STOCK PRICE PREDICTION APP

Software Requirements Specification

Table of Contents

Software Requirements Specification

1. Introduction

1.1 Purpose

This document serves as the Software Requirements Specification (SRS) for the Stock Price Prediction App. Its primary purpose is to define the comprehensive functional and non-functional requirements of the application. This SRS will guide the development and implementation phases, ensuring that the final product meets the needs of its users and stakeholders by providing reliable and accurate stock price predictions through a user-friendly interface.

1.2 Scope

The Stock Price Prediction App is designed to allow users to select stocks, view historical data, analyze using technical indicators, and predict future prices using advanced forecasting models like Prophet and LSTM. The application caters to individual investors, financial analysts, and educational users who seek a tool for making informed investment decisions.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **LSTM**: Long Short-Term Memory
- **API**: Application Programming Interface
- **GUI**: Graphical User Interface

1.4 References

- Python Documentation
- Streamlit User Guide
- yfinance API Documentation
- Prophet Documentation

1.5 Overview

The remainder of this document is structured as follows:

- **Section 2: Overall Description**: Describes the general factors that affect the product and its requirements. This section covers product perspective, product functions, user characteristics, and constraints.
- **Section 3: System Features**: Detailed description of the system's features and their requirements.
- **Section 4: External Interface Requirements**: Interface requirements including user, hardware, software interfaces, and communications protocols.

- **Section 5: Other Non-functional Requirements**: Performance, safety, security, and software quality attributes.
- **Section 6: Appendices**: Provides additional information that is relevant to the SRS.

This structure ensures a clear and comprehensive exposition of the application's requirements and helps all stakeholders, including the development team and potential investors, to understand the functionalities and scope of the project effectively.

2. Overall Description

2.1 Product Perspective

The Stock Price Prediction App is a standalone application integrated with external financial data sources to provide real-time and historical stock data.

2.2 Product Functions

- Real-time stock data retrieval
- Historical data analysis
- Technical indicators computation
- Stock price forecasting

2.3 User Classes and Characteristics

The primary users of the application are financial analysts, retail investors, and educational institutions using the app for teaching financial concepts.

2.4 Operating Environment

The app will be web-based, accessible via any standard web browser and hosted on cloud infrastructure to ensure scalability and availability.

2.5 Design and Implementation Constraints

The app will be developed using Python and Streamlit, relying on public APIs for data, which may limit the data's availability or accuracy.

2.6 Assumptions and Dependencies

The successful operation depends on the continuous availability of stock price data from external APIs and the accuracy of the forecasting models used.

3. System Features

3.1 Stock Ticker Selection

- **3.1.1 Functional Requirements:**
    - FR3.1.1.1: The system must allow users to select a stock ticker from a dropdown list that includes a variety of commonly traded stock tickers.
    - FR3.1.1.2: The system should update dynamically to reflect the data related to the selected stock ticker.
- **3.1.2 Performance Requirements:**
    - PR3.1.2.1: The dropdown list for stock tickers must load within two seconds of initiating the application.

3.2 Data Fetching and Analysis

- **3.2.1 Functional Requirements:**
    - FR3.2.1.1: The system shall fetch historical stock data from external sources using the **yfinance** API upon user selection of a ticker.
    - FR3.2.1.2: The system must provide an error message if the selected stock ticker data is unavailable.
- **3.2.2 Security Requirements:**
    - SR3.2.2.1: Ensure that all data transfers from external sources are performed over secure channels (HTTPS).

3.3 Technical Indicators

- **3.3.1 Functional Requirements:**
    - FR3.3.1.1: The system must compute technical indicators such as EMA (Exponential Moving Average), MACD (Moving Average Convergence Divergence), and RSI (Relative Strength Index) for the fetched data.
    - FR3.3.1.2: Display these indicators in a comprehensible format on the user interface.

3.4 Forecasting Models

- **3.4.1 Functional Requirements:**
    - FR3.4.1.1: Implement the Prophet model to forecast stock prices for up to one year into the future based on historical data.
    - FR3.4.1.2: Implement an LSTM neural network to provide short-term stock price predictions.
- **3.4.2 Performance Requirements:**

    - PR3.4.2.1: The system must generate and display forecasts within 30 seconds after user requests.
    - PR3.4.2.2: The system must maintain accuracy levels as per predefined benchmarks during testing phases.

3.5 Visualization of Data and Predictions

- **3.5.1 Functional Requirements:**
    - FR3.5.1.1: Display the historical and predicted stock price data using interactive charts.
    - FR3.5.1.2: Visual representations should include options for plotting EMA fast, EMA slow, MACD, MACD signal, and RSI.
- **3.5.2 Usability Requirements:**
    - UR3.5.2.1: Ensure the interface is intuitive and the charts are easily interpretable by users with basic knowledge of stock markets.
    - UR3.5.2.2: Provide tooltips and help documents within the application to guide new users on the interpretation of the data.

4. External Interface Requirements

4.1 User Interfaces

- **UI4.1.1**: The application shall present a graphical user interface (GUI) accessible via web browsers like Chrome, Firefox, Safari, and Edge.
- **UI4.1.2**: The GUI shall display a dropdown menu for stock ticker selection, interactive charts for data visualization, and controls for initiating data fetching and predictions.
- **UI4.1.3**: The GUI must be responsive and adapt to both desktop and mobile screen sizes to ensure usability across different devices.

4.2 Software Interfaces

- **SI4.2.1**: The system must interface with the **yfinance** API to fetch historical stock data.
    - **Specification**: yfinance API, URL: https://finance.yahoo.com
    - **Data Interchange Format**: JSON
- **SI4.2.2**: The application will use the Prophet library for generating future stock price forecasts.
    - **Specification**: Prophet library, maintained by Facebook.
- **SI4.2.3**: The system shall use TensorFlow and Keras for constructing and training the LSTM model.

  - **Specification**: TensorFlow version 2.x, Keras version 2.3.x or newer.

4.3 Communications Interfaces

- **CI4.3.1**: All external communications shall use HTTPS to ensure data security during transmission.
- **CI4.3.2**: The application must comply with HTTP/1.1 standards for all web-based communications.
- **CI4.3.3**: Real-time data fetching from external APIs must be performed over secure network connections to protect user data and privacy.

4.4 Hardware Interfaces

- **HI4.4.1**: No specific hardware interfaces are required for user interaction beyond standard input devices (keyboard, mouse) and output devices (monitor).
- **HI4.4.2**: The application must be capable of running on server hardware that supports HTTPS protocol for secure data transmission.

Additional Considerations

- **Integration Details**: Include details on how external libraries and APIs are integrated into the application, specifying any required middleware or additional security layers.
- **Compatibility and Support**: Define the compatibility requirements with external systems, such as operating systems, database management systems, or third-party software that the application will interact with.
- **Error Handling and Logging**: Specify how the system will handle errors during interaction with external interfaces, including logging mechanisms and user notifications.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **PR5.1.1**: The application should be capable of handling up to 100 simultaneous users without performance degradation.
- **PR5.1.2**: Response times for fetching historical data from external sources should not exceed 5 seconds.
- **PR5.1.3**: Forecast computations should complete within 30 seconds from the initiation of the request by the user.
- **PR5.1.4**: The system should be available with a 99.9% uptime, excluding scheduled maintenance.

5.2 Security Requirements

- **SR5.2.1**: All data transmitted over the internet must be encrypted using SSL/TLS.

- **SR5.2.2**: User data should be stored securely, with sensitive information (e.g., browsing history) encrypted in transit and at rest.
- **SR5.2.3**: Implement standard authentication measures to protect user accounts, including password protection and optional two-factor authentication.
- **SR5.2.4**: The application must comply with applicable data protection regulations (e.g., GDPR, HIPAA) to ensure the privacy of user data.

5.3 Safety Requirements

- **SA5.3.1**: The system should have backup and recovery procedures in place to handle system failures without data loss.
- **SA5.3.2**: Regular security audits and updates should be performed to ensure the system is protected against new vulnerabilities.
- **SA5.3.3**: Emergency stop mechanisms or features should be accessible to administrators to halt operations in the event of a critical system failure.

5.4 Software Quality Attributes

- **SQA5.4.1**: **Reliability**: The system should perform consistently under defined conditions, maintaining a low rate of failures.
- **SQA5.4.2**: **Usability**: The application should have an intuitive interface that is easy to navigate, even for users with limited technical expertise.
- **SQA5.4.3**: **Maintainability**: The code should be well-documented and structured to facilitate updates and maintenance.
- **SQA5.4.4**: **Scalability**: The application should be designed to easily accommodate growth in users and data without a need for complete redesign.
- **SQA5.4.5**: **Interoperability**: The system should be able to operate with various external systems and technologies without restrictive dependencies.

6. Appendices

6.1 Glossary

- **G6.1.1**: **API (Application Programming Interface)**: A set of routines, protocols, and tools for building software applications, specifying how software components should interact.
- **G6.1.2**: **SSL/TLS (Secure Sockets Layer/Transport Layer Security)**: Protocols for establishing authenticated and encrypted links between networked computers.
- **G6.1.3**: **LSTM (Long Short-Term Memory)**: A type of recurrent neural network used in the field of deep learning.
- **G6.1.4**: **MACD (Moving Average Convergence Divergence)**: A trend-following momentum indicator that shows the relationship between two moving averages of a security’s price.

- **G6.1.5**: **RSI (Relative Strength Index)**: A momentum oscillator that measures the speed and change of price movements.

6.2 Analysis Models

- **G6.2.1**: Detailed architectural diagrams and data flow diagrams illustrating the application's structure.
- **G6.2.2**: Sequence diagrams and use case diagrams that detail functional operations and interactions within the system.

6.3 Issues and Solutions

- **G6.3.1**: A log of identified issues during the initial phases of system design and the corresponding solutions implemented.
- **G6.3.2**: Troubleshooting guidelines and corrective actions for common problems anticipated during the operation of the system.

6.4 Assumptions and Dependencies

- **G6.4.1**: List of all assumptions made during the system design and analysis phases.
- **G6.4.2**: Details of external dependencies that affect the system's design and functionality.

6.5 Reference Documents

- **G6.5.1**: Links to or copies of external documents referenced in the SRS.
- **G6.5.2**: Documentation on third-party tools and libraries used in the system, such as Streamlit, yfinance, Prophet, and TensorFlow.

6.6 Acronyms and Abbreviations

- **G6.6.1**: Comprehensive list of acronyms and abbreviations used throughout the SRS, along with their definitions.

Additional Notes

- **Note 1**: It's essential to ensure that all appendices are clearly referenced in the main body of the SRS. This helps in maintaining continuity and ease of understanding.
- **Note 2**: Appendices should be kept updated in line with any changes to the system requirements or project scope to ensure consistency across the documentation.