

Software Design Document (SDD)

Virtual Stock Market Application

Team 9: Christopher McCracken, Darren Gabrido, Louis Brian Santos, Tai Martinez

1. Overview

The virtual stock market application is a functioning desktop application that simulates an environment where users can practice and learn how to partake in trading in the U.S. Stock Market. This application will not address actual transactions of stock buying or selling with real money but rather allows the user to mimic investing without any actual stake. The desktop application shall use an U.S. Stock Market API to implement real time online stock data to track the user's gains and losses per stock of companies. Users will be able to continually update and keep track of their trading on a portfolio linked to their account through the application's use of an online database. Users shall also be able to create multiple portfolios on a single account to help comprehend how certain trading factors will affect the user's money in the application. Under those portfolio, users are able to view their net gains and net losses per stock and the net gains and net losses per portfolio. This scope will help the user to learn the dynamics of the U.S. Stock Market by familiarizing themselves with how trading is completed.

2. Stakeholders

1. Client

Role: Provide developers with clear requirements, review project and provide feedback, review final deliverables.

Concerns: Lack of documentation, project doesn't start on time

2. Developers

Role: Designing, testing, and documenting the creation of an application.

Concerns: Unrealistic deadlines and requested specifications by the customer, changes to the scope of the project

3. User/Customer

Role: Use of the final functioning desktop application i.e. trading in the virtual stock market

Concerns: Difficult to understand and messy navigation of user interface, glitches/bugs in the application, long loading time for requested data

4. U.S. Stock Market Database(World Trading Data API)

Role: Supply the desktop application with current and updated U.S. Stock Market data for various companies

Concerns: Data is not continually updated (not live) and the database is down.

3. System Architecture

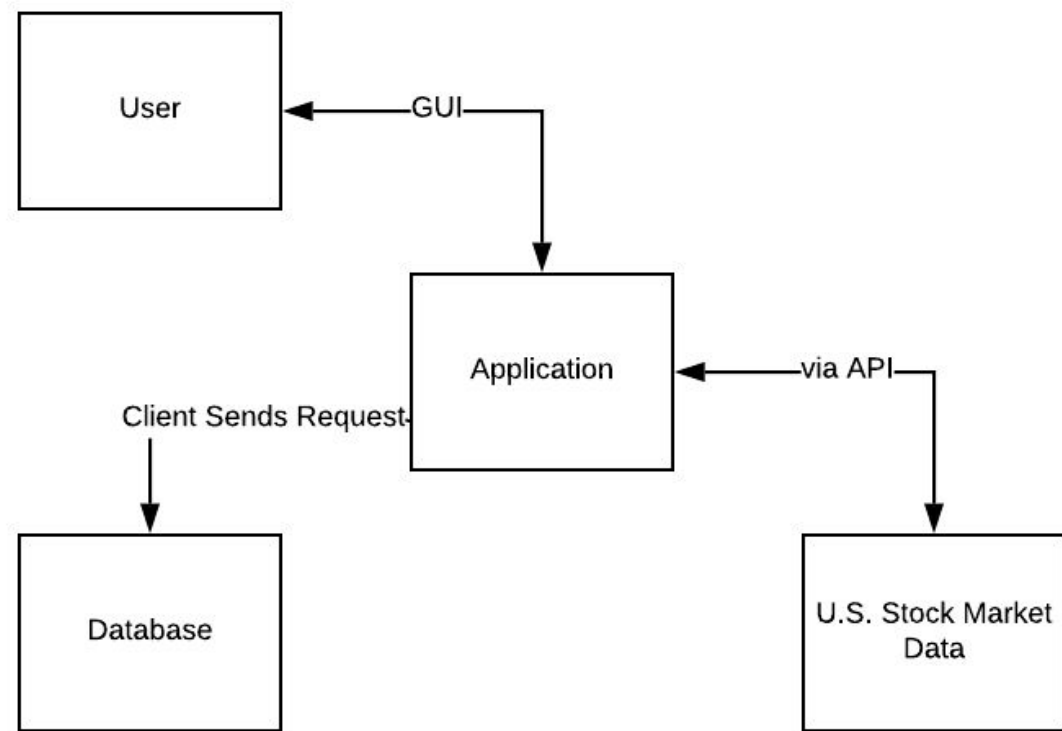


Diagram 1: Three-Tier Architecture

As shown above, the system architecture will be a three- tier architecture composed of the user, the actual desktop application, the database for user data, and the live U.S. Stock Market data. Initially, the user will directly access the application and the application will display all data to the user. The application will also access an online database for user data. The purpose of this database is so that users are able to login to the application and continually view how their stocks have developed over time. Users will login through the application which will then connect to the database to pull the user's data and display it as it continually updates through the users usage of the application. The application also communicates with the U.S. stock market data through the World Trading Data¹ API to have live stock market data that is accessible to the user through the application. The application will continually update the numbers from the API in the stock market for accurate trading results for the user.

4. Detailed Design

4.1 Logical Viewpoint

Class: Portfolio
-name: String -portfolios: List{}
+CurrentPortInfo(name: string): string

Class: Main
-portfolioName: string -tickerPrice: float -ticker: string -purchaseCount: int -amountSpent: float -tickers: Array[1...20] -newPortfolio: Portfolio
+buySellStocks(portName: string): void +promptForTickers(): tickers: string +getPortfolioInfo(name: string): PortfolioName: string +createPortfolio(name: string): newPortfolio: Portfolio

Class: ReadAPIData
-stockToRequest: String -tickers: String -stockName: String -stockSymbol: String -stockPrice: Float -lastUpdateTime: String
+GrabData(tickers) +ConvertToJSON(stockRequestJSON: string):stockRequestDict: dictionary

Class: Database
-userName: String -password: String
+AccessDB(): void +FindData(name: string, password: string): data: string

Class: DisplayGUI
-PortfolioName: string
+buySellStocks(): void +createPortfolioWindow(): void +viewPortfolioWindow(): void +setupUI(app: Application): void +inputDialog(): void +main(): void

Class Description:

Portfolio: Creates the general portfolio structure for the user.

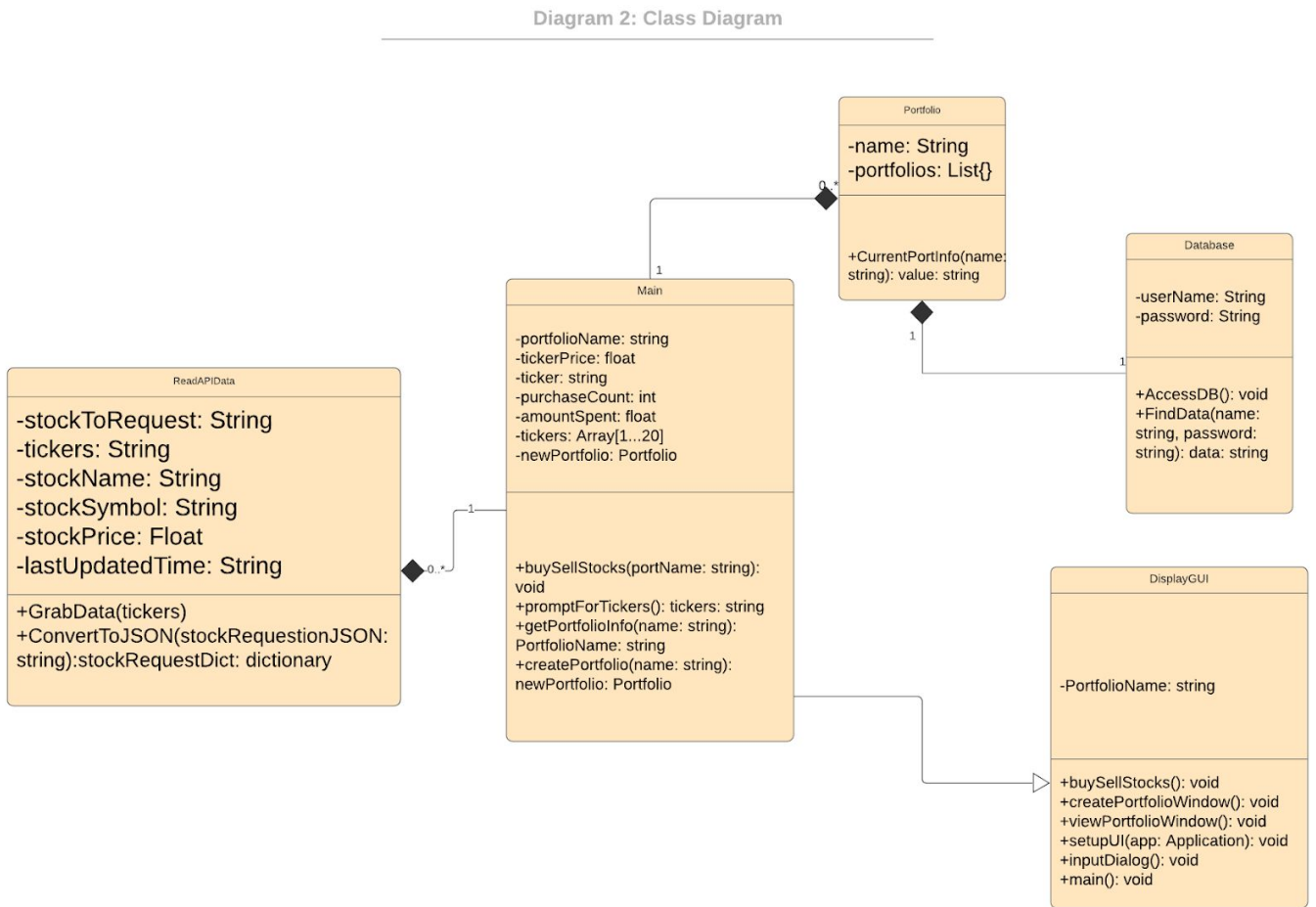
Main: Collects portfolio information and prompts the user for which company they would like to trade for. It then processes user actions to buy or sell that stock. Obtain the user's Portfolio from the Portfolio class and gets stock market data from the ReadAPIData class.

ReadAPIData: Performs a request to grab specific ticker data from API. This data is then converted from JSON text to a Python Dictionary to allow for data to be more easily managed. This data grabbed holds all specific stock information requested.

Database: Allows the user to log into their database which stores the user data for their trades. The specific database will be accessed through a username and password.

DisplayGUI: Visual representation of application actions and user's data which will be done done through PyQt². It imports all the data from the Main class to display it.

Class Diagram (Diagram2):



4.2 Interaction Viewpoint

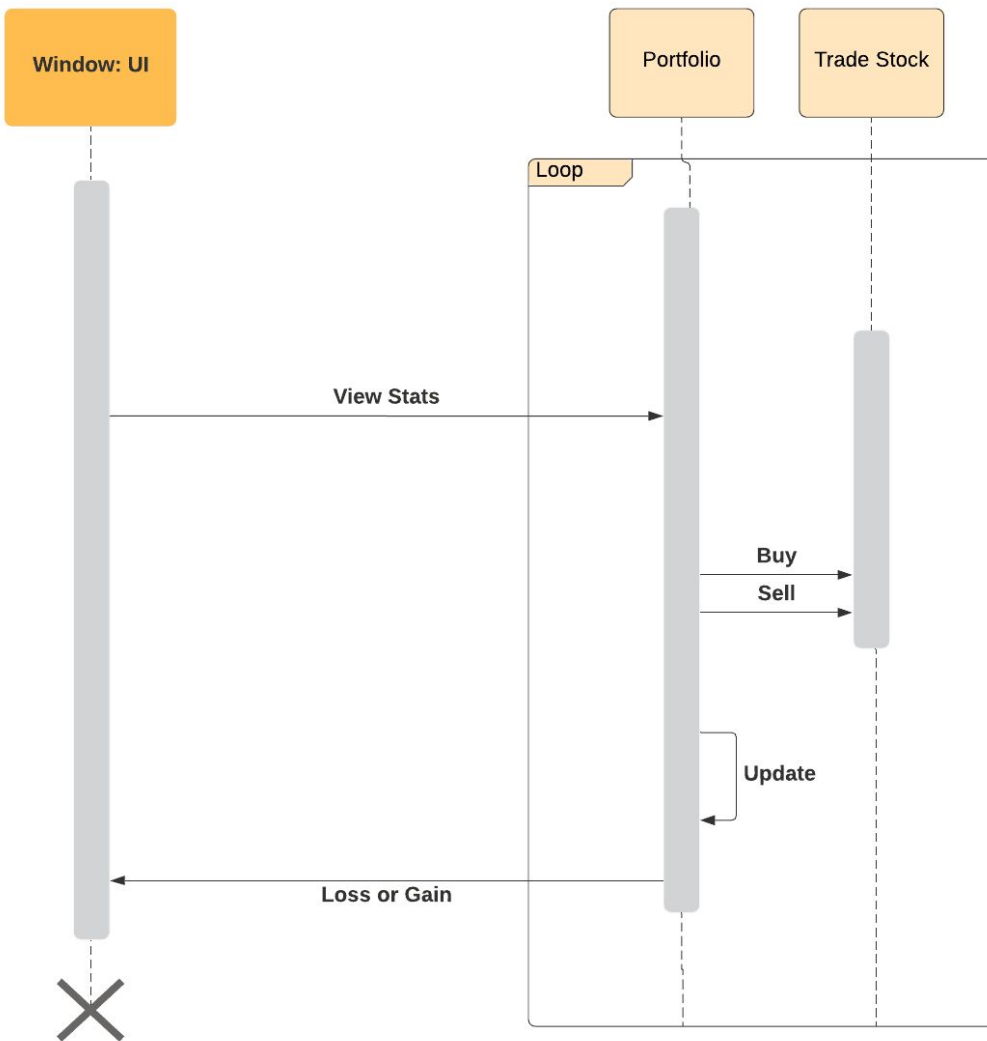


Diagram 3: General Sequence Diagram

The sequence diagram above shows the general sequence for the overall desktop application. Users will interact with the UI to view their trading statistics in their portfolio. The portfolio continually will update as the user trades stock by either buying or selling that stock. The UI will then display the losses or gains made by that user.

Use Case: User Login

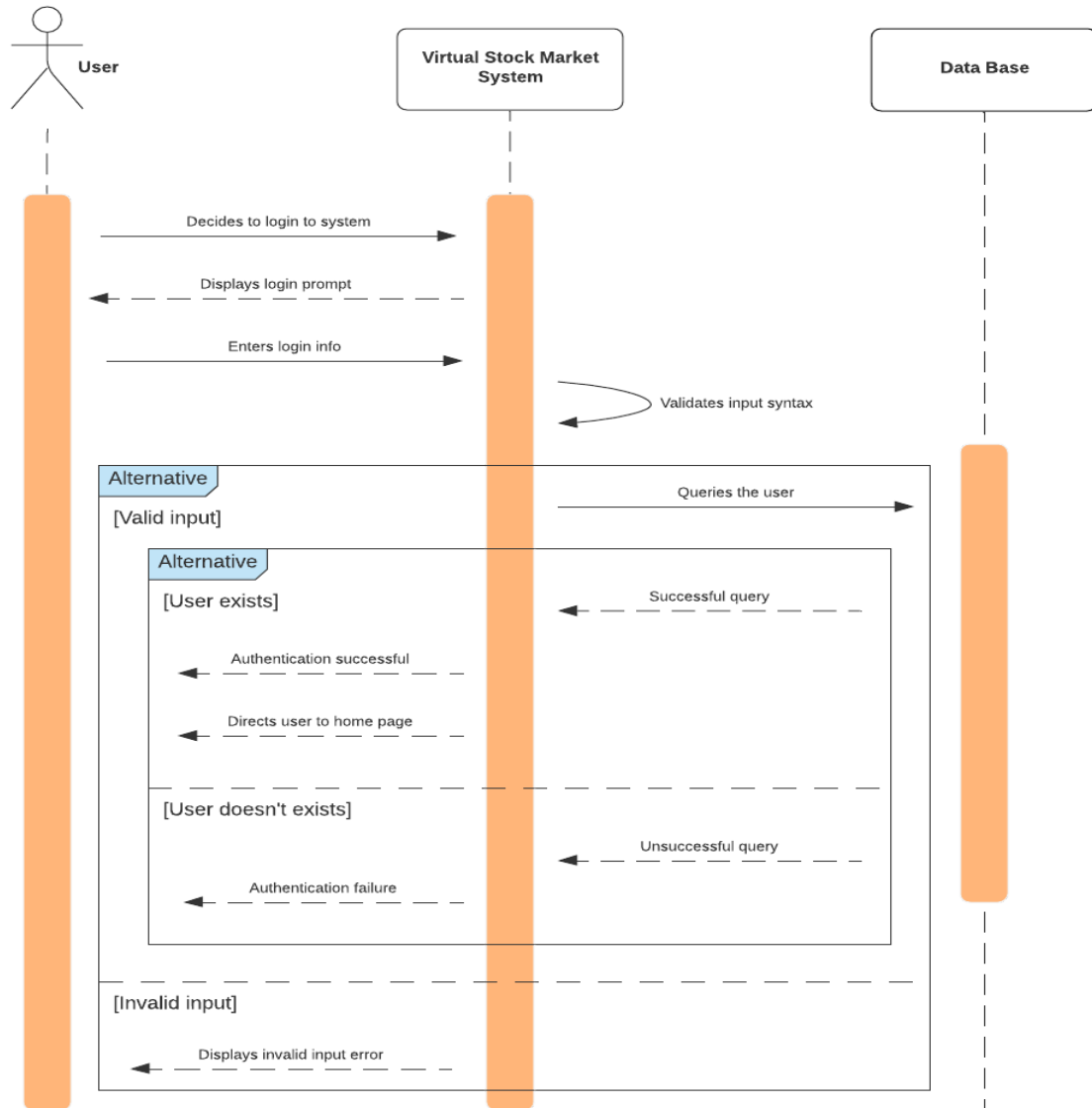


Diagram 4.a: User Login Use Case Sequence Diagram

This user login sequence diagram shown above, displays a user logging into the virtual stock market system. The user opts to login into the system. The system displays the login prompt to the user. The user enters the login information and the system validates the user's input syntax. If the input syntax is valid, the system queries the user's details from the database. If the user exists, the authentication is successful and the user is directed to the system's homepage. If the user doesn't exist, the user's authentication has failed and is displayed to the user. If the user's input syntax is not valid, the system displays an invalid input error message.

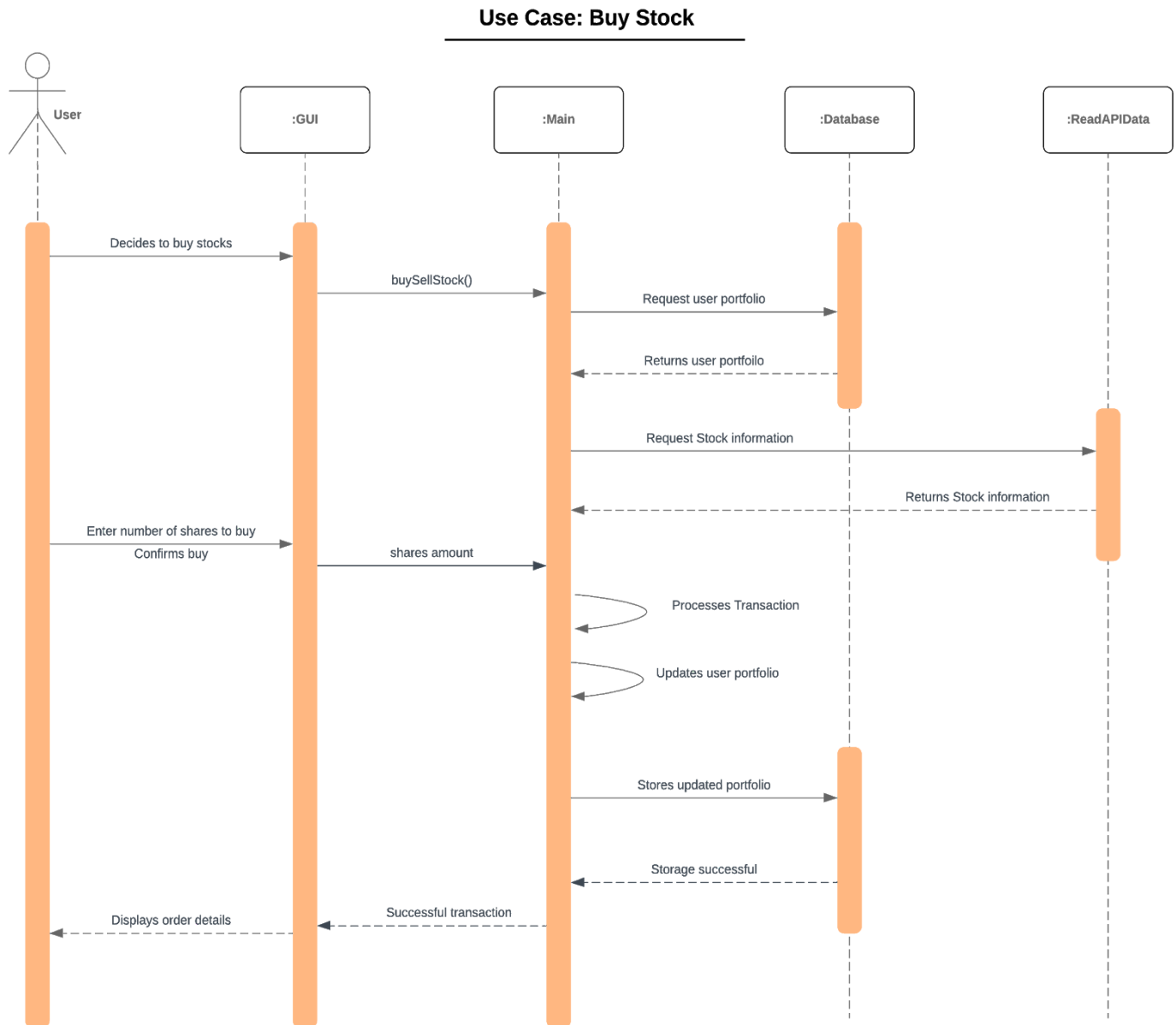


Diagram 4.b: Buy Stock Use Case Sequence Diagram

This buy stock sequence diagram shown above, displays the purchase of a stock from the virtual stock market by a user. The user opts to purchase a company's stock. The system retrieves the user's portfolio from the database and the company's stock information from the ReadAPIData class. When the user confirms the amount of shares to purchase, the system processes the transaction and updates the user's portfolio. The updated portfolio is then stored into the database. The user's purchase is successful and the order details are displayed to the user.

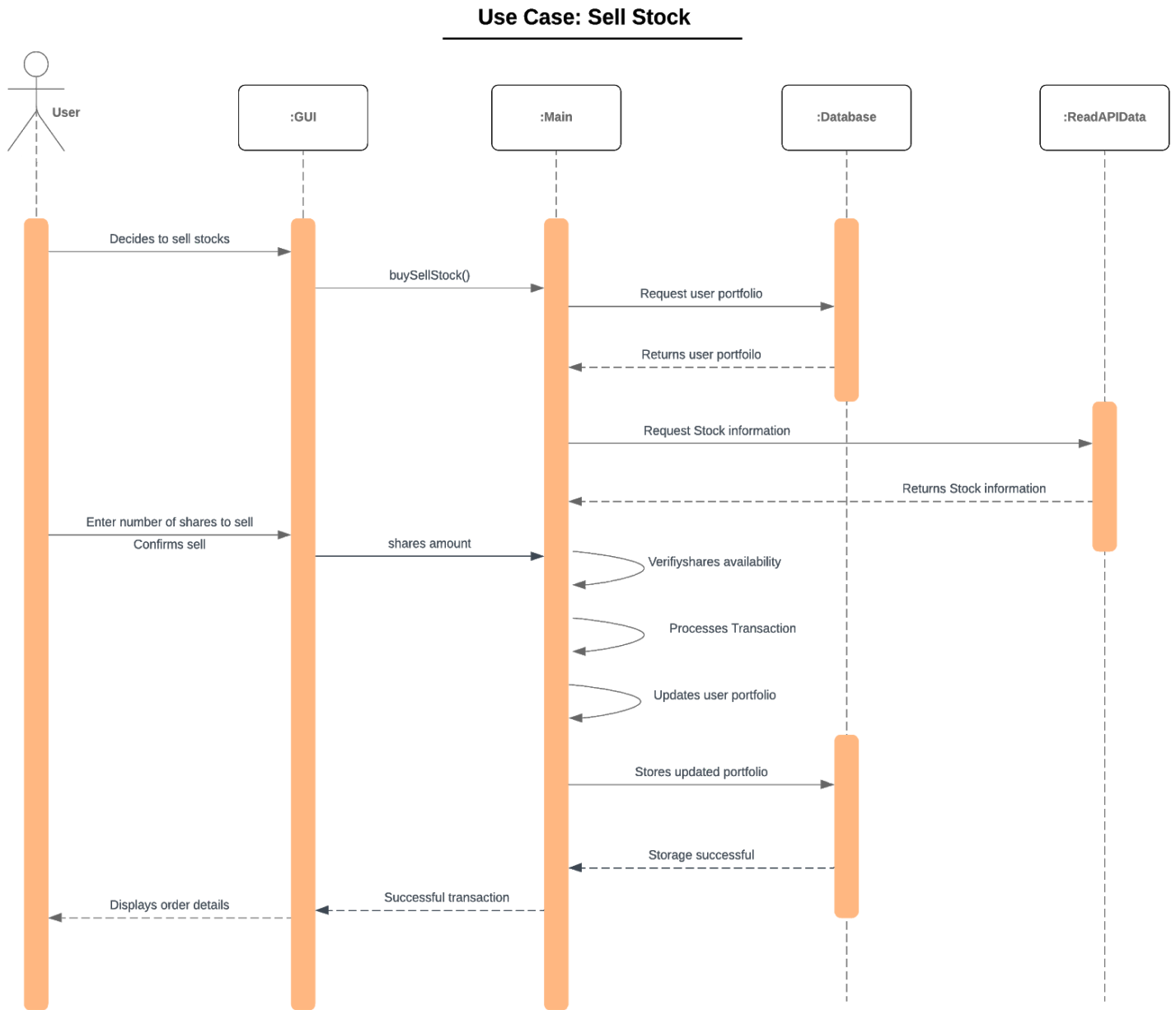


Diagram 4.c: Sell Stock Use Case Sequence Diagram

This sell stock sequence diagram shown above, displays the sale of a stock to the virtual stock market by a user. The user opts to sell a company's stock. The system retrieves the user's portfolio from the database and the company's stock information from ReadAPIData. When the user confirms the amount of shares to sell, the system processes the transaction and updates the user's portfolio. The updated portfolio is then stored into the database. The user's purchase is successful and the order details are displayed to the user.

Use Case: View Portfolio

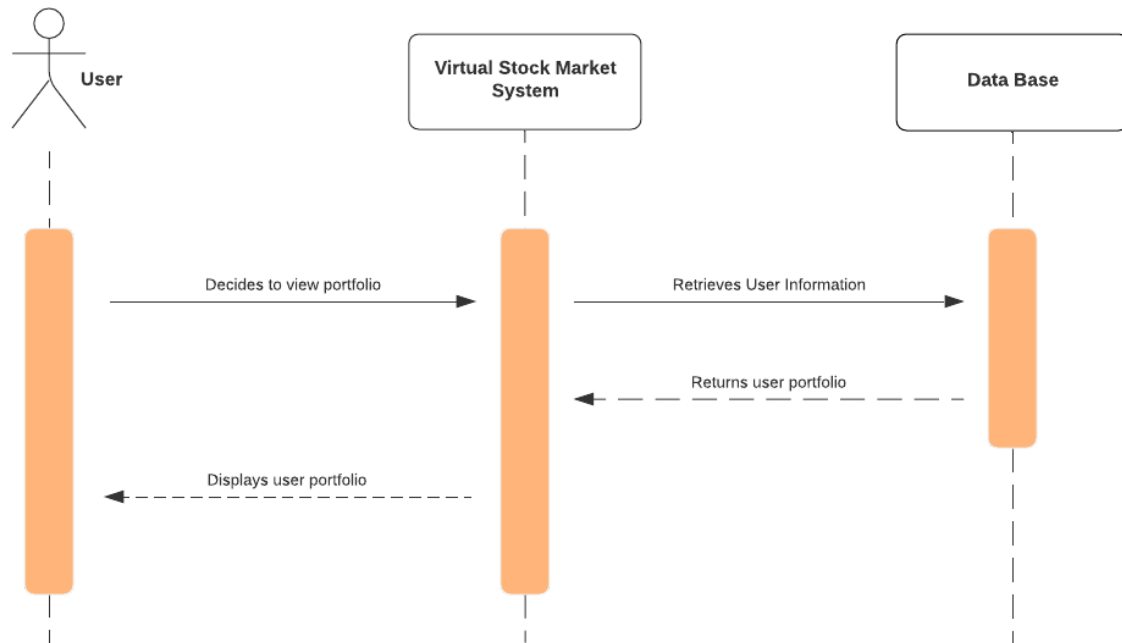


Diagram 4.d: View Portfolio Use Case Sequence Diagram

This view portfolio sequence diagram shown above, displays a user viewing their portfolio. The user opts to view portfolio. The system retrieves the user's portfolio from the database. The system displays the portfolio to the user.

4.3 State Dynamic Viewpoint

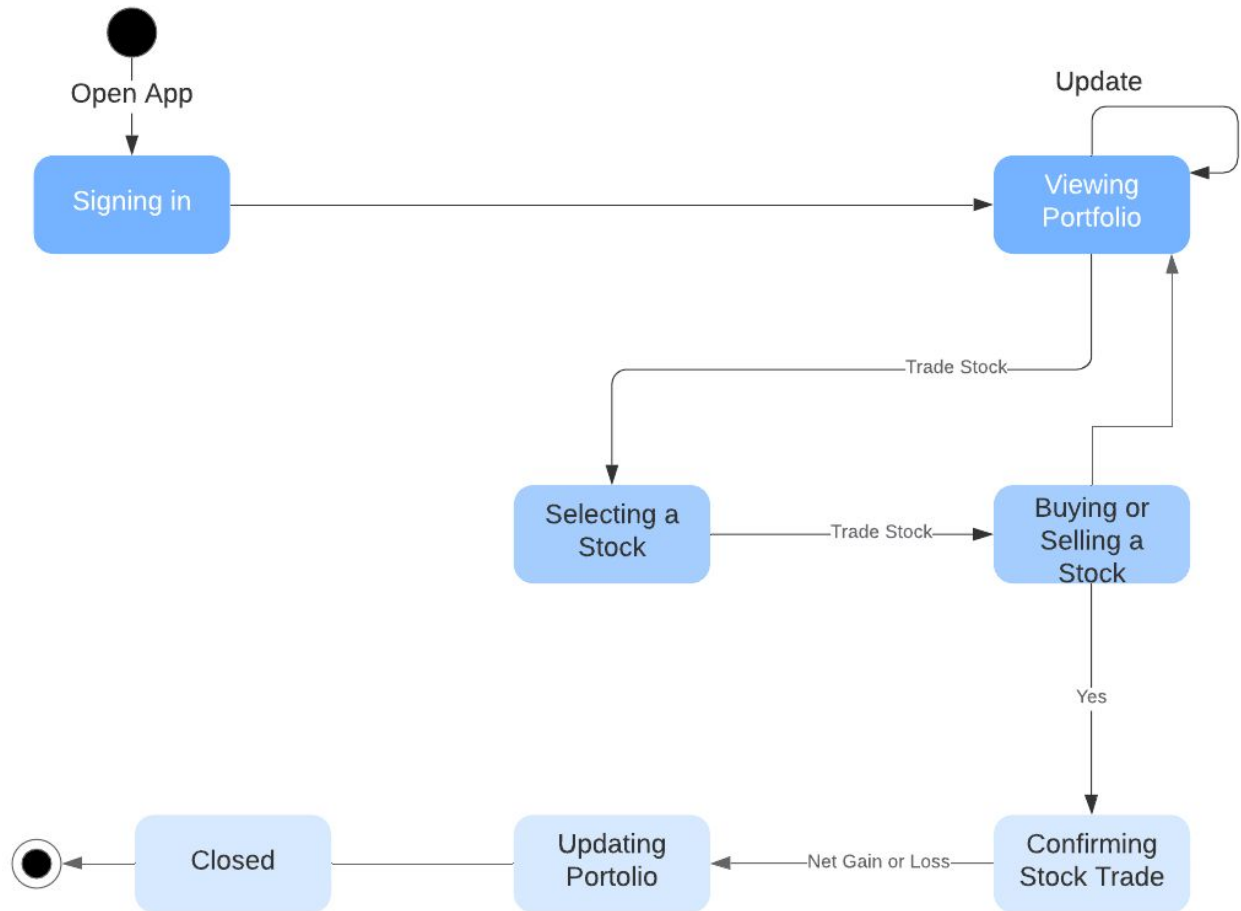


Diagram 5 General State Diagram: The diagram shown above is a state diagram for the general function of the desktop application. The user first signs into their personal account then navigates to the Portfolio page where they are then viewing their portfolio. From that point, they can navigate to select a stock to trade and then buy or sell that stock. If they choose to do so, they will confirm that stock trade before the system updates their portfolio.

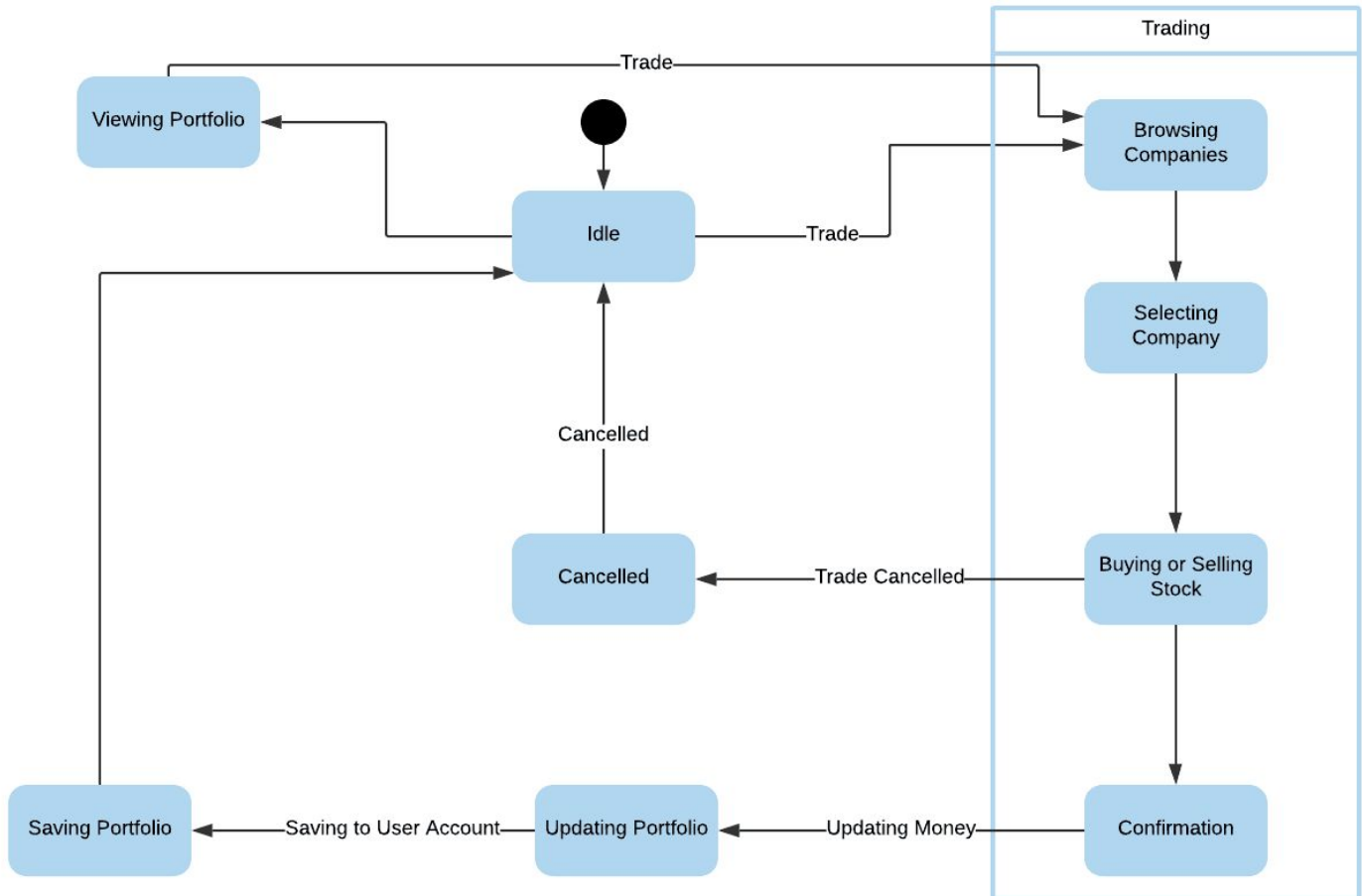
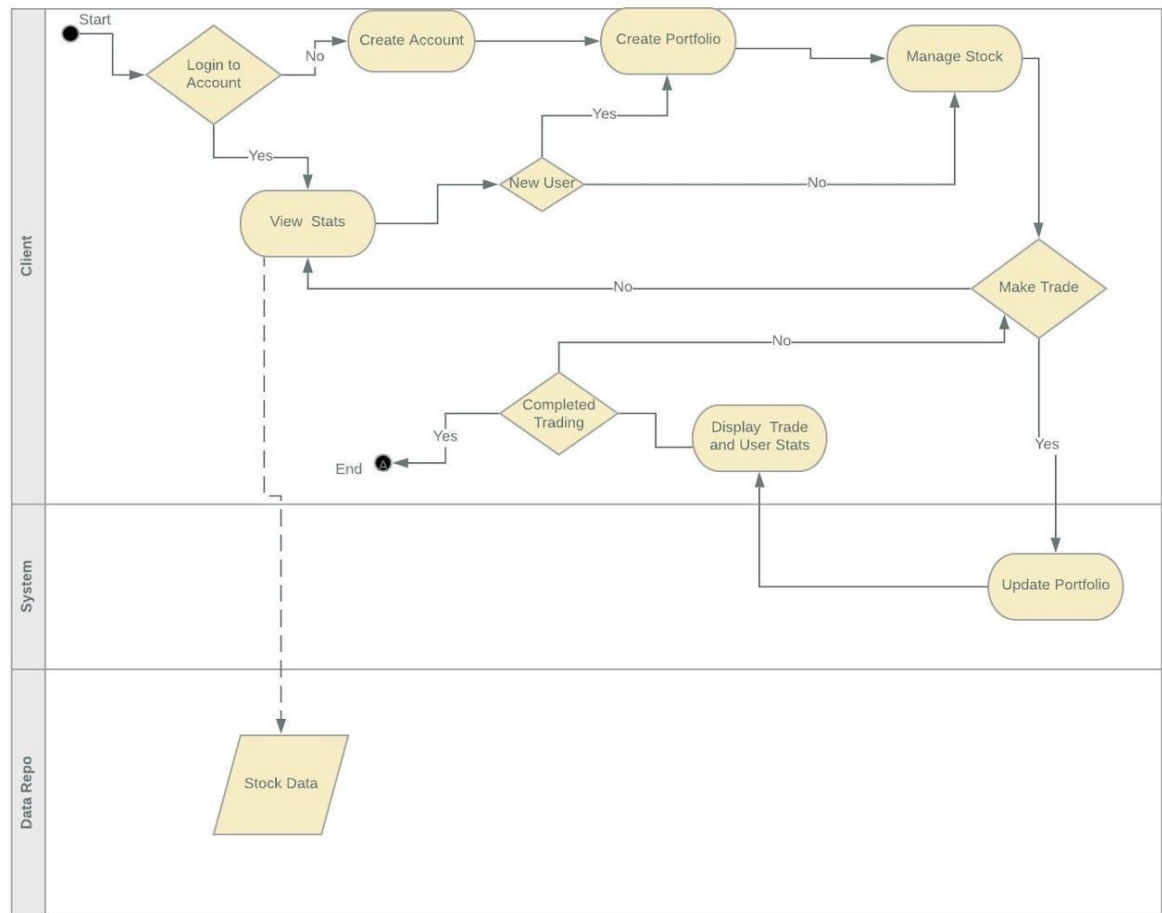


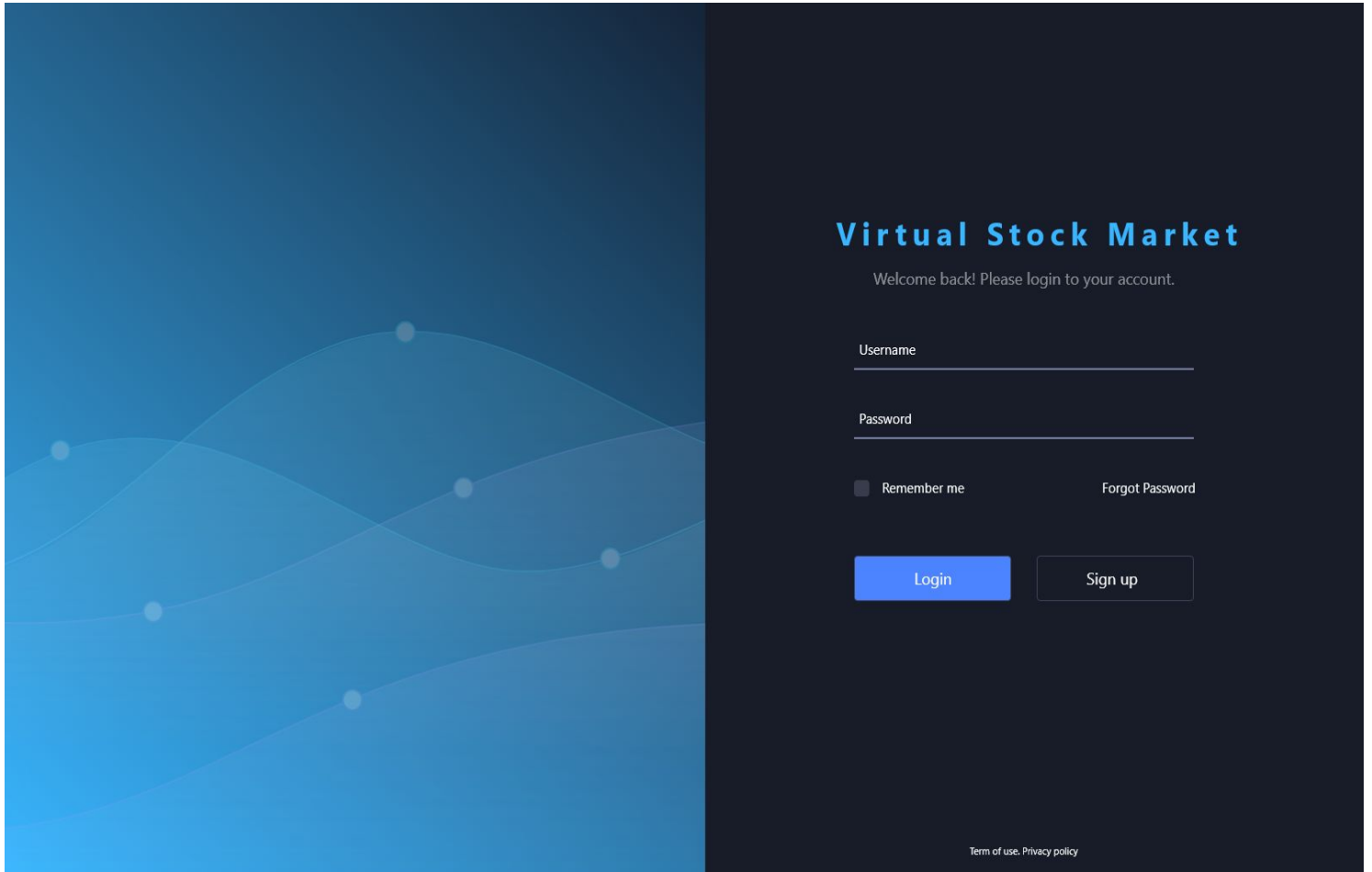
Diagram 6 Detailed State Diagram: The diagram shown above is a more detailed state diagram showing a specific process the user will go through to buy or sell a stock.

4.4 Updated Data Flow or Activity Viewpoint

The overview and solution of the application was unclear therefore, it was decided that this will be a desktop application that is based on a simulation of trading in the stock market. With that being said the following activity diagram is an updated version. The modifications include the changing from “new game” decision to a “new user” decision. Furthermore the end point of the activity diagram was moved and is only when the user has picked yes for the “completed trading” decision.



5. Human Interface Design



Virtual Stock Market

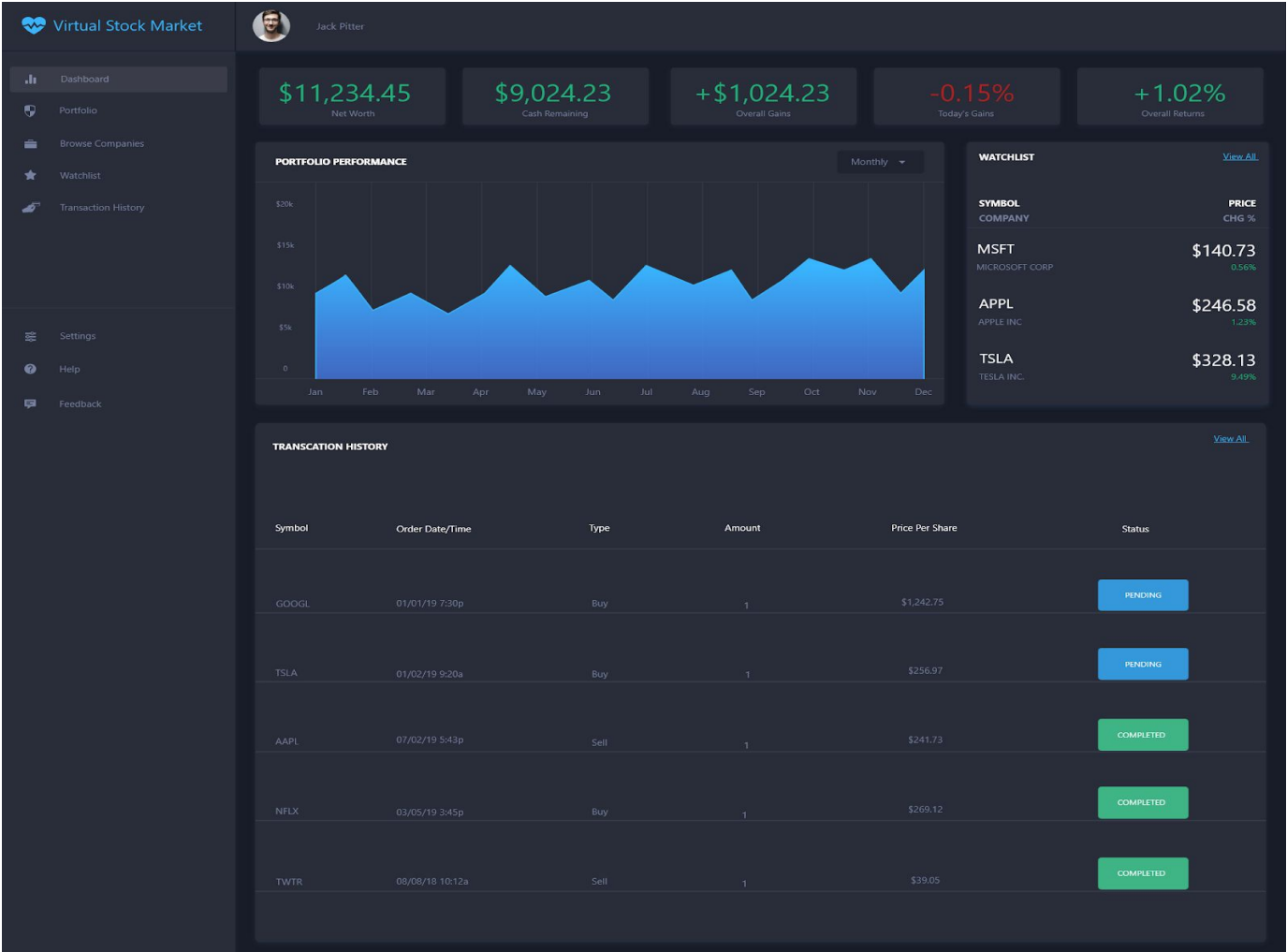
Welcome back! Please login to your account.

Username

Password

☐ Remember me [Forgot Password](#)

[Term of use](#) [Privacy policy](#)



Virtual Stock Market

Dashboard

Portfolio

Browse Companies

Watchlist

Transaction History

Settings

Help

Feedback

Jack Pitter

Browse Companies

QgXSearch

DISPLAYING 8 RESULTS

G GENPACT LTD.	\$38.19 -0.93%	TRADE
GOOGL ALPHABET INC. CL A	\$1,254.30 0.41%	TRADE
GOOG ALPHABET INC. CL C	\$1,265.13 0.33%	TRADE
GE GENERAL ELECTRIC CO.	\$9.00 0.56%	TRADE
GSK GLAXOSMITH LINE PLC ADR	\$43.82 -0.45%	TRADE
GILD GILEAD SCIENCES INC.	\$63.32 -4.06%	TRADE
GS GOLDMAN SACHS GROUP INC.	\$214.23 1.51%	TRADE
GM GENERAL MOTORS CO.	\$36.74 2.57%	TRADE

Q g

Trade Order

\$9,0243 Available

\$1,265.13

1

\$1,297.76

\$1,297.76*

6. Updates / Modifications to SRS

Added Constraints:

- a. Stock Market Knowledge: Currently, there is a limited knowledge in understanding of the U.S. Stock Market functions
- b. API Requests: For the API (World Trading Data¹), there is a limit of 250 requests per day with 20 stocks per request

Added Risks:

Risk: With the API in use (World Trading Data¹), the team exceeds the limited number of requests per day (250 requests of up to 20 stocks per request) for the free version. If the limit is exceeded by the user the user cannot continue to use the application for viewing and purchasing new stocks.

Severity (High/Med/Low): High

Likelihood (High/Med/Low): Low

Mitigation: Communicate with the team to plan out limiting the number of requests per day used. Ensure application design prevents user from easily being able to go over number of daily requests or simply have users use their own free/premium token allowing from 250 up to unlimited stock pulls. Granted 250 requests per day should be more than enough to functionally use this application.

Risk: User portfolio data is lost.

Severity (High/Med/Low): High

Likelihood (High/Med/Low): Low

Mitigation: Implement database system allowing for user portfolio data to be stored not only on their local machine, but also a remote database.

7. Glossary/References

Non-Standard Acronyms

1. API: Application Programming Interface
2. U.S. : United States of America
3. UI: User Interface
4. GUI: Graphical User Interface

References

1. “World Trading Data,” *World Trading Data*. . “<https://www.worldtradingdata.com/>”
2. “PyQt,” *Qt*. The Qt Company. “<https://www.qt.io/>”

8. Prototype Architecture & Design – Unless you plan to implement 100% of your SRS, then you need to provide additional design insight specific to your prototype (likely need to provide a prototype version of Sections 3-4) (25%)

The prototype version will be available via the link:
<https://github.com/Chrithtoph/StockMarketApplication>

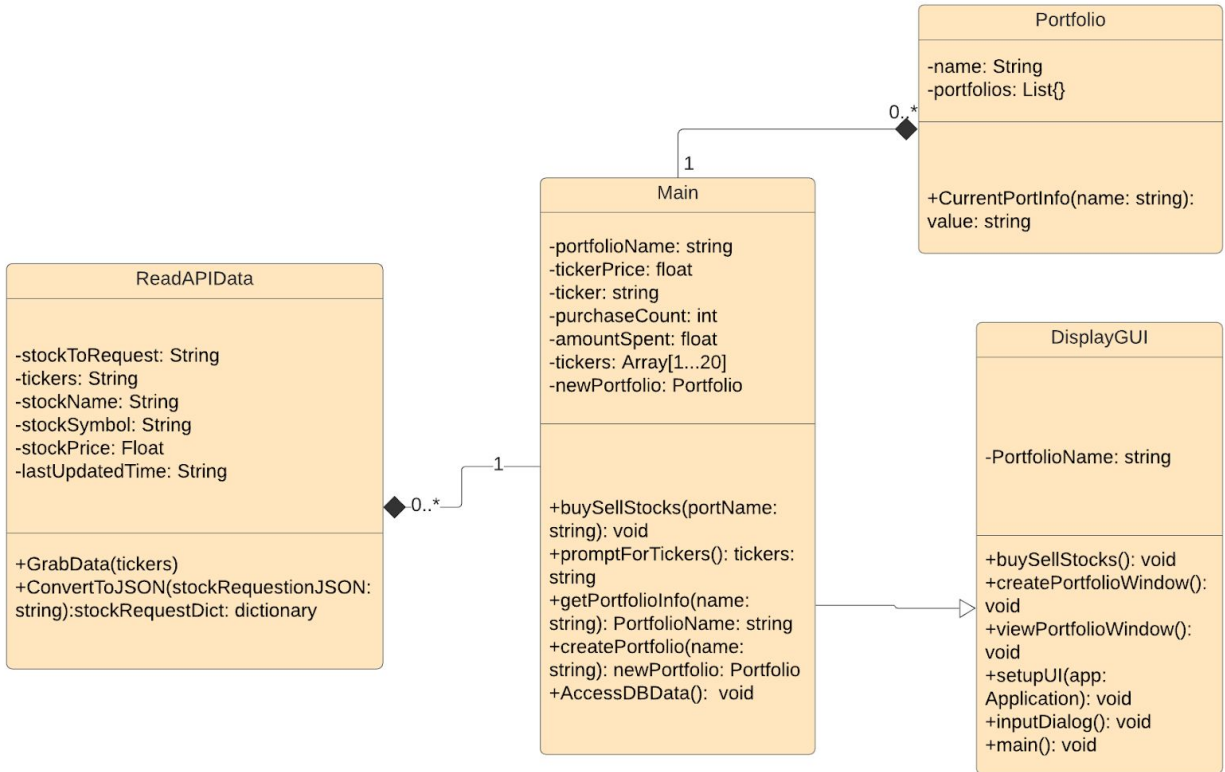
Prototype Architecture:

The desktop application prototype will have the same three-tier architecture composed of the user, the actual desktop application, the database for the user, and the live U.S. Stock Market data (shown in diagram 1). The difference with the prototype is in the database section of architecture. More specifically, due to the time constraints, the prototype will only support a single user rather than multiple users using the database therefore it will be a local database. This means that the database will not have to hold the data of multiple users. With that being said, the login username and password for the single user will also not be encrypted for the prototype. The purpose of the database in the prototype architecture will be to hold the dollar amounts for the user based on their net gains and losses throughout their trading process. As with the system architecture, in the prototype architecture the application will access the online database which enables the users to continually view how their stocks have developed over time. The application will also still communicate to the live U.S. stock market data through the World Trading Data¹ API to continually provide the application and user with live stock market data.

Prototype Design:

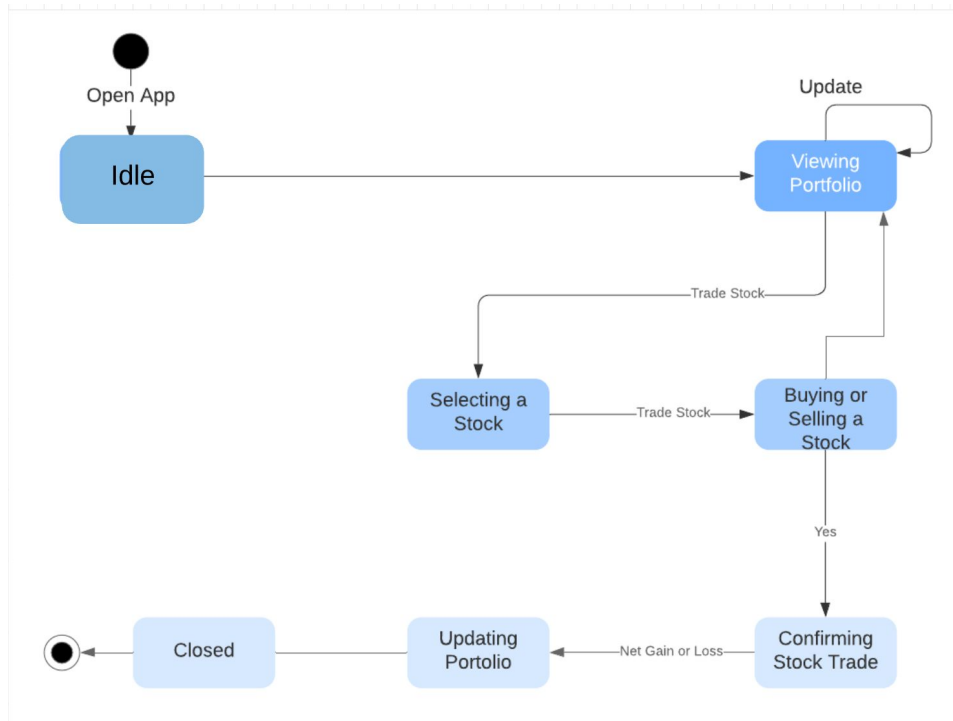
Due to time constraints, limited desktop application development knowledge among the team and limited programming language capabilities, for the overall design of the prototype, there will be a lower scale version of GUI/UI than shown previously in the human interface design images.

Logical Viewpoint: In the prototype class diagram shown below, the Database class was removed as the database will now only store the dollar amounts for a single user as described above. Previously, the Database class was connected to the Portfolio class because it would store the portfolios for multiple users, however, the database will now be accessed by a method in the Main class which handles user trading actions and portfolio information.



Interaction Viewpoint: For the prototype version, the only change to the interaction viewpoint design is for the sequence diagrams in which there will not be an implementation of Diagram 4.a: User Login Use Case Sequence Diagram. All other sequence diagrams previously described will be implemented in full.

State Dynamic Viewpoint: For the prototype version of state dynamic viewpoint design, there was a change to diagram 5 in which the signing in state was changed to idle as there will not be a sign in state implemented with the prototype. The updated general state diagram is shown below. The detailed state diagram still follows the prototype design and will be implemented in full.



Activity diagram: The prototype design will also change the updated activity diagram to remove the login components that will not be implemented in the prototype. The prototype activity diagram is shown below.

