Software Engineering
Software Requirements Specification (SRS) Document

[Stock Tracker App]

[09/25/2023]

[0.1]

By: [David Vasquez, Salomon Perez, Ro Mei]

[Honor Code]

Table of Contents

1. Introduction

1.1. Purpose

[The goal of your project and the objectives it wishes to accomplish]

The goal of the Stock Tracking App is to allow its users to monitor changes in the stock market whenever they wish to do so. The app will allow its users to select the stocks they want to track, check news information that are deemed relevant to stocks, and allow its users to access historical stock data.

1.2. Document Conventions

[Full description of the main objectives of this document in the context of your project.
Here's how you should begin this section:
"The purpose of this Software Requirements Document (SRD) is to..."
"In it, we will . . ., . . ., and . . .."]

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the Stock Tracking App (STA). Client-oriented requirements describe the system from the client's perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3. Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.
For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.
API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

1.4. Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

- Section 1: Stakeholders, Users, Project Managers, Developers and Users.
- Section 2: Stakeholders, Developers, Project Managers.
- Section 3: Developers, Project Managers.
- Section 4: Developers, Project Managers.
- Section 5: Developers, Project Managers.
- Section 6: Developers, Project Managers.

1.5. Project Scope

[Specify how the software goals align with the overall business goals and outline the benefits of the project to business.]

The goal of the software is to provide an easy-to-use, and accurate, interface for Stock Trackers, News Readers, and Stock Market Researchers, as well as provide users with the flexibility to meet their needs. This aligns with the overall business goals of the stock tracking app as a stock tracking system requires fast, efficient, and accurate services in order to fulfill the needs of its users.

The benefits of the project to it users include:

- Allowing its users to be able to check their preferred stocks from anywhere, without having to give up too much personal information (SSN, Government issued ID, etc.) like you'd do in stock trading apps.
- Giving users relevant stocks-related information that they may otherwise not have heard of, allowing our users to be caught up to everything deemed relevant by the API in the stocks world..
- Providing historical stock data to our users, allowing them to be able to research and study the market.

1.6. Technology Challenges

[Any technological constraints that the project will be under. Any new technologies you may need to use]

1.7. References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

- Ticker News - Polygon. (n.d.). Polygon.io. https://polygon.io/docs/stocks/get_v2_reference_news
- Morah, C. (2022). What are all of the major US stock exchanges? *Investopedia*. https://www.investopedia.com/ask/answers/08/security-market-usa.asp
- Srivastava, S. (2023, June 30). Stock Trading App Development: A Complete guide. Appinventiv. https://appinventiv.com/blog/stock-trading-app-development/
- Chen, J. (2022). Tracking Stock: Definition, Benefits, Risks, and example. Investopedia. https://www.investopedia.com/terms/t/trackingstocks.asp#:~:text=A%20tracking%20stock%20is%20a%20specialized%20equity%20security%20issued%20by,independent%20of%20the%20parent%20stock.

2. General Description

2.1. Product Perspective

[Describe the context and origin of the product.]

The stock app originated from an idea of having a single place that is beginner friendly for keeping up with stock news and price updates, as well as historical stock data, of any selected stock, all intended for beginning traders.

2.2. Product Features

[A high-level summary of the functions the software would perform and the features to be included.]

The product features include the ability for users to create a listing of stocks of their choice to follow current market trends. The software also includes the ability to see news related to stocks chosen by the user allowing them to be informed on events involving the company. The ability to see historical information related to pricing of the stock is another feature that will allow users to see how the stock has performed in the past.

2.3. User Class and Characteristics

[A categorization and profiling of the users the software is intended for and their classification into different user classes]

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser. It does however require a user to be familiar with basic stock market knowledge and the concept of financial growth and shrinkage.

2.4. Operating Environment

[Specification of the environment the software is being designed to operate in.]

The application is designed to operate on the web across many different devices.

2.5. Constraints

[Any limiting factors that would pose challenge to the development of the software. These include both design as well as implementation constraints.]

The use of a free API that will provide the stock data may not be able to quickly provide updated information in comparison to a subscription-based service.

2.6. Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans. The application will also use the Polygon API (https://polygon.io/docs/stocks/getting-started) that provides the application with current stock price information, news, and historical data related to the stocks.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

3.1. Primary

[All the requirements within the system or sub-system in order to determine the output that the software is expected to give in relation to the given input. These consist of the design requirements, graphics requirements, operating system requirements and constraints if any.]

- FR0: The system will allow the user to lookup stocks and see current pricing information.
- FR1: The system will allow the user create a list of stocks to track and follow updates on.
- FR2: The system will allow the user to view news and historical data related to the stock of their choice, as well as compare them.

3.2. Secondary

[Some functions that are used to support the primary requirements.]

- The ability for users to create accounts that will keep their lists saved.
- A personal portfolio for Stock Trackers.
- Search bar for Stock Trackers to look up stocks.
- News tab for News Recipients.
- Pin system that allows Stock Researchers to select and compare stocks.

4. Technical Requirements

4.1. Operating System and Compatibility

[The environments that will be needed to operate the system]

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

4.2.2. Hardware Interfaces

[All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.]

The Stock Tracker application will run on any hardware device that has access to the internet, the ability to display webpages and interact with web pages. User can use the app on smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

[Determination of all the communication standards to be utilized by the software as a part of the project]

It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the Polygon API and return the current date and time.

4.2.4. Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to the used, the database management system and libraries describing the need and the purpose behind each of them.]
We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

5. Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

5.1. Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

- NFR0(R): The local copy of the vehicle violation database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the vehicle violation database) will consume less than 50MB of memory
- NFR2(R): The novice user will be able to create and print a ticket in less than 5 minutes.
- NFR3(R): The expert user will be able to create and print a ticket in less than 1 minute.

5.2. Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

5.3. Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

- NFR4(R): The system will only be usable by authorized users.

5.4. Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

5.4.1. Availability

[Details]

5.4.2. Correctness

[Details]

5.4.3. Maintainability

[Details]

5.4.4. Reusability

[Details]

5.4.5. Portability

[Details]

5.5. Process Requirements

5.5.1. Development Process Used

[Software Process Model]

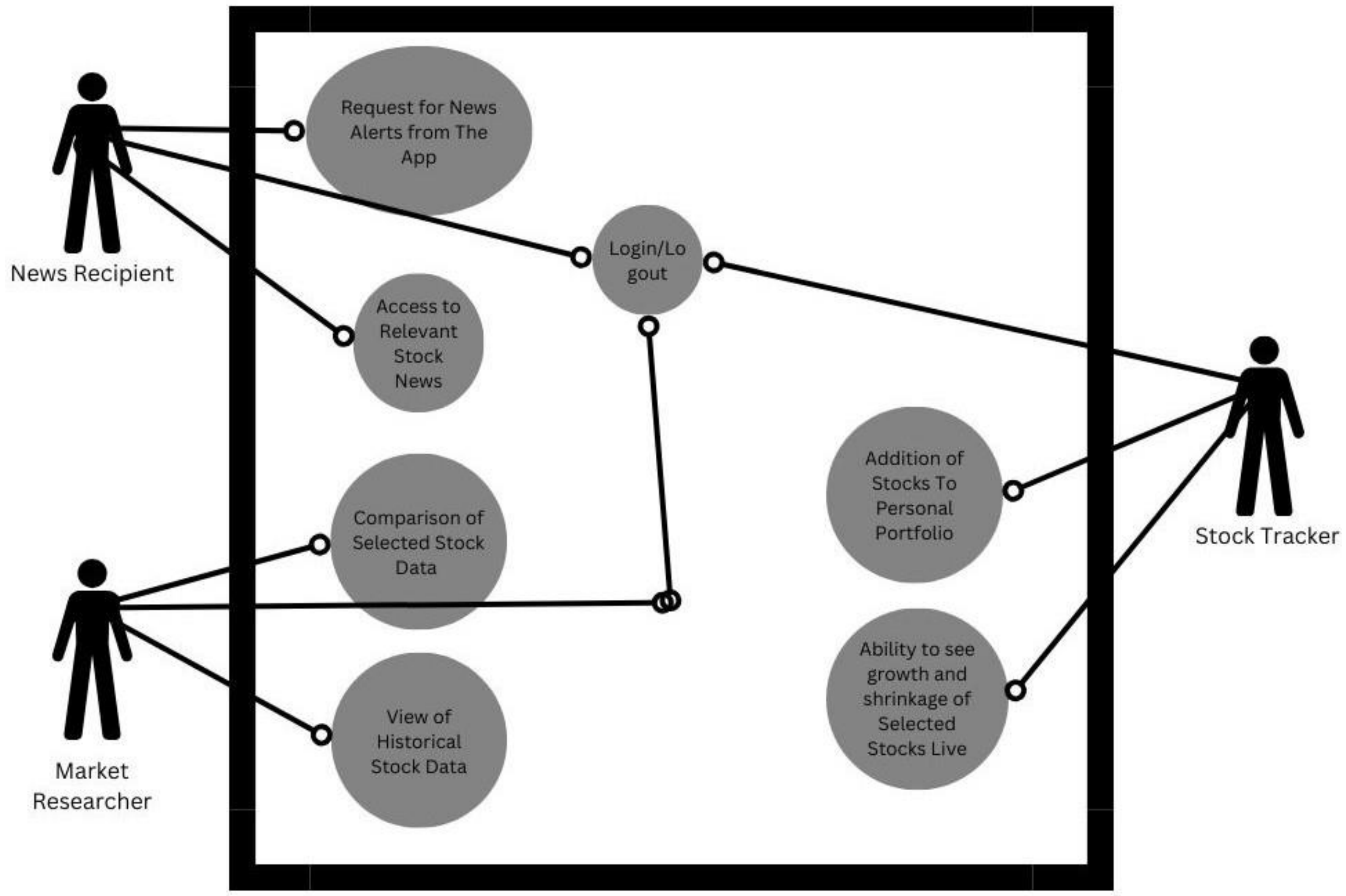
5.5.2. Time Constraints

5.5.3. Cost and Delivery Date

5.6. Other Requirements

TBD

5.7. Use-Case Model Diagram



News Recipient
Request for News Alerts from The App
Login/Logout
Access to Relevant Stock News
Comparison of Selected Stock Data
View of Historical Stock Data
Addition of Stocks To Personal Portfolio
Ability to see growth and shrinkage of Selected Stocks Live
Stock Tracker
Market Researcher

5.8. Use-Case Model Descriptions

5.8.1. Actor: News Recipient (Responsible Team Member: Ro Mei)

- **Request for News Alerts from the App**: [Allows the News Recipient to opt for alerts containing news shown on the app.]
- **Access to Relevant Stock News**: [Allows the News Recipient access to a news tab containing relevant stock-related news.]

5.8.2. Actor: Market Researcher (Responsible Team Member: David Vasquez)

- **Comparison of Selected Stock Data**: [Allows Market Researchers to compare up to two stocks and their historical data.]
- **View of Historical Data**: [Market Researchers have access to historical data of stocks, allowing them to study and analyze the stocks available.]

5.8.3. Actor: Stock Tracker (Responsible Team Member: Salomon Perez)

- **Addition of Stocks to Personal Portfolio**: [Allows the Stock Tracker to look up and add stocks to their personal portfolio, to keep track of them for later use.]
- **Ability to see growth and Shrinkage of Selected Stocks Live**: [The Stock Tracker is able to see the live growth/shrinkage of their selected stocks.]

5.9. Use-Case Model Scenarios

5.9.1. Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:

5.9.2. Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:

5.9.3. Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:

- **System State on Completion**:

6. Design Documents

6.1. Software Architecture

6.2. High-Level Database Schema

6.3. Software Design

6.3.1. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.2. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.3. State Machine Diagram: Actor Name (Responsible Team Member)

6.4. UML Class Diagram

7. Scenario

7.1. Brief Written Scenario with Screenshots