**Software Requirements Specification (SRS)**

**1. Introduction**

* **1.1 Purpose: To build a fully automated stock market trading app that predicts the highest weekly stock price and executes trades via the FYERS API, while also tracking and displaying performance on a public-facing website.**
* **1.2 Document Conventions: This document uses a numbered outline format for clarity. Key terms are bolded for emphasis. All dates are in YYYY-MM-DD format.**
* **1.3 Project Scope: This app will autonomously analyze NSE stock data, identify the stock with the highest predicted weekly gain (including brokerage fees), and automatically execute a trade after email authorization. A dashboard hosted on Vercel will track the profit and trade history. This is for a single user (myself) and is a self-contained system.**
* **1.4 References: The FYERS API documentation, jugaad-data Python library documentation, research papers on time-series forecasting models (e.g., LSTM), and this initial project brief.**

**2. Overall Description**

* **2.1 Product Perspective: This app is a standalone tool designed to automate a specific trading strategy. It is not integrated with other major financial platforms but relies on external APIs to function.**
* **2.2 Product Functions: The app will perform data fetching, machine learning prediction, trade execution, email authorization, and website data display.**
* **2.3 User Classes and Characteristics: The sole user is a single investor who needs an automated tool for a specific high-frequency trading strategy. The user has access to a dedicated email account for trade authorization.**
* **2.4 Constraints: The system must be hosted on an old laptop acting as a local cloud server. Data must be stored in a local SQL database. The trading platform is limited to the FYERS API. The deadline for completion is March 1, 2026.**

**3. Functional Requirements**

* **3.1 Data Ingestion:**
  + **The app must use the FYERS API and the jugaad-data library to fetch OHLC data for all stocks on the NSE.**
  + **The app must update this data daily to ensure it is current.**
  + **The system must store this data in a local SQL database for future use and portability.**
* **3.2 Machine Learning Model:**
  + **The system must train a time-series prediction model on the historical data.**
  + **The model must be able to predict the weekly high for all stocks.**
  + **The model's accuracy must be evaluated using metrics appropriate for high-value forecasting.**
* **3.3 Backend & Trading API:**
  + **The backend must identify the single stock with the highest predicted profit after factoring in FYERS brokerage fees.**
  + **The system must send an email for trade authorization to a pre-defined address.**
  + **Upon receiving a confirmation, the system must use the FYERS API to automatically execute a buy order, holding the position for one week.**
* **3.4 User Interface (UI):**
  + **The app must have a website dashboard hosted on Vercel.**
  + **The website must display the trade log, including buy/sell dates, prices, brokerage, and net profit.**
  + **The website must show the overall change in profit over time.**

**4. External Interface Requirements**

* **4.1 User Interface: The website on Vercel must be a simple, clean, and responsive dashboard.**
* **4.2 Hardware Interfaces: The app will be hosted on a dedicated laptop acting as a local server.**
* **4.3 Software Interfaces: The app will interact with the FYERS API for both data and trade execution, the jugaad-data Python library for data acquisition, a local SQL database (PostgreSQL) running on the dedicated laptop server, and an email API for authorization.**

**5. Non-Functional Requirements**

* **5.1 Performance: The entire prediction and trade authorization process must be completed in a timely manner to avoid missing trading opportunities.**
* **5.2 Security: All API keys and credentials for FYERS must be stored securely. The email authorization process must be robust to prevent unauthorized trades.**
* **5.3 Reliability: The app's trading process must include safeguards to prevent accidental double-trades or errors.**
* **5.4 Maintainability: The code must be modular to allow for easy updates to the trading strategy or the machine learning model.**
* **5.5 Portability: The entire system must be self-contained within the old laptop, which will act as the server.**