République Tunisienne
*****
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
*****
Université de Monastir
*****
Institut Supérieur d'Informatique et de Mathématiques de Monastir
*****
Département Informatique

N° d'ordre : L14-INFO

Project Memory End of Studies

Presented with a view to obtaining the

National Bachelor of Science Diploma

Computer Science

Speciality :

Software and Information System Engineering

By
Jaziri Ahmed

Development of a mobile application and web back office for real estate investment

Defended on 6/24/2025 in front of the jury composed of:

Mr. : JAOUA Zied	**President**
Ms. : MILLI Manel	**Reporter**
Ms. : BALI Nadia	**Educational Supervisor**
Mr. : ZOUARI Khalil	**Technical Supervisor**

SUMMARY

This work is part of our Final Year Project at the Higher Institute of Computer Science and Mathematics of Monastir for 2024-2025. Conducted at **«KZ IT Services»**, we developed **«KORPOR»**, a real estate investment platform with a mobile app and web back-office. Using MySQL, Express-Node.js, React, and Vite, the platform offers fractional property ownership with AI for valuations and recommendations. Blockchain technology secures transactions while SCRUM methodology guided our development process.

Keywords: Blockchain Technology, AI, MySQL, Express-Node.js, React, Vite, Real Estate Investment.

ABSTRACT

Ce projet s'inscrit dans le cadre de notre projet de fin d'études à l'Institut Supérieur d'Informatique et de Mathématiques de Monastir pour 2024-2025. Réalisé chez **«KZ IT Services»**, nous avons développé **«KORPOR»**, une plateforme d'investissement immobilier avec application mobile et back-office web. Utilisant MySQL, Express-Node.js, React et Vite, la plateforme permet la propriété fractionnée avec IA pour évaluations et recommandations. La blockchain sécurise les transactions tandis que la méthodologie SCRUM a guidé notre processus de développement.

Keywords: Blockchain Technology, AI, MySQL, Express-Node.js, React, Vite, Investissement Immobilier.

Dedication

To the memory of my beloved father, whose guidance and wisdom continue to light my path.
Though no longer with us, your presence remains in every achievement of my life.

To my loving mother, whose strength and endless support shaped who I am today.

To my sister and brother, whose companionship and encouragement have been constant sources of joy and motivation.

To my little Aryouma, whose innocence and love bring happiness to our family every day.

To Mme Nadia, my professors and mentors, who have guided me with knowledge and patience throughout my academic journey.

To my friends, whose encouragement made this journey worthwhile.

This work is dedicated to all of you, but especially to you, Father.

Acknowledgement

With profound gratitude, I acknowledge those whose wisdom and support were instrumental to the Korpor platform's development:

Academic Committee
Dr. Zied JAOUA & **Prof. MANEL MILLI** — for their invaluable evaluation and guidance.

Supervisor
Dr. NADIA BALI — whose mentorship illuminated the path through complex technological challenges.

Company Leader
Mr. KHALIL EZOUIRI — for his visionary leadership and providing the opportunity to contribute to this innovative platform.

Institution
Faculty and staff who fostered an environment of innovation and excellence.

This work stands as testament to the transformative potential of integrating blockchain security with AI-driven insights to democratize investment opportunities.

With appreciation,
Ahmed Jaziri
June 9, 2025

TABLE OF CONTENTS

LIST OF FIGURES

"Real estate cannot be lost or stolen, nor can it be carried away. Purchased with common sense, paid for in full, and managed with reasonable care, it is about the safest investment in the world."[1]

— Franklin D. Roosevelt
32nd President of the United States

General Introduction

In today's rapidly evolving financial landscape, traditional investment methods are often burdened by opaque processes, cumbersome bureaucracy, and significant entry barriers. Investors have long struggled with outdated systems that impede transparency, elevate risks, and complicate access to promising opportunities. Such challenges not only limit diversification but also expose users to uncertainties that modern technology can easily overcome.

Korpor was conceived to transform this paradigm by delivering a fully integrated, AI and blockchain-powered mobile investment platform. By harnessing advanced data analytics, machine learning, and cutting-edge blockchain technology, Korpor streamlines every facet of the investment process. The application offers a seamless user onboarding experience, intuitive project listings enriched with AI-driven recommendations, and a secure, automated investment flow that simplifies transactions while ensuring that every operation is recorded immutably on the blockchain.

Security and trust are at the heart of Korpor's design. By employing state-of-the-art encryption, blockchain-based transparency, and strict compliance measures, the platform safeguards sensitive financial data and guarantees that every transaction is executed within a secure and verifiable framework. Continuous monitoring, performance optimization, and the immutable nature of blockchain records further ensure that the application remains resilient, scalable, and resistant to fraud.

Developed under a flexible **Agile framework** that combines iterative development with strategic project management best practices, Korpor is designed to rapidly adapt to evolving market trends and user needs. This methodical approach allows for regular feedback, swift enhancements, and the seamless integration of innovative features throughout the development lifecycle.

Document Structure:

- **Chapter 1: Project Context** — Explores the industry challenges and the vision that inspired Korpor's creation.
- **Chapter 2: Analysis and Specification** — Outlines the comprehensive requirements gathering, analysis, architectural design, and the selection of cutting-edge tools and technologies.
- **Subsequent Chapters** — Document the progressive implementation of core features—from AI-enhanced project recommendations and blockchain-secured transactions to comprehensive portfolio management.

Through this structured approach, we demonstrate how **Korpor** leverages modern technology to reimagine investment management, offering a secure, transparent, and dynamic solution that is set to redefine digital financial engagement.

CHAPTER 1

Project Context

Introduction

The aim of this chapter is to present the general framework of the Korpor project, a solution dedicated to real estate investment. In this chapter, we'll discuss successively:

The presentation of the host organization and the context and challenges of the real estate sector and the analysis of existing solutions and identification of their limitations.

1.1 Project Context

This work is part of the end-of-study project for the national diploma of Applied Bachelor's degree in Computer Science from the Higher Institute of Computer Science and Mathematics of Monastir (ISIMM) for the year 2024/2025. we has the opportunity to do our end-of-study internship at the company « KZ IT Services », under the supervision of Mr. Khalil Zouari.

1.2 Hosting Company

The purpose of this section is to present the company within which I developed my project, as shown in Figure 1.1.

Figure 1.1: Hosting Company « KZ IT Services »

Table 1.1: KZ IT Services - Company Information

Aspect	Details
Company Name	KZ IT Services
Specialization	Custom software development, IT solutions, Digital transformation
Mission Focus	Delivering innovative, robust, and scalable IT solutions to drive client efficiency and success through quality and continuous improvement.
Size	2-10 employees
Location	Djerba, Tunisia

KZ IT Services is a dynamic software company dedicated to delivering innovative IT solutions tailored to modern business needs. they are specialize in designing and developing robust, scalable applications that drive efficiency and digital transformation. thier experienced team leverages cutting-edge technology to create customized software that exceeds client expectations. With a strong commitment to quality and continuous improvement, they build lasting partnerships based on trust and excellence. At « KZ IT Services », innovation is at the core of everything they do, empowering thier clients to achieve sustainable growth and success.

1.3 Preliminary Study

This preliminary study provides a review of some existing investment and asset management platforms. Further, the next section identifies some key concepts that will lead to further understanding of the domain in question.

1.3.1 Existing Solutions Study

After conducting extensive research on investment platforms similar to our concept across the global market, we carefully analyzed numerous applications based on their performance metrics and market position. From this comprehensive study, we specifically selected « Aseel » [2] and « Stake » [3] for in-depth analysis [4, 5] due to their exceptional performance and status as leading companies in the real estate investment platform sector.

The Aseel Platform

Aseel [2] is a portal through which users can invest in different real estate projects with ease. The interface allows the clients to surf various investment opportunities, view

the details of the properties, and then make an informed decision. Aseel introduces transparency in the investment process by offering financial data, updates regarding projects, and returns that are estimated. This platform comes with an easy-to-use dashboard through which one tracks their investments and manages their assets without any hassle. The interface of the Aseel Platform is shown in Figure 1.2.

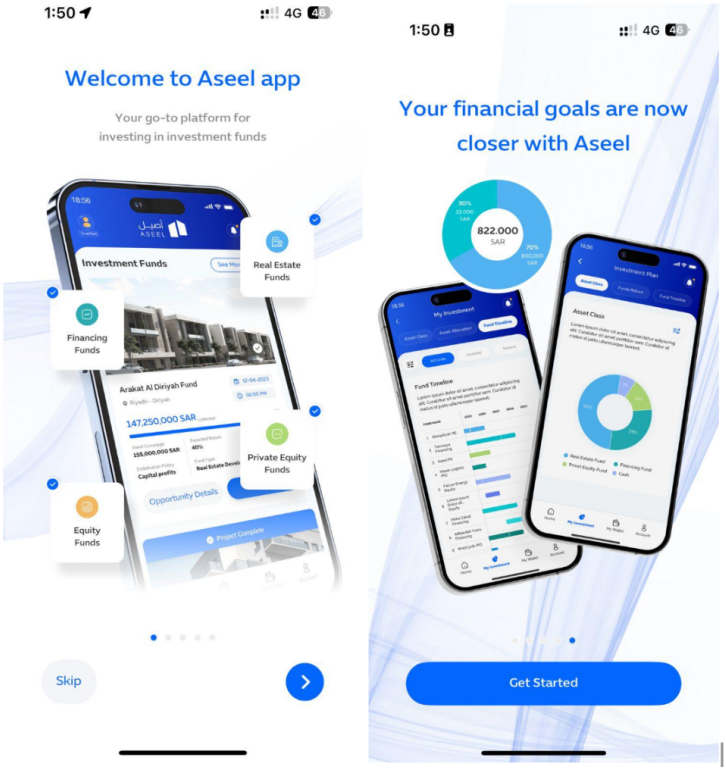


Figure 1.2: Interface of « The Aseel Platform »

The Stake Platform

Stake [3] is an online investment platform that deals with real estate crowdfunding. It provides the opportunity to invest in fractions of property ownership, hence diversifying a portfolio without huge capital. On Stake, there are AI-powered recommendations based on user preferences, seamless payment integration, and a secure environment for investment. Besides, liquidity is guaranteed by enabling exit options for investors who may want to sell their shares in ongoing projects. Figure 1.3 illustrates the interface of the Stake Platform.

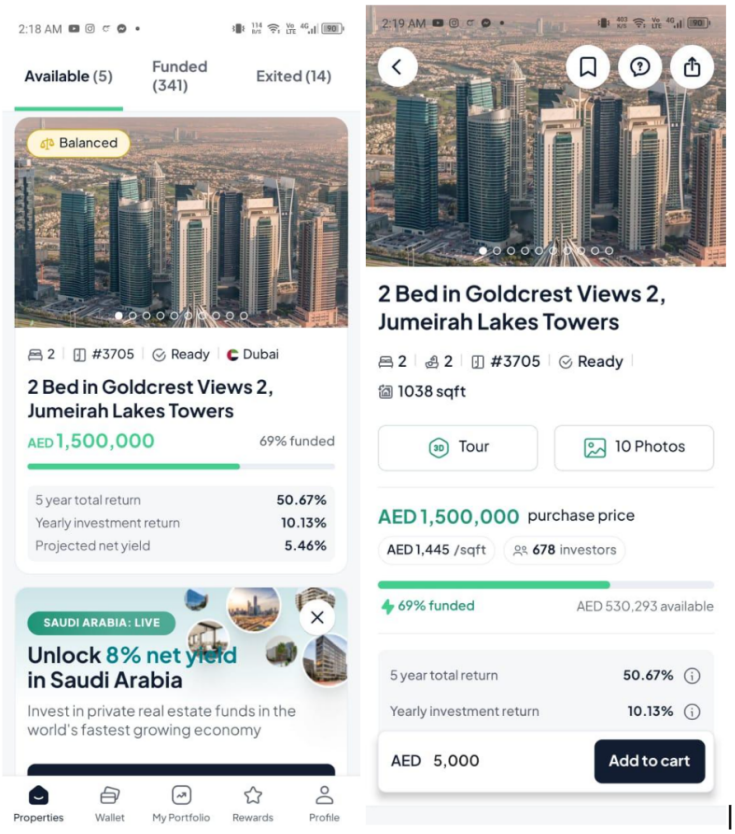


Figure 1.3: Interface of « The Stake Platform »

1.3.2 Comparative and Critical Analysis

We can summarize all that comes from our analysis based on a number of criteria used for the evaluation of these applications [6, 7].

- **Speed (C1)**: The platform should obtain value for the user as fast as possible and effectively, anticipating their proliferating expectations.
- **Costs (C2)**: With minimum software development costs, it is important to keep the pricing predictable and acceptable.
- **Quality (C3)**: Since the market expects quality, any kind of error might affect brand reputation. Improvement of the platform should be regular.
- **Reliability (C4)**: Since modern-day investment platforms need to make sure of minimum downtime and maximum availability of services, this factor is critical.
- **Security (C5)**: Such an investment platform enforces access rights, roles, and contribution rights through a powerful security system.
- **Performance (C6)**: Crucial features include AI-powered recommendations going through seamlessly, easy transaction tracking, and investment monitoring.

- **Stability (C7)**: The platform should have a proven track record, regular updates, and a large user base to ensure its longevity.
- **Resilience (C8)**: In order to prevent data loss and guarantee a smooth experience for investors, it must be able to restore lost functionalities should issues occur.
- **User Experience (C9)**: The interface should be intuitive and user-friendly, hence allowing investors to move with ease through it, thus making wiser decisions.

Table 1.2 presents the evaluation of the existing solutions based on these criteria:

Table 1.2: Evaluation Table

Solution	C1	C2	C3	C4	C5	C6	C7	C8	C9
Stake	✓	✓	✓	✓	✓	×	✓	✓	✓
Aseel	✓	✓	✓	×	✓	×	✓	×	✓

1.3.3 Proposed Solution

The **Korpor** platform is a dual-benefit real estate solution for agencies and investors, prioritizing a symbiotic ecosystem where agency efficiency enhances investor opportunities and experience.

Benefits for Real Estate Agencies

- **Expanded Market Reach & Sales Scalability:** Showcase properties to more investors and scale sales via blockchain-recorded fractional ownership, broadening the investor base.
- **Streamlined Operations & Enhanced Presence:** Efficiently manage listings, client interactions, and bolster digital brand image, gaining insights into market trends.

Benefits for Investors

- **Accessible & Simplified Investments:** Discover diverse real estate opportunities, including fractional ownership, with simplified processes that handle paperwork and rent collection.
- **AI-Powered Guidance & Secure Transactions:** Receive personalized AI recommendations and experience secure, transparent, blockchain-recorded transactions with robust regulatory adherence.
- **Seamless Portfolio Management:** Track investments in real-time via a user-friendly dashboard and mobile app.

1.4 Development methodology

Meeting project delivery deadlines is a critical challenge in software development. Common issues include insufficient technical specifications, poor time management with emerging technologies, and sudden requirement changes. To address these challenges, we follow an agile methodology using Git [8] for version control and GitHub [9] for collaborative development.

1.4.1 SCRUM

Scrum is an agile development approach that is used to create software using incremental and iterative methods. Scrum is a quick, flexible, and efficient agile methodology that is intended to provide value to the client at every stage of the project's development [10]. Scrum is founded on empiricism and lean thinking, employing an iterative, incremental approach guided by the three pillars of transparency, inspection, and adaptation [11, 10]. Scrum's main goal is to meet customer needs by fostering an atmosphere of open communication, group accountability, and constant improvement, underpinned by the Scrum values of Commitment, Focus, Openness, Respect, and Courage [10]. The development process begins with a broad concept of what must be constructed, developing a list of features that the product owner desires, and arranging them according to priority (product backlog).

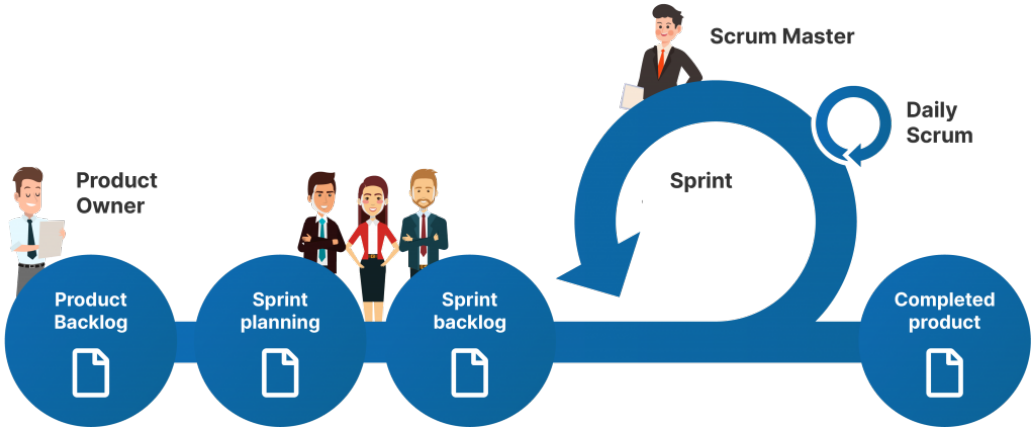


Figure 1.4: Agile Scrum Framework Process

1.4.2 Agile Scrum roles and responsibilities

The Product Owner

The Product Owner understands the customer and business requirements, then creates and manages the product backlog based on those requirements. Their key responsibilities include managing the scrum backlog, overseeing release management, and handling stakeholder management to ensure project alignment with business objectives.

Developers

In Scrum, the term developer or team member refers to anyone who plays a role in the development and support of the product and can include researchers, architects, designers, programmers, etc. Developers are responsible for delivering the work through the sprint and ensuring transparency during the sprint by meeting daily at the daily scrum to discuss progress and challenges.

Scrum Master

The Scrum Master is the role responsible for gluing everything together and ensuring that scrum is being done well. In practical terms, that means they help the product owner define value, the development team deliver the value, and the scrum team get better. The Scrum Master focuses on maintaining transparency, promoting empiricism, encouraging self-organization, and facilitating the Scrum events effectively.

1.4.3 The Scrum Events

The Scrum events are key elements of the Scrum Framework. They provide regular opportunities for enacting the Scrum pillars of Inspection, Adaptation and Transparency [10]. In addition, they help teams keep aligned with the Sprint and Product Goals, improve Developer productivity, and remove impediments and reduce the need to schedule too many additional meetings.

- **Sprint**: All work in Scrum is done in a series of short projects called Sprints. This enables rapid feedback loops.
- **Sprint Planning**: The Sprint starts with a planning session in which the Developers plan the work they intend to do in the Sprint. This plan creates a shared understanding and alignment among the team.
- **Daily Scrum**: The Developers meet daily to inspect their progress toward the Sprint Goal, discuss any challenges they've run into, and tweak their plan for the next day as needed.
- **Sprint Review**: At the end of the Sprint, the Scrum Team meets with stakeholders to show what they have accomplished and get feedback.
- **Sprint Retrospective**: Finally, the Scrum Team gets together to discuss how the Sprint went and if there are things they could do differently and improve in the next Sprint.

Conclusion

It is clear that planning and methodology are essential pillars to ensure the success of the project. By fully understanding the project framework, including the host organization's expectations and the challenges ahead, the team is better prepared to meet the challenges ahead.

This chapter lays the solid foundation on which the entire project will be built, providing a valuable guide for the next steps. The next chapter will allow us to analyze and specify the requirements developed for our project.

CHAPTER 2

Requirements Specification and Analysis

Introduction

In this chapter, we will present the analysis and specification of Requirements. We start by presenting the specification of the requirements, illustrating them using global use case diagram. Then we will present our project architecture and our working environment, and finally the product backlog and release planning, and we will close our chapter with a conclusion.

2.1 Requirements Specification

In this section, we will define the actors of our application and the functional and non-functional Requirements that our application aims to fulfill.

2.1.1 Identifying Actors

We define actors as a shorthand for the roles played by entities outside the system that interact directly with them [12, 13]. In our system, we identify four types of actors:

- **Super Admin**: Responsible for the global configuration of the platform, they have extended privileges to manage administrators, oversee security, and ensure compliance. They can also configure advanced features and control all system resources.
- **Admin**: In charge of the day-to-day management of the platform, they can add, modify, or delete listings, supervise agency and user profiles, and ensure smooth operations. They are also responsible for monitoring and assisting other actors.
- **Real Estate Agent**: Dedicated to creating and updating real estate listings, they manage property information, handle investor requests, and finalize transactions related to sales or rentals. They can also coordinate property visits and propose tailored offers.
- **Investor**: A user who wishes to browse and finance real estate projects. They have access to all available offers, can make investments in a few simple steps, and monitor the evolution of their portfolio. They also benefit from personalized insights to optimize their investments.

2.1.2 Functional Requirements

After several meetings with our client, the various functional requirements of our application are illustrated as follows:

For the Super Admin (Korpor)

- **Authenticate**: The super admin enters their credentials to access the advanced management console.
- **Log Out**: After viewing or updating global settings, they can securely log out.
- **Manage Admin Accounts**: Create, enable/disable, or modify admin profiles associated with different real estate companies.
- **Monitor Security & Compliance**: Oversee transactions, data integrity, and regulatory adherence using specialized reporting and audit tools.
- **View Global Reports**: Generate and analyze consolidated metrics (financials, user activity, transactions) for overall performance insights.
- **Moderate Content**: Review and remove any inappropriate or erroneous property listings or user-generated data.

For the Admin (Real Estate Company)

- **Authenticate**: The admin logs in with valid credentials to manage daily operations.
- **Log Out**: They can end their session to maintain account security.
- **Manage Real Estate Listings**: Add, update, or delete property listings visible to investors.
- **Oversee Real Estate Agents**: Create and manage agent accounts, assign properties, and monitor performance and commissions.
- **Track Transactions & Commissions**: Review incoming payments, calculate commissions owed to agents, and track the history of completed deals.
- **Address Investor Inquiries**: Respond to questions or concerns from investors, ensuring a smooth user experience.
- **Access Agency Dashboard**: View comprehensive statistics on properties, sales, rentals, and market trends.

For the Real Estate Agent

- **Authenticate**: The agent logs in to manage assigned properties and interact with potential investors.
- **Log Out**: Securely exit the account after completing tasks.

- **Manage Assigned Properties**: Create new listings, update property details, set prices, and upload images.
- **Handle Investment Requests**: Review purchase or rental offers, negotiate terms, and initiate contract finalization.
- **Contribute to AI Estimates**: Provide or refine data to improve AI-driven pricing and market analysis.
- **Maintain Client Relationships**: Communicate with investors, schedule property visits, and follow up on inquiries.
- **View Commissions**: Track earnings based on successful sales or rentals.

For the Investor (Mobile App User)

- **Create an account & authenticate**: Register to gain access to the platform's core features.
- **Log Out**: End the session to protect personal and financial data.
- **Browse Listings & Invest**: Explore available properties, filter according to preferences, and commit to an investment in a few steps.
- **Track Portfolio**: Monitor owned assets, property status, and receive real-time updates on performance.
- **Make Payments**: Use integrated payment methods (credit cards, digital wallets, etc.) to complete transactions.
- **Access AI Recommendations**: View data-driven insights and return-on-investment estimates generated by the system.
- **Manage Withdrawals & Earnings**: Withdraw profits, monitor rental income, or exit investments under the right conditions.

2.1.3 Non-functional Requirements

In order to ensure the proper functioning of the decision-making system and to avoid any kind of anomaly, the implemented solution must meet a set of non-functional requirements such as:

- **Maintainability**: The system must be designed for simplicity so that tasks, updates, and bug fixes can be executed with minimal complexity [14, 15].
- **Evolution**: Platform administration must remain attentive to user needs and feedback, continuously enhancing the services offered while preserving the application's utility and efficiency [16, 17].

- **Security**: Robust security measures are essential. The platform must enforce strong authentication protocols, access privileges, and comprehensive data encryption (both at rest and in transit) [18, 19]. The integration of blockchain technology further ensures the immutability and integrity of sensitive information [20, 21].
- **Efficiency**: The application must be effective in all circumstances, delivering prompt and reliable functionality regardless of external conditions [22, 23].
- **Performance**: The system must operate optimally across diverse environments. It should consistently provide a responsive and reliable experience, even under high transaction volumes or varying network conditions [24, 25].

General use case diagram

Below, we present the various actors of the application and the actions they are authorized to perform. The overall diagram is illustrated in Figure 2.1:

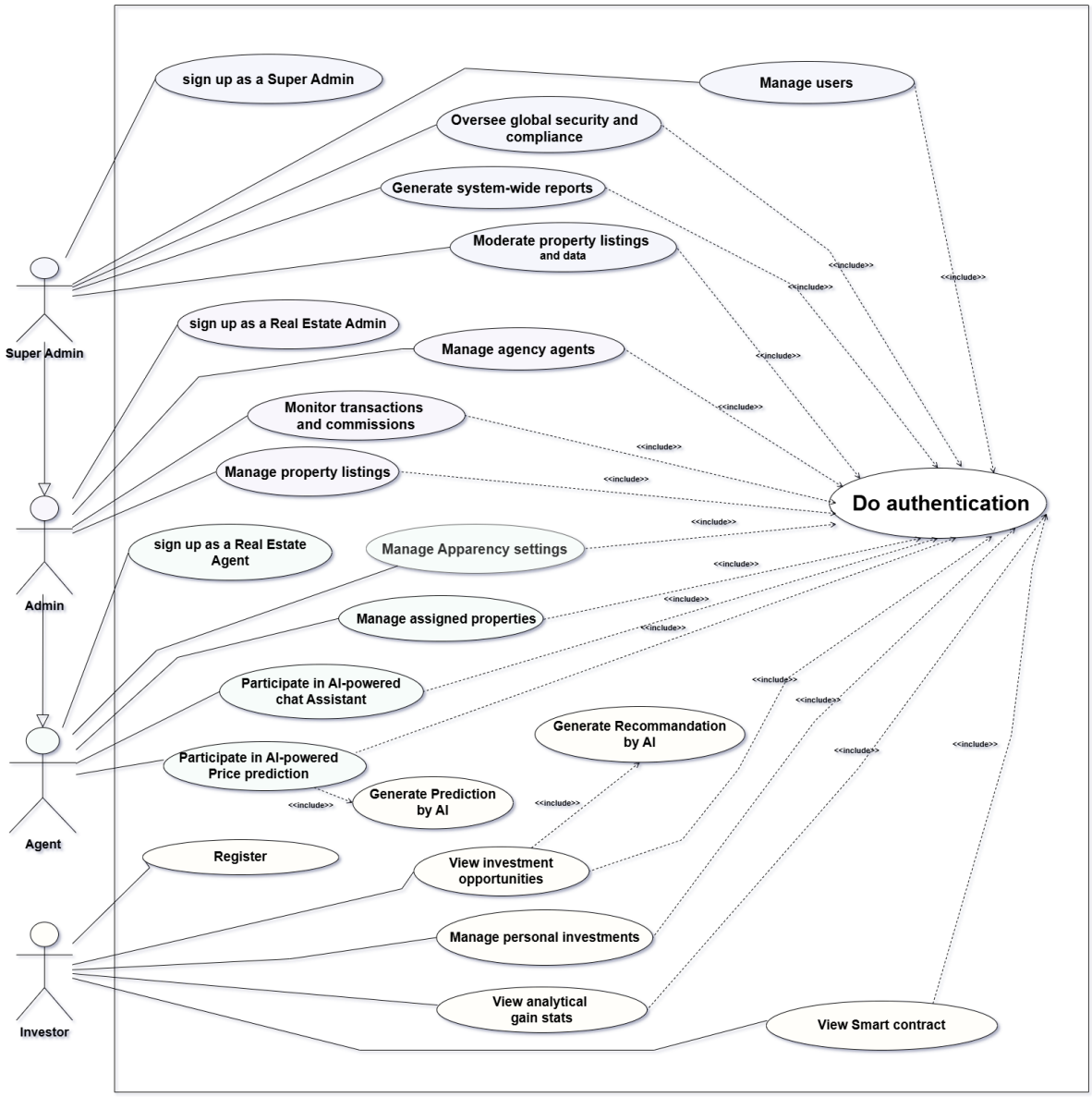


Figure 2.1: General use case diagram

2.2 Software architecture

Before starting the design and development of any computerized system, it is essential to prepare the architecture.

2.2.1 Physical architecture

Korpor's physical architecture, depicted in Figure 2.2, is a distributed system. Users interact via **Mobile Devices** (React Native app) and **Web Browsers** (React components), communicating via HTTP/JSON with backend services primarily on **Google Cloud Platform (GCP)**. GCP handles core logic and data, featuring a **Cloud Run JS Server** for the main backend, a **Cloud Run Flask Server** for AI models (prediction, recommendation, chatbots), **Cloud

SQL** for database storage, and a **Google Cloud VPS** for data scraping models. Blockchain operations leverage **Infura** for Solidity code and the **Sepolia Testnet** for contracts, using RPC and HTTP/JSON. An administrative **Web Panel** is hosted on a **Hosting.com server**.

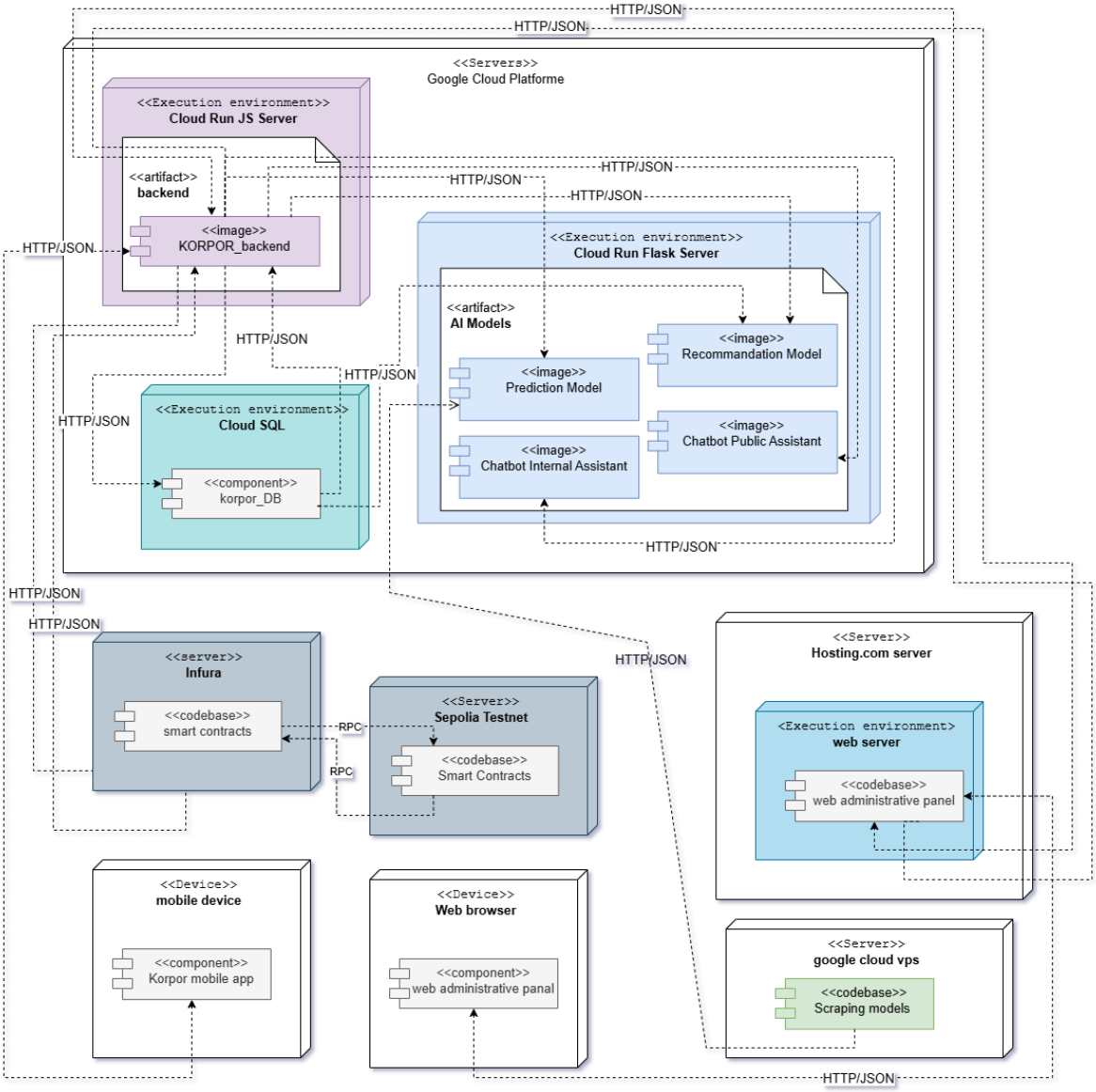


Figure 2.2: Deployment diagram

2.2.2 Logical architecture

Korpor's logical architecture follows the MVC (Model-View-Controller) pattern [26, 27] for maintainability.

- **Model**: Manages application data and business logic, interacting with the backend database.
- **View**: The frontend (React, TypeScript, TanStack) presents data to users and handles interactions.

- **Controller**: The Express.js backend processes requests from the View, interacts with the Model and external services (Clerk, AI, Blockchain), and determines the response.

User requests flow from the View (frontend) to the Controller (backend), which interacts with the Model (data) and services before returning data to update the View. This structure ensures scalability and separation of concerns. The architecture is illustrated in Figure 2.3.

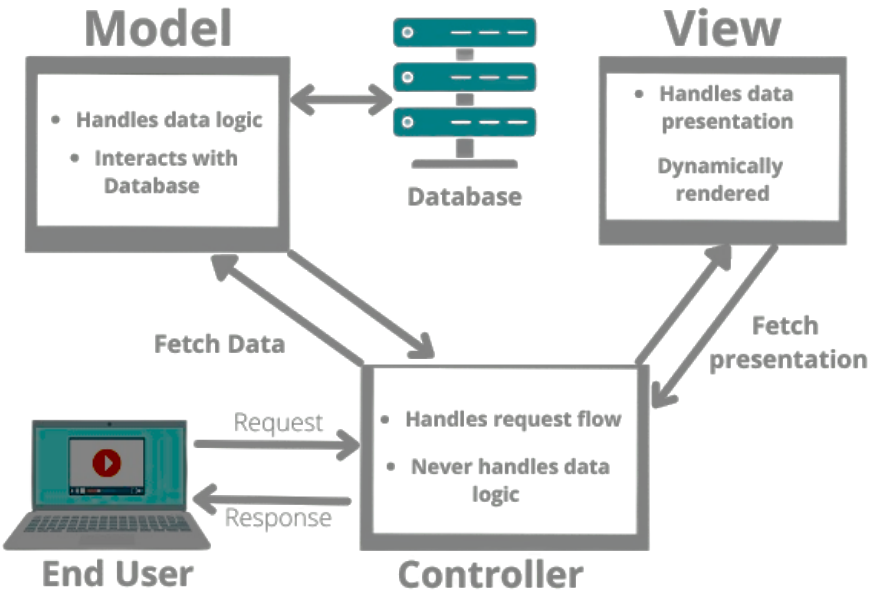


Figure 2.3: Logical architecture

2.3 Work Environment

In this part, we will talk about our work environment, focusing on different aspects: our material environment, the techniques we used in the realization of our project as well as the tools we used in our report, the product backlog and sprint planning, and finally, we will conclude this section [28, 29].

2.3.1 Physical environment

The work was carried out by a laptop computer that is equipped with these detailed features presented in Table 2.1 below:

Computer Name	MSI
Processor	i5 10th gen
Hard disk	512 Go SSD
RAM	24.0 Go
Operating system	Windows 11 Pro

Table 2.1: Physical environment

2.3.2 Used technologies

Expo

Expo is an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React.

TypeScript

TypeScript (abbreviated as TS) is a free and open-source high-level programming language developed by Microsoft that adds static typing with optional type annotations to JavaScript [30]. It is designed for the development of large applications and transpiles to JavaScript.

Tanstack

High-quality open-source software for web developers [31]. Headless, type-safe, & powerful utilities for Web Applications, Routing, State Management, Data Visualization, Datagrids/Tables, and more.

clerk

Clerk [18] is a complete suite of embeddable UIs, flexible APIs, and admin dashboards to authenticate and manage your users.

Maestro

Maestro [32] is the simplest, most powerful, and most trusted end-to-end testing platform for mobile and web apps.

Google cloud platform

Google cloud platform, or just GCP, is the cloud computing platform developed by Google. It has management, access and development of applications and services to individuals, companies, and governments through its global infrastructure.

GitHub

GitHub [9] is a cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

Express.js

Express.js [33] is a minimal and flexible Node.js [34] web application framework that provides a list of features for building web and mobile applications easily.

Postman

Postman [35] is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

Vite

Vite [36] is a modern build tool that provides a fast and optimized development experience for React 17 applications. It leverages native ES modules and offers a highly efficient development server with hot module replacement (HMR).

React

React [37], sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

MySQL

MySQL [38] is an open-source relational database management system. It is based on structured query language (SQL), which is used to add, access and manage content in a database.

Docker

Docker is an open platform for developing, shipping, and running applications [24]. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

Playright

Playwright [39] is an open-source testing and automation framework that can automate web browser interactions. To put it simply, you can write code that can open a browser [14].

Storybook

Storybook is a frontend workshop for building UI components and pages in isolation. It helps you develop and share hard-to-reach states and edge cases without needing to run your whole app [40].

StarUML

StarUML [41] is a sophisticated software modeler aimed to support agile and concise modeling. It provides eleven different types of diagrams and it accepts UML 2.x standards.

Node.js

Node.js [34] is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser, allowing developers to use JavaScript for server-side scripting.

2.3.3 Tools used for the report

Overleaf

Overleaf is a collaborative cloud-based LaTeX editor used to write, edit, and publish scientific papers.

Canva

Canva is a global company that empowers people to design anything and publish anywhere. Learn about its mission, values, commitments, awards, product, and careers.

2.3.4 Source code management with Git and GitHub

GitHub [9] was utilized for version control, employing a structured branching strategy for organized development. The ‘main‘ branch holds official release history, while ‘develop‘ serves as an integration branch for new features. Feature branches are created from ‘develop‘ for new tasks or bug fixes. Upon completion and testing, these are merged back into ‘develop‘ via pull requests, ensuring code review and quality control. Merging ‘develop‘ into ‘main‘ creates a new stable application version. This workflow is illustrated in Figure 2.4.

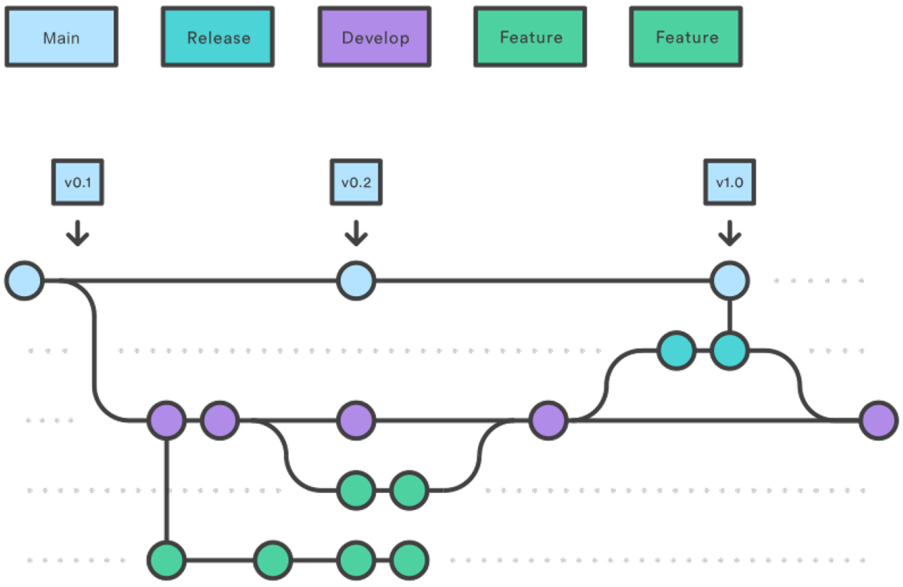


Figure 2.4: Git Workflow

2.4 Product backlog

The backlog was created before the sprints to plan the milestones and determine the content of each sprint based on project requirements [42]. It includes the following fields:

- **Code:** The unique identifier of the task.
- **Theme:** The subject of a user story.

- **User Story:** A short description of the functionality requested by the client.
- **Priority:** A value indicating the importance of the functionality [43, 44].
  - **Must:** The feature is essential and must be implemented.
  - **Should:** The feature should be implemented if possible.
  - **Could:** The feature is optional and may be deprioritized.

Table 2.2 shows the product backlog for our Korpor project:

Table 2.2: Korpor Product Backlog

Code	Theme	User story	Priority
Authentication & User Management			
PB001	Authentication	As a user, I want to create an account and authenticate securely	Must
PB002	User Management	As a user, I want to manage my profile	Must
PB003	Authentication	As a user, I want to securely reset my password	Must
PB004	Admin Management	As a Super Admin, I want to manage admin accounts for different real estate companies	Should
Super Admin Features			
PB005	Security	As a Super Admin, I want to monitor security and compliance across the platform	Could
PB006	Analytics	As a Super Admin, I want to generate and analyze global performance reports	Could
PB007	AI Integration	As a Super Admin, I want to chat with an AI assistant that can securely access database	Could
Admin Features			
PB008	Listing Management	As an Admin, I want to manage real estate listings in my company	Must
PB009	Agent Management	As an Admin, I want to oversee real estate agents and their permissions	Should
PB010	Transaction Management	As an Admin, I want to track transactions and calculate agent commissions	Should
PB011	Customer Service	As an Admin, I want to address investor inquiries and issues	Could
PB012	Analytics	As an Admin, I want to access a comprehensive agency dashboard	Could
PB013	AI Integration	As an Admin, I want to input property details and receive AI-powered valuation	Should
Real Estate Agent Features			

Continued on next page

PB014	Listing Management	As an Agent, I want to create and manage property listings	Must
PB015	Investment Management	As an Agent, I want to handle investment and purchase requests	Could
PB016	Data Management	As an Agent, I want to contribute data for AI-driven estimates	Could
PB017	Customer Relations	As an Agent, I want to maintain client relationships and communications	Could
PB018	Finance	As an Agent, I want to view my commissions on sales and rentals	Should
Investor Features			
PB019	Property Discovery	As an Investor, I want to browse available property listings	Must
PB020	Search Functionality	As an Investor, I want to filter properties based on my preferences	Could
PB021	Investment Process	As an Investor, I want to invest in properties through a simple process	Could
PB022	Portfolio Management	As an Investor, I want to track my investment portfolio in real-time	Must
PB023	Payment Processing	As an Investor, I want to make secure payments	Should
PB024	AI Recommendations	As an Investor, I want to receive personalized property recommendations	Must
PB025	AI Assistance	As an Investor, I want to consult an AI assistant for real estate legal questions	Could
PB026	Financial Prediction	As an Investor, I want to see predictions of potential earnings	Could
PB027	Finance Management	As an Investor, I want to manage my earnings and withdrawals	Could
PB028	Notifications	As an Investor, I want to receive push notifications about my investments	Should
AI & Machine Learning Features			
PB029	Recommendation System	As the System, I want to analyze user interactions for personalized recommendations	Must
PB030	Prediction System	As the System, I want to predict property valuations and rental prices	Should
PB031	Chatbot	As the System, I want to provide real estate legal information via NLP	Should
PB032	administrative chatbot	As the System, I want to provide real time data from database	Could
Blockchain: Smart Contract Features			

Continued on next page

PB033	Blockchain	As an Investor, I want my property investments to be secured via blockchain [21]	Must
PB034	Blockchain Management	As an Admin, I want to verify and validate blockchain transactions	Should
PB035	Data Integrity	As the System, I want to store transaction records immutably on blockchain	Must
PB036	System Monitoring	As a Super Admin, I want to monitor blockchain health and performance	Should

2.5 Sprint planning

In order to complete the project within the deadlines set by the internship agreement, planning is an important step in the process [45, 46]. It was therefore necessary to define the essential steps and estimate the time to be devoted to the completion of the various tasks. To do this, we made a GANTT chart.

In our project management, we opted for the proportional distribution method in order to estimate the costs [47, 48]. Figure 2.5 shows the Gantt chart that describes the progress of our project:

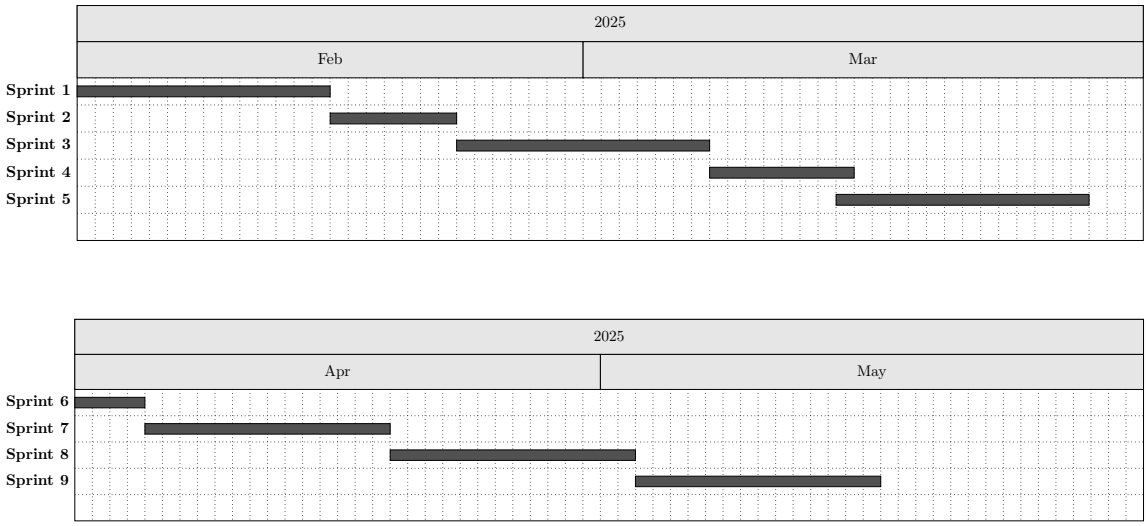


Figure 2.5: GANTT chart with sprint planning (February - May 2025)

Conclusion

Our Sprint 0 marked the exciting start of our KORPOR project [49, 45]. We defined global and specific objectives, developed a solid architecture, and configured an optimal working environment. With a clear vision of the initial product backlog and preliminary planning for upcoming sprints, we are ready to achieve our vision and achieve our goals successfully [42, 46].

CHAPTER 3

AUTHENTICATION & USER MANAGEMENT

Introduction

This chapter presents the authentication and user management system, which forms the security foundation of the Korpor platform. The implementation covers user registration, role-based access control, and comprehensive authentication workflows for both web and mobile interfaces.

3.1 Sprint 1: Authentication and User Management

Introduction

The first sprint focuses on establishing the core authentication system and user management functionality. This foundation is critical for all subsequent features as it defines user roles and access controls.

3.1.1 Analysis

The Authentication and User Management system forms the backbone of Korpor's security and user interaction model. It encompasses processes for user registration with role assignment, secure login with session and token management, password recovery mechanisms, and session termination. Figure 3.1 illustrates the global use cases for these core functionalities.

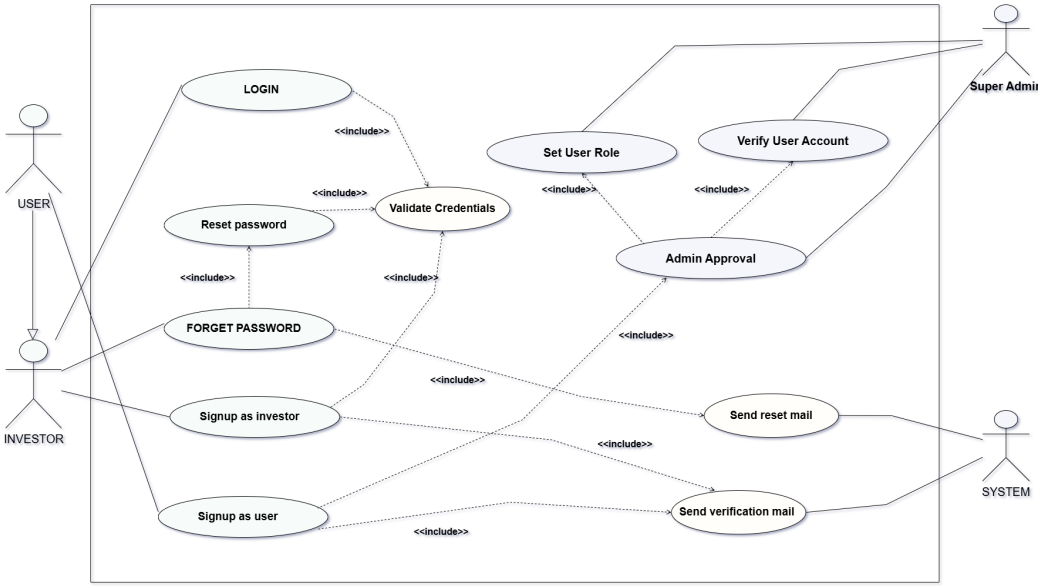


Figure 3.1: Global Use Case Diagram for Authentication and User Management

Sign-up Process

The sign-up process is illustrated in Figure 3.2 below. The diagram shows the authentication flow for new users registering in the system.

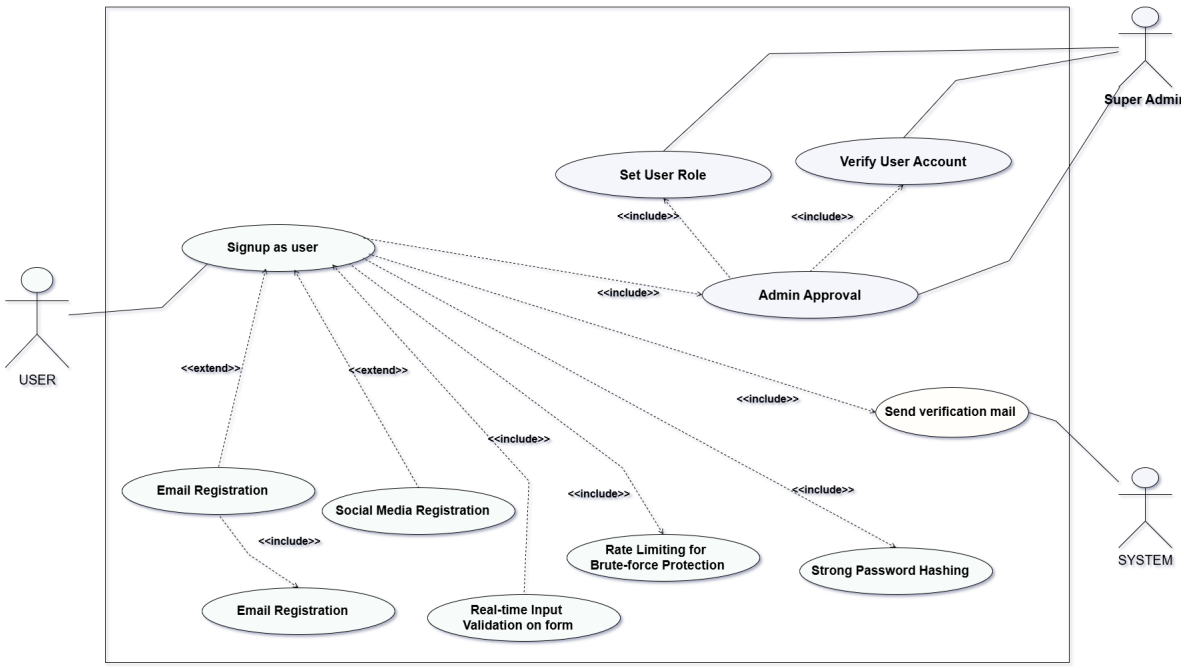


Figure 3.2: Authentication Sign-up Use Case Diagram

Figure 3.3 illustrates the activity flow of the sign-up process. The process begins when a new user navigates to the sign-up page and presses the sign-up button. The system then presents a form for the user to fill out. Upon submission, the system validates the entered information.

If the information is invalid, the user is prompted to correct the form. If the information is valid, the system saves the user's details in the database and sends a verification email to the user. The user must then check their email.

Upon successful email verification, the system redirects the user to an "email verification code" page or a similar confirmation step. The Super Admin is then involved in a two-step verification process. If the Super Admin accepts the user, their login is enabled. If the Super Admin refuses the user, the user's account is deleted. If the initial email verification step fails (e.g., there's an issue with the verification code), an error message is displayed to the user.

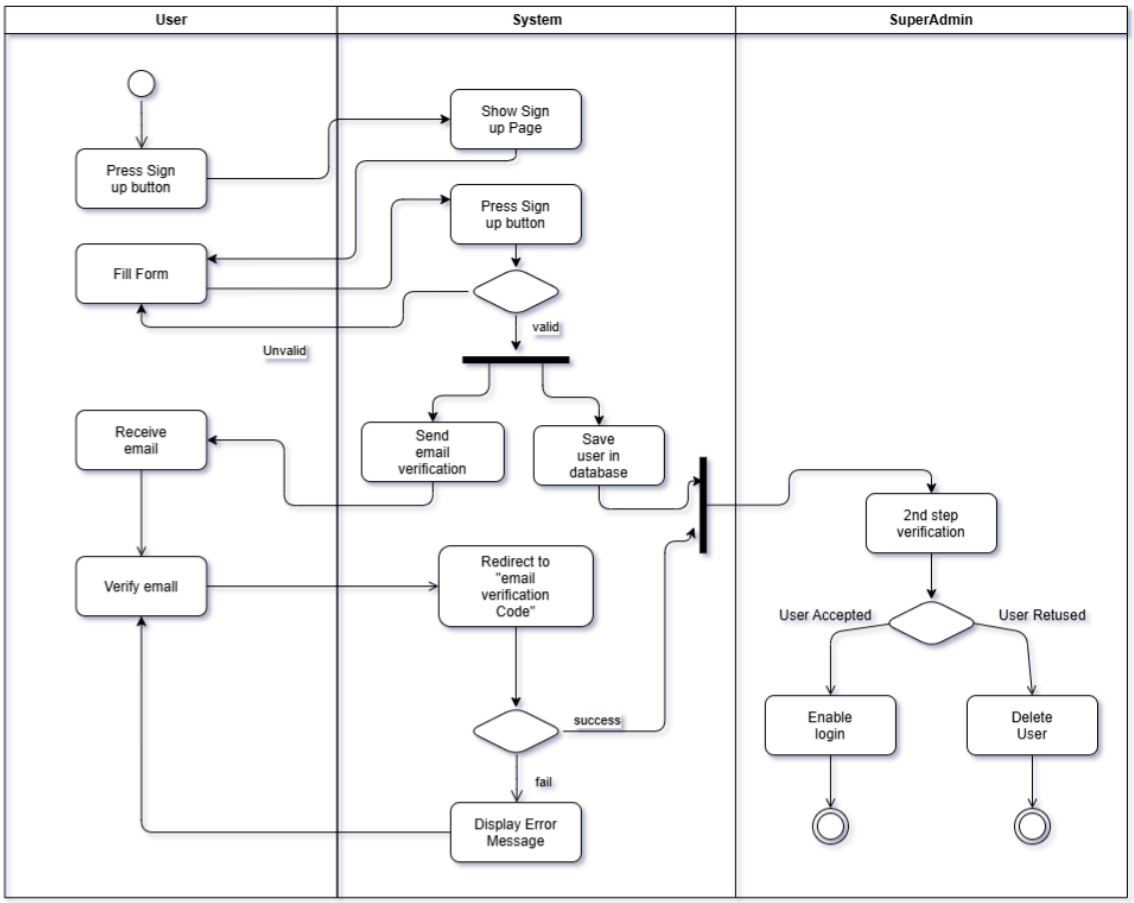


Figure 3.3: Authentication Sign-up Activity Diagram

Login Process

The login process allows existing users to access the system. The table below details the use case, as shown in Table 3.1.

Section	Details
Use Case	User Login
Actor	User (Super Admin, Admin, Agent, Investor)
Precondition	User has an existing, verified account. User is on the login page.
Main Scenario	User submits their valid credentials.
Postcondition	User is successfully logged into the system and can access features based on their role.
Exception	- Invalid credentials: System displays an error message. - Account locked/disabled: System displays an appropriate message. - System error: System displays a general error message.

Table 3.1: Login Process Details

Manage Users Process

The user management process enables administrators to create, view, update, and delete user accounts. The table below details the use case, as shown in Table 3.2.

Section	Details
Use Case	Assign Role
Actor	Super Admin
Precondition	Actor is logged into the system with appropriate administrative privileges.
Main Scenario	Super Admin assigns a role to a user.
Postcondition	User account is created, updated, or deleted as per the action taken. The list of users reflects the changes.
Exception	- Invalid input data: System displays validation errors. - Permission denied: System prevents unauthorized actions. - User not found (for update/delete): System displays an error message. - System error: System displays a general error message.

Table 3.2: Manage Users Process Details

Forget Password Process

The password recovery process enables users to securely reset their password when they cannot access their account. The table below details the use case, as shown in Table 3.3.

Section	Details
Use Case	Forget Password
Actor	User (Super Admin, Admin, Agent, Investor)
Precondition	User has an existing account with a valid email address
Main Scenario	User requests password reset and receives secure reset instructions via email
Postcondition	User successfully resets password and can access their account with new credentials
Exception	- Email not found: System displays error message. - Network error: System displays connectivity error message.

Table 3.3: Forget Password Process Details

The sign-up process includes user registration, role assignment, and account verification steps. During registration, users are categorized into one of the three user types: Super Admin, Admin, or Agent, with each type having different permissions and access levels within the system.

3.1.2 Modeling

Figure 3.4 depicts the class diagram for the authentication system. It showcases the key components and their relationships involved in user authentication and authorization.

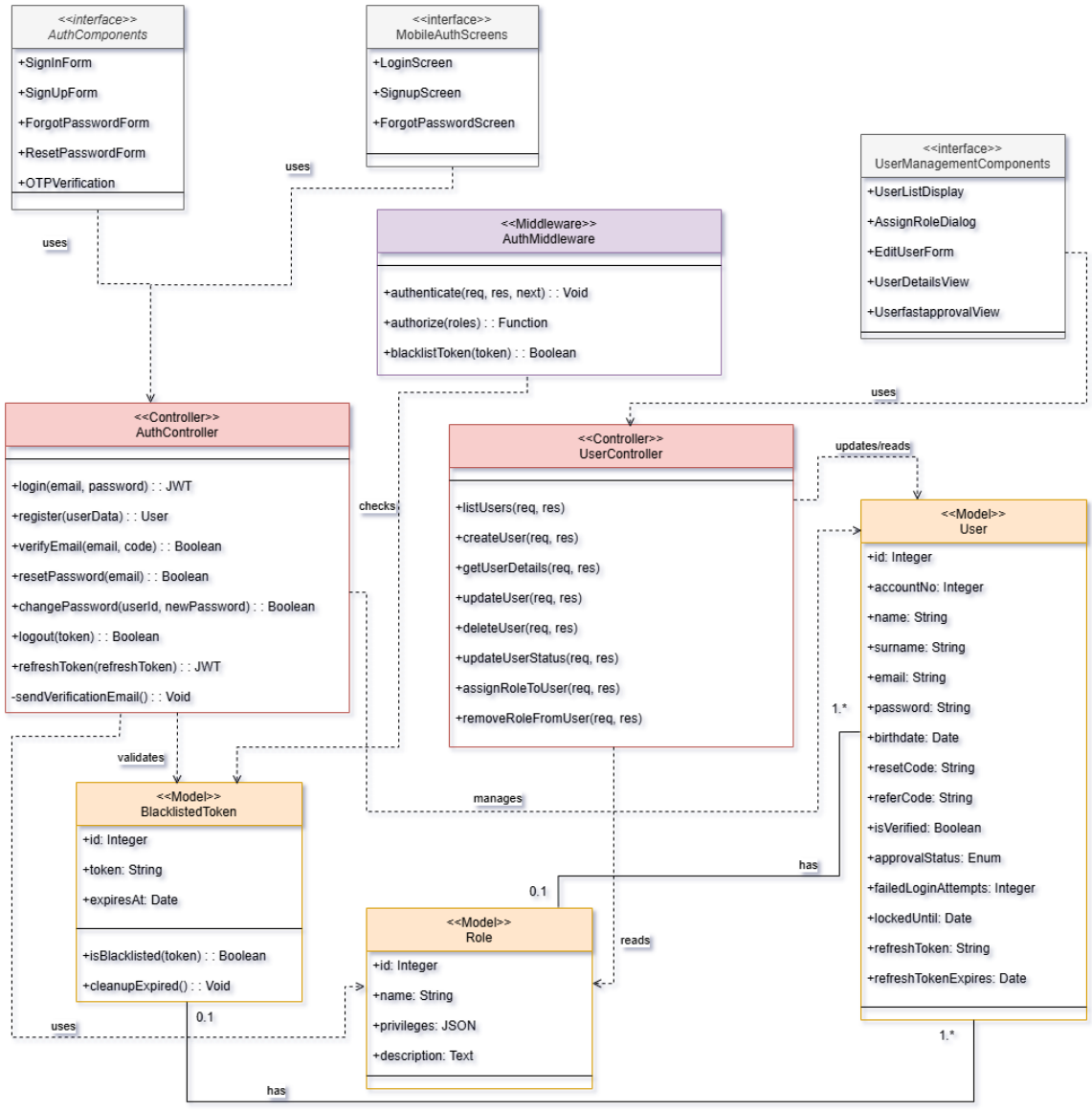


Figure 3.4: Authentication AND User Management System Class Diagram

- **AuthComponents**: Represents the UI elements for authentication, such as SignInForm, SignUpForm, ForgotPasswordForm, ResetPasswordForm, and OTPVerification. These components are used by **MobileAuthScreens**.
- **MobileAuthScreens**: Includes screens like LoginScreen, SignupScreen, and ForgotPasswordScreen that utilize **AuthComponents** and interact with the **AuthService**.
- **AuthService**: Acts as an intermediary between the frontend components/screens and the backend. It handles functions like signIn, signUp, verifyEmail, forgotPassword, resetPassword, logout, refreshToken, and error handling. It consumes **AuthRoutes**.
- **AuthRoutes**: Defines the API endpoints for authentication, such as /login, /register, /verify-email, /forgot-password, /reset-password, /logout, and /refresh-token. These routes map to methods in the **AuthController**.
- **AuthController**: Contains the core logic for authentication processes, including login, register, verifyEmail, resetPassword, changePassword, logout, refreshToken, and sendVerificationEmail. It interacts with the **User**, **Role**, and **BlacklistedToken** models and utilizes **AuthMiddleware**.
- **AuthMiddleware**: Provides middleware functions for authentication (authenticate) and authorization (authorize roles). It also manages blacklisted tokens (blacklistToken) and interacts with the **User** model.
- **User**: Represents the user entity with attributes like id, accountNo, name, surname, email, password, birthdate, resetCode, isVerified, approvalStatus, failedLoginAttempts, lockedUntil, refreshToken, and refreshTokenExpires. A User has one or more **Role**s.
- **Role**: Defines user roles with attributes like id, name, privileges (JSON), and description. Each User is associated with a Role (0..1 relationship shown, typically a User has at least one Role, but the diagram indicates a User can have zero or one Role, which might need clarification or represent a specific system design choice, e.g., a default role or a user awaiting role assignment).
- **BlacklistedToken**: Stores tokens that have been invalidated (e.g., after logout or password change) with attributes like id, token, and expiresAt. It includes methods like isBlacklisted and cleanupExpired. The **AuthController** uses this to validate tokens, and **AuthMiddleware** checks against it.

3.1.3 Implementation

The implementation of the authentication and user management system resulted in the following key user interfaces:

Sign-in Interface

The sign-in interface provides a secure and user-friendly means for users to authenticate. Figure 3.5 shows the implementation of this interface.

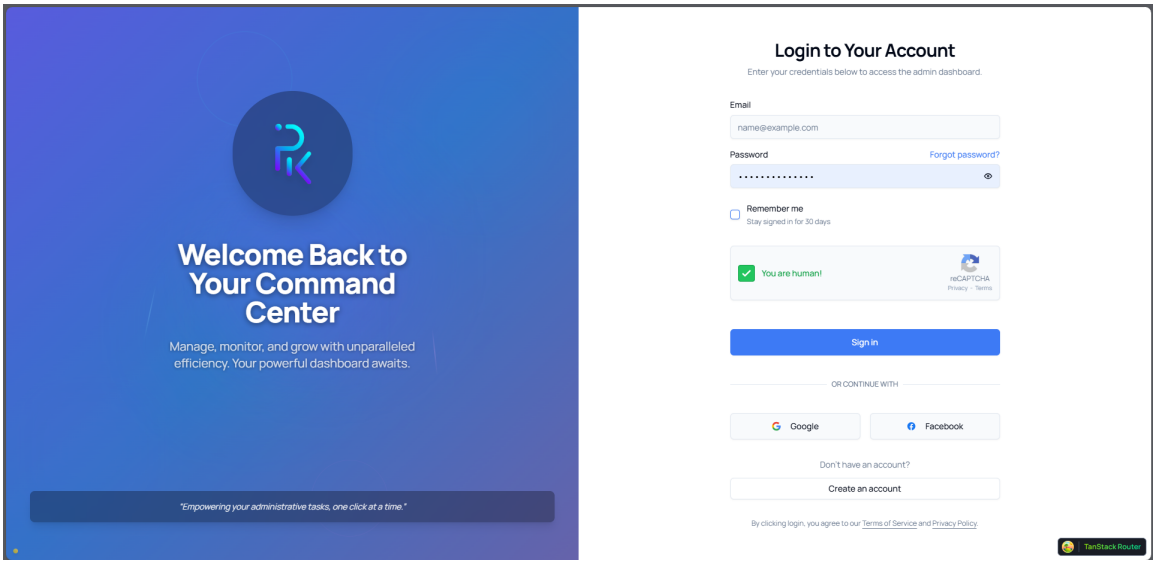


Figure 3.5: Sign-in Interface Implementation

Sign-up Interface

The sign-up interface allows new users to register for an account. It collects necessary information and begins the verification process. Figure 3.6 displays this implementation.

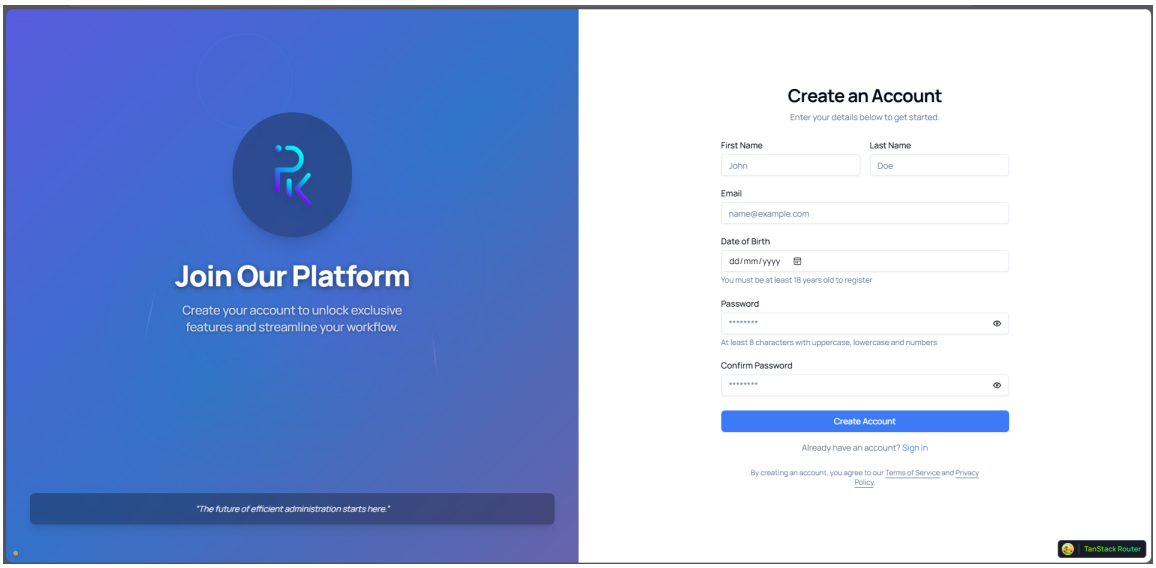


Figure 3.6: Sign-up Interface Implementation

User Management in Super Admin Dashboard

The Super Admin dashboard provides comprehensive user management capabilities, including the ability to view, create, edit, and deactivate user accounts. Figure 3.7 shows this powerful interface.

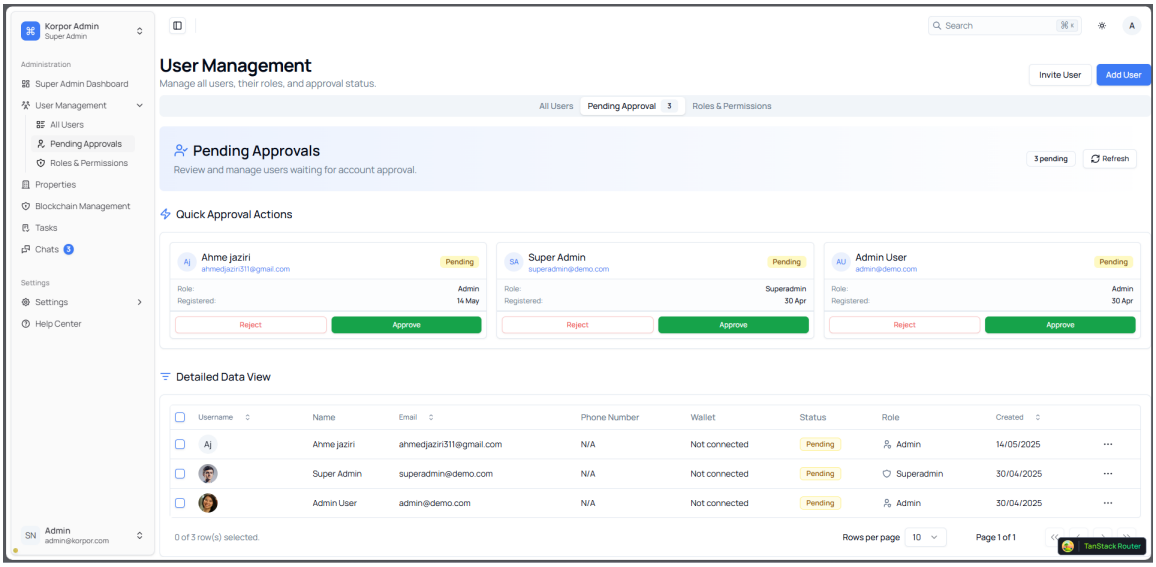
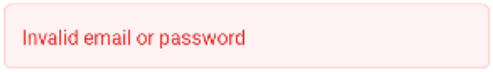
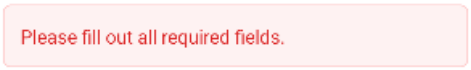


Figure 3.7: User Management Interface in Super Admin Dashboard

The mobile application provides a streamlined authentication experience for investors. The system includes comprehensive error handling to guide users through common authentication issues, as shown in Figure 3.8. The complete authentication flow includes login, signup, OTP verification, and password recovery interfaces designed for clarity and ease of use on mobile devices, as shown in Figure 3.9.

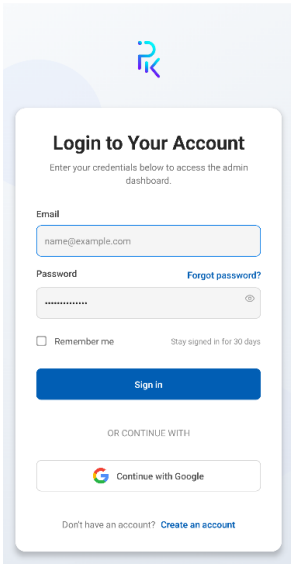


(a) Invalid Email or Password Error

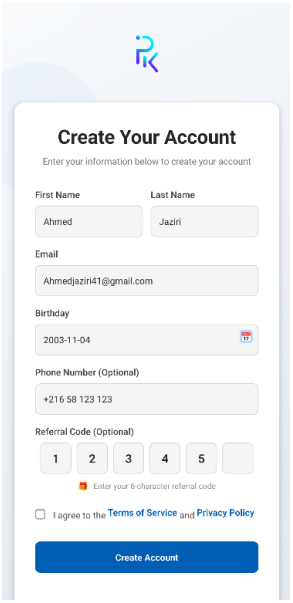


(b) Empty Fields Validation Error

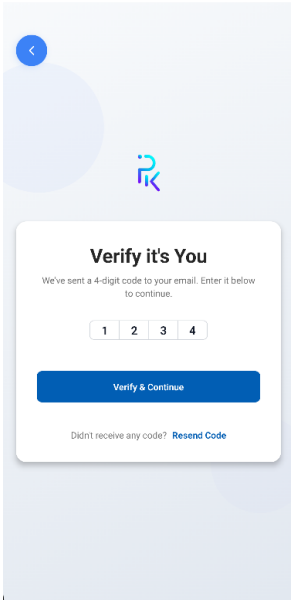
Figure 3.8: Authentication Validation Messages



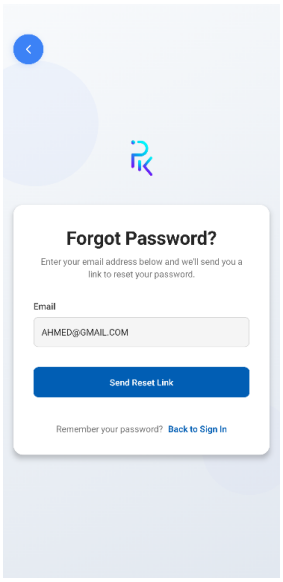
(a) Login Screen



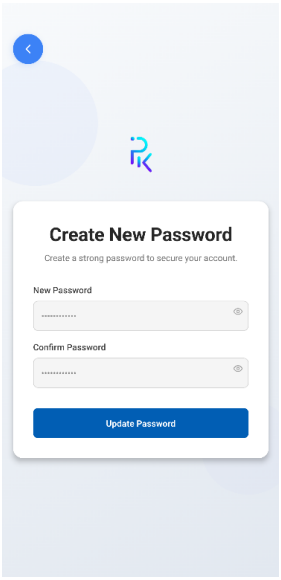
(b) Signup Screen



(c) OTP Screen



(d) Forgot Password Screen



(e) New Password Screen

Figure 3.9: Complete Mobile Application Authentication Flow

3.1.4 Test

To ensure the reliability and functionality of the authentication and user management system, we implemented comprehensive testing using Playwright, an end-to-end testing framework [39].

Playwright Test Results

The authentication system underwent rigorous testing through automated test scripts that verified all key functionality, including sign-up, sign-in, password reset, and user management operations. Figure 3.10 demonstrates the successful execution of these tests.

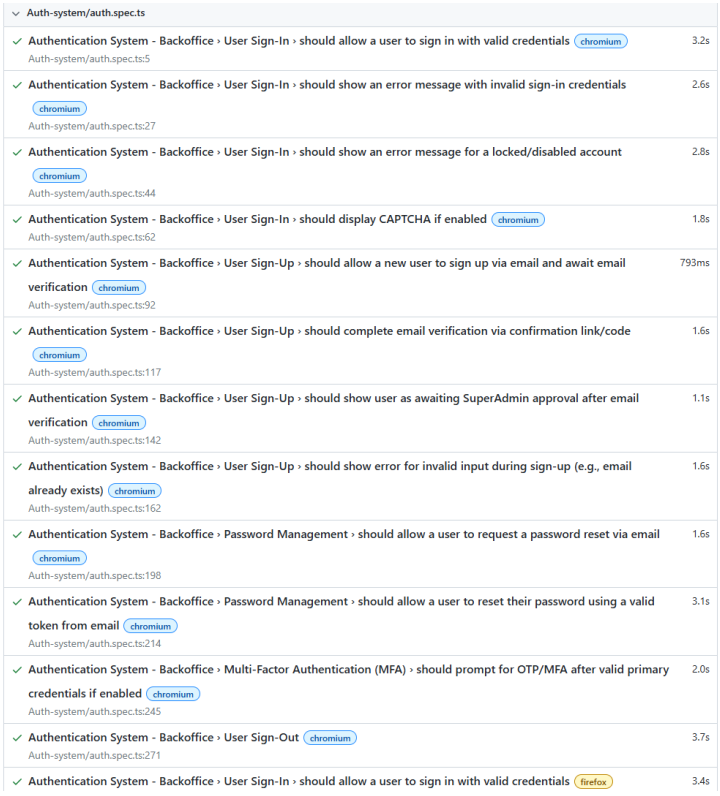


Figure 3.10: Playwright Test Results for Authentication System

All tests passed successfully, confirming the robustness of the implemented authentication system.

3.1.5 Retrospective

The authentication and user management sprint provided valuable insights into system development and team collaboration. Table 3.4 summarizes the key achievements, challenges, and lessons learned during this sprint.

Category	Details
What Went Well	• Successfully implemented secure authentication flow • Role-based access control working as expected • Mobile interface provides excellent user experience • All planned test scenarios passed • Strong team collaboration throughout the sprint
Challenges	• Initial complexity in setting up JWT token management • Integration testing required more time than estimated • Email verification system configuration challenges • Balancing security requirements with user experience

Table 3.4: Authentication Sprint Retrospective Summary

Conclusion

The Foundation sprint successfully established the secure authentication and user management infrastructure for the Korpor platform.

CHAPTER 4

Artificial Intelligence Features

Artificial intelligence is the new electricity.

— Andrew Ng

Introduction

This chapter explores the integration of artificial intelligence capabilities within our real estate platform. We have developed four distinct AI models, each addressing specific requirements within the ecosystem. These models collectively enhance user experience, improve decision-making processes, and provide valuable insights to various stakeholders in the real estate market.

The AI features presented in this chapter represent a significant competitive advantage for our platform, enabling more accurate property valuations, personalized recommendations, intelligent assistance, and efficient administrative operations. Each model has been carefully designed to solve real-world challenges faced by users interacting with real estate data and transactions.

4.1 Sprint 2: Data Collection and Scraping

Introduction

This section details the data collection processes implemented to gather the real estate market data required for our AI models.

4.1.1 Real Estate Data Scraping

Data Sources

We identified several real estate websites with significant listings for the Tunisian market: Properstar, Remax, Home in Tunisia, and Mubawab. These platforms offered a reasonable volume of property listings with the attributes needed for our models. For each website, I developed a dedicated Python script that navigated through the listings, extracted the relevant property details, and stored the information in CSV files. This approach gave us a foundation of raw data that could later be processed and used for training our prediction models.

Figure 4.1: properstar website

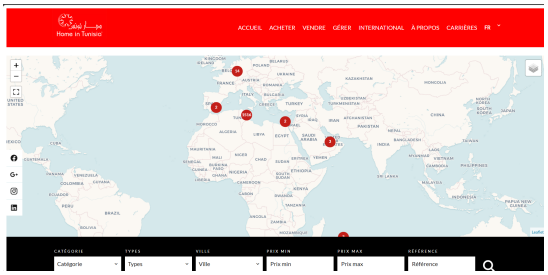


Figure 4.2: homeintunisia website

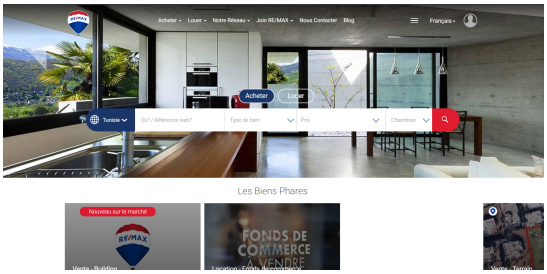


Figure 4.3: remax website

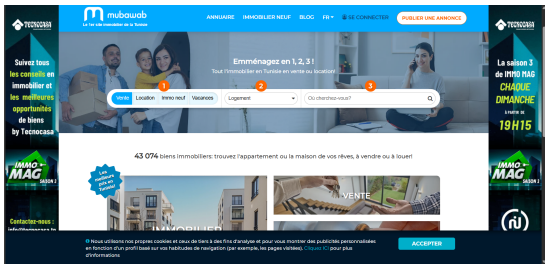


Figure 4.4: mubawab website

Our scraping system uses a distributed architecture with the following components:

- Scheduled jobs for regular data updates, Data validation and cleaning pipelines

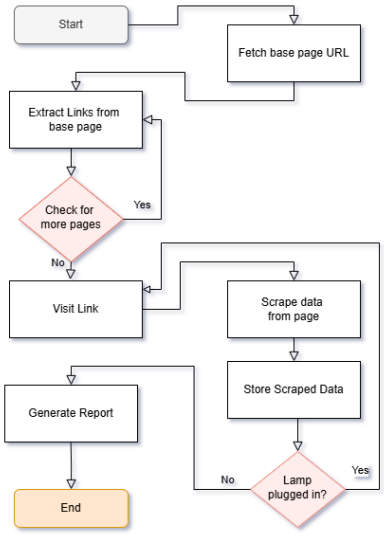
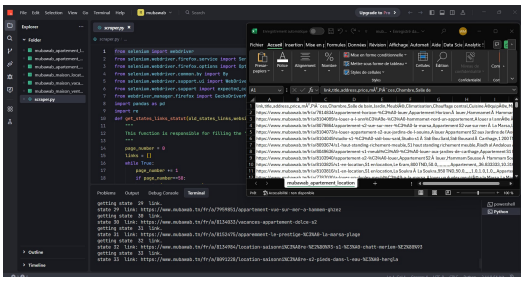


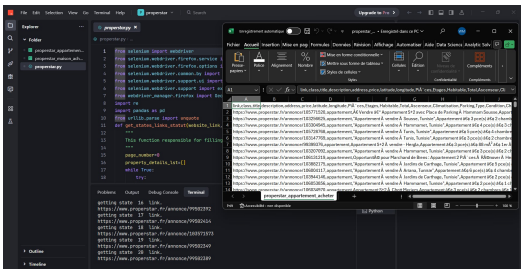
Figure 4.5: Data Scraping workchart

4.1.2 Implementation

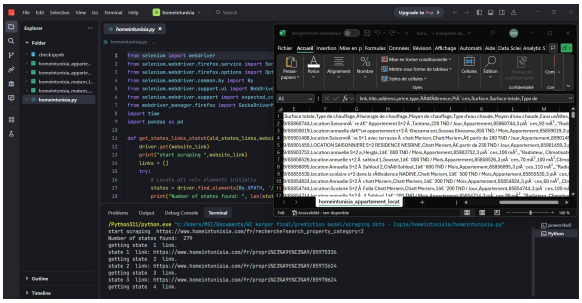
The data scraping system successfully collected comprehensive property information from multiple real estate platforms. Figure 4.6 demonstrates the structured output format of the scraped data, showing how property details are organized and stored for further processing by our AI models.



Mubwab Scraper



Properstar Scraper



Home in Tunisia Scraper

Figure 4.6: Data Scraping Output Results

4.2 Sprint 3: Property Valuation Prediction Model

Introduction

The Property Valuation Prediction Model is designed to estimate both the market value and potential rental income for real estate properties. This provides investors with crucial information to make informed investment decisions.

4.2.1 Analysis

Use Case Diagram

The property valuation model serves multiple actors within the Korpor ecosystem. Figure 4.7 illustrates the primary use cases for the AI-powered property valuation system.

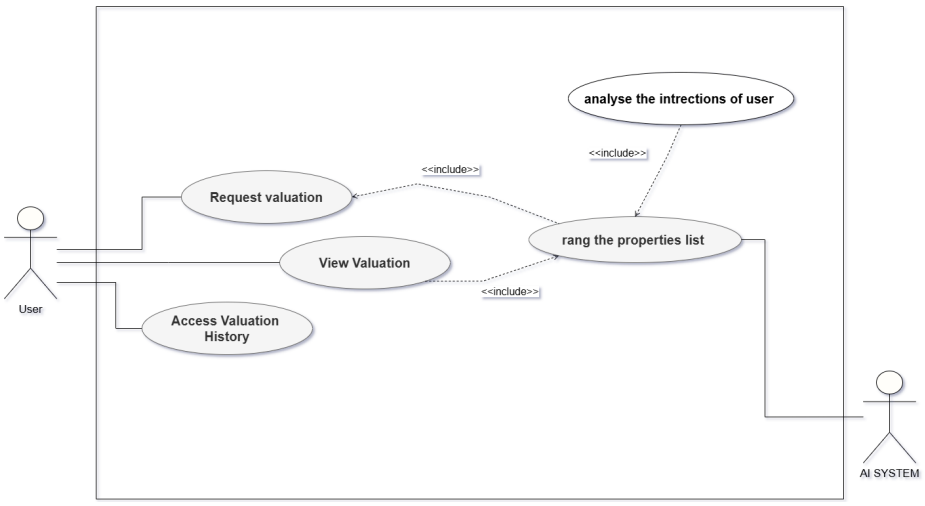


Figure 4.7: Property Valuation Model Use Case Diagram

Textual Use Case Descriptions

Use Case	**Request Property Valuation**
Actor	User (representing Super Admin, Admin, Real Estate Agent)
Precondition	User is authenticated and has access to valuation features
Main Scenario	User gets market value and rental income predictions for a property
Postcondition	Valuation is stored and available for future reference

Table 4.1: Property Valuation Use Case Description

4.2.2 Modeling

Class Diagram

The property valuation system follows object-oriented design principles. Figure 4.8 shows the main classes and their relationships.

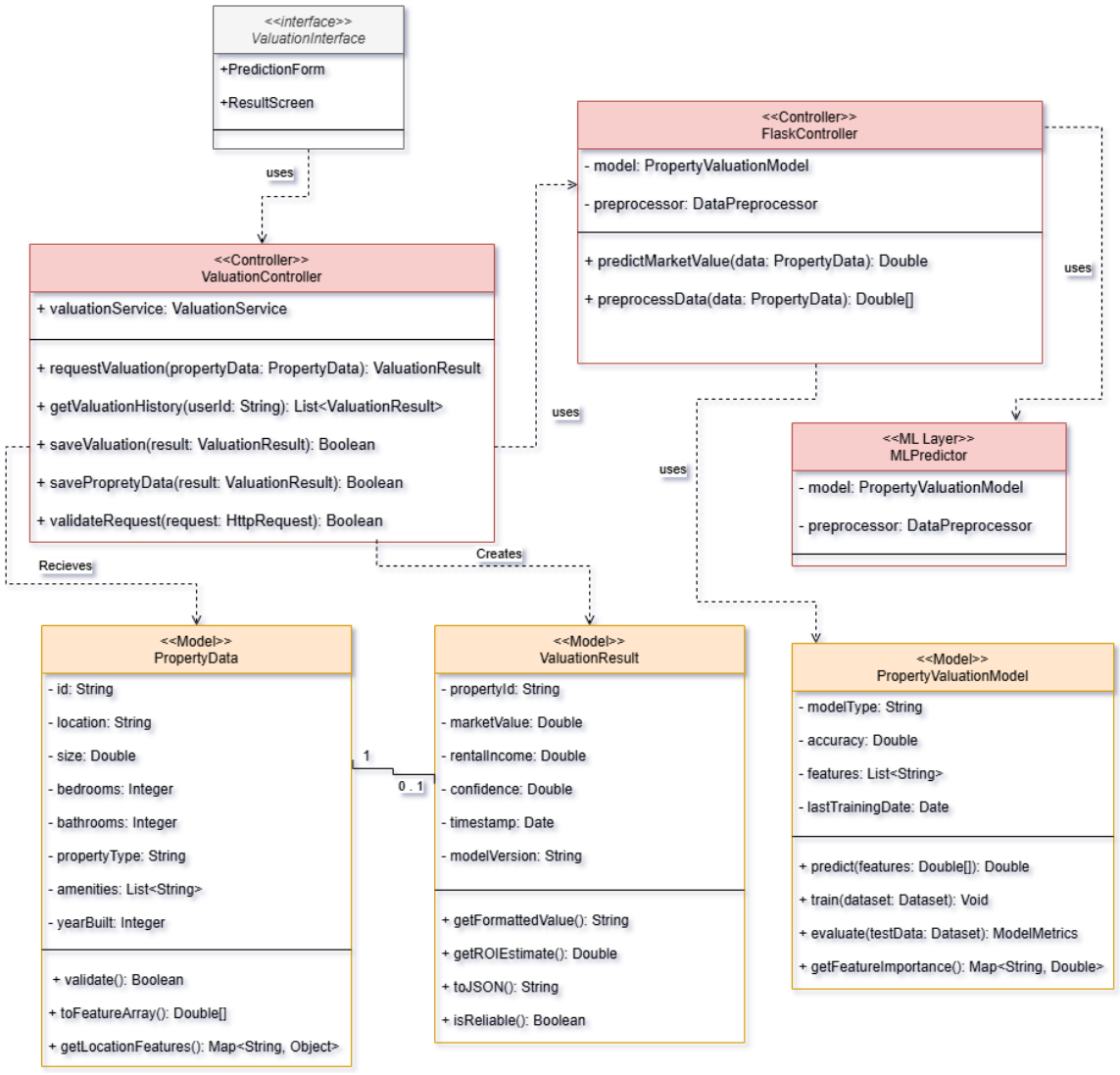


Figure 4.8: Property Valuation Model Class Diagram

The interaction sequence for the AI property valuation prediction model demonstrates the flow between different system components. Figure 4.9 illustrates the sequence of operations from user request to prediction result delivery.

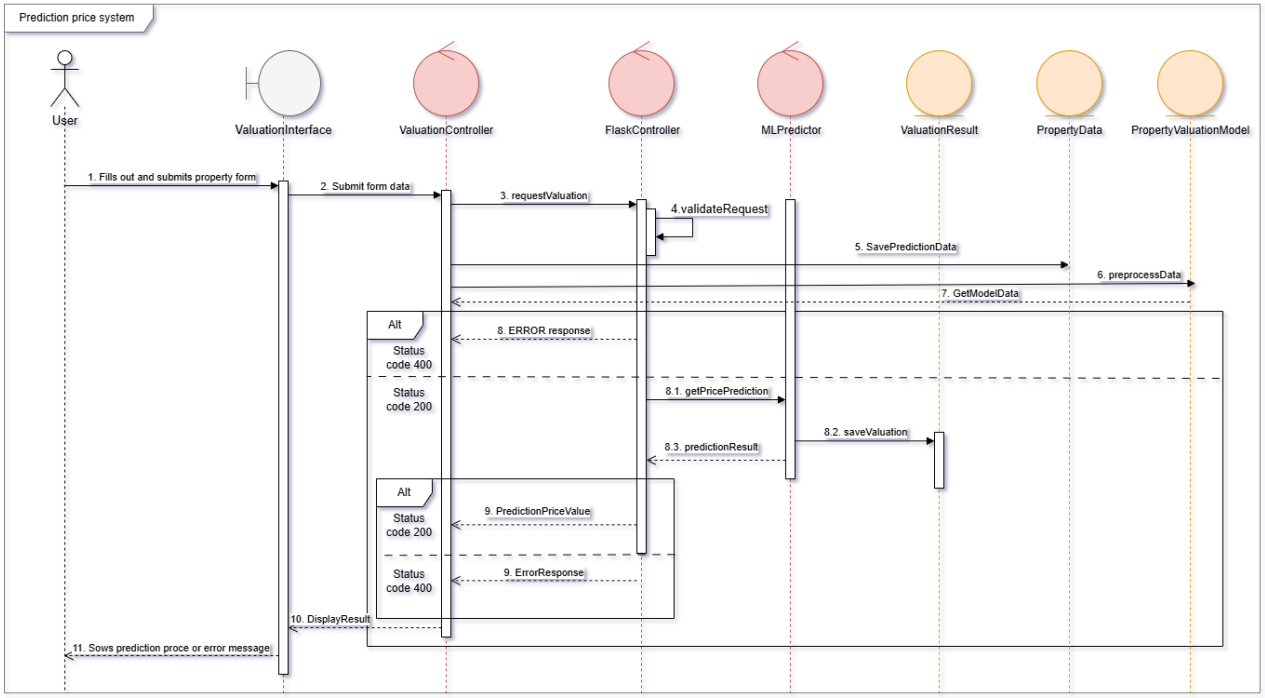


Figure 4.9: AI Property Valuation Prediction Sequence Diagram

To provide a clearer understanding of the property valuation process, Figure 4.10 presents a simplified workflow diagram that illustrates the step-by-step flow from user input to prediction results.

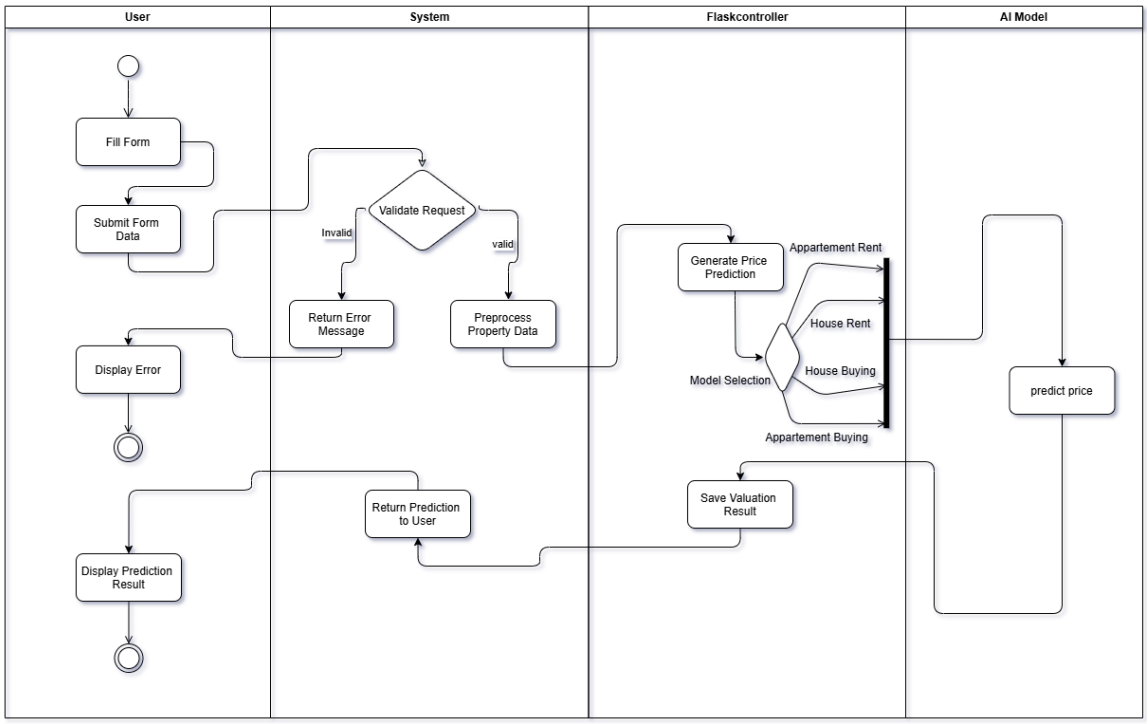


Figure 4.10: Property Valuation Prediction Workflow Diagram

4.2.3 Implementation

Data Features for Valuation

The accuracy of the property valuation model heavily relies on the quality and comprehensiveness of the input data. Figure 4.11 outlines the key geo-property data features utilized by the model. These features capture essential characteristics of a property and its location, enabling the model to learn complex relationships and predict market values and rental incomes effectively.

Figure 4.11: Geo-propriety Data Features for AI Models

Model Selection

To develop an accurate property valuation prediction model, several regression algorithms were evaluated. The following models were selected for training and comparison due to their distinct characteristics and common effectiveness in similar predictive tasks:

- **Linear Regression**: Chosen as a baseline model due to its simplicity and interpretability. It helps in understanding the linear relationships between the features and the target variables (market value and rental income).
- **Random Forest Regressor**: An ensemble learning method that operates by constructing a multitude of decision trees at training time. It is robust to overfitting, handles non-linear relationships well, and often provides high accuracy.
- **Gradient Boosting Regressor**: Another powerful ensemble technique that builds models in a stage-wise fashion. It is known for its high predictive accuracy and ability to optimize for various loss functions, making it suitable for complex regression tasks.

These models were trained on the prepared dataset, and their performances were evaluated to select the most suitable one for deployment in the Korpor platform.

Feature Importance Analysis

Understanding which features contribute most to the model's predictions is crucial for model interpretability and refinement. After training the selected model, a feature importance analysis was conducted. Figure 4.12 displays the top 15 most important features identified by the model. This analysis helps in validating the model's logic.

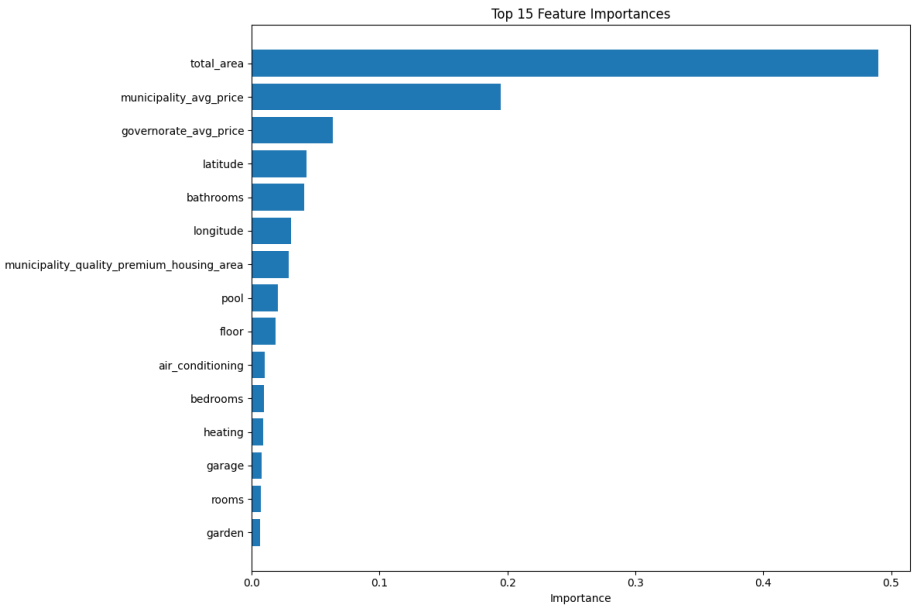
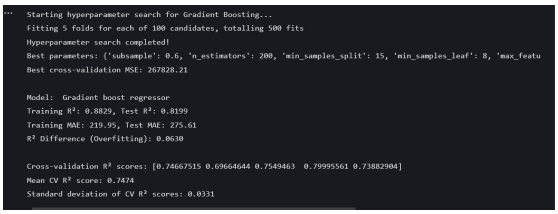


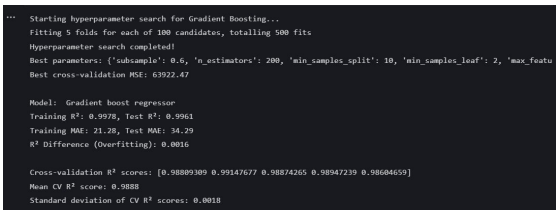
Figure 4.12: Top 15 Feature Importance for Property Valuation Model

Model Evaluation and Metrics

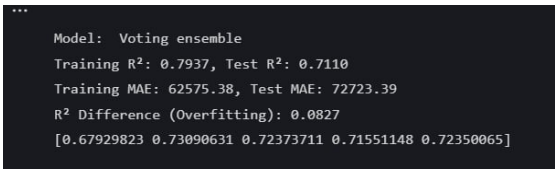
The performance of the trained regression models was rigorously evaluated using standard metrics to ensure reliability and accuracy. Key metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score were computed on a held-out test dataset. Figure 4.13 presents a summary of these performance metrics for the chosen valuation model. These results provide a quantitative assessment of the model's ability to generalize to unseen data.



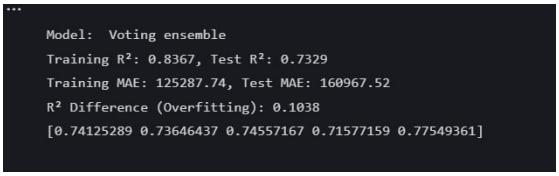
Apartment Location Prediction



House Location Prediction



Apartment Sale Prediction



House Sale Prediction

Figure 4.13: Prediction Model Test Metrics Summary

Prediction User Interface

Figure 4.14 shows the input form, and Figure 4.15 displays an example of the prediction results screen.

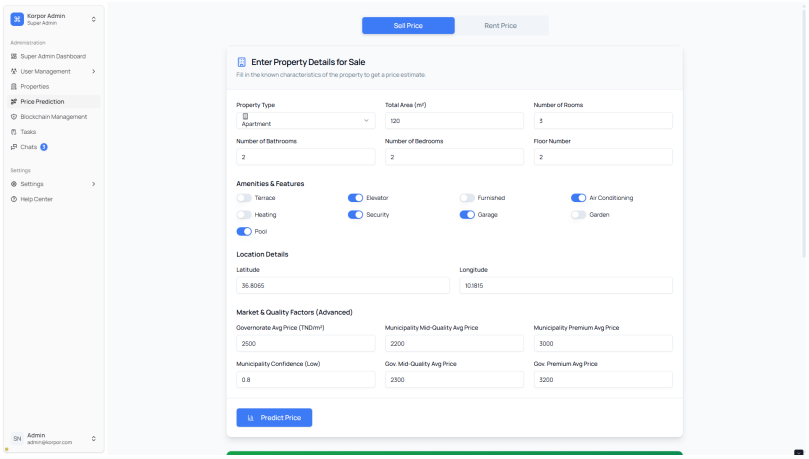


Figure 4.14: Property Details Input Form for Valuation

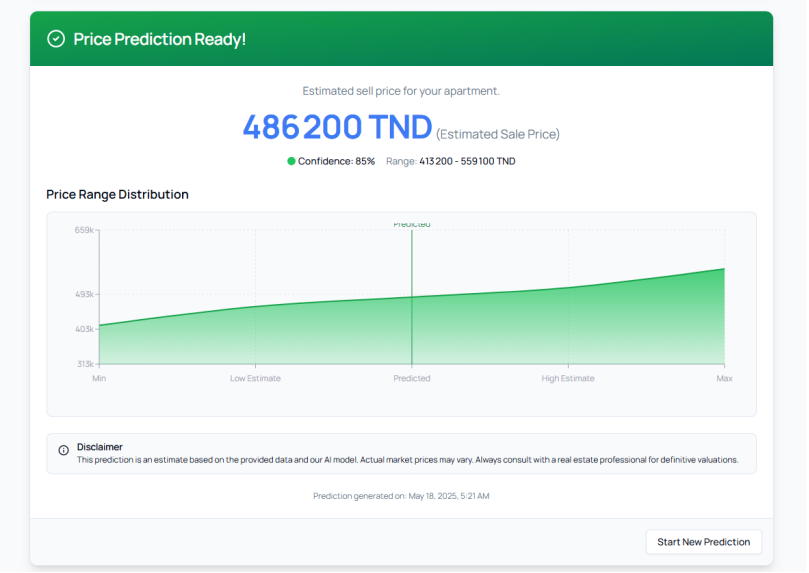


Figure 4.15: Valuation Prediction Results Screen

Mobile Investment Insights

The Korpor mobile application provides investors with direct access to AI-powered property valuations, including future evaluation insights generated by the prediction model. This feature empowers users to make data-driven investment decisions by visualizing potential growth and returns. Figure 4.16 showcases the mobile interface where these future evaluations are presented to the investor.

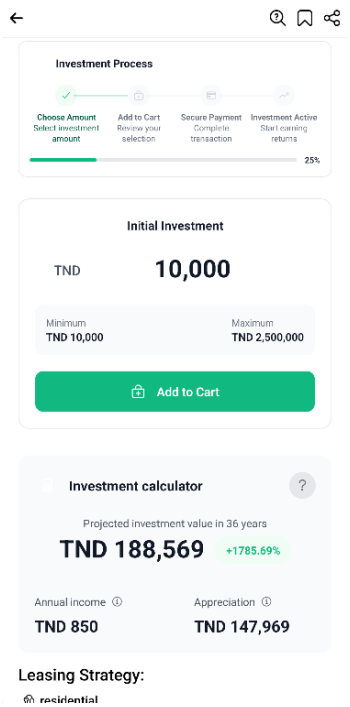


Figure 4.16: Mobile Interface for Future Property Evaluation Insights

4.2.4 Test

Mobile Interface Testing (Maestro)

To ensure a seamless and reliable user experience on the mobile platform, the interface for displaying AI-driven property evaluations underwent end-to-end testing using Maestro. The tests covered user flows for accessing predictions and interacting with the displayed data. Figure 4.17 shows the successful completion of these Maestro tests, validating the robustness of the mobile UI components related to the AI features.

Figure 4.17: Maestro Test Results for Mobile Prediction Interface

Model Performance Validation

The property valuation model underwent comprehensive testing to ensure accuracy and reliability. The model achieved satisfactory performance metrics across different property types and locations, validating its effectiveness for real-world deployment.

4.2.5 Retrospective

The property valuation prediction model sprint provided valuable insights into AI model development and deployment. Table 4.2 summarizes the key achievements, challenges, and lessons learned during this sprint.

Category	Details
What Went Well	• Successfully implemented multiple regression algorithms • Achieved good model performance metrics • Feature importance analysis provided valuable insights • Mobile interface integration worked seamlessly • Comprehensive testing validated model reliability
Challenges	• Data preprocessing required more time than anticipated • Model hyperparameter tuning was computationally intensive • Handling missing or incomplete property data • Balancing model complexity with interpretability

Table 4.2: Property Valuation Model Sprint Retrospective Summary

4.3 Sprint 4: Real Estate Assistant (NLP Chatbot)

Introduction

The Real Estate Assistant is an intelligent NLP-powered chatbot designed to provide investors with instant access to real estate legal information and guidance. This AI assistant helps users understand complex legal concepts, property regulations, and investment procedures through natural language conversations.

4.3.1 Analysis

Use Case Diagram

The Real Estate Assistant serves primarily investors who need legal guidance during their property investment journey. Figure 4.18 illustrates the main use cases for the AI assistant.

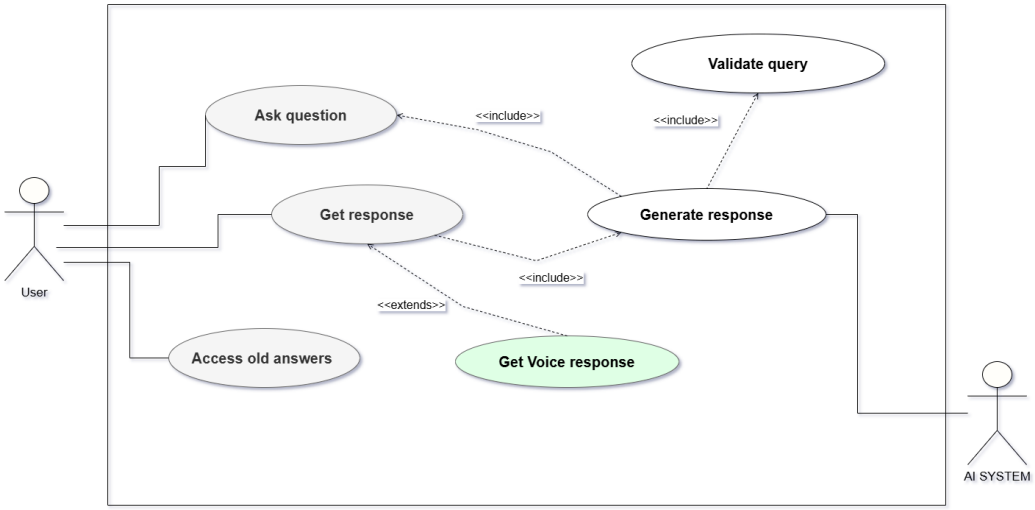


Figure 4.18: Real Estate Assistant Use Case Diagram

Textual Use Case Descriptions

Use Case	**Ask Legal Question**
Actor	Investor
Precondition	User is authenticated and has access to the mobile app
Main Scenario	User get comprehensive answer
Postcondition	Conversation is saved for future reference

Table 4.3: Real Estate Assistant Use Case Description

4.3.2 Modeling

Class Diagram

The Real Estate Assistant follows a modular architecture for NLP processing and knowledge management. Figure 4.19 shows the main classes and their relationships.

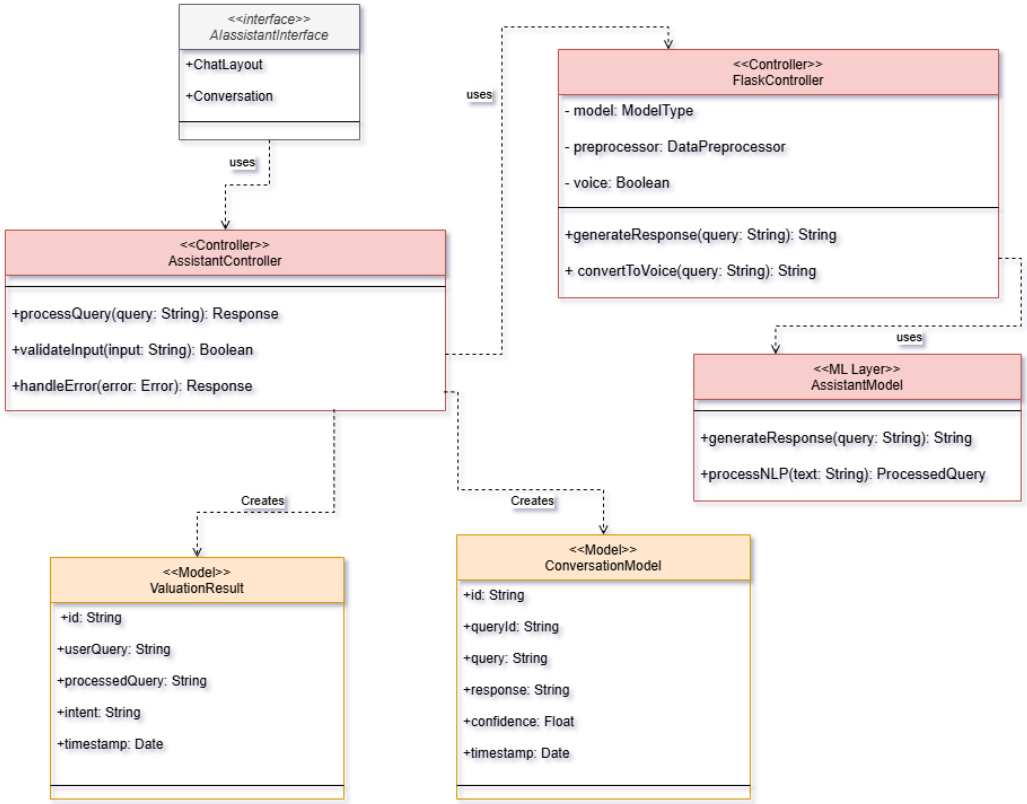


Figure 4.19: Real Estate Assistant Class Diagram

Sequence Diagram (MVC)

The interaction flow between the mobile interface, backend services, and NLP processing components is illustrated in Figure 4.20.

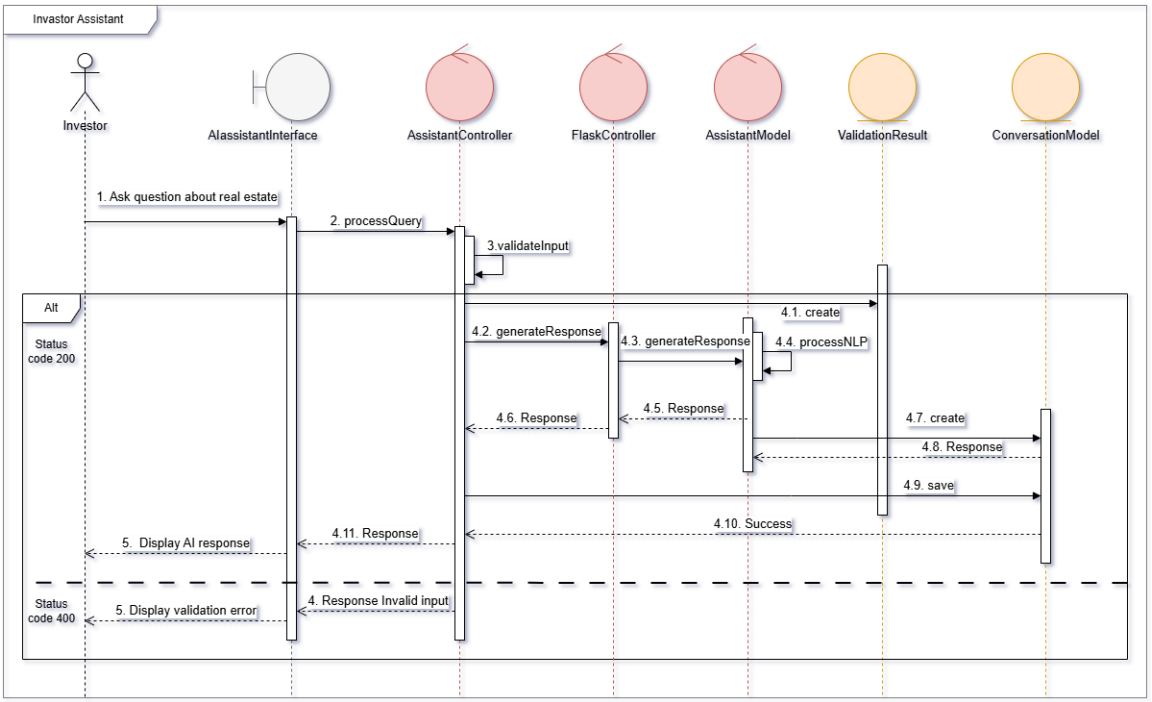


Figure 4.20: Real Estate Assistant MVC Sequence Diagram

Workflow Diagram

The Real Estate Assistant follows a structured workflow to process user queries and provide accurate legal information. Figure 4.21 illustrates the complete workflow from user input to response generation.

AI Approach Selection: RAG vs Fine-Tuning vs Prompt Engineering

The development of the Real Estate Assistant required careful consideration of different AI approaches to ensure optimal performance for domain-specific legal and regulatory queries. This section analyzes the three primary approaches and justifies our selection.

Prompt Engineering Prompt engineering is the process of crafting input text (prompts) that guide a pre-trained language model to respond in a specific way. The purpose is to optimize the interaction between users and the model without modifying the underlying model parameters.

Example Comparison:

- **Without prompt engineering:** User asks "What is the capital of France?" and receives a simple response: "Paris"
- **With prompt engineering:** The system uses a crafted prompt: "In a friendly tone, tell me: What is the capital of France?" resulting in: "Oh, that's easy! The capital of France is Paris!"

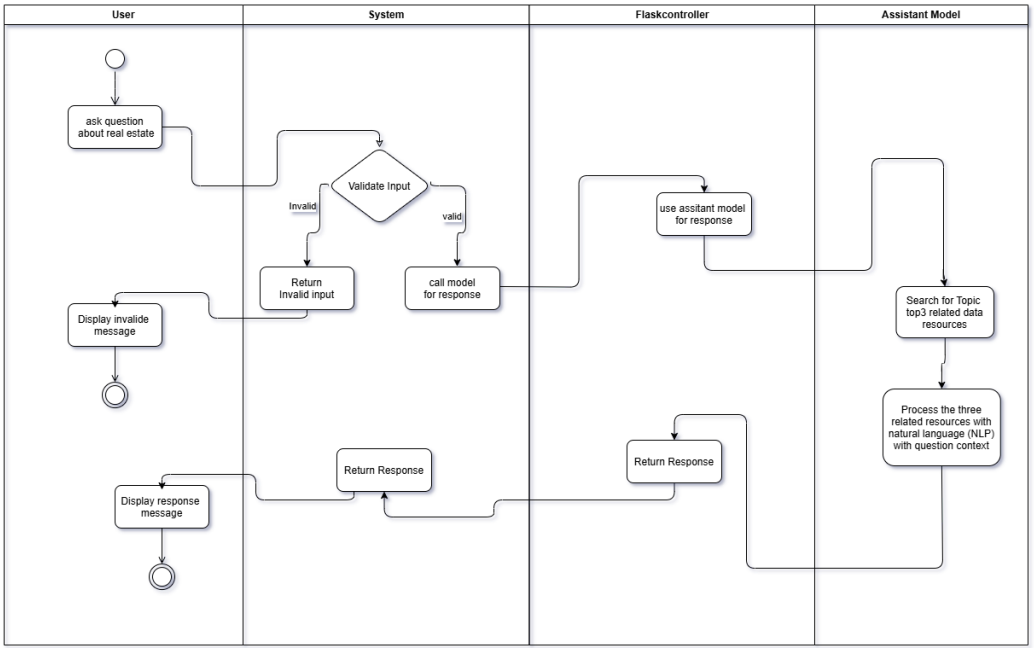


Figure 4.21: Real Estate Assistant Workflow Diagram

Fine-Tuning Fine-tuning involves training a pre-trained model using domain-specific datasets. While pre-trained models excel at general conversational abilities, they often struggle with specialized domains and may produce inaccurate responses or hallucinations when addressing intricate domain-specific questions.

Characteristics:

- Requires substantial computational resources (high-end GPUs, large memory)
- Needs cleaned, labeled datasets specific to the domain
- Demands technical expertise in large language model training
- Suitable for specialized tasks with sufficient training data

Retrieval-Augmented Generation (RAG) RAG combines retrieval and generation components to make large language models context-aware using external data sources. The retriever fetches relevant information from knowledge bases or vector databases, which the generator then uses alongside the original query to produce accurate, contextually relevant responses.

Conceptual Explanation: RAG is an AI architecture that enhances the capabilities of large language models by providing them with access to external knowledge sources. Unlike traditional language models that rely solely on their training data, RAG systems can dynamically retrieve and incorporate relevant information from external databases, documents, or knowledge bases to generate more accurate and contextually appropriate responses.

Standard RAG Workflow: The RAG process follows a systematic four-step workflow:

1. **Embed the user's query:** Convert the user's input into a vector representation that captures its semantic meaning

2. **Retrieve relevant documents:** Search through the knowledge base using the query embedding to find the most relevant documents or passages
3. **Add them into the model's prompt:** Combine the retrieved context with the original user query to create an enhanced prompt
4. **Generate a response:** Use the LLM with this additional context to produce a comprehensive and accurate response

Embedding Concept: Embedding is a method of representing objects such as text, images, and audio as points in a continuous vector space where the locations of those points are semantically meaningful to machine learning algorithms. In the context of RAG, embeddings enable the system to understand and compare the semantic similarity between the user's query and the documents in the knowledge base, ensuring that the most relevant information is retrieved.

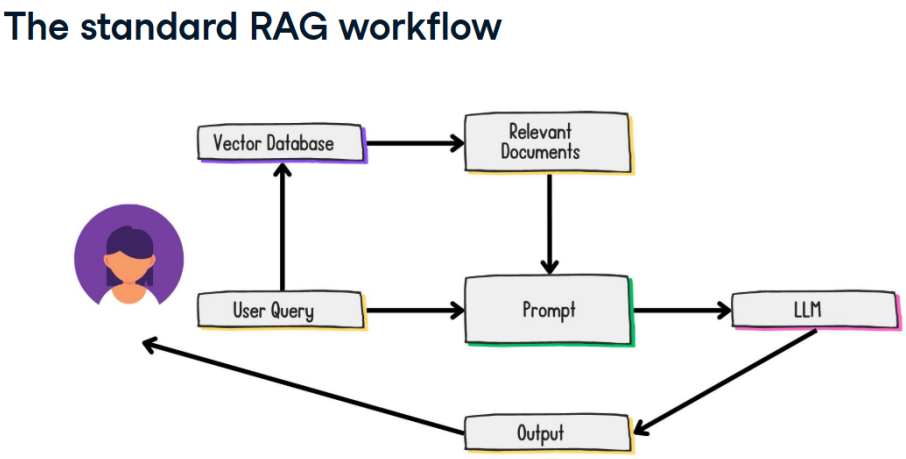


Figure 4.22: RAG Workflow Diagram

Advantages:

- Ideal for augmenting models with external, up-to-date, or domain-specific information
- Effective for constantly changing data (legal documents, regulations, FAQs)
- Requires limited resources and expertise compared to fine-tuning
- Can be implemented using established frameworks like LangChain and OpenAI API

Decision Rationale For our Real Estate Assistant, we selected RAG over the other approaches for the following reasons:

1. **Domain Specificity:** Our model needs to respond using specialized knowledge about Tunisian real estate law and regulations that was not present in the pre-trained model's original training data

2. **Resource Efficiency:** RAG implementation requires significantly fewer computational resources and technical expertise compared to fine-tuning
3. **Data Characteristics:** Legal documents and regulations are constantly evolving, making RAG's ability to retrieve up-to-date information crucial
4. **Development Speed:** RAG allows for rapid prototyping and deployment using existing frameworks and APIs

Prompt engineering alone was insufficient for our use case, as we required the model to access and utilize domain-specific knowledge that extends beyond general conversational abilities.

Prompt Engineering Implementation

The following pseudo code demonstrates our prompt engineering approach for the Real Estate Assistant:

```
ALGORITHM: ChatPromptTemplate for Real Estate Assistant
INPUT: context, question, language
OUTPUT: structured_prompt

BEGIN
    // Initialize ChatPromptTemplate
    prompt = ChatPromptTemplate.from_template(
        "Based on the following context, provide a detailed answer to the user's query.

        Context: {context}
        User Query: {question}
        Response Language: {language}

        Instructions:
        1. Extract the relevant answer from the ANSWER section in the context
        2. Present the information in a clear, structured way
        3. Use the exact information from the context without adding external knowledge

        If the query is general conversation (like greetings, how are you, etc.),
        respond naturally in the specified language.

        If the query is about a topic covered in the context but is within general knowledge,
        respond with the exact information from context.

        If the query is not related to real estate, respond with the equivalent of
        'I'm specialized in Tunisian real estate law and regulations.'

        If no relevant information is found and you cannot provide a general answer,
```

```
        respond with the equivalent of 'I don't have specific information about this topic
        in my current knowledge base.'"
    )

    // Format the prompt with actual values
    formatted_prompt = prompt.format(
        context=context,
        question=question,
        language=language
    )

    RETURN formatted_prompt
END
```

4.3.3 Implementation

Large Language Model Selection

Choosing an appropriate LLM can be challenging given the numerous companies and different models available in the market. The host organization provided us with a Google Gemini API key, which constrained our selection to the available Gemini offerings.

Among the Gemini model family, we evaluated several options:

- **Gemini 2.5 models**: While these represent the latest technology, they are still in experimental release phase, designed primarily to gather feedback and deliver updates quickly to developers
- **Gemini Flash 2.0**: Selected as the optimal choice based on efficiency and cost considerations

Gemini Flash 2.0 was chosen for the following reasons:

- Superior efficiency and cost-effectiveness compared to other available models
- Lack of fine-tuning support, which reinforced our decision to implement RAG architecture
- Stable release status ensuring reliable performance for production deployment

Voice Generation Implementation

For voice generation capabilities, we selected ElevenLabs, an advanced AI-powered platform specializing in text-to-speech and voice synthesis technologies. ElevenLabs enables the generation of realistic, human-like voices suitable for various applications including narrations, audiobooks, podcasts, and video content.

The following pseudo code demonstrates our voice generation implementation:

```
ALGORITHM: Voice Generation for Real Estate Assistant
INPUT: text_response, user_preferences, voice_settings
```

```
OUTPUT: audio_stream

BEGIN
    // Initialize ElevenLabs API connection
    api_client = initialize_elevenlabs_client(API_KEY)

    // Voice configuration
    voice_id = get_selected_voice_id(user_preferences)
    voice_settings = {
        stability: 0.8,
    }

    // Text preprocessing
    cleaned_text = preprocess_text(text_response)
    chunks = split_text_into_chunks(cleaned_text, MAX_CHUNK_SIZE)

    // Voice generation process
    audio_segments = []
    FOR each chunk in chunks DO
        try:
            audio_segment = api_client.generate_voice(
                text=chunk,
                voice_id=voice_id,
                model="eleven_multilingual_v2",
                voice_settings=voice_settings
            )
            audio_segments.append(audio_segment)
        catch APIException as e:
            log_error("Voice generation failed: " + e.message)
            RETURN fallback_response()
        end try
    END FOR

    // Combine audio segments
    final_audio = concatenate_audio_segments(audio_segments)

    // Post-processing
    final_audio = apply_audio_filters(final_audio)
    final_audio = normalize_volume(final_audio)

    RETURN final_audio
END
```

NLP Model Architecture

The Real Estate Assistant utilizes advanced natural language processing techniques to understand and respond to user queries. The system employs:

- **Intent Recognition**: Identifies the purpose of user questions (property law, taxes, contracts, etc.)
- **Entity Extraction**: Extracts key information like property types, locations, and legal concepts
- **Context Management**: Maintains conversation history for coherent multi-turn dialogues
- **Response Generation**: Creates natural, informative responses based on legal knowledge base

Legal Knowledge Base

The assistant's knowledge base contains comprehensive information about:

- Tunisian real estate law and regulations
- Property investment procedures
- Tax implications and calculations
- Contract templates and requirements
- Common legal issues and solutions

Chat Interface Implementation

The mobile chat interface provides an intuitive way for investors to interact with the AI assistant. Figure 4.23 shows the chat interface design.

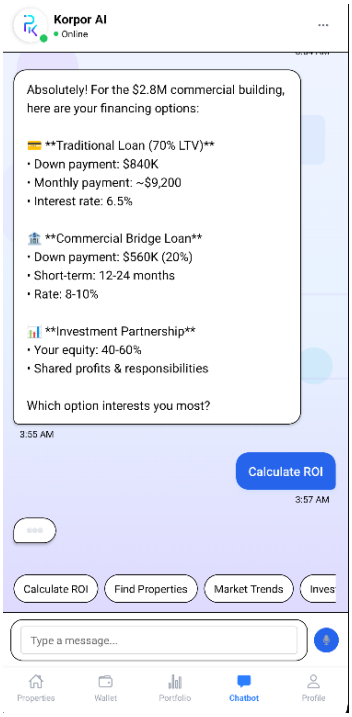


Figure 4.23: Mobile Chat Interface for Real Estate Assistant

4.3.4 Test

Test Scenarios

The Real Estate Assistant underwent extensive testing to ensure accurate responses and reliable performance. Table 4.4 presents the key test scenarios.

Scenario	Input	Expected Output	Status
Property tax query	"What are property taxes?"	Detailed tax information	✓
Contract question	"What's in a rental contract?"	Contract requirements	✓
Investment procedure	"How to buy property?"	Step-by-step guide	✓
Legal compliance	"Foreign investment rules?"	Regulatory information	✓
Context follow-up	Multi-turn conversation	Coherent responses	✓
Unclear question	Ambiguous query	Clarification request	✓

Table 4.4: Real Estate Assistant Test Scenarios

4.3.5 Retrospective

The Real Estate Assistant sprint demonstrated the successful integration of NLP technology into the platform. Table 4.5 summarizes the key outcomes and future improvements.

Category	Details
What Went Well	• Successfully implemented NLP chatbot functionality • Mobile interface provides intuitive user experience • Legal knowledge base effectively answers user queries • Comprehensive testing validated system reliability
Action Items	• Expand legal knowledge base coverage • Implement multilingual support for broader accessibility • Add conversation analytics for continuous improvement • Integrate voice recognition capabilities

Table 4.5: Real Estate Assistant Sprint Retrospective Summary

4.4 Sprint 5: Role-Based Backoffice Agent

Introduction

The Role-Based Backoffice Agent is an intelligent AI system designed to assist different user roles (Super Admin, Admin, Real Estate Agent) with automated task management, workflow optimization, and decision support. This AI agent adapts its behavior and recommendations based on the specific role and responsibilities of the authenticated user.

4.4.1 Analysis

Use Case Diagram

The Role-Based Backoffice Agent serves multiple user types with role-specific functionalities and automated assistance. Figure 4.24 illustrates the main use cases for each role.

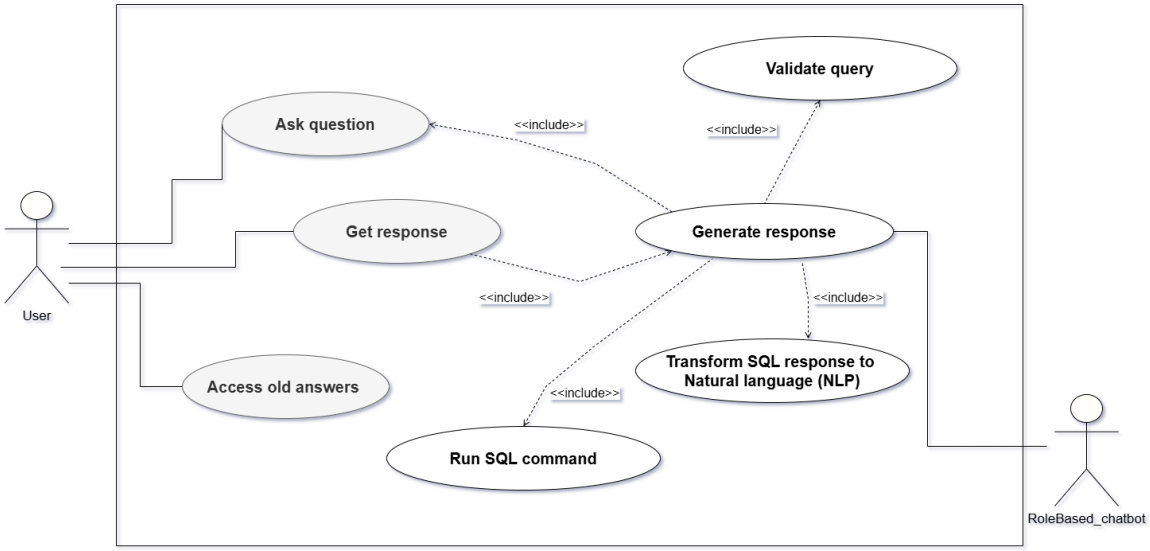


Figure 4.24: Role-Based Backoffice Agent Use Case Diagram

Textual Use Case Descriptions

Use Case	**agent for administrative tasks**
Actor	User (representing Super Admin, Admin, Real Estate Agent)
Precondition	User is authenticated with specific role permissions
Main Scenario	User get comprehensive answer from platform database
Postcondition	Tasks are efficiently managed with minimal manual intervention

Table 4.6: Role-Based Backoffice Agent Use Case Description

4.4.2 Modeling

Class Diagram

The Role-Based Backoffice Agent follows a role-based architecture with specialized services for each user type. Figure 4.25 shows the main classes and their relationships.

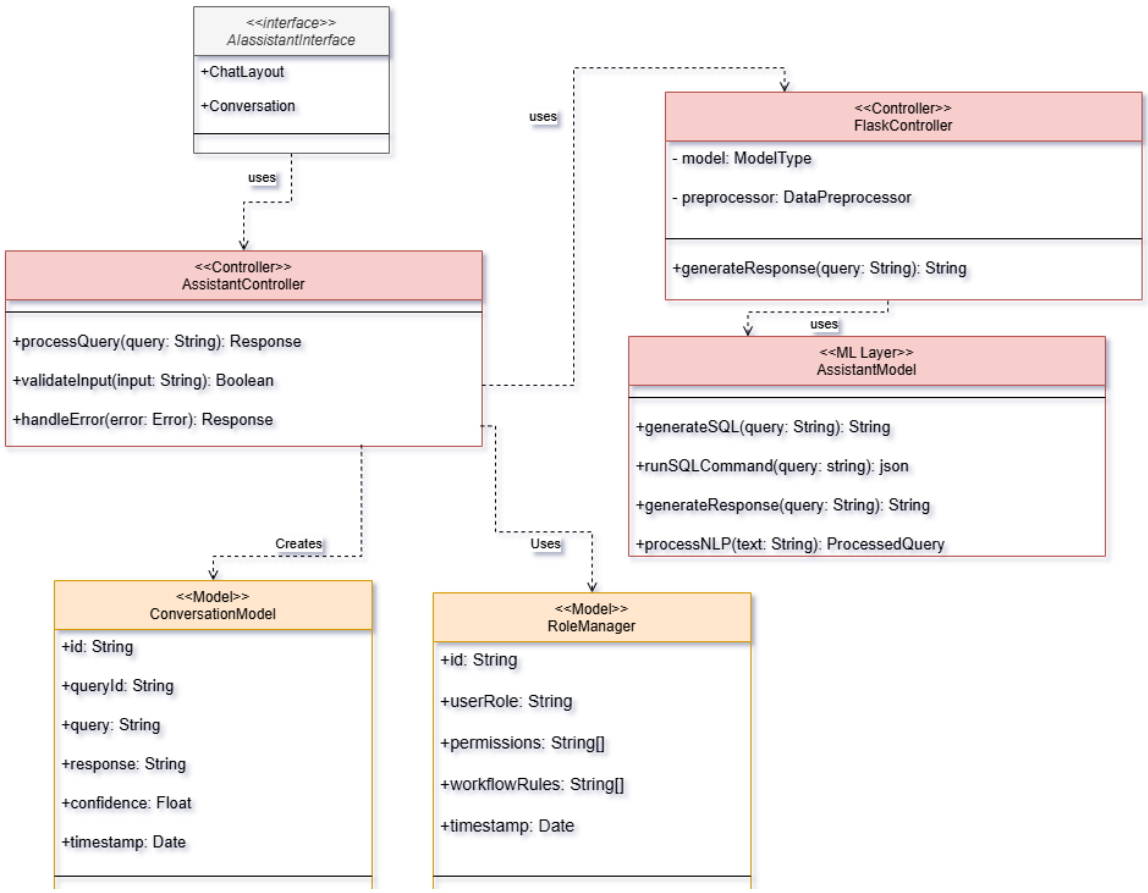


Figure 4.25: Role-Based Backoffice Agent Class Diagram

Sequence Diagram (MVC)

The interaction flow between the web interface, role management services, and AI processing components is illustrated in Figure 4.26.

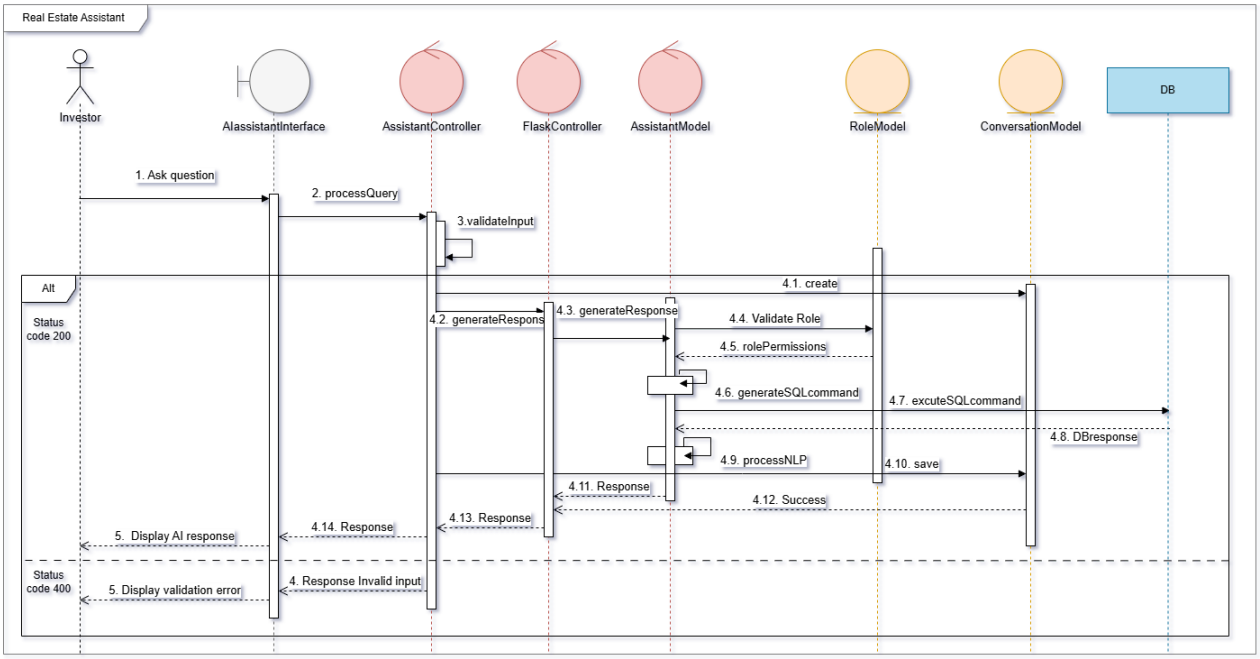


Figure 4.26: Role-Based Backoffice Agent MVC Sequence Diagram

4.4.3 Implementation

Role-Based AI Logic

The Backoffice Agent utilizes sophisticated role-based AI algorithms to provide personalized assistance:

- **Super Admin Features**: System monitoring, user management, platform analytics, security oversight
- **Admin Features**: Property management, user support, content moderation, performance tracking
- **Real Estate Agent Features**: Client management, property listings, sales tracking, commission calculations

Natural Language to SQL Processing

The following pseudo code demonstrates the complete workflow for converting natural language queries to SQL and generating human-readable responses:

Natural Language to SQL Generation

```
FUNCTION convert_question_to_sql(user_question, user_id):

    // Step 1: Get user information
    user_info = get_user_role_and_store(user_id)
```

```
    role = user_info.role          // 'user', 'admin', or 'super_admin'
    agency_id = user_info.agency_id

    // Step 2: Create database description with permissions
    database_info = build_database_description(role, agency_id)

    // Step 3: Ask AI to write SQL
    prompt = "Given this database: {database_info}
              User asks: {user_question}
              Write a MySQL query that respects user permissions.
              Only return the SQL code."

    sql_query = ask_ai_model(prompt)
    clean_sql = remove_formatting(sql_query)

    // Step 4: Check if SQL is safe
    IF is_sql_safe(clean_sql, role, agency_id):
        RETURN clean_sql
    ELSE:
        RETURN "Cannot create safe query"

FUNCTION get_user_role_and_store(user_id):
    query = "SELECT role, store_id FROM users WHERE id = ?"
    result = execute_database_query(query, user_id)
    RETURN result

FUNCTION build_database_description(role, agency_id):
    description = "Tables: users, properties, agency, transactions, clients

                   Access Rules:
                   - user: Can only see their agency's data
                    - admin: Can see all data in their agency
                   - super_admin: Can see everything"

    IF role != 'super_admin':
        description += f"IMPORTANT: Must filter by agency_id = {agency_id}"

    RETURN description

FUNCTION is_sql_safe(sql, role, store_id):
    // Block dangerous operations
    dangerous_words = ['DELETE', 'DROP', 'UPDATE', 'INSERT']
```

```
    FOR word IN dangerous_words:
        IF word IN sql.upper():
            RETURN false

    // Check role permissions
    IF role != 'super_admin' AND 'WHERE' NOT IN sql.upper():
        RETURN false  // Must have WHERE clause

    RETURN true
```

SQL Query Execution

```
FUNCTION run_sql_query(sql_query):

    TRY:
        // Step 1: Connect to database
        connection = connect_to_database()

        // Step 2: Run the query with timeout
        start_time = current_time()
        results = execute_query(connection, sql_query, timeout=30)
        end_time = current_time()

        // Step 3: Limit results if too many
        IF length(results) > 1000:
            results = first_1000_items(results)
            results.add("... (truncated)")

        // Step 4: Return success
        RETURN {
            'success': true,
            'data': results,
            'execution_time': end_time - start_time
        }

    CATCH error:
        // Step 5: Handle errors nicely
        user_message = make_error_friendly(error)
        RETURN {
            'success': false,
            'error': user_message
        }
```

```
    FINALLY:
        close_database_connection(connection)

FUNCTION make_error_friendly(database_error):
    error_text = string(database_error).lower()

    IF 'syntax' IN error_text:
        RETURN "There's a problem with the query format"
    ELSE IF 'timeout' IN error_text:
        RETURN "Query took too long to run"
    ELSE IF 'connection' IN error_text:
        RETURN "Cannot connect to database"
    ELSE:
        RETURN "Database error occurred"
```

Database Results to Natural Language

```
FUNCTION convert_results_to_answer(query_results, user_role):

    // Step 1: Check if we have data
    IF query_results is empty:
        RETURN "No data found for your request"

    // Step 2: Choose response style based on user role
    response_style = get_response_style(user_role)

    // Step 3: Prepare data for AI
    simplified_data = simplify_data_for_ai(query_results)

    // Step 4: Ask AI to write natural response
    prompt = f"Convert this data to natural language:
              Data: {simplified_data}
              Style: {response_style}
              Keep it short and clear."

    natural_response = ask_ai_model(prompt)
    clean_response = clean_ai_response(natural_response)

    RETURN clean_response

FUNCTION get_response_style(user_role):
    styles = {
        'super_admin': "Executive summary with key insights",
```

```
        'admin': "Management report with important numbers",
        'user': "Simple explanation in everyday language"
    }
    RETURN styles[user_role]

FUNCTION simplify_data_for_ai(results):
    // If too much data, summarize it
    IF length(results) > 50:
        summary = create_data_summary(results)
        RETURN summary
    ELSE:
        RETURN results

FUNCTION create_data_summary(large_dataset):
    summary = {
        'total_rows': length(large_dataset),
        'sample_data': first_10_rows(large_dataset),
        'key_numbers': extract_important_numbers(large_dataset)
    }
    RETURN summary
```

Main Processing Flow

```
FUNCTION process_user_request(user_question, user_id):

    // Step 1: Convert question to SQL
    sql_query = convert_question_to_sql(user_question, user_id)

    IF sql_query == "Cannot create safe query":
        RETURN "Sorry, I cannot process that request safely"

    // Step 2: Run the SQL query
    query_results = run_sql_query(sql_query)

    IF query_results.success == false:
        RETURN f"Database error: {query_results.error}"

    // Step 3: Convert results to natural language
    user_info = get_user_role_and_store(user_id)
    final_answer = convert_results_to_answer(query_results.data, user_info.role)

    // Step 4: Log the interaction (optional)
    log_interaction(user_question, sql_query, final_answer, user_id)
```

```
    RETURN final_answer

// Example usage
FUNCTION main():
    user_question = "How many properties do we have listed?"
    user_id = 3

    answer = process_user_request(user_question, user_id)
    print(answer)
```

Automated Workflow Management

The system automates routine tasks based on role permissions:

- Property approval workflows for admins
- User verification processes for super admins
- Client follow-up reminders for agents
- Performance report generation for all roles

Dashboard Implementation

Each role receives a customized dashboard with relevant tools and insights. Figure **??** shows the role-specific dashboard implementations.

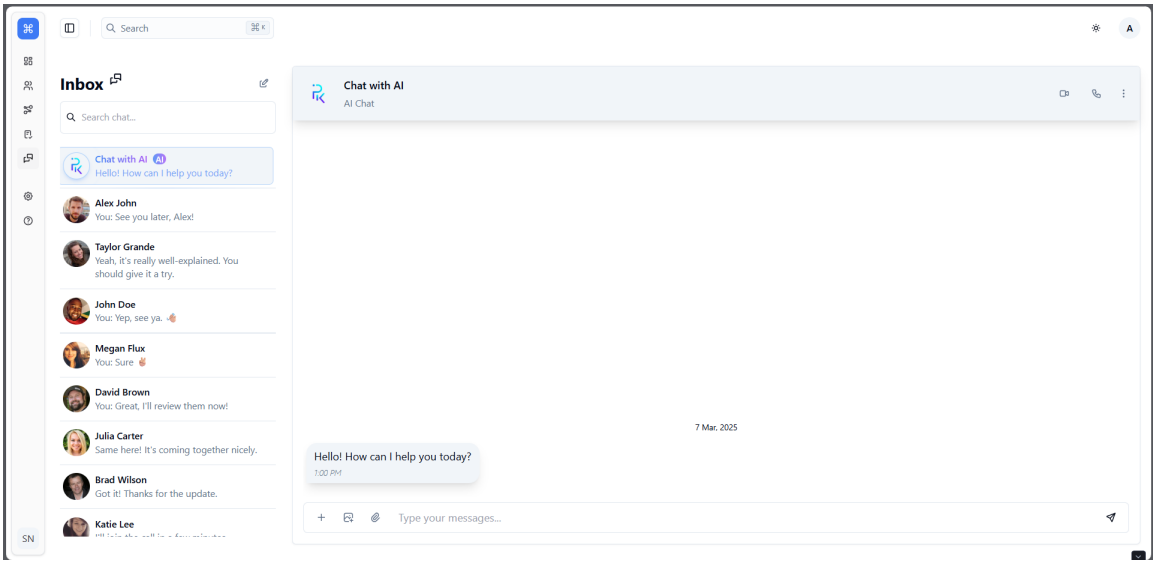


Figure 4.27: Web Interface for Real Estate Assistant

4.4.4 Test

Test Scenarios

The Role-Based Backoffice Agent underwent comprehensive testing across all user roles. Table 4.7 presents the key test scenarios.

Scenario	User Role	Expected Output	Status
Role identification	Super Admin	Admin dashboard access	✓
Task automation	Admin	Property approval workflow	✓
Performance insights	Real Estate Agent	Sales analytics	✓
Permission validation	Admin	Restricted super admin features	✓
AI recommendations	All roles	Role-specific suggestions	✓
Workflow management	Super Admin	System optimization tips	✓

Table 4.7: Role-Based Backoffice Agent Test Scenarios

4.4.5 Retrospective

The Role-Based Backoffice Agent sprint successfully delivered customized administrative support for different user roles. Table 4.8 summarizes the key achievements and future enhancements.

Category	Details
What Went Well	• Successfully implemented role-based AI assistance • Automated workflow management improved efficiency • Web interface provides comprehensive administrative tools • Role permissions and security controls work effectively
Action Items	• Add more sophisticated automation rules • Implement predictive analytics for task prioritization • Enhance dashboard customization options • Integrate with external business intelligence tools

Table 4.8: Role-Based Backoffice Agent Sprint Retrospective Summary

4.5 Sprint 6: Investor-Focused Recommendation System

Introduction

The Investor-Focused Recommendation System is an advanced AI model that analyzes investor behavior, preferences, and market conditions to provide personalized property investment recommendations. This system leverages machine learning algorithms to match investors with optimal investment opportunities based on their risk profile, budget, and investment goals.

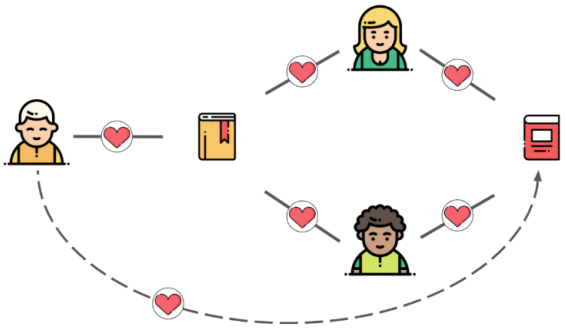


Figure 4.28: Collaborative Filtering Recommendation System Overview

4.5.1 Analysis

Use Case Diagram

The Recommendation System serves investors by analyzing their profiles and suggesting suitable investment opportunities. Figure 4.29 illustrates the main use cases for the recommendation engine.

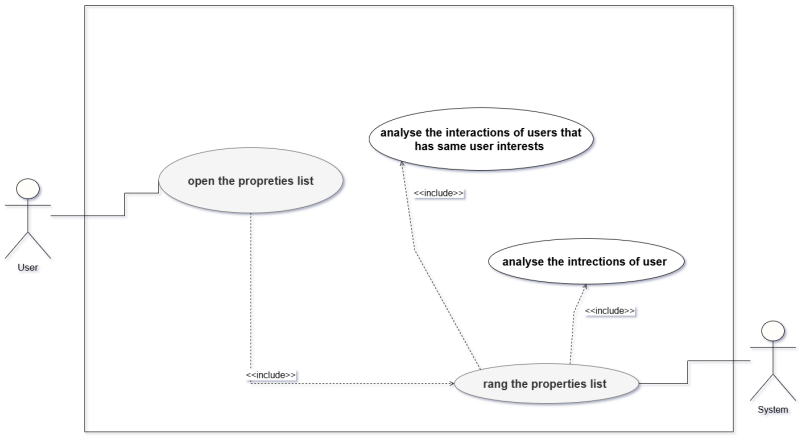


Figure 4.29: Investor-Focused Recommendation System Use Case Diagram

Textual Use Case Descriptions

Use Case	**Generate Investment Recommendations**
Actor	User (Investor)
Precondition	Investor has completed profile setup and preferences
Main Scenario	System matches properties with investor criteria
Postcondition	Investor receives tailored investment recommendations

Table 4.9: Investor-Focused Recommendation System Use Case Description

4.5.2 Modeling

Class Diagram

The Recommendation System uses collaborative filtering and content-based algorithms for personalized suggestions. Figure 4.30 shows the main classes and their relationships.

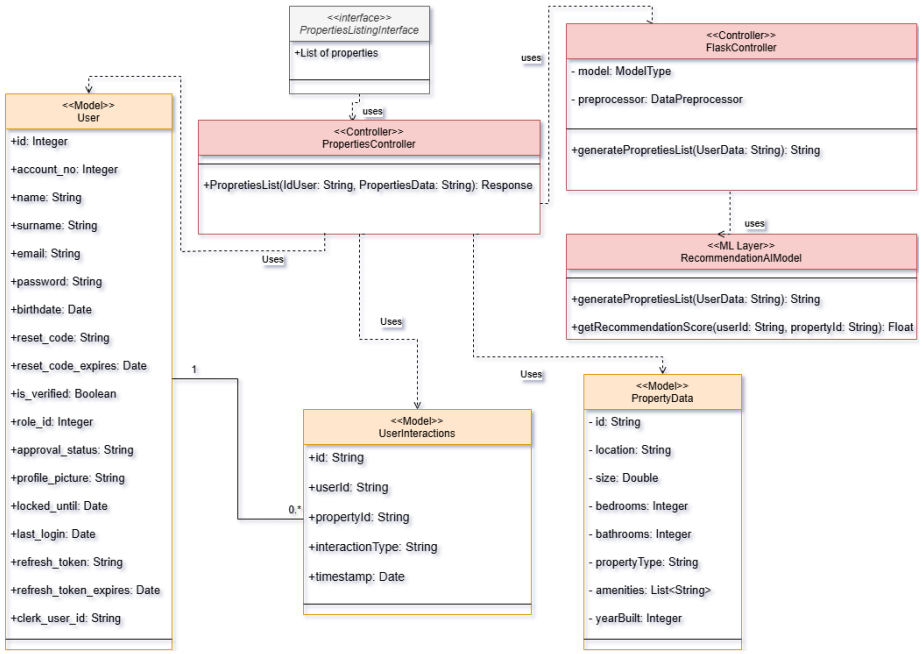


Figure 4.30: Investor-Focused Recommendation System Class Diagram

Sequence Diagram (MVC)

The interaction flow between the mobile interface, recommendation engine, and machine learning components is illustrated in Figure 4.31.

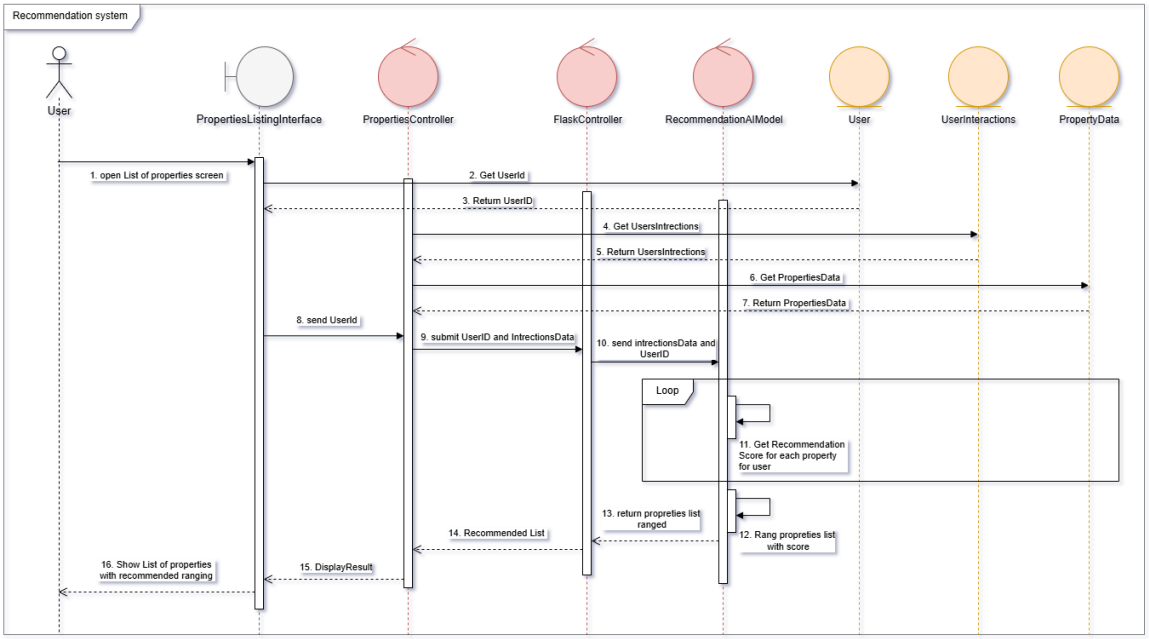


Figure 4.31: Recommendation System MVC Sequence Diagram

4.5.3 Implementation

Machine Learning Algorithms

The Recommendation System employs multiple ML techniques for optimal suggestions:

- **Collaborative Filtering**: Analyzes similar investor behaviors and preferences
- **Content-Based Filtering**: Matches properties based on investor criteria
- **Hybrid Approach**: Combines multiple algorithms for enhanced accuracy
- **Deep Learning**: Neural networks for complex pattern recognition

Investor Profiling

The system creates comprehensive investor profiles including:

- Risk tolerance assessment
- Investment budget and timeline
- Geographic preferences
- Property type preferences

- Previous investment history
- Market behavior analysis

Mobile Recommendations Interface

The mobile app provides an intuitive interface for viewing personalized recommendations. Figure 4.32 shows the mobile recommendation implementation.

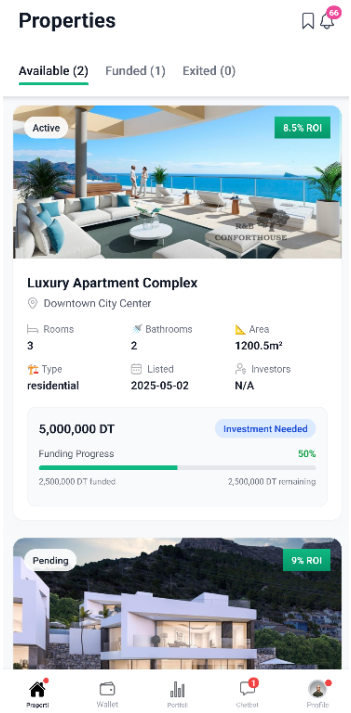


Figure 4.32: Mobile Interface for Investment Recommendations

Recommendation Algorithm Implementation

The following pseudo code demonstrates our hybrid recommendation algorithm that combines user preferences, collaborative filtering, and popularity scoring:

```
FUNCTION calculate_property_score(user_id, property_id):
    user_score = get_user_preference_score(user_id, property_id)
    similar_score = get_similar_users_score(user_id, property_id)
    popularity_score = get_popularity_score(property_id)

    final_score = (user_score * 0.5) + (similar_score * 0.3) + (popularity_score * 0.2)
    RETURN final_score

FUNCTION get_user_preference_score(user_id, property_id):
    user = get_user(user_id)
    property = get_property(property_id)
```

```
    score = 0

    IF property.type == user.preferred_type: score += 30
    IF property.location == user.preferred_location: score += 20
    score += count_matching_amenities(property.amenities, user.preferred_amenities) * 2

    RETURN score

FUNCTION get_similar_users_score(user_id, property_id):
    similar_users = find_similar_users(user_id, limit=10)
    total_score = 0
    user_count = 0

    FOR each similar_user IN similar_users:
        interaction = get_user_interaction(similar_user.id, property_id)
        IF interaction exists:
            weighted_score = calculate_interaction_score(interaction) * similar_user.similari
            total_score += weighted_score
            user_count += 1

    RETURN user_count > 0 ? total_score / user_count : 0

FUNCTION find_similar_users(user_id, limit):
    target_user = get_user(user_id)
    similar_users = []

    FOR each other_user IN get_all_users():
        IF other_user.id != user_id:
            similarity = calculate_user_similarity(target_user, other_user)
            IF similarity > 0.3:
                similar_users.add({'id': other_user.id, 'similarity': similarity})

    similar_users.sort_by_similarity_desc()
    RETURN similar_users.take(limit)

FUNCTION calculate_user_similarity(user1, user2):
    similarity = 0

    IF user1.preferred_type == user2.preferred_type: similarity += 0.3
    IF user1.preferred_location == user2.preferred_location: similarity += 0.3
    similarity += count_common_amenities(user1.amenities, user2.amenities) * 0.05
```

```
    RETURN min(similarity, 1.0)

FUNCTION calculate_interaction_score(interaction):
    base_scores = {"view": 10, "favorite": 30, "contact": 50, "invest": 90}
    score = base_scores[interaction.type]

    days_ago = days_since(interaction.timestamp)
    IF days_ago <= 7: score *= 1.2
    ELSE IF days_ago > 30: score *= 0.8

    RETURN score
```

Personalized Dashboard

Investors receive a personalized dashboard with recommendations and portfolio insights. Figure **??** shows the recommendation dashboard design.

4.5.4 Test

Test Scenarios

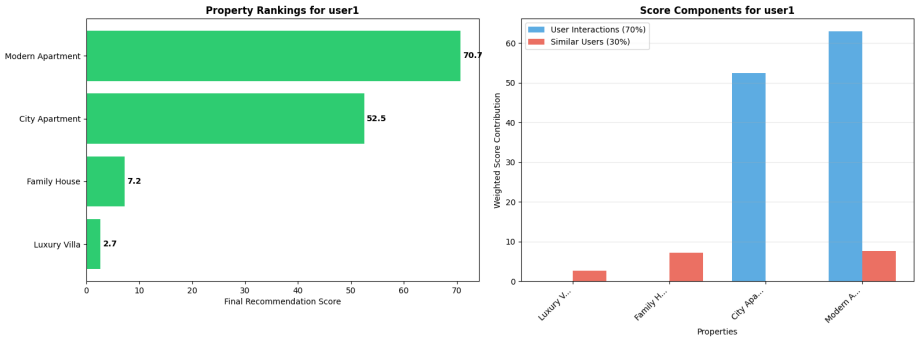
The Recommendation System underwent extensive testing to ensure accuracy and relevance. Table 4.10 presents the key test scenarios.

Scenario	Input	Expected Output	Status
Algorithm accuracy	Historical data	High precision score	✓
New investor profile	Basic preferences	Relevant recommendations	✓

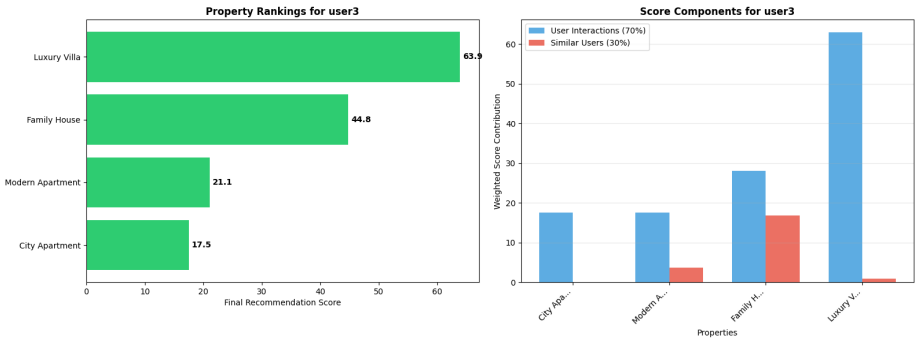
Table 4.10: Recommendation System Test Scenarios

Algorithm Visualization Results

The recommendation algorithm generates personalized property rankings based on user preferences and collaborative filtering. Figure 4.33 demonstrates the algorithm's output for different user profiles, showing both final scores and component breakdowns.



Property Rankings for User1



Property Rankings for User3

Figure 4.33: Recommendation Algorithm Visualization Results

4.5.5 Retrospective

The Investor-Focused Recommendation System sprint successfully delivered personalized investment guidance through advanced machine learning algorithms. Table 4.11 summarizes the key achievements and future improvements.

Category	Details
What Went Well	• Successfully implemented collaborative filtering algorithms • Mobile interface provides excellent user experience • Comprehensive testing validated system performance
Action Items	• Implement real-time model

Table 4.11: Investor-Focused Recommendation System Sprint Retrospective Summary

CHAPTER 5

Blockchain

Blockchain is the technology that allows trust without trusting.

— Vitalik Buterin [50]

Introduction

The blockchain payment integration represents a crucial component of the Korpor platform, focusing on secure investor payment processing and transparent transaction recording [51]. This sprint implements a hybrid approach combining traditional payment gateways with blockchain technology to ensure both user convenience and transaction security.

The main feature centers on investor payment flows where users can securely invest in real estate projects through Paymee payment gateway, with all transactions automatically recorded on the blockchain for transparency and immutability [52]. This integration provides investors with traditional payment convenience while leveraging blockchain's security and transparency benefits.

5.1 Sprint 7: Blockchain Payment Integration

5.1.1 Analysis

Use Cases Overview

The blockchain integration within the **Korpor** platform addresses four core functionalities: **Investment Recording** for immutable ownership proof, **Project Registration** for on-chain project verification, **Rent Distribution** for automated fair payout calculation, and **Transaction Verification** for comprehensive audit trails and regulatory compliance.

Objectives of Integration

The blockchain integration enhances the platform through three key objectives: **Security and trust** via cryptographically signed transactions [53], **Decentralization and immutability** ensuring data cannot be altered [54], and **Smart contract automation** for efficient business logic execution [55].

Use Case Diagram

The transaction management system serves as the core component for handling all blockchain-based investment activities within the Korpor platform. Figure 5.1 illustrates the main use cases for blockchain transaction management, showing how the system automatically saves all transactions as immutable smart contracts.

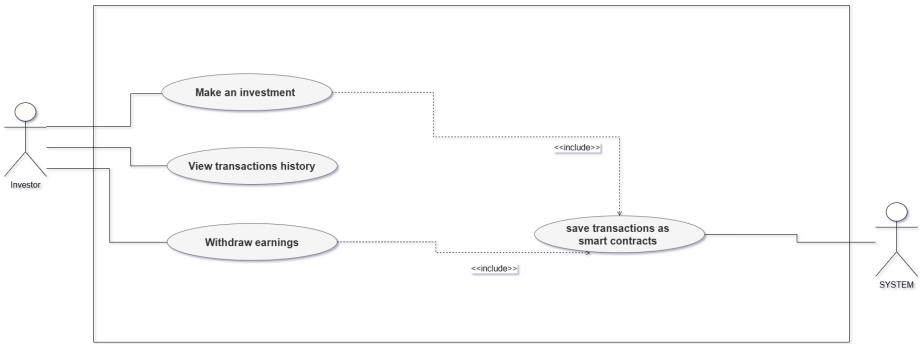


Figure 5.1: Blockchain Transaction Management Use Case Diagram

Textual Use Case Descriptions

Use Case	**Make an Investment**
Actor	Investor
Precondition	Investor is authenticated and has sufficient funds
Main Scenario	Investor selects a property and initiates investment through secure payment processing
Postcondition	Investment is recorded on blockchain and investor receives confirmation

Table 5.1: Make Investment Transaction Use Case Description

Use Case	**View Transactions History**
Actor	Investor
Precondition	Investor is authenticated and has previous transactions
Main Scenario	Investor accesses comprehensive transaction history with blockchain verification
Postcondition	Complete transaction timeline is displayed with verifiable blockchain records

Table 5.2: View Transaction History Use Case Description

Use Case	Withdraw Earnings
Actor	Investor
Precondition	Investor has available earnings from property investments
Main Scenario	Investor requests withdrawal and system processes payment securely
Postcondition	Earnings are transferred and withdrawal is recorded on blockchain

Table 5.3: Withdraw Earnings Transaction Use Case Description

Use Case	Save Transactions as Smart Contracts
Actor	System
Precondition	Transaction has been initiated and validated
Main Scenario	System automatically records transaction details in immutable smart contracts
Postcondition	Transaction is permanently stored on blockchain for transparency and audit

Table 5.4: Save Transactions as Smart Contracts Use Case Description

5.1.2 Modeling

The blockchain component is designed to integrate seamlessly within the overall architecture of the **Korpor** platform. This section outlines how the blockchain fits into the system and the technologies used for its implementation.

- **How blockchain fits into the overall system architecture**
  - The backend communicates with smart contracts deployed on the Ethereum blockchain.
  - Transactions such as investments, project registrations, and rent distributions are processed and recorded on-chain.
  - The blockchain acts as a complementary layer to ensure data integrity and traceability.
- **On-chain vs off-chain components**
  - *On-chain:*
    - Smart contracts handle investments, project registrations, and rent distributions.
    - Data recorded on-chain is immutable and publicly verifiable.
  - *Off-chain:*
    - User data, authentication, and detailed analytics are managed by the backend and stored in a centralized database (MySQL).
    - Interaction with the blockchain is facilitated through API endpoints and external services [56].

5.1.3 Implementation

Technologies Used

The blockchain implementation utilizes several key technologies:

- **Solidity:** Smart contract programming language for Ethereum blockchain [57]
- **Ethers.js:** JavaScript library for blockchain interaction [58]
- **Sepolia Testnet:** Ethereum test network for safe development
- **Infura:** Cloud-based Ethereum node access [59]
- **MetaMask:** Cryptocurrency wallet for decentralized applications [60]

Third-Party Payment Integration: Paymee

In the Korpor platform, secure and reliable payment processing is fundamental to enabling users to invest in real estate projects with confidence. While blockchain handles transparency and decentralization for on-chain interactions, fiat payments from users need to be handled via trusted third-party providers. For this purpose, we chose **Paymee** as our main payment gateway.

Role of Paymee in the Architecture

Paymee acts as a bridge between the user's traditional banking system and our decentralized investment platform. When a user decides to invest in a project, the fiat transaction is processed securely via Paymee. Upon confirmation, the platform triggers an on-chain event that records the investment using a smart contract, ensuring both real-world and blockchain-level consistency.

- Paymee provides a secure and verified method for processing payments via bank cards or wallets.
- It offers real-time transaction status updates, which are essential for synchronizing with blockchain confirmations.
- The integration ensures that only successful payments are recorded on-chain, reducing fraud and improving traceability [61].

Justifying the Choice of Paymee

Several third-party payment providers are available in Tunisia, including **Flouci** and **Konnect** , however, **Paymee** was chosen due to its distinct advantages in several key areas:

- **Regulatory Compliance:** Fully compliant with local regulations and has established partnerships with most major Tunisian banks.

- **Developer Experience:** Offers clear documentation, a stable sandbox environment, and responsive customer support.
- **Payment Mode:** Supports a wide range of payment methods, including both local and international options.
- **Pricing:** Offers simplified and transparent pricing with competitive rates.
- **User Experience:** Provides a minimalistic and intuitive user interface that reduces drop-off rates during transactions.
- **Flexibility:** Highly flexible and easily integrates into diverse use cases, adapting to the requirements of individual investors.

Table 5.5: Comparison of Payment Providers: Paymee vs Flouci vs Konnect

Feature	Paymee	Flouci	Konnect
Regulatory Compliance	Fully compliant with local regulations	Limited compliance	Enterprise-focused, may have specific requirements
Developer Experience	Clear documentation, stable sandbox, responsive support	Less stable APIs, limited documentation	Lengthy onboarding, limited flexibility
Payment Model	Ideal for recurring, high-trust investments	Not suited for investment models	Enterprise-focused with fixed pricing
Pricing	Simplified and transparent	Unclear pricing structure	Complex pricing and fees
User Experience	Minimalistic UI, reliable webhooks, low drop-offs	Less reliable, mobile payment focus	Enterprise UI, longer process, limited customization
Flexibility	High flexibility in fee structures	Low flexibility	Low flexibility, rigid fee structures

Based on these factors, **Paymee** was selected as the most suitable payment provider for the Korpor platform, playing a pivotal role in bridging traditional finance with our blockchain-based investment infrastructure.

Smart Contract Design

Smart Contracts Overview A smart contract is a self-executing contract with the terms of the agreement directly written into lines of code [62]. These contracts run on blockchain platforms, such as Ethereum, and are designed to automatically enforce and execute the terms of an agreement without the need for intermediaries.

Smart contracts operate on decentralized networks, ensuring transparency, immutability, and security. They allow parties to interact and transact with one another in a trustless environment, where the contract's logic is executed automatically when predefined conditions are met [63].

Smart Contract Responsibilities The smart contract in the Korpor application is responsible for managing the critical aspects of the platform. The main responsibilities include:

- **Recording Investments:** The smart contract records each user's investment when they invest in a project.
- **Project Registration:** Real estate companies can register new projects.
- **Rent Distribution:** The contract ensures that rental income is distributed fairly to investors.
- **Transaction Verification:** All actions taken within the contract are securely logged on the blockchain.

Data Structures and Functions In the smart contract, several data structures and functions are employed to manage and store key information, facilitating efficient interaction with the system [64].

Data Structures

The contract uses ‘structs‘ to represent complex data types like project and investor details.

Project Struct: The ‘Project‘ struct stores the details of each project:

```
struct Project {
    uint256 projectId;
    string projectName;
    address companyAddress;
    uint256 totalFunding;
    uint256 rentIncome;
    uint256 numInvestors;
}
```

Investor Struct: The investor struct stores individual investment details for tracking purposes.

Mobile Payment Interface

The mobile application provides a seamless interface for investors to manage their blockchain-based transactions. Figure 5.2 demonstrates the complete mobile payment workflow including investment initiation, payment processing, and withdrawal functionality.

Investment Interface Payment Process Withdrawal Interface

Figure 5.2: Mobile Payment Interface: Investment, Payment, and Withdrawal Process

Admin Blockchain Visualization

The super admin dashboard provides comprehensive visualization and management capabilities for blockchain smart contracts. Figure 5.3 shows the administrative interface for monitoring and managing blockchain transactions and smart contract interactions.

Figure 5.3: Super Admin Blockchain Smart Contract Visualization Dashboard

5.1.4 Test

Transaction Flow Testing

The blockchain payment integration was thoroughly tested through a complete end-to-end transaction flow, demonstrating the seamless integration between Paymee payment gateway and

blockchain recording. The following test scenario validates the entire investment process:

Initiate New Transaction The investor initiates a new investment transaction through the platform interface, selecting the desired property and investment amount.

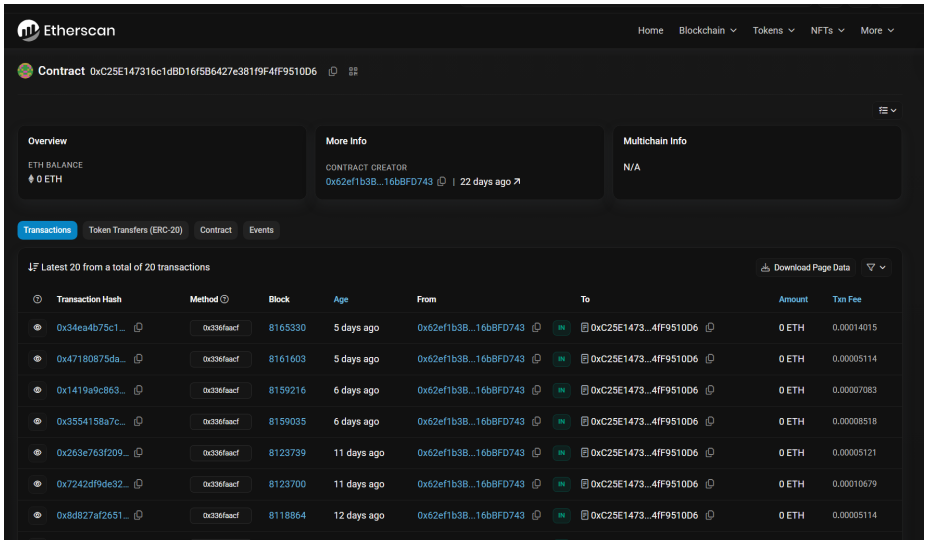


Figure 5.4: New Transaction Initiation Interface

Database Transaction Recording Upon payment initiation, the transaction information is immediately saved to the database with a "pending" status, ensuring proper tracking of the investment process.

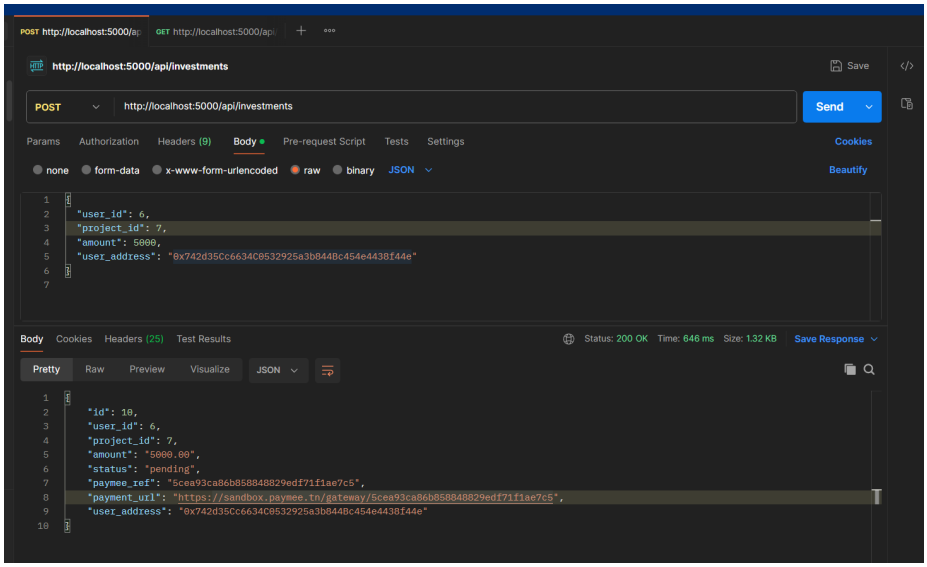


Figure 5.5: Transaction Saved with Pending Status

Payment Confirmation and Blockchain Recording After successful payment confirmation, the Paymee callback is executed, triggering the blockchain investment recording process. The system generates a transaction hash confirming the successful blockchain entry.

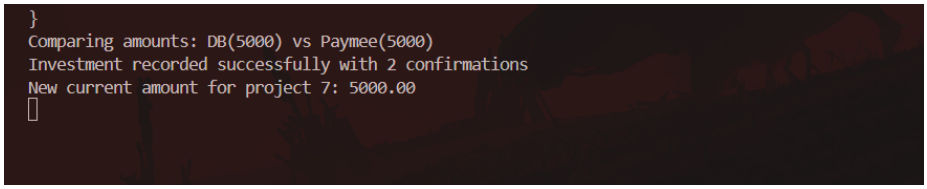


Figure 5.6: Payment Confirmation Success Message

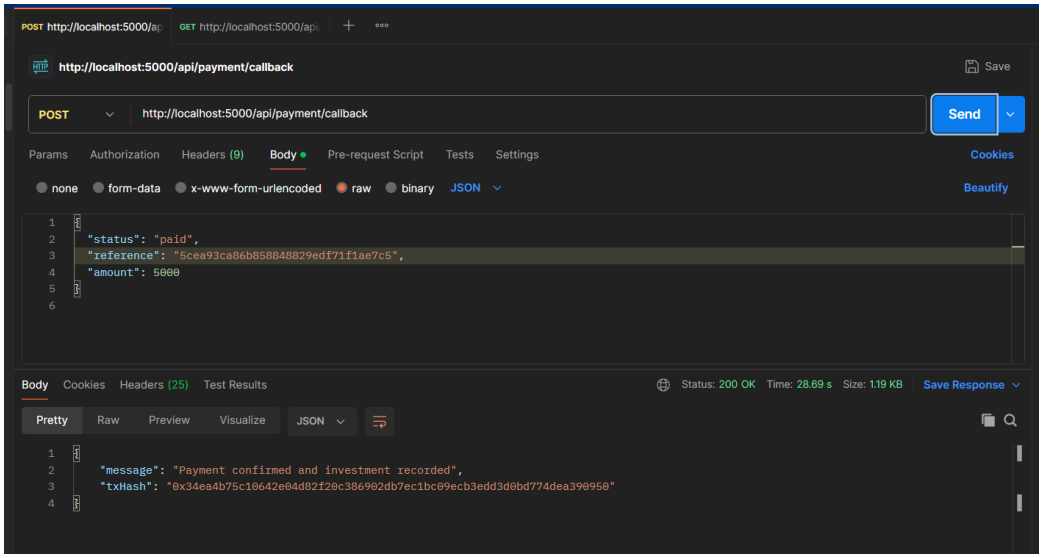


Figure 5.7: Blockchain Transaction Hash Generated

Transaction Status Update The investment status is updated to "confirmed" in the database, the current investment amount is adjusted, and the blockchain record hash is permanently stored for future reference and verification.

id	user_id	project_id	amount	status	paymee_ref	payment_url	user_address	created_at	updated_at	tx_hash
te 10	6	7	5000.00	confirmed	5cea93ca86b858848829edf71f1ae7c5	https://sandbox.paymee.tn/gateway/5cea93ca86b85884...	0x742d35Cc6634C0532925a3b844Bc454e4438f44e	2025-04-22 00:46:44	2025-04-22 00:52:40	0x34ea4b75c10642e04

Figure 5.8: Transaction Status Updated to Confirmed

Blockchain Verification The transaction is successfully recorded on the Ethereum Sepolia testnet, providing immutable proof of the investment that can be independently verified through Etherscan.

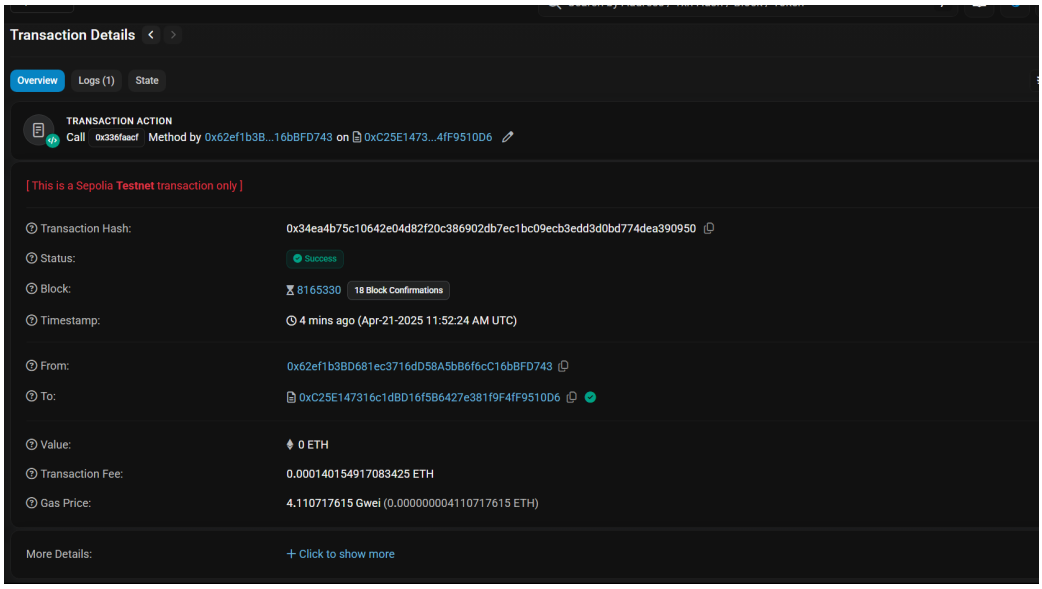


Figure 5.9: Transaction Record on Etherscan Sepolia Blockchain

Smart Contract Validation

Smart contract functions were thoroughly tested on the Sepolia testnet, validating investment recording, project registration, and rent distribution mechanisms. All test scenarios passed successfully, confirming the reliability of the blockchain implementation and the seamless integration with the Paymee payment gateway.

5.1.5 Retrospective

The blockchain integration sprint successfully delivered secure payment processing and transparent transaction recording. Table 5.6 summarizes the key achievements and future improvements.

Category	Details
What Went Well	• Successfully integrated Paymee payment gateway with blockchain • Achieved secure and transparent investment recording • Smart contracts deployed and tested successfully on Sepolia • Payment flow validation completed without major issues
Action Items	• Implement gas optimization for smart contract operations • Add automated monitoring for transaction failures

Table 5.6: Blockchain Integration Sprint Retrospective Summary

CHAPTER 6

Platform Operations

The best investment on earth is earth.

— Louis Glickman

Introduction

This chapter presents the core investment features that complement the AI capabilities and blockchain infrastructure of the Korpor platform. These features focus on enhancing the investor experience through comprehensive property management and portfolio tracking. The implementation covers property listing management and investment portfolio analytics for both web and mobile interfaces.

6.1 Sprint 8: Property Management

6.1.1 Introduction

The Property Management sprint focuses on establishing comprehensive property listing and management functionality. This foundation enables real estate agents and administrators to efficiently manage property portfolios, update listings, and maintain property information across the platform.

6.1.2 Analysis

Use Case Diagram

The property management system serves multiple actors within the Korpor ecosystem, including real estate agents, administrators, and property owners.

Textual Use Case Descriptions

Use Case	Manage Property Listings
Actor	Real Estate Agent, Administrator
Precondition	User is authenticated with appropriate permissions
Main Scenario	User creates, updates, or removes property listings

Table 6.1: Property Management Use Case Description

6.1.3 Modeling

Class Diagram

The property management system follows object-oriented design principles with entities for properties, listings, and management operations.

Sequence Diagram

The interaction sequence demonstrates the flow between user interface, backend services, and database operations for property management tasks.

6.1.4 Implementation

Property CRUD Operations

The implementation includes comprehensive Create, Read, Update, and Delete operations for property management with advanced filtering and search capabilities.

Media Management

Property images and documents are managed through secure upload and storage systems with thumbnail generation and compression optimization.

Property Status Tracking

Real-time tracking of property availability, investment status, and funding progress with automated status updates.

6.1.5 Test

Functional Testing

Comprehensive testing of all property management operations including creation, modification, and deletion workflows across different user roles.

Performance Testing

Load testing for property search and filtering operations to ensure optimal performance with large property datasets.

6.1.6 Retrospective

The Property Management sprint successfully delivered comprehensive property administration capabilities. Table **??** summarizes the key achievements and future improvements.

Category	Details
What Went Well	• Successfully implemented full CRUD operations for properties • Media management system works efficiently • Search and filtering provide excellent user experience • Role-based access control functions properly
Action Items	• Implement advanced property comparison features • Add bulk property import/export functionality • Enhance property analytics and reporting • Integrate with external property databases

Table 6.2: Property Management Sprint Retrospective Summary

6.2 Sprint 9: Investor Portfolio

6.2.1 Introduction

The Investor Portfolio sprint focuses on providing comprehensive investment tracking and portfolio management capabilities. This enables investors to monitor their investments, track performance, and analyze their real estate portfolio across multiple projects and properties.

6.2.2 Analysis

Use Case Diagram

The portfolio system serves investors by providing detailed analytics, performance tracking, and investment management tools.

Textual Use Case Descriptions

Use Case	**Track Investment Portfolio**
Actor	Investor
Precondition	Investor has active investments in the platform
Main Scenario	Investor views portfolio performance and investment analytics
Postcondition	Comprehensive portfolio insights are displayed

Table 6.3: Investor Portfolio Use Case Description

6.2.3 Modeling

Class Diagram

The portfolio system architecture includes entities for investments, performance metrics, and analytics calculations.

Sequence Diagram

The interaction flow demonstrates portfolio data retrieval, calculation processes, and visualization generation.

6.2.4 Implementation

Portfolio Analytics Engine

Advanced analytics engine calculating ROI, performance metrics, diversification indices, and risk assessments for investor portfolios.

Investment Tracking

Real-time tracking of investment performance, rental income distribution, and capital appreciation across all investor holdings.

Reporting and Visualization

Interactive charts and comprehensive reports providing insights into portfolio performance and investment trends.

6.2.5 Test

Analytics Validation

Comprehensive testing of portfolio calculations and performance metrics to ensure accuracy and reliability of financial data.

User Interface Testing

Extensive testing of portfolio dashboards and reporting features across web and mobile platforms.

6.2.6 Retrospective

The Investor Portfolio sprint successfully delivered comprehensive investment tracking capabilities. Table 6.4 summarizes the key achievements and future improvements.

Category	Details
What Went Well	• Successfully implemented portfolio analytics engine • Real-time investment tracking works accurately • Interactive visualizations provide excellent insights • Mobile portfolio interface is user-friendly
Action Items	• Add predictive portfolio modeling capabilities • Implement portfolio rebalancing recommendations • Enhance tax reporting and documentation • Integrate with external financial planning tools

Table 6.4: Investor Portfolio Sprint Retrospective Summary

CHAPTER 7

Conclusion & Future Work

The best way to predict the future is to create it.

— Abraham Lincoln

7.1 Summary of Achievements

7.2 Challenges Encountered

7.3 Future Improvements

7.4 Lessons Learned

Bibliography

[1] Franklin D. Roosevelt. *The Public Papers and Addresses of Franklin D. Roosevelt*. Vol. 7. Quoted in numerous real estate publications and historical archives. Random House, 1938.

[2] Assil Eye Institute. *Assil Eye Institute*. https://assileye.com/. Accessed: 2024. 2024.

[3] Stake. *Stake: The modern way for anyone to invest in real estate*. https://getstake.com/welcome. Accessed: 2024. 2024.

[4] Jeremy Moser. *How to Conduct a Competitive Analysis of Software Solutions*. G2 Learn. Jan. 2024. URL: https://learn.g2.com/software-competitive-analysis (visited on 04/14/2025).

[5] Team Asana. *How to create a competitive analysis (with examples)*. Asana. Feb. 2024. URL: https://asana.com/resources/competitive-analysis-example (visited on 04/18/2025).

[6] Kai Tomboc. *UX competitive analysis*. Lyssna Blog. Apr. 2024. URL: https://www.lyssna.com/blog/ux-competitive-analysis/ (visited on 05/02/2025).

[7] Ovidijus Jurevicius. *The Complete Guide to Competitive Analysis*. Strategic Management Insight. Apr. 2024. URL: https://strategicmanagementinsight.com/tools/competitive-analysis/ (visited on 05/24/2025).

[8] *Git*. Software Freedom Conservancy. URL: https://git-scm.com/ (visited on 03/20/2025).

[9] *GitHub*. GitHub, Inc. URL: https://github.com/ (visited on 03/20/2025).

[10] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. The official Scrum Guide, November 2020 version. Scrum.org. 2020. URL: https://scrumguides.org/scrum-guide.html (visited on 05/15/2025).

[11] Atlassian Team. *Three Pillars of Scrum: Understanding Scrum's Core Principles*. Atlassian Agile Coach. 2023. URL: https://www.atlassian.com/agile/project-management/3-pillars-scrum (visited on 04/20/2025).

[12] Alistair Cockburn. *Writing Effective Use Cases*. Boston, MA: Addison-Wesley Professional, 2002. ISBN: 978-0201702255.

[13] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley Professional, 2000. ISBN: 978-0201702255.

[14] Jennifer Davis and Ryn Daniels. *DevOps Foundations: Modern Software Development Practices*. O'Reilly Media, 2023. ISBN: 978-1492097136.

[15] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. 2nd ed. Addison-Wesley Professional, 2018. ISBN: 978-0134757599.

[16] Mary Poppendieck and Tom Poppendieck. "Lean Software Development: An Agile Toolkit". In: *Agile Software Development Series* (2012).

[17] Henrik Kniberg. *Lean from the Trenches: Managing Large-Scale Projects with Kan-*

ban. Pragmatic Bookshelf, 2013. ISBN: 978-1934356852.

[18] *Clerk Authentication Documentation*. Clerk, Inc. 2023. URL: https://clerk.com/docs (visited on 05/27/2025).

[19] OWASP Foundation. *OWASP Top Ten Security Principles*. Open Web Application Security Project. 2021. URL: https://owasp.org/www-project-top-ten/ (visited on 05/30/2025).

[20] Hui Wang, Yuming Chen, and Kaili Zhu. "Blockchain Applications in Real Estate: Current State and Future Prospects". In: *International Journal of Strategic Property Management* 27.1 (2023), pp. 32–49. DOI: 10.3846/ijspm.2023.18372.

[21] McKinsey & Company. *Blockchain in Real Estate: Transforming Property Transactions*. McKinsey & Company. 2023. URL: https://www.mckinsey.com/industries/real-estate/our-insights/blockchain-in-real-estate-transforming-property-transactions (visited on 04/22/2025).

[22] Gene Kim et al. "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations". In: *IT Revolution Press* (2018).

[23] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. 4th ed. Addison-Wesley Professional, 2021. ISBN: 978-0136886099.

[24] Docker Documentation Team. *Docker Architecture and Container Design*. Docker, Inc. 2023. URL: https://docs.docker.com/get-started/overview/ (visited on 05/27/2025).

[25] Nicole Forsgren and Jez Humble. *DevOps Metrics That Matter: Measuring Success in 2023*. Google Cloud. 2023. URL: https://cloud.google.com/blog/products/devops-sre/the-2023-accelerate-state-of-devops-report-now-out (visited on 04/22/2025).

[26] Andri Sunardi and Suharjito. "MVC Architecture : A Comparative Study Between Laravel Framework and Slim Framework in Freelancer Project Monitoring System Web Based". In: *Procedia Computer Science* 157 (2019), pp. 134–141. DOI: 10.1016/j.procs.2019.08.150. URL: https://doi.org/10.1016/j.procs.2019.08.150.

[27] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994. ISBN: 978-0201633610.

[28] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. 2nd ed. Addison-Wesley Professional, 2004. ISBN: 978-0321278654.

[29] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017. ISBN: 978-0134494166.

[30] *TypeScript Documentation*. Microsoft Corporation. 2023. URL: https://www.typescriptlang.org/docs/ (visited on 05/30/2025).

[31] *TanStack - High-quality open-source software for web development*. TanStack. 2023. URL: https://tanstack.com/ (visited on 05/30/2025).

[32] *Maestro Documentation*. Maestro Labs, Inc. 2022. URL: https://maestro.mobile.dev/ (visited on 05/30/2025).

[33] *Express - Node.js web application framework*. Node.js Foundation. URL: https://expressjs.com/ (visited on 05/23/2025).

[34] *Node.js*. OpenJS Foundation. URL: https://nodejs.org/en (visited on 05/18/2025).

[35] *Postman*. Postman, Inc. URL: https://www.postman.com/ (visited on 03/20/2025).

[36] *Vite*. Evan You and Vite contributors. URL: https://vitejs.dev/guide/ (visited on 05/23/2025).

[37] *React*. Meta Platforms, Inc. URL: https://react.dev/ (visited on 04/16/2025).

[38] *MySQL Documentation*. Oracle Corporation. URL: https://dev.mysql.com/doc/ (visited on 05/23/2025).

[39] *Playwright End-to-End Testing Documentation*. Microsoft Corporation. 2023. URL: https://playwright.dev/docs/intro (visited on 05/30/2025).

[40] Ethan Marcotte. *Responsive Web Design: Patterns and Principles for the Modern Web*. 3rd ed. A Book Apart, 2023. ISBN: 978-1937557522.

[41] *StarUML*. MKLabs Co., Ltd. URL: https://staruml.io/ (visited on 04/18/2025).

[42] Jeffrey Schwarz. *The Scrum Product Backlog: A Guide for Product Owners*. Scrum.org. 2019. URL: https://www.scrum.org/resources/blog/scrum-product-backlog-guide-product-owners (visited on 05/30/2025).

[43] Dai Clegg and Richard Barker. *The MoSCoW Method: A Prioritization Technique for Project Management*. ProductPlan. 2021. URL: https://www.productplan.com/glossary/moscow-prioritization/ (visited on 05/27/2025).

[44] Dai Clegg and Richard Barker. *Case Method Fast-Track: A RAD Approach*. Addison-Wesley, 2004. ISBN: 978-0201624328.

[45] Jeff Sutherland and J.J. Sutherland. *Scrum: The Art of Doing Twice the Work in Half the Time*. Currency, 2020. ISBN: 978-0385346474.

[46] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional, 2012. ISBN: 978-0137043293.

[47] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, 2005. ISBN: 978-0131479418.

[48] James Grenning. "Planning Poker or How to avoid analysis paralysis while release planning". In: *Renaissance Software Consulting* 3 (2002). URL: https://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf.

[49] Scaled Agile, Inc. *SAFe for Lean Enterprises 6.0*. Scaled Agile, Inc. 2024. URL: https://www.scaledagileframework.com/ (visited on 04/22/2025).

[50] Vitalik Buterin. *Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform*. 2014. URL: https://ethereum.org/en/whitepaper/ (visited on 05/12/2025).

[51] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (2008). URL: https://bitcoin.org/bitcoin.pdf (visited on 05/10/2025).

[52] Don Tapscott and Alex Tapscott. *Blockchain Revolution: How the Technol-*

ogy Behind Bitcoin Is Changing Money, Business, and the World. Portfolio, 2016. ISBN: 978-1101980132.

[53] Zibin Zheng et al. "Blockchain challenges and opportunities: a survey". In: *International Journal of Web and Grid Services* 14.4 (2018), pp. 352–375. DOI: 10.1504/IJWGS.2018.095647.

[54] Andreas M. Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, 2018. ISBN: 978-1491971949.

[55] Massimo Bartoletti and Livio Pompianu. "An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns". In: *Financial Cryptography and Data Security*. Springer, 2017, pp. 494–509. DOI: 10.1007/978-3-319-70278-0_31.

[56] Xiwei Xu et al. "A Taxonomy of Blockchain-Based Systems for Architecture Design". In: *IEEE International Conference on Software Architecture (ICSA)*. 2019, pp. 243–252. DOI: 10.1109/ICSA.2017.33.

[57] *Solidity Documentation*. Ethereum Foundation. 2023. URL: https://docs.soliditylang.org/ (visited on 05/15/2025).

[58] *ethers.js Documentation*. ethers. 2023. URL: https://docs.ethers.org/ (visited on 05/20/2025).

[59] *Infura Web3 API Documentation*. Consensys. 2023. URL: https://docs.infura.io/ (visited on 05/22/2025).

[60] *MetaMask Documentation*. ConsenSys. 2023. URL: https://docs.metamask.io/ (visited on 05/20/2025).

[61] Seyed Mojtaba Hosseini Bamakan, Amir Motavali, and Ali Babaei Bondarti. "A survey of blockchain consensus algorithms performance evaluation criteria". In: *Expert Systems with Applications* 154 (2020), p. 113385. DOI: 10.1016/j.eswa.2020.113385.

[62] Nick Szabo. "Formalizing and Securing Relationships on Public Networks". In: *First Monday* 2.9 (1997). URL: https://firstmonday.org/ojs/index.php/fm/article/view/548.

[63] Maximilian Wöhrer and Uwe Zdun. "Smart contracts: security patterns in the ethereum ecosystem and solidity". In: (2018), pp. 2–8. DOI: 10.1109/IWBOSE.2018.8327565.

[64] Chris Dannen. *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Apress, 2017. ISBN: 978-1484225349.