PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND

SCIENTIFIC RESEARCH

UNIVERSITY OF AUGUST 20, 1955 - SKIKDA

Faculty of Science

Department of Computer Science

Specialization: Artificial Intelligence (AI)

Master's Thesis

Theme

AI-POWERED WEB APPLICATION FOR HOTEL RESERVATION SYSTEM

Prepared By:

- **CHORFI DHIA EDDINE**

Supervised by:

- **LAHSASNA ADEL**

Academic Year: 2025/2026

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

ACKNOWLEDGMENT

First and foremost, I thank **ALLAH TAEALAA** for His grace and for granting me the guidance, the strength, and the perseverance to bring this project to completion.

I am deeply grateful to **My Parents**, whose constant support, encouragement, and provision allowed me to devote myself to this work and to complete it in comfort and peace of mind.

I extend my sincere thanks to my supervisor, **LAHSASNA ADEL**, for his valuable guidance, his insightful advice, and the time and attention he generously gave throughout the course of this work.

Finally, I would like to thank **the Members of the Examination** committee for accepting to read and evaluate this thesis, and for the consideration they have devoted to it.

Abstract

The digital transformation of the hospitality sector has made online reservation systems central to the way hotels operate, while artificial intelligence (AI) has become a decisive competitive factor through dynamic pricing and the automated analysis of guest opinion. These intelligent capabilities, however, remain largely fragmented, costly, or opaque, and so out of reach for the independent hotel owner. This thesis addresses that gap through the design and implementation of Hotelify, a complete, full-stack web application for hotel reservation in which AI is a built-in capability. The system unites the conventional functions of reservation and hotel management with two integrated models, namely a LightGBM model that predicts and recommends room prices in support of dynamic pricing, and a fine-tuned DistilBERT model, trained with focal loss to counter class imbalance, that analyses the sentiment of guest reviews. These capabilities are organised around three roles, those of the guest, the owner, and the administrator, and are presented transparently, so that the owner retains authority over the decisions that the system informs. The application was built with a React frontend, a Node.js and Express backend, a separate FastAPI AI service, and a MongoDB database, orchestrated with Docker Compose. The evaluation shows the system to be functionally complete and soundly engineered; the sentiment analysis model attains strong and balanced performance, while the price prediction model, despite strong offline metrics, is affected by a target leakage limitation that is identified as the foremost subject of future work. Overall, the work demonstrates how intelligent and explainable decision support can be brought, within a single integrated platform, into the service of the guests, owners, and administrators who together make up a hotel reservation platform.

Keywords: Hotel Reservation System (HRS), Web Application, Artificial Intelligence (AI), Dynamic Pricing, Price Prediction, Sentiment Analysis, LightGBM, DistilBERT, Full-Stack Development, MongoDB.

Résumé

La transformation numérique du secteur de l'hôtellerie a placé les systèmes de réservation en ligne au cœur du fonctionnement des hôtels, tandis que l'intelligence artificielle (IA) est devenue un facteur concurrentiel déterminant à travers la tarification dynamique et l'analyse automatisée de l'opinion des clients. Ces capacités intelligentes demeurent toutefois fragmentées, coûteuses ou opaques, et donc hors de portée de l'hôtelier indépendant. Ce mémoire répond à ce besoin par la conception et la réalisation de Hotelify, une application web complète de réservation hôtelière dans laquelle l'IA est une fonctionnalité intégrée. Le système réunit les fonctions classiques de réservation et de gestion hôtelière avec deux modèles intégrés : un modèle LightGBM qui prédit et recommande les prix des chambres au service d'une tarification dynamique, et un modèle DistilBERT affiné, entraîné avec une fonction de perte focale pour pallier le déséquilibre des classes, qui analyse le sentiment des avis des clients. Ces capacités s'organisent autour de trois rôles, ceux du client, du propriétaire et de l'administrateur, et sont présentées de manière transparente, de sorte que le propriétaire conserve l'autorité sur les décisions que le système éclaire. L'application a été développée avec une interface React, un serveur Node.js et Express, un service d'IA FastAPI distinct et une base de données MongoDB, orchestrés au moyen de Docker Compose. L'évaluation montre que le système est fonctionnellement complet et solidement conçu : le modèle d'analyse de sentiment atteint des performances solides et équilibrées, tandis que le modèle de prédiction des prix, malgré de bonnes métriques hors ligne, est affecté par une limite de fuite de cible (target leakage) identifiée comme l'axe prioritaire des travaux futurs. Dans l'ensemble, ce travail démontre comment une aide à la décision intelligente et explicable peut être mise, au sein d'une plateforme unique et intégrée, au service des clients, des propriétaires et des administrateurs qui composent ensemble une plateforme de réservation hôtelière.

Mots-clés : Système De Réservation Hôtelière (SRH), Application Web, Intelligence Artificielle (IA), Tarification Dynamique, Prédiction Des Prix, Analyse De Sentiment, LightGBM, DistilBERT, Développement Full-Stack, MongoDB.

ملخَّص

أدّى التحوّل الرقميّ في قطاع الضيافة إلى جعل أنظمة الحجز عبر الإنترنت محورًا لعمل الفنادق، فيما غدا الذكاء الاصطناعي (AI) عاملًا تنافسيًّا حاسمًا عبر التسعير الديناميكيّ والتحليل الآليّ لآراء النزلاء. ومع ذلك، تبقى هذه القدرات الذكيّة مجزّأةً أو مكلفةً أو معتِمةً في توصياتها، فتظلّ بعيدةً عن متناول مالك الفندق المستقلّ. وتعالج هذه المذكّرة هذه الفجوة عبر تصميم وتنفيذ Hotelify، وهو تطبيق ويب كامل لحجز الفنادق يكون فيه الذكاء الاصطناعي قدرةً مدمجةً لا إضافةً لاحقة. ويجمع النظام بين الوظائف التقليديّة للحجز وإدارة الفندق ونموذجين مدمجين، هما نموذج LightGBM الذي يتنبّأ بأسعار الغرف ويوصي بها خدمةً للتسعير الديناميكيّ، ونموذج DistilBERT المضبوط بدقّةٍ والمدرَّب بدالّة الخسارة البؤريّة (focal loss) لمعالجة اختلال توازن الأصناف، الذي يحلّل مشاعر مراجعات النزلاء. وتتنظّم هذه القدرات حول ثلاثة أدوار، هي الضيف والمالك والمدير، وتُعرض بشفافيّةٍ تُبقي للمالك سلطة القرار الذي يُرشده النظام إليه. وقد بُني التطبيق بواجهة React وخادم Node.js وExpress وخدمة ذكاء اصطناعيّ منفصلة بـ FastAPI وقاعدة بيانات MongoDB، منسَّقةً عبر Docker Compose. ويُظهر التقييم أنّ النظام مكتملٌ وظيفيًّا ومتينُ الهندسة؛ إذ يبلغ نموذج تحليل المشاعر أداءً قويًّا ومتوازنًا، بينما يتأثّر نموذج التنبّؤ بالأسعار — رغم قوّة مقاييسه غير المتّصلة — بقيد تسرّب الهدف (target leakage) الذي حُدّد بوصفه أولويّة العمل المستقبليّ. وإجمالًا، يُبيّن هذا العمل كيف يمكن وضع دعمٍ ذكيٍّ وقابلٍ للتفسير لاتّخاذ القرار، ضمن منصّةٍ واحدةٍ متكاملة، في خدمة الضيوف والمُلّاك والمديرين الذين تتألّف منهم معًا منصّة حجز الفنادق.

الكلمات المفتاحية: نظام حجز الفنادق (HRS)، تطبيق ويب، الذكاء الاصطناعي (AI)، التسعير الديناميكي، التنبّؤ بالأسعار، تحليل المشاعر، LightGBM، DistilBERT، التطوير الكامل (Full-Stack)،قاعدة البيانات MongoDB.

Table of Contents

Chapter 01: General Introduction 1

1. General Context 2
2. Motivation 2
3. Problem Statement 3
4. Objectives 3
5. Proposed Solution 4
6. Thesis Organization 4

Chapter 02: Background & Literature Review 5

1. Introduction 6
2. Overview of Hotel Reservation Concepts 6
   - 2.1. Hotel 7
   - 2.2. Hotel Reservation 7
   - 2.3. Room Type 8
   - 2.4. Availability 9
   - 2.5. Occupancy Rate 9
   - 2.6. Revenues 9
3. Hotel Reservation Systems 10
   - 3.1. Types of Hotel Reservation System 11
   - 3.2. Importance and Benefits of Hotel Reservation Systems 12
4. Evolution of Hotel Reservation Systems 13
   - 4.1. The Manual Reservation Era (Pre-1940s) 14
   - 4.2. The Emergence of Early Mechanical and Centralised (1940s–1950s) 14
   - 4.3. The Emergence of Computerised Reservation Systems (1960s–1980s) 15
   - 4.4. The Rise of Web-Based and Internet-Based Reservation Systems (1990s-2000s) 15
   - 4.5. AI-Powered Hotel Reservation Systems (The Modern Era) 16
5. Components of a Hotel Reservation System 17
   - 5.1. Booking Engine 17
   - 5.2. Central Reservation System (CRS) 18
   - 5.3. Channel Manager 19
   - 5.4. Property Management System (PMS) Integration 19
   - 5.5. Payment Gateway 19

5.6. Reporting and Analytics Tools 20
5.7. Database Management 21
5.8. System Integration: The Interconnected HRS Architecture 21
6. Core Functionalities of a Hotel Reservation System 21
6.1. Search and Availability Checking 22
6.2. Booking and Reservation Management 23
6.3. User Account Management and Role-Based Access 23
6.4. Payment Processing 24
6.5. Reporting, Analytics, and AI-Driven Decision Support 24
7. AI in Tourism and Hospitality 25
7.1. The Scope of AI Adoption in Tourism and Hospitality 25
7.2. Dynamic Pricing and Demand Forecasting in Hospitality 26
7.3. Natural Language Processing and Sentiment Analysis of Guest Reviews 27
7.4. The Convergence of AI and Web-Based HRS: Rationale for the Present System 29
8. Conclusion 29
Chapter 03: System Analysis & Design 31
1. Introduction 32
2. System Analysis 32
2.1. Overview of the Proposed System (Hotelify) 33
2.2. System Actors and Their Roles 33
.2.3 Functional Requirements 34
2.4. Non-Functional Requirements 36
3. System Modeling (UML Diagrams) 38
3.1. Use Case Diagram 39
3.2. Class Diagram 42
3.3. Sequence Diagrams 45
3.4. Activity Diagram 52
4. Database Design 55
4.1. Conceptual Design 55
4.2. Logical Structure and Data Dictionary 57
5. System Architecture Design 61
.5.1 Overall Architecture 62

5.2. AI Service Integration within the Architecture 64
6. AI Models Design (Pipeline) 65
6.1. Dataset Description 65
6.2. Price Prediction and Recommendation Model for Dynamic Pricing 67
6.3. Sentiment Analysis Model Pipeline 70
6.4. Unified Comparison of Both Pipelines 73
7. User Interface Design Pipeline 75
7.1. Guest Interface 76
7.2. Owner Dashboard 76
7.3. Admin Dashboard 76
8. Conclusion 77
Chapter 04: Implementation, Experimental Results & Evaluation 78
1. Introduction 79
2. Development Environment and Technologies 79
2.1. General Development Environment and Tools 80
2.2. Frontend Technologies 80
2.3. Backend Technologies 81
2.4. AI Service Technologies 82
2.5. AI Model Development Tools 83
2.6. Database and Data Management 84
2.7. AI-Assisted Development Tool 84
3. System Implementation 85
3.1. Database Implementation 85
3.2. Backend Implementation 86
3.3. AI Service Implementation 87
3.4. Frontend Implementation 87
3.5. System Integration and Deployment 88
4. AI Models: Training and Experimental Results 88
4.1. Price Prediction Model Results 88
4.2. Sentiment Analysis Model Results 98
5. Presentation of the Web Application 102
5.1. Unauthenticated User Interface 102
5.2. Guest Interface 110

5.3. Owner Dashboard 124
5.4. Admin Dashboard 137
6. System Evaluation 151
6.1. Functional Evaluation 151
6.2. Non-Functional Evaluation 152
6.3. Comparison with Existing Solutions 155
6.4. Achievement of Objectives 156
7. Conclusion 158
Chapter 05: General Conclusion 159
1. Conclusion 160
2. Contributions 160
3. Limitations 161
4. Future Work 161
References 163

List of Figures

Figure 2.1: Hotel exterior 7
Figure 2.2: Common Hotel Room Type Classifications and Their Key Attributes 8
Figure 2.3: Online Hotel Booking Software Market Size,Trends and Forecast to 2033.....11
Figure 2.4: Evolution of Hotel Reservation Systems from Manual Processes to AI-Powered Platform 14
Figure 2.5: Hotel Booking Engine 18
Figure 2.6: Architecture of the CRS and Its Integration with Hotel Distribution Channels 18
Figure 2.7: Channel Manager Architecture for Real-Time Synchronization Between PMS and Online Distribution Channel 19
Figure 2.8: Secure Payment Gateway Integration in a Hotel Reservation System 20
Figure 2.9: Reporting and analytics tools interface for Hotel Rservation System 20
Figure 2.10: Architecture of a Unified Hotel Database and its interaction with the rest of the system components 21
Figure 2.11: The Five Core Functionalities of a Hotel Reservation System and Their Interrelation.....22
Figure 2.12: Conceptual Flow of AI-Driven Dynamic Pricing in Hospitality 27
Figure 2.13: Conceptual Flow of Sentiment Analysis of Guest Reviews Using a Pre-Trained Language Model (BERT).....28
Figure 3.1: Hotelify Logo 33
Figure 3.2: Classification Map of Different UML Diagrams.....38
Figure 3.3: Use Case Diagram of the Hotelify System.....39
Figure 3.4: Class Diagram of the Hotelify System 42
Figure 3.5: Sequence Diagram of the Authentication and Token Management Scenario .46
Figure 3.6: Sequence Diagram of the Price Prediction Scheduled Job 48
Figure 3.7: Sequence Diagram of Guest and Owner Access to Price Predictions.....49
Figure 3.8: Sequence Diagram of the Guest Review and Sentiment Analysis Scenario 51
Figure 3.9: Activity Diagram of the Reservation Process and Management 53
Figure 3.10: Entity-Relationship Diagram of the Hotelify Database 56
Figure 3.11: The Layered Architecture of the Hotelify Platform Overall Architecture.....62
Figure 3.12: The Design Pipeline of the Price Prediction and Recommendation Model....68
Figure 3.13: The Design Pipeline of the Sentiment Analysis Model 71
Figure 3.14: The User Interface Design Pipeline 75
Figure 4.1: Development Tools & Technologies 80
Figure 4.2: Frontend Technologies.....81
Figure 4.3: Backend Technologies.....82
Figure 4.4: AI Service Technologies.....83
Figure 4.5: AI Model Development Tools.....84
Figure 4.6: Database & Data Management Technologies 84
Figure 4.7: AI-Assisted Development Tool 85
Figure 4.8: Distribution of the Base Price.....90
Figure 4.9: Boxplots of the Lead Time and the Base Price.....90
Figure 4.10: Base Price Distribution by Reserved Room Type 91

Figure 4.11: Evolution of the Average Base Price Over the Year 92
Figure 4.12: SHAP Summary Plot of the Selected Model....96
Figure 4.13: SHAP Dependence Plot for the Recent Average Price 96
Figure 4.14: SHAP Waterfall Plot Explaining an Individual Prediction 97
Figure 4.15: Training of The Final Model with Early Stopping 97
Figure 4.16: Distribution of the Three Sentiment Classes 99
Figure 4.17: Distribution of Review Length, in Words and Characters, per Class 100
Figure 4.18: The 20 Most Frequent Words In Each Sentiment Class....100
Figure 4.19: The Hero Area and Search Bar of the Home Page 103
Figure 4.20: Browsing Hotels by Country on the Home Page, Light Mode and in Dark Mode....103
Figure 4.21: The Authentication Guard Dialog....104
Figure 4.22: The Login Page 105
Figure 4.23: A Failed Login Reported Through an Error Snackbar 105
Figure 4.24: The Account Registration Page 106
Figure 4.25: Requesting a Password Reset by Email....107
Figure 4.26: Confirmation that a Reset Link Has Been Sent 107
Figure 4.27: The Password Reset Email....108
Figure 4.28: Setting A New Password 109
Figure 4.29: Confirmation That the Password Has Been Reset Successfully 109
Figure 4.30: The Access Code Page 110
Figure 4.31: The Home Page as Seen by a Signed-In Guest 111
Figure 4.32: The Results of a Hotel Search....111
Figure 4.33: The Top of the Hotel Details Page....112
Figure 4.34: The Full-Screen Photo Gallery of a Hotel 112
Figure 4.35: The Available Rooms with the AI Price Forecast (Prediction) for each Room Type 113
Figure 4.36: The Explanation of an AI Price Forecast (Prediction), Shown for Increasing And Decreasing Prices 114
Figure 4.37: The Ratings Overview of a Hotel 114
Figure 4.38: Guest Reviews, each Annotated with the Analysed Sentiment 114
Figure 4.39: The Detailed Sentiment Breakdown of a Review 115
Figure 4.40: The First Step of Booking, Collecting the Guest Information 115
Figure 4.41: The Second Step of Booking, Selecting the Payment Method 116
Figure 4.42: The Third Step of Booking, the Final Review with the Option to Confirm Now or Later 116
Figure 4.43: The Outcome when the Booking is Confirmed Immediately....117
Figure 4.44: The Outcome when the Booking is Saved to be Confirmed Later....117
Figure 4.45: The Reservations Page, with the Status tabs and a Pending Reservation....118
Figure 4.46: A Confirmed Reservation 119
Figure 4.47: The Dialog for Editing a Reservation....119
Figure 4.48:A Checked-Out Reservation, Offering to Write a Review, and a Checked-In Reservation, Counting Down to Check-Out....120
Figure 4.49: The Account Page of the Guest....121

Figure 4.50: The Form for Editing the Account Details 122
Figure 4.51: The Page on Which the Guest Manages their Own Reviews 122
Figure 4.52: The Dialog for Editing a Review 123
Figure 4.53: The Confirmation Dialog Shown before a Review is Deleted 123
Figure 4.54: The Favourites Page of the Guest 124
Figure 4.55: KPI cards summarising the hotel's financial 125
Figure 4.56: The Revenue Trend Chart 126
Figure 4.57: The Operational Analytics 126
Figure 4.58: The Commercial Analytics 127
Figure 4.59: The AI Insights 128
Figure 4.60: The Recent Reservation Log 128
Figure 4.61: The Hotel Profile 129
Figure 4.62: The Edit Hotel Information Dialog 130
Figure 4.63: The Add a New Room Type Dialog 131
Figure 4.64: The Edit Room Type Dialog 131
Figure 4.65: The Price Recommendation Produced for Two Different Selected Check-In Dates 132
Figure 4.66: An Accepted Price Recommendation 133
Figure 4.67: A Rejected Price Recommendation 133
Figure 4.68: The Guest Reviews of the Property 134
Figure 4.69: The Reservations Page 135
Figure 4.70: The Owner's Account 136
Figure 4.71: The Identity Verification Dialog Requiring the Account Password before the Access Code is Revealed 137
Figure 4.72: The Access Code Revealed for a Limited Time 137
Figure 1.73: The Platform-Wide KPI Cards Covering the Platform's Commercial Results 138
Figure 4.74: The Growth and Performance Charts 139
Figure 4.75: The Distribution Charts 140
Figure 4.76: The Geographic and Cancellation Charts 140
Figure 4.77: The Per-Hotel AI Insights 141
Figure 4.78: The Recent Reservations Across the Platform 141
Figure 4.79: The Authentication Activity Log 142
Figure 4.80: The Owners Tab 142
Figure 4.81: The Create New Owner Dialog 143
Figure 4.82: The Owner Details Dialog 143
Figure 4.83: The Edit Owner Dialog 144
Figure 4.84: The Guests Tab 144
Figure 4.85: The Catalogue of All Platform Hotels 145
Figure 4.86: The Create Hotel Dialog 146
Figure 4.87: The Hotel Profile 147
Figure 4.88: The Hotel's Room Types 147
Figure 4.89: The Hotel's Guest Reviews 148
Figure 4.90: The Platform-Wide Monitoring of Reservations 148

Figure 4.91: The Training Results and Evaluation Metrics of the Price Prediction and Sentiment Analysis Models 150
Figure 4.92: The Administrator's Profile and the Access Code Panel 150

List of Tables

Table 3.1: The users Collection 57
Table 3.2: The guests Collection.....57
Table 3.3: The owners Collection 58
Table 3.4: The admins Collection 58
Table 3.5: The hotels Collection 58
Table 3.6: The roomtypes Collection 59
Table 3.7: The facilities Collection.....59
Table 3.8: The reservations Collection 59
Table 3.9: The payments Collection 60
Table 3.10: The reviews Collection 60
Table 3.11: The sentimentanalysis Collection 60
Table 3.12: the pricePredicyions Collection 61
Table 3.13: the favorites Collection 61
Table 3.14: A Sample of the Hotel Booking Demand Dataset 66
Table 3.15: A Sample of the HRAST Dataset 67
Table 3.16: A Comparison of the Design Choices Underlying the Price Prediction and Sentiment Analysis Pipelines 73
Table 4.1: A Sample of the Processed Price Dataset.....89
Table 4.2: Total Bookings and Cancellations by Month 91
Table 4.3: A sample set of the feature-engineering dataset, showing a representative subset of the engineered features. 93
Table 4.4: Best Cross-Validation MAPE Obtained by the Randomized Search for Each Candidate Model 94
Table 4.5: Tuned Hyperparameters Selected by the Search for Each Candidate Model....94
Table 4.6: Evaluation of the three candidate models on the test set.....94
Table 4.7: Global Feature Importance of the Selected Model, Measured by the Mean Absolute SHAP Value 95
Table 4.8: A Sample Set of the Processed Sentiment Dataset.....98
Table 4.9: Summary Statistics of the Review Length, by Class 99
Table 4.10: Per-Epoch Results of the Initial Training of the Model 101
Table 4.11: Overall Evaluation of the Model 101
Table 4.12: Training Loss of the Final Model 101
Table 4.13: Verification of the Functional Requirements Against the Delivered System.151
Table 4.14: Qualitative Comparison of the Proposed System with Existing Categories of Solution.....155

CHAPTER 01



GENERAL INTRODUCTION

1. General Context

The tourism and hospitality industry has undergone a profound digital transformation over the past two decades. The way in which travellers discover, compare, and book accommodation has shifted almost entirely from the physical and the telephonic to the online, and a stay at a hotel now typically begins with a search on a website or an application rather than at a reception desk. Online hotel reservation systems have consequently become central infrastructure for the sector, mediating the relationship between travellers seeking accommodation and the hotels that provide it, and shaping the commercial fortunes of properties of every size.

In parallel with this shift, artificial intelligence (AI) has moved from the periphery of the industry to its core. Two applications in particular have proved especially consequential. The first is dynamic pricing, in which the nightly rate of a room is adjusted in response to demand, seasonality, and other signals rather than being fixed, with the aim of maximising revenue without deterring guests. The second is the analysis of guest opinion, in which the large volume of written reviews that a property accumulates is processed automatically to extract the sentiment it carries, turning unstructured text into a measurable indicator of guest satisfaction. Together, intelligent pricing and the automated understanding of feedback have come to define the competitive frontier of modern hotel commerce.

This convergence of online reservation and AI forms the general context of the present work, which sits at their intersection: a web application for hotel reservation in which intelligence is not an afterthought but a built-in capability.

2. Motivation

The motivation for this work arises from a gap between the capabilities described above and their availability to those who would most benefit from them. Intelligent pricing and reputation analysis confer a clear commercial advantage, and the large platforms and established commercial tools that provide them are powerful; yet these capabilities tend to be fragmented across several separate and often costly products, to be concentrated in the hands of large intermediaries, or to be delivered as opaque recommendations that the hotelier is expected to trust without explanation. Independent operators, who face the same competitive pressures as large chains but lack their resources, are frequently left without such tools in an integrated, affordable, and intelligible form; and the guests they serve are, in turn, denied the smoother and better-informed booking experience that those same tools could enable.

There is, therefore, a genuine value in a single platform that brings these intelligent capabilities, transparently and accessibly, into a complete reservation system that serves all of its participants: guests reaching better booking decisions through a smoother experience, owners managing their properties more responsively, and an administrator overseeing the platform as a whole. To this practical motivation is added an academic

one: the design and construction of such a system is an opportunity to engineer a complete, full-stack web application, end to end, in which two distinct AI models are not merely trained in isolation but integrated meaningfully into a working product. The combination of these practical and academic aims is what motivates the present project.

3. Problem Statement

The problem that this work addresses can be stated as follows. A hotel reservation platform must serve, within a single coherent system, the distinct but interdependent needs of three actors. The guest needs to discover and compare hotels, to reach a well-informed booking decision, and to manage reservations through an experience that is smooth and clear. The hotel owner needs to manage the property, its rooms, and its reservations, to respond intelligently to changing demand, to keep sight of guest satisfaction, and to see a consolidated view of performance. The administrator needs to oversee the platform as a whole, namely its users, its hotels and their owners, its reservations, and the behaviour of its intelligent models. In current practice, these needs are met, if at all, by an assortment of disconnected tools, and the intelligent capabilities that could support better decisions across all three actors are typically expensive, reserved to large platforms, or presented without justification.

The central problem, then, is the absence of an integrated, transparent, and accessible platform that unites conventional reservation and management functions with intelligent and explainable decision support, organised around the distinct roles of the people who use it. It is this problem that the present work sets out to solve.

4. Objectives

In response to this problem, the work pursues the following objectives:

1. To design and implement a complete, full-stack web application for hotel reservation, rather than an isolated component.
2. To support the distinct actors of the platform, namely the guest, the hotel owner, and the administrator, each under access control appropriate to its responsibilities.
3. To enhance the platform with a price prediction and recommendation capability that guides the pricing of rooms.
4. To incorporate the sentiment analysis of guest reviews, turning written feedback into a measurable indicator of satisfaction.
5. To provide owners and administrators with integrated analytics through which they can understand and oversee their operations.
6. To realise the system with sound engineering qualities, namely security, modularity, maintainability, and deployability.

These objectives define the criteria against which the success of the work is judged, and the extent to which each has been achieved is assessed explicitly in the evaluation of the system presented in Chapter 4.

5. Proposed Solution

The solution proposed in this work is Hotelify, an artificial-intelligence-powered web application for hotel reservation that addresses the problem above within a single, coherent platform. Hotelify combines the conventional functions of a reservation system, namely the browsing of hotels, the making and management of bookings, and the administration of properties, with two integrated AI capabilities: a model that predicts and recommends room prices in support of dynamic pricing, and a model that analyses the sentiment of guest reviews. These capabilities are organised around three roles, those of the guest, the hotel owner, and the administrator, each served by an interface shaped to its needs. The intelligent features are presented transparently and in support of human judgement: guests are helped towards better-informed booking decisions, owners receive pricing guidance that they remain free to accept or reject, and administrators are given oversight of the platform and of its models. The detailed design, implementation, and evaluation of this solution form the substance of the chapters that follow.

6. Thesis Organization

The remainder of this thesis is organised into four chapters. Chapter 2 reviews the background and the related literature, situating the work within the existing body of knowledge on reservation systems, dynamic pricing, and sentiment analysis. Chapter 3 presents the analysis and design of the system, covering its requirements, its modelling, its database, its architecture, and the design of its AI pipelines and user interfaces. Chapter 4 describes the implementation, reports the experimental results of the two models, presents the finished application, and evaluates the completed system against its requirements and objectives. Finally, Chapter 5 draws together the general conclusions of the work and sets out its limitations and the directions identified for future development.

CHAPTER 02



BACKGROUND & LITERATURE REVIEW

1. Introduction

The development of any intelligent information system must be grounded in a thorough understanding of the domain it is intended to serve. For a hotel reservation system enhanced with AI, this requires not only familiarity with the technical architecture of reservation platforms, but also a clear grasp of the underlying hospitality concepts, the historical trajectory of the field, and the role that AI has come to play within the tourism and hospitality sector. This chapter establishes that foundation, providing the conceptual and theoretical background upon which the analysis, design, and implementation presented in the subsequent chapters are built.

The purpose of this chapter is twofold. First, it aims to define and clarify the core concepts of the hotel reservation domain, ensuring that the terminology and ideas used throughout the remainder of this work are precise and consistently understood. Second, it situates the present project within the broader landscape of hotel reservation systems, examining how such systems are structured, how they have evolved over time, and how AI is increasingly being integrated to enhance their capabilities. Together, these objectives provide the reader with the necessary perspective to appreciate both the relevance and the contribution of the system developed in this work.

The chapter is organised as follows. It begins with an overview of the fundamental concepts of the hotel reservation domain, including the notions of a hotel, a reservation, room types, availability, occupancy rate, and revenue. It then introduces hotel reservation systems as integrated platforms, discussing their principal types and their importance and benefits within the tourism sector. The chapter proceeds to trace the historical evolution of these systems, from early manual methods through to the modern, AI-powered era. Building on this, it examines the components and the core functionalities that together define a complete reservation system. Finally, it explores the growing role of AI in tourism and hospitality, focusing in particular on the two AI capabilities that are central to the present project: dynamic pricing and the sentiment analysis of guest reviews. Throughout, the discussion remains at a conceptual and introductory level, deliberately reserving the technical design and implementation details for the chapters that follow.

2. Overview of Hotel Reservation Concepts

The hotel reservation industry is built upon a set of fundamental concepts that must be clearly understood before examining hotel reservation systems (HRS) and their technological enhancements. These concepts determine how accommodation services are provided, managed, and improved. Core elements such as hotel operations, reservation processes, room types, availability management, occupancy rates, and revenue generation form the theoretical basis of modern hospitality management. They are closely interconnected and collectively affect service quality, operational efficiency,

and financial performance across the tourism sector. This section, therefore, presents the foundational concepts upon which hotel reservation systems are designed and evaluated.

2.1. Hotel

A hotel is both a physical establishment and a commercial enterprise that provides paid accommodation services to a diverse range of guests, including tourists, business travellers, and long-term visitors [1]. Beyond lodging, hotels offer a broad spectrum of hospitality services aimed at ensuring guest comfort and satisfaction throughout their stay. In the broader economic context, the hotel industry plays a vital role in supporting tourism and generating revenue at both local and national levels [2].

Central to the hotel business model is the management of customer relationships. Hotels are fundamentally service-oriented organisations whose operations revolve around transactions and interactions between guests and staff [3]. In recent decades, the traditional reliance on intermediaries to reach customers has been progressively replaced by digital distribution channels, particularly the internet, which has significantly reduced operational costs and expanded market reach [4]. This digital transition has positioned hotels to engage directly with potential guests, enabling personalised service delivery and more efficient reservation processes.



Figure 2.1: *Hotel exterior*

2.2. Hotel Reservation

A hotel reservation is the formal process by which a guest secures accommodation in advance for a specified period, whether for leisure or business purposes [5]. The reservation process typically involves capturing essential guest information, including contact details, check-in and check-out dates, the number of guests, and any special requests, all of which facilitate smooth check-in procedures and enable the hotel to meet guest expectations effectively.

Reservations may be made directly through the hotel or via third-party online travel agencies (OTAs), which are digital platforms that aggregate accommodation listings and allow users to search, compare, and book hotels through web or mobile applications [6]. The shift towards online reservation platforms has greatly enhanced the convenience and accessibility of hotel booking, benefiting both guests and hotel operators. For guests, pre-booking provides certainty and peace of mind; for hotels, it enables more accurate operational planning and resource allocation. Overall, hotel reservations constitute a fundamental component of modern travel management, enabling personalised, secure, and well-organised stays [5].

2.3. Room Type

Room types represent a critical dimension of hotel inventory management, as they define the categories of accommodation available within a property and directly influence guest selection and pricing strategies. Each room type is characterised by attributes such as bed configuration, room size, occupancy capacity, included amenities, and associated pricing tier. Common room categories include Standard Single, Standard Double, Twin, Deluxe, Suite, Family, and Accessible rooms, among others, although classifications may vary across hotel brands and star ratings [7].

As illustrated in Figure 2.2, room types are grouped into standard and enhanced tiers, each with defined occupancy limits and pricing levels, and all are managed through a unified HRS inventory layer that governs availability, pricing logic, and booking validation.

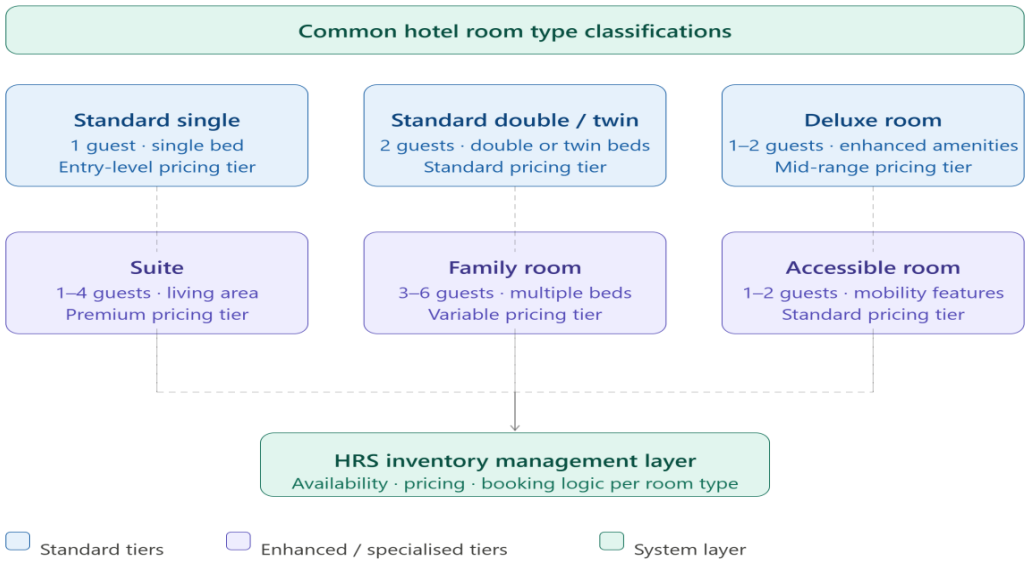


Figure 2.2: *Common Hotel Room Type Classifications and Their Key Attributes*

From a system perspective, room types are the primary unit around which availability tracking, pricing, and reservation logic are organised. When a guest submits a booking request, the reservation system validates availability at the room-type level rather than for individual rooms, thereby optimising inventory utilisation and reducing the risk of

overbooking. In AI-powered hotel reservation systems, room type data also serves as a key input feature for price prediction models and personalised recommendation engines, both of which rely on room classification to generate accurate and contextually relevant outputs.

2.4. Availability

Availability in the context of hotel reservation systems refers to the real-time status of rooms, indicating whether a given room type is open for booking within a requested date range. It involves the continuous updating of room occupancy records and the synchronisation of inventory data between the hotel's internal systems and external distribution channels [8], such as OTAs. Effective availability management is essential for preventing double bookings, supporting accurate capacity planning, and providing guests with reliable information at the point of reservation, all of which directly contribute to guest satisfaction and operational integrity [8].

2.5. Occupancy Rate

The occupancy rate is a key performance indicators (KPI) that reflects the proportion of a hotel's rooms that are occupied over a given period relative to the total room capacity available during that same period. As a general measure of how efficiently a hotel's room inventory is utilised, it is among the most widely used indicators in the hospitality industry and a central input to revenue management [9].

Because a hotel's capacity is consumed over time rather than at a single instant, the occupancy rate is expressed in terms of room nights, where one room night denotes the availability, or the occupancy, of one room for one night. The standard formula is therefore:

$$Occupancy\ Rate = \frac{Occupied\ Room\ Nights}{Available\ Room\ Nights} \times 100$$

where the available room nights over a period are given by the hotel's total number of rooms multiplied by the number of nights in that period:

$$Available\ Room\ Nights = Total\ Rooms \times\ Number\ of\ Nights\ in\ the\ Period$$

A higher occupancy rate thus indicates a fuller utilisation of the available capacity over the period considered.

2.6. Revenues

Revenue, in the context of hotel reservation, represents the total financial income generated from room bookings over a defined period. In general terms, it is determined by how many rooms are booked, for how many nights, and at what price [9].

For a period containing n bookings, the revenue may be expressed as the sum, over those bookings, of the price of each booking's room multiplied by its number of nights and its number of rooms:

$$Revenue = \sum_{i=1}^{n} Room\ Price_i \times Nights_i \times Rooms_i$$

where Room Price_i, Nights_i, and Rooms_i denote, respectively, the nightly price, the number of nights, and the number of rooms of the i-th booking.

Together, the occupancy rate and the revenue constitute the two primary performance metrics by which hotels assess their commercial standing, guide their pricing decisions, and optimise the allocation of their resources, which makes them essential inputs to AI approaches to revenue management.

3. Hotel Reservation Systems

A Hotel Reservation System (HRS) is an integrated information system used in the tourism and hospitality industry to manage, process, and facilitate hotel room bookings efficiently [1]. It serves as a software platform that enables hoteliers to accept and manage online reservations, monitor occupancy rates, and optimise revenue generation [6]. Rather than functioning as a single monolithic tool, an HRS is composed of multiple interconnected components that work in concert to capture, process, and manage reservation data effectively.

HRS platforms allow guests to book rooms remotely through secure internet-based interfaces that safeguard personal and financial information. These systems frequently integrate with OTAs, enabling guests to compare prices, facilities, and availability across multiple properties before confirming a reservation [6]. In addition to improving the guest experience, modern HRS solutions help hotels reduce overbooking, automate confirmation workflows, enhance customer support, and respond to the growing preference for direct online booking [1] [6] [10].

The growing adoption of online hotel reservation platforms is evidenced by consistent market expansion. As illustrated in Figure 2.2, the Online Hotel Booking Software Market was valued at USD 4.22 billion in 2024 and is projected to reach USD 7.24 billion by 2033, reflecting a compound annual growth rate (CAGR) of 6.2% [6].

Figure 2.3 figure illustrates the increasing profits from online hotel reservation, indicating a growing trend among guests to reserve hotel online:

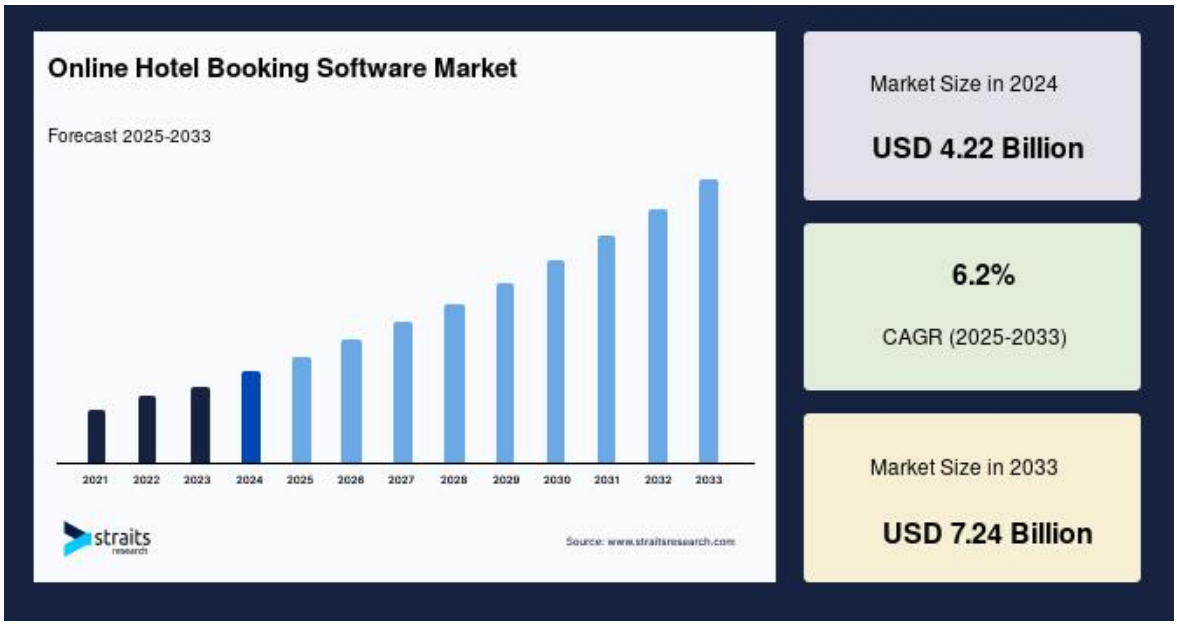


Figure 2.3: *Online Hotel Booking Software Market Size,Trends and Forecast to 2033*

3.1. Types of Hotel Reservation System

Several distinct types of HRS exist, each varying in its architectural design, deployment model, and intended operational context. The most prominent types are described below.

❖ Manual Hotel Reservation System

The earliest form of reservation management relied on pen-and-paper logbooks, telephone calls, and in-person interactions. Such systems, which predate digital technology, are now largely obsolete, as they were inherently prone to human error and incapable of scaling with increasing demand [10].

❖ On-premise Reservation System

An on-premise reservation system is installed and operated locally on the hotel's own servers and hardware infrastructure. While it offers greater control over data security and system configuration, it requires dedicated IT expertise for ongoing maintenance and upgrades, making it less flexible than cloud-based alternatives [11].

❖ Central Reservation System (CRS)

A Central Reservation System (CRS) is a centralised platform used primarily by hotel groups and chains to manage room inventory, pricing, reservations, and guest data across multiple properties from a unified interface [12]. CRS solutions enable real-time pricing and availability visibility for reservation agents, support centralised booking modifications and cancellations, and facilitate advanced

analytics and automated pricing strategies. They are commonly integrated with OTAs and property-level systems to streamline distribution [6].

- **Cloud-Based/Web-Based Hotel Reservation System**

  the prevailing standard in the hospitality industry, offering online access from any internet-enabled device while significantly reducing the burden of IT infrastructure and maintenance. These systems automate core reservation tasks, synchronise data across channels in real time, and simplify operational management [10] [12].

  This type of HRS forms the architectural foundation of the present project. The system developed in this work is a web-based application that incorporates AI capabilities, enabling users to access its full range of services online.

3.2. Importance and Benefits of Hotel Reservation Systems

Hotel reservation systems play a multifaceted and strategically important role within the tourism sector, generating benefits that span operational, commercial, and guest experience dimensions. The most significant of these are outlined below.

- **Attracting and Retaining Guests**

  By enabling guests to easily search, compare, and book accommodation online, HRS platforms actively encourage travel and increase tourist flow. The convenience of online booking extends guests' willingness to engage with tourism services, contributing to longer stays and higher conversion rates [13].

- **Supporting Marketing and Promotion**

  Reservation systems function as effective digitalmarketing tools, facilitating the promotion and distribution of accommodation services across avariety of online channels, including OTAs and metasearch engines. This expanded reachallows hotels to target a wider audience of potential guests and improve brand visibility [13].

- **Enhancing Competitiveness**

  By improving service efficiency and enabling rapid responses toguest demands, modern HRS platforms strengthen the competitive positioning of hotels in adynamic marketplace [13].

- **Facilitating Access and Simplifying Booking Processes**

  nline reservation systemseliminate the complexity associated with traditional booking methods, making the process ofsearching for hotels, checking availability, and securing reservations straightforward andaccessible to users worldwide [13].

❖ Supporting Guest Decision-Making

Reservation platforms provide guests with detailedproperty information, pricing comparisons, and verified reviews, enabling more informedaccommodation choices [14].

❖ Improving Operational Efficiency

HRS platforms automate numerous administrativefunctions, including availability updates, booking confirmation workflows, and guestcommunication. This automation reduces manual workload and allows hotel staff to focus onservice delivery and operational activities [12].

❖ Reducing Overbooking and Reservation Errors

Real-time synchronisation of roomavailability across all reservation channels minimises the risk of double bookings and humanerror inherent in manual processes [15].

❖ Increasing Revenue and Direct Bookings

By facilitating direct reservations through the hotel'sown web platform, HRS solutions reduce dependence on third-party OTAs and the associatedcommission fees. Some systems further enhance revenue through dynamic pricing andupselling mechanisms [12].

❖ Enabling 24/7 Online Access

The continuous availability of online reservation systems allowsguests to book at any time and from any location, increasing booking opportunities andbroadening the hotel's potential guest base [15].

❖ Collecting and Leveraging Guest Data

Reservation systems capture valuable data regardingguest preferences, booking histories, and demographic profiles. This information supportstargeted marketing campaigns, service personalisation, and the development of loyaltyprogrammes [15].

4. Evolution of Hotel Reservation Systems

The hotel reservation system has undergone a significant transformation over the past century,progressing from entirely manual procedures to sophisticated, AI-driven digital platforms. Thisevolution has been shaped by successive waves of technological innovation and by thesustained demand for greater efficiency, accuracy, and guest satisfaction in the tourism andhospitality industry.

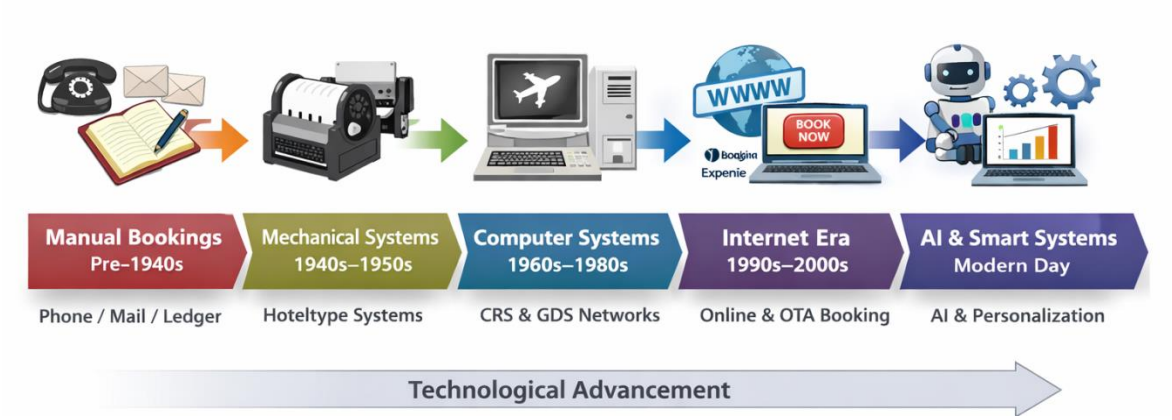


Figure 2.4: *Evolution of Hotel Reservation Systems from Manual Processes to AI-Powered Platform*

This section traces the major stages of this evolution, from the manual reservation era throughthe advent of computerised systems, the rise of internet-based platforms, and, ultimately, theintegration of AI in modern HRS solutions.

4.1. The Manual Reservation Era (Pre-1940s)

Prior to the introduction of computerised systems, hotel reservations were handled entirelythrough manual means, relying on telephone calls, postal correspondence, and telegrams.Reservation data was recorded by hand in logbooks, a process that was heavily dependent onthe accuracy and coordination of individual staff members [16]. The limitations of this approachwere considerable: double bookings, lost reservations, and departmental miscommunicationwere frequent occurrences that adversely affected both guest satisfaction and operationalperformance [16].

The absence of centralised data management made real-time tracking of room availabilityvirtually impossible. Reservation requests were processed individually, and updates were notshared across departments in a timely manner, leading to frequent discrepancies in roomallocation [16]. Furthermore, the system was inherently labour-intensive and non-scalable,meaning that as hotel operations grew in volume, the inadequacies of the manual approachbecame increasingly apparent. These limitations created an industry-wide demand for morereliable, systematised reservation solutions [22].

4.2. The Emergence of Early Mechanical and Centralised (1940s–1950s)

The shortcomings of manual reservation methods prompted the first efforts to formalise andautomate reservation processes within the hospitality industry. A notable milestone was theintroduction of the Hoteltype system by Westin Hotels in 1947, which represented the firststructured attempt to standardise reservation management by replacing informal handwrittenrecords with a more organised, rule-based approach [16].

A more significant breakthrough followed in the 1950s, when the conceptual framework for CRS emerged from a collaboration between American Airlinesand IBM. Originally developed for airline reservations, this model was subsequently adapted forthe hotel industry, introducing the principle of centralised reservation data accessible acrossmultiple channels through a unified system [16]. This approach addressed several criticalshortcomings of earlier methods, including the risk of duplicate bookings, inefficiencies ininter-departmental communication, and the absence of coordinated inventory management.These foundational concepts of standardisation, centralisation, and process optimisationcontinue to underpin modern HRS architectures.

4.3. The Emergence of Computerised Reservation Systems (1960s–1980s)

The adoption of computerised reservation systems in the 1960s marked a pivotal turning pointin the evolution of HRS. Initially developed for the airline industry, these systems wereprogressively adapted for hotel operations, enabling the automation of reservation processesthrough computerised databases [16] [17]. Key improvements included a significant reduction inhuman errors such as duplicate bookings, faster access to and updating of reservation records,and the centralisation of inventory and pricing data across multiple channels and properties [16] [17].

During the 1970s and 1980s, these systems evolved further to support higher transactionvolumes, incorporating functionalities such as inventory management, rate control, andautomated reporting tools. Concurrently, Global Distribution Systems (GDS) — includingplatforms such as Amadeus and Sabre — began to emerge, connecting hotels, airlines, andtravel agencies through global networks and enabling real-time reservation access on aninternational scale [17] [6]. This period established the foundational technical concepts ofreal-time data processing, system networking, and cross-platform integration that continue todefine modern HRS design [17].

4.4. The Rise of Web-Based and Internet-Based Reservation Systems (1990s-2000s)

The 1990s brought about a further transformation in hotel reservation systems, as computerisedplatforms evolved into fully networked, internet-based environments. In 1995, Promus HotelCorporation introduced a real-time central reservation system that offered instant bookingconfirmation and significantly improved the speed and reliability of the reservation process,reducing delays and enhancing guest satisfaction [16].

Simultaneously, the widespread adoption of the internet enabled hotels to establish their ownonline booking platforms, allowing guests to make direct reservations without relying ontraditional intermediaries [16] [17]. OTAs also emerged during this period as powerful distributionchannels, aggregating hotel listings and enabling price and availability

comparisons acrossmultiple providers — greatly expanding market reach while also increasing competitive pressureand commission costs for hotels [16].

The integration of Property Management Systems (PMS) with CRS solutions during this periodcreated more comprehensive hotel operations management environments, connectingreservation functions with front desk management, billing, housekeeping, and reporting [16].The technical innovations of this era — including web-based interfaces, relational databaseconnectivity, and API-driven system integration — laid the groundwork for the scalable, flexibleHRS architectures that characterise contemporary hospitality technology [17]. The presentsystem builds directly upon this foundation, aiming to realise the core capabilities establishedduring this era while extending them with AI-driven functionality.

4.5. AI-Powered Hotel Reservation Systems (The Modern Era)

The most recent stage in the evolution of HRS is characterised by the integration of artificialintelligence technologies, transforming reservation platforms from static digital tools intodynamic, intelligent systems capable of adaptive decision-making and personalised servicedelivery [18]. The key AI technologies now embedded within modern HRS platforms include thefollowing.

- **Recommendation systems**

  Analyse guest preferences and historical booking behaviour tosuggest suitable properties, room types, and services, creating a more personalised andengaging booking experience [18].

- **Chatbots and Virtual Assistants**

  Provide immediate, conversational support to guestsstroughout the reservation process, improving accessibility and enabling round-the-clockservice without the need for human intervention [18].

- **Dynamic Pricing with Price Prediction**

  Enables the automated adjustment of room ratesbased on real-time demand signals, historical pricing patterns, and competitive market data.This capability allows hotels to maintain pricing competitiveness while maximising revenue yield [18].

- **Sentiment Analysis**

  applies natural language processing (NLP) [19] techniques, the field devoted to enabling computers to process, interpret, and generate human language, to extract theemotional polarity from guest reviews and feedback, providing hotel managers with structured insights into guest satisfaction and service quality. This data informs operational improvementsand supports evidence-based decision-making [18].

The integration of AI has fundamentally repositioned HRS platforms as intelligent, user-centricsystems that enhance both operational performance and the overall guest experience. Thepresent project incorporates several of these AI capabilities — including price prediction andsentiment analysis — as core functional components, the details of which will be elaborated insubsequent chapters.

5. Components of a Hotel Reservation System

As outlined in the preceding section, a modern Hotel Reservation System is not a singleapplication but an ecosystem of interoperable components, each fulfilling a distinct functionalrole. Together, these components form an integrated architecture in which data flowscontinuously and in real time, enabling accurate inventory management, seamless guestinteractions, and data-driven decision-making. This section describes each core component inturn.

5.1. Booking Engine

The booking engine serves as the primary guest-facing interface within the HRS architecture.Integrated directly into the hotel's web application or mobile application, it enables guests tocheck real-time room availability, view pricing, select room types, and submit reservations [20].Its core functions include displaying available inventory and associated rates, processingpayment transactions securely, promoting ancillary services such as room upgrades, andautomatically forwarding confirmed bookings to the hotel's internal systems.

Through API-based integrations, the booking engine achieves real-time synchronisation with CRS, Channel Manager, and Property Management System (PMS), ensuring that live inventory data is accurately reflected and that new bookings areimmediately distributed across all connected channels [20]. This integration eliminates the riskof data conflicts such as duplicate bookings and contributes directly to increased direct bookingrevenue.

A key component of a hotel reservation system is the booking engine, which serves as the guest user interface. The booking engine is directly integrated into the hotel's website (wep application) or mobile application. This engine enables guests to perform a set of actions, including checking real-time availability, viewing rates, selecting rooms, and submitting reservations. [20]

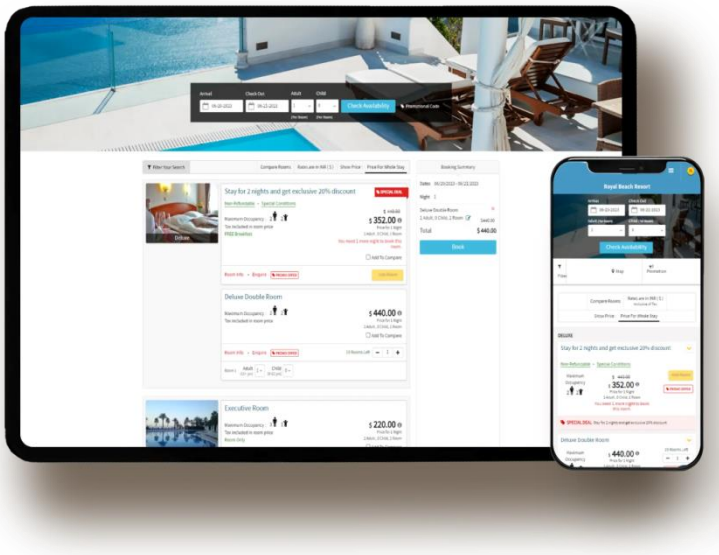


Figure 2.5: *Hotel Booking Engine*

5.2. Central Reservation System (CRS)

The CRS constitutes the core operational hub of the hotel reservation system, functioning as thesingle authoritative source of inventory, pricing, and booking data across all distributionchannels [21]. Its primary role is to aggregate incoming reservations from the booking engine,OTAs, and Global Distribution Systems (GDSs), and to distribute real-time updates to the PMS and Channel Manager. By maintaining pricing parity and availability synchronisation across allconnected systems, the CRS ensures that the HRS operates as a unified platform rather than acollection of disconnected modules [21].

In AI-driven reservation environments, the CRS serves an additional function: it supplies thestructured, historical data streams that power predictive models for revenue optimisation,demand forecasting, and dynamic pricing



Figure 2.6: *Architecture of the CRS and Its Integration with Hotel Distribution Channels*

5.3. Channel Manager

The channel manager functions as the distribution and synchronisation layer within the HRSarchitecture. It maintains bidirectional connectivity between the CRS/PMS and external saleschannels — including OTAs, metasearch engines, and direct booking engines — automaticallypropagating inventory changes, rate adjustments, and booking confirmations in real time. Byeliminating the need for manual updates across multiple platforms, the channel managerensures system-wide consistency, prevents overbooking, and feeds consolidated performancedata to the central analytics layer.

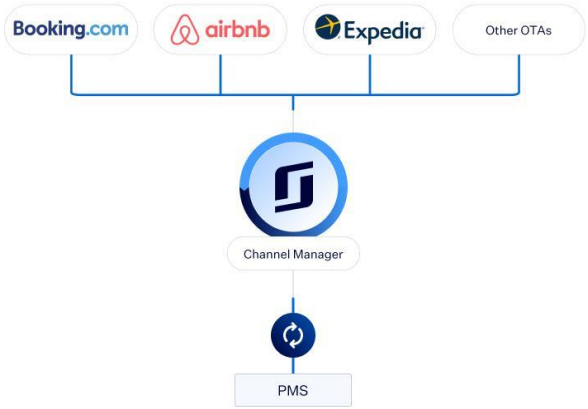


Figure 2.7: *Channel Manager Architecture for Real-Time Synchronization Between PMS and Online Distribution Channel*

5.4. Property Management System (PMS) Integration

PMS integration acts as the operational link connecting the reservation ecosystem to theproperty's internal workflow. Through bidirectional APIs, the integrated system receivesreservations from the booking engine and CRS, updates room status and guest data within the PMS to reflect current occupancy, and returns real-time availability and room service informationto the channel manager and booking engine [22]. This continuous data exchange eliminatesinformation gaps across the system, enabling seamless automation of check-in and check-outprocedures, invoicing, and inventory management at the property level.

5.5. Payment Gateway

The payment gateway is the secure transaction-processing component of the HRS. It integratesdirectly with the booking engine and PMS workflow to authorise, capture, and reconcilepayments in real time during the booking process or upon guest arrival [23]. The gatewaysupports multiple payment methods, including card payments and cash on arrival, whilemaintaining compliance with the Payment Card Industry Data Security Standard (PCI-DSS),which mandates the secure processing, encryption, and protection of cardholder data to preventfraud [23]. Upon successful transaction processing, the

gateway automatically updates guestaccounts and financial records across the CRS and reporting modules, ensuring financialaccuracy without manual intervention.



Figure 2.8: *Secure Payment Gateway Integration in a Hotel Reservation System*

5.6. Reporting and Analytics Tools

Reporting and analytics tools constitute the intelligence layer of the HRS, aggregatingtransactional data from all other components — the booking engine, CRS, Channel Manager,PMS, and payment gateway — to generate real-time dashboards and performance metrics [24].KPIs tracked by this component include the occupancy rate,Average Daily Rate (ADR), and Revenue per Available Room (RevPAR). ADR measures theaverage revenue earned per occupied room per day, while RevPAR combines occupancy rateand ADR to reflect overall revenue efficiency across the entire room inventory. Together, thesemetrics provide revenue managers with actionable insights for optimising pricing strategies anddistribution channel performance [24].



Figure 2.9: *Reporting and analytics tools interface for Hotel Rservation System*

5.7. Database Management

Database management forms the foundational data layer of the HRS, responsible for the securestorage and organisation of all reservation records, guest profiles, room inventories, pricingdata, historical bookings, and transaction logs [25]. Modern HRS deployments rely oncloud-based or unified database architectures that maintain a single, authoritative data sourceaccessible to all system components via APIs. This design ensures data consistency, supportsrapid querying and retrieval, and provides the accurate, structured datasets required to power AI-driven features such as price prediction and demand forecasting [25].



Figure 2.10: *Architecture of a Unified Hotel Database and its interaction with the rest of the system components*

5.8. System Integration: The Interconnected HRS Architecture

The components described above do not operate in isolation; they are designed to interoperatecontinuously through APIs and real-time synchronisation protocols. A reservation initiatedthrough the booking engine propagates immediately through the CRS and Channel Manager,updates the PMS via API integration, triggers payment processing through the gateway, recordsall transactional data in the unified database, and generates performance insights through thereporting and analytics tools. This tightly coupled, event-driven architecture is essential forbuilding a robust, AI-enhanced hotel reservation web application capable of delivering accuracy,operational efficiency, and sustained revenue growth. The following sections introduce the corefunctionalities built upon this architectural foundation, followed by an examination of AItechnologies relevant to the tourism and hospitality sector.

6. Core Functionalities of a Hotel Reservation System

The components described in the preceding section collectively enable a set of well-defined operational capabilities that a hotel reservation system must provide in order to function effectively. While components refer to the structural building blocks of the system architecture, functionalities refer to what the system is actually capable of doing

from both the guest and operator perspectives. Understanding these functionalities is essential for the analysis, design, and evaluation of any HRS, and forms the conceptual basis upon which the present system is developed.

The core functionalities of a hotel reservation system can be grouped into five interrelated categories: search and availability checking, booking and reservation management, user account management, payment processing, and reporting and analytics. Each is discussed in turn below.

Figure 2.11 provides an overview of these five functionalities and illustrates how they relate to one another within a unified reservation workflow.

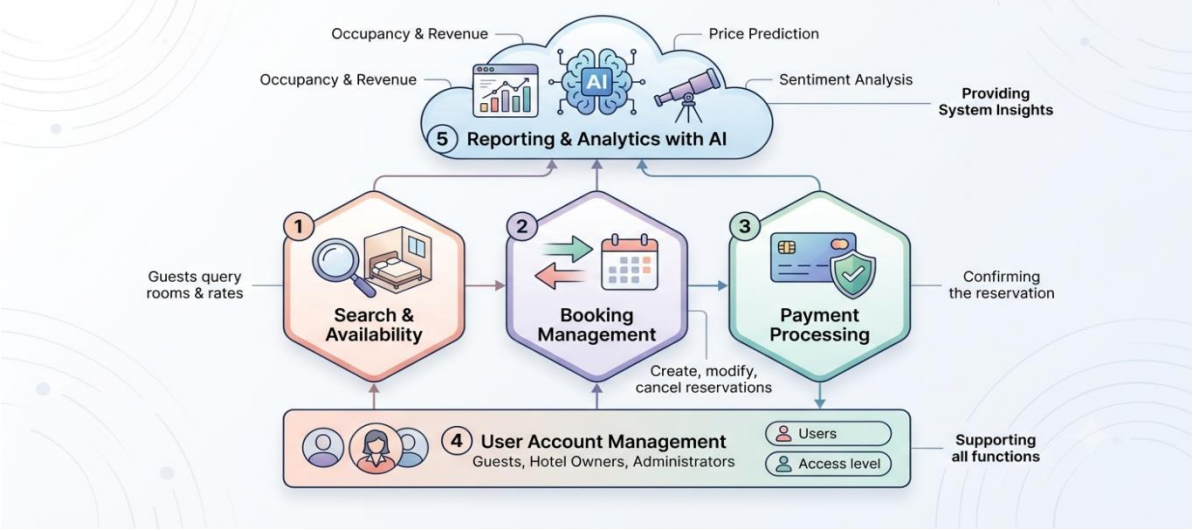


Figure 2.11: *The Five Core Functionalities of a Hotel Reservation System and Their Interrelation*

6.1. Search and Availability Checking

The search and availability checking functionality constitutes the entry point of any reservation workflow. It enables guests to query the system for available room types across a specified date range, filter results by price range, amenities, occupancy capacity, or hotel location, and receive real-time inventory information. From a technical perspective, this functionality depends on the continuous synchronisation of room inventory data across the booking engine, the CRS, and the channel manager, as established in Section 5. Any inconsistency in this data layer directly translates into incorrect availability information, with consequences including overbooking and guest dissatisfaction [9].

In web-based HRS implementations, the search interface typically presents results ranked by relevance or price, together with room type descriptions, photographs, and available amenities. The accuracy and responsiveness of this functionality are critical determinants of the booking conversion rate, as guests who encounter availability errors or slow query responses are likely to abandon the reservation process [21].

6.2. Booking and Reservation Management

Once a guest selects a room type and date range, the booking functionality manages the full lifecycle of a reservation, from initial creation through to confirmation, modification, and cancellation. At the point of booking, the system validates room availability in real time, assigns the reservation a unique identifier, captures all required guest information, and triggers an automated confirmation message delivered to the guest by email or in-application notification [1].

Reservation management also encompasses modification and cancellation workflows. Guests may need to update their check-in or check-out dates, change the room type, or cancel a reservation subject to the hotel's cancellation policy. The system must propagate these changes immediately across all connected components, including the PMS and channel manager, to ensure that room inventory is updated without delay [22]. In addition, the HRS must implement overbooking controls, which prevent the same room type from being booked simultaneously by multiple guests when no inventory remains. These controls typically operate through transaction-level locking mechanisms within the database layer [25].

For hotel operators, reservation management extends to administrative functions such as viewing all active bookings, filtering reservations by date, room type, or guest profile, managing group reservations, and generating booking reports. In the present system, this functionality is accessible through role-specific dashboards designed for hotel owners and system administrators, ensuring that each actor interacts only with the data and operations relevant to their role.

6.3. User Account Management and Role-Based Access

A modern HRS serves multiple categories of users simultaneously, each with distinct informational needs and operational permissions. Accordingly, user account management is a fundamental system functionality that encompasses registration, authentication, profile management, and role-based access control (RBAC) [26]. RBAC is a security model in which system permissions are assigned based on a user's defined role rather than on individual identity, thereby simplifying access governance and reducing the risk of unauthorised operations.

In the context of the present system, three primary roles are defined: guest users, who search for hotels, manage reservations, and submit reviews; hotel owners, who manage property listings, room types, pricing, and reservation data through a dedicated owner dashboard; and the system administrator, who oversees the entire platform, manages user accounts, and monitors system-level analytics through an admin dashboard. Each role is associated with a distinct set of permissions enforced at both the API and interface levels.

A core requirement of this functionality is the secure authentication of users, ensuring that each actor is reliably identified before being granted access to their designated operations. Equally important is the management of guest profiles, which maintain booking histories, personal preferences, and contact information. These profiles support personalised service delivery and provide the data foundation upon which the sentiment analysis and recommendation features of the system are built. The specific design of the authentication mechanism is addressed in the system design chapter.

6.4. Payment Processing

Payment processing is a transactional functionality that converts a provisional reservation into a confirmed booking by capturing and validating guest payment data. As described conceptually in Section 5.5, this functionality typically relies on a payment gateway that handles authorisation and confirmation. From a functionality perspective, the system must record the chosen payment method, provide immediate feedback on whether a reservation has been successfully confirmed, and generate a booking record or receipt accordingly [23].

In full-scale commercial deployments, this functionality is associated with strict security requirements, including encrypted data transmission and compliance with the PCI-DSS standard for the protection of cardholder data [23]. Within the scope of the present project, however, payment is handled through a simplified mechanism appropriate to the academic and prototypical nature of the system, rather than a full commercial gateway integration. This design choice keeps the focus of the work on the reservation workflow and the AI-driven functionalities, while still preserving the conceptual role of payment within the overall reservation lifecycle. The specific approach adopted is detailed in the relevant later chapters.

From the hotel owner's perspective, the payment functionality also relates to revenue tracking, enabling owners to view total revenue generated per period, per room type, or per reservation, in alignment with the revenue metrics defined in Section 2.6. These records feed directly into the reporting and analytics layer described below.

6.5. Reporting, Analytics, and AI-Driven Decision Support

Reporting and analytics represent the intelligence layer of the system's functionality, aggregating operational data to generate actionable insights for hotel owners and administrators. Standard reporting capabilities include the generation of occupancy rate summaries, revenue reports, booking trend visualisations, and channel performance indicators such as ADR and RevPAR, all of which were introduced in Section 5.6.

Beyond standard reporting, the present system extends this functionality with AI-driven decision support capabilities. The integration of a price prediction model enables the system to generate recommended room rate adjustments based on predicted demand patterns, historical occupancy data, and seasonal trends. Simultaneously, the sentiment

analysis module processes guest review text to produce polarity scores that indicate the overall satisfaction level associated with each property. These AI outputs are surfaced directly within the owner dashboard as actionable KPI elements, enabling hotel operators to make informed pricing and service improvement decisions without requiring specialised data science expertise.

This AI-augmented reporting functionality reflects the broader industry trajectory described in Section 4.5, in which HRS platforms are evolving from passive data repositories into proactive decision-support environments. The design of these AI components is addressed in Chapter 3, while their implementation, experimental results, and evaluation are presented in Chapter 4.

7. AI in Tourism and Hospitality

The integration of AI into the tourism and hospitality sector has emerged as one of the most consequential technological developments of the past decade. Driven by advances in machine learning (ML) [27], the branch of artificial intelligence concerned with algorithms that improve their performance on a task through exposure to data rather than through explicit programming, NLP, and data analytics, AI is progressively reshaping how tourism services are designed, delivered, and optimised. A growing body of academic research confirms that AI applications in this sector span a wide spectrum, from demand forecasting and dynamic pricing to personalised guest services and automated review analysis [28] [29].

This section situates the AI components of the present system within the broader landscape of AI adoption in the tourism and hospitality industry, providing the theoretical and empirical grounding for the design choices made in subsequent chapters.

7.1. The Scope of AI Adoption in Tourism and Hospitality

The adoption of AI in tourism and hospitality has accelerated considerably since the early 2010s, with the volume of published research in the field growing from a handful of studies per year to several hundred annually by 2024 [30]. A systematic review by Doborjeh et al. [27] covering 146 peer-reviewed articles published between 2010 and 2021 identified demand forecasting, destination behaviour analysis, customer service enhancement, and experience personalisation as the four dominant application domains, with ML algorithms, particularly ensemble methods [31], techniques that combine the predictions of multiple base models to achieve higher accuracy than any individual model alone, and deep learning (DL) [32] architectures, a class of ML methods based on multi-layered artificial neural networks capable of learning hierarchical feature representations directly from raw data, consistently outperforming traditional statistical models across all domains.

The tourism sector's particular suitability for AI-driven approaches stems from the inherent complexity and volatility of its demand dynamics. Hotel bookings are influenced

by a multitude of factors, including seasonality, local events, macroeconomic conditions, competitor pricing, lead time, and guest demographics, all of which interact in non-linear ways that traditional forecasting methods struggle to capture accurately [29]. AI models, and gradient boosting methods [33], an ensemble technique that builds an additive sequence of decision trees in which each new tree is trained to correct the residual errors of its predecessors, such as LightGBM [34] and XGBoost [35] in particular, are well-suited to learning these complex relationships from large historical datasets, generating predictions that adapt dynamically as new booking data becomes available [36].

7.2. Dynamic Pricing and Demand Forecasting in Hospitality

Among the most impactful applications of AI in the hospitality sector are dynamic pricing and demand forecasting, two closely related capabilities that directly support the revenue management function described earlier in this chapter. Dynamic pricing refers to the continuous adjustment of room rates in response to changing demand, competitive conditions, and other contextual factors, whereas demand forecasting concerns the prediction of future occupancy and booking volumes. Together, they allow hotels to align pricing with anticipated demand, thereby improving both occupancy rates and revenue performance [29].

The value of these capabilities in a hospitality context is considerable. Traditionally, pricing decisions were made manually by revenue managers who relied on historical occupancy curves and fixed seasonal rules, an approach that is slow to react and prone to leaving revenue unrealised during periods of fluctuating demand. AI-driven approaches address this limitation by learning demand patterns from historical booking data and producing pricing guidance that adapts as conditions change. For an individual hotel, this translates into more competitive rates, reduced reliance on intuition, and the ability to respond to demand shifts in a timely and systematic manner [29] . The academic literature consistently reports that ML models outperform conventional statistical techniques in predicting hotel prices and demand, owing to their capacity to capture the non-linear relationships among the many factors that influence booking behaviour [36] [37].

This general logic is summarised in Figure 2.12, which illustrates, at a conceptual level, how a range of demand-related factors is transformed into a recommended room rate.

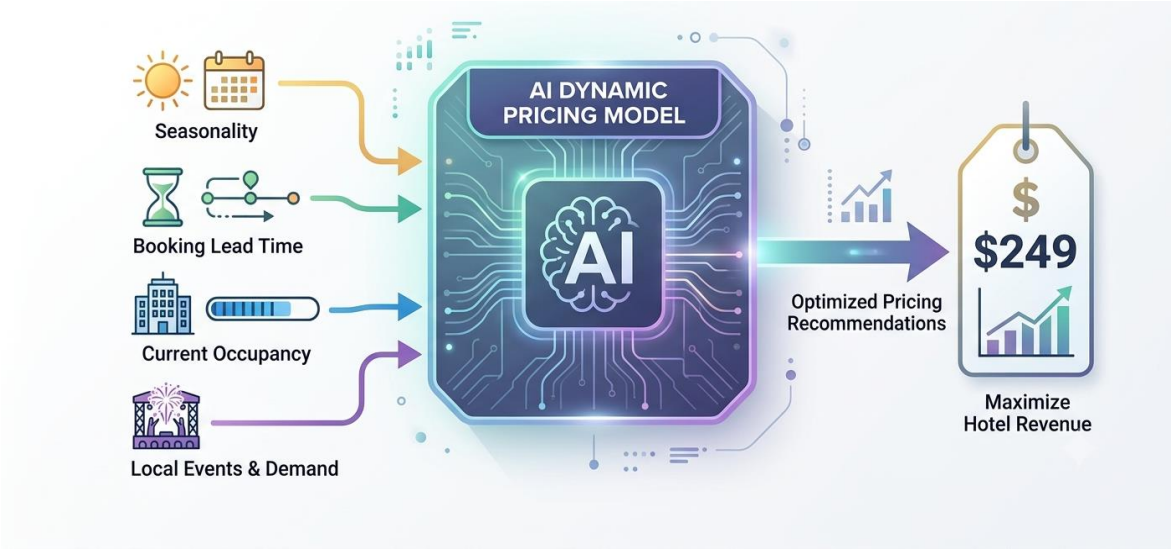


Figure 2.12: *Conceptual Flow of AI-Driven Dynamic Pricing in Hospitality*

In the present system, this capability is realised through a price prediction feature that assists hotel owners in setting appropriate room rates based on learned demand patterns. At this stage, it is sufficient to note that this feature is conceived as an intelligent decision-support tool embedded within the owner dashboard. The design of this feature is presented in Chapter 3, while its concrete realisation, experimental evaluation, and performance results are reported in Chapter 4.

7.3. Natural Language Processing and Sentiment Analysis of Guest Reviews

Online guest reviews have become a primary channel through which travellers communicate their satisfaction and service quality experiences. The volume of review data generated across OTA platforms and hotel websites is sufficiently large to preclude manual analysis at scale, making automated sentiment analysis an operationally significant tool for hotel managers. Sentiment analysis, as a subfield of NLP, refers to the computational identification of the emotional polarity expressed within a text, typically classified as positive, neutral, or negative, and is the task to which the second AI component of the present system is addressed [38].

The evolution of sentiment analysis methods in the hospitality domain mirrors broader advances in NLP. Early approaches relied on lexicon-based methods and shallow ML classifiers such as Support Vector Machines (SVM) [39], a supervised algorithm that classifies data by finding the hyperplane that maximally separates classes in feature space, and Naive Bayes [40], a probabilistic classifier that applies Bayes' theorem under the simplifying assumption of conditional independence between features, which achieved reasonable performance on balanced datasets but proved brittle when faced with domain-specific vocabulary, sarcasm, or code-switching [38]. The introduction of transformer-based [41] pre-trained language models, the transformer being a neural network architecture that relies on self-attention mechanisms to model relationships

between all tokens of a sequence in parallel, beginning with Bidirectional Encoder Representations from Transformers (BERT) [42] in 2018, represented a paradigm shift in the field. By learning contextualised token representations from large unlabelled text corpora, BERT and its lighter and domain-adapted derivative architectures substantially outperform their predecessors on hotel review classification tasks [43].

Wen et al. [38] demonstrated that BERT-based models applied to hotel review sentiment classification significantly outperform both conventional NLP baselines and Long Short-Term Memory (LSTM) [44]-based sequence models, a recurrent neural network architecture designed to capture long-range dependencies in sequential data, achieving high accuracy scores on Chinese hotel review datasets while maintaining competitive inference latency. This finding is consistent with results reported across multiple languages and hotel review corpora, establishing transformer-based sentiment classifiers as the current empirical benchmark for this task [38] [43]. A comprehensive systematic review of sentiment analysis for hotel reviews by Abdelgwad et al. [38] further confirmed that DL models, particularly those based on contextualised embeddings, now dominate performance rankings across all standard evaluation metrics.

Figure 2.13 illustrates, in conceptual terms, how a raw guest review is processed by a pre-trained language model to produce a structured sentiment classification.



Figure 2.13: *Conceptual Flow of Sentiment Analysis of Guest Reviews Using a Pre-Trained Language Model (BERT)*

Building on this body of work, the present system includes a sentiment analysis feature that automatically classifies the guest reviews submitted through the platform according to their expressed polarity. The purpose of this feature is to transform the unstructured textual feedback left by guests into a structured indicator of satisfaction that hotel owners can interpret at a glance, complementing the quantitative performance indicators generated by the reporting and analytics component. In this way, the sentiment analysis feature contributes to a more complete, evidence-based picture of service quality for

each property. The design of this feature is elaborated in Chapter 3, and the details of its realisation together with its experimental evaluation are reported in Chapter 4.

7.4. The Convergence of AI and Web-Based HRS: Rationale for the Present System

The preceding subsections have established that both dynamic pricing and sentiment analysis are mature, empirically validated AI applications within the tourism and hospitality domain, with well-established methodological precedents and demonstrable operational benefits. The central contribution of the present system is to integrate both capabilities within a unified, web-based hotel reservation platform that serves multiple user roles simultaneously.

This integration addresses a gap that remains common in practice: many hotels either rely on AI-powered pricing tools that are decoupled from their reservation systems, or they possess review data without the analytical infrastructure to extract actionable insights from it. By embedding intelligent decision-support directly within the reservation platform itself, rather than treating it as a separate, external tool, the present system aims to establish a coherent link between the operational data generated by the reservation process and the AI-driven insights delivered to hotel owners. In this way, the data produced through everyday bookings and guest reviews is continuously translated into pricing guidance and satisfaction indicators that support better-informed management decisions.

Conceptually, keeping the intelligent components modular and well-separated from the core reservation logic ensures that the system remains extensible: additional AI models or analytical features can be incorporated in future iterations without disrupting the fundamental booking workflow. This separation of concerns is consistent with established practice for AI-integrated web applications [28] and supports the scalability required for a system intended to serve multiple hotels across diverse markets. The specific technologies and architectural decisions through which this modularity is achieved are presented in the subsequent chapters.

The system design, data modelling decisions, and AI component specifications that emerge from this theoretical grounding are presented in the following chapters.

8. Conclusion

This chapter has established the conceptual and theoretical foundation required to understand, analyse, and develop an AI-powered hotel reservation system. Beginning with the fundamental concepts of the hospitality domain, it clarified the meaning of hotels, reservations, room types, availability, occupancy rate, and revenue, providing the precise vocabulary that underpins the remainder of this work. These concepts are not isolated definitions; rather, they are the interconnected elements upon which every reservation system, from the simplest to the most sophisticated, ultimately depends.

The chapter then examined hotel reservation systems as integrated platforms, reviewing their principal types, their importance and benefits within the tourism sector, and the historical trajectory through which they have evolved. This evolution, from labour-intensive manual methods to centralised, computerised, internet-based, and ultimately AI-powered systems, revealed a consistent trend towards greater automation, accuracy, and intelligence. Understanding this progression is essential, as the present project positions itself as a continuation of this trajectory, building upon the achievements of internet-based systems while extending them with modern AI capabilities.

A detailed examination of the components and core functionalities of a hotel reservation system demonstrated that such a system is not a single application but an ecosystem of interoperable parts working in concert to deliver a seamless reservation experience. The chapter highlighted how these components and functionalities interconnect through real-time data exchange, and how this integrated architecture provides the foundation upon which intelligent features can be layered.

Finally, the chapter explored the role of AI in tourism and hospitality, focusing on the two capabilities most relevant to this project: dynamic pricing and the sentiment analysis of guest reviews. The academic literature confirms that both are mature, empirically validated applications with clear operational value, providing strong justification for their inclusion in the present system. By embedding these AI capabilities within a unified, web-based reservation platform, the system developed in this work aims to bridge a gap that remains common in practice, transforming operational data into actionable, intelligent decision-support.

With this conceptual foundation now in place, the following chapter shifts the focus from background and theory to the intelligent components and AI modelling that form the core innovation of this work, before proceeding to the analysis, design, and implementation of the complete system in the subsequent chapters.

CHAPTER 03



SYSTEM ANALYSIS & DESIGN

1. Introduction

Chapter 2 established the conceptual and technical foundations of this work and surveyed the relevant literature, framing the problem that the Hotelify platform sets out to address. Building directly upon those foundations, the present chapter turns from understanding the problem to engineering a solution, presenting the complete analysis and design of the system. Its purpose is to specify what the system must do and to determine, in a rigorous and technology-independent manner, how it should be structured in order to do it, thereby producing the blueprint that the subsequent implementation will follow.

The chapter is organised as a progression from requirements toward a fully specified design. It opens with the system analysis, which identifies the actors of the system and elicits its functional and non-functional requirements. On this basis, the system is modelled using the Unified Modeling Language (UML), through the use case, class, sequence, and activity diagrams that capture its behaviour and its structure. The design of the database then translates this model into a concrete organisation of the system's data. Attention next moves to the overall system architecture and to the way the AI service is integrated within it. The design of the two AI models follows, presented as a pair of pipelines that set out the methodology by which each model is conceived. Finally, the chapter describes the pipeline through which the user interface (UI) of the system is designed.

Throughout, the chapter remains deliberately at the level of design. It is concerned with the algorithms, techniques, structures, and methods that define the solution, whereas the concrete tools, technologies, and results that bring the design to life are reserved for Chapter 4. This separation keeps the design reasoning independent of any particular implementation and allows it to be assessed on its own terms.

2. System Analysis

System analysis is the foundational phase of any software engineering project, in which the problem domain is studied and the expectations placed upon the system are identified and formalised before any design or implementation begins. Its purpose is to answer the question of what the system must accomplish, independently of how these goals will later be realised. This section presents the analysis of the proposed system, Hotelify, beginning with a general overview of the system, followed by an identification of its actors and their roles, and concluding with a structured specification of its functional and non-functional requirements.

2.1. Overview of the Proposed System (Hotelify)

Hotelify is the AI-powered web application developed in the context of this project, representing the practical realisation of the system entitled ***AI-Powered Web Application for Hotel Reservation System***. It serves simultaneously as the working prototype of the research and as the commercial identity under which the system is presented. The objective of Hotelify is to provide a complete, web-based hotel reservation platform that not only supports the full reservation lifecycle, from searching for accommodation to confirming a booking, but also augments this process with intelligent, data-driven capabilities that distinguish it from conventional reservation systems.



Figure 3.1: *Hotelify Logo*

At its core, Hotelify connects three categories of users within a single integrated environment: guests seeking to reserve accommodation, hotel owners managing their properties, and a system administrator overseeing the entire platform. Guests are able to browse hotels, examine room types, check availability, and make reservations, while also benefiting from intelligent pricing guidance and consulting the experiences of previous guests. Hotel owners are provided with a dedicated dashboard through which they manage their hotel, its room types, pricing, and reservations, supported by analytical insights into its performance. The administrator supervises the platform as a whole, managing user accounts and monitoring system-wide activity.

What sets Hotelify apart from a traditional reservation system is the integration of two AI capabilities directly into its workflow. The first is a price prediction and recommendation capability, which forecasts the future evolution of room prices and translates these forecasts into actionable recommendations for both guests and owners. The second is a sentiment analysis capability, which automatically interprets the textual reviews submitted by guests and converts them into structured indicators of satisfaction. Together, these capabilities transform Hotelify from a passive booking tool into an intelligent decision-support system. In addition, the platform incorporates a set of automated background processes that maintain the consistency of its data and the currency of its predictions without manual intervention. The detailed analysis of these features, beginning with the system's actors, is presented in the subsections that follow.

2.2. System Actors and Their Roles

An actor is any external entity that interacts with the system to achieve a particular goal. In Hotelify, three primary human actors are identified, each associated with a distinct set of responsibilities and access privileges. The clear definition of these actors is essential, as it governs the functional requirements, the access control model, and the structure of the user interfaces presented later in this chapter.

❖ Guest

The guest is the principal end user of the platform and represents the demand side of the reservation process. A guest may register an account, authenticate, and manage a personal profile. Once authenticated, a guest can search for hotels, browse available room types, consult price predictions and the associated booking recommendations, make and manage reservations, and submit textual reviews of the hotels in which they have stayed. The guest therefore interacts primarily with the booking workflow and with the guest-oriented intelligent features of the system.

❖ Hotel Owner

The hotel owner represents the supply side of the platform and is responsible for a single hotel listed on Hotelify, as the designation itself implies. Through a dedicated owner dashboard, the owner manages their hotel in its entirety, including its description, facilities, images, and room types, and sets the pricing for each room type. The owner consults the reservation data relating to their hotel, examines analytical reports on its performance, and makes use of the price prediction and recommendation capability to inform decisions regarding the adjustment of room-type prices. The owner additionally has access to the sentiment indicators derived from the reviews of their hotel. The scope of the owner's authority is strictly limited to the single property they own.

❖ Administrator

The administrator is the supervisory actor responsible for the integrity and oversight of the platform as a whole. Through an administrative dashboard, the administrator manages user accounts, oversees the hotels registered on the platform, monitors reservations across all properties, and consults platform-wide analytical reports, including aggregated AI insights. The administrator operates at the highest level of privilege within the system, with a scope that spans the entire platform rather than any single property.

The distinction between these three actors, and in particular the strict separation between the property-level scope of the owner and the platform-level scope of the administrator, forms the conceptual basis of the RBAC model that secures the system, as elaborated in the non-functional requirements.

2.3. Functional Requirements

The functional requirements specify the concrete services and operations that Hotelify must provide. They describe the observable behaviour of the system in response to user actions and internal events. For clarity, the functional requirements are organised according to the actor or subsystem to which they relate.

❖ Authentication and Account Management

The system shall allow users to register, authenticate, and manage their accounts. Specifically, the system shall enable account registration, secure login and logout, the retrieval of the authenticated user's profile, and a complete password reset procedure through which a user may recover access to their account by means of a secure, single-use reset link delivered to their registered email address.

❖ Guest Functional Requirements

The system shall allow a guest to search for hotels according to criteria such as location and dates, to browse the room types offered by each hotel together with their availability and pricing, and, for a chosen check-in date, to obtain a price prediction that constitutes in itself a booking recommendation. This prediction expresses the expected price trend, the percentage change relative to the current price, and a supporting explanation, on the basis of which the guest decides whether to book immediately or to defer the reservation. The system shall further allow a guest to create a reservation, to view and manage their existing reservations, to cancel a reservation in accordance with the applicable conditions, and to submit a textual review of a hotel following their stay.

❖ Hotel Owner Functional Requirements

The system shall allow a hotel owner to manage the hotel they own, including the management of property details, facilities, images, and room types, and to define the price associated with each room type. The system shall allow the owner to view and manage the reservations relating to their hotel and to consult analytical reports on its performance. For a date of the owner's choosing, the system shall provide a price prediction that, expressing the expected price trend, the percentage change relative to the current price, and a supporting explanation, constitutes in itself the recommendation on which the owner relies to decide whether to adjust the prices of their room types. The system shall also present the owner with the sentiment indicators derived from the reviews of their hotel.

❖ Administrator Functional Requirements

The system shall allow the administrator to manage the user accounts registered on the platform, to oversee the hotels listed by owners, to monitor reservations across all properties, and to consult platform-wide analytical reports. These reports shall include aggregated KPIs and AI insights spanning all hotels on the platform.

❖ AI Functional Requirements

The system shall provide a price prediction and recommendation service that forecasts the future evolution of room-type prices and derives from these forecasts an interpretable recommendation, accompanied by a supporting explanation, indicating whether the price trend for a given date is expected to

increase, decrease, or remain stable. The system shall additionally provide a sentiment analysis service that classifies each guest review according to its expressed polarity and makes the resulting indicators available to owners and to the administrator.

❖ Automated (Scheduled) Functional Requirements

Beyond the operations initiated directly by users, the system shall perform a number of automated background processes on a predefined schedule. The system shall automatically check out reservations whose departure time has elapsed, executing this process every hour. The system shall automatically regenerate the room-price predictions on a daily basis, at midnight local time, so that the predictions presented to users remain current. The system shall likewise update the historical record of room prices on a daily basis, at midnight local time, in order to maintain the temporal data upon which the prediction capability depends. These scheduled processes ensure that the data and intelligent outputs of the system remain consistent and up to date without requiring any manual intervention.

2.4. Non-Functional Requirements

Whereas the functional requirements describe the services the system provides, the non-functional requirements describe the quality attributes that govern how the system delivers those services. They constrain and characterise the behaviour of the system rather than defining new operations. The principal non-functional requirements of Hotelify are presented below.

❖ Security

Security is a paramount concern for Hotelify, given that it manages personal data, reservation records, and authentication credentials. The system adopts a stateless, token-based authentication mechanism [45], in which access is granted through signed tokens, complemented by a token rotation strategy [46] that issues a fresh credential upon each renewal and thereby enables the detection of token reuse. Authentication tokens are never stored in plain form; only their hashed representations [47], irreversible fixed-length digests computed from the original token, are retained, and their consumption is performed atomically [48], as a single indivisible operation that cannot be interleaved with a concurrent request, to prevent race conditions. Access to the operations of the system is further governed by a RBAC model, which grants each request only the privileges associated with the role of the requesting actor, enforcing in particular the separation between the property-level scope of owners and the platform-level scope of the administrator. At the level of the web layer, the system applies a set of protective HTTP security headers [49], directives returned in the server's response that instruct the browser to restrict certain behaviours known to enable

common web attacks, to mitigate common web vulnerabilities, and it regulates cross-origin access [50] so that only authorised client origins are permitted to communicate with the application interface. Communication with the internal AI service is confined to the trusted server environment rather than being exposed publicly.

❖ Performance and Scalability

The system is expected to respond to user interactions with minimal latency, ensuring that searches, reservations, and the consultation of intelligent insights are processed promptly. To this end, the communication between the application backend and the AI service is designed around a batch processing approach for the large volume of records handled during the daily prediction cycle, in preference to processing records individually. The architecture is conceived to accommodate multiple hotels across diverse geographical markets, and the modular separation of the AI capability into an independent service supports the future scaling and extension of the system without disruption to its core reservation logic.

❖ Reliability and Data Consistency

The system must preserve the integrity and consistency of its data at all times. The automated scheduled processes contribute directly to this requirement by ensuring that reservation statuses, price predictions, and price histories are maintained automatically and remain mutually consistent. Critical operations affecting authentication credentials are performed atomically in order to avoid inconsistent intermediate states.

❖ Usability

The system must offer an intuitive and coherent user experience across its three categories of users. Each actor is presented with an interface tailored to their role and responsibilities, exposing only the operations and information relevant to that role, thereby reducing complexity and the likelihood of error. The interface is designed to be clear, consistent, and responsive across the range of devices through which the platform may be accessed.

❖ Maintainability and Modularity

The system is structured so that its principal concerns are clearly separated, with the reservation logic, the data layer, and the AI capability organised as distinct, loosely coupled parts. This modular organisation facilitates the maintenance, testing, and future evolution of the system, and permits the AI models to be updated or replaced without necessitating changes to the reservation workflow.

❖ **Portability**

The system is designed to operate consistently across different deployment environments. The organisation of its services into self-contained, independently deployable units supports its portability and ensures that it can be provisioned reliably in varied settings, an aspect whose concrete realisation is addressed in the implementation chapter.

3. System Modeling (UML Diagrams)

Following the analysis of the system's requirements, this section presents the modelling of Hotelify using the Unified Modeling Language (UML) [51]. UML is a standardised, general-purpose visual modelling language widely adopted in software engineering for specifying, visualising, and documenting the artefacts of a software system. It provides a collection of diagram types that capture both the structural and the behavioural aspects of a system, thereby offering a precise and communicable representation that bridges the gap between requirement analysis and system design.

Figure 3.15 illustrates the classification map of different UML diagrams for each type, capturing either the structural or behavioral aspect of the system.

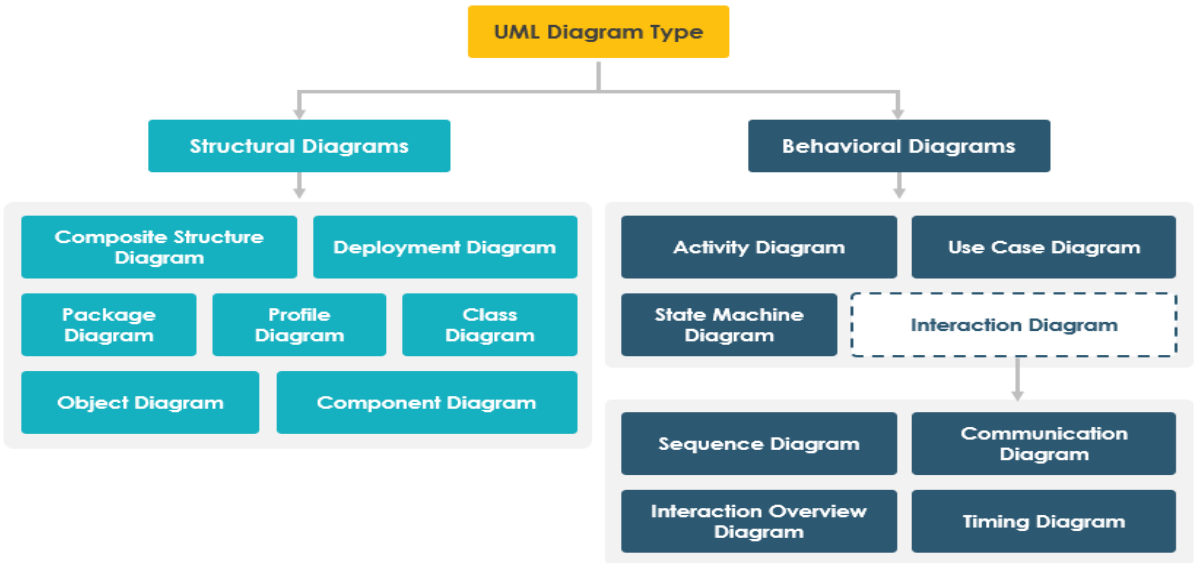


Figure 3.2: *Classification Map of Different UML Diagrams*

The UML diagrams presented in this chapter were produced using StarUML [52], a sophisticated and widely used modelling environment that supports the full range of UML diagram types and conforms to the UML specification. StarUML was selected for its robust support of use case, class, sequence, and activity diagrams, its clear and professional rendering, and its ability to maintain consistency between the model and its graphical representation.

In the present work, four different types of complementary UML diagrams are used to model the system. The use case diagram captures the functional interactions between

the actors and the system; the class diagram describes the static structure of the system's principal entities; the sequence diagrams represent the dynamic interactions underlying selected key scenarios; and the activity diagram models the workflow of the reservation process. Together, these diagrams provide a comprehensive view of Hotelify from both the functional and the structural perspectives.

3.1. Use Case Diagram

The use case diagram is a behavioural UML diagram that describes the functionality of a system from the perspective of its users. It depicts the actors who interact with the system, the use cases that represent the services the system offers, and the relationships that connect them [51]. As such, it constitutes a direct visual translation of the functional requirements identified in the system analysis, and serves as the foundation upon which the subsequent structural and behavioural models are built.

Figure 3.16 presents the complete use case diagram of Hotelify, encompassing all of its actors and their associated use cases. The diagram and its constituent elements are described in the remainder of this subsection.

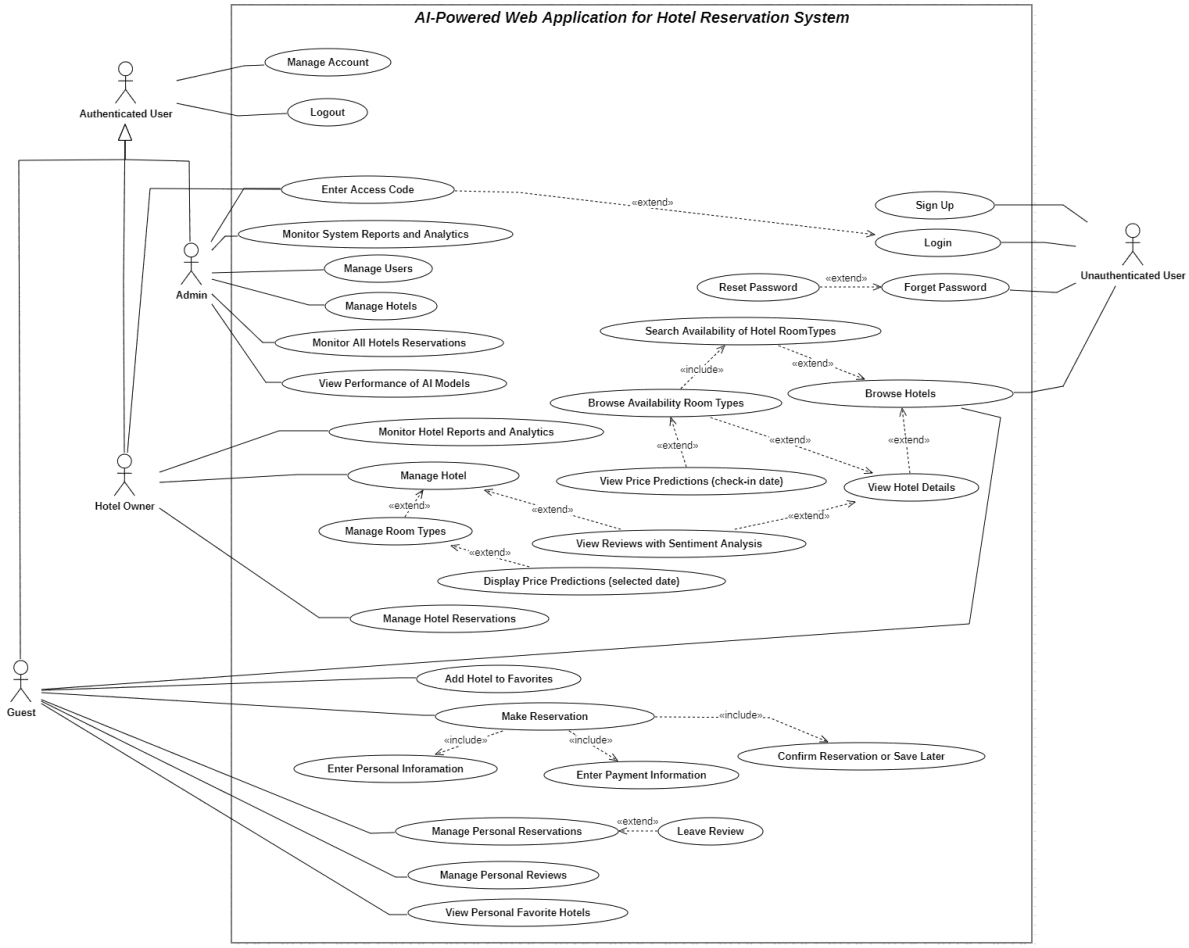


Figure 3.3: *Use Case Diagram of the Hotelify System*

3.1.1. Actors

The diagram identifies five actors, organised into a generalisation hierarchy that reflects the progressive levels of access within the system.

- **The Unauthenticated User:** represents any visitor who interacts with the platform without having logged in. This actor is able to perform the public operations of the system, namely browsing hotels and the services that branch from it, as well as initiating the authentication operations of logging in, signing up, and recovering a forgotten password.
- **The Authenticated User:** is an abstract actor that represents any user who has successfully logged into the system. It is not instantiated directly but serves as a generalisation of the three concrete authenticated roles. This actor groups the operations common to every logged-in user, specifically logging out and managing one's account.
- **The Guest, Hotel Owner, and Admin**: are the three concrete authenticated actors, each inheriting the common authenticated operations through a generalisation relationship with the Authenticated User. The Guest is the end user who searches for and reserves accommodation; the Hotel Owner manages the single hotel they own; and the Admin supervises the platform as a whole. The roles and responsibilities of each were detailed in the system analysis (Section 2.2).

3.1.2. Relationships

The diagram employs the four standard relationships defined by UML for use case modelling: association, generalisation, inclusion, and extension.

- **Association Relationships:** connect an actor to the use cases it directly participates in. For example, the Guest is associated with the use cases of making a reservation, managing personal reservations, leaving a review, and adding a hotel to favourites, while the Admin is associated with managing users, managing hotels, and monitoring system-wide reports and analytics.
- **Generalisation Relationships:** are used both between actors and, in one case, between use cases. Between actors, the Guest, Hotel Owner, and Admin each generalise the Authenticated User, thereby inheriting its common operations. This hierarchical organisation avoids the repetition of shared authentication operations across the three roles and yields a cleaner, more maintainable model.
- **Inclusion («include») Relationships:** express mandatory sub-behaviour that is always executed as part of a base use case. The most significant example is the Make Reservation use case, which includes the Enter Personal Information, Enter Payment Information, and Confirm Reservation or Save Later use cases, indicating that these steps are invariably performed whenever a reservation is made. Similarly, Browse Availability Room Types includes Search Availability of Hotel RoomTypes.

- **Extension («extend») Relationships:** express optional or conditional behaviour that augments a base use case only under specific circumstances. Several extension relationships structure the browsing experience of the system: View Hotel Details extends Browse Hotels, Browse Availability Room Types extends View Hotel Details, and View Price Predictions extends Browse Availability Room Types. This chain reflects the natural progression of a user navigating from the list of hotels down to the detailed availability and pricing of individual room types. The conditional nature of these operations is appropriately captured by the extension relationship, since a user may browse hotels without necessarily proceeding to view the details, availability, or price predictions of any particular property.
  Two extension relationships merit particular attention as they embody the AI-driven character of the system. The View Reviews with Sentiment Analysis use case extends both View Hotel Details and Manage Hotel, reflecting the fact that the sentiment-annotated reviews are consulted both by visitors and guests when examining a hotel and by the owner when managing their property. Likewise, the Enter Access Code use case extends Login and is associated exclusively with the Hotel Owner and the Admin, expressing that the entry of an access code is a conditional step in the login process required only for these two privileged roles, and never for an ordinary guest.

3.1.3. Key Use Cases

While the diagram comprises a substantial number of use cases, several are central to the distinctive purpose of the system and warrant a brief description.

- **The Make Reservation Use Case:** constitutes the core transactional functionality of the platform. It mandates the entry of personal and payment information and the confirmation of the reservation, ensuring that every booking follows a complete and consistent process.
- **The View Price Predictions (Check-In Date) and Display Price Predictions (Selected Date) Use Cases:** embody the price prediction and recommendation capability of the system. The former is available to guests and visitors, who are presented with the price prediction automatically upon selecting a check-in date, while the latter is reserved for the Hotel Owner, who must explicitly select a date in order to obtain the prediction used to inform pricing decisions. The deliberate separation of these two use cases reflects the distinct manner in which each role accesses the prediction within the system.
- **The View Reviews with Sentiment Analysis Use Case:** represents the sentiment analysis capability, through which the textual reviews submitted by guests are automatically classified and presented as structured satisfaction indicators, consulted by visitors, guests, and owners alike.

Together, these use cases translate the functional requirements of Section 2.3 into a coherent behavioural model, establishing the basis for the structural modelling presented in the following subsection.

3.2. Class Diagram

The class diagram is the principal structural diagram in UML. It describes the static architecture of a system by representing its classes, their attributes and operations, and the relationships that connect them [51]. Whereas the use case diagram presented in the previous subsection captures what the system does from the perspective of its actors, the class diagram captures how the system is structured internally, defining the entities that constitute its data model and the manner in which they relate to one another. As such, it forms the conceptual bridge between the functional analysis and the design of the system's database, which is presented in the following section.

Figure 3.17 presents the class diagram of the Hotelify system. It comprises fifteen classes, including two embedded classes, together with eight enumerations that define the constrained value sets used throughout the model. The diagram and its constituent elements are described in the remainder of this subsection.

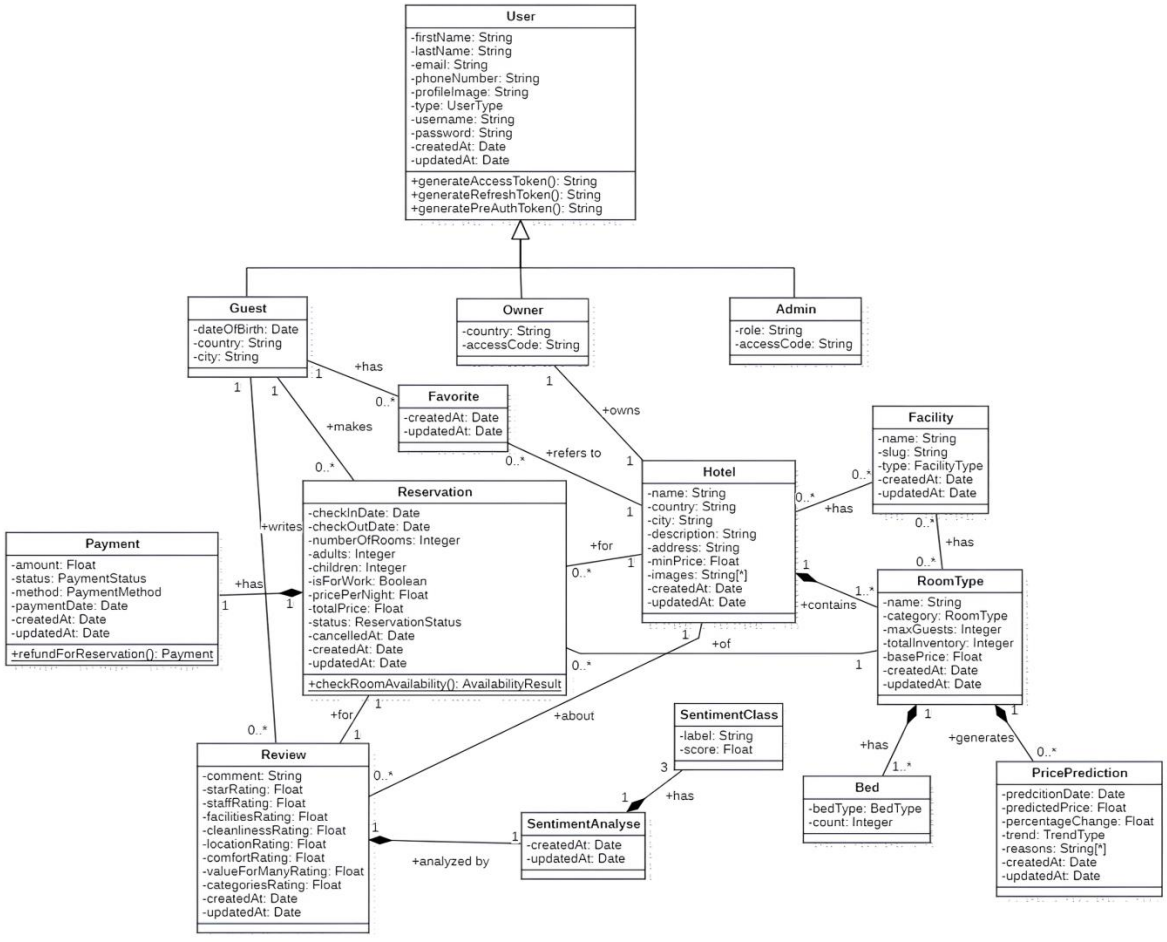


Figure 3.4: *Class Diagram of the Hotelify System*

3.2.1. Classes and Their Responsibilities

The classes of the system can be grouped, according to their roles, into user classes, core domain classes, AI classes, and supporting classes.

- **The User Classes:** model the actors of the system. The User class is the central abstraction that holds the attributes common to every user, namely the personal identification details (first name, last name, email, phone number, username), the authentication credential (password), the profile image, and the user type. It also defines three operations responsible for issuing the authentication tokens of the system: generateAccessToken, generateRefreshToken, and generatePreAuthToken. Three specialised classes inherit from User through a generalisation relationship: the Guest class, which adds the date of birth and place of residence; the Owner class, which adds the country and the access code required for privileged login; and the Admin class, which adds the administrative role and its own access code. This inheritance hierarchy reflects the three roles identified during the system analysis while avoiding the duplication of the shared user attributes.
- **The Core Domain Classes:** model the central business entities of the reservation process. The Hotel class represents a property listed on the platform, with its descriptive attributes (name, location, description, address, images) and its minimum price. The RoomType class represents a category of room within a hotel, defined by its name, category, capacity, total inventory, and base price. The Reservation class represents a booking made by a guest, holding the stay details (check-in and check-out dates, number of rooms, occupancy, and whether the stay is for work), the pricing information, and the reservation status. It additionally defines the static operation checkRoomAvailability, which determines the availability of a room type for a given period. The Payment class records the financial transaction associated with a reservation, comprising the amount, status, method, and payment date, together with the static operation refundForReservation, which processes refunds for cancelled bookings. The Review class captures a guest's evaluation of a hotel following their stay, holding the textual comment, the overall star rating, and a set of category-specific ratings, all expressed as real-valued scores to accommodate fractional averages.
- **The AI Classes:** model the outputs of the two AI capabilities of the system. The PricePrediction class stores the result of the price prediction and recommendation model, comprising the predicted price for a given date, the percentage change relative to the current price, the predicted trend, and the list of reasons that explain and justify the prediction. The SentimentAnalyse class stores the result of the sentiment analysis model applied to a guest review. Its sentiment scores are modelled through the embedded SentimentClass, which captures the label and confidence score of a sentiment category, instantiated three times to represent the positive, negative, and neutral classes.
- **The Supporting Classes:** complete the model. The Favorite class is an associative class that records a hotel a guest has marked as a favourite. The Facility class

represents an amenity that may be offered by a hotel or a room type. Finally, the embedded Bed class describes a bed configuration within a room type, capturing the bed type and the number of beds of that type.

3.2.2. Enumerations

To enforce constrained sets of permissible values and to improve the clarity and type-safety of the model, eight enumerations are defined. The UserType enumeration distinguishes the three user roles (guest, owner, admin). The ReservationStatus enumeration defines the lifecycle states of a reservation (pending, confirmed, cancelled, checked-in, checked-out, and no-show). The PaymentStatus and PaymentMethod enumerations define, respectively, the states of a payment (unpaid, paid, refunded) and the means by which it may be settled (card or payment at the property). The RoomCategory enumeration defines the available room classifications, and the BedType enumeration the permissible bed configurations. The TrendType enumeration defines the three possible directions of a price prediction (increase, decrease, stable), and the FacilityType enumeration distinguishes facilities applicable to a hotel from those applicable to a room type.

3.2.3. Relationships

The classes of the diagram are connected through three categories of relationship: generalisation, composition, and association.

- **Generalisation:** is used to model the inheritance of the three concrete user roles from the abstract User class, as described above. The Guest, Owner, and Admin classes each specialise User, inheriting its attributes and operations.
- **Composition:** is used where a part cannot meaningfully exist independently of the whole that contains it. Several composition relationships are present in the model. A Hotel is composed of one or more RoomTypes, since a room type has no existence outside the hotel to which it belongs. A RoomType is composed of one or more Beds and generates zero or more PricePredictions, both of which are entirely dependent on the room type. A Reservation is composed of exactly one Payment, and a Review is composed of exactly one SentimentAnalyse, which is in turn composed of three SentimentClass instances. In each case, the lifecycle of the part is bound to that of its containing whole.
- **Association:** is used to model the remaining relationships between independent entities, each annotated with a descriptive name and the appropriate multiplicities. An Owner owns exactly one Hotel, reflecting the design decision that each owner is responsible for a single property, created on their behalf by the administrator. A Guest makes zero or more Reservations, each of which is for exactly one Hotel and of exactly one RoomType. A Guest writes zero or more Reviews, each about one Hotel and corresponding to exactly one Reservation. A Guest may also mark hotels as favourites, modelled through the Favorite associative class, which links one Guest to one Hotel. Finally, both Hotel and RoomType maintain a many-to-many association with Facility, since a given

amenity may be offered by many hotels or room types, and each hotel or room type may offer many amenities.

Together, these classes, enumerations, and relationships provide a complete and precise structural model of the Hotelify system. This model directly informs the design of the database, presented in the following section, in which the classes are translated into the collections and documents of the system's data store.

3.3. Sequence Diagrams

A sequence diagram is a behavioural UML diagram that models the dynamic interactions between the objects of a system over time. It depicts the participating entities as vertical lifelines and represents the messages exchanged between them in chronological order, from top to bottom [51]. Whereas the use case diagram identifies what the system does and the class diagram describes how it is structured, the sequence diagram reveals how the system behaves when a particular scenario is executed, making explicit the order of operations and the collaboration between components.

Given the breadth of the Hotelify system, it is neither necessary nor practical to model every possible interaction. Instead, three representative scenarios were selected, each of which captures a distinct and significant aspect of the system's behaviour: the authentication and token management process, the price prediction workflow, and the review and sentiment analysis process. These scenarios were chosen because they exercise the most characteristic and technically meaningful interactions of the platform, including those that involve its AI capabilities. Each scenario is presented and described in the following subsections.

3.3.1. Scenario 1: Authentication and Token Management

The first scenario models the authentication process of the system, together with the management of the security tokens that govern user sessions. This scenario is of particular importance because Hotelify implements a two-tier authentication mechanism: a single-step process for guests and a two-step process, reinforced by an access code, for the privileged roles of owner and administrator. Figure 3.18 presents the corresponding sequence diagram.

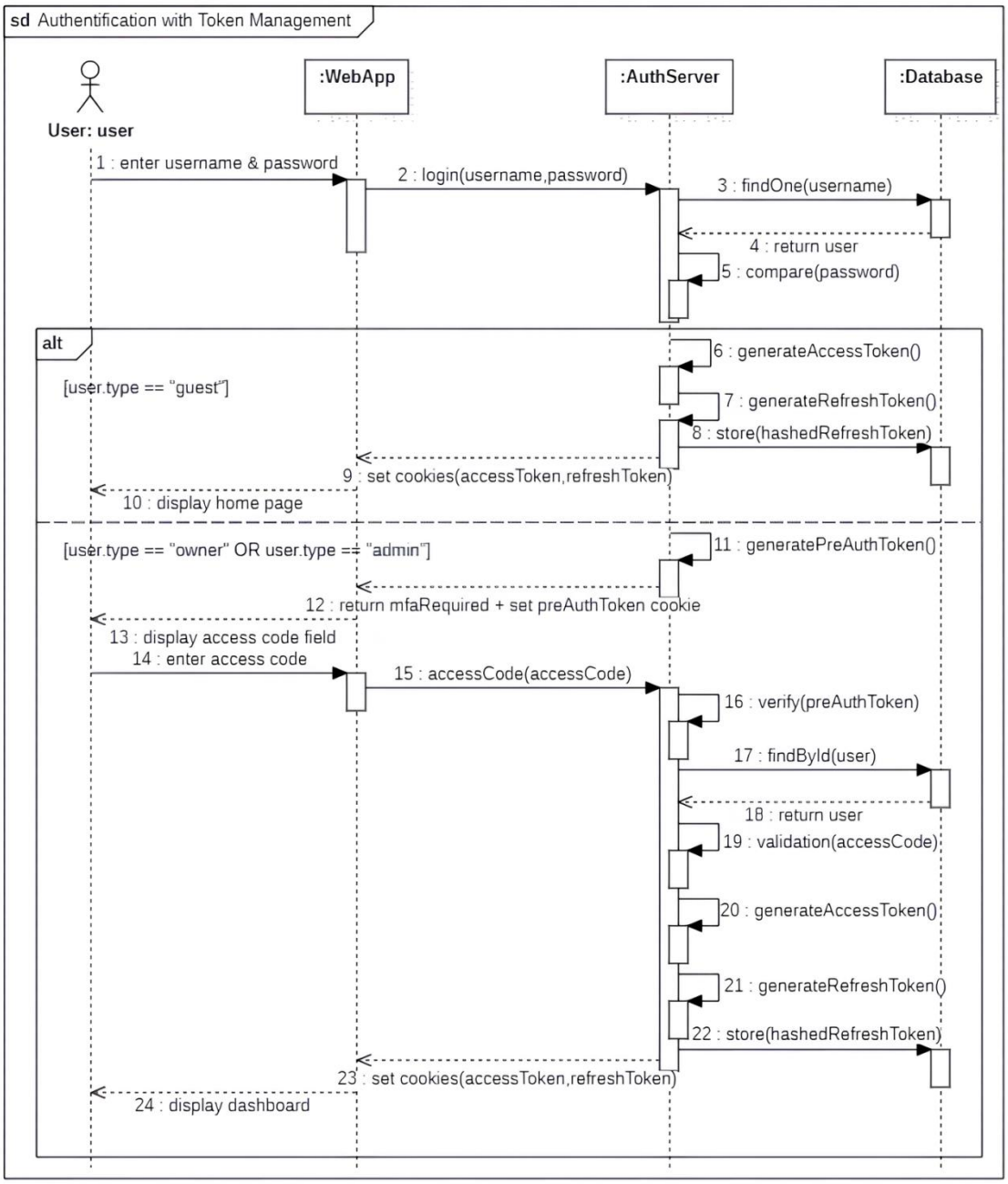


Figure 3.5: *Sequence Diagram of the Authentication and Token Management Scenario*

- **Participating Lifelines:** The interaction involves four participants. The User represents the actor attempting to log in, whether a guest, an owner, or an administrator. The WebApp represents the client-side application through which the user interacts with the system. The AuthServer represents the server-side authentication logic that processes login requests, validates credentials, and issues tokens. The Database represents the persistent data store from which user records are retrieved and in which refresh tokens are stored.

- **Common Authentication Phase:** The scenario begins with the user entering their username and password into the WebApp, which forwards these credentials to the AuthServer through a login request. The AuthServer retrieves the corresponding user record from the Database and compares the submitted password against the stored credential. Once the user's identity and type are established, the process branches according to the user's role, a divergence modelled by an alternative (alt) combined fragment comprising two operands.
- **First Branch — Guest Authentication:** When the authenticated user is a guest, the process completes in a single step. The AuthServer generates an access token and a refresh token, stores the hashed refresh token in the Database, and returns the tokens to the WebApp as secure cookies. The WebApp then grants the guest access by displaying the home page. This direct path reflects the design decision to provide guests with immediate access following successful credential verification, without imposing an additional authentication factor.
- **Second Branch — Owner And Administrator Authentication:** When the authenticated user is an owner or an administrator, an additional verification step is required. Rather than issuing a session immediately, the AuthServer generates a short-lived pre-authentication token (PreAuthToken) and returns it to the WebApp, indicating that a second authentication factor is required. The WebApp responds by displaying the access code field to the user. Upon entering the access code, the WebApp submits it, together with the pre-authentication token, to the AuthServer. The AuthServer verifies the pre-authentication token, retrieves the user record from the Database, and validates the submitted access code. Only once this second factor is confirmed does the AuthServer generate the access and refresh tokens, store the hashed refresh token in the Database, and return the session cookies to the WebApp, which then grants access by displaying the dashboard.
- **Token Management:** This scenario makes explicit the role of the three distinct tokens that underpin the security of the system. The pre-authentication token is a short-lived, single-purpose credential issued exclusively to owners and administrators to authorise the access code step, ensuring that the second factor can only be submitted by a user who has already passed the first. The access token is the credential that authorises a user's subsequent requests during an active session. The refresh token enables the renewal of the access token without requiring the user to log in again; for security, only its hashed form is persisted in the Database, in accordance with the principle of never storing credentials in plain form. Together, these tokens establish a secure, role-sensitive session management mechanism, the implementation details of which are addressed in Chapter 4.

3.3.2. Scenario 2: Price Prediction Workflow

The second scenario models the workflow of the price prediction and recommendation capability, which constitutes one of the two core AI features of the system. This workflow

is best understood in two complementary parts, each represented by a separate sequence diagram. The first part describes how the price predictions are generated and persisted by an automated scheduled job, while the second describes how guests and owners subsequently access these predictions. The deliberate separation between the generation of predictions and their consultation is a central design decision: predictions are computed in advance and stored, so that they can be retrieved instantly when needed, without incurring the cost of real-time inference during user interaction.

a) Price Prediction Scheduled Job

The first part of the workflow, presented in Figure 3.19, models the automated job responsible for generating the price predictions. This interaction is entirely system-driven and involves no human actor.

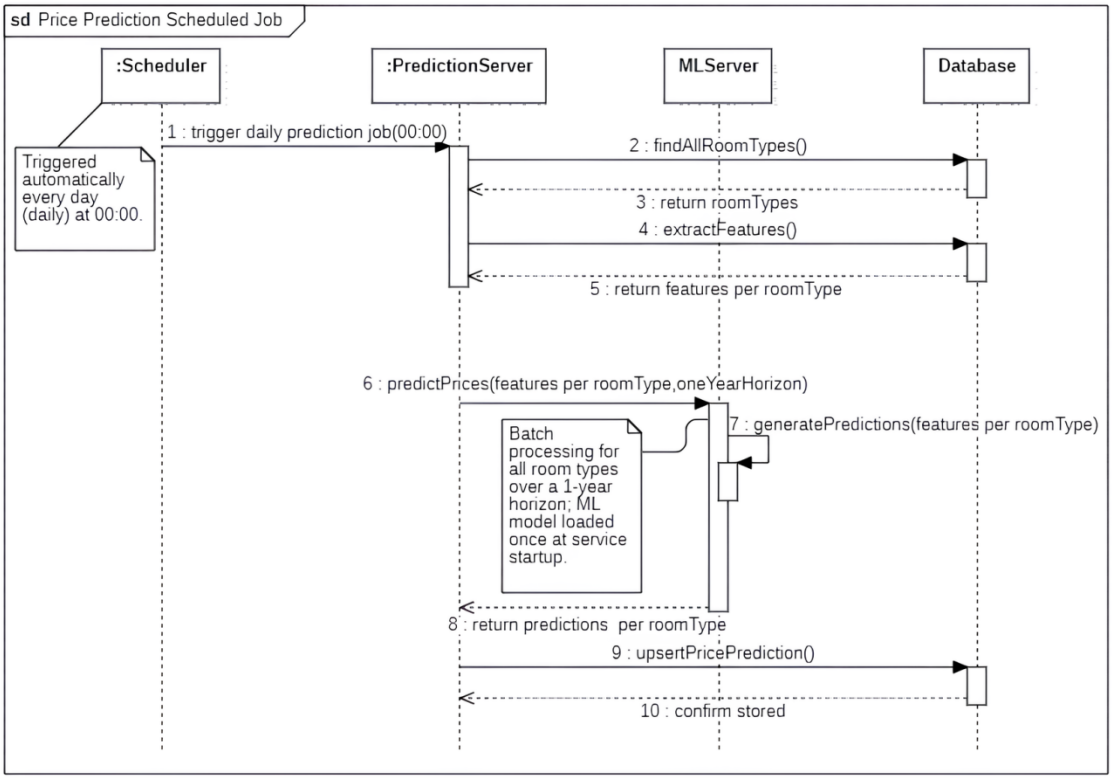


Figure 3.6: *Sequence Diagram of the Price Prediction Scheduled Job*

The interaction involves four participants. The Scheduler is the component that automatically triggers the job every day at midnight. The PredictionServer represents the server-side logic that orchestrates the prediction process. The MLServer represents the dedicated ML service that hosts the price prediction model and produces the predictions. The Database is the persistent data store from which the input data is read and into which the resulting predictions are written.

The job begins when the Scheduler triggers the daily prediction process at midnight. The PredictionServer first retrieves all room types from the Database, then extracts, for each

room type, the set of features required by the prediction model. These features are derived from the reservation records, the historical price records, and the temporal factors associated with each room type, such as the season and the presence of holidays or events. Once the features have been assembled, the PredictionServer submits them to the MLServer in a single batch request covering a one-year horizon. The MLServer generates, for each room type, a series of predictions spanning the coming year, each prediction comprising the predicted price, the expected trend, the percentage change relative to the current price, and the set of reasons that explain the prediction. The predictions are returned to the PredictionServer, which persists them in the Database, overwriting any existing predictions to ensure that the stored data remains current.

Two design characteristics of this job are made explicit in the diagram. First, the predictions for all room types are produced through a single batch request rather than through individual requests per room type, an approach that significantly improves the efficiency of the daily process given the large volume of records involved. Second, the ML model is loaded once when the service starts, rather than on each invocation, avoiding redundant loading overhead. The concrete realisation of these mechanisms is addressed in Chapter 4.

b) Guest and Owner Access to Price Predictions

The second part of the workflow, presented in Figure 3.20, models how the two categories of users consult the predictions that have been generated and stored by the scheduled job. Because the predictions are already persisted in the Database, this interaction does not involve the ML service at all; the predictions are simply retrieved and displayed. The interaction therefore involves four participants: the User (a guest or an owner), the WebApp, the PredictionServer, and the Database. As the two roles access the predictions through different entry points, the interaction is structured around an alternative (alt) combined fragment with two operands.

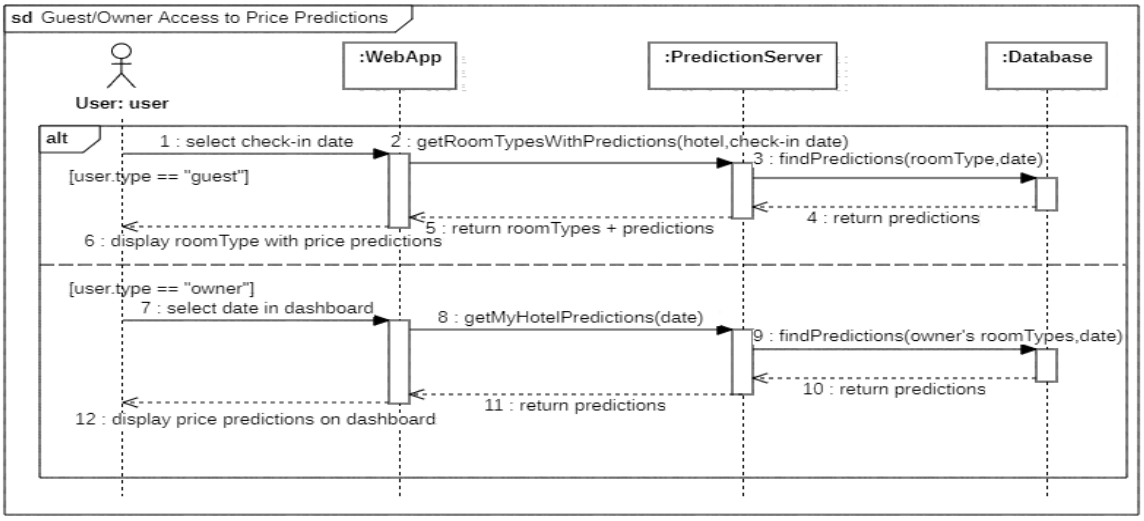


Figure 3.7: *Sequence Diagram of Guest and Owner Access to Price Predictions*

- **First Operand:** corresponding to the guest, the prediction is presented automatically as part of the room browsing experience. When the guest selects a check-in date, the WebApp requests the room types together with their predictions from the PredictionServer for the chosen hotel and date. The PredictionServer retrieves the corresponding predictions from the Database and returns the room types accompanied by their predictions to the WebApp, which displays them to the guest. In this way, the guest is presented with the price prediction for each room type as an integral part of viewing availability, supporting the decision of whether to book immediately or to defer the reservation.
- **Second Operand:** corresponding to the owner, the prediction is consulted deliberately rather than automatically. When the owner selects a date within their dashboard, the WebApp requests the predictions for the owner's hotel from the PredictionServer. The request is scoped to the owner's own property, ensuring that an owner can only access the predictions relating to the hotel they own. The PredictionServer retrieves the relevant predictions from the Database and returns them to the WebApp, which displays them on the dashboard, enabling the owner to decide whether to adjust the prices of their room types.

The distinction between the two operands reflects the differing roles of the prediction for each user: for the guest, it serves as immediate, contextual guidance during booking, whereas for the owner, it serves as a deliberate decision-support tool for pricing management. Together, the two parts of this scenario illustrate the complete lifecycle of a price prediction, from its automated generation to its role-specific consultation.

3.3.3. Scenario 3: Guest Review and Sentiment Analysis

The third scenario models the process by which a guest submits a review of a hotel and the system automatically analyses the sentiment expressed within it. This scenario represents the second of the two core AI capabilities of the system and illustrates how the sentiment analysis is integrated directly into the review submission workflow. Figure 3.21 presents the corresponding sequence diagram.

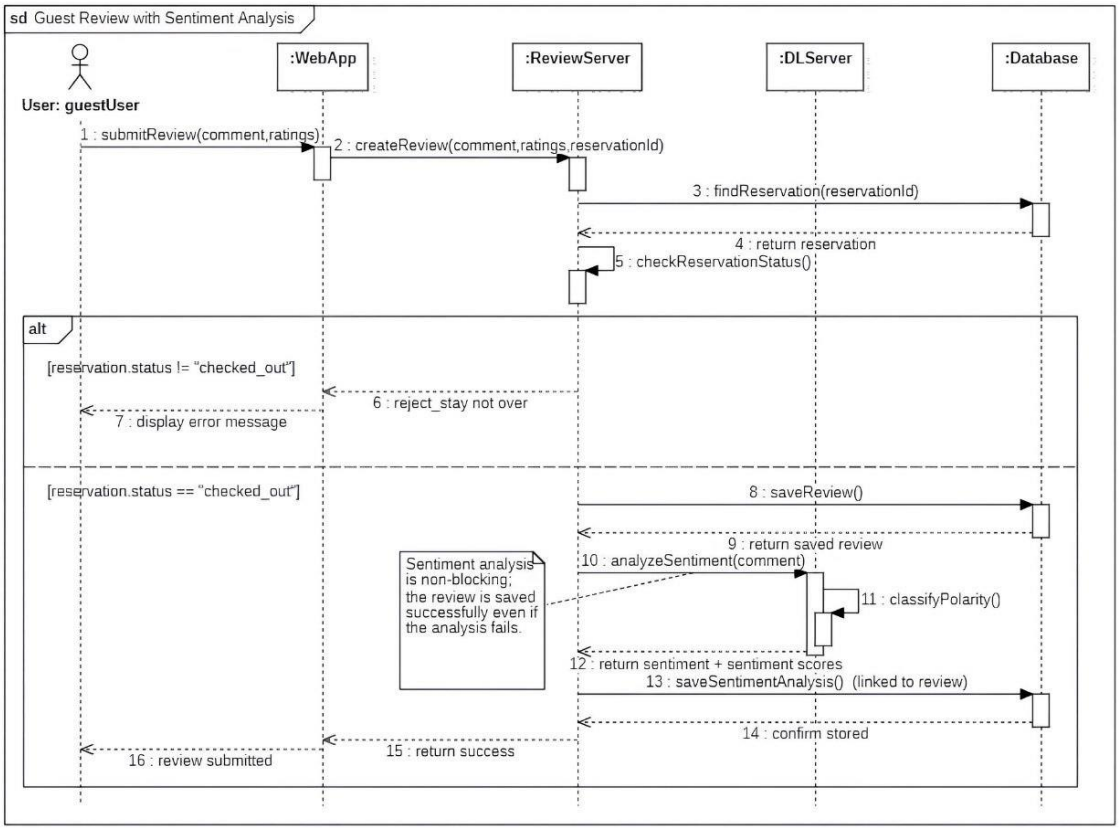


Figure 3.8: *Sequence Diagram of the Guest Review and Sentiment Analysis Scenario*

- **Participating Lifelines:** The interaction involves five participants. The User represents the guest submitting the review. The WebApp represents the client-side application. The ReviewServer represents the server-side logic responsible for processing reviews. The DLServer represents the dedicated DL service that performs the sentiment classification of the review text; it is distinguished from the ML service used for price prediction, reflecting the fact that the sentiment analysis capability is built upon a DL model. Finally, the Database represents the persistent data store in which reservations, reviews, and sentiment results are held.
- **Review Submission and Eligibility Check:** The interaction begins when the guest submits a review, comprising a textual comment and a set of ratings, through the WebApp, which forwards it to the ReviewServer. Before the review can be accepted, the ReviewServer must verify that the guest is eligible to submit it. To this end, it retrieves the associated reservation from the Database and checks its status, since a review may only be submitted once the corresponding stay has been completed. This conditional verification is modelled by an alternative (alt) combined fragment with two operands.
  In the first operand, corresponding to the case in which the reservation has not yet been checked out, the ReviewServer rejects the submission and returns an

error, which the WebApp displays to the guest. No review is created and no sentiment analysis is performed. In the second operand, corresponding to the case in which the reservation has been checked out, the review submission proceeds.

- **Review Persistence and Sentiment Analysis:** Once eligibility is confirmed, the ReviewServer saves the review to the Database. It then forwards the review comment to the DLServer for sentiment analysis. The DLServer classifies the polarity of the comment and returns the result, comprising the sentiment together with the confidence scores for the positive, negative, and neutral classes. The ReviewServer persists this sentiment result in the Database, linking it to the corresponding review, before returning a success response to the WebApp, which confirms to the guest that the review has been submitted.
- **Non-Blocking Sentiment Analysis:** A notable design characteristic of this scenario, highlighted in the diagram, is that the sentiment analysis is performed in a non-blocking manner: the review is saved and the submission succeeds even if the sentiment analysis were to fail. This design decision ensures that the core functionality of submitting a review remains robust and is never compromised by a failure in the auxiliary AI service. The concrete realisation of this behaviour, together with the underlying sentiment analysis model, is detailed in Chapter 4.

Together, the three scenarios presented in this section provide a comprehensive view of the dynamic behaviour of the Hotelify system, covering its security mechanism, its price prediction workflow, and its sentiment analysis process. They complement the static structure described by the class diagram and prepare the ground for the modelling of the reservation workflow presented in the following section.

3.4. Activity Diagram

An activity diagram is a behavioural UML diagram that models the flow of control of a process as a sequence of actions and decisions [51]. Whereas the sequence diagrams of the previous section model the interactions between system components for specific scenarios, the activity diagram provides a higher-level view of a complete business process from beginning to end, making explicit the order of actions, the decision points, and the alternative paths that the process may follow. It is therefore particularly well suited to representing the reservation process, which is the central workflow of the Hotelify system.

Figure 3.22 presents the activity diagram modelling the complete lifecycle of a reservation, from its creation by the guest through to its terminal state. To distinguish the responsibilities of the different participants, the diagram is organised into three swimlanes (partitions): one for the Guest, one for the Owner, and one for the Scheduler, the latter representing the automated scheduled job responsible for the automatic check-out of reservations. This partitioning makes clear, for each action, which participant is responsible for performing it.

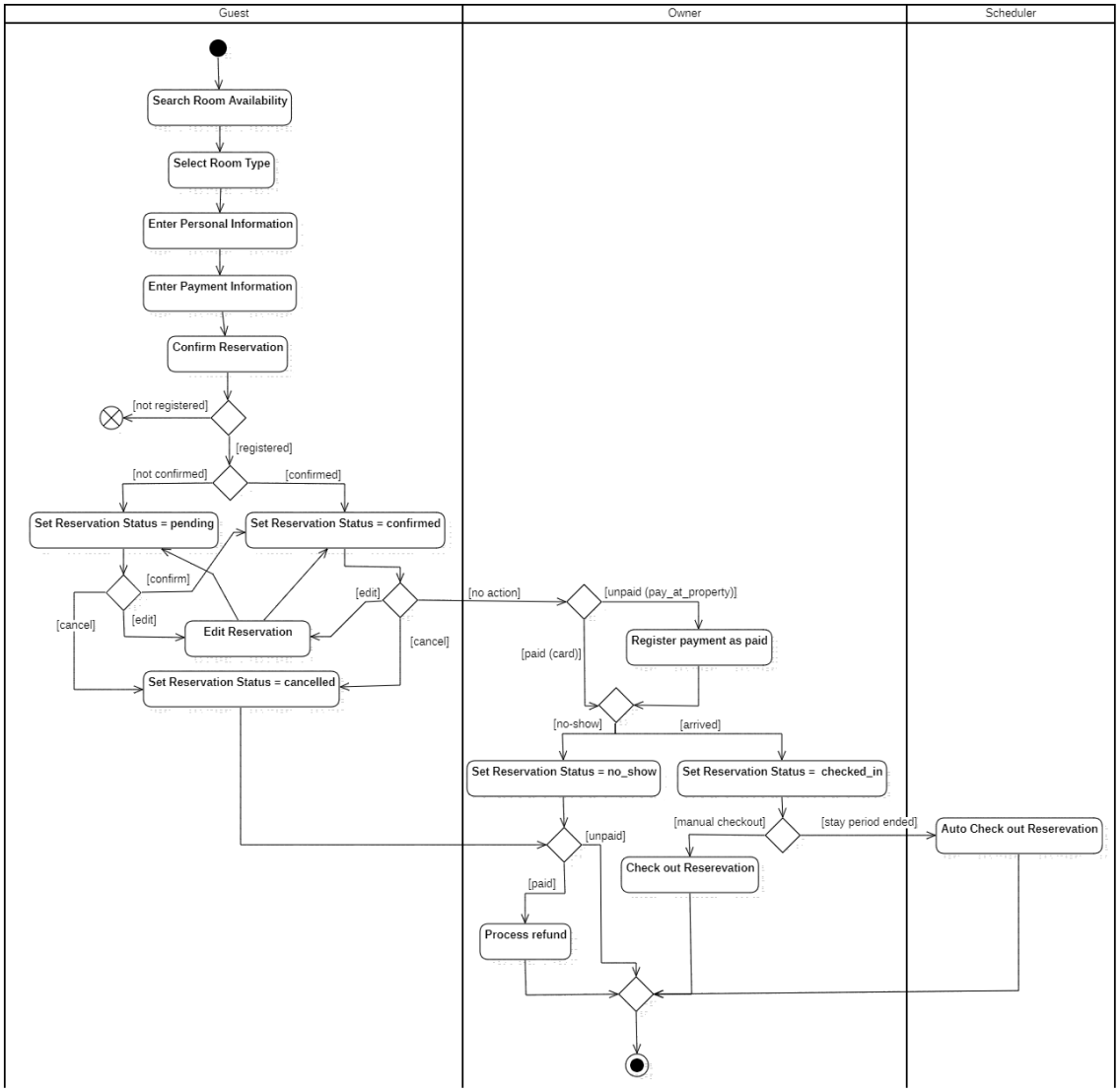


Figure 3.9: *Activity Diagram of the Reservation Process and Management*

3.4.1. The Reservation Creation Process

The process begins in the Guest partition. The guest first searches for the availability of the room types of a chosen hotel and selects an available room type, thereby initiating the booking. The guest then proceeds through the three integral stages of the booking process: entering their personal information, entering their payment information, and confirming the reservation.

At this point, a decision determines whether the reservation has been successfully registered. If the three stages are not completed correctly, the reservation is not registered at all, and the flow terminates at a flow-final node without entering the reservation lifecycle. It is important to note that this outcome does not correspond to a cancellation; it simply means that no reservation was created. If, on the other hand, the reservation is successfully registered, the process continues to determine its initial state.

A second decision then establishes the initial status of the registered reservation. If the guest has not yet confirmed the reservation, its status is set to pending; if the guest has completed and confirmed the three stages, its status is set to confirmed. The payment status is determined in parallel by the payment method chosen during the booking: a reservation paid by card is recorded as paid upon confirmation, whereas a reservation designated for payment at the property remains unpaid until settled in person.

3.4.2. Guest-Side Reservation Management

Once a reservation has been registered, the guest may manage it according to its current status. When the reservation is in the pending state, the guest has three available actions: cancelling the reservation, which sets its status to cancelled; editing the reservation, after which it remains pending; or confirming the reservation, which transitions it to the confirmed state. When the reservation is in the confirmed state, the guest may either cancel it, again setting its status to cancelled, or edit it, after which it remains confirmed. If the guest takes no further action on a confirmed reservation, control passes to the Owner partition for the subsequent stages of the lifecycle. Beyond these states, the guest can perform no direct action on the reservation, as its management becomes the responsibility of the owner.

3.4.3. Owner-Side Reservation Management

When a confirmed reservation reaches the Owner partition, the owner's available actions depend on the payment status. If the reservation was designated for payment at the property and is therefore still unpaid, the owner may register the payment as paid, once the guest has effectively paid at the hotel; if the reservation was already paid by card, this step is not required.

The owner then manages the guest's arrival. If the guest arrives on the check-in date, the owner records the arrival and the reservation status becomes checked-in; if the guest fails to appear, the owner marks the reservation as no-show. For a checked-in reservation, the owner subsequently manages the departure: if the guest departs before or on the check-out date, the owner records the check-out manually, and the reservation status becomes checked-out.

3.4.4. Automatic Check-Out and Refund Handling

The departure of a guest may also be handled automatically. As shown in the Scheduler partition, if a checked-in reservation reaches the end of its stay period without having been checked out manually, the scheduled job automatically sets its status to checked-out. This automated transition complements the manual action of the owner and ensures that the status of every reservation remains accurate without requiring manual intervention.

Finally, the diagram models the handling of refunds. Whenever a reservation reaches the cancelled or no-show state, a decision determines whether a payment had been made. If the reservation had been paid, the owner processes a refund, and the payment status becomes refunded; if no payment had been made, no refund is necessary. All terminal

paths of the process, namely the checked-out, cancelled, and no-show outcomes, converge at a merge node and lead to the final node, marking the end of the reservation lifecycle.

Together, this activity diagram provides a complete and precise model of the reservation process and its management, clearly delineating the responsibilities of the guest, the owner, and the automated scheduler, while distinguishing throughout between the status of the reservation and the status of its payment.

4. Database Design

Building upon the structural model established by the class diagram, this section presents the design of the database that underpins the Hotelify system. As noted in Section 3.2, the classes identified during the structural modelling are translated here into the concrete data structures of the system's data store. Hotelify employs a document-oriented database [53] (MongoDB), in which data is organised into collections of documents rather than into the tables of a relational system [54]. This section first presents the conceptual design of the database through an entity-relationship diagram (ERD) [55], and then details the logical structure and constraints of each collection.

4.1. Conceptual Design

The conceptual design provides a high-level, implementation-independent view of the data, identifying the principal entities of the system and the relationships between them. It is represented by an ERD, presented in Figure 3.23, which was produced using the dbdiagram.io [56] modelling tool.

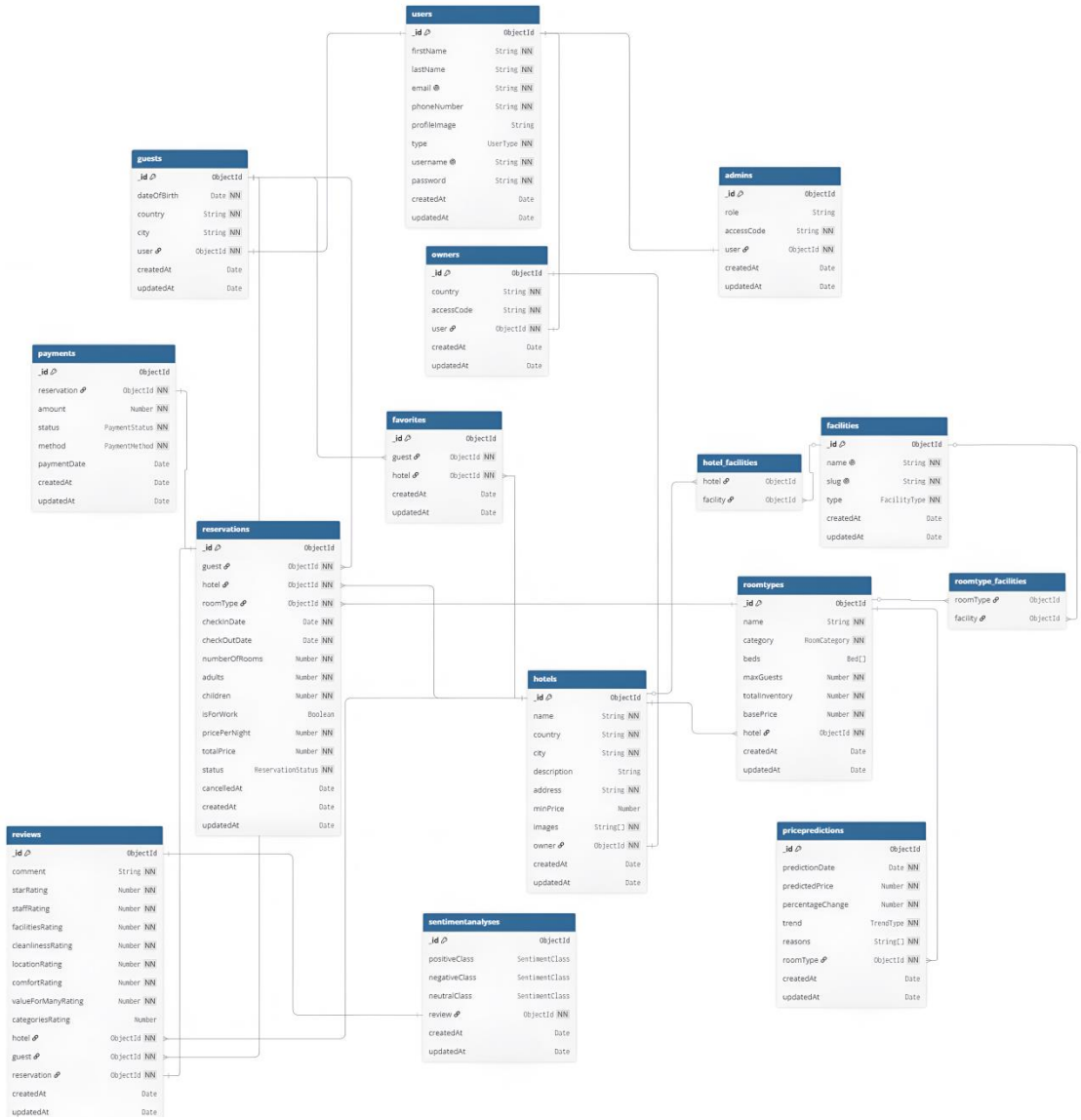


***Figure 3.10**: Entity-Relationship Diagram of the Hotelify Database*

The ERD comprises thirteen primary entities, corresponding to the collections of the database: users, guests, owners, admins, hotels, roomtypes, facilities, reservations, payments, reviews, sentimentanalyses, pricepredictions, and favorites. Whereas the class diagram of Section 3.2 modelled the inheritance of the guest, owner, and administrator roles from a common user class, the database design realises this relationship through separate collections: the guests, owners, and admins collections each reference the users collection, in which the attributes common to all users are stored. This separation reflects the manner in which the role hierarchy is concretely persisted in the document store.

The diagram makes explicit the principal relationships of the data model. Each owner is associated with exactly one hotel, and each hotel is composed of one or more room types. A reservation references the guest who made it, together with the hotel and the room type it concerns. Each reservation is associated with one payment, and a review is linked to the hotel, the guest, and the reservation to which it pertains, as well as to the

sentiment analysis derived from it. Each price prediction references the room type to which it applies, and each favourite links a guest to a hotel. Finally, both hotels and room types maintain a many-to-many relationship with the facilities collection, modelled in the document store as arrays of facility references and represented in the diagram, for clarity, through junction entities. Certain structures, such as the bed configurations of a room type and the three sentiment classes of a sentiment analysis, are not modelled as separate entities but are embedded directly within their parent documents, in accordance with the document-oriented nature of the database.

4.2. Logical Structure and Data Dictionary

This subsection presents the detailed logical structure of each collection, serving simultaneously as the schema specification and as the data dictionary of the database. For each collection, the following tables document the name of every field, its data type, the constraints to which it is subject (such as whether it is required, unique, or bounded by a permitted range or set of values), and a brief description of its purpose. Fields whose type references another collection denote relationships, while fields whose type is an embedded structure denote data nested within the document itself. Timestamp fields (createdAt and updatedAt), which are automatically maintained for every document, are omitted from the tables for the sake of concision.

Table 3.1: *The users Collection*

Field	Type	Constraints	Description
firstName	String	Required, 3–50 chars	The user's first name
lastName	String	Required, 3–50 chars	The user's last name
email	String	Required, unique, 5–100 chars	The user's email address
phoneNumber	String	Required, 8–15 chars	The user's contact number
profileImage	String	Optional	URL of the profile image
type	String	Required, enum {guest, owner, admin}	The role of the user
username	String	Required, unique, 3–100 chars	The user's unique username
password	String	Required, min 8 chars	The hashed authentication credential

Table 3.2: *The guests Collection*

Field	Type	Constraints	Description
dateOfBirth	Date	Required	The guest's date of birth
country	String	Required, 2–30 chars	The guest's country of residence
city	String	Required, 2–50 chars	The guest's city of residence
user	Reference → users	Required	The associated user account

Table 3.3: *The owners Collection*

Field	Type	Constraints	Description
country	String	Required, 2–30 chars	The owner's country
accessCode	String	Required	The access code used for privileged login
user	Reference → users	Required	The associated user account

Table 3.4: *The admins Collection*

Field	Type	Constraints	Description
role	String	Default "super_admin"	The administrative role
accessCode	String	Required	The access code used for privileged login
user	Reference → users	Required	The associated user account

Table 3.5: *The hotels Collection*

Field	Type	Constraints	Description
name	String	Required, 3–100 chars	The name of the hotel
country	String	Required, 2–30 chars	The country of the hotel
city	String	Required, 2–50 chars	The city of the hotel
description	String	Optional	A textual description of the hotel
address	String	Required, 2–100 chars	The address of the hotel
minPrice	Number	Optional	The lowest room price offered
images	Array of String	Required, ≥ 1 image	The image URLs of the hotel
facilities	Array of Reference → facilities	Optional	The facilities offered by the hotel
owner	Reference → owners	Required	The owner of the hotel

Table 3.6: *The roomtypes Collection*

Field	Type	Constraints	Description
name	String	Required, 3–50 chars	The name of the room type
category	String	Required, enum (8 categories)	The classification of the room type
beds	Array of embedded Bed	Each: bedType (enum), count (1–5)	The bed configuration of the room type
maxGuests	Number	Required, 1–10	The maximum occupancy
totalInventory	Number	Required, ≥ 3	The total number of rooms of this type
basePrice	Number	Required, ≥ 10	The base price per night
facilities	Array of Reference → facilities	Optional	The amenities of the room type
hotel	Reference → hotels	Required	The hotel to which the room type belongs

Table 3.7: *The facilities Collection*

Field	Type	Constraints	Description
name	String	Required, unique, 3–50 chars	The name of the facility
slug	String	Required, unique, lowercase	A URL-friendly identifier
type	String	Required, enum {hotel, roomType}	The scope of the facility

Table 3.8: *The reservations Collection*

Field	Type	Constraints	Description
guest	Reference → guests	Required	The guest who made the reservation
hotel	Reference → hotels	Required	The reserved hotel
roomType	Reference → roomtypes	Required	The reserved room type
checkInDate	Date	Required	The arrival date
checkOutDate	Date	Required	The departure date
numberOfRooms	Number	Required, [1,30]	The number of rooms reserved
adults	Number	Required, [1,30]	The number of adults
children	Number	Required, [0,10]	The number of children
isForWork	Boolean	Default false	Whether the stay is for business
pricePerNight	Number	Required, ≥ 0	The price per night
totalPrice	Number	Required, ≥ 0	The total price of the stay
status	String	Required, enum (6 states)	The reservation status

cancelledAt	Date	Optional	The cancellation timestamp, if any

Table 3.9: *The payments Collection*

Field	Type	Constraints	Description
reservation	Reference → reservations	Required	The associated reservation
amount	Number	Required, ≥ 0	The payment amount
status	String	Required, enum {unpaid, paid, refunded}	The payment status
method	String	Required, enum {card, pay_at_property}	The payment method
paymentDate	Date	Optional	The date the payment was settled

Table 3.10: *The reviews Collection*

Field	Type	Constraints	Description
comment	String	Required, 3–1000 chars	The textual review
starRating	Number	Required, [0,5]	The overall star rating
staffRating	Number	Required, [0,10]	The staff rating
facilitiesRating	Number	Required, [0,10]	The facilities rating
cleanlinessRating	Number	Required, [0,10]	The cleanliness rating
locationRating	Number	Required, [0,10]	The location rating
comfortRating	Number	Required, [0,10]	The comfort rating
valueForManyRating	Number	Required, [0,10]	The value-for-money rating
categoriesRating	Number	[0,10], computed average	The averaged category rating
hotel	Reference → hotels	Required	The reviewed hotel
guest	Reference → guests	Required	The reviewing guest
reservation	Reference → reservations	Required	The associated reservation

Table 3.11*: The sentimentanalysis Collection*

Field	Type	Constraints	Description
positiveClass	Embedded {label, score}	score [0,1]	The positive sentiment class and its confidence
negativeClass	Embedded {label, score}	score [0,1]	The negative sentiment class and its confidence
neutralClass	Embedded {label, score}	score [0,1]	The neutral sentiment class and its confidence

review	Reference → reviews	Required	The analysed review

Table 3.12: *the pricePredicyions Collection*

Field	Type	Constraints	Description
predictionDate	Date	Required	The date the prediction applies to
predictedPrice	Number	Required, ≥ 10	The predicted price
percentageChange	Number	Required, [−100,100]	The change relative to the current price
trend	String	Required, enum {increase, decrease, stable}	The predicted price trend
reasons	Array of String	Required, ≥ 1 reason	The explanations supporting the prediction
roomType	Reference → roomtypes	Required	The room type the prediction applies to

Table 3.13: *the favorites Collection*

Field	Type	Constraints	Description
guest	Reference → guests	Required	The guest who saved the favourite
hotel	Reference → hotels	Required	The favourited hotel

Together, these collections constitute the complete data model of the Hotelify system, providing a structured, consistent, and constraint-enforced foundation for the storage and retrieval of all the data manipulated by the application. With the data design now established, the following section turns to the overall architecture of the system.

5. System Architecture Design

Having defined the behaviour of the system through its use cases and sequence diagrams, its structure through the class diagram, and its data through the database design, the remaining design concern is the organisation of the software into a set of cooperating components and the manner in which these components communicate. This section presents the architecture of the Hotelify platform from a high-level, technology-independent standpoint. It first describes the overall layered organisation [57] of the system, an architectural style in which the software is decomposed into a stack of layers, each depending only on the layer immediately beneath it, and then examines how the AI service is integrated within that organisation. Both subsections refer to the same architectural model, created using a tool called Mermaid [58], illustrated in Figure 3.9, which captures the tiers of the system, their constituent components, and the communication paths between them.

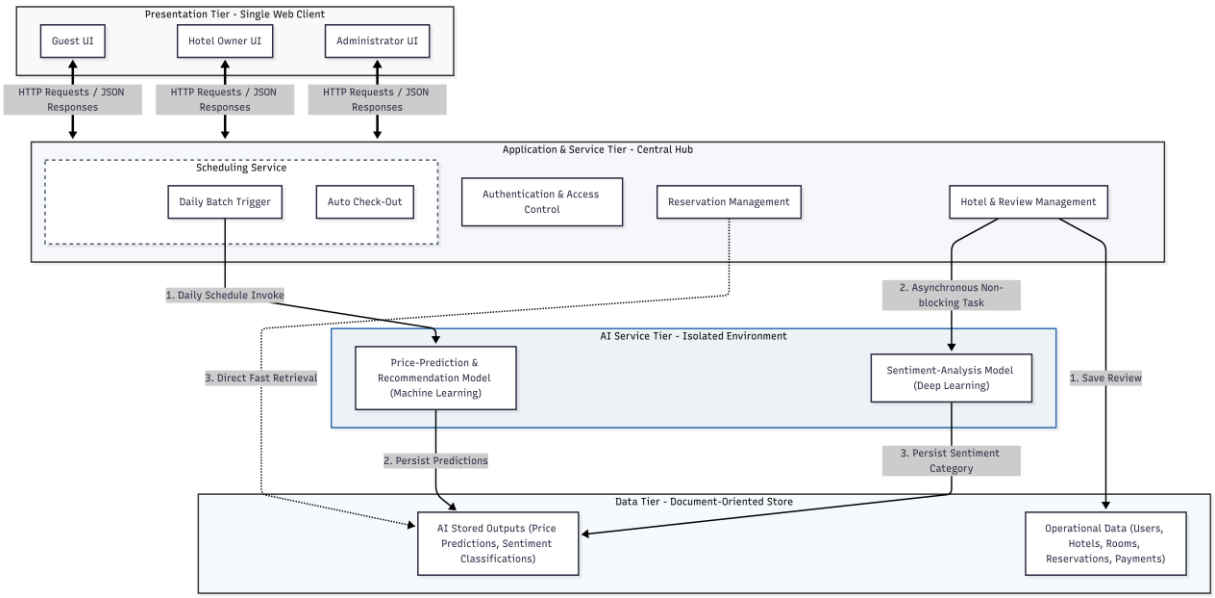


Figure 3.11: *The Layered Architecture of the Hotelify Platform Overall Architecture*

5.1. Overall Architecture

Hotelify adopts a layered, multi-tier architectural style, in which the system is decomposed into a set of horizontal tiers, each assigned a distinct and cohesive responsibility, and each depending only on the services exposed by the tier beneath it. This separation of concerns is a well-established means of improving modularity, testability, and maintainability, since it allows each tier to evolve independently as long as the contracts between tiers are preserved. As shown in Figure 3.23, the architecture comprises four tiers: the presentation tier, the application and service tier, the AI service tier, and the data tier.

❖ The Presentation Tier

constitutes the point of contact between the system and its users. It provides a single web-based client interface through which the three categories of user, namely the guest, the hotel owner, and the administrator, interact with the platform according to their respective privileges. This tier is responsible solely for rendering information and capturing user actions; it holds no business logic of its own, and instead delegates every operation to the tier below it through well-defined requests, receiving structured responses in return. Concentrating all presentation responsibilities within a single tier ensures a consistent user experience and keeps the interface independent of the way the underlying services are realised.

❖ The Application and Service Tier

This tier is located at the center of the architecture, which embodies the business logic of the system and orchestrates the cooperation of all other tiers. This tier

exposes the operations that fulfil the functional requirements established earlier in the chapter, grouped into cohesive functional units. The principal units include authentication and access control, which governs identity verification and the enforcement of role-based privileges; reservation management, which handles the booking life cycle and its associated state transitions; and hotel and review management, which administers hotel content and the submission of guest reviews. In addition, this tier hosts a scheduling service responsible for the recurring, time-driven operations of the system, namely the periodic automatic check-out of expired reservations and the daily generation of price predictions and price history. Acting as the coordinating hub of the system, the application tier mediates every interaction between the presentation tier above it and the data and AI tiers beneath it, so that neither the user interface nor the data store ever communicates with the models directly.

❖ The AI Service Tier

encapsulates the two predictive capabilities that distinguish the platform. It is deliberately designed as an independent service rather than as a component embedded within the application tier, so that the computational requirements of the models remain isolated from the core business logic, and so that the models can be invoked and maintained without affecting the rest of the system. This tier houses the price prediction and recommendation model, which is based on ML, and the sentiment analysis model, which is based on DL. For each room type, the first model produces a predicted price together with a trend, a percentage change, and the reasons underlying the estimate. These outputs constitute the recommendation in their own right, since it is from them that a guest or an owner derives a pricing decision, rather than from any separate recommendation mechanism. The second model classifies the sentiment expressed in a guest review into one of three categories. The internal design of both models is detailed in Section 6; the present section is concerned only with their position within the architecture.

❖ The Data Tier

provides durable storage for the entire system. It is organised as a document-oriented store, in which data is held in collections of documents rather than in relational tables, a model that aligns naturally with the nested and heterogeneous structure of the platform's data, as established in the database design of Section 4. Beyond the operational data of users, hotels, room types, reservations, payments, reviews, and favourites, this tier also persists the outputs of the AI tier, namely the price predictions generated for each room type and the sentiment classifications derived from guest reviews. Storing these results alongside the operational data is a deliberate architectural choice whose rationale is examined in the following subsection.

The four tiers communicate through clearly defined request and response exchanges. The presentation tier issues requests to the application tier and renders the responses it receives; the application tier, in turn, retrieves and persists data in the data tier and invokes the AI service tier whenever intelligent processing is required. By routing all communication through the application tier, the architecture maintains a single locus of control in which security, validation, and coordination are uniformly enforced.

5.2. AI Service Integration within the Architecture

While the previous subsection situated the AI service within the overall structure, this subsection examines more closely how that service is integrated into the operation of the system. The integration of the two models is governed by a single principle: intelligent processing must enrich the user experience without ever degrading the responsiveness of the platform. To honour this principle, the two models are invoked through two distinct interaction patterns, each suited to the nature of the task it serves. Both patterns are depicted in Figure 3.9.

❖ The Price Prediction and Recommendation Model

This model is integrated through a scheduled, batch-oriented pattern. Rather than being computed at the moment a user requests it, the prediction for every room type is generated in advance by the scheduling service of the application tier, which invokes the model once per day as a single batch operation. The resulting predictions, each comprising the predicted price together with its trend, percentage change, and explanatory reasons, are then persisted in the data tier. Consequently, when a guest selects a check-in date, or when an owner consults a date within the management dashboard, the corresponding recommendation is retrieved directly from the data tier as pre-computed information, and no invocation of the model occurs along this path. This design decouples the cost of inference from the user's interaction. It guarantees a near-instantaneous response at the point of consultation, it avoids redundant computation by reusing a single daily prediction across all users, and it ensures that the recommendation presented to every user remains consistent.

❖ The Sentiment Analysis Model

This model is integrated through an event-driven, asynchronous pattern. Its execution is triggered by a specific event, the submission of a guest review, rather than by a schedule. When a review is submitted, the application tier accepts and stores it immediately and forwards its text to the AI service tier for classification as a separate, non-blocking task. Because the analysis is performed asynchronously, the submission is acknowledged without waiting for the model to complete, and the resulting sentiment classification is persisted and associated with the review once it becomes available. This arrangement ensures that the analytical enrichment of reviews never delays the interaction, thereby preserving

the responsiveness of the interface even though an additional computational step is involved.

Taken together, these two patterns reflect a consistent integration philosophy. The application tier remains the sole mediator of the AI service: it decides when each model is invoked, supplies the required input, and persists the returned results, while the presentation and data tiers never communicate with the models directly. The systematic storage of model outputs in the data tier, whether produced on a schedule or in response to an event, allows the results of intelligent processing to be served to users with the same speed and reliability as ordinary operational data. In this way, the AI capabilities are woven into the architecture as a first-class concern, yet without compromising the modularity, responsiveness, or robustness of the platform as a whole.

6. AI Models Design (Pipeline)

The intelligence of the Hotelify platform rests on the two predictive models whose architectural placement was described in the previous section. The present section turns from where these models reside to how they are designed. For each model, it sets out the complete design pipeline, that is, the ordered sequence of stages through which raw data is transformed into a trained, validated, and deployable model. In keeping with the scope of this chapter, the discussion remains at the level of methodology: it identifies the algorithms, techniques, and strategies adopted at each stage and the reasoning behind them, whereas the concrete implementation, the experimental work, and the quantitative results are deferred to Chapter 4. The section begins by describing the datasets on which the two models were designed, and then presents, in turn, the pipeline of the price prediction and recommendation model and that of the sentiment analysis model.

6.1. Dataset Description

The design of any data-driven model begins with the data from which it learns. Two distinct, publicly available datasets, both obtained from the Kaggle platform, were used to design the two models of the system. Each dataset was selected because its content corresponds closely to the predictive task it serves. This subsection describes each dataset only as an input to the design process; consistent with the boundary of this chapter, it is not revisited in Chapter 4.

6.1.1. Hotel Booking Demand Dataset

The price prediction and recommendation model was designed using the Hotel Booking Demand dataset [59], obtained from the Kaggle platform. This dataset records a large collection of real hotel booking transactions, each described by a rich set of attributes that capture the circumstances of the booking. Among these attributes are the hotel name,the room type reserved,the lead time between reservation and arrival, the components of the arrival date, the length of stay over weekday and weekend nights,and

the average daily rate (ADR) associated with the reservation. The temporal nature of these records, together with the presence of a rate attribute, makes the dataset well suited to the design of a model that learns how room prices vary with demand-related conditions over time.

To convey the structure of the data underlying the model, Table 3.14 presents a small sample set of records restricted to the principal features that proved relevant throughout the design pipeline.

Table 3.14: *A Sample of the Hotel Booking Demand Dataset*

#	hotel	Is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	reserved_room_type	adr
1	Resort Hotel	0	342	2015	July	27	1	0	0	C	0
2	Resort Hotel	0	737	2015	July	27	1	0	0	C	0
3	Resort Hotel	0	7	2015	July	27	1	0	1	A	75
4	Resort Hotel	0	13	2015	July	27	1	0	1	A	75
5	Resort Hotel	0	14	2015	July	27	1	0	2	A	98
6	City Hotel	0	13	2015	August	35	27	0	1	F	153
7	City Hotel	0	0	2015	August	35	27	0	1	A	122
8	City Hotel	1	1	2015	August	35	27	0	1	E	160
9	City Hotel	0	8	2015	August	35	27	0	1	A	90
10	City Hotel	1	15	2015	August	35	27	0	2	A	80

6.1.2. Hotel Reviews: Aspects, Sentiments and Topics (HRAST) Dataset

The sentiment analysis model was designed using the Hotel Reviews: Aspects, Sentiments and Topics (HRAST) dataset [60], likewise obtained from the Kaggle platform. This dataset consists of textual guest reviews of hotels, each accompanied by indicators of the sentiment it expresses. For the purposes of the present model, the relevant content of each record is the review text together with its sentiment, the latter being conveyed through three indicator attributes denoting positive, negative, and neutral sentiment. The dataset therefore provides the pairing of free-text reviews with sentiment labels that the supervised design of a sentiment classifier requires

As with the previous dataset, Table 3.15 presents a small sample set of records limited to the features that were retained for the design of the sentiment analysis model.

Table 3.15: *A Sample of the HRAST Dataset*

#	review	positive	negative	neutral
1	1 elevator not working.	0	1	0
2	1 euros to park per day on their own premises.	0	0	1
3	1 hour queue to check-in.	0	1	0
4	1 min walk to metro entrance and also few min walk to the central station where you can access rail trains.	1	0	0
5	1 minute walk from the main metro & train station, which made getting around Copenhagen extremely easy.	1	0	0
6	A Broken phone in the room.	0	1	0
7	A cleaning kit would be a real comfort.	0	1	0
8	A closed kitchen for dinner one of the nights (at 7pm) however, the bar staff saved this by cooking us pizzas - so 10 out of 10 for a great rescue.	1	0	0
9	A coffee maker and tea seems the minimum in a room of this standard.	0	1	0
10	A fan is available but very noisy and not in good condition.	0	1	0

6.2. Price Prediction and Recommendation Model for Dynamic Pricing

The price prediction and recommendation model is the component responsible for estimating, for each room type and for a given date, the price that best reflects prevailing demand conditions, and for accompanying that estimate with the contextual information from which a pricing decision can be drawn. As established earlier in the chapter, the prediction itself constitutes the recommendation, since the predicted price, together with its trend, its percentage change, and the reasons behind it, is precisely what guides the guest or the owner, rather than the output of any separate recommendation mechanism. Because it learns a mapping from structured, tabular features to a continuous price value, this model belongs to the family of ML methods. Its design follows a structured pipeline whose stages are summarised in Figure 3.25 and detailed in the remainder of this subsection.

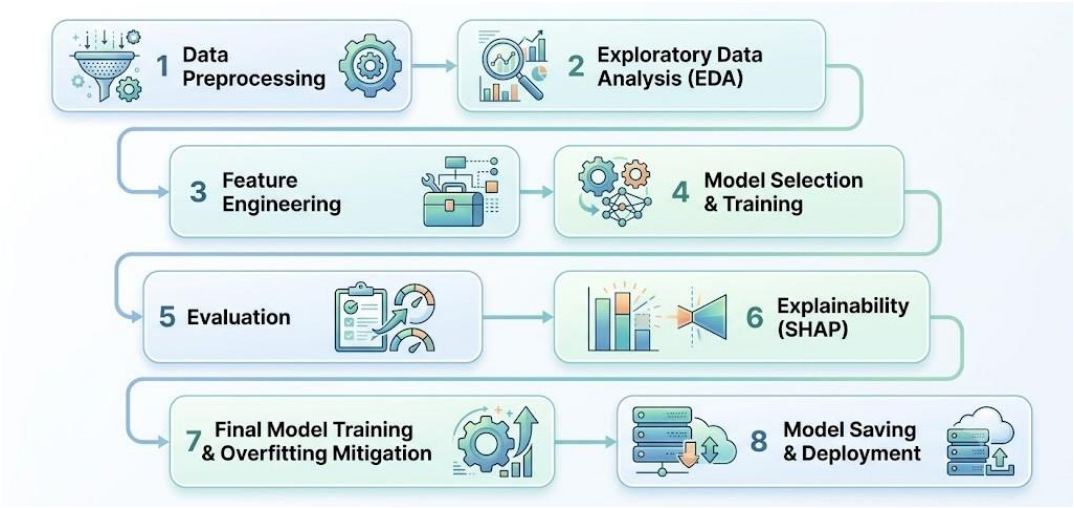


Figure 3.12: *The Design Pipeline of the Price Prediction and Recommendation Model*

6.2.1. Data Preprocessing

The pipeline begins by preparing the raw booking records for analysis. This stage encompasses the handling of missing and inconsistent values, the removal of duplicate and invalid records, the encoding of categorical attributes into a numerical form suitable for learning, and the derivation of a per-night price target from the recorded rate and stay information. Its outcome is a clean, consistent dataset that faithfully represents the booking conditions while being amenable to the subsequent stages.

6.2.2. Exploratory Data Analysis (EDA)

Before modelling, the prepared data is examined in order to understand its characteristics. This analysis studies the distribution of the price target, the relationships between the candidate features and that target, the seasonal and temporal patterns present in the bookings, and the presence of outliers. The purpose of this stage is methodological: it informs the feature engineering and modelling decisions that follow rather than producing findings in its own right.

6.2.3. Feature Engineering

Building on the insights of the exploratory analysis, this stage constructs the set of features from which the model learns. Beyond the attributes already present in the raw data, a number of derived features are engineered to express demand dynamics more explicitly. These include temporal aggregations computed over rolling time windows, indicators of booking velocity and of the time remaining until a room type becomes fully booked, and a tiering of room prices intended to improve the generalisability of the model across hotels of differing price levels. The selection of the final feature set is itself iterative, with features retained or discarded according to their contribution to predictive quality.

6.2.4. Model Selection and Training

With the feature set established, several candidate regression algorithms are considered, including a Random Forest Regressor [61] and Gradient Boosting algorithms such as XGBoost and LightGBM. Because the data is inherently temporal, model selection relies on a time-series cross-validation scheme [62] that respects the chronological order of the bookings and thereby prevents any leakage of future information into the past. The hyperparameters of each candidate are tuned through a randomized search [63] over the parameter space. This stage identifies the algorithm and configuration best suited to the prediction task.

6.2.5. Evaluation

The candidate models are assessed according to a deliberately composite strategy rather than a single metric. The design adopts complementary error measures, namely the mean absolute percentage error (MAPE) [64], the root mean square error (RMSE) [65], and the mean absolute error (MAE) [65], each capturing a different facet of predictive accuracy. These three measures are formally defined below, immediately following the description of this stage. To these is added a measure of the gap between training and test performance, which serves as an explicit indicator of overfitting. Combining these measures into a single weighted score allows the selection of a model that is not only accurate but also stable and generalisable. The concrete results of this assessment are reported in Chapter 4.

The three error measures adopted for the evaluation of the price prediction model are defined as follows.

- **Mean Absolute Percentage Error (MAPE):** expresses the average absolute deviation between the predicted and the actual price as a percentage of the actual price:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

  where n is the number of observations in the evaluation set, y_i is the actual price of observation i, and $\hat{y}_i$ is the price predicted by the model for that same observation.

- **Root Mean Square Error (RMSE):** measures the square root of the average squared deviation between predicted and actual prices, penalising larger errors more heavily than smaller ones:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  where n, y_i, and $\hat{y}_i$ retain the meaning defined above.

- **Mean Absolute Error (MAE):** measures the average magnitude of the errors between predicted and actual prices, without regard to their direction:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where n, y_i, and $\hat{y}_i$ retain the meaning defined above.

6.2.6. Explainability (SHAP)

So that the model is not treated as an opaque predictor, its behaviour is interpreted using SHapley Additive exPlanations (SHAP) [66], a game-theoretic technique that attributes to each feature a contribution toward a given prediction. This analysis reveals which demand factors most strongly influence the predicted price and underpins the reasons that accompany each recommendation, thereby linking the internal logic of the model to the explanatory output presented to users.

6.2.7. Final Model Training and Overfitting Mitigation

Once the evaluation has identified the most suitable algorithm, its configuration, and the most informative features, a final, optimised model is trained for deployment. Whereas the earlier training served to compare and select, this stage aims to extract the best possible generalisation from the chosen configuration. To this end, the model is retrained on the full extent of the available data, and overfitting is actively countered through an early-stopping mechanism [67] that halts training once validation performance ceases to improve, complemented by the regularisation [68] inherent to the selected algorithm. The result is an optimised model that preserves the accuracy observed during evaluation while remaining robust on unseen data.

6.2.8. Model Saving and Deployment

Finally, the optimised model is serialised and prepared for deployment within the AI service described in Section 5. Its design anticipates the scheduled, batch mode of invocation through which daily predictions are generated and stored, so that the trained model can be integrated into the running system without further modification.

6.3. Sentiment Analysis Model Pipeline

The sentiment analysis model is the component responsible for determining the opinion expressed in a guest's review and assigning it to one of three sentiment categories, namely positive, negative, or neutral. By converting unstructured review text into a structured sentiment label, the model enables the platform to summarise guest opinion at scale. Because it is built upon a neural language model that learns rich contextual representations of text, this model belongs to the family of deep-learning (DL) methods. Its design follows the pipeline summarised in Figure 3.26 and detailed below.

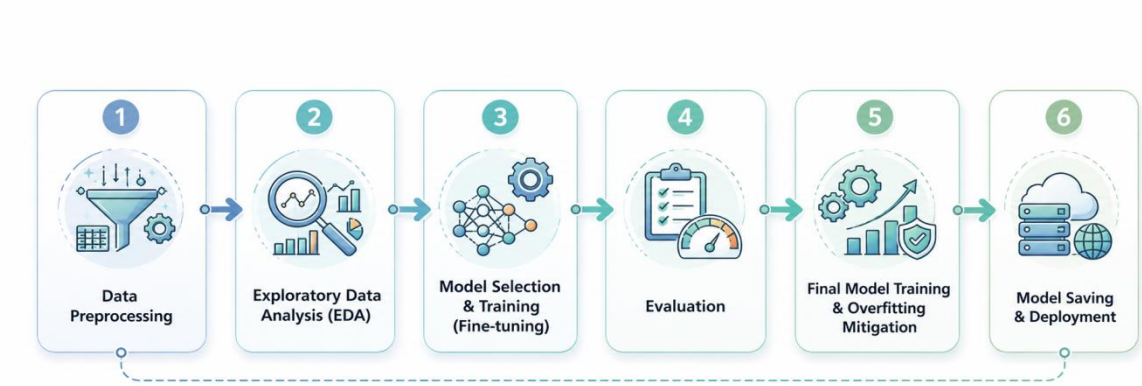


Figure 3.13: *The Design Pipeline of the Sentiment Analysis Model*

6.3.1. Data Preprocessing

The pipeline begins by transforming the raw review dataset into a form suitable for supervised learning. The records are first inspected for missing values, duplicates, and class composition. Only the attributes relevant to the task are retained, namely the review identifier and its text, which are given clear and consistent names. The review text is then cleaned by normalising it to lowercase and removing surrounding whitespace, while excessively short reviews, defined as those containing fewer than two words or shorter than eight characters, are discarded as uninformative. Finally, the three separate sentiment indicators are consolidated into a single categorical sentiment label, producing a compact dataset in which each cleaned review is paired with exactly one sentiment class.

6.3.2. Exploratory Data Analysis (EDA)

The cleaned dataset is then examined in order to understand the nature of the classification problem. The central observation that guides the design is the distribution of reviews across the three sentiment classes, which is markedly imbalanced, the neutral class being substantially less represented than the positive and negative classes. As before, this analysis is methodological in purpose, shaping the choices made in the stages that follow rather than reporting findings.

6.3.3. Model Selection and Training

The classifier is built upon DistilBERT [69], a compact, pre-trained, transformer-based language model chosen for offering a strong balance between linguistic representational power and computational efficiency. The review text is converted into the input form expected by the model through subword tokenization, with a maximum sequence length fixed at one hundred and twenty-eight tokens, a value established by measuring the distribution of review lengths so as to retain the informative content of the reviews while limiting unnecessary computation. The pre-trained model is then fine-tuned on the hotel-review data, so that its general understanding of language is specialised to the sentiment task. To address the class imbalance identified during the exploratory analysis, the

training strategy adopts a class-weighted, focal-loss [70] objective that increases the influence of the under-represented neutral class and concentrates learning on the harder examples.

6.3.4. Evaluation

The quality of the model is assessed using metrics appropriate to an imbalanced, multi-class problem. Beyond overall accuracy [71], the design emphasises the per-class precision [71], recall [71], and F1-score [71], together with their macro-average [72], so that strong performance on the majority classes cannot mask weak performance on the minority class. These metrics are formally defined below, immediately following the description of this stage. This evaluation strategy ensures that the model is judged on its ability to recognise all three sentiments fairly. The numerical results obtained are presented in Chapter 4.

The classification metrics adopted for the evaluation of the sentiment analysis model are defined as follows.

- **Accuracy:** measures the overall proportion of reviews for which the predicted sentiment label matches the true label, across all three classes:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} 1(y_i = \hat{y}_i)$$

where n is the total number of reviews in the evaluation set, y_i is the true sentiment label of review i, $\hat{y}_i$ is the label predicted by the model, and $1(\cdot)$ is an indicator function equal to 1 when its argument is true and 0 otherwise.

For a given sentiment class, precision, recall, and F1-score are expressed in terms of the true positives (TP), false positives (FP), and false negatives (FN) obtained when that class is treated as the positive class and the remaining two classes as negative.

- **Precision:** measures the proportion of reviews predicted as belonging to a given class that are indeed correctly classified:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** measures the proportion of reviews actually belonging to a given class that are correctly identified by the model:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-score:** combines precision and recall into a single measure through their harmonic mean, so that a model is penalised for weakness in either dimension:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **Macro-average:** extends a per-class metric to a single, class-balanced summary value by computing the unweighted arithmetic mean of that metric across all classes:

$$Macro - average(M) = \frac{1}{C}\sum_{c=1}^{C} M_c$$

where C is the number of sentiment classes (three, in the present system) and M is the value of the metric M (precision, recall, or F1-score) computed for class c.

6.3.5. Final Model Training and Overfitting Mitigation

Mitigation Following the same two-stage logic adopted for the pricing model, the configuration validated during evaluation is used to train a final, optimised model for deployment. Whereas the first training phase set aside part of the data to estimate performance, the final phase trains the model on the complete dataset so as to exploit all available examples. Overfitting is guarded against by limiting the number of training epochs, by monitoring validation performance to stop training at its optimal point, and by the regularisation effects already present in the architecture. This produces an optimised classifier that generalises reliably to new, unseen reviews.

6.3.6. Model Saving and Deployment

The optimised model is then saved and made ready for integration into the AI service. Consistent with the event-driven, non-blocking mode of invocation described in Section 5, the model is prepared to classify each newly submitted review as an asynchronous task, so that its design aligns directly with its role in the running system.

6.4. Unified Comparison of Both Pipelines

The two pipelines presented in the previous subsections were designed independently, yet they share a common methodological backbone while diverging wherever the nature of the data or of the learning task requires it. This subsection draws them together in a structured comparison of their design choices, set out in Table 3.16, which places the two pipelines side by side so that their shared stages and their points of divergence become apparent.

Table 3.16 summarises and contrasts the two pipelines across the dimensions that most clearly distinguish their design.

Table 3.16: *A Comparison of the Design Choices Underlying the Price Prediction and Sentiment Analysis Pipelines*

Dimension	Price Prediction Model	Sentiment Analysis Model
Learning paradigm	Machine Learning (ML)	Deep Learning (DL)

Task type	Regression (continuous price)	Multi-class classification (three sentiments)
Input data	Structured, tabular booking records	Unstructured review text
Dataset (source)	Hotel Booking Demand (Kaggle)	HRAST (Kaggle)
Core model	Gradient boosting model, selected among candidate regressors	Fine-tuned DistilBERT transformer
Feature preparation	Explicit feature engineering	Subword tokenization (learned representations)
Principal data challenge	Temporal dependence and information leakage	Class imbalance
Strategy for the challenge	Time series cross-validation	Class-weighted focal loss
Evaluation focus	Regression error metrics and the train/test gap	Accuracy with per-class precision, recall, and F1 (macro-average)
Explainability	SHAP feature attribution	Not applied
Overfitting mitigation in final training	Early stopping and regularization	Limited epochs, early stopping, and architectural regularization
Model output	Predicted price with trend, percentage change, and reasons (the recommendation)	Sentiment labels(positive, negative, or neutral), and Score for each sentiment
Invocation pattern	Scheduled, batch	Event-driven, asynchronous (non-blocking)

As the comparison makes clear, the two pipelines rest on the same sequence of core stages. Both begin with data preprocessing, proceed through EDA, model selection and training, and evaluation, and conclude with a final, optimised training phase that guards against overfitting before the model is saved and deployed. The price prediction pipeline adds two stages that the sentiment pipeline does not require: feature engineering, made necessary by its structured tabular input, and explainability through SHAP, which supplies the reasons that accompany each recommendation. The sentiment pipeline, learning directly from raw text, folds the role of feature construction into the representations its model learns internally and therefore omits a separate feature engineering stage.

The divergence runs deeper than the list of stages. The price prediction model addresses a regression task over engineered tabular features and must guard primarily against temporal leakage, which it does through a validation scheme that respects chronological order. The sentiment model addresses a classification task over textual representations and must guard primarily against class imbalance, which it does through a class-weighted objective. The two models also enter the running system through different doors, the pricing model through scheduled batch invocation and the sentiment model through event-driven asynchronous invocation, as described in Section 5. Taken together, the shared backbone and these deliberate divergences show that a single, coherent design methodology was adapted to two distinct predictive problems

7. User Interface Design Pipeline

The user interface (UI) is the layer through which every category of user ultimately experiences the platform, and its quality determines how effectively the capabilities of the system can be reached. Its design was therefore approached not as an afterthought but as a deliberate pipeline, a repeatable sequence of stages that turns the needs of each user into a coherent, consistent, and usable interface. The same pipeline was applied to the three interface domains of the system, beginning with the guest interface, proceeding to the owner dashboard, and concluding with the administrator dashboard, an order that moves from the broad public-facing experience toward the more specialised administrative workspaces. Figure 3.27 summarises this UI design pipeline.

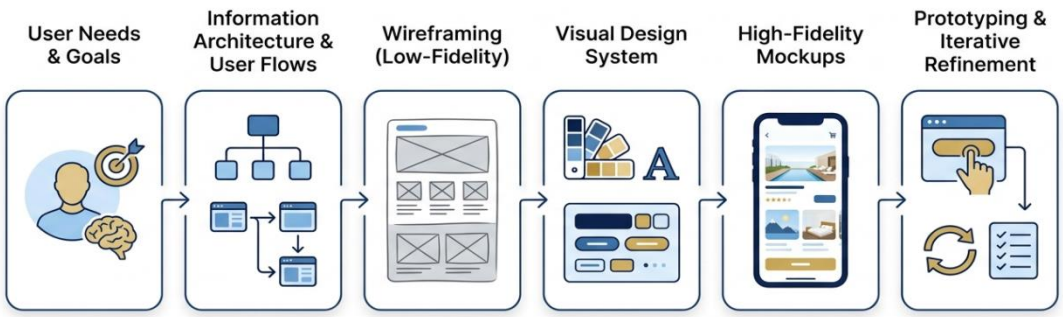


Figure 3.14: *The User Interface Design Pipeline*

The pipeline begins with the identification of user needs and goals, drawn from the actors and the functional requirements established earlier in this chapter, so that every interface is anchored in what its users must accomplish rather than in arbitrary aesthetic preference. It then proceeds to the definition of the information architecture and the user flows, which organise the content into a logical structure and map the paths a user follows to complete each task. On this foundation, low-fidelity wireframes are produced to settle the layout and the placement of elements before any visual styling is introduced. A shared visual design system is then established, comprising a consistent colour palette built on the platform's navy and gold identity, a unified typographic scale, spacing rules, and a library of reusable UI components. Applying this design system to the wireframes yields the high-fidelity design of each screen, which is finally assembled into interactive prototypes and refined iteratively. Throughout these stages, the design adheres to cross-cutting principles of responsiveness across devices, accessibility, and visual consistency, the last of which is reinforced by the shared design system so that the three interface domains remain coherent even though each serves a different audience. The realised interfaces themselves are presented in Chapter 4; the present section is concerned only with the design methodology and with the considerations that shaped each domain.

7.1. Guest Interface

The guest interface is the public face of the platform and serves its largest and most varied audience. Its design therefore prioritises clarity, ease of discovery, and a frictionless path through the reservation journey, so that a first-time visitor can move from searching for a hotel to confirming a booking with minimal effort. The information architecture is organised around this journey, from browsing and searching hotels, through inspecting a hotel and the details of its room types, to the booking flow and the subsequent management of one's reservations and account. A particular design consideration is the presentation of the price recommendation: when a guest selects a check-in date, the predicted price together with its trend and the reasons behind it must be surfaced in a form that is immediately understandable and that supports, rather than complicates, the booking decision. Because this audience is the least familiar with the system, the interface is designed to guide the guest gently through each step and to remain legible and usable across a wide range of devices.

7.2. Owner Dashboard

The owner dashboard is a focused management workspace intended for hotel owners, each of whom is responsible for a single hotel. Its design shifts the emphasis from discovery toward operational efficiency, presenting the owner with an actionable overview of the entities under their control and the means to manage them in as few steps as possible. The information architecture gathers the recurring management tasks, namely the administration of room types and their availability, the handling of reservations, and the monitoring of guest reviews together with their analysed sentiment. The dashboard also presents the pricing insight produced for the owner's own room types, allowing the owner to consult the predicted price for a chosen date and to act upon it. Because this audience works with denser and more analytical information than the guest, the design balances information density with clarity and makes deliberate use of data visualisation, so that trends and figures can be grasped at a glance.

7.3. Admin Dashboard

The administrator dashboard provides system-wide oversight and governance, and its design addresses the broadest scope of any interface in the platform. Whereas the owner dashboard is confined to a single hotel, the administrator's workspace spans every hotel, owner, user, and reservation in the system. Its information architecture is therefore organised around comprehensive management views that allow the administrator to navigate large collections of entities, to administer hotels and to provision the owner accounts associated with them, and to consult the platform-wide analytics and the results of the AI models. The design priority here is legible control over high-density information, achieved through structured, table-oriented views and a clear navigation scheme that keeps the many administrative functions organised and reachable. As with the other two

domains, the shared design system ensures that this administrative interface, despite its distinct purpose, remains part of a single, coherent visual whole.

8. Conclusion

This chapter has presented the complete analysis and design of the Hotelify platform, carrying it from a statement of requirements to a coherent and fully specified blueprint. The analysis first delimited the system by identifying its actors and by establishing the functional and non-functional requirements it must satisfy. These requirements were then given precise form through UML modelling, whose use case, class, sequence, and activity diagrams expressed both the behaviour expected of the system and the structure underlying it.

From this model, the design advanced through its successive concerns. The database design organised the system's data into a coherent structure faithful to the modelled domain. The system architecture arranged the software into cooperating tiers and defined the place of the AI service among them, together with the two patterns by which that service is invoked. The design of the two AI models was then set out as a pair of pipelines, one for price prediction and recommendation and one for sentiment analysis, each described as a sequence of methodological stages culminating in an optimised, deployable model. Finally, the design of the UI was presented as a pipeline applied in turn to the guest, owner, and administrator interfaces, unified by a shared visual design system.

Taken together, these elements constitute a complete and internally consistent design in which every part, from the data and the architecture to the intelligent models and the interfaces, is aligned with the requirements established at the outset. The chapter has deliberately confined itself to the design of the solution. The realisation of this design through the concrete technologies that implement it, and the experimental evaluation of its behaviour and performance, form the subject of the chapter that follows.

CHAPTER 04



IMPLEMENTATION, EXPERIMENTAL RESULTS & EVALUATION

1. Introduction

The preceding chapter was concerned with what the system should be and how it should be structured. It analysed the requirements of the platform, modelled its actors and their interactions, designed its database, set out its overall architecture, and described, at the level of method, the pipelines by which its two AI models and its user interfaces were to be built. That treatment was deliberately confined to design: it named concepts, algorithms, and techniques but not the concrete tools with which they would be realised, and it described methods without presenting their results.

The present chapter completes the passage from design to a working system. Where Chapter 3 asked, in principle, what the system should do and how it should be organised, this chapter shows what was actually built, with which technologies, to what effect, and how well. It is organised into five further sections that follow the natural order of construction and assessment.

The first of these, the development environment and technologies, identifies the concrete tools and libraries used to build each part of the system and explains the role of each. The second, the system implementation, describes the methodology by which the database, the backend, the AI service, and the frontend were realised and integrated, and how the whole was orchestrated for deployment. The third turns to the two AI models and reports their training and experimental results in full, from the preparation of the data through the selection of the models to their final evaluation. The fourth presents the finished application itself, following it through the eyes of each of its actors, namely the unauthenticated visitor, the guest, the owner, and the administrator, and showing how the intelligent features are woven into their experience. The fifth and final section evaluates the completed system, verifying its functions, assessing its quality attributes, comparing it with existing solutions, and judging the extent to which it fulfils the objectives of the work.

Taken together, these sections demonstrate that the design of Chapter 3 has been carried through into a complete, functioning, and intelligent application, and they prepare the ground for the general conclusion that follows.

2. Development Environment and Technologies

The realisation of the design presented in Chapter 3 rested on a deliberate selection of tools and technologies, each chosen for the role it plays in the system. This section introduces these technologies, organised by the area of the system to which they belong, and explains for each one its purpose and the reasoning behind its selection. The aim here is to establish what was used and why, while the manner in which these technologies were combined to build the system is the subject of Section 3. Because the platform is composed of three cooperating applications, a web client, a server, and a separate AI service, the technologies are presented per area, beginning with the tools

common to the whole project and proceeding through the frontend, the backend, the AI service, the development of the AI models, and the database.

2.1. General Development Environment and Tools

The development of all three applications was carried out within a single code editor, Visual Studio Code (VS Code) [73]. It was chosen because it supports the two programming languages used in the project within one consistent workspace, and because its integrated terminal, debugging facilities, and rich ecosystem of extensions allowed the frontend, the backend, and the AI service to be developed, run, and inspected without leaving the editor. Source control was handled with Git [74], and the project was hosted remotely on GitHub [75], which together preserved a complete history of the work, protected it against loss, and made each stage of development reproducible.

To keep the three applications isolated yet runnable together, the project relied on containerisation through Docker [76], coordinated by Docker Compose [77]. Containerisation guarantees that each application runs in a consistent, self-contained environment regardless of the host machine, which removes configuration discrepancies and simplifies running the full system. Dependencies were managed with the package managers native to each ecosystem, namely npm [78] for the JavaScript applications and pip [79] for the Python service. Finally, the application programming interfaces (APIs) of the server and the AI service were exercised and validated during development using Postman [80], which made it possible to send requests to each endpoint and inspect its responses in isolation, before the web client was connected to them.

Figure 4.28 provides a visual overview of the main tools and technologies used during the development process.



Figure 4.1: *Development Tools & Technologies*

2.2. Frontend Technologies

The web client was built with JavaScript as its programming language and React [81] as its core library. React was selected for its component-based model, which encourages the interface to be composed of small, reusable, and independently testable units, and for its declarative and efficient approach to updating the displayed content as the application's

data changes. To manage the state shared across these components in a predictable manner, the application uses Redux Toolkit [82], which centralises the application state and reduces the amount of repetitive code that state management would otherwise require.

The visual layer was constructed with Material UI (MUI) [83], a comprehensive library of ready-made, accessible interface components. MUI was chosen because it accelerates development while remaining fully themeable, which allowed the platform's navy and gold visual identity and its chosen typography to be applied consistently across every component through a single theme. Navigation between the views of the single-page application (SPA) [84] is handled by React Router [85], which maps each URL to the appropriate view without reloading the page. Communication with the server is carried out through Axios [86], a promise-based client used to send requests to the backend and to retrieve the data, including the outputs of the AI models, that the interface displays.

Figure 4.29 illustrates the principal technologies that compose the frontend layer of the system.



Figure 4.2: *Frontend Technologies*

2.3. Backend Technologies

The server was implemented on the Node.js [87] runtime, which executes JavaScript outside the browser and therefore allows a single language to be used across both the client and the server. Its non-blocking, event-driven model is well suited to a service whose work is dominated by handling many concurrent requests. On top of this runtime, the Express [88] framework provides the routing and middleware structure through which the server exposes its Representational State Transfer (REST) [89] API to the client and to the AI service.

Security and access control rest on several dedicated technologies. Authentication is built on JSON Web Tokens (JWT) [45], which carry a user's verified identity in a compact, self-contained, and stateless form, supporting the access, refresh, and pre-authentication tokens defined in the design. Sensitive credentials are never stored in plain form: passwords are hashed with bcrypt [90], and stored refresh tokens are protected by cryptographic hashing. Two further middleware tools harden the server at the boundary, namely Helmet [91], which sets protective HTTP response headers, and Cross-Origin

Resource Sharing (CORS) [50], which governs precisely which client origins are permitted to call the API. The recurring, time-driven operations of the system, namely the hourly automatic check-out and the daily generation of price predictions and price history, are driven by the node-cron [92] scheduling library, which triggers each task on its defined schedule.

Figure 4.30 presents the main technologies employed within the backend architecture.



Figure 4.3: *Backend Technologies*

2.4. AI Service Technologies

The AI service was implemented in Python, whose mature ecosystem for data and ML makes it the natural language in which to host and serve the trained models. The service itself is built with FastAPI [93], a modern web framework chosen for its high performance, its asynchronous request handling, and its automatic validation and documentation of the data exchanged at each endpoint, all of which suit a service whose role is to expose the models cleanly to the rest of the system. The FastAPI application is run by uvicorn [94], an Asynchronous Server Gateway Interface (ASGI) [95] server that handles the incoming requests. The two trained models, published as Kaggle Models [96], are retrieved and loaded into the service when it starts up by means of kagglehub [97], so that they are held in memory and ready to serve requests throughout the lifetime of the service.

Figure 4.31 shows the technologies that support the deployment and operation of the AI service.



Figure 4.4: *AI Service Technologies*

2.5. AI Model Development Tools

The two models were developed and trained on Kaggle [96], a cloud platform that provided both the notebook environment and the graphics processing unit (GPU) resources needed to train them efficiently. The preparation and manipulation of the data relied on pandas [98] and numpy [99], which together supply the data structures and numerical operations used throughout preprocessing and feature engineering, while matplotlib [100] and seaborn [101] were used to produce the visualisations that supported the exploratory data analysis.

For the price prediction model, scikit-learn [101] provided the supporting machinery of preprocessing pipelines, model selection, and validation, including the chronological splitting and randomized search strategies, as well as the random forest baseline, while XGBoost and LightGBM supplied the gradient boosting regressors among which the final model was chosen. The interpretation of the model was carried out with the SHAP library, which computes the feature contributions underlying its predictions. For the sentiment analysis model, the Transformers library [102] provided the pre-trained DistilBERT model together with the tooling required to fine-tune it, and it operates on top of the PyTorch [103] DL framework, which performs the underlying training computations.

Figure 4.32 highlights the key tools involved in the development and training of the AI models.

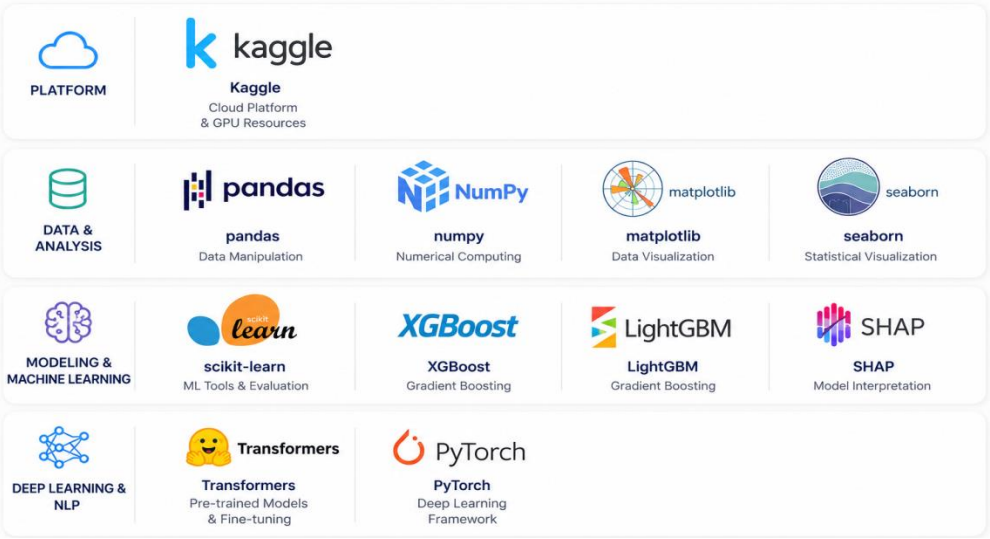


Figure 4.5: *AI Model Development Tools*

2.6. Database and Data Management

The system's data is stored in MongoDB [104], a document-oriented database whose flexible, document-based storage aligns naturally with the nested and heterogeneous structure of the platform's collections, as established in the database design of Chapter 3. The server interacts with the database through Mongoose [105], an Object Data Modeling (ODM) library that adds a schema definition, validation, and a structured modelling layer on top of MongoDB, allowing the application to work with well-defined data models rather than raw documents. To populate the database with its initial content, the generated hotel data was loaded in bulk using the mongoimport [106] utility.

Figure 4.33 provides a visual summary of the technologies used for data storage and management.

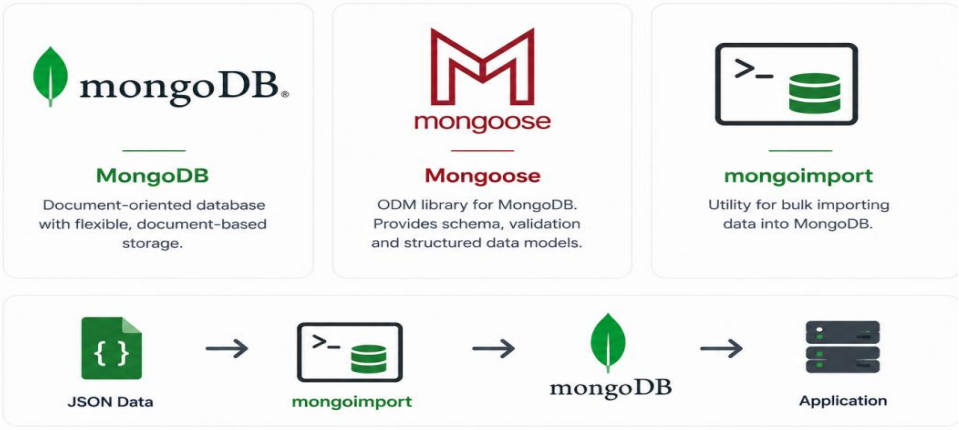


Figure 4.6: *Database & Data Management Technologies*

2.7. AI-Assisted Development Tool

Alongside the technologies that constitute the system itself, one supporting tool was used during its development, namely Claude AI [107], an AI assistant developed by Anthropic that can be applied to tasks such as writing, analysis, and programming.Within this project it served strictly as an auxiliary aid rather than as a foundation of the work. It was used to help organise certain ideas, to support understanding and analysis at particular stages, and to assist with phrasing, reviewing, and technical exploration.

Its usefulness, however, was contingent on prior knowledge and effort on the part of the developer. Drawing any benefit from it required an existing understanding of the problem, independent research, and sufficient experience to direct it correctly and to judge the soundness of its suggestions. It was therefore not a substitute for technical knowledge or for the personal work that the project demanded, but only a means of improving the efficiency of some tasks that had already been understood and executed.

Figure 4.34 presents the AI assistant used as a supporting tool during the development process.



Figure 4.7: *AI-Assisted Development Tool*

3. System Implementation

Building on the technologies introduced in the previous section, this section describes how the system was actually constructed. It follows the same layering as the architecture of Chapter 3, presenting in turn the implementation of the database, the backend, the AI service, and the frontend, and concluding with the way these parts were integrated and deployed as a single running system. The emphasis throughout is on the methods and mechanisms by which each part was realised rather than on its source code, so that the behaviour and the results presented later in this chapter can be understood in light of how the system works.

3.1. Database Implementation

The thirteen collections defined in the database design of Chapter 3 were realised as a set of schemas declared through the ODM layer. Each schema specifies the fields of its collection together with their data types, their validation rules, their default values, and the constraints derived from the design, so that the structure of every document is enforced before it is stored. The enumerations established in the class diagram were implemented as restricted fields that admit only their permitted values, and the relationships between entities were expressed by storing references from one document to another, for example from a hotel to the owner who manages it, or from a reservation

to the guest and the room type it concerns. This referencing approach keeps the independent entities of the system distinct while preserving the links between them, and embedding was reserved for the tightly bound data that is always accessed together with its parent.

Beyond the schemas, the implementation defined the indexes needed to support the queries the system performs most frequently, so that lookups remain efficient as the data grows. A Time To Live (TTL) index was applied to the stored authentication tokens, which causes expired tokens to be removed automatically once their lifetime elapses, directly realising the token expiry behaviour required by the design. Finally, the database was populated with realistic content: the generated set of hotels, spread across several North African and Middle Eastern countries, together with their room types, facilities, and mapped images, was loaded in bulk using the import utility, yielding a fully seeded database suitable for development and for the demonstration of the system.

3.2. Backend Implementation

The server was implemented as a layered application that separates the handling of requests from the business logic and the data access. Each route of the REST API delegates an incoming request to a controller, which validates and interprets it, and then to the service logic that carries out the operation against the data models. This organisation keeps the codebase maintainable and mirrors the application tier of the architecture, in which the server acts as the single coordinating point between the client, the database, and the AI service.

Authentication and session management were a central concern of the implementation. When a user signs in, the server issues a short-lived access token that accompanies each subsequent request, and a longer-lived refresh token that allows a new access token to be obtained without re-entering credentials. Refresh tokens are rotated, meaning that each use issues a fresh token and invalidates the previous one, and they are never stored directly: only a hashed form, produced with the SHA-256 algorithm, is kept, and it is removed automatically by the TTL index once it expires. For the privileged roles of owner and administrator, an additional verification step was implemented in which a pre-authentication token grants only the right to submit an access code, and full access is issued only once that code is confirmed, realising the two-factor authentication of the design.

Access to every protected operation is governed by RBAC, enforced through middleware that inspects the authenticated user's role before a request is allowed to proceed, so that guests, owners, and administrators each reach only the operations permitted to them, and owner-scoped data is confined to the hotel a given owner manages. At the boundary of the server, the security middleware introduced earlier is applied to every request, and a complete password reset flow was implemented to allow users to recover access safely. Lastly, the recurring operations of the system were implemented as scheduled jobs: an hourly job automatically checks out the reservations whose stay has ended, and a daily

job, running at midnight, invokes the AI service to produce the price predictions for every room type, stores them, and updates the corresponding price history. In these exchanges the server acts as a client of the AI service, sending it work and persisting the results it returns.

3.3. AI Service Implementation

The AI service was implemented as an independent application, separate from the server, whose sole responsibility is to host the two trained models and expose them through a small, well-defined API. To avoid the cost of loading a model on every request, the service loads both models only once, when it starts: a startup routine retrieves the price prediction model and the sentiment analysis model, published as Kaggle Models, and holds them in memory for the entire lifetime of the service, so that every subsequent request is served immediately.

The service exposes one capability for each model. The price prediction capability is designed to be called in bulk by the server's daily job: it receives a batch covering all room types, applies the model, and returns for each one the predicted price together with its trend, its percentage change, and the reasons behind it. The sentiment capability receives the text of a guest review and returns its sentiment classification, and it is invoked asynchronously by the server so that the analysis never delays the guest. In both cases, the data presented to a model at the moment of prediction is transformed in exactly the same way as during training, since the price model carries its preprocessing within the same pipeline used to train it and the sentiment model applies the same tokenisation, which guarantees that the models behave consistently in production and in the laboratory.

3.4. Frontend Implementation

The web client was implemented as a single-page application assembled from small, reusable components. Each component was organised with a consistent internal structure that separates its markup, its styling, and its styled building blocks, which keeps the interface code uniform and easy to extend. A single central theme encodes the platform's navy and gold identity and its typography, and because every visual component draws from this theme, the three interface domains share one coherent appearance.

The state shared across the application, such as the identity of the signed-in user and the data retrieved from the server, is held in a central store divided into focused slices, each responsible for one area of the application. Data is fetched from the server asynchronously through this store and delivered to the components that display it, with the requests themselves carried out by the HTTP client introduced earlier. The guest interface, the owner dashboard, and the administrator dashboard were each implemented as a set of routed views, and navigation between them is guarded

according to the user's role, so that each user can reach only the views intended for them. A particular part of this work was the presentation of the intelligent outputs of the system: when a guest selects a check-in date, or an owner selects a date in the dashboard, the interface retrieves the pre-computed prediction and renders the predicted price, its trend, its percentage change, and its reasons in a readable form, and the analysed sentiment of guests is surfaced alongside the reviews to which it belongs.

3.5. System Integration and Deployment

Once the individual parts were built, they were brought together into a single, runnable system. The web client, the server, the AI service, and the database were each placed in their own container, and these containers were orchestrated with Docker Compose, which declares the four services, their configuration, and the network that joins them, and allows the entire platform to be started with a single command in a consistent environment.

Within this arrangement, the services communicate over a shared internal network and address one another by service name rather than by fixed network addresses, so that the server reaches the AI service and the database reliably regardless of where the system is run, and the configuration that differs between environments is supplied externally rather than embedded in the code. This integration turns the multi-tier architecture of Chapter 3 into a set of cooperating running processes, and it made the complete system straightforward to launch, exercise, and demonstrate as a whole, which is the form in which its results and behaviour are presented in the remainder of this chapter.

4. AI Models: Training and Experimental Results

This section presents the experimental results obtained when the two model design pipelines of Chapter 3 were carried out in practice. For each model, it follows the stages defined in that design and reports the concrete outcome of each, namely the data produced by the preparatory stages, the findings of the exploratory analysis, the behaviour observed during training and selection, and the measures by which the resulting model was evaluated. The results that Chapter 3 deliberately deferred are thus delivered here, in the order in which they were obtained. The price prediction model is treated first, followed by the sentiment analysis model.

4.1. Price Prediction Model Results

This subsection reports the results of the price prediction pipeline, presenting the concrete output of each of its stages in turn.

4.1.1. Data Preprocessing

Preprocessing cleaned the raw booking records and reduced them to the fields relevant to the prediction of price. The outcome is a dataset of 79,806 records described by eight

fields: the hotel, the reserved room type, the three components of the arrival date, the per-night base price that forms the basis of the target (this price was correctly converted from Euro to US Dollar to match the currency used by the system, as the raw dataset used Euro), an indicator of whether the booking was cancelled, and the lead time. Table 4.17 shows a sample set of the resulting dataset.

Table 4.1: *A Sample of the Processed Price Dataset*

#	hotel	reserved_room_type	arrival_date_year	arrival_date_month	arrival_date_day_of_month	base_price	is_canceled	lead_time
1	Resort Hotel	C	2015	July	1	219.76	0	342
2	Resort Hotel	A	2016	July	16	131.57	0	151
3	Resort Hotel	D	2015	July	8	167.9	0	5
4	City Hotel	A	2016	September	27	148.35	1	16
5	City Hotel	A	2016	July	16	78.20	0	156
6	City Hotel	A	2017	August	18	102.92	0	0

4.1.2. Exploratory Data Analysis (EDA)

The exploratory analysis examined the cleaned data in order to characterise the price and to uncover the factors that govern it. It began with the distribution of the base price. As Figure 4.35 shows, the price is strongly right-skewed: the great majority of rooms fall between roughly 50 and 200 dollars, with a pronounced concentration around one 100 dollars and a long, thin tail extending toward the upper end of the data. The boxplots in Figure 4.36 confirm this picture for the price and add the same examination for the lead time. The lead time is widely dispersed, ranging from same-day bookings to almost a full year in advance, with a median close to 50 days and a considerable population of high outliers, while the base price, whose median lies near 116 dollars, likewise carries a body of high-value outliers above its main mass. Both quantities are therefore skewed and heavy-tailed, a property that the later modelling must accommodate.

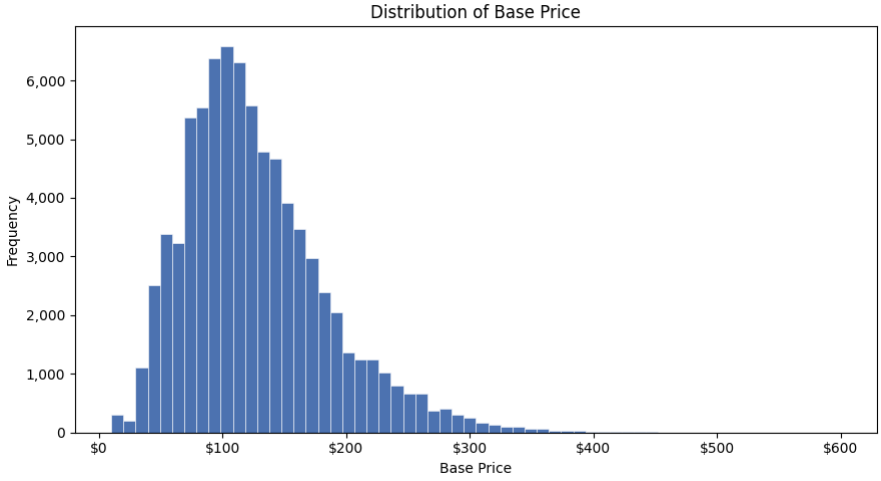


Figure 4.8: *Distribution of the Base Price*

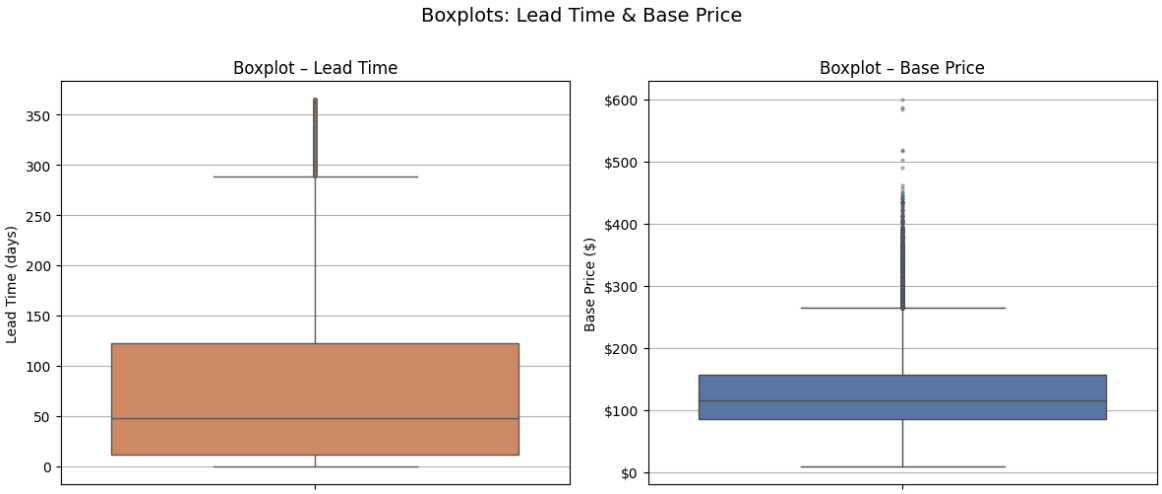


Figure 4.9: *Boxplots of the Lead Time and the Base Price*

The price was then examined across the reserved room types. Figure 4.37 reveals a clear and consistent ordering: certain room types command markedly higher prices, with median values approaching 100 dollars, whereas others cluster near 200 dollars, and the spread of prices within each type differs as well. This strong dependence of price on room type identifies the room type as a primary determinant of price and motivates its central role in the features constructed later.

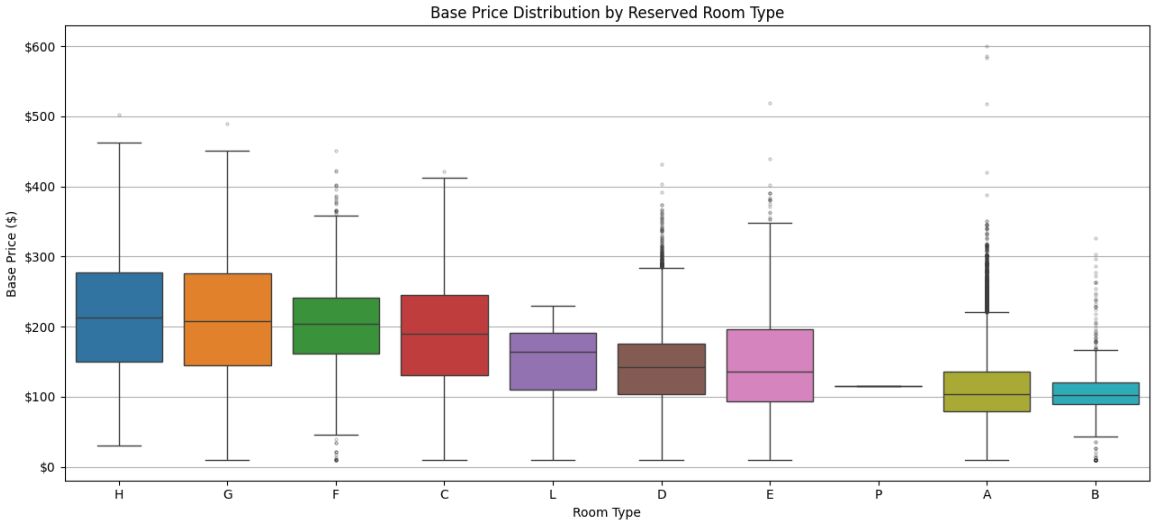


Figure 4.10: *Base Price Distribution by Reserved Room Type*

Turning from price to demand, the analysis aggregated the bookings and their cancellations over the calendar. Table 4.18 reports these totals by month. Demand is plainly seasonal: it builds through the spring to a marked peak in July and August and recedes in the late autumn and winter, while cancellations rise and fall together with bookings and amount to a substantial fraction of all reservations across the year. The same totals examined week by week confirm this seasonal rhythm at a finer resolution.

Table 4.2: *Total Bookings and Cancellations by Month*

Month	Total Bookings	Total Cancellations
January	4,237	1,004
February	5,608	1,366
March	6,842	1,773
April	7,188	2,303
May	7,515	2,336
June	7,150	2,288
July	9,536	3,113
August	10,681	3,524
September	5,825	1,531
October	6,046	1,552
November	4,453	1,001

December	4,725	1,319

This seasonality is mirrored in the price itself. Figure 4.38 traces the average base price across the year, both by month and by week, and shows the price climbing to a clear peak at the height of summer before declining toward the low season. Demand and price therefore move together over time, which establishes the temporal dimension as a further essential determinant of price.

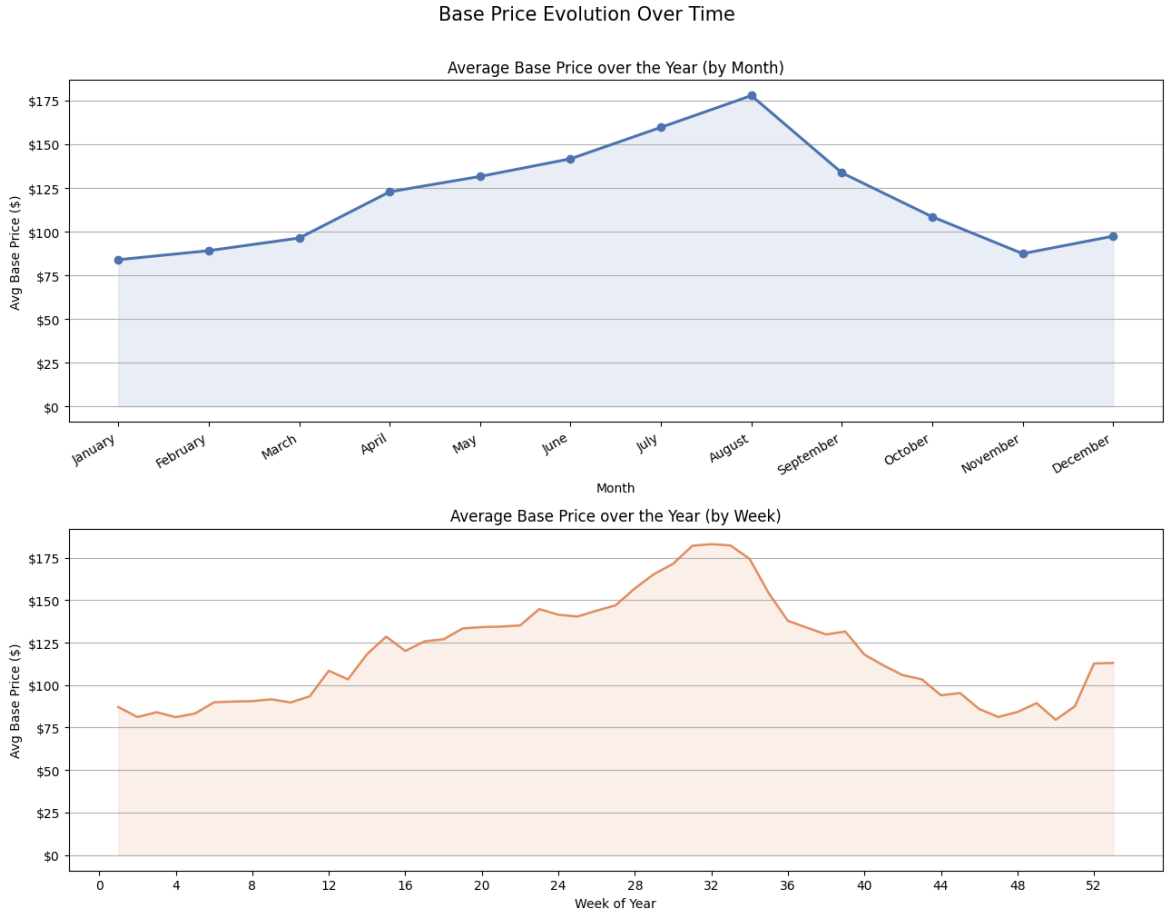


Figure 4.11: *Evolution of the Average Base Price Over the Year*

Taken as a whole, the exploratory analysis shows that the base price is a skewed quantity governed chiefly by the room type and by the season and time of year, against a backdrop of strongly seasonal demand. These observations directly shaped the feature engineering described next.

4.1.3. Feature Engineering

Guided by these findings, the feature engineering stage transformed the per-booking records into a dataset suited to learning how price varies with demand over time. Rather than one row per reservation, each record now represents a single room type of a hotel observed for a given check-in date from a given point in time, so that the same future date is seen at successively shorter lead times as it approaches. The result is a dataset of

77,378 records described by 25 features, with the per-night price as the target to be predicted.

These features fall into several groups. The temporal context of each observation is captured through the day and the year and through a cyclical, sine and cosine encoding of the month, the day of the week, and the week of the year, which represents these recurring cycles without imposing a false break between their endpoints, together with the lead time in days and a categorical season. Calendar effects are marked by flags for holidays and for events. The state of demand is expressed through the counts of bookings and of cancellations, the number of available rooms, the occupancy rate, the booking velocity, and the estimated number of days until the room type is fully booked. Finally, recent history is summarised by the average price and the average occupancy over the preceding 7 days, which endow the model with a short-term memory of how each room type has lately been priced and filled. Table 4.19 presents a sample set of the resulting dataset, restricted to a representative subset of these features.

Table 4.3: *A sample set of the feature-engineering dataset, showing a representative subset of the engineered features.*

#	hotel_id	room_type_cat	check_in_date	lead_time_days	season	occupancy_rate	booking_velocity	days_to_sellout	previous_7d_avg_price	predicted_price
1	1	deluxe	2015-07-13	51	peak	0.41	0.00	90.0	91.42	91.42
2	1	standard	2016-03-17	3	low	0.52	0.71	42.0	40.25	41.40
3	1	superior	2017-06-23	28	shoulder_high	0.42	0.14	90.0	195.50	195.50
4	2	standard	2016-03-11	64	low	0.60	0.57	90.0	75.30	59.77
5	2	standard	2017-05-10	125	shoulder_high	0.36	0.00	90.0	124.20	124.20
5	2	deluxe	2015-10-01	17	shoulder	0.10	0.14	90	140.01	0.049

Through these engineered features, the seasonal, room type, lead time, and demand signals surfaced by the exploratory analysis are made explicitly available to the model, providing the demand-aware basis on which the prediction of price is learned.

4.1.4. Model Selection and Training

With the feature set in place, the three candidate algorithms were tuned and compared. Each candidate was embedded in a pipeline that first encoded the categorical features, namely the room type and the season, so that training and later prediction would treat the data identically, and the hyperparameters of each were then searched by a randomized search whose folds respected the chronological order of the data. The search reported, for every candidate, its best configuration and the cross-validation error it achieved. Table 4.20 gives the best cross-validation MAPE of each model, and Table 4.21 lists the tuned hyperparameters that produced it.

Table 4.4: *Best Cross-Validation MAPE Obtained by the Randomized Search for Each Candidate Model*

Candidate Model	Best Cross-Validation MAPE
Random Forest	0.1563
LightGBM	0.1592
XGBoost	0.1606

Table 4.5: *Tuned Hyperparameters Selected by the Search for Each Candidate Model*

Model	Tuned hyperparameters
LightGBM	n_estimators= 300; num_leaves= 25; max_depth= 4; learning_rate= 0.03; min_child_samples= 30; min_split_gain=0.05; colsample_bytree=0.85; subsample= 0.75; subsample_freq= 3; reg_alpha= 1.0; reg_lambda= 0.0; path_smooth= 0.1
XGBoost	n_estimators= 600; max_depth= 3; learning_rate= 0.02; min_child_weight= 7; gamma= 0.0; subsample= 0.8; colsample_bytree= 0.9; colsample_bylevel= 0.9; reg_alpha= 2.0; reg_lambda= 3.0
Random Forest	n_estimators= 700; max_depth= 15; min_samples_split= 10; min_samples_leaf= 30; max_features= 0.8; max_samples= 0.65; min_impurity_decrease= 0.0; ccp_alpha= 0.001

The three models reached very similar cross-validation errors, which left the final choice to be settled by a fuller evaluation on data the models had not seen.

4.1.5. Evaluation

The three tuned models were then assessed on the held-out test set according to the composite strategy established in the design, which weighs several complementary measures rather than a single one. Table 4.22 reports, for each model, its error on both the training and the test set, its mean absolute error and root mean square error, the time it took to train and to predict, the ratio that measures the gap between its training and test error, and the composite score that combines these quantities, a score for which a lower value denotes a better overall balance.

Table 4.6: *Evaluation of the three candidate models on the test set*

Model	Train MAPE	Test MAPE	MAE	RMSE	Training Time (s)	Prediction Time (s)	MAPE Gap Ratio	Composite Score
LightGBM	0.1256	0.1592	15.46	24.00	7.56	0.19	0.2675	0.4753
XGBoost	0.1310	0.1606	15.60	23.93	2.26	0.17	0.2260	0.5000
Random Forest	0.1158	0.1563	15.03	24.28	90.85	0.74	0.3497	0.5000

The three models were closely matched in raw accuracy, with their test errors separated

by less than half a percentage point. The decisive differences lay elsewhere. The random forest attained the lowest test error but also the widest gap between its training and test performance, a sign of greater overfitting, and it was by far the slowest to train. The composite score, which rewards the best balance of accuracy, generalisation, and efficiency, was lowest for LightGBM, which paired competitive accuracy with a controlled generalisation gap and a short training time. LightGBM was therefore selected as the model to carry forward.

4.1.6. Explainability (SHAP)

To understand how the selected model reaches its predictions, its behaviour was analysed with SHAP, which assigns to every feature a contribution toward each prediction. Aggregating these contributions over the data gives the global importance of each feature, reported in Table 4.23.

Table 4.7: *Global Feature Importance of the Selected Model, Measured by the Mean Absolute SHAP Value*

Feature	Mean absolute SHAP value
previous_7d_avg_price	36.19
room_type_cat	2.37
week_of_year_cos	2.20
season	1.82
lead_time_days	1.27
booking_velocity	1.21
available_rooms	1.15
year	0.93
cancellation_counts	0.81
week_of_year_sin	0.58

One feature dominates the model's behaviour. The average price of the room type over the preceding 7 days carries a mean absolute SHAP value of about 36, more than 15 times that of the next most influential feature, which confirms that the recent price history is by far the strongest driver of the predicted price. The features that follow, although far smaller in effect, are precisely those that the exploratory analysis had identified as relevant, namely the room type, the position within the yearly cycle and the season, the lead time, and the indicators of demand such as the booking velocity and the number of available rooms. The remaining features each contributed less than the tenth listed and play only a minor role.

This global picture is complemented by three further views. The summary plot in Figure 4.39 shows not only that the recent average price is the most influential feature but also the direction of its effect, with high recent prices pushing the prediction upward and low ones pulling it down. The dependence plot in Figure 4.40 isolates this feature and shows that its contribution rises steadily as its value increases, confirming a clear and consistent positive relationship. Finally, the waterfall plot in Figure 4.41 explains a single prediction,

showing how the model departs from the average prediction of the dataset and how each feature, led by the recent average price, adds to or subtracts from the final estimate. Together, these analyses confirm that the model is interpretable and that its reasoning agrees with domain expectations, and they provide the basis for the reasons that accompany each recommendation in the system.

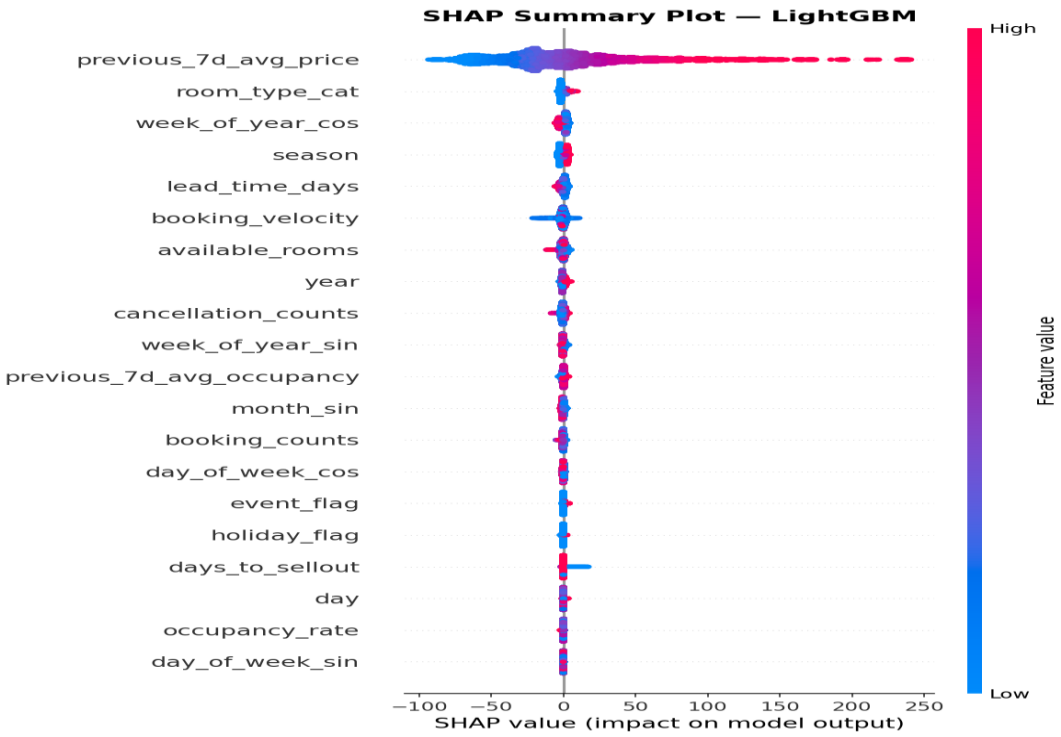


***Figure 4.12**: SHAP Summary Plot of the Selected Model*

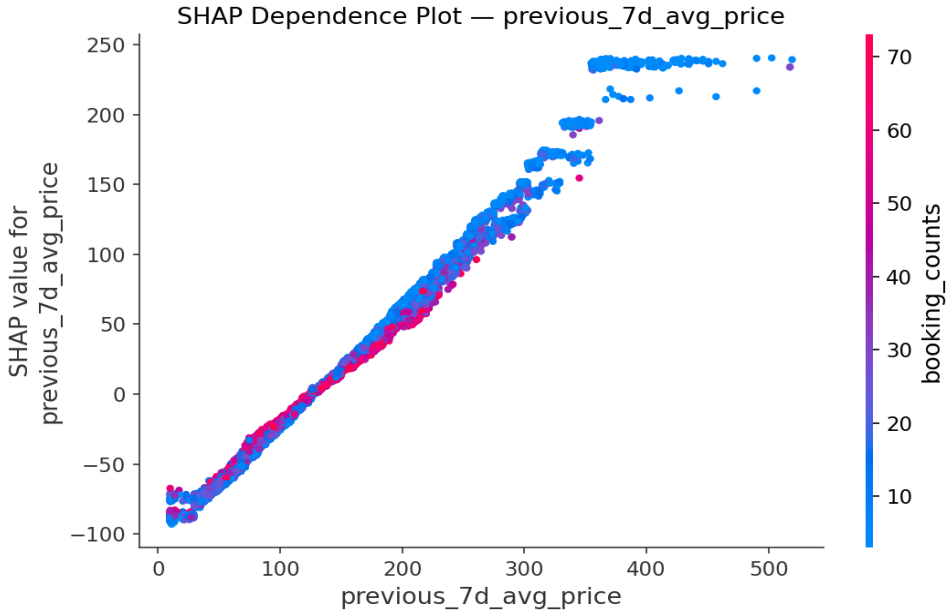


***Figure 4.13:** SHAP Dependence Plot for the Recent Average Price*

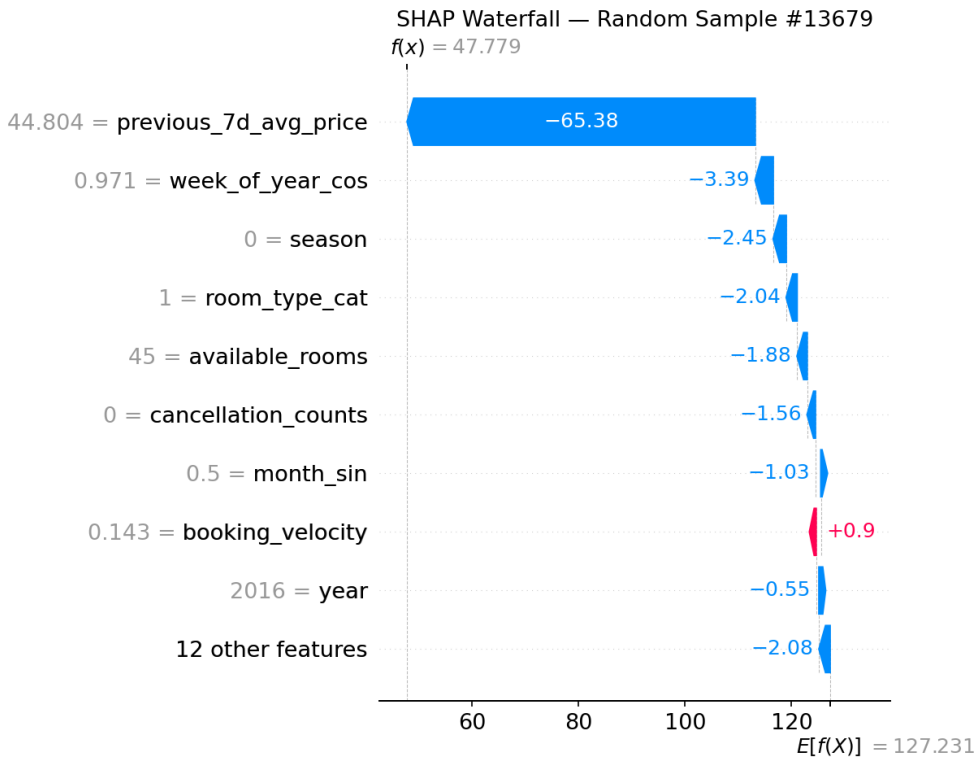


Figure 4.14: *SHAP Waterfall Plot Explaining an Individual Prediction*

4.1.7. Final Model Training and Overfitting Mitigation

Once LightGBM had been selected and understood, it was retrained as the final model with an explicit safeguard against overfitting. Training was monitored on a validation set and halted automatically once the validation error stopped improving, so that the model would not continue to fit the training data at the expense of its ability to generalise. As Figure 4.42 shows, the validation error improved up to the best iteration and training was then stopped early at that point. The final model reached a training error of 0.1431 and a test error of 0.1573, measured as MAPE.

```
Training until validation scores don't improve for 50 rounds
[50]    valid_0's mape: 0.175931        valid_0's l2: 733.114
[100]   valid_0's mape: 0.160963        valid_0's l2: 671.184
[150]   valid_0's mape: 0.157359        valid_0's l2: 656.894
[200]   valid_0's mape: 0.157493        valid_0's l2: 653.015
Early stopping, best iteration is:
[155]   valid_0's mape: 0.157252        valid_0's l2: 656.432
Evaluated only: mape
Best iteration: 155
Final MAPE Train: 0.1431
Final MAPE Test: 0.1573
```

Figure 4.15: *Training of The Final Model with Early Stopping*

The effect of this safeguard is visible when these figures are compared with those of the selection stage. The gap between the training and test error narrowed markedly, from roughly three and a half percentage points to slightly over one, while the test error itself

improved slightly. In other words, the final model fits the training data a little less tightly but generalises noticeably better, which is exactly the outcome the overfitting mitigation was intended to achieve.

4.1.8. Model Saving and Deployment

The final model, together with the preprocessing it carries within its pipeline, was then saved to a single file, ***dynamic_pricing_model.pkl***, so that the complete fitted model could be reloaded and used for prediction without any retraining. In this form it is ready to be loaded by the AI service at startup and invoked by the daily batch job, as described in the implementation, closing the gap between the model developed here and its role in the running system.

4.2. Sentiment Analysis Model Results

This subsection reports the results of the sentiment analysis pipeline, presenting the concrete outcome of each of its stages in turn.

4.2.1. Data Preprocessing

Preprocessing cleaned the raw guest reviews and reduced them to the form required for supervised classification. The outcome is a dataset of 23,110 labelled comments, each described by an identifier, its cleaned text, and the single sentiment label assigned to it. Table 4.24 shows a sample set of the resulting dataset.

Table 4.8: *A Sample Set of the Processed Sentiment Dataset*

comment_id	comment	sentiment
146	a comfortable seat at reception for arrival.	positive
102	a bit far from the center but quiet.	neutral
91	a bigger room would have been better.	negative

4.2.2. Exploratory Data Analysis (EDA)

The exploratory analysis examined the cleaned data in order to understand the classification problem and the nature of the text. Its first and most consequential finding concerns the distribution of the three classes. As Figure 4.43 shows, the dataset is strongly imbalanced: positive reviews make up 51.1% of the data and negative reviews 45.4%, while neutral reviews account for only 3.4%. This severe under-representation of the neutral class is the central difficulty of the task and is precisely what the class-weighted focal-loss strategy of the design was intended to counter.

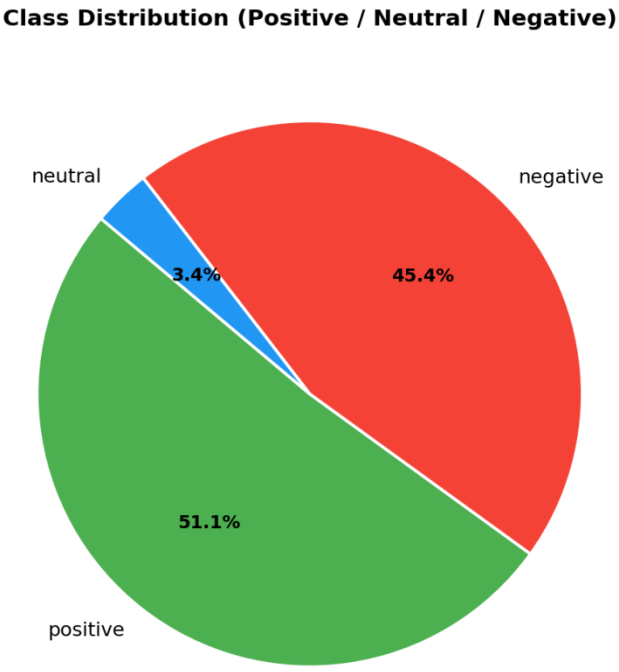


Figure 4.16: *Distribution of the Three Sentiment Classes*

The exact composition of the data, together with the length of the reviews, is summarised in Table 4.25. The dataset comprises 11,818 positive, 10,501 negative, and only 791 neutral comments. The reviews are uniformly short, averaging about 10 words and 58 characters overall, with the neutral and negative reviews tending to be marginally longer than the positive ones.

Table 4.9: *Summary Statistics of the Review Length, by Class*

Class	Num. Comments	Min Words	Mean Words	Max Words	Min Chars	Mean Chars	Max Chars
Positive	11,818	2	9.17	123	9	53.23	678
Neutral	791	2	11.05	46	8	59.46	261
Negative	10,501	2	11.51	97	8	62.33	490
Overall	23,110	2	10.30	123	8	57.58	678

The distribution of these lengths is shown per class in Figure 4.44. In every class the length is right-skewed, with the overwhelming majority of reviews containing only a handful of words, so that the maximum sequence length of 128 tokens fixed in the design comfortably accommodates essentially the entire dataset. Finally, the content of the reviews was examined through the most frequent words of each class, shown in Figure 4.45. The vocabulary is clearly discriminative: positive reviews are dominated by approving terms such as good, nice, clean, friendly, and comfortable, alongside the aspects most often praised, namely the location and the staff; negative reviews are marked by complaints about the room and the bathroom and by words such as small,

shower, noise, noisy, and water; and neutral reviews are characterised by plainer, descriptive language. This confirms that the text carries a clear and learnable sentiment signal.

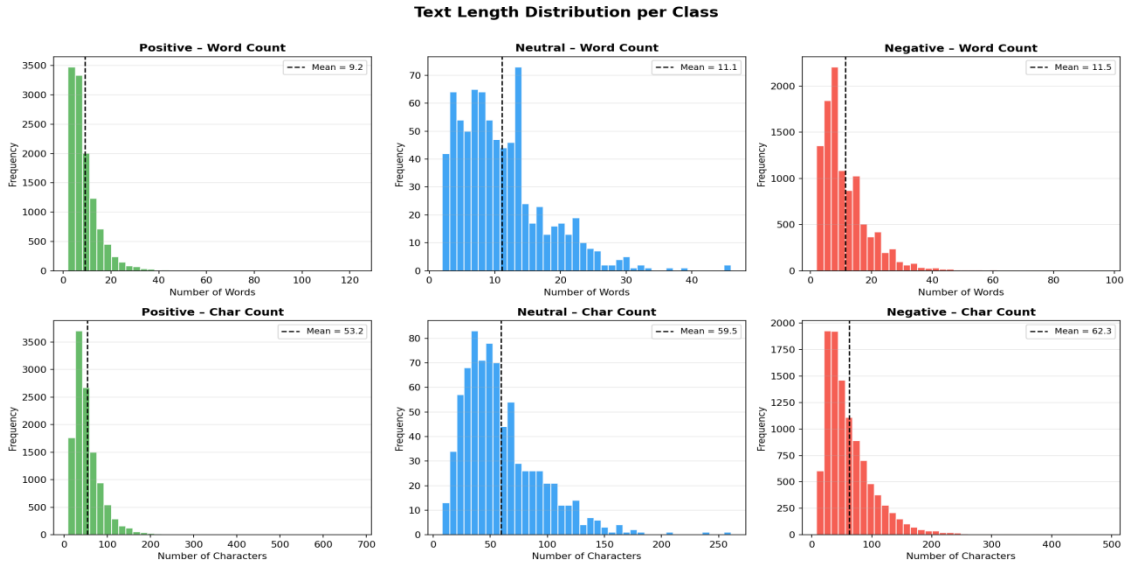


Figure 4.17: *Distribution of Review Length, in Words and Characters, per Class*

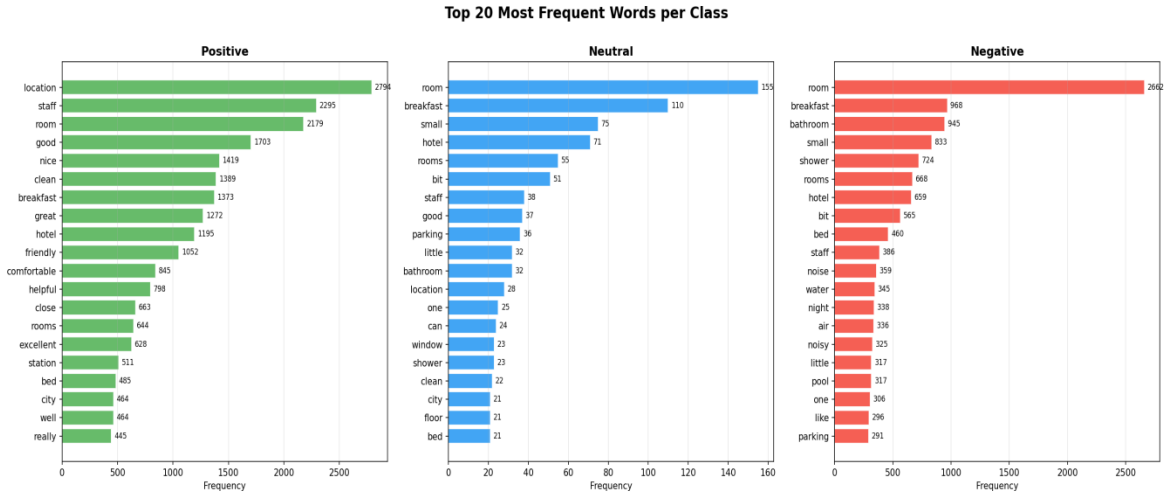


Figure 4.18: *The 20 Most Frequent Words In Each Sentiment Class*

4.2.3. Model Selection and Training

The DistilBERT model chosen in the design was first trained while its behaviour was monitored on a validation set, so as to study how it learned and to determine the amount of training it should receive. The per-epoch record of this initial training is given in Table 4.10. It exhibits the characteristic signature of overfitting: the training loss falls steadily as the epochs proceed, whereas the validation loss, after improving over the first epochs, rises continuously thereafter. The classification metrics qualify this picture. The accuracy, precision, recall, and F1-score on the validation set improve over the first three epochs and then effectively plateau, so that little is gained beyond the third epoch even as the validation loss continues to climb. The third epoch therefore emerges as the optimal

point, attaining a validation F1-score close to 0.936, almost the highest reached at any epoch, while its validation loss remains among the lowest, before overfitting sets in. This analysis fixed the number of epochs to be used when training the final model.

Table 4.10: *Per-Epoch Results of the Initial Training of the Model*

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1-score
1	0.7277	0.7119	0.9254	0.9231	0.9254	0.9236
2	0.7174	0.7621	0.9275	0.9324	0.9275	0.9298
3	0.3613	0.9633	0.9356	0.9355	0.9356	0.9355
4	0.3477	1.1296	0.9340	0.9357	0.9340	0.9348
5	0.1309	1.3609	0.9351	0.9351	0.9351	0.9351
6	0.2878	1.3825	0.9362	0.9366	0.9362	0.9364

4.2.4. Evaluation

The fine-tuned model was then evaluated, with the results reported in Table 4.27. On unseen reviews the model achieved a test F1-score of 0.936, together with a precision of 0.936, a recall of 0.937, and an accuracy of 0.937, indicating strong and well-balanced performance across the classes. The considerably higher training F1-score of 0.990 set against this test score reveals a tendency toward overfitting, which the final training stage was designed to contain; the test performance nonetheless remains high.

Table 4.11: *Overall Evaluation of the Model*

Train F1-score	Test F1-score	Precision	Recall	Accuracy
0.9900	0.9363	0.9356	0.9370	0.9370

4.2.5. Final Model Training and Overfitting Mitigation

Guided by this finding, the final model was trained and stopped at the third epoch, the point identified above as the best balance between fit and generalisation, so that training would end before overfitting could degrade it. Table 4.28 reports the training loss recorded during this final run, which falls from 0.946 to 0.615 as the training advances toward that point. Stopping the training at the selected epoch is the principal means by which overfitting is mitigated, and it is reinforced by the class-weighted focal-loss objective, which throughout offsets the dominance of the two majority classes over the scarce neutral class. The result is the deployable final model, trained precisely to the point that the earlier analysis had identified as optimal for generalisation.

Table 4.12: *Training Loss of the Final Model*

Step	Training Loss
1000	0.945999
2000	0.614568

4.2.6. Model Saving and Deployment

The final fine-tuned model was saved together with its tokenizer, so that any review presented to it later is encoded into tokens in exactly the same way as during training. The two were then deployed within the AI service, where the model classifies each newly submitted review as described in the implementation, completing the path from the trained classifier to its role in the running system.

5. Presentation of the Web Application

The preceding sections have described how the system was built and how its models perform. This section presents the system as the user finally encounters it, namely the realised web application. It is organised by interface, beginning with the unauthenticated user interface, then the guest interface, and finally the owner and administrator dashboards, and within each interface it follows the operations that a user actually performs. Each operation is illustrated with annotated screenshots of the running application, so that the design set out in Chapter 3 can be seen in its concrete, working form.

5.1. Unauthenticated User Interface

Before a visitor signs in, the platform exposes a distinct set of pages that are common to everyone and are not tied to any particular role. These comprise the public landing page, the mechanism that guards the actions reserved for registered users, and the authentication pages through which a visitor signs in, registers, or recovers access, including the additional verification required of the privileged roles. This subsection presents these pages in turn.

5.1.1. Home Page

The Home page is the entry point of the platform and the first page presented to any visitor who has not yet signed in, that is, to an unauthenticated visitor. As shown in Figure 4.46, it opens with a header carrying the Hotelify logo, a control for switching between a light and a dark theme, and the Log In and Sign Up actions, followed by a prominent hero area that states the purpose of the platform and offers a search bar. Through this bar the visitor specifies a destination, the check-in and check-out dates, and the number of guests (adults and children) and finally the number of rooms, and then launches a search. The page therefore allows a visitor to begin looking for a stay immediately, without first creating an account.

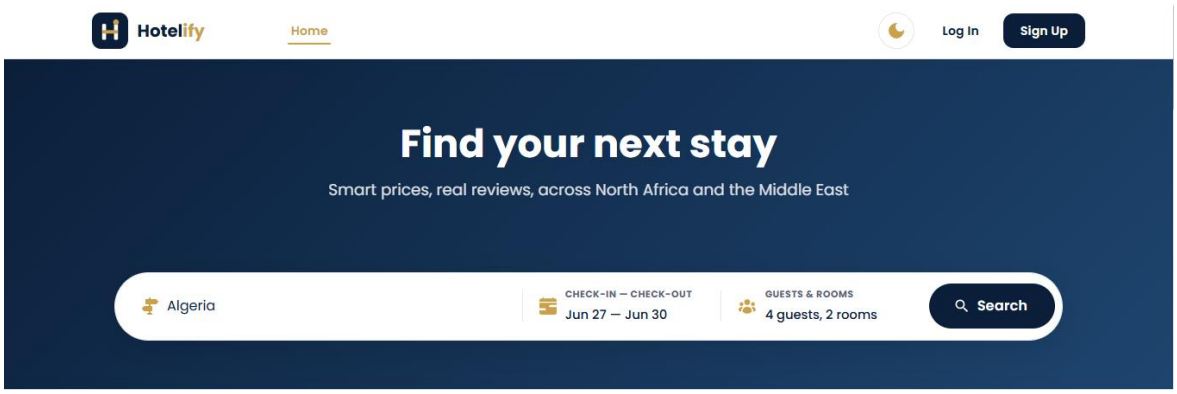


Figure 4.19: *The Hero Area and Search Bar of the Home Page*

Below the search bar, the page presents the available hotels grouped by country, each country introducing a horizontally scrollable row of hotel cards, as shown in Figure 4.47 in both the light and the dark theme. Each card displays a carousel of photographs of the hotel, its name and location, a rating badge accompanied by a qualitative label and the number of reviews, a star rating, the starting price per night, and a heart icon for adding the hotel to a list of favourites. These elements give the visitor a concise overview of each hotel and embody the two capabilities announced in the hero area, namely informed pricing and reviews drawn from real guests.

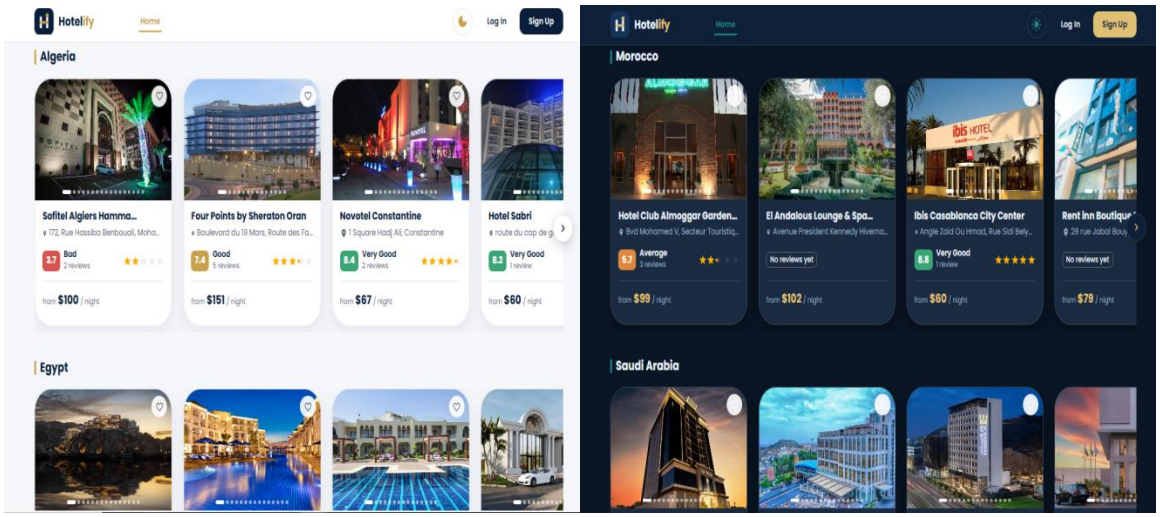


Figure 4.20*: Browsing Hotels by Country on the Home Page, Light Mode and in Dark Mode*

The Home page thus serves as the hub from which the rest of the experience unfolds. From it, the visitor may search and browse freely, open a hotel to inspect its details, or move toward the Log In and Sign Up pages. Two of the actions offered on this page, however, namely adding a hotel to the favourites and booking a stay, are reserved for registered users, and any attempt to perform them without an account is intercepted by the mechanism described next.

5.1.2. Authentication Guard

Certain actions are meaningful only for a registered user, because they are bound to a personal account. Adding a hotel to the favourites stores a preference against the user's account, and booking a stay creates a reservation in that user's name. When an unauthenticated visitor attempts either of these actions, the system does not simply ignore the attempt; instead it presents the authentication guard shown in Figure 4.48.

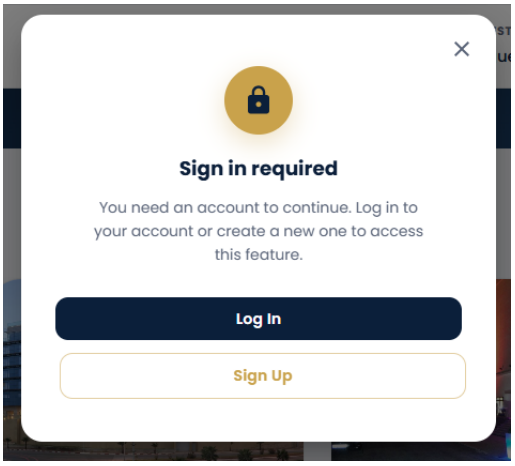


Figure 4.21: *The Authentication Guard Dialog*

This dialog informs the visitor that an account is required in order to continue and offers a direct route to either the Log In or the Sign Up page. Its role is twofold: it prevents protected operations from being carried out without an authenticated identity, and it guides the visitor toward authentication at the precise moment the need arises, so that the visitor can sign in or register and then return to complete the intended action. It is in this way the bridge between the freely accessible browsing of the Home page and the operations that require an account.

5.1.3. Login

The Login page, shown in Figure 4.49, authenticates a visitor who already has an account. It presents a field for the username and a field for the password, the latter with a control to reveal or hide its contents, together with the Log In action. The page also offers two secondary routes: a link to recover a forgotten password, and a link to the registration page for visitors who do not yet have an account. On successful authentication the user gains access to the operations reserved for their role.

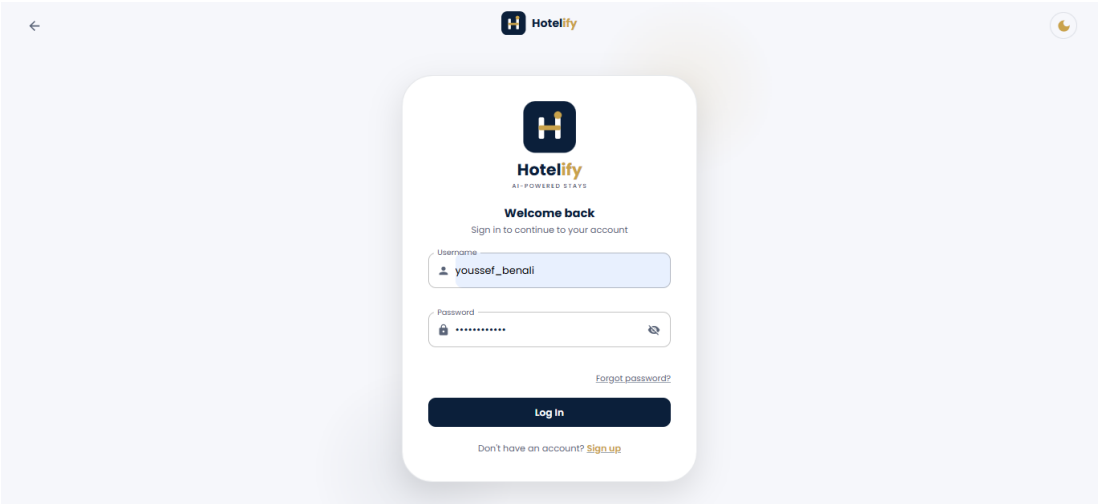


Figure 4.22: *The Login Page*

When authentication fails, for instance because the username or the password is incorrect, the system reports the failure through a Snackbar, as shown in Figure 4.50. A Snackbar is a brief, unobtrusive message that appears at the edge of the screen to inform the user of the outcome of an action and then disappears of its own accord. The application uses Snackbars consistently throughout the interface: a Snackbar in a warning colour signals that an operation has failed or that an error has occurred, while a Snackbar in a success colour confirms that an operation has completed successfully. Because this feedback mechanism is shared across the whole application, it is described here once, and later occurrences are referred to only briefly.

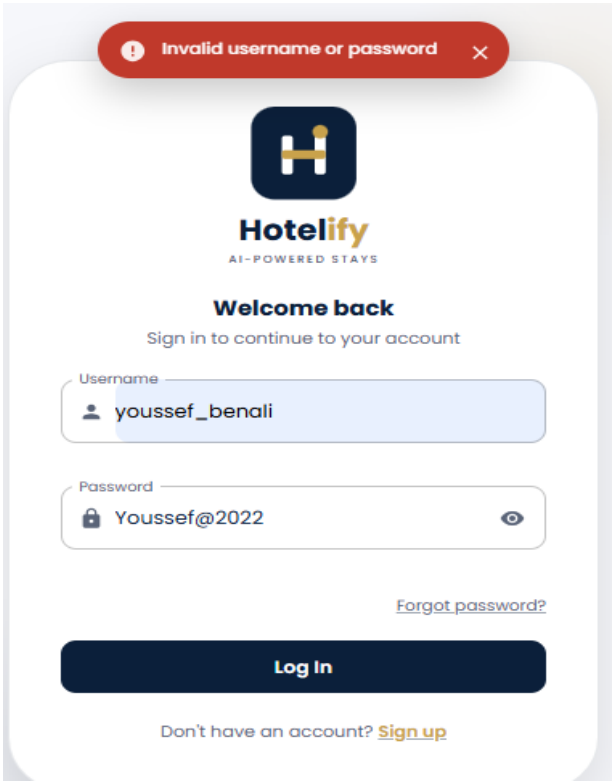


Figure 4.23: *A Failed Login Reported Through an Error Snackbar*

5.1.4. Account Registration

A visitor without an account creates one through the Sign Up page, shown in Figure 4.51. To keep the registration clear despite the amount of information required, the page organises its fields into labelled groups. A first group gathers the personal information of the guest, namely the first and last name, the email address, and the phone number. A second group collects the account credentials, namely the chosen username, the password, and its confirmation, with controls to reveal or hide the entered passwords. A final group records further details of the guest, namely the date of birth, the country, and the city. Once these are provided, the Sign Up action creates the account, after which the guest may sign in; a link to the Login page is also offered for visitors who in fact already have an account.

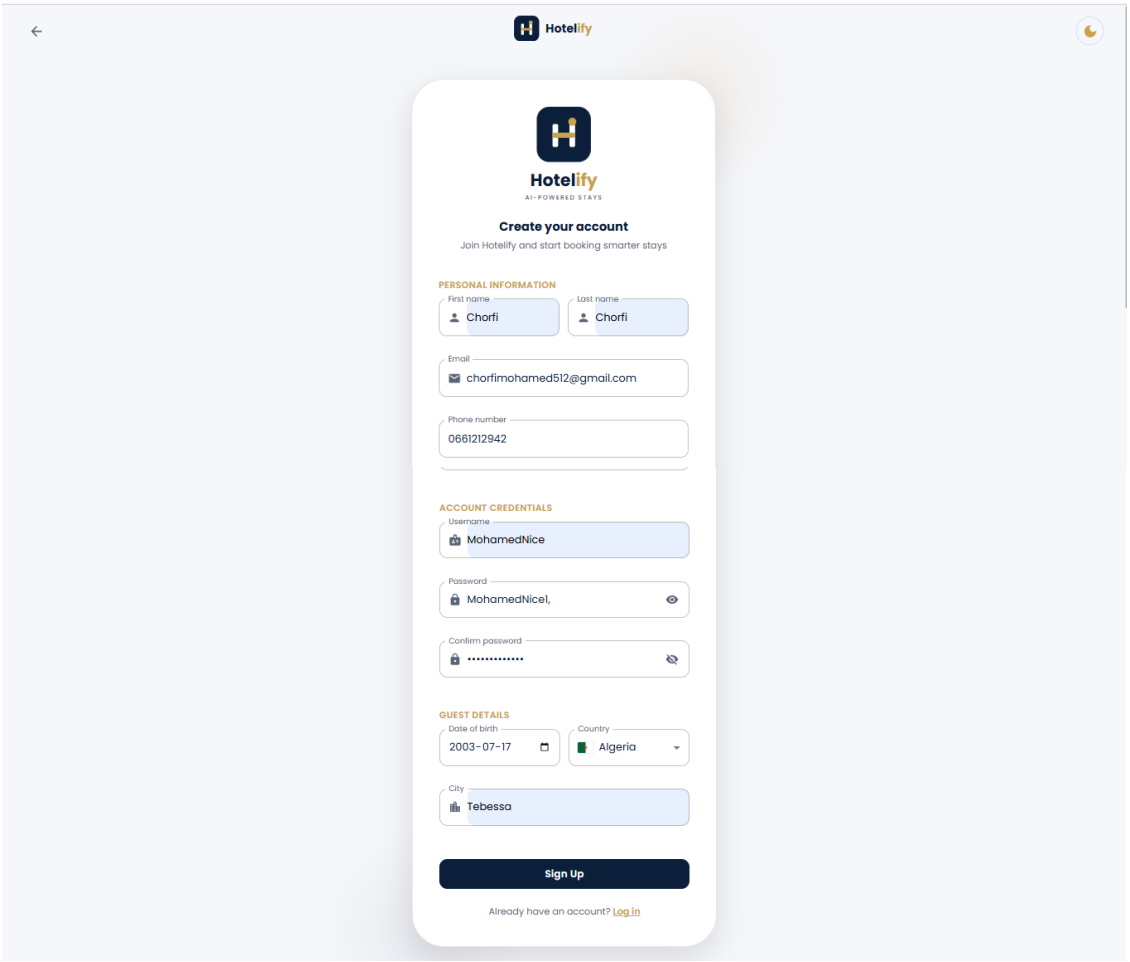


Figure 4.24: *The Account Registration Page*

5.1.5. Password Recovery

A user who has forgotten their password can recover access through a dedicated, multi-step operation that is reached from the Login page. The first step, shown in Figure 4.52, asks the user to enter the email address associated with the account so that a reset link can be sent to it.

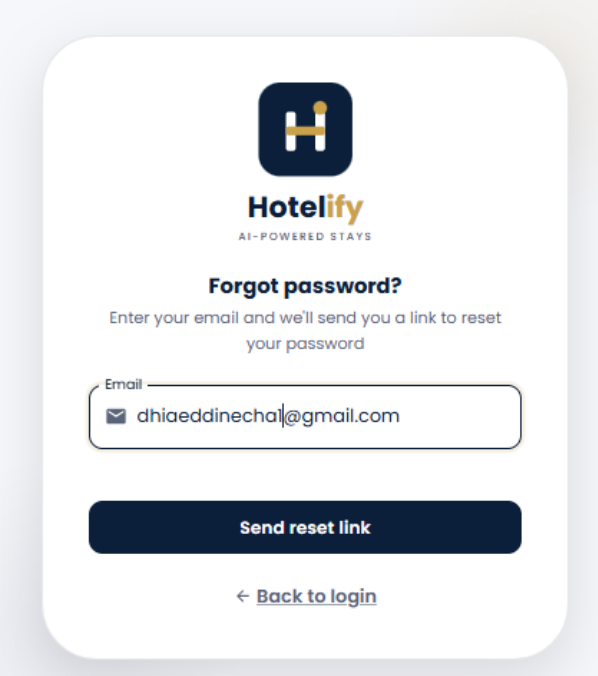


Figure 4.25: *Requesting a Password Reset by Email*

The system then displays the confirmation shown in Figure 4.53. The wording of this message is deliberately neutral, stating that a reset link has been sent if an account exists for the given address, so that it confirms the request without revealing whether any account is in fact registered under that address, a precaution that protects the privacy of the platform's users.

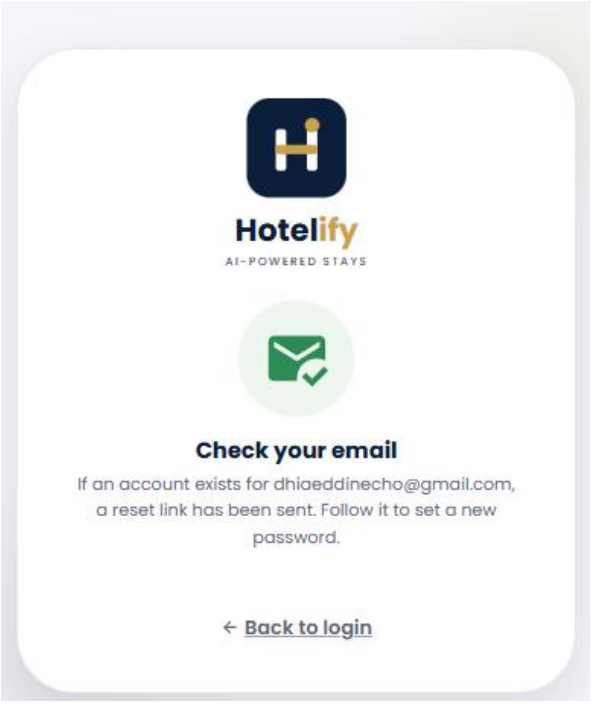


Figure 4.26: *Confirmation that a Reset Link Has Been Sent*

If an account does exist, the user receives the email shown in Figure 4.54, which contains the reset action and states clearly that the link remains valid for only 15 minutes and that the message may be ignored if the reset was not requested. This time limit, together with the cautionary note, reflects the secure password reset flow implemented in the backend.

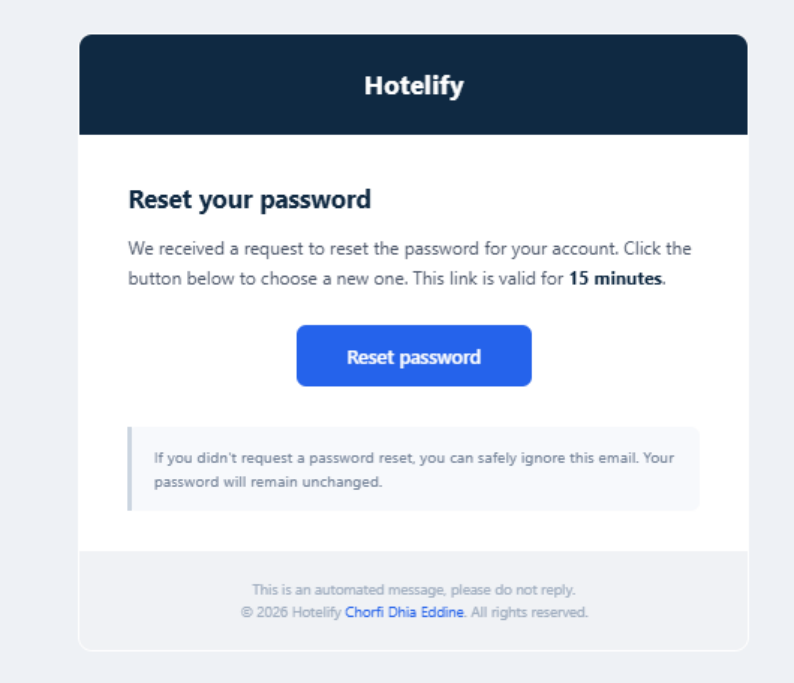


Figure 4.27: *The Password Reset Email*

Following the link leads to the page shown in Figure 4.55, where the user enters and confirms a new password. Once the reset is submitted, the system presents the success page shown in Figure 4.56, which confirms that the password has been updated and invites the user to sign in with the new credentials.

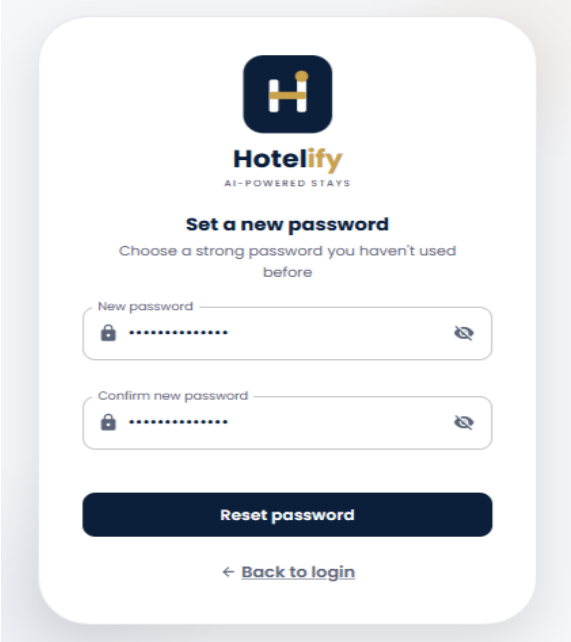


Figure 4.28: *Setting A New Password*

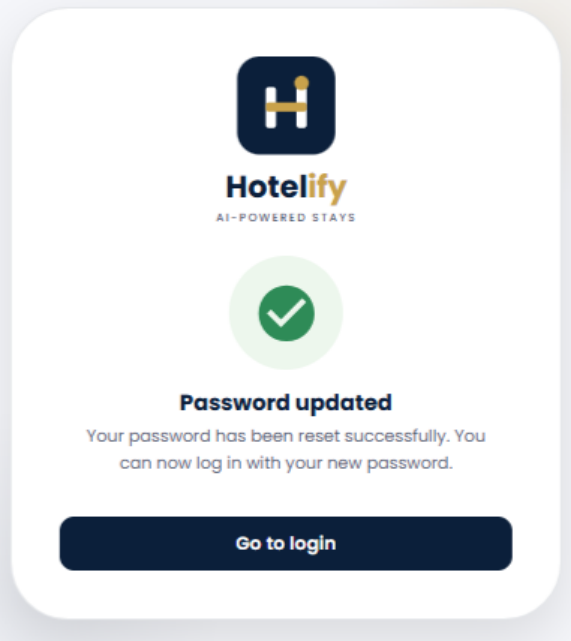


Figure 4.29: *Confirmation That the Password Has Been Reset Successfully*

5.1.6. Access Code

The authentication pages described so far are common to every visitor. One further authentication step, however, applies only to the two privileged roles, namely the hotel owner and the administrator. After such a user has signed in with a username and password, the system requires a second factor before the session is granted, presenting the access code page shown in Figure 4.57. On this page the user enters the access code that completes the sign-in, after which full access is granted. This page realises the two-factor authentication of the design, in which an account of a privileged role is not fully

authenticated by its password alone but only once the access code has also been verified. It is the final page of the authentication flow, and with it the pages seen by an unauthenticated user are complete.

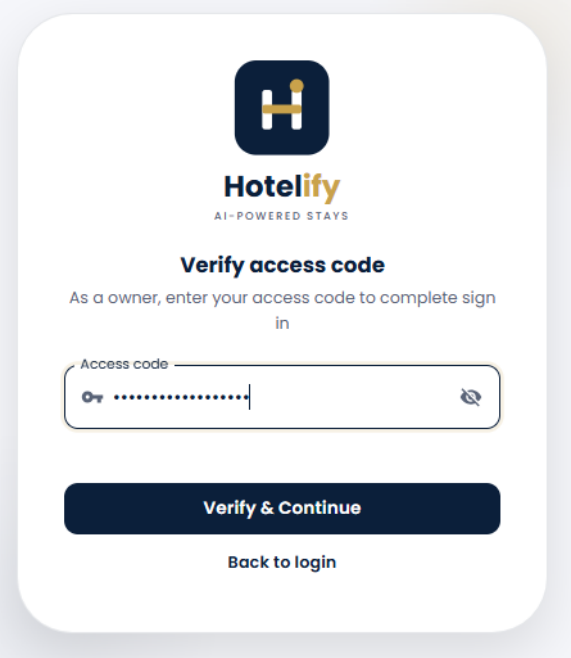


Figure 4.30: *The Access Code Page*

5.2. Guest Interface

This subsection presents the operations available to a guest once signed in, the role through which the platform is actually used to find and book a stay. Whereas the preceding subsection covered the pages seen before authentication, the pages described here all assume an authenticated guest. They are presented operation by operation, following the path a guest naturally takes: returning to a personalised Home page, searching for hotels, examining a hotel in detail, booking a stay, and then managing the reservations, the account, the reviews, and the favourites that the account accumulates.

5.2.1. The Registered Guest's Home Page

Once the guest has signed in, the Home page retains the search bar and the country-grouped hotel listings described earlier, but its header gains the features of an authenticated session, as shown in Figure 4.58. A link to the guest's reservations now appears, and a profile menu, opened from the account avatar, gives access to the guest's account, reviews, and favourites, as well as to the action for signing out. Moreover, the hotels that the guest has added to the favourites are now marked with a filled heart, so that previously saved preferences are immediately visible.

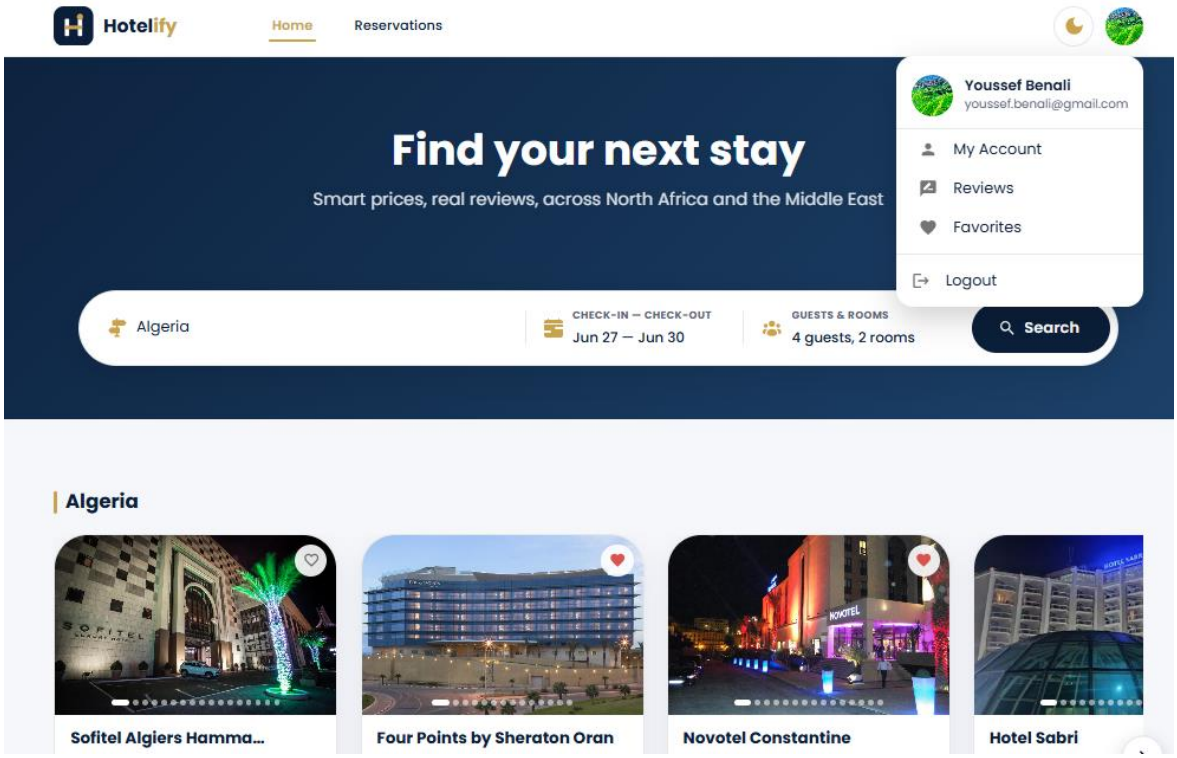


Figure 4.31: *The Home Page as Seen by a Signed-In Guest*

5.2.2. Searching for Hotels

When the guest submits a search from the Home page, the system presents the results page shown in Figure 4.59. The search bar remains at the top so that the criteria can be refined, and beneath it the hotels matching the chosen destination and dates are listed, introduced by a count of the available hotels and grouped by country in the same card format as on the Home page. From any of these cards the guest opens the hotel to view its details.

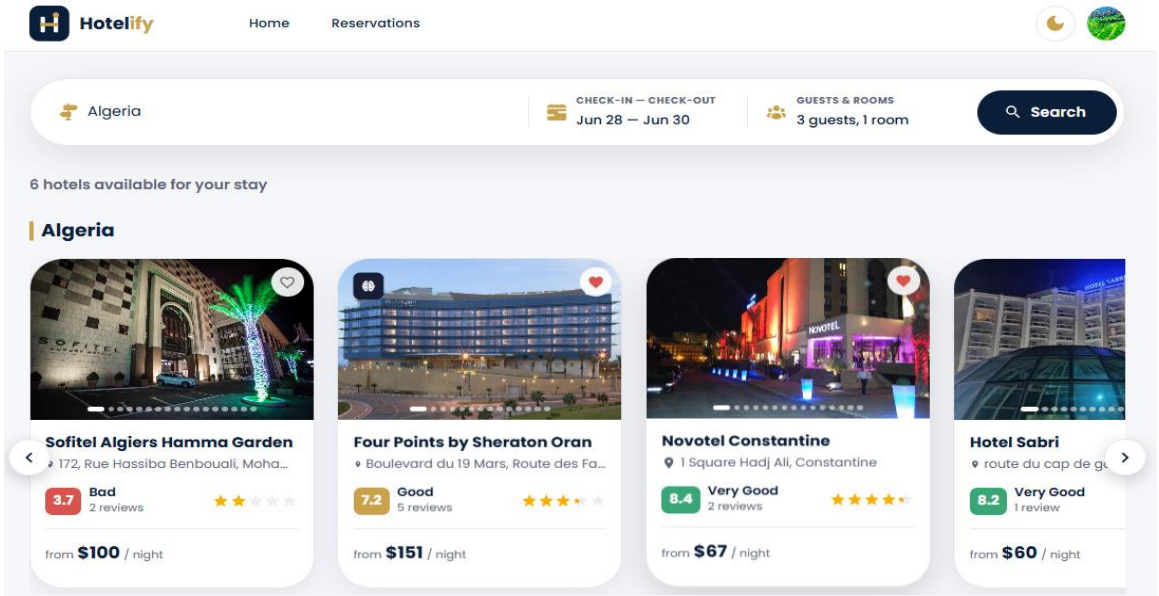


Figure 4.32: *The Results of a Hotel Search*

5.2.3. Viewing Hotel Details

Opening a hotel leads to its details page, which gathers everything the guest needs in order to evaluate the hotel and choose a room. The page begins, as shown in Figure 4.60, with a gallery of photographs, the hotel name, its location, and a control for adding it to the favourites, followed by a short description under "About this hotel" and a list of the hotel facilities such as the internet access, the restaurant, the pool, and the front-desk service.

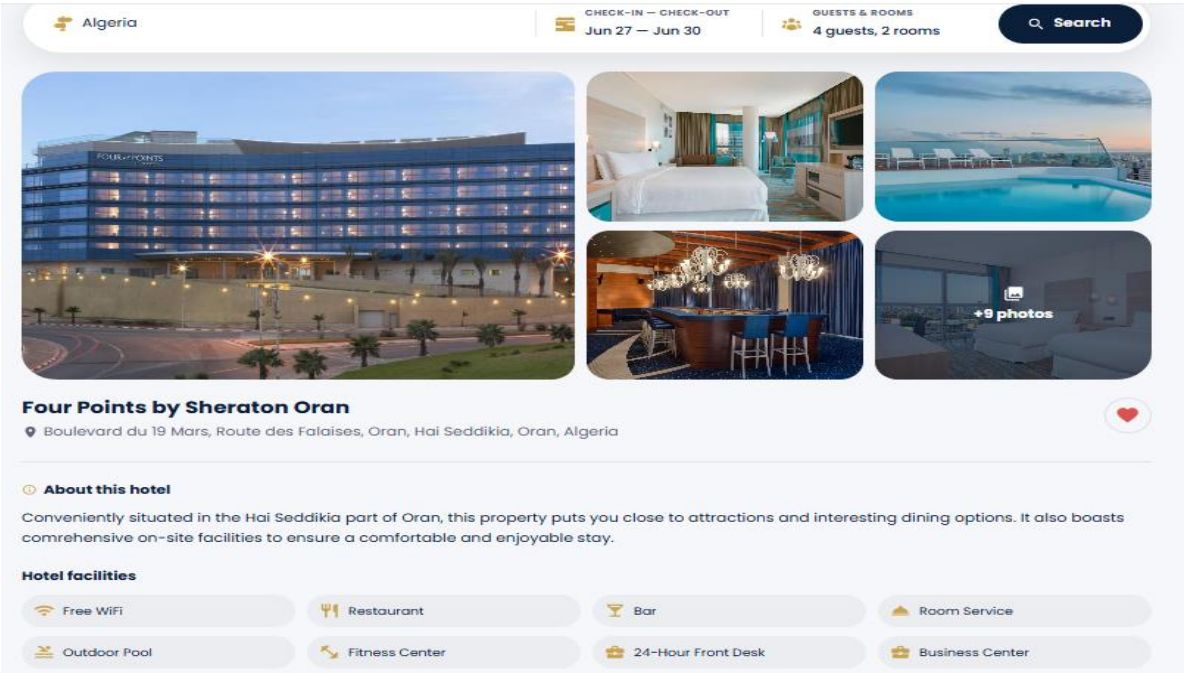


Figure 4.33: *The Top of the Hotel Details Page*

Selecting the gallery opens the full-screen viewer shown in Figure 4.61, in which the guest can browse all of the hotel's photographs in turn, using the navigation controls and the thumbnail strip, with the current position indicated within the total number of images.

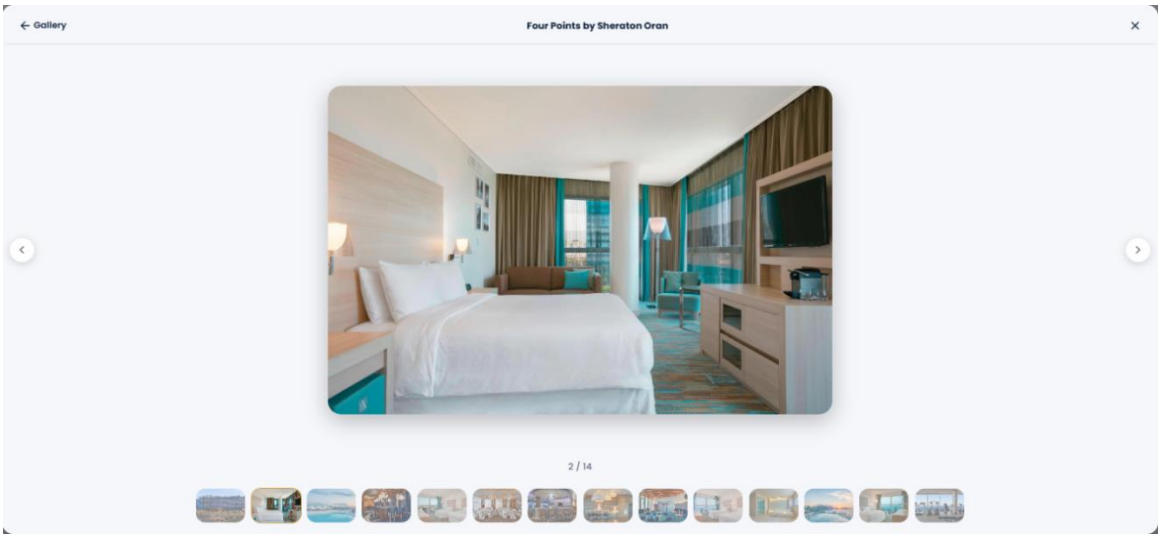


Figure 4.34: *The Full-Screen Photo Gallery of a Hotel*

The heart of the page is the list of available rooms, shown in Figure 4.62. Each room is presented with its bed configuration and capacity, its amenities, the number of rooms still available, its nightly price, and a "Book now" action. Most importantly, each room carries an AI forecast: the price predicted for the selected check-in date, accompanied by the percentage change and an arrow indicating whether the price is expected to rise or to fall. This forecast is the output of the price prediction and recommendation model presented directly to the guest, and Figure 4.28 shows it both for a room whose price is forecast to rise and for one whose price is forecast to fall.

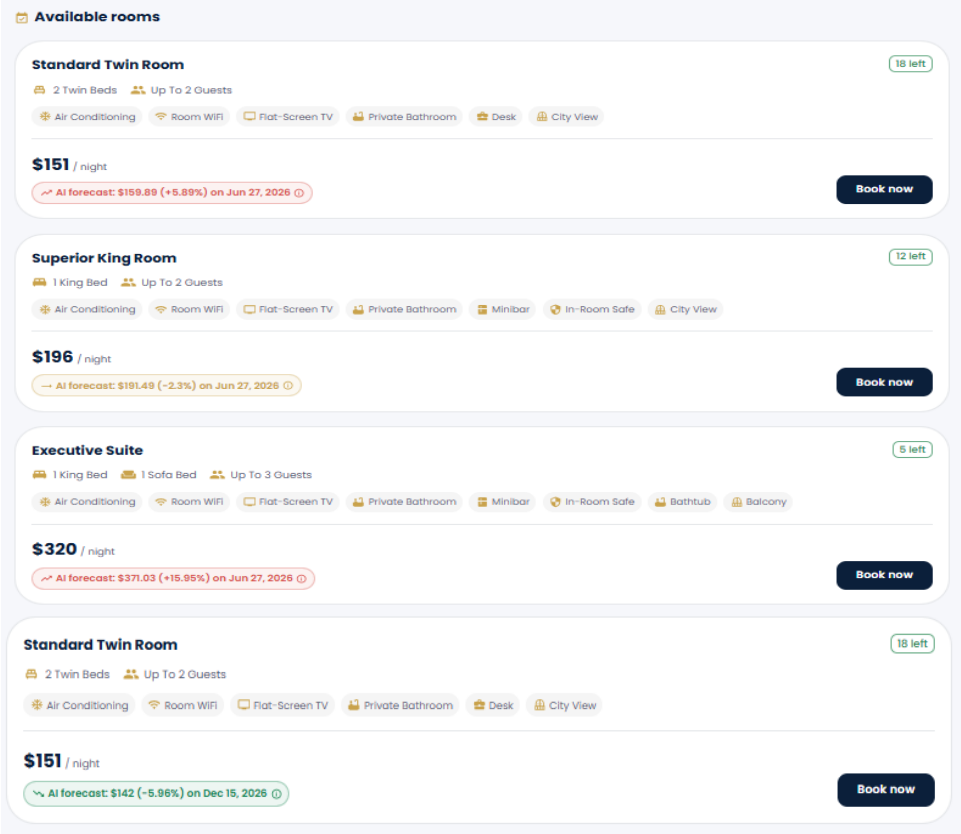


Figure 4.35: *The Available Rooms with the AI Price Forecast (Prediction) for each Room Type*

The forecast can be expanded to reveal the reasons behind it, as shown in Figure 4.63. These reasons form a short, readable explanation of why the price is expected to move as it does, citing factors such as the recent average price of the room, how far in advance the booking is being made, the category of the room, the current availability, and the pace at which the room is being booked. They are precisely the explanatory output of the model and allow the guest to understand, rather than merely receive, the recommendation. Figure 4.63 presents these explanations across different scenarios, including both an expected increase and an expected decrease in price.

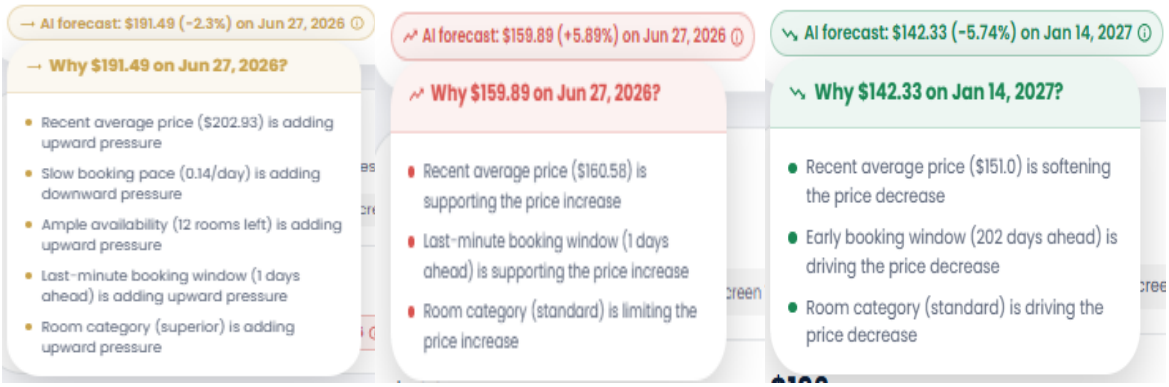


Figure 4.36: *The Explanation of an AI Price Forecast (Prediction), Shown for Increasing And Decreasing Prices*

Below the rooms, the page summarises the opinions of past guests. Figure 4.64 shows the ratings overview, with an overall score and a qualitative label, the number of reviews, and a breakdown into categories such as the staff, the facilities, the cleanliness, the comfort, the value for money, and the location.



Figure 4.37: *The Ratings Overview of a Hotel*

The individual reviews follow, under "What guests are saying", as shown in Figure 4.65. Each review displays its author and origin, the room and the dates of the stay, a star rating and an overall score, and the text of the review. Alongside each review the system shows a sentiment label, namely positive, neutral, or negative, together with a confidence percentage; this label is the output of the sentiment analysis model applied to the text of the review. Figure 4.65 includes positive, neutral, and negative examples.

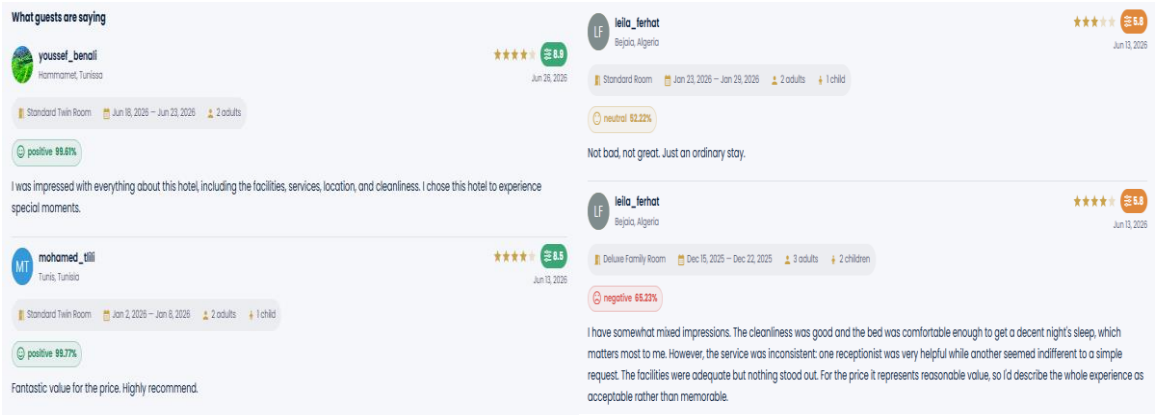


Figure 4.38: *Guest Reviews, each Annotated with the Analysed Sentiment*

The sentiment label can itself be expanded, as shown in Figure 4.66, to reveal the full breakdown of the analysis across the three classes, giving the proportion attributed to the positive, the neutral, and the negative sentiment. In this way the two AI capabilities of the platform, the prediction of price and the analysis of sentiment, are both made visible and actionable on the very page where the guest decides whether and what to book.

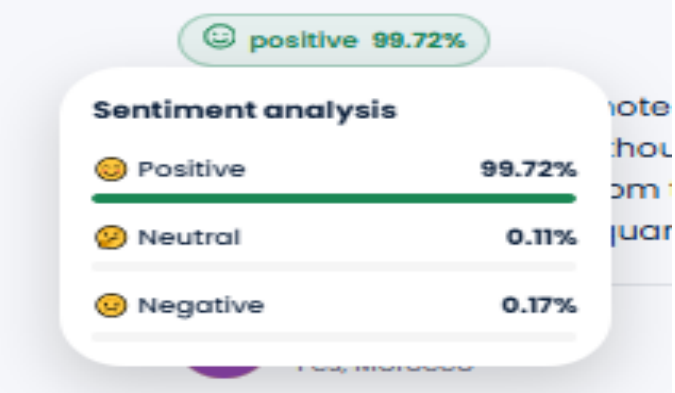


Figure 4.39: *The Detailed Sentiment Breakdown of a Review*

5.2.4. Booking a Stay

When the guest chooses to book a room, the system guides them through a three-step process, with a booking summary, listing the hotel, the room, the dates, the number of nights, rooms, and guests, and the total price, shown alongside at every step so that the cost and the details of the booking remain in view throughout.

The first step, shown in Figure 4.67, collects the guest's information. The personal details are pre-filled from the account, sparing the guest from re-entering them, and a single option allows the guest to indicate whether the trip is for business before continuing to the payment step.

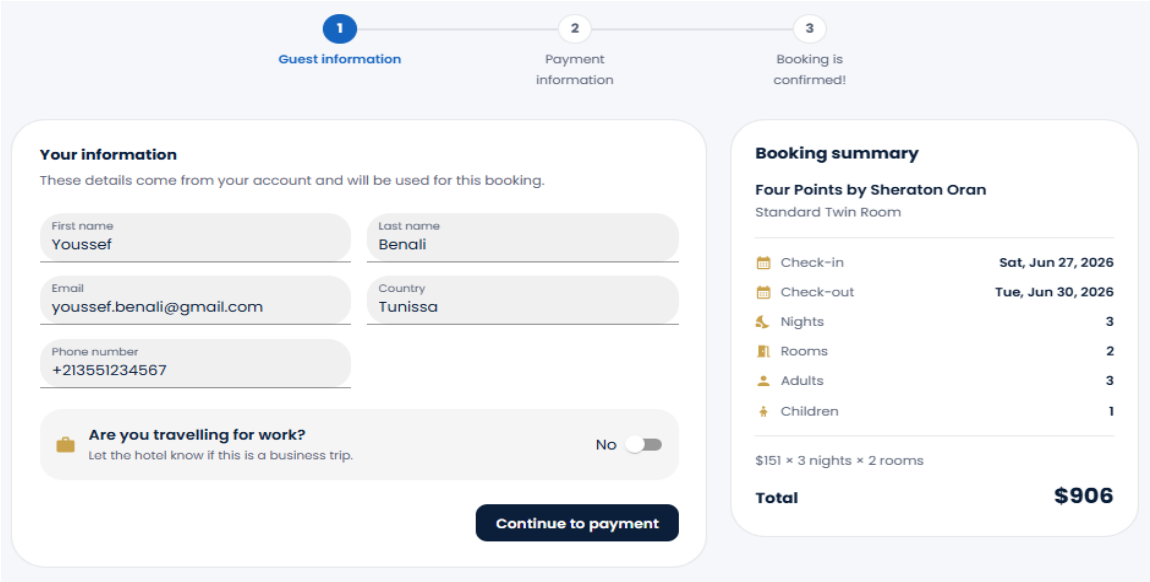


Figure 4.40: *The First Step of Booking, Collecting the Guest Information*

The second step, shown in Figure 4.68, concerns the payment. The guest selects how to pay, either by card or at the property, and confirms the payment details. As the page itself states, this is a demonstration checkout in which no real card is charged, reflecting the prototype nature of the system.

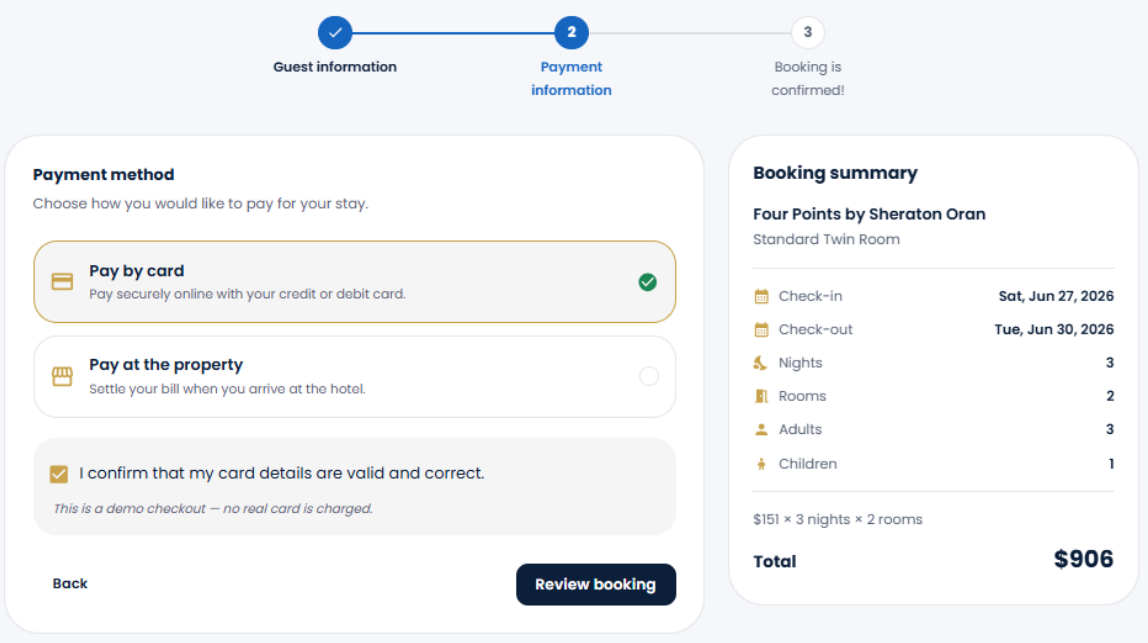


Figure 4.41: *The Second Step of Booking, Selecting the Payment Method*

The third step, shown in Figure 4.69, is a final review before the booking is committed. Here the guest is offered two distinct choices. Confirming the booking immediately marks the reservation as confirmed and its payment as paid, whereas saving it for later holds the reservation in a pending state to be confirmed afterwards.

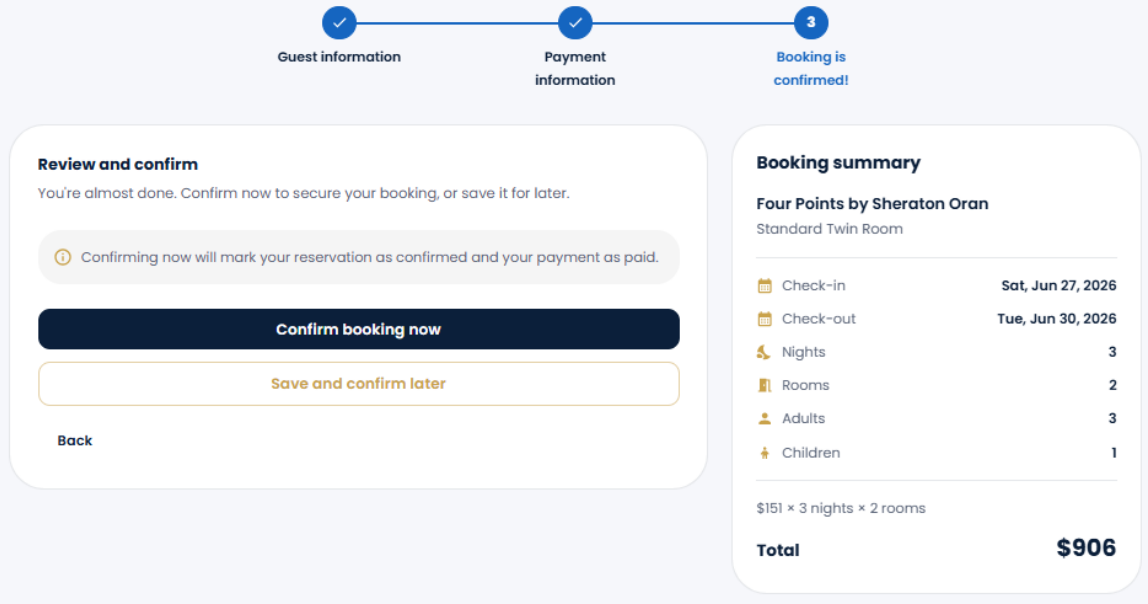


Figure 4.42: *The Third Step of Booking, the Final Review with the Option to Confirm Now or Later*

These two choices lead to two corresponding outcomes. If the guest confirms immediately, the system shows the confirmation in Figure 4.70, stating that the booking is confirmed, that the payment has been received, and that the reservation has been saved. If instead the guest chooses to confirm later, the system shows the outcome in Figure 4.71, stating that the booking is reserved with a pending status and an unpaid payment, and that it may be confirmed afterwards from the guest's reservations. These two states, confirmed and pending, mirror the reservation lifecycle defined in the design.

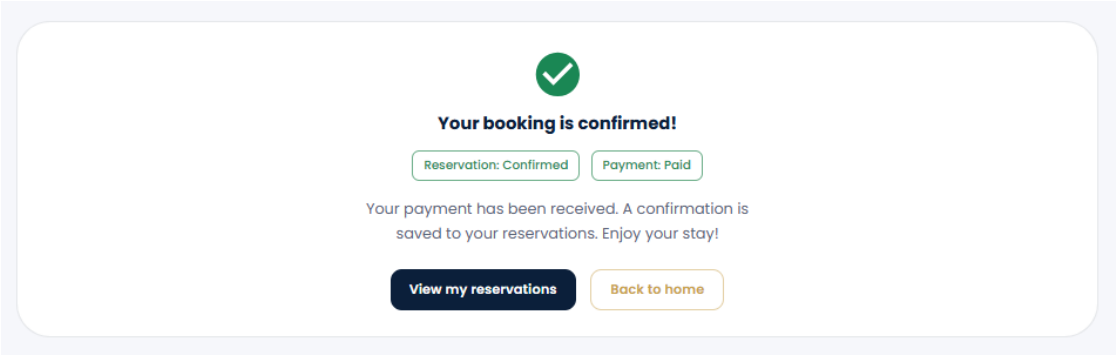


Figure 4.43: *The Outcome when the Booking is Confirmed Immediately*

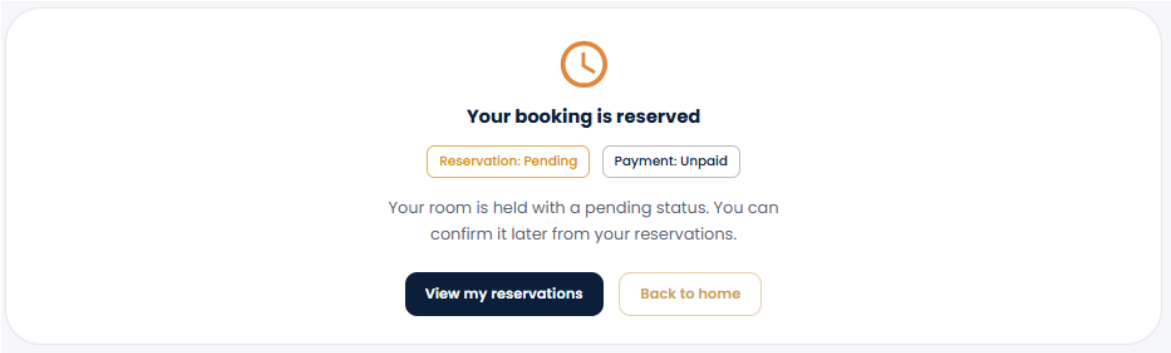


Figure 4.44: *The Outcome when the Booking is Saved to be Confirmed Later*

5.2.5. Managing Reservations

Once a guest has made bookings, the reservations page becomes the place where those bookings are managed. As shown in Figure 4.72, the page organises the guest's reservations by their status, offering a set of tabs that separate all reservations from those that are pending, confirmed, checked in, checked out, cancelled, or marked as a no-show, with the number of reservations in each state shown beside its tab. Each reservation is presented as a card carrying the hotel and room, the dates and the party size, the total price, and, for an upcoming stay, a live countdown to the moment of check-in.

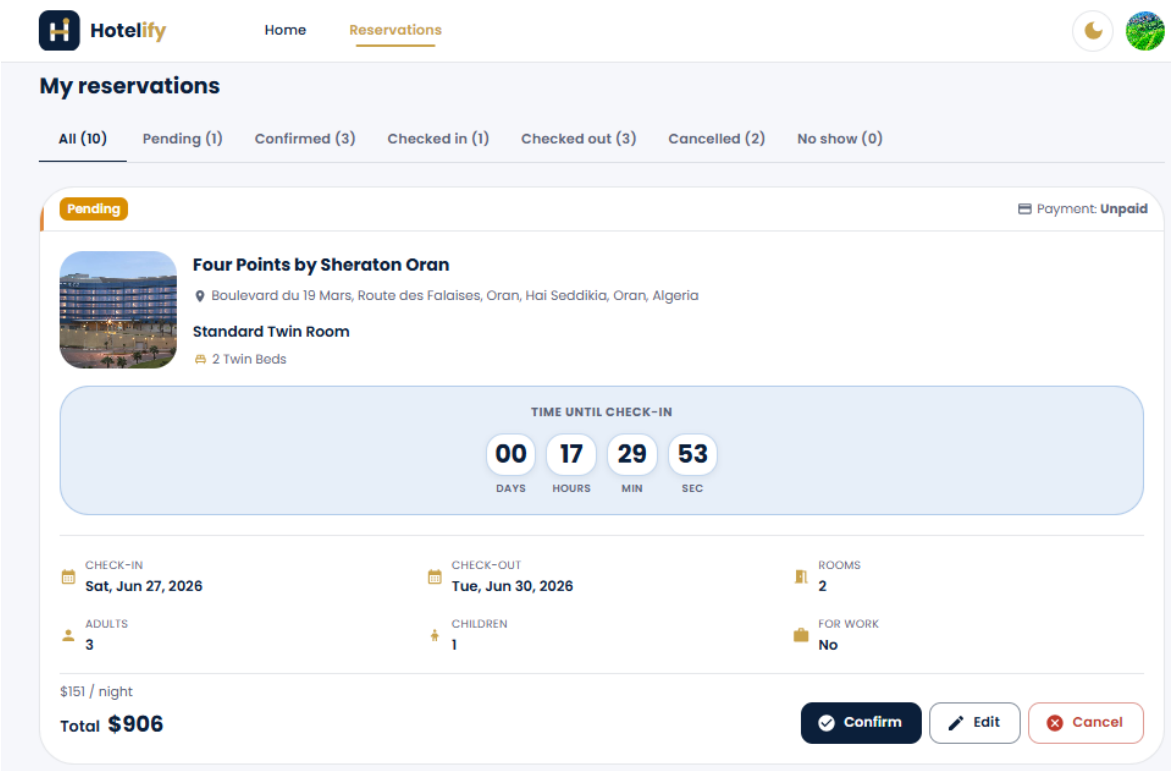


Figure 4.45: *The Reservations Page, with the Status tabs and a Pending Reservation*

The actions offered on a reservation depend on its state. A pending reservation, such as the one in Figure 4.72, is held but not yet paid, and the guest may confirm it, edit it, or cancel it. Confirming a pending reservation settles its payment, upon which a success Snackbar reports that the payment has been recorded and the reservation moves to the confirmed state, as shown in Figure 4.73. Editing a reservation opens the dialog shown in Figure 4.74, in which the guest may adjust the dates, the number of adults, children, and rooms, and whether the trip is for business, before saving the changes. Cancelling a reservation, being an action that cannot be undone, first asks the guest to confirm the cancellation.

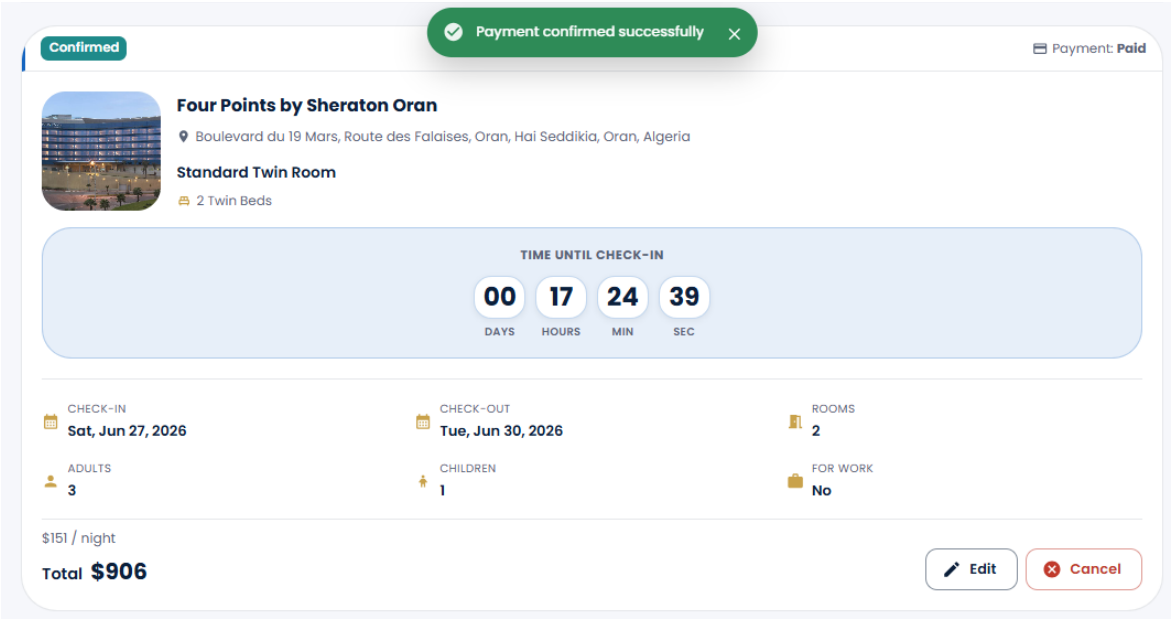


Figure 4.46: *A Confirmed Reservation*

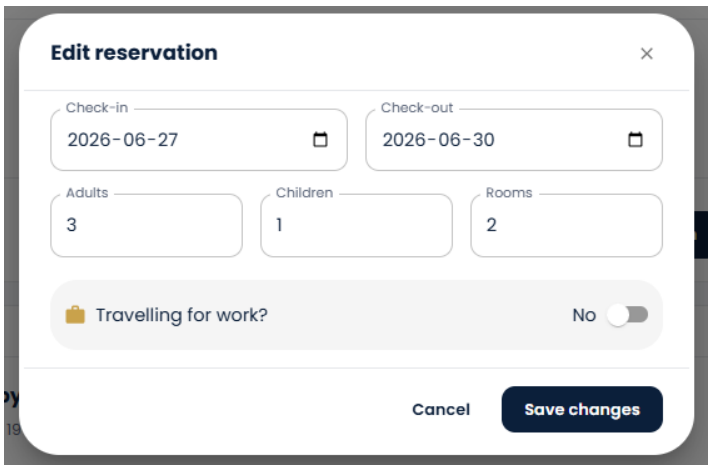


Figure 4.47: *The Dialog for Editing a Reservation*

A confirmed reservation may still be edited or cancelled while the stay is upcoming. Once the stay begins, the reservation becomes checked in, and its card then counts down to the moment of check-out rather than check-in; once the stay ends, it becomes checked out, and the guest is invited to write a review of the hotel. Both of these states are shown in Figure 4.75.

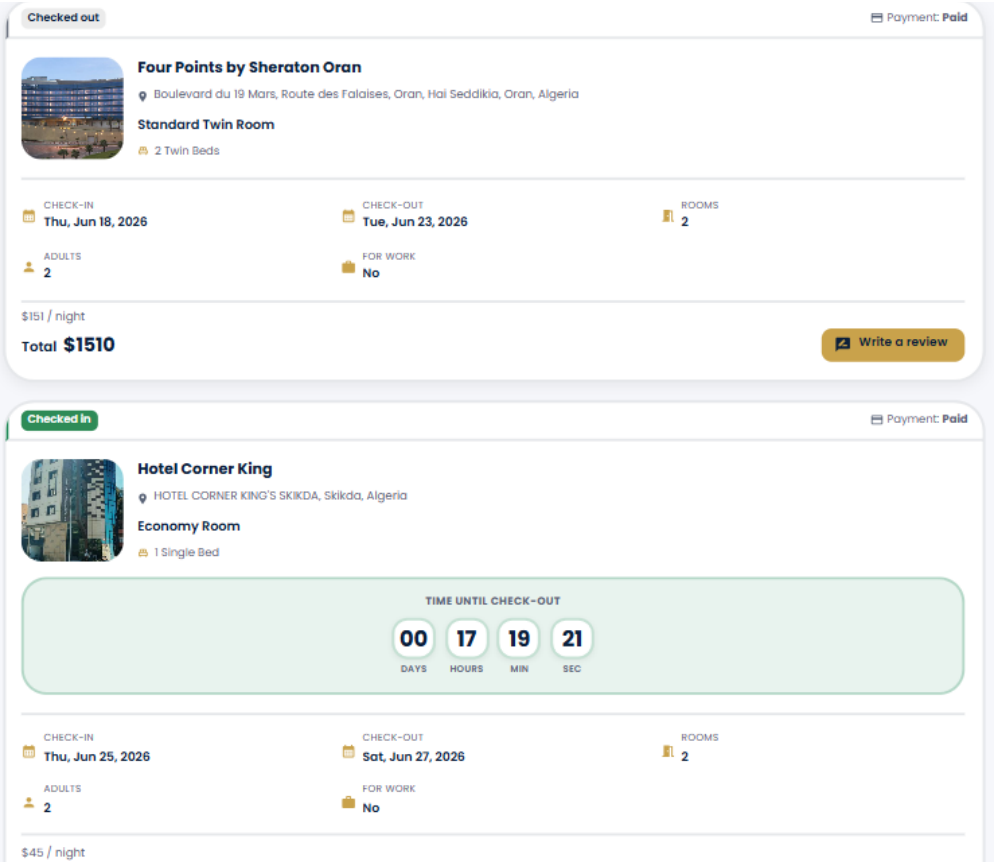


Figure 4.48:*A Checked-Out Reservation, Offering to Write a Review, and a Checked-In Reservation, Counting Down to Check-Out*

Two further states are not illustrated here but complete the lifecycle. A cancelled reservation, whether cancelled by the guest, and a no-show, recorded when a confirmed guest does not arrive, each appear under their respective tab with the corresponding status, and are kept without further actions. Presenting them in this way gives the guest a complete and durable history of every reservation, whatever its eventual fate, and not merely of the active ones.

The importance of these reservations extends well beyond the individual guest, for they are also the raw material on which the intelligence of the platform feeds. As established in the design of the price prediction model, that model does not learn from prices alone but from the state of demand, expressed through such quantities as the number of bookings and cancellations, the occupancy rate, the pace at which a room is being booked, the time remaining until it is fully booked, and the number of rooms still available. Every one of these quantities is computed from the reservations made against a hotel's rooms. Each booking a guest makes, and each cancellation, therefore alters the demand signals of the hotel concerned, and through them the prices that the model forecasts for its rooms in the daily prediction. In this sense a reservation is not an isolated transaction but an input to the system as a whole: it shapes the availability of the hotel and, by way of the demand it represents, the very recommendations that future guests will see.

5.2.6. My Account

The account page allows the guest to view and manage their personal information. As shown in Figure 4.76, it presents the guest's profile, namely the avatar, the name, and the username, together with the recorded details such as the first and last name, the email address, the phone number, the country, the city, and the date of birth, and it offers the action to edit them.

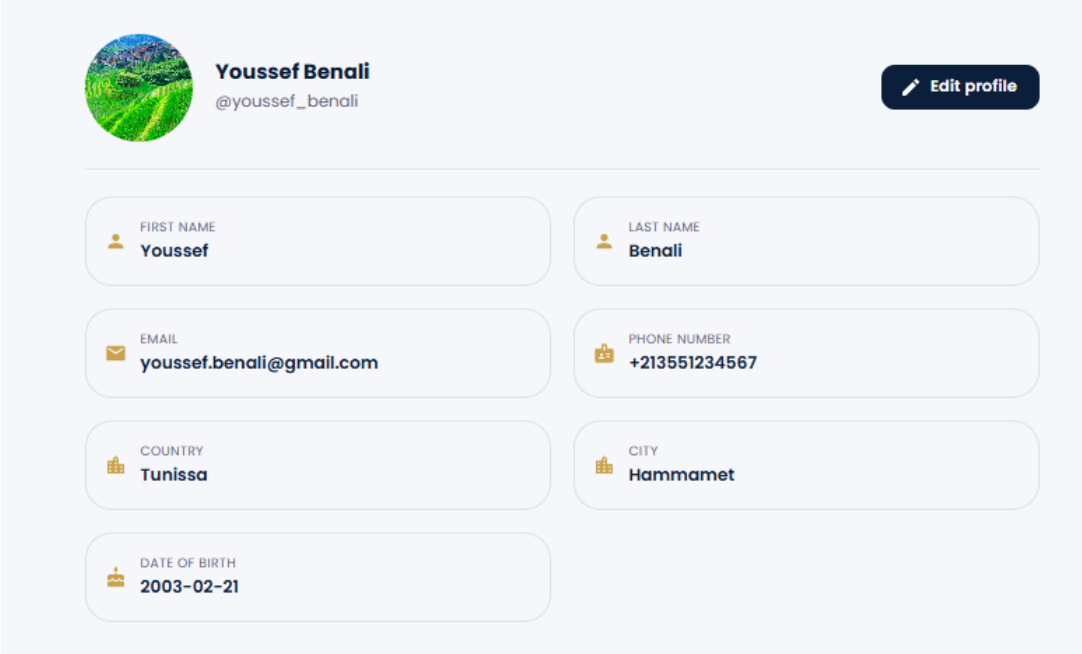


Figure 4.49: *The Account Page of the Guest*

Choosing to edit opens the form shown in Figure 4.77, in which the same details become editable and the profile photograph may be changed. Once the guest saves the changes, the updated information replaces the former values, with a Snackbar confirming the success of the operation.

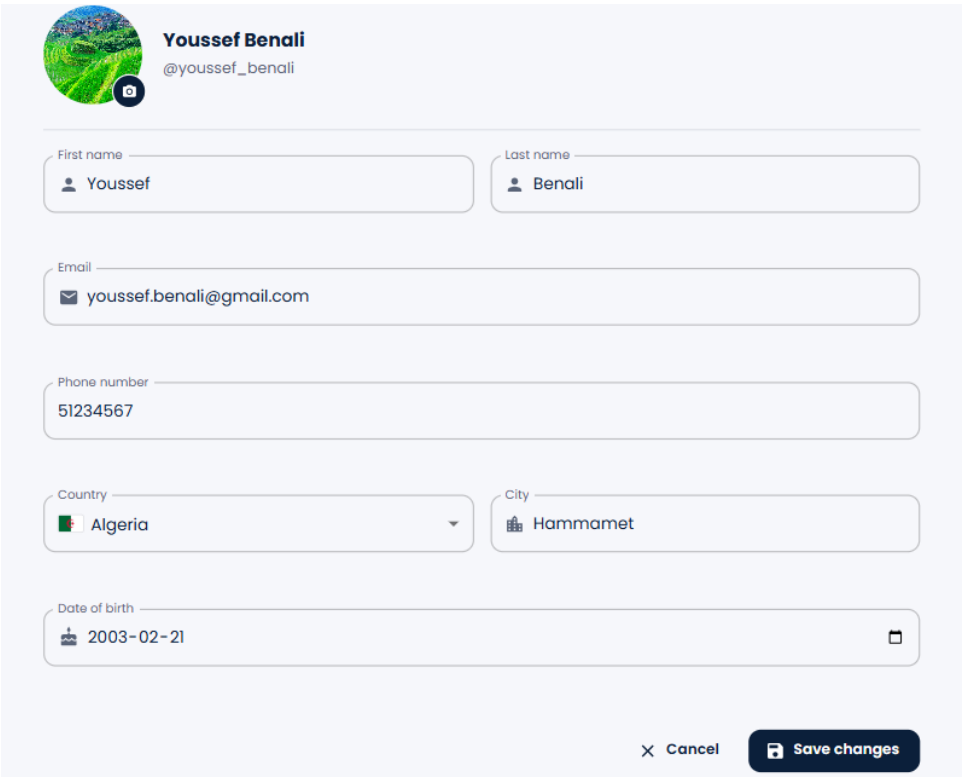


Figure 4.50: *The Form for Editing the Account Details*

5.2.7. Managing Reviews

Just as the reservations page gathers a guest's bookings, the reviews page gathers the reviews that the guest has written, as shown in Figure 4.78. Each entry shows the hotel and room concerned, the text of the review, and the analysed sentiment of that text, presented in the same form as on the hotel page, alongside the actions to edit or to delete the review.

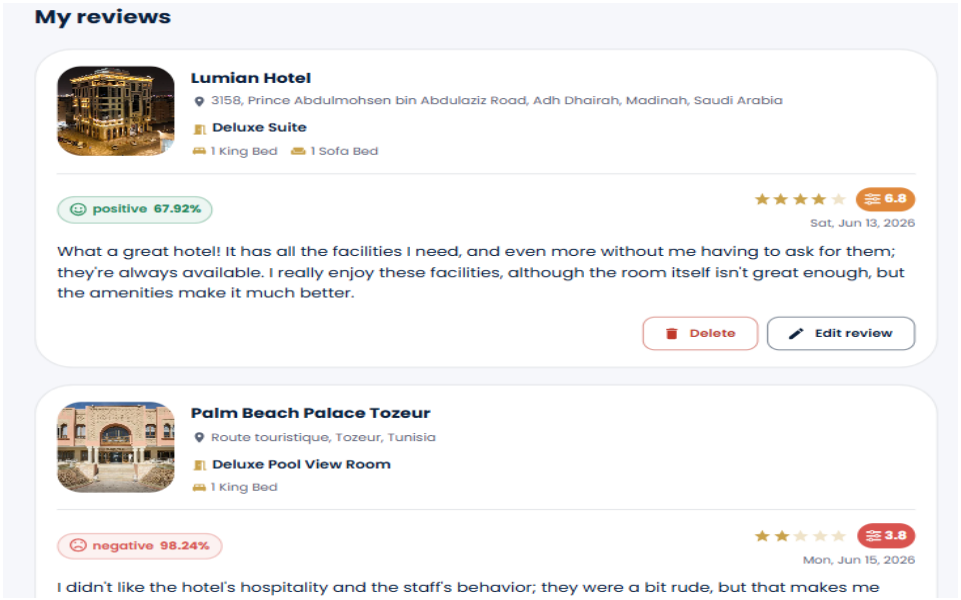


Figure 4.51: *The Page on Which the Guest Manages their Own Reviews*

Editing a review opens the dialog shown in Figure 4.79, where the guest may revise the overall star rating, the written comment, and the rating given to each category, such as the staff, the facilities, the cleanliness, the comfort, the value for money, and the location, each scored on a scale out of ten.

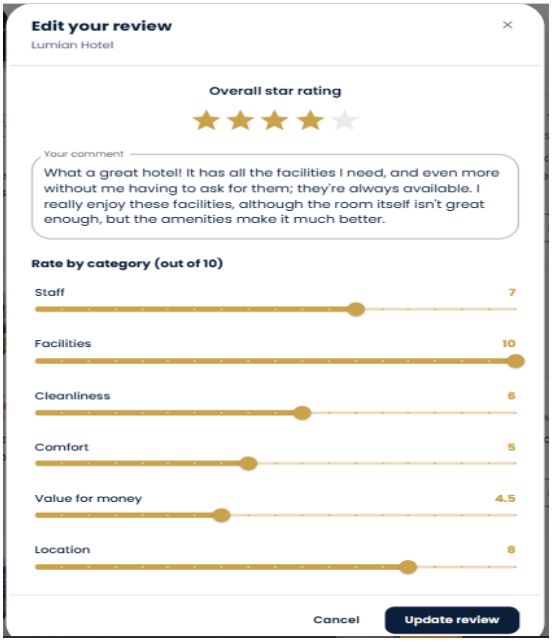


Figure 4.52: *The Dialog for Editing a Review*

Deleting a review, by contrast, is an irreversible action, and the system therefore guards it with the confirmation dialog shown in Figure 4.80. This dialog states plainly which review is about to be deleted and warns that the action cannot be undone, and it proceeds only once the guest explicitly confirms. Such confirmation dialogs are the standard safeguard the application places before any destructive action, including the cancellation of a reservation discussed earlier; having been described here, they are referred to only briefly when they recur.

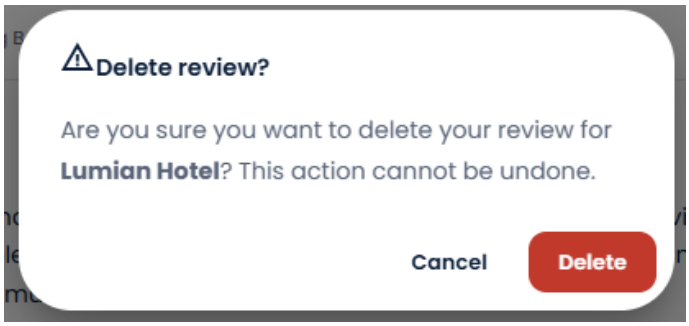


Figure 4.53: *The Confirmation Dialog Shown before a Review is Deleted*

5.2.8. Favorites

Finally, the favourites page collects the hotels that the guest has saved by means of the heart control encountered throughout the interface. As shown in Figure 4.81, these hotels are presented as the same cards used elsewhere, each marked with a filled heart,

so that the guest may return to a hotel of interest at any time or remove it from the list. This page completes the personal space of the registered guest.

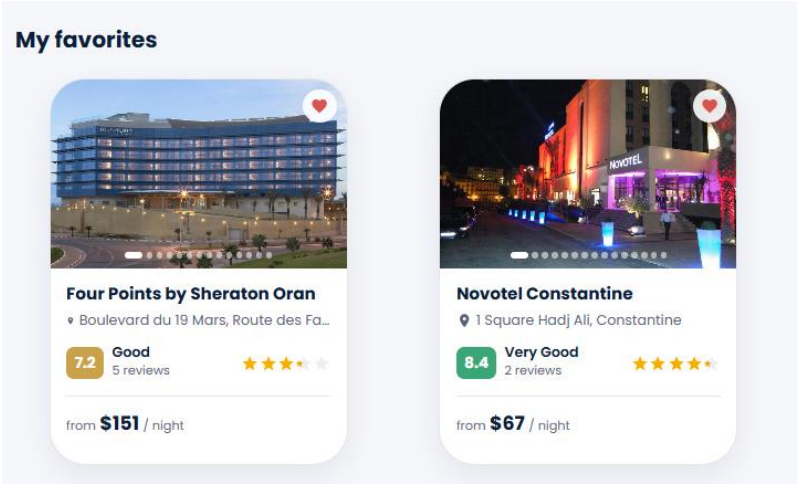


Figure 4.54: *The Favourites Page of the Guest*

With the favourites page, the operations available to the guest are complete. The presentation now turns from the guest to the two privileged roles, beginning with the dashboard of the hotel owner.

5.3. Owner Dashboard

The preceding subsection presented the platform from the standpoint of the guest, the user for whom the system was primarily intended. The present subsection turns to the hotel owner, the actor responsible for a single property within the platform. As established in Chapter 3, every owner account is provisioned by the administrator and is bound to exactly one hotel, and access to the owner's workspace is protected by the two-factor mechanism described in Section 5.1, whereby the owner confirms their identity with a dedicated access code before the dashboard is opened. Once authenticated, the owner is presented not with the public, exploratory interface of the guest but with a dedicated management dashboard, whose design deliberately favours information density and analytical clarity, in keeping with the interface design principles set out in Chapter 3.

The dashboard is organised around a persistent navigation sidebar that gives access to its three areas: the Home page, which concentrates the analytics of the hotel; the Hotel page, through which the owner manages the property itself and its room types; and the Reservations page, through which the owner oversees bookings. A top bar, common to the whole dashboard, provides a shortcut to the public site, a light and dark theme toggle, and the owner's account menu. This subsection presents the Home and Hotel pages in turn, organised, as before, around the operations that the owner actually performs.

5.3.1. Home Page

The Home page is the analytical heart of the owner dashboard. It draws together, in a single scrollable view, the indicators, charts, and records that describe the commercial and operational health of the hotel, so that the owner can assess performance at a glance and identify where attention is required. A period selector at the top of the page, offering windows of 7 days, 30 days, 90 days, and 12 months, governs the entire page, recomputing every indicator and chart for the chosen interval and thereby allowing the same dashboard to serve both short-term operational monitoring and longer-term strategic review.

At the top of the page sits a band of KPI cards that condense the hotel's standing into a set of headline figures, shown in Figure 4.82. The first group reports the financial and occupancy fundamentals: the total revenue generated over the selected period, an estimated profit derived from that revenue after accounting for an approximate operating cost of 30 per cent, the occupancy rate, the average daily rate (ADR), and the revenue per available room (RevPAR), the last two being the standard yield measures of the hospitality industry. Alongside these, operational counters report the total number of bookings, distinguishing active from cancelled, together with the check-ins and check-outs due on the current day, which support the owner's immediate front-desk planning. A second group complements these with behavioural and quality measures: the average stay duration, the number of cancellations and no-shows, the count of paid invoices and of pending payments with their outstanding value, the proportion of repeat guests, and the top-performing room type by revenue. Read together, these indicators give the owner a comprehensive yet immediate sense of how the hotel is performing financially, how fully it is occupied, and how reliably its guests are paying and returning.

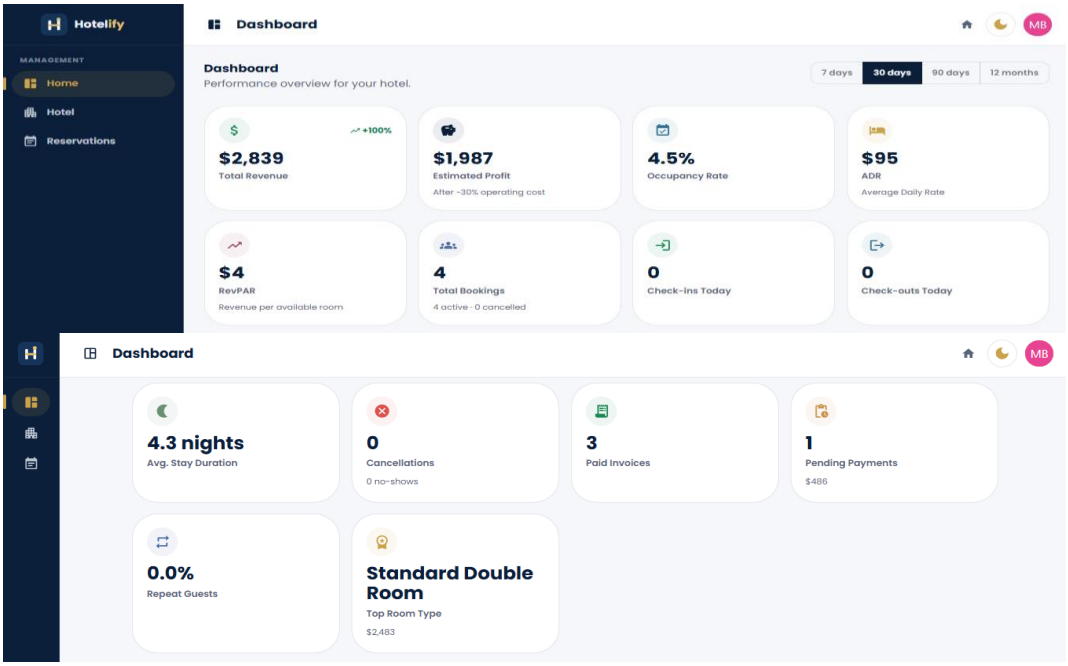


Figure 4.55: *KPI cards summarising the hotel's financial*

Beneath the indicators, a sequence of charts renders the same information over time and across categories, turning static figures into trends that are easier to interpret. The first of these, shown in Figure 4.83, traces the revenue trend, plotting the revenue accruing from active bookings across the selected period so that the owner can see at once whether income is rising or falling and relate that movement to particular dates.

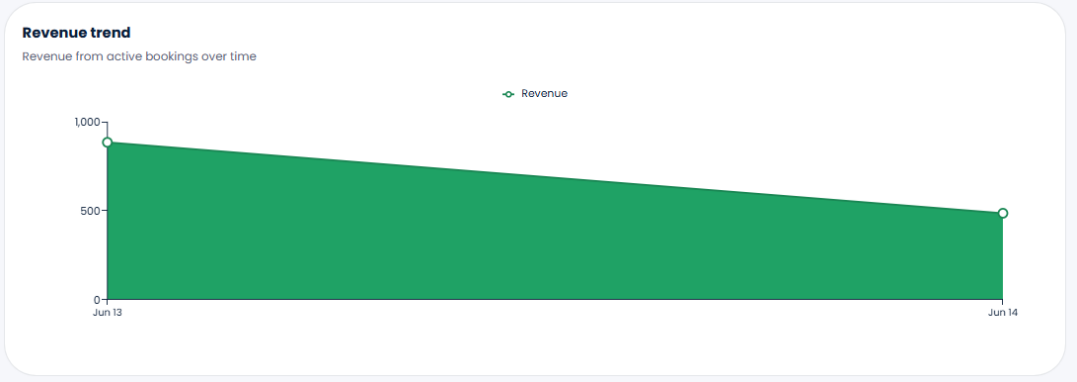


Figure 4.56: *The Revenue Trend Chart*

Figure 4.84 groups four operational charts. The bookings trend records the number of bookings made in each sub-period; the booking status chart shows the distribution of reservations by their status, separating, for example, those already checked out from those merely confirmed, a distinction that, as established in Chapter 3, is independent of whether payment has been settled; the occupancy trend plots the number of rooms occupied by check-in date, revealing the rhythm of demand over time; and the revenue by room type chart attributes income to each category of room, showing the owner which products contribute most to turnover.



Figure 4.57: *The Operational Analytics*

Figure 4.85 presents a second cluster of four charts oriented towards the commercial and demand profile of the hotel. The top-performing rooms chart ranks room types by the number of bookings they attract; the payment methods chart shows the split between guests who pay at the property and those who pay by card; the guest segments chart distinguishes business from leisure travellers; and the cancellations trend follows cancelled bookings over time, displaying an explicit empty state when no cancellations fall within the selected period. Collectively, these views help the owner understand who is booking, why, and how, and so inform decisions on pricing, promotion, and room allocation.



Figure 4.58: *The Commercial Analytics*

Of particular importance are the two AI panels shown in Figure 4.86, which bring the outputs of the system's two models directly into the owner's strategic view. The first is a price prediction panel that forecasts the nightly price of each room type over the coming 12 months, plotting a separate curve for each category and giving the owner forward visibility of how prices are expected to evolve. The second is a review sentiment panel that aggregates the polarity of the hotel's analysed guest reviews into a single distribution of positive, neutral, and negative opinion. These two panels are summary, at-a-glance expressions of the price prediction and sentiment analysis models whose construction and evaluation were presented in Section 4; their actionable, fine-grained counterparts appear on the Hotel page, discussed below.

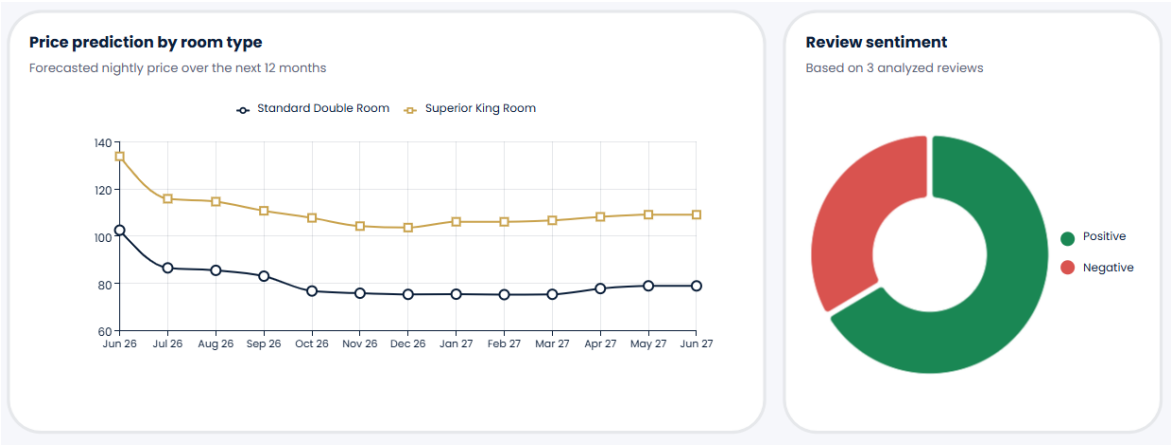


***Figure 4.59**: The AI Insights*

Finally, the Home page closes with a reservation log, shown in Figure 4.87, that lists the most recent bookings of the selected period in tabular form. For each reservation it records the guest, the room type, the reservation status, the total amount, and the date on which the booking was made, giving the owner a concise operational record that complements the aggregated charts above it with concrete, individual transactions.

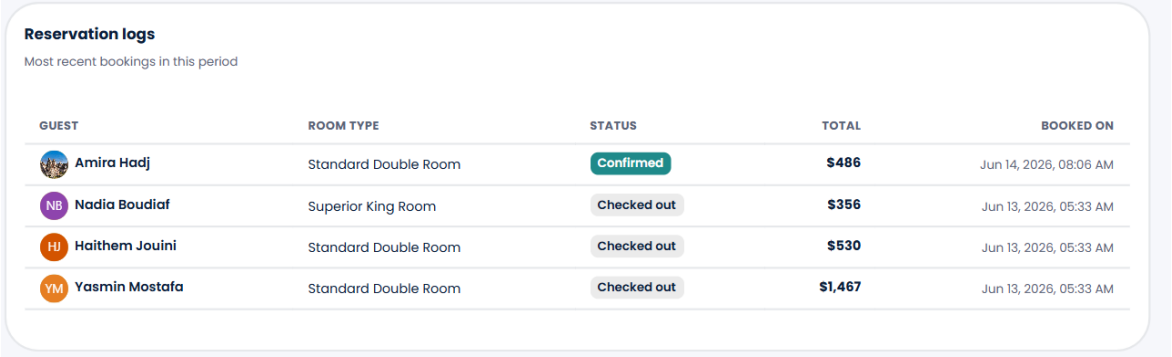


***Figure 4.60:** The Recent Reservation Log*

5.3.2. Hotel Page

Where the Home page is concerned with observation, the Hotel page is concerned with management. It is the workspace through which the owner curates the single property bound to their account: its public presentation, its catalogue of room types, the prices at which those rooms are offered, and the feedback they receive. Its operations are presented here in turn.

The page opens with the hotel's profile, shown in Figure 4.88. A gallery presents the property's photographs, with the principal image displayed prominently and the remainder accessible through a thumbnail strip and a counter indicating how many further photographs are available. Below the gallery, the hotel's identity is set out, namely its name, its full address, and a descriptive paragraph, followed by the list of hotel-wide facilities such as free WiFi, a restaurant, room service, a 24-hour front desk, free parking, and a fitness centre. An Edit hotel control gives access to the editing operation described next.

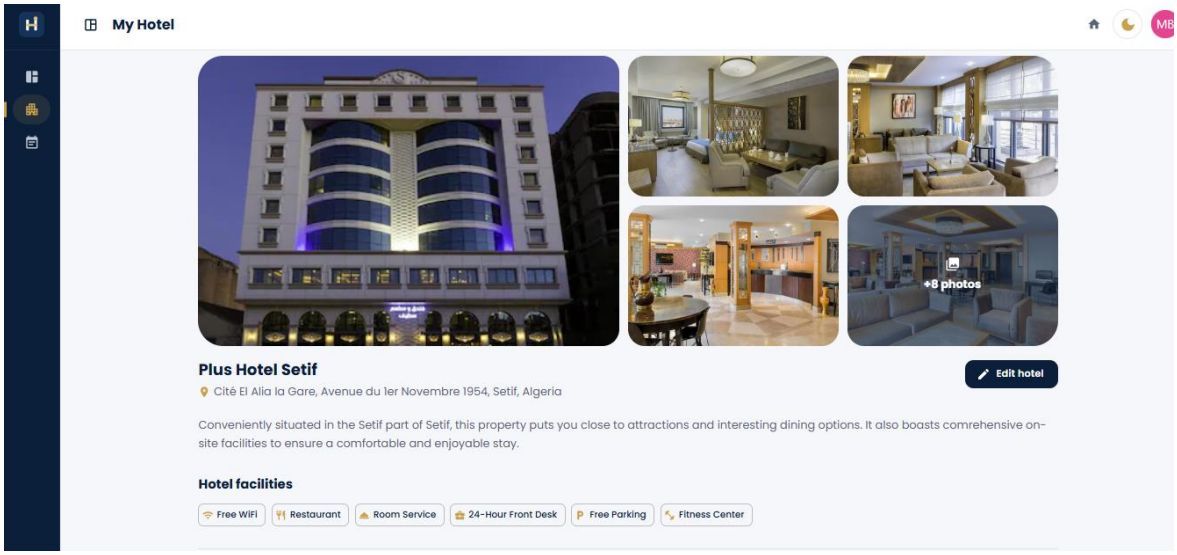


Figure 4.61: *The Hotel Profile*

Selecting Edit hotel opens the dialog shown in Figure 4.89, through which the owner revises the hotel's information. The dialog exposes the editable identity fields, namely name, description, address, country, and city, together with an image manager in which existing photographs may be removed and new ones added, newly added images being marked as such until the change is saved. Below these, the facilities are presented as removable chips, and a combined selector allows the owner either to choose from a predefined list of facilities or to define an entirely new one and add it. Confirming the change with Save changes persists the new information and reports the outcome through the success Snackbar introduced in Section 5.1, whereas Cancel dismisses the dialog without applying any change.

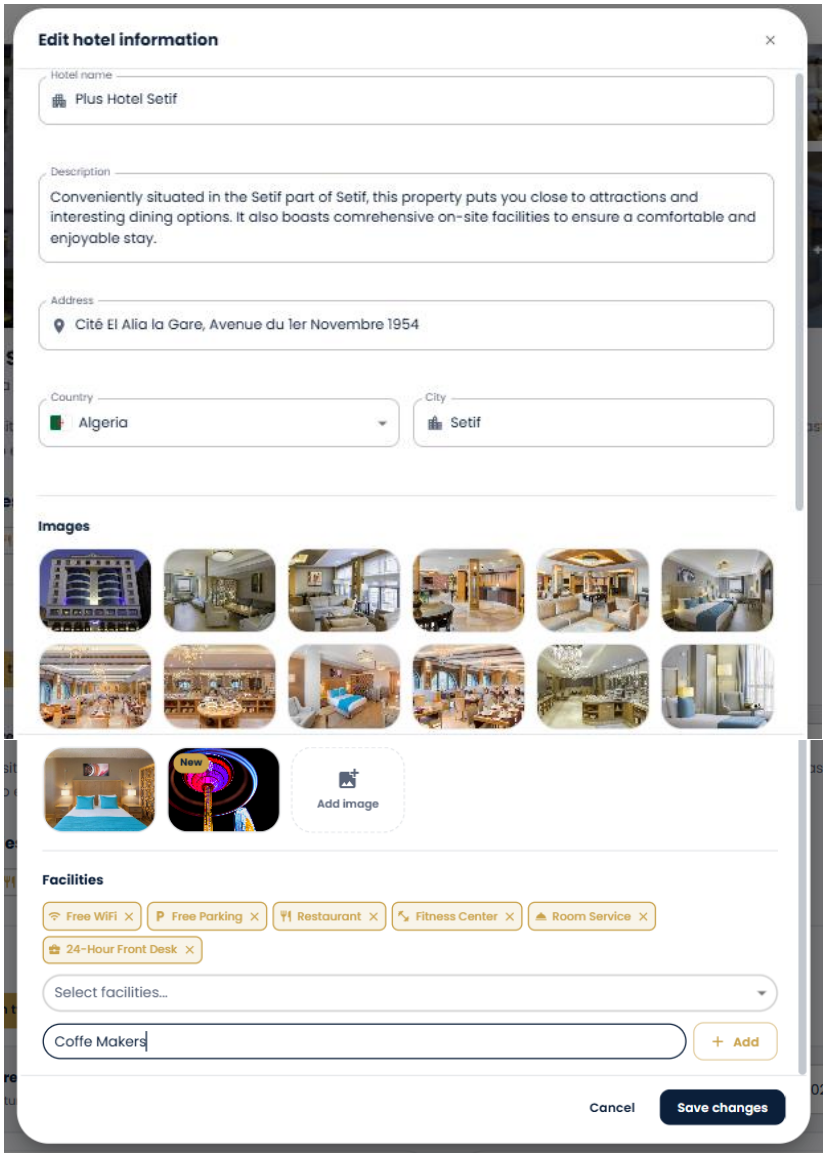


Figure 4.62: *The Edit Hotel Information Dialog*

Below its profile, the Hotel page lists the room types that the hotel offers, each presented as a card stating the room's name and category, its capacity and bed configuration, the number of rooms of that type, its facilities, and its base nightly price. The owner manages this catalogue through three operations. A new room type is created through the dialog shown in Figure 4.90, in which the owner specifies the room's name and category, its base price, its maximum number of guests and its total number of rooms, its bed configuration, and its facilities, the last again selectable from a predefined list or definable anew. An existing room type is revised through the closely related dialog shown in Figure 4.91, which presents the same fields populated with the room's current values. A room type may also be removed, an action mediated by the shared confirmation dialog established earlier in this chapter, so that the owner does not delete a room type inadvertently.

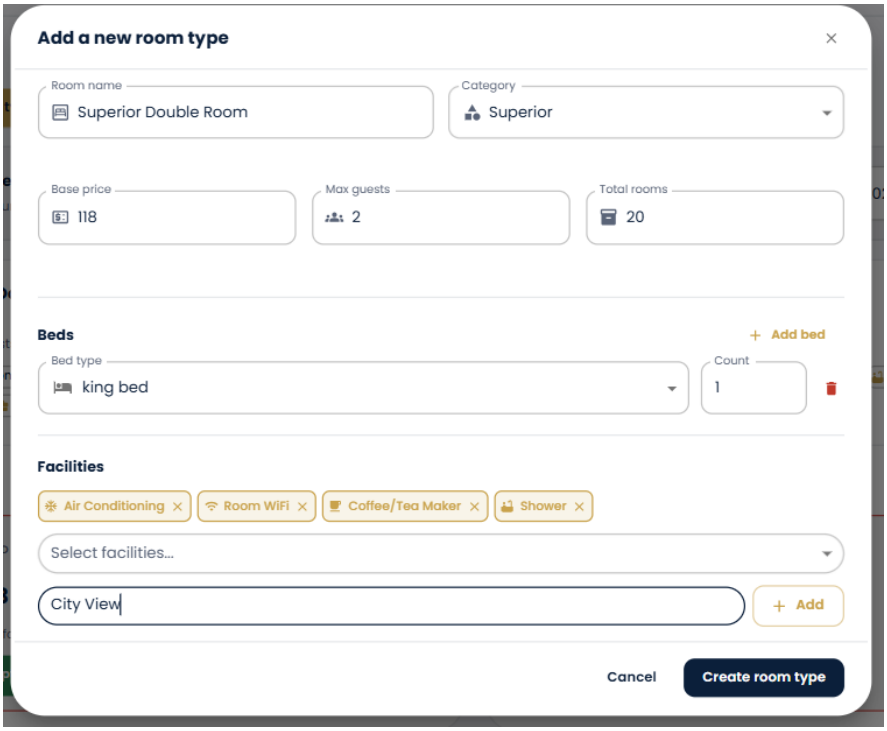


Figure 4.63: *The Add a New Room Type Dialog*

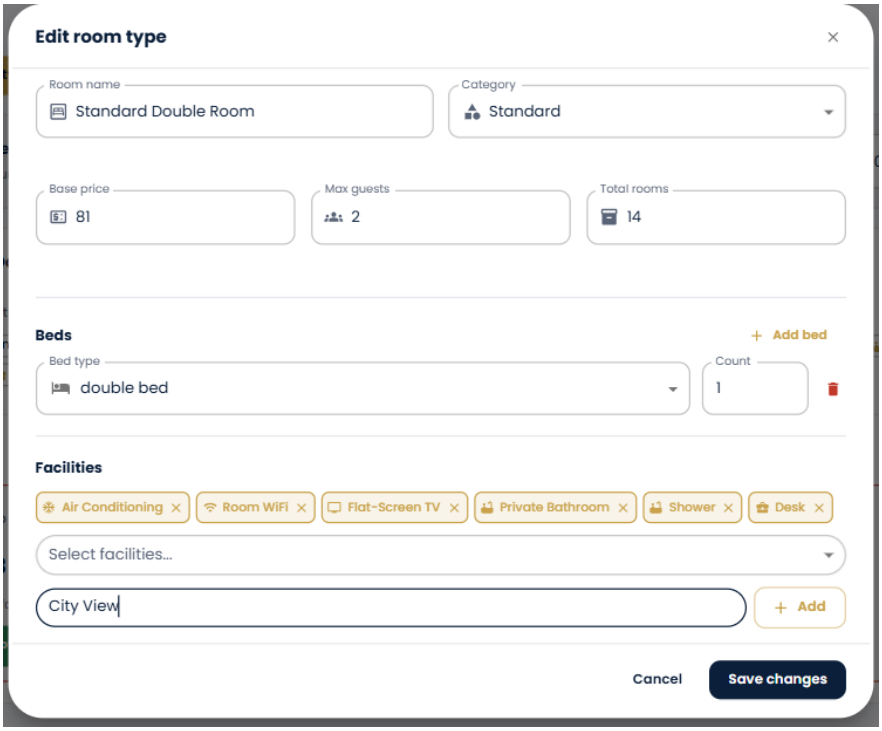


Figure 4.64: *The Edit Room Type Dialog*

The Hotel page is also where the price prediction model is placed directly into the owner's hands as an actionable recommendation tool. Above the room type cards, a price prediction panel invites the owner to pick a future check-in date, upon which the system forecasts, for every room type, the nightly price recommended for that date. As Figure 4.92 illustrates for two different chosen dates, each room card then displays, beneath its

base price, a predicted price together with the percentage by which it differs from the base price and a trend indicator showing whether the recommendation is to raise, lower, or hold the price; the recommendation may thus differ in both magnitude and direction from one date to another, reflecting the demand conditions that the model associates with each date. A Why? control beside each prediction reveals the factors underlying the recommendation, in the same manner described for the guest's view of the forecast in Section 5.2.

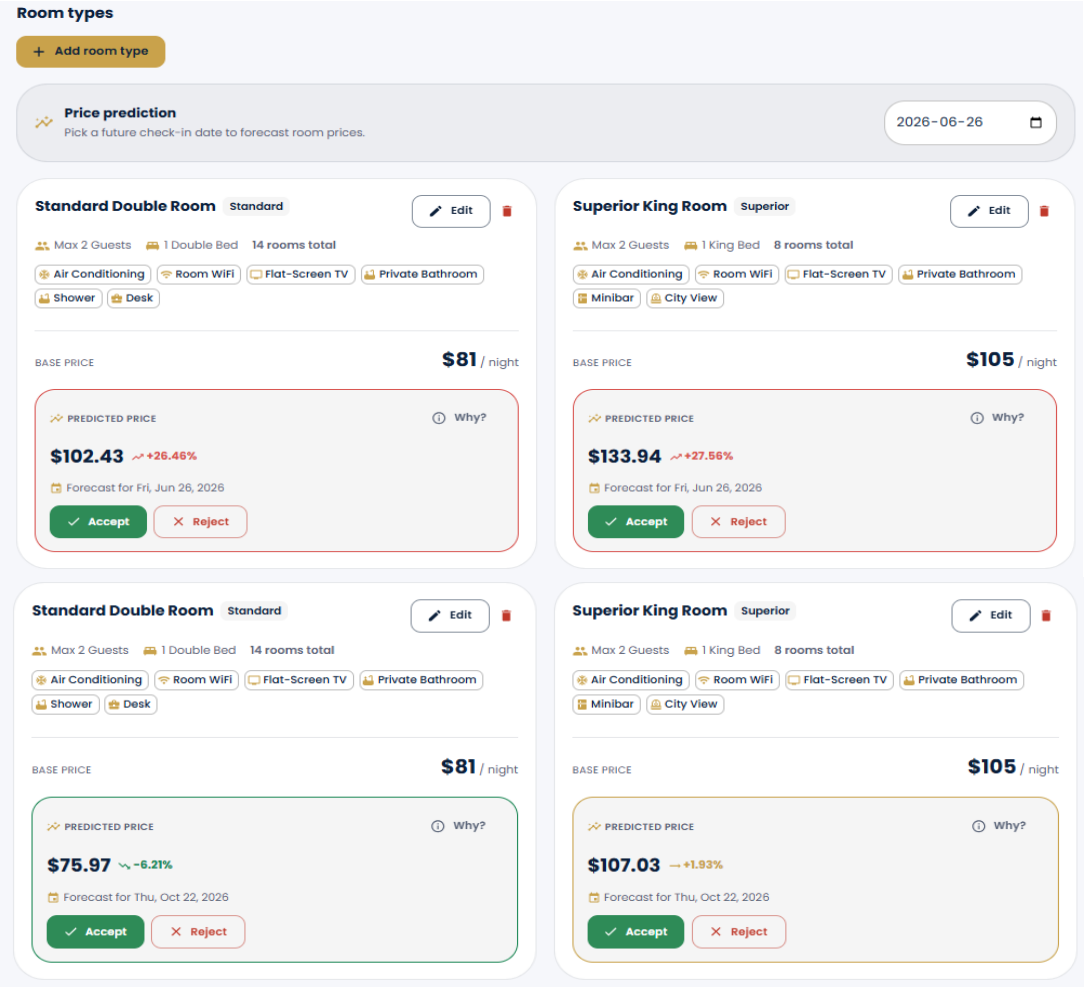


Figure 4.65: *The Price Recommendation Produced for Two Different Selected Check-In Dates*

For each recommendation the owner retains full control through two explicit actions. Accepting a recommendation, as shown in Figure 4.93, applies the predicted price to the room type as its new base price and marks the card accordingly; rejecting it, as shown in Figure 4.94, discards the recommendation and leaves the existing base price unchanged, again marking the card to record the decision. In this way the model acts as a decision-support tool rather than as an autonomous pricing authority: it proposes a price, with a justification, while the final pricing decision remains with the owner. This realises, in the running application, the price prediction and recommendation capability whose model was developed and evaluated in Section 4.

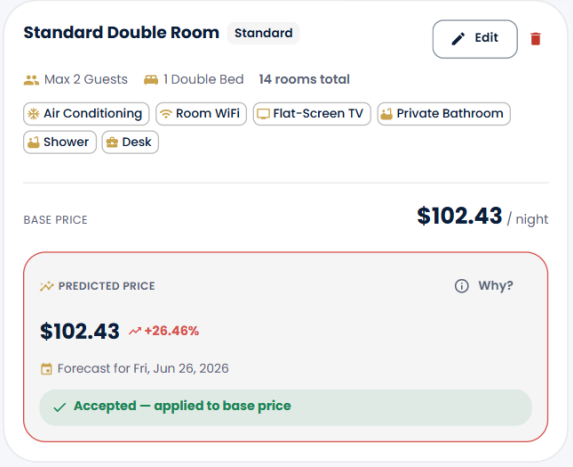


Figure 4.66: *An Accepted Price Recommendation*

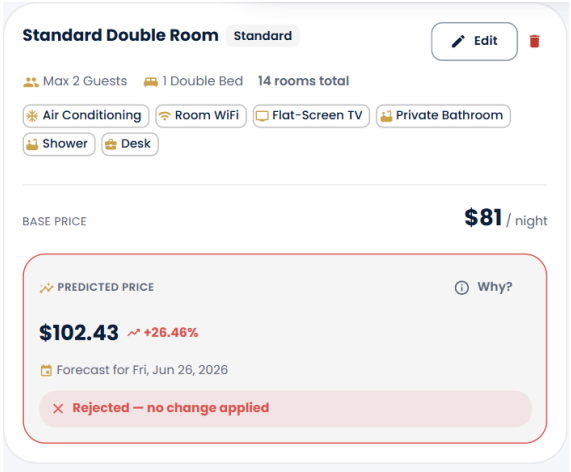


Figure 4.67: *A Rejected Price Recommendation*

Finally, the Hotel page presents the guest reviews of the property, shown in Figure 4.95, closing the loop between the experiences of guests and the awareness of the owner. The section opens with an overall rating and a count of reviews, followed by a breakdown of scores across the rating categories of staff, facilities, cleanliness, comfort, value for money, and location, which lets the owner see precisely which aspects of the stay guests rate most and least favourably. Beneath this summary, the individual reviews are listed, each showing the reviewer, their room type and stay dates, their numeric rating, and the review text. Crucially, each review is accompanied by a sentiment label and an associated confidence produced by the sentiment analysis model, for instance a review classified as negative or as positive with the model's confidence in that judgement, so that the owner not only reads the feedback but sees it automatically classified by polarity. This is the per-review, fine-grained counterpart of the aggregate sentiment panel on the Home page, and it brings the sentiment analysis model of Section 4 to bear directly on the owner's understanding of guest satisfaction.

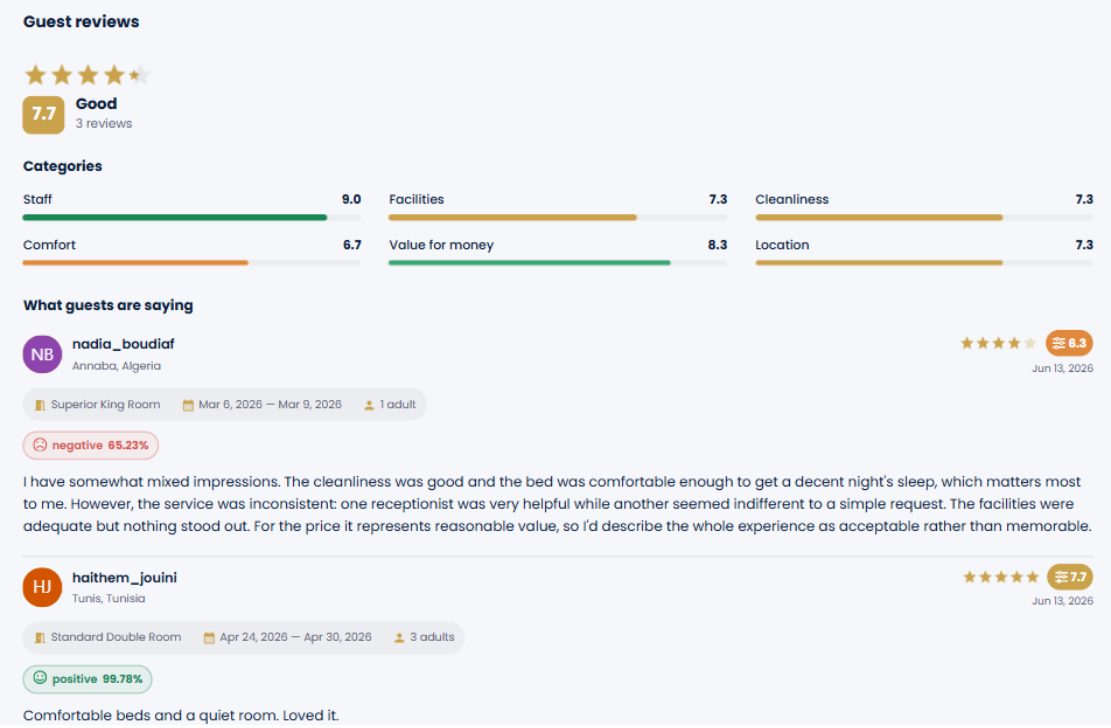


Figure 4.68: *The Guest Reviews of the Property*

5.3.3. Reservations Page

The third area of the owner dashboard is the Reservations page, through which the owner administers the bookings made against the hotel across their entire lifecycle. Whereas the reservation log on the Home page offered only a brief, read-only record of recent bookings, this page is the full operational instrument for managing them. It is shown in Figure 4.96.

At the head of the page, a row of status filter tabs allows the owner to narrow the list to reservations of a particular status, namely all, pending, confirmed, checked in, checked out, cancelled, or no show, each tab displaying the number of reservations it contains. The owner can thus triage the workload at a glance and concentrate on the bookings that require action.

Below the tabs, each reservation is presented as a detailed card. Every card states, in its header, two distinct pieces of status information that, as established in Chapter 3, vary independently of one another: the reservation status, which records where the booking stands in its lifecycle, and the payment status, which records whether the booking has been paid and by which method. The body of the card then sets out the guest's identity and contact details, the booked room type and its bed configuration, the check-in and check-out dates, the number of rooms, the numbers of adults and children, whether the stay is for work, and the date on which the booking was made, supplemented, where relevant, by the date of payment or of cancellation. The card closes with the nightly rate and the total amount due.

The actions offered by a card depend on the reservation's current status, so that the owner is only ever presented with the operations that are valid for that booking. A confirmed reservation, for example, may be checked in, marked as a no show, or marked as paid; a checked-in reservation may be checked out; and a cancelled reservation that had already been paid may be refunded. Reservations that have reached a terminal state, such as one already checked out, present no further actions. Each such action updates the reservation and reports its outcome through the shared feedback mechanisms introduced earlier in this chapter.

To avoid redundancy, not every status is illustrated separately, since the card layout is common to all of them and they differ only in their badge and in the actions they permit. Two statuses not depicted in the figure nonetheless merit a brief description. A pending reservation is one that has been created but not yet confirmed, and its purpose is to let the owner track bookings that are still awaiting confirmation before they enter the active part of the lifecycle. A no show records a confirmed reservation whose guest failed to arrive, a status the owner assigns through the corresponding action so that the booking is closed accurately rather than left open. Together with the statuses shown, these give the owner a complete and faithful account of every booking, which is essential both for day-to-day front-desk operations and for the reliability of the analytics presented on the Home page.

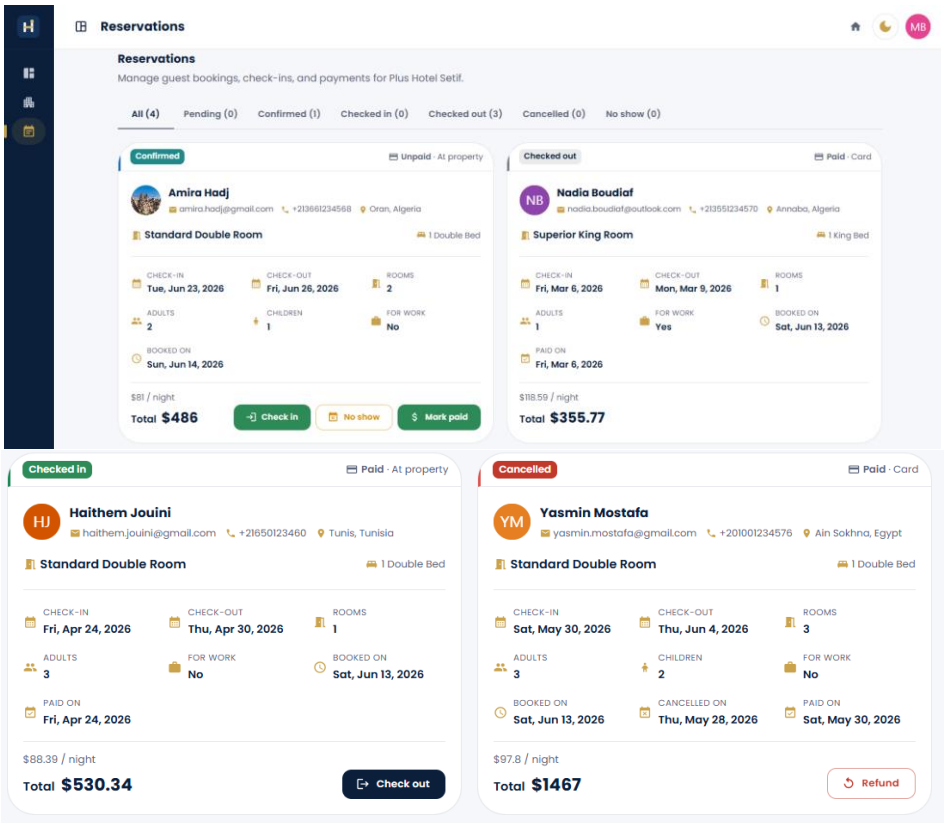


Figure 4.69: *The Reservations Page*

5.3.4. My Account

The final area of the owner dashboard is the My Account page, shown in Figure 4.97, through which the owner manages their own personal account. In its presentation of the owner's profile, namely the first name, last name, email, telephone number, and country, and in the Edit profile operation through which these details are revised, this page mirrors the guest account page described in Section 5.2, and the editing operation is therefore not repeated here.

What distinguishes the owner's account page, and warrants explanation, is the access code panel at its top. The access code is the secret second factor used in the two-step sign-in introduced in Section 5.1, by which the owner confirms their identity when entering the dashboard. Because this code is security-sensitive, the panel keeps it masked by default and does not display it on the page.

Revealing the code is therefore guarded by a deliberate re-authentication step. When the owner selects Reveal, the system does not show the code immediately but first opens the identity verification dialog shown in Figure 4.98, which requires the owner to re-enter their account password and warns that the code will be visible only briefly. Once the password is verified, the code is revealed, as shown in Figure 4.99, together with a control to copy it and a visible countdown indicating that it will be hidden again automatically after 15 seconds, after which the panel returns to its masked state. This design allows the owner to retrieve their second factor when they need it, for instance when signing in from a new device, while protecting it against casual observation and incidental exposure on screen.

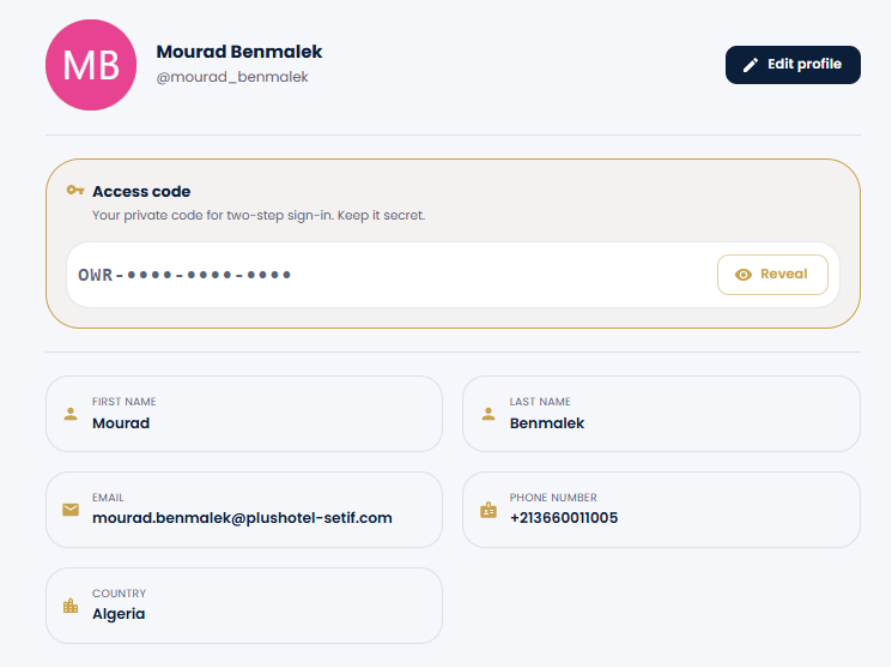


Figure 4.70: *The Owner's Account*

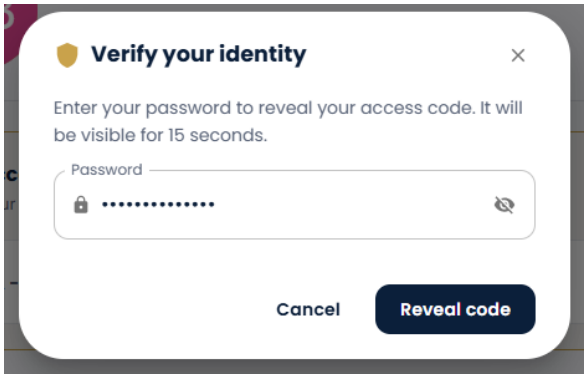


Figure 4.71: *The Identity Verification Dialog Requiring the Account Password before the Access Code is Revealed*

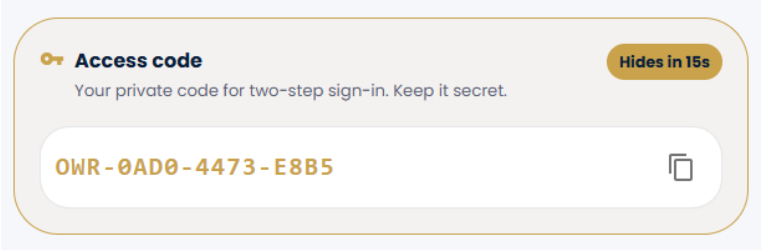


Figure 4.72: *The Access Code Revealed for a Limited Time*

5.4. Admin Dashboard

The final subsection of the application presentation concerns the administrator, the actor with the broadest authority in the system. Unlike the owner, whose remit is confined to a single hotel, the administrator oversees the platform in its entirety: every hotel, every owner, every guest, and every reservation. As with the owner, access to the administrator's workspace is protected by the two-step sign-in described in Section 5.1. Once authenticated, the administrator is presented with a dashboard that, while sharing the visual language and layout of the owner dashboard, is platform-wide in scope. Its navigation sidebar gives access to its areas: the Home page, which aggregates the analytics of the whole platform; the Users page, through which owners and guests are managed; the Hotels page, through which hotels are managed; the Reservations page, which lists every reservation on the platform; and the AI Models page, through which the system's models are monitored. This part presents the Home and Users pages; the remaining pages are presented in the part that follows.

5.4.1. Home Page

The administrator's Home page is the platform-wide counterpart of the owner's Home page presented in Section 5.3.1. It shares the same organisation, namely a period selector governing a scrollable arrangement of indicator cards, charts, and records, but where the owner's page reported on a single hotel, this page aggregates the same classes of information across the entire platform. Because the structure and the chart types are largely those already described for the owner, the following discussion concentrates on

the platform-wide meaning of each element and on the panels that are specific to the administrator.

The page opens with a band of KPI cards, shown in Figure 4.100, whose indicators describe the health of the whole business. A first group reports the platform's commercial fundamentals: the total revenue transacted across all hotels, the platform commission earned from that revenue at a rate of 10 per cent, the total number of bookings, and the average booking value. A second group quantifies the platform's reach, namely the total number of hotels and of those currently active, the total number of owners and of guests, and the number of newly registered users in the period broken down by role, giving the administrator an immediate measure of how large the platform is and how quickly it is growing. A third group reports aggregate operational quality: the platform-wide occupancy rate, the average ADR across all hotels, the cancellation rate, the number and value of pending payments, and the monthly growth. Together these indicators allow the administrator to judge, at a glance, whether the platform is growing, earning, and operating soundly.

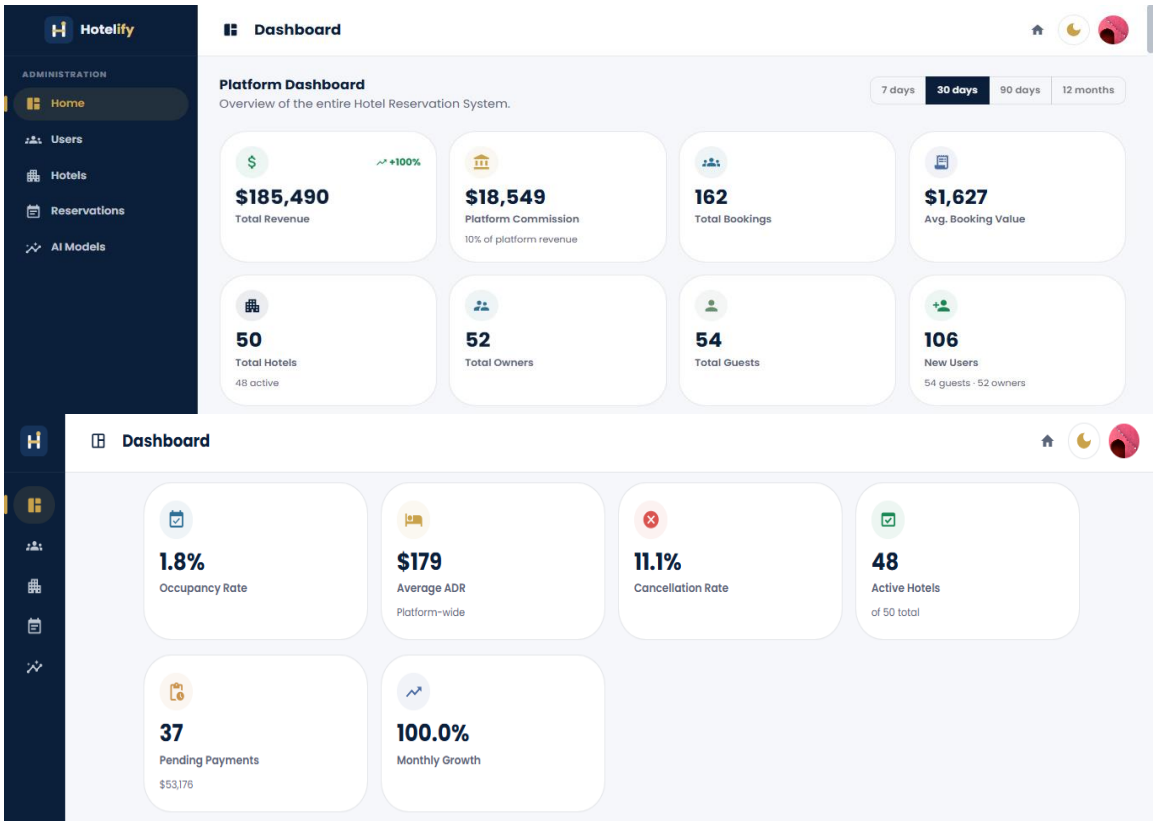


Figure 1.73: *The Platform-Wide KPI Cards Covering the Platform's Commercial Results*

Beneath the indicators, Figure 4.101 presents four charts concerned with growth and with the platform's strongest performers. The bookings trend plots the volume of bookings over time, and the user growth chart traces the registration of new guests and owners across the period, so that the administrator can follow the expansion of both sides of the marketplace. Two ranking charts then identify the platform's leaders: the top

hotels by revenue and the top owners by revenue, which highlight the properties and the operators that contribute most to the platform's turnover and so warrant particular attention.



***Figure 4.74**: The Growth and Performance Charts*

Figure 4.102 groups four distribution charts that characterise where the platform's activity comes from and how it is composed. The revenue by country chart attributes turnover to each of the countries in which the platform operates, revealing its strongest markets; the booking status chart shows the distribution of all reservations across their possible statuses, namely checked out, pending, cancelled, no show, and confirmed; the payment methods chart shows the platform-wide split between payment at the property and payment by card; and the guest segments chart shows the balance between leisure and business travellers. These views give the administrator a structural understanding of demand that complements the headline indicators.



Figure 4.75: *The Distribution Charts*

Figure 4.103 presents two further charts. The hotels per country chart shows the geographic distribution of the properties themselves, complementing the revenue by country view by distinguishing where hotels are located from where revenue is earned; and the cancellations trend follows cancelled bookings over time at the level of the whole platform.



Figure 4.76: *The Geographic and Cancellation Charts*

Of particular note is the per-hotel AI panel shown in Figure 4.104, which extends the administrator's oversight to the outputs of the system's two models. By means of a hotel selector, the administrator may choose any hotel on the platform and inspect, for that hotel, the price prediction by room type over the coming 12 months and the distribution of sentiment across its analysed reviews. Whereas the owner sees these insights only for their own property, the administrator can examine them for every hotel, which makes this panel an instrument of platform-wide quality monitoring for both the price prediction and the sentiment analysis models presented in Section 4.



Figure 4.77: *The Per-Hotel AI Insights*

As on the owner's page, a recent reservations table, shown in Figure 4.105, lists the latest bookings; here, however, it spans the entire platform and accordingly records, for each reservation, not only the guest, the status, the total, and the booking date but also the hotel and the owner concerned, so that the administrator can trace any booking to the property and the operator behind it.

Recent reservations

Latest bookings across the platform

GUEST	HOTEL	OWNER	STATUS	TOTAL	BOOKED ON
Youssef Benali	Four Points by Sheraton Oran	Meriem Bouzid	Confirmed	$906	Jun 26, 07:28 AM
Youssef Benali	Four Points by Sheraton Oran	Meriem Bouzid	Confirmed	$906	Jun 26, 07:27 AM
Youssef Benali	Four Points by Sheraton Oran	Meriem Bouzid	Confirmed	$2,718	Jun 26, 07:23 AM
Islam Chorfi	Novotel Constantine	Hichem Ouali	Confirmed	$402	Jun 25, 06:37 PM
Islam Chorfi	Radisson Blu Hotel, Cairo Heliopolis	Amr El-Hawary	Confirmed	$720	Jun 25, 06:14 PM
Youssef Benali	Four Points by Sheraton Oran	Meriem Bouzid	Checked out	$1,510	Jun 17, 10:40 AM
Dhia Chorfi	Radisson Blu Hotel, Cairo Heliopolis	Amr El-Hawary	Confirmed	$1,440	Jun 17, 09:11 AM
Youssef Benali	Hotel Corner King	Nassim Hamici	Checked out	$90	Jun 16, 05:41 PM
Amira Hadj	The Mirage Resort & SPA	Wissem Belhadj	Confirmed	$530	Jun 14, 08:17 AM
Amira Hadj	Plus Hotel Setif	Mourad Benmalek	Checked out	$486	Jun 14, 08:06 AM
Youssef Benali	Four Points by Sheraton Oran	Meriem Bouzid	Cancelled	$3,528	Jun 13, 09:25 PM
Youssef Benali	Hotel Sabri	Wafa Saidani	Cancelled	$300	Jun 13, 09:23 PM
Khaled Ibrahim	Timgad Hotel Hazem	Djamila Kerboua	Checked out	$337	Jun 13, 05:33 AM
Mariam Khalil	Ibis Ouarzazate Centre	Hamza Zeghari	Pending	$1,092	Jun 13, 05:33 AM

Figure 4.78: *The Recent Reservations Across the Platform*

Finally, and uniquely to the administrator, the Home page presents an authentication activity log, shown in Figure 4.106. This records the most recent account events on the platform, namely sign-ups, logins, and logouts, identifying for each the user, their role, the action performed, and the time at which it occurred. The log gives the administrator visibility of who is accessing the system and when, serving as a lightweight audit and security facility over the authentication system whose implementation was described earlier in this chapter.

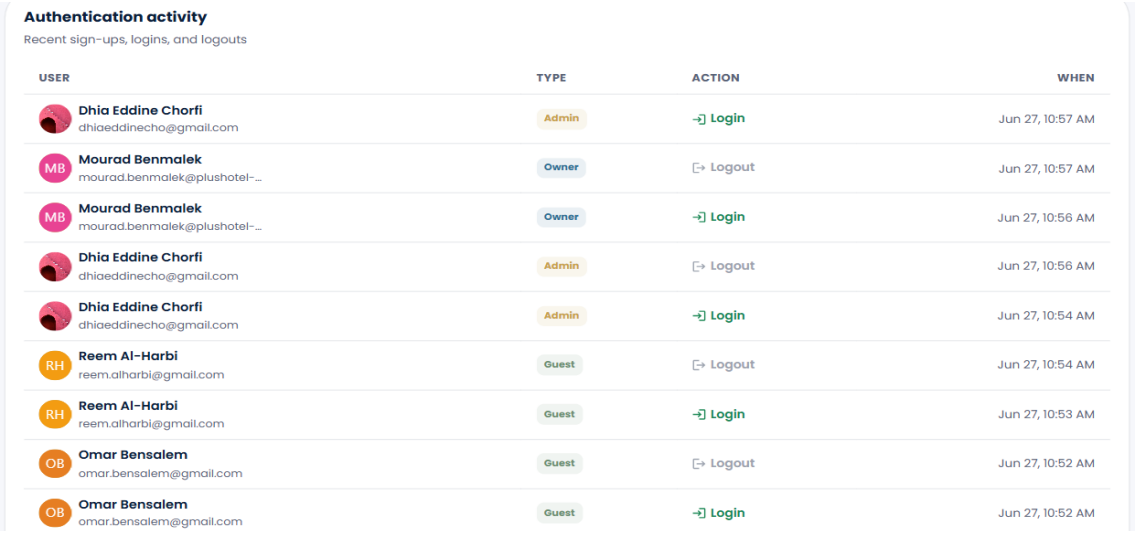


Figure 4.79: *The Authentication Activity Log*

5.4.2. Users Page

The Users page is the workspace through which the administrator manages the platform's people: the hotel owners and the guests. As shown in Figure 4.107, the page separates these two populations into tabs, each labelled with its current count, and provides a search field for locating an individual within the selected population. The owners tab, shown in the same figure, lists each owner as a card bearing their photograph, name, username, contact details, and country, together with controls to view, edit, or remove the account; a Create owner action sits above the list.

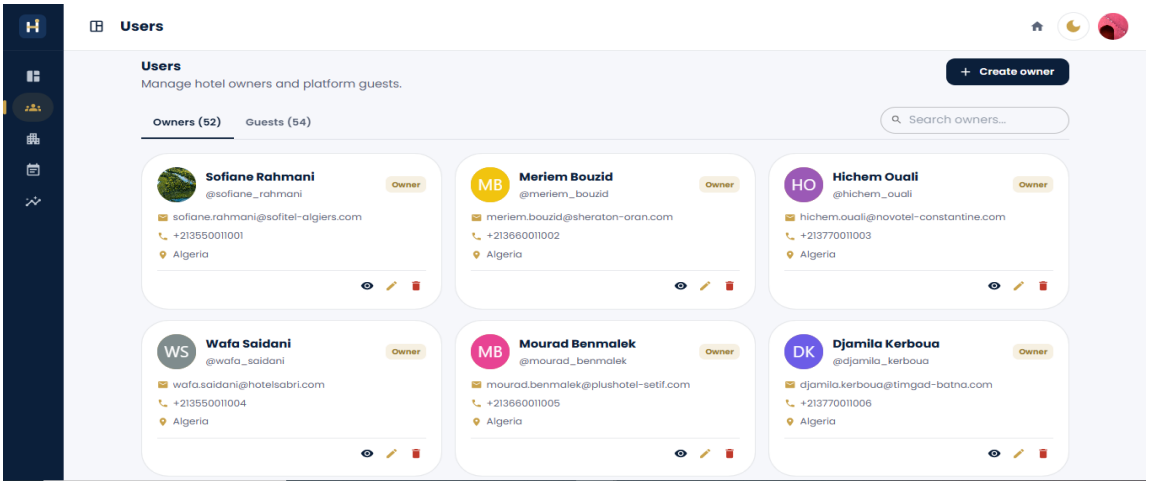


Figure 4.80: *The Owners Tab*

The presence of a Create owner action reflects the role model, established in Chapter 3, in which owner accounts are provisioned by the administrator rather than self-registered. Selecting it opens the dialog shown in Figure 4.108, in which the administrator enters the new owner's identity and contact details together with the credentials, namely the username and password, with which the owner will sign in. The hotel that the owner is to

manage is created for them separately, through the Hotels page presented in the following part.

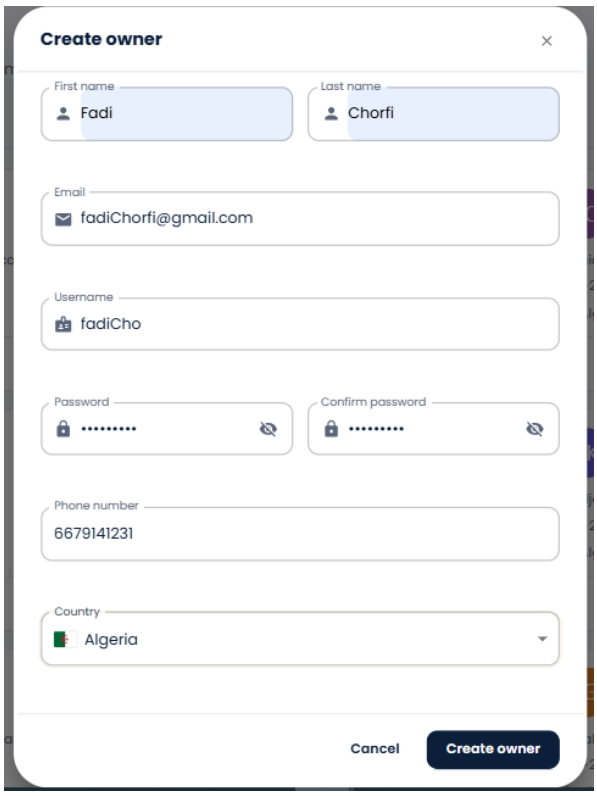


Figure 4.81: *The Create New Owner Dialog*

The view control opens the owner details dialog shown in Figure 4.109, a read-only summary that presents the owner's full profile, including the date on which the account joined the platform. The edit control opens the dialog shown in Figure 4.110, through which the administrator may revise any of the owner's details, including replacing their profile photograph. The remove control, being destructive, is mediated by the shared confirmation dialog so that an account is not deleted inadvertently.

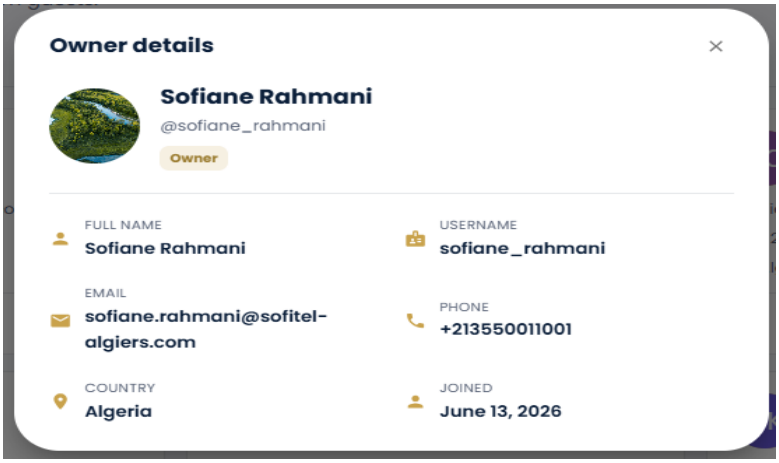


Figure 4.82: *The Owner Details Dialog*

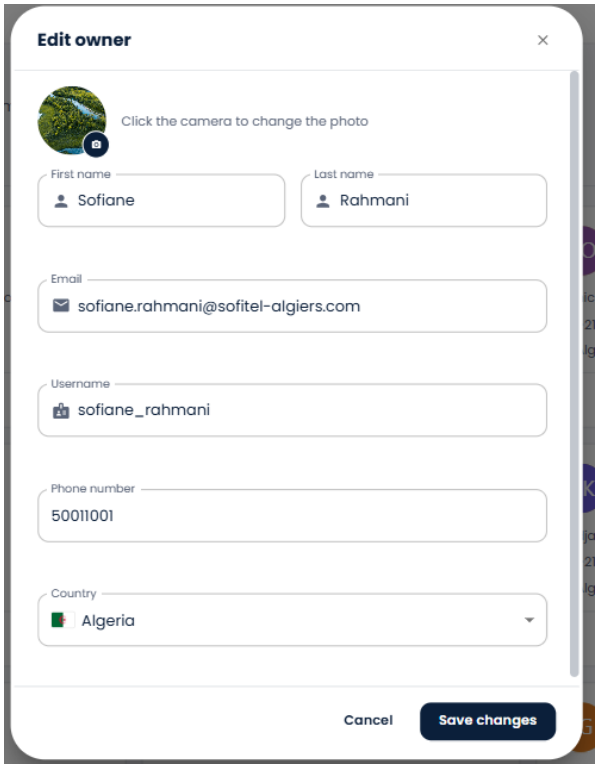


Figure 4.83: *The Edit Owner Dialog*

Switching to the guests tab, shown in Figure 4.111, presents the platform's guests in the same card format. Here, however, the available controls are deliberately narrower: a guest card offers only the options to view the guest's details or to remove the account, with no facility either to create a guest or to edit one. This asymmetry is intentional and again follows from the role model: unlike owners, guests register themselves through the public sign-up described in Section 5.1, so that the administrator's role with respect to them is one of oversight and moderation rather than of provisioning.

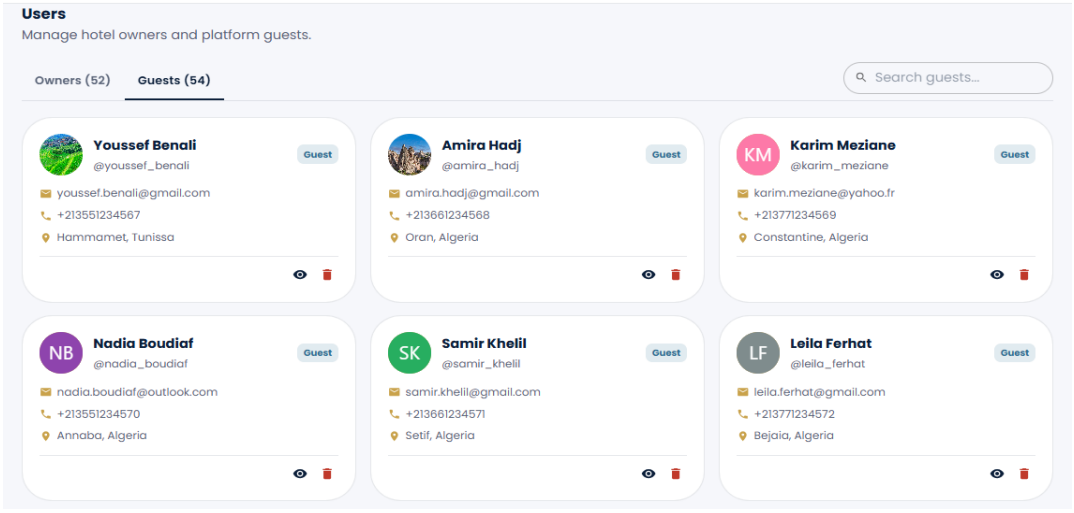


Figure 4.84: *The Guests Tab*

5.4.3. Hotels Page

The Hotels page is the administrator's catalogue of every property on the platform. As shown in Figure 4.112, it lists the hotels as cards, each displaying the property's photograph and starting price, its name and address, and the owner to whom it belongs, together with controls to view the hotel's details or to remove it; a search field locates a hotel by its name or its owner, and a Create hotel action sits above the list. The cards offer no direct editing control, since the content of a hotel is maintained by its own owner through the Hotel page described in Section 5.3.2; the administrator's role here is to create hotels, to inspect them, and, where necessary, to remove them.

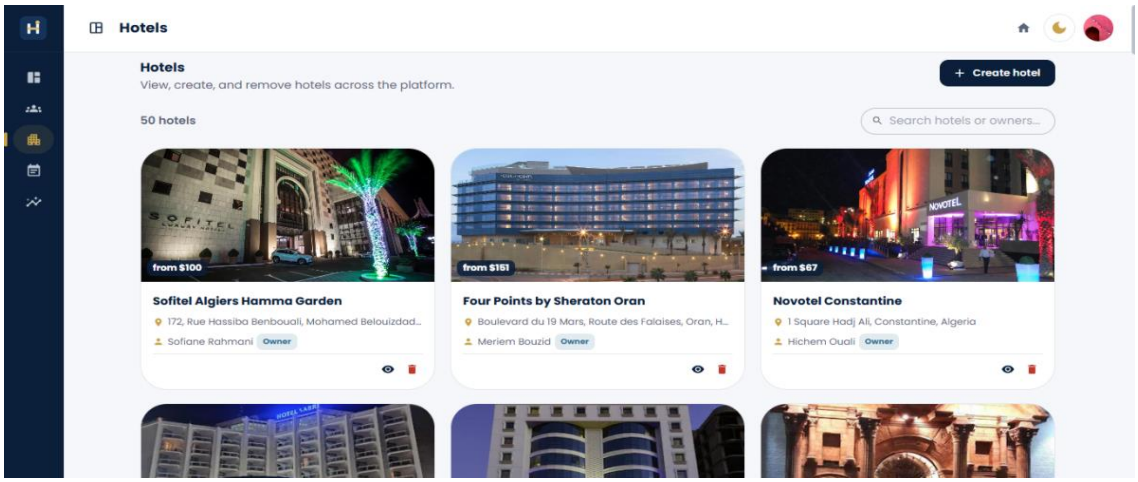


Figure 4.85: *The Catalogue of All Platform Hotels*

The Create hotel action completes the provisioning of an owner that was begun on the Users page. Selecting it opens the dialog shown in Figure 4.113, whose defining feature is the owner selector at its top: through it, the administrator assigns the new hotel to a previously created owner, thereby establishing the one-to-one binding between an owner and their hotel that was set out in Chapter 3. The remainder of the dialog captures the hotel's information, namely its name, description, address, country, and city, its photographs, and its facilities, using the same fields and the same facility selector already described for the owner's editing of a hotel in Section 5.3.2. Once created, the hotel becomes the single property that its assigned owner manages.

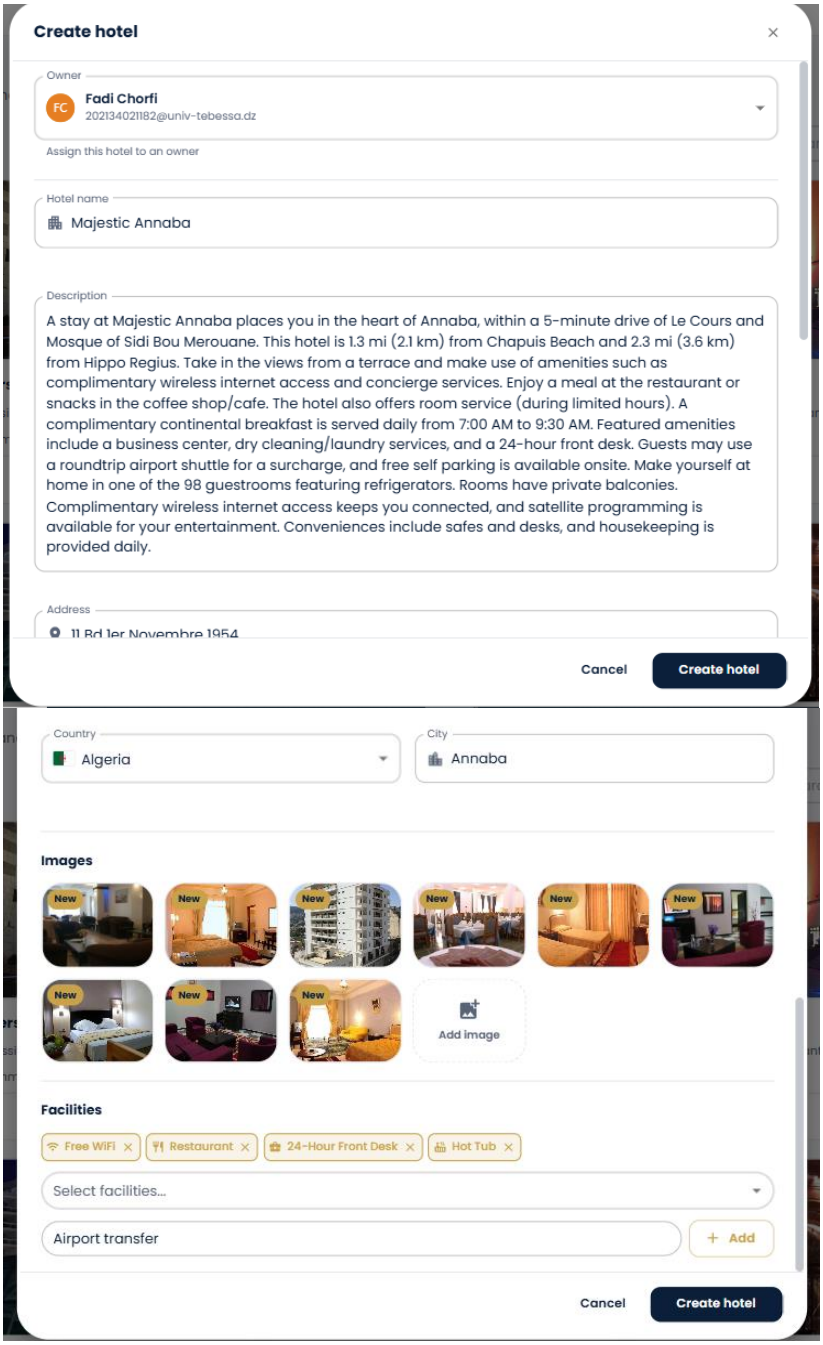


Figure 4.86: *The Create Hotel Dialog*

5.4.4. Hotel Details Page

Selecting the view control on a hotel card opens the Hotel Details page, shown across Figures 4.114 to 4.116, through which the administrator examines a single property in full. It opens, in Figure 4.114, with the hotel's profile: its photograph gallery, its name, address, and description, the owner to whom it belongs, and its facilities. It continues, in Figure 4.115, with the hotel's room types, each presented as a card stating its name and category, its capacity and bed configuration, the number of rooms, its facilities, and its base nightly price. It closes, in Figure 4.116, with the hotel's guest reviews, comprising the overall rating, the breakdown of scores by category, and the individual reviews, each

carrying the sentiment label produced by the sentiment analysis model in the manner described in Section 5.3.2.

It is instructive to contrast this page with the owner's Hotel page of Section 5.3.2, since the two present much of the same information to different ends. The owner's page is a management workspace: through it the owner edits the hotel's information, creates, edits, and removes room types, and accepts or rejects the price recommendations produced for each room. The administrator's Hotel Details page, by contrast, is purely observational: it exposes the same hotel, its rooms, and its reviews for inspection but offers none of these editing or pricing controls, and it additionally identifies the owner responsible for the property. This difference is a direct expression of the two roles. The owner operates a single hotel and therefore needs the means to manage it, whereas the administrator supervises every hotel and therefore needs only to observe each one, leaving its operation to its owner.

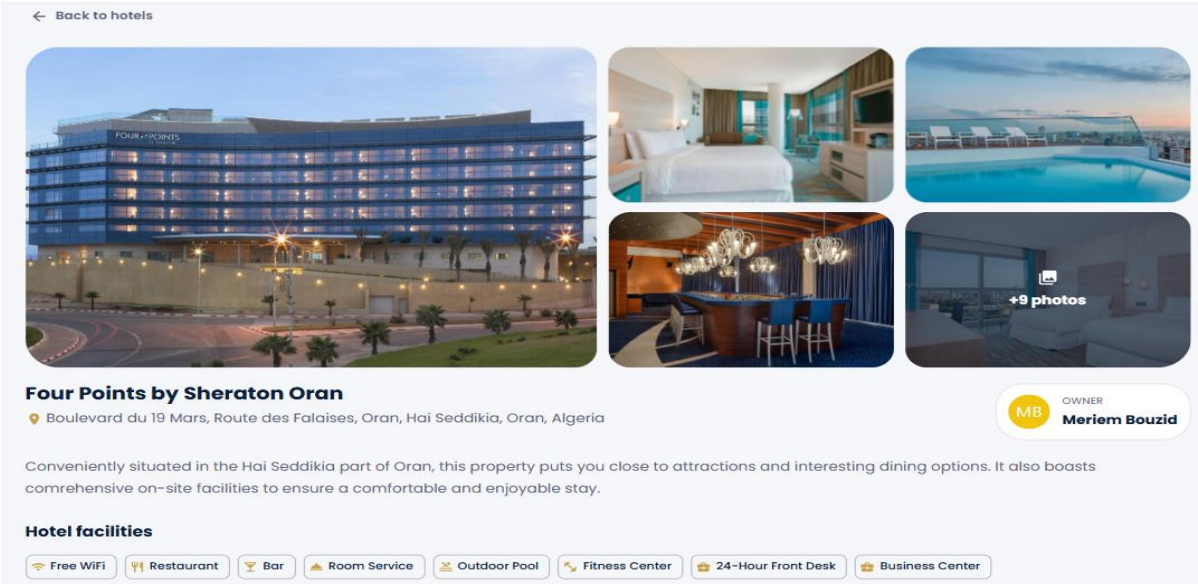


Figure 4.87: *The Hotel Profile*

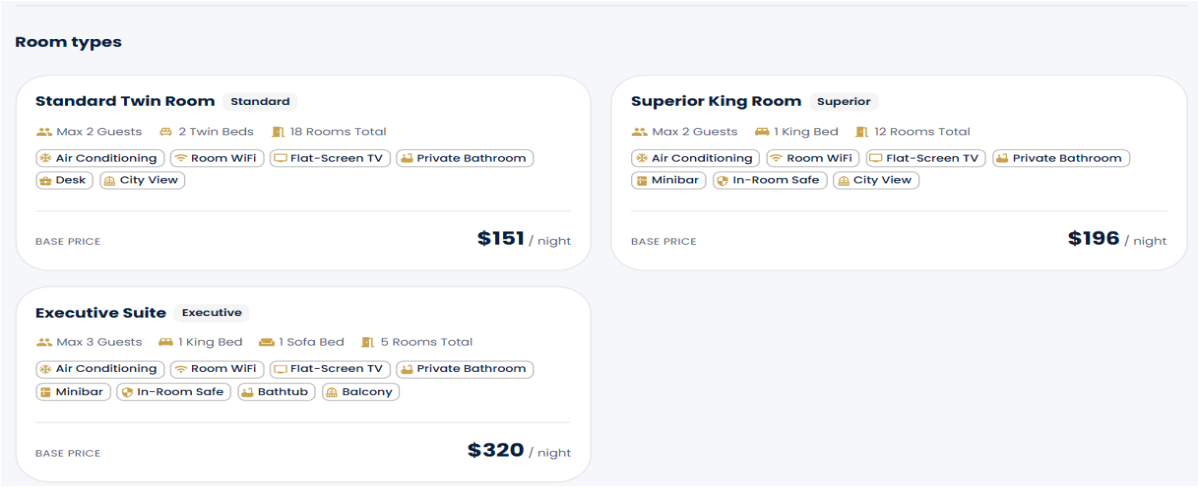


Figure 4.88: *The Hotel's Room Types*

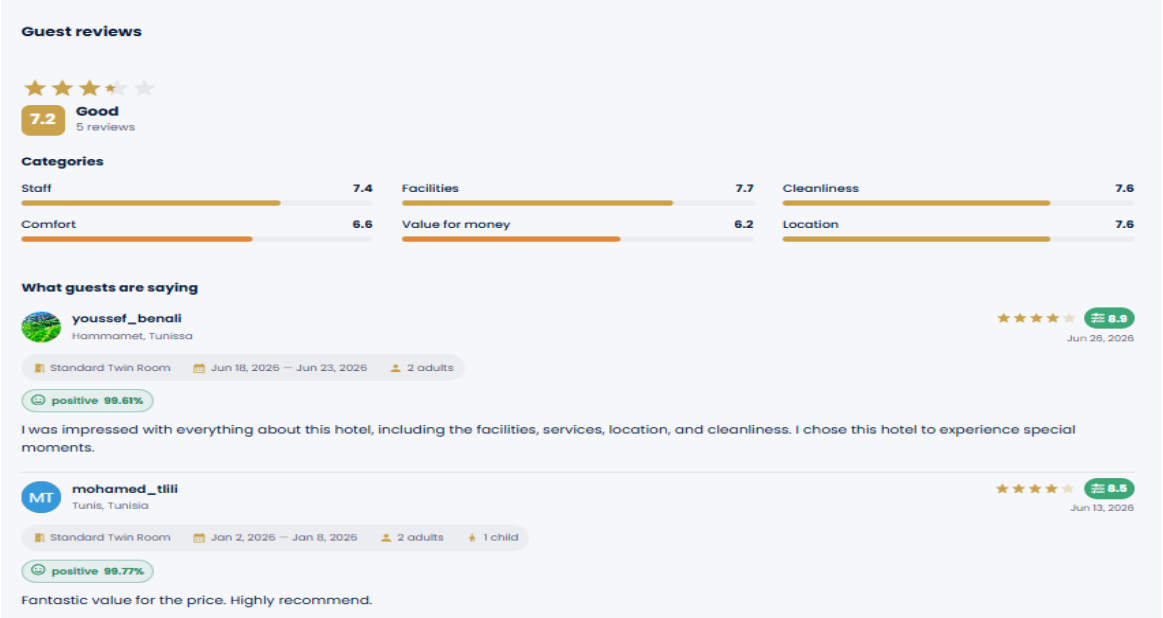


Figure 4.89: *The Hotel's Guest Reviews*

5.4.5. Reservations Page

The administrator's Reservations page, shown in Figure 4.117, provides oversight of every booking on the platform. Like the owner's reservations page, it organises bookings by status through a row of counted filter tabs, namely all, pending, confirmed, checked in, checked out, cancelled, and no show, and presents each booking as a card; here, however, each card identifies not only the guest, the room type, the nightly rate, the total, the booking date, and the status, but also the hotel and its owner, and a search field locates a booking by hotel, owner, or guest. The essential difference from the owner's page is one of purpose: whereas the owner's reservations page offered the operations of front-desk management, such as checking guests in and out and recording payments, the administrator's page is purely a monitoring instrument and offers no such actions, since the handling of a booking belongs to the owner of the hotel concerned. The administrator's interest is in the platform as a whole, in seeing at any moment how many bookings stand at each stage of their lifecycle across every hotel.

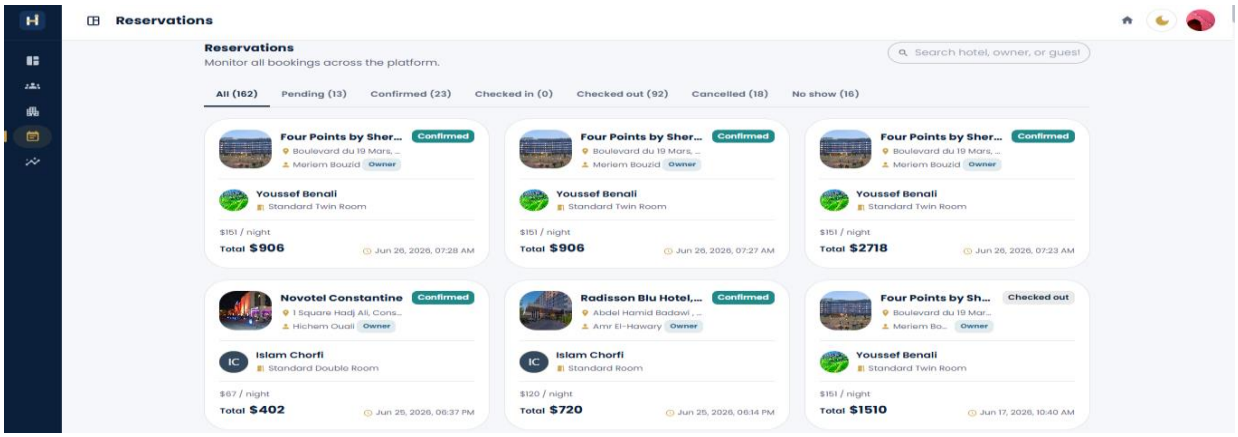


Figure 4.90*: The Platform-Wide Monitoring of Reservations*

5.4.6. AI Models Page

The AI Models page brings the platform's two ML models into the administrator's view. As shown in Figure 4.118, it presents each model, the price prediction model and the sentiment analysis model, on its own card, stating the algorithm that implements it, namely LightGBM and a fine-tuned DistilBERT respectively, the nature of its task, a short description of its purpose, the dataset on which it was trained, and the set of evaluation metrics that quantify its performance. For the price prediction model these are the regression error measures reported in Section 4, namely the test and training MAPE, the MAE, and the RMSE, supplemented by the training and prediction times; for the sentiment analysis model they are the classification measures, namely the test and training F1 scores, the accuracy, the precision, and the recall. Two summary bar charts accompany the cards, grouping the error metrics, for which lower values are better, and the performance metrics, for which higher values are better, so that the two models can be appraised at a glance. In effect, this page surfaces, within the running application, the experimental results that were established and discussed in detail in Section 4.

It should be made clear, however, that this page represents only an initial realisation of the administrator's relationship with the models. As presented here, it is a reporting view: it displays the results obtained when the models were trained and allows the administrator to read and compare them. The administrator's principal role with respect to the AI Models is intended to be more active than this. Because the platform's data is not static, as new hotels are added, booking patterns shift, and fresh reviews accumulate, the performance of models trained on earlier data may decline over time, and the appropriate response is to retrain them on more recent data. The administrator is the actor responsible for this governance: for monitoring the models' continued performance, for recognising when a model has degraded or when changes in the system warrant an update, and for triggering retraining accordingly. The present page is therefore best understood as the foundation of that role rather than its full expression, and the development of richer monitoring together with a managed retraining workflow is identified among the directions for future work discussed in Chapter 5.

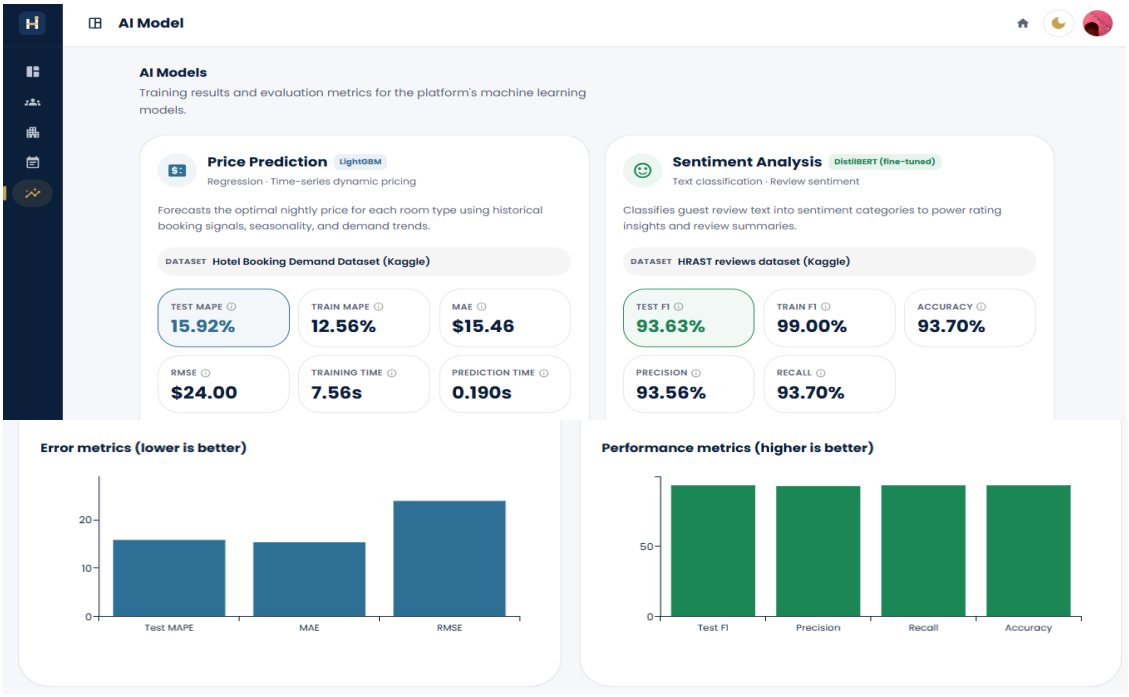


Figure 4.91: *The Training Results and Evaluation Metrics of the Price Prediction and Sentiment Analysis Models*

5.4.7. My Account Page

The administrator's My Account page, shown in Figure 4.119, completes the dashboard. It is the administrator's counterpart of the owner's account page described in Section 5.3.4 and behaves in exactly the same way: it presents the administrator's profile together with the Edit profile operation, and it carries the same access code panel, whose code is revealed only after the identity verification step and only for a limited time, precisely as described in Section 5.3.4. The page differs only in detail, the account's role being shown here as administrator.

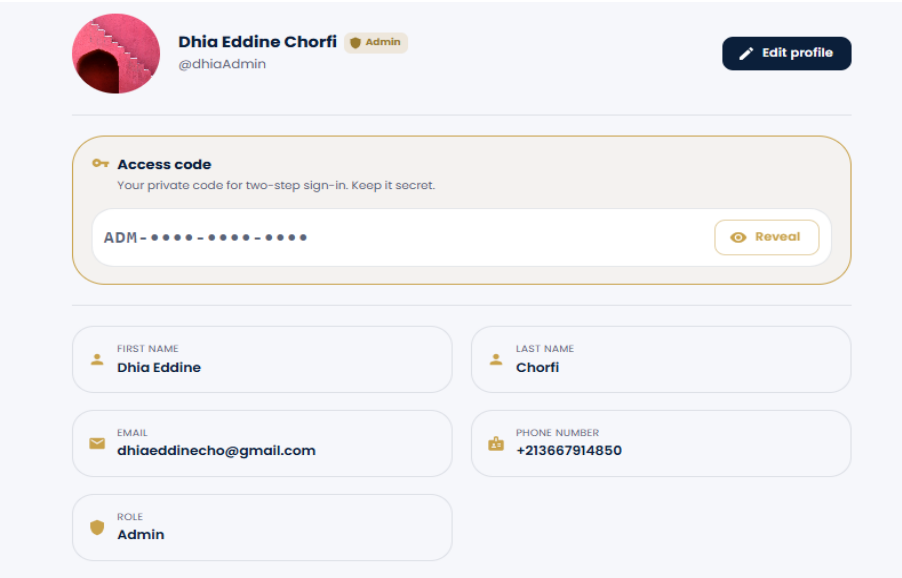


Figure 4.92: *The Administrator's Profile and the Access Code Panel*

With the administrator's dashboard, the presentation of the web application is complete. The preceding subsections have followed the system through the eyes of each of its actors in turn, namely the unauthenticated visitor, the guest, the owner, and the administrator, showing how the design of Chapter 3 and the implementation of the present chapter come together in a working application, and how the price prediction and sentiment analysis models are woven into the experience of the owner and the administrator. The remainder of this chapter turns from presentation to assessment, evaluating the system against its functional and non-functional requirements, comparing it with existing solutions, and judging the extent to which it meets the objectives set out at the start of this work.

6. System Evaluation

Having presented the design, the implementation, and the finished application, this section evaluates the system against the criteria by which its success should be judged. The evaluation proceeds in four parts. The functional evaluation verifies that the system performs the operations required of it; the non-functional evaluation assesses the qualities with which it performs them, such as security, performance, usability, and maintainability; the comparison with existing solutions situates the system among the available alternatives and clarifies the contribution it makes; and the assessment of objectives returns to the aims that motivated this work and judges the extent to which they have been achieved. Throughout, care is taken to distinguish properties that have been demonstrated or measured from those that follow from the system's design, so that the claims made are precise and defensible.

6.1. Functional Evaluation

The functional evaluation establishes whether the system provides the capabilities specified for it in the requirements analysis of Chapter 3. Because the application has been presented in full in the preceding section, this verification can be carried out directly: each functional requirement is matched against the operation that realises it in the running system, with a reference to the part of this chapter in which that operation was demonstrated. Table 4.29 sets out this mapping, organised by the actor to whom each requirement belongs.

Table 4.13: *Verification of the Functional Requirements Against the Delivered System*

Actor	Functional requirement	Status
Common	Guest self-registration and account creation	Met
Common	Authentication: login and logout	Met
Common	Password reset	Met
Common	Two-step sign-in for privileged roles via access code	Met
Common	Server-side role-based authorisation	Met

Common	Public browsing and search of hotels	Met
Guest	View hotel details, room types, and facilities	Met
Guest	View the price forecast with its explanation	Met
Guest	View the sentiment of reviews	Met
Guest	Create a booking and select a payment method	Met
Guest	View and cancel reservations	Met
Guest	Submit reviews	Met
Guest	Manage favourites	Met
Guest	Manage personal profile	Met
Owner	View the single-hotel analytics dashboard	Met
Owner	Edit hotel information	Met
Owner	Create, edit, and remove room types	Met
Owner	Accept or reject the price recommendation	Met
Owner	View guest reviews with their sentiment	Met
Owner	Manage reservations: check-in, check-out, payment, no show, refund	Met
Owner	Manage account and reveal the access code	Met
Admin	View the platform-wide analytics dashboard	Met
Admin	Manage owners: create, view, edit, remove	Met
Admin	Manage guests: view, remove	Met
Admin	Create and assign hotels to owners; view and remove hotels	Met
Admin	Monitor all reservations across the platform	Met
Admin	Review the AI model metrics	Met
Admin	Manage account	Met
System	Daily price prediction batch for every room type	Met
System	Daily update of the price history	Met
System	Hourly automatic check-out of due reservations	Met

As the table shows, every functional requirement identified in Chapter 3 is realised by a corresponding operation in the delivered system, across all of its actors, across its automated background tasks, and across both its conventional and its AI features. The system is therefore judged to be functionally complete with respect to its specification.

6.2. Non-Functional Evaluation

The non-functional evaluation concerns the quality attributes of the system, that is, the manner in which it carries out its functions. Where a claim rests on a measurement it is identified as such; otherwise the claim is to be understood as a property of the system's design and implementation rather than as an experimentally verified result.

❖ Security

Security received particular attention, since the system mediates access to personal data and to commercial controls such as pricing. Authentication is built on JSON Web Tokens (JWT) with a rotating access and refresh token scheme: user

passwords are stored only as salted bcrypt hashes, and refresh tokens only as SHA-256 hashes held in documents carrying a time-to-live index, so that a compromise of the store does not expose usable credentials and expired tokens are removed automatically. Authorisation is enforced on the server by RBAC middleware, so that each request is checked against the permissions of its role rather than relying on the interface to conceal unavailable actions. The privileged roles, namely the owner and the administrator, are further protected by the two-step sign-in, in which a secret access code supplements the password, and the access code itself is disclosed only after a fresh password verification and only for a limited time, as described in Section 5.3.4. Standard hardening measures, including HTTP security headers and a restrictive cross-origin policy, complete the design. Collectively these mechanisms address the principal authentication and authorisation risks by construction.

❖ Performance and Responsiveness

The system is designed to remain responsive in normal use. The frontend is a single-page application (SPA), so that navigation updates the view without full page reloads, and it communicates with the backend through a concise REST interface. On the AI side, two deliberate choices prevent the models from impeding the user: the price prediction model is served from memory, loaded once when the service starts rather than on each request, and its forecasts are produced by a scheduled daily batch rather than synchronously during an interaction, while sentiment analysis is performed when a review is submitted along a path that does not block the core reservation flow. The prediction time of the price model, reported in Section 4 as approximately 0.19 seconds, is a measured indication that inference is fast enough not to constitute a bottleneck. Beyond these properties, no formal load testing was conducted; claims about behaviour under heavy concurrent load are therefore not made here, and the architecture's suitability for scaling is treated separately below.

❖ Usability

The application aims to be clear and consistent to use. A single visual language, namely a restrained navy and gold palette with the Poppins typeface, is applied across every interface, and the layout adapts to the size of the screen, so that the system is usable on mobile devices as well as on the desktop. Each role is given an interface shaped to its needs rather than a single undifferentiated view: the guest is offered an exploratory, content-rich experience, while the owner and the administrator are given denser, analytical workspaces. Feedback is immediate and uniform, with success and failure reported through a common notification mechanism and destructive actions guarded by confirmation dialogs. Of particular note for usability is the treatment of the AI features: rather than presenting the price recommendation as an opaque number, the interface accompanies it with a trend, a magnitude, and an on-demand explanation of the factors behind it, and it

leaves the final decision with the owner; similarly, review sentiment is surfaced legibly alongside each review. This emphasis on transparency is intended to make the system's intelligent features intelligible and trustworthy to non-technical users.

❖ Maintainability and Modularity

The system is structured to be maintainable and extensible. It is divided into independent services, namely a frontend, a backend, and a separate AI service, that communicate over well-defined interfaces, so that each may be developed, tested, and replaced in isolation. Within the frontend, presentation is separated from logic by externalising component styles into dedicated files and by centralising state management; within the backend, the REST API isolates the data and business logic from the clients that consume them. The decoupling of the AI service is especially valuable, since it allows a model to be retrained and redeployed without disturbing the rest of the system, which is the very capability on which the administrator's intended model-governance role, described in Section 5.4.6, depends.

❖ Scalability and Deployability

The architecture is conducive to scaling and to reliable deployment, although scaling was not exercised experimentally in this work. Authentication is stateless, relying on signed tokens rather than on server-side sessions, which in principle allows the backend to be replicated across instances without shared session state. The separation of the AI service from the backend means that the heavier work of model serving can be scaled independently of the core application. The whole system is containerised and orchestrated with Docker Compose, with inter-service communication addressed by service name rather than by a fixed host, so that the same configuration runs consistently across development and deployment environments and individual services can be rebuilt or relocated without reconfiguring their neighbours.

❖ Reliability

The system is designed to degrade gracefully where its intelligent features are concerned. Because sentiment analysis and price forecasting run on paths that are separate from, and non-blocking with respect to, the core reservation workflow, a fault or a delay in the AI service does not prevent guests from browsing and booking or owners from managing their hotels. The conventional and the intelligent parts of the system are thus loosely coupled, which limits the effect of a failure in one upon the other.

6.3. Comparison with Existing Solutions

To clarify the contribution of the system, it is useful to situate it among the kinds of solution already available to hotels and their guests. Three broad categories are relevant. The first is the large OTA, such as Booking.com or Expedia, through which guests discover and book hotels at very large scale. The second comprises the operational and commercial tools used by hotels themselves: the PMS, which administers rooms, reservations, and guests, and the revenue management system (RMS), which advises on pricing. The third is the separate class of online reputation tools that aggregate and analyse guest reviews. These categories are mature and robust and, in the case of the commercial platforms, operate at a scale far beyond that of an academic prototype; the present comparison therefore does not concern scale, market readiness, or breadth of integration, for which these solutions are plainly superior, but rather the way in which capabilities are combined and made available to the individual hotel owner.

Viewed in this light, the distinguishing characteristic of the proposed system is integration. The capabilities that a small hotel would otherwise obtain, if at all, from several separate and often costly products are here combined within a single platform: reservation management, an owner-facing analytical dashboard, an AI price recommendation, and review sentiment analysis coexist in one coherent system organised around a unified model of three roles. Two features distinguish the system even from the tools that offer comparable functions individually. The first is the explainability of its pricing: where commercial pricing advice is typically delivered as an opaque figure, the system presents each recommendation with a trend, a magnitude, and an on-demand justification, and it preserves the owner's authority by requiring an explicit acceptance. The second is the native integration of review sentiment into the same interfaces in which owners and administrators already work, rather than as a separate reputation product. Table 4.30 summarises this qualitative comparison across the dimensions most relevant to the system's purpose.

Table 4.14: *Qualitative Comparison of the Proposed System with Existing Categories of Solution*

Dimension	Large OTAs	Hotel PMS / RMS	Proposed system (Hotelify)
Reservation management	Yes, traveller-facing, very large scale	Yes, operational	Yes, integrated
Owner-facing integrated analytics	Limited partner dashboards	Yes	Yes
AI price recommendation	Generally not exposed to the individual owner	Often available, usually as a separate paid tool	Built in
Explainability of the price recommendation	Typically opaque	Typically opaque	Trend, magnitude, and on-demand reasons

Owner control over pricing decision	Limited	Varies	Explicit accept or reject
Integrated review sentiment analysis	Reviews shown, sentiment not typically surfaced per review to the owner	Usually a separate reputation product	Native, per review and in aggregate
Unified guest, owner, and administrator platform	Separate applications	Staff-facing	One coherent system
Architecture	Proprietary, closed	Proprietary, closed	Open full-stack implementation
Primary scope and scale	Global, very large	Established commercial	Academic prototype, one hotel per owner

The comparison is necessarily qualitative, since the internal workings and current feature sets of proprietary commercial systems are not available for direct measurement; its purpose is to position the system rather than to benchmark it. So positioned, the contribution of the system is the delivery, in an open and integrated full-stack implementation, of intelligent and transparent decision support to the individual hotel owner.

6.4. Achievement of Objectives

The final element of the evaluation returns to the objectives that motivated this work and assesses, objective by objective, the degree to which each has been met; these objectives correspond to the aims stated in the general introduction.

- ✓ **The First Objective**

  Was to design and implement a complete, full-stack web application for hotel reservation, rather than an isolated component. This objective has been fully achieved: the delivered system comprises a react frontend, a node.js and express backend, a separate AI service, and a mongodb database, integrated and orchestrated with docker compose, and it supports the entire reservation process from discovery through booking to the post-stay review.

- ✓ **The Second Objective**

  Was to support the distinct actors of a reservation platform under appropriate access control. This too has been achieved: the system implements three roles, namely guest, owner, and administrator, each with an interface tailored to its responsibilities, governed by server-side RBAC and, for the privileged roles, by a two-step sign-in.

✓ The Third Objective

Was to enhance the platform with a price prediction and recommendation capability. At the level of design and integration this objective has been met: a regression model was trained, evaluated, and embedded in the system, where it produces, for each room type and each future date, a recommended price together with a trend and an explanation that the owner may accept or reject, and its offline evaluation, reported in section 4, yielded strong error metrics. It must be stated precisely, however, that a target leakage issue identified during development affects the reliability of the model's predictions at inference time, so that the strength of the offline metrics overstates the quality of the recommendations actually produced in operation. The nature of this issue and the means of addressing it are examined in the limitations and future work of chapter 5. The capability is therefore delivered and fully integrated, but its predictive quality is, at present, a known limitation rather than a settled result.

✓ The Fourth Objective

Was to incorporate sentiment analysis of guest reviews. This objective has been fully and robustly achieved: a fine-tuned distilbert model, trained with focal loss to counter the class imbalance of the data, classifies each review and attains strong and well-balanced test metrics, as reported in section 4; and its output is integrated throughout the system, both per review and in aggregate, for guests, owners, and administrators alike.

✓ A Fifth Objective

Was to give owners and administrators the means to understand and oversee their operations through integrated analytics. This has been achieved through the owner and administrator dashboards presented in sections 5.3 and 5.4, which combine operational, commercial, and AI insights in a single view at the level of a single hotel and at the level of the whole platform respectively.

✓ A Final Objective

Concerned the engineering qualities of the system: that it should be secure, modular, maintainable, and deployable. As argued in the non-functional evaluation above, these qualities are reflected in concrete design and implementation choices, from the token-based security model and the RBAC to the service-oriented architecture and the containerised deployment.

Taken together, the objectives of this work have been substantially achieved. The system is functionally complete, soundly engineered, and distinguished by its integrated and transparent use of AI; the principal qualification to this assessment is the predictive quality of the price prediction model, which remains a known limitation and the foremost subject of the future work set out in Chapter 5.

7. Conclusion

This chapter has documented the realisation of the system in full, carrying the design of Chapter 3 through to a complete and working application. It began by identifying the tools and technologies with which each part of the system was built, and then described the methodology by which the database, the backend, the AI service, and the frontend were implemented and integrated into a single, containerised whole.

The chapter then reported the training and experimental results of the two AI models. The price prediction model, a LightGBM regressor selected through a composite scoring procedure and interpreted with the aid of SHAP analysis, was trained and evaluated and yielded strong offline error metrics; the sentiment analysis model, a fine-tuned DistilBERT trained with focal loss to address the imbalance of its data, attained strong and balanced classification metrics. The finished application was then presented in detail, interface by interface and operation by operation, showing how the unauthenticated visitor, the guest, the owner, and the administrator each experience the system, and how the price recommendation and the review sentiment are surfaced, transparently and under the user's control, within the interfaces of the owner and the administrator.

Finally, the system was evaluated. It was shown to be functionally complete with respect to its specification, to embody sound engineering qualities in its security, its modularity, and its deployment, and to occupy a distinctive position among existing solutions through its integrated and explainable use of AI. The evaluation also established, with precision, the principal qualification to these results: while the sentiment analysis model is robust, the price prediction model is affected by a target leakage issue that limits the reliability of its predictions in operation, notwithstanding its strong offline metrics.

The system delivered in this chapter therefore meets the objectives set for it, with that single, clearly identified reservation. The general conclusion that follows draws together the contributions of the work as a whole and examines this limitation, alongside the other directions for future development, in its treatment of the limitations and future work.

CHAPTER 05



GENERAL CONCLUSION

1. Conclusion

This work set out to address a clear problem: the absence of an integrated, transparent, and accessible platform that serves, within a single system, the interdependent needs of guests, hotel owners, and administrators, uniting the conventional functions of a reservation system with intelligent and explainable decision support. The preceding chapters have, in turn, framed that problem, grounded it in the existing body of knowledge, designed a system to solve it, and built and evaluated that system.

Chapter 1 established the context, the motivation, and the objectives of the work. Chapter 2 reviewed the background and the related literature on reservation systems, dynamic pricing, and sentiment analysis. Chapter 3 analysed the requirements of the platform and designed it in full, from its actors and use cases, through its data and architecture, to the design of its AI pipelines and its user interfaces. Chapter 4 then carried that design into a working system: it implemented and integrated the application's parts, trained and evaluated its two AI models, presented the finished application through the eyes of each of its actors, and evaluated the completed system against its requirements and objectives.

The outcome is Hotelify, a complete, full-stack web application for hotel reservation in which a price prediction and recommendation capability and a review sentiment analysis capability are integrated, transparently and under the user's control, around a unified model of three roles. As established in the evaluation of Chapter 4, the system is functionally complete and soundly engineered, and the objectives set for the work have been substantially achieved, subject to the single, clearly identified reservation concerning the predictive quality of the price prediction model. The work therefore succeeds in delivering, in an open and integrated implementation, the kind of intelligent and transparent decision support that motivated it.

2. Contributions

The principal contribution of this work is the design and realisation of a single, coherent platform that integrates capabilities which are ordinarily fragmented across separate products. Within one system, and around a unified model of the guest, the owner, and the administrator, Hotelify combines the conventional functions of hotel reservation and management with two distinct AI capabilities, namely the prediction and recommendation of room prices and the analysis of review sentiment.

Beyond integration, two further contributions deserve emphasis. The first is the transparency of the system's intelligent guidance: rather than issuing pricing advice as an opaque figure, the platform accompanies each recommendation with a trend, a magnitude, and an on-demand explanation, and it preserves the owner's authority by requiring an explicit decision, so that the AI informs rather than dictates. The second is the delivery of these capabilities as a complete, open, full-stack implementation, in which two separately developed models, a gradient boosting regressor and a fine-tuned

transformer, are not merely trained in isolation but woven meaningfully into a deployed, role-based application. Together, these contributions place intelligent and intelligible decision support at the service of the platform's users as a whole, helping guests towards better booking decisions, supporting owners in the management of their hotels, and giving administrators oversight of the system and of its models.

3. Limitations

Notwithstanding these contributions, the work has limitations that should be stated plainly. The foremost of these concerns the price prediction model. As reported in the evaluation of Chapter 4, a target leakage issue, arising during training from a feature that was derived from the prediction target itself, inflated the model's apparent accuracy and, at inference time, caused its predictions to collapse towards a narrow range. In consequence, although the model's offline error metrics are strong, the quality of the price recommendations it produces in operation is materially weaker than those metrics suggest. This is the principal limitation of the system, and it qualifies the achievement of the price prediction objective.

Several further limitations are more modest in nature. The system was not subjected to formal load testing, so its behaviour under heavy concurrent use has not been measured, and claims of scalability rest on the properties of its architecture rather than on experiment. The booking process records a payment method and status but is not connected to a real payment gateway, payment being represented at the level of the model rather than executed. The sentiment analysis model, while robust, was evaluated on a dataset whose neutral class was sparsely represented, which constrains confidence in its performance on that class in particular. The system also does not implement any automatic retraining of its models: the two models are trained once and thereafter remain fixed, and the platform neither detects a decline in their performance nor responds to changes in its data by retraining them; the administrator's relationship with the models, as delivered, is one of monitoring their reported metrics rather than of automated governance, and a richer monitoring and retraining mechanism is proposed as future work rather than provided as an implemented feature. Finally, the system embodies the deliberate design choice that each owner is bound to a single hotel, which suits the project's scope but would require extension to serve operators of multiple properties.

4. Future Work

These limitations indicate, in turn, the most immediate directions for future work. The first and most important is to resolve the price prediction model's target leakage: by removing or correctly constructing the offending feature, redesigning the time-dependent features so that they draw only on information genuinely available at prediction time, and retraining and re-evaluating the model so that its operational

predictions match the quality promised by its offline metrics. This is the foremost priority, since it concerns the system's central intelligent capability.

Building upon the administrator's role described in Chapter 4, a natural further direction is to develop the governance of the models into a managed workflow: continuous monitoring of their performance in operation, the detection of degradation as the platform's data evolves, and a controlled mechanism for retraining and redeploying them, so that the models remain accurate over time rather than fixed at the moment of their first training. Beyond the models, the system would be strengthened by integration with a real payment gateway, by formal load and scalability testing to substantiate its architectural promise, by enriching the sentiment dataset, particularly its neutral class, and by extending the data model to allow an owner to manage more than one property. Richer forms of recommendation, drawing on the demand signals that the platform already accumulates, offer a further avenue for investigation. A further, forward-looking direction is to enrich the platform with an additional and interactive form of AI in the shape of a conversational assistant, or chatbot. Such an assistant would help all of the system's users, namely guests, owners, and administrators alike, to understand the platform, to reach better decisions, and to interact with it more naturally, by responding to questions in plain language and guiding each user through the tasks relevant to their role.

In closing, this work has shown that the conventional and the intelligent functions of a hotel reservation platform can be united, transparently and accessibly, within a single full-stack application. The system delivered here meets the objectives that motivated it, carries a clearly identified limitation that defines its most pressing future work, and stands, taken as a whole, as a complete and coherent demonstration of how AI can be brought, intelligibly and under the user's control, into the service of the guests, owners, and administrators who together make up a hotel reservation platform.

References

[1] R. Bemile, A. Achampong and E. Danquah, “Online Hotel Reservation System,” *Ijiset,* vol. 1, no. 9, p. 1–12, 2019.

[2] N. Vrček, "From the editor," *J. Inf. Organ. Sci,* vol. 33, no. 2, p. 41–54, 2009.

[3] J. Pandţa, L. Đeri, A. Galamboš and T. Galamboš, “Two-factor Analysis of Employee Motivation at Postal Traffic – Department in Novi Sad,” *European,* vol. 12, no. 2, pp. 15-25, 2015.

[4] A. Gomaa and A. E.-M. Ahmed, “Understanding the determinants of hotel booking intentions and moderating role of habit,” *Int. J. Hosp. Manag.,* vol. 54, p. 52–67, 2016.

[5] M. Nason, "What Is Hotel Reservation | TouristSecrets," TouristSecrets, 08 November 2023. [Online]. Available: https://www.touristsecrets.com/travel-guide/hotels/what-is-hotel-reservation/.

[6] H. Jordan, "What is a Hotel Reservation System?," HotelTechReport, 13 November 2024. [Online]. Available: https://hoteltechreport.com/news/hotel-reservations-system.

[7] D. Elphick, "Your complete guide to types of hotel rooms," SiteMinder, 14 September 2022. [Online]. Available: https://www.siteminder.com/r/hotel-room-types/.

[8] L. Liuyi, D. Yufeng, G. Xiaolong and L. Liang, "Availability management of hotel rooms under cooperation with online travel agencies," *International Journal of Hospitality Management,* vol. 50, pp. 145-152, 2015.

[9] S. Vaishnavi, K. A. Reddy and T. Mahenderker, “Online Hotel Reservation System,” *IJRASET,* vol. 13, no. 4, pp. 1542-1547, 2025.

[10] S. Colcol, "How Hotel Reservation Systems Work," Little Hotelier, 2016. [Online]. Available: https://www.littlehotelier.com/blog/learn-about-tech/hotel-reservation-systems-work/.

[11] D. Bansal, "How do Hotel Reservation Systems work? It's types & benefits," aperfectstay, 10 May 2024. [Online]. Available: https://aperfectstay.ai/blog/hotel-reservation-system.

[12] M. Takahashi, "The Best Hotel Reservation System: Key Features and Benefits Explained," prostay, 13 August 2025. [Online]. Available: https://www.prostay.com/blog/best-hotel-reservation-system-key-features-benefits/.

[13] G. Myskiv and S. Nycz-Wojtan, “Reservation systems as a tool of tourist services marketing,” *Economics, Entrepreneurship, Management,* vol. 9, no. 2, pp. 7-18, 2022.

[14] O. Vlahović, Ž. Rađenović, D. Perović, V. Vujačić and K. Davidović, "Digital Transformation in Tourism: The Role of E-Booking Systems," *Croatian Regional Development Journal,* vol. 5, no. 2, pp. 129-145, 2024.

[15] C. Shine, "Hotel Reservation System: Software Guide for 2026," SiteMinder, 16 January 2026. [Online]. Available: https://www.siteminder.com/r/hotel-reservation-system/.

[16] A. A. Stephan, "The Evolution of Hotel Room Reservation Systems,from Pencil and Paper to Online Reservation System," *IJARSCT,* vol. 2, no. 4, pp. 358-360, 2022.

[17] M. Deputat, M. Podolian, V. Zhupnyka, K. Terletska and P. Gorishevskyy, "Evolution of information systems and technologies in the hospitality and tourism sector: A historical perspective," *Multidisciplinary Science Journal,* vol. 6, no. 0729, 2024.

[18] S. B. Ibirinoye-Abdulkareem, k. J. Adedotun, j. O. Adekunle and A. K. Raji, "Ai-Powered Website Design for Hotel Booking Systems Improving User Experience and Operational Efficiency," *International Journal of African Innovation & Multidsciplinary Res.,* vol. 8, no. 2, pp. 133-146, 2025.

[19] D. Jurafsky and J. H. Martin, Speech and Language Processing, Stanford, CA, USA, 2023.

[20] J. Hollander, "What is a Booking Engine? Complete Guide to Hotel Booking Systems," HotelTechReport, 21 March 2025. [Online]. Available: https://hoteltechreport.com/news/hotel-booking-engine-guide.

[21] M. Dawson, "Central Reservation System (CRS): Guide for hotels," SiteMinder, 25 June 2025. [Online]. Available: https://www.siteminder.com/r/central-reservation-system/.

[22] S. Colcol, "What is PMS integration? How it works for hotels," SiteMinder, 11 April 2024. [Online]. Available: https://www.siteminder.com/r/pms-integration/.

[23] " A Quick Guide to Payment Gateway Integration with Your Hotel PMS," WebRezPro, 12 July 2018. [Online]. Available: https://webrezpro.com/a-quick-guide-to-payment-gateway-integration-with-your-hotel-pms/.

[24] M. lozano, "10 Best Hotel Business Intelligence Tools," Room Master, 25 December 2024. [Online]. Available: https://www.roommaster.com/blog/hotel-business-intelligence-software.

[25] Minal, "How to Design a Database for Booking and Reservation Systems," GeeksforGeeks, 23 July 2025. [Online]. Available: https://www.geeksforgeeks.org/dbms/how-to-design-a-database-for-booking-and-reservation-systems/.

[26] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, “Role-based access control models,” *IEEE Computer,* vol. 29, no. 2, pp. 38-47, 1996.

[27] T. Mitchell and M. Hill, Machine Learning, New York, NY, USA: McGraw-Hill, 1997.

[28] Z. Doborjehr, N. Hemmington, M. Doborjeh and N. Kasabov, "Artificial intelligence: a systematic review of methods and applications in hospitality and tourism," *International Journal of Contemporary Hospitality Management,* vol. 34, no. 3, p. 1154–1176, 2022.

[29] H. Henriques and L. N. Pereirsa, "Hotel demand forecasting models and methods using artificial intelligence: a systematic literature review," *Tourism & Management Studies,* vol. 20, no. 3, p. 39–51, 2024.

[30] W. Ming and T. W. Y. Billy, "Artificial intelligence research in tourism and hospitality journals: trends, emerging themes, and the rise of generative AI," *Tourism and Hospitality,* vol. 6, no. 2, p. 63, 2025.

[31] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Proc. 1st Int. Workshop Multiple Classifier Systems (MCS)*, Cagliari, Italy, 2000.

[32] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, p. 436–444, 2015.

[33] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist,* vol. 29, no. 5, pp. 1189-1232 , 2001.

[34] G. Ke, Q. Men, F. Thomas, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017.

[35] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016.

[36] O. Francisco B, S.-F. Moesio W, B. Gabriel A, F. João Paulo, C. Penna and M. Péricles B. C, "Machine learning and time series analysis to forecast hotel room prices," in *Lecture Notes in Networks and Systems*, Springer, 2025.

[37] R. Das, H. Chadha and S. Banerjee, “Multi-layered market forecast framework for hotel revenue management by continuously learning market dynamics,” *Journal of Revenue and Pricing Management,* vol. 20, p. 351–367, 2021.

[38] A. Ameur, S. Hamdi and S. Ben Yahia, “Sentiment Analysis for Hotel Reviews: A Systematic Literature Review,” *ACM Computing Surveys,* vol. 26, no. 51, pp. 1-38, 2023.

[39] "Support-vector networks," *Mach. Learn,* vol. 20, no. 3, p. 273–297, 1995.

[40] I. Rish, "An empirical study of the naive Bayes classifier," in *Proc. IJCAI Workshop Empirical Methods Artif. Intell.*, Seattle, WA, USA, 2001.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All you Need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017.

[42] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (NAACL-HLT)*, Minneapolis, MN, USA, 2019.

[43] Y. Wen, Y. Liang and X. Zhu, “Sentiment analysis of hotel online reviews using the BERT model and ERNIE model—Data from China,” *PLOS ONE,* vol. 18, no. 3, pp. 1-14, 2023.

[44] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, p. 1735–1780, 1997.

[45] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," IETF, RFC 7519, 2015.

[46] T. Lodderstedt, J. Bradley, A. Labunets and D. Fett, "Best Current Practice for OAuth 2.0 Security," IETF, RFC 9700, 2025.

[47] National Institute of Standards and Technology, "Secure Hash Standard (SHS)," NIST, Gaithersburg, MD, USA, 2015.

[48] J. Gray, "The transaction concept: Virtues and limitations," in *Proc. 7th Int. Conf. Very Large Data Bases (VLDB)*, Cannes, France, 1981.

[49] OWASP Foundation, "OWASP Secure Headers Project," 2023. [Online]. Available: https://owasp.org/www-project-secure-headers/.

[50] A. van Kesteren, "Cross-Origin Resource Sharing," W3C Recommendation, 2014.

[51] G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language User Guide,

USA: Addison Wesley, 1998, p. 512.

[52] MKLab, "StarUML Documentation," 2024. [Online]. Available: https://docs.staruml.io/.

[53] J. S. Pramod and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Boston, MA, USA: Addison-Wesley, 2012.

[54] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM,* vol. 13, no. 6, p. 377–387, 1970.

[55] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM Transactions on Database Systems (TODS),* vol. 1, no. 1, pp. 9-36, 1976.

[56] dbdiagram.io, "Database Relationship Diagrams Design Tool," dbdiagram.io, 2024. [Online]. Available: https://dbdiagram.io/.

[57] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice, Boston, MA, USA: Addison-Wesley, 2012.

[58] Mermaid Contributors, "Mermaid — Diagramming and Charting Tool," 2024. [Online]. Available: https://mermaid.js.org/.

[59] N. Antonio, A. Ana de and L. Nunes, "Hotel booking demand datasets," *Data in Brief,* vol. 22, pp. 41-49, 2019.

[60] C. Andreou, N. Tsapatsoulis and V. Anastasopoulou, "A Dataset of Hotel Reviews for Aspect-Based Sentiment Analysis and Topic Modeling," in *IEEE*, 2023.

[61] L. Breiman, "Random Forests," *Mach. Learn,* vol. 45, no. 1, p. 5–32, 2001.

[62] C. Bergmeir and B. José M, “On the use of cross-validation for time series predictor evaluation,” *Information Sciences,* vol. 191, p. 192–213, 2012.

[63] Y. B. James Bergstra, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res (JMLR),* vol. 13, no. 10, p. 281–305, 2012.

[64] M. Arnaud de, B. Golden, G. Bénédicte Le and F. Rossi, “Mean Absolute Percentage Error for regression models,” *Neurocomputing,* vol. 192, pp. 38-48, 2016.

[65] W. Cort J and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res,* vol. 30, pp. 79-82, 2005.

[66] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[67] L. Prechelt, "Early Stopping - But When?," in *Neural Networks: Tricks of the Trade*,

Berlin, Germany: Springer, 2002.

[68] H. Arthur E and K. Robert W, “Ridge Regression: Biased Estimation for Nonorthogonal Problems,” *Technometrics,* vol. 12, no. 1, pp. 55-67, 1970.

[69] V. Sanh, L. Debu, J. Chaumond and T. Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,* 2019.

[70] L. Tsung-Yi, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Venice, Italy, 2017.

[71] P. David Martin Ward, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *J. Mach. Learn. Technol.,* vol. 2, no. 1, p. 37–63, 2011.

[72] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.,* vol. 45, no. 4, p. 427–437, 2009.

[73] Microsoft Corp, "Visual Studio Code," code.visualstudio.com, 2024. [Online]. Available: https://code.visualstudio.com/.

[74] Software Freedom Conservancy, "Git," git-scm.com, 2024. [Online]. Available: https://git-scm.com/.

[75] G. Inc, "GitHub," github.com, 2024. [Online]. Available: https://github.com/.

[76] I. Docker, "Docker," docker.com, 2024. [Online]. Available: https://www.docker.com/.

[77] I. Docker, "Docker Compose Overview," 2024. [Online]. Available: https://docs.docker.com/compose/.

[78] I. npm, "npm," npmjs.com, 2024. [Online]. Available: https://www.npmjs.com/.

[79] Python Packaging Authority, "pip," pip.pypa.io, 2024. [Online]. Available: https://pip.pypa.io/.

[80] I. Postman, “Postman,” postman.com, 2024. [Online]. Available: https://www.postman.com/.

[81] I. Meta Platforms, "React," react.dev, 2024. [Online]. Available: https://react.dev/.

[82] Redux Toolkit Team, "Redux Toolkit," redux-toolkit.js.org, 2024. [Online]. Available: https://redux-toolkit.js.org/.

[83] MUI Team, "Material UI," mui.com, 2024. [Online]. Available: https://mui.com/.

[84] M. Michael S and P. Josh C, Single Page Web Applications, Shelter Island, NY, USA:

Manning Publications, 2013.

[85] I. Remix Software, "React Router," reactrouter.com, 2024. [Online]. Available: https://reactrouter.com/.

[86] Axios Contributors, "Axios," axios-http.com, 2024. [Online]. Available: https://axios-http.com/.

[87] OpenJS Foundation, "Node.js," nodejs.org, 2024. [Online]. Available: https://nodejs.org/.

[88] OpenJS Foundation, "Express," expressjs.com, 2024. [Online]. Available: https://expressjs.com/.

[89] F. Roy Thomas, *Architectural Styles and the Design of Network-based Software Architectures,* Irvine, CA, USA: Ph.D. dissertation, 2000.

[90] bcrypt.js Contributors, "bcrypt," npmjs.com, 2024. [Online]. Available: https://www.npmjs.com/package/bcrypt.

[91] Helmet.js Contributors, "Helmet," helmetjs.github.io, 2024. [Online]. Available: https://helmetjs.github.io/.

[92] node-cron Contributors, "node-cron," npmjs.com, 2024. [Online]. Available: https://www.npmjs.com/package/node-cron.

[93] S. Ramírez, "FastAPI," fastapi.tiangolo.com, 2024. [Online]. Available: https://fastapi.tiangolo.com/.

[94] L. Encode OSS, "Uvicorn," uvicorn.org, 2024. [Online]. Available: https://www.uvicorn.org/.

[95] L. Encode OSS, "ASGI Specification," 2024. [Online]. Available: https://asgi.readthedocs.io/.

[96] I. Kaggle, "Kaggle," kaggle.com, 2024. [Online]. Available: https://www.kaggle.com/.

[97] I. Kaggle, "kagglehub," GitHub.com, 2024. [Online]. Available: https://github.com/Kaggle/kagglehub.

[98] pandas Development Team, "pandas," pandas.pydata.org, 2024. [Online]. Available: https://pandas.pydata.org/.

[99] NumPy Developers, "NumPy," numpy.org, 2024. [Online]. Available: https://numpy.org/.

[100] Matplotlib Development Team, "Matplotlib," matplotlib.org, 2024. [Online].

Available: https://matplotlib.org/.

[101] scikit-learn Developers, "scikit-learn," scikit-learn.org, 2024. [Online]. Available: https://scikit-learn.org/.

[102] Hugging Face, "Transformers," 2024. [Online]. Available: https://huggingface.co/docs/transformers.

[103] PyTorch Foundation, "PyTorch," pytorch.org, 2024. [Online]. Available: https://pytorch.org/.

[104] I. MongoDB, "MongoDB Manual," 2024. [Online]. Available: https://www.mongodb.com/docs/manual/.

[105] Mongoose Contributors, "Mongoose," mongoosejs.com, 2024. [Online]. Available: https://mongoosejs.com/.

[106] I. MongoDB, "mongoimport," 2024. [Online]. Available: https://www.mongodb.com/docs/database-tools/mongoimport/.

[107] Anthropic, "Claude," anthropic.com, 2024. [Online]. Available: https://www.anthropic.com/claude.

الجمهورية الجزائرية الديمقراطية الشعبية

Republique Algérienne Démocratique Populaire
وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur de la Recherche Scientifique

Université 20 août 1955 Skikda — **جامعة 20 أوت 1955 – سكيكدة**

Faculte des Sciences
Departement d'informatique

كلية العلوم
قسم الإعلام الآلي

الرقم: .... / ق/ أ/ أ/ ك.ع/ 2026

Autorisation de Dépôt de Mémoire de Master

Je soussigné : .....Lahsasna Adel.....

Certifie que l'étudiant(e) : .....Chorfi Dhia Eddine.....

Spécialité : .....Intelligence Artificielle (IA).....

Ayant soutenu le projet intitulé : ...AI-Powered Web Application for Hotel Reservation System...

A apporté les corrections nécessaires sur son manuscrit de Master.

Signature de l'encadreur

تم المسح الضوئي بـ CamScanner