Software Requirements Specification

Version 1.0

February 3rd, 2024

Financial Service Application

Michael Barry

Ahmed Issa

Angel Lopez

Dillon Bridgewater

Jeff Spinner

Steiner Christensen

Table of Contents

[1.0 Introduction 3](#_Toc157977114)

[1.1 Purpose 3](#_Toc157977115)

[1.2 Scope of Project 3](#_Toc157977116)

[1.3 Relative Terminology 3](#_Toc157977117)

[1.4 References 4](#_Toc157977118)

[1.5 Overview 4](#_Toc157977119)

[2.0 Overall Description 4](#_Toc157977120)

[2.1 System Environment 4](#_Toc157977121)

[2.1.1 Interfaces 5](#_Toc157977122)

[2.1.2 Memory 7](#_Toc157977123)

[2.1.1 Operations 7](#_Toc157977124)

[2.2 User Cases 9](#_Toc157977125)

[2.3 User Characteristics 10](#_Toc157977126)

[2.4 Non-Functional Requirements 10](#_Toc157977127)

[3.0 Requirements Specification 10](#_Toc157977128)

[3.2 Detailed Function Analysis: 10](#_Toc157977129)

[3.2.1 System Feature 1: 10 stock summary view 10](#_Toc157977130)

[System Feature 2: Charting Page 11](#_Toc157977131)

[3.2.1.1 Introduction 11](#_Toc157977132)

[3.2.1.2 Stimulus 11](#_Toc157977133)

[3.2.1.3 Associated Functional Requirements 11](#_Toc157977134)

[3.3 Performance Requirements 13](#_Toc157977135)

[3.4 Design Constraints 13](#_Toc157977136)

[3.5 Software system attributes 13](#_Toc157977137)

[3.5.1 Security 13](#_Toc157977138)

[3.5.2 Maintainability 13](#_Toc157977139)

# 1.0 Introduction

## 1.1 Purpose

The purpose of this document is to present a detailed description of the Financial Web Service, referred to as the system. This document will outline the system's objectives, key features, user interfaces, operational constraints, and its interactions with external stimuli. It aims to serve as a comprehensive guide for both stakeholders and developers involved in the project. The document will facilitate a clear understanding of the system's functionality, requirements, and the technological framework within which it operates. By providing this information, the document seeks approval from the project's stakeholders and ensures a unified vision for the development team.

## 1.2 Scope of Project

This software system is designed as a robust financial web service that aims to enhance the user experience in accessing and analyzing stock market data. The project will enable users to view live and historical data on various stocks, interact with dynamic graphs, and obtain detailed company information, all within a user-friendly web interface. The system will leverage a remote MySQL database for data storage and employ Python and Django for backend operations, with React powering the frontend.

Key functionalities include a "Market Watch" feature displaying the current performance of major indices like Dow, S&P 500, and NASDAQ; interactive stock graphs showing over 20 years of data; and a comprehensive database of stock and company information. The system is intended to support individual investors, financial analysts, and educational users by providing them with tools to make informed decisions based on up-to-date and historical financial data.

## 1.3 Relative Terminology

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Financial Web Service (System) | The web-based application being described by this document. |
| User | An individual who interacts with the system, ranging from novice investors to experienced financial analysts. |
| Stock Market Data | Information pertaining to stock prices, market trends, and financial metrics. |
| Dynamic Graphs | Interactive visual representations of stock market data over time. |
| Market Watch | A feature of the system that displays real-time performance data of major stock market indices. |
| MySQL Database | The database technology used for storing and managing the system's data. |
| Python & Django | The programming language and web framework used for backend development. |
| React | The JavaScript library used for building the system's user interface. |
| API (Application Programming Interface) | Software intermediaries that allow two applications to communicate with each other. |
| RapidAPI | An API marketplace used by the system for accessing financial data. |
| Alpha Vantage | A provider of APIs for historical and real-time financial data accessed by the system. |
| Server | The computer that is running the backend program and the database. |
| Frontend | The part of the System that is visible to the user and runs on their computer. |
| Backend | A program written in Python that uses the DJango API to manage communication between the frontend, Alpha Vantage, and the MySQL database. It provides information from the Database and/or Alpha Vantage when the frontend requests it. |
| API Call | Any time the System requests data using the Alpha Vantage API, an API call is used. These are a limited resource. |

## 1.4 References

* Alpha Vantage API Documentation
* Django Documentation
* React Documentation
* MySQL Database Documentation

## 1.5 Overview

The following chapter, "Overall Description," provides a comprehensive overview of the system's functionality, including informal requirements and the system's context. This section is designed to lay a foundation for the technical specifications detailed in the subsequent chapter, aimed at developers.

The third chapter, "Requirements Specification," delves into the technical details of the system's functionalities, tailored for the development team. It specifies the system requirements in a manner that ensures clarity and precision, facilitating the development process.

Both sections collectively describe the entirety of the software product, catering to different audiences through varied linguistic approaches.

# 2.0 Overall Description

## 2.1 System Environment

A diagram of a stock data manager

Description automatically generated

Figure 1 - System Environment

The system is an independent, self-contained system that does not serve as serve as component of a larger system. It is designed to be a standalone product that interacts with external financial data providers and presents the data through a web-based interface. The system comprises several interlinked components:

* **Data Fetching:** Interacts with external financial APIs to retrieve stock data.
* **Data Management:** Processes and stores the retrieved data in a database.
* **User Interface:** Displays the information to the user in an accessible and interactive format.

### 2.1.1 Interfaces

#### 2.1.1.1 System Interfaces

* **Alpha Vantage via Rapid API:** The system communicates with Alpha Vantage through RapidAPI, to manage the limited number of API calls available, the system intelligently utilizes a MySQL database hosted online. The backend, developed with Django and Python, is responsible for pulling data from Alpha Vantage, processing it, and storing it in the MySQL database. This approach ensures optimal use of the available API calls and allows for a reliable and up-to-date stock database.
* **Remote MySQL Database:** Thedatabase serves as the repository for stock data. The backend, powered by Django and Python, interacts with this database to store, organize, and maintain the integrity of the financial data. Centralized data storage enhances data accessibility, allowing the system to efficiently fetch and present information to users. The database also supports long-term storage for historical stock data.
* **Backend (Django with Python):** The backend operations utilize Django, a powerful web framework, utilizing Python. This component manages communication between the frontend, Alpha Vantage via RapidAPI, and the MySQL database. It ensures that the front-end receives accurate stock data while optimizing the utilization of API calls. The backend handles data processing, ensuring the information presented to users is both meaningful and relevant.
* **Frontend (HTML, CSS, JavaScript, possibly React):** The user interface is developed using a combination of HTML, CSS, and JavaScript, possibly incorporating React to enhance interactivity. This approach ensures a responsive and visually appealing experience across various devices. The front-end interacts seamlessly with the backend to fetch and display stock data. The layout is optimized for varying devices, offering a consistent design and user experience. The main page showcases the main stock movers of the day, along with interactive graphs. Below, there will be smaller information about all the other stocks, all clickable to take the user to detailed stock information presented with interactive graphs on a separate page. Simultaneously, the second page will provide a way for the user to change the stock they are looking at without going back to the main page. Finally, there will be another page about the team and the project.

#### 2.1.1.2 User Interfaces

The front-end is optimized for diverse devices, ensuring a seamless experience for users accessing the Financial Web Service from different platforms. The system will ensure a user-friendly design, aiming to provide the interface with a clear sense of affordance. Having a consistent design, layout, and interactions across the interface will maintain clarity and efficiency. The content layout will draw inspiration from other financial service systems. This approach aims to provide users with a clear understanding of the functions each element performs. The system will consist of at least three pages.

* **Main Page:**
* Top Movers Section: Showcases the top stock movers of the day, providing users with immediate insights into market dynamics.
* Interactive Graph: Visual representations of stock movements enhance user engagement and understanding.
* Additional Stock Information: Following the top movers, a section featuring concise information about additional stocks is presented. Users can click on any stock to access detailed information on a separate page.
* **Detailed Stock Information Page:**
* Enhanced Stock Detail: This page offers additional details for analysis of the selected stock. Users will have access to company data, including a comprehensive description, closing price, and other relevant information.
* Interactive Graphs: Users can explore historical and real-time stock data through interactive graphs, offering a comprehensive view of a selected stock's performance.
* Dynamic Stock Changing: Users have access to a navigation tool to change between the viewed stock without navigating back to the main page, improving overall efficiency.
* **Team and Project Page:**
* Project Overview: Provides information about the Financial Web Service, its objectives, and key features.
* Team Information: Introduces the team members working on the project, fostering transparency and user connection.

#### Software Interfaces

* Alpha Vantage API: The interfacing API is an important aspect of the system which provides access to raw stock market data. That raw data is transformed into a digestible format for enhanced analysis within the system.
* **Mnemonic:** RAPI
* **Specification number:** 1.0
* **Version Number:** Latest
* **Source:** Marketplace for accessing various APIs, including Alpha Vantage
* MySQL Database: The interfacing software stores the raw data collected from Alpha Vantage API on a MySQL server. This raw data is processed, cleaned, and sorted for frontend use. Allowing ease of access remotely.
* **Mnemonic:** MySQLDB
* **Specification number:** 1.0
* **Version Number:** Latest
* **Source:** Online database for storing and managing stock-related data
* Django: The framework serves as the backend of the system, providing a seamless connection. It is a straightforward approach to linking various components and functions.
* **Mnemonic:** Django
* **Specification number:** 3.0
* **Version Number:** Latest
* **Source:** Python web framework used for backend development
* React: The React library is crucial to the frontend of the system. Enabling a smooth and efficient implementation to the user interface.
* **Mnemonic:** React
* **Specification number:** 17.0
* **Version Number:** Latest
* **Source:** JavaScript library for building the user interface

#### 2.1.1.4 Communications Interfaces

In our Financial Web Service project, various communication interfaces are essential for the seamless exchange of information between different components. These interfaces ensure effective communication and coordination within the system.

* **Frontend-Backend Interaction:**
* The user interface (Frontend) communicates with the backend (Brain) to request and receive real-time stock information.
* This communication relies on HTTP requests, providing a standardized method for fetching data and maintaining a responsive user experience.
* **Backend-Alpha Vantage via Rapid API Communication:**
* The backend communicates with Alpha Vantage via RapidAPI to obtain up-to-date stock data.
* Secure API calls are employed to fetch essential financial information, supporting the accuracy of the information presented to users.
* **Backend-MySQL Database Interaction:**
* The backend interacts with the online MySQL Database to store, organize, and retrieve stock data efficiently.
* This communication involves SQL queries, ensuring data integrity and facilitating smooth data operations within the system.
* This communication takes place entirely within the server.
* **Frontend-User Interaction:**
* User interactions with the frontend trigger requests to the backend for specific stock data or actions.
* These interactions, managed through responsive user interface components, allow users to navigate through the system and access detailed stock information.
* **Backend-Server Communication:**
* The backend operates on a server, and communication with the server is crucial for data processing, retrieval, and storage.
* Server communication is vital to maintaining the integrity of financial data and ensuring the overall reliability of our Financial Web Service.

### 2.1.2 Memory

In the context of the Financial Web Service, the memory aspect refers to how the system manages and handles data, both in terms of storage and retrieval. This section outlines the key elements related to memory operations within the system.

* **Data Storage in MySQL Database:**
* **Description:** The MySQL database serves as the memory repository for storing stock-related information.
* **Details:**
* The database stores real-time and historical stock data retrieved from Alpha Vantage via RapidAPI.
* Stock data is organized in a structured manner to facilitate efficient retrieval and processing.
* **Frontend-Backend Data Interaction:**
  + **Description:** The frontend and backend collaborate to manage data interactions between the user interface and the MySQL database.
  + **Details:**
    - Frontend sends requests to the backend for specific stock data.
    - The backend retrieves the requested data from the MySQL database.
    - This interaction ensures responsive data access for the user interface.

### 2.1.1 Operations

The operations in the Financial Web Service outline the key actions and functionalities that the system performs to provide users with accurate and timely stock information. These operations cover the interaction between various components, ensuring the smooth flow of data and maintaining the overall functionality of the system.

* **Fetch Stock Data from Alpha Vantage via RapidAPI:**

A diagram of a data flow

Description automatically generated

Figure 2 – Stock Data Fetching and Processing

* **Description:** The backend communicates with Alpha Vantage through RapidAPI to retrieve both real-time and historical stock data.
* **Steps:**
* Backend initiates an API call to Alpha Vantage via RapidAPI.
* The response, containing stock data, is processed and stored in the MySQL database.
* If necessary, the database is updated with the latest information.
* **Retrieve Stock Data from MySQL Database:**

**A diagram of a stock data

Description automatically generated**



Figure 3 - Stock Data Retrieval

* **Description:** The backend interacts with the MySQL database to access stored stock information efficiently.
* **Steps:**
* Frontend requests specific stock data from the backend.
* Backend retrieves the requested data from the MySQL database.
* The data is sent to the front-end for display, ensuring a quick response to user queries.
* **Switch Viewed Stock on Detailed Page:**

**A diagram of a stock data

Description automatically generated**

Figure - Stock Previewing

* **Description:** The front-end allows users to seamlessly switch between different stocks on the detailed stock information page.
* **Steps:**
* User selects a different stock from the available options on the detailed page.
* Frontend sends a request to the backend to change the viewed stock. Backend retrieves the data for the
* newly selected stock from the MySQL database.
* The front-end updates the page with details and graphs of the newly chosen stock.

## 2.2 User Cases

In the Financial Web Service, user cases outline the key interactions between the system and its users. These scenarios help us understand how individuals, ranging from novice investors to experienced financial analysts, will interact with our platform.

* **User Accesses Stock Information:**
* **Trigger:** User opens the Financial Web Service.
* **Basic Path:**
* Frontend requests stock data from the backend.
* Backend checks the database for up-to-date stock information.
* If data is outdated, the backend fetches the latest data from Alpha Vantage via RapidAPI and updates the database.
* Frontend displays current stock information, including main stock movers and interactive graphs.
* **User Explores Stock Details:**
* **Trigger:** User clicks on a specific stock from the main page.
* **Basic Path:**
* Frontend sends a request to the backend for detailed information on the selected stock.
* Backend retrieves the relevant data from the database.
* Frontend displays a new page with interactive graphs and comprehensive details about the chosen stock.
* **User Changes Viewed Stock on Detailed Page:**
* **Trigger:** User, on the detailed stock information page, selects a different stock.
* **Basic Path:**
* Frontend sends a request to the backend to switch the viewed stock.
* The backend retrieves the new stock's data from the database.
* Frontend updates the page with the details and graphs of the newly selected stock.
* **User Navigates to Team and Project Page:**
* **Trigger:** User clicks on the designated page about the team and project.
* **Basic Path:**
* 1.Frontend sends a request to the backend for information about the team and project.
* 2.Backend retrieves relevant details.
* 3.Frontend displays a page with information about the team members and an overview of the

## 2.3 User Characteristics

* **General Financial Interest**
* **Description:** Users with a general interest in financial markets, seeking to stay informed about top and safe stocks prices and market trends
* **Characteristics:**
  + Look for a user-friendly interface to easily access financial data.
  + Seek summarized information about major stock indices and top-performing stocks.

## 2.4 Non-Functional Requirements

The Financial Web Service will be hosted on a server with internet access. The physical server will be an Oracle Cloud Virtual Machine VM. Standard.E2.1. Micro instance. The server will have its firewall allow the port 8000 open to incoming traffic for communication with the Financial Web Service. The database will be hosted on this server too.

Users will need a web browser capable of running JavaScript and must have a computer with internet access.

# 3.0 Requirements Specification

## 3.2 Detailed Function Analysis:

### 3.2.1 System Feature 1: 10 stock summary view

#### 3.2.1.1 Introduction

Main page to show prices and trends of available stocks.

#### 3.2.1.2 Stimulus

Accessing stock website.

#### 3.2.1.3 Associated Functional Requirements

##### 3.2.1.3.1 Find Stock

|  |  |
| --- | --- |
| Requirement Name | Find Stock |
| XRef | 2.2.1 |
| Trigger | User Accesses Stock Website |
| Basic Path | 1. Frontend asks Backend for stock data 2. Backend determines if stock data stored in Database is up to date. 3. Backend provides up-to-date stock data. 4. Frontend displays stock data as a table. 5. Frontend sorts stocks alphabetically, the user can change this to be by price. 6. The user selects a stock by clicking the row in which its data is contained. |
| Alternative Paths | If in step 2, the stock data stored on the database is not up to date, the Backend will request up to date data from Alpha Vantage. It will then update the data in the Database. |
| Postcondition | If the data in the database was not up to date, it will be updated. |
| Other | The stock information on the table includes daily high, daily low, daily volume, and percent change from previous day |

|  |  |
| --- | --- |
| Requirement Name | View Team and Project Page |
| XRef | 2.2.4 |
| Trigger | User Selects Team or Project Page |
| Basic Path | 1. Frontend sends a request to the backend for information about the team and project. 2. Backend retrieves relevant details. 3. Frontend displays a page with information about the team members and an overview of the |

### System Feature 2: Charting Page

### 3.2.1.1 Introduction

The Charting Page is a page that shows a chart of stock prices over time for a particular stock. There is a small list of alternative stocks to switch to. There are options to change features about the graph,

### 3.2.1.2 Stimulus

Selecting a stock from the 10-stock summary screen.

### 3.2.1.3 Associated Functional Requirements

##### 3.2.1.3.1 View default chart

|  |  |
| --- | --- |
| Requirement | View default chart |
| XRef | 2.2.2 |
| Trigger | Select Stock from main page |
| Basic Path | 1. Frontend displays a chart with the data for a default stock. 2. The user selects another stock from list to change chart. |
| Other | The data displayed on this chart is a line graph or optionally a candle graph whose vertical axis is the price of the stock for the date and time, and whose horizontal axis is the date and time. |

##### 3.2.1.3.2 Change options of chart

|  |  |
| --- | --- |
| Requirement | Change options of chart |
| XRef | 2.2.3 |
| Trigger | User presses chart’s option button |
| Precondition | User has selected a stock to be dispalyed on the chart page |
| Basic Path | 1. Frontend displays a list of buttons representing options that the user can toggle or otherwise change the state of. 2. User selects an option. 3. Frontend draws chart with new option reflected. |
| Alternative paths | If during step 2, the user does not select an option, the chart does not change.  If, during step 2, the user toggles on the logarithmic scale option, the vertical axis of the chart, representing stock price, is scaled in logarithmic increments.  If, during step 2, the user toggles off the logarithmic scale option, the vertical axis of the chart is scaled with linear increments.  If, during step 2, the user toggles on the line graph option (and by necessity, automatically turns off the candles option), the chart is displayed with a series of connected diagonal line segments that move between discrete points, with each line segment starting at the point whose vertical component is proportional to the previous price of the stock, whose horizontal component is proportional to the time for that previous price, and whose ending point is made up of a vertical component proportional to the next price of the stock, and whose horizontal component is proportional to the time during which that next price was the price.  If, during step 2, the user toggles on the candle option (and thus off the line option), the chart displays a series of “candles”. Each candle represents a discrete change in the price of the stock, and the candles heightis proportional to the change in the price of the stock, and the candles width is proportional to the chronological duration during which that change in the price of stock happened. The candles are red if the price decresed during the time interval, and the candles are green if the price increased during the time interval. |

##### 3.2.1.3.3 Change the Chronological Range of the Chart

|  |  |
| --- | --- |
| Requirement | Change the Chronological Range of the Chart |
| XRef | 2.2.3 |
| Trigger | The user changes the minimum and or maximum date and time of the chart. |
| Precondition | User has selected a stock to be dispalyed on the chart page. |
| Basic Path | 1. The frontend requests data from the backend for the data for the displayed stock for the time range in between the minimum and maximum date times specified by the user. 2. The frontend re-draws the chart using this new data. |
| Postcondition | If the database did not have data for the desired date range before, it will use an API call if one is available and will store that data for the desired range. |

## 3.3 Performance Requirements

Client-side performance needs to maintain limited memory usage via trash collection and storing as little data client side as possible at a time. Additionally consider the visual performance of maintaining and updating the chart. Server-side performance is concerned with long-term storage of values in the database controlled through pruning certain data and storing data in most condensed form. Additionally, the client will use caching to control the number of requests to the server from a client. Supports up to 100 simultaneous users and responds to user requests in under 2s.

## 3.4 Design Constraints

Time constraints and limited development time requires simpler designs to allow for the project to be completed on time. We are constrained by the API: it limits requests to 125 per day per key, of which we have 5, which limits how up to date the data provided can be.

## 3.5 Software system attributes

### 3.5.1 Security

The system will be structured to maintain the security of data stored on the server, such as the Alpha Vantage API keys and server file structure. The system will not store any other private information so there is little risk of data breaches.

To maintain security while the frontend is communicating with the backend, Django will be set to use HTTPS, so that the messages exchanged are encrypted.

Specific IP’s need to be allowlisted before they are allowed to log in to the database.

### 3.5.2 Maintainability

The system will be easy to maintain because it is simple. The primary difficulty of maintainability will be the database and API functionality. The system will have to maintain a server with a database it can access to store historical data which must be available for it to function. Another avenue that must be maintained is the Access to the Alpha vantage API that is used to update the stock data.