CLI Stock Trading System

https://github.com/kechavious/OOPFinalProject

Group Members

Yifei (Zora) Zhao (yz9704)

Muyao (Jerry) Kong (mk9014)

Gordon (Kangzheng) Zou (kz2538)

Date of Submission:

Thursday, August 14, 2025 (GMT+8)
Time in Beijing/Hong Kong

Thursday, August 14, 2025 Eastern Time (ET)

Table of Work

(Please write x in the boxes to mention what each student achieved in this project)

	Yifei Zhao	Muyao (Jerry) Kong	Gordon Zou
Project Description	X	X	X
Uses Cases Diagram(s) and description	X	X	
Sequence Diagrams	X	X	X
Class diagram(s)	X	X	X
Implementation	X	X	X
Conclusion		X	X

Table of Contents

System Analysis

Project Description

General description:
The Command-line stock trading system allows users to make their stock investment in the secondary market at an ease. The program allows users to select possible stocks, place stock trade(s) and view their investment summary/portfolio. The system gives the stock value each day during the investment period and performs automatic profit/loss calculation at the end of the whole term. In addition, we have introduced a novel feature: **Risk Warning**. Larger investment with no hedging usually means greater returns on a certain stock, but that also implies that the investor would face a significantly higher risk compared to other diversified portfolios. When a user makes a huge, or sometimes irrational investment decision (such as putting money in merely one particular stock for over a million dollars), the program will display a warning message. We hope that with this functionality, our users will be smarter and less prone to the loss of assets.

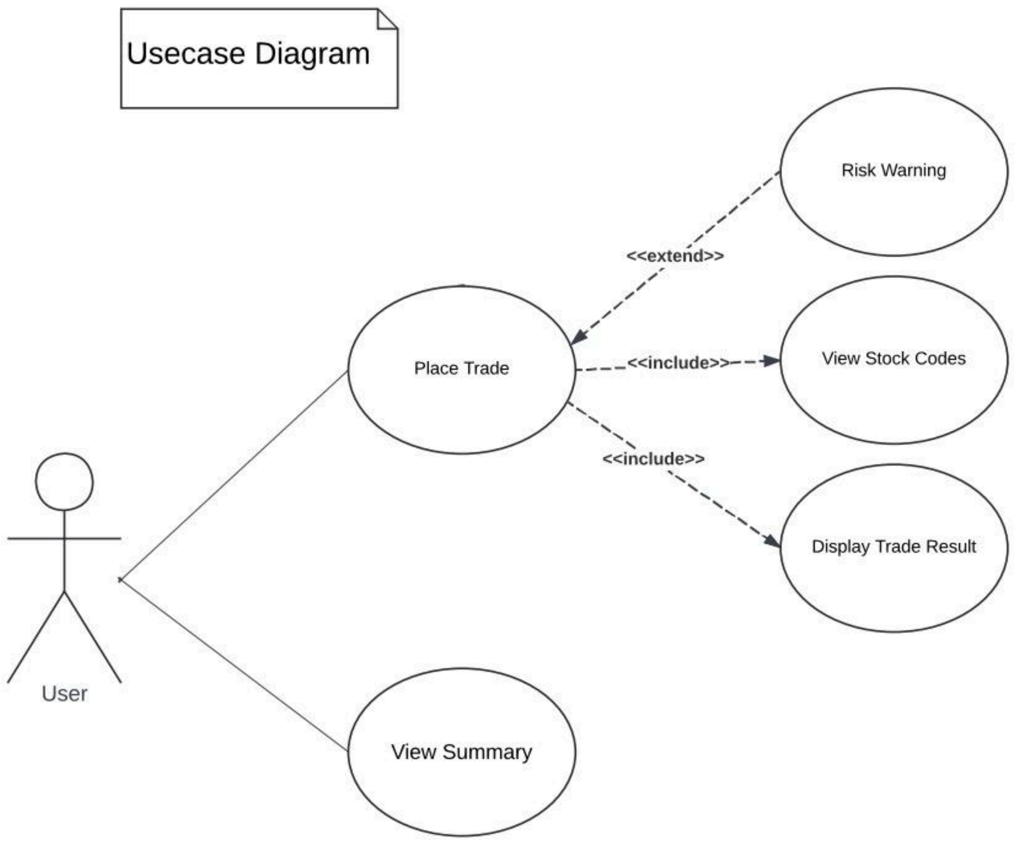
To enhance user experience, we want to help users navigate through the jungle of possible stocks to invest by sorting stock codes (AAPL, AMZN, TSLA for example) alphabetically. In a growing market as of today, every day there are thousands of new stocks being listed in the secondary market. The sorting feature resolves this predicament: our users could find their desirable stock based alphabetically, possibly saving the hassle of "searching through the mess".

Goals
The system aims to provide users with the most accurate and full-sided view of their stock investment with the minimalist command-line design. Users are able to place trades, view and manage their portfolios, and monitor daily stock value changes with automated profit/loss calculations. It supports multiple trades and provides a total summary of all transactions. The system displays available stock codes in alphabetical order before placing a trade, and shows a risk warning when the investment amount exceeds a defined threshold. Making the program easy to run and when users used the program, we did a good job for making the User Menu simplicity and efficiency. We also want to **improve communication skills** when three people have different tasks and how we should advance our work together.

Benefits:
- Educational Value: Providing a safe environment for the people who do not understand the stock trading system for learning the fundamentals of stock trading, risk assessment and profit/loss calculation.
- Monetary Value: Providing a safe platform for a myriad of possible investors. With a clear command-line user interface, investors will be able to place trades, view portfolios (view trade summary), and navigate through stocks at an ease.
- Simplicity and Efficiency: No external API integration that ensures easy setup and fast execution on any JAVA supported systems.
- The design system: What we do is the modular class structure allows future expansion to include advanced features such as real-time data, historical record and portable analysis.

Special Requirement:
To keep everything simple and easy to implement/test, instead of importing external APIs or real-time stock data, we use java.lang.Math.random() to simulate stock prices at the time of buying and selling. To make the simulated data more realistic, we limited the random() function to generate the percentage of fluctuation no higher than 15% of the price of the last day.

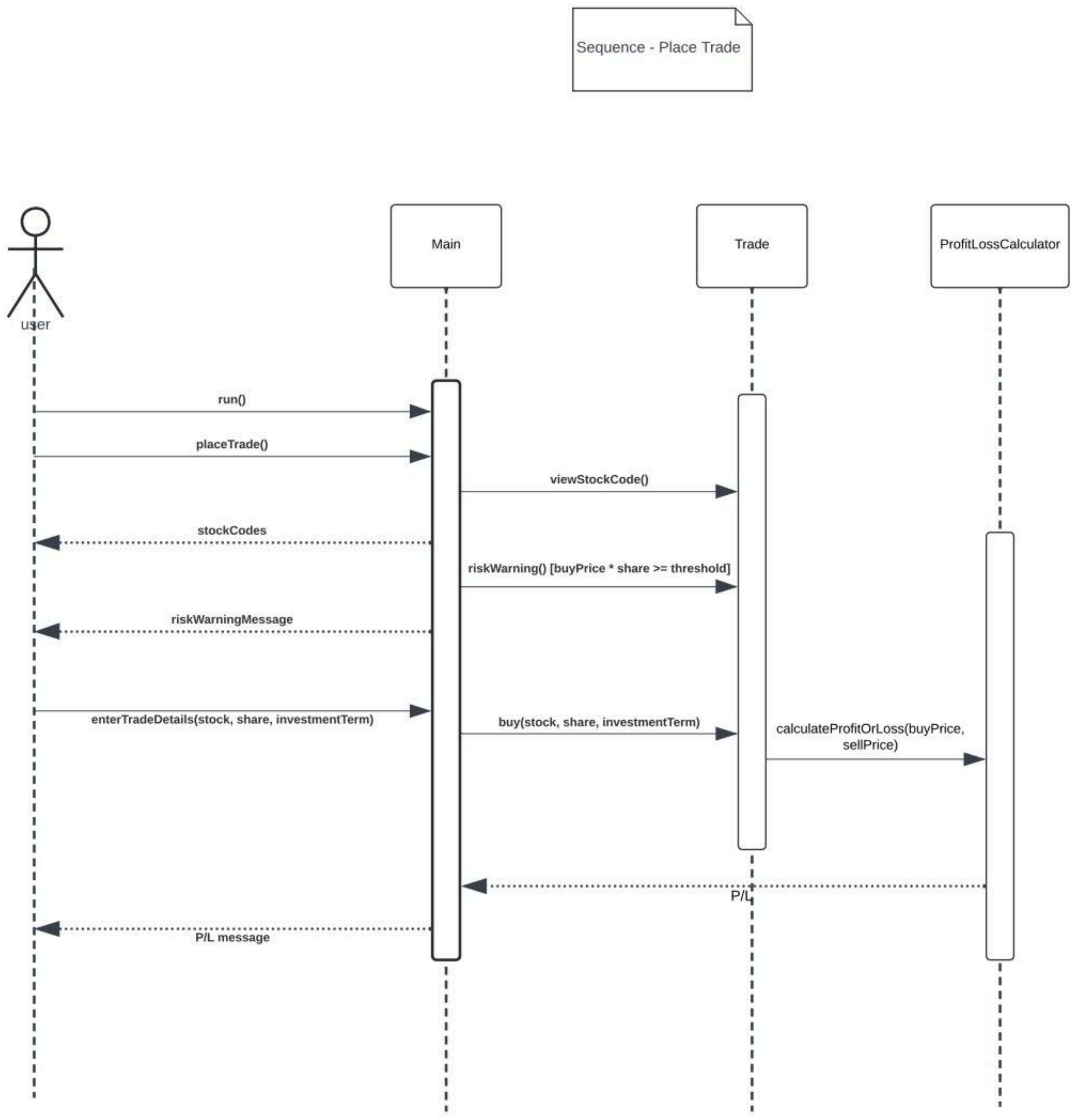
Uses Cases Diagram(s) and use cases description

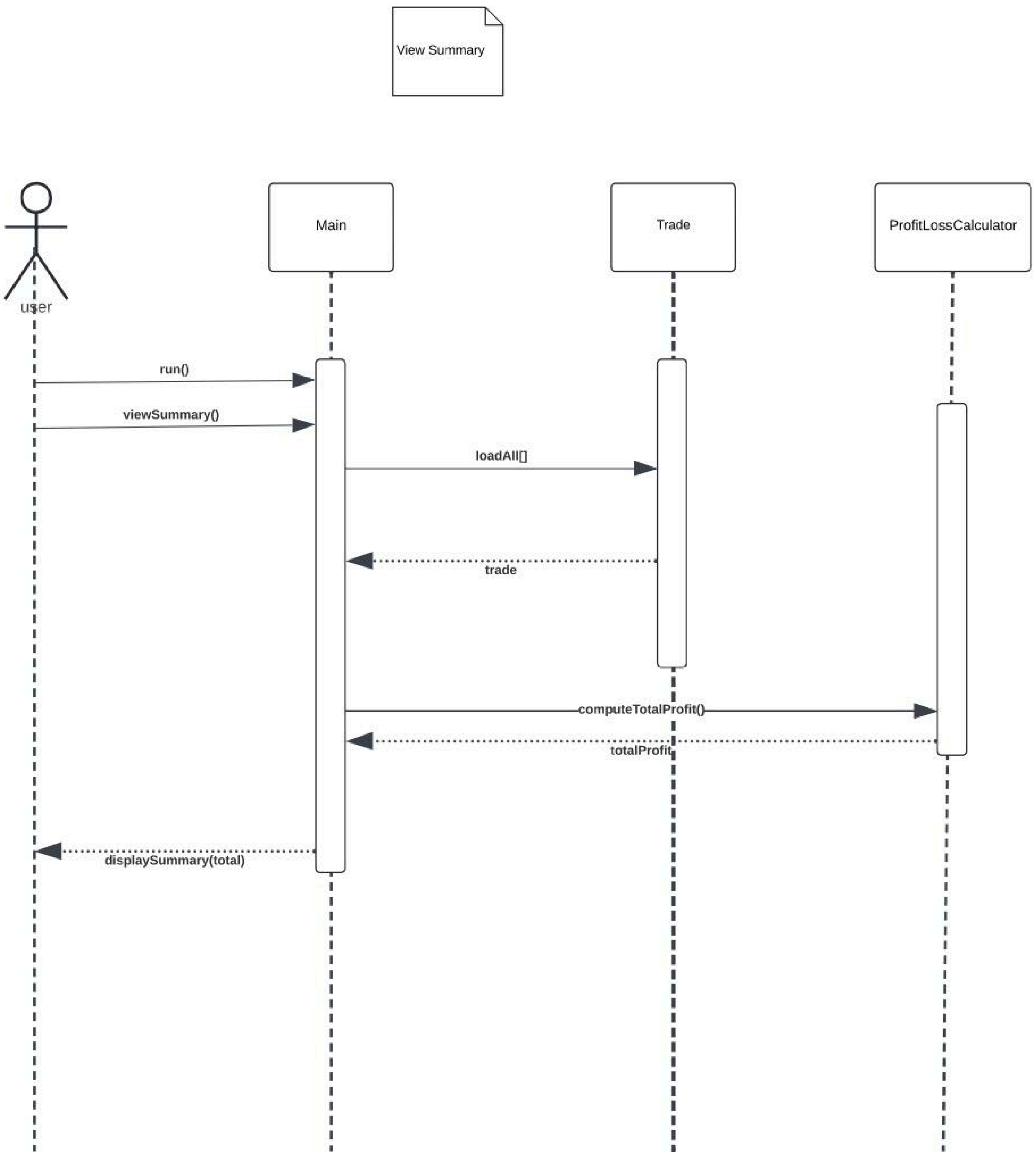


UC Reference name	Actors	Related Usecase	Overview
Place Trade	User	View Stock Codes, Display Trade Result, Risk Warning	User places a trade by selecting a stock code, entering shares and term. System shows codes, calculates P/L, and may show a risk warning if the trade amount exceeds the threshold.
View Stock Codes	User	–	Display available stock codes in alphabetical order for user selection.
Display Trade Result	User	–	Show calculated profit or loss for the placed trade.
Risk Warning	User	–	Display a caution message if the trade amount meets or exceeds the risk threshold.
View Summary	User	–	Show all trades made so far and the total profit/loss.

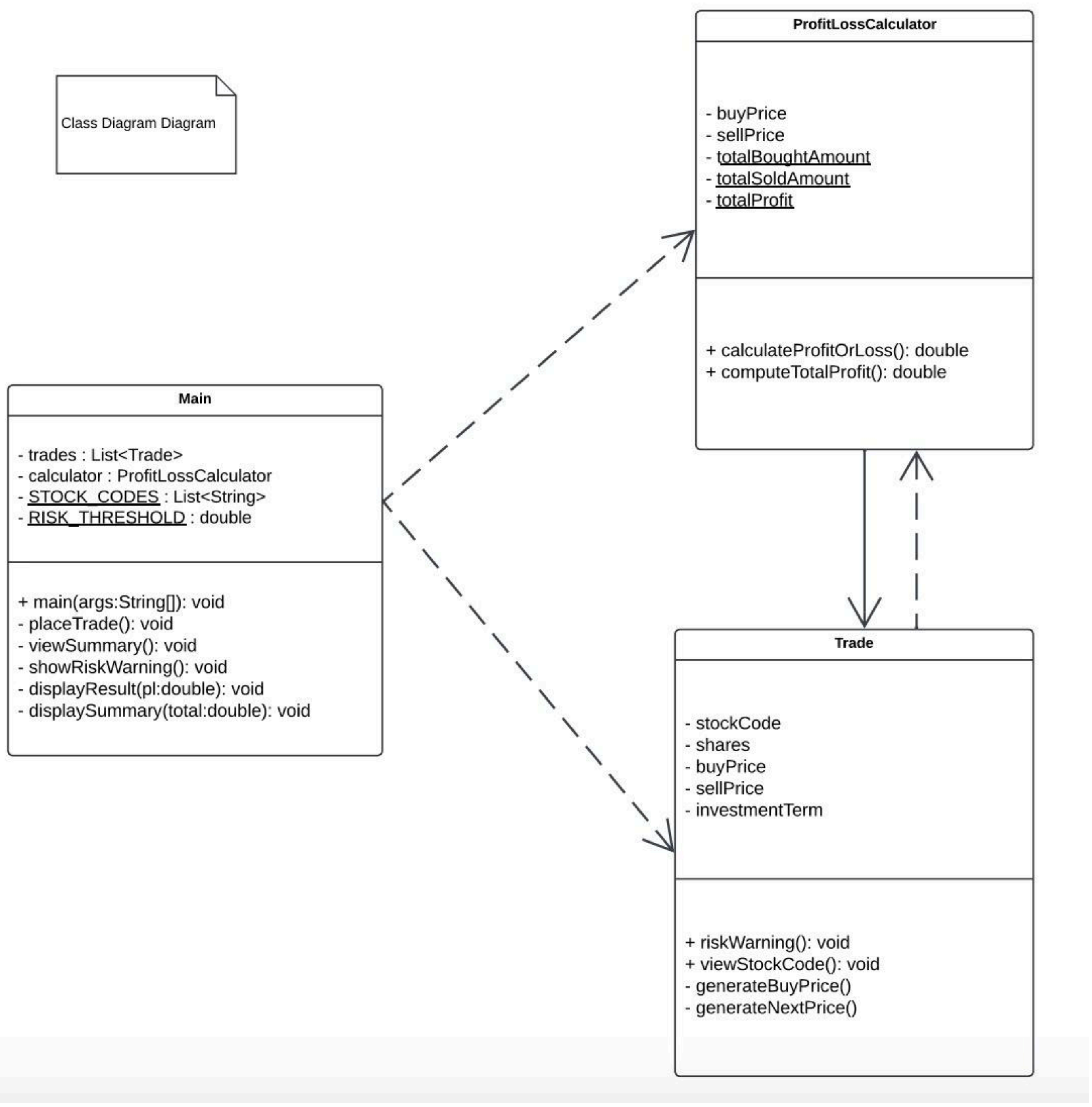
System Design

Sequence Diagram and Sequence Description: Place Trade


Sequence - Place Trade
user
Main
Trade
ProfitLossCalculator
run()
placeTrade()
viewStockCode()
stockCodes
riskWarning() [buyPrice * share >= threshold]
riskWarningMessage
enterTradeDetails(stock, share, investmentTerm)
buy(stock, share, investmentTerm)
calculateProfitOrLoss(buyPrice, sellPrice)
P/L
P/L message


View Summary
user
Main
Trade
ProfitLossCalculator
run()
viewSummary()
loadAll[]
trade
computeTotalProfit()
totalProfit
displaySummary(total)

Class Diagram



Conclusion

Throughout the project, our team spent approximately two weeks developing the system and shaping it into a realistic simulation of a stock trading application. We adopted a **divide-and-conquer** approach, breaking the large task into smaller, manageable components so that each member could actively contribute to both coding and diagram design.

For the diagramming phase, we utilized the **Lucid UML Platform**. Since there were three members in our team, each person selected their preferred diagram to create, ensuring balanced participation and efficiency. This phase took roughly three days to complete.

After finalizing the diagrams, we transitioned to the coding phase, where each member was responsible for one core component of the system:

- **Main.java** – Handling overall program flow, user interaction, and execution logic.
- **ProfitLossCalculator.java** – Computing and aggregating profits and losses across trades.
- **Trade.java** – Managing trade data and encapsulating related calculations.

Upon completing the individual coding tasks, we conducted group reviews to discuss and refine the overall design, improving efficiency and code quality. Finally, we collaborated on the **commenting phase**, adhering to standard Java documentation conventions. We ensured that each class and method included clear explanations of its purpose, functionality, and intended usage.

Working Summary:

- Successfully designed and proposed a **CLI Stock Trading System** implemented by **Java** and more focusing on **simulation basic trading operations** without relying on external APIs.
- Implemented key features such as **trade entry, profit/loss calculation, multitrading support** and a **total summary display**
- Integrated additional usability features including **Risk Level Indicator** for large investment and **Alphabetical Support** of stock codes before placing trades.
- Ensured the **system remained easy** to implement by us and to **test by using Math.random()** from Lang package of the Java library **for price simulation**.
-

Difficulties:

- We deciding the balance between **ensuring realism** and **simplicity in stock simulation** and given the **absence of real- market data**
- Handling multiple trades while ensuring **accuracy in cumulative profi/loss calculations**.
- Designing an **intuitive CLI interface** that remains user-friendly despite the lack of a graphical interface.

Recommendations:

- In the future iterations and future programming, we should consider adding data persistence such as **CSV** or **SQL database** to **keep historical trade records** and **make it more real** than the project we have done.
- Enhances the **risk prediction by adding market volatility factors**, sort the stock risk factors based on indicators such as maximum retracement rate, sharp-ratio or sector base performance variations.
- And if project scope allows in the future work projects, just integrate **optional real-time API** support for actual **stock data retrieval** while retaining the **simulation mode** for offline use.
- Adding a **new system design** for the **prototype** would be closer to the **real life stock trading system**.

Potential enhancements could include **integrating real-time market data, enabling order history tracking, supporting portfolio management, and implementing advanced analytics.**

- Splitting the task should be more **efficient** within the **communication skills**.
- To strengthen **data security** and **privacy,** all **trade records** should be stored in **encrypted form**, ensuring that **sensitive financial information** is protected from **unauthorized access** or **potential data breaches.**