# SafeTrade: Escrow Marketplace



| | |
|---|---|
| **Submitted by** | **Muhammad Sabir Hussain** |
| | **Maria Khadim** |
| | **Lubna Khizar** |
| **Roll No.** | **0129-BSCS-20** |
| | **0081-BSCS-20** |
| | **0150-BSCS-20** |
| **Session** | **2020 - 2024** |
| **Supervised by** | **Dr. Umair Sadiq** |
| | **Lecturer** |


**BS(HONS) IN COMPUTER SCIENCE**


**DEPARTMENT OF COMPUTER SCIENCE**

**GC UNIVERSITY LAHORE**

# Safetrade: Secure Escrow Marketplace

**Submitted to GC University Lahore in partial fulfillment of the requirements for the award of degree of**

## BS(HONS)
## IN
## COMPUTER SCIENCE


| | |
|---|---|
| **Submitted by** | **Muhammad Sabir Hussain** |
| | **Maria Khadim** |
| | **Lubna Khizar** |
| **Roll. No** | **0129-BSCS-20** |
| | **0081-BSCS-20** |
| | **0150-BSCS-20** |
| **Session** | **2020 - 2024** |
| **Supervised by** | **Dr. Umair Sadiq** |
| | **Lecturer** |

## DEPARTMENT OF COMPUTER SCIENCE

## GC UNIVERSITY LAHORE

# Declaration

**Muhammad Sabir Hussain** Roll No. **0129-BSCS-20**, **Maria Khadim** Roll No. **0081-BSCS-20**, and **Lubna Khizar** Roll No. **0150-BSCS-20** students in the **Computer Science**, and program for the session 2020-2024, affirm that the content presented in this thesis, **Safetrade** is our original work. We confirm that it has not been duplicated, published, or submitted as research, a thesis, or any form of publication in any educational institution or research organization in Pakistan or abroad.

**Date:** _____

**Student's Signature:**

Muhammad Sabir Hussain(Roll No.0129-BSCS-20)

_____

Maria Khadim(Roll No.0081-BSCS-20)

_____

Lubna Khizar(Roll No.0150-BSCS-20)

_____

# Research Completion Certificate

It is certified that the research work contained in this thesis titled **Safetrade** has been carried out by **Muhammad Sabir Hussain, Maria Khadim and Lubna Khizar** Roll No. **0129-BSCS-20, 0081-BSCS-20 and 0150-BSCS-20** under my supervision.

---

**Dr. Umair Sadiq**
**Lecturer**

**Date:** _____

**Submitted Through**

---

**Prof. Dr. Muhammad Waqas Anwar**
**Chairperson**
**Department of Computer Science**
**GC University Lahore**

---

**Controller of Examination**
**GC University Lahore**

# Acknowledgements

On this occasion, we express our gratitude to the **Almighty Allah** for bestowing His blessings upon us, leading our efforts to a successful conclusion. We extend our deepest thanks to the **Holy Prophet Muhammad (Peace be upon him)** the most revered figure in history. We wish to convey our heartfelt appreciation to our esteemed teacher for their invaluable guidance and support during critical moments, steering us in the right direction. Furthermore, we'd like to acknowledge our supervisor, **Dr. Umair Sadiq** for their unwavering assistance, constructive feedback, and mentor-ship. We also extend our thanks to our esteemed chairperson, **Prof. Dr. Muhammad Waqas Anwar** for granting us access to essential resources and facilities. We appreciate the contributions of our other teachers as well. Finally, we are grateful to our friends and family for their unwavering support and encouragement throughout our journey.

# *Dedication*

*This project work is dedicated to the ALMIGHTY ALLAH for making us being able to start up and finish in sound health. Also to our Parents and Teachers who encouraged and helped us to complete our research.*

# *Abstract*

**SafeTrade**, an innovative platform, facilitates secure transactions for buying and selling products and digital goods. With an emphasis on security, the application employs escrow protection for all transactions. Payments from buyers are held in escrow during transactions and are released only upon the seller fulfilling their obligations, effectively preventing fraud. In cases where the seller fails to deliver, payments are refunded to the buyer.

In addition to the customer-facing application, **SafeTrade** incorporates a robust system panel for staff and administrators. This panel facilitates various operational tasks, including managing categories, user management, handling support tickets, communicating with customers regarding support issues, overseeing finances (deposits and withdrawals), managing disputes, customer suspensions, and approving KYC verifications.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction


FIGURE 1.1: Safetrade logo

## 1.1 Introduction

In today's digital age, ensuring the security of online transactions is paramount. **SafeTrade** emerges as a groundbreaking platform designed to facilitate secure and reliable transactions for both physical products and digital goods. The core of **SafeTrade's** innovation lies in its stringent security measures and user verification processes, aimed at fostering a trustworthy marketplace environment.

**SafeTrade** differentiates itself through the implementation of escrow protection for all transactions. This mechanism ensures that payments from buyers are securely held in escrow accounts during transactions and are only released once the seller has met all obligations. Such a system effectively mitigates the risk of fraud, providing peace of mind to both buyers and sellers. In scenarios where a seller fails to deliver the promised goods or services, **SafeTrade** guarantees a full refund to the buyer, thereby upholding the platform's commitment to fairness and security.

A critical feature of **SafeTrade** is the requirement for sellers to undergo Know Your Customer (KYC) verification. This process involves thorough vetting of sellers' identities, enhancing the overall security and integrity of the marketplace. Only KYC verified users are permitted to sell on **SafeTrade**, ensuring that all transactions are conducted with verified and trustworthy parties.

Communication within the platform is another layer of security, as all interactions between buyers and sellers are documented. These records serve as valuable evidence in case of disputes, further safeguarding the interests of both parties. Additionally, **SafeTrade** offers a dedicated customer support system where users can register support or dispute tickets, ensuring prompt and efficient resolution of any transaction-related issues.

The platform is equipped with a built-in wallet feature, enabling seamless payment operations. Users can easily deposit or withdraw funds for their transactions, streamlining the buying and selling process. For sellers, **SafeTrade** provides a comprehensive listing system across various categories, accompanied by a rating system. Buyers can leave feedback based on their experiences, contributing to the public profiles of sellers by showcasing completed orders, recent offers, and overall ratings and reviews.

Beyond its customer-facing capabilities, **SafeTrade** integrates a robust system panel designed for staff and administrators. This backend system supports a range of operational tasks including category management, user management, support ticket handling, financial oversight (deposits and withdrawals), dispute management, customer suspensions, and KYC verification approvals.

## 1.2 Literature Review

The literature review for SafeTrade delves into the existing body of knowledge surrounding secure online transactions, escrow services, KYC (Know Your Customer) verification processes, and the overall landscape of digital marketplaces.

This review will examine the critical components that underpin the SafeTrade platform, highlighting the significance of each element in fostering a secure and reliable online trading environment.

### 1.2.1 Escrow Services

Escrow services have been widely recognized as a critical component in mitigating transaction-related risks in online marketplaces. According to Huang et al. (2017), escrow accounts act as neutral intermediaries that hold funds until all parties fulfill their contractual obligations. This mechanism significantly reduces the risk of fraud by ensuring that payments are only released upon successful delivery of goods or services

### 1.2.2 KYC (Know Your Customer) Verification

KYC processes are integral to the security framework of financial transactions and online platforms. A study by Kshetri (2021) emphasizes the role of KYC in preventing identity theft, money laundering, and other forms of financial fraud. By verifying the identities of users, platforms can ensure that they are dealing with legitimate individuals, thereby reducing the likelihood of fraudulent activities

### 1.2.3 Digital Marketplaces and User Trust

Trust is a fundamental aspect of digital marketplaces, as noted by Pavlou and Gefen (2004). Their research indicates that trust in online transactions is influenced by factors such as the reputation of sellers, transparency of the platform, and the presence of secure payment systems

### 1.2.4 Dispute Resolution Mechanisms

Effective dispute resolution mechanisms are crucial for maintaining user trust in online platforms. The work of Siering, Muntermann, and Rajagopalan (2018) explores the impact of dispute resolution systems on user satisfaction and trust. Their findings suggest that platforms with robust dispute resolution processes are more likely to retain users and foster positive transaction experiences

### 1.2.5 Administrative and Operational Management

The administrative and operational aspects of digital marketplaces are critical for their smooth functioning. Research by Turban et al. (2018) discusses the importance of efficient user management, category management, and financial oversight in maintaining the integrity and effectiveness of online platforms

## 1.3 Background of The Problem

In the realm of online commerce, counterparty risks are a significant concern for both buyers and sellers. These risks refer to the possibility that one party in a transaction may not fulfill their contractual obligations, leading to potential financial loss and a breakdown in trust. Common scenarios include sellers failing to deliver the purchased goods or services, and buyers not completing the payment after receiving the goods. Such risks are exacerbated in digital marketplaces where parties often have no prior relationship and must rely solely on the platform's mechanisms for security and dispute resolution.

Counterparty risks can manifest in various forms:

1. **Fraudulent Sellers**: Sellers might take payment without delivering the promised goods or services, leaving buyers without recourse.

2. **Non-Payment by Buyers**: Buyers might receive goods or services but refuse to complete the payment, causing financial losses for sellers.

3. **Delivery Issues**: There could be discrepancies in the quality or quantity of the delivered goods, leading to disputes.

4. **Scams and Deception**: Fraudsters may exploit the lack of stringent verification processes to deceive honest participants, undermining trust in the marketplace.

## 1.4 Aims of the Project

1. **Escrow protected marketplace**: The project's primary aim is to develop a secure marketplace allowing buyers or sellers to engage in transactions over any kind of product or service without worrying about counter-party risks.

2. **Communication Channels**: Safetrade will implement communication channels for buyer and sellers to communicate related to transacton. These channels also facilitate in communication with support team on support tickets and disputes.

3. **Funds Management**: Implementation of a wallet system to enable users to securely manage their funds within the platform i.e.,, Deposits and Withdrawals of funds within the platform.

4. **Reliable Customer Support**: SafeTrade will provide dedicated customer support services to promptly address user inquiries, report issues, and resolve disputes.

5. **Empowered Seller Feedback Mechanism**: Integration of a feedback and rating system will empower buyers to provide feedback and rate sellers based on their transaction experiences, aiding other users in making informed decisions.

## 1.5 Scope of the Project

The scopes of the project are enlisted below:

1. **User-Friendly Platform Development**: Designing an intuitive and user-friendly online marketplace accessible via web browsers and mobile devices. The marketplace should enable the buyers to search offers on Safetrade. Search implementation should enable the buyer fine-grained control over search criteria. The platform should provide seamless experience to both buyers and sellers during transaction and support if needed.

2. **Communication Feature Implementation**: Development of Chat functionality to enable users to communicate with each other during transactions. This functionality should also be integrated with System Panel allowing communication with Support Staff over support tickets or disputes

3. **Wallet Management System**: Wallet implementation to allow users to deposit, withdraw or use funds in transactions within Safetrade Platform. The wallet operations are sensitive enough to consider secure and managed balance operations. There should be a way for user to generate wallet statement.

4. **Comprehensive Customer Support System**: Establishing a dedicated customer support system capable of efficiently handling user inquiries, issues, and disputes to ensure a satisfactory user experience.

5. **Feedback and Rating System Integration**: Incorporating a feedback and rating system to allow buyers to provide feedback and rate sellers based on their transaction experiences, enhancing transparency and trust.

6. **System Panel Development for Administration**: Creating a robust system panel to facilitate the management and administration of various system operations, including user management, category management, support ticket handling, and dispute resolution, ensuring smooth platform operation and scalability.

## 1.6 Limitations

Some of the limitations this app has are as follow:

1. **Challenges with Deposit and Withdrawals**: SafeTrade faces challenges with deposit and withdrawals, particularly in regions where certain financial instruments, or services such as PayPal or Stripe not available in the region. We had to rely on manual deposits or withdrawals through Mobile Wallets and Bank Transfers then have manual verification of transaction.

2. **Regulatory Compliance**: Compliance with regulatory frameworks presents a significant challenge for SafeTrade. The platform must adhere to various legal requirements and regulations governing financial transactions, consumer protection, and data privacy. Failure to comply with these regulations could result in legal repercussions and fines.

3. **Privacy Concerns and KYC Data**: SafeTrade collects sensitive data for Know Your Customer (KYC) verification, raising concerns about data privacy and compliance with data protection laws. Safeguarding this data from unauthorized access, breaches, or misuse is paramount to ensure user trust and regulatory compliance.

4. **Technical Issues**: Like any online platform, SafeTrade is susceptible to technical issues and challenges. These may include system downtime, server outages, software bugs, and cybersecurity threats. Addressing and mitigating these technical issues in a timely manner is essential to maintain platform functionality and user satisfaction.

# Chapter 2

# Requirement Specification

Requirement specifications are an essential part of the completion of any project for proper functioning. So we divided them in two parts:

1. Non-Functional Requirements
2. Functional Requirements

## 2.1 Non-functional Requirements

These requirements are not related to the application interface. Usually required for optimal performance, security needed to improve our application working, which affects the user review in using that application. i.e. How the system will behave in terms of non-functional aspects such as performance, flexibility, etc.

### 2.1.1 Security

Users must be logged in to the system to perform any operation and only registered users are allowed to interact with the system [3]. All important operations are handled by a super administrator only. The app must ensure that no one can access or bypass the personal information of any person.

### 2.1.2 Performance

The application should be well optimized so that it can work well [1]. All the features of the application must ensure that work is done on time and the app performs well on different devices. Moreover, the response time should be fast.

### 2.1.3 Usability

The system should be easy to use or in other words user friendly. So that everyone can understand and use 'Faculty E-facilitator'.

### 2.1.4 Flexibility And Scalability

The system should adapt to future changes. It should not be complicated in terms of scalability in the future.

### 2.1.5 Reliability

The application should be reliable. It should be ready for use at any time. It should work properly whenever needed.

### 2.1.6 Maintainability

The system should be easy to maintain without having to cost much to maintain.

### 2.1.7 Modification

The system must ensure its modification anytime without affecting other modules.

## 2.2 Functional Requirements

Functional requirements are those that define the behavior of any application. So, these requirements give a basic flow of what are the basic requirements to be followed by the system. Following are the functionalities required for our web application to function properly:

### 2.2.1 Marketplace

1. The system shall allow public users to search for offers (products or services).
2. The system shall allow users to view offer details, including product/service details and seller information.
3. The system shall provide the functionality to save offers for later viewing.
4. The system shall display saved offers to authenticated users.

### 2.2.2 Dashboard

1. The system shall enable sellers to post and edit sell offers.
2. The system shall provide users with access to their control panel for managing offers.
3. The system shall display user purchases, providing a summary of all transactions.
4. The system shall allow users to update their profile information.
5. The system should allow users access to wallet dashboard

### 2.2.3 Wallet

1. The system shall allow users to check their wallet balance and view wallet statements.

2. The system shall enable users to deposit funds into their wallet.

3. The system shall verify deposits and mark them as successful, increasing the wallet balance accordingly.

4. The system shall allow users to withdraw funds from their wallet.

5. The system shall process withdrawals and mark them as successful, decreasing the wallet balance accordingly.

6. The system shall track the status of deposits and withdrawals for users to monitor.

### 2.2.4 Transactions

1. The system shall facilitate the initiation of transactions between buyers and sellers.

2. The system shall lock payments in escrow upon transaction initiation.

3. The system shall allow sellers to mark items as delivered.

4. The system shall enable buyers to confirm or disapprove the delivery of items.

5. The system shall release payment to the seller upon buyer approval.

6. The system shall revert payment back to the buyer if the buyer disapproves the delivery or wins a dispute.

### 2.2.5 Support

1. The system shall allow users to create support tickets for transaction-related issues.
2. The system shall enable support staff to perform transaction actions and resolve disputes.
3. The system shall facilitate communication between users and support staff through a messaging system.

### 2.2.6 User Verification

1. The system shall require sellers to submit KYC documents for verification.
2. The system shall allow audit staff to verify and approve KYC submissions.

### 2.2.7 User Registration and Onboarding

1. The system shall provide a registration process for new users.
2. The system shall allow users to log in to their accounts.
3. The system shall guide users through an onboarding flow where they can choose a display name for the marketplace.
4. The system shall prompt users to upload a profile picture during the onboarding process.
5. The system shall ask users if they want to sell items. If a user wants to sell, the system shall prompt the user to submit KYC documents for verification.

### 2.2.8 System Panel

1. The system shall allow system users to sign in.

2. The system shall allow root users to add more system users.
3. The system shall allow root users to manage permissions and roles of other system users.
4. The system shall allow support staff to list support tickets and assign a ticket to others or themselves.
5. The system shall allow support staff to work on assigned tickets, including chatting with the user.
6. The system shall allow support staff to resolve dispute tickets.
7. The system shall allow support staff to study data on disputes and communicate with counterparts.
8. The system shall allow root users to check activity logs of the platform.
9. The system shall allow accounts staff to list deposits and withdrawals.
10. The system shall allow accounts staff to update the status of deposits and withdrawals.

## 2.3 Tools and Techniques

Following are tools and techniques used in development of this Safetrade:

- NestJS
- Passport.js
- PostgreSQL
- pg_cron (CRON jobs on DB level)
- DrizzleORM
- Socket.IO (websockets)

- NextJS
- React
- React Query
- Typescript
- Webpack
- Nginx
- Tailwind CSS
- Redux
- Github & Github Actions
- TurboRepo (for monorepo)
- PM2 (process manager)
- Shell Scripting
- Linux
- Google Cloud
- CI/CD
- PlantUML
- OpenAPI & Swagger
- Design Patterns
- ... and more

# Chapter 3

# Project Design

This chapter defines the application design phase. The design patterns, methods and processes to be used in project design are listed in this chapter.

## 3.1 Methodology

The methodology is a term used for software systems that define how a system should be designed, how it should be planned, and how to control its development. There are many methodologies used nowadays for developing web applications. One of them is Waterfall Methodology which follows an iterative approach to developing systems.

### 3.1.1 Iterative Methodology

Iterative methodology is an approach to software development that breaks down the project into small segments called iterations. Each iteration involves a cycle of planning, design, implementation, testing, and evaluation, which allows for the continuous refinement of the product based on feedback and new requirements. This approach enables developers to manage changes more effectively, identify

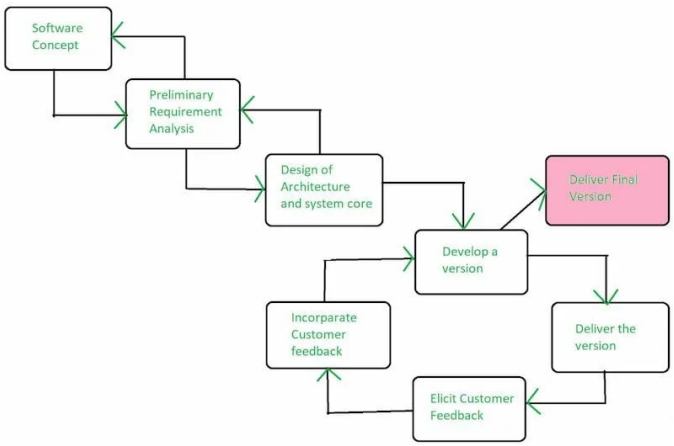


FIGURE 3.1: Iterative Model

and address issues early, and progressively enhance the functionality and performance of the software. The iterative process is particularly beneficial for complex projects where requirements are expected to evolve over time, ensuring that the final product aligns closely with user needs and expectations [2].

#### 3.1.1.1 Why choose Iterative model

The iterative model is preferred development methodology for this projects because of our projects requirements and significant advantages of iterative model over other SDLC models.

1. **Flexibility in Requirements:**

   The iterative methodology allows for adapting to changes in requirements throughout the development process. Given the evolving nature of marketplace and financial technologies, being able to incorporate feedback and new requirements is crucial.

2. **Risk Management:**

   By developing the project in iterations, risks can be identified and mitigated early. This approach helps in addressing technical challenges, regulatory compliance issues, and security concerns systematically.

3. **Continuous Improvement:**

Iterative development supports continuous improvement of the product. Each iteration builds upon the previous one, allowing for incremental enhancements and refinements based on user feedback and testing outcomes.

4. **Early Detection of Issues:**

   Regular iterations allow for early detection of issues and bugs, facilitating timely resolution. This is particularly important for features like escrow payments, KYC verification, and transaction security where reliability is paramount.

5. **Scalability and Maintainability:**

   The iterative approach makes the system more scalable and maintainable. By breaking down the development into manageable chunks, it becomes easier to implement new features, fix issues, and improve the system over time.

6. **Focus on Usability:**

   Each iteration provides an opportunity to test the usability of the system with actual users, allowing for adjustments to improve the user experience. This is essential for ensuring that users can easily navigate the marketplace, wallet, transaction, and support functionalities.

By adopting the iterative methodology, the SafeTrade project benefits from a structured yet flexible approach to development, ensuring that the final product is robust, user-friendly, and aligned with market needs.

## 3.2 Design Diagrams

This section contains different UML diagrams depicting different behaviors of Safetrade system, its interactions with the user, structure and deployment:

### 3.2.1 Use Case Diagram

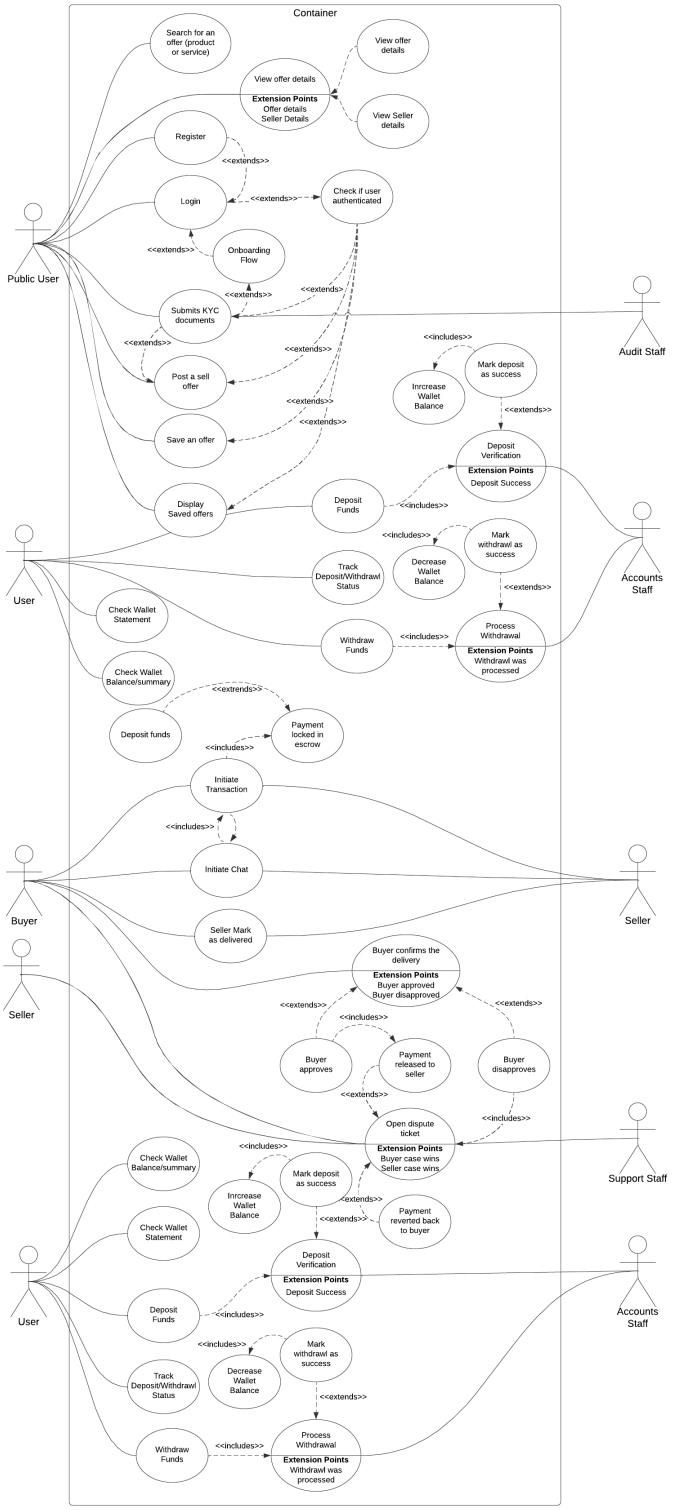


FIGURE 3.2: Safetrade's Use-Case

## 3.3 Activity Diagram

#### 3.3.0.1 User verification, onboarding, KYC and selling

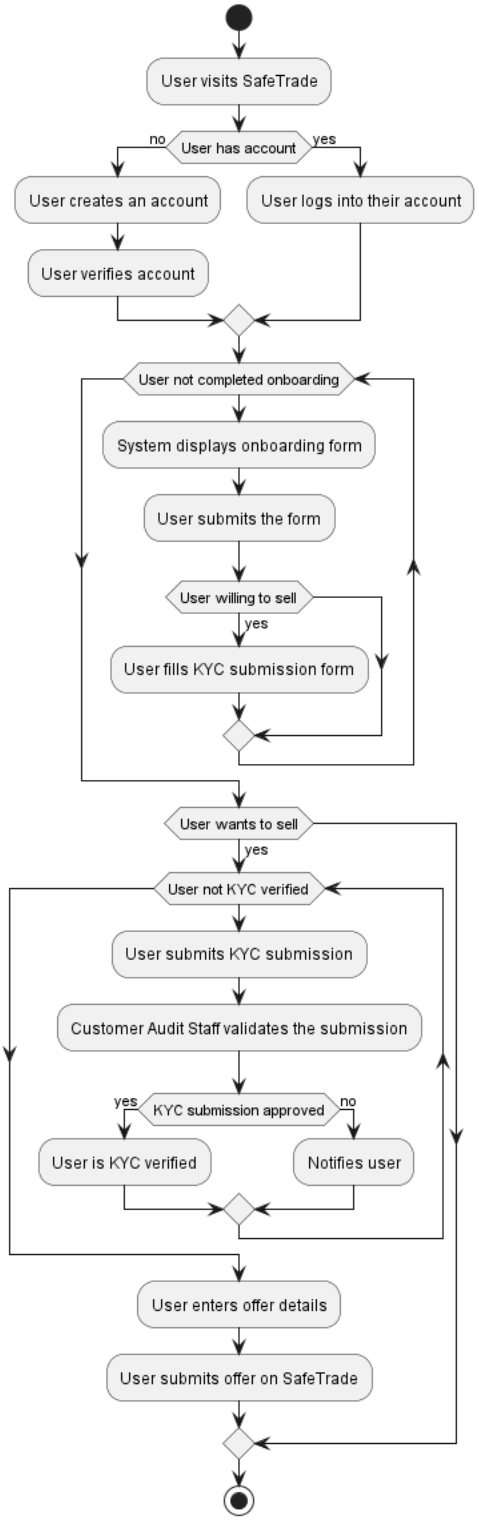


FIGURE 3.3: User verification, onboarding, KYC and sell activity

#### 3.3.0.2 User initiate transaction

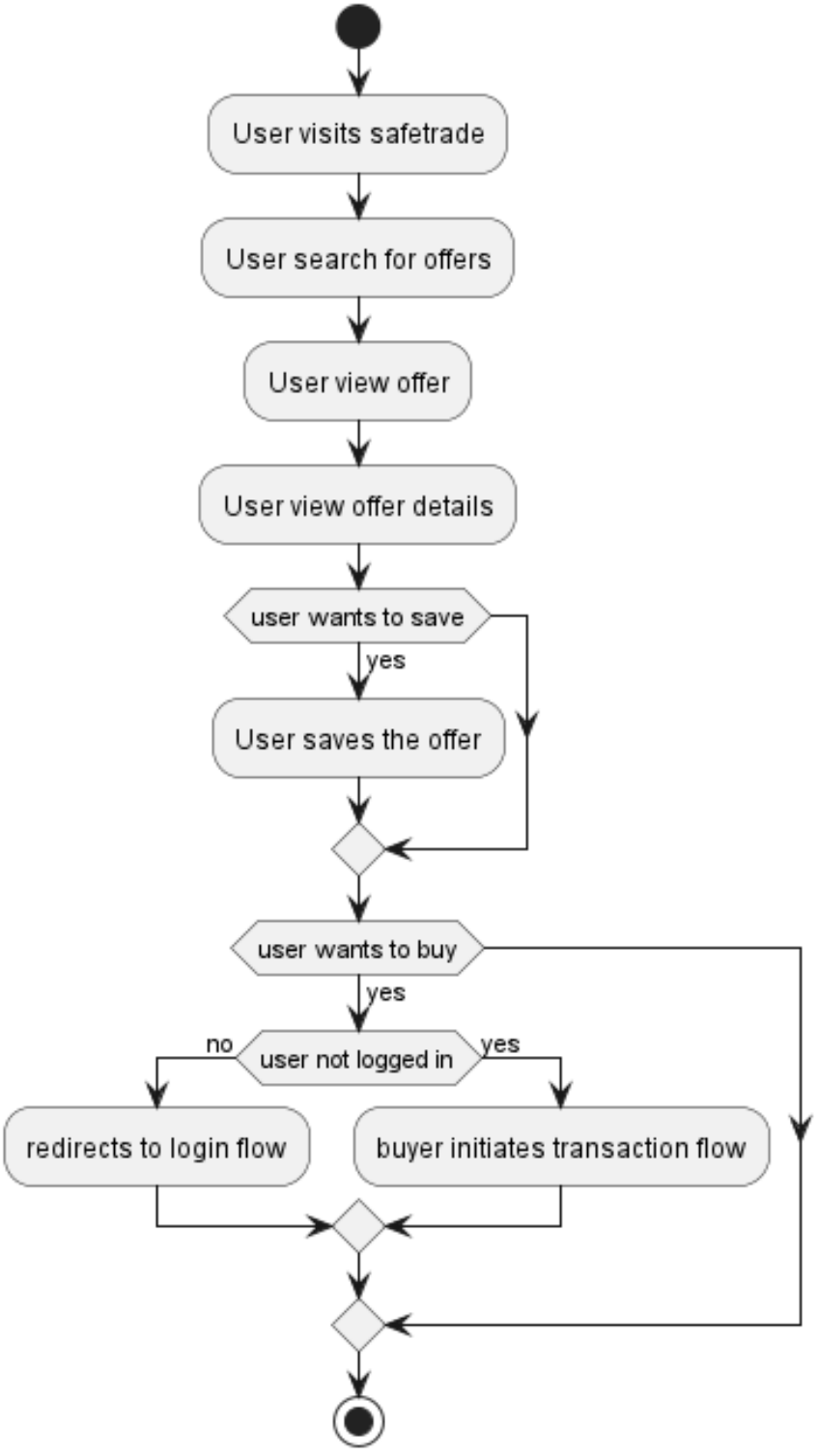


FIGURE 3.4: User initiate transaction activity

#### 3.3.0.3 Transaction

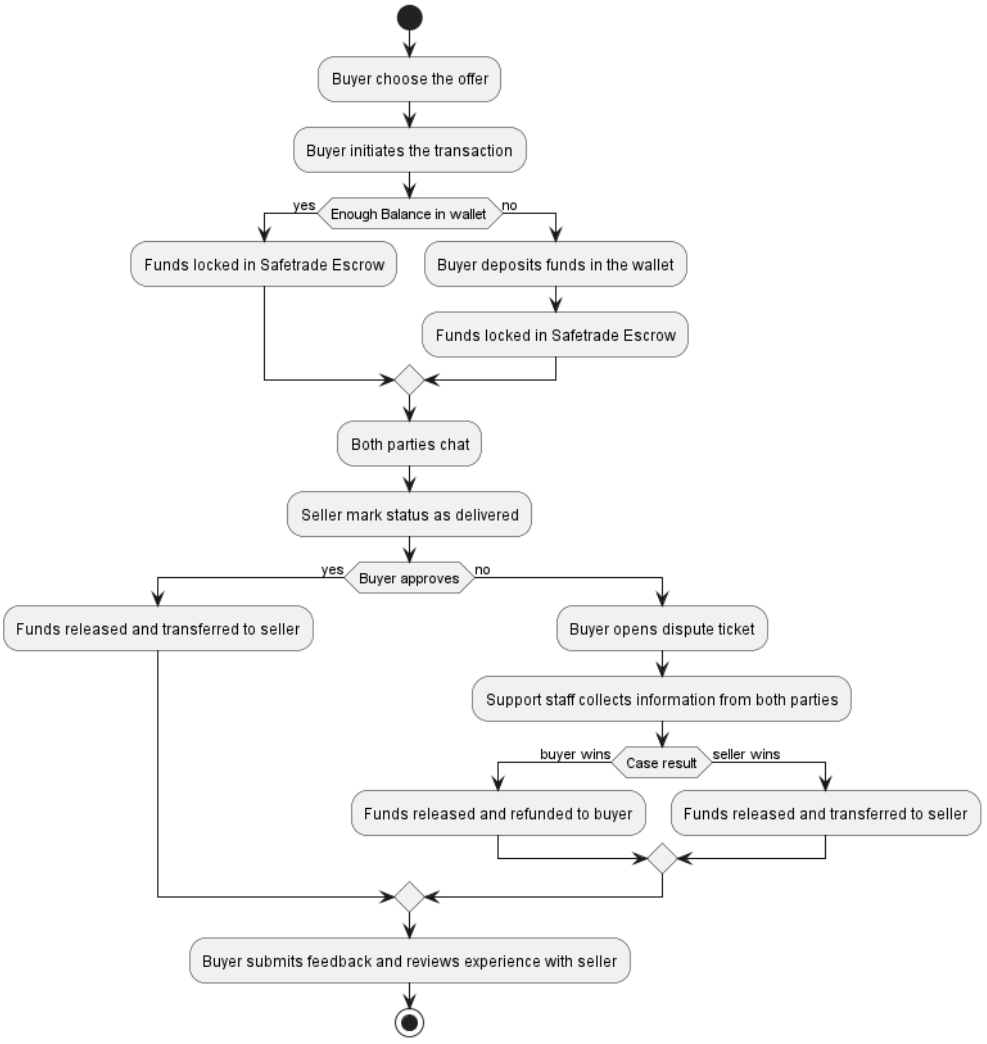


FIGURE 3.5: Transaction activity

### 3.3.1 Entity Relationship Diagrams

#### 3.3.1.1 Financial

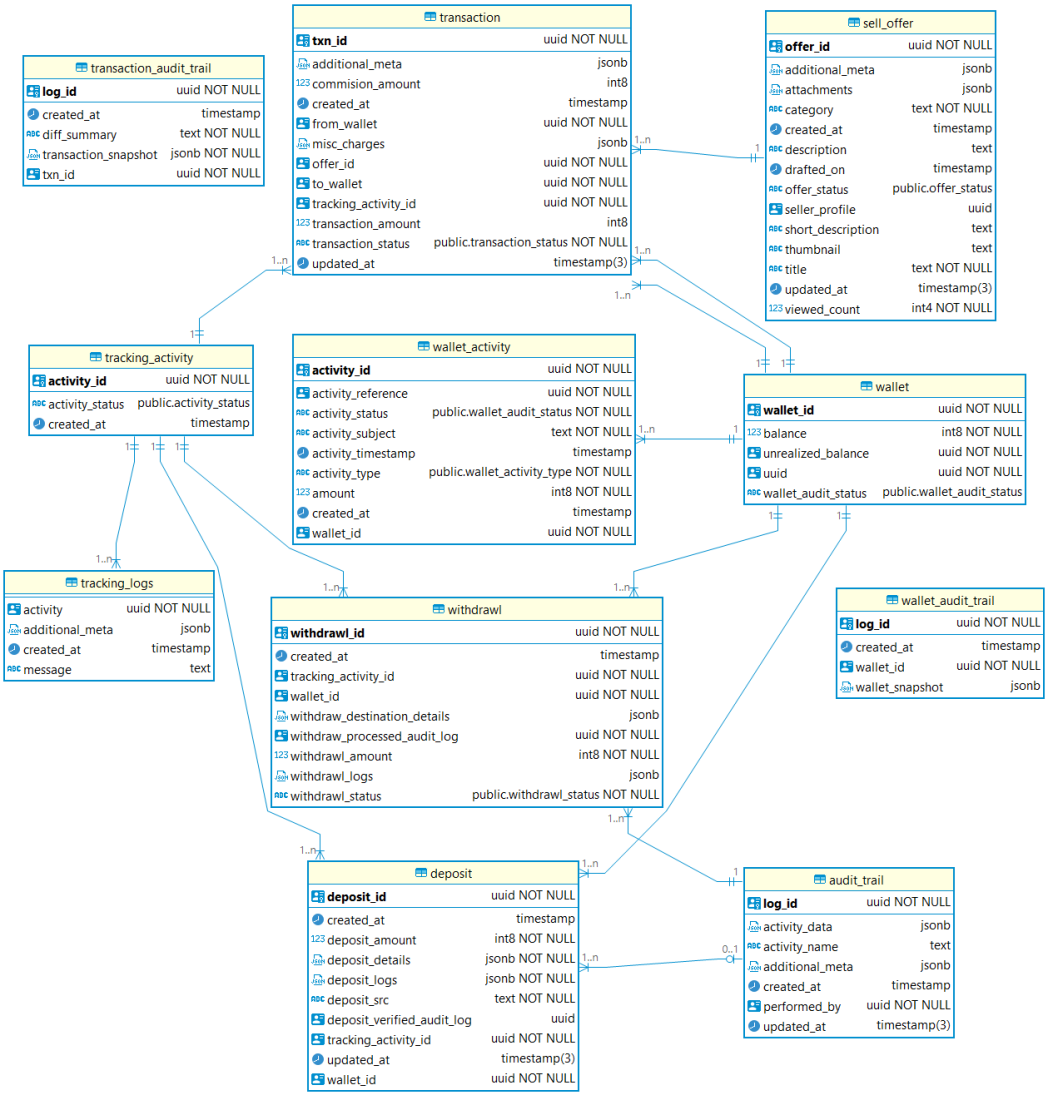


FIGURE 3.6: Financial ERD

#### 3.3.1.2 System User Ops

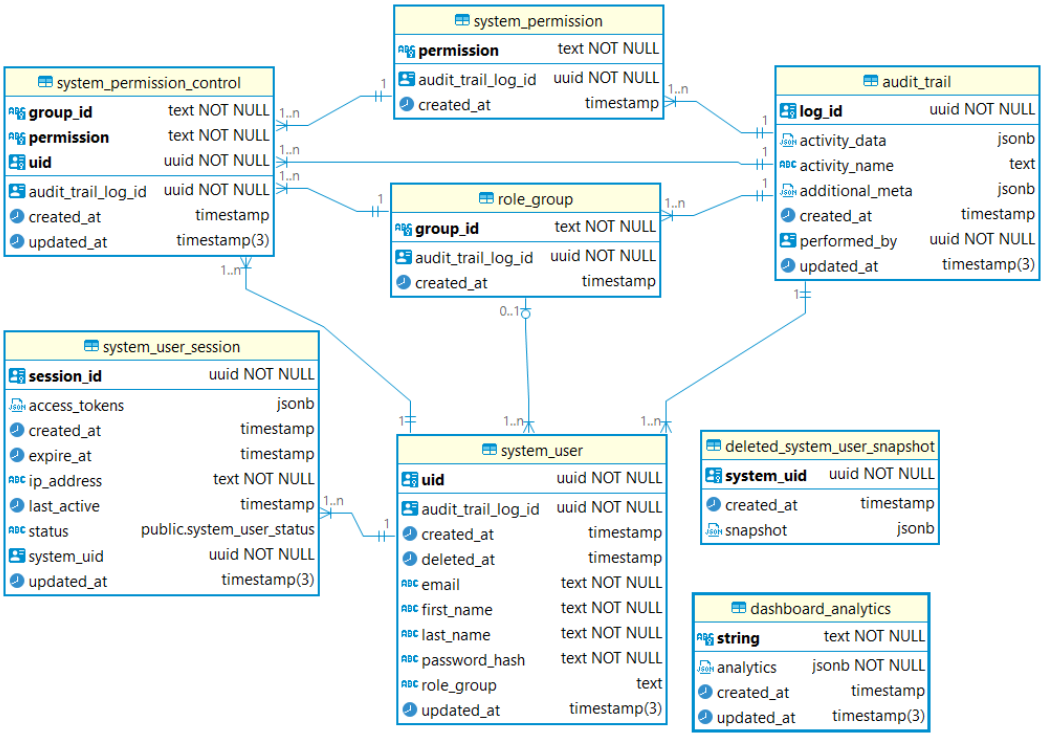


FIGURE 3.7: System User Ops ERD

#### 3.3.1.3 User Ops

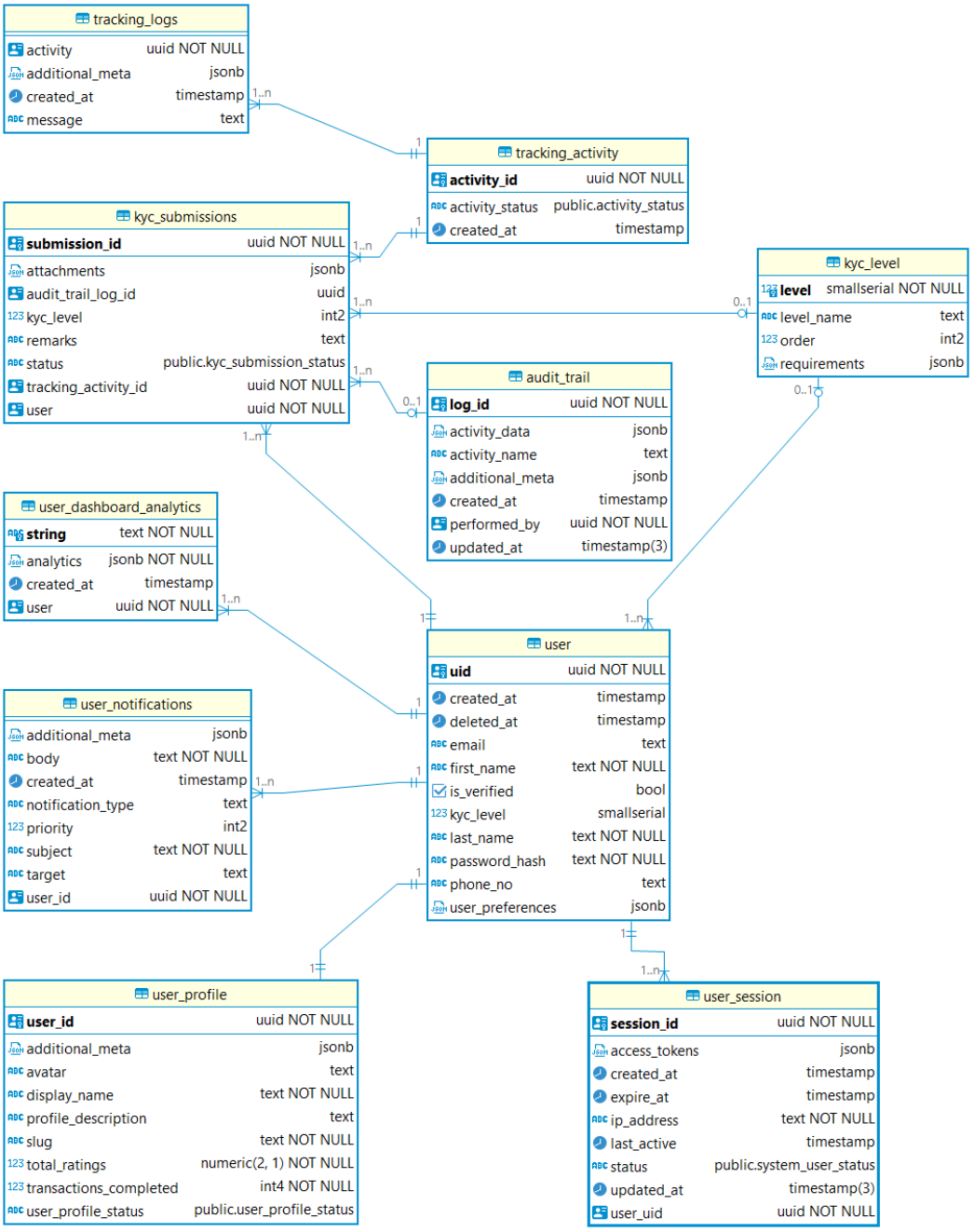


FIGURE 3.8: User Ops ERD

#### 3.3.1.4 Marketplace

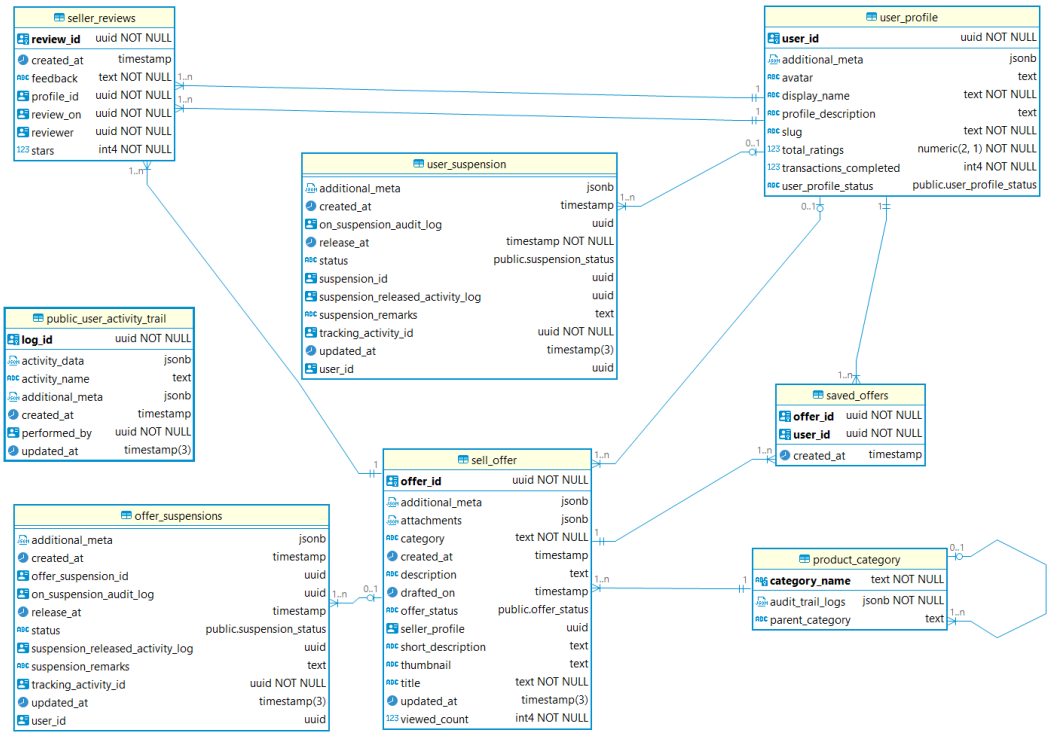


FIGURE 3.9: Marketplace ERD

#### 3.3.1.5 Chats

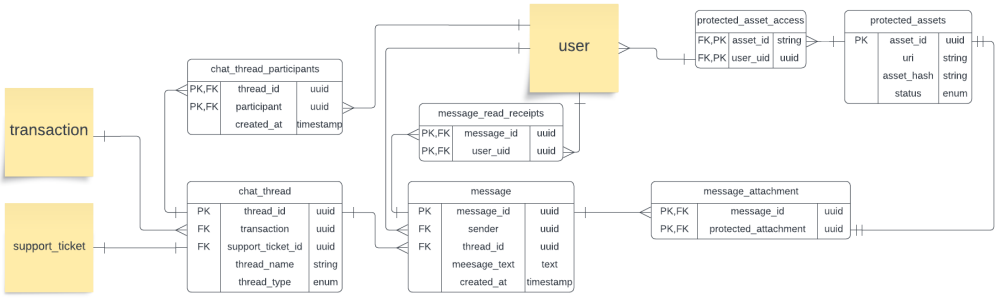


FIGURE 3.10: Chat ERD

### 3.3.2 Class Diagram

We will be using MVC structure for our backend server. For simplicity purposes we will only be including Services Classes in the Class diagrams.

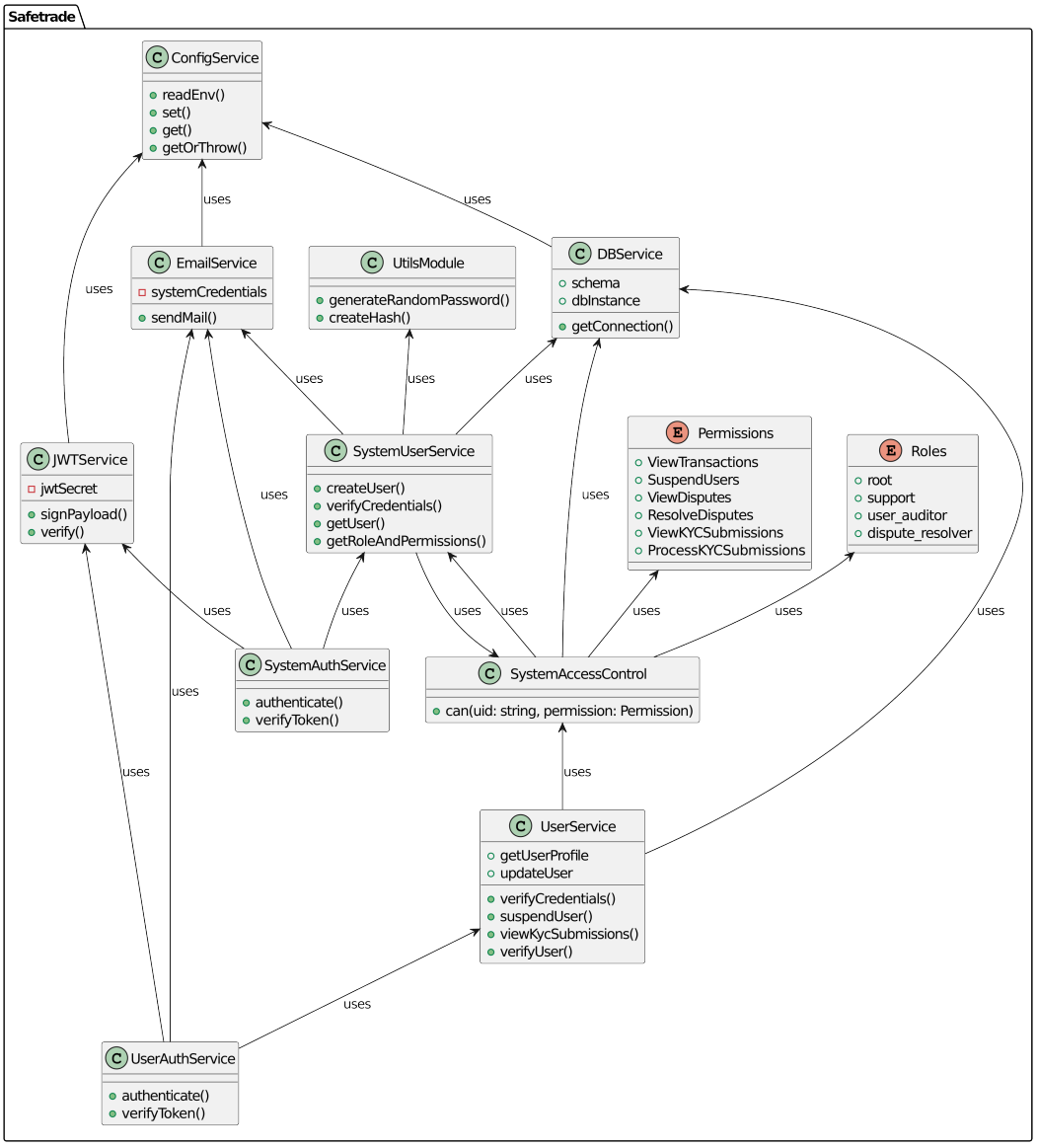


FIGURE 3.11: UserOps Class Diagram

#### 3.3.2.1 Complete diagram

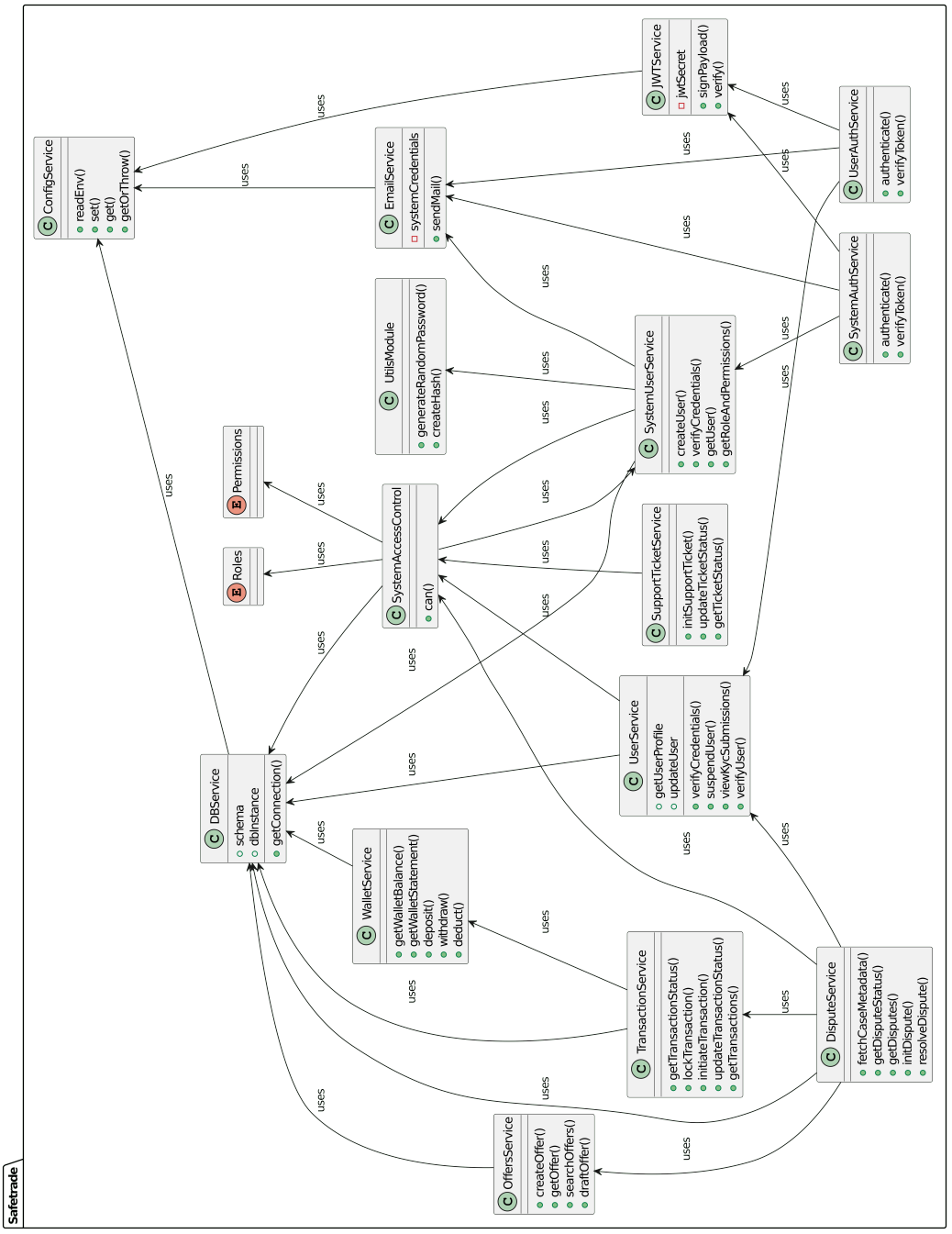


FIGURE 3.12: Safetrade Operations Class Diagram

### 3.3.3 Deployment Diagram

The deployment architecture of our application is structured to ensure scalability, high availability, and efficient management of resources. Below is a detailed

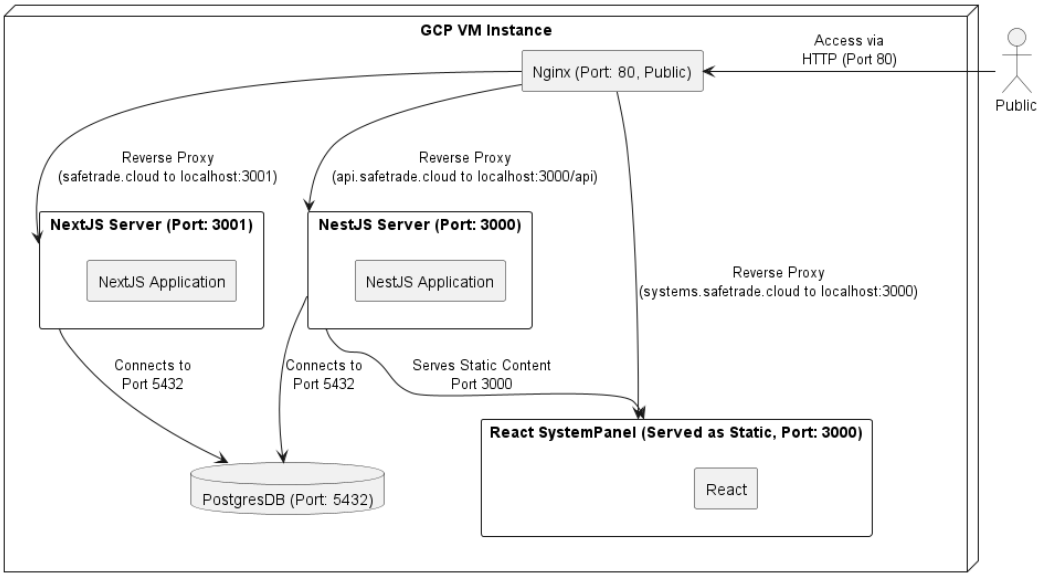


FIGURE 3.13: Deployment Diagram

explanation of the deployment flow:

#### 3.3.3.1 Google Cloud Load Balancer

The instance hosting our application is placed behind a Google Cloud Load Balancer. The load balancer serves as the entry point for all incoming traffic and distributes the traffic across multiple instances of our application to ensure balanced load distribution and high availability.

#### 3.3.3.2 Public Accessibility via Nginx

The Nginx server, listening on port 80, is configured to handle public requests. The Nginx server acts as a reverse proxy, routing the incoming requests to the appropriate services based on the subdomain and URL path. The reverse proxy rules are as follows:

- Requests to `safetrade.cloud` are forwarded to the NextJS server on `localhost:3001`.

- Requests to `api.safetrade.cloud` are forwarded to the NestJS server on `localhost:3000/api`.
- Requests to `systems.safetrade.cloud` are forwarded to the React System-Panel served statically on `localhost:3000`.

#### 3.3.3.3 NestJS and NextJS Servers

Both the NestJS and NextJS servers are managed by PM2 in a cluster environment. This setup allows for better resource utilization and fault tolerance by enabling multiple instances of the applications to run concurrently.

#### 3.3.3.4 Database Connection

The NestJS and NextJS servers connect to a Postgres database running on port 5432. This database is responsible for storing and managing all persistent data required by the applications.

#### 3.3.3.5 Scalability

As the application demand increases, we can scale the number of instances running behind the Google Cloud Load Balancer. This horizontal scaling approach ensures that the application can handle increased traffic and load without compromising on performance or availability. Additional instances can be spun up as needed, and the load balancer will automatically distribute the incoming requests among the available instances.

## 3.4 Use Cases

### 3.4.1 Use cases list

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| **Marketplace** | | |
| UCM01 | Search for an offer | Allow users to search for offers (products or services) on the platform. |
| UCM02 | View Offer Details | Enable users to view detailed information about a specific offer, including product/service details. |
| UCM03 | View Seller Details | Provide users with information about the seller of a particular offer, or seller past reviews and other offers. |
| UCM04 | Login | Allow users to log in to their accounts. |
| UCM05 | Register | Allow new users to register on the platform. |

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| UCM06 | Onboarding | Guide new users through an onboarding process, including choosing a display name, uploading a profile picture, and deciding whether they want to sell items (with KYC submission if they choose to sell). |
| UCM07 | Submits KYC documents | Allow sellers to submit KYC documents for verification. |
| UCM08 | Post a sell offer | Enable sellers to post new offers for sale. |
| UCM09 | Display saved offers | Show saved offers to authenticated users. |
| UCM10 | Saved Offers | Allow users to save offers for later viewing. |
| **Wallet** | | |
| UCW01 | Check Wallet Balance/Summary | Allow users to check their wallet balance and view a summary. |
| UCW02 | Check Wallet Statement | Provide users with a detailed statement of their wallet transactions. |

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| UCW03 | Deposit Funds | Enable users to deposit funds into their wallet. |
| UCSW01 | Deposit Verification | Verify deposited funds and mark them as successful. |
| UCW04 | Withdraw Funds | Allow users to withdraw funds from their wallet. |
| UCSW02 | Process Withdrawal | Process withdrawal requests and mark them as successful. |
| **Transaction** | | |
| UCT01 | Initiate Transaction | Facilitate the initiation of transactions between buyers and sellers. |
| UCT02 | Initiate Chat | Allow buyers and sellers to communicate via chat during a transaction. |
| UCT03 | Seller Delivers Goods | Enable sellers to mark items as delivered. |
| UCT04 | Buyer Mark as delivered | Allow buyers to confirm the delivery of items. |

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| UCT05 | Open Dispute | Allow users to open a dispute if there is an issue with the transaction. |
| **Dashboard** | | |
| UCD01 | Update profile information | Allow users to update their profile information. |
| UCD02 | Edit an offer | Enable users to edit their posted offers. |
| UCD03 | Display purchases | Show users a summary of their purchases. |
| UCD04 | Message | Facilitate communication between users and support staff through messaging. |
| UCD05 | Open Support Ticket | Allow users to create support tickets for issues related to their transactions or account. |
| **System Panel** | | |
| UCSP01 | Login to panel | Allow system users to sign in to the system panel. |

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| UCSP02 | Manage System Users | Allow root users to add, edit, and delete system users. |
| UCSP03 | View audit logs of system users | Enable root users to view activity logs of all system users. |
| UCSP04 | View support tickets | Allow support staff to view a list of support tickets. |
| UCSP05 | View Dispute Tickets | Enable support staff to view dispute tickets. |
| UCSP06 | Assign ticket to staff | Allow support staff to assign support or dispute tickets to themselves or other staff members. |
| UCSP07 | Resolve Tickets/disputes | Enable support staff to resolve support and dispute tickets. |
| UCSP08 | Chat with counterparts | Allow support staff to communicate with users and other staff members via chat. |
| UCSP09 | Approves KYC Submission | Enable audit staff to review and approve KYC submissions. |

TABLE 3.1: Use cases list

| **UCID** | **Use case name** | **Description** |
|---|---|---|
| UCSP10 | Approves Deposit | Allow accounts staff to approve deposit transactions. |
| UCSP11 | Process Withdrawal | Enable accounts staff to process and approve withdrawal requests. |

#### 3.4.1.1 Use Case: Search for an offer

Table 3.2: Use Case: Search for an Offer

| | |
|---|---|
| **use case-id:** | UCMP01 |
| **use case-name:** | Search for an Offer |
| **Description:** | This use case allows users to search for available offers using keywords. |
| **Primary-Actors:** | User |
| **Preconditions:** | Offers must be available in the database. |
| **Post-condition:** | The user is presented with a list of offers matching the search criteria. |
| **Trigger:** | The user enters keywords into the search bar and initiates the search. |
| **Secondary-Actor:** | Database |

#### 3.4.1.2 Use Case: Complete KYC Submissions

Table 3.3: Use Case: Complete KYC Submissions

| | |
|---|---|
| **use case-id:** | UCMP07 |
| **use case-name:** | Complete KYC Submissions |
| **Description:** | This use case allows sellers to submit their KYC documents for verification. |
| **Primary-Actors:** | Seller |
| **Preconditions:** | Seller must be logged into the platform.<br><br>Seller must have decided to sell items on the platform. |
| **Post-condition:** | The KYC documents are submitted and pending approval from audit staff. |
| **Trigger:** | The seller navigates to the KYC submission page and uploads the required documents. |
| **Secondary-Actor:** | Audit Staff, Database |

#### 3.4.1.3 Use Case: Approve a KYC submission

Table 3.4: Use Case: Approve KYC Submission

| | |
|---|---|
| **use case-id:** | UCSP09 |
| **use case-name:** | Approve KYC Submission |
| **Description:** | This use case allows audit staff to review and approve KYC submissions from users. |
| **Primary-Actors:** | Audit Staff |
| **Preconditions:** | Audit staff must be logged into the system panel.<br><br>There must be pending KYC submissions to review. |
| **Post-condition:** | KYC submissions are reviewed and approved or rejected. |
| **Trigger:** | The audit staff selects a pending KYC submission and performs the review. |
| **Secondary-Actor:** | Database |

#### 3.4.1.4 Use Case: Post a sell offer

Table 3.5: Use Case: Post a Sell Offer

| | |
|---|---|
| **use case-id:** | UCMP08 |
| **use case-name:** | Post a Sell Offer |
| **Description:** | This use case allows sellers to post new offers for sale. |
| **Primary-Actors:** | Seller |
| **Preconditions:** | Seller must be logged into the platform.<br><br>Seller must have completed KYC verification. |
| **Post-condition:** | The new offer is listed on the marketplace and visible to potential buyers. |
| **Trigger:** | The seller clicks on the 'Post Offer' button and submits the offer details. |
| **Secondary-Actor:** | Database |

#### 3.4.1.5 Use Case: Deposit Funds

Table 3.6: Use Case: Deposit Funds

| | |
|---|---|
| **use case-id:** | UCW03 |
| **use case-name:** | Deposit Funds |
| **Description:** | This use case allows users to deposit funds into their SafeTrade wallet. |
| **Primary-Actors:** | User |
| **Preconditions:** | User must be logged into the platform. |
| **Post-condition:** | Funds are credited to the user's SafeTrade wallet balance. |
| **Trigger:** | The user selects 'Deposit Funds' and attaches the bank transfer receipt. |
| **Secondary-Actor:** | Accounts Staff, Database |

#### 3.4.1.6 Use Case: Initiate Transaction

Table 3.7: Use Case: Initiate Transaction

| | |
|---|---|
| **use case-id:** | UCT01 |
| **use case-name:** | Initiate Transaction |
| **Description:** | This use case allows users to initiate a transaction for purchasing an offer. |
| **Primary-Actors:** | Buyer |
| **Preconditions:** | Buyer must be logged into the platform.<br><br>Buyer must have sufficient funds in their SafeTrade wallet. |
| **Post-condition:** | The transaction is initiated, and funds are held in escrow until the seller fulfills the order. |
| **Trigger:** | The buyer clicks on the 'Buy Now' button and confirms the purchase. |
| **Secondary-Actor:** | Seller, Database |

#### 3.4.1.7 Use Case: Open Dispute

Table 3.8: Use Case: Open Dispute

| | |
|---|---|
| **use case-id:** | UCT05 |
| **use case-name:** | Open Dispute |
| **Description:** | This use case allows users to open a dispute if there is an issue with a transaction. |
| **Primary-Actors:** | Buyer, Seller |
| **Preconditions:** | The transaction must be in progress or completed.<br><br>The user must have a valid reason for opening the dispute. |
| **Post-condition:** | The dispute is recorded, and support staff are notified. |
| **Trigger:** | The user clicks on the 'Open Dispute' button and submits details of the issue. |
| **Secondary-Actor:** | Support Staff, Database |

#### 3.4.1.8 Use Case: Resolve Tickets/Disputes

Table 3.9: Use Case: Resolve Tickets/Disputes

| | |
|---|---|
| **use case-id:** | UCSP07 |
| **use case-name:** | Resolve Tickets/Disputes |
| **Description:** | This use case allows support staff to resolve support tickets and disputes raised by users. |
| **Primary-Actors:** | Support Staff |
| **Preconditions:** | Support staff must be logged into the system panel.<br>There must be open support tickets or disputes. |
| **Post-condition:** | The support tickets or disputes are resolved,<br>and users are notified of the resolution.<br>The locked amount is refunded to the buyer or released<br>to the seller after resolution. |
| **Trigger:** | The support staff selects an open ticket or dispute and takes necessary action to resolve it. |
| **Secondary-Actor:** | Buyer, Seller, Database |

#### 3.4.1.9 Use Case: Process Withdrawal

Table 3.10: Use Case: Process Withdrawal

| | |
|---|---|
| **use case-id:** | UCW05 |
| **use case-name:** | Process Withdrawal |
| **Description:** | This use case allows accounts staff to process withdrawal requests from users. |
| **Primary-Actors:** | Accounts Staff |
| **Preconditions:** | Accounts staff must be logged into the system panel.<br><br>There must be pending withdrawal requests from users. |
| **Post-condition:** | Withdrawal requests are processed and funds are transferred to the users. |
| **Trigger:** | The accounts staff selects a pending withdrawal request and processes it. |
| **Secondary-Actor:** | Database, Bank System |

#### 3.4.1.10 Use Case: Initiate Support Ticket

Table 3.11: Use Case: Initiate Support Ticket

| | |
|---|---|
| **use case-id:** | UCSD05 |
| **use case-name:** | Initiate Support Ticket |
| **Description:** | This use case allows users to initiate a support ticket in case of any issues. |
| **Primary-Actors:** | User |
| **Preconditions:** | User must be logged into the platform. |
| **Post-condition:** | A support ticket is created and assigned to support staff for resolution. |
| **Trigger:** | The user clicks on 'Open Support Ticket' and submits details of the issue. |
| **Secondary-Actor:** | Support Staff, Database |

# Chapter 4

# Implementation and Evaluation

This chapter defines the application design phase. The design patterns, methods and processes to be used in project design are listed in this chapter.

## 4.1 Implementation

### 4.1.1 Planning

The SafeTrade project was meticulously planned to ensure a robust, scalable, and maintainable system. Our codebase is structured as a monorepo, integrating multiple components to streamline development and deployment. The frontend is divided into two primary sections: the system panel and the client-facing application. For the system panel, we opted for React due to its efficient client-side rendering (SPA) capabilities, as SEO was not a priority, and server-side rendering was unnecessary. For the client-facing application, Next.js was chosen to leverage server-side rendering and server actions, providing an enhanced user experience and improved performance. On the backend, we utilized NestJS, a framework built on top of Express, to ensure maintainability and scalability as the application grows. We selected TypeScript for type safety across our codebase and

implemented DrizzleORM with PostgreSQL for robust and reliable database management. Additional tools such as PM2 for process management, Nginx as our reverse proxy, and TailwindCSS for styling were incorporated to ensure smooth operation and aesthetic consistency. To facilitate seamless API development, we employed ts-rest and TRPC. Spam Detection module is for future reference in the figure below.

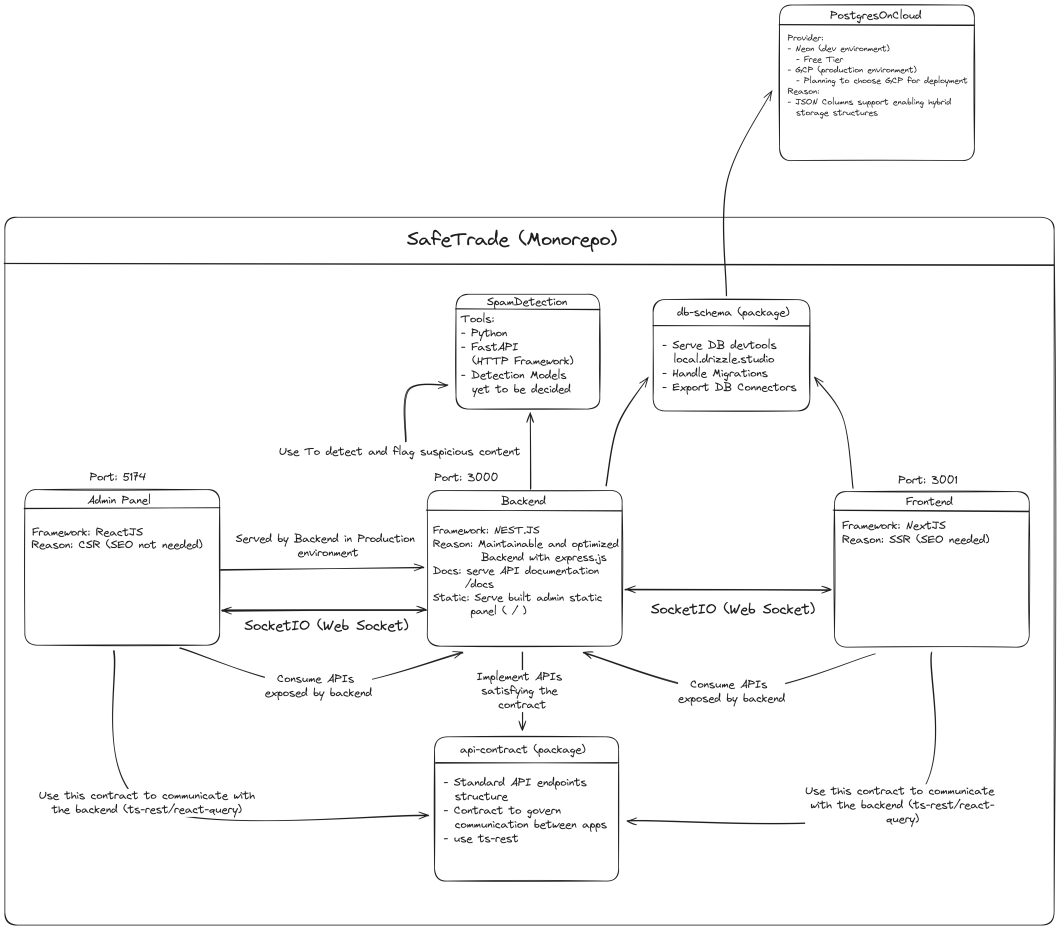


FIGURE 4.1: Safetrade Architecturel

## 4.2 Languages and Tools

### 4.2.1 React

React was chosen for our system panel to provide a single-page application (SPA) experience. Its client-side rendering capabilities were ideal since SEO was not a

concern, and there was no need for server-side rendering. React's component-based architecture also enhances code reusability and maintainability.

### 4.2.2 NestJS

NestJS, built on top of Express, was selected for our backend framework due to its structured and developer-friendly patterns, which facilitate application scalability and maintainability. Using TypeScript with NestJS ensures type safety and reliable API development, making the codebase easier to manage and less prone to errors.

### 4.2.3 NextJS

For our client-facing application, Next.js was the optimal choice, allowing us to utilize server-side rendering and server actions. This approach improves performance and user experience by delivering pre-rendered pages, reducing load times, and enhancing SEO capabilities.

### 4.2.4 DrizzleORM and PostgreSQL

DrizzleORM was chosen alongside PostgreSQL to manage our database needs. DrizzleORM provides a type-safe and flexible ORM solution, while PostgreSQL offers a robust, reliable, and scalable database system.

### 4.2.5 PM2

PM2 is used as our process manager to ensure that our application runs smoothly and efficiently. It provides features like process monitoring, auto-restarting on failure, and load balancing, which are crucial for maintaining high availability and performance.

### 4.2.6 Nginx

Nginx serves as our reverse proxy, handling client requests and distributing them to the appropriate backend services. It improves security, load balancing, and scalability, ensuring that our application can handle high traffic and provide a seamless user experience.

### 4.2.7 TailwindCSS

TailwindCSS is utilized for styling our application. Its utility-first approach allows for rapid development and consistent design across the application, making it easier to maintain and extend the styling as the project evolves.

### 4.2.8 ts-rest and TRPC

These tools are used for seamless API development. ts-rest and TRPC enable type-safe, end-to-end type sharing between the client and server, reducing the risk of errors and improving the overall developer experience.

### 4.2.9 TypeScript

TypeScript is used throughout our codebase to ensure type safety and reliability. By catching errors at compile time, it enhances code quality and reduces the likelihood of runtime issues, making the development process more efficient and the final product more robust.

### 4.2.10 Functional Testing

The system's functionality is tested and checked according to the business requirements. A number of test cases are made with possible inputs and expected outputs and with bugs or errors if any. The modules of the application which we will be testing are given next.

### 4.2.11 Types of testing

In this phase, we tested our project's frontend and backend using different approaches.

**Unit Testing**

Unit testing is the practice of evaluating isolated code components or units to verify their correct functioning. In our project, this entails assessing the performance of functions responsible for tasks such as timetable generation and user authentication.

**Integration Testing**

Integration testing evaluates how various modules or components interact to confirm their smooth coordination. We will employ this testing method to guarantee effective integration among the frontend, backend, and database elements of our application.

**Functional Testing**

Functional testing evaluates the functional aspects of our application, such as user authentication, timetable generation, and user profile management, to ensure they meet specified requirements and user expectations.

### User Interface Testing

UI testing focuses on the visual and interactive aspects of our application. It ensures that the user interface is user-friendly, responsive, and functions correctly across various devices and browsers.

### Regression testing

Regression testing is a process that validates whether recent modifications to the codebase have not introduced new issues or negatively impacted existing features [4]. It serves as a safeguard to maintain the application's stability as it undergoes development and changes over time.

# Test Cases

Table 4.1: Test cases list

| **TCID** | **Name** | **Description** | **Expected Result** |
|---|---|---|---|
| **Marketplace** | | | |
| TCMP01 | Search for an offer | Verify that users can search for offers using keywords. | Search results displayed |
| TCMP02 | View Offer Details | Verify that users can view the details of a selected offer. | Offer details displayed |
| TCMP03 | View Seller Details | Verify that users can view details about the seller of an offer. | Seller details displayed |
| TCMP04 | Login | Verify that users can log in with valid credentials. | User logged in |
| TCMP05 | Register | Verify that new users can register with valid details. | User registered |
| TCMP06 | Onboarding | Verify that new users are guided through the onboarding process. | Onboarding completed |

| | | | |
|---|---|---|---|
| TCMP07 | Submit KYC documents | Verify that sellers can submit KYC documents. | KYC documents submitted |
| TCMP08 | Post a sell offer | Verify that sellers can post new offers. | Offer posted |
| TCMP09 | Display saved offers | Verify that users can view their saved offers. | Saved offers displayed |
| TCMP10 | Save Offer | Verify that users can save an offer for later viewing. | Offer saved |
| **Wallet** | | | |
| TCW01 | Check Wallet Balance/Summary | Verify that users can check their wallet balance and view a summary. | Wallet balance displayed |
| TCW02 | Check Wallet Statement | Verify that users can view a detailed statement of their wallet transactions. | Wallet statement displayed |
| TCW03 | Deposit Funds | Verify that users can deposit funds into their wallet. | Funds deposited |
| TCW04 | Deposit Verification | Verify that deposited funds are verified and marked as successful. | Deposit verified |

| | | | |
|---|---|---|---|
| TCW05 | Withdraw Funds | Verify that users can withdraw funds from their wallet. | Funds withdrawn |
| TCW06 | Process Withdrawal | Verify that withdrawal requests are processed successfully. | Withdrawal processed |
| **Transaction** | | | |
| TCT01 | Initiate Transaction | Verify that transactions can be initiated between buyers and sellers. | Transaction initiated |
| TCT02 | Initiate Chat | Verify that buyers and sellers can communicate via chat during a transaction. | Chat initiated |
| TCT03 | Seller Delivers Goods | Verify that sellers can mark items as delivered. | Item marked as delivered |
| TCT04 | Buyer Marks as Delivered | Verify that buyers can confirm the delivery of items. | Delivery confirmed |
| TCT05 | Open Dispute | Verify that users can open a dispute if there is an issue with the transaction. | Dispute opened |
| **Dashboard** | | | |

| | | | |
|---|---|---|---|
| TCD01 | Update Profile Information | Verify that users can update their profile information. | Profile updated |
| TCD02 | Edit an Offer | Verify that users can edit their posted offers. | Offer edited |
| TCD03 | Display Purchases | Verify that users can view a summary of their purchases. | Purchases displayed |
| TCD04 | Message Support | Verify that users can communicate with support staff through messaging. | Message sent |
| TCD05 | Open Support Ticket | Verify that users can create support tickets for issues related to their transactions or account. | Support ticket created |
| **System Panel** | | | |
| TCSP01 | Login to Panel | Verify that system users can sign in to the system panel. | User logged in |
| TCSP02 | Manage System Users | Verify that root users can add, edit, and delete system users. | System users managed |

| | | | |
|---|---|---|---|
| TCSP03 | View Audit Logs | Verify that root users can view activity logs of all system users. | Audit logs displayed |
| TCSP04 | View Support Tickets | Verify that support staff can view a list of support tickets. | Support tickets displayed |
| TCSP05 | View Dispute Tickets | Verify that support staff can view dispute tickets. | Dispute tickets displayed |
| TCSP06 | Assign Ticket to Staff | Verify that support staff can assign support or dispute tickets to themselves or other staff members. | Ticket assigned |
| TCSP07 | Resolve Tickets/Disputes | Verify that support staff can resolve support and dispute tickets. | Ticket resolved |
| TCSP08 | Chat with Counterparts | Verify that support staff can communicate with users and other staff members via chat. | Chat initiated |
| TCSP09 | Approve KYC Submission | Verify that audit staff can review and approve KYC submissions. | KYC approved |

| | | | |
|---|---|---|---|
| TCSP10 | Approve Deposit | Verify that accounts staff can approve deposit transactions. | Deposit approved |
| TCSP11 | Process Withdrawal | Verify that accounts staff can process and approve withdrawal requests. | Withdrawal processed |

#### 4.2.11.1 Test Case Deposit Funds

Table 4.2: TCW03: Deposit Funds

| | |
|---|---|
| **Test Case ID** | TCW03 |
| **Title** | Deposit Funds |
| **Purpose** | To verify that users can deposit funds into their wallet. |
| **Precondition** | The user is logged in and on the wallet deposit page. |
| **Test Data** | • User account details<br>• Deposit amount |
| **Steps to Perform** | 1. Navigate to the wallet deposit page.<br>2. Enter the deposit amount.<br>3. Select the payment method.<br>4. Confirm the deposit. |
| **Expected Result** | The funds are deposited into the user's wallet and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the deposit was successful.<br>• The wallet balance is updated accordingly. |
| **Status** | Pass |

#### 4.2.11.2 Test Case Withdraw Funds

Table 4.3: TCW05: Withdraw Funds

| | |
|---|---|
| **Test Case ID** | TCW05 |
| **Title** | Withdraw Funds |
| **Purpose** | To verify that users can withdraw funds from their wallet. |
| **Precondition** | The user is logged in and on the wallet withdrawal page. |
| **Test Data** | • User account details<br>• Withdrawal amount |
| **Steps to Perform** | 1. Navigate to the wallet withdrawal page.<br>2. Enter the withdrawal amount.<br>3. Select the withdrawal method.<br>4. Confirm the withdrawal. |
| **Expected Result** | The funds are withdrawn from the user's wallet and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the withdrawal was successful.<br>• The wallet balance is updated accordingly. |
| **Status** | Pass |

#### 4.2.11.3 Test Case Submit KYC documents

Table 4.4: TCMP07: Submit KYC documents

| | |
|---|---|
| **Test Case ID** | TCMP07 |
| **Title** | Submit KYC documents |
| **Purpose** | To verify that sellers can submit KYC documents. |
| **Precondition** | The user is logged in and on the KYC submission page. |
| **Test Data** | • User identification documents |
| **Steps to Perform** | 1. Navigate to the KYC submission page.<br>2. Upload the required identification documents.<br>3. Submit the documents for verification. |
| **Expected Result** | The KYC documents are submitted successfully and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the KYC documents were submitted successfully. |
| **Status** | Pass |

#### 4.2.11.4 Test Case Open Dispute

Table 4.5: TCT05: Open Dispute

| | |
|---|---|
| **Test Case ID** | TCT05 |
| **Title** | Open Dispute |
| **Purpose** | To verify that users can open a dispute if there is an issue with the transaction. |
| **Precondition** | The user is logged in and has a transaction with an issue. |
| **Test Data** | • Transaction details<br>• Issue description |
| **Steps to Perform** | 1. Navigate to the transaction details page.<br>2. Click on the "Open Dispute" button.<br>3. Enter the issue description.<br>4. Submit the dispute. |
| **Expected Result** | The dispute is opened successfully and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the dispute was opened successfully. |
| **Status** | Pass |

#### 4.2.11.5 Test Case Post a sell offer

Table 4.6: TCMP08: Post a sell offer

| | |
|---|---|
| **Test Case ID** | TCMP08 |
| **Title** | Post a sell offer |
| **Purpose** | To verify that sellers can post new offers. |
| **Precondition** | The user is logged in and on the sell offer posting page. |
| **Test Data** | • Offer details |
| **Steps to Perform** | 1. Navigate to the sell offer posting page.<br>2. Enter the offer details.<br>3. Submit the offer. |
| **Expected Result** | The sell offer is posted successfully and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the sell offer was posted successfully. |
| **Status** | Pass |

#### 4.2.11.6 Test Case Search for an offer

Table 4.7: TCMP01: Search for an offer

| | |
|---|---|
| **Test Case ID** | TCMP01 |
| **Title** | Search for an offer |
| **Purpose** | To verify that users can search for offers using keywords. |
| **Precondition** | The user is on the marketplace search page. |
| **Test Data** | • Search keywords |
| **Steps to Perform** | 1. Navigate to the marketplace search page.<br>2. Enter search keywords.<br>3. Click the search button. |
| **Expected Result** | The search results are displayed successfully. |
| **Actual Result** | • The system displays the search results matching the keywords. |
| **Status** | Pass |

#### 4.2.11.7 Test Case Initiate Transaction

Table 4.8: TCT01: Initiate Transaction

| | |
|---|---|
| **Test Case ID** | TCT01 |
| **Title** | Initiate Transaction |
| **Purpose** | To verify that transactions can be initiated between buyers and sellers. |
| **Precondition** | The user is logged in and has selected an offer to purchase. |
| **Test Data** | • Offer details |
| **Steps to Perform** | 1. Navigate to the selected offer page.<br>2. Click on the "Buy Now" or equivalent button.<br>3. Confirm the transaction details.<br>4. Initiate the transaction. |
| **Expected Result** | The transaction is initiated successfully and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the transaction was initiated successfully. |
| **Status** | Pass |

#### 4.2.11.8 Test Case Seller Delivers Goods

Table 4.9: TCT03: Seller Delivers Goods

| | |
|---|---|
| **Test Case ID** | TCT03 |
| **Title** | Seller Delivers Goods |
| **Purpose** | To verify that sellers can mark items as delivered. |
| **Precondition** | The user is logged in and has a transaction in progress. |
| **Test Data** | • Transaction details |
| **Steps to Perform** | 1. Navigate to the transaction details page.<br>2. Click on the "Mark as Delivered" button. |
| **Expected Result** | The item is marked as delivered successfully and a confirmation message is displayed. |
| **Actual Result** | • The system displays a success message that the item was marked as delivered. |
| **Status** | Pass |

## 4.3 User Interface

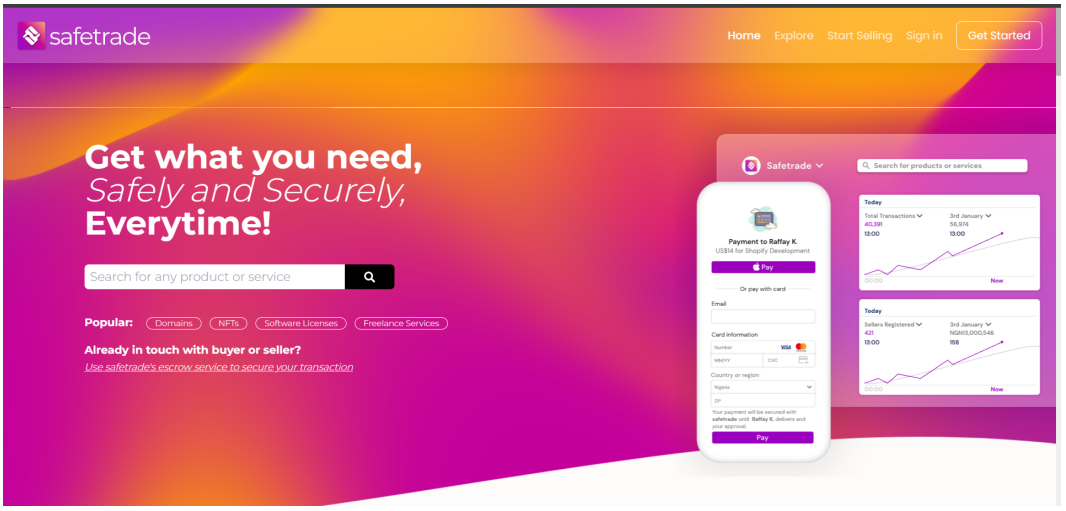


FIGURE 4.2: Safetrade Homepage

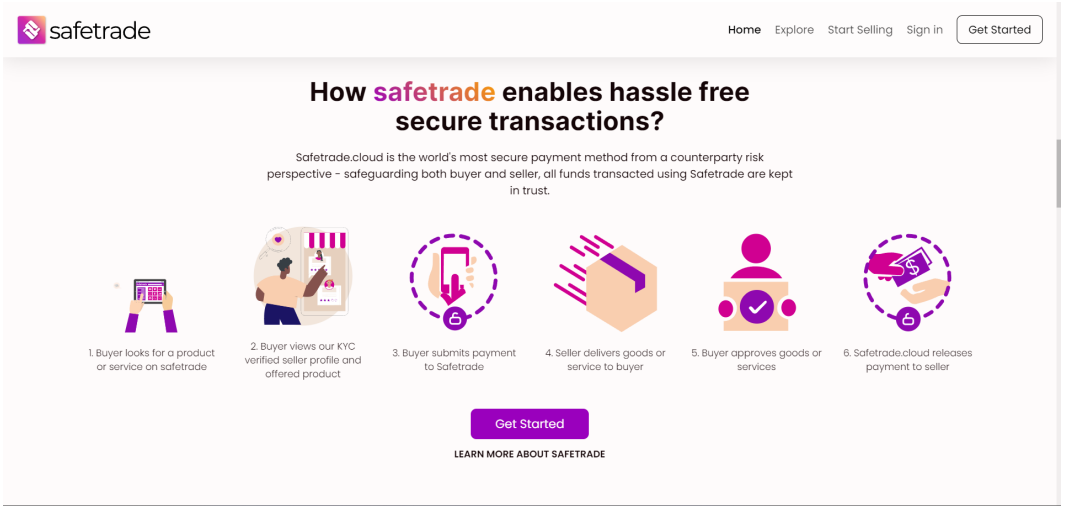


FIGURE 4.3: Safetrade transaction steps

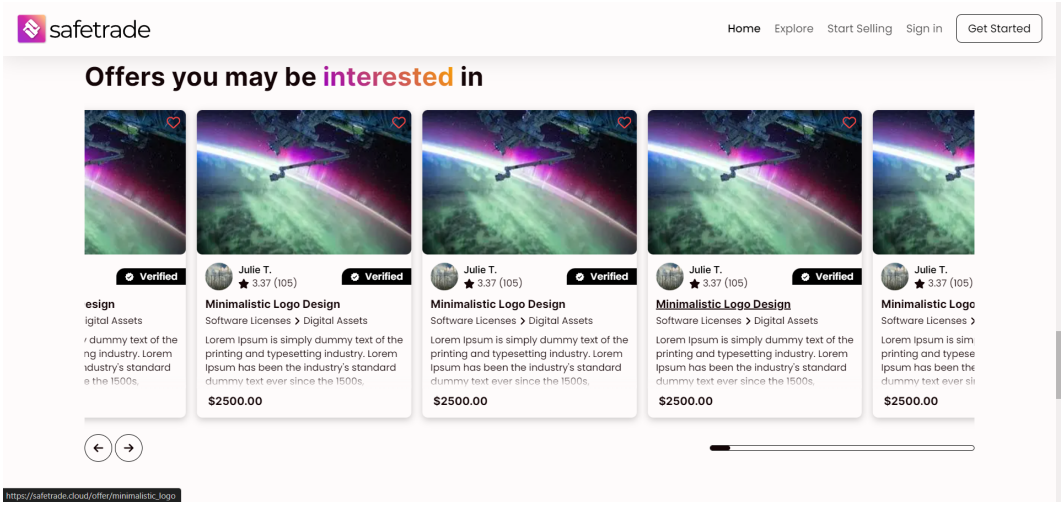


FIGURE 4.4: Homepage offers section

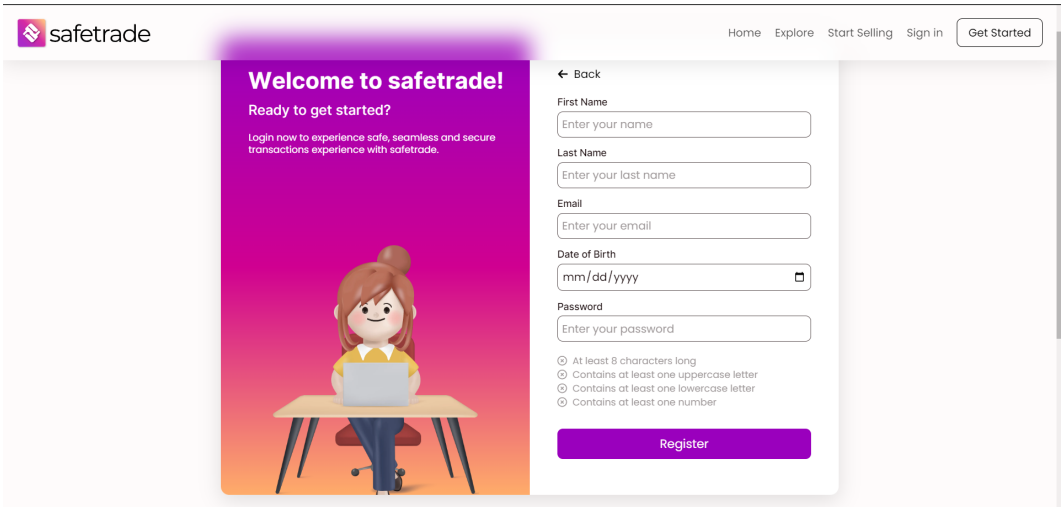


FIGURE 4.5: Safetrade Signup page

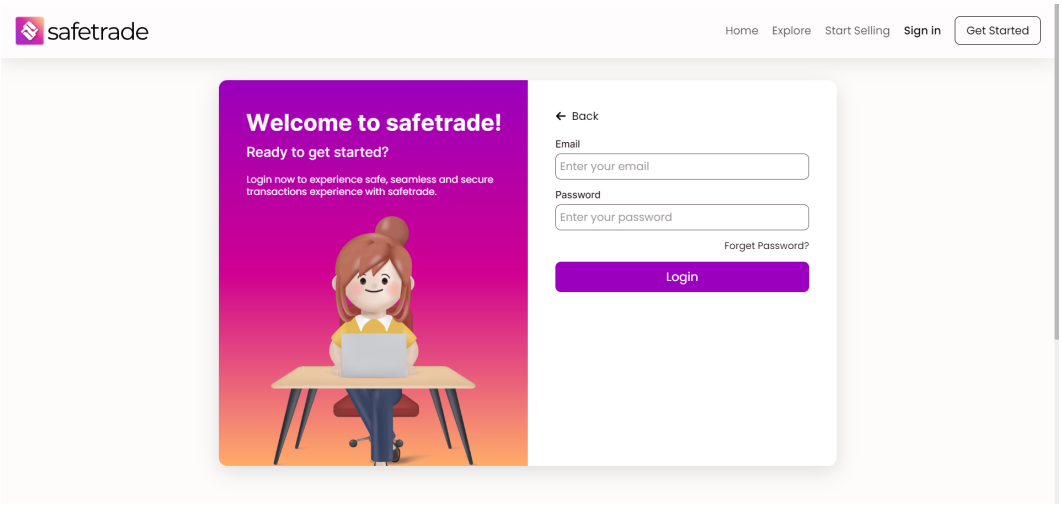


FIGURE 4.6: Safetrade login page

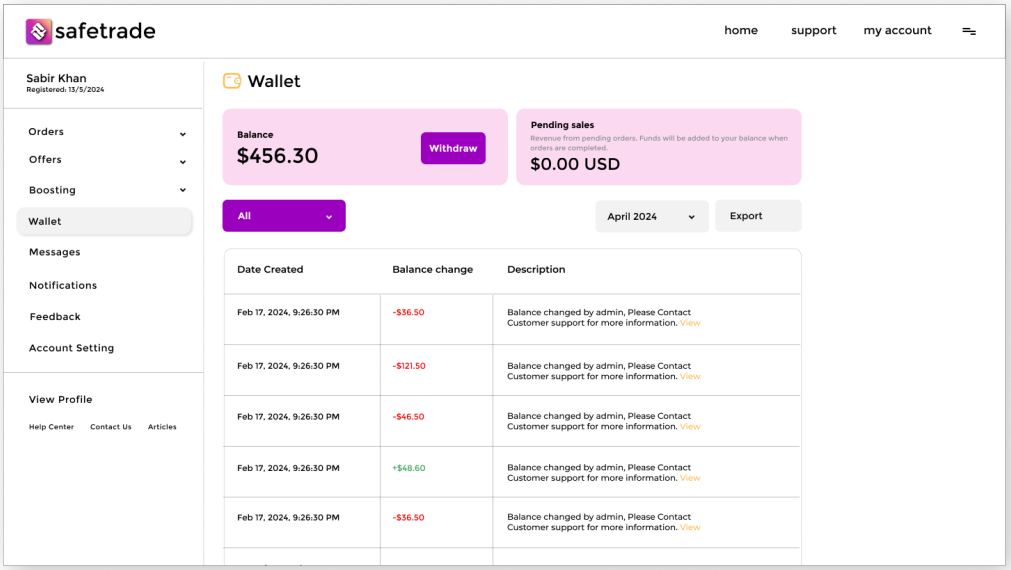


FIGURE 4.7: Safetrade Wallet

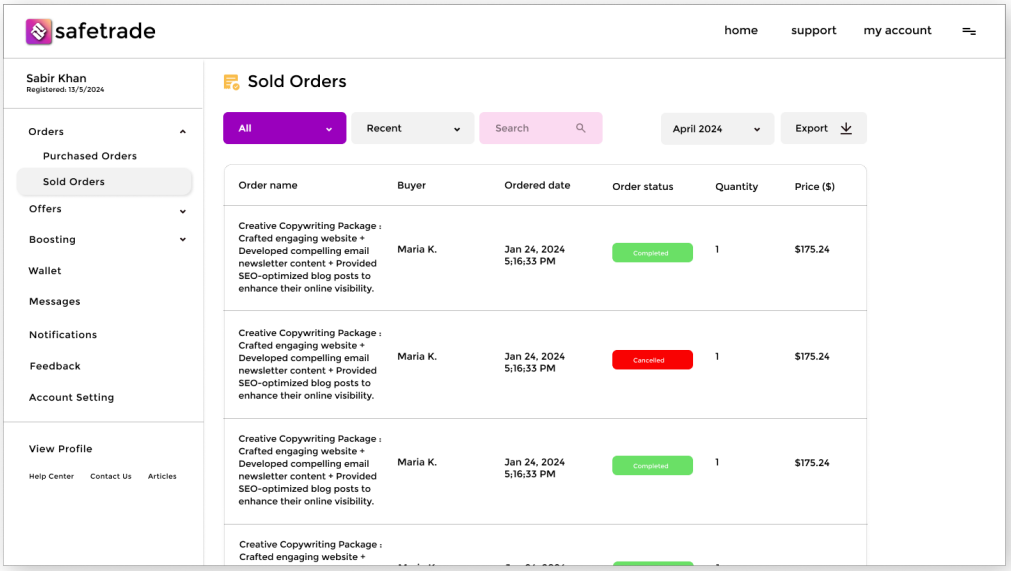


FIGURE 4.8: Safetrade orders dashboard

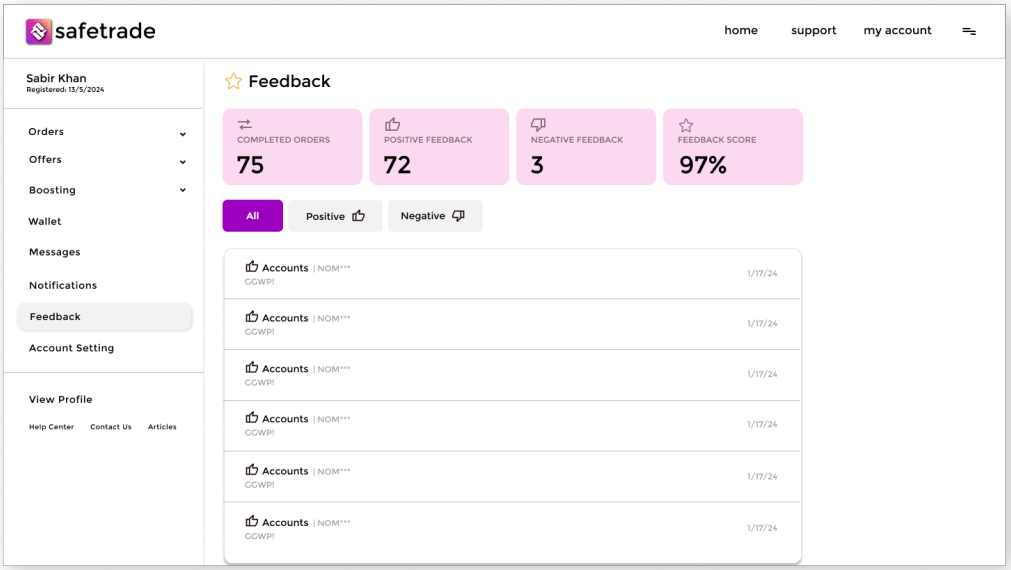


FIGURE 4.9: Safetrade feedback dashboard

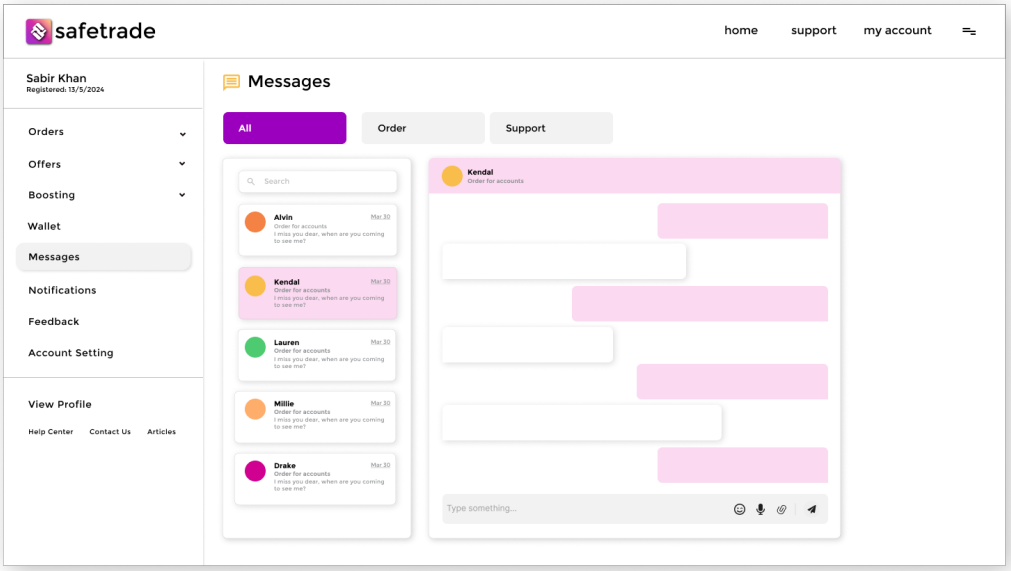


FIGURE 4.10: Safetrade messages

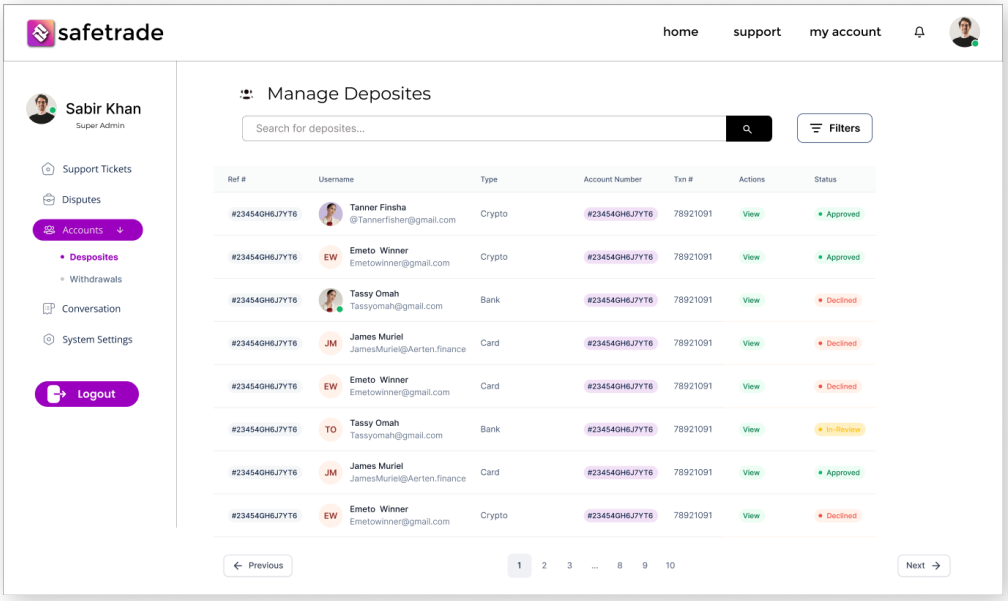


FIGURE 4.11: Safetrade system panel: deposits

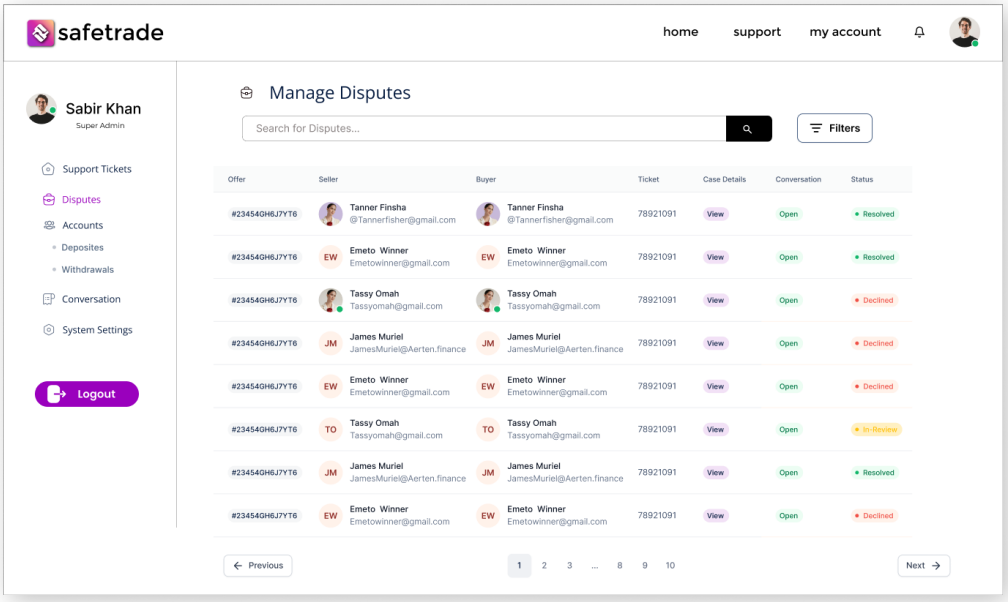


FIGURE 4.12: Safetrade system panel: disputes

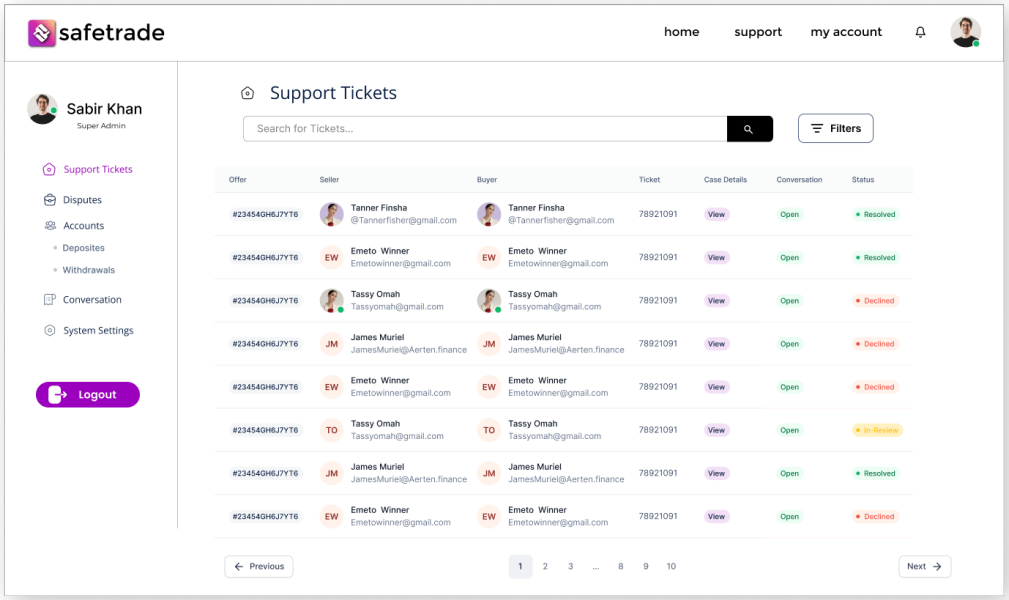


FIGURE 4.13: Safetrade system panel: support tickets

# Chapter 5

# Conclusion & Future Work

## 5.1 Conclusion

SafeTrade represents a significant advancement in secure online transactions for both physical and digital goods. By incorporating key features such as escrow protection, KYC verification, and a robust dispute resolution system, SafeTrade mitigates counterparty risks and ensures a secure marketplace environment. The platform's emphasis on security and user trust fosters a safe trading experience, which is crucial in the modern digital economy.

The successful implementation of features such as a built-in wallet for seamless payment operations, a comprehensive customer support system, and detailed seller profiles enhances the overall functionality and user experience. Additionally, the development of a robust system panel for administrators and staff ensures efficient management and operational oversight.

## 5.2 Future Work

While SafeTrade has successfully addressed many critical aspects of secure online transactions, there are several areas for future enhancement:

- **Integration of Additional Payment Methods:** Expanding the range of supported payment methods, including cryptocurrencies where legally permissible, to cater to a broader user base and provide more flexibility in transactions.
- **Mobile Application Development:** Developing native mobile applications for both Android and iOS platforms to provide users with a more accessible and convenient way to engage with SafeTrade.
- **Advanced Fraud Detection Algorithms:** Implementing machine learning algorithms to enhance fraud detection and prevention capabilities, thereby increasing the overall security of the platform.
- **Global Expansion:** Adapting the platform to comply with international regulations and expanding its services to global markets, ensuring that SafeTrade can serve a diverse and international user base.
- **User Experience Enhancements:** Continuously improving the user interface and experience based on user feedback, ensuring that the platform remains user-friendly and intuitive.
- **Scalability Improvements:** Enhancing the platform's architecture to handle increased traffic and transaction volumes as SafeTrade grows, ensuring that performance remains robust and reliable.
- **Blockchain Integration:** Exploring the integration of blockchain technology for additional transparency and security in transaction processing.
- **Enhanced Data Privacy Measures:** Continuously updating and improving data privacy measures to comply with evolving privacy regulations and to protect user data effectively.

By focusing on these areas, SafeTrade can continue to evolve and meet the changing needs of its users, maintaining its position as a trusted platform for secure online transactions.

# Bibliography

[1] A. Magliocchetti J. Garcia and P. Molli. *An Evaluation of Modern Web Application Frameworks*. 2017.

[2] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2002.

[3] Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. 2011.

[4] M. R. Meybodi T. Hemmati, H. Haghighi. *A Study of Test Case Prioritization Techniques in Regression Testing*. 2009.

# Index