Project Plan – Stock Market Simulator

User Case Description

The Stock Market Simulator is designed for beginners in stock trading who want to practice with real-world data. Users can manage a portfolio, buy and sell stocks using market and limit orders, and view stock performance through candlestick charts over different timeframes (day, month, year, hour). The system allows multiple users to log in with username and password to manage their own portfolios. The purpose is to help users learn stock trading strategies before using real money.

Class List

1. **User**: Manages user data (username, password) and tracks portfolios.
1. **Portfolio**: Manages a collection of stocks that a user owns and their buying power.
2. **Stock**: Represents different type of stocks, including the current price and ticker symbol.
3. **Order**: Handles the two types of orders (market and limit), including price and quantity.
4. **MarketData**: Retrieves and stores real-world stock market data for use in the simulator.
5. **Graph**: Manages the display of stock performance in candlestick charts.
6. **TransactionHistory**: Tracks the user's previous buy/sell actions.

Data and Function Members

1. **User:**
   Attributes:
   Username: string
   Password: string
   portfolio: Portfolio – manages the user’s stock.
   Function:
   login () – user authenticate.
   createAccount() – Creates a new account.
   getPortfolo() – Retrieves the user’s portfolio.
   Logout() – logs the user out.

2. **Portfolio:**
   Attributes:
   Stocks: vector<Stock> - collection of stock owned by the user
   BuyingPower: double – the money available for buying.
   Functions:
   addStock(Stock stock) – Add a stock to the portfolio.
   RemoveStock(string tickerSymbol) – Remove a stock from the portfolio by symbol.
   updateStockPrice(string tickerSymbol, double newPrice) – Updates the price of a stock in the portfolio.
   viewPortfolio() – Display the stock in portfolio.

3. **Stock**
   Attributes:
   tickerSymbol: string
   companyName: string
   currentPrice: double
   quantityOwned: int
   stockType: string

   Functions:
   getPrice() – returns the current price of the stock.
   getDetails() – returns details like the company name, stock type and ticker symbol.

BuyStock(int quantity) – buys a specific quantity of the stock.
sellStock(int quantity) – sell a specific quantity of the stock.
getType() – returns the type of stock.

4. **Order:**
   Attributes:
   orderType: string – limit or market.
   stock: Stock
   price: double – for limit orders.
   quantity: int
   Functions:
   executeMarketOrder() – Buys/sells the stock at current price.
   executeLimitOrder() – Execute the order when the stock price on a specific price.

5. **MarketData:**
   Attributes:
   apiEndpoint: string
   stockData: map<string, Stock> - stock data retrieved from API.
   Functions:
   fetchStockData(string, tickerSymbol) – retrieves real world data for a single stock.
   updateStockPrice(string tickerSymbol) – updates the price of the stock in the system.

6. **Graph:**
   Attributes:
   stockPrices: vector<double> - price history for candlestick chart
   timeFrama: string – day, month, years….
   Function:
   plotCandleStickChart(Stock stock) – display a candlestick chart for the stock.
   viewByTimeframe(string timeframe): Show the chart data by day/month/year.

7. **TransactionHistory:**
   Attributes:
   transactions: vector<string> - list of all the past transactions
   Function:
   recordTransaction(string transactionDetails) – records detail of transaction.
   viewHistory() - display the user's past transactions.

Relationship between Classes:

- **User** has **Portfolio.**
- **Portfolio** has different **Stock** object.
- **Order** interacts with **Stock** to executes trading.
- **MarketData** provides live data to the **Stock** and **Graph.**
- **Graph** displays data provided by **MarketData.**
- **TransactionHistory** logs all action act by **Order.**

Project Task List and Timeline:

1. Design class structure (3 days). Hau Leung Lai
2. Implement User and Portfolio classes (4 days): Quantai Li
3. Implement Stock and Order classes (4 days): Yi Jiang
4. Implement MarketData class (API integration) (5 days): Everyone
5. Implement Graph class (candlestick chart) (5 days): Hau Leung Lai
6. Build user interface (login, buy/sell function**)** (5 days): Hau Leung Lai
7. Test and debug each module (3 days): Everyone
8. Finalize project and documentation (3 days): Everyone

User Interaction Description:

The program will be accessed through a terminal or graphical interface by the user. The user will login with the username and password to access his portfolio. They are allowed to execute buying or selling of the stock either via keyboard commands or mouse clicks. They can also view stock performance in different time frames using candlestick charts, going from days to months or years. Also, user will see a list of different stock types, such as Technology, Healthcare or Energy, and can navigate through them. By clicking on a specific stock(keyboard / mouse), users are able to see the stock's detail, such as the current price, type, and company name. After every trade, confirmation will appear or maybe some error messages for invalid input will be shown. Those function can let

Unit Testing and Debugging Plan:

The project will follow a unit testing to test each class and function to verify correctness, each class and function will be test individually. The tests will focus on:

- API integration for real-time stock data.
- Candlestick chart plotting.
- login/logout function.
- Execution of buy/sell function.

Also, the debugging plan will focus on catching invalid trades, invalid input and make sure the timeline function display correctly.