UE21CS352B

Object Oriented Analysis & Design using Java

E Section

Mini Project Title: Stock Trade Portfolio Mangement

SRN	Name
PES1UG21CS257	K Prajwal
PES1UG21CS295	Kushal U
PES1UG21CS299	Lalataaksh S
PES1UG21CS315	Mahesh KM

Problem Statement:

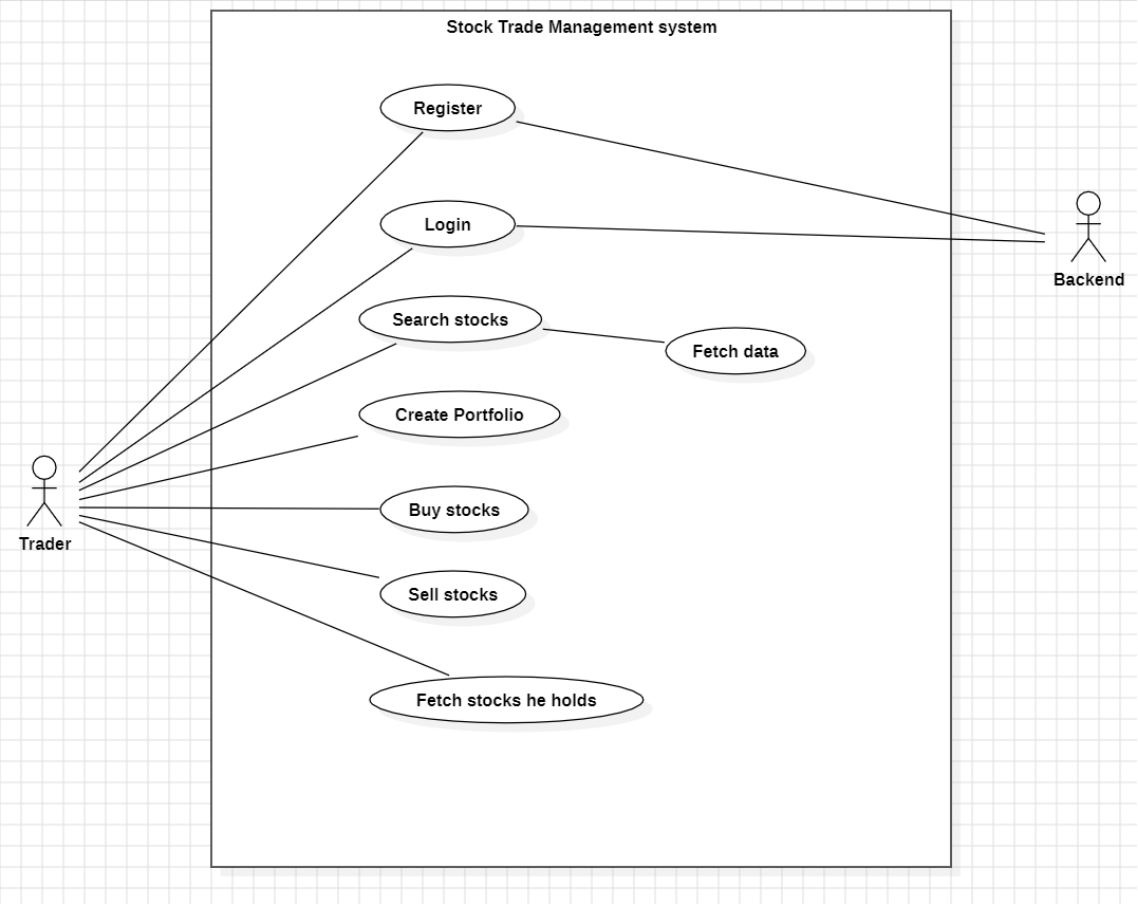
Creating a stock trading portfolio management system is a sophisticated project that involves integrating various functionalities. Such a system would allow traders to have a comprehensive view of their investments, make informed decisions based on real-time data visualization, and execute trades efficiently. By inputting a company's ticker symbol, the system could provide detailed information and facilitate the management of stock quantities, which is crucial for maintaining a balanced and diversified portfolio. This could significantly streamline the trading process and enhance the user experience for traders.

Functionalities:

1. Trader registration and Authentication
2. Visualize real-time stock details.
3. Purchase stokes
4. Sell the stocks
5. Update stock quantity

Models:

1. **Use case Diagram:**



Use case Specifications:

1. Name: Register

Summary: Create account for Stock Trade Management

Actor: Trader

Pre-condition: He/She must have knowledge of stocks and is interested in managing his trading portfolio.

Description: User creates valid userId and password which is used for future authentication.

Exception: If network failure occures before completion of registration.

Alternative Flows: User is redirected to registration page again, steps are rolled back.

Post-conditions: User is directed to login page. UserId and password is stored in backend database.

2 .Name: Login

Summary: User Logins to his pre-created Trade management account.

Actor: Trader

Pre-condition: User must have trade management account i.e, valid userID and password.

Description: User feeds his/her valid userId and password in specified fields and clicks on login button. Userid is used as key to search database and retrive password and is authenticated.

Exception: If in case userId or password is invalid.

Alternative Flows: The user remains in same login page, valid error message is displayed to user.

Post-conditions: User is directed to home page of website.

3 .Name: Search stocks

Summary: User can fetch details of stocks he is interested

Actor: Trader, Backend

Pre-condition: There must exist stock which user is interested

Description: user feeds stock id to search field and enter stock details are feteched from backend and displayed to user.

Exception:if that stock does not exist.

Alternative Flows: Appropriate error message is displayed.

Post-conditions: Stock details are displayed in frontend.

4. Name: Create Portfolio

Summary: For managing stocks user creates portfolio where information about stocks is put at one place for giving overall picture of stocks held by user.

Actor: Trader, backend

Pre-condition: user must have an account and should be logged in.

Description: User clicks of create new portfolio button user gives name for portfolio and this is stored in backend assigning new ID.

Exception: if user is not logged in.

Alternative Flows: user is redirected to login page.

Post-conditions: Portfolio account is created and stored in backend.

5. Name: Buy Stocks

Summary: User can invest in stock by clicking invest button and select quantity and data of purchase.

Actor: Trader

Pre-condition : He must have portfolio

Description: User can invest in stock by clicking invest button and select quantity and data of purchase.

Exception: if that stock does not exist

Alternative Flows: Appropriate error message is displayed.

Post-conditions: Stock is added to portfolio.

6 . Name: Sell Stocks

Summary: Trader can sell stock that is there in his portfolio.

Actor: Trader

Pre-condition: Trader must have stock that he wants to sell in his portfolio.

Description: Trader selects on stock present in portfolio and selects recoder as sold.

Exception: NA

Alternative Flows: NA

Post-conditions: Particular stock must be removed from users portfolio.

7. Name: Fetch stocks he Holds.

Summary: Trader can see all the stocks he holds or invested in his portfolio home page.

Actor: Trader, Backend

Pre-condition: Trader must have at portfolio.

Description: Trader clicks on his portfolio account backend fetchs stocks under this protfolio and he will be directed to page where all his stocks displayed.

Exception: NA

Alternative Flows: NA

Post-conditions: NA

8 .Name: Visualize Stock Trend

Summary: Plot's value of stock over a period. use case used by other usecases particularly Search stocks and Fetch stocks he holds.

Actor: Backend

Pre-condition: Valid StockID

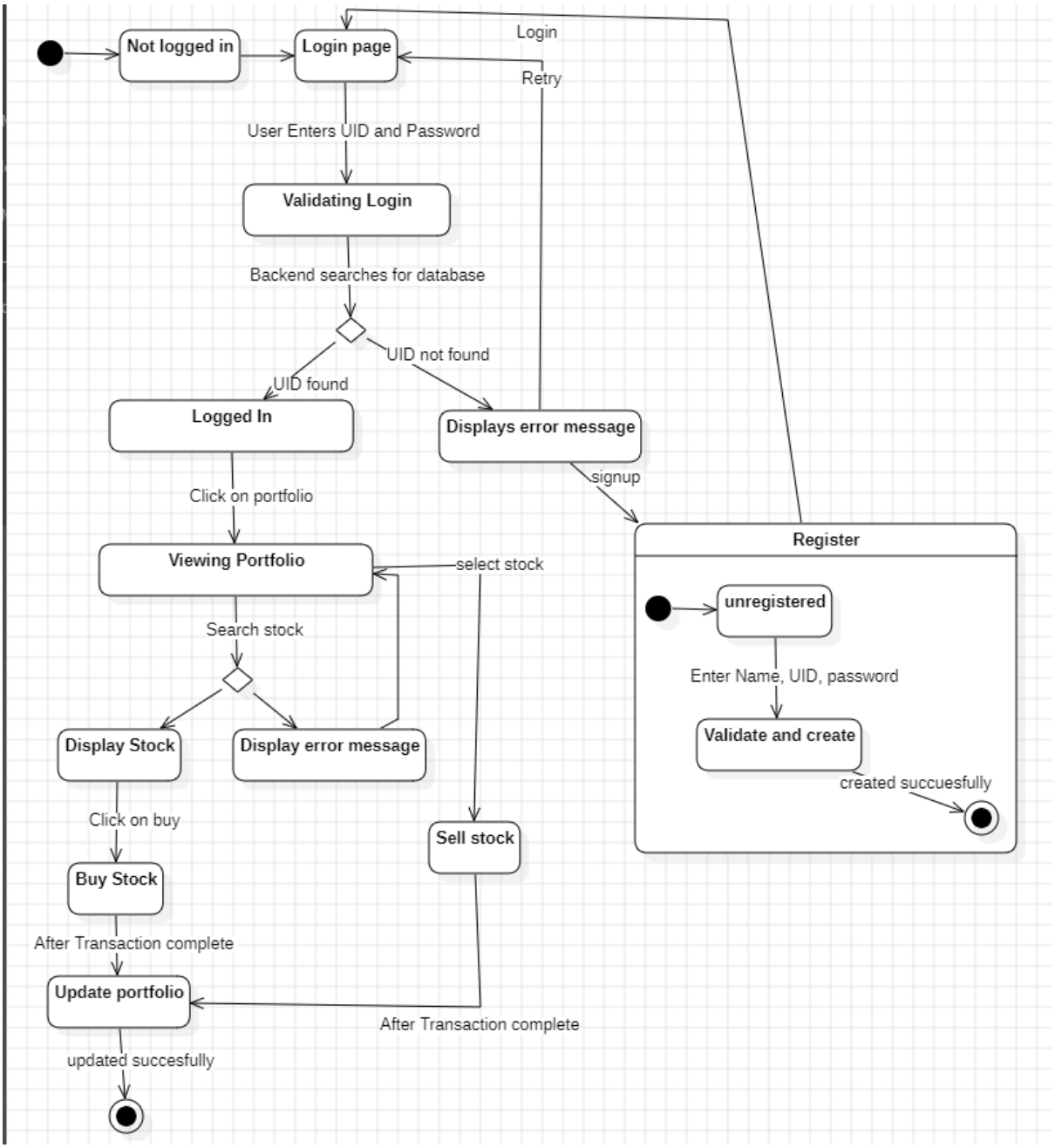
Description: User searchs StockID and plots value of stock over a period along with other details.

Exception: Non Valid StockID

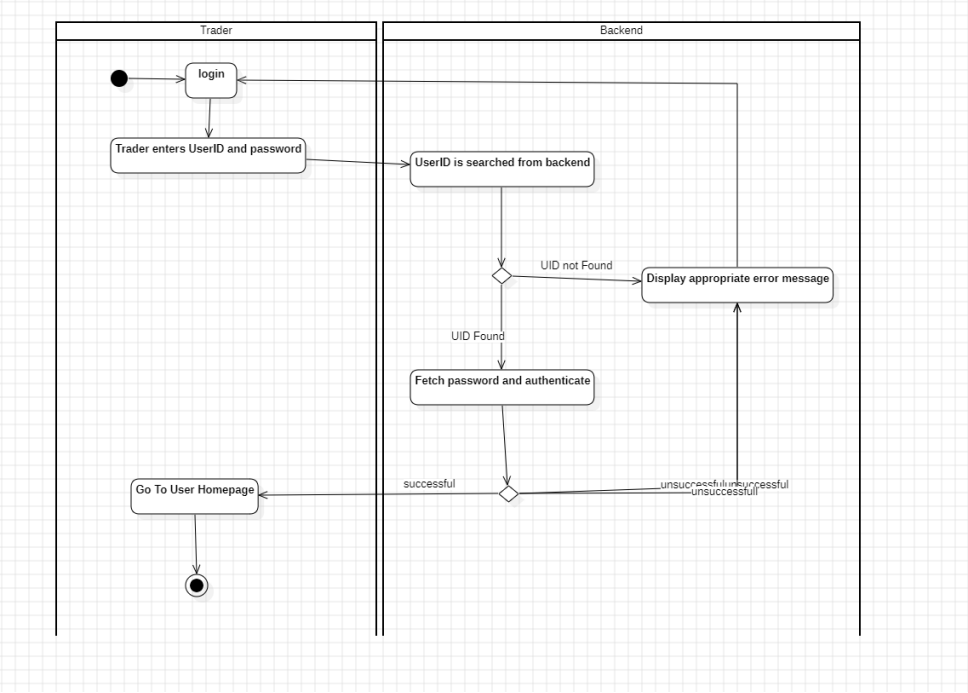
Alternative Flows: Display appropriate error message

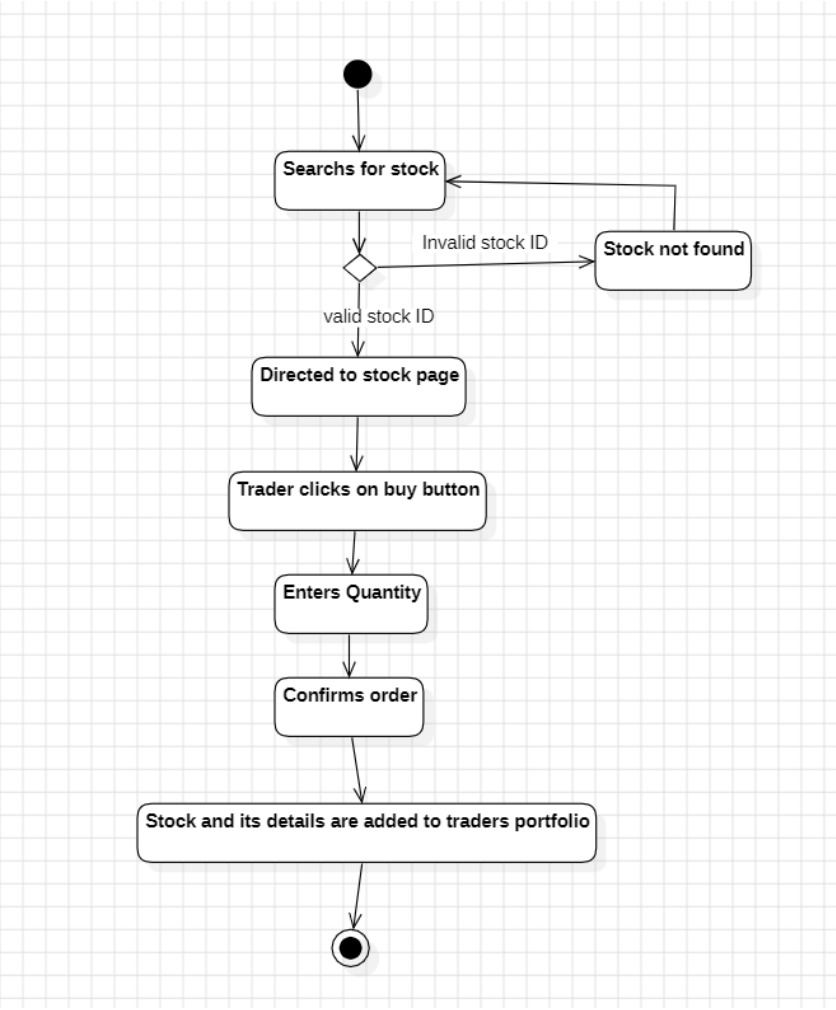
Post-conditions: Plot is displayed.

State Diagram

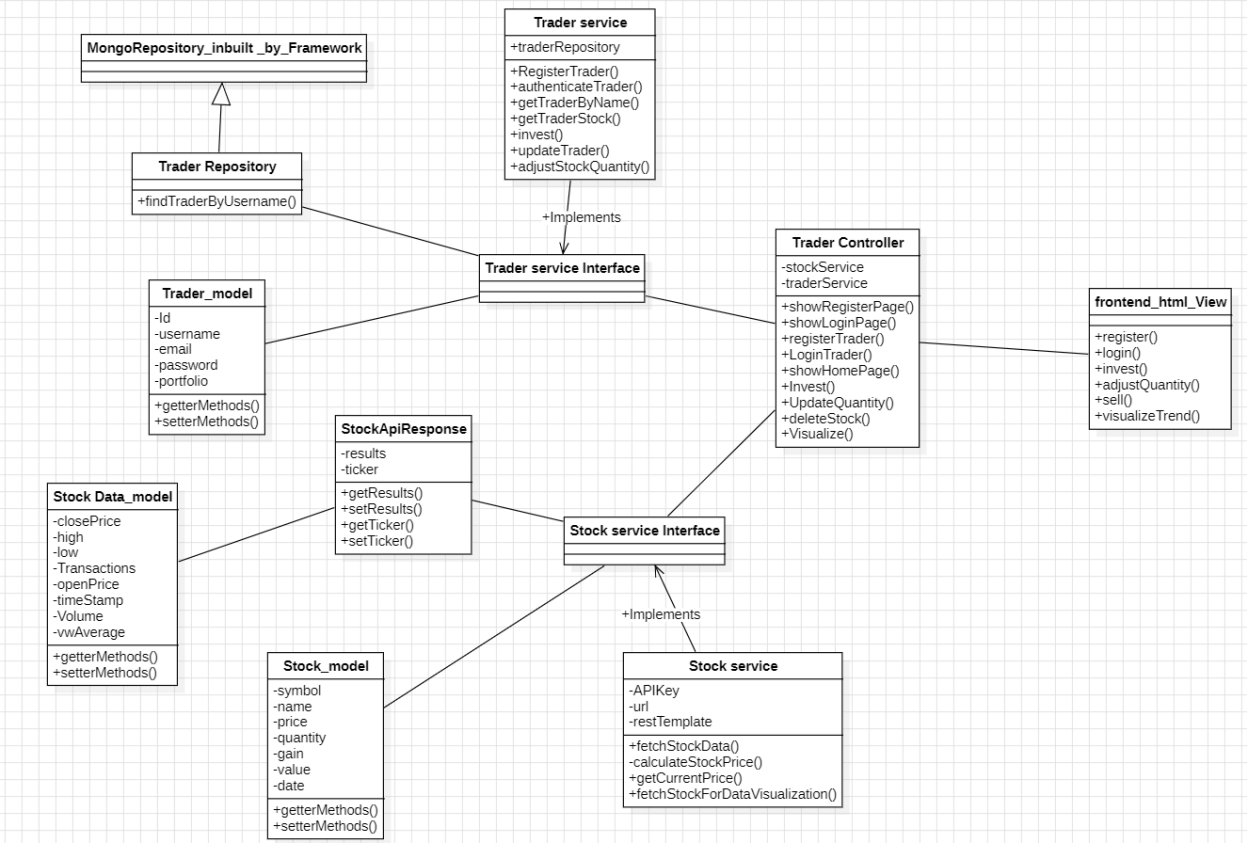


2. Activity Diagram


Trader
Backend
login
Trader enters UserID and password
UserID is searched from backend
UID not Found
Display appropriate error message
UID Found
Fetch password and authenticate
Go To User Homepage
successful
unsuccessful
unsuccessfull


Searchs for stock
Invalid stock ID
Stock not found
valid stock ID
Directed to stock page
Trader clicks on buy button
Enters Quantity
Confirms order
Stock and its details are added to traders portfolio

3. Class Diagram



Architectural Patterns:

Model-View-Controller (MVC): MVC separates the application into three main components: Model (business logic and data), View (presentation layer), and Controller (handles user input and updates the model and view accordingly). This separation of concerns promotes modularity and maintainability.

Event-Driven Architecture (EDA): EDA decouples components by enabling them to communicate asynchronously through events. This can be useful for handling real-time data updates, event sourcing, and integrating with external systems.

Design Principles:

Single Responsibility Principle (SRP):

Example: In a Stock Trade Portfolio Management system, the Stock class should be responsible only for representing the properties and behavior of a single stock entity. It should not be responsible for handling user authentication or processing orders.

This property is true for even Trader, StockData, ApiResponse classes.

Open Closed Principle (OCP):

Example: The Trader service interface class is used, and separate Trader service class implements this interface so that in future if any new operations are to be done will not modify existing code instead create new class which extends this interface.

This property holds for even Stock service class.

Dependency Injection (DI):

Example: Instead of directly instantiating dependencies within a class, dependencies such as data access objects (DAOs) or external services are injected into the class through constructor injection or setter injection. For instance, a TraderService class might depend on a TraderRepository, which can be injected at runtime.

Design Patterns:

Singleton Pattern (Creational):

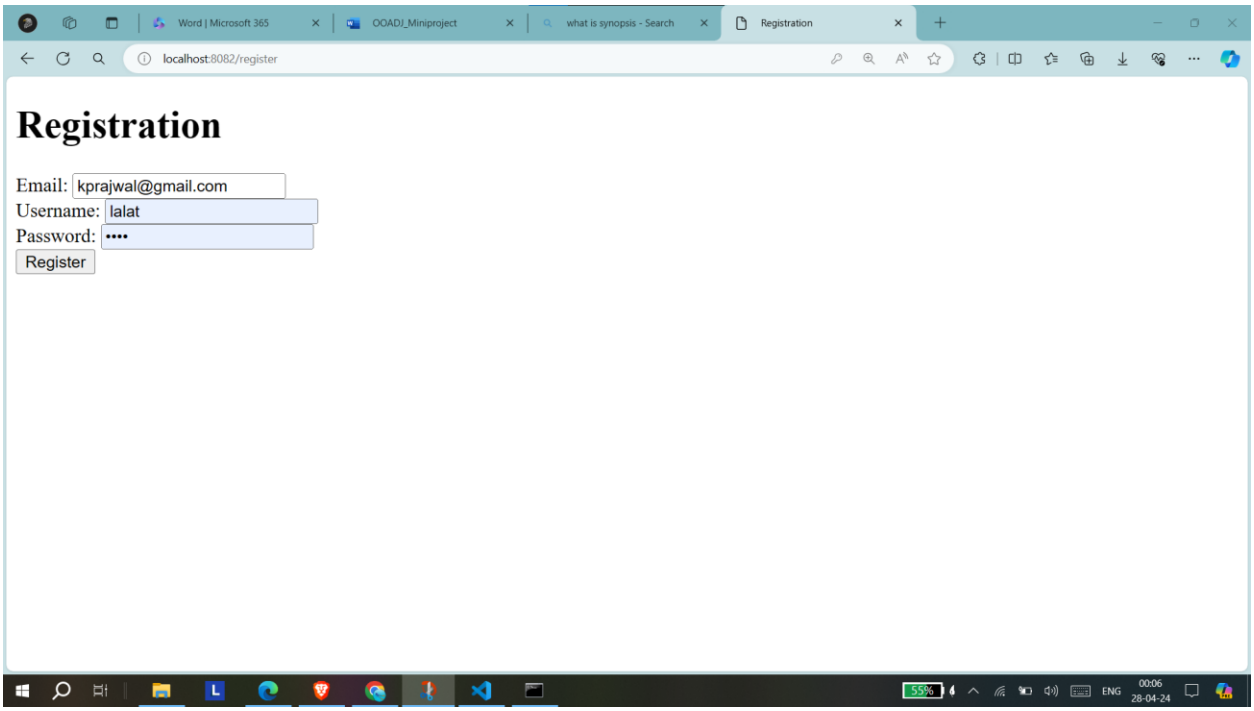
MongoDB Repository class has one object instance to access, store and modify traders' details in database. It acts as an entity through which backend database is handled efficiently, maintaining consistency and following ACID Properties.

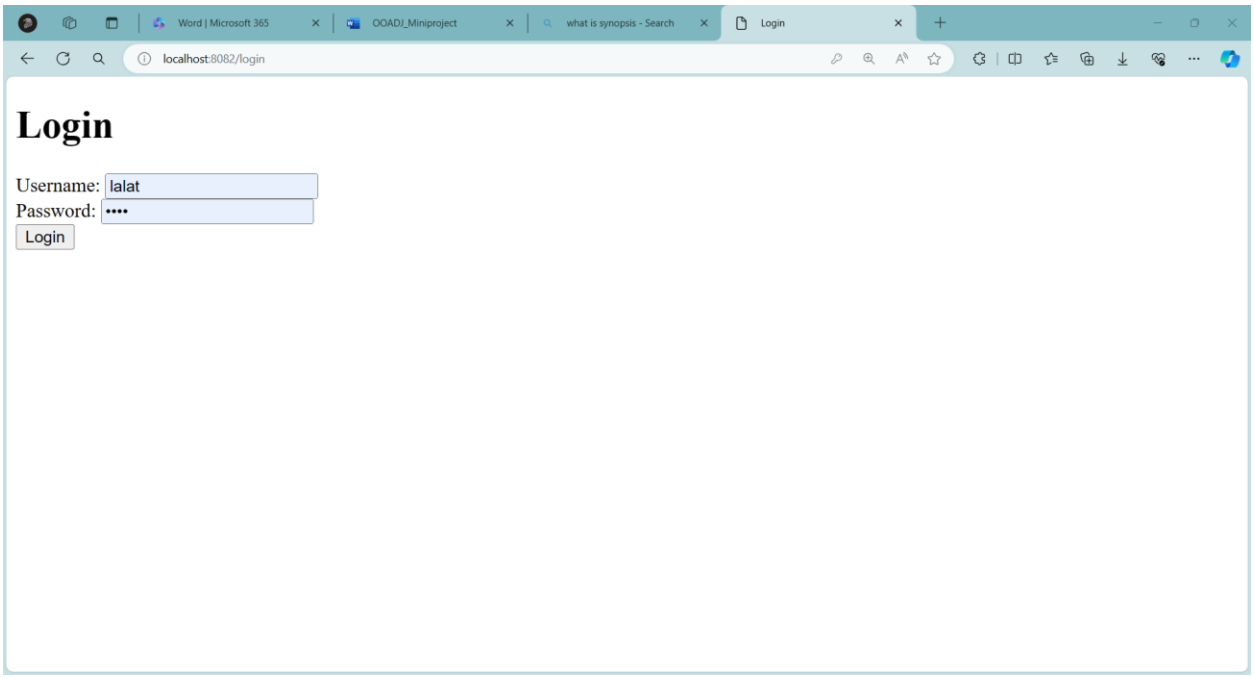
Facade Pattern (Structural):

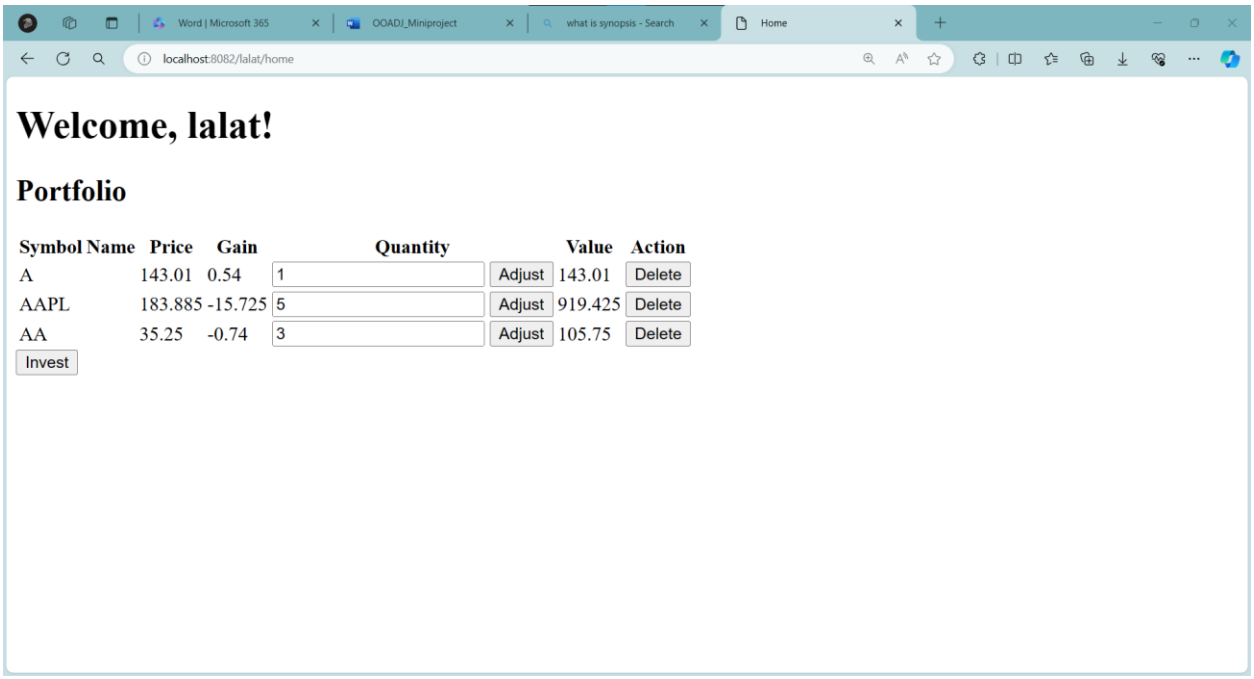
Facade Pattern is used in a stock trade portfolio management application to simplify the interface and interactions between the application's subsystems, promote loose coupling, manage complexity, and provide a unified access point for client code. It improves the maintainability, flexibility, and scalability of the application by abstracting away the details of the underlying subsystems.

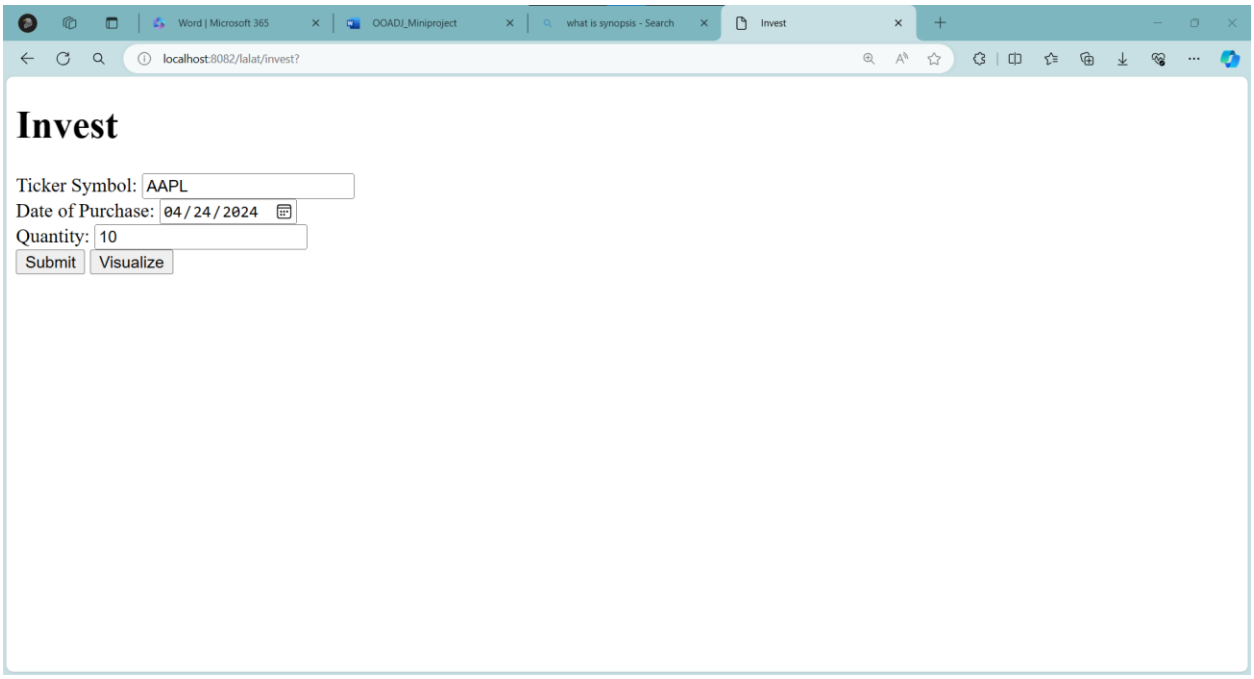
Command Pattern (Behavioral):

Example: The command pattern can be used to encapsulate requests for executing specific actions in the system, such as registration, login, placing a buy order, selling a stock, or generating a report. Each command object contains all the information needed to perform the action, allowing for easy undo/redo functionality or logging of user actions.


Word | Microsoft 365
OOADJ_Miniproject
what is synopsis - Search
Registration
localhost:8082/register
Registration
Email: kprajwal@gmail.com
Username: lalat
Password: ••••
Register
55%
ENG
00:06
28-04-24


Word | Microsoft 365
OOADJ_Miniproject
what is synopsis - Search
Login
localhost:8082/login
Login
Username: lalat
Password: ••••
Login


Word | Microsoft 365
OOADJ_Miniproject
what is synopsis - Search
Home
localhost:8082/lalat/home
Welcome, lalat!
Portfolio
Symbol Name Price Gain Quantity Value Action
A 143.01 0.54 1 Adjust 143.01 Delete
AAPL 183.885 -15.725 5 Adjust 919.425 Delete
AA 35.25 -0.74 3 Adjust 105.75 Delete
Invest


Word | Microsoft 365
OOADJ_Miniproject
what is synopsis - Search
Invest
localhost:8082/lalat/invest?
Invest
Ticker Symbol: AAPL
Date of Purchase: 04/24/2024
Quantity: 10
Submit
Visualize


Word | Microsoft 365
OOADJ_Miniproject
what is synopsis - Search
Stock Trend Map
localhost:8082/AAPL/visualize
Stock Price
Price
200
195
190
185
180
175
170
165