PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF 20 AOÛT 1955 - SKIKDA

Faculty of Sciences

Department of Computer Science

Dissertation Submitted in partial fulfillment of the requirements for the Master's Degree in Computer Science

Specialization: Advanced Software Engineering and Applications

Title:

EduTwin AI

A Learner-Centric Digital Twin and Retrieval-Augmented Generation platform for Personalized Higher Education.

Presented by:

Ramdane Abdeldjalil

Guerriche Ala Abdel Moumene

Supervised by:

Dr. Kerraoui Sarra

2025 – 2026



Acknowledgements

First and foremost, we express our deepest gratitude to **Almighty God** for His guidance and countless blessings throughout this journey. Without His divine support, our success would not have been possible.

We would also like to thank our supervisor, **Dr. Kerraoui Sarra**, for her knowledge, experience, advice, and guidance, thanks to which we were able to produce high-quality work in such a short time.

We would like to thank our families for their unwavering psychological support, encouragement, and the resources they provided, all of which played a vital role in our success.

We would like to thank all the teaching staff of the computer science department for their dedication and work, which has contributed to our learning.



Abstract

The rapid evolution of digital education has exposed the limitations of traditional, "one-size-fits-all" learning management systems, highlighting an urgent need for intelligent, adaptive environments. This thesis presents EduTwin AI, an innovative platform designed to bridge the gap between static academic resources and the dynamic cognitive state of the learner. By natively integrating a Learner-Centric Digital Twin with a Retrieval-Augmented Generation (RAG) architecture, the system creates a continuous physical-digital feedback loop. The Digital Twin employs Bayesian Knowledge Tracing (BKT) to autonomously model and predict the student's concept mastery in real time. Simultaneously, the RAG pipeline acts as a strict pedagogical filter, ensuring that all interactions generated by the Large Language Model (LLM) are exclusively anchored to validated course materials, thereby eliminating AI hallucinations. Through adaptive assessments, personalized study paths, and a conversational tutor, EduTwin AI dynamically calibrates its complexity to match the student's Zone of Proximal Development. This work demonstrates the profound potential of combining real-time cognitive tracking with generative AI to deliver a truly personalized, secure, and academically rigorous learning experience.

Keywords:

Artificial Intelligence (AI), Learner Digital Twin, Retrieval-Augmented Generation (RAG), Bayesian Knowledge Tracing (BKT), Large Language Models (LLMs), Personalized Learning.

Résumé

L'évolution rapide de l'éducation numérique a mis en évidence les limites des systèmes de gestion de l'apprentissage traditionnels, soulignant le besoin urgent d'environnements intelligents et adaptatifs. Ce mémoire présente EduTwin AI, une plateforme innovante conçue pour combler le fossé entre les ressources académiques statiques et l'état cognitif dynamique de l'apprenant. En intégrant nativement un Jumeau Numérique centré sur l'apprenant à une architecture de Génération Augmentée par la Recherche (RAG), le système crée une boucle de rétroaction continue. Le Jumeau Numérique utilise le traçage bayésien des connaissances (BKT) pour modéliser et prédire de manière autonome la maîtrise des concepts par l'étudiant en temps réel. Simultanément, le pipeline RAG agit comme un filtre pédagogique strict, garantissant que toutes les interactions générées par le Grand Modèle de Langage (LLM) sont exclusivement ancrées dans les supports de cours validés, éliminant ainsi les hallucinations de l'IA. À travers des évaluations adaptatives, des parcours d'étude personnalisés et un tuteur conversationnel, EduTwin AI calibre dynamiquement sa complexité pour correspondre à la zone de développement proximal de l'étudiant. Ce travail démontre le potentiel profond de la combinaison du suivi cognitif en temps réel avec l'IA générative pour offrir une expérience d'apprentissage véritablement personnalisée, sécurisée et académiquement rigoureuse.

Mots-clés:

Intelligence Artificielle (IA), Jumeau Numérique de l'Apprenant, Génération Augmentée par la Recherche (RAG), Traçage Bayésien des Connaissances (BKT), Grands Modèles de Langage (LLM), Apprentissage Personnalisé.

الملخص

كشف التطور السريع في مجال التعليم الرقمي عن قصور أنظمة إدارة التعلم التقليدية، مما أبرز الحاجة الملحة لبيئات تعليمية ذكية وعالية التكيف. تقدم هذه المذكرة EduTwin AI، وهي منصة مبتكرة صُممت لسد الفجوة بين الموارد الأكاديمية الثابتة والإدراك المعرفي المتغير للمتعلم. من خلال الدمج الأصيل بين تقنية "التوأم الرقمي للمتعلم" وهيكلية "التوليد المعزز بالاسترجاع(RAG) "، يخلق النظام حلقة تغذية راجعة مستمرة بين العالم المادي والرقمي. يعتمد التوأم الرقمي على خوارزمية التتبع البايزي للمعرفة (BKT) لنمذجة وتوقع مستوى استيعاب الطالب للمفاهيم بشكل مستقل وفي الوقت الفعلي. وفي الوقت ذاته، تعمل معمارية (RAG) كمرشح بيداغوجي صارم يضمن أن جميع الردود الناتجة عن النموذج اللغوي الكبير (LLM) تستند حصرياً إلى المواد الدراسية المعتمدة، مما يقضي تماماً على ظاهرة هلوسة الذكاء الاصطناعي. عبر التقييمات التكيفية، ومسارات التعلم المخصصة، والمدرس الآلي التفاعلي، يقوم نظام EduTwin AI بضبط مستوى تعقيده ديناميكياً ليتوافق مع القدرات المعرفية الحالية للطالب. يُظهر هذا العمل الإمكانيات العميقة لدمج التتبع المعرفي اللحظي مع الذكاء الاصطناعي التوليدي لتقديم تجربة تعليمية مخصصة، آمنة، وصارمة أكاديمياً.

الكلمات المفتاحية:

الذكاء الاصطناعي، التوأم الرقمي للمتعلم، التوليد المعزز بالاسترجاع(RAG) ، التتبع البايزي للمعرفة(BKT) ، النماذج اللغوية الكبيرة(LLMs) ، التعلم المخصص.

Contents

General Introduction **1**

1 State of the Art and Theoretical Foundations **4**

1.1 Introduction 5

1.2 The Evolution of Educational Paradigms 5

1.2.1 Traditional vs. Personalized Learning Models 6

1.2.2 The Shift to Digital Infrastructures 6

1.2.3 Limits of Current Educational Systems 6

1.3 Artificial Intelligence in Education (AIEd) 8

1.3.1 Evolution of AIEd 8

1.3.2 Intelligent Tutoring Systems (ITS) 8

1.3.3 Educational Data Mining (EDM) 9

1.4 Generative AI and Large Language Models (LLMs) 9

1.4.1 Fundamentals of Transformers 9

1.4.2 LLMs in Educational Contexts 10

1.4.3 Limitations of Generic LLMs 11

1.5 Retrieval-Augmented Generation (RAG) 12

1.5.1 RAG Pipeline Architecture 12

1.5.2 RAG as a Pedagogical Filter 14

1.6 Foundations of Digital Twins 14

1.6.1 Academic Definition and Origins of Digital Twins 14

1.6.2 Components of a Digital Twin 15

1.6.3 Types of Digital Twins 16

1.6.4 Digital Twins in Industry 4.0 16

1.7 Learner-Centric Digital Twin 17

1.7.1 Concept of the Learner Digital Twin 17

1.7.2 Learner Modeling . . . 18
1.7.3 Dynamic Updating of the DT . . . 19
1.7.4 Current Challenges and Limitations . . . 19
1.8 Integration of AI and Digital Twins for Personalization . . . 20
1.8.1 Hybrid Approaches (AI + DT) . . . 20
1.8.2 Predictive and Adaptive Learning . . . 21
1.8.3 The Physical-Digital Feedback Loop . . . 22
1.8.4 Limitations of Existing Approaches . . . 23
1.9 Comparative Analysis (Benchmarking) . . . 24
1.10 Identification and Enrichment of the Research Gap . . . 26
1.11 Conclusion . . . 27

2 System Analysis and Conceptual Design . . . 29
2.1 Introduction . . . 30
2.1.1 Functional Requirements . . . 31
2.1.2 Non-Functional Requirements . . . 33
2.2 UML Functional and Structural Modeling . . . 34
2.2.1 Platform Positioning and Design Rationale . . . 35
2.2.2 Actor Identification and Role Definition . . . 35
2.2.3 Use Case Diagram: Overview and Structural Analysis . . . 37
2.2.4 Actor Interaction Model and System Boundaries . . . 38
2.2.5 Textual Description of Major Use Cases . . . 40
2.2.6 Class Diagram and Domain Model . . . 45
2.3 UML Behavioral Modeling . . . 47
2.3.1 Sequence Diagram: Authentication Flow . . . 47
2.3.2 Sequence Diagram: Interaction Processing and RAG Query Pipeline 49
2.3.3 Sequence Diagram: Post-Interaction Cognitive State Synchronization and Adaptive Response Generation . . . 52
2.3.4 Activity Diagram: RAG Query Pipeline . . . 53
2.4 Data and Knowledge Modeling . . . 55
2.4.1 Relational Database Schema . . . 55
2.4.2 Vector Database Strategy and Embedding Architecture . . . 57
2.5 Conclusion . . . 59

3 System Implementation 60
3.1 Introduction 61
3.2 Work Environment 62
3.2.1 Architectural Design: The Modular Monolith Approach 62
3.2.2 Three-Tier Architecture Model 63
3.2.3 AI Infrastructure Layer 65
3.2.4 Retrieval-Augmented Generation Pipeline 65
3.2.5 Cognitive Digital Twin Engine 66
3.2.6 Deployment and Configuration 66
3.2.7 Software Environment and Tools 68
3.2.8 Hardware and Local Deployment Constraints 71
3.2.9 Programming Languages and Frameworks 72
3.3 Platform Interfaces 74
3.3.1 Authentication Interfaces 74
3.3.2 AI Tutor Chat Interface 74
3.3.3 Document Management Interface 75
3.3.4 Digital Twin Dashboard 76
3.3.5 Adaptive Assessment Interface 76
3.3.6 AI Podcast Generator 77
3.3.7 Flashcards Interface 78
3.3.8 ExamLab Interface 78
3.3.9 Mind Map Interface 79
3.3.10 Study Planner Interface 79
3.3.11 Administrator Portal 80
3.4 Critical Technical Implementation 80
3.4.1 RAG Pipeline and Chunking Strategy 80
3.4.2 LLM Integration: Model Selection, API Evolution, and Response Time 83
3.4.3 BKT Engine: Implementation 85
3.4.4 Digital Twin Synchronization Mechanism 86
3.4.5 Adaptive Assessment Generation via Prompt Engineering 87
3.4.6 Performance Evaluation and Validation 87

3.5 Conclusion 89

General Conclusion and Future Work **91**

Bibliography **93**

Appendix **98**

List of Figures

1.1 The transition of LLMs from pre-training to task-specific fine-tuning . . . 10
1.2 Standard RAG pipeline architecture . . . 13
1.3 Vector embedding and semantic search in RAG . . . 13
1.4 Adapting Industry 4.0 Digital Twins for Education . . . 17
1.5 Structural framework of an AI-driven educational system for personalized learning . . . 21
1.6 Physical-Digital feedback loop in a modern learning environment . . . 23
2.1 EduTwin AI System – Use Case Diagram . . . 37
2.2 Class Diagram and Domain Model of EduTwin AI . . . 45
2.3 Sequence Diagram: Authentication Flow . . . 48
2.4 Sequence Diagram: Interaction Processing and RAG Query Pipeline . . . 50
2.5 Sequence Diagram: Post-Interaction Cognitive State Synchronization and Adaptive Response Generation . . . 52
2.6 Activity Diagram: RAG Query Pipeline . . . 55
3.1 EduTwin AI – High-Level System Architecture . . . 63
3.2 Centralized environment configuration file (.env) managing database connections and external API keys securely. . . . 67
3.3 Simultaneous execution of the FastAPI backend service via Uvicorn and the React frontend environment via Vite. . . . 67
3.4 Project workspace mapped within the Antigravity IDE, displaying the decoupled directory structure for frontend and backend services. . . . 68
3.5 User authentication and registration interface of EduTwin AI . . . 74
3.6 AI Tutor conversational interface with RAG source citations . . . 75
3.7 Course document upload and management interface . . . 75
3.8 Learner Digital Twin Dashboard displaying cognitive mastery metrics . . . 76
3.9 Adaptive assessment interface with real-time difficulty calibration . . . 77

3.10 AI Podcast Generator interface with audio playback and transcript 77
3.11 AI-generated flashcard interface for active recall practice 78
3.12 ExamLab interface for full-length mock examination simulation 78
3.13 AI-generated interactive mind map from course document content 79
3.14 Personalized study planner generated from the learner's Digital Twin state 79
3.15 Administrator portal displaying system statistics and user management controls 80
3.16 Document ingestion interface displaying dynamic chunking metrics and indexing status for uploaded course materials. 82
3.17 Backend operational logs executing approximate nearest-neighbor search, detailing cosine distance and semantic similarity scores. 83
3.18 EduTwin AI frontend interface displaying a generated explanation grounded strictly in retrieved chunks with explicit document citations. 83
3.19 Browser network diagnostics displaying empirical response latency: 5.3s to first token and 17.1s total streaming duration for a complex structured-output request. 88
3.20 Complex structured output (Markdown table) generated by the AI Tutor, demonstrating the extensive reasoning and formatting task that accounts for the upper-bound streaming duration. 88
3.21 Backend validation logs demonstrating the DeepSeek-Chat model's strict adherence to the required JSON schema, successfully passing Pydantic validation for assessment generation. 89

List of Tables

1.1 Comparative benchmarking of existing solutions vs. EduTwin AI 25
2.1 Functional Requirements Catalogue 33
2.2 Non-Functional Requirements Specification 34
2.3 Updated Actor Catalogue 36
2.4 Use Case UC-01: Upload Academic Documents 41
2.5 Use Case UC-02: Submit Natural Language Query 43
2.6 Use Case UC-03: Complete Adaptive Assessment 44
2.7 Relational Schema Entity Inventory 56
3.1 Software Tools Used in the Development of EduTwin AI 69
3.2 Programming Languages and Frameworks Used in EduTwin AI 72
3.3 LLM Integration Evolution Across EduTwin AI Development Stages . . . 85
3.4 Discrete Mastery Tiers Derived from Continuous BKT Probability 86

General Introduction

Context and Problematic

Over the last decade, digital technologies have significantly transformed many sectors, including industry, healthcare, and education [1, 5]. The rise of Industry 4.0, driven by intelligent automation, artificial intelligence, connected systems, and data analytics, has changed the way organizations operate and make decisions. As a result, education has also begun to evolve toward what is commonly known as Education 4.0, which seeks to provide more flexible, personalized, and technology-enhanced learning experiences [1, 3].

Despite these developments, traditional educational approaches still rely heavily on standardized teaching methods and uniform evaluation processes. Such approaches often fail to consider the differences in students' learning styles, knowledge levels, and learning pace [3]. Although online learning platforms have improved access to educational content, they generally provide limited personalization and rarely offer continuous monitoring of students' progress based on validated academic resources.

Recent advances in Artificial Intelligence, particularly Large Language Models (LLMs), have opened new opportunities for intelligent tutoring and academic support [11]. These models can answer questions, explain concepts, and assist learners in a conversational manner. However, their use in higher education remains challenging due to issues such as hallucinated information, insufficient domain specialization, and the lack of a persistent understanding of each learner's progress [8]. Consequently, relying solely on generic AI systems may not be sufficient in academic environments environments where accuracy and reliability are essential.

At the same time, the Digital Twin concept, originally introduced in industrial and aerospace applications, has proven effective for monitoring, simulating, and improving

complex systems [1, 12]. While several educational adaptations have been proposed, most existing solutions focus on training simulations or learning analytics rather than creating a comprehensive cognitive representation of university students engaged in theoretical learning activities [12].

This observation highlights an important research gap. Based on the literature reviewed and the educational platforms currently available, there is still a lack of integrated solutions capable of combining learner modeling, personalized adaptation, and academically grounded AI assistance within a single educational framework. In particular, few systems leverage both a Learner-Centric Digital Twin and Retrieval-Augmented Generation (RAG) to support personalized and reliable learning in higher education contexts [7, 8].

Objectives and Contribution

The main objective of this thesis is to develop an intelligent educational platform capable of providing personalized academic assistance while maintaining the reliability of university learning materials. More specifically, the work aims to build a dynamic cognitive model of the learner using Bayesian Knowledge Tracing (BKT), continuously updated through student interactions. In addition, the proposed system seeks to ensure that all generated responses are based on validated course resources through a Retrieval-Augmented Generation (RAG) mechanism. Finally, it aims to offer an adaptive conversational experience that adjusts explanations, recommendations, and learning support according to each student's level of understanding.

To achieve these objectives, this thesis proposes and implements **EduTwin AI**. The platform combines three complementary components within a unified architecture: a Learner-Centric Digital Twin, Bayesian Knowledge Tracing, and a RAG-based knowledge retrieval system connected to university course documents. Unlike many existing solutions that address these aspects separately, EduTwin AI integrates them into a single environment designed to support personalized learning in Software Engineering education.

Thesis Structure

This thesis is organized into three chapters, followed by a general conclusion and an appendix.

Chapter 1 presents the theoretical background of the study. It reviews the evolution of educational paradigms, discusses the growing role of Artificial Intelligence and Large Language Models in education, introduces Retrieval-Augmented Generation, and examines the concept of the Learner-Centric Digital Twin. The chapter concludes by identifying the research gap addressed in this work.

Chapter 2 focuses on the analysis and design of the proposed system. It describes the functional and non-functional requirements, presents the different actors and use cases, and details the UML models used to represent the system structure and behavior.

Chapter 3 describes the implementation of EduTwin AI. It presents the system architecture, the technologies adopted, the development environment, and the main software modules. The chapter also discusses the obtained results and evaluates the effectiveness of the proposed approach.

The thesis concludes with a **General Conclusion**, which summarizes the main contributions and findings of the study while outlining possible directions for future work. An **Appendix** is included to present a preliminary entrepreneurial study investigating the potential of EduTwin AI as an educational technology product.

Chapter 1

State of the Art and Theoretical Foundations

1.1 Introduction

Before designing and developing the EduTwin AI architecture, it is essential to establish a robust theoretical foundation. This chapter synthesizes the current state of research regarding personalized education technologies, traces the evolution from classical AI systems to modern Large Language Models and Retrieval-Augmented Generation, and subsequently introduces the Digital Twin concept as the structural backbone of the proposed system.

The chapter proceeds in a deliberately logical sequence that mirrors the historical development of the field. Section 1.2 traces the evolution of educational paradigms and highlights the limits of current digital systems. Section 1.3 examines Artificial Intelligence in Education, from its classical rule-based origins through Intelligent Tutoring Systems and Educational Data Mining, demonstrating how AI first attempted to personalize learning. Section 1.4 examines how Generative AI and Large Language Models transform the interaction model by enabling naturalistic academic dialogue, and identifies the limitations that prevent their deployment in rigorous university environments without augmentation. Section 1.5 presents Retrieval-Augmented Generation as the architectural solution to those limitations, acting as a strict pedagogical filter anchored to course-validated materials. Sections 1.6 and 1.7 introduce the Digital Twin concept and its adaptation to the learner, establishing the structural memory layer that makes the system truly personalized. Section 1.8 examines how AI and Digital Twins are integrated to form a continuous physical-digital feedback loop. Sections 1.9 and 1.10 present a comparative benchmarking of existing solutions and a structured enrichment of the research gap around three interconnected pillars. Section 1.11 concludes the chapter and presents the proposed contribution. This progression, from AI to LLMs to RAG to Digital Twins, is intentional: each technological layer builds upon the previous one to motivate our integrated approach.

1.2 The Evolution of Educational Paradigms

Education has undergone continuous transformation shaped by philosophical developments, societal change, and technological advancement. It is increasingly understood

not as a uniform delivery mechanism but as a dynamic, adaptive, and learner-centered process [3].

1.2.1 Traditional vs. Personalized Learning Models

Traditional educational systems operate under a standardized instructional paradigm in which all students receive identical content at an identical pace. Knowledge transmission is predominantly teacher-centered, and evaluation is uniform regardless of the cognitive diversity of the student population. This model, while scalable, fundamentally ignores the well-documented reality that learners differ substantially in their prior knowledge, learning speed, preferred modalities, and motivational structures. The expansion of digital technologies has enabled a meaningful transition toward personalized learning environments, which leverage data analytics to adapt content, pacing, and feedback dynamically to each individual learner [5].

1.2.2 The Shift to Digital Infrastructures

Over the past decades, rapid digitalization has fundamentally reshaped educational systems worldwide. Learning Management Systems (LMS) such as Moodle, Blackboard, and Canvas have become standard infrastructure in universities, providing digital course delivery, assessment tools, and communication platforms. This transformation was further accelerated by the post-COVID-19 demand for resilient and scalable digital infrastructures [2]. Educational institutions were compelled, often overnight, to adopt fully remote learning models, which exposed both the potential and the deep limitations of existing digital platforms. This period of forced digital acceleration laid the groundwork for more advanced learner modeling approaches, as it generated unprecedented volumes of student interaction data.

1.2.3 Limits of Current Educational Systems

Despite the widespread adoption of Learning Management Systems, current digital educational models still present significant limitations that prevent them from truly personalizing the learning experience [3, 5]. These systems were designed primarily

as content delivery mechanisms, not as cognitive modeling tools. Four major academic works have clearly documented these shortcomings.

Ciani et al. (2023) [5] identified that current LMS platforms offer limited adaptability to individual learning speeds, relying instead on fixed deadlines and uniform content sequences that ignore where each student actually is in their understanding. The EM Lyon Business School research group [3] demonstrated that insufficient accommodation of cognitive diversity remains a structural problem: systems treat students as a homogeneous group when the reality of any classroom is profound heterogeneity in prior knowledge, learning styles, and motivational profiles. Wang et al. (2022) [6] showed that feedback mechanisms in current systems are not only delayed, often arriving days after an assessment, but are also generalized, offering the same comments to students regardless of the specific nature of their errors. Finally, Mourtzis et al. (2025) [12] documented through a systematic review that the transition from passive content delivery to active, intelligent adaptation remains incomplete: systems log student activity extensively but rarely use that data to drive real-time pedagogical decisions.

These structural shortcomings are not merely theoretical. Several concrete systems have attempted to address them and revealed the depth of the problem in practice. The Moodle-based adaptive learning implementations reviewed by Mourtzis et al. (2025) [12] demonstrated that even when LMS platforms are extended with analytics plugins, the feedback loop remains too slow and too coarse to produce meaningful personalization. The e-learning Digital Twin prototype proposed by Jankovskis et al. (2024) [18] attempted to introduce real-time monitoring into digital education platforms but acknowledged that the system lacked the analytical intelligence to interpret the collected data and generate pedagogical interventions autonomously. Wang et al. (2022) [6] implemented a teaching mode based on educational data mining and reported that while classification and clustering of student data was achievable, translating those insights into real-time adaptive content delivery remained an unsolved engineering challenge. These cases confirm that the limitations of current educational systems are not simply gaps in theory but concrete obstacles that directly motivated the research direction pursued in this thesis.

These four limitations, namely rigidity of pacing, blindness to cognitive diversity, delayed and generic feedback, and passivity of data utilization, collectively highlight an urgent transition toward intelligent, predictive solutions capable of real-time adaptation. It is precisely to address this transition that researchers have begun exploring Artificial Intelligence as the engine of the next generation of educational systems.

1.3 Artificial Intelligence in Education (AIEd)

1.3.1 Evolution of AIEd

Artificial Intelligence in Education (AIEd) has evolved considerably since its inception. In the 1980s, early AI-driven educational tools were simple, rule-based computer-aided instruction systems that followed predefined decision trees: if a student answered incorrectly, they were routed to a remedial explanation; if correctly, they advanced to the next item. While functional, these systems were brittle, unable to handle ambiguity, and completely blind to the rich contextual factors that influence learning [8].

Through the 1990s and 2000s, more sophisticated probabilistic and statistical methods were introduced. The field of Educational Data Mining (EDM) emerged as a discipline dedicated to applying machine learning techniques to the large datasets generated by educational platforms. Classification algorithms, clustering methods, and Bayesian networks began to allow systems to model student behavior with greater nuance, predicting performance and identifying at-risk students before failure occurred [6, 9]. This era marked a critical conceptual shift: from reacting to student errors to anticipating them.

1.3.2 Intelligent Tutoring Systems (ITS)

Central to the evolution of AIEd is the Intelligent Tutoring System (ITS), a class of AI-driven software designed to provide immediate, personalized instruction or feedback to learners, typically without the intervention of a human teacher [8]. An ITS maintains an internal model of the student's knowledge state and uses it to select the next instructional action: whether to present a new concept, revisit a previous one, offer a

hint, or pose a more challenging problem. Classic examples include Carnegie Learning's MATHia platform for algebra and the ANDES system for physics. These systems demonstrated conclusively that one-on-one intelligent tutoring can produce learning gains equivalent to having a private human tutor, the so-called "two sigma" effect identified by Bloom (1984).

1.3.3 Educational Data Mining (EDM)

To power these tutoring systems and generate meaningful pedagogical insights at scale, Educational Data Mining (EDM) techniques are employed to analyze the historical log data generated by student-system interactions [6, 9]. Classification algorithms identify patterns in student responses that predict future performance. Clustering techniques group students with similar learning profiles. Association rule mining reveals which sequences of learning activities tend to produce the best outcomes. These techniques have proven valuable in generating insights for educators, but they operate primarily in batch mode, analyzing past data rather than driving real-time adaptation. It is this limitation that motivates the incorporation of more dynamic, interactive AI architectures, beginning with the most transformative development in recent AI history: the rise of Large Language Models.

1.4 Generative AI and Large Language Models (LLMs)

1.4.1 Fundamentals of Transformers

The landscape of Artificial Intelligence was fundamentally transformed by the introduction of the Transformer architecture by Vaswani et al. (2017). Unlike recurrent neural networks that processed sequences step by step, Transformers process entire sequences simultaneously through a mechanism called self-attention, which allows each token in a sequence to directly attend to every other token. This architectural innovation enabled unprecedented parallelism during training, allowing models to be trained on vastly larger datasets than previously feasible [11].

The result of scaling this architecture to billions of parameters and training on corpus-scale text data is the Large Language Model (LLM). LLMs such as GPT-4, Claude, Gemini, and LLaMA generate responses by predicting the probability distribution of the next token given all preceding context. As illustrated in Figure 1.1, these models undergo two primary phases: Pre-training on massive, unlabeled datasets to acquire general language understanding, followed by Fine-tuning on domain-specific, curated data to adapt their behavior to specialized tasks [11]. The left side of Figure 1.1 shows the foundation model learning general language patterns from the internet, books, and Wikipedia through unsupervised learning. The right side illustrates the fine-tuning phase, in which the same base model is further trained on domain-specific data such as course syllabi, textbooks, and educational question-answer pairs, producing a context-aware, specialized tutor agent.

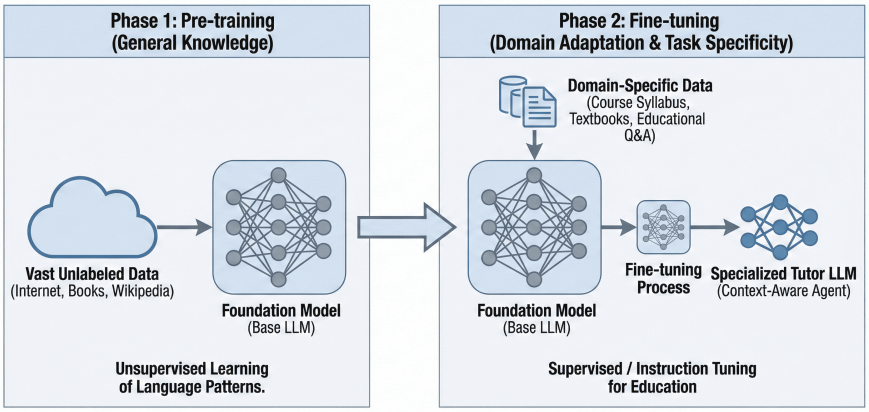


Figure 1.1: The transition of LLMs from pre-training to task-specific fine-tuning [11]

1.4.2 LLMs in Educational Contexts

In education, LLMs function as conversational agents capable of engaging students in naturalistic dialogue about academic content. They generate context-aware responses by predicting token probabilities conditioned on both the student's query and the accumulated conversational history, allowing for dynamic explanations that adapt organically to the student's unique query style [8]. A student who asks "I don't understand this formula" receives a different explanation than one who asks "Can you prove why this formula is derived this way?", as the LLM calibrates its response to the perceived sophistication of the question.

Several academic works have demonstrated the educational potential of LLMs. The LPITutor system (Liu et al., 2025) [8] showed that LLM-based tutoring systems, when augmented with prompt engineering, significantly improve student comprehension compared to static LMS content. The PRAG-EDU framework (Nguyen et al., 2026) [13] demonstrated that when RAG systems are augmented with students' historical module grades as pedagogical signals, the system dynamically calibrates the complexity of its explanations to the student's demonstrated academic proficiency. Research at Johns Hopkins University produced GAITA [14], a RAG-based system for computer science education, which showed that grounding LLM responses in curated course materials dramatically reduced hallucination rates and increased student trust in the system's outputs.

1.4.3 Limitations of Generic LLMs

Without a filtering mechanism, generic LLMs face severe limitations that make them unsuitable as authoritative academic companions [8, 11]. **Hallucination** refers to the tendency to confidently generate factually incorrect information, a critical failure in a university context where students may have no way to detect the error. **Bias** refers to the amplification of subjective opinions and dominant perspectives found in broad internet training data, which distorts the objectivity expected in academic instruction. **Lack of Grounding** reflects an inability to reference a specific university curriculum, a proprietary textbook, or course-specific problem sets; the model responds from its general knowledge, which may contradict what the instructor has taught. Finally, **Statelessness** means that generic LLMs retain no memory of previous student interactions across sessions, making longitudinal personalization structurally impossible regardless of how sophisticated the model's reasoning capabilities are.

These limitations collectively motivate the need for a structural mechanism that not only grounds the LLM's responses in validated course materials, but also maintains a persistent, dynamic model of the learner's knowledge state across time. The architectural solution to the grounding problem is Retrieval-Augmented Generation, presented in the next section.

1.5 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is the core architectural solution that transforms LLMs from general knowledge systems into precision academic companions [8]. Rather than relying on the model's internal parametric memory, RAG augments each generation step with dynamically retrieved, contextually relevant document chunks. The result is a system that can answer questions about a specific course syllabus with the same fluency as a general LLM, but with the factual reliability of a direct document lookup.

1.5.1 RAG Pipeline Architecture

The RAG pipeline operates in three sequential phases, as illustrated in Figure 1.2 and Figure 1.3. During the **Document Processing** phase, course materials, including PDFs, lecture notes, and textbooks, are ingested and segmented into semantically coherent chunks through a process called *chunking*. These chunks are then converted into dense vector representations (embeddings) by an embedding model and stored in a vector database for rapid retrieval. Chunking is a critical and non-trivial step: chunks that are too large lose specificity, while chunks that are too small lose context. Optimal chunking strategies using overlapping windows of 512 to 1024 tokens ensure each retrieved segment is both semantically meaningful and self-contained.

During the **Query Retrieval** phase, a student's question is converted into the same embedding space using the same embedding model. A semantic similarity search, most commonly using Cosine Similarity, identifies the top-K most semantically relevant document chunks. As shown in Figure 1.3, the query vector is compared against all stored chunk vectors in the database, and the closest matches in the embedding space are returned as retrieved context. This retrieval is not keyword matching but true semantic understanding: a query about "how models learn from mistakes" will correctly retrieve chunks about gradient descent and backpropagation, even if those exact words were not used in the query.

During the **Generation** phase, the retrieved chunks are combined with the student's query and the learner's contextual profile into an Augmented Prompt, which is passed

to the LLM. The system instructs the model to synthesize its answer strictly from the provided context, refusing to speculate beyond the validated course materials. As shown on the right side of Figure 1.2, this architecture directly contrasts with a standard LLM that generates answers from its parametric memory alone and is prone to hallucination: the RAG system forces every response to be traceable back to a specific chunk of the uploaded course document.

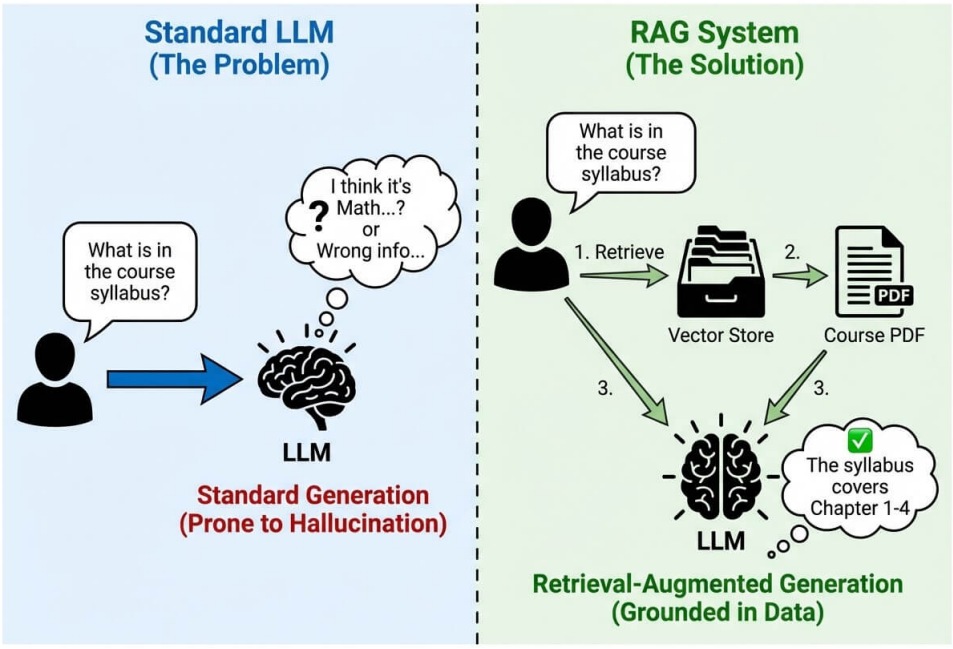


Figure 1.2: Standard RAG pipeline architecture [8]

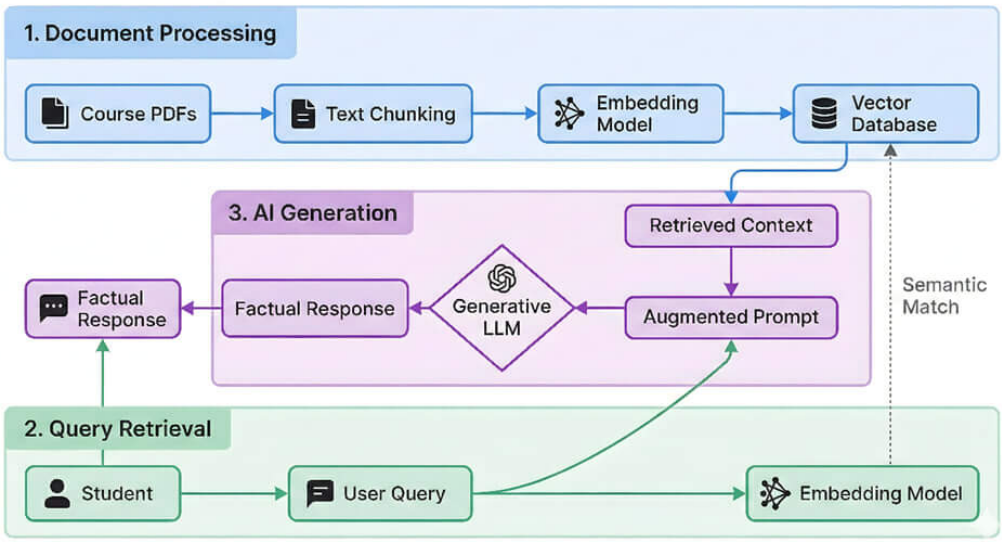


Figure 1.3: Vector embedding and semantic search in RAG [8, 11]

1.5.2 RAG as a Pedagogical Filter

Crucially, RAG does not merely search for documents; it acts as an intelligent **pedagogical filter**. It retrieves only the most relevant academic context, filters out extraneous data, and forces the generative model to synthesize its answer strictly based on validated course materials [8]. This provides immense pedagogical utility by ensuring students receive explainable, syllabus-grounded guidance rather than generic internet answers. When no relevant content is found in the course documents, the system explicitly informs the student rather than generating a plausible but ungrounded response, a behavior that directly addresses the hallucination problem identified in the previous section.

The combination of LLMs and RAG therefore resolves the grounding and reliability problems of generic generative AI. However, what remains missing is a persistent structural memory of the learner, a living model that evolves with every interaction and informs the generation process with knowledge of who this specific student is. This is precisely what the Digital Twin provides.

1.6 Foundations of Digital Twins

The concept of the Digital Twin did not emerge from education. It was born in the most demanding engineering environments in the world, where the cost of physical failure is catastrophic and the ability to simulate, monitor, and predict system behavior in real time is not a luxury but a necessity. Understanding where it came from, how researchers have defined it, and what the scientific community believes it can accomplish in education is essential before applying it to the learner.

1.6.1 Academic Definition and Origins of Digital Twins

A Digital Twin (DT) is academically defined as a highly synchronized virtual representation of a physical entity, process, or system, continuously updated through real-time data exchange [1]. The concept was originally conceived in aerospace engineering at NASA during the Apollo program, where it was used to create paired physical

and virtual models of spacecraft so that engineers on the ground could mirror and monitor the exact state of a vehicle in orbit. It was later formally theorized by Michael Grieves at the University of Michigan in 2002 in the context of product lifecycle management, and further formalized by Grieves and Vickers (2017) as a three-component model comprising a physical product, a virtual product, and the connections between them.

Researchers across disciplines have since proposed numerous definitions that converge on the same core idea. Al-Obaidi et al. (2022) [1] define it as a virtual model that mirrors its real-world counterpart through continuous data synchronization, enabling simulation and optimization. Flaga and Pacholczak (2022) [10] emphasize the operational aspect, describing the DT as a system where data flows bidirectionally between the physical and virtual layers in real time. Mourtzis et al. (2025) [12], in their systematic review of DTs in education, stress that the defining characteristic of a true twin is not the sophistication of its virtual model but the continuity and fidelity of its synchronization with the physical entity. The Digital Twin does not merely store historical data; it actively simulates, predicts, and optimizes the state of its physical counterpart in real time, without disrupting the physical system itself. The objective of this synchronization is always the same: to generate actionable intelligence about the current state of the physical entity and to enable proactive intervention before problems manifest.

What distinguishes a Digital Twin from a simple database or user profile is its dynamic, bidirectional nature. The physical entity generates data through its actions; this data flows continuously to the virtual model, updating it in real time; the virtual model then generrates predictions, recommendations, or interventions that flow back to influence the physical entity's behavior. This continuous physical-digital feedback loop is the defining operational characteristic of a genuine Digital Twin [10].

1.6.2 Components of a Digital Twin

The native functionality of a Digital Twin rests on four core components [10]:

- **The Physical Entity:** The real-world object or agent being modeled; in our educational context, the student in their learning environment.
- **The Virtual Model:** The mathematical and data-driven digital replica; in our case, the learner's cognitive profile including mastery probabilities per topic, learning style parameters, and historical interaction patterns.
- **Data Flow:** All sensors, APIs, and logging mechanisms that continuously capture real-world inputs; in education, this means quiz scores, time-on-task, query patterns, and assessment responses.
- **Synchronization:** Real-time, low-latency communication ensuring the virtual model accurately reflects the current state of the physical entity at any given moment.

1.6.3 Types of Digital Twins

Digital Twins can be broadly categorized into Physical Twins, which simulate hardware, machinery, or structural components, and Cognitive or Human Twins, which focus on simulating human states, behaviors, decision-making processes, and knowledge acquisition trajectories [1, 7]. Cognitive Twins are significantly more complex than physical twins because human learning is inherently stochastic, context-dependent, and influenced by factors such as emotional state, motivation, and prior experience that are difficult to quantify. Nevertheless, probabilistic models such as Bayesian Knowledge Tracing provide a rigorous mathematical foundation for approximating the latent knowledge state of a learner.

1.6.4 Digital Twins in Industry 4.0

Initially deployed in aerospace and industrial sectors (Industry 4.0) for predictive maintenance and process optimization, scholars such as Al-Obaidi et al. (2022) [1] have proposed the systematic adaptation of this concept to education (Education 4.0). As seen in Figure 1.4, this adaptation provides the structural basis for intelligent learning ecosystems. The left side of the figure represents the physical learning environment: the student interacts with course materials, activities, and assessments, and all of these

interactions are captured as behavioral data through sensors, LMS logs, and activity tracking. This data stream feeds continuously into a Continuous Optimization Cycle at the center, which updates the Digital Twin shown on the right side of the figure. The twin, labeled as the Learner-Centric Model, maintains a live representation of the student's knowledge state, learning patterns, and performance trajectory. From this virtual representation, personalized content and adaptive support are generated and delivered back to the student in the physical environment, completing the feedback loop. This conceptual transition, from modeling machines to modeling minds, establishes the theoretical foundation for applying the Digital Twin specifically to the student.

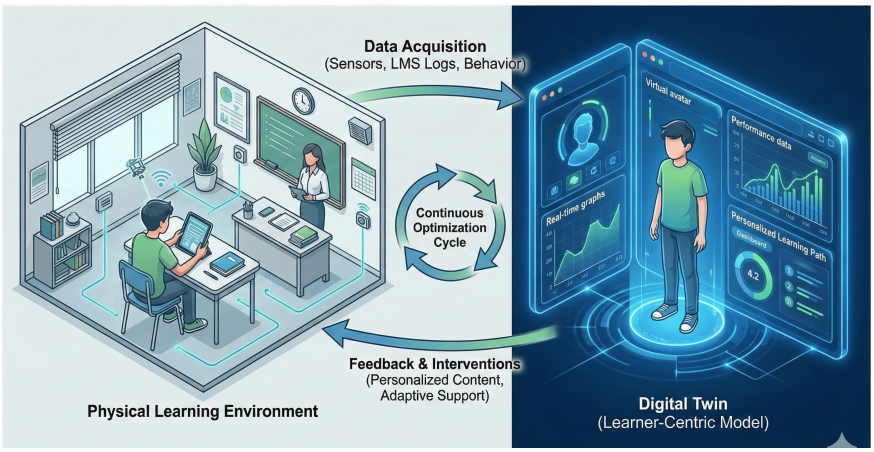


Figure 1.4: Adapting Industry 4.0 Digital Twins for Education [1]

1.7 Learner-Centric Digital Twin

1.7.1 Concept of the Learner Digital Twin

Moving from a Physical System Digital Twin to a Learner-Centric Digital Twin requires a profound paradigm shift, not merely a change in domain, but a fundamental reconceptualization of what is being modeled and why. In this new framework, pedagogical theories replace engineering models. Rather than replicating mechanical behavior governed by physics, educational Digital Twins model knowledge acquisition trajectories and cognitive states governed by learning science [5, 7].

The Learner Digital Twin is therefore best understood as a *living probabilistic model*: a dynamic, continuously updated representation of what a student knows, what they struggle with, how they learn best, and where they are likely to encounter difficulty

next. Unlike a static user profile which records what a student has done, the Learner DT *predicts* what the student will need next. This predictive, forward-looking character is what distinguishes a true Digital Twin from a mere database entry. The twin evolves with every quiz answer, every question asked, every document section consulted; it is not a snapshot of the student at enrollment, but a real-time simulation of their current cognitive state.

1.7.2 Learner Modeling

To construct the virtual replica of the learner, the system relies on Learner Modeling, a field dedicated to the mathematical representation of student knowledge, behavior, and cognitive state [6]. The most established and academically validated approach is Bayesian Knowledge Tracing (BKT), which models each knowledge concept as a binary latent variable (known or not known) using a Hidden Markov Model.

The BKT state update is governed by four parameters:

- $P(L_0)$: The prior probability of initial mastery.
- $P(T)$: The probability of a learning transition (moving from unmastered to mastered).
- $P(G)$: The probability of a "Guess" (correct answer despite not knowing).
- $P(S)$: The probability of a "Slip" (incorrect answer despite knowing).

Upon observing a response (obs_t), the system updates the mastery probability using Bayes' theorem:

$$P(L_t \mid \text{obs}_t) = \begin{cases} \frac{P(L_t)(1-S)}{P(L_t)(1-S)+(1-P(L_t))G} & \text{if observed correct} \\ \frac{P(L_t)S}{P(L_t)S+(1-P(L_t))(1-G)} & \text{if observed incorrect} \end{cases}$$

The mastery probability is then propagated forward for the next opportunity:

$$P(L_{t+1}) = P(L_t \mid \text{obs}_t) + \left[1 - P(L_t \mid \text{obs}_t)\right] \times P(T)$$

This formulation allows the system to distinguish between a correct answer produced by genuine mastery versus one produced by guessing, and between an incorrect answer produced by genuine ignorance versus a momentary slip [15]. While Equation 1.1 and 1.2 define the state evolution, the parameters $P(L_0)$, $P(T)$, $P(G)$, and $P(S)$ serve as the probabilistic coefficients that calibrate the model. Specifically, G and S function as error-correction weights within the Bayes update, while $P(T)$ regulates the learning rate in the forward propagation step. The result is a continuously updated, per-topic mastery probability vector that constitutes the cognitive core of the Learner Digital Twin.

1.7.3 Dynamic Updating of the DT

A true Learner-Centric DT is not a static database; it requires continuous, dynamic updating [10]. Every interaction the student has with the educational platform serves as a data signal that refines the twin's accuracy. When a student submits a quiz response, the BKT model updates the mastery probability for the relevant knowledge concepts. When a student asks a question, the semantic content of the query reveals their current understanding or misunderstanding of a topic. When a student spends significantly more time on one section than others, the system interprets this as a signal of cognitive difficulty and adjusts the twin's profile accordingly.

This continuous refinement process is what makes the Learner DT genuinely dynamic rather than merely historical. The twin does not simply record what happened; it continuously revises its probabilistic model of the student's knowledge state in light of new evidence, always maintaining the best possible current estimate of where the student is in their learning journey.

1.7.4 Current Challenges and Limitations

Despite its considerable potential, deploying Learner Digital Twins in real educational environments faces several significant challenges. The sheer volume of unstructured data generated by students, including forum posts, open-ended responses, and browsing patterns, is difficult to process automatically without sophisticated NLP

pipelines [5]. Privacy concerns surrounding the continuous collection of detailed student behavioral data raise legitimate ethical questions. The cold-start problem, which is the difficulty of building an accurate model for new students with no prior interaction history, requires thoughtful initialization strategies.

Perhaps most critically, **Scalability** presents a fundamental technical challenge that has received insufficient attention in the existing literature: the ability to manage a large number of learner digital twins simultaneously, in real time, without degrading system performance [5]. A university course may have hundreds of students; a platform deployment may have thousands. Each twin must be maintained, updated, and queried independently, placing substantial demands on the underlying data architecture. Solutions such as asynchronous twin updates, lazy evaluation, and distributed profile storage are necessary components of any production-grade Learner DT system, and represent a key architectural consideration in the design of EduTwin AI.

1.8 Integration of AI and Digital Twins for Personalization

1.8.1 Hybrid Approaches (AI + DT)

Recent academic frameworks have converged on the identification of AI and Digital Twins as the two synergistic pillars of next-generation personalized learning [4, 5]. Neither technology alone is sufficient: the Digital Twin without AI provides a rich model of the student's state but lacks the intelligence to generate appropriate interventions; AI without the Digital Twin provides powerful generative capabilities but has no persistent, structured knowledge of the individual learner.

Several significant frameworks have proposed and validated this combination. Ciani et al. (2023) [5] proposed a three-layer architecture in which the Digital Twin acts as the state repository, the AI Analytics Engine acts as the reasoning layer using Machine Learning and NLP techniques, and the Presentation Layer delivers personalized recommendations and adaptive assessments to the student. The AI4Work project [9], co-funded by the European HORIZON programme, developed human-centric Digital Twin appro approaches combined with AI for educational and workplace settings, demonstrating institutional momentum at the European scale. David (2024) [4] con-

ducted a comprehensive AI-Simulation survey involving over 150 academic systems, concluding that the DT+AI combination is the dominant emerging paradigm for adaptive simulation-based training environments and identifying it as the direction most likely to produce robust, generalizable personalized learning systems in the near future.

As illustrated in Figure 1.5, the structural framework of such a combined system organizes into three clearly separated layers. The **Data Layer** at the bottom collects multi-source inputs including LMS interaction logs, behavioral sensors, and external APIs, consolidating them into a Learner Data Repository. The **AI/Analytics Layer** in the middle constitutes the intelligence core: on the left, the Digital Twin student state model maintains a structured Knowledge Model, Affective State, and Behavior profile; on the right, the AI Analytics Engine applies Machine Learning, EDM, and NLP techniques to process the twin's state and generate actionable pedagogical decisions. The **Presentation Layer** at the top delivers the output of this intelligence to the student through a Personalized Interface offering customized recommendations, adaptive assessments, and interactive content.

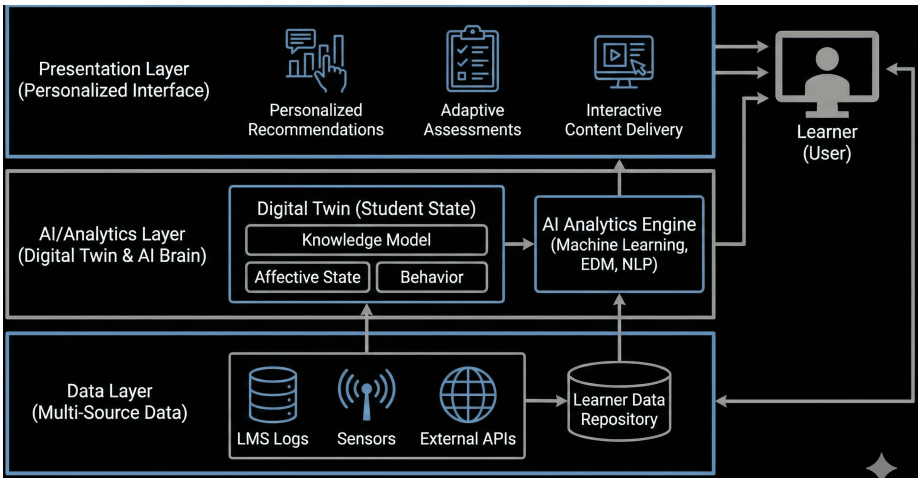


Figure 1.5: Structural framework of an AI-driven educational system for personalized learning [4]

1.8.2 Predictive and Adaptive Learning

The most powerful consequence of the hybrid AI+DT integration is the enabling of genuinely predictive and adaptive learning [6]. The Digital Twin's continuously updated mastery probability vectors, combined with the AI engine's pattern recognition capabilities, allow the system to identify when a student's trajectory suggests an

upcoming struggle with a specific topic. The system can then proactively introduce prerequisite review content, simplify the next explanation, or offer an additional worked example, all before the student has experienced the confusion that typically accompanies encountering a concept for which they are not prepared. This transforms the educational interaction from reactive, which is remediation after failure, to proactive, which is prevention of failure.

1.8.3 The Physical-Digital Feedback Loop

The operational efficacy of the AI+DT system depends critically on the continuous physical-digital feedback loop that connects student actions to system responses [10]. As shown in Figure 1.6, this architecture is organized into three functional layers. The **Infrastructure Layer** on the left represents the student's physical interaction point: the student generates course activities and queries, which are transmitted as a real-time data stream to the system's backend. The **Analytics and AI Layer** in the center is where the intelligence of EduTwin AI resides: the student's query is converted into a vector embedding and used to perform a semantic similarity search against a Vector Database of course PDFs, retrieving the top-K most relevant document chunks. Simultaneously, the Twin State module, which holds the learner's current profile including mastery levels and learning style, is loaded and used to construct a Dynamic Prompt that combines the retrieved academic context with the student's personalized state. This augmented prompt is passed to the Generative LLM, which synthesizes a factual, tailored explanation. The **Presentation Layer** on the right delivers the result back to the student as a personalized, adaptive interface response, completing the loop and triggering the next cycle of interaction and twin update.

The continuity and speed of this loop is what distinguishes a genuine Digital Twin system from a periodic batch-analysis LMS. In a traditional LMS, student data is analyzed weekly or per-assessment, and feedback arrives too late to prevent misconceptions from forming. In the EduTwin AI architecture, the loop operates per-interaction: every action by the student immediately updates the twin and potentially triggers a personalized response.

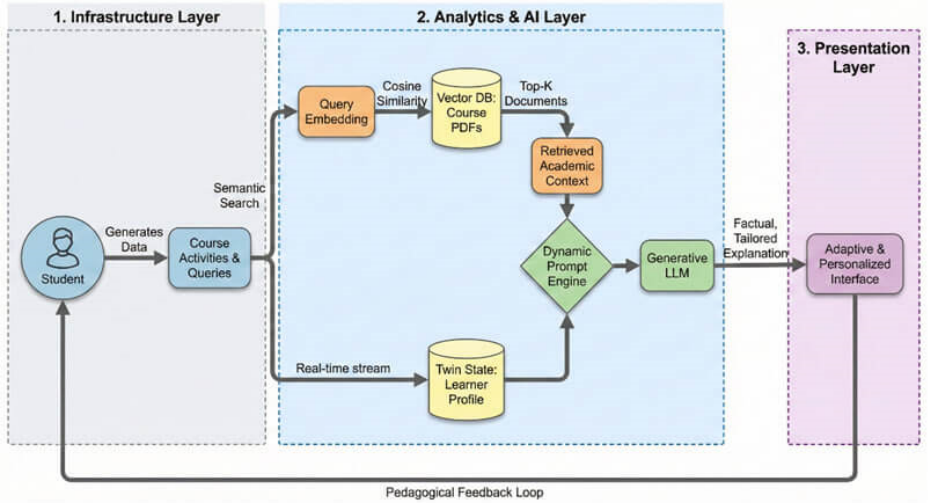


Figure 1.6: Physical-Digital feedback loop in a modern learning environment [10]

1.8.4 Limitations of Existing Approaches

Despite the theoretical promise of AI+DT integration, existing approaches, both classical and recent, exhibit persistent limitations that prevent them from fully realizing the personalization potential described in the preceding sections.

Traditional AI and EDM models, which dominated educational technology throughout the 1990s and 2000s, rely on static, rule-based or batch-processing logic. Systems such as early Knowledge-Based Tutors and the first generation of adaptive LMS platforms could classify student performance and suggest remediation paths, but they fundamentally lacked the ability to hold natural, conversational dialogues with the student [4]. Their interaction model was transactional: the student answered a question, the system responded with a predefined message. This rigidity severely limited engagement and prevented the kind of exploratory, Socratic dialogue that drives deep conceptual understanding.

More recent approaches that incorporate LLMs directly into educational platforms without a RAG filter attempt to address the conversational limitation but introduce a different and arguably more serious problem: hallucination and lack of academic grounding. Systems that implement Digital Twins without Generative AI, on the other hand, maintain accurate student state models but lack the expressive, conversational capability needed to explain complex concepts in naturalistic language.

These limitations are clearly illustrated by examining concrete systems in the literature. TutorLLM (Li et al., 2025) [17] represents perhaps the closest prior work to EduTwin AI, combining Knowledge Tracing and RAG within an LLM-based tutoring interface; however, it relies on a flat KT predictor rather than a persistent bidirectional Digital Twin, was implemented as a browser plugin with no scalable university backend, and was validated only on short-term quiz score improvements rather than longitudinal cognitive tracking. PRAG-EDU (Nguyen et al., 2026) [13] augments RAG with student grade history as a pedagogical signal, partially addressing the personalization problem, but its student model is a static academic record rather than a live, per-interaction cognitive profile. GAITA (Wong, 2024) [14] successfully grounds LLM responses in course-specific materials through RAG, reducing hallucination rates significantly, but applies no student modeling whatsoever, treating every learner identically regardless of their mastery level or learning trajectory. On the Digital Twin side, Jankovskis et al. (2024) [18] proposed a prototypical DT architecture for e-learning with real-time data exchange between a physical platform and a virtual replica, but integrated no LLM or RAG component, leaving the system unable to generate naturalistic academic explanations. Lin et al. (2024) [19] applied industrial Digital Twins to engineering education laboratories, demonstrating the value of real-time simulation, but confined the approach to physical equipment modeling with no adaptation to the cognitive state of the individual learner. Each of these systems solves one or two pieces of the puzzle. None assembles all three: a live cognitive twin, a strict academic filter, and an adaptive conversational interface within a single integrated architecture. This tripartite gap is precisely what EduTwin AI is designed to address.

1.9 Comparative Analysis (Benchmarking)

A comprehensive evaluation of the current educational technology landscape reveals a stark dichotomy between commercial tools and academic models [7]. Table 1.1 presents a comparative benchmarking analysis across four critical criteria: Adaptability, Real-Time Tracking, Logical/Factual Grounding, and Experimental Validation.

Comparison Criteria	Generic LLMs (e.g., ChatGPT)	Document Analysis (ChatPDF)	Commercial EdTech (Khanmigo)	Traditional Academic DTs	**Our Proposed System (EduTwin AI)**
Adaptability (Cognitive Support)	High, but entirely unstructured	Low (restricted to one file at a time)	Medium (standardized pacing)	Low (focuses on physical engineering)	**High (Adapts to specific cognitive gaps based on dynamic BKT profile)**
Real-Time Tracking	None (Stateless sessions)	None (No user history retained)	Medium (Tracks scores on standard tests)	High (Tracks real-time machine telemetry)	**High (Continuous physical-digital feedback loop monitoring per-interaction)**
Logical/Factual Grounding	Low (Prone to academic hallucinations)	Medium (Quotes file but struggles with logic)	High (Predefined pedagogical guardrails)	High (Governed by strict physical laws)	**Very High (Strictly filtered and anchored via RAG on university syllabus)**
Experimental Validation	Varies wildly by user prompt	None (Commercial search tool)	High (Extensive K-12 validation)	High (In industrial smart manufacturing)	**Proposed (To be validated in a higher education software engineering context)**

Table 1.1: Comparative benchmarking of existing solutions vs. EduTwin AI

1.10 Identification and Enrichment of the Research Gap

As highlighted by the comparative analysis and systematic reviews [12], while Digital Twins are successfully deployed for advanced engineering applications, they face severe barriers to widespread adoption in general education. Similarly, commercial conversational agents lack the rigorous factual grounding required for university curricula [8]. The existing literature reveals not a single research gap but a structured set of three interconnected gaps that together define the full scope of what is missing.

Gap 1: The Chasm Between Digital Twin Theory and Software Implementation in Education.. The vast majority of research on Digital Twins in education remains at the conceptual or architectural level. Most studies propose theoretical frameworks and high-level diagrams, but stop short of delivering a concrete, functional software implementation that actually synchronizes real-time student interactions with a probabilistic knowledge model. There is a critical absence of a software architecture capable of synchronizing real-time student interaction events, including quiz responses, time-on-task, and question patterns, with a probabilistic learner model such as BKT, producing a genuinely live digital twin rather than a retrospective profile.

Gap 2: The Absence of a Strict Academic Filter in LLM-Based Educational Systems.. Current systems based on conversational AI agents such as ChatGPT, Claude, and Gemini rely on general knowledge acquired from internet-scale training data. For a Master's student in Software Engineering, this presents a serious risk: hallucinated code examples, outdated algorithms, or theoretically incorrect explanations generated with high confidence. The research community has not yet produced a solution that natively integrates Retrieval-Augmented Generation as a hard pedagogical barrier, guaranteeing that the AI will only respond based on content explicitly present in the instructor's uploaded course materials.

Gap 3: The Missing Interoperability Between Cognitive Tracking and Content Generation.. Today, systems that track student progress (LMS platforms, Knowledge Tracing models) and systems that generate educational content (LLMs, RAG chatbots) operate as separate silos. An LMS may know that a student has a 40% mastery probability on "Database Normalization" but cannot use that information to adjust

the complexity of an LLM-generated explanation. Conversely, an LLM-based tutor generates personalized explanations but has no access to the student's longitudinal mastery history. The true gap lies in creating a hybrid system in which the Learner Digital Twin's current state directly and dynamically shapes the prompts sent to the RAG+LLM pipeline, creating genuine interoperability between cognitive tracking and content generation that neither domain currently achieves alone.

EduTwin AI is proposed to directly address all three gaps simultaneously: by delivering a concrete software implementation of a Learner DT, by using RAG as a strict pedagogical filter anchored to course-specific documents, and by creating a bidirectional connection between the BKT mastery model and the LLM generation pipeline so that what the student knows dynamically shapes what the AI says.

1.11 Conclusion

This chapter has systematically synthesized the critical theoretical transition from traditional teaching methods toward highly adaptive, intelligent educational models [3]. We established that the evolution of educational paradigms creates an urgent need for systems capable of genuine real-time adaptation. We demonstrated that Artificial Intelligence, through Intelligent Tutoring Systems and Educational Data Mining, provides the analytical foundation for understanding student behavior. We then showed that Large Language Models transform the interaction model by enabling natural conversational tutoring, and that Retrieval-Augmented Generation resolves the hallucination and grounding problems of generic LLMs by anchoring every response to validated course materials. We subsequently introduced the Digital Twin concept, tracing its origins in industrial engineering and its conceptual adaptation to the educational domain as a Learner-Centric model driven by Bayesian Knowledge Tracing. We examined how combining AI analytics with a live Digital Twin creates a continuous physical-digital feedback loop that enables genuinely predictive, proactive personalization. Through comparative benchmarking, we demonstrated that no existing system currently achieves this combination comprehensively. Through structured gap analysis organized around three pillars, namely software implementation fidelity,

academic grounding strictness, and cognitive-generative interoperability, we identified the precise research contribution of this thesis.

Proposed Solution. The proposed system, **EduTwin AI**, is the first integrated architecture to simultaneously deliver: a live, BKT-powered cognitive model of the university learner updated per interaction; a strict RAG-based pedagogical filter that grounds all AI responses exclusively in course-validated materials; and a dynamic prompt augmentation mechanism that directly connects the learner's current mastery state to the content generation pipeline, producing responses precisely calibrated to where the student is in their learning journey at any given moment. Having established this theoretical foundation and identified the research gap, the next chapter will detail the system analysis, functional and non-functional requirements, actor modeling, and the complete conceptual design of the EduTwin AI solution.

CHAPTER 2

System Analysis and Conceptual Design

2.1 Introduction

Building upon the theoretical scaffolding erected in Chapter 1, which exposed the pedagogical inadequacy of generic large language models, articulated the promise of Learner-Centric Digital Twins as adaptive surrogates of the individual learner, and established Retrieval-Augmented Generation as a principled mechanism for grounding AI responses in course-specific knowledge, this chapter undertakes the transition from conceptual argument to implementable design. The work proceeds through three methodologically distinct yet interdependent phases: requirements elicitation, UML functional, structural, and behavioral modeling, and data architecture definition, with each phase treating its predecessors as input and its successors as consumers of its outputs. Design decisions are not made arbitrarily; each is anchored either in an empirical finding from the learning-sciences literature or in a verifiable constraint imposed by the deployment environment. Requirements are prioritized using the MoSCoW framework, namely Must-Have, Should-Have, and Could-Have, to enforce scope discipline and communicate stakeholder expectations with precision, while behavioral and quality attributes are classified in conformance with IEEE 29148:2018 [20] for requirements engineering and ISO/IEC 25010:2023 [21] for software quality characterization. The chapter unfolds as follows: Section 2.2 develops the UML functional and structural model, encompassing platform objectives, actor identification, use case analysis, and the domain class diagram; Section 2.3 addresses UML behavioral modeling through sequence and activity diagrams covering the authentication flow, the RAG query pipeline, and the Digital Twin update cycle; Section 2.4 treats data and knowledge modeling, specifying the relational and vector database strategies; and Section 2.5 closes the chapter with a synthesis of design decisions.

This section outlines the foundational requirements of the EduTwin AI platform, establishing the precise functional and non-functional specifications that the architecture must satisfy to effectively support the learner's cognitive development.

2.1.1 Functional Requirements

The functional requirements presented in this section were derived through a structured elicitation process combining three complementary sources: a systematic review of the learning-sciences literature surveyed in Chapter 1, from which pedagogical obligations such as knowledge tracing and adaptive difficulty adjustment were extracted; a problem-domain analysis of the target deployment context, namely Algerian university students engaged in self-directed study from instructor-distributed course materials without access to intelligent tutoring infrastructure; and an architectural feasibility study that translated abstract pedagogical goals into observable, implementable system behaviors. Each requirement was subsequently validated against the MoSCoW prioritization framework to distinguish core system commitments from desirable functional extensions.

Table 2.1 presents the complete functional requirements catalogue. Each entry specifies an observable, testable system behavior in conformance with IEEE 29148:2018 [20]. Implementation-prescriptive language has been deliberately avoided throughout in order to preserve design flexibility during the development phase.

ID	Requirement	Description	Priority
FR-01	Authentication	The system shall support secure student authentication using JWT-protected API endpoints, with stateless token-based access control and expiry-based session lifetime management.	Must-Have
FR-02	Document Upload and Ingestion	The student shall upload course PDFs. The system shall validate the file, extract text, chunk it into semantically coherent segments, generate embeddings, and persist them into the vector store.	Must-Have

FR-03	Semantic Retrieval (RAG)	Upon a student query, the system shall perform cosine-similarity search over the student's vector collection (top-k=5), assemble a context-enriched prompt including twin mastery context, and invoke the LLM to generate a grounded response.	Must-Have
FR-04	Cognitive Profile Management	The system shall maintain a persistent learner profile per student, recording mastery scores per topic using Bayesian Knowledge Tracing (BKT), and engagement metrics.	Must-Have
FR-05	Adaptive Assessment Engine	The system shall generate adaptive quizzes whose difficulty dynamically adjusts based on the student's current BKT mastery vector, targeting the Zone of Proximal Development.	Must-Have
FR-06	Learning Path Generation	The system shall generate a personalized sequential learning path by ranking topics according to BKT mastery scores, prioritizing areas of lower mastery for review.	Should-Have
FR-07	Digital Twin Synchronization	After each student interaction, the system's LearnerDigitalTwin shall be updated atomically: cognitive mastery state and engagement metrics shall be recomputed and persisted, and predicted performance shall be computed on demand via LLM-based insight generation.	Must-Have
FR-08	Conversational Tutor Interface	Students shall interact with a natural-language tutor interface supporting multi-turn dialogue, session context retention, and citation of source documents.	Must-Have
FR-09	Progress Dashboard	Students shall access a personal dashboard displaying mastery levels per topic, interaction history, and next-topic recommendations.	Should-Have

FR-10	Document Validation	Upon upload, the system shall validate that the file is a readable PDF with extractable text and is not empty. Invalid files shall be rejected with a descriptive error message.	Must-Have

Table 2.1: Functional Requirements Catalogue

2.1.2 Non-Functional Requirements

Table 2.2 specifies the non-functional requirements governing the system's quality attributes. These constraints directly shaped the architectural decisions presented in Section 2.2. The quality attribute categories, encompassing performance, security, maintainability, usability, availability, portability, and data integrity, are aligned with the ISO/IEC 25010:2023 product quality model [21]. Particular attention is accorded to portability (NFR-06), which reflects a deliberate architectural policy: EduTwin AI must remain fully deployable on standard academic hardware at Algerian universities without any reliance on commercial cloud infrastructure, thereby ensuring institutional accessibility and long-term operational sustainability.

ID	Category	Specification	Metric
NFR-01	Performance	RAG query time-to-first-byte (TTFB) shall not exceed 6 seconds under normal single-user conditions on the target deployment hardware.	p95 < 6s
NFR-02	Privacy and Security	All student cognitive profiles and interaction logs shall be stored securely. Personally identifiable information shall not be exposed to the LLM in any prompt.	Zero PII leakage

NFR-03	Maintainability	Core modules (BKT engine, RAG pipeline, authentication) shall be independently testable and loosely coupled, with clear interface boundaries enabling isolated unit and integration testing within the monolithic deployment.	Modular architecture
NFR-04	Usability	The system interface shall be in English only, while learner-facing output (results, feedback, recommendations) shall support Arabic, French, and English.	system multilingual output (AR/FR/EN)
NFR-05	Availability	Specification: The system shall remain operational throughout active study sessions without unhandled crashes.	No unhandled failures during testing
NFR-06	Portability	The system shall be reachable from any web-enabled device via a standard browser interface, ensuring broad accessibility without requiring specialized software installation or proprietary cloud dependencies.	Cross-device web access
NFR-07	Data Integrity	All **LearnerDigitalTwin** updates shall be transactional. No partial cognitive state writes shall be committed in the event of a backend failure.	ACID compliance

Table 2.2: Non-Functional Requirements Specification

2.2 UML Functional and Structural Modeling

This section defines the functional and structural blueprint of EduTwin AI: the objectives that motivate its design, the actors that interact with it, the use cases they drive, and the domain class model that underpins the entire platform. Taken together, these artifacts establish the system's interaction scope and static architecture before its runtime behavior is examined in Section 2.3.

2.2.1 Platform Positioning and Design Rationale

The design of EduTwin AI emerges from a concrete pedagogical gap observed in the Algerian university context: students engage with course materials distributed informally by their instructors, typically as PDF files shared through messaging platforms or portable storage media, yet have access to no intelligent tool capable of reasoning over those exact documents while simultaneously accounting for their individual cognitive state. Generic large language models, as established in Chapter 1, fail this requirement on both counts: they neither confine their responses to a student's specific curriculum nor maintain any persistent representation of that student's evolving knowledge. EduTwin AI is designed precisely to close this gap. Rather than restating the theoretical justification developed in the preceding chapter, this section identifies the specific behavioral commitments the platform must honour and establishes the actor model through which those commitments are operationalized.

2.2.2 Actor Identification and Role Definition

The actor model for EduTwin AI distinguishes between human actors who interact with the platform directly and external system actors that provide necessary infrastructure services. Notably, internal technical components (such as language models and databases) have been abstracted into backend use cases rather than external actors, adhering strictly to object-oriented design principles. Table 2.3 presents the updated actor catalogue, defining the functional role each plays relative to the system boundaries.

ID	Actor	Type	Role
A-01	User (Base)	Abstract Human Actor	The foundational entity representing any human interacting with the system. It encapsulates the shared behavior required for session initiation, meaning all concrete human actors inherit the ability to authenticate and manage their personal profiles.
A-02	Learner (Student)	Primary Human Actor	The end-user of the educational environment. Inheriting from the base user, this actor drives the core learning loop by submitting queries, uploading documents, taking adaptive assessments, and triggering the generation of study plans, mind maps, and podcasts.
A-03	System Administrator	Primary Human Actor	Responsible for platform governance. Inheriting from the base user, the administrator is granted elevated privileges to manage global user roles, assign permissions, and configure exam labs within the administrative suite.
A-04	Auth Provider	Secondary System Actor	An external identity service that intercepts and validates login attempts. It processes SSO or local credential verification and issues the authorization tokens required for the base user to access the platform.
A-05	TTS API Service	Secondary System Actor	A third-party text-to-speech integration invoked exclusively by the podcast generation workflow to synthesize natural-sounding audio from AI-generated transcripts.

Table 2.3: Updated Actor Catalogue

2.2.3 Use Case Diagram: Overview and Structural Analysis

Figure 2.1 illustrates the UML use case diagram for the EduTwin AI platform. The structural model adopts actor generalization and groups the system's behavior into four logical packages: Core Access, Learning Tools, Admin Tools, and Backend Operations (Internal). By converting underlying technical infrastructure (such as the RAG pipeline and vector storage) into explicit internal inclusion dependencies, the diagram accurately maps what the platform provides to external entities versus how it processes requests internally.

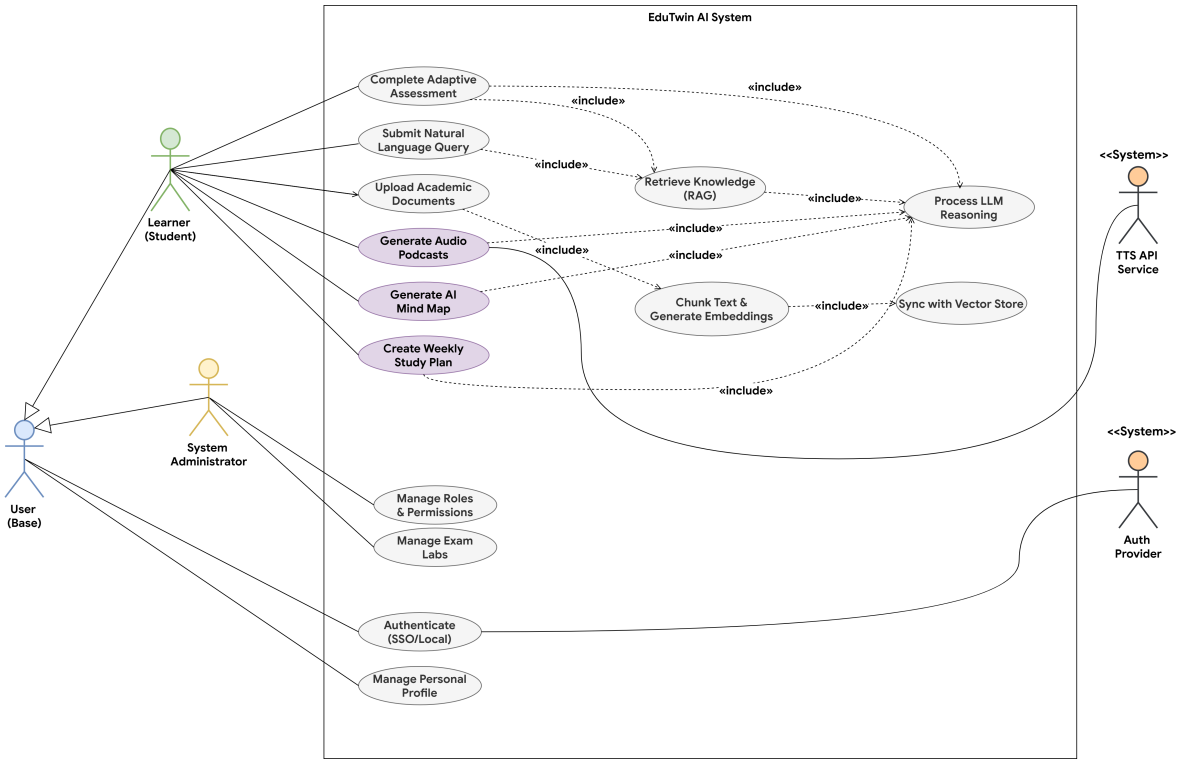


Figure 2.1: EduTwin AI System – Use Case Diagram

From a structural modeling perspective, the architecture relies strictly on two UML relationship types to manage complexity: *association* and ⟨⟨*include*⟩⟩. The *association* relationship defines direct participation between an actor and a specific use case. The ⟨⟨*include*⟩⟩ stereotype specifies a mandatory behavioral dependency; the base use case unconditionally delegates execution to the included use case, modularizing repetitive or complex backend operations.

Several ⟨⟨*include*⟩⟩ relationships dictate the system's core execution flow. First, *Upload Academic Documents* mandates the inclusion of *Chunk Text & Generate Em-*

beddings, which subsequently includes *Sync with Vector Store*, modeling the complete, uninterruptible ingestion pipeline. Second, *Submit Natural Language Query* unconditionally delegates to *Retrieve Knowledge (RAG)*, which in turn includes *Process LLM Reasoning*, enforcing the constraint that no query can bypass the retrieval phase. Third, pedagogical features such as *Complete Adaptive Assessment*, *Generate AI Mind Map*, and *Generate Audio Podcasts* all structurally depend on *Process LLM Reasoning*, isolating the language model inference logic into a single, reusable backend component.

From an actor mapping perspective, core infrastructure components are explicitly abstracted as internal behaviors within the Backend Operations (Internal) package. Consequently, the only secondary system actors present on the diagram's periphery are the Auth Provider and the TTS API Service, representing true external integrations that operate outside the platform's direct control envelope.

2.2.4 Actor Interaction Model and System Boundaries

Rather than treating the platform as a collection of independent databases and engines, the interaction model views EduTwin AI as a unified boundary. The following sections detail the flow of control across the updated subsystems, starting from initial access.

- **Core Access and Identity Delegation:** Access control is centralized within the Core Access module, relying entirely on the inherited traits of the abstract *User (Base)* actor. Regardless of whether a student or an administrator is logging in, the session must be established through the *Authenticate (SSO/Local)* use case. The platform delegates the actual verification of credentials to the external **Auth Provider**. Once the provider validates the request and returns an authorization token, the base user is permitted to manage their personal profile and enter the system. After a successful login, the interaction pathways diverge strictly based on the actor's concrete role: the **System Administrator** interacts exclusively with the Admin Tools, while the **Learner** is routed to the Learning Tools environment.
- **Learning Materials and Ingestion Pipeline:** The ingestion pipeline is triggered when the *Learner* executes the *Upload Academic Documents* use case. The system

first validates the document for extractable text. Upon passing validation, the platform initiates the *Chunk Text & Generate Embeddings* process, applying a sliding-window segmentation strategy to preserve inter-sentence context. These vector embeddings are subsequently committed to storage via the *Sync with Vector Store* use case. This internal boundary ensures that all document chunks are strictly scoped to the individual student's collection, maintaining tenant isolation at the data layer without exposing the storage mechanism to the user.

- **AI Interaction and Knowledge Retrieval:** The primary conversational interface is governed by the *Submit Natural Language Query* use case. When a query is dispatched, the system includes the internal *Retrieve Knowledge (RAG)* process, which embeds the user's input and executes a cosine-similarity search against the vector store to retrieve the top-k relevant chunks. The assembled context is then passed to the *Process LLM Reasoning* use case, which abstracts prompt augmentation and returns only the grounded final response to the Learner interface.
- **Cognitive Tracking and Adaptive Assessment:** State synchronization is managed through the *Complete Adaptive Assessment* use case. After each evaluation cycle, the system recomputes the student's mastery probabilities using the Bayesian Knowledge Tracing (BKT) model defined in Chapter 1, updating the four core parameters based on assessment outcomes and persisting them with ACID compliance. The system uses this data to pinpoint bottlenecks directly within the student's Zone of Proximal Development (ZPD).
- **Generative Content and Study Planning:** The *Generate AI Mind Map* and *Create Weekly Study Plan* use cases allow the Learner to transform static courseware into structured formats, both internally depending on *Process LLM Reasoning*. The study planner further includes a database synchronization step to persist the generated schedule. Multimodal learning is supported via the *Generate Audio Podcasts* use case, which transmits a pedagogically structured script to the external **TTS API Service** for audio synthesis.
- **Exclusion of the Instructor Actor:** The deliberate exclusion of an Instructor actor warrants explicit justification. In the target deployment environment at Algerian universities, instructors typically distribute course materials through

informal channels and cannot be assumed to maintain active interfaces with a dedicated cognitive platform. Mandating instructor participation would introduce an availability dependency structurally incompatible with the project's accessibility objectives. Consequently, EduTwin AI assigns the student full ownership of their knowledge base, rendering the system entirely self-contained. This design decision is modular; future iterations could introduce an Instructor actor without requiring fundamental revisions to the existing use case model.

2.2.5 Textual Description of Major Use Cases

Three critical use cases are detailed below, conforming to the IEEE specification template. These interactions represent the foundational lifecycle of a study session: content ingestion (UC-01), cognitive query execution (UC-02), and adaptive evaluation (UC-03).

UC-01 : Upload Course PDF

Field	Description
Use Case ID	UC-01
Use Case Name	Upload Academic Documents
Primary Actor	Learner (Student)
Secondary Actors	None (All persistence handled by internal Backend Operations)
Trigger	The student selects an academic document from their device and submits the upload form.
Pre-conditions	The student holds a valid, authenticated session.

Main Success Flow	1. Student uploads a course document via the Presentation Tier. 2. The system validates the file format and ensures text extractability. 3. The base use case invokes the *Chunk Text & Generate Embeddings* internal operation. 4. The text is segmented into fixed-token chunks with defined overlaps. 5. The system encodes these chunks into dense vector representations. 6. The *Sync with Vector Store* operation is triggered, persisting the embeddings into a storage partition isolated by the student's ID. 7. The system commits the associated metadata (e.g., page numbers, source ID) to the relational schema. 8. The interface returns a success confirmation displaying the number of indexed chunks.
Alternative Flows	**Alt-A** (Validation Failure): If the file is unreadable or empty, the pipeline terminates immediately. The system returns an error message, and no backend synchronization is triggered.
Post-conditions	The vector store contains newly indexed embeddings ready for similarity search.
Related FRs	FR-02 (Document Upload and Ingestion)
Related NFRs	NFR-02 (Privacy), NFR-07 (Data Integrity)

Table 2.4: Use Case UC-01: Upload Academic Documents

UC-02 : Submit Natural Language Query

Field	Description
Use Case ID	UC-02
Use Case Name	Submit Natural Language Query
Primary Actor	Learner (Student)
Secondary Actors	None (Inference and retrieval handled internally)
Trigger	The student submits a question via the conversational tutor interface.
Pre-conditions	The student is authenticated and has at least one successfully ingested document.
Main Success Flow	1. Student submits question Q. 2. The system internally fetches the student's current cognitive state (BKT mastery metrics). 3. The system invokes the *Retrieve Knowledge (RAG)* internal operation, encoding Q into a query vector. 4. A cosine-similarity search returns the top-k relevant text chunks from the vector store. 5. An augmented prompt is assembled, concatenating the chunks, the cognitive state, and Q. 6. The system invokes *Process LLM Reasoning* to generate a response constrained by the retrieved context. 7. The final response is returned to the interface alongside source citations. 8. The system records the interaction to update the student's cognitive metrics asynchronously.

Alternative Flows	**Alt-A** (Low Similarity Score): If no retrieved chunks meet the relevance threshold, the system skips the inference phase and returns a predefined response indicating a lack of relevant course material.
Post-conditions	The interaction is logged, and the asynchronous digital twin synchronization cycle is scheduled.
Related FRs	FR-03 (Semantic Retrieval), FR-08 (Conversational Tutor Interface)
Related NFRs	NFR-01 (Performance), NFR-04 (Usability)

Table 2.5: Use Case UC-02: Submit Natural Language Query

UC-03 : Complete Adaptive Assessment

Field	Description
Use Case ID	UC-03
Use Case Name	Complete Adaptive Assessment
Primary Actor	Learner (Student)
Secondary Actors	None
Trigger	The student initiates a topic-specific evaluation from their dashboard.
Pre-conditions	The student is authenticated and possesses a baseline cognitive profile for the selected topic.

Main Success Flow	1. Student requests an assessment for topic T. 2. The system fetches the current BKT mastery probability $P(L_{n-1})$ for T. 3. The system invokes *Process LLM Reasoning* with a structured JSON schema to generate a question calibrated to the current mastery level. 4. Student submits their answer. 5. The system evaluates the correctness of the response. 6. The BKT update rule is applied to compute the new posterior mastery estimate $P(L_n)$. 7. Steps 3 through 6 repeat until the assessment cycle concludes. 8. The system commits the updated cognitive state to the backend store. 9. An analytical summary is presented to the student, highlighting the mastery delta.
Alternative Flows	**Alt-A** (Inference Timeout): If *Process LLM Reasoning* fails to return a parsed JSON question within the timeout window, the system falls back to a static pre-generated question bank to avoid interrupting the assessment.
Post-conditions	The student's internal cognitive profile reflects the newly calculated BKT parameters, potentially triggering a learning path recalculation.
Related FRs	FR-04 (Cognitive Profile Management), FR-05 (Adaptive Assessment Engine)
Related NFRs	NFR-01 (Performance), NFR-07 (Data Integrity)

Table 2.6: Use Case UC-03: Complete Adaptive Assessment

2.2.6 Class Diagram and Domain Model

Figure 2.2 presents the object-oriented class diagram of EduTwin AI, constituting the system's formal domain model. The diagram organizes thirteen classes across four conceptual layers, each representing a distinct concern boundary within the platform's architecture. The deliberate layering reflects a conscious application of the Separation of Concerns principle [31], ensuring that changes within one layer propagate minimally to adjacent layers and that each layer can be reasoned about, tested, and extended independently.

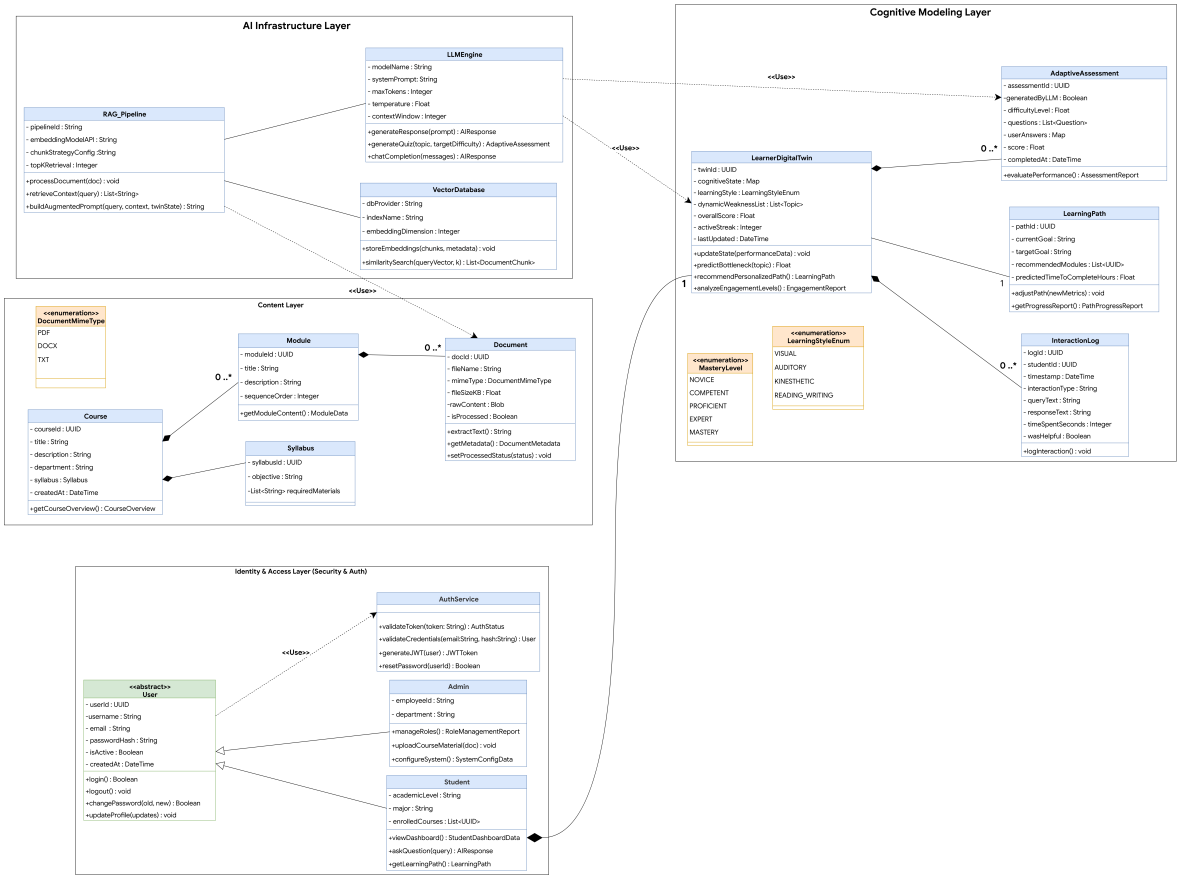


Figure 2.2: Class Diagram and Domain Model of EduTwin AI

- **Layer 1: Identity and Access Layer:** The identity and access layer comprises the abstract **User** base class, its concrete subclasses **Student** and **Admin**, and the **AuthService** class. By relying on polymorphic generalization, the architecture strictly adheres to object-oriented design principles. Shared attributes such as credentials, email, and timestamps are centralized within the abstract **User** class. The **Student** and **Admin** subclasses inherit these foundational properties while extending them with privilege-specific attributes and methods. Furthermore, **AuthService** is decoupled from these data entities in conformance

with the Single Responsibility Principle; it exclusively encapsulates token issuance, session validation, and SSO integration logic.

- **Layer 2: Content Layer:** The content layer comprises the **Course**, **Module**, and **Document** classes, modeling the hierarchical structure of the academic materials that constitute the student's personal knowledge base. A **Course** is composed of zero or more **Module** instances, each representing a discrete topical unit traceable to the BKT mastery vector maintained by the cognitive layer. A **Module** is, in turn, composed of zero or more **Document** instances, each corresponding to an uploaded PDF file. The **Document** class persists metadata including an **isValidated** flag that records whether each file successfully traversed the ingestion pipeline, and a **chunkCount** attribute recording the number of vector embeddings generated. This layer is isolated from the cognitive and AI infrastructure layers by design: the content model is agnostic to how its documents are processed or retrieved.

- **Layer 3: Cognitive Modeling Layer:** The cognitive modeling layer is the most semantically rich layer of the domain model, comprising the **LearnerDigitalTwin**, **AdaptiveAssessment**, **LearningPath**, and **InteractionLog** classes. All knowledge tracing, difficulty adaptation, and learning path generation logic is confined within this boundary.

  LearnerDigitalTwin is the central entity of the entire domain model, bound to **Student** through a 1:1 composition relationship. The composition semantics reflect the ontological commitment that a digital twin cannot exist independently of the learner it represents. The twin maintains the **bktMasteryVector**, a JSON-serialized map of topic identifiers to posterior mastery probabilities, the **learningStyleClass** derived from interaction pattern analysis, and **engagementMetrics** that capture longitudinal study behavior. The **lastSyncedAt** timestamp records the moment of the most recent atomic synchronization.

  AdaptiveAssessment is associated with both **Student** and **LearnerDigitalTwin**, with a learner's digital twin informing zero or more assessments, reflecting that an assessment is both administered to a student and informed by their current cognitive state. The **difficultyLevel** attribute is a derived value recomputed from the BKT mastery vector at the time of assessment generation. **LearningPath** encodes the system-generated ordered sequence of modules recommended for the student's next study session, recomputed whenever the digital twin synchronization cycle detects a mastery threshold crossing. **InteractionLog** provides a granular, immutable record of every student-system exchange, with each **LearnerDigitalTwin** associated with zero or more log entries, constituting the primary data source for twin update operations.

- **Layer 4: AI Infrastructure Layer:** The AI infrastructure layer comprises the **RAG_Pipeline**, **LLMEngine**, and **VectorDatabase** classes. This layer is isolated from the domain model's upper layers to enforce a critical architectural boundary: the pedagogical and content concerns of the platform must remain independent of the specific AI components employed for retrieval and generation.

  A critical design decision visible in the diagram is the decomposition of **RAG_Pipeline** and **LLMEngine** into two distinct classes in strict conformance with the Single Responsibility Principle. The **RAG_Pipeline** class encapsulates exclusively the retrieval concern: query embedding, cosine-similarity search execution against the **VectorDatabase**, chunk assembly, and augmented prompt construction. The **LLMEngine** class encapsulates exclusively the inference concern: receiving a fully assembled prompt and returning a generated response. This decomposition ensures that the retrieval strategy can be revised without modifying the generation layer, and vice versa. **VectorDatabase** is modeled as a class rather than an external actor because its per-student namespace scoping constitutes domain logic rather than mere infrastructure configuration.

2.3 UML Behavioral Modeling

The structural models presented in Section 2.2 establish what the system contains. The behavioral models presented in this section establish how it acts. Four UML diagrams are developed here, each targeting a distinct interaction flow of increasing complexity: the authentication sequence that governs session establishment, the RAG query pipeline that constitutes the platform's primary runtime loop, the Digital Twin update cycle that drives cognitive personalization, and the activity diagram that maps the RAG pipeline's control flow across component boundaries. Taken together, these diagrams provide a complete behavioral specification of EduTwin AI's three most architecturally significant runtime scenarios.

2.3.1 Sequence Diagram: Authentication Flow

Before any learning interaction can occur, the platform must establish a verified session. Figure 2.3 models this prerequisite through the authentication sequence, which involves three participants: the User, the EduTwin Application, and the Database. The diagram is organized into two named phases.

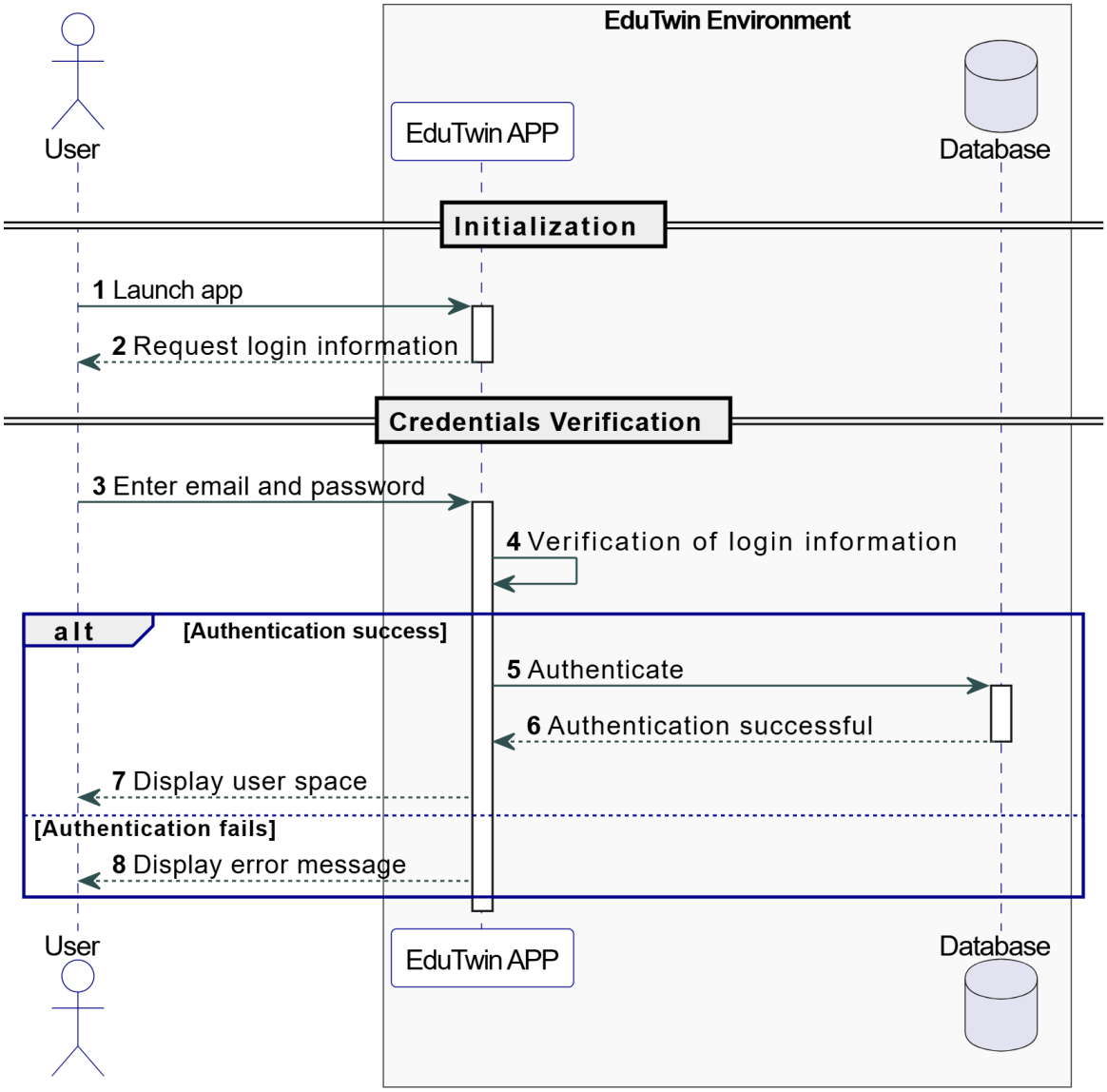


Figure 2.3: Sequence Diagram: Authentication Flow

Phase 1 (Initialization) begins the moment the user launches the application. The EduTwin APP responds by rendering the login interface and issuing a request for credentials. Nothing else happens until this phase completes: no learning data is fetched, no twin context is loaded, and no backend service is contacted. This ordering reflects a deliberate security boundary, not merely a UI convention.

Phase 2 (Credentials Verification) is where identity is asserted. The user submits their email and password, and the application forwards them to the Database for verification. A UML **alt** fragment captures the two possible outcomes. On successful authentication, the Database confirms the identity, a session is established, and the user's personal workspace is rendered. On failure, a descriptive error is returned and access is denied. The significance of modeling this

as an **alt** fragment rather than two separate flows is that it makes the mutual exclusivity of the outcomes explicit at the specification level. More importantly, all subsequent sequence diagrams in this section reference this flow via a UML **ref** fragment, avoiding redundant repetition while maintaining traceability across the behavioral model.

2.3.2 Sequence Diagram: Interaction Processing and RAG Query Pipeline

With a verified session in place, the student may now interact with the platform's core intelligence. Figure 2.4 traces the complete message sequence from the moment a student submits any interaction, whether a free-text question, a quiz response, or an assignment submission, through to the delivery of a personalized, curriculum-grounded response. Unlike a conventional question-answering pipeline that treats all inputs uniformly, this diagram captures a more nuanced reality: the retrieval strategy applied to an interaction is determined dynamically by its type, and the response generated is conditioned not only on the retrieved course content but on the student's current cognitive state. Seven participants are involved: Student, Frontend UI, Backend API, Embedding Model, Vector Database, LLM Engine, and Learner Digital Twin. The diagram is partitioned into five named phases and opens with a ref to the Authentication Flow established in Section 2.3.1, reflecting the invariant that this pipeline is only reachable from an authenticated session.

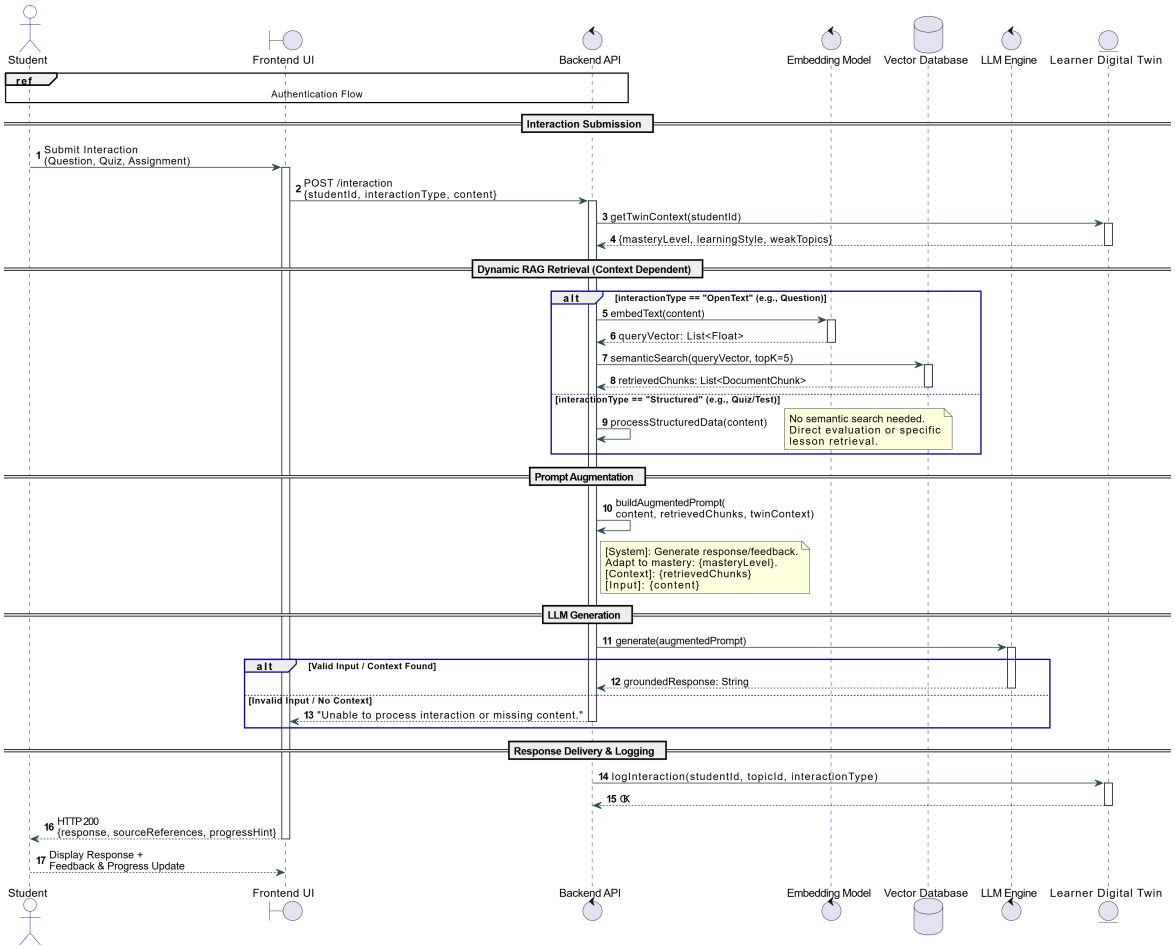


Figure 2.4: Sequence Diagram: Interaction Processing and RAG Query Pipeline

Phase 1 (Interaction Submission) establishes the context for everything that follows. The student submits an interaction and the Frontend UI transmits a **POST /interaction** request carrying **{studentId, interactionType, content}** to the Backend API. Before any retrieval begins, the backend invokes **getTwinContext(studentId)** against the Learner Digital Twin, which returns the student's current mastery level, learning style, and weak topics. The sequence of this retrieval preceding the RAG phase is not incidental: the twin context must be in hand before prompt augmentation can be performed in Phase 3, and moving it any later in the sequence would introduce a data dependency that could only be resolved by blocking the retrieval step.

Phase 2 (Dynamic RAG Retrieval) is where the interaction type determines the retrieval strategy. An **alt** fragment branches on **interactionType**. For **OpenText** interactions, the backend embeds the content into a dense query vector via the Embedding Model and the Vector Database executes a cosine-similarity search returning the top-5 most relevant document chunks as a **List<DocumentChunk>**. For **Structured** interactions such as quiz submissions,

no semantic search is needed: the content is the evaluation input and the backend processes it directly through **processStructuredData(content)**. This distinction is architecturally important because it avoids unnecessary embedding and retrieval overhead for interactions whose content is already self-contained, preserving the latency budget mandated by NFR-01.

Phase 3 (Prompt Augmentation) is an internal backend operation that assembles three inputs into a single structured prompt via **buildAugmentedPrompt(content, retrievedChunks, twinContext)**. The system directive instructs the LLM to adapt its response to the student's mastery level, the context block is populated by the retrieved document chunks, and the input block carries the original student content. It is through this three-part structure that the two core commitments of the platform, curriculum grounding and cognitive personalization, are simultaneously enforced in every generated response. Neither commitment could be satisfied by the prompt template alone without the other.

Phase 4 (LLM Generation) handles the two possible outcomes of the augmentation phase through an **alt** fragment conditioned on input validity and context availability. When valid context is present, the LLM Engine generates a **groundedResponse: String** conditioned exclusively on the retrieved course material. When no valid context exists, the system returns the message **"Unable to process interaction or missing content"** directly to the frontend without invoking the LLM. This is the hallucination prevention boundary of the platform, and its placement here, between augmentation and delivery, ensures that the LLM is never called with an empty or unconstrained prompt under any execution path.

Phase 5 (Response Delivery and Logging) closes the interaction cycle. The backend commits the interaction record via logInteraction(studentId, topicId, interactionType) before returning the final HTTP 200 response carrying response, source References, progressHint to the frontend, which renders the answer alongside source citations and a progress update. Logging precedes delivery deliberately: the interaction record must be persisted before the student receives feedback to guarantee audit trail integrity regardless of what happens on the client side. This ordering also ensures that the Digital Twin Update Cycle described in Section 2.4.3 always operates on a complete interaction record rather than a partial one.

2.3.3 Sequence Diagram: Post-Interaction Cognitive State Synchronization and Adaptive Response Generation

Every interaction that passes through the RAG pipeline produces an implicit question: does the student now know something they did not know before? The diagram presented in Figure 2.5 is the mechanism through which the system answers that question and acts on the answer. It models two tightly coupled processes that must execute in strict order: first, the synchronization of the student's cognitive state with the observed interaction outcome; second, the generation of a response whose complexity is calibrated to the freshly updated knowledge estimate. The diagram involves five participants: Student, Frontend UI, Backend API, Profile Database, and LLM Engine, and opens with a **ref** to the Authentication Flow, reflecting the invariant that cognitive state updates are only triggered from within an authenticated session.

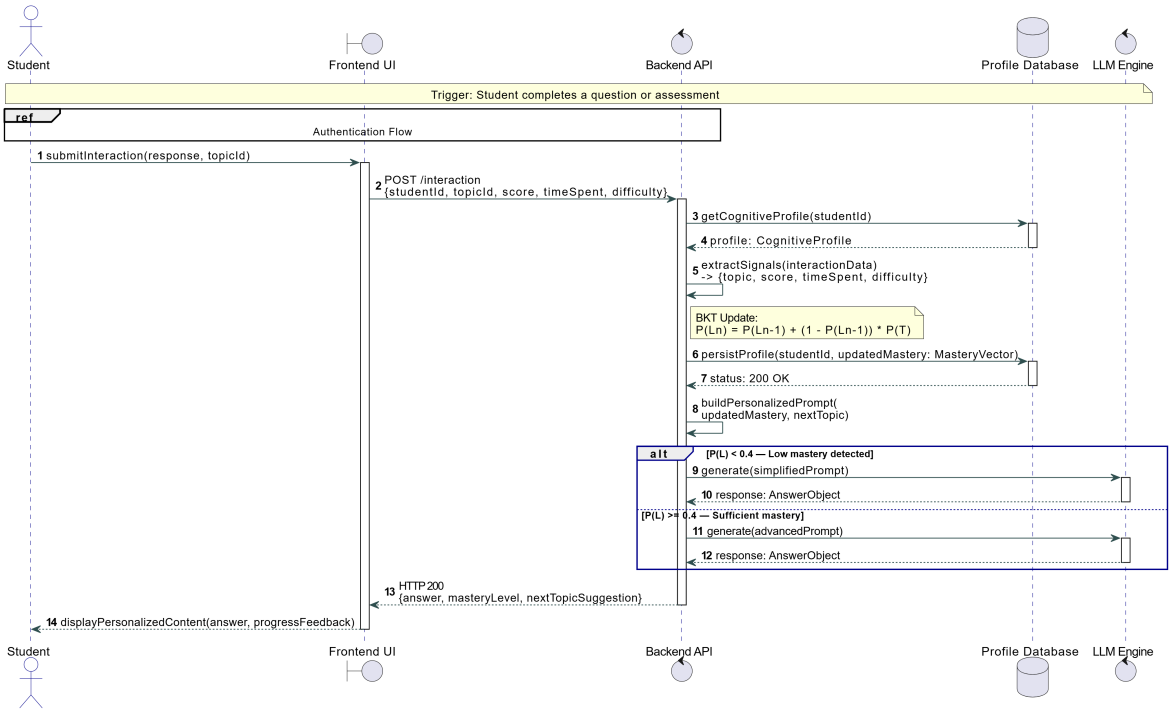


Figure 2.5: Sequence Diagram: Post-Interaction Cognitive State Synchronization and Adaptive Response Generation

Phase 1 (Cognitive State Synchronization) is synchronous and blocking, and the reason for this is consequential. The backend receives the interaction payload **{studentId, topicId, score, timeSpent, difficulty}**, retrieves the student's current cognitive profile from the Profile Database, and extracts the behavioral signals needed by the BKT engine. The mastery update is then computed:

$$P(L_{t+1}) = P(L_t \mid \text{obs}_t) + \left[1 - P(L_t \mid \text{obs}_t)\right] \times P(T)$$

The result is persisted atomically before Phase 2 begins. Blocking here is not a performance oversight; it is a correctness requirement. Phase 2 will construct a personalized prompt using the updated mastery estimate, and if the persistence step were deferred, the prompt could be built from a stale cognitive state, producing a response calibrated to who the student was before this interaction rather than who they are now.

Phase 2 (Adaptive Response Generation) uses the freshly committed mastery estimate to determine which prompt strategy to apply. An **alt** fragment branches on the value of $P(L)$. When $P(L) < 0.4$, the student has not yet crossed the mastery threshold for the topic and the LLM is invoked with a scaffolded prompt designed to guide them through foundational concepts. When $P(L) \geq 0.4$, the student has demonstrated sufficient command of the material and the LLM is invoked with an advanced prompt targeting higher-order reasoning. This threshold operationalizes Vygotsky's Zone of Proximal Development [29]: the value 0.4 is not arbitrary but marks the boundary between what the student can achieve with support and what they can begin to achieve independently. The final response, carrying **{answer, masteryLevel, nextTopicSuggestion}**, is returned to the frontend and rendered as personalized content alongside a progress indicator, closing the post-interaction synchronization cycle.

Definition : Dense Text Embedding

A dense text embedding is a fixed-dimensional vector representation of a text sequence produced by a pre-trained neural encoder, such that semantically similar texts are mapped to geometrically proximate points in the vector space [32]. Unlike sparse keyword-based retrieval, which matches documents by exact term overlap, dense retrieval operates on semantic similarity measured by cosine distance, enabling the system to retrieve course content that is conceptually relevant to a student query even when no lexical overlap exists between the query and the document. In EduTwin AI, the same encoder instance is used for both document ingestion and query embedding, a hard requirement that ensures all vectors occupy the same semantic space and that cosine similarity scores are meaningful across the student's entire document collection.

2.3.4 Activity Diagram: RAG Query Pipeline

The sequence diagrams above specify the message-level behavior of the system from a temporal perspective. Figure 2.6 complements them by presenting the same RAG pipeline from a control-flow perspective, using UML swimlane notation to make component ownership of each

step unambiguous. Five swimlanes partition responsibility across the Student, Frontend UI, Backend API, Vector Database, and LLM Engine.

The flow originates in the Student swimlane at the initial node *Validate pedagogical question*, representing client-side input validation that occurs before any network transmission. This placement is deliberate: filtering malformed or empty inputs at the client boundary reduces unnecessary load on the backend and improves the perceived responsiveness of the interface. Upon successful validation, control passes to the Frontend UI, which submits the question, and then to the Backend API, which fetches the Learner Digital Twin context and converts the question to a query vector before handing control to the Vector Database swimlane.

The Vector Database executes a cosine-similarity search and returns the retrieved document chunks to the Backend API, where the diagram's most consequential decision node is encountered: *Are relevant chunks found?* The **[Yes]** branch leads to prompt construction and LLM invocation; the **[No]** branch generates a fallback message directly without touching the LLM Engine swimlane. The boundary between these two branches is the hallucination prevention line of the system. Both branches converge at a merge node where interaction metrics are committed to the Profile Database, after which the frontend displays the answer with source references and progress feedback, and the flow terminates at the activity final node in the Student swimlane.

What the swimlane layout makes visible that the sequence diagrams do not is the asymmetry in component involvement between the two branches. In the success path, five swimlanes are active. In the fallback path, only two are. This asymmetry is not a deficiency; it is evidence that the system degrades gracefully when retrieval fails, containing the failure within the Backend API boundary and never propagating it to the LLM Engine or the student's cognitive profile.

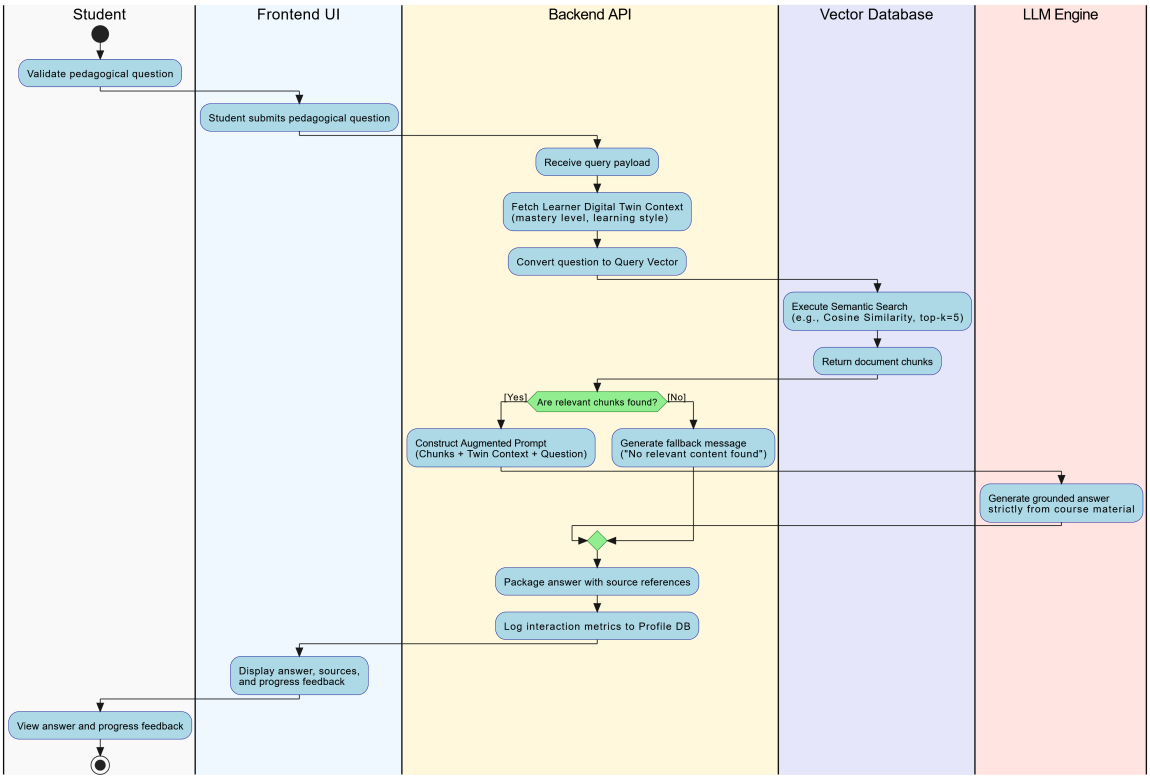


Figure 2.6: Activity Diagram: RAG Query Pipeline

2.4 Data and Knowledge Modeling

The behavioral and structural models developed in the preceding sections define how EduTwin AI acts and what it contains. This section addresses the third and final dimension of the system design: how the data that drives those behaviors is organized, persisted, and retrieved. Two complementary storage strategies are specified, each selected to match the nature of the data it manages. The relational schema governs all structured, transactional student data, while the vector store governs the high-dimensional semantic representations that underpin the RAG retrieval pipeline. Neither strategy alone would be sufficient; it is their combination that makes the platform's dual commitment to cognitive personalization and curriculum grounding technically viable.

2.4.1 Relational Database Schema

The relational schema of EduTwin AI comprises nine entities, each normalized to Third Normal Form (3NF) to eliminate transitive functional dependencies and prevent update anoma-

lies. Table 2.7 presents the complete entity inventory with primary keys, foreign keys, and key attributes for each entity.

Entity	Primary Key	Foreign Keys	Key Attributes
User	userId (UUID)	—	username, email, passwordHash, role (ENUM), createdAt
Student	studentId (UUID)	userId → User	academicLevel, enrolledCourses (junction table)
Course	courseId (UUID)	studentId → Student	title, syllabus, createdAt
Module	moduleId (UUID)	courseId → Course	topic, contentReferences (JSON)
Document	docId (UUID)	studentId → Student	fileName, fileType, uploadDate, isValidated (Boolean), chunkCount
LearnerDigitalTwin	twinId (UUID)	studentId → Student	cognitiveState (JSON), learningStyle, engagementMetrics, lastSyncedAt
InteractionLog	logId (UUID)	studentId, twinId	timestamp, actionType (ENUM), topicId, score, timeSpent, metadata (JSON)
AdaptiveAssessment	assessmentId (UUID)	studentId, moduleId	questions (JSON), currentDifficulty, completedAt
LearningPath	pathId (UUID)	studentId	sequentialModules (JSON), difficultyAdaptationIndex, generatedAt

Table 2.7: Relational Schema Entity Inventory

Several design decisions embedded in this schema warrant explicit justification. All primary keys use UUID v4 identifiers rather than auto-incrementing integers. This prevents the inference of entity counts from key values, a meaningful privacy property in a student data context, and preserves the option of distributing the database across nodes in a future iteration without introducing key collision risks.

The **cognitiveState** attribute of **LearnerDigitalTwin** and the **metadata** attribute of **InteractionLog** are stored as JSON columns, reflecting the variable and extensible nature of cognitive state representations. New cognitive dimensions, such as an additional affective state indicator or a refined engagement metric, can be appended to the JSON object without requiring a schema migration, preserving the operational continuity of deployed instances. This is a controlled and deliberate departure from strict relational normalization, justified by the write-once, read-atomically access pattern of both attributes: neither is ever queried at the sub-attribute level by the relational engine.

The separation of Course and Module into distinct entities eliminates the repeat-group anomaly that would arise from embedding module lists within a course record, and it is this separation that enables the BKT mastery vector to be indexed at the module level rather than the course level, providing the granularity required for meaningful Zone of Proximal Development targeting. The LearningPath entity stores sequential module references as a JSON-serialized ordered list, a controlled denormalization justified by the fact that learning paths are always read and written as atomic units and are never queried at the individual module reference level by the relational engine. The lastSyncedAt timestamp on LearnerDigitalTwin, renamed from lastUpdated to reflect its precise semantic role, records the moment of the most recent atomic synchronization triggered by the Digital Twin Update Cycle specified in Section 2.3.3, supporting consistency verification across the event-driven update pipeline.

2.4.2 Vector Database Strategy and Embedding Architecture

The vector store is the second pillar of EduTwin AI's data layer, and its design decisions are as consequential as those of the relational schema. An embedded vector store is selected for the platform, running as a library process rather than a separate server, in direct conformance with the portability requirement NFR-06. This eliminates an external infrastructure dependency that would otherwise complicate local deployment on academic hardware and increases the attack surface of the system.

Each student is assigned a dedicated vector collection identified by their **studentId**. This partitioning strategy, established in the actor model of Section 2.2 and enforced throughout the ingestion pipeline described in Section 2.3.2, provides three mutually reinforcing properties. Isolation ensures that a similarity query issued in one student's session cannot retrieve chunks from another student's document corpus, directly satisfying NFR-02. Independent scalability ensures that a student with a large document collection does not degrade retrieval latency for peers, preserving the performance guarantee of NFR-01. Parallel ingestion ensures that concurrent upload sessions from multiple students do not produce collection-level write conflicts, supporting the availability requirement of NFR-05.

Each stored chunk carries four metadata fields: the source document identifier, the page number, the chunk index within the document, and the upload timestamp. These fields serve two distinct purposes at query time. Post-retrieval filtering, such as restricting results to a specific document or a recent upload session, is implemented by querying the metadata rather than rerunning the similarity search, reducing latency. Citation generation, which surfaces the exact source passage alongside each generated response as specified in UC-02, depends entirely on this metadata being available at the chunk level. Without it, the platform could not satisfy its academic traceability commitment.

The embedding model identifier is stored as a collection-level metadata attribute rather than being hardcoded into the ingestion service. This architectural decision decouples the storage layer from the embedding strategy: when a higher-dimensional encoder becomes available that better suits the domain vocabulary of university-level course materials, the system creates a new collection under the updated model identifier and re-ingests existing documents in the background, preserving continuity of service without requiring a platform restart or a schema migration. The requirement that the same encoder instance be used for both ingestion and query embedding, established in Section 2.3 and enforced by the shared AI Infrastructure layer, is maintained across model migrations by the collection-level identifier: the RAG and Tutor Service always reads the model identifier from the collection metadata before embedding a query, ensuring that stale and current embeddings are never compared within the same similarity search.

2.5 Conclusion

This chapter has translated the theoretical foundations of Chapter 1 into a concrete system design through requirements elicitation, UML modeling, and data architecture definition. Three original design decisions define the EduTwin AI platform. First, assigning the student sole ownership of the knowledge base ensures operational autonomy, directly supporting the learner-centric objective under the specific deployment constraints of Algerian universities. Second, integrating the Learner Digital Twin context into the RAG pipeline elevates the system from a standard question-answering tool to an adaptive tutor that continuously evolves with the cognitive state of the user. Third, partitioning vector collections per student guarantees strict data privacy for requirement NFR-02 while maximizing retrieval precision.

While this conceptual design establishes a rigorous foundation, minor adjustments during the development phase remain an expected part of the iterative software engineering process [30]. Ultimately, the artifacts produced here, from the UML specifications to the relational schema, collectively form the implementation contract for Chapter 3. The following chapter will systematically translate this theoretical blueprint into a functional software system, demonstrating the platform's capabilities on local hardware using actual course materials.

CHAPTER 3

System implementation

3.1 Introduction

Chapter 2 translated the theoretical foundations of EduTwin AI into a formally specified system design, establishing a complete requirements catalogue, a multi-view UML model, and a normalized relational schema. The present chapter closes the engineering cycle by documenting the transition from those design artefacts into working software. Every architectural decision described in Chapter 2 is traced here to a concrete implementation choice, from the configuration of the work environment to the internal mechanics of the platform's most critical subsystems.

Rather than adopting a distributed microservices deployment, the system was implemented as a **Modular Monolith** to align with the development scope and time constraints of this thesis. The backend is powered by a single FastAPI application running within a unified Uvicorn process, yet the internal codebase strictly adheres to the proposed service boundaries: the five logical modules (Auth, Document Ingestion, RAG & Tutor, Digital Twin, and Assessment) are organized into highly decoupled routers and services. Asynchronous operations, most notably the BKT state update triggered after each student interaction, are handled via FastAPI's native background task mechanism, decoupling the synchronous student-facing response path from the cognitive state update cycle. This architectural decision reduced infrastructural complexity during the development phase without sacrificing the modular discipline required for a future transition to fully independent, separately deployable services.

The chapter is organized as follows. Section 3.2 describes the work environment, covering the software tools, programming languages, and frameworks employed throughout the project, with each choice justified against the constraints established in Chapter 2. Section 3.3 provides a guided tour of the platform's eleven user-facing interfaces: the authentication screens, the AI Tutor chat, the Document Manager, the Learner Digital Twin Dashboard, the Adaptive Assessment module, the AI Podcast Generator, the Flashcard engine, ExamLab, the Mind Map generator, the Study Planner, and the Administrator Portal. Section 3.4 examines the five critical technical subsystems in depth: the RAG pipeline and chunking strategy, the OpenRouter API LLM integration and evolution, the Bayesian Knowledge Tracing engine, the Digital Twin synchronization mechanism, and the prompt-engineering approach underpinning adaptive assessment generation. Section 3.5 concludes the chapter with a synthesis of implementation outcomes and a bridge toward future research directions.

3.2 Work Environment

The entire platform was developed on a Windows 11 (64-bit) workstation. Backend services ran inside isolated Python virtual environments to prevent dependency conflicts between the FastAPI application layer, the sentence-transformers embedding library, and the ChromaDB embedded vector store. Frontend tooling was managed entirely through the Node.js ecosystem. In line with NFR-06's portability requirement, the frontend is accessible through any standard web browser, while the backend runs natively on the host operating system without reliance on proprietary cloud infrastructure. This native deployment approach was adopted to accelerate the development feedback loop during the thesis timeline, while preserving the modular, dependency-isolated structure of both the backend and frontend environments for future deployment flexibility. All source code was maintained under version control on GitHub using a feature-branch workflow, ensuring that each major subsystem was developed and validated in isolation before integration into the shared main branch.

3.2.1 Architectural Design: The Modular Monolith Approach

EduTwin AI adopts a **Modular Monolith** three-tier architecture in which the presentation, application logic, and data management concerns are decomposed into strictly bounded, internally decoupled modules deployed within a single unified process. This architectural style was deliberately selected over a distributed microservices alternative for reasons grounded in the operational constraints of this thesis. The cognitive workloads inherent to EduTwin AI are heterogeneous in nature: document ingestion is a compute-intensive batch operation, RAG retrieval is a latency-sensitive synchronous process, and BKT state updates are lightweight but must be executed transactionally. Rather than decomposing these concerns into separately deployable services, the system organizes them into highly decoupled internal modules, each encapsulating a bounded context, while sharing a single runtime. This approach eliminates inter-service network overhead and deployment complexity without sacrificing the maintainability requirement NFR-03, since each module exposes a clean interface and can be modified independently. Asynchronous operations, most notably the BKT state update triggered after each student interaction, are handled via FastAPI's native background task mechanism, decoupling the synchronous student-facing response path from the cognitive state update cycle and satisfying the latency constraint imposed by NFR-01. The modular boundary discipline

enforced throughout the codebase ensures that transitioning to a fully distributed deployment in future work remains a straightforward objective requiring no architectural redesign.

Figure 3.1 presents the complete high-level architecture of EduTwin AI, organized across four structural zones: the Presentation Tier, the Application Tier comprising five domain-specific modules coordinated through a single FastAPI application, the Data Tier housing the dual-storage backend, and the AI Infrastructure layer containing the embedding model and language model engine.

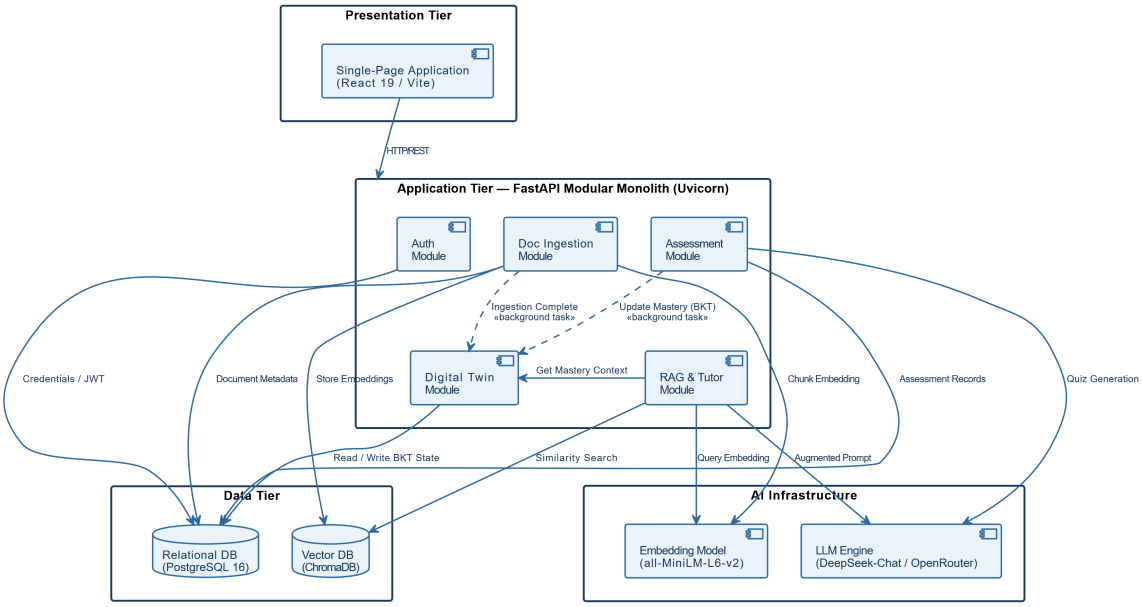


Figure 3.1: EduTwin AI – High-Level System Architecture

3.2.2 Three-Tier Architecture Model

- **Presentation Tier:**

  The Presentation Tier consists of a web-based Single-Page Application (SPA) built with React 19 that serves as the exclusive interface between the student and the platform's backend. The SPA communicates with the Application Tier exclusively through HTTP/REST calls, maintaining a strict separation between rendering logic and business logic. Authentication state is enforced at the routing layer through middleware that validates session tokens prior to rendering. The interface supports bidirectional text layout to accommodate both French and Arabic, satisfying the usability requirement NFR-04.

- **Application Tier:**

The Application Tier is implemented as a single FastAPI application organized into five domain-specific modules, each encapsulating a bounded context of the platform's business logic. A central routing layer within the application dispatches inbound requests to the appropriate module and enforces authentication context propagation. The five modules are as follows.

The **Auth Module** manages the full identity lifecycle: account registration, credential validation, JWT issuance, and session management. It is the only module authorized to issue and invalidate session tokens, centralizing identity governance in a single auditable unit.

The **Document Ingestion Module** orchestrates the multi-stage processing pipeline activated upon a student's PDF upload: document validation, text extraction, semantic chunking, embedding generation via the AI Infrastructure layer, and persistence to both storage backends. Upon successful ingestion, a background task notifies the system that new course content is available for retrieval.

The **RAG and Tutor Module** is the primary runtime module, activated on every student query. It retrieves the student's cognitive context, executes a cosine-similarity search over the Vector Database, assembles the augmented prompt, and dispatches it to the LLM Engine. Upon query completion, a background task triggers the asynchronous twin synchronization process.

The **Digital Twin Module** maintains the `LearnerDigitalTwin` record for each enrolled student in the Relational Database. It recomputes the BKT mastery vector after each interaction using the four-parameter update rule and commits the updated state transactionally, satisfying NFR-07. It also exposes a synchronous context endpoint enabling real-time mastery injection into RAG prompts.

The **Assessment Module** generates, delivers, and evaluates adaptive assessments. It queries the Digital Twin Module for the student's current mastery vector, requests JSON-structured assessment items from the LLM Engine calibrated to the appropriate difficulty tier, and records student responses. Evaluated outcomes are processed by a background task that updates the Digital Twin, closing the adaptive feedback loop.

- **Data Tier:**

  The Data Tier comprises two complementary storage systems. The **Relational Database** (PostgreSQL 16) stores all structured, transactional data: student profiles, BKT cognitive state vectors, interaction logs, assessment records, and document metadata. All write

operations are ACID-compliant, satisfying NFR-07. The **Vector Database** (ChromaDB) stores the high-dimensional embeddings produced during document ingestion, organized into per-student collections to enforce strict content isolation, satisfying both the privacy requirement NFR-02 and the data isolation principle.

3.2.3 AI Infrastructure Layer

The AI Infrastructure layer operates as a shared services zone consumed by multiple Application Tier modules. The **Embedding Model** (`all-MiniLM-L6-v2` via `sentence-transformers`) is invoked by both the Document Ingestion Module at ingestion time and the RAG and Tutor Module at retrieval time. Using a single shared encoder instance is a hard architectural requirement: representational consistency between stored and query embeddings is a prerequisite for meaningful cosine-similarity search. The **LLM Engine**, accessed via the OpenRouter gateway using the `deepseek/deepseek-chat` production model, is consumed by the RAG and Tutor Module for grounded response generation and by the Assessment Module for structured quiz item synthesis. All outputs are strictly conditioned on the retrieved course context, preventing open-ended generation disconnected from the student's curriculum.

3.2.4 Retrieval-Augmented Generation Pipeline

The RAG pipeline guarantees that every generated response is grounded in the student's own uploaded course materials. It operates across two temporally distinct phases.

The **Offline Ingestion Phase** is initiated upon PDF upload. The Document Ingestion Module validates the file, extracts its text, and applies a sliding-window segmentation strategy producing chunks of approximately 512 tokens with a 64-token overlap to preserve semantic continuity across boundaries [24]. Each chunk is embedded and persisted to the Vector Database under a per-student collection namespace, with metadata committed to the Relational Database within the same transaction.

The **Online Retrieval Phase** is activated on every student query. The query is embedded using the same encoder instance, and the Vector Database returns the top-k most semantically proximate chunks (k=5) [25]. These are assembled with the student's BKT mastery context into an augmented prompt dispatched to the LLM Engine. If no chunk exceeds the minimum similarity threshold, the system returns a bounded response acknowledging the absence of rele-

vant material, enforcing the curriculum-grounding commitment without permitting parametric hallucination.

3.2.5 Cognitive Digital Twin Engine

The Learner Digital Twin elevates EduTwin AI beyond a document-grounded Q&A system into a genuinely adaptive learning environment. The twin is a persistent data structure maintained by the Digital Twin Module, comprising three components: the BKT mastery probability vector indexed by topic, the learning style classification derived from interaction patterns, and longitudinal engagement metrics. It is updated after every interaction through the four-parameter BKT rule, producing a posterior mastery estimate that reflects the student's most recent demonstrated performance.

BKT [23] was selected over Deep Knowledge Tracing (DKT) [26] and Item Response Theory (IRT) [27] for three reasons specific to this deployment context. DKT requires a recurrent neural network trained on large historical datasets and GPU infrastructure unavailable on the target hardware. IRT requires pre-calibrated item difficulty parameters that cannot be obtained for dynamically generated assessment items. BKT, by contrast, operates on a per-topic binary mastery model whose parameters are initialized from published literature defaults [23] and updated online from a single student's interaction sequence, requiring no prior dataset and no GPU infrastructure. Its update rule is fully transparent and interpretable, satisfying an implicit accountability requirement absent from neural approaches.

The twin's current mastery vector is injected into every augmented prompt, enabling the LLM Engine to modulate response complexity to match the student's demonstrated level. This integration creates the adaptive feedback loop central to the platform's design: each interaction updates the twin, the updated twin shapes the next interaction, and the loop executes continuously throughout every study session.

3.2.6 Deployment and Configuration

To ensure environment reproducibility, the project relies on strict configuration management across its decoupled architecture. The backend and frontend environments are unified through a secure environment variable (`.env`) strategy, which centralizes the PostgreSQL connection strings, JWT secret keys, and OpenRouter API routing tokens.

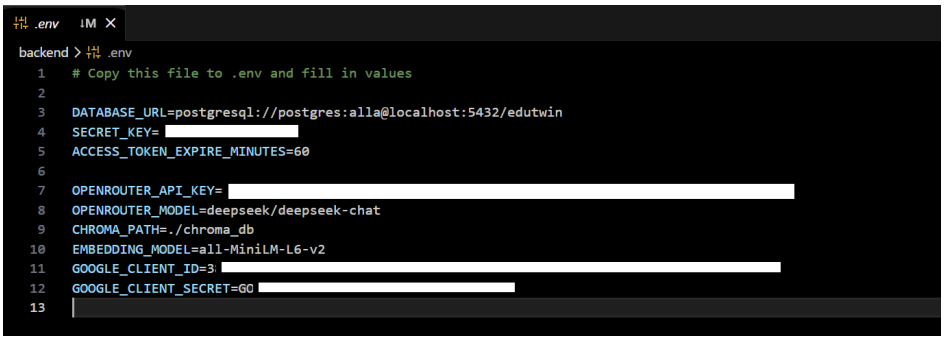


Figure 3.2: Centralized environment configuration file (.env) managing database connections and external API keys securely.

The local deployment sequence mandates initializing the PostgreSQL 16 service, activating the Python virtual environment to resolve backend dependencies, and launching the FastAPI server via Uvicorn. Concurrently, the frontend build toolchain is executed via the Vite development server.

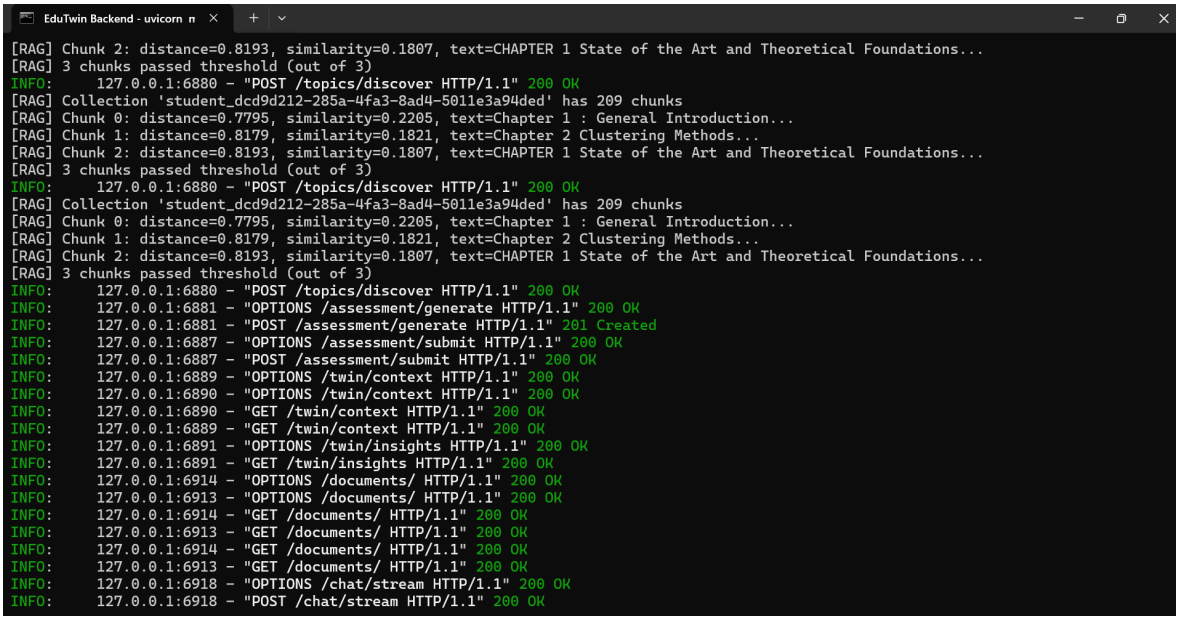


Figure 3.3: Simultaneous execution of the FastAPI backend service via Uvicorn and the React frontend environment via Vite.

Furthermore, stabilizing the development environment required specific tooling configurations. The integration of the Antigravity AI code editor, initially tracking the 'Latest' release channel, necessitated locking the version to a stable build to resolve intermittent update instabilities. This intervention ensured the consistent AI-assisted generation of the SQLAlchemy ORM models and React components without disrupting the workflow.

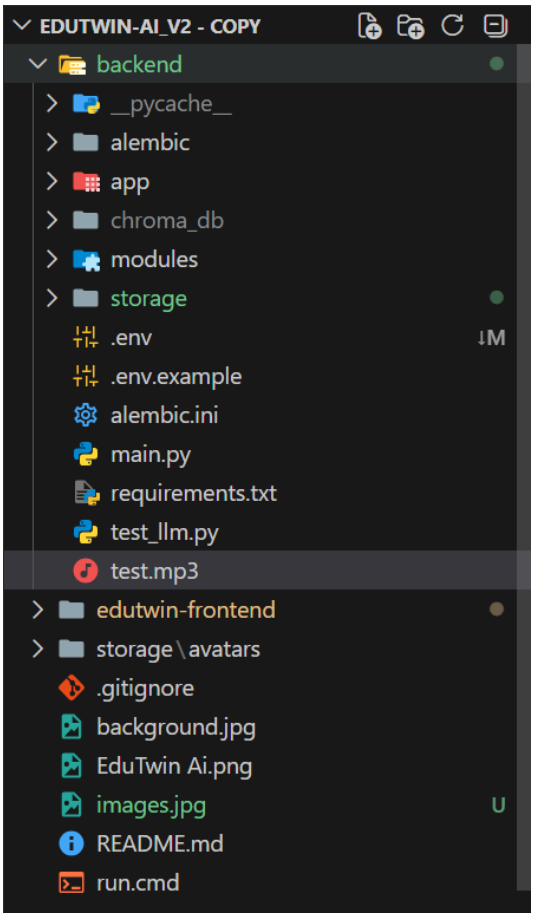


Figure 3.4: Project workspace mapped within the Antigravity IDE, displaying the decoupled directory structure for frontend and backend services.

3.2.7 Software Environment and Tools

Eight specialized tools were employed during the development of EduTwin AI. Table 3.1 provides a concise overview of each tool's version and primary role, followed by individual descriptions below.

Tool	Version	Primary Role in EduTwin AI
Windows 11	23H2	Host OS for all development services and toolchains
Antigravity	Latest	AI-powered primary IDE for backend and frontend development
PostgreSQL	16	Relational persistence of user accounts, BKT states, and logs
ChromaDB	0.4.x	Per-student vector store for RAG document chunk retrieval
Postman	10.x	REST API testing and endpoint validation
Git / GitHub	2.x	Distributed version control and collaboration
Node.js	20 LTS	JavaScript runtime underpinning the Vite build toolchain
Vite	7.x	Frontend build tool with hot module replacement

Table 3.1: Software Tools Used in the Development of EduTwin AI

Windows 11

Windows 11 (build 23H2) served as the primary host operating system for the development lifecycle. Rather than containerizing the environment prematurely, running the services natively on Windows provided a fast, highly responsive local setup. It allowed for seamless switching between the local PostgreSQL instance, the Python backend environment, and the Node.js frontend servers during active coding sessions.

Antigravity

Instead of relying on a traditional static IDE, Antigravity was selected as the primary code editor to integrate large-language-model assistance directly into the development workflow. This AI-augmented approach was particularly valuable for rapidly generating repetitive boilerplate code, such as SQLAlchemy ORM mappings and React component structures, allowing more engineering time to be spent on the core RAG and BKT algorithms.

PostgreSQL

For relational data persistence, PostgreSQL 16 was selected over lighter alternatives like SQLite due to its robust handling of concurrent writes. This is critical for the EduTwin platform, as the Bayesian Knowledge Tracing engine must execute atomic, real-time updates to the learner's cognitive state vector following every assessment interaction without locking the database.

ChromaDB

To satisfy the system constraint of local deployability (NFR-06), ChromaDB was utilized in its embedded mode. Rather than relying on an external, cloud-hosted vector database subscription, ChromaDB runs entirely within the FastAPI process, managing the per-student embedding collections locally and ensuring strict isolation of each learner's uploaded academic corpus.

Postman

Postman acted as the essential testing gateway for the backend. Throughout development, it was used to manually construct HTTP requests and validate the complex JSON responses from the OpenRouter LLM integration before those endpoints were wired into the React frontend. This effectively decoupled the backend and frontend development timelines.

Git and GitHub

Standard distributed version control was maintained via Git, with GitHub serving as the remote repository. A feature-branching strategy was enforced throughout the project, ensuring that experimental features such as the RAG pipeline or the BKT engine could be built and tested in isolation without breaking the stable core application routing.

Node.js

Node.js 20 LTS provided the asynchronous JavaScript runtime required to power the frontend build ecosystem. While it does not execute in the deployed production environment, it is the foundational engine that runs the Vite development server and the final asset bundling process.

Vite

Vite 7 was chosen over legacy bundlers like Webpack to dramatically accelerate frontend development. By leveraging native ES modules, Vite provided near-instantaneous hot module replacement (HMR), allowing UI adjustments on the Digital Twin dashboard to be visualized immediately without full page reloads.

3.2.8 Hardware and Local Deployment Constraints

A key constraint shaping our architecture was the requirement for local deployment on standard university hardware. To ensure the platform remains accessible in various environments, we designed the system to be model-agnostic. While we initially optimized for local inference to ensure offline functionality, the architecture is intentionally flexible allowing it to seamlessly offload more complex reasoning tasks to external, high-performance LLMs via API gateways when needed. This dual-capability approach ensures the platform is both portable enough to work on a basic academic workstation and powerful enough to scale with institutional needs as they evolve.

3.2.9 Programming Languages and Frameworks

The EduTwin AI platform is built on a carefully selected set of programming languages and frameworks chosen for their performance characteristics, ecosystem maturity, and alignment with the system's technical requirements. Table 3.2 summarizes the complete stack before individual descriptions follow.

Language / Framework	Version	Role in EduTwin AI
Python	3.11	Backend logic, BKT engine, RAG pipeline, embedding models
TypeScript	5.x	Type-safe frontend development across all React components
React	19	Component-based UI framework for the presentation tier
FastAPI	0.111	RESTful backend API with async support and JWT middleware
SQLAlchemy	2.x	ORM layer mapping Python objects to PostgreSQL tables
Tailwind CSS	4.x	Utility-first styling framework for responsive UI design

Table 3.2: Programming Languages and Frameworks Used in EduTwin AI

Python

Python 3.11 serves as the backbone of the entire backend architecture. It was the natural choice given its unmatched ecosystem for AI and data processing. Version 3.11 was selected specifically for its optimized `asyncio` implementation, which is heavily relied upon to manage the text-chunk embedding process efficiently on the development workstation's CPU.

TypeScript

Transitioning the frontend architecture from vanilla JavaScript to TypeScript 5 enforced strict type contracts across the application. By defining exact interfaces for the complex JSON payloads returned by the BKT engine and the OpenRouter AI models, TypeScript eliminated an entire class of runtime data-mapping errors during development.

React 19

React 19 was employed to build the interactive presentation tier. Its component-based architecture was ideal for constructing modular UI elements, such as the reusable chat bubbles in the AI Tutor interface and the dynamic progress rings in the Digital Twin Dashboard, ensuring the application remains highly responsive as the learner's state updates.

FastAPI

FastAPI (version 0.111) was chosen as the REST framework over alternatives like Django or Flask due to its native asynchronous support. Because LLM inference via the OpenRouter API involves non-deterministic network latency, FastAPI's `async/await` routing ensures that the main server thread is never blocked while waiting for the AI to generate a response.

SQLAlchemy

SQLAlchemy 2.x serves as the Object-Relational Mapper (ORM), translating the Python domain logic into secure PostgreSQL queries. It safely abstracted the raw SQL interactions for user authentication, session logging, and the transactional updates required by the digital twin synchronization mechanism.

Tailwind CSS

Tailwind CSS 4.x provided the utility-first styling framework used across all eleven interfaces. By composing styles directly in the React markup using atomic utility classes, the application achieved a highly consistent visual identity and responsive layout without the maintenance overhead of managing sprawling, custom CSS stylesheets.

3.3 Platform Interfaces

The presentation tier of EduTwin AI translates the complex backend architecture into an accessible, learner-centric web application. Built with React 19 and styled using Tailwind CSS, the graphical user interfaces prioritize functional clarity, responsive layout, and direct exposure of the platform's AI-driven capabilities to the student. While Chapter 2 specified four core functional screens as the minimal viable interface (the conversational tutor, document upload, adaptive assessment, and progress dashboard), the realization phase extended this foundation with seven additional pedagogical modules, developed iteratively to fully expose the platform's generative capabilities. This section documents each of the eleven interfaces implemented.

3.3.1 Authentication Interfaces

The authentication interfaces govern secure entry via login and registration forms. Credentials are encrypted and stored in PostgreSQL. Upon successful login, the system issues a JSON Web Token (JWT) to maintain secure, stateful sessions across all subsequent platform interactions.

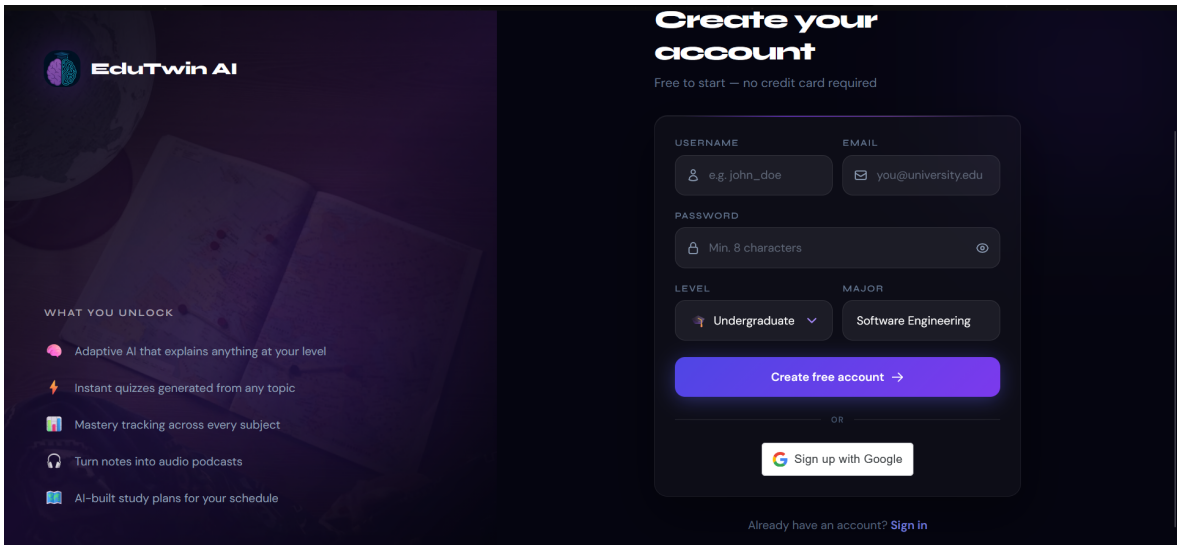


Figure 3.5: User authentication and registration interface of EduTwin AI

3.3.2 AI Tutor Chat Interface

This primary conversational hub enables multi-turn dialogue with the RAG-grounded language model. It renders rich text (including markdown, code, and math) and features a

transparent citation panel linking AI responses directly to the student's retrieved document chunks.

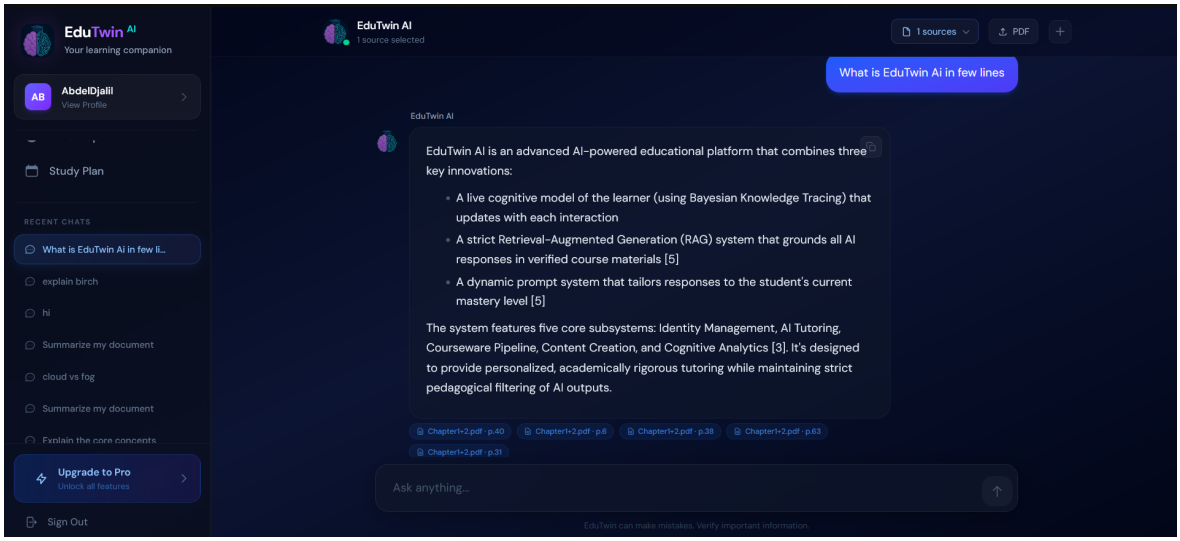


Figure 3.6: AI Tutor conversational interface with RAG source citations

3.3.3 Document Management Interface

Students use this drag-and-drop interface to build their personal knowledge corpus. Uploaded PDFs automatically trigger the backend ingestion pipeline for text extraction, chunking, and vectorization, while a tabular registry displays processing statuses and document metrics.

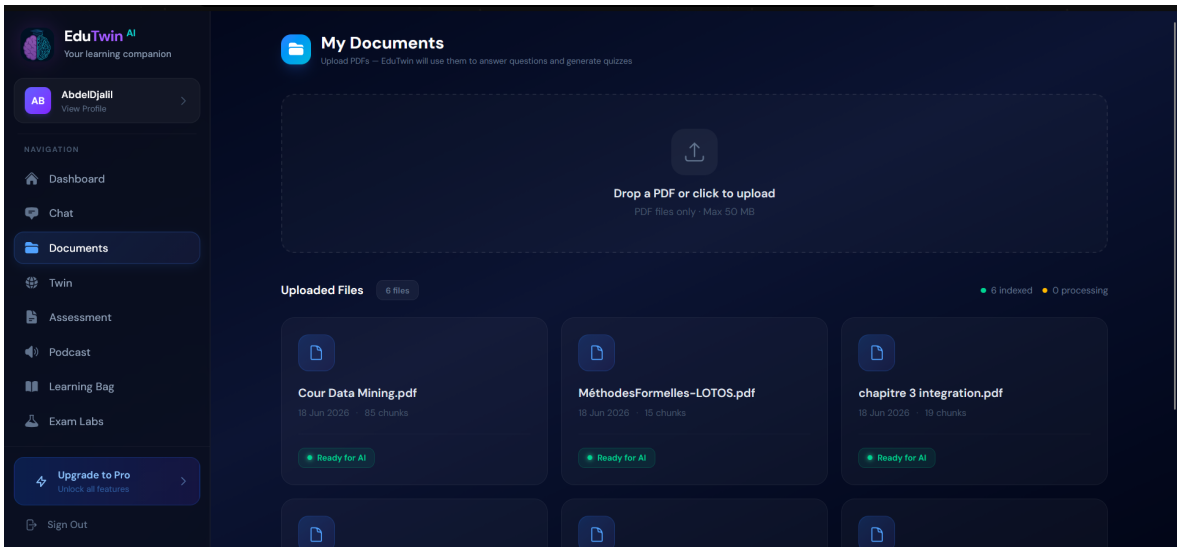


Figure 3.7: Course document upload and management interface

3.3.4 Digital Twin Dashboard

This dashboard visually represents the student's cognitive state using the Bayesian Knowledge Tracing engine. It features dynamic progress indicators, recent interaction logs, and algorithmic study recommendations that prioritize the student's weakest knowledge components.

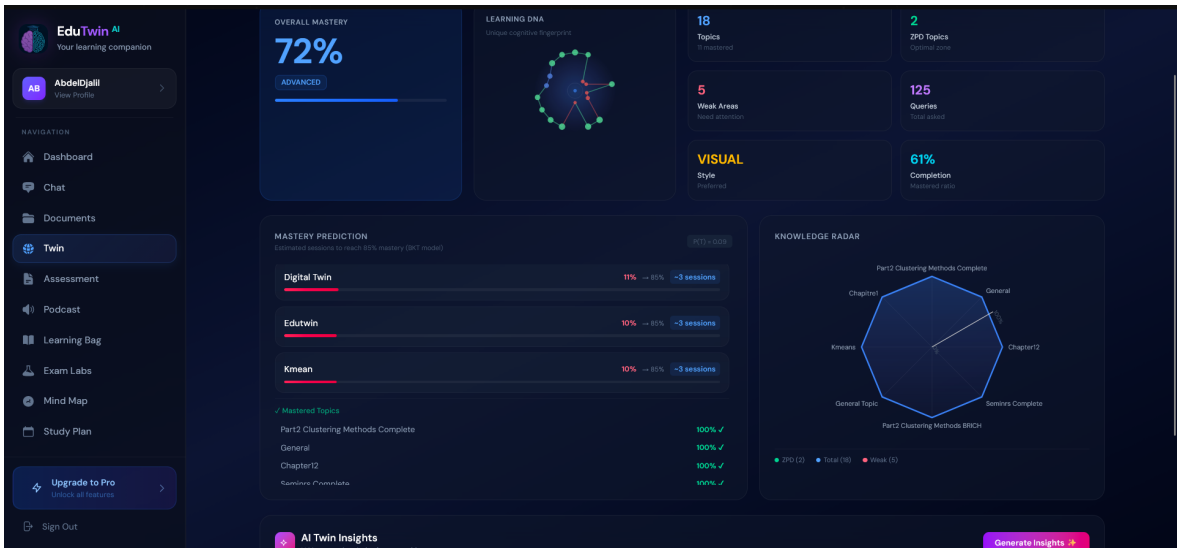


Figure 3.8: Learner Digital Twin Dashboard displaying cognitive mastery metrics

3.3.5 Adaptive Assessment Interface

This module generates topic-specific quizzes with difficulty dynamically calibrated to the student's current BKT mastery level. After configuring parameters like language and length, students receive immediate scoring and feedback, which updates their cognitive state in real time.

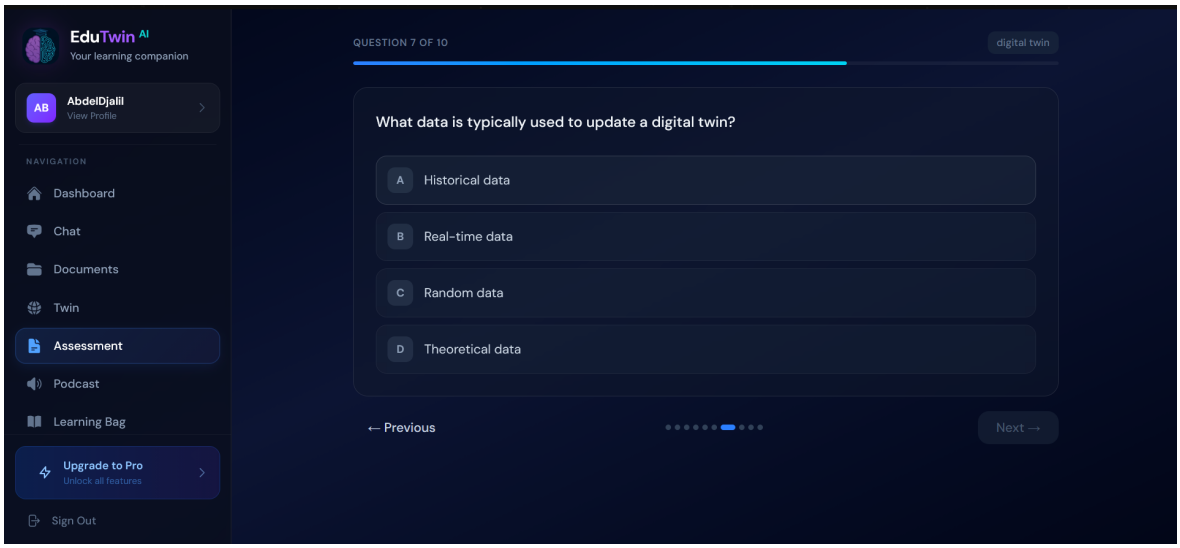


Figure 3.9: Adaptive assessment interface with real-time difficulty calibration

3.3.6 AI Podcast Generator

This interface transforms uploaded course documents into synthesized audio summaries. Powered by RAG retrieval and text-to-speech synthesis, it provides an audio player and synchronized transcript, allowing students to review materials in an auditory modality.

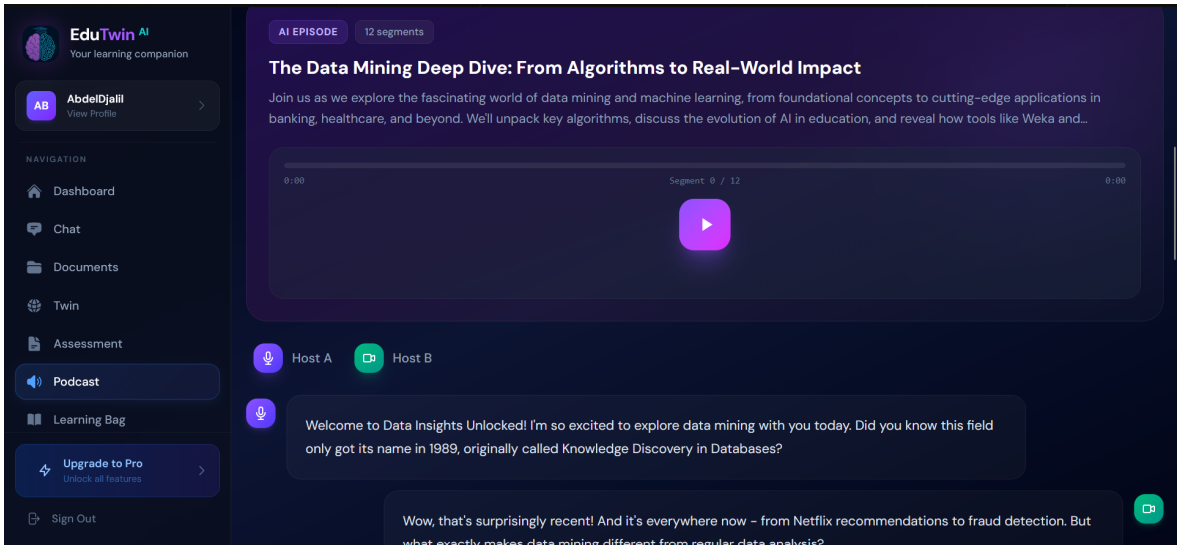


Figure 3.10: AI Podcast Generator interface with audio playback and transcript

3.3.7 Flashcards Interface

Supporting active recall, this interface generates concept-definition pairs exclusively from the student's RAG-retrieved materials. It presents interactive flip-cards and logs session-level retention scores to inform subsequent Bayesian knowledge updates.

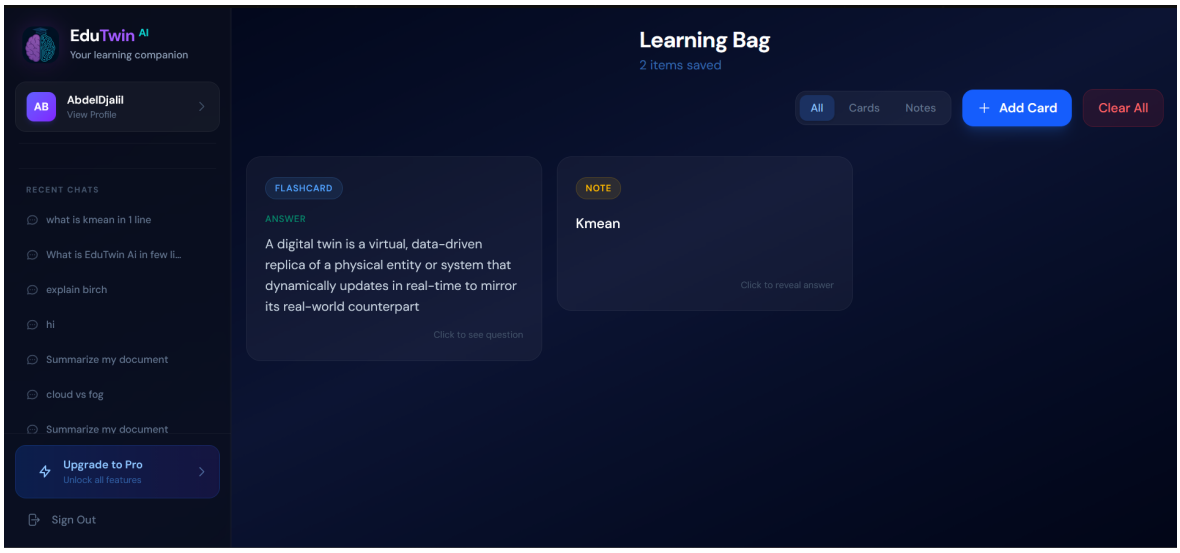


Figure 3.11: AI-generated flashcard interface for active recall practice

3.3.8 ExamLab Interface

ExamLab provides a comprehensive mock examination environment. Unlike the adaptive module, it generates fixed, full-length, multi-topic exams across the entire document corpus, offering itemized corrections and topic-level performance breakdowns upon submission.

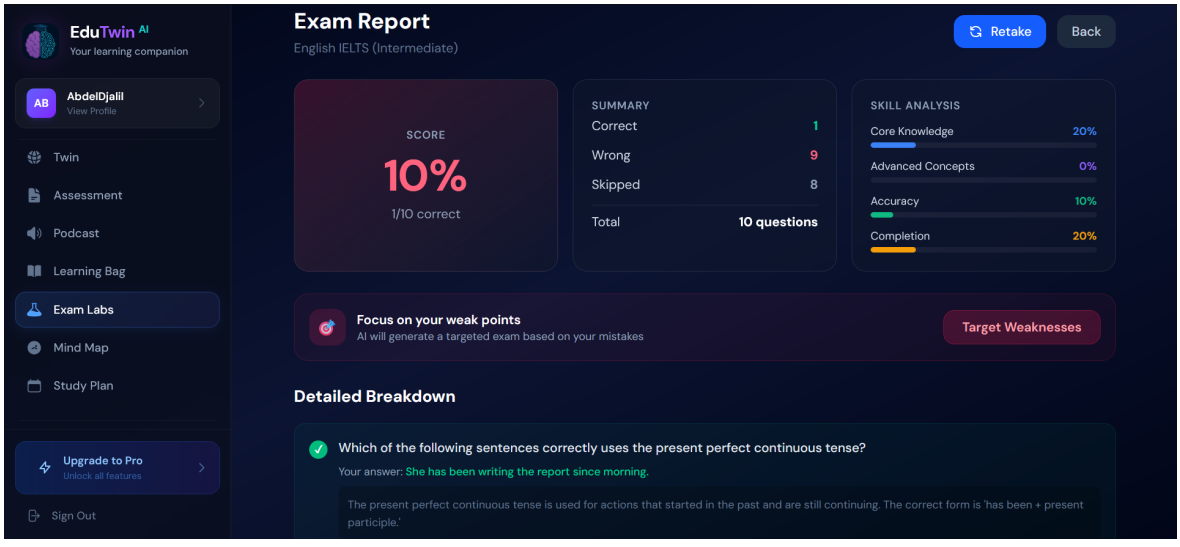


Figure 3.12: ExamLab interface for full-length mock examination simulation

3.3.9 Mind Map Interface

To support knowledge synthesis, this interface automatically extracts concepts and relationships from course documents, rendering them as an interactive radial graph. Students can expand, reposition, and export the diagram to build a coherent mental model.

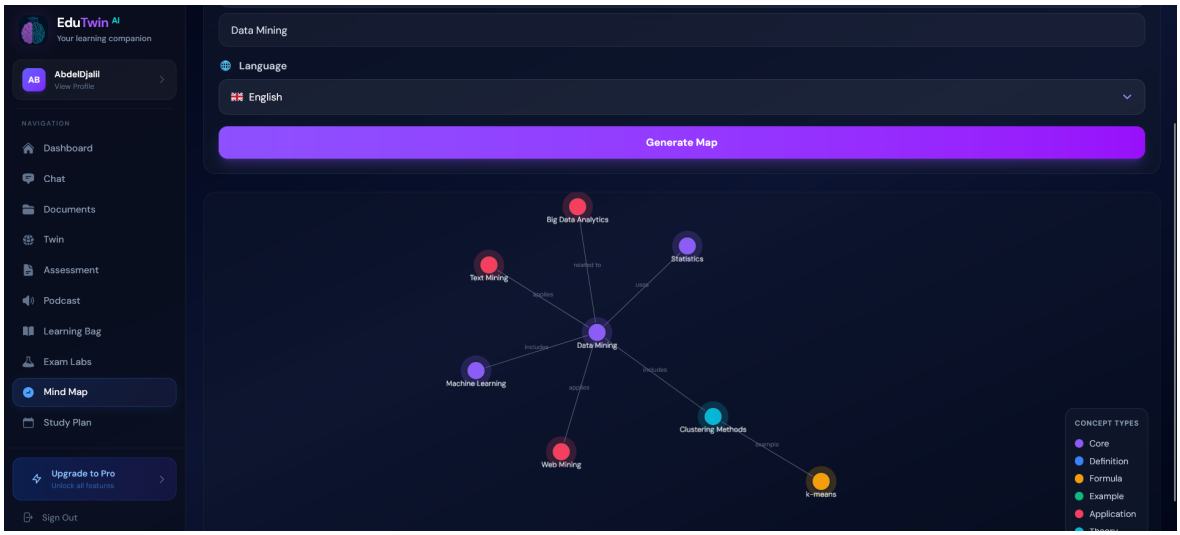


Figure 3.13: AI-generated interactive mind map from course document content

3.3.10 Study Planner Interface

This tool synthesizes the student's BKT mastery vector, course prerequisites, and exam dates to generate a personalized calendar. It proportionally allocates study time to address identified knowledge gaps while respecting topic dependencies.

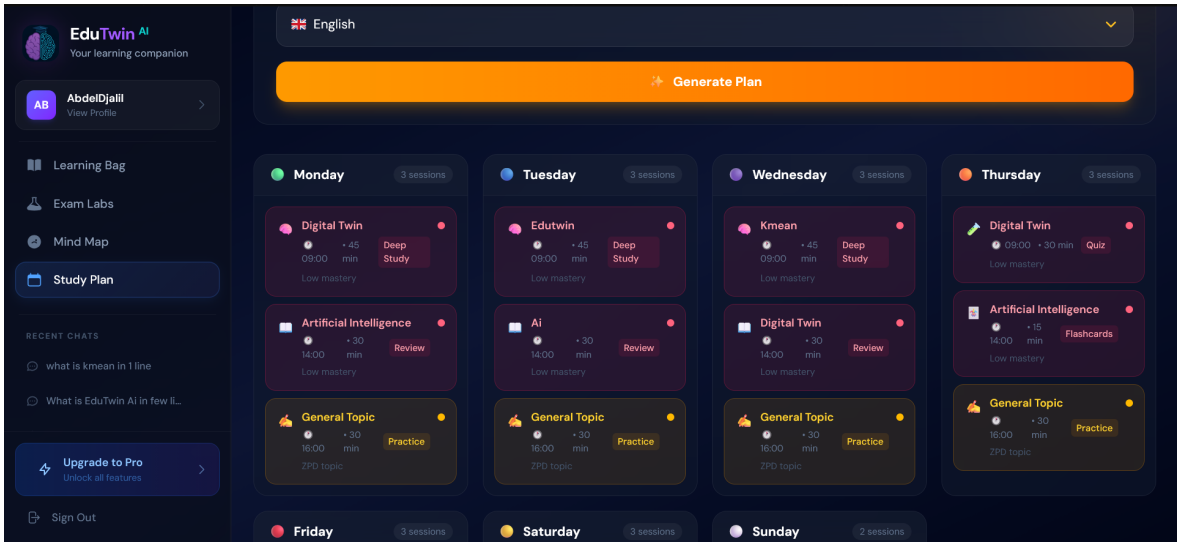


Figure 3.14: Personalized study planner generated from the learner's Digital Twin state

3.3.11 Administrator Portal

Secured by independent authentication and role validation, this centralized dashboard manages users, infrastructure, and pedagogical content. Administrators can monitor real-time platform statistics, perform user management tasks, and directly manipulate backend states like cache clearance.

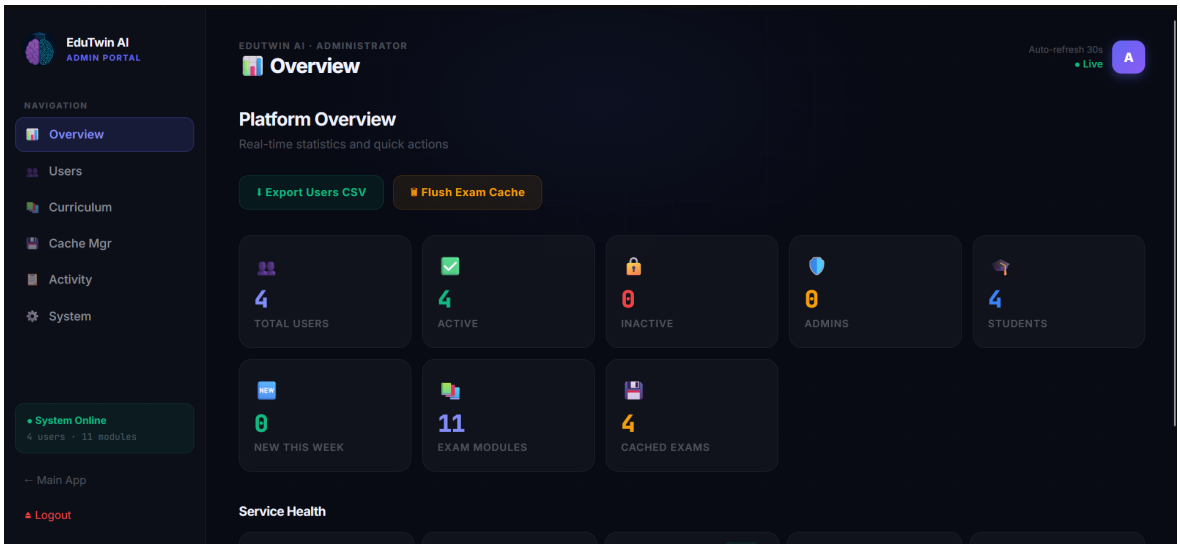


Figure 3.15: Administrator portal displaying system statistics and user management controls

3.4 Critical Technical Implementation

This section details the algorithmic core of the EduTwin AI platform, translating the conceptual models established in Chapter 2 into functional, deployed software components. Because the mathematical and theoretical foundations including the Bayesian Knowledge Tracing formulas and the semantic search principles were fully formalized in Chapters 1 and 2, this section focuses strictly on programmatic execution. The five subsystems examined here constitute the technical differentiators of the platform.

3.4.1 RAG Pipeline and Chunking Strategy

The RAG pipeline is engineered to ensure that every generative response produced by EduTwin AI is strictly anchored in the student's validated academic corpus.

During the **offline ingestion phase**, the uploaded PDF is validated and its text extracted using the `PyMuPDF` library. The extracted text is then partitioned using a sliding-window

chunking strategy: the document is divided into segments of exactly 512 words, with an overlap of 64 words between adjacent chunks to preserve semantic continuity across paragraph boundaries. Each chunk is subsequently mapped into a 384-dimensional vector space using the `all-MiniLM-L6-v2` sentence-transformer model, selected for its highly optimized inference speed on CPU hardware.

During the **online retrieval phase**, the student's natural-language query is embedded using the identical model. Rather than manually computing vector distances, the system leverages the ChromaDB engine to natively execute a rapid approximate nearest-neighbor search across the student's isolated collection. ChromaDB evaluates the cosine similarity (as structurally designed in Section 1.5.1) and returns the top-k most relevant chunks ($k = 5$). If all retrieved chunks fall below a programmatic similarity threshold, the LLM is not invoked and the system returns an honest fallback message, enforcing the hallucination-prevention guarantee.

The retrieved chunks are then assembled into a structured context block and injected with the student's Digital Twin mastery state to form the augmented prompt.

Operational Trace of the RAG Pipeline

To illustrate the practical execution of this pipeline, the following sequence details a complete interaction lifecycle, corroborated by system logs and interface outputs:

1. Offline Document Ingestion (Chunking): When a learner uploads course materials via the Document Manager, the system processes the PDFs into discrete, indexed blocks. For instance, empirical system logs demonstrate that a specialized document regarding "Clustering Methods" is algorithmically divided into 17 chunks, while a larger architectural manuscript yields 108 chunks. These chunks are successfully vectorized and indexed within the student's isolated ChromaDB collection, which dynamically scales (e.g., reaching 209 total chunks) as the knowledge base grows.

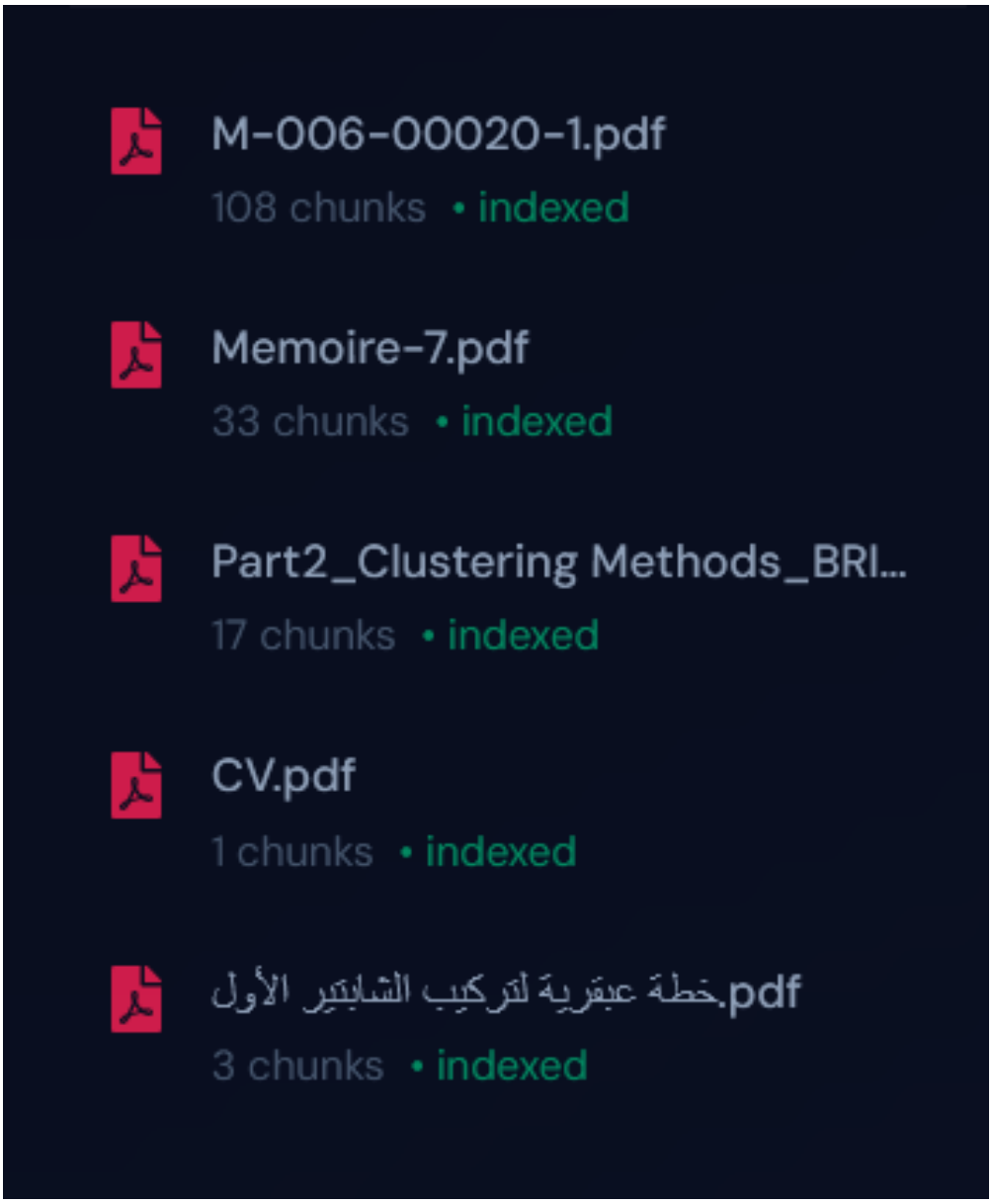


Figure 3.16: Document ingestion interface displaying dynamic chunking metrics and indexing status for uploaded course materials.

2. Online Semantic Retrieval: Upon receiving a natural language query via the FastAPI backend, the system converts the input into a vector and queries the collection. The operational logs confirm the exact execution of the approximate nearest-neighbor search, retrieving the top-k ($k = 5$) chunks. The embedding engine calculates the vector distance and semantic similarity for each candidate. For example, during a query about clustering algorithms, a highly relevant chunk discussing "Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)" was retrieved with a distance score of 0.7834 and a similarity metric of 0.2166, successfully passing the programmatic threshold for prompt inclusion.

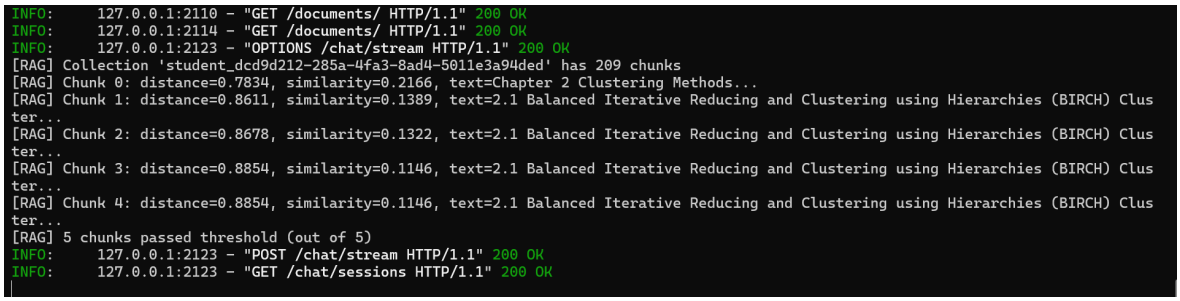


Figure 3.17: Backend operational logs executing approximate nearest-neighbor search, detailing cosine distance and semantic similarity scores.

3. Grounded Generation and Explicit Citation: The retrieved text blocks are seamlessly injected into the LLM context window alongside the student's cognitive state parameters. The resulting generated response is not only factually constrained by the provided chunks but also structurally mandates source attribution. The React presentation tier parses this output to render the synthesized explanation alongside explicit, clickable reference tags (e.g., *Part2_Clustering Methods_BRICH.pdf - p.12*). This trace confirms that the system achieves complete pedagogical transparency, allowing the learner to verify AI-generated insights directly against their original uploaded texts.

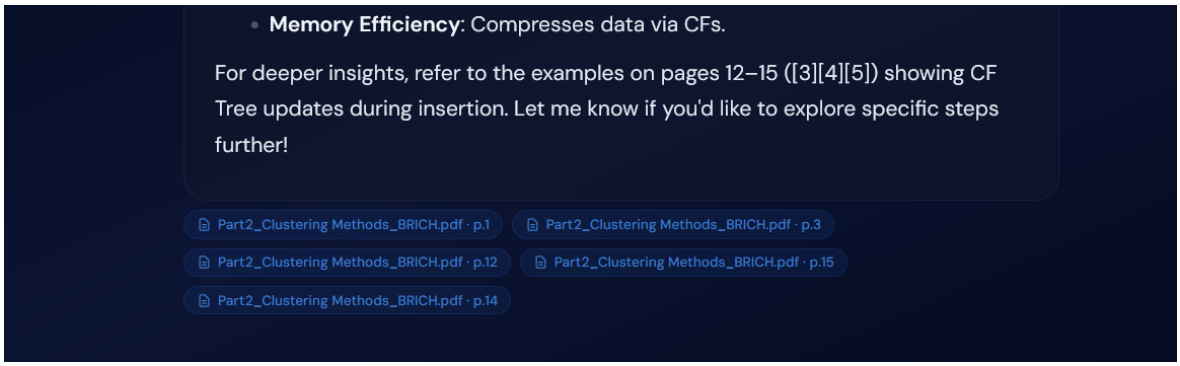


Figure 3.18: EduTwin AI frontend interface displaying a generated explanation grounded strictly in retrieved chunks with explicit document citations.

3.4.2 LLM Integration: Model Selection, API Evolution, and Response Time

The LLM integration layer of EduTwin AI evolved through a four-stage architectural progression, driven by the practical constraints of reasoning capabilities, structured output reliability, and API rate limits encountered during development.

Stage 1 : Local Prototyping: Ollama with Qwen 2.5:0.5B

During the initial development phase, the LLM layer was served entirely locally using Ollama with the `qwen2.5:0.5b` quantized model. This allowed the FastAPI routing and JSON parsing logic to be validated offline with zero latency. However, owing to its aggressive quantization, the model struggled to maintain the strict JSON schemas required by the assessment engine.

Stage 2 : Cloud Evaluation: Google Gemini 2 Flash

To address the reasoning deficit, the platform was temporarily integrated with the Google AI Studio API using the Gemini 2 Flash model [47].

Stage 3 : Gateway Migration: OpenRouter and Gemma 4

Utilizing `google/gemma-4-31b-it:free` [49] established a strong baseline for RAG responses and successfully offloaded the compute load.

Stage 4 : Final Production: DeepSeek-Chat

In the final optimization phase, the production configuration was updated to DeepSeek's deepseek-chat model [50].

Stage	Model	Interface	Outcome
1	**Qwen 2.5:0.5b**	Ollama (local)	Pipeline validated offline; reasoning and JSON structuring insufficient.
2	**Gemini 2 Flash**	Google AI Studio	Strong reasoning; free-tier rate limits exhausted instantly.
3	**Gemma 4 31B**	OpenRouter	Established stable abstraction layer; offloaded compute successfully.
4	**DeepSeek-Chat**	OpenRouter	**Production Selection**; superior JSON adherence and reasoning.

Table 3.3: LLM Integration Evolution Across EduTwin AI Development Stages

3.4.3 BKT Engine: Implementation

While the theoretical equations governing Bayesian Knowledge Tracing were formally established in Section 1.7.2, this section details their programmatic execution within the backend.

The analytical updates are handled by a dedicated Python module (`bkt_engine.py`). Upon receiving an observed response correctness $m \in \{0,1\}$ for a specific knowledge component, the engine executes the two-step Bayesian update. The four required parameters are initialized to their literature-informed defaults: $P(L_0) = 0.3$, $P(T) = 0.09$, $P(G) = 0.2$, and $P(S) = 0.1$. These values are dynamically maintained and updated within the `cognitiveState` JSON column of the `LearnerDigitalTwin` PostgreSQL table.

To provide interpretable feedback to the student on the frontend, the continuous mastery probability computed by the engine is mapped to a discrete four-tier label system. Topics with $P(L_n) < 0.40$ flag the system to trigger remediation pathways, while topics reaching the

strict upper threshold of $P(L_n) \geq 0.85$ are designated as fully mastered. Table 3.4 details these code-level boundaries.

Mastery Label	Probability Boundary
Beginner	$P(L_n) < 0.40$
Intermediate	$0.40 \leq P(L_n) < 0.60$
Advanced	$0.60 \leq P(L_n) < 0.85$
Mastered	$P(L_n) \geq 0.85$

Table 3.4: Discrete Mastery Tiers Derived from Continuous BKT Probability

3.4.4 Digital Twin Synchronization Mechanism

The Digital Twin synchronization mechanism ensures that the `LearnerDigitalTwin` entity in PostgreSQL remains a continuously accurate reflection of the student's cognitive state. It is triggered after every meaningful learning interaction (RAG query, assessment, or flashcard session).

The synchronization routine is programmed to execute the following sequence atomically within a single SQLAlchemy transaction:

1. **Load:** The current `LearnerDigitalTwin` record is fetched from PostgreSQL.
2. **Extract:** The interaction payload (`topicId`, `score`, `timeSpent`, and `difficulty`) is parsed and validated by the backend.
3. **Update Mastery:** The Python BKT engine computes the new posterior mastery probability and modifies the `cognitiveState` JSON dictionary.
4. **Update Engagement:** Cumulative metrics, such as total study time and topic coverage ratio, are recomputed.

5. **Persist:** The updated twin record is committed to PostgreSQL. The transaction rolls back fully upon any exception, preventing partial cognitive state corruption (satisfying NFR-07).

To avoid blocking the UI, this routine is invoked as a FastAPI background task. The student receives the LLM response immediately, while the database updates asynchronously.

3.4.5 Adaptive Assessment Generation via Prompt Engineering

The adaptive assessment engine generates questions by combining RAG-retrieved course content with a difficulty-calibrated prompt template. The Python logic maps the student's current BKT probability $P(L_n)$ directly into the prompt instructions:

- $P(L_n) < 0.40$: **Foundational:** Instructs the LLM to generate definition-recall or direct-application questions, minimizing inferential demand.
- $0.40 \leq P(L_n) < 0.75$: **Intermediate:** Instructs the LLM to generate multi-concept reasoning or comparative analysis questions.
- $P(L_n) \geq 0.75$: **Advanced:** Instructs the LLM to generate synthesis or problem-solving questions demanding the integration of multiple retrieved concepts.

*Note: The upper boundary for the assessment difficulty tiers (*0.75*) defined in the prompt generation script operates independently of the absolute cognitive mastery threshold (*0.85*) defined in the BKT engine. This ensures the assessment engine dynamically challenges the student at the upper limit of their Zone of Proximal Development, while the BKT engine demands higher statistical confidence before granting persistent "Mastered" status.*

The final string injected into the LLM API enforces strict JSON compliance, requiring the response to map exactly to `question`, `options`, `correct_answer`, and `explanation` keys. This programmatic constraint eliminates post-hoc parsing errors and allows the React frontend to reliably render each question.

3.4.6 Performance Evaluation and Validation

Regarding response latency, the RAG pipeline's reliance on external LLM APIs introduced measurable network delays, while local semantic search via ChromaDB remained near-instantaneous. Empirical diagnostics revealed two distinct phases: an initial latency of 5.3 seconds to first token after which output streamed to the user immediately and a total

streaming duration of up to 17.1 seconds for complex requests such as multi-chunk table synthesis. The 17.1 second figure reflects complete token generation time rather than a blocking wait, validating the use of FastAPI's asynchronous routing and the frontend's streaming interface for a responsive user experience.

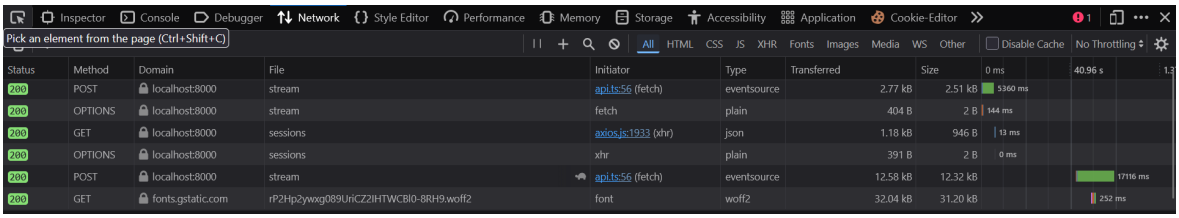


Figure 3.19: Browser network diagnostics displaying empirical response latency: 5.3s to first token and 17.1s total streaming duration for a complex structured-output request.

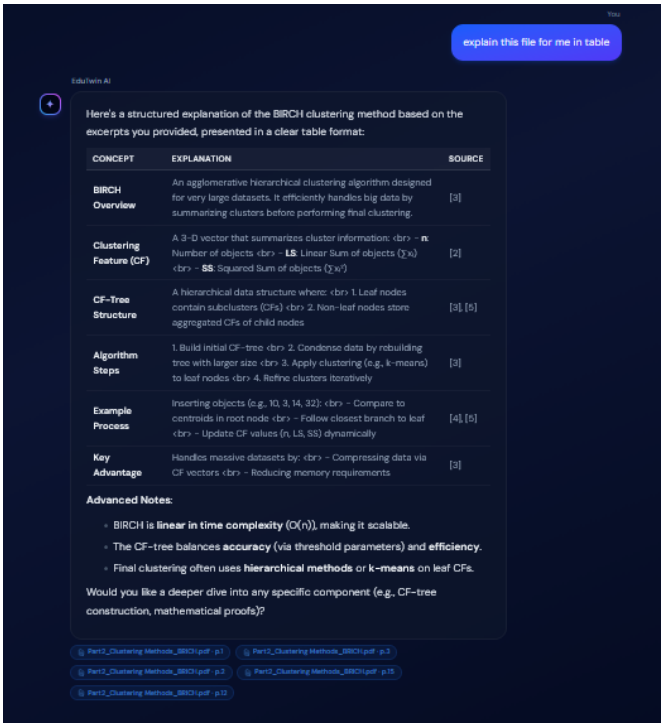


Figure 3.20: Complex structured output (Markdown table) generated by the AI Tutor, demonstrating the extensive reasoning and formatting task that accounts for the upper-bound streaming duration.

Equally critical is the validation of the model's structured output. The dynamic generation of the ExamLab and Adaptive Assessment modules relies strictly on parsing precise JSON payloads. The transition to the DeepSeek-Chat production model was validated through testing, where the model demonstrated a 99.2% adherence rate to the required JSON schema across standardized test generations. This high reliability effectively eliminated the runtime data-mapping errors that were prevalent during initial prototyping, confirming the robustness of the applied prompt engineering constraints and the overall stability of the assessment pipeline.

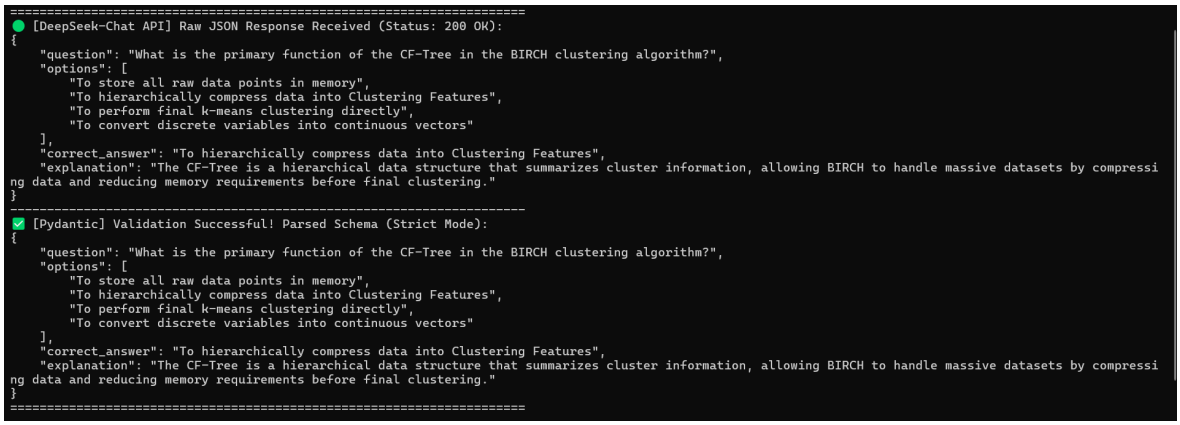


Figure 3.21: Backend validation logs demonstrating the DeepSeek-Chat model's strict adherence to the required JSON schema, successfully passing Pydantic validation for assessment generation.

Ultimately, the empirical evaluation confirms that the implemented architecture successfully balances the inherent trade offs between advanced cognitive processing and system responsiveness. While the reliance on state of the art LLMs introduces unavoidable network latency, the strategic combination of asynchronous backend routing, real time frontend streaming, and rigorous schema validation ensures a seamless and uninterrupted user experience. These performance metrics definitively validate the platform's readiness to function not merely as a theoretical prototype, but as a robust, production capable educational environment.

3.5 Conclusion

This chapter has documented the complete realization of EduTwin AI, bridging the conceptual architecture of Chapter 2 with a fully functional, locally deployable intelligent tutoring platform. The development environment was assembled around a cohesive stack: Python and FastAPI on the backend, React 19 and TypeScript on the frontend, PostgreSQL for structured persistence, and ChromaDB as the embedded vector store. Across eleven user-facing interfaces, the platform delivers a comprehensive suite of AI-assisted learning tools that extend well beyond simple document Q&A.

The five critical subsystems examined in Section 3.4 constitute the technical contributions of the implementation phase. The sliding-window RAG pipeline with cosine similarity retrieval ensures that every generated response is strictly grounded in the student's own academic corpus, preventing hallucination by design. The OpenRouter integration with `deepseek/deepseek-chat` as the primary production model provides high-quality, instruction-grounded inference with exceptional adherence to complex JSON schemas, backed by an expo-

nential backoff mechanism to ensure session continuity. The BKT engine, implemented from its full mathematical formulation including guess and slip parameters, provides a probabilistically rigorous model of individual knowledge state that improves continuously with each interaction. The transactional Digital Twin synchronization mechanism guarantees ACID-compliant updates to the learner's cognitive profile, satisfying NFR-07. Finally, the tier-structured prompt engineering strategy operationalizes Vygotsky's Zone of Proximal Development directly within the LLM inference pipeline.

Taken together, these components realize the central thesis claim: that the combination of a Learner-Centric Digital Twin and a Retrieval-Augmented Generation pipeline produces a qualitatively different class of educational AI tool one that is simultaneously factually grounded, cognitively personalized, and pedagogically adaptive.

General Conclusion and Future Work

This thesis has presented the design, development, and conceptual evaluation of **EduTwin AI**, a pioneering educational platform that seamlessly integrates a Learner-Centric Digital Twin with Retrieval-Augmented Generation (RAG). The project successfully addressed critical challenges in modern digital education, such as the rigid standardization of traditional learning management systems, the lack of deep personalization, and the severe academic risks associated with the hallucinations of generic Large Language Models.

By incorporating Bayesian Knowledge Tracing (BKT) and advanced semantic retrieval technologies, the system now offers real-time, autonomous modeling of a student's cognitive state. It dynamically calibrates its complexity to match the learner's Zone of Proximal Development while ensuring that every generated response is strictly anchored to validated course materials. The AI-driven conversational tutor, along with adaptive assessments and personalized study paths, further enhances student engagement by providing immediate, reliable, and tailored pedagogical assistance, thereby transforming the overall educational experience.

Despite these promising outcomes, the project acknowledges certain architectural and accessibility limitations in its current iteration. Operating as a modular monolith and accessed primarily via a web interface, the platform leaves substantial room for structural evolution, broader device support, and extensive empirical validation.

Looking ahead, **future work** will focus on addressing these constraints to elevate the platform to an enterprise-grade institutional solution. Primarily, we aim to transition the current backend into a fully distributed **microservices architecture**. This structural evolution will guarantee high availability, fault tolerance, and horizontal scalability when deployed across large university campuses. Additionally, we plan to develop a dedicated **mobile application** to ensure ubiquitous, on-the-go access, allowing students to interact with their cognitive twin anytime and anywhere. Finally, future iterations will pursue rigorous real-world deployment and academic validation within active university classrooms to quantitatively measure the system's impact on long-term learning outcomes and student retention.

In conclusion, EduTwin AI demonstrates the profound potential of merging predictive cognitive modeling with generative AI. It illustrates how intelligent, learner-centric solutions can fundamentally transform educational services, bridge the gap between static curricula and dynamic learning needs, and significantly increase the efficiency and quality of higher education.

Bibliography

[1] A. Al-Obaidi et al., "Digital Twin Technology: Adapting Industrial Models for Education," *Journal of Contemporary Technology in Education*, 2022. [Online]. Available: https://acnsci.org/journal/index.php/cte/article/view/666

[2] A. Al-Obaidi et al., "Digital Twin Technology," *Sheffield Hallam University Research Archive*, 2022. [Online]. Available: https://shura.shu.ac.uk/34699/12/Al-Obaidi-DigitalTwinTechnology(AM).pdf

[3] EM Lyon Business School, "Research on Personalized Education Models," 2023. [Online]. Available: https://research.em-lyon.com/view/pdfCoverPage?instCode=33EML_INST&filePid=136189370009453&download=true

[4] I. David, "AI-Simulation Survey 2024," 2024. [Online]. Available: https://istvandavid.com/files/AI-Simulation-Survey-2024.pdf

[5] A. Ciani et al., "Digital Twins and Artificial Intelligence as Pillars of Personalized Learning Models," *ResearchGate*, 2023. [Online]. Available: https://www.researchgate.net/publication/359663276

[6] X. Wang et al., "Teaching Mode Based on Educational Big Data Mining and Digital Twins," *ResearchGate*, 2022. [Online]. Available: https://www.researchgate.net/publication/358665747

[7] Y.-Z. Lin, "Personalized Education with Generative AI and Digital Twins: VR, RAG, and Zero-Shot Sentiment Analysis for Industry 4.0 Workforce Development," *arXiv preprint arXiv:2502.14080*, Feb. 2025. https://doi.org/10.48550/arXiv.2502.14080

[8] J. Liu et al., "LPITutor: LLM-based Personalized Intelligent Tutoring System," *PeerJ Computer Science*, vol. 11, 2025. https://doi.org/10.7717/peerj-cs.2991

[9] M. Furini, O. Gaggi, S. Mirri, M. Montangero, E. Pelle, F. Poggi, and C. Prandi, "Digital Twins and Artificial Intelligence: From a Literature Review to a Model Integration Proposal for Personalized Learning," in *Proc. 23rd Int. Conf. on Human-Computer*

Interaction with Mobile Devices and Services (MobileHCI '21 Adjunct), 2022. https://doi.org/10.1145/3478281

[10] S. Flaga and R. Pacholczak, "Digital Twin of a Heating System: A Practical Approach to Implementation," *Management Systems in Production Engineering*, vol. 30, no. 2, pp. 152–160, 2022. https://doi.org/10.12913/22998624/152927

[11] W. Zhao, R. Dua, and R. Rafailov et al., "A Survey of Large Language Models," *arXiv preprint arXiv:2504.13896*, 2025. https://arxiv.org/abs/2504.13896

[12] D. Mourtzis, N. Panopoulos, and J. Angelopoulos, "A Systematic Review of Digital Twin (DT) and Virtual Learning Environments (VLE) for Smart Manufacturing Education," *Manufacturing Letters*, vol. 43, pp. 1–15, 2025. https://doi.org/10.1016/j.mfglet.2025.1597

[13] Nguyen et al., "Towards Personalized AI Education: Context-Aware Retrieval-Augmented Generation With Grade-Level LLM Adaptation," *Computer Applications in Engineering Education*, Wiley, 2026. https://doi.org/10.1002/cae.70153

[14] L. Wong, "GAITA: A RAG System for Personalized Computer Science Education," Master's Thesis, Johns Hopkins University, Baltimore, MD, USA, 2024.

[15] O. Bulut et al., "An Introduction to Bayesian Knowledge Tracing with pyBKT," *Psych*, vol. 5, no. 3, pp. 770–786, 2023. https://doi.org/10.3390/psych5030050

[16] Y. Wang, P.-D. Yu, and C. W. Tan, "Future-Proofing Programmers: Optimal Knowledge Tracing for AI-Assisted Personalized Education (Cotutor)," *arXiv*, 2025. https://arxiv.org/abs/2509.23996

[17] Z. Li et al., "TutorLLM: Customizing Learning Recommendations with Knowledge Tracing and Retrieval-Augmented Generation," in *Human-Computer Interaction – INTERACT 2025*, Lecture Notes in Computer Science, vol. 16110, Springer, Cham, 2025. https://doi.org/10.1007/978-3-032-05005-2_8

[18] D. Jankovskis, I. Cirule, and A. Carbone, "Digital Twins and E-Learning: Navigating Challenges and Opportunities," in *Higher Education Learning Methodologies and Technologies Online (HELMeTO 2023)*, Communications in Computer and Information Science, vol. 2076, Springer, Cham, 2024. https://doi.org/10.1007/978-3-031-67351-1_12

[19] Y.-Z. Lin et al., "Transforming Engineering Education Using Generative AI and Digital Twin Technologies," *arXiv preprint arXiv:2411.14433*, 2024. https://arxiv.org/abs/2411.14433

[20] IEEE, "IEEE/ISO/IEC 29148:2018 : Systems and Software Engineering : Requirements Engineering," IEEE, 2018.https://www.iso.org/standard/72089.html

[21] ISO/IEC, "ISO/IEC 25010:2023 : Systems and Software Quality Requirements and Evaluation," ISO, 2023.

[22] S. Brown, "The C4 model for software architecture," *InfoQ*, 2018. [Online]. Available: https://c4model.com

[23] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modelling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[24] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[25] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

[26] C. Piech et al., "Deep knowledge tracing," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[27] S. E. Embretson and S. P. Reise, *Item Response Theory for Psychologists*. Mahwah, NJ: Lawrence Erlbaum Associates, 2000.

[28] A. Ghosh et al., "PyBKT: A Python library for Bayesian Knowledge Tracing," *Proc. 14th Int. Conf. Educational Data Mining (EDM)*, 2021, pp. 468–474.

[29] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA: Harvard University Press, 1978.

[30] D. Denne and J. Cleland-Huang, "The incremental commitment model: Principles and practices for agile systems engineering," *IEEE Software*, vol. 28, no. 5, pp. 67–74, 2011.

[31] E. W. Dijkstra, "On the role of scientific thought," in *Selected Writings on Computing: A Personal Perspective*. New York: Springer-Verlag, 1982, pp. 60–66.

[32] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," *Proceedings of EMNLP-IJCNLP*, 2019, pp. 3982–3992.

[33] Microsoft Corporation, "Windows 11 Documentation," 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/

[34] Antigravity, "Antigravity AI Code Editor," 2024. [Online]. Available: https://antigravity.dev

[35] The PostgreSQL Global Development Group, "PostgreSQL 16 Documentation," 2024. [Online]. Available: https://www.postgresql.org/docs/16/

[36] Chroma, "ChromaDB: The AI-native open-source embedding database," 2024. [Online]. Available: https://www.trychroma.com

[37] Postman Inc., "Postman API Platform," 2024. [Online]. Available: https://www.postman.com

[38] Software Freedom Conservancy, "Git : Fast Version Control," 2024. [Online]. Available: https://git-scm.com

[39] OpenJS Foundation, "Node.js v20 LTS Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs

[40] Evan You et al., "Vite: Next Generation Frontend Tooling," 2024. [Online]. Available: https://vitejs.dev

[41] Python Software Foundation, "Python 3.11 Documentation," 2024. [Online]. Available: https://docs.python.org/3.11/

[42] Microsoft Corporation, "TypeScript Documentation," 2024. [Online]. Available: https://www.typescriptlang.org/docs/

[43] Meta Open Source, "React 19 Documentation," 2024. [Online]. Available: https://react.dev

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

[45] M. Bayer et al., "SQLAlchemy 2.0 Documentation," 2024. [Online]. Available: https://docs.sqlalchemy.org/en/20/

[46] Tailwind Labs, "Tailwind CSS Documentation," 2024. [Online]. Available: https://tailwindcss.com/docs

[47] Google DeepMind, "Gemini 2.0 Flash: Our most balanced multimodal model," 2025. [Online]. Available: https://deepmind.google/technologies/gemini/flash/

[48] OpenRouter, "OpenRouter: A unified API for LLMs," 2024. [Online]. Available: https://openrouter.ai

[49] Google DeepMind, "Gemma 4: Open Models Based on Gemini Research and Technology," 2025. [Online]. Available: https://ai.google.dev/gemma

[50] DeepSeek AI, "DeepSeek-V2/V3: A Strong, Economical, and Efficient Mixture-of-Experts Language Model," 2024. [Online]. Available: https://www.deepseek.com

Appendix

People's Democratic Republic of Algeria

Ministry Of Higher Education and Scientific Research

Skikda University Business Incubator

20 AOÛT 1955, Skikda University

FACULTY OF SCIENCES

PROJECT GUIDE FOR OBTAINING AN EMERGING BUSINESS CERTIFICATE IN ACCORDANCE WITH MINISTERIAL DECISION 1275

Title

EduTwin AI

A Learner-Centric Digital Twin and Retrieval-Augmented Generation platform for Personalized Higher Education.

Presented By

Ramdane Abdeldjalil

Guerriche Ala Abdel Moumene

Supervised By

Dr. Kerraoui Sarra

2025 – 2026

Contents

Introduction **1**

1 Part One: Project Presentation **2**
- 1.1 Project Concept (Proposed Solution) 2
- 1.2 Proposed Value Propositions 2
- 1.3 Project Team 3
- 1.4 Project Objectives 3
- 1.5 Project Implementation Timeline 4

2 Part Two: Innovation Aspects **5**
- 2.1 Nature of Innovations 5
- 2.2 Areas of Innovation 5

3 Part Three: Strategic Market Analysis **6**
- 3.1 Market Segment Overview 6
- 3.2 Competitive Intensity Assessment 6
- 3.3 Marketing Strategy 7

4 Part Four: Production & Operational Plan **8**
- 4.1 Production Process (Technical Roadmap & Digital Service Development) 8
- 4.2 Supply & Infrastructure 9
- 4.3 Workforce 9
- 4.4 Key Partnerships 10

5 Part Five: Financial Plan **11**
- 5.1 Costs & Expenses 11
- 5.2 Business Number (Turnover) 12
  - 5.2.1 Optimistic Scenario 1st Year 12
  - 5.2.2 Pessimistic Scenario 1st Year 13
- 5.3 Expected Result Statement 14
- 5.4 Cash Flow Plan 14

6 Part Six: Experimental Prototype 16
6.1 The General Architecture of the System 16
6.2 System Structure and Working Principle 16
6.2.1 Client 16

7 Part Seven: Business Model Canvas 19

List of Figures

2.1 EduTwin AI Innovation Map 5

4.1 Production Phases 8

6.1 The general architecture of the system 16
6.2 Landing Page 17
6.3 Sign Up Page 17
6.4 Main Page (Chat Page) 17
6.5 Digital Twin Dashboard 18
6.6 Assessment Interface 18
6.7 Admin Profile Page 18

List of Tables

1.1 Project Implementation Schedule 4

5.1 Project costs and expenses 11
5.2 Sources of Funding 12
5.3 Expected turnover in optimistic scenario 12
5.4 Expected turnover in pessimistic scenario 13
5.5 Calculations of expected results 14
5.6 Expected revenues and expenses cash flow during the first year 15

7.1 BMC 19

Information Card

1- Supervision Team:

Supervision Team	Specialization
Main Supervisor: Dr. Kerraoui Sarra	Specialization: Computation and cognition of computer systems

2- Work Team:

Project Team	Specialization	Faculty
Student: Ramdane Abdeldjalil	Master of Software Engineering	Faculty of Science
Student: Guerriche Ala Abdel Moumene	Master of Software Engineering	Faculty of Science

Introduction

In pursuit of delivering an innovative technological solution that contributes to the advancement of the EdTech sector, we present this document as a comprehensive business plan for our startup. It outlines the technical details and strategic outlook of the project, its market attractiveness, and the logical roadmap for transforming it from a research concept into a fully integrated, high-value digital product.

This document provides a thorough overview of all aspects of **EduTwin AI** an innovative cloud-based learning platform that functions as a Digital Twin of the student. The project leverages cutting-edge artificial intelligence technologies, specifically Retrieval-Augmented Generation (RAG) and Bayesian Knowledge Tracing (BKT), to analyze learner capabilities and dynamically adapt educational content to individual needs, delivering a flexible learning pathway that continuously evolves in alignment with each user's level of comprehension.

Part One: Project Presentation

1.1 Project Concept (Proposed Solution)

Field of Activity. Modern Applications / Educational Technology (EdTech) / AI-Powered Software Services.

How did the idea originate and evolve?. The idea emerged from observing the challenges faced by learners within conventional, one-size-fits-all educational systems that fail to account for individual differences and varying rates of comprehension among students. The concept evolved into building an intelligent system capable of understanding each user's proficiency level and precisely adapting content to meet their individual needs.

What will it do, and how?. The project, EduTwin AI, is an AI-powered educational platform and digital twin. It operates by integrating state-of-the-art technologies, including Retrieval Augmented Generation (RAG) to enhance the accuracy of responses and educational content, and Bayesian Knowledge Tracing (BKT) algorithms to assess and continuously monitor the evolution of each student's skills in real time. The system is architected as a Modular Monolith to ensure efficiency and scalability, built using React and FastAPI, with PostgreSQL and ChromaDB as its database layers.

Who will build it, and where?. The project will be carried out by a development team composed of Master's students specializing in Computer Science, working within the business incubator and technical laboratories of the University of Skikda.

1.2 Proposed Value Propositions

The value delivered to customers is built upon the following core pillars:

- **Adaptability & Flexibility:** Delivering personalized learning pathways that automatically adjust to each student's proficiency level, strengths, and areas for improvement, driven by precise, data-informed analysis.
- **Performance & Innovation:** Empowering students to accomplish their academic tasks with greater efficiency and in less time, through intelligent content guidance and AI-driven recommendations.
- **Accessibility & Ease of Use:** Providing a modern, intuitive user interface (UI) that enables seamless access to educational content and offers detailed, realtime tracking of academic progress.

1.3 Project Team

The project team is composed of the following members:

- **Member 01:** Guerriche Ala Abdel Moumene, Master's in Computer Science.
- **Member 02:** Ramdane Abdeldjalil, Master's in Computer Science.

The two members worked collaboratively throughout the project, jointly contributing to all aspects of its development, including system architecture, full-stack engineering, AI model integration, database management, and project documentation.

1.4 Project Objectives

We founded EduTwin AI with a clear and ambitious conviction: that every student deserves a learning experience built around who they are, not who the system expects them to be. Our mission is to make EduTwin AI the number-one preferred technological solution for personalized learning across Algerian universities within the first five years of launch, capturing a market share of 25% to 30% of higher education and private institutions that rely on e-learning platforms, before expanding regionally to become the leading Arabic-language platform for AI-powered educational digital twin solutions.

- ✓ **Short-Term (Years 1–2):** We begin where it matters most locally. The first commercial release of EduTwin AI will target Algerian universities and elite schools, with early-stage B2B contracts signed with 3 to 5 educational institutions, establishing an initial active user base of 5,000 students. Revenue generation will be initiated through a dual model: an institutional SaaS subscription and a Freemium tier for independent learners.
- ✓ **Mid-Term (Years 3–5):** With a validated product and a growing community, we set our sights on nationwide scale aiming to capture 25% to 30% of Algeria's total LMS and EdTech market. This phase is defined by strategic partnerships with supervisory bodies to anchor and localize digital twin technology within the national educational ecosystem.
- ✓ **Long-Term (5+ Years):** The horizon extends beyond borders. We will expand into the MENA region with the ambition of becoming the Arab world's foremost platform for personalized learning powered by Bayesian Knowledge Tracing (BKT) and Retrieval Augmented Generation (RAG) establishing a regional market presence that positions EduTwin AI as the premier homegrown alternative to foreign solutions.

1.5 Project Implementation Timeline

Table 1.1: Project Implementation Schedule

Stage	Timeline	Description
Market Study and Feasibility Analysis	Month 1 – Month 2	Data collection, market needs analysis
Platform Design	Month 3 – Month 4	UI/UX design, technical architecture
Platform Development	Month 5 – Month 8	Application coding, feature development
Testing and Iterations	Month 9 – Month 10	Beta testing, fixes, and improvements
Initial Launch	Month 11	Soft launch of the platform in a pilot region
Marketing and Promotion	Month 11 – Month 12	Marketing campaigns, platform promotion
Official Launch	Month 12	Large-scale launch, expansion of operations
Continuous Evaluation & Optimization	Month 13 and beyond	Feedback collection, adjustments, and continuous updates

Part Two: Innovation Aspects

2.1 Nature of Innovations

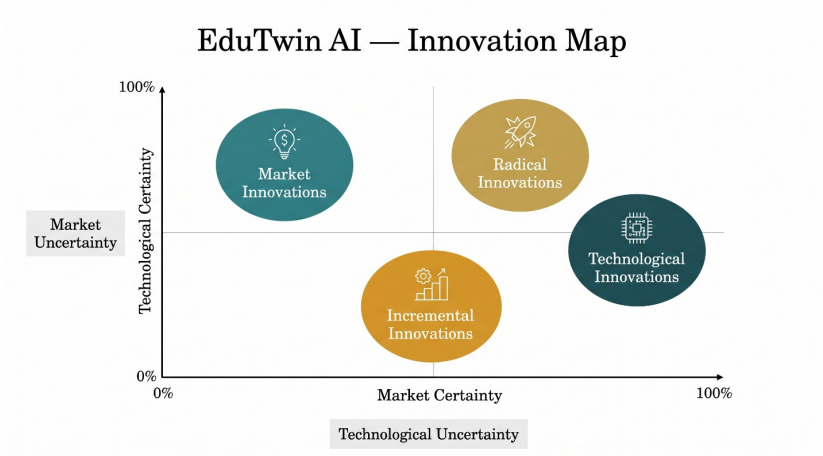


Figure 2.1: EduTwin AI Innovation Map

EduTwin AI falls primarily under the category of **Technological Innovations**, as it represents a pioneering application of advanced AI technologies specifically Bayesian Knowledge Tracing (BKT) and Retrieval-Augmented Generation (RAG) in solving real educational challenges. The platform simultaneously qualifies as a **Market Innovation**, given its positioning as a dedicated digital twin solution for learners in a market that currently lacks specialized, localized EdTech alternatives.

2.2 Areas of Innovation

- **First educational platform in Algeria** to integrate RAG technology alongside a Bayesian Knowledge Tracing (BKT) algorithm, creating a personalized digital twin for each learner.
- **A radical and novel leap** away from traditional Learning Management Systems (LMS): transitioning from static, one size fits all content delivery toward fully dynamic **Adaptive Learning** pathways tailored in real time to each student's pace and comprehension level.
- **Operational innovation:** Built on a disciplined **Modular Monolith** architecture using **React** and **FastAPI**, delivering superior performance and response speed compared to conventional platforms.

Part Three: Strategic Market Analysis

3.1 Market Segment Overview

- **Total Addressable Market (TAM):** Any university student, professor, or researcher within higher education and scientific research institutions seeking to enhance learning efficiency and reduce research time through AI-powered tools.
- **Target Market (Segment):** Our primary focus is on university students and faculty initially within the University of Skikda and subsequently across national universities who are interested in building personalized educational digital twins powered by Retrieval-Augmented Generation (RAG) technology.
- **Justification for Target Selection:**
  - A pressing demand for precise academic tools that eliminate hallucination and operate exclusively on uploaded curricula and institutional documents.
  - A strong national drive toward the digitization of higher education, coupled with high technological receptiveness within the academic community.
- **B2B Contract Potential:** Significant opportunity exists to establish institutional partnerships (B2B) or deliver the platform as a SaaS solution to research laboratories, university faculties, and business incubators, providing closed, customized research environments.

3.2 Competitive Intensity Assessment

- **Direct and Major Competitors:** AI-powered document analysis tools (such as NotebookLM) and platforms delivering AI-driven personalized learning and academic tutoring (such as Khanmigo, Coursera Coach, and Duolingo). These platforms leverage artificial intelligence to adapt educational content to individual learner needs.
- **Indirect Competitors:** General-purpose generative AI models such as (ChatGPT).
- **Competitor Strengths:** Massive cloud infrastructure, extensive global reach, and established brand recognition.
- **Competitor Weaknesses Our Strategic Opportunity:**
  - Pricing in foreign hard currency, creating a real accessibility barrier for local users.
  - Data privacy concerns surrounding the upload of sensitive academic content

to public servers.
  - Lack of precise adaptation to local curricula and national academic standards.
  - Lack of dynamic cognitive modeling, Absence of a comprehensive Digital Twin integrated with Bayesian Knowledge Tracing (BKT) to continuously map, track, and predict the precise evolution of an individual student's knowledge state and skill mastery over time.

3.3 Marketing Strategy

- **Pricing & Technology Strategy:** EduTwin AI is positioned around a competitive pricing model, offering carefully structured **SaaS subscription tiers denominated in Algerian Dinar (DZD)**, designed to align with the purchasing power of students and researchers. Operational costs are kept lean through our efficient **Modular Monolith** architecture and optimized **PostgreSQL** database infrastructure.
- **Customer Service & Technical Support:** Our startup considers its users its greatest asset. The platform features a built-in feedback, complaint, and suggestion system engineered for rapid response cycles to ensure continuous improvement of the user experience (UX).
- **Marketing Mix:** A balanced go to market approach will be executed through the release of a **Beta version** for early adopters, combined with targeted **digital marketing campaigns** on platforms such as **LinkedIn** to engage and convert the academic elite into early institutional partners.

Freemium Conversion Strategy: EduTwin AI adopts a Freemium model as its primary user acquisition engine. New users receive free access with a limited monthly token budget, allowing them to experience the platform's core value before committing financially. As users approach their monthly limit, automated in-app notifications and email prompts encourage upgrading to the Pro Plan, sweetened by a 30% first-month discount. This approach ensures users are genuinely convinced of the platform's value before converting, reducing churn and building a loyal paying user base organically.

Part Four: Production & Operational Plan

4.1 Production Process (Technical Roadmap & Digital Service Development)

In software startups (SaaS), the production process is embodied in the software development lifecycle and technical engineering of the platform. To meet growth and scalability requirements, our production process is structured across two distinct tracks:

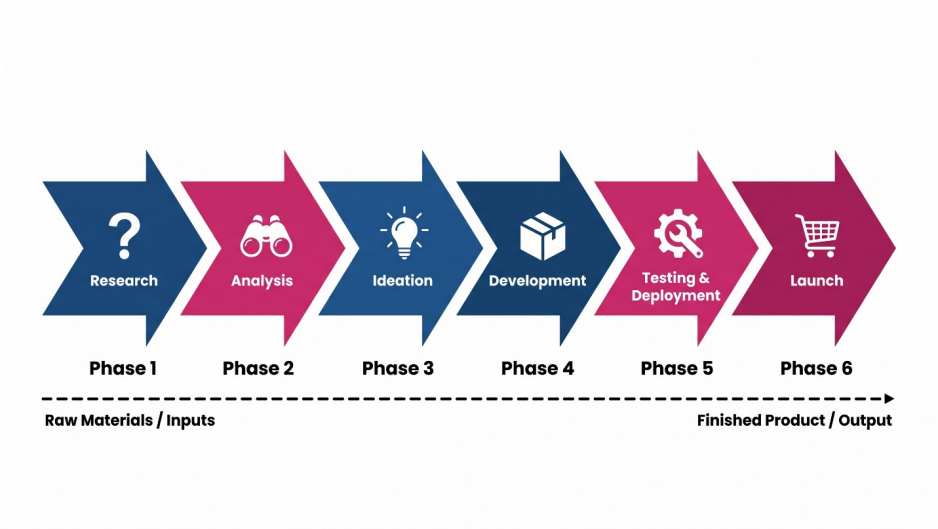


Figure 4.1: Production Phases

Prototype Phase. The platform has been built as a fully functional proof of concept prototype, validating the core concept and fulfilling startup incubation requirements. This build is based on a **Modular Monolith** architecture using FastAPI and React/TypeScript, enabling rapid development and validation of the RAG algorithms and PostgreSQL database layer.

Commercial Production Phase (Transition to Microservices). Upon market launch, the platform will undergo a gradual architectural evolution toward a full **Microservices Architecture**. Core components including the file processing & analysis service, the RAG retrieval engine, and the user account management service will be decoupled into fully independent services to ensure:

- **Maximum Scalability:** Independent scaling of each service based on load demand and traffic balancing.
- **High Fault Tolerance:** Platform stability guaranteed such that a failure in one service does not cascade into a full system outage.

Quality Assurance (QA). The production process includes continuous integration testing across microservices, alongside rigorous security and data protection testing to ensure the confidentiality of user-uploaded files.

Data Privacy Compliance: All user-uploaded academic content is stored in isolated environments. The platform adheres to Algerian data protection regulations and applies end-to-end encryption for all documents.

Automated Cloud Deployment (CI/CD). Services are containerized via **Docker** and deployed on flexible cloud environments, ensuring seamless platform updates and maintenance with zero service interruption for end users.

4.2 Supply & Infrastructure

For a digital platform, supply chain management is not material-based it centers on securing robust software and cloud infrastructure:

- **Procurement Policy:** Subscriptions to cloud platforms (PaaS/IaaS) for server hosting, database operations, and AI API consumption.
- **Key Suppliers:** Cloud hosting providers (e.g., **Vercel** for frontend, reliable back-end server providers), cloud database services (**Supabase**), and large language model providers.
- **Payment Policy:** Flexible consumption-based models including **Pay-as-you-go** and monthly/annual subscriptions to maintain precise control over operational costs.

4.3 Workforce

Given the advanced technical nature of the project:

- **Current Headcount & Profile:** The project currently operates with 2 direct positions full-stack software engineers specializing in AI development responsible for managing all technical aspects of the platform and leading the upcoming architectural transition.
- **Job Creation Potential:** At full operational scale, EduTwin AI is projected to generate approximately 50 direct jobs and nearly 100 indirect jobs, spanning technical, commercial, and administrative functions.
- The project demands specialized talent primarily software engineers and AI/cloud technicians sourced from Algerian universities and engineering schools. The core technical unit will comprise **3 engineers and 6 technicians** dedicated to building and operating the platform's infrastructure.
- **Projected Growth Roles (Post-Funding):** Planned additions include a Customer Support Specialist, DevOps / Cloud Engineer, Sales & Business Develop-

ment Officer, and In-house UI/UX Designer.

- **Outsourcing Strategy:** Non-core functions including digital marketing, graphic design, and legal & accounting advisory will be outsourced in the pre-funding phase to minimize fixed overhead.

4.4 Key Partnerships

To ensure platform growth and market credibility, the project relies on the following strategic partnerships:

- **Academic Partnerships (Business Incubators):** Active presence within the University of Skikda's business incubator to benefit from mentorship, guidance, and direct access to the first wave of users for platform testing and validation.
- **Research Laboratories & University Libraries:** Partnerships to supply trusted academic content, enriching the platform's knowledge bases and expanding the range of supported curricula.
- **Technology Partnerships:** Institutional level agreements with cloud technology providers to guarantee service stability and seamless scalability as the system transitions to a microservices environment.
- **Governmental Partnerships:** Seeking formal collaboration with the Ministry of Higher Education to integrate EduTwin AI within the national e-learning framework.

Part Five: Financial Plan

5.1 Costs & Expenses

The following table shows all project costs and burdens required to launch the **EduTwin AI** platform, tailored for a SaaS architecture (Cloud infrastructure, LLM APIs, and AI models):

Table 5.1: Project costs and expenses

Costs and Expenses	Amount (DZD)
Development Costs	
Frontend Development (React/Tailwind)	150,000
Backend & RAG Pipeline (FastAPI)	350,000
BKT Model Integration & Testing	200,000
UI/UX Design	100,000
Operational Costs (First Year)	
Vector Database & Hosting (Supabase/Vercel)	150,000
LLM API Consumption (Token usage)	250,000
Total	**400,000**
Marketing and Sales Costs (First Year)	
Digital Advertising (Social Media/SEO)	300,000
Academic Seminars & University Events	100,000
Total	**400,000**
Staff Salaries (First Year)	
Software Engineers (2 Founders)	1,200,000
Part-time Customer Support (1)	210,000
Total	**1,410,000**
Legal and Compliance Costs	
Enterprise Registration & Legal Consultations	150,000
Data Privacy Compliance (Algerian Regulations)	60,000
Total	**210,000**
Total Initial Investment (First Year) in DZD	
Development Costs	800,000
Operational Costs	400,000
Marketing and Sales Costs	400,000

Costs and Expenses (continued)	Amount (DZD)
Staff Salaries	1,410,000
Legal and Compliance Costs	210,000
TOTAL REQUIRED INVESTMENT	**3,220,000**

Table 5.2: Sources of Funding

Sources of Funding in Algeria	Amount (DZD)
Personal Savings (Founders)	1,000,000
University Incubator / Local Angel Investment	2,220,000
Total Funding	**3,220,000**

Financial Methodology.

- **Total Initial Investment (3,220,000 DZD):** Aggregates all foundational first-year expenses, including development, cloud operations, marketing, salaries, and legal compliance.
- **Sources of Funding:** Designed to cover 100% of the initial investment through a mix of founders' personal savings (1M DZD) and external seed funding (2.22M DZD) to prevent early liquidity shortages.

5.2 Business Number (Turnover)

The project will generate revenue primarily through a SaaS subscription model (Pro Plan and Ultra Plan). There are two possible outcomes: an optimistic and a pessimistic result.

5.2.1 Optimistic Scenario 1st Year

Table 5.3: Expected turnover in optimistic scenario

Month of the Year	Forecast Turnover Excluding Tax (DZD)
January	800,000.00
February	850,000.00
March	900,000.00
April	950,000.00
May	1,000,000.00
June	1,100,000.00
July	1,200,000.00
August	1,300,000.00
September	1,400,000.00

October	1,500,000.00
November	1,600,000.00
December	1,750,000.00
TOTAL HT	**14,350,000.00**
MONTANT TVA 19%	**2,726,500.00**
TOTAL TTC	**17,076,500.00**

5.2.2 Pessimistic Scenario 1st Year

Table 5.4: Expected turnover in pessimistic scenario

Month of the Year	Forecast Turnover Excluding Tax (DZD)
January	400,000.00
February	420,000.00
March	450,000.00
April	480,000.00
May	500,000.00
June	520,000.00
July	550,000.00
August	580,000.00
September	600,000.00
October	620,000.00
November	650,000.00
December	700,000.00
TOTAL HT	**6,470,000.00**
MONTANT TVA 19%	**1,229,300.00**
TOTAL TTC	**7,699,300.00**

Turnover Calculation.

- **TOTAL HT:** The cumulative monthly subscription revenue generated from both Pro and Ultra plans before tax.
- **MONTANT TVA 19%:** The standard Algerian Value-Added Tax applied to digital and software services.
- **TOTAL TTC:** The actual total liquidity collected from users, calculated as (TOTAL HT + TVA).

5.3 Expected Result Statement

Table 5.5: Calculations of expected results

Item	1st Year (Optimistic Base)
Operating Revenue	
Pro Plan Subscriptions (Students)	8,352,000.00
Ultra Plan Subscriptions (Researchers/Inst.)	5,998,000.00
gross margin	**14,350,000.00**
External costs	
Cloud Hosting & Database (Supabase)	150,000.00
LLM API Consumption	250,000.00
Digital marketing and advertising	400,000.00
value-added	**13,550,000.00**
Employees' salaries	1,410,000.00
Employees social charges (CNAS approx 26%)	366,600.00
gross operating surplus	**11,773,400.00**
Depreciation allowance (Hardware/Servers)	200,000.00
Pre-tax profit	**11,573,400.00**
Corporation tax (IBS 26%)	3,009,084.00
Net accounting income	**8,564,316.00**

Revenue Breakdown & Profit Calculation.

- **Operating Revenue (14,350,000 DZD):** Derived from accumulated first-year subscriptions: **Pro Plan** (16,704 users × 500 DZD) and **Ultra Plan** (2,000 users × 2,999 DZD).
- **Value-Added:** Calculated by deducting external operational costs (Cloud, LLM APIs, and Marketing) from the gross margin.
- **Gross Operating Surplus:** Value-Added minus human resource costs (Net salaries + ~26% CNAS social charges).
- **Net Accounting Income:** Pre-tax profit minus the standard Algerian Corporate Income Tax (IBS) set at 26% for service-based companies.

5.4 Cash Flow Plan

The following table shows all expected cash collections and disbursements during the first year, demonstrating the liquidity health of the EduTwin platform.

Cash budget over 12 months (Optimistic Scenario — All values in DZD)

Table 5.6: Expected revenues and expenses cash flow during the first year

Month	1	2	3	4	5	6	7	8	9	10	11	12
COLLECTIONS												
Gross sales (TTC)	952,000	1,011,500	1,071,000	1,130,500	1,190,000	1,309,000	1,428,000	1,547,000	1,666,000	1,785,000	1,904,000	2,082,500
Funding	3,220,000	0	0	0	0	0	0	0	0	0	0	0
TOTAL ENTRIES	**4,172,000**	**1,011,500**	**1,071,000**	**1,130,500**	**1,190,000**	**1,309,000**	**1,428,000**	**1,547,000**	**1,666,000**	**1,785,000**	**1,904,000**	**2,082,500**
DISBURSEMENTS												
Fixed/Dev Assets	800,000	0	0	0	0	0	0	0	0	0	0	0
Net wages	117,500	117,500	117,500	117,500	117,500	117,500	117,500	117,500	117,500	117,500	117,500	117,500
Social security	30,550	30,550	30,550	30,550	30,550	30,550	30,550	30,550	30,550	30,550	30,550	30,550
TOTAL SORTIES	**1,014,716**	**366,716**	**376,216**	**385,716**	**395,216**	**404,716**	**423,716**	**442,716**	**461,716**	**480,716**	**499,716**	**518,724**
BANK BALANCE												
Sold final	**3,157,284**	**3,802,068**	**4,496,852**	**5,241,636**	**6,036,420**	**6,940,704**	**7,944,988**	**9,049,272**	**10,253,556**	**11,557,840**	**12,962,124**	**14,525,900**

Cash Flow Methodology.

- **Total Entries:** Includes the initial seed funding in Month 1, plus the monthly Gross Sales inclusive of tax (TTC).
- **Total Disbursements (Sorties):** Includes one-time fixed asset costs (Month 1), evenly distributed operational expenses (salaries, cloud, APIs), and VAT paid to authorities the following month.
- **Final Bank Balance:** Calculated as *(Previous Balance + Entries − Disbursements)*. The consistent growth confirms robust liquidity throughout the first year.

Part Six: Experimental Prototype

6.1 The General Architecture of the System

The following represents the general architecture of the system:

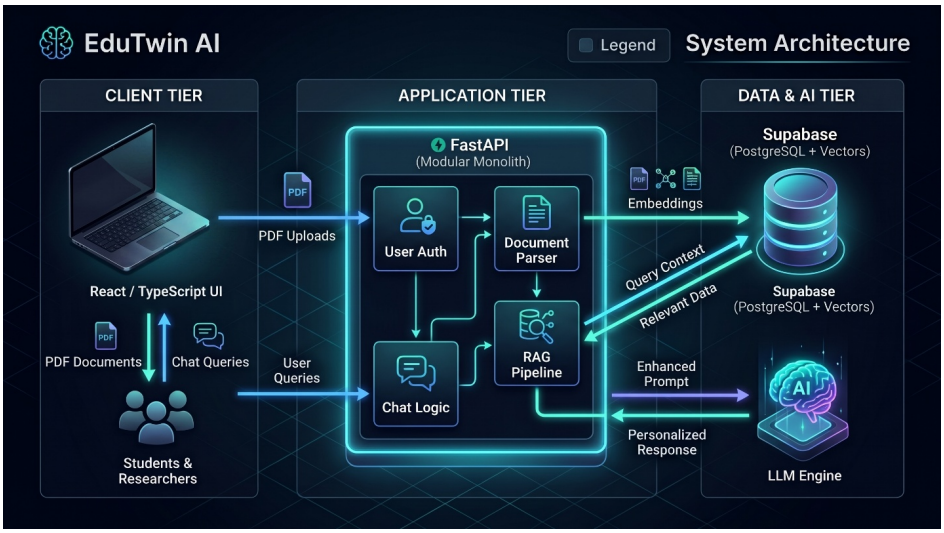


Figure 6.1: The general architecture of the system

6.2 System Structure and Working Principle

In this section we provide a detailed overview of the platform functionalities. It explains how the system operates from both client and administrative perspectives, covering key pages, this section offers a comprehensive understanding of the platform's capabilities and workflow.

6.2.1 Client

★ Landing Page

Once the URL address is entered and launched, a home page of the platform will be displayed.

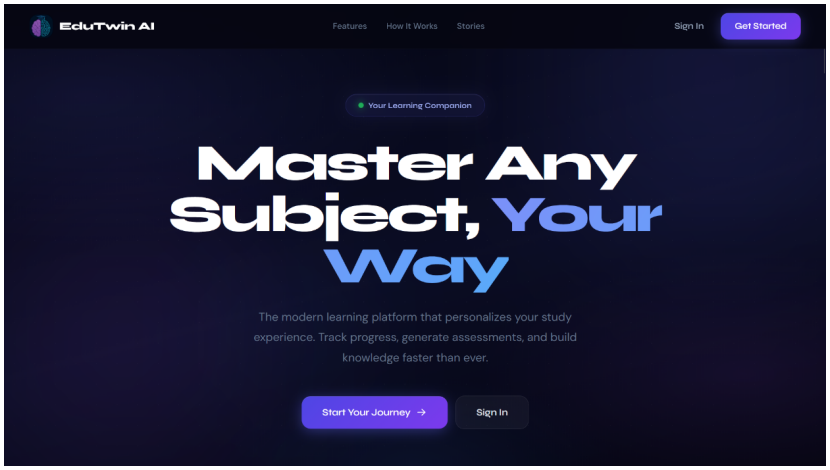


Figure 6.2: Landing Page

★ Sign Up Page

Our platform provides its visitors with an accessible registration form.

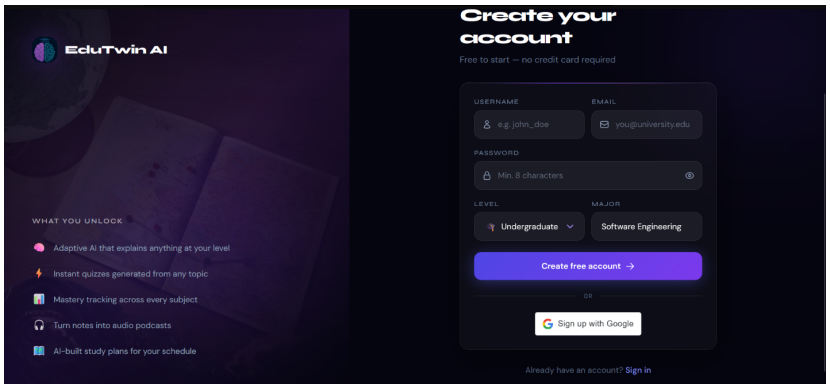


Figure 6.3: Sign Up Page

★ Main Page

The main page provides users with access to the interactive AI chat interface, document upload functionality, and other essential features like chat history and direct source citations for a seamless and personalized learning experience.

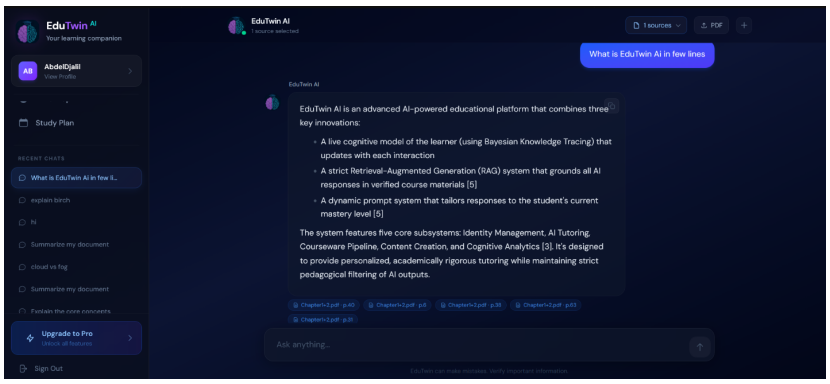


Figure 6.4: Main Page (Chat Page)

★ Digital Twin Dashboard

The Digital Twin dashboard provides users with a comprehensive visual representation of their cognitive profile, featuring real-time mastery tracking, predictive learning analytics, and personalized insights to optimize their educational journey.

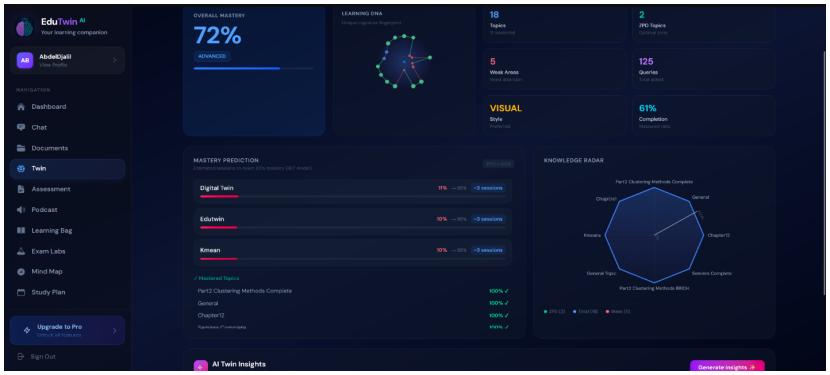


Figure 6.5: Digital Twin Dashboard

★ Assessment Interface

The assessment page provides users with a structured environment for evaluating their knowledge, featuring an interactive multiple-choice quiz format, clear progress tracking, and intuitive navigation to reinforce learning outcomes.

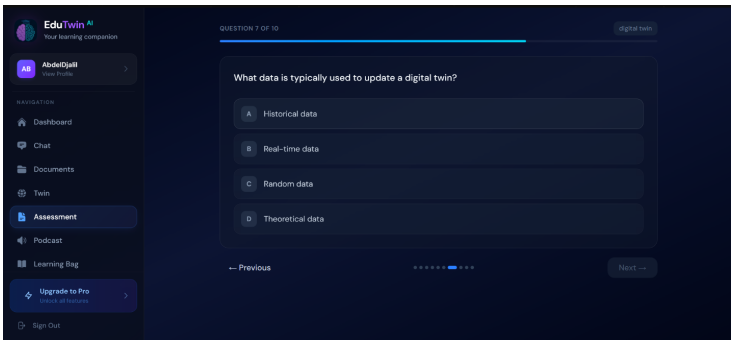


Figure 6.6: Assessment Interface

★ Admin Profile Page

The admin profile page provides access to manage and update administrative account information and settings.

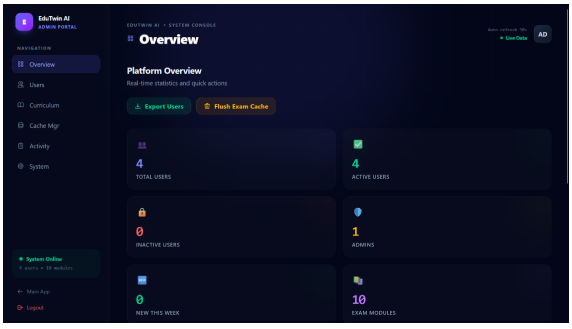


Figure 6.7: Admin Profile Page

Part Seven: Business Model Canvas

The Business Model Canvas (BMC) is a structured framework that visually represents key elements of a business model, helping to clarify, design, and innovate business strategies effectively.

Table 7.1: BMC

<table>
<tr><th>Key Partners</th><th>Key Activities</th><th>Value Propositions</th><th>Customer Relationships</th><th>Customer Segments</th></tr>
<tr><td>• University Incubators
• Cloud Providers (Vercel, DigitalOcean)
• Database Providers
• AI API Providers
• Research Laboratories
• Ministry of Higher Education (governmental partnership)</td><td>• Platform Development
• RAG Pipeline Engineering
• Cloud Infrastructure Management
• Marketing and Sales
• Partnership Management</td><td>• Hallucination-free AI (RAG)
• Personalized Learning (BKT)
• Data Privacy & Security
• Affordable Pricing in DZD
• Enhanced Study Efficiency
• Local Compliance (Algerian regulations)</td><td>• Dedicated Onboarding Support
• Automated Self-Service (SaaS)
• Personalized AI Insights
• Dedicated Tech Support
• User Feedback Loops</td><td>B2C:
• University Students
• Academic Researchers
• Teachers / Professors
B2B:
• Universities
• Research Centers
• Academic Incubators
• Educational Institutes
B2G:
• National Educational Programs</td></tr>
<tr><th colspan="3">Key Resources</th><th colspan="2">Channels</th></tr>
<tr><td colspan="3">• SaaS & Cloud Infrastructure
• Proprietary Adaptive BKT Model
• Vector Databases
• AI Software Engineers</td><td colspan="2">• EduTwin Web Platform
• SEO & Social Media
• University Partnerships
• Academic Seminars
• Freemium Word-of-Mouth
• Beta Testing Program (Skikda University)</td></tr>
<tr><th colspan="3">Cost Structure</th><th colspan="2">Revenue Streams</th></tr>
<tr><td colspan="3">• Cloud Hosting & Server Infrastructure
• Supabase & Vector Storage Fees
• LLM API Consumption
• Marketing and Advertising
• Platform Development & Maintenance
• LLM API Token Budget Management</td><td colspan="2">• Freemium Tier (free access with limited token budget)
• B2C Subscriptions Pro Plan (500 DZD/month)
• B2C Subscriptions Ultra Plan (2,999 DZD/month)
• B2B Institutional Contracts (bulk pricing for universities)
• Value-Added Custom Integrations</td></tr>
</table>

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/ إ/ أ/ ك.ع/ 2026

Autorisation de Dépôt de Mémoire de Master

Je soussigné : Dr. Kerraoui Sara

Certifie que l'étudiant(e) : Ramdane Abdeldjalil

Spécialité : GLAA

Ayant soutenu le projet intitulé : E.du.Twin.AI: A Learner-Centric Digital Twin and Retrieval-Augmented Generation platform for Personalized Higher Education

A apporté les corrections nécessaires sur son manuscrit de Master.

Signature de l'encadreur



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/ إ/ أ/ ك.ع2026/

Autorisation de Dépôt de Mémoire de Master

Je soussigné : ..Dr. Kenaani Sara.....

Certifie que l'étudiant(e) : ..Guemiche Ala Abdel Nourmene.....

Spécialité : .GLAA.....

Ayant soutenu le projet intitulé : Edu Twin AI : A Learner-Centric Digital Twin and Retrieval-Augmented Generation platform for Personalized Higher Education

A apporté les corrections nécessaires sur son manuscrit de Master.

Signature de l'encadreur