**1. Introduction**

### 1.1 Description

This document outlines the software requirements for the Company Stock Prediction Project. The project aims to develop a comprehensive system that leverages historical stock data and statistical models to predict future stock prices. It will provide actionable buy/sell suggestions and detailed visualizations to enhance decision-making for investors and traders.

**1.2 Problem Statement**

Investors and traders need accurate tools to predict stock prices and make informed decisions on buying and selling stocks. Traditional methods and tools often lack the integration of modern predictive models and real-time data analysis, resulting in suboptimal investment strategies. This project aims to develop a system that leverages historical stock data and advanced statistical models to predict future stock prices, thus providing users with actionable buy/sell suggestions and detailed visualizations to enhance their decision-making process.

**1.3 Scope**

The project covers the development of a full-stack application with the following features:

* Fetching and storing historical stock data from the Yahoo Finance API.
* Viewing top financial stocks
* Adding stocks to watchlist
* Providing stock details and visualizing stock data
* Predicting future stock prices of a company using the VAR model.
* Providing buy/sell suggestions based on predictions.
* Enabling user interaction through a web interface.

**2. Pre-requisites**

* Basic knowledge of stock market concepts and trading.
* Understanding of web development technologies (React, Node.js).
* Familiarity with Python and statistical modeling.
* Experience with databases, particularly MongoDB.

**3. Functional Requirements**

**3.1 Data Collection and Storage**

* Fetch historical stock data from Yahoo Finance API.
* Store the fetched data in MongoDB.

**3.2 Data Retrieval and Model Prediction**

* Retrieve historical stock data from MongoDB.
* Train the VAR model on the retrieved data.
* Predict future stock prices using the trained model.
* Store the predicted values in CSV files.

**3.3 User Interface**

* Develop a web interface using React for user interaction.
* Display top financial stocks and allow users to search for specific stocks.
* Visualize historical and predicted stock prices using React Highcharts.
* Allow users to add stocks to a watchlist.
* Provide a feature to toggle between dark and light themes.

**3.4 Buy/Sell Suggestions**

* Analyze predicted stock prices to provide buy/sell suggestions.
* Allow users to input parameters (ticker, initial capital, start date, end date) for personalized suggestions.

**4. Non-Functional Requirements**

* **Performance**: The system should handle multiple user requests efficiently.
* **Scalability**: The architecture should support easy scaling to accommodate more users and additional features.
* **Reliability**: Ensure high availability and minimal downtime.
* **Usability**: The user interface should be intuitive and easy to navigate.
* **Security**: Protect user data and ensure secure communication between the frontend and backend.
* **Maintainability**: Code should be well-documented and follow best practices for easy maintenance and updates.

**5. Solution Approach**

**5.1 Data Collection and Storage**

* **Data Source**: Fetch stock data from the Yahoo Finance (yfinance) API, including daily prices and other financial metrics.
* **Data Storage**: Store the fetched data in a MongoDB database, leveraging its flexibility and efficiency for handling large volumes of time-series data.

**5.2 Backend Development**

* **Data Retrieval**: Use Node.js for the backend to handle requests from the frontend and interact with the MongoDB database for data retrieval.
* **Model Prediction**: Host the VAR (Vector Autoregression) model using a Flask application to predict future stock prices. This modular approach keeps prediction logic separate and maintainable.
* **CSV Storage**: Store predicted stock prices in CSV files for easy integration and manipulation.

**5.3 Frontend Development**

* **React Application**: Develop the frontend using React to provide features like viewing top financial stocks, searching for stock details, and visualizing data.
* **Charts Visualization**: Utilize React Highcharts for rendering interactive stock price charts, enabling users to analyze historical and predicted prices visually.
* **User Interaction**: Allow users to input ticker symbols to view stock information, manage a watchlist, toggle themes, and receive buy/sell suggestions.

**5.4 Model Training and Prediction**

* **VAR Model**: Train the VAR model using historical stock data for each company individually, considering variables such as open, high, low, close, and volume for accurate predictions.
* **Prediction**: Predict future stock prices for each company and save the results in CSV files for further analysis and user interaction.

**5.5 Suggestion Feature**

* **Buy/Sell Recommendations**: Analyze predicted stock prices along with user inputs (initial capital, start date, end date) to provide actionable buy/sell suggestions aimed at maximizing profits.
* **CSV Integration**: Utilize the stored predicted values in CSV files to determine optimal buy/sell points and generate recommendations.