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 : LINFO**

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 dedicated to real estate investment

Defended on *** in front of the jury composed of:

Mr :	**President**
Mr :	**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
Prof. [Name] & **Dr. [Name]** — 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
May 8, 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. Investors can manage their portfolios effortlessly through a comprehensive dashboard, with real-time notifications, an interactive AI chatbot, and multi-language support delivering a personalized and globally accessible experience.

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 in a dynamic market environment.

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

- The first chapter, **Project Context**, delves into the industry challenges and the vision that inspired Korpor's creation.
- The second chapter, **Analysis and Specification of Needs**, outlines the comprehensive requirements gathering, needs 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—each developed through clearly defined sprints encompassing analysis, design, and deployment phases.

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"

"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 Service", 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" and "Stake" for in-depth analysis [2, 3] due to their exceptional performance and status as leading companies in the real estate investment platform sector.

The Aseel Platform

Aseel 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.

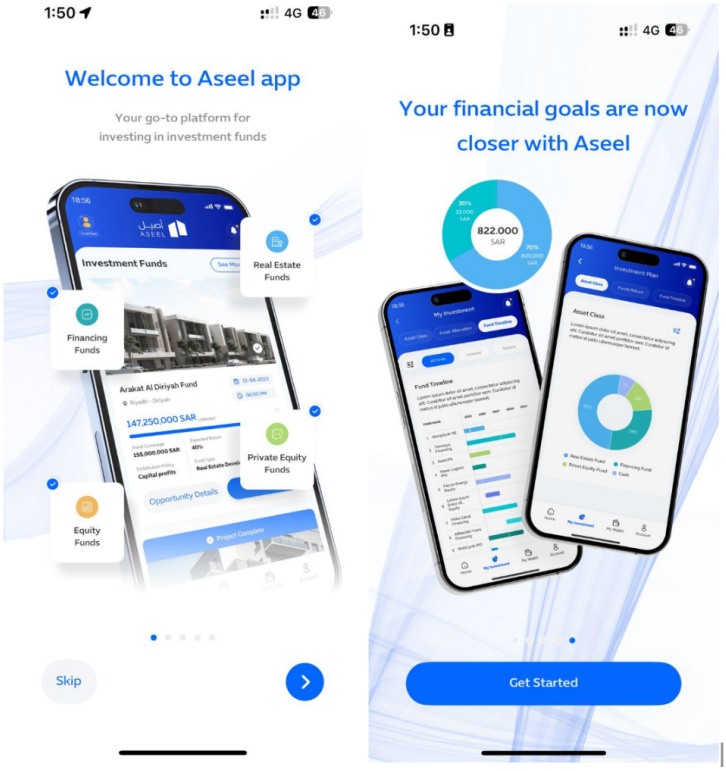


Figure 1.2: Interface of "The Aseel Platform"

The Stake Platform

Stake 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.

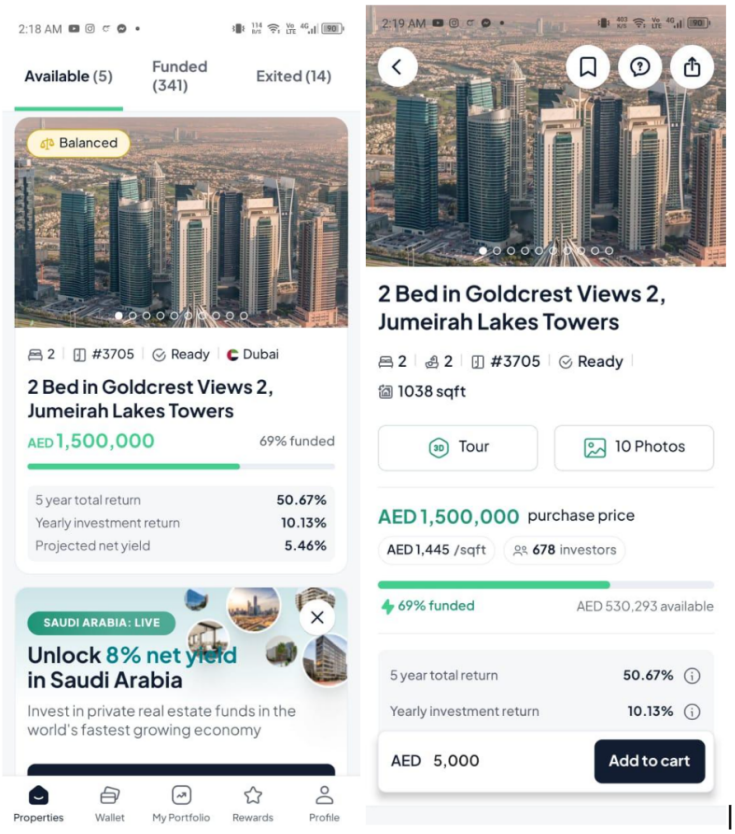


Figure 1.3: Interface of "The Stake Platform"

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 [4, 5].

- **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 1.1 presents the evaluation of the existing solutions based on these criteria:

Table 1.1: Evaluation Table

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

1.3.3 Proposed Solution

Having studied the already working platforms for investments, we found strengths and weaknesses that could define what was required from the project.

The **Korpor** platform will be offering the following features:

A listing of investment opportunities with deep financial insights into those opportunities and the abiltie of investing in fractions of property ownership.

- AI-driven recommendations of investments as per users' preferences [6].
- Smooth funding and payout mechanisms.
- Real-time portfolio performance tracking on a single screen/dashboard.
- Forum for interactive discussions on strategy and market trends among its users.
- Referral and Rewards System: An engaging system for rewarding users via referral.

Our proposed solution will look at:

- Developing an efficient mobile application for investment management.
- Increasing the level of users engagement with recommendations using the power of AI [7, 8, 9].
- Ensuring responsive and user-friendly interaction with the interface.
- Gaining the trust of investors by ensuring transparency and security in the investing platform.
- Enhancing the security of data and following all the financial regulations.

1.4 Development methodology

The completion of the project on its delivery date is the main problem of every software development team. One of the most common problems encountered in the production of software is insufficient technical specifications, poor time management in the face of the use of emerging technology, and sudden changes in needs. In order to avoid these critical issues, we follow an agile methodology for project management, using tools like Git [10] for version control and GitHub [11] 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 [12]. Scrum is founded on empiricism and lean thinking, employing an iterative, incremental approach guided by the three pillars of transparency, inspection, and adaptation [13, 12]. 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 [12]. 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).

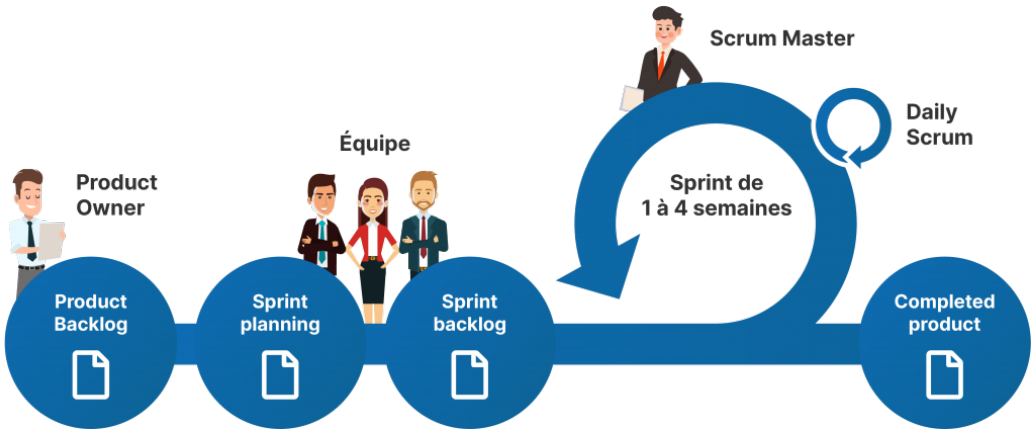


Figure 1.4: Agile Scrum Framework Process

1.4.2 Agile Scrum roles and responsibilities

The Product Owner

Understands the customer and business requirements, then creates and manages the product backlog based on those requirements.

Responsibilities:

- Managing the scrum backlog
- Release management
- Stakeholder management

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.

Responsibilities:

- Delivering the work through the sprint
- To ensure transparency during the sprint, they meet daily at the daily scrum

Scrum Master

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:

- Transparency
- Empiricism
- Self-organization
- The Scrum events

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 [12]. 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 needs developed in our project.

CHAPTER 2

Analysis and Specification of Requirements

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 the diagram of the global use cases. Then we will present our project architecture and our working environment, and finally we will present our product backlog and releases 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 [14, 15]. 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.
- **System**: The entity that automatically manages all basic functionalities, such as authentication, notification generation, transaction validation, and adherence to security protocols. It ensures the coherence and reliability of the application at all times.

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.
- **Configure Platform Features**: Define key parameters (payment methods, AI/blockchain integrations, etc.) and roll out feature updates.
- **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.

For the System

- **Automate Authentication**: Validate credentials, manage sessions, and maintain user roles and permissions.
- **Generate Notifications**: Send real-time alerts (e.g., new listings, completed transactions, commission updates) to relevant users.
- **Ensure Compliance & Security**: Leverage blockchain for data integrity, verify payments, and detect anomalies or fraudulent activities.
- **Coordinate AI Insights**: Aggregate and analyze real estate data to produce market predictions and price recommendations.
- **Maintain Transaction Consistency**: Update dashboards, user balances, and property statuses automatically upon each operation.
- **Optimize Performance**: Monitor server load, scale resources, and ensure a smooth, responsive application experience.

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 needs such as:

- **Maintainability**: The system must be designed for simplicity so that tasks, updates, and bug fixes can be executed with minimal complexity [16, 17].
- **Evolution**: Platform administration must remain attentive to user needs and feedback, continuously enhancing the services offered while preserving the application's utility and efficiency [18, 19].

- **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) [20, 21]. The integration of blockchain technology further ensures the immutability and integrity of sensitive information [22, 23].
- **Efficiency**: The application must be effective in all circumstances, delivering prompt and reliable functionality regardless of external conditions [24, 25].
- **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 [26, 27].

2.2 Requirements Analysis

In this section, we'll outline the various features that our app should offer, using a general use case diagram [14].

2.2.1 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:

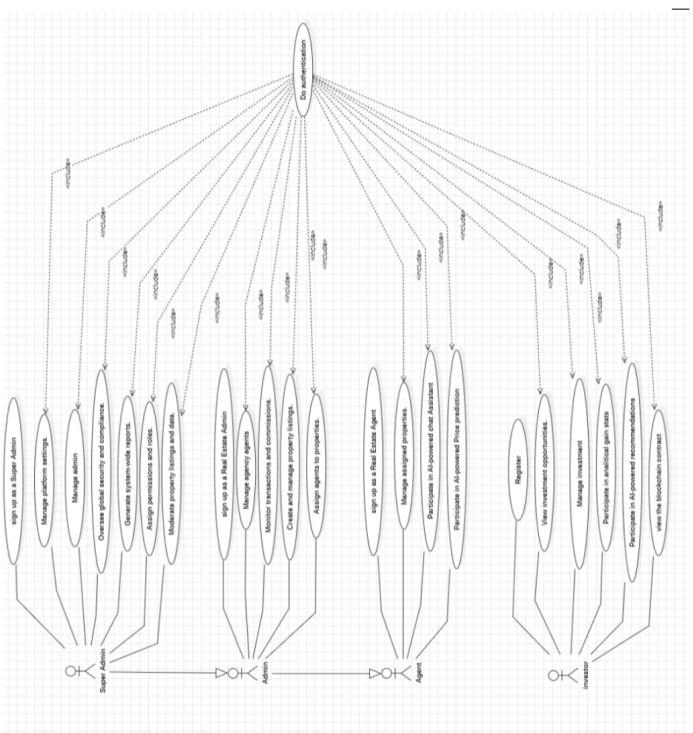


Figure 2.1: General use case diagram

2.3 Software architecture

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

2.3.1 Physical architecture

The physical architecture of Korpor leverages modern, scalable technologies to deliver a seamless investment platform. The frontend is built with Expo, React, and TypeScript using Vite [28] for rapid development and Tanstack for robust state management and data visualization, while Storybook supports isolated UI component development [29]. The backend relies on Express.js [30] with user authentication managed by Clerk [20], containerization via Docker [26], and MySQL [31] for data storage hosted on Microsoft Azure [32]. Integrated AI modules provide personalized insights [33], and blockchain technology ensures transactional security and data immutability [22, 23]. This setup is further supported by GitHub [11] for version control, Postman [34] for API testing, and end-to-end testing tools like Maestro and Playwright [35], with architectural designs and documentation maintained using StarUML [36] and Overleaf.

Figure 2.2: Deployment diagram

2.3.2 Logical architecture

To better manage code organization and ensure maintainability, we've designed our application based on the MVC (Model-View-Controller) [37, 38] architectural pattern:

- **Model**: Represents application data and business logic.
- **View**: Presents data to users through interfaces.
- **Controller**: Processes incoming requests, performs operations using Models, and returns appropriate Views.

Architecture Components:

- **Model:** The MySQL database serves as the core data source, responsible for storing and managing application data, including user profiles, real estate listings, transactions, and investment records. The data layer interacts with the backend through ORM or query-based operations, ensuring efficient data retrieval and persistence.

- **Controller:** The Express.js backend acts as the intermediary between the database and the frontend, handling user requests, business logic, and data validation. It processes API calls from the frontend and interacts with services such as Clerk for authentication, AI modules for predictive analytics, and blockchain integration for secure transactions.
- **View:** The frontend is built with React, TypeScript, and TanStack tools, ensuring a responsive and interactive UI. The frontend communicates with the backend via API requests, displaying dynamic content and allowing seamless user interaction.

The logical architecture is illustrated in Figure 2.3.

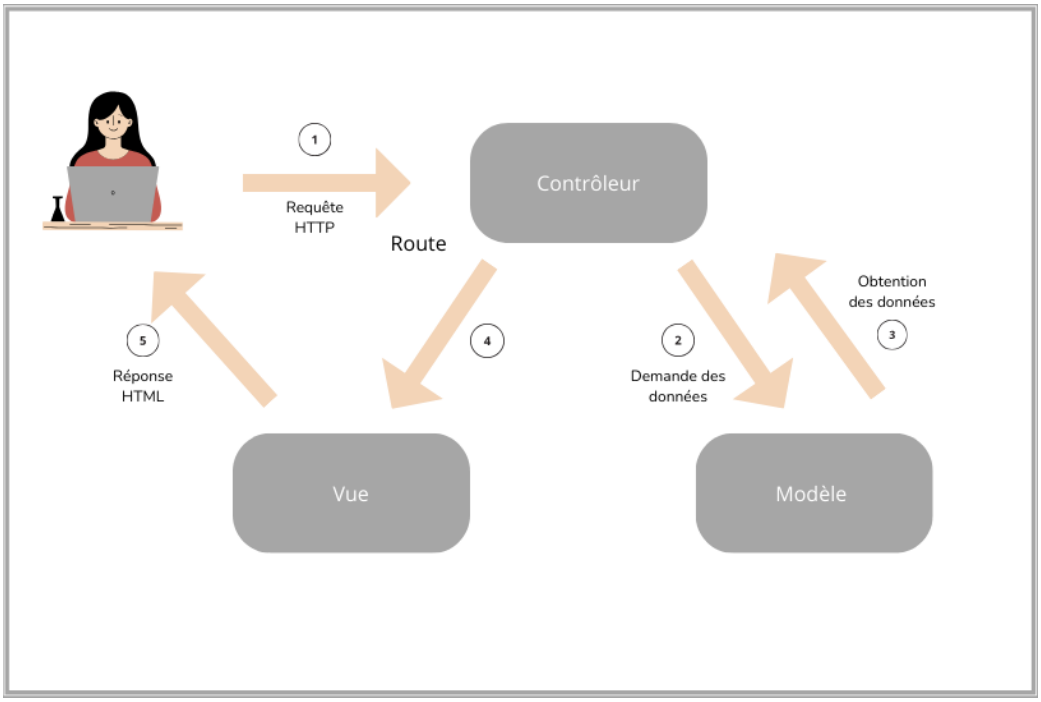


Figure 2.3: Logical architecture

Request Flow:

1. A user action (e.g., viewing property listings) triggers a request in the frontend.
2. The request is sent to the backend via an API call.
3. The Express.js controller processes the request, interacting with the database and other services.
4. Data is retrieved, processed, and returned as a response.
5. The frontend updates the UI dynamically based on the received data.

This structured approach ensures a scalable, secure, and high-performance system, optimizing Korpor's real estate investment and management operations [24].

2.4 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 [39, 40].

2.4.1 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

2.4.2 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 [41]. It is designed for the development of large applications and transpiles to JavaScript.

Tanstack

High-quality open-source software for web developers [42]. Headless, type-safe, & powerful utilities for Web Applications, Routing, State Management, Data Visualization, Datagrids/Tables, and more.

Clerk

Clerk [20] is a complete suite of embeddable UIs, flexible APIs, and admin dashboards to authenticate and manage your users.

Maestro

Maestro [43] 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 [11] 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 [30] is a minimal and flexible Node.js [44] web application framework that provides a list of features for building web and mobile applications easily.

Postman

Postman [34] 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 [28] 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 [45], sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

MySQL

MySQL [31] 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 [26]. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

Playright

Playwright [35] 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 [16].

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 [46].

StarUML

StarUML [36] 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 [44] 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.

2.4.3 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.

2.4.4 Source code management with Git and GitHub

We used GitHub [11] for storing our application code. We set up a repository with a structured branching strategy to ensure organized development.

The main branch maintains the history of official releases, while the develop branch is used for integrating new features. This approach allows us to maintain a stable production version while simultaneously developing new functionality.

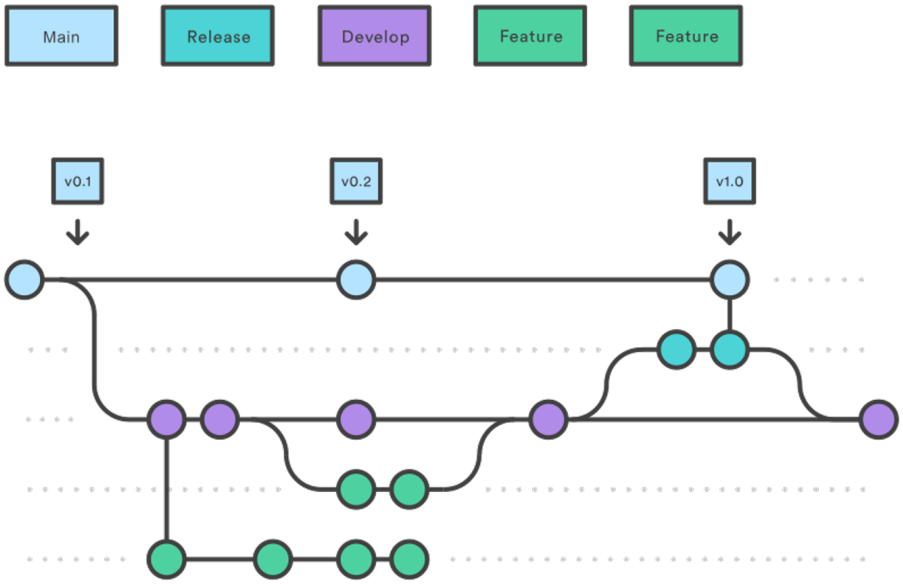


Figure 2.4: Git Workflow

For each new feature or bug fix, we create a feature branch from develop. Once the work is completed and tested, it is merged back into develop through a pull request process. This ensures code review and quality control before any changes are integrated. When a release is ready, develop is merged into main, creating a new stable version of the application.

2.5 Product backlog

The backlog was created before the sprints to plan the milestones and determine the content of each sprint based on project needs [**ProductBacklogGuide2024**, 47]. 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 [48, 49].
  - **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.
- **Won't:** The feature is not a priority at this time.

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 information	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	Configuration	As a Super Admin, I want to configure platform-wide features and settings	Could
PB007	Analytics	As a Super Admin, I want to generate and analyze global performance reports	Could
PB008	Moderation	As a Super Admin, I want to moderate content across the platform	Won't
PB009	AI Integration	As a Super Admin, I want to chat with an AI assistant that can securely access database information	Could
Admin Features			
PB010	Listing Management	As an Admin, I want to manage real estate listings in my company	Must
PB011	Agent Management	As an Admin, I want to oversee real estate agents and their permissions	Should
PB012	Transaction Management	As an Admin, I want to track transactions and calculate agent commissions	Should
PB013	Customer Service	As an Admin, I want to address investor inquiries and issues	Could

Continued on next page

PB014	Analytics	As an Admin, I want to access a comprehensive agency dashboard	Could
PB015	AI Integration	As an Admin, I want to input property details and receive AI-powered valuation	Should
Real Estate Agent Features			
PB016	Listing Management	As an Agent, I want to create and manage property listings	Must
PB017	Investment Management	As an Agent, I want to handle investment and purchase requests	Could
PB018	Data Management	As an Agent, I want to contribute data for AI-driven estimates	Could
PB019	Customer Relations	As an Agent, I want to maintain client relationships and communications	Could
PB020	Finance	As an Agent, I want to view my commissions on sales and rentals	Should
Investor Features			
PB021	Property Discovery	As an Investor, I want to browse available property listings	Must
PB022	Search Functionality	As an Investor, I want to filter properties based on my preferences	Could
PB023	Investment Process	As an Investor, I want to invest in properties through a simple process	Could
PB024	Portfolio Management	As an Investor, I want to track my investment portfolio in real-time	Must
PB025	Payment Processing	As an Investor, I want to make secure payments through various methods	Should
PB026	AI Recommendations	As an Investor, I want to receive personalized property recommendations	Must
PB027	AI Assistance	As an Investor, I want to consult an AI assistant for real estate legal questions	Could
PB028	Financial Prediction	As an Investor, I want to see predictions of potential earnings	Could
PB029	Finance Management	As an Investor, I want to manage my earnings and withdrawals	Could
AI & Machine Learning Features			

Continued on next page

PB030	AI System	As the System, I want to analyze user interactions for personalized recommendations	Must
PB031	AI Prediction	As the System, I want to predict property valuations and rental prices	Should
PB032	AI Forecasting	As the System, I want to forecast potential investment returns	Should
PB033	NLP Integration	As the System, I want to provide real estate legal information via NLP	Could
PB034	Security	As the System, I want to secure AI database access for authorized queries	Could
Blockchain & Smart Contract Features			
PB035	Blockchain	As the System, I want to create and manage virtual contracts for transactions [22]	Must
PB036	Blockchain	As an Investor, I want my property investments to be secured via blockchain [23]	Must
PB037	Blockchain Management	As an Admin, I want to verify and validate blockchain transactions	Should
PB038	Data Integrity	As the System, I want to store transaction records immutably on blockchain	Must
PB039	System Monitoring	As a Super Admin, I want to monitor blockchain health and performance	Should
System & Security Features			
PB040	Authentication	As the System, I want to automate authentication and session management	Should
PB041	Notifications	As the System, I want to generate real-time notifications for relevant users	Could
PB042	Data Consistency	As the System, I want to maintain transaction consistency across the platform	Should
PB043	Security	As the System, I want to ensure secure communication between AI and database	Should
DevOps & Infrastructure			
PB044	CI/CD	As a Developer, I want CI/CD pipelines to build project images on GitHub [27]	Must
PB045	Deployment	As a Developer, I want to automate image deployment to GCP registry	Must

Continued on next page

PB046	Deployment	As a Developer, I want to auto-deploy back-end services and database	Must
PB047	Containerization	As a Developer, I want to containerize application components with Docker	Must
PB048	Web Deployment	As a Developer, I want to automatically deploy the web app frontend	Should
PB049	Mobile Deployment	As a Developer, I want to automate mobile app deployments to Play Store	Should
PB050	Versioning	As a Developer, I want to implement versioning for mobile app releases	Should
PB051	Monitoring	As an Admin, I want to monitor system health and performance	Could
PB052	Configuration	As a Super Admin, I want to manage environment configurations	Could
Mobile App Specific Features			
PB053	Mobile UI/UX	As an Investor, I want a responsive, intuitive mobile interface [50]	Should
PB054	Notifications	As an Investor, I want to receive push notifications about my investments	Should
PB055	Offline Access	As an Investor, I want offline access to my basic portfolio information	Should

2.6 Sprint planning

In order to complete the project within the deadlines set by the internship agreement, planning is an important step in the process [51, 52]. 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 [53, 54]. Figure 2.5 shows the Gantt chart that describes the progress of our project:

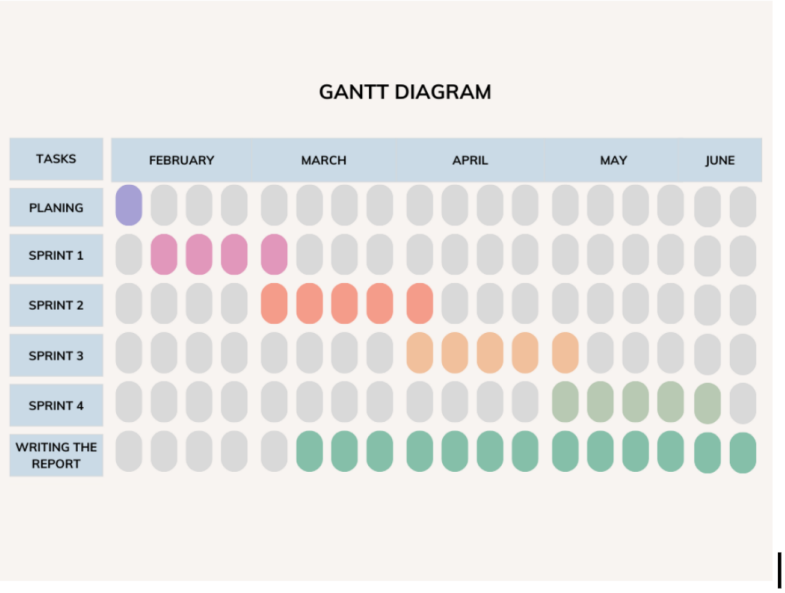


Figure 2.5: GANTT chart with sprint planning

Conclusion

Our Sprint 0 marked the exciting start of our KORPOR project [55, 51]. 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 [47, 52].

CHAPTER 3

Foundation

Introduction

This chapter details the implementation phase of our project, which follows an agile methodology with four sprints. Each sprint focuses on delivering specific features and functionality according to the project backlog. The implementation utilizes a modern tech stack consisting of React [45] with Vite [28] for the frontend, Node.js [44] with Express.js [30] for the backend, MySQL [31] for database management, and Tailwind CSS [56] for styling.

Our first sprint focused on establishing essential foundational components of the system, with the following key deliverables:

3.0.1 Web Backoffice Authentication

- Admin dashboard login and authentication system
- Role-based access control for backoffice users (Super Admin, Admin, Agent)
- User management interface for creating and managing user accounts
- Permission management system for different user roles
- Security logs and audit trails for backoffice activities
- Session management and secure token handling

3.1 Sprint 1: Authentication and User Management

3.1.1 Overview

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.

3.1.2 User Types

The system supports three distinct user types, each with different permissions and capabilities:

- **Super Admin**: Has complete access to all system features and can manage admins and agents.
- **Admin**: Can manage agents and has access to administrative features within their assigned scope.
- **Agent**: Has limited access to the system based on their assigned responsibilities.

3.1.3 Authentication System

Sign-up Process

The sign-up process is illustrated in Figure 3.1 below. The diagram shows the authentication flow for new users registering in the system.

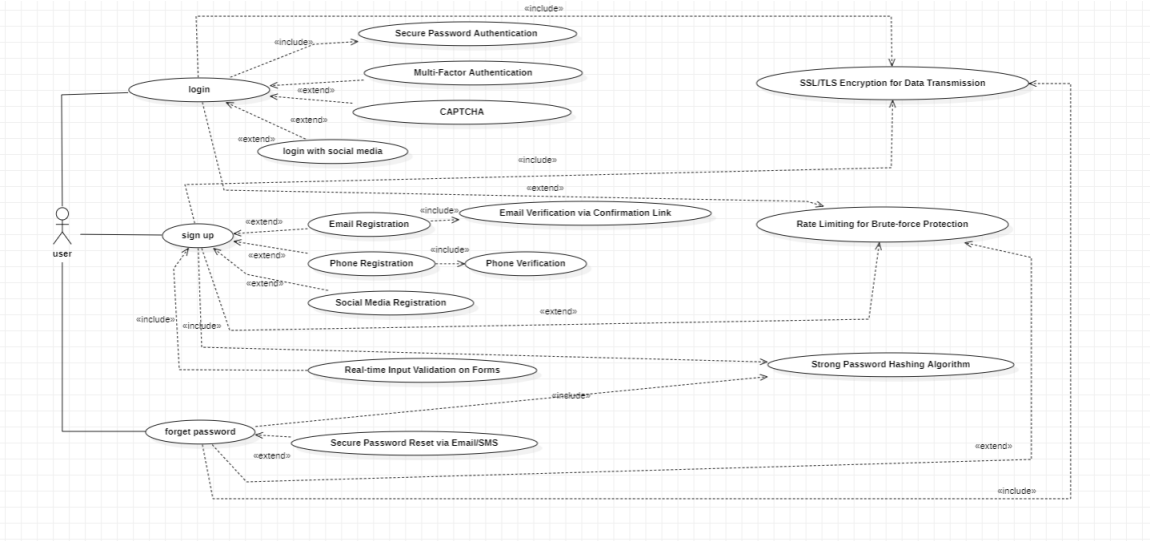


Figure 3.1: Authentication Sign-up Use Case Diagram

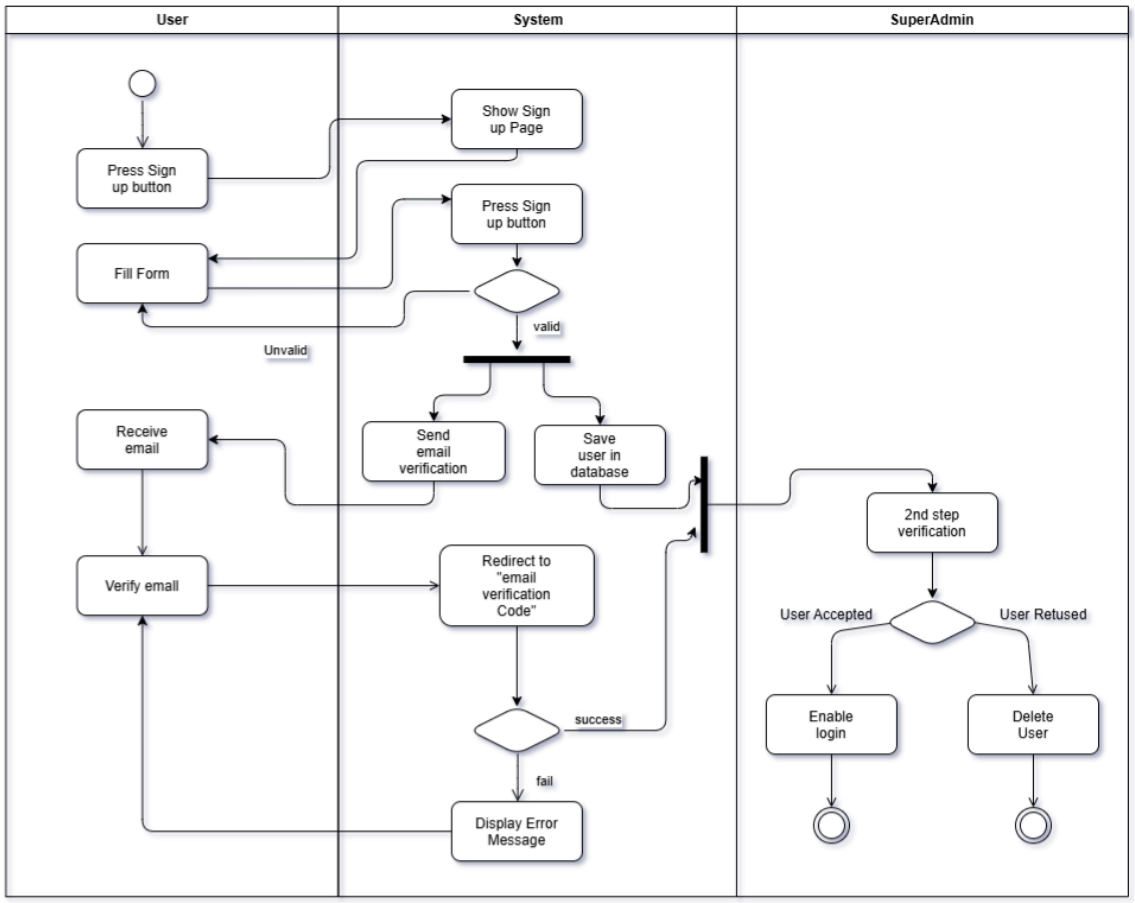


Figure 3.2: Authentication Sign-up Activity Diagram

Figure 3.2 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.

Login Process

The login process allows existing users to access the system. The table below details the use case.

Section	Details
Use Case	User Login
Actor	User (Super Admin, Admin, Agent)
Precondition	User has an existing, verified account. User is on the login page.
Main Scenario	1. User enters their credentials (e.g., email and password). 2. User clicks the login button. 3. System verifies the credentials. 4. If credentials are valid, the system grants access and redirects the user to their respective dashboard.
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.

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.

Section	Details
Use Case	Manage User Accounts
Actor	Super Admin
Precondition	Actor is logged into the system with appropriate administrative privileges.
Main Scenario	1. Actor navigates to the user management section. 2. To create a user: Actor fills in user details (name, email, role, etc.) and submits the form. System creates the new user account. 3. To view users: System displays a list of users. Actor can search/filter the list. 4. To update a user: Actor selects a user, modifies their details, and saves the changes. System updates the user account. 5. To delete a user: Actor selects a user and confirms deletion. System deactivates or deletes the user account.
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

Figure 3.3 depicts the class diagram for the authentication system. It showcases the key components and their relationships involved in user authentication and authorization.

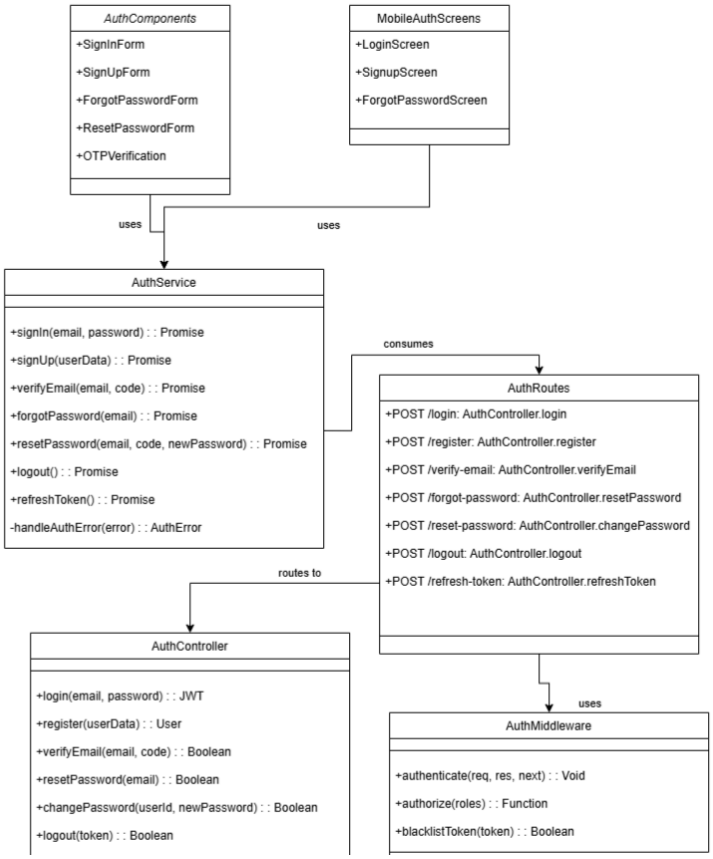
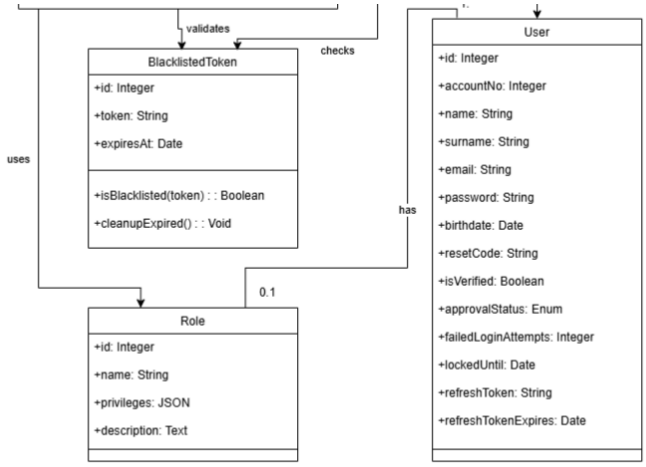


Figure 3.3: Authentication System Class Diagram



- **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.

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 needs 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 AI Development Overview

4.1.1 Technology Stack

4.1.2 Data Preprocessing Pipeline

Feature Engineering

Dataset Creation

4.1.3 Knowledge Base Construction

Information Extraction

Knowledge Organization

4.2 Data Collection and Scraping

4.2.1 Overview

This section details the data collection processes implemented to gather the real estate market data required for our AI models. Effective data acquisition is a critical foundation for the AI capabilities implemented in our platform.

4.2.2 Real Estate Data Scraping

Data Sources

We implemented automated scraping systems to collect real estate data from various sources:

- Real estate listing websites such as PropertyStar Tunisia [57] for comprehensive property listings and pricing data
- Property transaction records from public databases
- Real estate market reports and analytics platforms
- RE/MAX Tunisia [58] for detailed property valuation data in both sale and rental markets
- Al-Mindhar [59] for blog content related to legal aspects, investment strategies, and real estate regulations in Tunisia to train knowledge-based AI assistants

Scraping Architecture

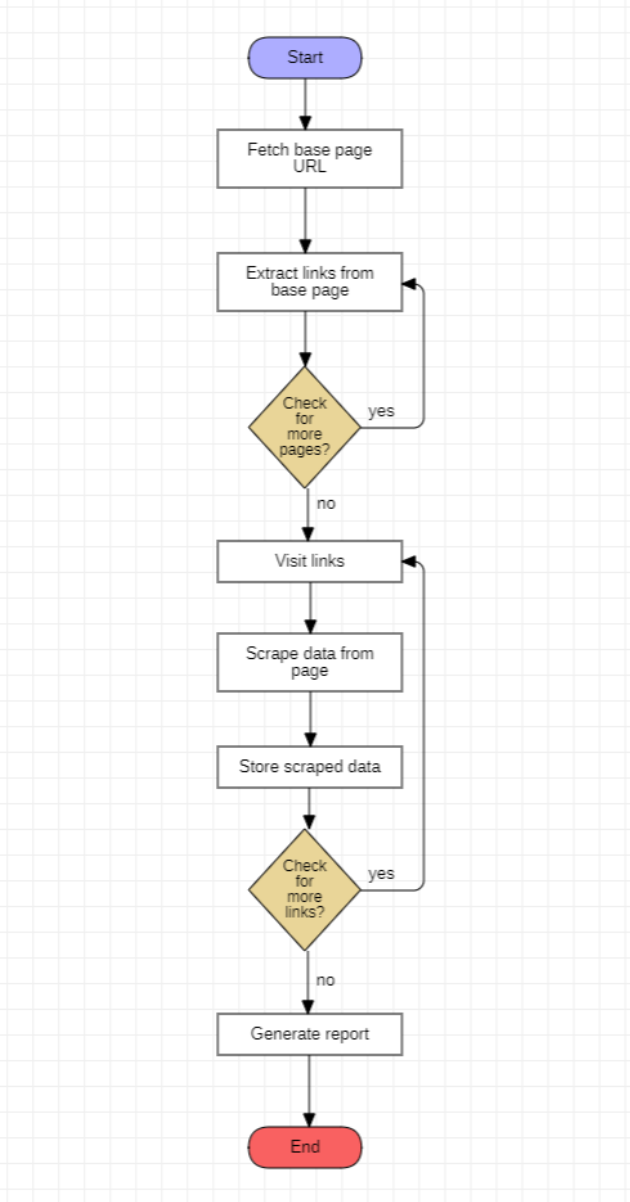


Figure 4.1: Data Scraping workchart

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

- Rate-limiting and request throttling to respect website policies
- Scheduled jobs for regular data updates
- Data validation and cleaning pipelines

4.2.3 Data Storage and Management

Database Schema

The collected data is stored in a structured database with the following key entities:

- Property listings (with pricing history)
- Location data (neighborhoods, cities, regions)
- Property features and amenities
- Transaction records and market trends

Data Versioning and Updates

To maintain data quality and currency, we implemented:

- Automated data refresh cycles for each source
- Version control for dataset updates
- Conflict resolution for data from multiple sources
- Data quality monitoring and alerting

4.3 Property Valuation Prediction Model

4.3.1 Overview and Objectives

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.

4.3.2 Model Architecture and Training Process

Model Selection

After evaluating multiple approaches, we implemented:

Training Methodology

4.3.3 Prediction Capabilities

4.3.4 Performance Metrics and Evaluation

4.3.5 Integration with Property Listings

CHAPTER 5

Blockchain & Backoffice Features

Introduction

In recent years, blockchain technology has emerged as a powerful tool for building trust, enhancing transparency, and decentralizing processes across various industries [60, 61]. In the context of Korpor, a platform aimed at facilitating real estate investments, blockchain plays a crucial role in ensuring the reliability and traceability of financial operations.

Traditional real estate investment systems often rely on centralized authorities and intermediaries, which may introduce delays, additional costs, or even fraudulent behavior. By integrating blockchain, Korpor addresses these limitations by recording key transactions such as investments, rent distributions, and project creation directly on a decentralized ledger [62]. This ensures that all stakeholders can verify actions in a transparent and immutable manner, without needing to trust a single party.

Moreover, smart contracts enable the automation of operations like rent distribution and investment validation, reducing human error and increasing efficiency [63, 64].

5.1 Overview of Use Cases

The blockchain integration within the **Korpor** platform addresses several core functionalities that benefit from decentralization and transparency. The main use cases implemented include:

- **Investment Recording**
  - Immutable proof of ownership for the investor.
  - Transparent public records of all investment flows.
  - Verifiable data for audits or external reviews.
- **Project Registration**

  - Proof of project existence on-chain.
  - Traceability of project updates and funding status.
  - Protection against unauthorized modifications.

- **Rent Distribution**
  - Calculate and distribute rental income fairly to investors.
  - Automate distribution without manual intervention.
  - Log each rent payout on the blockchain for transparency.

- **Transaction Verification**
  - Trust in the platform.
  - User empowerment to track their funds and participation.
  - Regulatory compliance through transparent financial trails.

5.2 Objectives of Blockchain Integration

The integration of blockchain technology into the **Korpor** platform serves several key objectives aimed at enhancing the system's reliability, transparency, and automation capabilities:

- **Security and trust in transactions**
  - Ensures that each transaction is securely signed and verified.
  - Builds trust among users by preventing unauthorized modifications [65].
  - Enhances transparency through verifiable on-chain records.

- **Decentralization and immutability**
  - Eliminates single points of failure by distributing data across the blockchain.
  - Guarantees that data, once written, cannot be altered [66].
  - Promotes resilience and long-term reliability of the system.

- **Automation through smart contracts**
  - Enables automatic execution of business logic (e.g., rent distribution) [67].
  - Reduces manual intervention and human error.
  - Increases system efficiency through self-executing contracts.

5.3 Technical Architecture

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 [68].
- **Technologies used**
  - **Solidity:** A high-level programming language specifically designed for writing secure, efficient, and decentralized smart contracts on the Ethereum blockchain [69].
  - **Ethers.js:** A popular JavaScript library used to interact with the Ethereum blockchain, providing functionality to manage accounts, send transactions, and call smart contract functions [70].
  - **Sepolia Testnet:** A test network for Ethereum that allows developers to deploy and test their smart contracts in a safe, controlled environment without spending real cryptocurrency.

- **Infura:** A cloud-based platform that provides access to Ethereum nodes via JSON-RPC, eliminating the need to run a full Ethereum node [71].
- **MetaMask:** A widely-used browser extension and mobile application that serves as a cryptocurrency wallet for interacting with decentralized applications [72].

5.3.1 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.

Figure 5.1: Paymee Framework

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 [73].

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 needs of individual investors.

Table 5.1: 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.

5.4 Smart Contract Design

5.4.1 Smart Contracts Overview

A smart contract is a self-executing contract with the terms of the agreement directly written into lines of code [64]. 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 [74].

5.4.2 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.

5.4.3 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 [75].

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 holds information about each investor’s investment:

```
struct Investor {
    address investorAddress;
    uint256 amountInvested;
    bool hasReceivedRent;
}
```

Core Functions

Several key functions implement the core logic of the platform:

recordInvestment:

```
function recordInvestment(uint256 projectId) public payable {
    require(msg.value > 0, "Investment amount must be greater than zero.");
    projects[projectId].totalFunding += msg.value;
    investments[msg.sender] += msg.value;
    projects[projectId].numInvestors++;
}
```

recordProject:

```
function recordProject(string memory name, address company) public {
    uint256 projectId = projectCounter++;
    projects[projectId] = Project(projectId, name, company, 0, 0, 0);
}
```

distributeRent:

```
function distributeRent(uint256 projectId) public {
    uint256 rentAmount = projects[projectId].rentIncome / projects[projectId].numInve
    for (uint256 i = 0; i < projects[projectId].numInvestors; i++) {
        address investor = projects[projectId].investors[i].investorAddress;
        payable(investor).transfer(rentAmount);
        projects[projectId].investors[i].hasReceivedRent = true;
    }
}
```

getInvestmentDetails:

```
function getInvestmentDetails(address investor) public view returns (uint256) {
    return investments[investor];
}
```

5.4.4 Deployment & Testing

Deployment

The smart contracts were deployed following a structured workflow [76]:

- Compiled and deployed using  `Remix IDE` onto the **Sepolia Testnet**
- Deployment transactions signed through  `MetaMask` with test ETH
- Contracts made publicly accessible on  `Etherscan` for validation and transparency
- Deployed contract addresses:
  - **Investment Contract:** `0xC25E147316c1dBD16f5B6427e381f9F4fF9510D6`
  - **Projects Contract:** `0xed3c0f14c4b767A65955B71dD1f8328f51f38DE0`
  - **Rent Distribution Contract:** `0x74Ca4D7B2856dddCb1a074A9e40A7fC33deD6437`

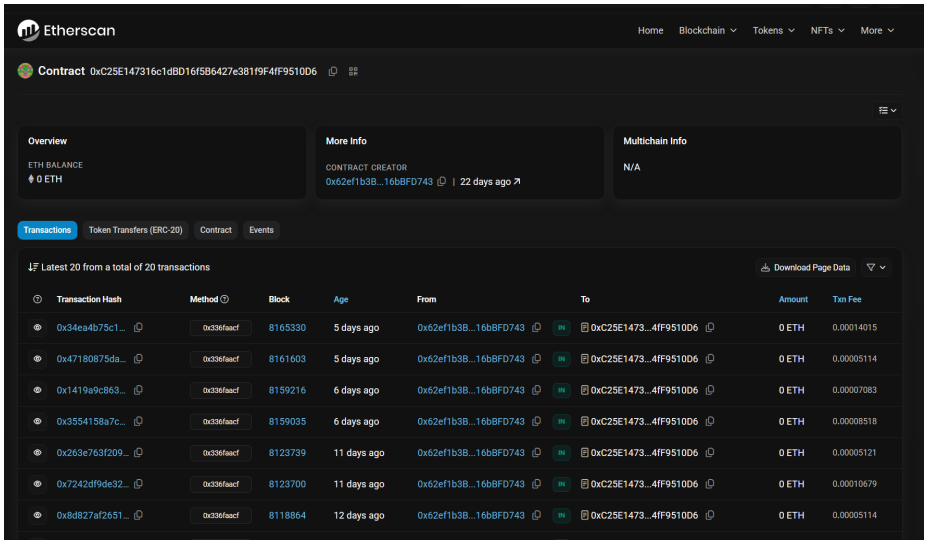


Figure 5.2: Investment Contract in Etherscan after deployment

Testing

The testing process used the following tools:

- **Remix IDE:** For writing, testing, and deploying Ethereum smart contracts
- **Hardhat:** For local blockchain simulation, testing, and deploying contracts

The testing approach included:

- **Unit Testing:** Testing core functions individually
- **Integration Testing:** Validating full transaction flows
- **Event Listening:** Capturing and validating event emissions
- **Performance Testing:** Ensuring efficient operation under load [77]

5.5 Integration Flow

The integration flow can be summarized as follows [78]:

5.5.1 Investment Flow

- The frontend initiates a payment session via Paymee's API
- Paymee redirects the user to complete the payment
- Upon success, Paymee calls back our backend with the transaction details
- After verification, the backend invokes the appropriate smart contract to log the investment on the blockchain

5.5.2 Rent Distribution Flow

- The backend initiates the rent distribution process by invoking the smart contract
- The smart contract calculates each investor's share based on their investment proportion
- For each investor, the smart contract emits an event indicating the calculated share
- The backend listens for these events, verifies the data, and triggers the payment through Paymee's API
- System records are updated after successful transactions

5.6 Backoffice Features

5.6.1 Administrative Dashboard

The backoffice provides administrators with a comprehensive dashboard to manage the platform. Key features include:

- Real-time monitoring of blockchain transactions
- Project management interface for adding, editing, and removing properties
- User management with role-based access control [79]
- Financial reporting and analytics
- Blockchain transaction verification tools

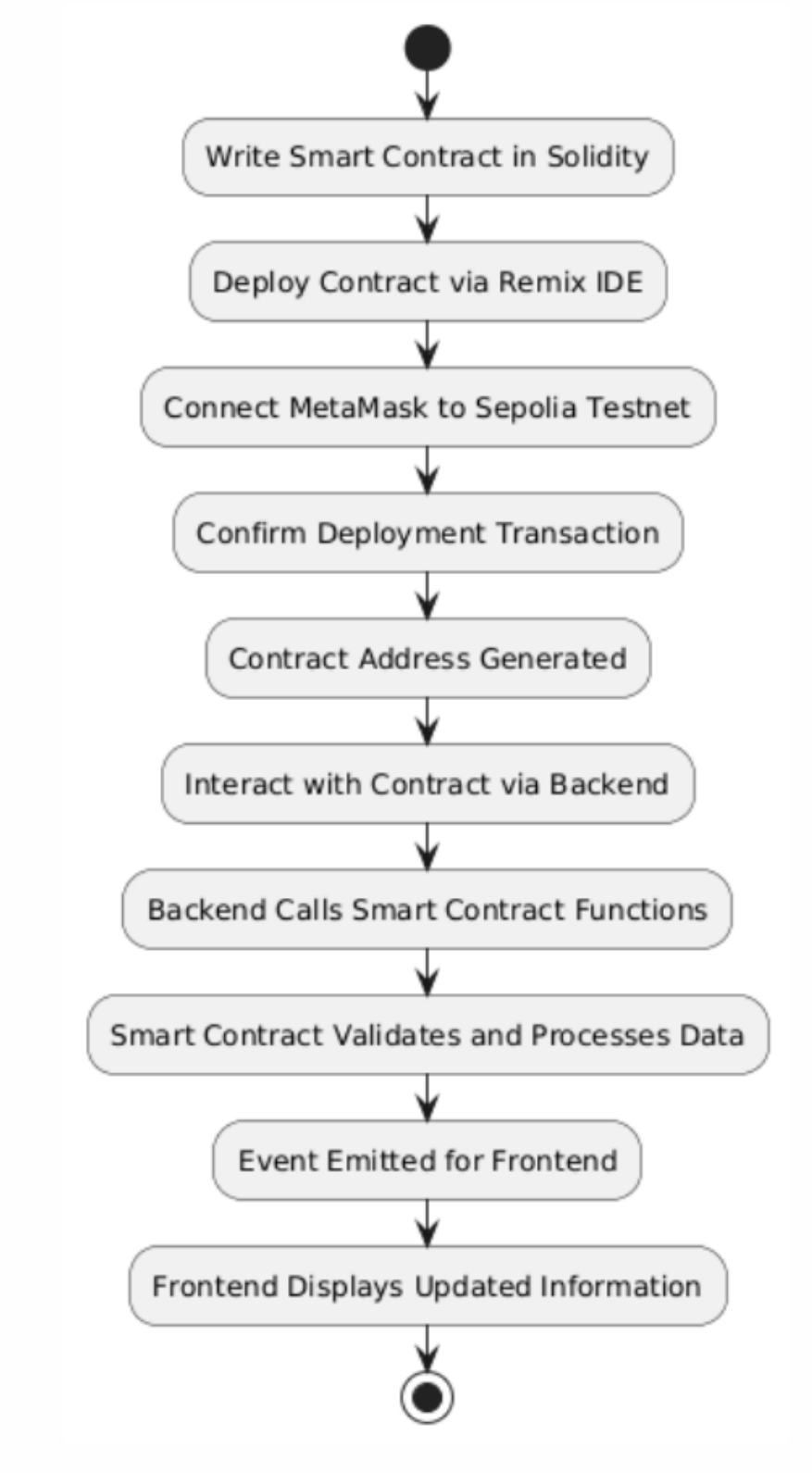


Figure 5.3: Flowchart: Smart Contract Deployment & Interaction

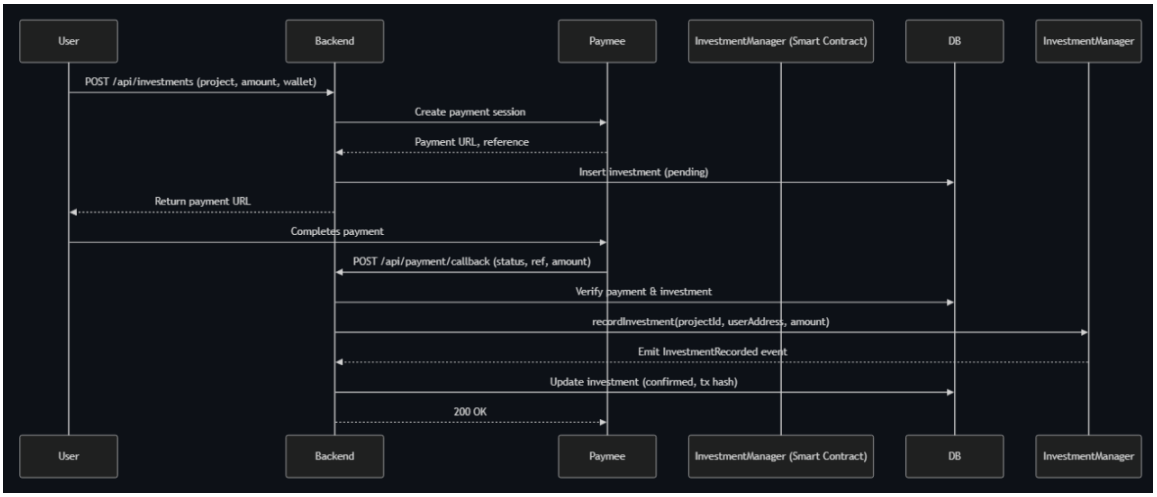


Figure 5.4: Sequence Diagram of the Investment Process

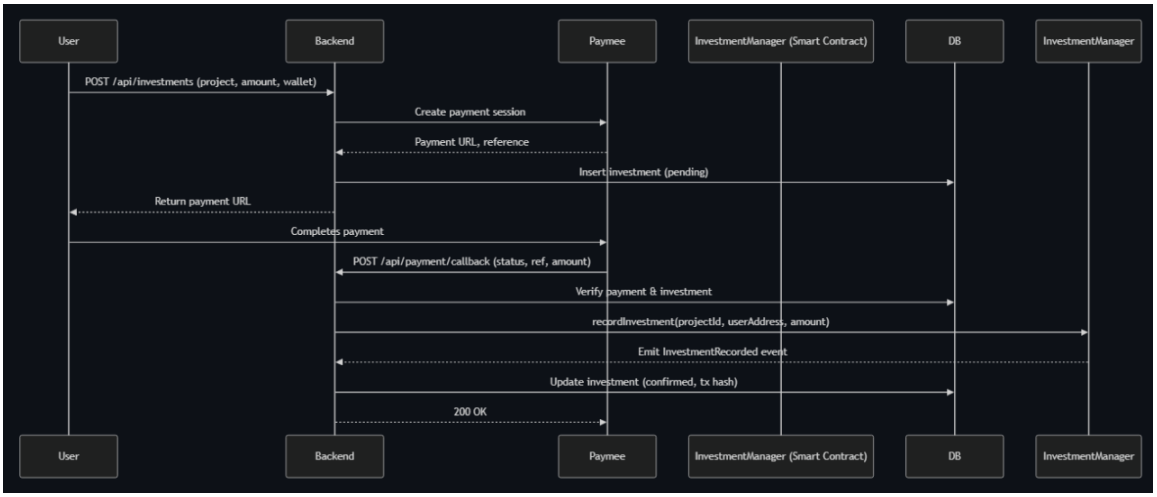


Figure 5.5: Sequence Diagram of Rent Distribution

Figure 5.6: Blockchain Transaction Monitoring in Administrative Dashboard

5.6.2 Transaction Management

Administrators can monitor and manage all types of transactions:

- View all investment records with blockchain verification links
- Monitor rent distribution status and history
- Verify payment processing through the Paymee integration
- Generate financial reports and audit trails

Figure 5.7: Transaction Management Interface with Blockchain Verification

Conclusion

The blockchain integration and backoffice features of the Korpor platform provide a robust foundation for secure, transparent, and efficient real estate investment management. By leveraging blockchain technology for critical financial operations while maintaining a user-friendly interface through the backoffice, the platform successfully bridges the gap between traditional finance and decentralized systems [80].

The smart contracts deployed on the Ethereum Sepolia Testnet validate core operations such as investment recording and rent distribution, ensuring the accuracy and transparency of financial flows. All interactions are securely handled and recorded on-chain, offering verifiable proof of activity and strengthening participant confidence.

By guaranteeing the integrity of payments and providing clients with clear, tamper-proof records, the blockchain integration significantly enhances user trust and overall satisfaction with the Korpor platform.

CHAPTER 6

DevOps & Mobile App Features

6.1 Introduction

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

APPENDIX A

Technical Documentation

APPENDIX B

User Manual

APPENDIX C

Project Timeline

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] 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).

[3] 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).

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

[5] 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).

[6] RECAP. *AI, fintech, and automation on the real estate industry*. LinkedIn. Feb. 2024. URL: https://www.linkedin.com/pulse/ai-fintech-automation-real-estate-industry-recapeg-wup1f (visited on 03/15/2025).

[7] Mobile Reality Team. *How AI in Real Estate is Transforming the Future of Property and Investment*. The Mobile Reality. Mar. 2024. URL: https://themobilereality.com/blog/proptech/ai-in-real-estate (visited on 03/19/2025).

[8] HouseCanary Team. *5 AI Tools for Real Estate Investors*. HouseCanary Blog. Dec. 2024. URL: https://www.housecanary.com/blog/5-ai-tools-for-real-estate-investors (visited on 02/10/2025).

[9] Daniel Smilkov et al. "Tensorflow.js : Machine learning for the web and beyond". In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 309–321.

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

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

[12] 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).

[13] 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).

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

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

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

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

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

[19] Henrik Kniberg. *Lean from the Trenches: Managing Large-Scale Projects with Kanban*. Pragmatic Bookshelf, 2013. ISBN: 978-1934356852.

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

[21] 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).

[22] 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.

[23] 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).

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

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

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

[27] 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).

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

[29] *Storybook Documentation*. Storybook Contributors. 2022. URL: https://storybook.js.org/docs/ (visited on 05/30/2025).

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

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

[32] Microsoft Azure Team. *Azure Cloud Platform Services*. Microsoft Corporation. 2024. URL: https://azure.microsoft.com/en-us/services/ (visited on 05/27/2025).

[33] Michael R. Johnson and Wei Zhang. "Advanced AI Techniques in Property Valuation: Challenges and Opportunities". In: *Journal of Real Estate Technology* 15.2 (2024), pp. 78–95. DOI: 10.1080/14735789.2024.2175633.

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

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

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

[37] 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.

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

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

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

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

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

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

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

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

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

[47] 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).

[48] 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).

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

[50] Jakob Nielsen and Kara Pernice. "Responsive Design and User Experience: Impact on Mobile Interfaces". In: *User Experience Magazine* 19.3 (2019).

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

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

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

[54] 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.

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

[56] *Tailwind CSS*. Tailwind Labs Inc. URL: https://tailwindcss.com/ (visited on 04/16/2025).

[57] *PropertyStar Tunisia - Real Estate Listings*. PropertyStar Ltd. 2023. URL: https://www.properstar.co.uk/tunisia (visited on 11/15/2023).

[58] *RE/MAX Tunisia Real Estate*. RE/-MAX Tunisia. 2023. URL: https://www.remax.com.tn/ (visited on 11/15/2023).

[59] *Al-Mindhar Real Estate Platform*. STE SK ALMINDHAR. 2023. URL: https://al-mindhar.com/ (visited on 11/15/2023).

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

[61] Don Tapscott and Alex Tapscott. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Portfolio, 2016. ISBN: 978-1101980132.

[62] Jan Veuger. "Trust in a viable real estate economy with disruption and blockchain". In: *Facilities* 36.1/2 (2018), pp. 103–120. DOI: 10.1108/F-11-2017-0106.

[63] 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).

[64] 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.

[65] 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.

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

[67] 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.

[68] 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.

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

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

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

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

[73] 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.

[74] 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.

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

[76] Karl Wüst and Arthur Gervais. "Do you need a Blockchain?" In: (2018), pp. 45–54. DOI: 10.1109/CVCBT.2018.00011.

[77] Reza M. Parizi et al. "Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains". In: (2018), pp. 103–113.

[78] Fran Casino, Thomas K. Dasaklis, and Constantinos Patsakis. "A systematic literature review of blockchain-based applications: Current status, classification and open issues". In: *Telematics and Informatics* 36 (2019), pp. 55–81. DOI: 10.1016/j.tele.2018.11.006.

[79] Johannes Botha, Christiaan Schutte, and Herman Steyn. "An Architecture for Blockchain Back-Office Systems for Supply Chain Management using Business Process Model and Notation". In: *South African Journal of Industrial Engineering* 32.2 (2021), pp. 59–73. DOI: 10.7166/32-2-2544.

[80] Merlinda Andoni et al. "Blockchain technology in the energy sector: A systematic review of challenges and opportunities". In: *Renewable and Sustainable Energy Reviews* 100 (2019), pp. 143–174. DOI: 10.1016/j.rser.2018.10.014.