Software Requirements Specification (SRS)

Stock Exchange Simulator

Version 1.0

Rohaan (CS-19091)

Umer (CS-19095)

Sohaib (CS-19096)

Talha (CS-19097)

Department of Computer and Information Systems
NED University of Engineering and Technology

Submitted to
Mr. Kashif Asrar

December 6, 2021

Contents

1 Introduction **4**
1.1 Document Purpose 4
1.2 Product Scope 4
1.3 Intended Audience and Document overview 4
1.4 Definitions, Acronyms and Abbreviations 5
1.5 Document Conventions 6
1.6 References and Acknowledgments 6

2 Overall Description **7**
2.1 Product Perspective 7
2.2 Product Functionality 7
2.3 Users and Characteristics 8
2.3.1 End User 8
2.4 Operating Environment 8
2.5 Design and Implementation Constraints 8
2.5.1 API usage limit 8
2.5.2 Firebase Constraints 9
2.5.3 Internet Connectivity 9
2.5.4 Hardware Constraints 9
2.5.5 OS Constraint (Windows only) 9
2.6 User Documentation 9
2.7 Assumptions and Dependencies 9
2.7.1 Alphavantage API 9
2.7.2 Firebase 10
2.7.3 Computer 10

3 Specific Requirements **11**
3.1 External Interface Requirements 11
3.1.1 User Interfaces 11
3.1.2 Hardware Interfaces 15
3.1.3 Software Interfaces 15

3.1.4 Communications Interfaces 16
3.2 Functional Requirements 16
3.2.1 General 16
3.2.2 Sign Up 16
3.2.3 Sign In 16
3.2.4 Home Screen (dashboard showing portfolio and tracked stocks) 17
3.3 Behaviour Requirements 17
3.3.1 Use Case View 17

4 Other Non-Functional Requirements 19
4.1 Performance Requirements 19
4.1.1 Application start-up time 19
4.1.2 Switching pages 19
4.1.3 Authentication 19
4.1.4 Stock price quote 19
4.1.5 Buy/Sell Stocks Transactions 19
4.2 Safety and Security Requirements 20
4.2.1 Safety Requirements 20
4.2.2 Security Requirements 20
4.3 Software Quality Attributes 20
4.3.1 Readability 20
4.3.2 Maintainability 20
4.3.3 Adaptability 21
4.3.4 Reuseability 21
4.3.5 Robustness 21
4.3.6 Usability 21

Chapter 1

Introduction

Stock Exchange Simulator is a desktop application for simulating a real stock exchange. It will allow its users to "buy" and "sell" stocks at real time prices. This section gives an overview of the project and this SRS document.

1.1 Document Purpose

The purpose of this SRS is to fully specify all the requirements of the first release of Stock Exchange Simulator (version 1.0) for all stake-holders. The document covers software requirements, hardware requirements, user-interface descriptions, program flows, functional requirements and non-functional requirements.

1.2 Product Scope

This intention of this project is to simulate a stock exchange environment for users to learn the basics of stock exchange. This learning process would be simple, stress free, and enjoyable in a gaming environment. This environment allows players to hone their skills through competition with other players using virtual money to buy and sell stocks based on real stock market data. Through experience, players will gain confidence in their investing abilities. We hope to make the difficult task of learning to invest in a high risk, stock exchange market as an enjoyable experience.

1.3 Intended Audience and Document overview

This SRS is intended for the following audiences:

- **Main Development Team**: The development team will be writing, testing, and deploying the code. The document aims to detailed and unambiguous in its description about the project for the ease of development.

- **Other Developers**: As this project is going to be open source and anyone can contribute to its code, this document also serves purpose of explaining the requirements of the project to developers other than the ones in the main development team so that they can understand the project's functionalities and contribute easily.

- **End Users**: The end users of this application are the people for whom this application is intended for and who will use this application for learning about stock exchange and improve their stock trading skills. The document can be read by the end users to gain in depth insight on how this application works

1.4 Definitions, Acronyms and Abbreviations

- **API**: **A**pplication **P**rogramming **I**terface - A software intermediary that allows two applications to talk to each other. (for example our application will communicate with Alpha Vantage API to get stocks related information)

- **CSS: C**ascading **S**tyle **S**heets - A language for styling user-interfaces made with HTML.

- **Electron JS**: A JavaScript tool to make desktop applications.

- **Firebase:** A platform/service developed by Google for creating mobile and web applications. Includes various services and features like authentication, database, analytics, and more.

- **Firestore:** A cloud database service included in Firebase

- **Framework / Library (in context of programming languages):** Collection of code that perform some often-required functionality, that can be used by anyone.

- **HTML**: **H**yper **T**ext **M**arkup **L**anguage - A language for making structure of user-interfaces

- **JavaScript (JS for short)**: A programming language, for including logic, interactivity, etc.
- **Node JS**: A JavaScript tool to execute JavaScript outside
- **SRS: S**oftware **R**equirements **S**pecification web-browsers.

1.5 Document Conventions

This document is made using LaTeX, a software system for document prepration. It uses Arial font size 12 pt and 1" margins.

1.6 References and Acknowledgments

- Alpha Vantage API - https://www.alphavantage.co/
- Firebase - https://firebase.google.com/

Chapter 2

Overall Description

2.1 Product Perspective

Stock Exchange Simulator is an application for users to learn about stock exchanging without risking real money. This document covers the requirements for the first version (Version 1.0) of the application.

2.2 Product Functionality

Following are the major functionalities of this project:

- Users will be able to sign up for a new account.
- Users will be able to sign in using an existing account.
- Users will be able to view their portfolio including all their stock holdings (and details therof) and cash.
- Users will be able to view details of the stocks they have selected to be tracked.
- Users will be able to see history of their trading.
- Users will be able to find out price of any stock by inputting the stock's symbol or company name.
- Users will be able to "buy" any amount of any stock they want (provided they have enough cash).
- Users will be able to "sell" any of their stocks.

- Users will be able to view a leaderboard of all users of the application. (leaderboard ranks users by their total value)

2.3 Users and Characteristics

This application only has one category of user: an end user.

2.3.1 End User

The end user is the one who wants to learn stock trading and/or practice trading stress-free with no risk of losing actual money. They should be able to perform all functionalities as mentioned in the Product Functionality section above.

2.4 Operating Environment

The application is desktop-based and version 1.0 would work on **Microsoft Windows**, since the initial development would be done on computers running Windows. Later versions of the application would support other major desktop OS's like MacOS and Linux. The targeted Windows versions are Windows 10 and above. The application would require **a stable connection to the internet** because stock prices are fetched through an API and the backend is managed by google Firebase, both of which require an internet connection. The API that will be used to get stocks related data is Alpha Vantage API (https://www.alphavantage.co/).

2.5 Design and Implementation Constraints

2.5.1 API usage limit

Version 1.0 would be using the free version of the Alpha Vantage API, which limits 5 API requests per minute and 500 requests per day for an API key. This API usage limit must be kept in mind while developing the application as trying to access the API at a higher frequency than the specified limits would not give the required results, and so proper error handling must be done.

2.5.2 Firebase Constraints

The application will use firebase for its backend. If firebase functionality isn't available at any time then the application won't work. Also, firebase's free version has limits like database access limits, so those must be kept in mind by the developer.

2.5.3 Internet Connectivity

As the application will use a third party API and Google Firebase, an internet connection is necessary for the application to work.

2.5.4 Hardware Constraints

To run this application, a computer with a mouse and keyboard is required. Since the development IDE is quite lightweight, a minimum RAM of 4 GB and a minimum disk space of 15 GB would suffice.

2.5.5 OS Constraint (Windows only)

As already highlighted, version 1.0 is targeted for Windows, specifically Windows 10 and above. Therefore computer's running windows 10 or above would be required for development of the application.

2.6 User Documentation

The software would come with a user manual which would include the following:

- Basics of stocks and stock exchange.
- Guide on how to navigate and use the application.
- Guide on how to get a free (or premium (paid), if user desires) Alpha Vantage API key.

2.7 Assumptions and Dependencies

2.7.1 Alphavantage API

It is assumed that the Alphavantage API would be working correctly and respond fast all the time as the API is an important factor in many function-

alities of the app. It is also assumed that any user won't be using the app in such a way that it exceeds the API limits of Alphavantage (which should be sufficient for most use cases). Moreover the API may change in terms of its syntax, functions, features, etc.

2.7.2 Firebase

Similarly it is assumed that firebase's services like authentication and database won't be down at any time. Moreover it is assumed that functionality limits of firebase won't be exceeded.

2.7.3 Computer

Although the application is not very CPU intensive, it is assumed that the application would be run on a computer with decent hardware specifications and a good internet connection.

Chapter 3

Specific Requirements

3.1 External Interface Requirements

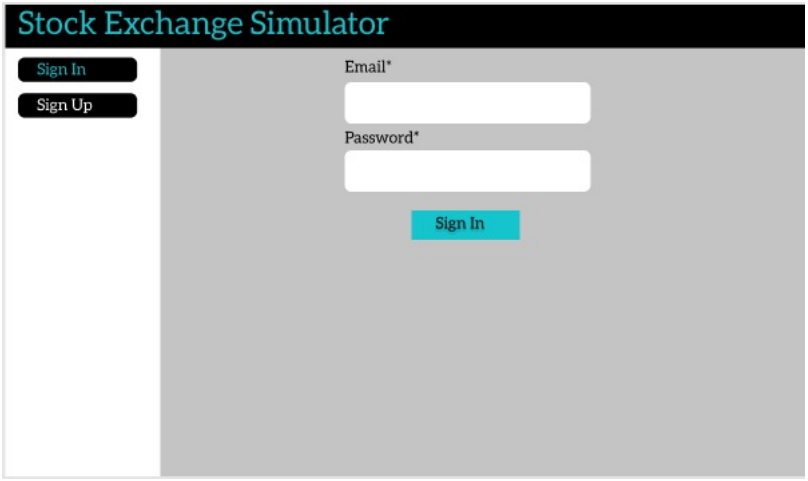
3.1.1 User Interfaces

The application's user interfaces can be classified into 2 categories depending on authentication state; i.e: the interfaces available when not signed in and when signed in. The application's general layout is to look uniform and consistent, especially the header and navigation bar, regardless of authentication state.

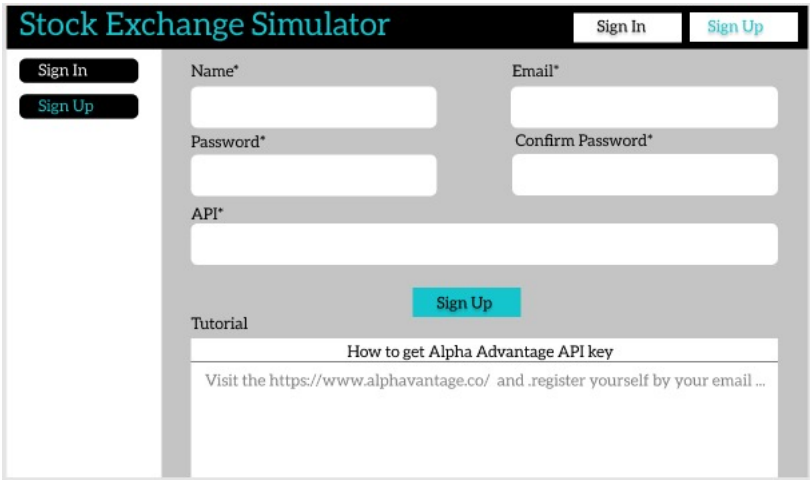
When not signed in

When not signed in, the navigation bar should only have buttons for signing in and signing up.

- **Sign Up page:** A form for signing up with fields for name, email address, password, confirm password, and API key, followed by a "Sign Up" button. and a simple guide on how to get an Alpha Vantage API.



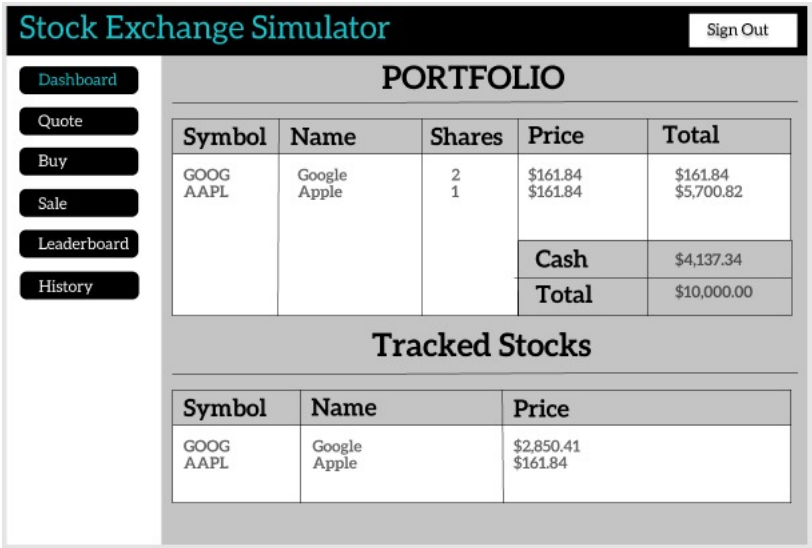
- **Sign In page:** A sign in form with fields for email address, and password.



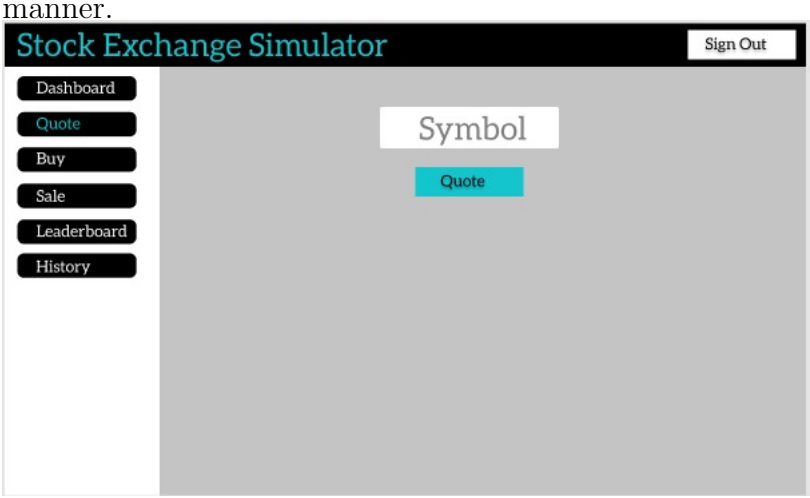
When signed in

When signed in, all the main features of the applications are to be accessible. The navigation bar in this state should include buttons for dashboard, quote, buy, sell, history, and leaderboard.

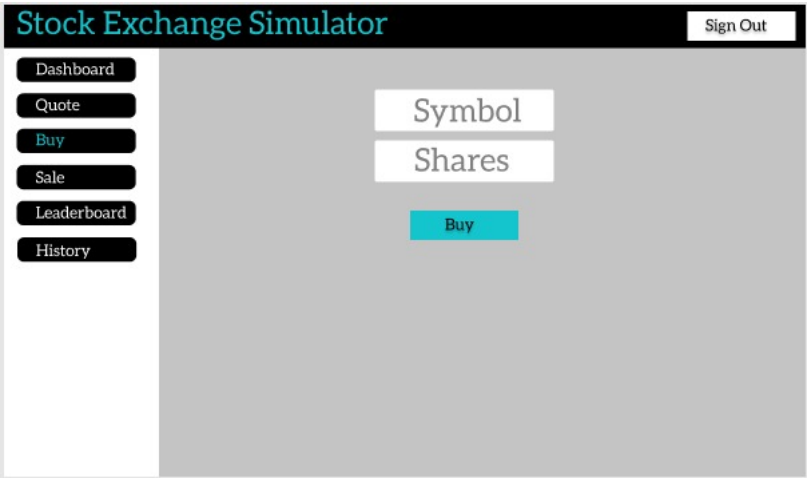
- **Dashboard Page:** The main / home screen of the application showing a dashboard summarizing the user's performance. Firstly a table showing all stock holdings and cash would be shown. Next, data of the user's tracked stocks would be shown.



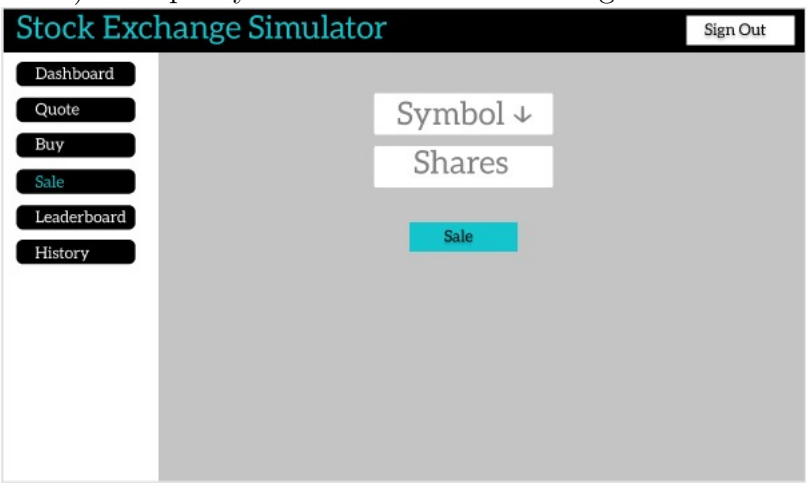
- **Quote Page:** A page to get latest information about any stock. An input field for searching stock (that would give autocomplete suggestions) with name of company or symbol of stock should be present. When searched, stock's information should be provided in user friendly manner.



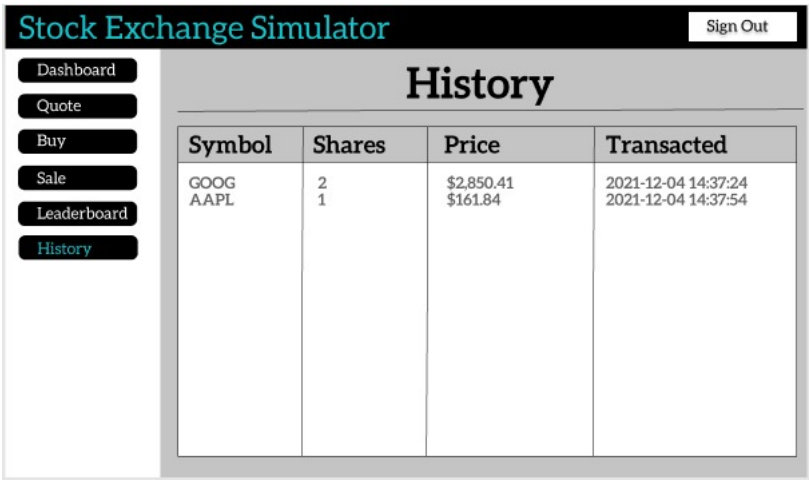
- **Buy**: A page to buy any stock. Input fields for symbol of stock and amount of stocks would be present.



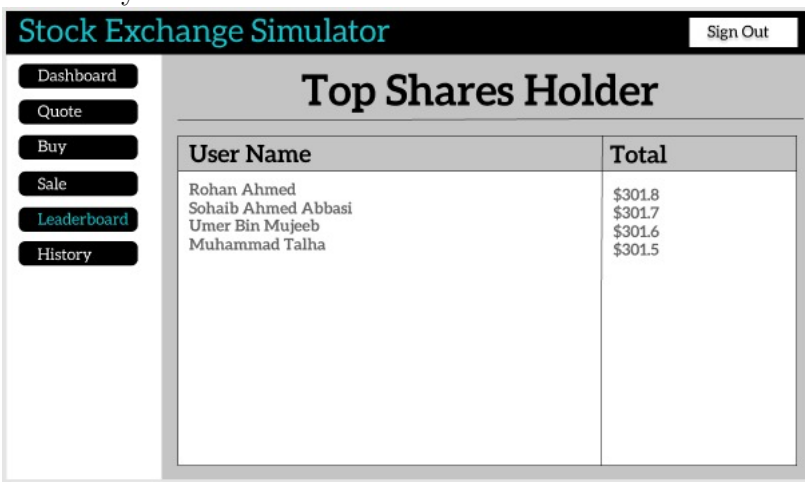
- **Sell**: A page to sell any of users stocks. Option to select a stock (drop-down) and specify amount to sell would be given.



- **History**: A history of user's stock transaction in tabular form should be present.



- **Leaderboard**: Leaderboard showing a table of all users on the app ranked by their total value.



3.1.2 Hardware Interfaces

This application is completely software based and has no hardware interfaces.

3.1.3 Software Interfaces

- The operating system for version 1.0 is Microsoft Windows (version 10 and above)
- The application will use Firebase Auth for user authentication
- The database to be used is Firebase's FireStore.

- The application will use Alphavantage API to get latest stocks related information.

3.1.4 Communications Interfaces

The application will be communicating with Alphavantage API using HTTP requests. In addition to that it will be using firebase for backend services including authentication and database.

3.2 Functional Requirements

3.2.1 General

The Application requires users to sign in to use its features. As such, the first screen visible should be the sign in screen with option to go to signup screen. Once signed in, the rest of the application should be accessible. Each user will be given an initial cash amount of $10,000.

3.2.2 Sign Up

The Sign Up page should have a sign up form including fields for name, email address, password, confirm password, and API key, followed by a "Sign Up" button. In addition to the form, a simple guide on how to get an Alpha Vantage API key should be present below the form. Constraints on password for version 1 are just that the password should atleast be of 4 characters. If valid data is entered, the user should be registered, signed in to the application, and navigated to the home screen. The data validation checks to be included are valid email address format, matching and valid passwords, and valid API key. If invalid data is entered however, appropriate error message(s) must be displayed.

3.2.3 Sign In

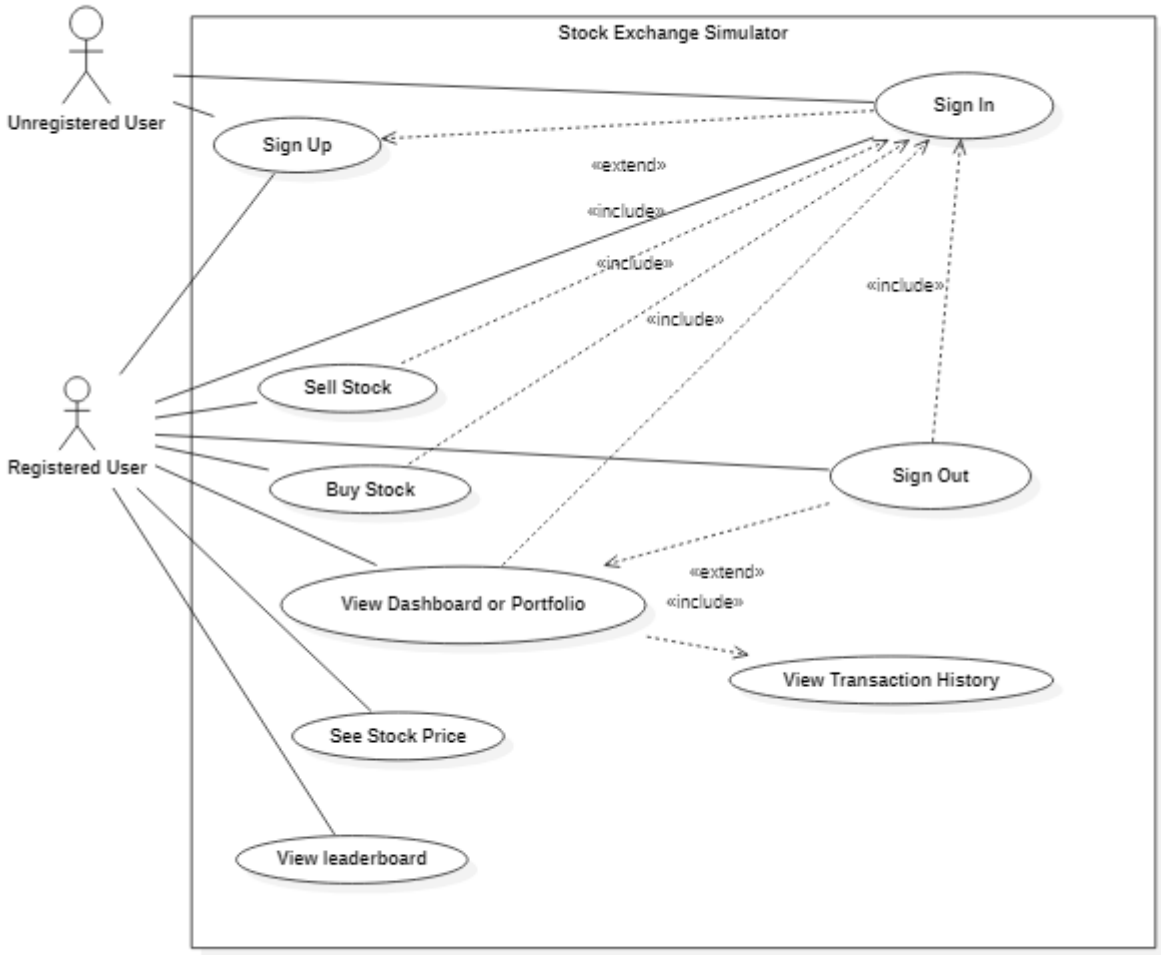
A similar layout to the Sign Up page should be displayed with the difference being in the fields of the form and no api key tutorial. The fields included should be email address and password. On successful sign in, the user should be redirected to home screen. On incorrect credentials, appropriate error message(s) must be displayed.

3.2.4 Home Screen (dashboard showing portfolio and tracked stocks)

This functional requirement covers product functionalities number 3 and 4 from section 2.2. The home screen of the application should show details of user's portfolio including information of their stocks and cash.

3.3 Behaviour Requirements

3.3.1 Use Case View



The actors are:

- **Unregistered Users** - Users who have not signed up yet or are signed up but do not sign in
- **Registered Users** - Users who have signed up and are signed in.

The use cases are:

- **Sign Up** - Users sign up by providing their personal information on a registration page form.
- **Sign In** - Users sign in by entering their username and password.
- **Sign Out** - End signed in session so no one else accesses an account.
- **See Stock Price** - View stock listings.
- **Buy Stock** - Buy stock and add it to account.
- **Sell Stock** - Sell stock already in account.
- **View Dashboard or Portfolio** - Users view their stocks, their valuations and other statistics on a dashboard page.
- **View Transaction History** - Users view their stock buying and selling history.
- **View Leaderboard** - Users see all users along with their valuations and ranked accordingly.

Chapter 4

Other Non-Functional Requirements

4.1 Performance Requirements

4.1.1 Application start-up time

The application should not take more than 10 seconds to load

4.1.2 Switching pages

The application should load any new page clicked on by the user in no more than 5 seconds

4.1.3 Authentication

Results of authentication (for example successful or failed sign-in or signup) should be obtained in no more than 5 seconds.

4.1.4 Stock price quote

Searching for a stock and getting its price from the quote page should not take more than 15 seconds

4.1.5 Buy/Sell Stocks Transactions

Result of buying and selling a stock should not take more than 10 seconds.

4.2 Safety and Security Requirements

4.2.1 Safety Requirements

This product does not have any particular safety hazards other than the common problems associated with long periods of computer usage such as the following:

- eye strain / weakening of eyesight
- headaches
- postural issues and muscular imbalances

As such, it is the responsibility of the end user themself to avoid using the application for prolonged periods of time and take regular breaks.

4.2.2 Security Requirements

- Users' passwords must be stored safely and not leaked in any case
- Every user's data must be protected from tampering by any other user

Security related to authentication would be satisfied easily as Google's trusted Firebase system would be used for authentication. However for database security, the developers must design the Firestore database and define its security and access rules properly.

4.3 Software Quality Attributes

4.3.1 Readability

The developers must ensure that the code they write is clean and highly readable, for ease of future development. Bad practices must be avoided. Clear and consise comments should be provided.

4.3.2 Maintainability

Although high readability would play a part in making the code maintainable, additional efforts must be made in this regard such as modularization of code as much as possible and making each module perform small independent tasks.

4.3.3 Adaptability

The code must be written in such a way that it can be easily adapted to changes in syntax, functionality, etc. of Firebase and Alpha Vantage API.

4.3.4 Reuseability

The modules of the program must be reusable to make it easy to extend and enhance this application in future versions.

4.3.5 Robustness

The main reason for crashes in the application could be change in syntax and functionality of Firebase or Alpha Vantage API. Developers must keep this fact in mind and expect such changes. User's data must not be affected in such situations and the program must not give any output that is not easy to understand for a common user.

4.3.6 Usability

The application's functions and user interface must be designed while keeping in mind that most of the end users would be common people who may not have much technical expertise and just want to learn stock trading. As such the application must be very simple, easy to use, and intuitive.