American College of Technology (ACT)

Department of Computer Science

Postgraduate Project

Ethiopian Stock Market Simulation Platform

Name	ID
Biniyam Kefelegn	010/2022
Sebele Fekede	074/2022
Tesfaye Fekadu	083/2022

18-JAN-2025

American College of Technology

Department of Computer Science

Postgraduate Project

Ethiopian Stock Market Simulation Platform

Examined by

Name	Signature	Date
1.________________	____________	___________
2.________________	____________	___________

For Advisor

1.________________	____________	___________

For coordinator

1.________________	____________	___________

Submitted to department of Computer Science

Acknowledgment

We express our heartfelt gratitude to the Almighty for guiding us through every challenge encountered on this journey.

Our deepest appreciation goes to our advisor, Dr. Anmaw, for providing invaluable guidance, insights, and unwavering support throughout the project. We are also grateful to the Department of Computer Science for offering us this unique opportunity to work as a team—a collaborative experience that will undoubtedly enrich our future careers.

We are proud to highlight that our project represents a pioneering effort in our country, serving as Ethiopia's first stock market simulation platform. This innovative endeavor not only provides a glimpse into what a fully functional stock market could look like but also sets a precedent for future developments in the field.

Our sincere thanks also extend to all the instructors who directed us on the right path, as well as to the staff members and friends whose constructive suggestions and constant support have been instrumental. While we have strived to develop a comprehensive system, we recognize that there is always room for improvement. We warmly welcome any feedback that can help us refine and enhance the system further.

Chapter 1: Introduction ... 1
1.1 Background of the project ... 1
1.2 Statement of the Problem ... 1
1.3 Objectives of the project ... 2
1.3.1 General Objective: ... 2
1.3.2 Specific Objectives: ... 2
1.4 Scope of the project ... 3
1.5 Limitation of the project ... 3
1.6 System Development Methodology ... 3
1.6.1 System Development Approach ... 3
1.6.2 Requirement Analysis and Design ... 4
1.6.3 System Development Tools ... 5
1.7 Significance of the Project ... 6
1.8 Beneficiaries of the Project ... 6
1.9 Feasibility Study ... 7
1.9.1 Technical Feasibility ... 7
1.9.2 Operational Feasibility ... 8
1.9.3 Economic Feasibility ... 9
1.10 Project schedule ... 10
1.11 Project Budget ... 12
Chapter 2: Requirement Analysis ... 13
2.1 Current System Description ... 13
2.1.1 Major function of the current system ... 13
2.1.2 Problem of Existing System ... 13

2.2 Requirement Gathering 14
2.2.1 Requirement Gathering Methods 14
2.2.2 Business Rules 14
2.3 Proposed System Description 15
2.3.1 Overview 15
2.3.2 Functional Requirements 16
2.3.3 Nonfunctional Requirements 17
Chapter 3: System Model 19
3.1 Scenarios 19
3.1.1 Use Case Model 19
3.1.2 Use Case Diagram 21
3.1.1. Description of use case diagram 23
3.1.3 Activity Diagram 27
3.1.4 Class Model 30
3.1.5 Data Dictionary 31
3.1.6 Sequence Diagram 38
Chapter 4: System Design 41
4.1 Introduction 41
4.2 Proposed Software Architecture 41
4.2.1 System Decomposition 41
4.2.2 Hardware and Software Mapping 42
4.2.3 Persistent Data Modeling 44
4.2.4 Access Control and Security 45
4.2.5 Packages diagram 48
4.2.6 Design Pattern (Architecture Layers) 50

4.2.7 Object Design Document (ODD) 52

4.2.8 User Interface Design 55

Chapter 5: Implementation 61

5.1 Mapping Models to Code 61

5.1.1 Users App 61

5.1.2 Stocks App 63

5.2 Suspicious Activity App 65

5.3 Testing and Evaluation 66

5.4 System Maintenance 67

Chapter 6: Conclusion and Recommendation 67

6.1 Conclusion 67

6.2 Recommendation 68

Reference 71

Figure 1 Use case diagram.. 22
Figure 2 User Registration Activity.. 27
Figure 3 User Login Activity Diagram.. 28
Figure 4: Stock Direct Purchase by Trader.. 29
Figure 5 Class Model.. 30
Figure 6 User Registration Sequence Diagram.. 39
Figure 7 User Login Sequence Diagram... 40
Figure 8 Direct Stock Purchase Sequence Diagram 41
Figure 9 Hardware and software mapping.. 43
Figure 10 Persistent data modeling.. 45
Figure 11 Package diagram.. 48
Figure 12 Login Page.. 56
Figure 13 Registration Page.. 56
Figure 14 Listed Company stock list 57
Figure 15 Placing Bid order... 58

Table 1 Project Schedule 11
Table 2 Project budget 12
Table 3 trader operation use case 23
Table 4 Listed Company Administrator Operations use case 25
Table 5 Regulator Operations 25
Table 6 Trading Engine Operations 26
Table 7 Company 31
Table 8 Dividend 31
Table 9 Disclosure 32
Table 10 Stock 32
Table 11 Orders 33
Table 12 Trade 34
Table 13 Suspicious Activity 35
Table 14 User Table 37
Table 15 Notification 37
Table 16 Audit Log 38
Table 17 Access Control and Security 47
Table 18 Class Interface 55

Chapter 1: Introduction

1.1 Background of the project

The Ethiopian Capital Market Authority (ECMA) is establishing a formal securities exchange to stimulate economic growth and provide a platform for raising capital. This marks a transformative step in Ethiopia's financial landscape, offering unprecedented opportunities for investors, brokers, and regulators. However, the Ethiopian financial market is still in its infancy, and a significant lack of practical experience among stakeholders threatens the success of this initiative.

The Stock Market Simulation Platform is designed to address this gap by providing an interactive and practical learning environment. This platform mimics the functionality of a real stock exchange, allowing users to simulate trading activities, understand market behavior, and test regulatory policies without real-world financial risks. It empowers participants to gain hands-on experience in stock trading, market analysis, and regulatory compliance. [1]

By fostering understanding and market readiness among traders, listed company admins, and the regulators, this platform plays a critical role in supporting the successful launch of Ethiopia's stock market.

1.2 Statement of the Problem

Ethiopia's new financial market faces challenges in developing skilled and knowledgeable participants. Due to a lack of practical experience, those participants may not be well-prepared to handle real-life situations related to trading, market analysis, and regulatory oversight. The absence of a structured educational tool makes it difficult for stakeholders to understand key market operations and comply with regulatory standards.

There is a need for an interactive simulation tool that provides practical experience in stock market trading, market analysis, and regulatory compliance, following the guidelines set by the ECMA.

1.3 Objectives of the project

1.3.1 General Objective:

To develop an interactive, user-friendly stock market simulation platform that prepares Ethiopian traders, companies and regulators for the upcoming Ethiopian stock exchange.

1.3.2 Specific Objectives:

1. Role-Based Registration and Simulation: Design a platform that allows users to register under specific roles (e.g., Traders, Listed Company Admins) and actively participate in simulated stock market activities.
2. Advanced Trading Engine: Develop a trading engine supporting two key order types (market and limit orders) with real-time matching and execution capabilities based on the Price-Time Priority Algorithm.
3. Regulatory Monitoring and Testing Module: Create a module enabling ECMA regulators to oversee simulated market activities, enforce compliance, and test policy impacts effectively.
4. Email Notification System: Implement an email-based notification mechanism to inform traders during user registration (sending OTP), account approval, and order execution.
5. Configurable Trading Hours: Provide regulators with the capability to define the platform's working hours, ensuring trading activities are restricted to specific time windows.
6. Listed Company Administration: Enable company admins to publish their stocks, upload company disclosures, and set declared dividend ratios.
7. Suspicious Activity Detection and Trader Suspension: Implement a system to detect suspicious trading activities and allow regulators to suspend traders as necessary.
8. Audit Trail Recording: Record comprehensive audit trails for critical actions, including order executions, to ensure accountability and compliance.
9. Portfolio Management: Allow traders to view and manage their personal investment portfolios, tracking their holdings and performance over time.

1.4 Scope of the project

The platform is designed to simulate essential functions of a stock market environment, focusing on features such as trading simulation, order matching, price discovery, regulatory compliance. It offers role-specific functionality for traders, listed company administrators, and regulators, ensuring a tailored experience for each type of user. While the platform enables simulated trading and learning in a secure environment, it does not handle actual financial transactions or operate as a real trading system.

1.5 Limitation of the project

- Only two order types (market and limit orders) are supported, which may limit complex trading strategies.
- External factors, such as unforeseen technical challenges or incomplete adherence to real-world policies, may affect the accuracy of the simulation.
- Suspicious Activity Detection: Relies on predefined rules for identifying suspicious behaviors, potentially missing complex fraudulent activities.
- Email Notification Dependency: Depends on external email services for sending notifications, which may experience delays or outages affecting communication reliability.
- Lack of Advanced Analytical Tools: The platform currently lacks sophisticated analytical tools and dashboards, limiting users' ability to perform in-depth market analysis and trend visualization.
- AI-Driven Surveillance Not Implemented: Advanced AI-driven surveillance for anomaly detection is not yet integrated, potentially reducing the effectiveness of fraud detection.

1.6 System Development Methodology

1.6.1 System Development Approach

The development of the Ethiopian Stock Market Simulation Platform follows an agile software development methodology to ensure iterative progress, continuous feedback, and adaptability throughout the development lifecycle. Given the absence of a stock market platform in Ethiopia, the platform draws extensively on guidelines and resources from the Ethiopian Capital Market Authority (ECMA) website, along with global best practices researched from platforms such as Investopedia and other real-world stock market simulations. The methodology is divided into the following key phases:

1.6.2 Requirement Analysis and Design

Requirement Analysis

- ➢ Refer to ECMA directives, manuals, and regulatory frameworks available on their official website to ensure platform compliance with all local regulations. [1]
- ➢ Supplement local research by examining global best practices and documentation from platforms like Investopedia to understand the dynamics of stock market operations and adapt them to Ethiopia's context. [2]

Designing

- ➢ Develop the platform's architecture to include core modules such as the trading engine, user management, portfolio management, and regulatory module.
- ➢ Design user-friendly interfaces tailored for different roles: traders, listed company admin user and administrators or regulators.
- ➢ Outline the Profit and Dividend Calculation module.

Development

Develop core modules, including:

- ➢ **User Management**:
  - ✓ Build a comprehensive user management system supporting multiple roles such as traders, listed company administrators, and regulators. Include KYC verification and role-based permissions
- ➢ **Trading Engine**:
  - ✓ Develop a high-performance trading engine capable of efficiently processing stock buy/sell orders and matching transactions with precision.
  - ✓ Ensure compliance with ECMA's regulatory framework, integrating rules for trading restrictions, transaction limits, and suspension management.

Testing and Quality Assurance

Testing:

Conduct comprehensive testing at all levels:

- **Unit Testing**: Test individual components (e.g., trading engine, portfolio updates) for correctness.
- **Integration Testing**: Validate the smooth interaction between modules such as trading, portfolio management.

Deployment and Training

- The platform is still in the development phase, and deployment has not yet been completed.

1.6.3 System Development Tools

To efficiently develop the Ethiopian Stock Market Simulation Platform, the following tools and technologies are being utilized:

- **Development Frameworks:**
  - ✓ **Backend:** Django, for handling server-side logic, APIs
  - ✓ **Frontend:** Angular, for building a responsive, interactive, and user-friendly interface.
- **Programming Languages:**
  - ✓ **Python:** For robust and scalable backend development.
  - ✓ **Typescript/JavaScript:** For efficient frontend development with Angular.
- **Database Management:**
  - ✓ **PostgreSQL:** For managing structured and reliable data storage.
- **Version Control and Collaboration:**
  - ✓ **Git:** For source code versioning and management.
  - ✓ **GitHub:** For team collaboration, issue tracking, and code review.
- **Additional Tools:**

- ✓ **Postman:** For API testing and debugging.
- ✓ **VS Code:** As the primary IDE for development.

1.7 Significance of the Project

This project is crucial for Ethiopia's capital market development as it provides a practical learning tool for future market participants. By simulating real market conditions, the platform will help users gain confidence, understand trading mechanisms, develop regulatory compliance skills, and prepare for the real Ethiopian stock exchange. It will also serve as a useful tool for ECMA to test regulatory policies in a controlled environment.

1.8 Beneficiaries of the Project

The Ethiopian Stock Market Simulation Platform is designed to address the knowledge and skill gaps among participants in Ethiopia's emerging financial market. The beneficiaries of the project include the following

- ➢ **Traders (Public Individuals)**:
  - ✓ Experience a practical, risk-free environment to learn stock trading and portfolio management. [3]
  - ✓ Gain confidence and hands-on experience to participate in the Ethiopian stock market.
  - ✓ Understand how regulatory policies and market dynamics influence trading strategies and investment decisions. [3]

- ➢ **Listed Company Representatives (Company Admins)**:
  - ✓ Manage company stocks, including publishing shares and setting declared dividend ratios.
  - ✓ Simulate interactions with the market to prepare for real-world listing scenarios. [3]
  - ✓ Gain insights into investor behavior and the impact of trading activities on company stocks.

- **Regulators (ECMA Representatives)**:
    - ✓ Monitor simulated market activities to ensure compliance with ECMA regulations. [3]
    - ✓ Test and refine regulatory policies in a controlled environment to mitigate implementation risks.
    - ✓ Simulate market scenarios to assess policy impacts and improve governance mechanisms. [3]

1.9 Feasibility Study

The feasibility study assesses the Ethiopian Stock Market Simulation Platform's technical, economic, and operational viability. It ensures the platform meets stakeholder needs and achieves its objectives effectively.

1.9.1 Technical Feasibility

This section assesses the technical viability of developing and implementing the platform.

- **Platform Design and Development**
    - **Backend**: Django is leveraged for its scalability, reliability, and ability to handle complex server-side logic and APIs.
    - **Frontend**: Angular is chosen for creating a highly responsive and interactive user experience.
    - **Database**: PostgreSQL provides robust, structured data management with support for complex queries and scalability.

- **Technical Expertise**
    - The team includes skilled developers with experience in Python, Django, Angular, PostgreSQL, and RESTful API development.
    - Tools like Git and GitHub enable efficient source code management and seamless collaboration among team members.

- **Scalability and Future Upgrades**
    - The platform's modular architecture ensures adaptability for future enhancements, including advanced analytics, sentiment analysis tools, and risk management modules.

- **Expected Benefits**
    - **Empowering Traders**:
        - ✓ Provides a risk-free environment to learn, practice, and build confidence in stock trading.
    - **Supporting Regulators**:
        - ✓ Refines policies and tests market mechanisms to ensure compliance and integrity.
    - **Enhancing Company Readiness**:
        - ✓ Equips companies to manage stocks, simulate listings, and understand market dynamics.
    - **Economic Growth**:
        - ✓ Reduces market errors and boosts liquidity through training and awareness.
    - **Technological Advancement**:
        - ✓ Promotes innovation and lays a foundation for future market tools.

1.9.2 Operational Feasibility

This aspect evaluates whether the project aligns with the needs of its stakeholders and can be effectively implemented.

- **Regulators (ECMA Representatives):**
    - ✓ The platform serves as a comprehensive testing ground for regulatory policies, enabling ECMA to monitor compliance, assess policy impacts, and refine governance mechanisms.

- **Traders (General Public):**
    - ✓ Empowers individuals to enhance their trading knowledge and confidence by simulating real-world stock market activities in a risk-free environment.
- **Listed Company Representatives (Company Admins):**
    - ✓ Allows company admins to simulate the listing process, manage stocks, and setting up the declared dividend ratio on the system.

Ease of Use:

- ✓ The platform is designed with a highly intuitive and user-friendly interface, ensuring all participants can easily access and navigate its features.
- ✓ Training programs and resources are provided to enable stakeholders to maximize the platform's capabilities and achieve their objectives.

1.9.3 Economic Feasibility

This aspect evaluates the financial viability of the project, including cost analysis and expected benefits.

- **Cost Analysis**
    - ❖ **Infrastructure Costs:** Cloud hosting services and database management tools.
    - ❖ **Maintenance Costs:** Regular updates and system monitoring.
    - ❖ **Training Costs:** User training for ECMA staff, brokers, and investors.
- **Expected Benefits**
    - ❖ Reducing the cost of errors in the real market by providing a risk-free training platform.
    - ❖ Increasing investor participation and market liquidity through better awareness.

- ❖ Supporting ECMA in refining policies, minimizing regulatory implementation risks.

1.10 Project schedule

The Ethiopian Stock Market Simulation Platform will be completed in approximately 4 months, divided into four phases with specific milestones and activities. This schedule ensures timely delivery while allowing for iterative development and stakeholder engagement.

Phase	Days	Key Activities	Critical Modules/Tasks
Planning Phase	Days 1-5	- Create product backlog - Define project scope and success criteria	N/A

Analysis Phase	Days 6-15	- Analyze requirements and prioritize user stories - Refine acceptance criteria - Estimate effort for tasks	Define stock market simulation requirements and dependencies
Design Phase	Days 16-30	- Design application architecture and workflows - Develop database schema - Define branding and UX/UI - Create technical documentation	Database Design for Transactions and User, User Portfolios, Order , Trade, Notification, Dividend and Listed Company Table
Development Phase	Days 31-90	- Backend API development - Stock market simulation engine - Web UI development - Integration of third-party services	**Stock Market Simulation Engine (Days 31-60)** **Backend APIs (Days 60-75)** **and UI Integration (Days 75-90)**
Testing Phase	Days 91-100	- Test backend, APIs, and web platform - Conduct user acceptance testing - Analyze and resolve bugs	**Testing Stock Simulation and Transaction Modules**
Implementation Phase	Days 101-108	- Deploy backend and web platform - Configure domain - Provide training and support - Collect final feedback	**Deployment and Domain Setup**
Maintenance Phase	Ongoing	- Fix bugs - Add new features - Optimize performance - Conduct backups and monitor user feedback	Continuous optimization and future feature updates

Table 1 Project Schedule

1.11 Project Budget

Comprehensive Budget Table

Category	**Item**	**Frequency**	**Cost (ETB)**	**Description**
One-Time Costs	Printing and Laminating	One-time	1,000	For initial presentations and materials.
	.gov.et domain Registration for year from ethiotelecom	One-time	550	Purchase and configure a custom domain.
	Initial Marketing	One-time	3,000	Digital ads and promotional campaigns setup.
	Content Creation	One-time	8,000	Hire content creator for web content and documentation.
Recurring Costs (4 Mo)	Broadband Internet (6 MB)	4 months reserve	4,375	Internet subscription for platform operation.
	VPS Hosting from Hostwinds	One Year reserve	($49.99) 6200	Web hosting for backend operations (8 GB RAM, 4 CPUs).
	Miscellaneous Maintenance	4 months reserve	6000	Reserve for unexpected maintenance during initial period.
Contingency Reserve	Platform Maintenance	One-time	10,000	Reserve fund for unforeseen issues and updates.

Table 2 Project budget

Chapter 2: Requirement Analysis

2.1 Current System Description

Ethiopia currently lacks both a formal stock market and a simulation platform to facilitate understanding and practice of stock market operations. As such, no existing system in Ethiopia performs the functionalities that this project aims to address.

2.1.1 Major function of the current system

The absence of an operational stock market system means that:

- ➢ No mechanisms exist for simulating trading activities or analyzing market dynamics.
- ➢ Regulatory bodies lack tools to test compliance frameworks and policies.
- ➢ Educational resources and platforms for stakeholders to understand stock markets are non-existent.

2.1.2 Problem of Existing System

The lack of a stock market and simulation platform results in several challenges:

- ➢ **Knowledge Gap**: Stakeholders, including potential traders, companies and regulators, lack practical exposure to stock market operations, trading strategies, and compliance requirements.
- ➢ **No Practical Training Tools**: There is no simulated environment where users can practice trading, market analysis, or regulatory testing in a risk-free setting.
- ➢ **Regulatory Challenges**: The Ethiopian Capital Markets Authority (ECMA) has no platform to test regulatory policies or simulate the impact of those policies.
- ➢ **Limited Public Awareness**: The public has minimal access to tools or educational resources to understand stock markets, limiting their readiness for a functional exchange in the future

2.2 Requirement Gathering

2.2.1 Requirement Gathering Methods

To build a simulation platform that addresses the challenges, the following methods were employed:

- **Observation**: Studying simulation platforms in established markets to identify industry best practices, design considerations, and feature sets.
- **Document Review**: Reviewing ECMA directives, legal documents, and regulatory frameworks to ensure alignment with Ethiopia's evolving capital market regulations.

2.2.2 Business Rules

Regulatory Compliance:

The platform must strictly adhere to ECMA’s regulatory frameworks, ensuring compliance in all simulated activities.

Role-Based Access:

Only registered users, including **Traders**, **Listed Company Representatives**, and **Regulators**, are allowed access to functionalities relevant to their roles.

Trading Engine Simulation:

The trading engine must accurately replicate real-world stock trading mechanisms, supporting multiple order types (e.g., market and limit orders) and employing a Price-Time Priority Algorithm for execution.

Work Hour Management:

Regulators have the authority to define the platform's active trading hours, ensuring trading activities occur only during designated periods.

Trader Suspension:

Regulators can suspend traders for policy violations, ensuring a fair and compliant trading environment.

Listed Company Operations:

Listed company representatives can manage their stocks, including publishing shares and setting declared dividend ratios, in compliance with ECMA regulations.

2.3 Proposed System Description

2.3.1 Overview

The proposed system is an Ethiopian Stock Market Simulation Platform, designed to mimic real-world trading activities, provide market analysis tools, and incorporate regulatory oversight features. This platform will serve as a comprehensive learning tool for stakeholders, including

- **Traders**:
  - Designed for individual participants to learn effective trading strategies, enhance their portfolio management skills, and understand the intricacies of market operations.
  - Traders can place buy/sell orders, track their portfolio, and receive notifications about their trades, simulating a complete trading experience.
- **Listed Company Administrators**:
  - Enables representatives of listed companies to manage their company profiles, publish stocks, and declare dividends.
  - They can also monitor their company's trading activity, ensuring a realistic simulation of administrative responsibilities in a stock market ecosystem.
- **Regulators**:
  - Provides tools for simulating regulatory oversight, approving or rejecting trader and company registrations, monitoring trading activities, and managing compliance violations.

- Regulators can generate comprehensive reports, set market alerts, and test the effectiveness of regulatory policies in a controlled environment.

By providing a secure, virtual environment, the Ethiopian Stock Market Simulation Platform empowers stakeholders to gain practical experience, deepen their understanding of stock market dynamics, and contribute to the successful establishment and sustainability of Ethiopia's upcoming stock exchange.

2.3.2 Functional Requirements

- User Registration and Role Management
  - ✓ **Multiple Roles:** Supports Traders, Company Admins, and Regulators.
  - ✓ **Role-Based Access Control:** Ensures secure and tailored access to platform features based on user roles.
  - ✓ **Email OTP Verification:** Sends OTPs via email during user registration to enhance account security.

- Trading Engine
  - ✓ **Order Placement:** Supports market and limit order types.
  - ✓ **Order Matching:** Utilizes the Price-Time Priority Algorithm for fair and efficient trade execution.
  - ✓ **Market Depth Simulation:** Provides real-time order books, bid-ask spreads, and partial matching capabilities.
  - ✓ **Transaction Fee Calculation:** Automatically applies predefined transaction fees to each trade.

- Portfolio Management
  - ✓ **Virtual Portfolios:** Allows traders to manage, monitor, and track their investment portfolios.

- Regulatory Tools
  - ✓ **Compliance Monitoring:** Enables regulators to oversee market activities and ensure adherence to regulations.

- ✓ **Suspicious Activity Detection:** Identifies and flags unusual trading behaviors for further investigation.
- ✓ **Trader Suspension:** Allows regulators to suspend traders based on detected suspicious activities.
- ✓ **Audit Trail Recording:** Logs critical actions, including order executions, to maintain accountability and support compliance efforts.
- ✓ **Report Generation:** Facilitates the creation of detailed compliance and regulatory reports.

➢ Company Administration

- ✓ **Stock Publication:** Enables company admins to publish their stocks on the platform.
- ✓ **Financial Disclosures:** Allows company admins to upload and manage their company's financial disclosures, ensuring transparency.

➢ Email Notification System

- ✓ **OTP Delivery:** Sends OTPs via email during user registration for account verification.
- ✓ **Account Approval Alerts:** Notifies users via email when their accounts are approved.
- ✓ **Order Execution Notifications:** Alerts traders via email when their orders are matched and executed.

2.3.3 Nonfunctional Requirements

2.3.3.1 Performance

The platform must provide a responsive and efficient user experience, handling trading simulations, regulatory operations, and administrative tasks with minimal latency and seamless interactions.

2.3.3.2 Scalability

The system should support the addition of new features and the ability to scale to accommodate an increasing number of users and activities as the platform grows alongside Ethiopia's market readiness.

2.3.3.3 Availability

The platform must remain highly accessible and reliable, ensuring continuous operation during the designated transaction periods set by the system's regulators. Outside these periods, the platform should remain available for non-transactional activities such as portfolio management, monitoring, and administrative tasks, with minimal downtime for maintenance.

2.3.3.4 Reliability

The system must ensure the accuracy and consistency of all simulations, including trading, order matching, and compliance monitoring, to provide a realistic and dependable market environment.

2.3.3.5 Maintainability

The platform’s codebase must be modular and well-documented to allow for straightforward updates, troubleshooting, and the integration of new functionalities as needed.

2.3.3.6 Security

Robust security measures must be implemented, including secure authentication, role-based access control, and data encryption, to safeguard user data and trading activities.

2.3.3.7 Usability

The user interface must be designed for ease of use, providing clear navigation and accessible features tailored to the needs of traders, listed company administrators, and regulators.

Chapter 3: System Model

3.1 Scenarios

3.1.1 Use Case Model

The Use Case Model provides a structured representation of the interactions between users (actors) and the system as well as the role of the trading engine as a sub-system. Highlighting the key functionalities offered by the Ethiopian Stock Market Simulation Platform. It serves as the foundation for understanding system requirements and user roles.

3.1.1.1 Actor Identification

The platform supports the following primary actors:

1. **Trader**:

Role: The trader represents an individual participant in the simulation who interacts with the system to perform trading activities.

Responsibilities:

- Registers with the platform to gain access.
- Places buy or sell orders using the trading interface.
- Monitors order statuses, manages portfolios.
- Receives notifications about executed trades, and system updates.

2. **Listed Company Admin:**

Role: Acts as a representative of a listed company managing stock-related activities.

Responsibilities:

- Registers the company and manages its profile.
- Publishes company stocks and declares dividends for shareholders.
- Monitors trading activities involving their listed stocks.
- Generates reports and manages stock visibility on the platform.

3. **Regulator**:

Role: Represents the governing body overseeing the simulation to ensure compliance and regulatory adherence.

Responsibilities:

- Approves or rejects user registrations for traders and listed company admins.
- Monitors market activities for compliance and can suspend traders if necessary.
- Sets system working hours to regulate trading periods.
- Generates compliance and market reports to ensure transparency and fairness.

4. Trading Engine (Sub-System):

Role: The trading engine operates as a core sub-system, automating critical trading functionalities.

Responsibilities:

- Matches buy and sell orders using a price-time priority algorithm.
- Executes trades in real-time, updating user portfolios and order statuses.
- Sends notifications to users upon successful trade execution.
- Logs all transactions for regulatory and auditing purposes.

3.1.1.2 Use case identification

Use cases describe the various actions or services the system provides to its actors. Each actor interacts with specific use cases to perform tasks. Below are the identified use cases grouped by actor:

1. Trader Use Cases:

- **Login** – Access the system.
- **Logout** – Exit the system (extends from login).
- **Place Order** – Execute a trade.
- **Track Order Status** – Monitor ongoing orders.
- **Manage Portfolio** – Handle investment portfolios.
- **Receive Different Notifications** – Get alerts and messages.
- **Generate Report** – Obtain performance or trade summaries.

2. Listed Company Admin Use Cases:

- **Login** – Access admin functionalities.
- **Logout** – Exit the system (extends from login).

- **Manage Company Profile** – Update company details.
- **Publish and Manage Company Stock** – Handle company shares and market availability.
- **Set Declared Company's Dividend** – Manage dividend declarations.
- **Monitor Company Trading Activity** – Oversee company-related trades.
- **Generate Report** – Create various reports.

3. Regulators Use Cases:

- **Login** – Access regulatory functions.
- **Approve/Reject User Registration** – Manage new user accounts.
- **Set System Working Hour** – Control operational hours.
- **Monitor Market Activity and Suspend Traders** – Oversee trading and enforce suspensions.
- **Generate Reports** – Create regulatory reports.

4. Sub-System (Trading Engine) Use Cases:

- **Order Matching and Processing** – Match buy/sell orders.
- **Trade Execution and Processing** – Execute transactions.
- **Send Order Execution Notifications** – Notify users about executed orders.
- **Transaction Logging** – Log trading activity for records.

3.1.2 Use Case Diagram

The use case diagram visually represents the interactions between actors (Trader, Listed Company Admin, Regulator) and their respective functionalities. It also highlights the role of the trading engine as a sub-system for automating trading-related processes. Refer to the provided diagram for detailed visualization.

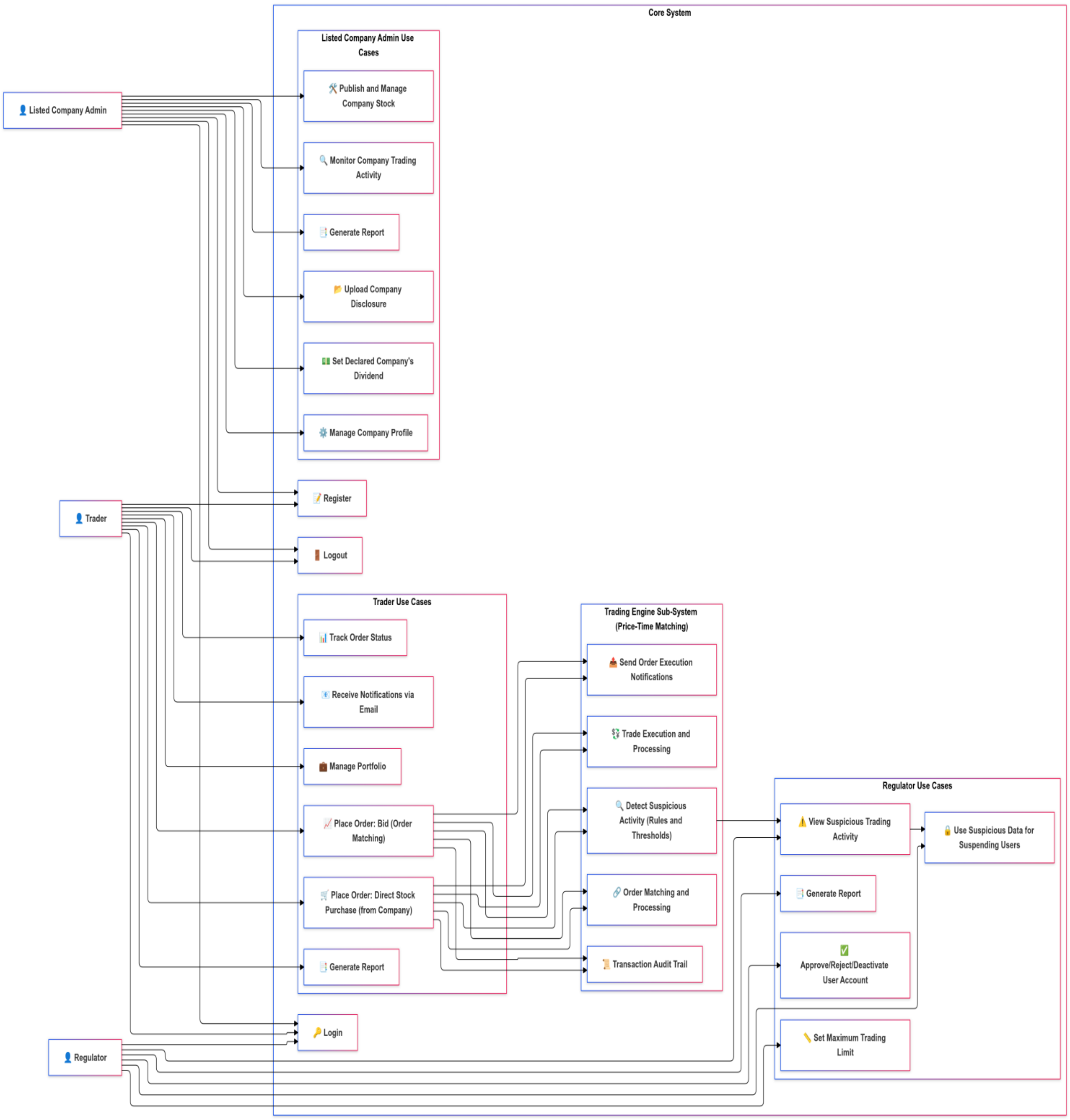

Core System
Listed Company Admin Use Cases
Listed Company Admin
Publish and Manage Company Stock
Monitor Company Trading Activity
Generate Report
Upload Company Disclosure
Set Declared Company's Dividend
Manage Company Profile
Trader
Register
Logout
Trader Use Cases
Track Order Status
Receive Notifications via Email
Manage Portfolio
Place Order: Bid (Order Matching)
Place Order: Direct Stock Purchase (from Company)
Generate Report
Trading Engine Sub-System (Price-Time Matching)
Send Order Execution Notifications
Trade Execution and Processing
Detect Suspicious Activity (Rules and Thresholds)
Order Matching and Processing
Transaction Audit Trail
Regulator Use Cases
View Suspicious Trading Activity
Use Suspicious Data for Suspending Users
Generate Report
Approve/Reject/Deactivate User Account
Set Maximum Trading Limit
Login
Regulator

Figure 1 Use case diagram

3.1.1. Description of use case diagram

Use Case: Trader Operations

Actors	**Trader**
Description	This use case allows traders to register, place orders, manage portfolios, track orders, and receiving system notifications.
Preconditions	The trader must have a KYC approved and active account.
Post conditions	User can Login, place orders, manage portfolios, receive system notifications and can receive dividend for their owned stocks
Events	➢ **Register User**: Traders register by providing required details such as username, email, and password and provided required document (KYC documents). ➢ **Login**: Traders log in to access their trading dashboard and to perform trading activity. ➢ **Place Order:** Traders place buy/sell orders in the system using different order types. ➢ **Track Order Status**: Traders monitor the status of their orders in real time. ➢ **Manage Portfolio**: Traders can view, analyze, and manage their stock holdings. ➢ **Receive Notifications**: Traders receive alerts on order execution, portfolio updates.
Alternative Events	➢ **Invalid Login Details**: Displays an error if incorrect credentials are provided, prompting the trader to re-enter their details. ➢ **KYC is not verified**: Displays and error if user is not verified by the regulators. ➢ **Order Cancellation**: Alerts the trader if an order cannot be matched or executed due to lack of order matching at the end of the day.
Exceptions	**System Downtime**: The system notifies the trader of temporary unavailability during order placement, ensuring the issue is resolved promptly.

Table 3 trader operation use case

Use Case: Listed Company Administrator Operations

Actors	Listed Company Admin
Description	This use case supports listed companies in publishing stocks, declaring dividends, monitoring activities, and generating reports.
Preconditions	Listed company administrators must have a verified and approved account.
Post conditions	Stocks are listed, dividends declared, or reports generated successfully.
Events	➢ **Register User**: Admins register by providing company details for system inclusion. ➢ **Login**: Admins log in to manage their company profile and stock-related activities. ➢ **Manage Company Profile**: Admins update company details such as stock offerings and contact information. ➢ **Publish Stock**: Admins list and update their company's stock for trading. ➢ **Declare Dividends**: Admins announce dividends for shareholders. ➢ **Monitor Trading Activity**: Admins monitor activities associated with their listed stocks. ➢ **Generate Report**: Admins generate stock performance and trading activity reports.
Alternative Events	➢ **Invalid Login Details**: Displays an error if incorrect credentials are provided, prompting the trader to re-enter their details. ➢ **KYC is not verified**: Displays and error if user is not verified by the regulators. ➢ **Invalid Stock Data**: Displays an error if incomplete or invalid stock data is entered during publishing.

Exceptions	System Error: Temporarily halts stock publishing or dividend declarations during a system malfunction.

Table 4 Listed Company Administrator Operations use case

Use Case: Regulator Operations

Actors	Regulator
Description	This use case allows regulators to monitor market activities, ensure compliance, and approve/reject user registrations.
Preconditions	Regulators must have authorized accounts with sufficient permissions.
Post conditions	Regulator can Set System working hour for each days, suspend traders from buying/selling specific stock or global in the platform and oversee the overall trading activity.
Events	➢ **Login**: Regulators log in to monitor and manage compliance activities. ➢ **Approve/Reject Users**: Regulators validate and approve or reject trader and listed company registrations. ➢ **Set System Working time**: Regulators will set system working time. ➢ **Suspend traders:** Regulators can suspend traders from trading specific stock or from the platform. ➢ **Generate Reports**: Regulators create detailed compliance and activity reports.
Alternative Events	➢ **Invalid Login Details**: Displays an error if incorrect credentials are provided, prompting the trader to re-enter their details.
Exceptions	Data Retrieval Issue: Alerts regulators if there is an issue accessing compliance or trading data and escalates the issue for resolution.

Table 5 Regulator Operations

Use Case: Trading Engine Operations

Use Case	Trading Engine Operations
Actors	Trading Engine (Sub-System)
Description	The trading engine automates core trading functionalities, ensuring efficient order matching, execution, and transaction logging.
Preconditions	Valid buy and sell orders must be placed by traders within the system's trading hours.
Post conditions	Orders are matched and executed, user portfolios are updated, and transactions are logged and send the notification for the traders.
Events	**Order Matching and Processing**: Matches buy and sell orders using a price-time priority algorithm. **Trade Execution and Processing**: Executes matched orders and updates the order book and user portfolios. **Send Order Execution Notifications**: Notifies users about the successful execution of their trades. **Transaction Logging**: Records all completed transactions for compliance and auditing purposes.
Alternative Events	**Order Mismatch**: If no matching order is found for a placed order, it remains in the pending queue until a match is available and will be cancelled at the end of day.
Exceptions	System Downtime: If the trading engine encounters a failure, ongoing operations are paused, and pending transactions are queued for later processing.

Table 6 Trading Engine Operations

3.1.3 Activity Diagram

User Registration Activity

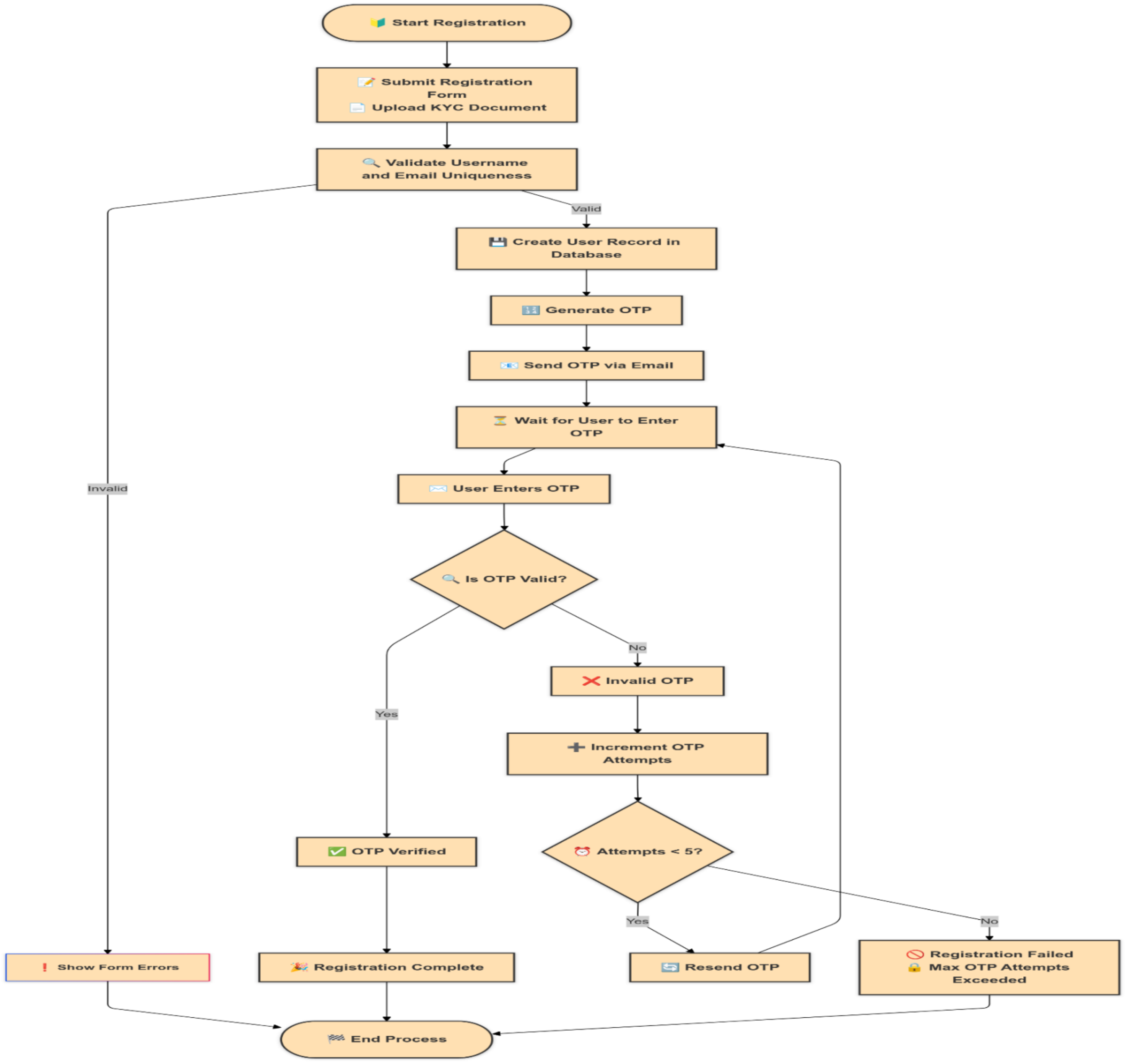


Figure 2 User Registration Activity

User Login Activity Diagram

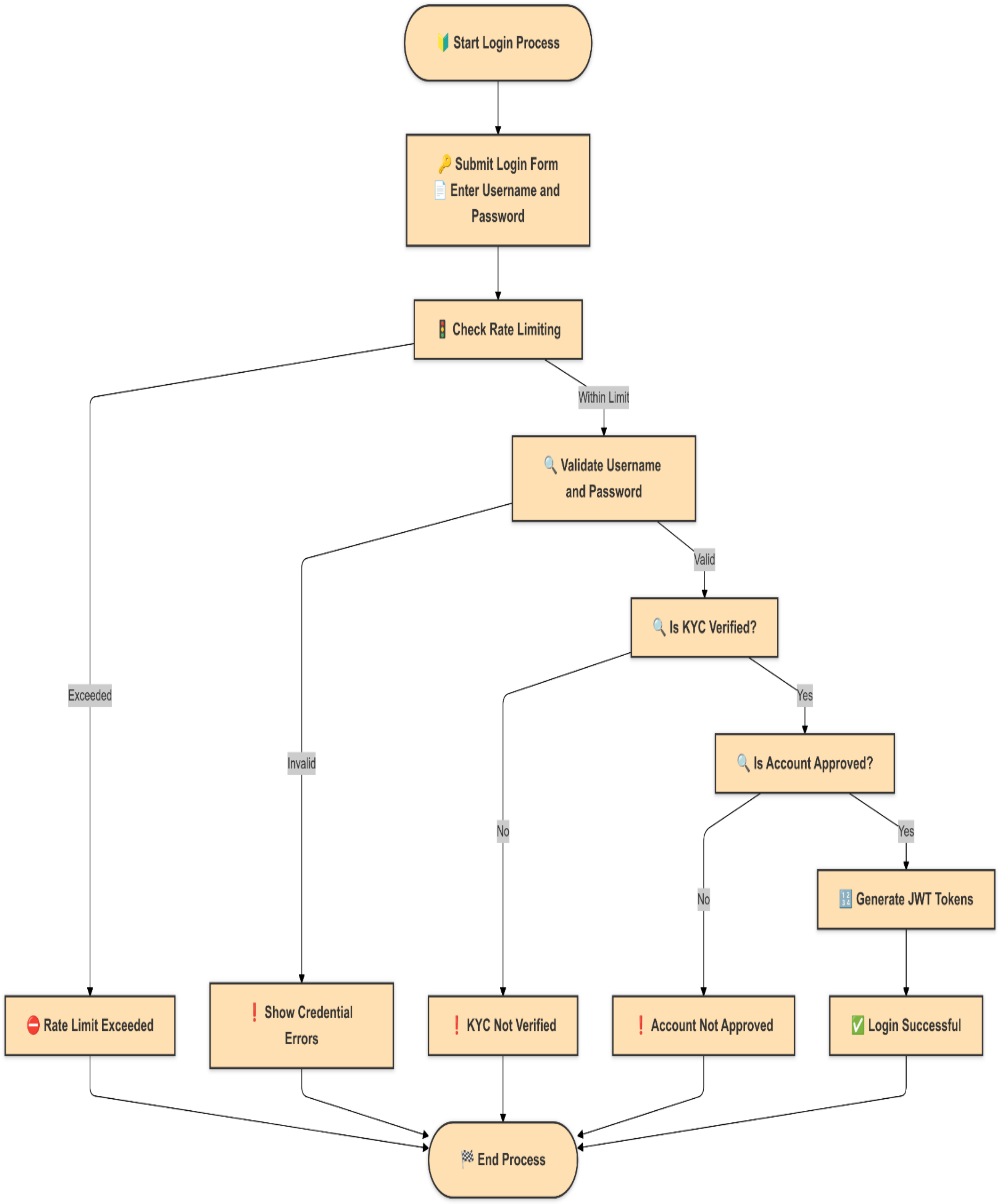


Figure 3 User Login Activity Diagram

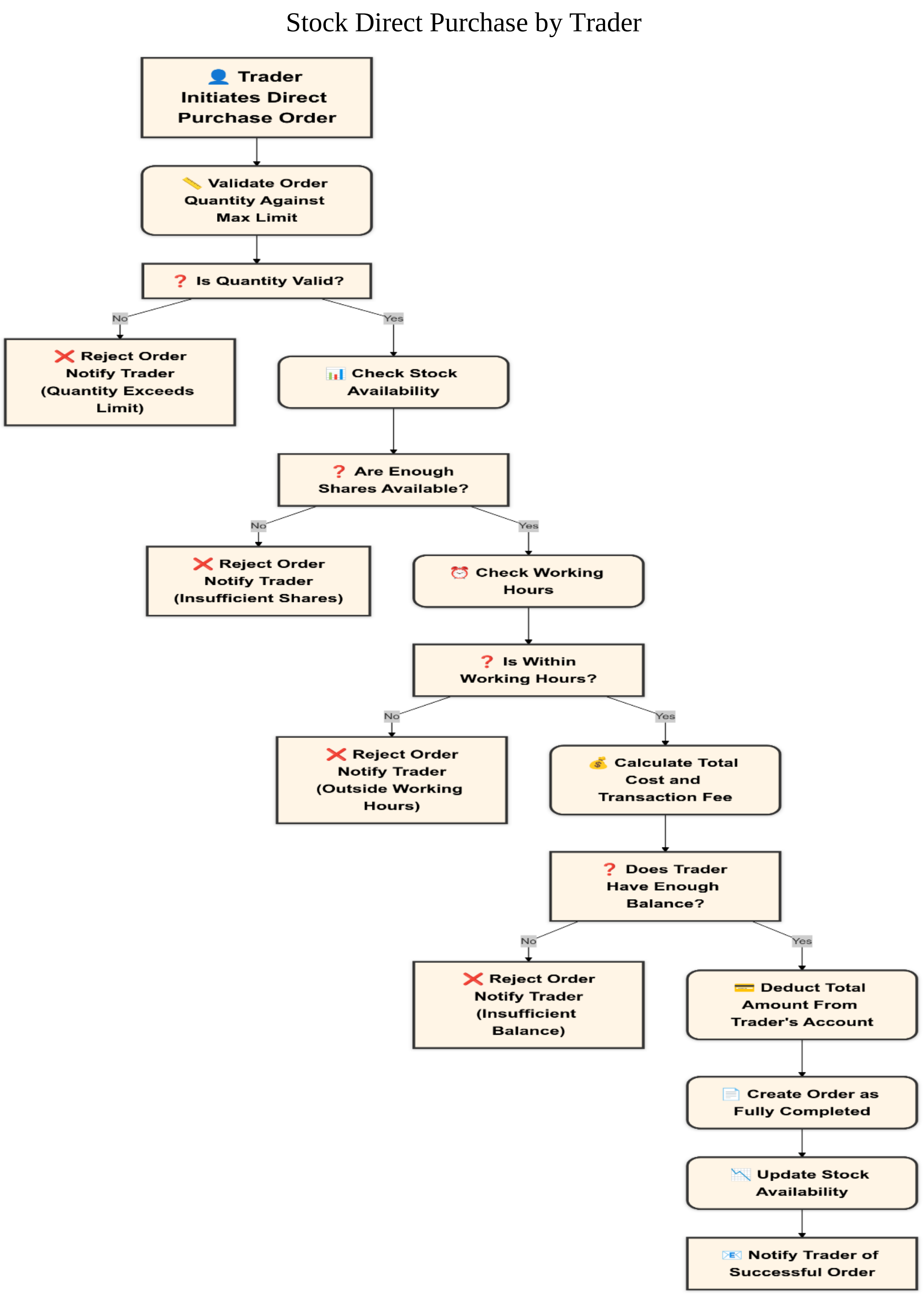

Stock Direct Purchase by Trader
Trader Initiates Direct Purchase Order
Validate Order Quantity Against Max Limit
Is Quantity Valid?
No
Yes
Reject Order Notify Trader (Quantity Exceeds Limit)
Check Stock Availability
Are Enough Shares Available?
No
Yes
Reject Order Notify Trader (Insufficient Shares)
Check Working Hours
Is Within Working Hours?
No
Yes
Reject Order Notify Trader (Outside Working Hours)
Calculate Total Cost and Transaction Fee
Does Trader Have Enough Balance?
No
Yes
Reject Order Notify Trader (Insufficient Balance)
Deduct Total Amount From Trader's Account
Create Order as Fully Completed
Update Stock Availability
Notify Trader of Successful Order

Figure 4: Stock Direct Purchase by Trader

3.1.4 Class Model

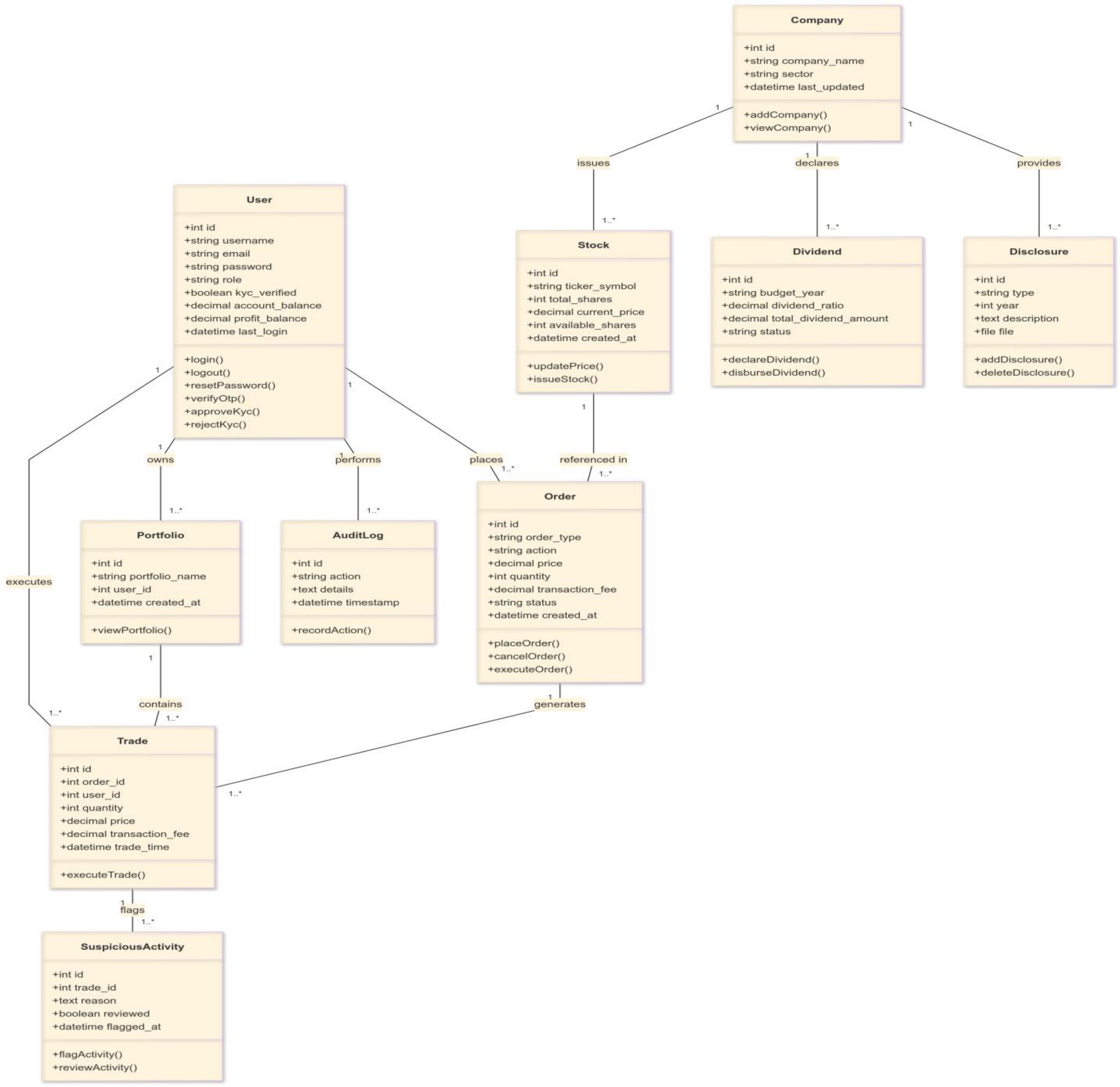


Figure 5 Class Model

3.1.5 Data Dictionary

Company

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Company	**PK** (Primary Key)
company_name	Company Name	VARCHAR	150 characters	Name/title of the Company	NOT NULL
sector	Sector	VARCHAR	150 characters	The industry or sector the Company operates in	—
last_updated	Last Updated	DATETIME	8 bytes	Timestamp of the last update for the record	NOT NULL

Table 7 Company

Dividend

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Dividend record	**PK** (Primary Key)
budget_year	Budget Year	VARCHAR	50 characters	Budget or fiscal year (e.g., "2025")	NOT NULL
dividend_ratio	Dividend Ratio	DECIMAL	—	Ratio (e.g., 0.05 for 5%)	Default = 0.0
total_dividend_amount	Total Dividend	DECIMAL	—	Total amount allocated as dividends	Default = 0.0
status	Status	VARCHAR	100 characters	Current status of the dividend ("Approved,","Paid," etc.)	—

Table 8 Dividend

Disclosure

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Disclosure	**PK** (Primary Key)
type	Type	VARCHAR	100 characters	Disclosure type ("Financial," "Annual," "Quarterly")	NOT NULL
year	Year	INTEGER	4 bytes	Relevant year of the disclosure	NOT NULL
description	Description	TEXT	—	Detailed description of the disclosure	—
file	File	TEXT (or similar)	—	URL/path or blob reference to the uploaded document	—

Table 9 Disclosure

Stock

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Stock	**PK** (Primary Key)
ticker_symbol	Ticker Symbol	VARCHAR	20 characters	Symbol used to identify the stock (e.g., "AAPL")	NOT NULL, UNIQUE
total_shares	Total Shares	INTEGER	4 bytes	Total number of shares authorized/issued	Default = 0
current_price	Current Price	DECIMAL	—	Current trading price per share	—
available_shares	Available Shares	INTEGER	4 bytes	Number of shares still available (not sold)	Default = 0
created_at	Created At	DATETIME	8 bytes	Timestamp of when the stock was added	NOT NULL

Table 10 Stock

Orders

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Order	**PK** (Primary Key)
order_type	Order Type	VARCHAR	50 characters	Type of order ("Limit," "Market," etc.)	NOT NULL
action	Action	VARCHAR	50 characters	"Buy" or "Sell"	NOT NULL
price	Price	DECIMAL	—	Limit price, if applicable	Default = 0.0
quantity	Quantity	INTEGER	4 bytes	Number of shares to buy or sell	Default = 1
transaction_fee	Transaction Fee	DECIMAL	—	Brokerage or platform fee associated with the order	Default = 0.0
status	Status	VARCHAR	50 characters	Current status ("Open," "Filled," "Canceled")	—
created_at	Created At	DATETIME	8 bytes	Timestamp of when the order was placed	NOT NULL
stock_id	Stock ID	INTEGER	4 bytes	FK referencing **Stock**	Foreign Key (FK)
user_id	User ID	INTEGER	4 bytes	FK referencing **User**	FK

Table 11 Orders

Trade

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each Trade	**PK** (Primary Key)
quantity	Quantity	INTEGER	4 bytes	Number of shares traded	NOT NULL
price	Price	DECIMAL	—	Price per share at time of trade	NOT NULL
transaction_fee	Transaction Fee	DECIMAL	—	Fee charged for executing the trade	Default = 0.0
trade_time	Trade Time	DATETIME	8 bytes	Timestamp of when the trade was executed	NOT NULL
stock_id	Stock ID	INTEGER	4 bytes	FK referencing **Stock**	FK
user_id	User ID	INTEGER	4 bytes	FK referencing **User**	FK

Table 12 Trade

Suspicious Activity

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each suspicious activity record	**PK** (Primary Key)
reason	Reason	TEXT	—	Explanation of why the trade/activity is suspicious	NOT NULL
reviewed	Reviewed	BOOLEAN	1 byte	Indicates if the suspicious activity was reviewed	Default = FALSE
flagged_at	Flagged At	DATETIME	8 bytes	Timestamp of when it was flagged	NOT NULL
trade_id	Trade ID	INTEGER	4 bytes	FK referencing **Trade**	FK

Table 13 Suspicious Activity

User Table

Field	**Name**	**Data Type**	**Size**	**Description**	**Constraints**
id	ID	INTEGER	4 bytes	Unique identifier for the User	**PK** (Primary Key)
username	Username	VARCHAR	150 characters	Chosen login name	NOT NULL, UNIQUE
password	Password	VARCHAR	255 characters	Hashed user password	NOT NULL
email	Email	VARCHAR	255 characters	User's email address	NOT NULL, UNIQUE
role	Role	VARCHAR	15 characters	E.g., "Trader," "Regulator," "Company Admin"	Default = "Trader"
is_approved	Is Approved	BOOLEAN	1 byte	Indicates if the user is approved	Default = FALSE
kyc_document	KYC Document	FILE	—	File uploaded for KYC verification	NULLABLE
kyc_verified	KYC Verified	BOOLEAN	1 byte	Indicates if the user passed KYC/AML checks	Default = FALSE

company_id	Company ID	INTEGER	4 bytes	ID of the associated company (if applicable)	NULLABLE
account_balance	Account Balance	DECIMAL	max_digits=15, decimal_places=2	Main balance in user's account	Default = 0.00
profit_balance	Profit Balance	DECIMAL	max_digits=15, decimal_places=2	Profit or realized gains stored separately	Default = 0.00
date_registered	Date Registered	DATETIME	8 bytes	Timestamp when the user registered	Default = CURRENT_TIMESTAMP
last_login	Last Login	DATETIME	8 bytes	Timestamp of the last login activity	NULL if never logged in
otp_code	OTP Code	VARCHAR	6 characters	One-Time Password sent to the user	NULLABLE
otp_sent_at	OTP Sent At	DATETIME	8 bytes	Timestamp of when the OTP was sent	NULLABLE
otp_verified	OTP Verified	BOOLEAN	1 byte	Indicates if the OTP was successfully verified	Default = FALSE
otp_attempts	OTP Attempts	INTEGER	4 bytes	Number of OTP retry attempts	Default = 0

Table 14 User Table

Portfolio

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for the Portfolio	**PK** (Primary Key)
quantity	Quantity	INTEGER	4 bytes	Number of shares within the portfolio (aggregate)	Default = 0
average_purchase_price	Average Purchase Price	DECIMAL	—	Weighted average price for the shares in the portfolio	Default = 0.0
total_investment	Total Investment	DECIMAL	—	Cumulative amount invested	Default = 0.0
user_id	User ID	INTEGER	4 bytes	FK referencing **User** (1-to-1 relationship)	FK, UNIQUE

Table 15 Notification

Audit Log

Field	Name	Data Type	Size	Description	Constraints
id	ID	INTEGER	4 bytes	Unique identifier for each audit record	**PK** (Primary Key)
action	Action	VARCHAR	100 characters	Description of the action performed	NOT NULL
details	Details	TEXT	—	Additional details or context for the action	—

timestamp	Timestamp	DATETIME	8 bytes	When the action occurred	NOT NULL
user_id	User ID	INTEGER	4 bytes	FK referencing **User** who performed the action	FK

Table 16 Audit Log

3.1.6 Sequence Diagram

User Registration sequence diagram

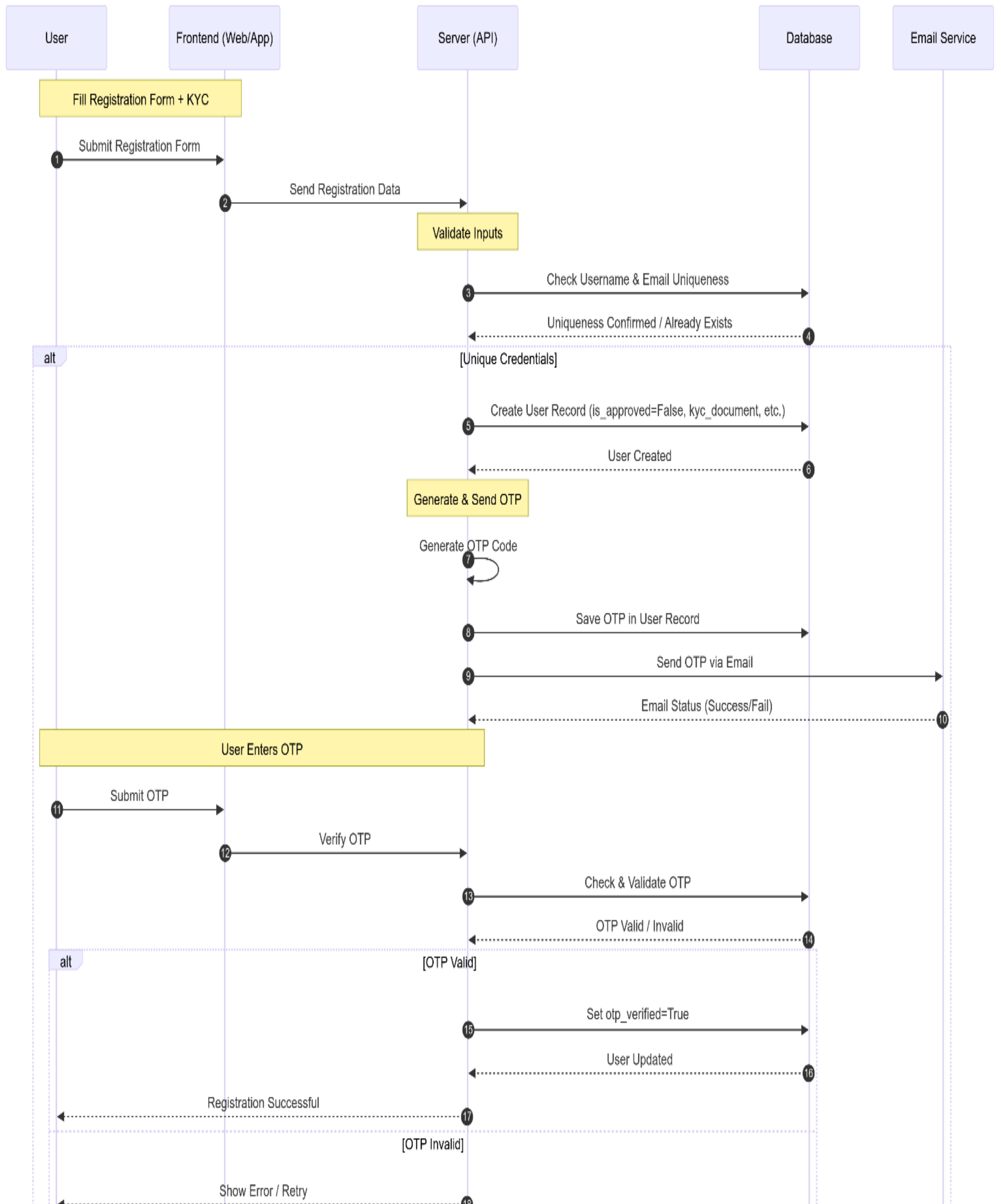
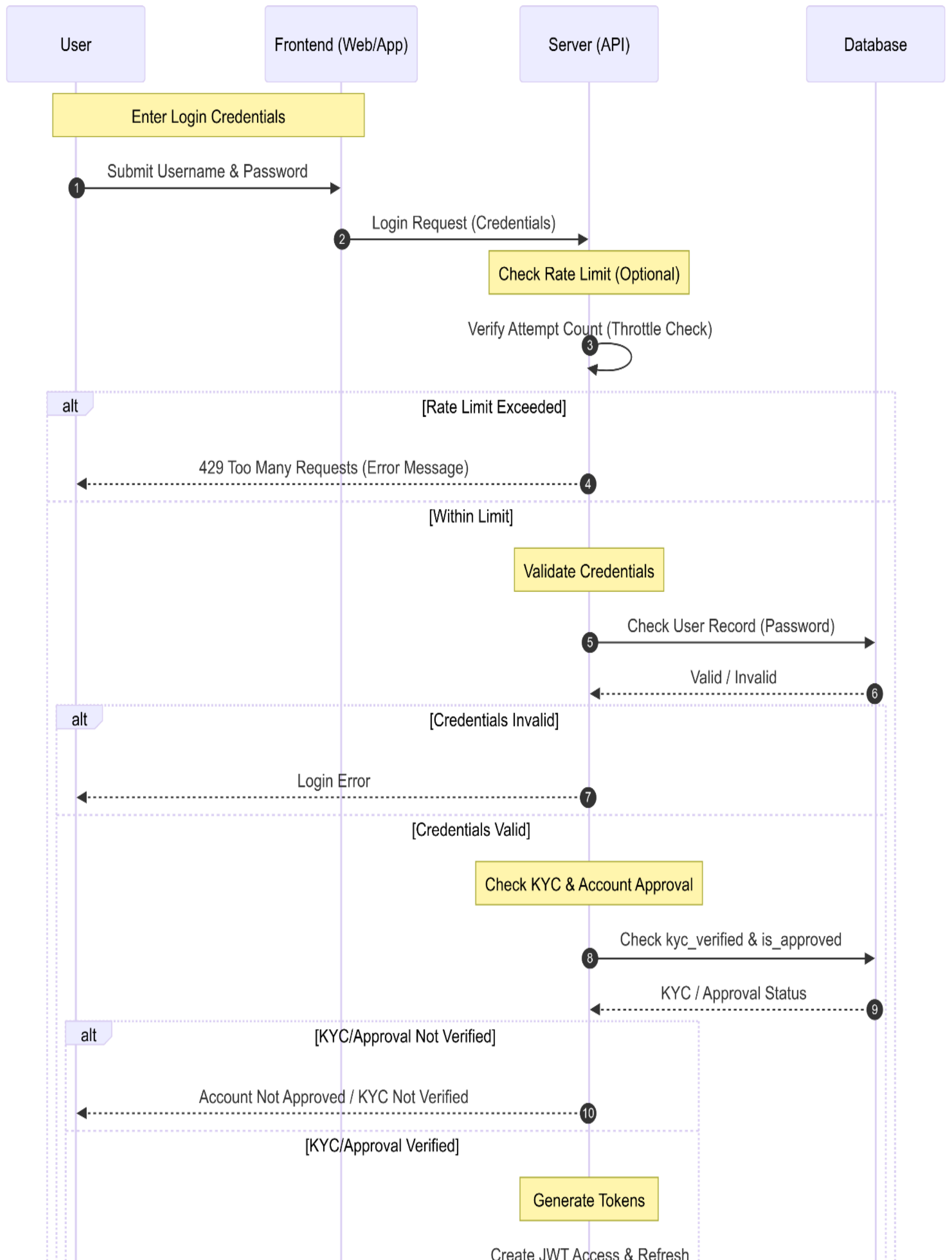


Figure 6 User Registration Sequence Diagram

User Login Sequence



User
Frontend (Web/App)
Server (API)
Database
Enter Login Credentials
1 Submit Username & Password
2 Login Request (Credentials)
Check Rate Limit (Optional)
3 Verify Attempt Count (Throttle Check)
alt
[Rate Limit Exceeded]
4 429 Too Many Requests (Error Message)
[Within Limit]
Validate Credentials
5 Check User Record (Password)
6 Valid / Invalid
alt
[Credentials Invalid]
7 Login Error
[Credentials Valid]
Check KYC & Account Approval
8 Check kyc_verified & is_approved
9 KYC / Approval Status
alt
[KYC/Approval Not Verified]
10 Account Not Approved / KYC Not Verified
[KYC/Approval Verified]
Generate Tokens
Create JWT Access & Refresh

Figure 7 User Login Sequence Diagram

Direct stock Purchase Sequence Diagram

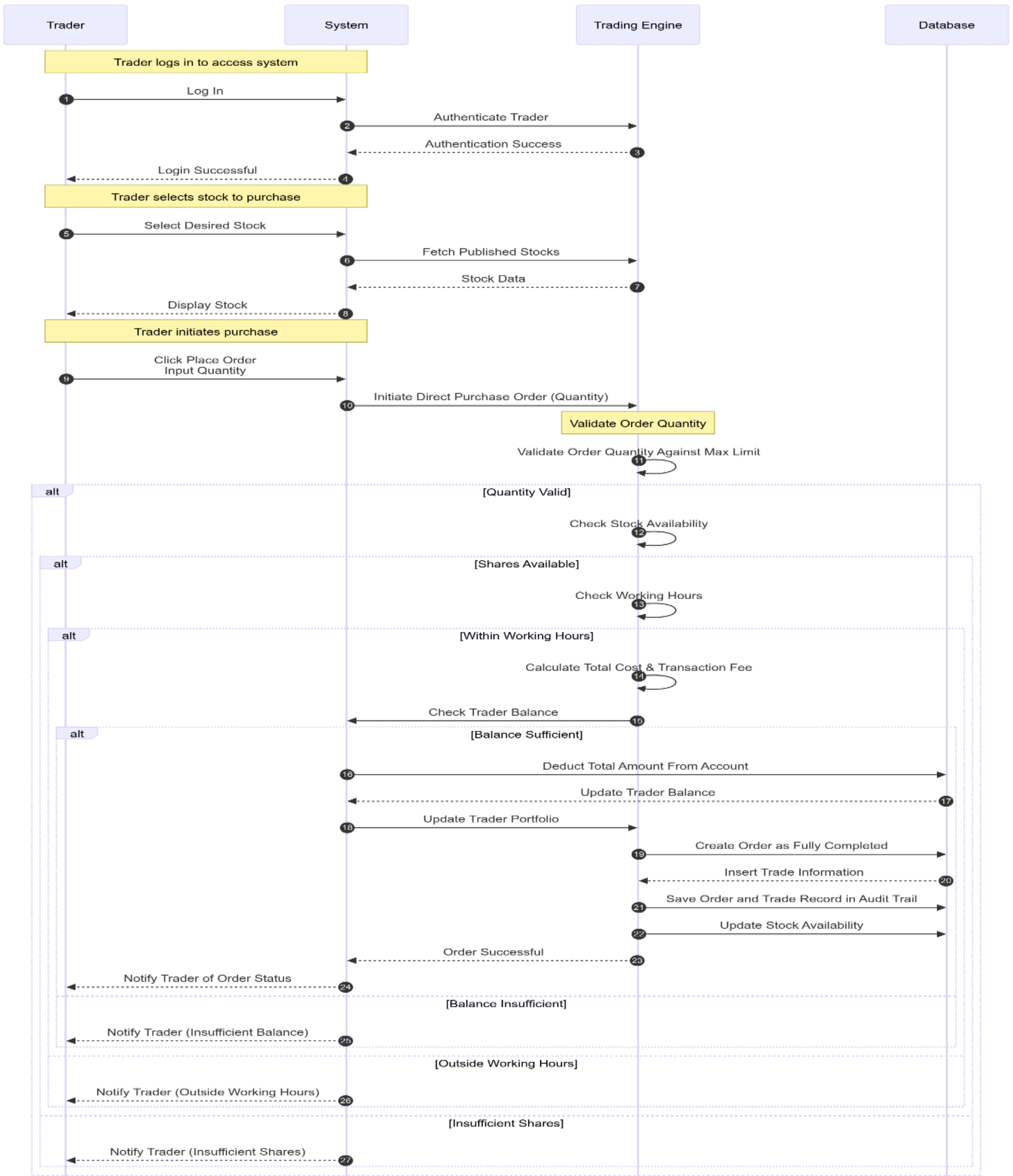

Trader
System
Trading Engine
Database
Trader logs in to access system
1 Log In
2 Authenticate Trader
3 Authentication Success
4 Login Successful
Trader selects stock to purchase
5 Select Desired Stock
6 Fetch Published Stocks
7 Stock Data
8 Display Stock
Trader initiates purchase
9 Click Place Order Input Quantity
10 Initiate Direct Purchase Order (Quantity)
Validate Order Quantity
11 Validate Order Quantity Against Max Limit
alt [Quantity Valid]
12 Check Stock Availability
alt [Shares Available]
13 Check Working Hours
alt [Within Working Hours]
14 Calculate Total Cost & Transaction Fee
15 Check Trader Balance
alt [Balance Sufficient]
16 Deduct Total Amount From Account
17 Update Trader Balance
18 Update Trader Portfolio
19 Create Order as Fully Completed
20 Insert Trade Information
21 Save Order and Trade Record in Audit Trail
22 Update Stock Availability
23 Order Successful
24 Notify Trader of Order Status
[Balance Insufficient]
25 Notify Trader (Insufficient Balance)
[Outside Working Hours]
26 Notify Trader (Outside Working Hours)
[Insufficient Shares]
27 Notify Trader (Insufficient Shares)

Figure 8 Direct Stock Purchase Sequence Diagram

Chapter 4: System Design

4.1 Introduction

This Ethiopian Stock Market Simulation Platform is a comprehensive system designed to emulate real-world stock trading environments. It facilitates user registration, KYC (Know Your Customer) verification, role-based access control, stock listing, order placement, trade execution, portfolio management, regulatory compliance, and real-time Email notifications. The platform is architected using Django for the backend, Django Rest Framework (DRF) for the API layer, and PostgreSQL for robust data storage. Deploying on a Virtual Private Server (VPS) ensures scalability, security, and high availability. This system is modular, comprising distinct applications such as **Users**, **Stocks**, **Regulations**, **Notifications**, and **Surveillance**, each handling specific functionalities to maintain a clear separation of concerns and facilitate maintainability.

4.2 Proposed Software Architecture

4.2.1 System Decomposition

To manage the complexity and enhance scalability, the platform is decomposed into several interconnected Django applications, each responsible for a specific domain:

1. **Users App (Authentication and Authorization)**
   - ➢ **Responsibilities**:
     - ✓ User registration and authentication.
     - ✓ Role assignment (Trader, Company Admin).
     - ✓ KYC document handling and verification.
     - ✓ OTP generation and validation for secure access.

2. **Stocks App (Trading and Portfolio Management)**
    - ➢ **Responsibilities**:
        - ✓ Managing listed companies stocks.
        - ✓ Facilitating order placement (buy/sell) and trade execution.
        - ✓ Maintaining user portfolios and handling dividends.
        - ✓ Logging daily closing prices and financial disclosures.
        - ✓ Detecting and managing suspicious trading activities.
3. **Regulations App (Compliance and Oversight)**
    - ➢ **Responsibilities**:
        - ✓ Defining and managing regulatory rules (e.g., daily trade limits).
        - ✓ Suspending traders from trading activities.
        - ✓ Setting and enforcing trading working hours.
4. **Notifications App (Communication)**
    - ➢ **Responsibilities**:
        - ✓ Sending real-time email notifications for trade executions, KYC approvals, suspensions, and other critical events.
        - ✓ Ensuring notifications are dispatched promptly without being stored in the database.
5. **Surveillance App (Monitoring and Security)**
    - ➢ **Responsibilities**:
        - ✓ Monitoring trades for anomalies and potential fraudulent activities.
        - ✓ Flagging suspicious trades for regulator review.
        - ✓ Integrating with the Regulations App to enforce suspensions based on surveillance findings.

This modular approach ensures each component can be developed, tested, and maintained independently, promoting scalability and ease of updates.

4.2.2 Hardware and Software Mapping

The system leverages a combination of hardware and software resources to ensure seamless operation and performance. The hardware includes servers hosting the Django backend and PostgreSQL database, while the software stack encompasses Django Rest Framework

(DRF), PostgreSQL, and the SendGrid cloud-based email service provider for communication. User devices (laptops, desktops, or mobile devices) serve as client interfaces, interacting with the backend through secure HTTPS APIs. Admin access is facilitated via a web-based admin panel hosted on the same server infrastructure.

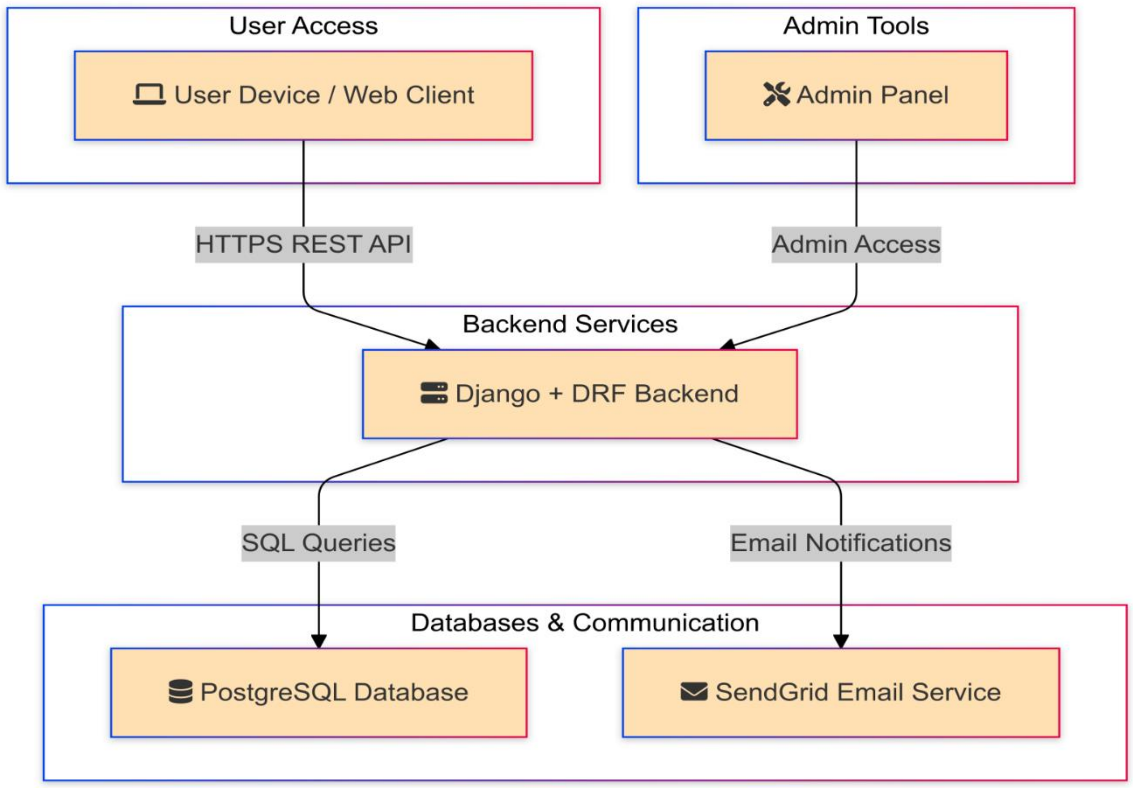


Figure 9 Hardware and software mapping

1. **User Access:**
   - **User Device / Web Client:** Represents end-users accessing the platform via web browsers or mobile devices.
2. **Backend Services:**
   - **Django + DRF Backend:** Core application responsible for handling business logic, API endpoints, and serving client requests.
   - **PostgreSQL Database:** Manages all persistent data storage, including user information, orders, trades, portfolios, and regulatory data.
   - **Email Server:** Handles the dispatching of email notifications for events such as trade executions, KYC verifications, and account updates.
3. **Admin Tools:**

- **Admin Panel:** Interface for administrators and regulators to manage users, oversee trading activities, enforce regulations, and perform other administrative tasks.

4.2.3 Persistent Data Modeling

Databases are the store houses of data used in the software system. The data is stored in tables inside a database. The general theme of database design is to handle information as an integrated whole, with a minimum redundancy and improved performance. Regardless of the type of data structure used, the objectives of the database are accuracy and integrity and successful recovery from failure, privacy and security of data, and good overall performance. [4]

Two essential settings for a database are

- **Primary key** - The field that is unique for all the record occurrences.
- **Foreign key** - The field used to set relation between tables. Normalization is a technique to avoid redundancy in the tables [4]

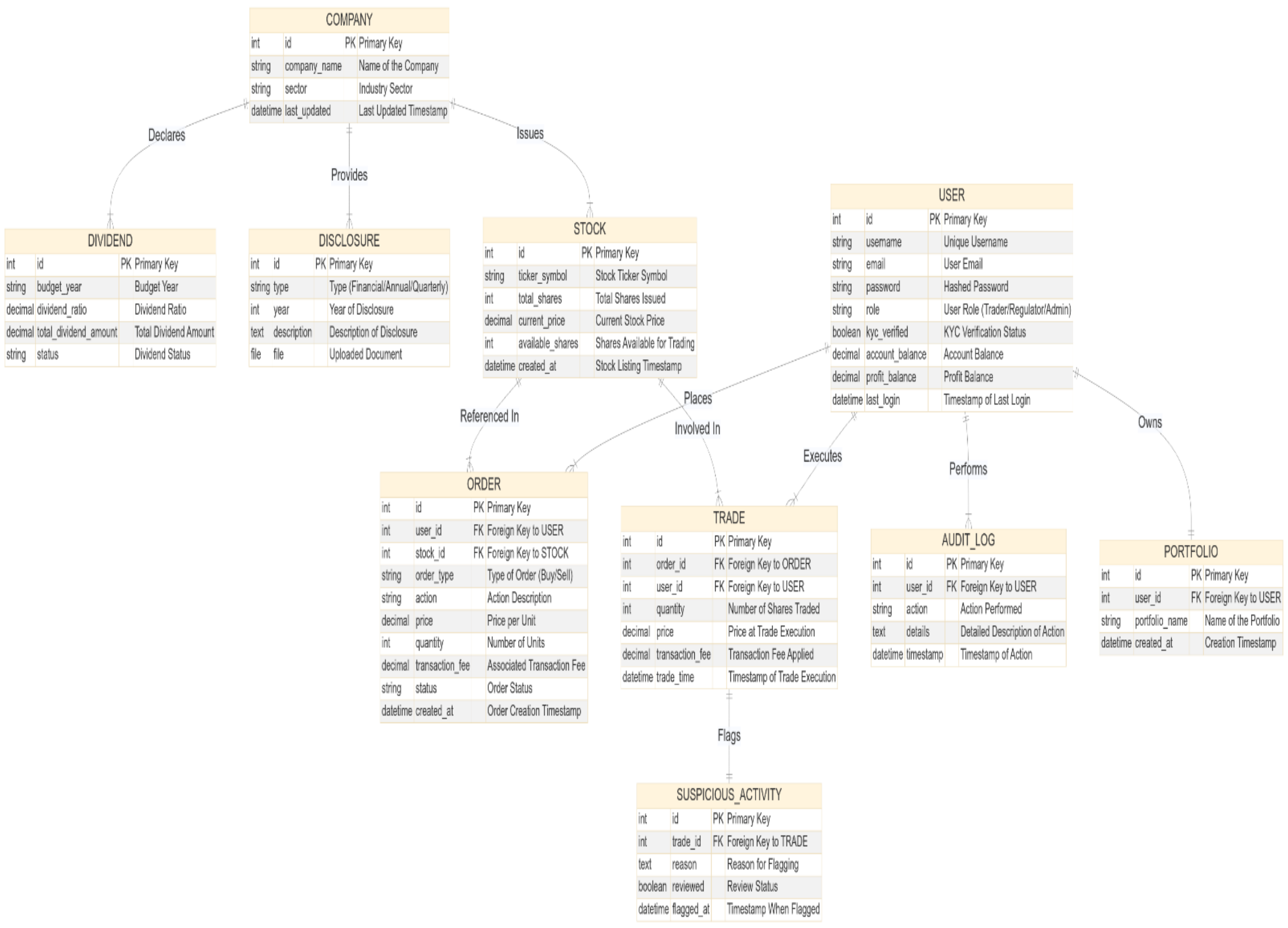

COMPANY
int id PK Primary Key
string company_name Name of the Company
string sector Industry Sector
datetime last_updated Last Updated Timestamp
Declares
Provides
Issues
DIVIDEND
int id PK Primary Key
string budget_year Budget Year
decimal dividend_ratio Dividend Ratio
decimal total_dividend_amount Total Dividend Amount
string status Dividend Status
DISCLOSURE
int id PK Primary Key
string type Type (Financial/Annual/Quarterly)
int year Year of Disclosure
text description Description of Disclosure
file file Uploaded Document
STOCK
int id PK Primary Key
string ticker_symbol Stock Ticker Symbol
int total_shares Total Shares Issued
decimal current_price Current Stock Price
int available_shares Shares Available for Trading
datetime created_at Stock Listing Timestamp
USER
int id PK Primary Key
string username Unique Username
string email User Email
string password Hashed Password
string role User Role (Trader/Regulator/Admin)
boolean kyc_verified KYC Verification Status
decimal account_balance Account Balance
decimal profit_balance Profit Balance
datetime last_login Timestamp of Last Login
Referenced In
Places
Involved In
Executes
Performs
Owns
ORDER
int id PK Primary Key
int user_id FK Foreign Key to USER
int stock_id FK Foreign Key to STOCK
string order_type Type of Order (Buy/Sell)
string action Action Description
decimal price Price per Unit
int quantity Number of Units
decimal transaction_fee Associated Transaction Fee
string status Order Status
datetime created_at Order Creation Timestamp
TRADE
int id PK Primary Key
int order_id FK Foreign Key to ORDER
int user_id FK Foreign Key to USER
int quantity Number of Shares Traded
decimal price Price at Trade Execution
decimal transaction_fee Transaction Fee Applied
datetime trade_time Timestamp of Trade Execution
AUDIT_LOG
int id PK Primary Key
int user_id FK Foreign Key to USER
string action Action Performed
text details Detailed Description of Action
datetime timestamp Timestamp of Action
PORTFOLIO
int id PK Primary Key
int user_id FK Foreign Key to USER
string portfolio_name Name of the Portfolio
datetime created_at Creation Timestamp
Flags
SUSPICIOUS_ACTIVITY
int id PK Primary Key
int trade_id FK Foreign Key to TRADE
text reason Reason for Flagging
boolean reviewed Review Status
datetime flagged_at Timestamp When Flagged

Figure 10 Persistent data modeling

4.2.4 Access Control and Security

The platform enforces strict access control and security measures to ensure data integrity, user privacy, and regulatory compliance. Access is primarily managed through role-based permissions, ensuring that each user type can only perform actions pertinent to their role.

User Roles:

1. **Trader**
   - **Permissions**:
     - ✓ Register and verify account via OTP.
     - ✓ Upload KYC documents.
     - ✓ Place buy/sell orders within set limits.
     - ✓ Execute direct purchases from listed companies.
     - ✓ View and manage personal portfolio.
     - ✓ Receive email notifications on trade executions and KYC status.
   - **Restrictions**:
     - ✓ Cannot approve KYC or manage regulations.
     - ✓ Cannot upload financial disclosures, publish stocks.
2. **Regulator**
   - **Permissions**:
     - ✓ Approve or reject user's KYC submissions.
     - ✓ Define and update regulations (e.g., daily trade limits).
     - ✓ Suspend traders from trading activities.
     - ✓ Review and mark suspicious activities.
     - ✓ View audit trail for compliance monitoring.
     - ✓ Define and update working hours for trading.

- **Restrictions**:
  - ✓ Cannot participate in trading activity and manage company disclosures and publishing stocks.

3. **Company Admin**
   - **Permissions**:
     - ✓ Upload and manage financial disclosures for listed companies.
     - ✓ Publish stock for the company they represent
     - ✓ View company-specific portfolios and viewing the company stock trading activity.
     - ✓ Input the declared dividend ratio and disburse the dividend for the traders
   - **Restrictions**:
     - ✓ Cannot suspend traders or review suspicious activities
     - ✓ Cannot Approve or reject user's KYC submissions
     - ✓ Cannot participate in trading activity

Access Control and Security:

Feature / Activity	**Requires Login**	**Trader**	**Regulator**	**Company Admin**
Register Account (Sign Up)	No	✓	✗	✓
OTP Verification	Yes	✓	✗	✓
Upload KYC Documents	Yes	✓	✗	✓
Approve/Reject KYC	Yes	✗	✓	✗
Place Buy/Sell Orders	Yes	✓	✗	✗
Direct Company Stock Purchase	Yes	✓	✗	✗

View/Manage Personal Portfolio	Yes	✓	✓	✓
Upload Disclosures	Yes	✗	✗	✓
Define/Update Regulations	Yes	✗	✓	✗
Suspend Trader	Yes	✗	✓	✗
Review/Mark Suspicious Activities	Yes	✗	✓	✗
View Audit Trial Logs	Yes	✗	✓	✗
Define/Update Working Hours	Yes	✗	✓	✗
Receive Email Notifications	Yes	✓	✗	✓

Table 17 Access Control and Security

Note: Regulators may have view permissions over all traders' portfolios for compliance monitoring.

Security Measures:

- **JWT Authentication**: Ensures secure and stateless user sessions.
- **OTP Verification**: Adds an additional layer of security during user registration.
- **Role-Based Access Control (RBAC)**: Restricts actions based on user roles.
- **Input Validation**: Prevents SQL injection, cross-site scripting (XSS), and other common attacks by validating and sanitizing all user inputs.
- **Secure File Handling**: KYC documents and disclosures are securely uploaded and stored with proper access restrictions.
- **Audit Logging**: All critical actions are logged for accountability and forensic analysis.

➢ **Rate Limiting and Throttling**: Implements rate limiting on API endpoints to prevent abuse and mitigate DDoS attacks

These measures collectively ensure that the platform remains secure, reliable, and compliant with industry standards.

4.2.5 Packages diagram

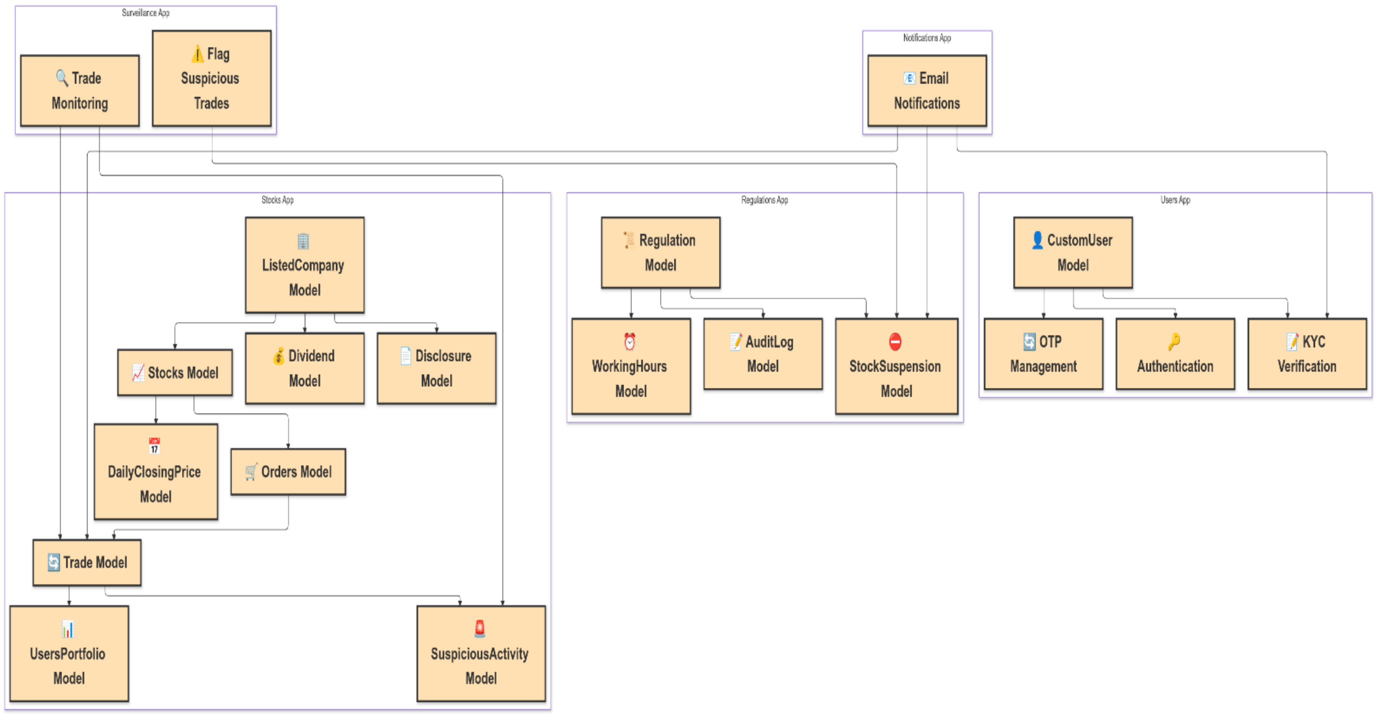


Figure 11 Package diagram

1. Users App

➢ **CustomUser Model:** Extends the default user model to include roles and other custom attribute (Trader, Regulator, Company Admin), KYC documents, and OTP codes.

- **Authentication:** Manages user login, logout, and session handling.
- **KYC Verification:** Handles the submission and approval of Know Your Customer (KYC) documents to ensure user compliance.
- **OTP Management:** Generates and validates One-Time Passwords for secure user verification processes.

2. Stocks App

- **ListedCompany Model:** Represents companies listed on the platform, storing details like company name and sector.
- **Stocks Model:** Manages stock information, including ticker symbols, current prices, total shares, and available shares.
- **Orders Model:** Facilitates the creation and management of buy and sell orders placed by traders.
- **Trade Model:** Executes trades by matching buy and sell orders, updating user portfolios accordingly.
- **UsersPortfolio Model:** Tracks individual user holdings, including the quantity of stocks owned and investment metrics.
- **Disclosure Model:** Allows company admins to upload financial disclosures and related documents.
- **Dividend Model:** Manages the distribution of dividends to shareholders based on company performance.
- **DailyClosingPrice Model:** Logs daily closing prices of stocks for historical reference and analysis.
- **SuspiciousActivity Model:** Flags and manages any trading activities that appear irregular or potentially fraudulent.

3. Regulations App

- **Regulation Model:** Stores and manages regulatory rules governing trading activities, such as daily trade limits and operational hours.

- **AuditLog Model:** Records all significant administrative and regulatory actions for accountability and compliance monitoring.
- **StockSuspension Model:** Manages the suspension of traders from trading activities based on regulatory decisions or detected suspicious activities.
- **WorkingHours Model:** Defines permissible trading hours to regulate when users can place orders, ensuring adherence to market operating times.

4. Notifications App

- **Email Notifications:** Handles the dispatching of email alerts for various platform events, including trade executions, KYC approvals/rejections, suspensions, and disclosures.

5. Surveillance App

- **Trade Monitoring:** Continuously monitors trading activities to identify unusual patterns or volumes that may indicate fraudulent behavior.
- **Flag Suspicious Trades:** Automatically flags trades that meet predefined suspicious criteria for further review by regulators.

4.2.6 Design Pattern (Architecture Layers)

The **Ethiopian Stock Market Simulation Platform** utilizes a **layered architectural pattern**, a standard design approach in the industry. This pattern organizes the system into distinct layers, each with specific responsibilities, promoting modularity, scalability, and ease of maintenance.

Layered Architecture Overview

1. **Presentation Layer (UI Subsystem)**
    - **Purpose:** Manages all user interactions and displays information to end-users.
    - **Components:** Web Client, Responsive Design, User Dashboards.

- ➢ **Responsibilities:** Render user-friendly interfaces, capture user inputs, and display data from the Business Logic Layer.

2. **Business Logic Layer (User Management & Trading Management Subsystems)**
   - ➢ **Purpose:** Encapsulates the platform's core functionalities and business rules.
   - ➢ **Components:** User Management Subsystem, Trading Management Subsystem.
   - ➢ **Responsibilities:** Authenticate users, enforce trading rules, and coordinate data flow between Presentation and Data Access Layers.
3. **Data Access Layer (Regulatory and Notification Subsystem)**
   - ➢ **Purpose:** Handles data storage, retrieval, and manipulation, ensuring data integrity.
   - ➢ **Components:** Regulatory Subsystem, Notification Subsystem, PostgreSQL Database.
   - ➢ **Responsibilities:** Perform CRUD operations, maintain audit logs, enforce regulations, and send user notifications.
4. **Infrastructure Layer**
   - ➢ **Purpose:** Provides essential services and infrastructure to support other layers.
   - ➢ **Components:** Email Server, Admin Panel.
   - ➢ **Responsibilities:** Manage communication services, support administrative tasks, and ensure system reliability.

Benefits of Layered Architecture

- **Separation of Concerns:** Each layer has a defined role, reducing interdependencies and simplifying maintenance.
- **Scalability:** Layers can be scaled independently to handle increasing demands efficiently.
- **Maintainability:** Isolated layers facilitate easier debugging, testing, and updates without impacting other components.

- **Reusability:** Common functionalities within layers can be reused across different parts of the application, enhancing consistency.
- **Flexibility:** New features or technologies can be integrated by modifying or adding layers without disrupting existing functionalities.

Implementation in Ethiopian Stock Market Simulation Platform

- **Presentation Layer:** Implemented through the **UI Subsystem**, offering intuitive interfaces for Traders, Regulators, and Company Admins.
- **Business Logic Layer:** Comprises the **User Management** and **Trading Management Subsystems**, handling authentication, KYC verification, order processing, and trade execution.
- **Data Access Layer:** Managed by the **Regulatory and Notification Subsystem**, interacting with the **PostgreSQL Database** to enforce regulations, maintain audit trails, and manage notifications.
- **Infrastructure Layer:** Supported by the **Email Server** and **Admin Panel**, ensuring efficient communication and administrative oversight.

4.2.7 Object Design Document (ODD)

The **Object Design Document (ODD)** provides a detailed blueprint of the Ethiopian Stock Market Simulation Platform's system architecture. It outlines the structure, behavior, and interactions of the system's objects, ensuring a clear understanding of how various components collaborate to achieve the platform's functionalities.

4.2.7.1 Class Interface

The **Class Interface** section delineates the primary classes within the Ethiopian Stock Market Simulation Platform, highlighting their attributes, methods, and relationships. This structured overview facilitates a comprehensive understanding of the system's object-oriented design.

Key Classes Overview

Class Name	Attributes	Methods	Relationships
CustomUser	- id: Integer - username: String - email: String - password: String - role: String - kyc_document: File - otp_code: String - is_approved: Boolean - kyc_verified: Boolean - account_balance: Decimal - profit_balance: Decimal	- register() - authenticate() - verify_otp() - upload_kyc()	One-to-One with UsersPortfolio One-to-Many with Orders, Trades
ListedCompany	- id: Integer - company_name: String - sector: String - last_updated: DateTime	- add_stock() - upload_disclosure() - issue_dividend()	One-to-One with Stocks One-to-Many with Disclosures
Stocks	- id: Integer - ticker_symbol: String - total_shares: Integer - current_price: Decimal - available_shares: Integer - max_trader_buy_limit: Integer	- publish_stock() - allocate_shares() - log_daily_closing_price()	Many-to-One with ListedCompany One-to-Many with Orders, Trades
Orders	- id: Integer - order_type: String - action: String - price: Decimal - quantity: Integer - status: String - transaction_fee: Decimal	- place_order() - cancel_order() - execute_order()	Many-to-One with CustomUser Many-to-One with Stocks One-to-One with Trade
Trade	- id: Integer - trade_time: DateTime - quantity: Integer	- execute_trade() - update_portfolio()	Many-to-One with CustomUser Many-to-One with

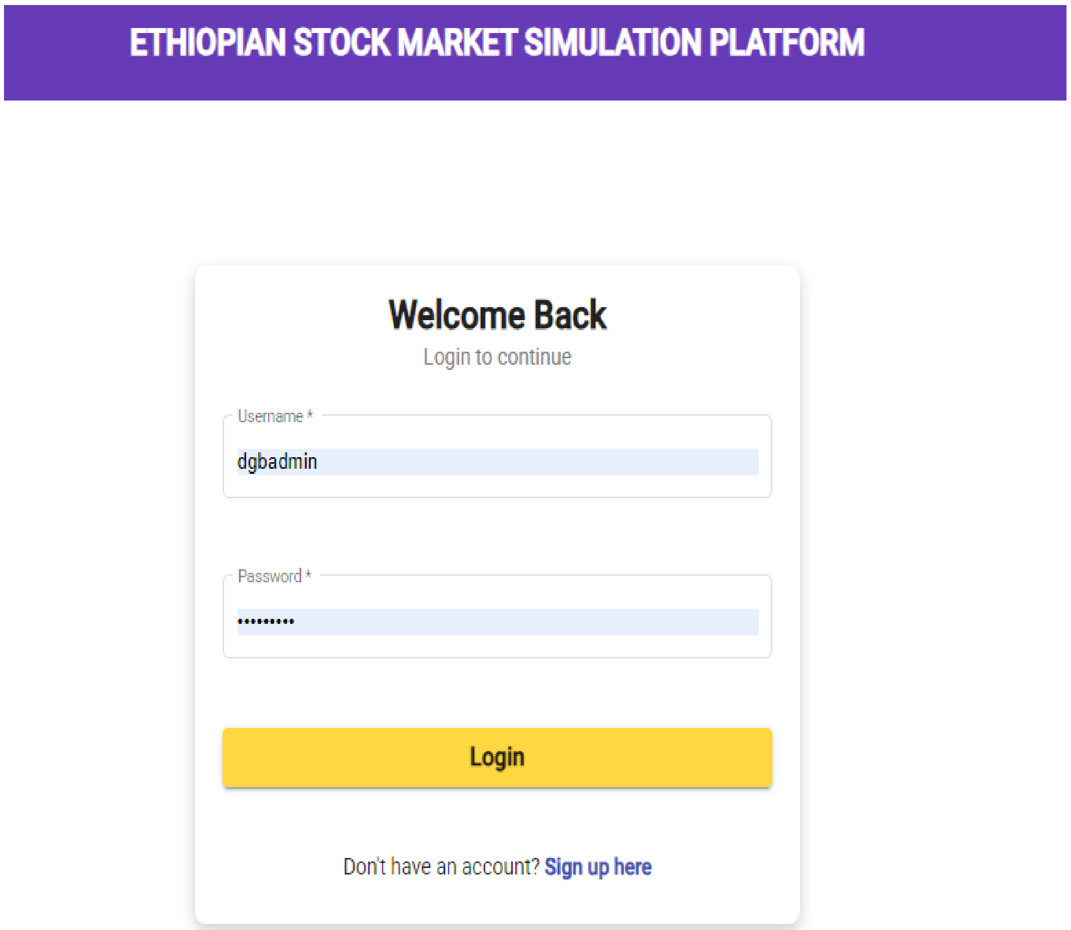
	- order_id: Integer - price: Decimal - transaction_fee: Decimal	- flag_suspicious_activity()	Stocks One-to-One with Orders One-to-Many with SuspiciousActivity
UsersPortfolio	- id: Integer - user_id: Integer - quantity: Integer - average_purchase_price: Decimal - total_investment: Decimal	- update_portfolio()	One-to-One with CustomUser
Disclosure	- id: Integer - company_id: Integer - type: String - year: Integer - file: File - description: String - uploaded_at: DateTime	- upload_disclosure() - update_disclosure()	Many-to-One with ListedCompany
Dividend	- id: Integer - company_id: Integer - budget_year: Integer - dividend_ratio: Decimal - total_dividend_amount: Decimal - status: String	- issue_dividend() - disburse_dividend()	Many-to-One with ListedCompany
DailyClosingPrice	- id: Integer - stock_id: Integer - date: Date - closing_price: Decimal	- log_closing_price() - retrieve_historical_prices()	Many-to-One with Stocks
Regulation	- id: Integer - name: String - value: String - description: String - created_by: Integer - created_at: DateTime - last_updated: DateTime	- create_regulation() - update_regulation() - delete_regulation()	One-to-Many with AuditLog One-to-Many with StockSuspension
StockSuspension	- id: Integer - trader_id: Integer - stock_id: Integer - suspension_type: String - initiator: String - reason: String - is_active: Boolean - created_at: DateTime - released_at: DateTime	- suspend_trader() - release_trader()	Many-to-One with Regulation Many-to-One with CustomUser Optional Many-to-One with Stocks

WorkingHours	- id: Integer - day_of_week: String - start_time: Time - end_time: Time	- define_working_hours() - update_working_hours()	One-to-Many with Orders
SuspiciousActivity	- id: Integer - reason: String - flagged_at: DateTime - reviewed: Boolean - trade_id: Integer	- flag_activity() - review_activity()	Many-to-One with Trade

Table 18 Class Interface

4.2.8 User Interface Design

The **User Interface Design** section details the visual and interactive elements of the Ethiopian Stock Market Simulation Platform, ensuring a seamless and intuitive user experience for all user roles. The platform's UI is crafted to be user-friendly, responsive, and accessible, catering to the diverse needs of Traders, Regulators, and Company Admins. The design emphasizes clarity, ease of navigation, and efficient access to essential functionalities.



ETHIOPIAN STOCK MARKET SIMULATION PLATFORM

Welcome Back

Login to continue

Username *

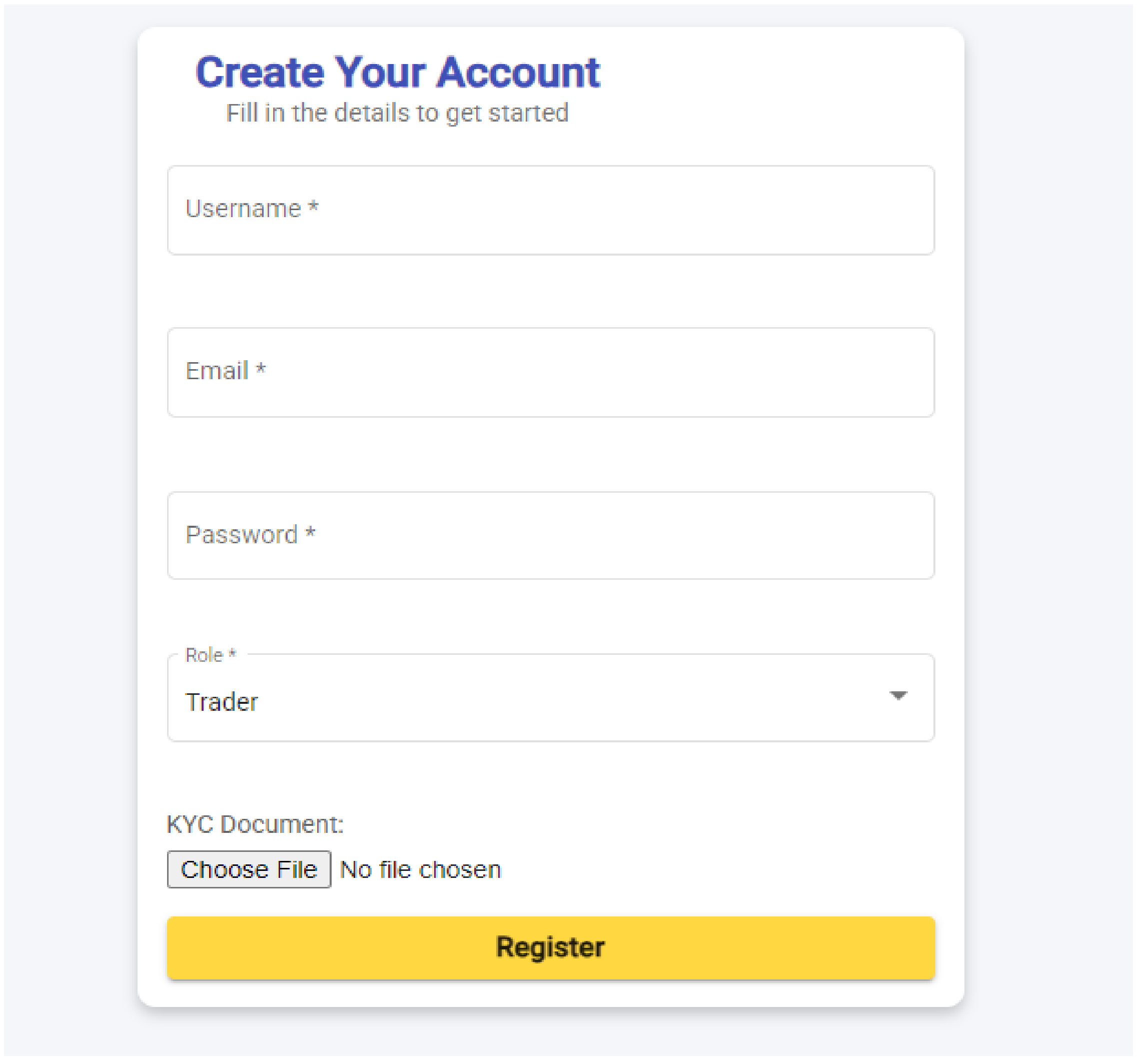
dgbadmin

Password *

•••••••••

Don't have an account? Sign up here

Figure 12 Login Page


Create Your Account

Fill in the details to get started

Username *

Email *

Password *

Role *

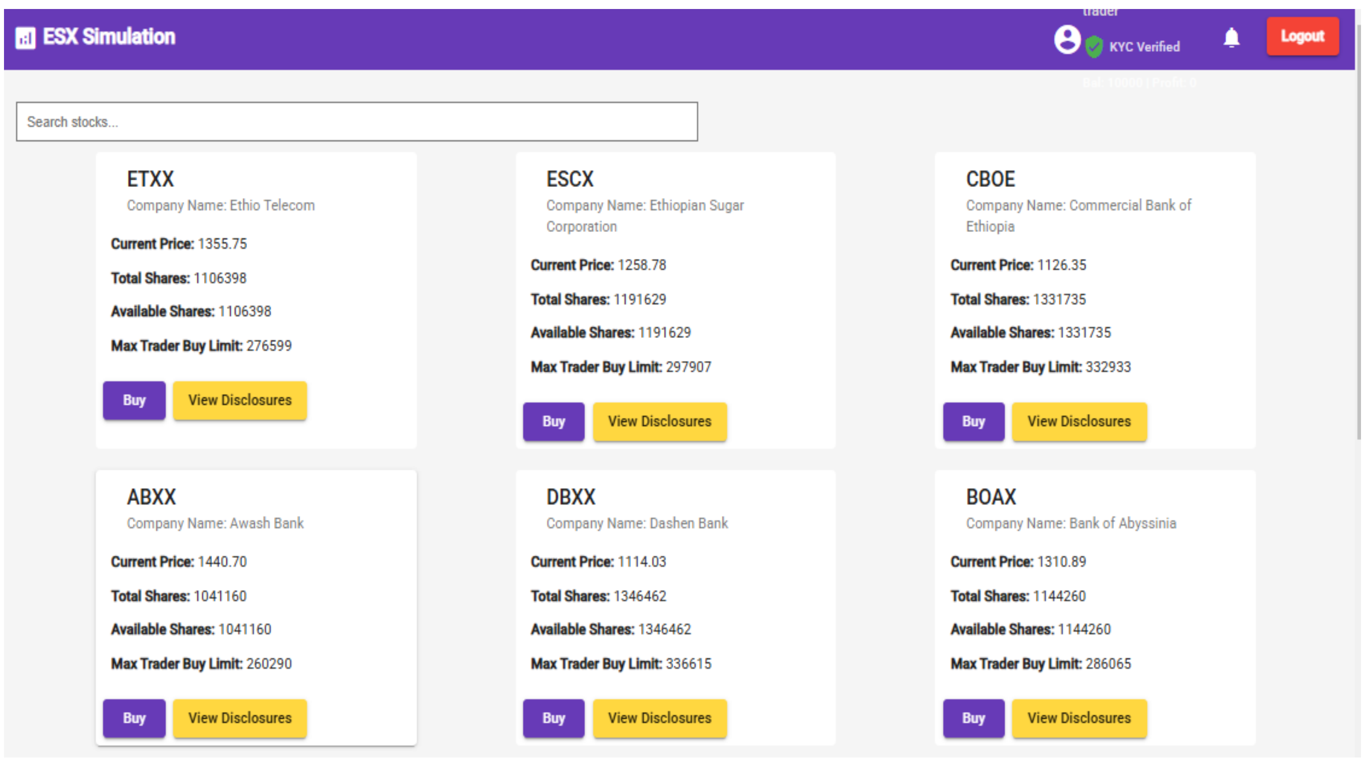
Trader

KYC Document:

Choose File No file chosen

Register

Figure 13 Registration Page



Bal: 10000 | Profit: 0

ETXX
Company Name: Ethio Telecom
Current Price: 1355.75
Total Shares: 1106398
Available Shares: 1106398
Max Trader Buy Limit: 276599

ESCX
Company Name: Ethiopian Sugar Corporation
Current Price: 1258.78
Total Shares: 1191629
Available Shares: 1191629
Max Trader Buy Limit: 297907

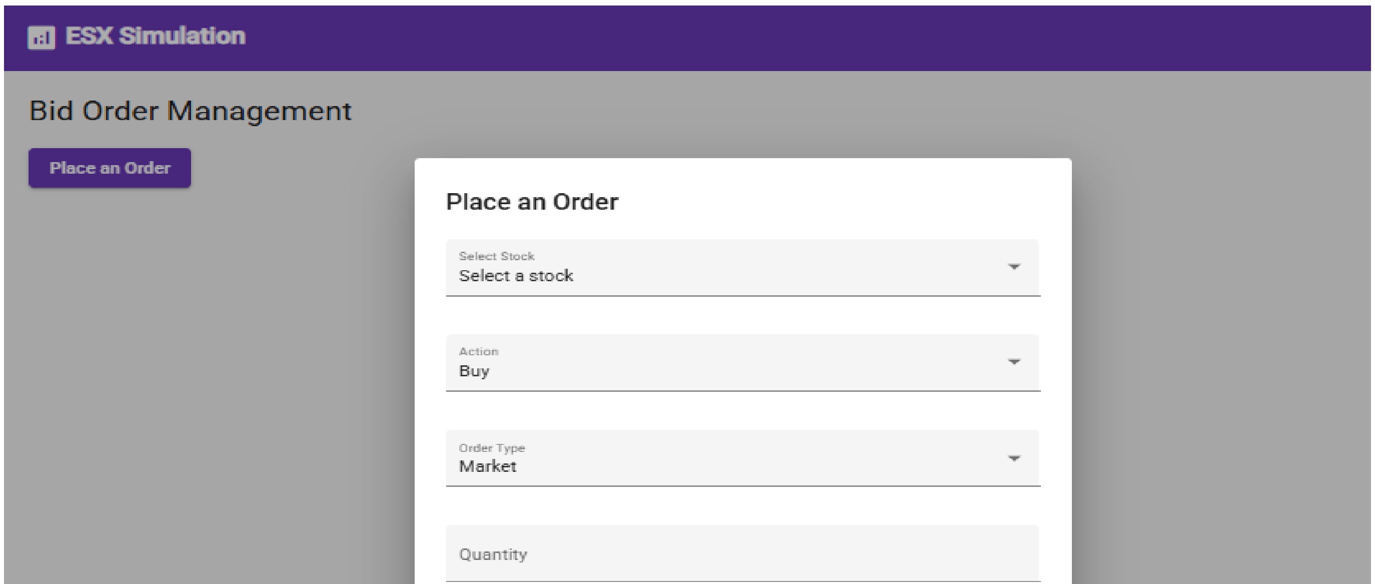
CBOE
Company Name: Commercial Bank of Ethiopia
Current Price: 1126.35
Total Shares: 1331735
Available Shares: 1331735
Max Trader Buy Limit: 332933

ABXX
Company Name: Awash Bank
Current Price: 1440.70
Total Shares: 1041160
Available Shares: 1041160
Max Trader Buy Limit: 260290

DBXX
Company Name: Dashen Bank
Current Price: 1114.03
Total Shares: 1346462
Available Shares: 1346462
Max Trader Buy Limit: 336615

BOAX
Company Name: Bank of Abyssinia
Current Price: 1310.89
Total Shares: 1144260
Available Shares: 1144260
Max Trader Buy Limit: 286065

Figure 14 Listed Company stock list



Bid Order Management

Place an Order

Select Stock: Select a stock
Action: Buy
Order Type: Market
Quantity

Figure 15 Placing Bid order

Key UI Components

1. **Login and Registration Pages**
   - **Features:**
     - ✓ User authentication via username and password.
     - ✓ OTP verification for secure access.
     - ✓ KYC document upload during registration.
   - **Design Considerations:**
     - ✓ Clean and straightforward layout.
     - ✓ Clear instructions and validation messages.
2. **User Dashboards**
   - **Traders:**
     - ✓ **Portfolio Overview:** Displays current holdings, total investment, and profit/loss metrics.
     - ✓ **Trade History:** Lists past trades with details like stock symbol, quantity, price, and date.
     - ✓ **Order Placement:** Interface to place buy/sell orders with options to set order type, quantity, and price.
   - **Regulators:**
     - ✓ **Regulation Management:** Tools to define and update trading regulations.

        - ✓ **Audit Logs:** Access to comprehensive logs of administrative and regulatory actions.
        - ✓ **Suspension Controls:** Interface to suspend or reinstate Traders based on activity reviews.
    - ➢ **Company Admins:**
        - ✓ **Disclosure Management:** Upload and manage financial disclosures for listed companies.
        - ✓ **Dividend Distribution:** Tools to issue and track dividend distributions to shareholders.

3. **Trading Interface**
    - ➢ **Features:**
        - ✓ Real-time stock price updates and charts.
        - ✓ Order book display showing current buy and sell orders.
        - ✓ Trade execution confirmation and notifications.
    - ➢ **Design Considerations:**
        - ✓ Interactive charts for better data visualization.
        - ✓ Responsive elements for trading on various devices.
4. **Notification Center**
    - ➢ **Features:**
        - ✓ Centralized hub for all email notifications and alerts.
        - ✓ Status indicators for pending actions like KYC verification or order execution.
    - ➢ **Design Considerations:**
        - ✓ Organized layout to easily track and manage notifications.
        - ✓ Clear categorization based on event types.
5. **Admin Panel**
    - ➢ **Features:**
        - ✓ Comprehensive management tools for overseeing platform operations.
        - ✓ User management capabilities to approve/reject KYC, suspend Traders, and manage roles.

    - ✓ Regulatory tools to define and enforce trading rules.
- ➢ **Design Considerations:**
    - ✓ Secure access with role-based permissions.
    - ✓ Intuitive navigation for efficient administrative tasks.

Chapter 5: Implementation

5.1 Mapping Models to Code

The **Ethiopian Stock Market Simulation Platform** transforms the designed models into functional Django components. Each application (Users, Stocks, Regulations, Notifications, etc.) is developed with clear responsibilities and well-organized code.

5.1.1 Users App

Model:

CustomUser

Below is a simplified sample of the custom user model extending Django's AbstractUser:

```
# users/models.py
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.utils import timezone
from ethio_stock_simulation.utils import generate_otp

class CustomUser(AbstractUser):
    ROLE_CHOICES = [
        ('trader', 'Trader'),
        ('regulator', 'Regulator'),
        ('company_admin', 'Company Admin'),
    ]
    role = models.CharField(max_length=15, choices=ROLE_CHOICES, default='trader')
    is_approved = models.BooleanField(default=False)
    kyc_document = models.FileField(upload_to='kyc_documents/', blank=True, null=True)
    kyc_verified = models.BooleanField(default=False)
    company_id = models.IntegerField(null=True, blank=True)
    account_balance = models.DecimalField(max_digits=15, decimal_places=2, default=0.00, null=True,
blank=True)
    profit_balance = models.DecimalField(max_digits=15, decimal_places=2, default=0.00, null=True,
blank=True)
    date_registered = models.DateTimeField(default=timezone.now)
    last_login = models.DateTimeField(null=True, blank=True)

    # OTP Fields
    otp_code = models.CharField(max_length=6, blank=True, null=True)
    otp_sent_at = models.DateTimeField(null=True, blank=True)
    otp_verified = models.BooleanField(default=False)
    otp_attempts = models.IntegerField(default=0)
```

Serializer:

UserSerializer

Defines the fields and validation logic for user registration and updates:

```
# users/serializers.py
from rest_framework import serializers
from .models import CustomUser

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = [
            'id', 'username', 'email', 'password', 'role', 'kyc_document',
            'kyc_verified', 'account_balance', 'profit_balance',
            'otp_verified', 'otp_attempts', 'is_approved'
        ]
        extra_kwargs = {
            'password': {'write_only': True},
            'kyc_verified': {'read_only': True},
            'is_approved': {'read_only': True},
            'account_balance': {'read_only': True},
            'profit_balance': {'read_only': True},
        }

    def create(self, validated_data):
        user = CustomUser.objects.create_user(
            username=validated_data['username'],
            password=validated_data['password'],
            email=validated_data['email'],
            role=validated_data['role'],
            kyc_document=validated_data.get('kyc_document', None),
        )
        return user
```

View:

RegisterUser

Manages the user registration process, including OTP generation and notification:

```
# users/views.py
from rest_framework import generics, status
from rest_framework.response import Response
from django.utils import timezone
from .serializers import UserSerializer
from .models import CustomUser
from notifications.utils import notify_user_email
```

```
class RegisterUser(generics.CreateAPIView):
    queryset = CustomUser.objects.all()
    serializer_class = UserSerializer

    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()

        # Generate and send OTP (simplified for example)
        otp = "123456"
        user.otp_code = otp
        user.otp_sent_at = timezone.now()
        user.save()

        subject = "Verify Your ESX Account"
        message = f"Dear {user.username},\n\nYour OTP is {otp}.\n\nThank you!"
        notify_user_email(subject, message, [user.email])

        return Response(
            {"detail": "Registration successful. OTP sent to your email."},
            status=status.HTTP_201_CREATED
        )
```

Key Functionalities:

- **User Registration:** Allows new users to register by providing essential details. An OTP is generated and sent to the user’s email for verification.
- **KYC Document Handling:** Users can upload KYC documents during registration.
- **Role Assignment:** Assigns roles (Trader, Regulator, Company Admin) based on user input.

5.1.2 Stocks App

Model:

Stocks

Represents company stocks and associated metadata:

```
# stocks/models.py
from django.db import models
from django.utils import timezone

class Stocks(models.Model):
    ticker_symbol = models.CharField(max_length=10, unique=True)
    company = models.ForeignKey('ListedCompany', on_delete=models.CASCADE, related_name='stocks')
    total_shares = models.IntegerField()
    current_price = models.DecimalField(max_digits=15, decimal_places=2)
    available_shares = models.IntegerField()
    max_trader_buy_limit = models.IntegerField(default=1000)
    created_at = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"{self.ticker_symbol} ({self.company.company_name})"
```

Serializer:

StocksSerializer

```
# stocks/serializers.py
from rest_framework import serializers
from .models import Stocks

class StocksSerializer(serializers.ModelSerializer):
    class Meta:
        model = Stocks
        fields = '__all__'
```

View:

StocksViewSet
Implements RESTful endpoints for managing stocks:

```
# stocks/views.py
from rest_framework import viewsets
from .models import Stocks
from .serializers import StocksSerializer
from rest_framework.permissions import IsAuthenticated

class StocksViewSet(viewsets.ModelViewSet):
    queryset = Stocks.objects.all()
    serializer_class = StocksSerializer
    permission_classes = [IsAuthenticated]
```

Key Functionalities:

- **Stock Management:** Handles essential operations on stocks, including viewing, creating, and updating.
- **Direct Purchase Logic:** Can be extended to bypass order matching for direct stock purchases.

Key Functionalities:

- **Email Dispatching:** Sends notifications (trade confirmations, KYC updates, regulatory alerts) without storing them in the database, optimizing performance.

5.2 Suspicious Activity App

Model: SuspiciousActivity

Tracks and manages suspicious trading activities.

```
# suspicious_activity/models.py
from django.db import models
from django.utils import timezone
from trades.models import Trade

class SuspiciousActivity(models.Model):
    reason = models.TextField()
    flagged_at = models.DateTimeField(default=timezone.now)
    reviewed = models.BooleanField(default=False)
    trade = models.ForeignKey(Trade, on_delete=models.CASCADE, related_name='suspicious_activities')

    def __str__(self):
        return f"Suspicious Activity for Trade ID: {self.trade.id}"
```

Serializer: SuspiciousActivitySerializer

Serializes suspicious activity data.

```
# suspicious_activity/serializers.py
from rest_framework import serializers
from .models import SuspiciousActivity

class SuspiciousActivitySerializer(serializers.ModelSerializer):
    class Meta:
        model = SuspiciousActivity
        fields = '__all__'
        read_only_fields = ['flagged_at', 'reviewed']
```

View: SuspiciousActivityViewSet

Manages creation and review of suspicious activities.

```
# suspicious_activity/views.py
from rest_framework import viewsets, status
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated
from .models import SuspiciousActivity
from .serializers import SuspiciousActivitySerializer
```

```
class SuspiciousActivityViewSet(viewsets.ModelViewSet):
    queryset = SuspiciousActivity.objects.all()
    serializer_class = SuspiciousActivitySerializer
    permission_classes = [IsAuthenticated]

    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        suspicious_activity = serializer.save()

        return Response(serializer.data, status=status.HTTP_201_CREATED)

    def update(self, request, *args, **kwargs):
        instance = self.get_object()
        instance.reviewed = True
        instance.save()
        serializer = self.get_serializer(instance)
        return Response(serializer.data)
```

Key Functionalities:

- **Activity Monitoring:** Detects and flags unusual trading patterns.
- **Review Process:** Allows regulators to review and mark activities as reviewed, ensuring accountability.

5.3 Testing and Evaluation

Upon completing the development of the **Ethiopian Stock Market Simulation Platform**, a comprehensive testing and evaluation process is essential to ensure its functionality, reliability, and security.

Unit testing is conducted to verify that each individual component, such as models, serializers, and views, operates correctly. The goal is to achieve at least 80% test coverage, ensuring that most parts of the application are thoroughly tested.

Integration testing follows, where different modules like Users, Stocks, and Trading are tested together to confirm they interact seamlessly. This includes scenarios such as user registration, trade execution, and notification dispatching. Additionally, **performance and stress testing** assesses the platform's ability to handle a numbers of simultaneous users and transactions, ensuring smooth operation.

Security testing is performed to identify and address potential vulnerabilities, safeguarding user data and maintaining system integrity. Finally, **User Acceptance**

Testing (UAT) involves real users Traders, Regulators, and Company Admins to validate the platform's usability and effectiveness in a real-world context. This structured testing approach guarantees that the platform meets high standards of quality and is ready for deployment.

5.4 System Maintenance

Ensuring the long-term success of the Ethiopian Stock Market Simulation Platform requires diligent system maintenance to guarantee its availability, performance, and security. Preventive maintenance involves regular monitoring of system performance and conducting routine security assessments to proactively identify and address potential issues. This includes applying necessary software updates and patches to keep the system secure and efficient.

Corrective maintenance addresses any unexpected problems or bugs that arise, ensuring they are resolved promptly to minimize downtime and user disruption. Additionally, system updates are regularly performed to upgrade the operating system, frameworks, and third-party components, maintaining compatibility and enhancing security. To support scalability, server resources are adjusted.

Security maintenance is a continuous effort, involving regular vulnerability scans and access reviews to enforce strict security protocols and protect sensitive data. Comprehensive **documentation and training** are maintained to ensure that the development and maintenance teams are well-equipped to manage the system effectively. By adhering to these maintenance practices, the platform remains reliable, secure, and capable of providing a seamless user experience.

Chapter 6: Conclusion and Recommendation

6.1 Conclusion

The **Ethiopian Stock Market Simulation Platform** is a groundbreaking initiative aimed at preparing Ethiopian stakeholders for the launch of the Ethiopian Stock Market. By integrating **user management**, **KYC verification**, **stock trading mechanisms**, and **regulatory oversight** within a robust Django framework, the platform effectively simulates real-world trading in a **risk-free environment**. Key features include seamless user registration with OTP verification, comprehensive KYC processes, advanced trading functionalities, real-time notifications, and proactive monitoring of suspicious activities. Leveraging **PostgreSQL's reliability** and **Django's extensibility**, the platform is designed to scale and adapt to evolving regulatory requirements. Drawing inspiration from established simulators like **Investopedia's Simulator**, the **Ethiopian Stock Market Simulation Platform** upholds industry-standard security and reliability, positioning itself as an essential tool for **education**, **strategy testing**, and **regulatory training** within Ethiopia's emerging stock market ecosystem.

6.2 Recommendation

To further enhance the **Ethiopian Stock Market Simulation Platform** and ensure its effectiveness as a training tool, the following recommendations are proposed:

1. **Advanced Analytical Tools and Dashboards**
    - **Implementation:** Integrate real-time dashboards using libraries like **Chart.js** or **D3.js** to visualize market trends and trading performance.

- **Benefits:** Provides users with actionable insights and enables regulators to monitor market activities more effectively.

2. **Extended Order Types and Financial Instruments**
    - **Implementation:** Introduce complex order types such as **stop-loss** and **iceberg orders**, and expand support to additional financial instruments like **bonds** and **commodities**.
    - **Benefits:** Offers a more realistic trading environment and accommodates a wider range of trading strategies.
3. **AI-Driven Surveillance and Anomaly Detection**
    - **Implementation:** Develop machine learning models to analyze trading patterns and detect fraudulent activities or market manipulations.
    - **Benefits:** Enhances the platform's ability to maintain market integrity and reduces reliance on manual monitoring.
4. **Automated Regulatory Compliance and Reporting**
    - **Implementation:** Automate the generation of compliance reports and streamline KYC verification workflows.
    - **Benefits:** Streamlines regulatory processes, ensuring timely and accurate compliance with minimal administrative overhead.
5. **Enhanced User Experience and Interface Improvements**
    - **Implementation:** Continuously refine the user interface based on feedback, incorporating features like **customizable dashboards** and **interactive tutorials**.
    - **Benefits:** Improves user satisfaction and facilitates easier adoption across diverse user groups.
6. **Integration with External Financial Data Providers**
    - **Implementation:** Connect the platform with external APIs to fetch real-time financial data and market indicators.
    - **Benefits:** Enriches the simulation experience by reflecting actual market dynamics, providing users with comprehensive information for informed trading strategies.

By implementing these enhancements, the **Ethiopian Stock Market Simulation Platform** will significantly improve its functionality, user engagement, and compliance capabilities. This will ensure the platform remains a leading tool for stock trading simulations and regulatory training, effectively preparing Ethiopian stakeholders for the successful launch and operation of the Ethiopian Stock Exchange.

Reference

[1] https://ecma.gov.et/. Ethiopian Capital Market Authority (ECMA). Ethiopian Capital Market Authority Directives and Regulatory Framework.. [Online]. https://ecma.gov.et/

[2] https://www.investopedia.com/. Investopedia. Stock Market Simulator: A Learning Tool. Accessed from Investopedia. [Online]. https://www.investopedia.com/

[3] ESX Academy. (2024) ESX Academy. [Online]. https://esxacademy.com/

[4] [Ambler 2004], "the object primer second edition (Scott W. Amber)," in *the object primer second edition (Scott W. Amber)*., 2004.

[5] https://www.djangoproject.com/. Django Software Foundation. Django Documentation: The Web Framework for Perfectionists with Deadlines. [Online]. https://www.djangoproject.com/

[6] capital market ethiopia. [Online]. https://www.capitalmarketethiopia.com/

[7] UNDP. undp. [Online]. https://www.undp.org/ethiopia/blog/ethiopia-capital-market-launches-regulatory-sandbox

Abbreviations

ECMA:	Ethiopian Capital Market Authority
ESX:	Ethiopian Stock Exchange
OTP:	One-Time Password
KYC:	Know Your Customer
RBAC:	Role-Based Access Control
VPS:	Virtual Private Server
CRUD:	Create, Read, Update, Delete
DRF:	Django Rest Framework
DDoS:	Distributed Denial of Service
JSON:	JavaScript Object Notation