**StockFlux - Stock Trend Prediction Application**

**Developers:** Riddhi Chaplot (E23CSEU0425) & Akinchan Jain (E23CSEU0423)

**Application Description:** This application, built using Python and Streamlit, provides a predictive analysis and visualization of stock trends. Users can input stock symbols to view candlestick charts, predictive price trends, and RSI-based buy/sell signals for a given portfolio.

**Table of Contents**

1. **Overview of StockFlux**
2. **Unique Features**
3. **Application Walkthrough**
4. **Code Breakdown**
   * **CSS Styling for Enhanced UI**
   * **Stock Data Fetching and Prediction**
   * **Candlestick Chart Visualization**
   * **RSI Calculation and Portfolio Signal Visualization**
5. **Predictive Model Description**
6. **Appendix: Full Code**

**1. Overview of StockFlux**

StockFlux is a stock trend prediction tool that uses historical data to provide insights and recommendations. By analyzing stock data, the application presents past and predicted price trends using a Long Short-Term Memory (LSTM) model, empowering users with data-driven guidance for stock trading.

**2. Unique Features**

* **Interactive User Interface:** The application’s UI is styled with CSS for enhanced readability and interactivity, using bright, user-friendly colors and Times New Roman font.
* **Predictive Model for Price Forecasting:** Leveraging a pre-trained LSTM model, the application predicts stock trends based on historical data.
* **Candlestick Chart Visualization:** Users can visualize recent stock trends via interactive candlestick charts.
* **Portfolio RSI Signal Visualization:** The application computes the Relative Strength Index (RSI) and classifies stocks as "Overbought," "Oversold," or "Hold" based on RSI thresholds.

**3. Application Walkthrough**

1. **Stock Symbol Input**: Users enter stock symbols (e.g., "RELIANCE.NS") to analyze trends.
2. **Show Data and Prediction**: Button fetches data, visualizes trends, and predicts future prices using an LSTM model.
3. **Candlestick Chart**: Displays recent price movements in a candlestick chart format.
4. **Portfolio RSI and Signals**: Provides RSI-based insights for a portfolio, showing each stock’s current condition (overbought, oversold, or hold).

**4. Code Breakdown**

**CSS Styling for Enhanced UI**

This section defines custom CSS styling for Streamlit components:

* **Font Style**: The entire interface uses Times New Roman.
* **Title Styling**: Bright gold color for "StockFlux" and balanced red for author names.
* **Input Boxes and Buttons**: Styled for dark themes with bright color highlights.
* **Divider and Backgrounds**: Divider line for section separation, consistent with overall color scheme.

**Stock Data Fetching and Prediction**

1. **Stock Symbol Input**: Allows users to input a stock symbol (e.g., "RELIANCE.NS").
2. **Data Fetching**: Downloads historical stock data from Yahoo Finance.
3. **Data Visualization**:
   * **Closing Price**: A simple time series of the stock’s closing price.
   * **Moving Averages**: Adds 100-day and 200-day moving averages to analyze trends.
4. **Predictive Model**: Loads an LSTM model to predict future prices based on historical data.
5. **Result Plotting**: Compares actual vs. predicted prices in a chart.

**Candlestick Chart Visualization**

This section uses the Plotly library to create candlestick charts of stocks.

* **User Input**: Enter multiple stock symbols for batch processing.
* **Date Range**: Last 50 trading days.
* **Candlestick Charts**: Interactive charts showing open, high, low, and close prices.

**RSI Calculation and Portfolio Signal Visualization**

The RSI (Relative Strength Index) is calculated based on a rolling window, with signals to indicate:

* **Overbought (RSI > 70)**: Suggested as a potential sell.
* **Oversold (RSI < 30)**: Suggested as a potential buy.
* **Hold**: RSI between 30 and 70.

The code snippet below demonstrates the RSI calculation and portfolio visualization.

**. Predictive Model Description**

The application uses a Long Short-Term Memory (LSTM) neural network model:

* **Training**: Trained on a large historical dataset for stock price prediction.
* **Implementation**: Loads a pre-trained Keras LSTM model saved as keras\_model.keras.
* **Scaling**: Data is normalized with MinMaxScaler to ensure efficient LSTM training and prediction.

**6. Appendix: Full Code**

**Note**: For best readability, view the full code within a Python IDE or copy it to a .py file. Ensure that all required libraries (Streamlit, yfinance, Keras, sklearn, and Plotly) are installed to run the code successfully.

This document provides a comprehensive guide to StockFlux, explaining its structure, functionalities, and how each part of the code contributes to the final application.