**Description of the Stock Prediction and Portfolio Management API**

This application provides two main functionalities: **stock price prediction** and **portfolio optimization**. It leverages machine learning and financial data analysis techniques to deliver insights.

**Stock Price Prediction**

The stock prediction feature uses a **Recurrent Neural Network (RNN)** model, specifically an **LSTM (Long Short-Term Memory)** network, to predict the next day's closing price of a given stock.

1. **Data Collection:**
   * Fetches the past one year of stock price data using the yfinance library.
   * Uses the "Close" prices to form a time series dataset.
2. **Preprocessing:**
   * A sliding window approach is applied, converting raw stock prices into a format suitable for supervised learning. Each data point consists of n previous prices as input and the next price as the target.
3. **Model Training:**
   * The LSTM model, which is well-suited for sequential data, is trained using this windowed dataset.
   * The model is structured with:
     + An LSTM layer to capture temporal dependencies.
     + Dense layers for further processing.
     + Mean Squared Error (MSE) as the loss function and Adam optimizer for training.
4. **Prediction:**
   * After training, the model predicts the next day’s closing price using the most recent n days of stock prices.
   * Predictions are stored in memory (predictions\_dict) for quick retrieval.
5. **API Endpoints:**
   * GET /prediction/<symbol>: Returns the predicted price for the given stock symbol. If a prediction is not available, it computes one using the above process.
   * GET /clear\_predictions: Clears all stored predictions.

**Portfolio Optimization**

This feature computes the optimal allocation of weights for a portfolio of stocks to maximize the **Sharpe Ratio**, a measure of risk-adjusted return.

1. **Data Collection:**
   * Fetches historical adjusted closing prices for the specified stock tickers using yfinance.
   * Calculates daily log returns based on the price data.
2. **Risk-Free Rate:**
   * Retrieves the 10-year U.S. Treasury rate from the FRED API to estimate the risk-free rate.
3. **Optimization Setup:**
   * Defines a portfolio optimization problem where:
     + The objective is to maximize the Sharpe Ratio.
     + Constraints ensure the sum of weights equals 1.
     + Each stock weight is capped at 50% to encourage diversification.
4. **Optimization Execution:**
   * Uses the scipy.optimize.minimize function with the SLSQP method to compute the optimal weights.
5. **API Endpoint:**
   * GET /calculate\_portfolio\_weights/<tickers>: Accepts a comma-separated list of stock tickers and returns:
     + The optimal allocation of weights for the portfolio.
     + The calculated Sharpe Ratio.
     + The risk-free rate used in the calculation.