Process MeNtOR 3.0

CoinMaster
Requirements Model

Version:	0.3
Print Date:	
Release Date:	
Release State:	Initial
Approval State:	Draft
Approved by:	
Prepared by:	
Reviewed by:	
Path Name:	
File Name:	Group11-SRS-CS2212B.doc
Document No:	

object oriented pty. ltd.

Document Change Control

Version	Date	Authors	Summary of Changes
0.1	Feb10	Ziyuan Li	Content 1 and content4
0.2	Feb11	Sihui He	Conent 5
0.3	Feb14	Mingkai Yang, Yuhan Zhang	Diagrams(Content 2 and Content3)

Document Sign-Off

Name (Position)	Initials	Date
Yuhan Zhang	YZ	Feb.9
Sihui He	SH	Feb.9
Mingkai Yang	MY	Feb.9
Ziyuan Li	ZL	Feb.9

Contents

1 Introduction

1.1 Purpose

This document details the requirements of the system EzCoin. EzCoin is like coinbase but for broker managers and with human intervention in the process of selecting and trading. EzCoin is a program developed intended to give cryptocurrency broker managers a convenient and efficient way to fetch and monitor real-time prices, execute buy and sell decisions according to strategies, and record transactions. Preventing unauthorized access to classified information, the software as well requires login credentials for each individual broker manager. It helps simplify managers' routine tasks by implementing Java algorithms and SQL databases.

1.2 Overview

CoinMaster is an assistant software that can help broker managers complete a series of cryptocurrency-related and broker-related tasks.

After the installation of the software, users (broker managers) should be able to click on it and run the program. A login window would pop up and ask for a username and password, which both should be enlisted during the installation. In later versions, a "Forget the password" button might be added in case of a user has trouble logging into the system. Upon the user entering the correct login information and clicking "submit", the primary interface would show up. The primary interface would contain three major sections.

On the right half of the interface would be the record of brokers. Each broker that is under the supervision of the user would take up one horizontal row. Broker's name, cryptocurrencies that the broker has interests in, and broker's trading strategies are the three corresponding columns. A broker may have multiple interested currencies and multiple ongoing strategies. Each broker's strategy slot is a drop-down list. Strategies are algorithms set up by the brokers and usually take the form of a pricing condition (i.e., When bitcoin is above $40,000, and ETH below $3500, buy $1000 worth of XRP). Below the form, there are two buttons, one for adding a broker and another for removing a broker. Below the two buttons, there is "Perform Trade" button. Before performing a trade, a broker and one corresponding strategy must be selected. The software will determine whether the conditions of the strategy has been met or not and will return the result of the trade by fetching real time price from CoinGecko.

On the left half of the interface, two sections are to be presented. A full trading record will be displayed on the top half. In the form, all successful transactions will be listed. Each transaction would contain broker name, cryptocurrency type, traded action (buy or sell), price, volume, and time stamp. Below the fully detailed record, an accumulated bar chart is shown. The horizontal axis is the strategies and vertical axis is the times the strategy has been successfully applied. Distinguishing between brokers, each broker's strategy (A-Z) will be filled with different colors. The chart can straightforwardly indicate that which strategies have been trigger the most or least often and which brokers have been most or least active in trading.

Further functionalities might be added in later versions. Currently considering: record of brokers' holding, profit and loss from trading actions, more variation of statistical charts.

1.3 References

https://www.coinbase.com/

https://www.coingecko.com/

https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency

2 Domain Model

A Domain Model Class Diagram is a high level object model of the classes that exist in the business domain supported with a general statement about each Class covered within the model.

Given that the model is largely used to structure the problem in terms of concepts and entities which participate in the use cases, emphasis is placed on the assignment of a unique name and description as well as the identification of candidate relationships between these domain model classes.

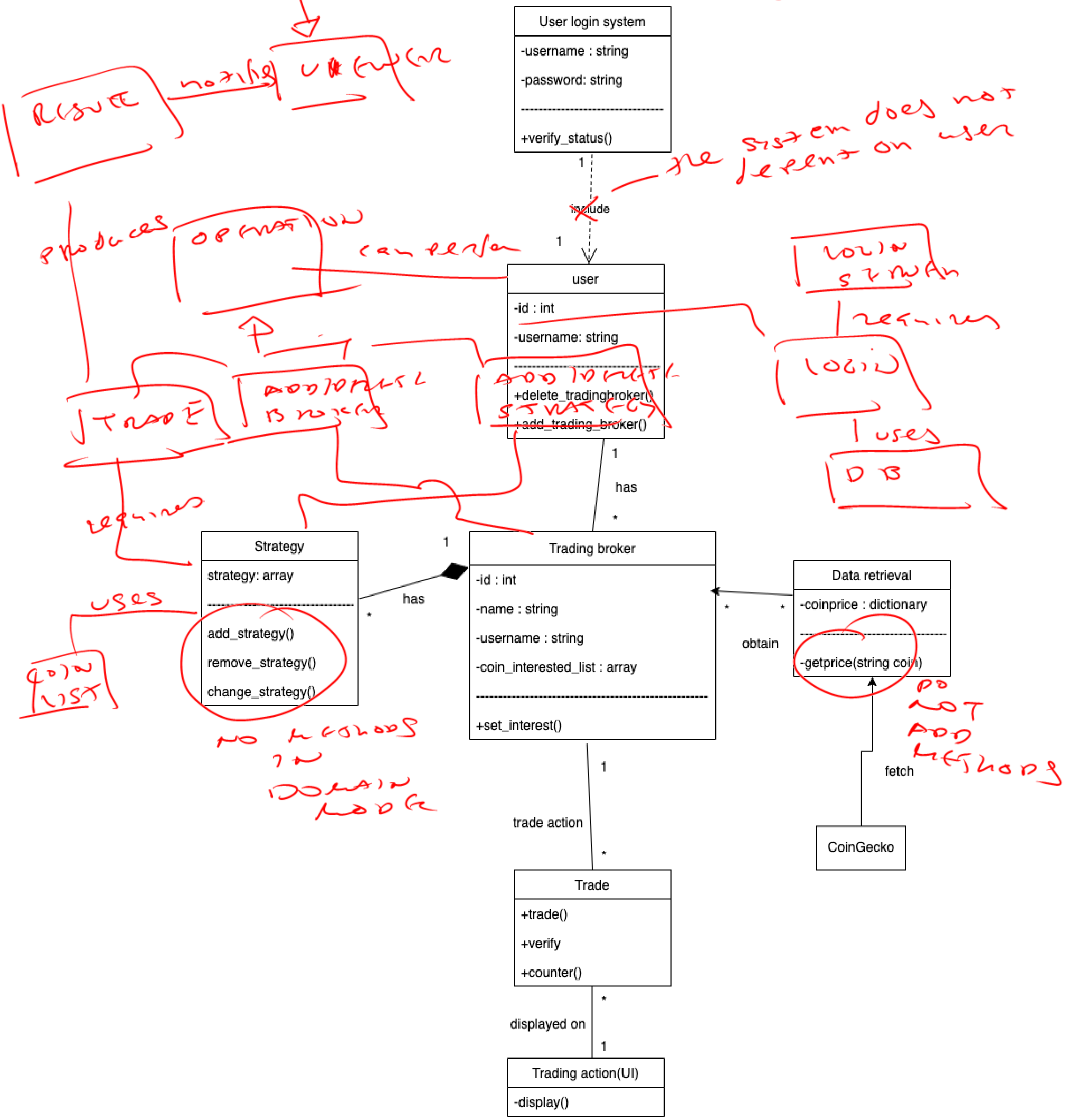
check your classes

histogram

table

2.1 Domain Model Class Diagram

expand



reuse

notify

user

the system does not depend on user

produces

operation

can perform

login system

requires

login

uses

DB

additional strategy

trade

additional broker

requires

uses

coin list

No methods in domain model

do not add methods

Fig. 2.1 Domain Model Class Diagram

2.2 Domain Model Class Definitions

Detailed below is an account of each business object contained within the business domain model. Group the definitions by subject area if required.

During the initial and core releases of the requirements model, the attributes and responsibilities assigned to each object will be preliminary and require refinement.

Repeat the following section for each business object in the diagrams in the previous section

2.2.1 User login system

Description	The object represents a user login system, when user start system, the users are greeted with a login window or form, where they can input their username and password.
Attributes	Username: *String* Password*: String*
Responsibilities	The User login system must verify the users. If the combination of username and password is not correct or no such user exists in the system's database, a pop-up window or a notification in the form will notify them that there is an error with the provided credentials and the application will terminate.
Business Rules	

2.2.2 User

Description	The object represents users who use this program. The user will operate trading action base on trading broker, their strategy and coin price.
Attributes	Id: *int* Username: *String*
Responsibilities	The object stores each user information in database. Initiating the addition of a new, or the removal of an existing, trading broker
Business Rules	

2.2.3 Trade broker

Description	The object represents every trade broker managed by the system.
Attributes	Id: int Name: string Username: string Coin_interested_list: array
Responsibilities	The object must define trading brokers, add the name of the broker. The object also must add a list of cryptocoins it is interested in getting prices for, attach a trading strategy to the broker. If the broker's name already exists, a message is displayed and the broker is not added.
Business Rules	

2.2.4 Strategy

Description	The object represents the strategies that associate each trading broker
Attributes	Strategy: array

Responsibilities	The object attaching a trading strategy to each trading broker (the logic of the trading strategy involves cryptocoin prices that the trading client has declared interest in.
Business Rules	

2.2.5 Data retrieval

Description	The object represents an api object that get coin data from CoinGecko website.
Attributes	Coinprice: dictionary
Responsibilities	The object must retrieve cryptocoin price data from CoinGecko's site; notifying each broker with the cryptocoin prices they declared interest in. once the data are fetched from CoinGecko, each data item (i.e., cryptocoin price) is sent to the right trading broker.
Business Rules	

2.2.6 Trade

Description	The object represents every trade made by users.
Attributes	
Responsibilities	The object must perform trade for users. Once the user has added trading brokers, and a list of cryptocoins for each trading broker and the trading strategy to be used, then, the user can initiate the trading by pressing the "Perform Trade" button.
Business Rules	

2.2.7 Trading Action

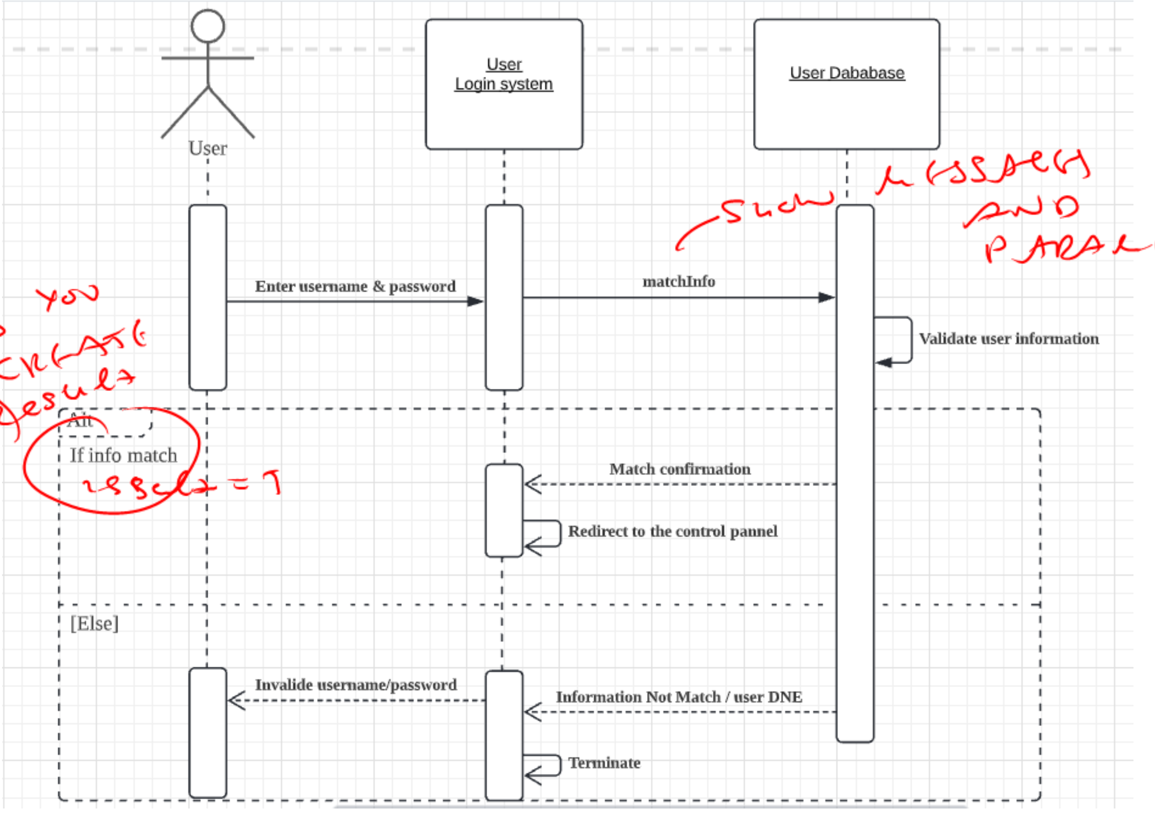
Description	The object represents main user interface that displaying the trading action for all trading clients
Attributes	
Responsibilities	The object must display on the UI the trading log and the trading activity.
Business Rules	

3 Sequence and Activity Diagrams

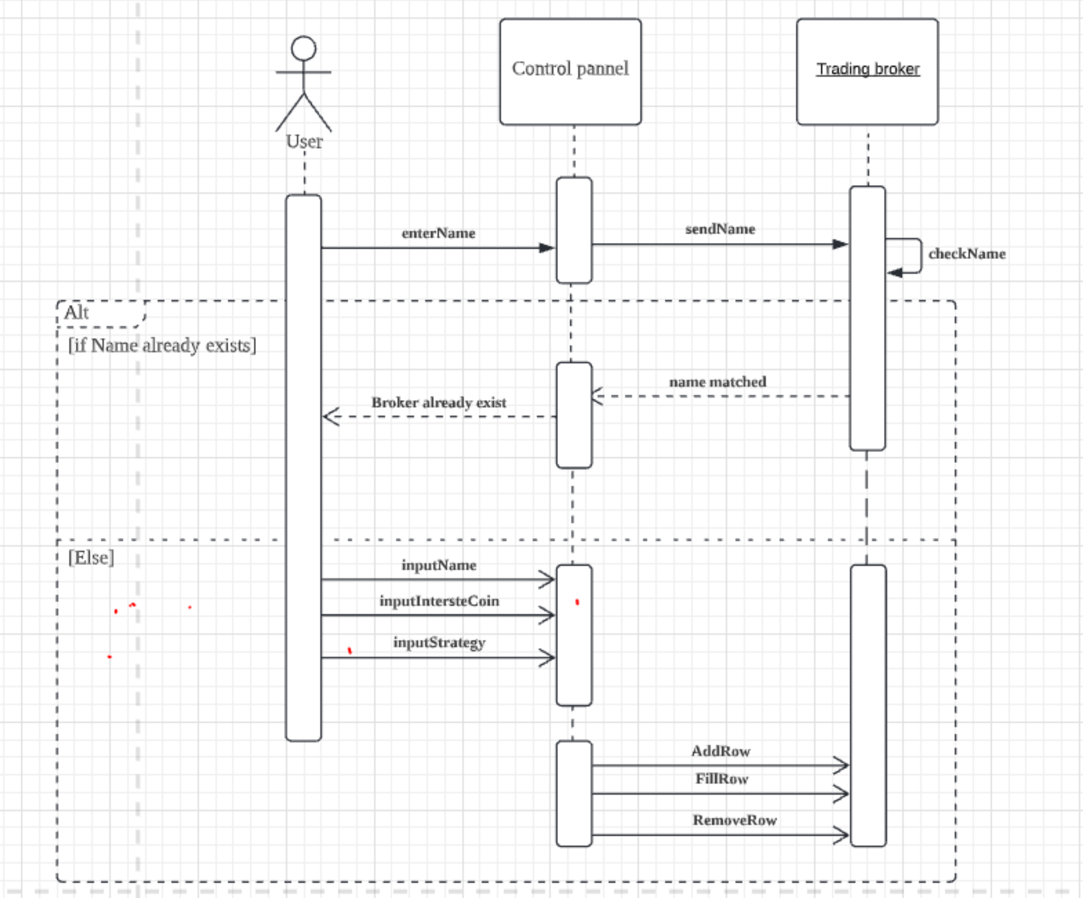
3.1 Sequencing Diagrams

For each use case diagram introduce the corresponding sequence diagram. Make sure you identify and associate each sequence diagram with the proper use case by maintaining unique Identifiers for use cases. Refer to the project description for which of the scenario you need to write collaboration diagrams. Provide one sub-subsection for each sequence diagram that corresponds to a Use case

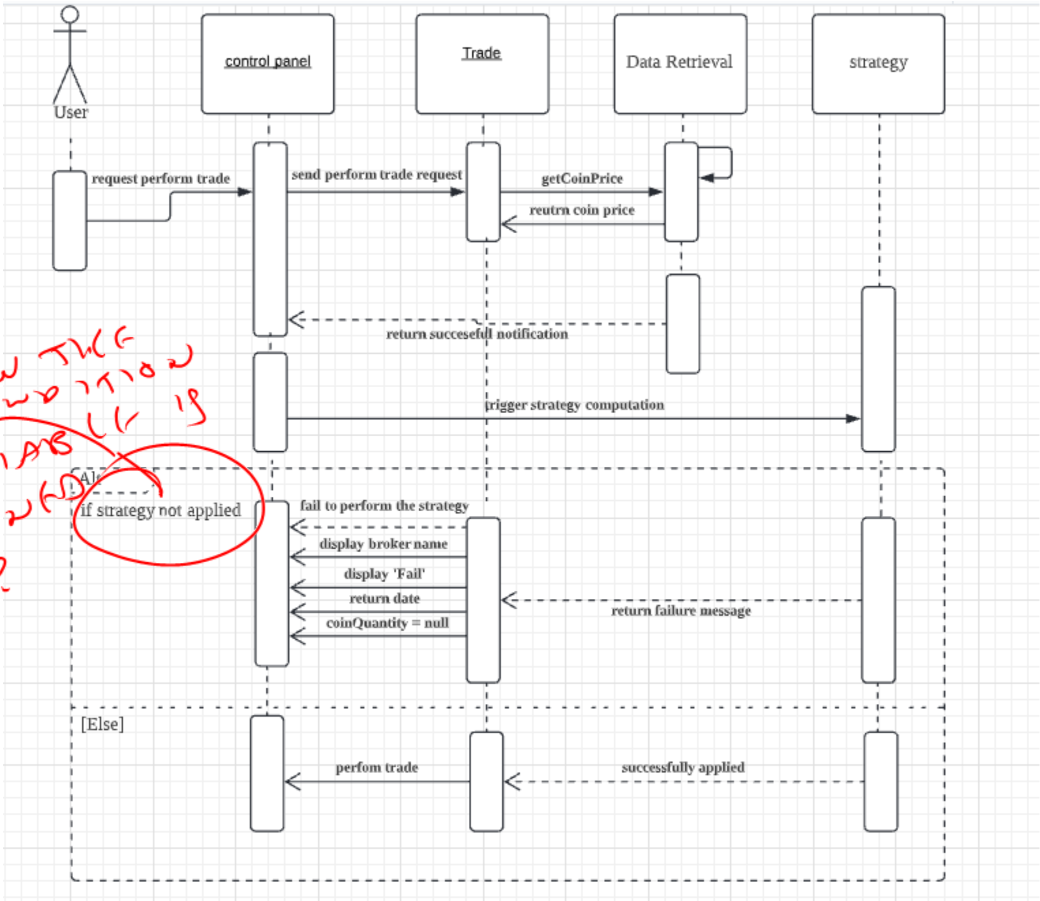
3.1.1 Sequence Diagram for Use-Case-1


User
User
Login system
User Dababase
Enter username & password
matchInfo
Validate user information
Alt
If info match
Match confirmation
Redirect to the control pannel
[Else]
Invalide username/password
Information Not Match / user DNE
Terminate
Check things you create results e-g
Show message and parameter
result = T

3.1.2 Sequence Diagram for Use-Case-2

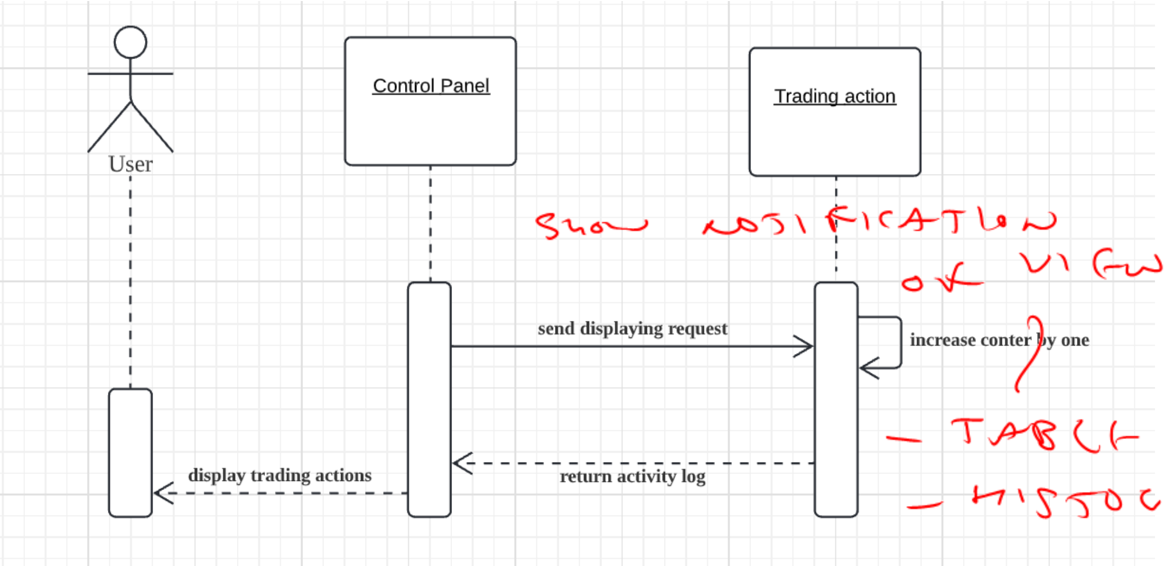


3.1.3 Sequence Diagram for Use-Case-3



how THE CONDITION VARIABLE IS OBTAINED ?

3.1.4 Sequence Diagram for Use-Case-4



SHOW NOTIFICATION OR VIEWING

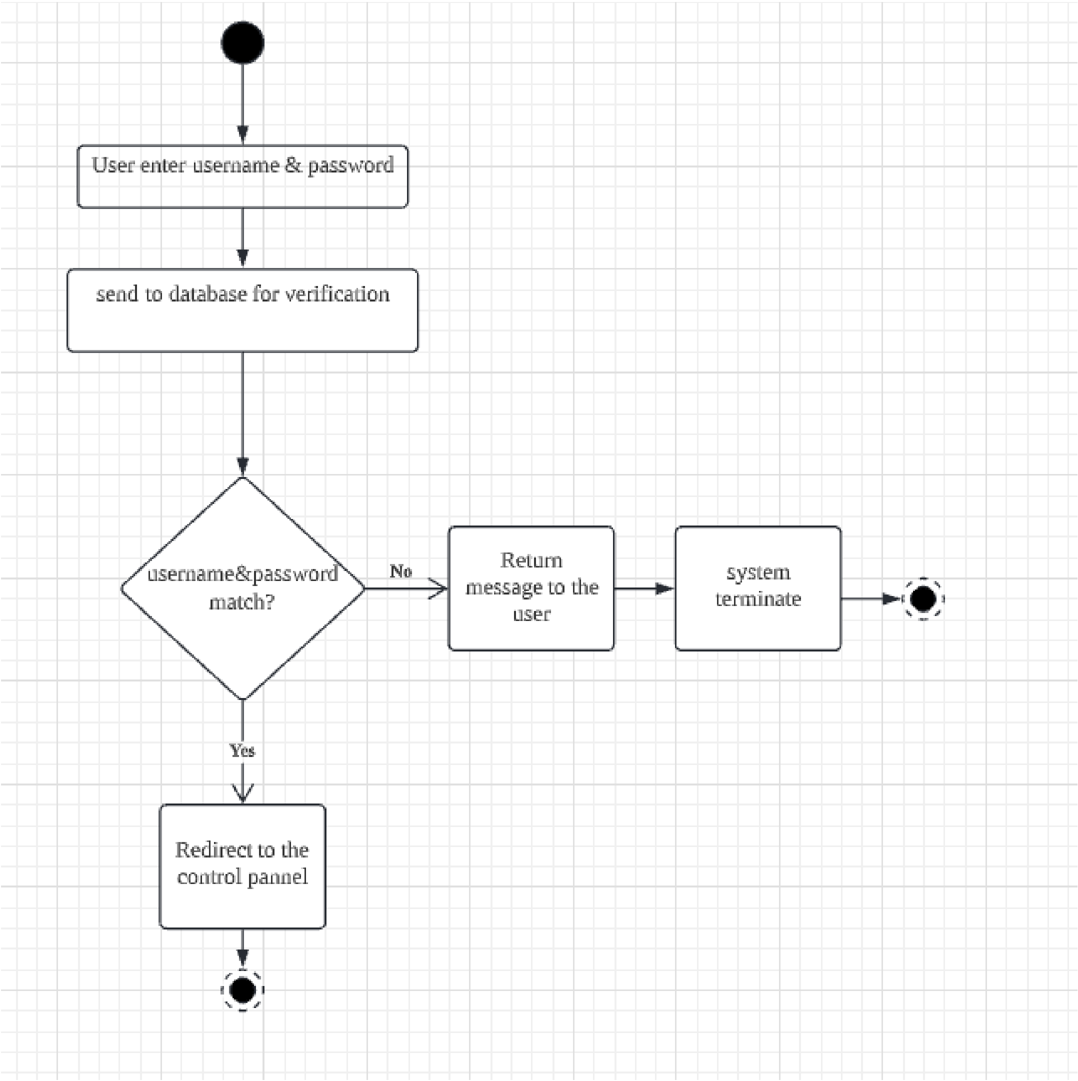
?

- TABLE
- HISTORICAL

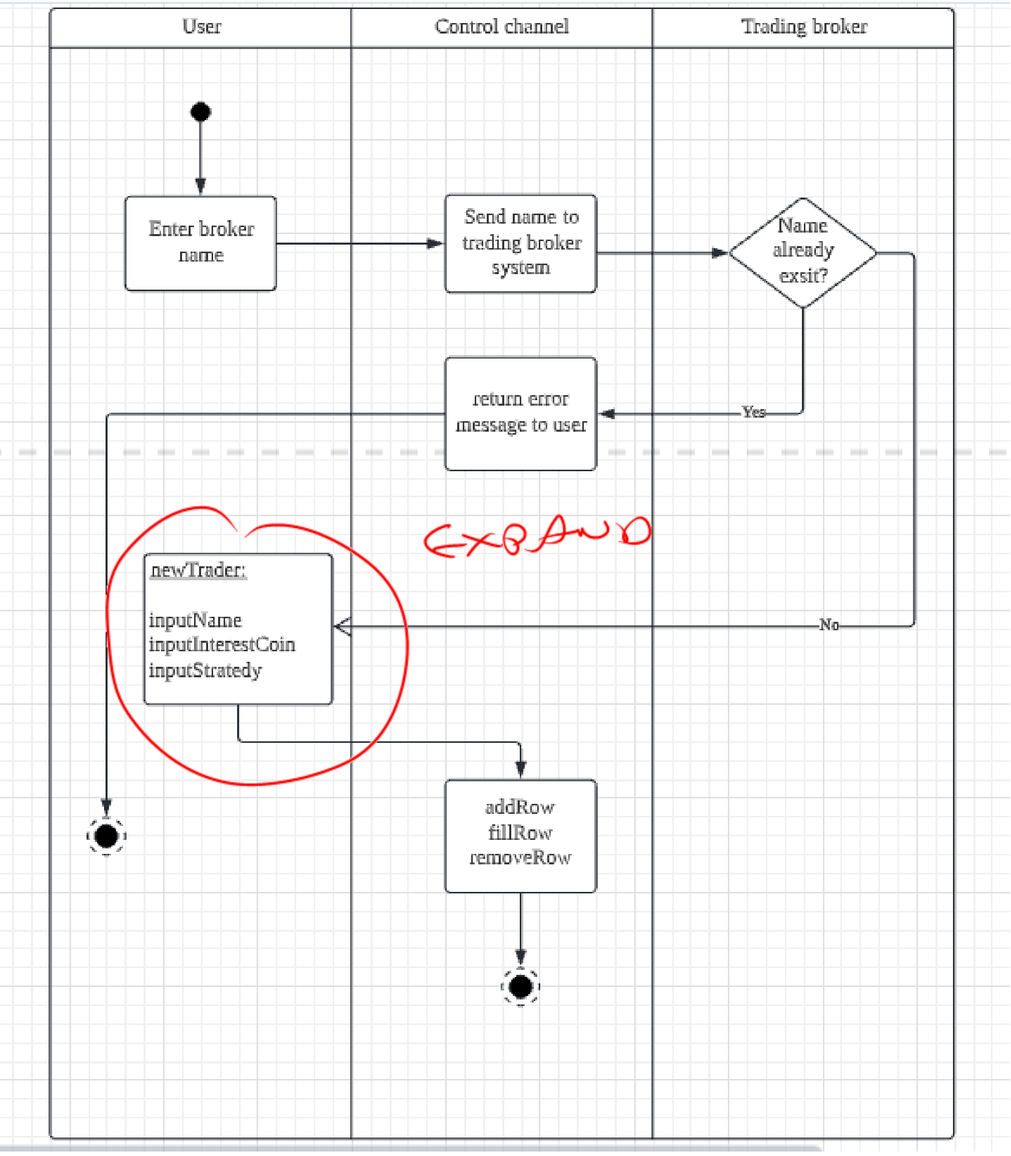
3.2 Activity Diagrams

For each use case introduce the corresponding activity diagram. Make sure you identify and associate each activity diagram with the proper use case by maintaining unique Identifiers for use cases. Refer to the project description for which of the *scenario you need to write collaboration diagrams. Provide one sub-subsection for each activity diagram that corresponds to a Use case*

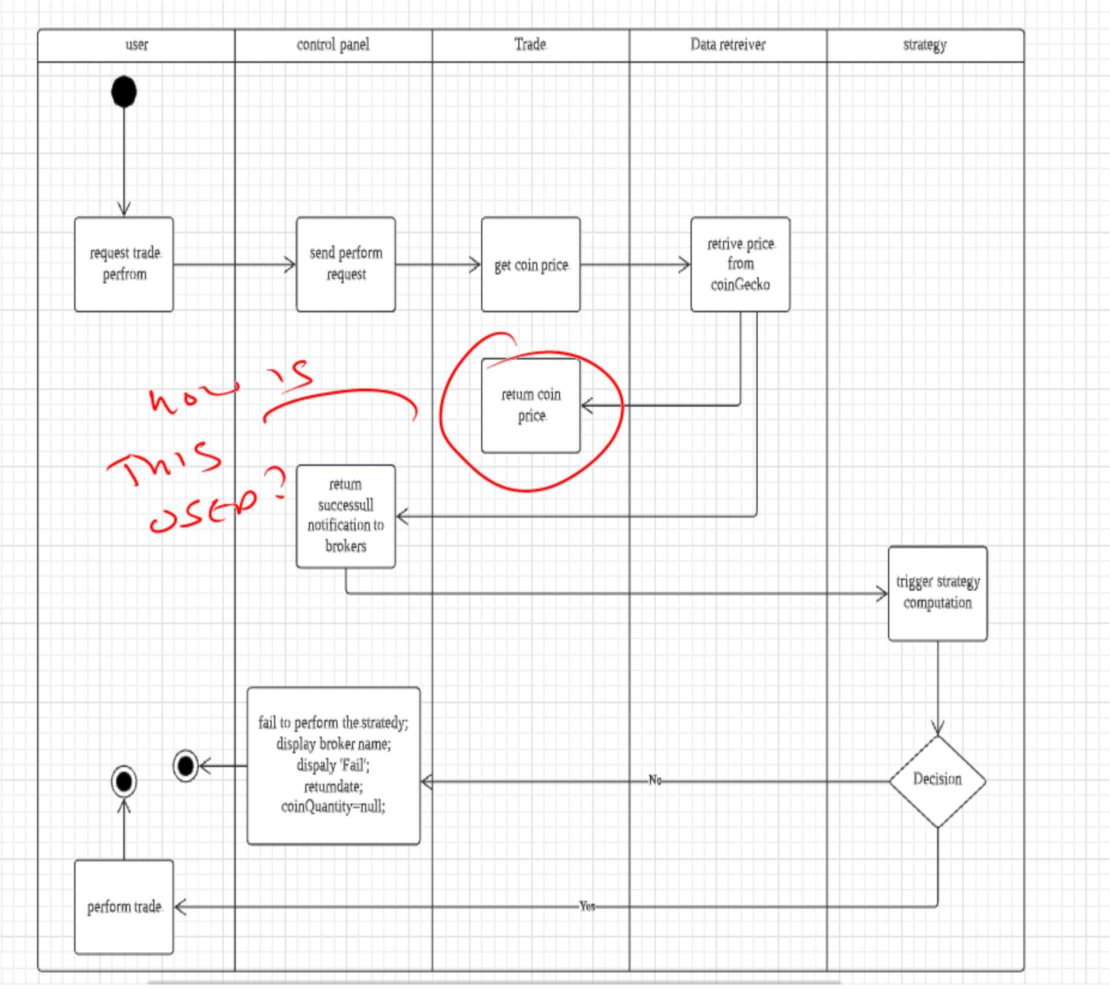
3.2.1 Activity Diagram for Use-Case-1


User enter username & password
send to database for verification
username&password match?
No
Return message to the user
system terminate
Yes
Redirect to the control pannel

3.2.2 Activity Diagram for Use-Case-2

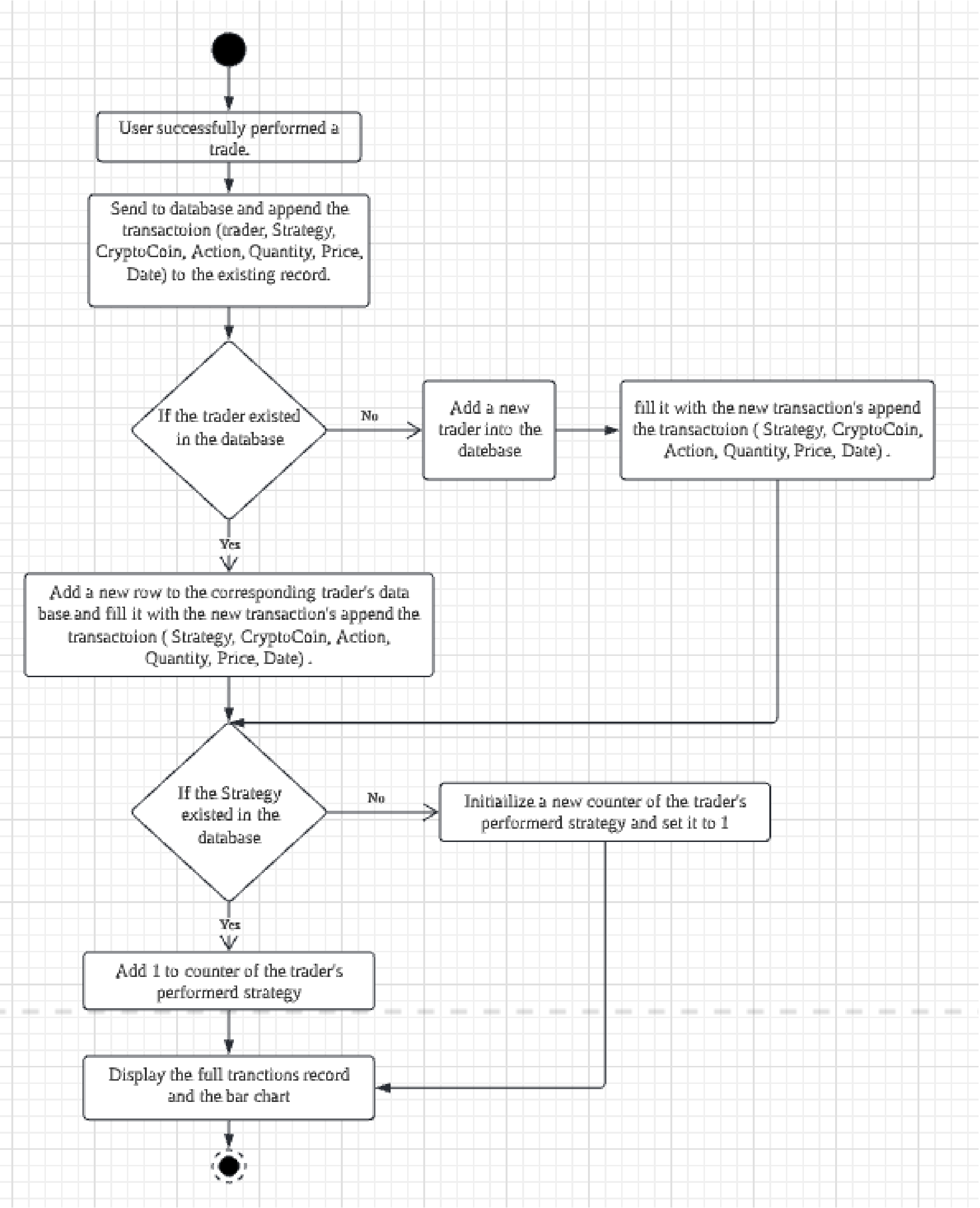


3.2.3 Activity Diagram for Use-Case-3


user
control panel
Trade
Data retreiver
strategy
request trade perfrom
send perform request
get coin price
retrive price from coinGecko
return coin price
return successull notification to brokers
trigger strategy computation
Decision
No
Yes
fail to perform the stratedy; display broker name; dispaly 'Fail'; returndate; coinQuantity=null;
perform trade

how is This used?

3.2.4 Activity Diagram for Use-Case-4


User successfully performed a trade.
Send to database and append the transactuion (trader, Strategy, CryptoCoin, Action, Quantity, Price, Date) to the existing record.
If the trader existed in the database
No
Add a new trader into the datebase
fill it with the new transaction's append the transactuion (Strategy, CryptoCoin, Action, Quantity, Price, Date) .
Yes
Add a new row to the corresponding trader's data base and fill it with the new transaction's append the transactuion (Strategy, CryptoCoin, Action, Quantity, Price, Date) .
If the Strategy existed in the database
No
Initiailize a new counter of the trader's performerd strategy and set it to 1
Yes
Add 1 to counter of the trader's performerd strategy
Display the full tranctions record and the bar chart

4 Non-Functional Requirements Specification

4.1 Overview

The non-functional requirements of the software contain a variety of features, which include enabling technologies, capacity planning, network, work station, and operational parameters. These features will ensure the software can function as expected and be successfully operated by the users.

4.2 Enabling Technologies - Target Development Environment

The main frame of the software will be developed using Java under Windows and MacOS systems. Depending on developers' preference, Visual Studio Code and Eclipse will be the primary code composers. MySQL database will be integrated into the system for the purpose of storing transactions.

4.3 Capacity Planning - Permanent Storage

The main frame of the software after installation will be anticipated to take up below 100 MB. Though all transactions will be recorded on the cloud server, as the record of transactions and the number of users increases on the offline platform, the program will gradually grow larger. 1GB of free storage will be strongly recommended for long-term usage of the software.

4.4 Network

As described in the introduction, the program fetches real-time cryptocurrency prices and is required to connect database on server, stable internet would be mandatory. Most modern networks should be able to satisfy the requirement.

4.5 Work Station

The software is intended to run smoothly on the two mainstream platforms, Windows and macOS. The minimum requirement for the version of Windows system accepted is Windows 7 and the minimum requirement for the version of macOS system accepted is MacOS X 10.0. Most modern processors and RAM will satisfy the requirements of the software. Also, a display setting of 1024 x 768 resolution and a 16-bit color palette should be used.

4.6 Operational Parameter

4.6.1 Usability

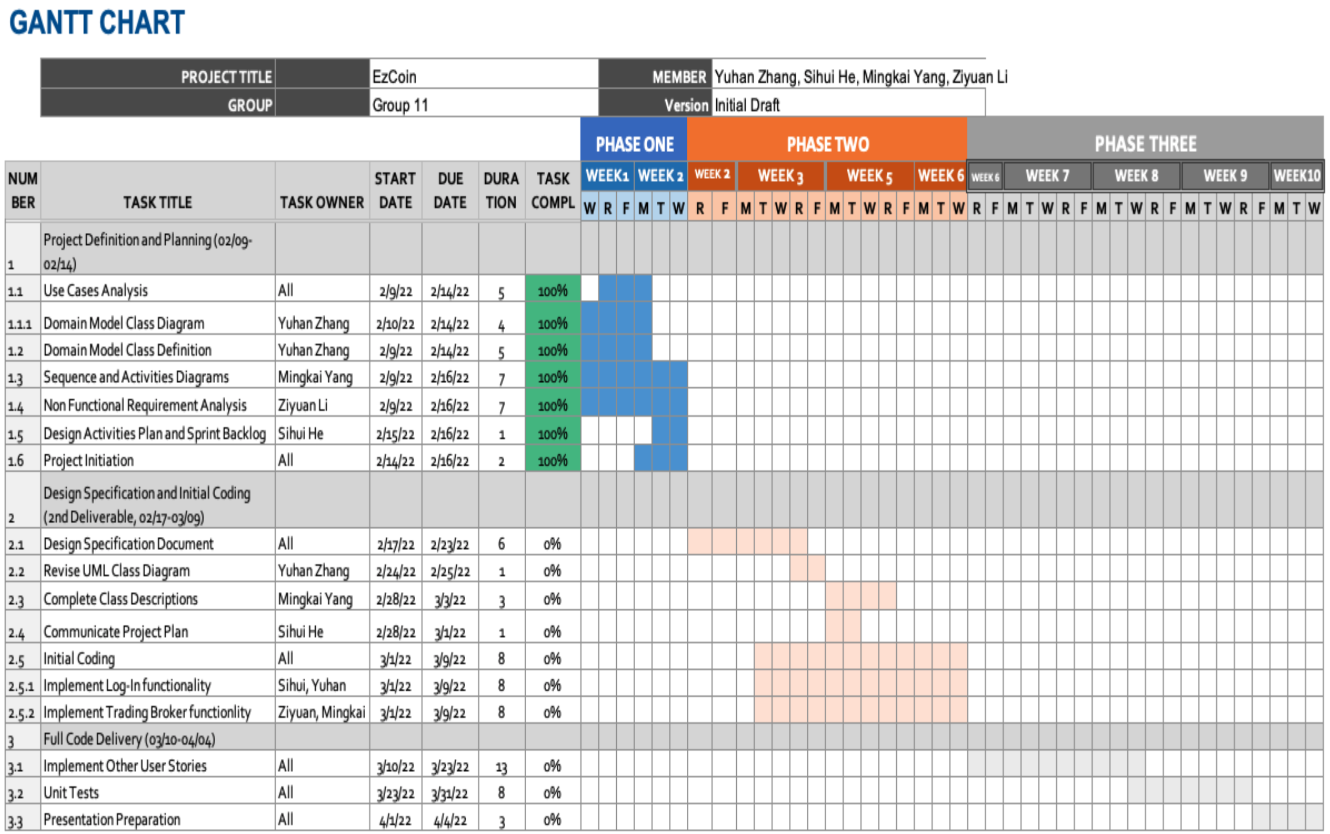
Anticipated users are brokers with no programming experience but with regular computer operation experience. The UI is designed to ensure that the user should have no major optical difficulties seeing 12 font characters. Since the coin's name will be abbreviated and a bar chart will be shown, the users are expected to have a certain degree of statistical and cryptocurrency-related knowledge.

Reliability

The system should be up and running 24/7, because the cryptocurrency market is up for the same schedule. Backup and logs should be stored and updated at regular intervals, which is every 24 hours. Notification about maintenance and upgrade will be sent in advance and take place during 3 – 4 AM EST.

5 Activities Plan

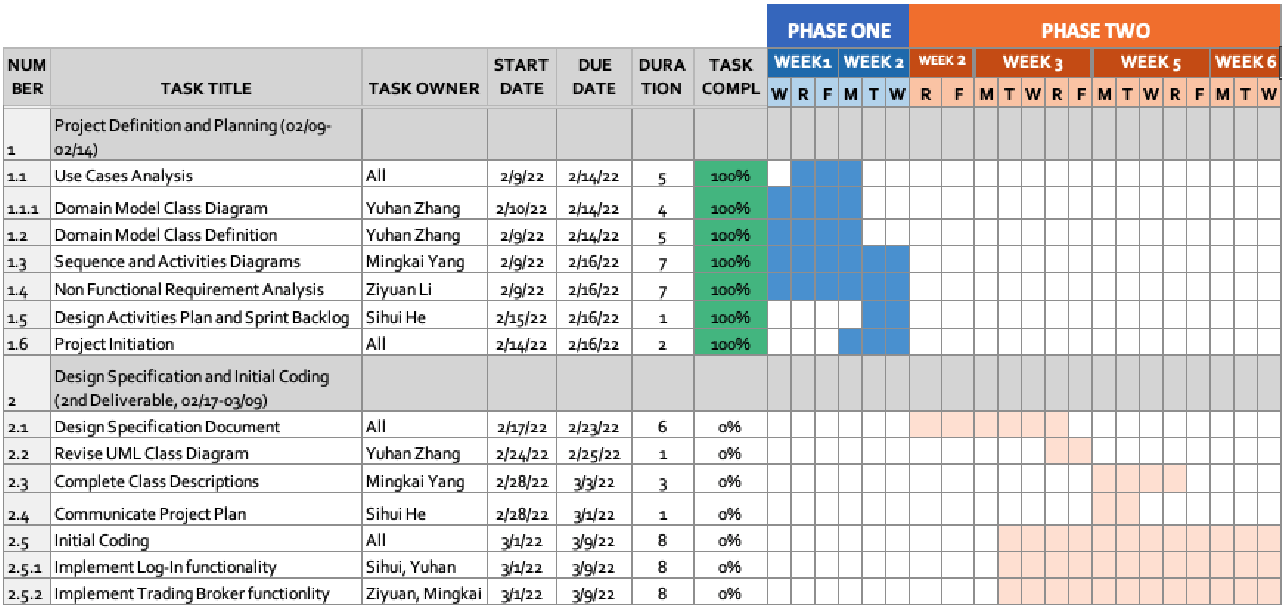
5.1 Gantt Chart



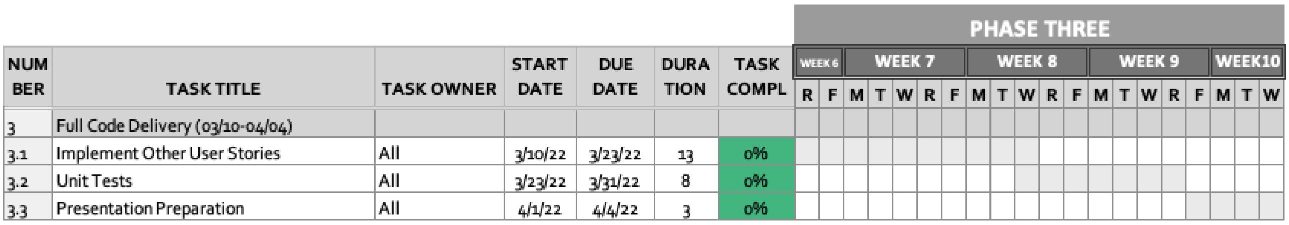
GANTT CHART

PROJECT TITLE	EzCoin	MEMBER	Yuhan Zhang, Sihui He, Mingkai Yang, Ziyuan Li
GROUP	Group 11	Version	Initial Draft

NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	TASK COMPL
1	Project Definition and Planning (02/09-02/14)					
1.1	Use Cases Analysis	All	2/9/22	2/14/22	5	100%
1.1.1	Domain Model Class Diagram	Yuhan Zhang	2/10/22	2/14/22	4	100%
1.2	Domain Model Class Definition	Yuhan Zhang	2/9/22	2/14/22	5	100%
1.3	Sequence and Activities Diagrams	Mingkai Yang	2/9/22	2/16/22	7	100%
1.4	Non Functional Requirement Analysis	Ziyuan Li	2/9/22	2/16/22	7	100%
1.5	Design Activities Plan and Sprint Backlog	Sihui He	2/15/22	2/16/22	1	100%
1.6	Project Initiation	All	2/14/22	2/16/22	2	100%
2	Design Specification and Initial Coding (2nd Deliverable, 02/17-03/09)					
2.1	Design Specification Document	All	2/17/22	2/23/22	6	0%
2.2	Revise UML Class Diagram	Yuhan Zhang	2/24/22	2/25/22	1	0%
2.3	Complete Class Descriptions	Mingkai Yang	2/28/22	3/3/22	3	0%
2.4	Communicate Project Plan	Sihui He	2/28/22	3/1/22	1	0%
2.5	Initial Coding	All	3/1/22	3/9/22	8	0%
2.5.1	Implement Log-In functionality	Sihui, Yuhan	3/1/22	3/9/22	8	0%
2.5.2	Implement Trading Broker functionlity	Ziyuan, Mingkai	3/1/22	3/9/22	8	0%
3	Full Code Delivery (03/10-04/04)					
3.1	Implement Other User Stories	All	3/10/22	3/23/22	13	0%
3.2	Unit Tests	All	3/23/22	3/31/22	8	0%
3.3	Presentation Preparation	All	4/1/22	4/4/22	3	0%



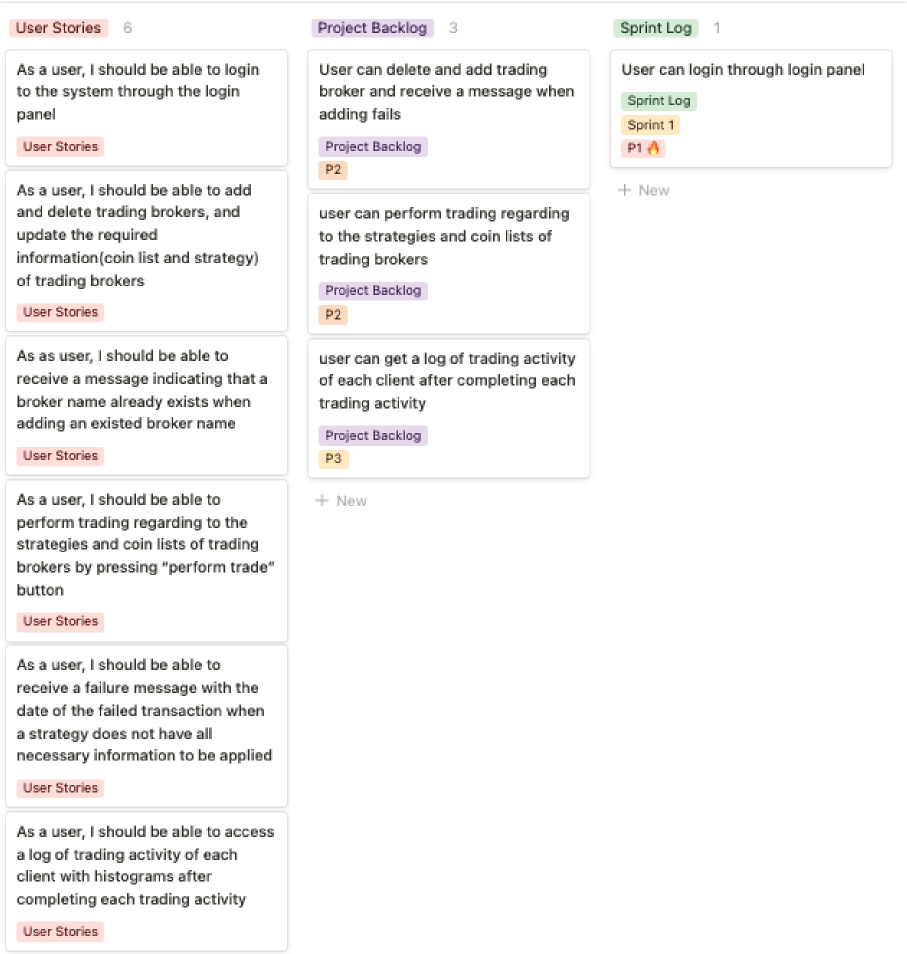
NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	TASK COMPL	PHASE ONE						PHASE TWO														
							WEEK1			WEEK 2			WEEK 2		WEEK 3					WEEK 5					WEEK 6		
							W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W
1	Project Definition and Planning (02/09-02/14)																										
1.1	Use Cases Analysis	All	2/9/22	2/14/22	5	100%		■	■	■																	
1.1.1	Domain Model Class Diagram	Yuhan Zhang	2/10/22	2/14/22	4	100%	■	■	■	■																	
1.2	Domain Model Class Definition	Yuhan Zhang	2/9/22	2/14/22	5	100%	■	■	■	■																	
1.3	Sequence and Activities Diagrams	Mingkai Yang	2/9/22	2/16/22	7	100%	■	■	■	■	■	■															
1.4	Non Functional Requirement Analysis	Ziyuan Li	2/9/22	2/16/22	7	100%	■	■	■	■	■	■															
1.5	Design Activities Plan and Sprint Backlog	Sihui He	2/15/22	2/16/22	1	100%					■	■															
1.6	Project Initiation	All	2/14/22	2/16/22	2	100%				■	■	■															
2	Design Specification and Initial Coding (2nd Deliverable, 02/17-03/09)																										
2.1	Design Specification Document	All	2/17/22	2/23/22	6	0%							■	■	■	■	■	■									
2.2	Revise UML Class Diagram	Yuhan Zhang	2/24/22	2/25/22	1	0%												■	■								
2.3	Complete Class Descriptions	Mingkai Yang	2/28/22	3/3/22	3	0%														■	■	■	■				
2.4	Communicate Project Plan	Sihui He	2/28/22	3/1/22	1	0%														■	■						
2.5	Initial Coding	All	3/1/22	3/9/22	8	0%										■	■	■	■	■	■	■	■	■	■	■	■
2.5.1	Implement Log-In functionality	Sihui, Yuhan	3/1/22	3/9/22	8	0%										■	■	■	■	■	■	■	■	■	■	■	■
2.5.2	Implement Trading Broker functionlity	Ziyuan, Mingkai	3/1/22	3/9/22	8	0%										■	■	■	■	■	■	■	■	■	■	■	■



NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	TASK COMPL	PHASE THREE																			
							WEEK 6		WEEK 7					WEEK 8					WEEK 9					WEEK10		
							R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W
3	Full Code Delivery (03/10-04/04)																									
3.1	Implement Other User Stories	All	3/10/22	3/23/22	13	0%	■	■	■	■	■	■	■	■	■	■										
3.2	Unit Tests	All	3/23/22	3/31/22	8	0%										■	■	■	■	■	■	■				
3.3	Presentation Preparation	All	4/1/22	4/4/22	3	0%																	■	■	■	■

5.2 Project Backlog and Sprint Backlog

In this Section, and assuming you follow a Scrum process model, provide a list of product backlog items so that you can select items for your Sprint backlog. Make sure the product backlog list and the tasks in each product backlog item are consistent with the Gantt Chart in Section 6.1. above.



User Stories 6

As a user, I should be able to login to the system through the login panel

User Stories

As a user, I should be able to add and delete trading brokers, and update the required information(coin list and strategy) of trading brokers

User Stories

As as user, I should be able to receive a message indicating that a broker name already exists when adding an existed broker name

User Stories

As a user, I should be able to perform trading regarding to the strategies and coin lists of trading brokers by pressing "perform trade" button

User Stories

As a user, I should be able to receive a failure message with the date of the failed transaction when a strategy does not have all necessary information to be applied

User Stories

As a user, I should be able to access a log of trading activity of each client with histograms after completing each trading activity

User Stories

Project Backlog 3

User can delete and add trading broker and receive a message when adding fails

Project Backlog

P2

user can perform trading regarding to the strategies and coin lists of trading brokers

Project Backlog

P2

user can get a log of trading activity of each client after completing each trading activity

Project Backlog

P3

+ New

Sprint Log 1

User can login through login panel

Sprint Log

Sprint 1

P1 🔥

+ New

5.3 Group Meeting Logs

In this Section you write minutes of each meeting, listing the attendance, what the topics of discussion in the meeting were, any decisions that were made, and which team members were assigned which tasks. These minutes must be submitted with the project report in each deliverable and will provide input to be used for the overall assessment of the project.

Present Group Members	**Meeting Date**	**Issues Discussed / Resolved**
All presented	Feb.6	First meeting, general discussion, task splitting and assigning
All presented	Feb.14	Discussion about use cases

6 Test Driven Development

Initial test cases will be provided in the form of a table as follows:

Test ID	The unique Id of the test case

Category	Which part of the system is tested (*e.g. evaluation of user credentials stored on file or DB*)
Requirements Coverage	The unique ID of the requirement tested (*e.g. UC1-Successful-User-Login*)
Initial Condition	Initial conditions required for the test case to run (*e.g. the system has been initiated and runs*)
Procedure	The list of steps required for this test case (*e.g.* *1. The user selects login* *2. The user provides a user name* *3. The user provides a password* *4. The user logs-in into the system and is presented with the main UI window*)
Expected Outcome	The expected outcome of the test case (*e.g. the login form closes, and the user is presented with the main UI window*)
Notes	Any other notes you may want to add for this test case, which are also reflected in the requirements specification (*e.g. the user should provide only alphanumeric user names and passwords without any special characters*)

7 Domain Dictionary (optional and as required)

7.1 Terms and Abbreviations

Term	**Definition**
Place term here	*Place a definition of the term here. Make the definition short and concise and consistent with other terms. Only used terms defined elsewhere in the domain dictionary.*
Place synonym here	*This is a synonym of <another term>*