Software Requirements Specification

for

<Innovesters>

Version 1.0 approved

Prepared by <TechTitans>

Anushree, Ezra, George, Jai, Marcus, Qin Yuan

<Nanyang Technological University>

17 Nov 2023

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
George	17/11/23	Introduction and Overall Description	1.0
Marcus	17/11/23	Appendix and APIs used	1.1
Ezra	17/11/23	External Interface	1.2
Qin Yuan	18/11/23	Software and Hardware Interfaces	1.3
Anushree	19/11/23	System Features	1.4
Jai	19/11/23	Other non-functional requirements	1.5

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) document is intended for the Innovestors web application, build version 1.0. The purpose of this SRS document is to describe the requirements specifications for the Innovestors web application to facilitate the development and production process for all stakeholders. This document will include the system features, limitations, functional and non-functional requirements, as well as use case descriptions. The Innovesters web application aims to provide users with a portfolio that is optimized based on real-time and recent information as well as users' specific risk tolerance levels, and allow users to make informed choices from there. This Innovestors web application is built using the following frameworks, front-end framework being React JS, back-end framework being Django as well as the usage of PostgreSQL as a database system.

1.2 Document Conventions

This section describes all standards and typographical conventions that we will be following when writing the SRS document. Refer to appendix A for the list of definitions (Data dictionary) for special terms used during the description of our report. This document follows the IEEE standards. Priorities of higher-level requirements are inherited by detailed-level requirements.

Font	Arial, Black
Heading	Times, Bold, Size 18
Subheading	Times, Bold, Size 14
Text	Size 11

1.3 Intended Audience and Reading Suggestions

This section will describe the different types of readers that the document is intended for and will provide suggestions to the sequence for reading the document.

This SRS document is meant for all stakeholders such as Innovestors users, Innovestors development team, Innovestors testing team as well as Innovestors project manager.

This document begins with the Introduction where it will state the purpose and product scope about the project. Thereafter, it will be followed by the overall description of the product which includes the application functionalities, several design constraints and assumptions of application. Then it will be followed by the introduction of both system functionalities and non-functional requirements of the product. This document will end with an appendix that encompasses the glossary and analysis models.

All stakeholders should begin reading the document from the introduction section - 1.1 Purpose, 1.2 Document Conventions and Appendix A (Data dictionary) to get a glimpse of what the product is about and has a brief introduction to the product.

The Innovestors development team can then proceed with reading section 2. Section 2 will provide the development team with a high-level description of product perspective, functionalities that will be useful in helping the development team to build their product. After which, section 4 (System Feature) in which developers can have a better understanding about the functionalities and features of the application.

On the other hand, the Innovestors users, testing team and project managers can read through this document in sequential order.

1.4 Product Scope

This section will provide a short description of the product's purpose, benefits, objective as well as goals.

In the fast-paced country of finance like Singapore, making informed investment decisions is crucial for individuals seeking financial growth. However, the abundance of information and the dynamic nature of the stock market often make it challenging for investors to identify the most promising stocks for their portfolios. As Singapore embraces the Smart Nation movement, our team, recognizing the need for a sophisticated yet user-friendly solution, introduces the Innovestors web application.

Innovestors aims to revolutionize the way Singaporeans approach stock investments by providing a comprehensive platform for optimized returns. The challenge lies in the complexity of stock market data and the varying expertise of individual investors. Many existing investment applications offer data presentation without clear guidance, providing a long list of stock options to users, making it difficult for users to make strategic and informed decisions. Our application addresses this issue by leveraging on yahoo APIs to analyze stock market trends, historical data, and other financial indicators. We tap on machine learning algorithms to filter out only the stocks which are likely to perform, based on the user's risk preference. Unlike traditional investment platforms, Innovestors goes beyond data presentation; it interprets the information to recommend the top-performing stocks tailored to each user's risk tolerance, financial goals, and investment horizon. In addition, with Innovestors, users have the option to find out more about a particular stock, and will be redirected to the relevant sites to read on the trending news regarding the stock. This keeps users always up to date, helping them make better investing decisions.

Innovestors aims to empower users with personalized investment recommendations based on real-time market conditions. Investors can access a curated list of stocks that align with their financial objectives, enabling them to make well-informed decisions for optimized returns. Additionally, the platform facilitates the tracking of users' own portfolio transactions, providing a holistic view of their investment journey.

1.5 References

- IEEE 830-1998 Template

- Javascript Documentation https://devdocs.io/javascript/
- Bootstrap Documentation https://getbootstrap.com/docs/4.1/getting-started/introduction/
- Bootstrap-React Documentation https://react-bootstrap.github.io/
- Yahoo Finance Website https://sg.finance.yahoo.com/
- Brevo Documentation https://developers.brevo.com/docs/getting-started
- Django Documentation https://docs.djangoproject.com/en/4.2/
- PostGresSQL Documentation https://www.postgresql.org/docs/

2. Overall Description

2.1 Product Perspective

The Innovesters software is a standalone product designed to provide users with a platform for optimizing and managing their stock portfolios. It operates independently, offering features such as portfolio optimization, stock selection, transaction logging, and access to information on the top 50 stocks. While it is a self-contained product, it may integrate with external financial data sources for real-time stock information. The system comprises major components like the Portfolio Optimizer, Playground, Transaction Log, and Stocks Information.

2.2 Product Functions

- Optimisation: users may generate the best portfolio based on current data and user preferences
- Playground: users may select several of their preferred stocks and generate the optimal portfolio based on their selection
- Transaction Log: users may keep track of the stocks which they bought or sold within Innovestors
- Stocks: users may read up on the latest news and key information about the top 50 stocks.

2.3 User Classes and Characteristics

- **Novice Traders**: Limited exposure to stock market operations, making them less familiar with complex financial concepts. They engage in occasional use for basic portfolio management, relying on simplified features and educational resources. Novice traders typically have limited technical expertise and prefer user-friendly interfaces for ease of navigation.

- **Experienced Traders**: Possess advanced knowledge of stock market dynamics and are well-versed in financial intricacies. They engage in frequent use for in-depth portfolio analysis and trading, utilizing advanced features such as algorithmic trading. These users exhibit high technical proficiency, demanding sophisticated tools and real-time data for effective decision-making.

- **Administrators**: Primarily involved in system configuration and maintenance, ensuring the smooth operation of the platform. Their interaction is infrequent and mainly focused on backend management tasks. Administrators possess technical expertise in system administration, prioritizing system stability and security.

2.4 Operating Environment

2.4.1 Software Environment

- Operating System: Compatible with Windows, macOS, and Linux.
- Web Browser Compatibility: Supports major browsers (Chrome, Firefox, Safari).

2.4.2 Hardware Environment

- Processor: Minimum dual-core processor.
- Memory: Minimum 4GB RAM.

2.4.3 External Interfaces

- Integration: Interfaces with external financial data sources for real-time stock information.

2.5 Design and Implementation Constraints

Innovesters relies on the Python/Django framework for its development, providing a solid foundation for the system. However, it is essential to note that our use of the yfinance library introduces a constraint. This library, although a valuable tool for fetching financial data, has been discontinued since 2017.

To tackle this, we combined yfinance with web scraping. This way, even if yfinance does not have all the data we need, we can still fetch complete stock information. Our goal is to ensure Innovesters provides accurate and current financial insights. Despite the hiccup with yfinance, we're committed to delivering a strong and dependable platform for all your financial needs.

2.6 User Documentation

- Video Demos: Video demonstrations for a more engaging and visual learning experience attached.
- User Manual: A simple guide for novice traders to optimize their portfolio (Appendix C)
- README files with specific instructions to set up the environment for both front-end and back-end.

2.7 Assumptions and Dependencies

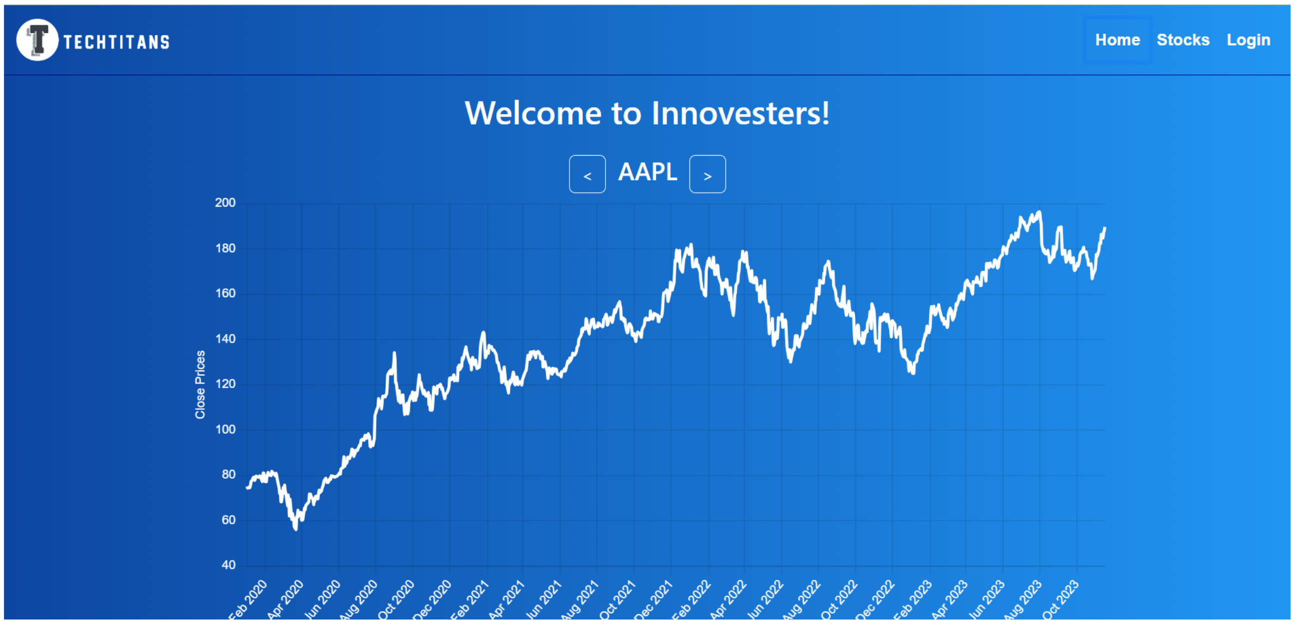
- **Assumptions**:
  - External financial data sources provide accurate and timely information.
  - Users have a basic understanding of stock market principles.
- **Dependencies**:
  - Availability of external financial data APIs.
  - Compliance with industry-standard security protocols.

3. External Interface Requirements

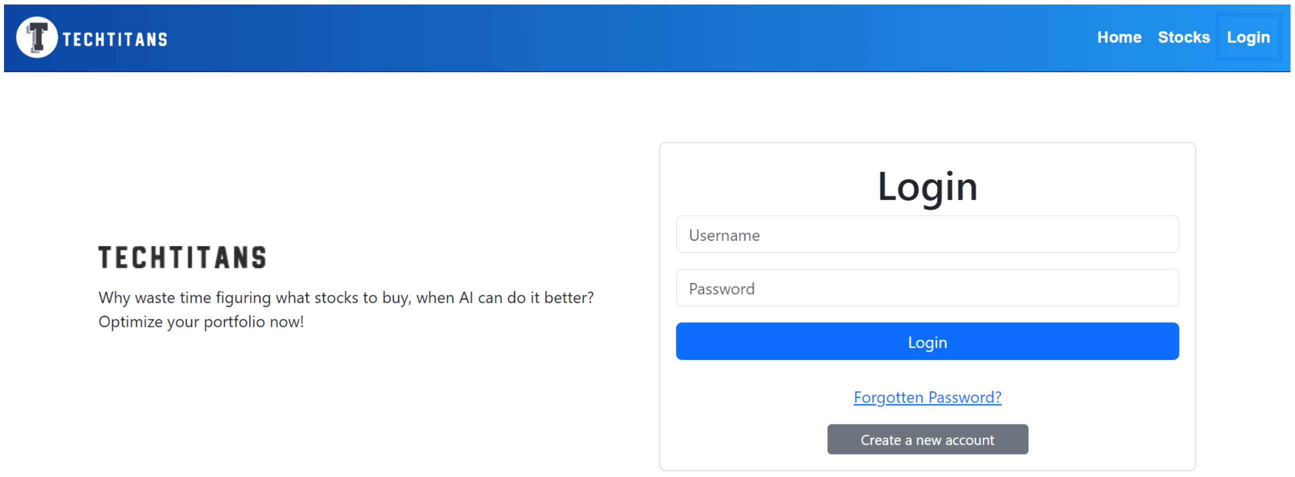
3.1 User Interfaces

This segment will provide a description of the logical characteristic of each interface. Sample screen images and user interfaces will be shown for better illustration.

A. Home Page before login



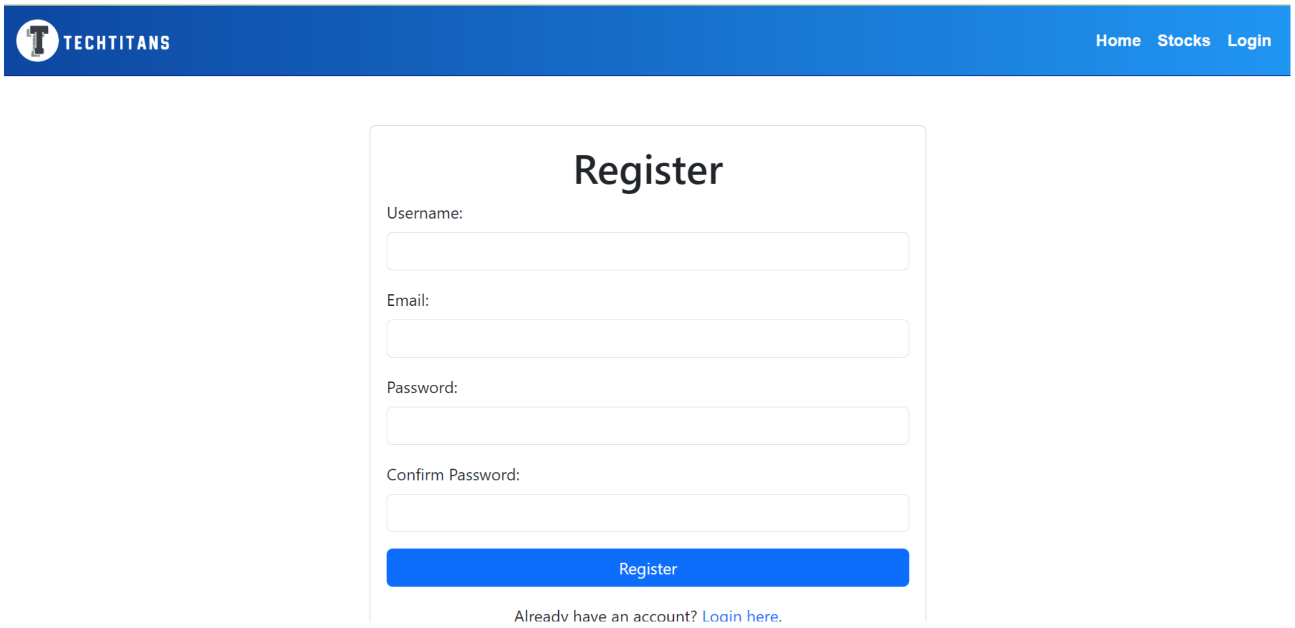
B. Login Page



TECHTITANS

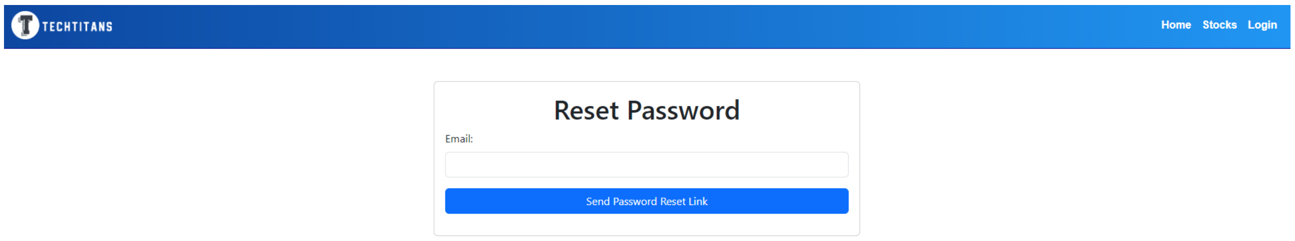
Why waste time figuring what stocks to buy, when AI can do it better? Optimize your portfolio now!

C. Register Account

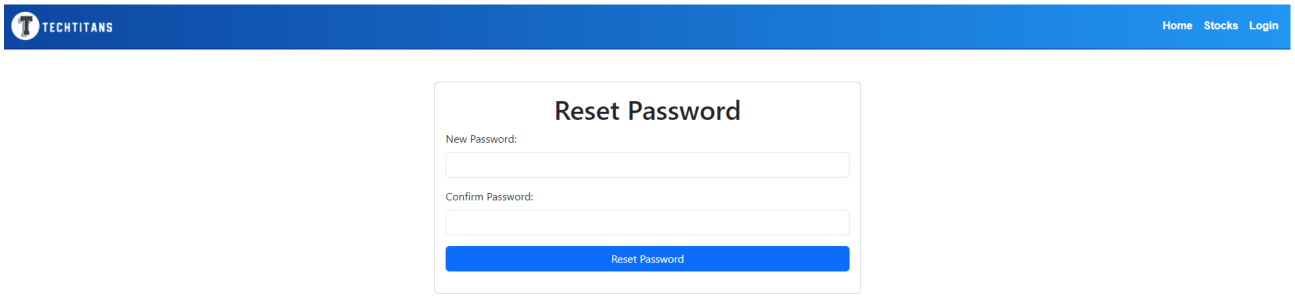


Already have an account? Login here.

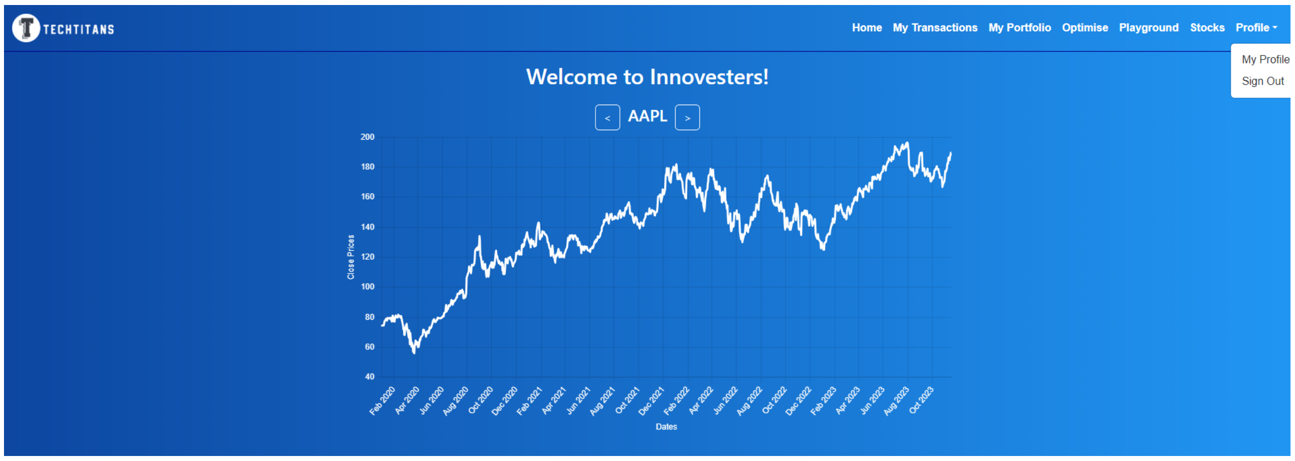
D. Send Password Reset Link Page



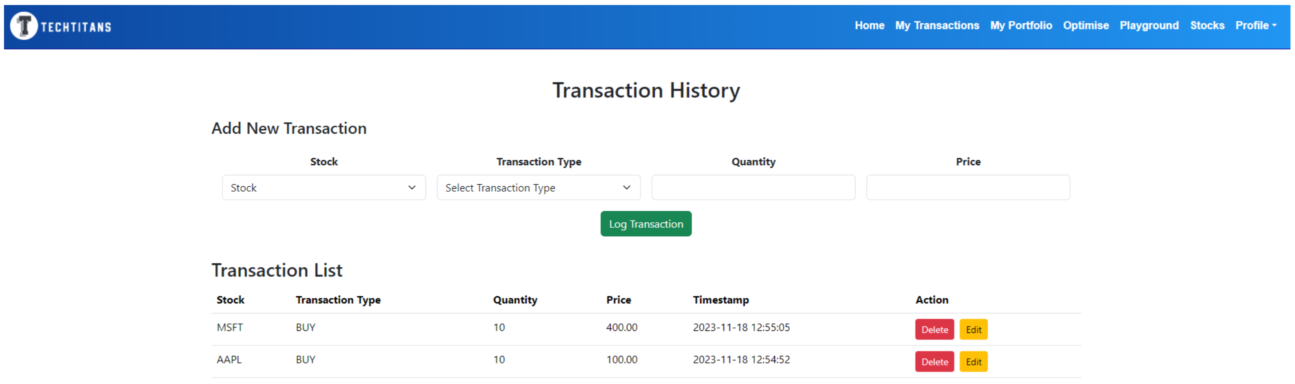
E. Reset Password Page



F. Home Page after user Logs in



G. Transaction History Page



Transaction History

Add New Transaction

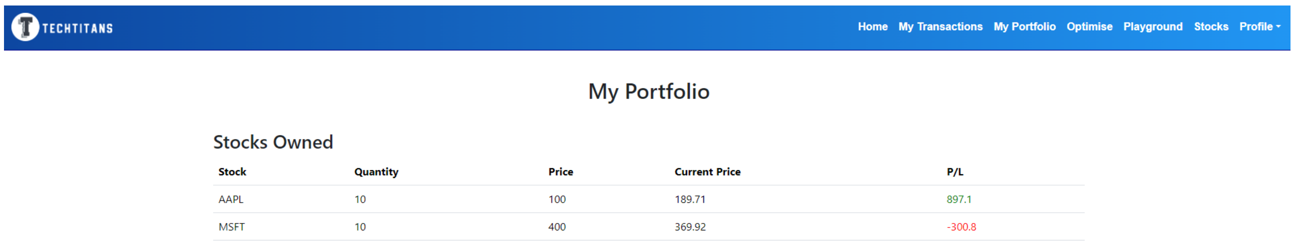
Stock	Transaction Type	Quantity	Price
Stock	Select Transaction Type		

Log Transaction

Transaction List

Stock	Transaction Type	Quantity	Price	Timestamp	Action
MSFT	BUY	10	400.00	2023-11-18 12:55:05	Delete Edit
AAPL	BUY	10	100.00	2023-11-18 12:54:52	Delete Edit

H. Portfolio Page

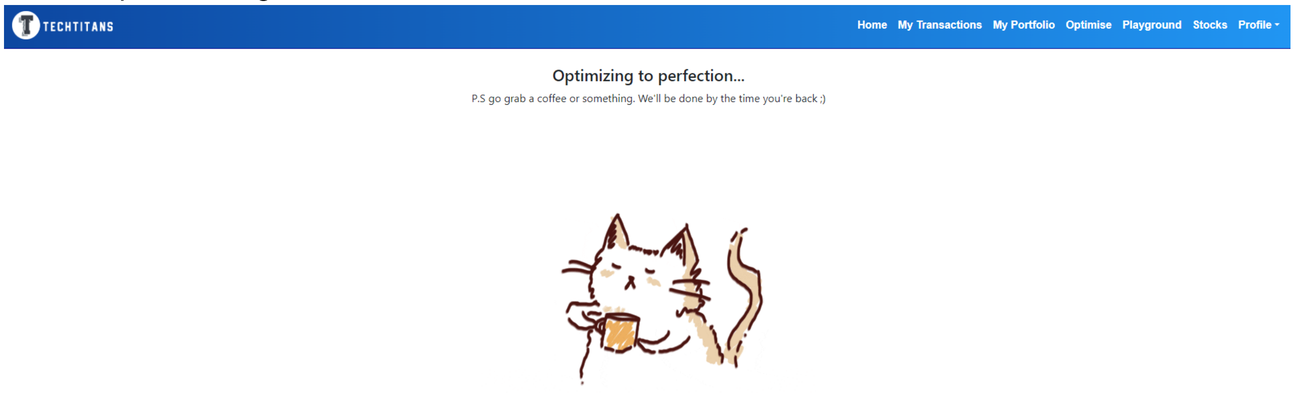


My Portfolio

Stocks Owned

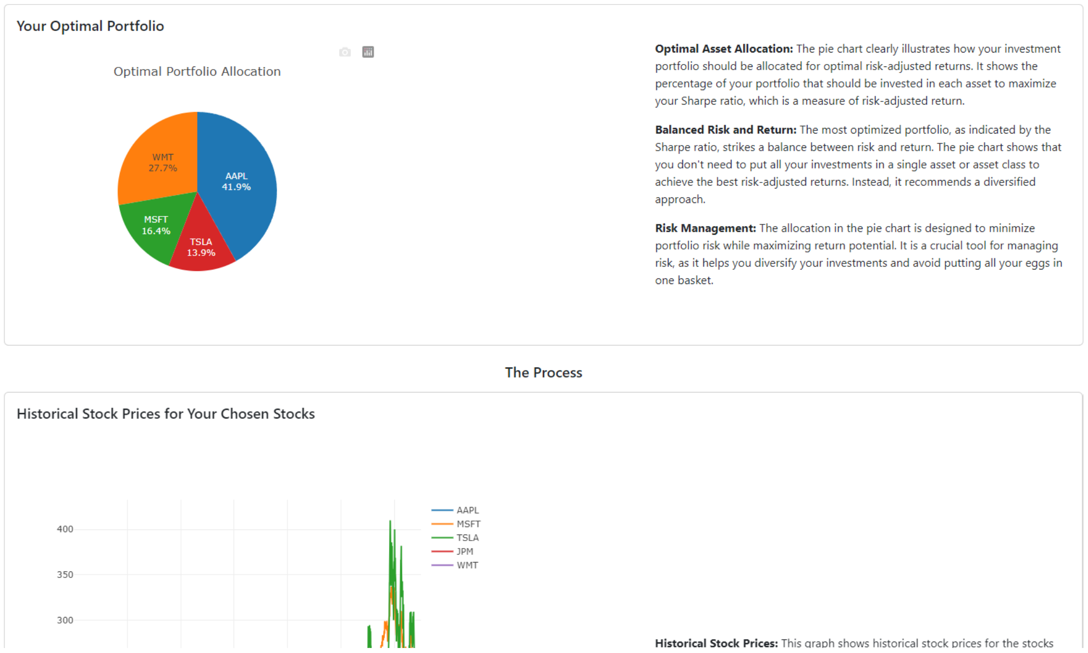
Stock	Quantity	Price	Current Price	P/L
AAPL	10	100	189.71	897.1
MSFT	10	400	369.92	-300.8

I. Optimize Page



Optimizing to perfection...

P.S go grab a coffee or something. We'll be done by the time you're back ;)



Your Optimal Portfolio

Optimal Asset Allocation: The pie chart clearly illustrates how your investment portfolio should be allocated for optimal risk-adjusted returns. It shows the percentage of your portfolio that should be invested in each asset to maximize your Sharpe ratio, which is a measure of risk-adjusted return.

Balanced Risk and Return: The most optimized portfolio, as indicated by the Sharpe ratio, strikes a balance between risk and return. The pie chart shows that you don't need to put all your investments in a single asset or asset class to achieve the best risk-adjusted returns. Instead, it recommends a diversified approach.

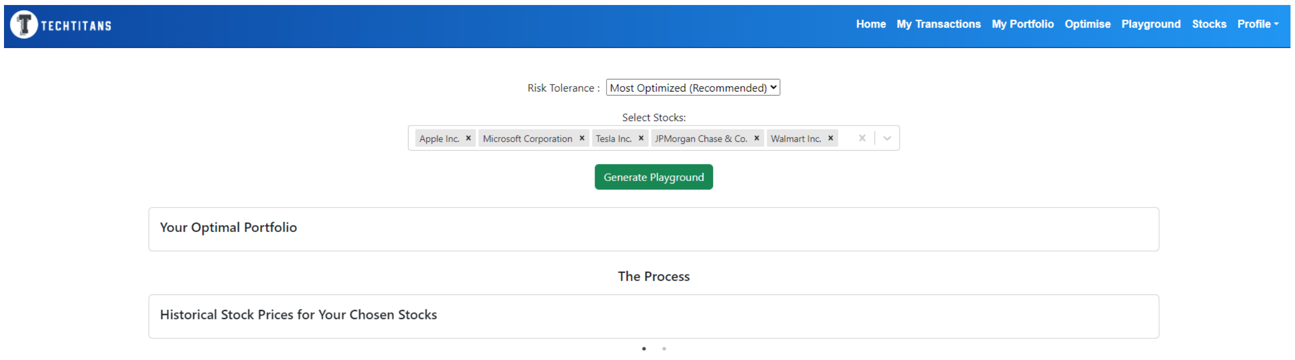
Risk Management: The allocation in the pie chart is designed to minimize portfolio risk while maximizing return potential. It is a crucial tool for managing risk, as it helps you diversify your investments and avoid putting all your eggs in one basket.

The Process

Historical Stock Prices for Your Chosen Stocks

Historical Stock Prices: This graph shows historical stock prices for the stocks

J. Playground Page



Risk Tolerance : Most Optimized (Recommended)

Select Stocks: Apple Inc. × Microsoft Corporation × Tesla Inc. × JPMorgan Chase & Co. × Walmart Inc. ×

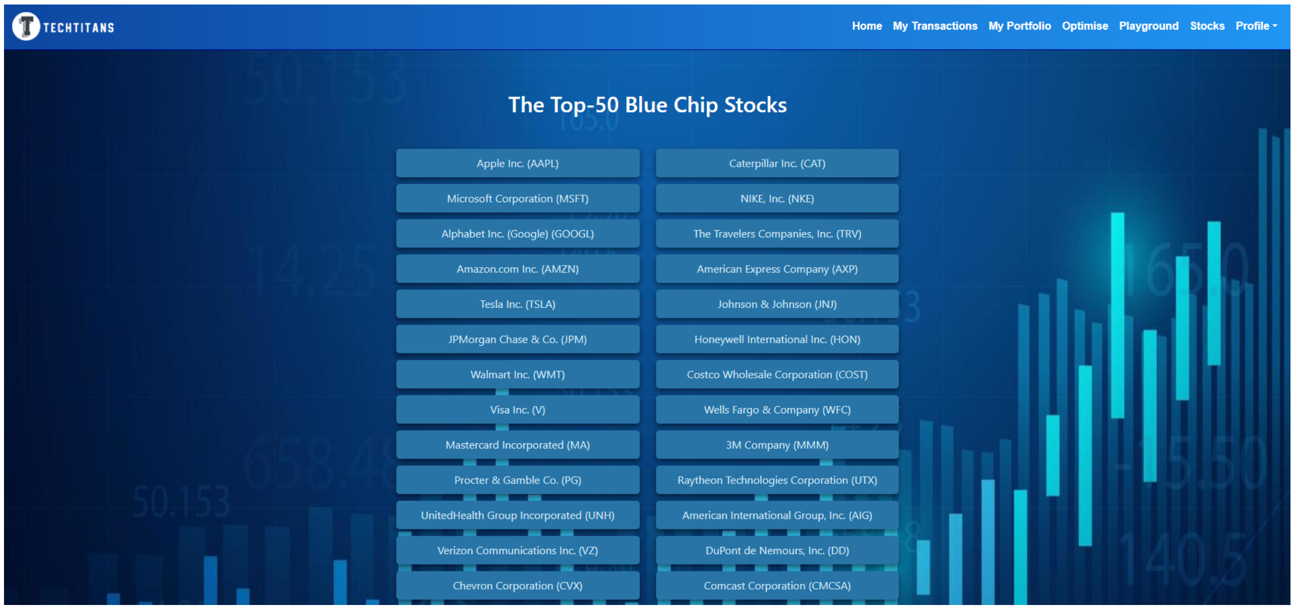
Generate Playground

Your Optimal Portfolio

The Process

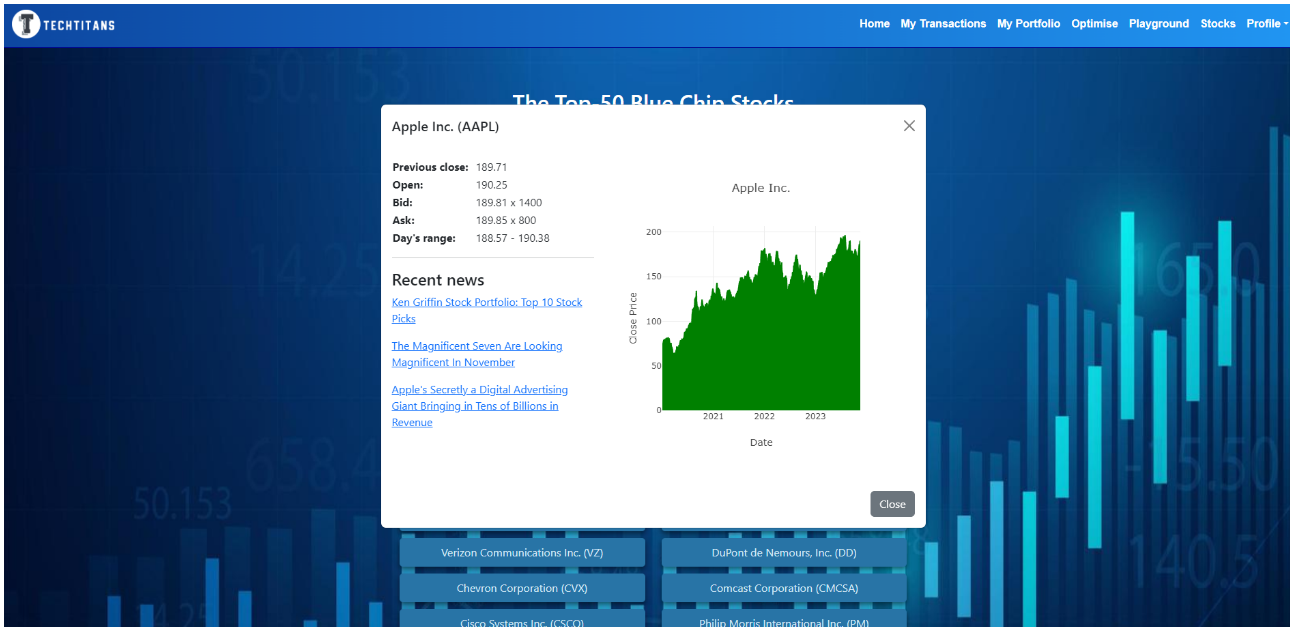
Historical Stock Prices for Your Chosen Stocks

K. Stocks Page



The Top-50 Blue Chip Stocks

Apple Inc. (AAPL)	Caterpillar Inc. (CAT)
Microsoft Corporation (MSFT)	NIKE, Inc. (NKE)
Alphabet Inc. (Google) (GOOGL)	The Travelers Companies, Inc. (TRV)
Amazon.com Inc. (AMZN)	American Express Company (AXP)
Tesla Inc. (TSLA)	Johnson & Johnson (JNJ)
JPMorgan Chase & Co. (JPM)	Honeywell International Inc. (HON)
Walmart Inc. (WMT)	Costco Wholesale Corporation (COST)
Visa Inc. (V)	Wells Fargo & Company (WFC)
Mastercard Incorporated (MA)	3M Company (MMM)
Procter & Gamble Co. (PG)	Raytheon Technologies Corporation (UTX)
UnitedHealth Group Incorporated (UNH)	American International Group, Inc. (AIG)
Verizon Communications Inc. (VZ)	DuPont de Nemours, Inc. (DD)
Chevron Corporation (CVX)	Comcast Corporation (CMCSA)



Apple Inc. (AAPL)

Previous close:	189.71
Open:	190.25
Bid:	189.81 x 1400
Ask:	189.85 x 800
Day's range:	188.57 - 190.38

Recent news

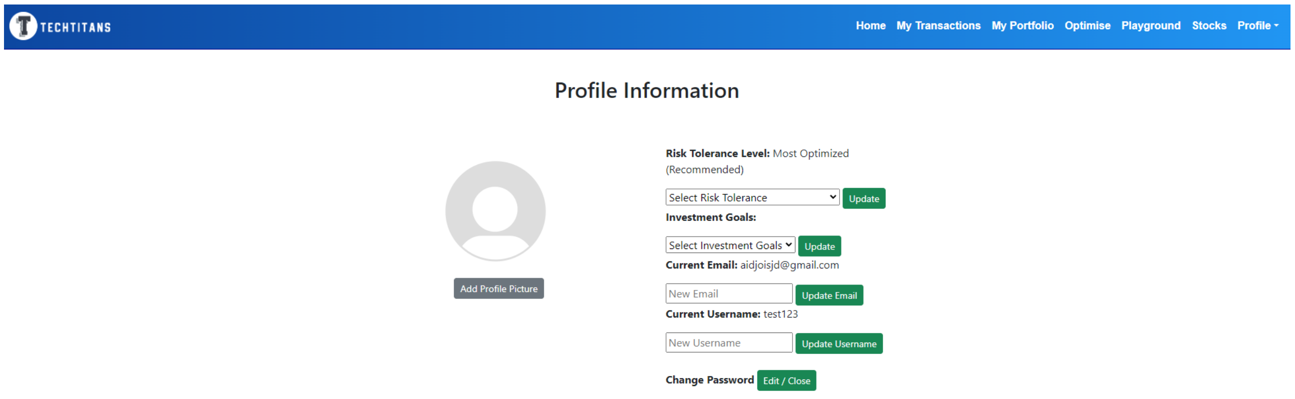
Ken Griffin Stock Portfolio: Top 10 Stock Picks

The Magnificent Seven Are Looking Magnificent In November

Apple's Secretly a Digital Advertising Giant Bringing in Tens of Billions in Revenue

Verizon Communications Inc. (VZ)	DuPont de Nemours, Inc. (DD)
Chevron Corporation (CVX)	Comcast Corporation (CMCSA)
Cisco Systems Inc. (CSCO)	Philip Morris International Inc. (PM)

L. Profile Page



Profile Information

Risk Tolerance Level: Most Optimized (Recommended)

Investment Goals:

Current Email: aidjoisjd@gmail.com

Current Username: test123

Change Password

3.2 Hardware Interfaces

This section will cover all hardware interface requirements for the Innovestors application to achieve its desired functionalities. This includes the supported device types that are needed, the nature of data and control interactions between the software and hardware as well as communication protocols being used.

A. Client-side Requirements
The Innovestors web application will support all desktop computers or laptops as well as being compatible with web services on smartphones. This device must support the usage of a desktop browser such as GoogleChrome or Firefox.

B. Server-side Requirements
The Innovestors backend server must be hosted and run on a server-computer. The backend server will perform basic functionalities such as Post, Get, Read, Update operations as well as connection using POST API with the client service on the back-end database. The PostGreSQL database must be connected and hosted on the local computer.

3.3 Software Interfaces

3.3.1 APIs Used

- Django REST Framework (DRF): Serves as a robust and adaptable toolkit within Django applications, specifically designed for constructing Web APIs with great power and flexibility.
- Backend API Endpoint: Generated via urls.py, facilitating the establishment of a connection between the frontend and backend.

- Yahoo Finance API: Utilized for gathering historical data points, enabling the creation of a chart displayed on the homepage, and optimization of a portfolio.

3.3.2 Database Used

We utilized PostgreSQL database for various functionalities related to user management, financial transactions, and system operations. The database is accessed in the following scenarios.

- User Registration: When a new user registers, their user data such as username, email, and encrypted password, is stored in the PostgreSQL database.

- User Authentication: During the login process, user credentials (username and password) are verified against the stored credentials in the PostgreSQL database.

- Token Generation: Upon successful login, an authentication token is generated and stored in the PostgreSQL database for the authenticated user as well as the sessionStorage of the webpage. This token is used for subsequent API requests to ensure secure authentication.

- User Profile Management: User profile information, including risk tolerance level, investment goals, email, username, password, and profile picture is stored in the PostgreSQL database. Any updates or modifications to the user profile are reflected in the database.

- Financial Transactions: All financial transactions, including buying, selling, and updating stock transactions, are recorded in the PostgreSQL database. The details of each transaction, such as stock, transaction type, price, and quantity, are stored for historical and analytical purposes.

- Password Change: When a user changes their password, the updated password information is stored in the PostgreSQL database.

- Email Change: If a user changes their email address, the new email information is updated in the PostgreSQL database.

- Username Change: When a user updates their username, the new username information is reflected in the PostgreSQL database.

- Password Reset Requests: Information about password reset requests, including the timestamp of the last password reset request, is stored in the PostgreSQL database.

- User Image Upload: User profile images are stored in the PostgreSQL database. When a user uploads a new profile image, the image data is saved in the database.

- Risk and Investment Updates: Changes to a user's risk tolerance level and investment goals are recorded in the PostgreSQL database.

- Optimizing Page: PostgreSQL stores information about the progress of optimization tasks, allowing the frontend to receive real-time updates on the status of ongoing tasks and results.

3.3.3 Frameworks and Libraries Used

Innovesters utilizes the following frameworks and libraries:

- Django Framework: Django is the primary web framework for building the MainApp. It provides a high-level, Python-based structure for rapid development, follows the Model-View-Controller (MVC) architectural pattern, and includes built-in features for authentication, ORM (Object-Relational Mapping), and templating.

- Django REST Framework (DRF): DRF extends Django to support building robust APIs. It provides serializers for data validation and transformation, class-based views for handling HTTP methods, and authentication classes. DRF is employed to create RESTful endpoints for communication between the frontend and backend.

- Channels: Channels is used for handling WebSocket connections and enabling real-time features in the application. It extends Django to handle protocols like WebSocket, allowing for asynchronous communication and features like live updates.

- Celery: Celery is an asynchronous task queue used for background processing. It is employed for handling tasks such as sending emails asynchronously, ensuring better performance and responsiveness.

- Redis: Redis serves as both a message broker for Celery tasks and a channel layer for Django Channels. It enhances the application's scalability and supports real-time functionality.

- PostgreSQL Database: PostgreSQL is the chosen relational database management system. It provides data persistence for user information, financial transactions, and system configurations, ensuring reliability and scalability.

- Rest Framework Token Authentication: Token Authentication from DRF is utilized for securing API endpoints. It ensures that only authenticated users can access certain resources by verifying the authenticity of tokens.

- Django CORS Headers: CORS Headers is used as middleware to handle Cross-Origin Resource Sharing (CORS) headers. It allows the frontend to make requests to the backend from a different domain securely.

- SendinBlue SMTP Service: The SendinBlue SMTP service is employed for sending email notifications and password reset emails to users. Used by Brevo.

- React Bootstrap: Enables responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

- Plotly.js Library: Utilized for rendering interactive and visually appealing plots and charts. It enhances the presentation of historical stock prices, portfolio allocations, and other graphical representations, aiding users in comprehending complex financial data.

- vaderSentiment Library: The vaderSentiment library is utilized for sentiment analysis on financial news headlines. It provides a quantitative measure of sentiment, aiding in the evaluation of the emotional tone associated with specific stocks.

- BeautifulSoup: Designed for pulling data out of HTML and XML files. It provides tools for navigating, searching, and modifying the parse tree.

- Requests: Python HTTP library that simplifies making HTTP requests. It abstracts the complexities of making requests behind a simple API, allowing easy integration of HTTP functionality into the application.

3.4 Communications Interfaces

The primary communication occurs through HTTP requests between the React frontend and Django REST Framework backend. The frontend leverages the fetch function to send asynchronous requests to various API endpoints, fetching real-time stock information and handling transaction-related functionalities. The backend, implemented in Django, responds to these requests, interacting with a PostgreSQL database to retrieve, update, or delete data based on user actions.

The message formatting adheres to REST protocol. For stock-related functionalities, RESTful communication is employed, using standard HTTP verbs such as GET and POST. The frontend sends requests to endpoints like /api-stocks/yahoo-web/.

The communication framework for the Yahoo Finance API involves the use of HTTP protocols with RESTful architecture for data exchange, employing JSON formatting for requests and responses. Security is ensured through HTTPS, encrypting data during transmission. The application optimizes data transfer rates through asynchronous operations and state management, allowing real-time updates and dynamic chart rendering based on the latest information from the Yahoo Finance API.

For web scraping, the application incorporates BeautifulSoup and Requests libraries. BeautifulSoup assists in parsing HTML and extracting relevant information, while Requests is employed for making HTTP requests to fetch data from web pages. These libraries enhance the application's capability to dynamically obtain and display relevant data, contributing to a comprehensive user experience.

Security measures include token-based authentication, where the frontend sends an authentication token along with each request to ensure secure communication with the backend. This helps protect sensitive user data, maintain user privacy and ensure only authorized users can access the API.

Data transfer rates are optimized through asynchronous operations, ensuring efficient communication between the React frontend and Django backend. Promises and async/await patterns are utilized to manage the asynchronous nature of API requests, allowing the application to update in real-time without affecting user interactions.

4. System Features

This section will include all system feature description and use cases.

4.1 Register

4.1.1 Description and Priority

Description	First time users can register for an account to create their account by simply clicking on the 'Register' button.
Priority	High
Frequency of Use	10 times a day

4.1.2 Stimulus/Response Sequences

Use Case ID:	001		
Use Case Name	Register		
Created By:	George Lai	Last Updated By:	George Lai
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	First time users can register for an account to create their account by simply clicking on the 'Register' button.
Preconditions:	1. The user must be connected to the internet 2. The user does not have an account prior to registration - Chosen email address does not exist in the database. 3. The user has navigated to the Register Interface
Postconditions:	1. The user has successfully registered an account for the application with a unique username and password and their account is added into the system database. Or 2. The user is alerted with the reason(s) why the registration of the account is unsuccessful.

Flow of Events:	1. The user enters a unique username 2. The user enters an email. Email must contain text before and after the '@'. 3. The user enters a password. Password must be at least 8 characters long 4. If the password matches the criteria, the user enters the password again to confirm it 5. If both passwords match, a new user account is created
Alternative Flows:	AF-S1: If the username already exists 1. Display message: "Username exists" 2. Return to step 1 and waits for input from user AF-S2: If an invalid email is entered 1. Display message: "Email must contain an @" 2. Return to step 2 and waits for input from user AF-S2 : If email already exists 1. Display message: "Email exists" 2. Return to step 2 and waits for input from user AF-S4: If the password is invalid 1. Display message: "Password must be at least 8 characters long and contain at least 1 number. Please enter a new password" 2. Return to step 3 and waits for input from user AF-S5: If the passwords do not match 1. Display message: "Passwords do not match. Please try again" 2. Return to step 4 and waits for input from user

4.1.3 Functional Requirements

1. Users must be able to create an account on the system.
1.1. The system must display text fields for user to enter his credentials
1.1.1. Text field must consist of username
1.1.2. Text field must consist of an password and an confirm password field

1.2. Users must fill in all of the fields before clicking on the 'Register' button

1.3. The system must verify all the fields filled in by the user before success creation of the account.
1.3.1. The system must make sure that the password and confirm password fields which the users inputted are the same.

1.3.2. The username that users entered must not exist before
1.3.3. The email that users entered must not exist before

1.4. The system must create an account for the user upon verification and store it in the database

4.2 Login

4.2.1 Description and Priority

Description	Existing users can login to their accounts by clicking on the login button.
Priority	High
Frequency of Use	20 times a day

4.2.2 Stimulus/Response Sequences

Use Case ID:	002		
Use Case Name	Login		
Created By:	Anushree Arora	Last Updated By:	Ezra Koh
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	Existing users can login to their accounts by clicking on the login button.
Preconditions:	1. The user must be connected to the internet 2. The user has an account 3. The user has navigated to the Login Interface
Postconditions:	1. The user has successfully logged into their account using their username and password. Or 2. The user is alerted with the reason(s) why the logging in of the account is unsuccessful. Or

	3. The user has successfully reset their password and logged in with the updated credentials
Flow of Events:	1. The user enters their username 2. The user enters their password 3. If the username and password is correct, the user successfully logs into the user home page
Alternative Flows:	AF-S3: If the username or password or both are not correct 1. Display message: "Invalid username and/or password. Please try again" 2. Return to step 1 OR 3. If the user wishes to reset their password. They click the 'Forgotten Password' button. 4. The user enters their email id and a reset link is sent to the email 5. The user resets their password 6. Return to step 1

4.2.3 Functional Requirements

1. Users must be able to log in to the system.
1.1. The system must display text fields for user to enter their credentials
1.1.1. Text field must consist of username and password
1.3. The system must verify all the fields filled in by the user before successful logging in
1.3.1. The system must make sure that the username is in the database and the password matches the password that corresponds to that username in the database
1.3.2. The system must display an error message if the username does not exist or the password does not match the username
1.3.3. The system must allow users to attempt to log in as many times as they want
1.4. Users must be able to reset their passwords if they have forgotten it
1.4.1. The system must display a text field for user to enter their email address
1.4.2. The system must verify that the email is valid
1.4.3. The system must send a link to the user's email that leads the user to a page where they can reset their password
1.4.4. The system must display text fields for user to enter their new password and confirm it
1.4.5. The system must verify that the password is sufficiently long and that the confirm password field matches the password field and display the respective error message if these criteria are not fulfilled
1.4.6. The system must update the user's password in the database after they enter valid inputs
1.4.7. Users must be able to log in to the system with their new password

4.3 Log Transactions

4.3.1 Description and Priority

Description	Users can log their transactions
Priority	Medium
Frequency of Use	5 times a day

4.3.2 Stimulus/Response Sequences

Use Case ID:	003		
Use Case Name	Transaction Log		
Created By:	Anushree Arora	Last Updated By:	Anushree Arora
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	Existing users can login to their accounts by clicking on the login button.
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the Transaction Log interface
Postconditions:	1. The user has successfully added a transaction Or 2. The user has edited a transaction Or 3. The user has deleted a transaction Or 4. The user can view their transaction history
Flow of Events:	1. The user selects a stock from the dropdown 2. The user selects the transaction type 3. The user enters a price 4. The user enters a quantity 5. The user clicks log transaction

Alternative Flows:	AF-S1: The user clicks on the 'edit' button 1. The user changes any of the transaction parameters 2. The user saves or cancels the changes 3. Return to step 1 AF-S1: The user clicks on the 'delete' button 1. The user can delete the transaction if it is a valid deletion 2. Return to step 1 AF-S5: The user logs an empty transaction 1. Display error message 2. Return to step 1 AF-S5: The user tries to log an invalid transaction 1. Display error message 2. Return to step 1

4.3.3 Functional Requirements

1. The system shall maintain a transaction history for each user.
2. Transactions shall be logged in an array or database, including details such as stock symbols, transaction type (buy/sell), quantities, prices, and timestamps.

4.4 View User Portfolio

4.4.1 Description and Priority

Description	Users can view their portfolio
Priority	Low
Frequency of Use	2 times a day

4.4.2 Stimulus/Response Sequences

Use Case ID:	004		
Use Case Name	View User Portfolio		
Created By:	Anushree Arora	Last Updated By:	Anushree Arora

Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	Users can view their portfolio
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the My Portfolio interface
Postconditions:	1. The user can view their portfolio
Flow of Events:	1. The user can view the various stocks they have purchased and the profit/loss that they are making on the stock.
Alternative Flows:	

4.4.3 Functional Requirements

1. The system maintains the user portfolio
2. The system integrates with the Yahoo Finance API to retrieve real-time stock data and determines the stocks profit/loss.

4.5 Optimize Portfolio

4.5.1 Description and Priority

Description	The system generates an optimized portfolio for the user
Priority	Medium
Frequency of Use	4 times a day

4.5.2 Stimulus/Response Sequences

Use Case ID:	004		
Use Case Name	Optimize Portfolio		

Created By:	Anushree Arora	Last Updated By:	Anushree Arora
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	The system generates an optimized portfolio for the user
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the Optimize interface
Postconditions:	1. The user can view their optimized portfolio and an overview of the algorithm that is working behind the scenes.
Flow of Events:	1. The user clicks on generate optimized portfolio 2. The system runs the machine learning and optimisation algorithm to generate an optimized portfolio based on the risk tolerance set in the user's profile.
Alternative Flows:	

4.5.3 Functional Requirements

1. Data Integration

    1.1 Gather and integrate financial news data from diverse sources, ensuring data relevance, and comprehensiveness.

    1.2 Acquire historical stock price data for the selected stock universe.

2. Sentiment Analysis

    2.1 Utilize machine learning models (e.g., Natural Language Processing techniques) to conduct sentiment analysis on news articles.

    2.2 Assign sentiment scores (positive, negative, neutral) to each article. Aggregate sentiment scores over a defined time period (e.g., daily, weekly).

3. Stock Selection and Ranking:

    3.1 Implement a machine learning-based ranking mechanism that considers sentiment scores along with other relevant factors (e.g., historical performance, fundamentals).

    3.2 Select a pool of potential stocks based on the ranking.

4. Efficient Frontier Optimization

    4.1 Utilize machine learning to estimate expected returns and volatility (standard deviation) for each selected stock.

    4.2 Generate a range of potential portfolios with varying asset allocations, including those that maximize returns for a given level of risk and those that minimize risk for a given level of return.

4.3 Apply machine learning optimization techniques, such as Mean-Variance Optimization, to identify the portfolio that best fits the user's risk-return preferences.

4.6 Playground Portfolio

4.6.1 Description and Priority

Description	The user select their own stocks and risk tolerance to generate an optimized portfolio
Priority	Medium
Frequency of Use	4 times a day

4.6.2 Stimulus/Response Sequences

Use Case ID:	005		
Use Case Name	Playground Portfolio		
Created By:	Anushree Arora	Last Updated By:	Anushree Arora
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	The system generates an optimized portfolio for the user
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the Playground interface
Postconditions:	1. The user can view their personalized optimized portfolio and an overview of the algorithm that is working behind the scenes.
Flow of Events:	1. The user selects their risk tolerance 2. The user selects their choice of stocks 3. The system runs the machine learning and optimisation algorithm to generate an optimized portfolio based on the risk tolerance and stocks selected.

Alternative Flows:	AF-S2: The user selects too few stocks 1. Display error message 2. Return to step 2

4.6.3 Functional Requirements

1. Data Integration

1.1 Gather and integrate financial news data from diverse sources, ensuring data relevance, and comprehensiveness.

1.2 Acquire historical stock price data for the selected stock universe.

2. Sentiment Analysis

2.1 Utilize machine learning models (e.g., Natural Language Processing techniques) to conduct sentiment analysis on news articles.

2.2 Assign sentiment scores (positive, negative, neutral) to each article. Aggregate sentiment scores over a defined time period (e.g., daily, weekly).

3. Stock Selection and Ranking:

3.1 Implement a machine learning-based ranking mechanism that considers sentiment scores along with other relevant factors (e.g., historical performance, fundamentals).

3.2 Select a pool of potential stocks based on the ranking.

4. Efficient Frontier Optimization

4.1 Utilize machine learning to estimate expected returns and volatility (standard deviation) for each selected stock.

4.2 Generate a range of potential portfolios with varying asset allocations, including those that maximize returns for a given level of risk and those that minimize risk for a given level of return.

4.3 Apply machine learning optimization techniques, such as Mean-Variance Optimization, to identify the portfolio that best fits the user's risk-return preferences.

4.7 View Stock Information

4.7.1 Description and Priority

Description	The user can view real-time information for the stocks used in the optimisation algorithm(50 blue chip stocks)
Priority	High
Frequency of Use	6 times a day

4.7.2 Stimulus/Response Sequences

Use Case ID:	007		
Use Case Name	View Stocks Information		
Created By:	Anushree Arora	Last Updated By:	Anushree Arora
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	The user can view real-time information for the stocks used in the optimisation algorithm(50 blue chip stocks)
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the Stocks interface
Postconditions:	1. The user can information for any stock they select from the list
Flow of Events:	1. The user selects a stock to view 2. The system opens a pop-up with real-time information of the stock, a chart and the most recent news headlines.
Alternative Flows:	

4.7.3 Functional Requirements

1. The system shall integrate with the Yahoo Finance API to retrieve real-time stock data.
2. Real-time stock data shall include current stock prices, financial ratios, and historical performance.
3. The system shall integrate relevant stock news feeds and make them available for user consumption.
4. The system shall display interactive stock price charts using a suitable Python charting library.

4.8 Update User Profile

4.7.1 Description and Priority

Description	The user can update their profile settings
Priority	Low
Frequency of Use	1 times a day

4.7.2 Stimulus/Response Sequences

Use Case ID:	008		
Use Case Name	Update user profile		
Created By:	Anushree Arora	Last Updated By:	Anushree Arora
Date Created:	10 November 2023	Date Last Updated:	14 November 2023

Actor:	User (Initiating Actor), Database
Description:	The user can update their profile settings
Preconditions:	1. The user must be connected to the internet 2. The user must be logged into their account 3. The user has navigated to the Profile interface
Postconditions:	1. The user has successfully updated their profile
Flow of Events:	1. The user adds the profile picture 2. The user sets their risk tolerance 3. The user sets their investment goals 4. The user updates their email 5. The user updates their username 6. The user updates their password
Alternative Flows:	

4.7.3 Functional Requirements

5. Other Nonfunctional Requirements

5.1 Usability Requirements

5.1.1. System should display prices up to 2 decimal points and in SGD currency.
5.1.2. System should use 12 hour clock to describe current time
5.1.3. System should show FAQ information in the local language according to the user's locale
5.1.4. System must be able to display all the information in the language preferred by the user
5.1.5. Users must not spend more than 5 minutes to create an account

5.2 Performance Requirements

5.2.1 The system provides responses to user interactions within a maximum acceptable time frame of 60 seconds while optimizing the portfolio.
5.2.2 The system is able to handle an increasing number of users and growing data volumes without a significant decrease in performance.
5.3.3 The system is able to support concurrent users without degrading performance.
5.4.4 The system is able to provide meaningful error messages and handle errors efficiently.

5.3 Safety Requirements

5.3.1 The system implements robust encryption mechanisms to secure sensitive user data, including financial information, login credentials, and personal details.
5.3.2 The system ensures the security of user accounts through robust authentication measures such as sending email notifications to allow password change.

5.4 Security Requirements

5.4.1 The system must not disclose the user's email address to other users
5.4.2 Other users must not be able to view other user's password, email addresses.
5.4.3 Other users must not be able to change other user's passwords and email addresses.
5.4.4 Other users must not be able to view user's stock - investments and stock history.
5.4.5 The system should not be disclosing the user's email address and password for any personal or commercial use.

5.5 Software Quality Attributes

5.5.1 Reliability - The system should consistently perform portfolio optimizations and provide accurate stock advice without unexpected failures. This will be measured using Mean Time

Between Failures (MTBF) of at least X hours. The verification will be conducted by stress testing to simulate heavy user loads and measure system stability.
5.5.2 Portability - The ability of the software to operate seamlessly across different devices and platforms would be measured by calculating the percentage of successful installations on different operating systems. The software has shown high portability after verifying this through testing the software on various operating systems and devices to ensure compatibility.
5.5.3 Robustness- The software's ability to gracefully handle unexpected inputs or conditions without crashing was measured using the percentage of system uptime during adverse conditions. After conducting testing with invalid inputs and simulating unexpected events, the software was deemed to be robust.
5.4.4 Interoperability -The software is able to interact seamlessly with other systems and financial data sources such as yfinance API and web scraping done on sites such as reuters, bloomberg.
5.5.5 Adaptability- The software is able to adapt to user requirements by giving them the freedom to choose stocks of their preferences, to choose risk that they are comfortable with , and allows them to modify stocks and user details as per preference.

5.6 Business Rules

5.6.1 The system will have different user roles, including regular users and administrators. Administrators will have the power to do regular maintenance for constant checking and removing spam user accounts.
5.6.2 Regular users can access and modify their own portfolios. The optimization of portfolios is automated based on user preferences and system algorithms. Administrators will play no part in handling user accounts and/or adjusting user stock choices.

6. Other Requirements

6.1 Comprehensive documentation shall be provided for system users, administrators, and developers.Comprehensive instructions provided for every decision a user can make in the interface.
6.2 The system utilizes a relational database to store user profiles, portfolio data, transaction history, and other relevant information.
6.3 The system maintains an audit trail to track and log significant user actions, changes to portfolios, and system events as per legal requirements.

Appendix A: Glossary

This segment will define all the terms, acronyms, abbreviations that are written in this SRS document.

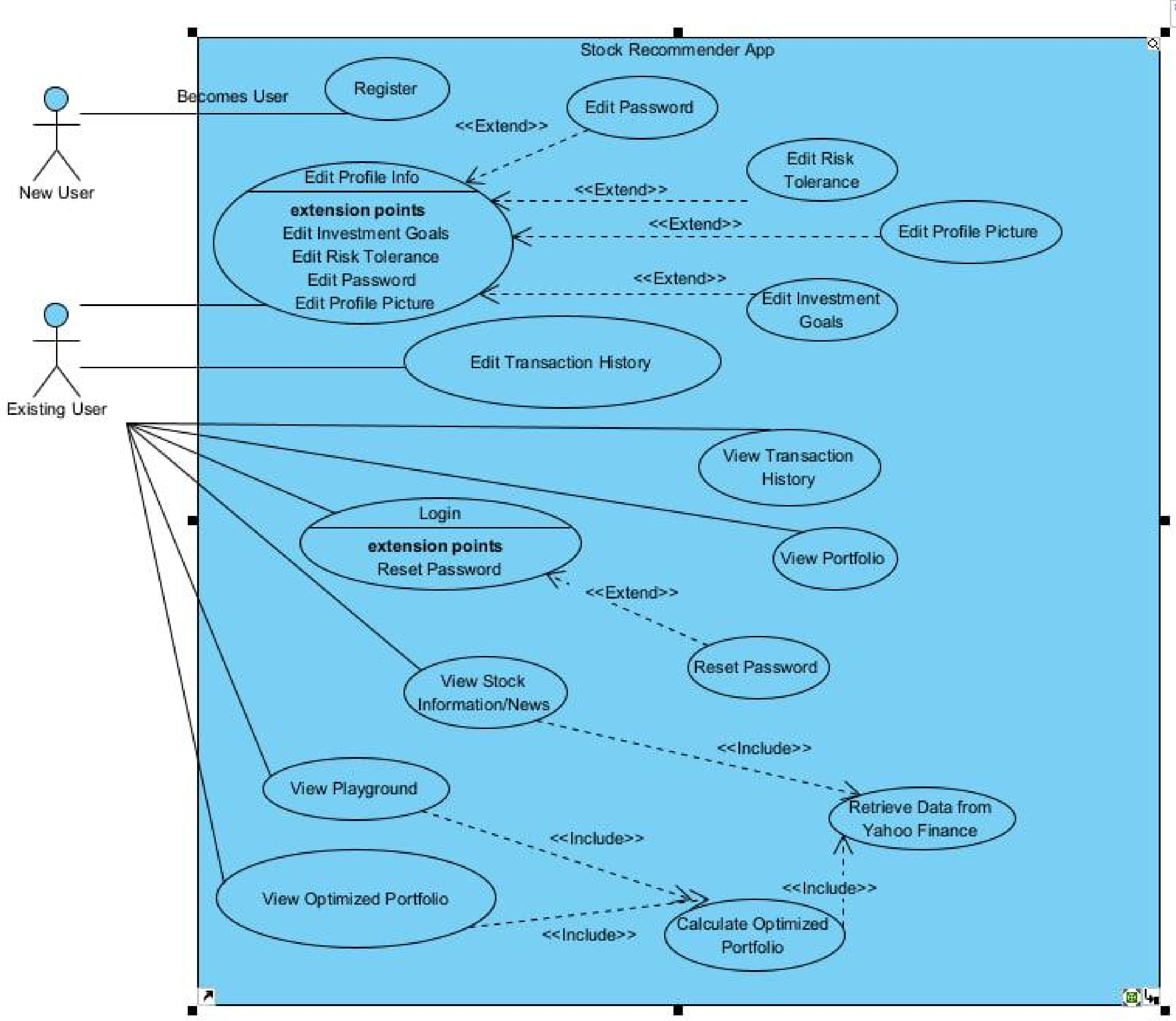
Created by	George Lai
Date created	17 November 2023
API	API stands for Application Programming Interface, which is a set of protocols for two or more computer programs to communicate with each other
User account	A means by which app users can get access to the platform. It includes a username as an identification set by user and an registered email address or contact number where the platform will send OTP for users to log in. The users can also log in with the password they set when registering.
Database	An organized collection of structured data, in this case, username, password, registered email, user portfolio risk preference, investment goals, profile picture, and transaction history.
Sharpe Ratio	A widely used method in measuring risk-adjusted relative returns
Sentiment Analysis	Machine learning algorithm to determine investors' opinions of a specific stock or asset. Sentiment may at times hint at future price action. This is also an example of how trading psychology can affect a market, assisting as a forecasting tool to determine possible future price changes in a particular asset.

Appendix B: Analysis Models

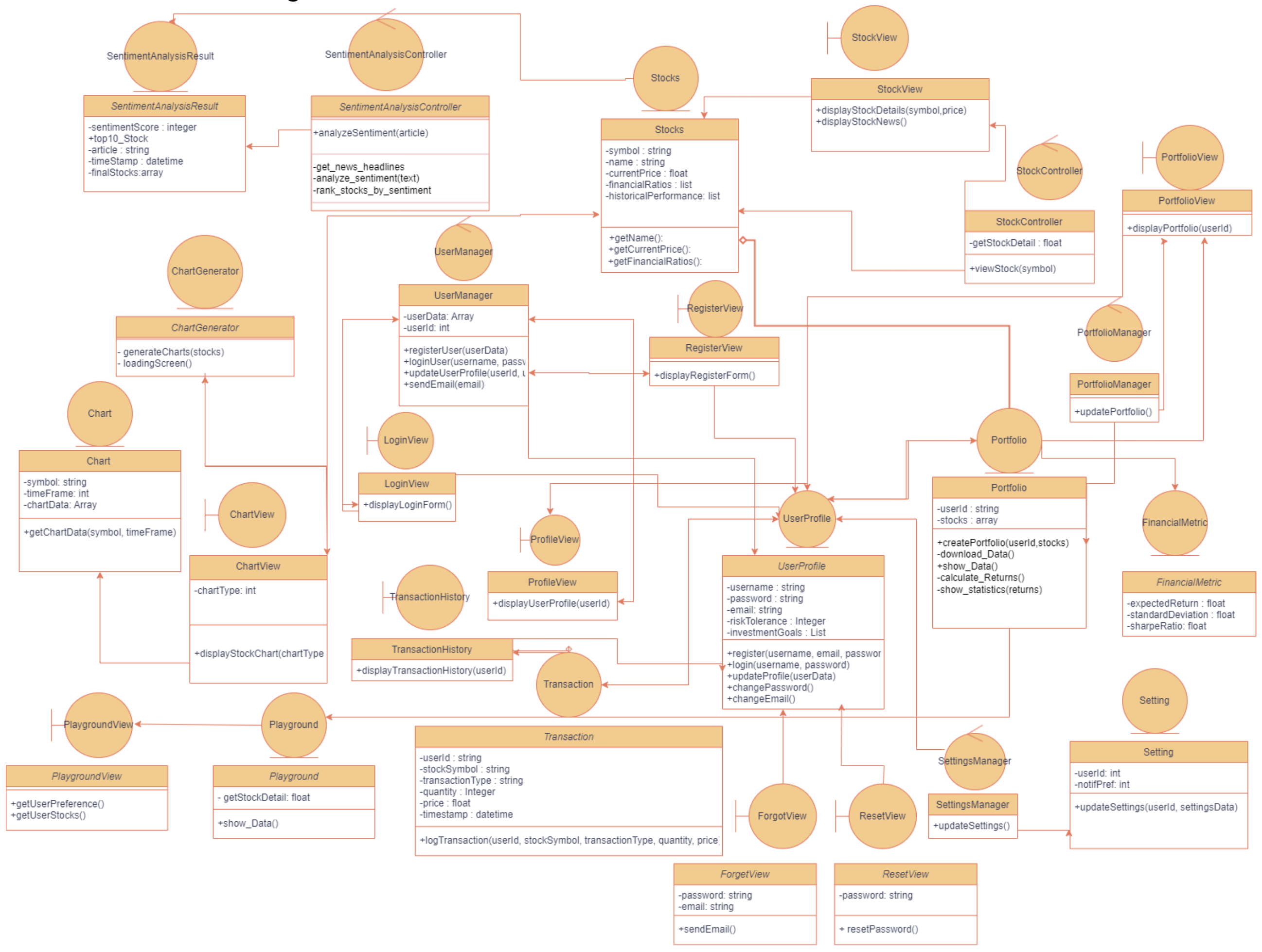
<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

This segment will consist of all analysis models. Clearer views of the pictures are attached in the folder.

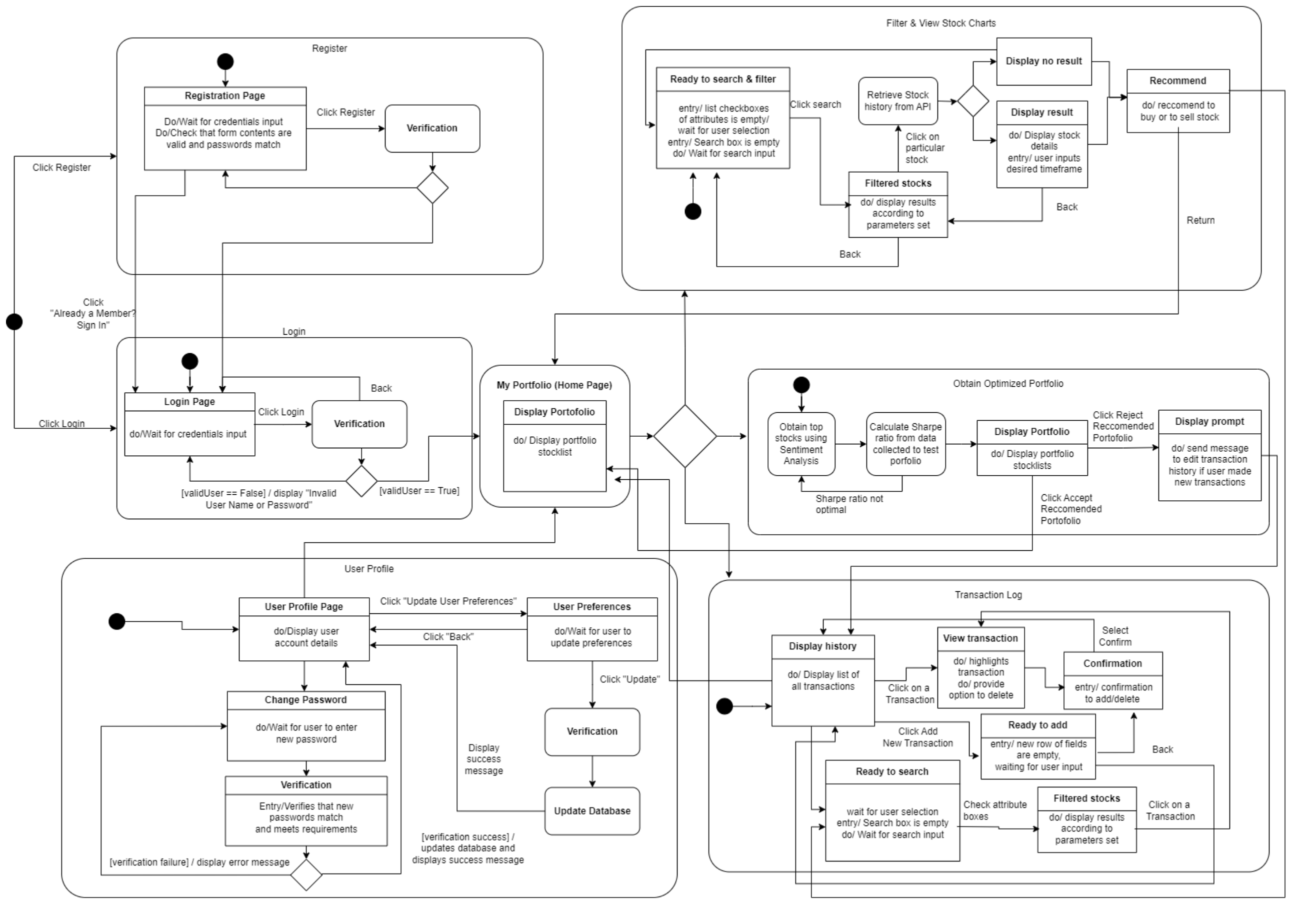
- Use Case Diagram



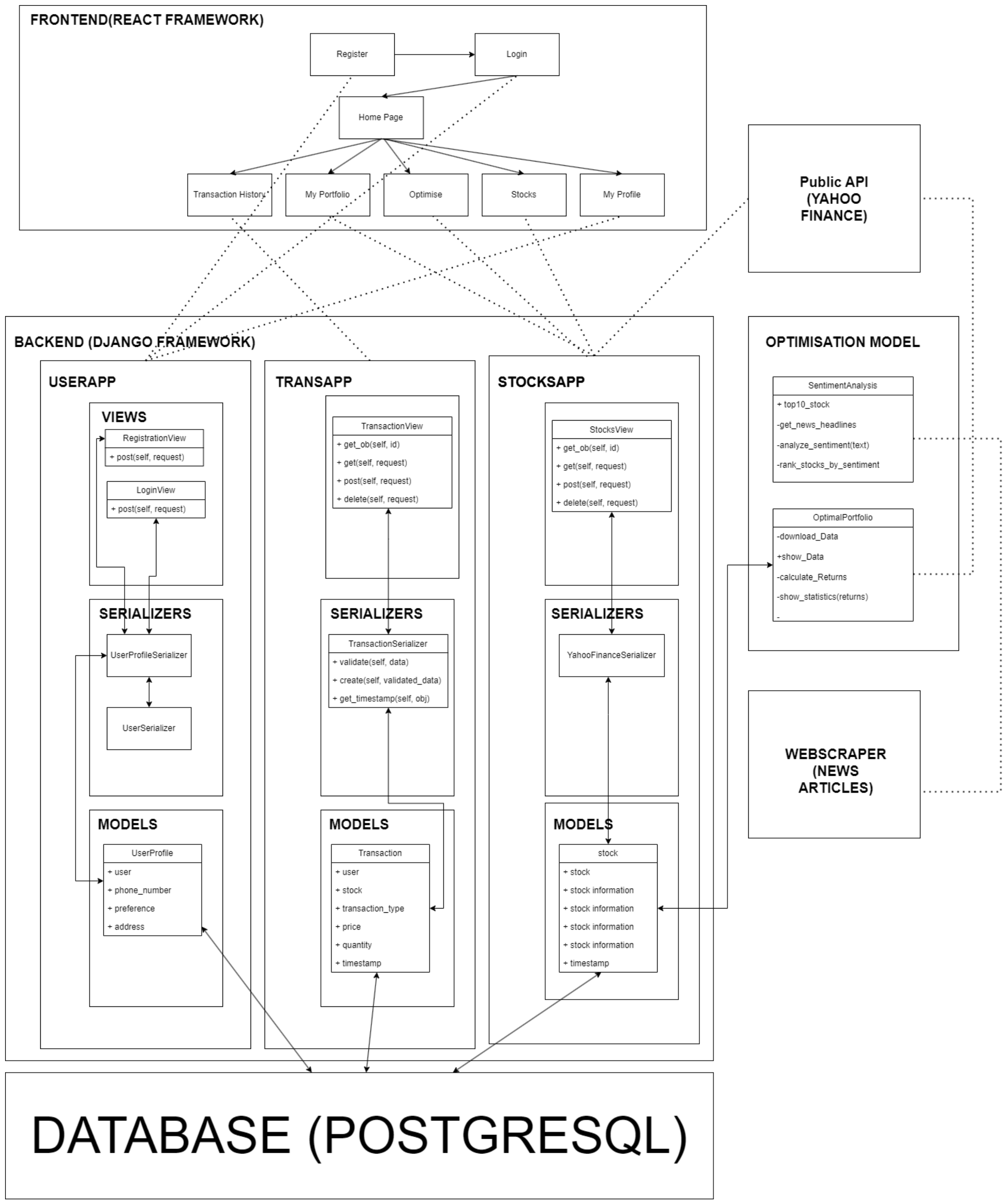
- UML Class Diagram



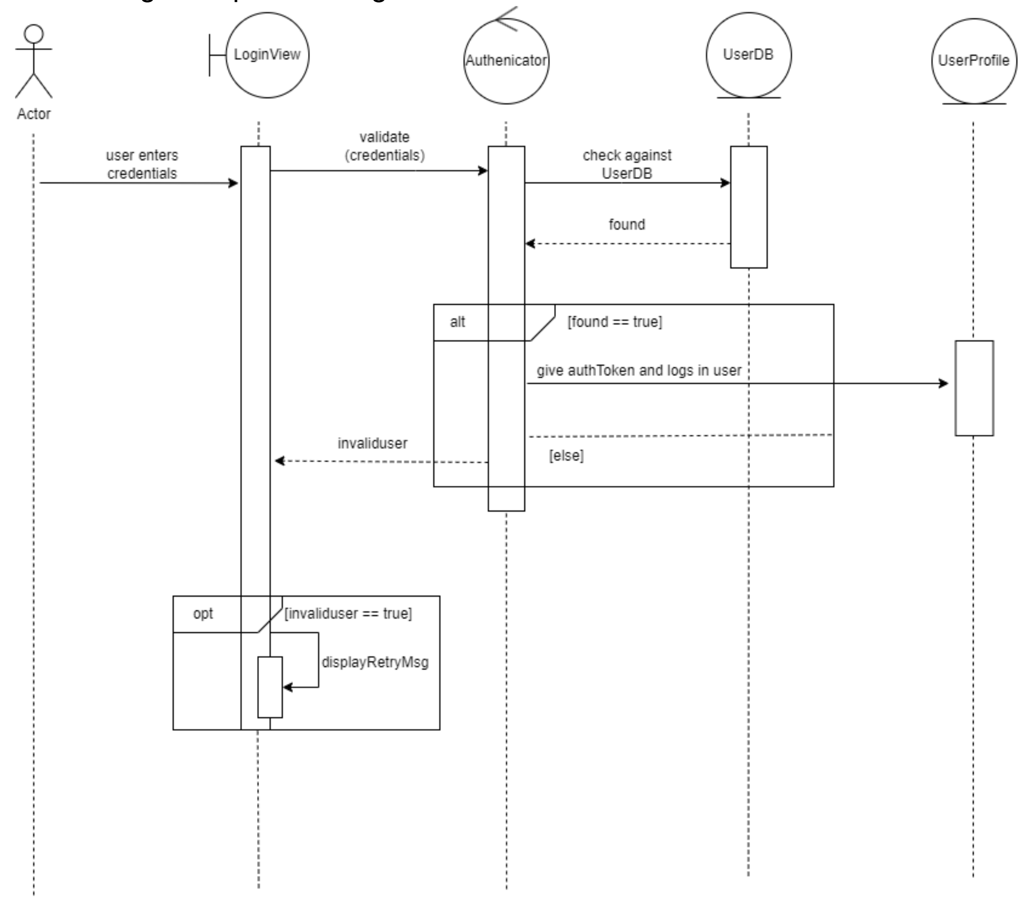
- Dialog Map (State Machine Diagram)


Register
Registration Page
Do/Wait for credentials input
Do/Check that form contents are valid and passwords match
Click Register
Verification
Click Register
Click "Already a Member? Sign In"
Click Login
Login
Login Page
do/Wait for credentials input
Click Login
Verification
Back
[validUser == False] / display "Invalid User Name or Password"
[validUser == True]
Filter & View Stock Charts
Ready to search & filter
entry/ list checkboxes of attributes is empty/ wait for user selection
entry/ Search box is empty
do/ Wait for search input
Click search
Retrieve Stock history from API
Click on particular stock
Filtered stocks
do/ display results according to parameters set
Back
Display no result
Display result
do/ Display stock details
entry/ user inputs desired timeframe
Back
Recommend
do/ reccomend to buy or to sell stock
Return
My Portfolio (Home Page)
Display Portofolio
do/ Display portfolio stocklist
Obtain Optimized Portfolio
Obtain top stocks using Sentiment Analysis
Calculate Sharpe ratio from data collected to test porfolio
Sharpe ratio not optimal
Display Portfolio
do/ Display portfolio stocklists
Click Reject Reccomended Portofolio
Click Accept Reccomended Portofolio
Display prompt
do/ send message to edit transaction history if user made new transactions
User Profile
User Profile Page
do/Display user account details
Click "Update User Preferences"
Click "Back"
User Preferences
do/Wait for user to update preferences
Click "Update"
Verification
Update Database
Display success message
[verification success] / updates database and displays success message
Change Password
do/Wait for user to enter new password
Verification
Entry/Verifies that new passwords match and meets requirements
[verification failure] / display error message
Transaction Log
Display history
do/ Display list of all transactions
Click on a Transaction
View transaction
do/ highlights transaction
do/ provide option to delete
Select Confirm
Confirmation
entry/ confirmation to add/delete
Click Add New Transaction
Ready to add
entry/ new row of fields are empty, waiting for user input
Back
Ready to search
wait for user selection
entry/ Search box is empty
do/ Wait for search input
Check attribute boxes
Filtered stocks
do/ display results according to parameters set
Click on a Transaction

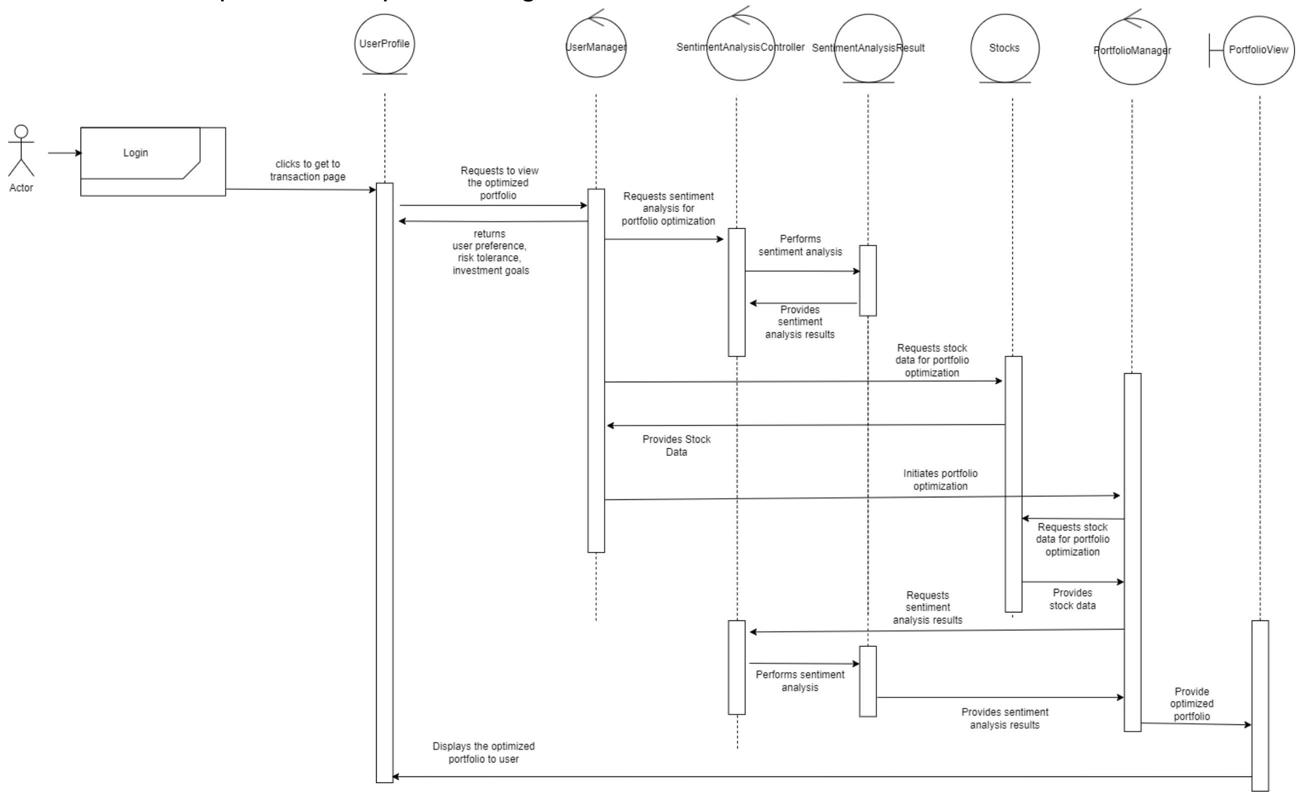
- Software Architecture



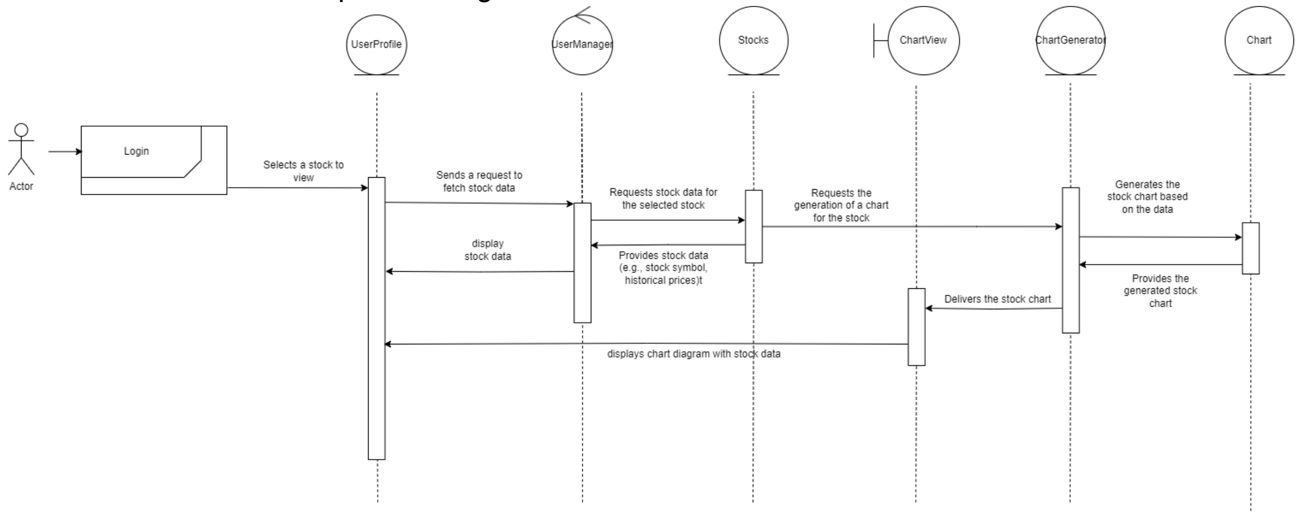
- UML Sequence Diagrams
  - Login Sequence Diagram



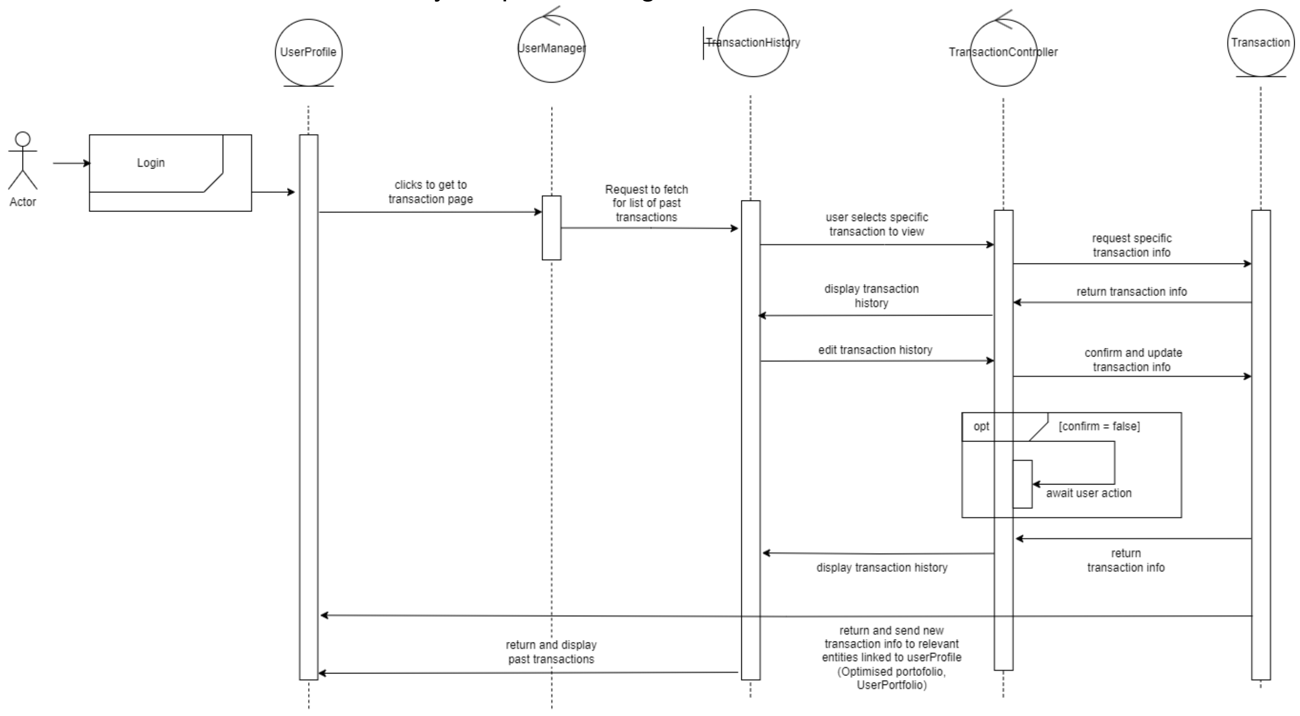
- Optimized Sequence Diagram


UserProfile
UserManager
SentimentAnalysisController
SentimentAnalysisResult
Stocks
PortfolioManager
PortfolioView
Actor
Login
clicks to get to transaction page
Requests to view the optimized portfolio
returns user preference, risk tolerance, investment goals
Requests sentiment analysis for portfolio optimization
Performs sentiment analysis
Provides sentiment analysis results
Requests stock data for portfolio optimization
Provides Stock Data
Initiates portfolio optimization
Requests stock data for portfolio optimization
Provides stock data
Requests sentiment analysis results
Performs sentiment analysis
Provides sentiment analysis results
Provide optimized portfolio
Displays the optimized portfolio to user

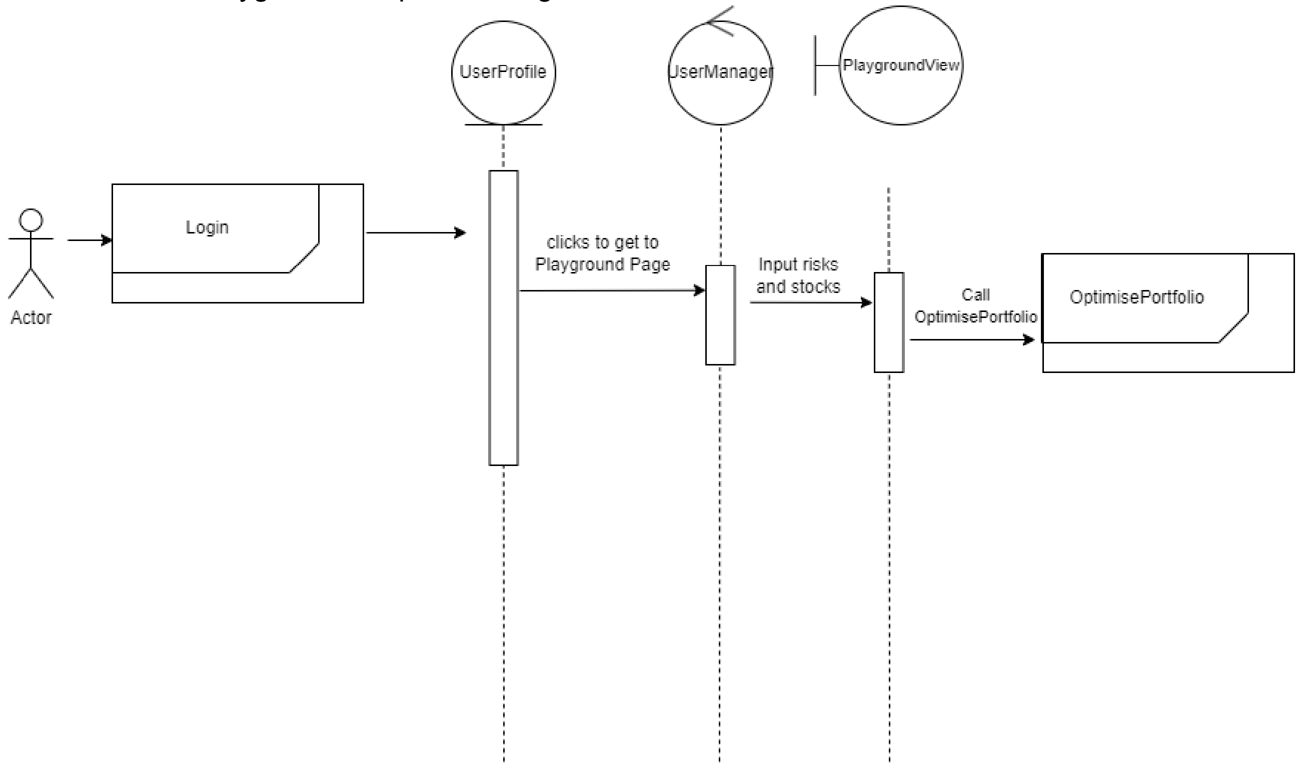
- Stocks Sequence Diagram


UserProfile
UserManager
Stocks
ChartView
ChartGenerator
Chart
Actor
Login
Selects a stock to view
Sends a request to fetch stock data
Requests stock data for the selected stock
Requests the generation of a chart for the stock
Generates the stock chart based on the data
display stock data
Provides stock data (e.g., stock symbol, historical prices)t
Provides the generated stock chart
Delivers the stock chart
displays chart diagram with stock data

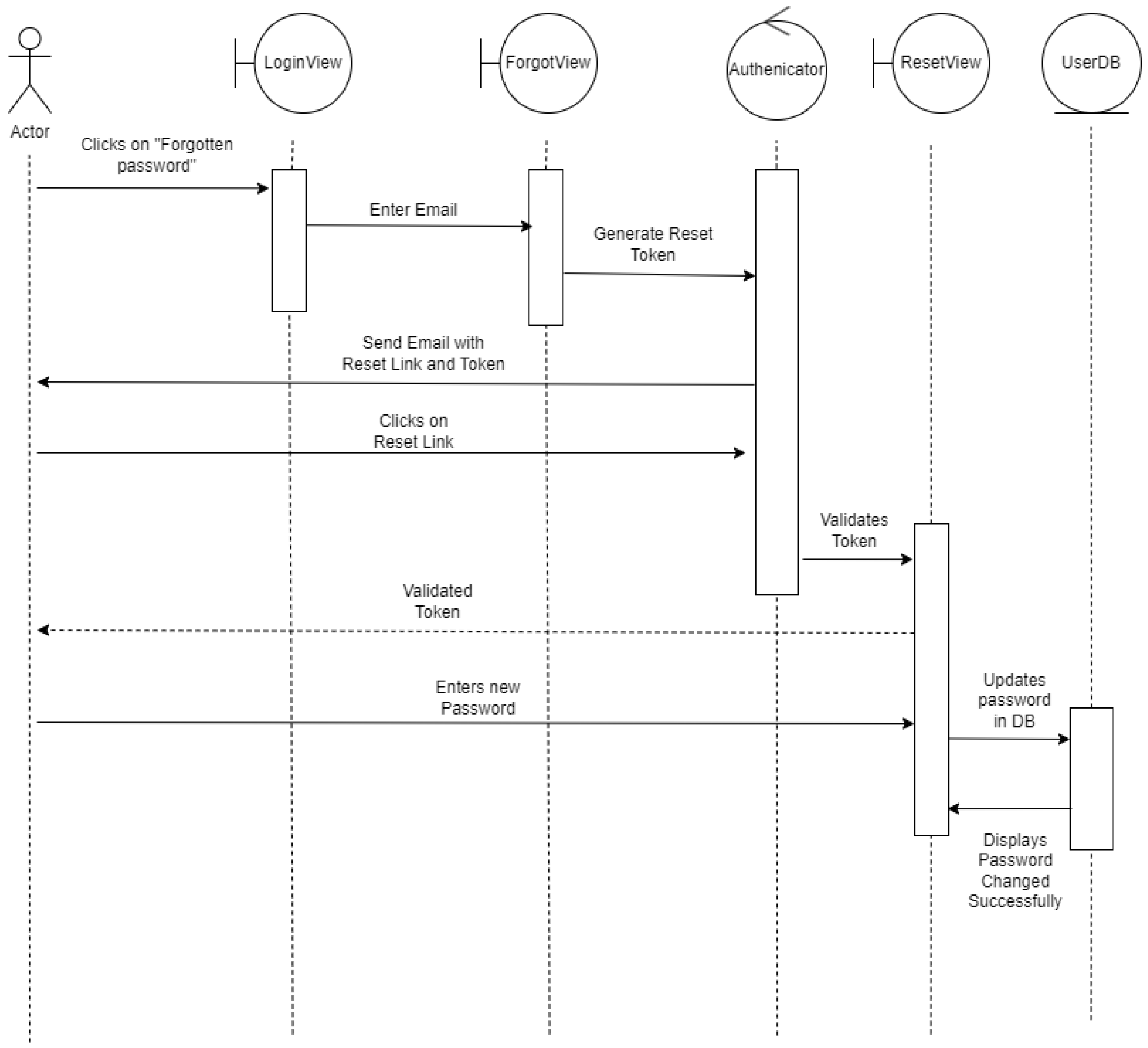
- Transaction History Sequence Diagram


UserProfile
UserManager
TransactionHistory
TransactionController
Transaction
Actor
Login
clicks to get to transaction page
Request to fetch for list of past transactions
user selects specific transaction to view
request specific transaction info
return transaction info
display transaction history
edit transaction history
confirm and update transaction info
opt
[confirm = false]
await user action
return transaction info
display transaction history
return and send new transaction info to relevant entities linked to userProfile (Optimised portofolio, UserPortfolio)
return and display past transactions

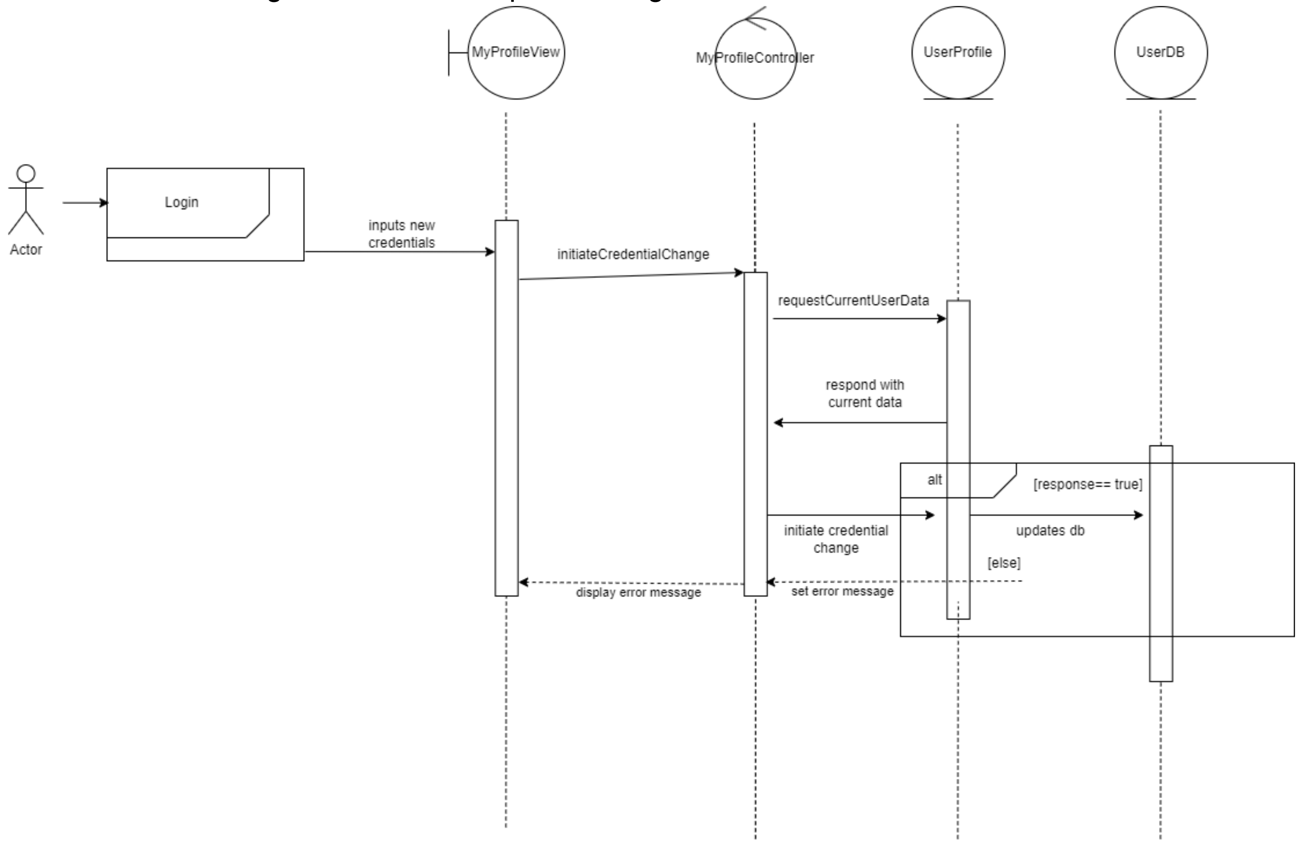
- Playground Sequence Diagram



- Reset Password Sequence Diagram



- Change Credentials Sequence Diagram



Appendix C: User Manual

Welcome to Innovesters - the essential stock analysis app that quickly creates an optimized portfolio that is personalized for you. Here's a step-by-step guide on how to use the app:

<u>Step 1: Open the Application</u>
To access Innovesters, open up any browser you wish and enter the web url "http://172.21.148.171:3000".

<u>Step 2: Login or Register</u>
If you don't have an account, you can quickly register a new one to start using the website.

<u>Step 3: Input Transaction History</u>
Once you're logged in, you can input the trades you have previously made in the relevant fields. You can also edit and delete these trade logs as needed.

<u>Step 4: Update the Risk Tolerance Level</u>
Now head to the profile page, where you can customize the Risk Tolerance Level and Investment Goals you wish. As a Novice Trader I recommend you to stick to "Most Optimized" Risk Tolerance Level.

<u>Step 5: Optimize your portfolio</u>
Now, we could head over to our Optimize page to Generate our portfolio, after clicking on it, a loading screen would appear. Once the optimized portfolio is generated, a pop up notification would appear, and the most optimized portfolio would appear for you. Using this portfolio, use your judgment and invest wisely.

That's all, these are just some simple steps for a novice trader, you could play around with the settings if you are experienced, but please do it responsibly. We hope that you will enjoy using Innovesters!