Process MeNtOR 3.o

Uni-SEP

Cryptocoin Trading System

Design Document

Version:	1.5
Print Date:	March 16th, 2022
Release Date:	
Release State:	
Approval State:	Core
Approved by:	
Prepared by:	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Tamer Ali Mohamed
Reviewed by:	
Path Name:	
File Name:	
Document No:	

object oriented pty. ltd.

Document Change Control

Version	Date	Authors	Summary of Changes
1.1	March 5th, 2022	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	Write a rough draft about the major design decisions with brainstormed ideas about the component diagram. Updated group meetings logs.
1.2	March 8th, 2022	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	Edited and finalized the major design decisions and did the introduction beside working on the architecture. Updated group meetings logs.
1.3	March 12th, 2022	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	finalized the component diagram and inserted the product backlog and sprint backlog. Updated group meetings logs.
1.4	March 13th, 2022	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	Done with the architectural styles beside starting the test driven development. Updated group meetings logs.
1.5	March 15th, 2022	Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	Finalized the test driven development and edited the whole project. Updated group meetings logs.

Document Sign-Off

Name (Position)	Signature	Date
Ian Guenther Green	IGG	March 5th, 2022
Vicky Jiang	VJ	March 5th, 2022
Hala Elewa	HE	March 5th, 2022
Ali Mohamed	AM	March 5th, 2022
Ian Guenther Green	IGG	March 8th, 2022
Vicky Jiang	VJ	March 8th, 2022
Hala Elewa	HE	March 8th, 2022
Ali Mohamed	AM	March 8th, 2022
Ian Guenther Green	IGG	March 12th, 2022
Vicky Jiang	VJ	March 12th, 2022
Hala Elewa	HE	March 12th, 2022

Ali Mohamed	AM	March 12th, 2022
Ian Guenther Green	IGG	March 13th, 2022
Vicky Jiang	VJ	March 13th, 2022
Hala Elewa	HE	March 13th, 2022
Ali Mohamed	AM	March 13th, 2022
Ian Guenther Green	IGG	March 15th, 2022
Vicky Jiang	VJ	March 15th, 2022
Hala Elewa	HE	March 15th, 2022
Ali Mohamed	AM	March 15th, 2022

Contents

1 Introduction

1.1 Purpose

This document will outline the design of the Cryptocurrency trading system. This includes how the Cryptocurrency trading system gets data about the prices from the CoinGecko cryptocurrency repository. As well as how it processes the trades to output in visual format in a histogram and table viewer. This document will include the major design decisions of the system and provide a component diagram, activities plan, and also a comprehensive list of test cases to make sure the system functions as required.

1.2 Overview

The SDD document contains the following information:

1. Major design decisions of the system and specify what classes we have used and how the data is described as a data coupling and check if the system is highly cohesive or not.
2. Component diagram that outlines the necessary classes and visualizes the different architectural styles that will be used in building the client software.
3. Activities planned and completed by team members.
4. Test cases that are important to take care of to avoid any misunderstanding.

1.3 Resources - References

CG-API: https://www.coingecko.com/api/documentations/v3

CG-TERMS: https://www.coingecko.com/en/glossary

Eclipse: http://www.eclipse.org/downloads/index.php

Maven: https://maven.apache.org/download.cgi

RESTAPI: https://www.w3schools.in/restful-web-services/intro/

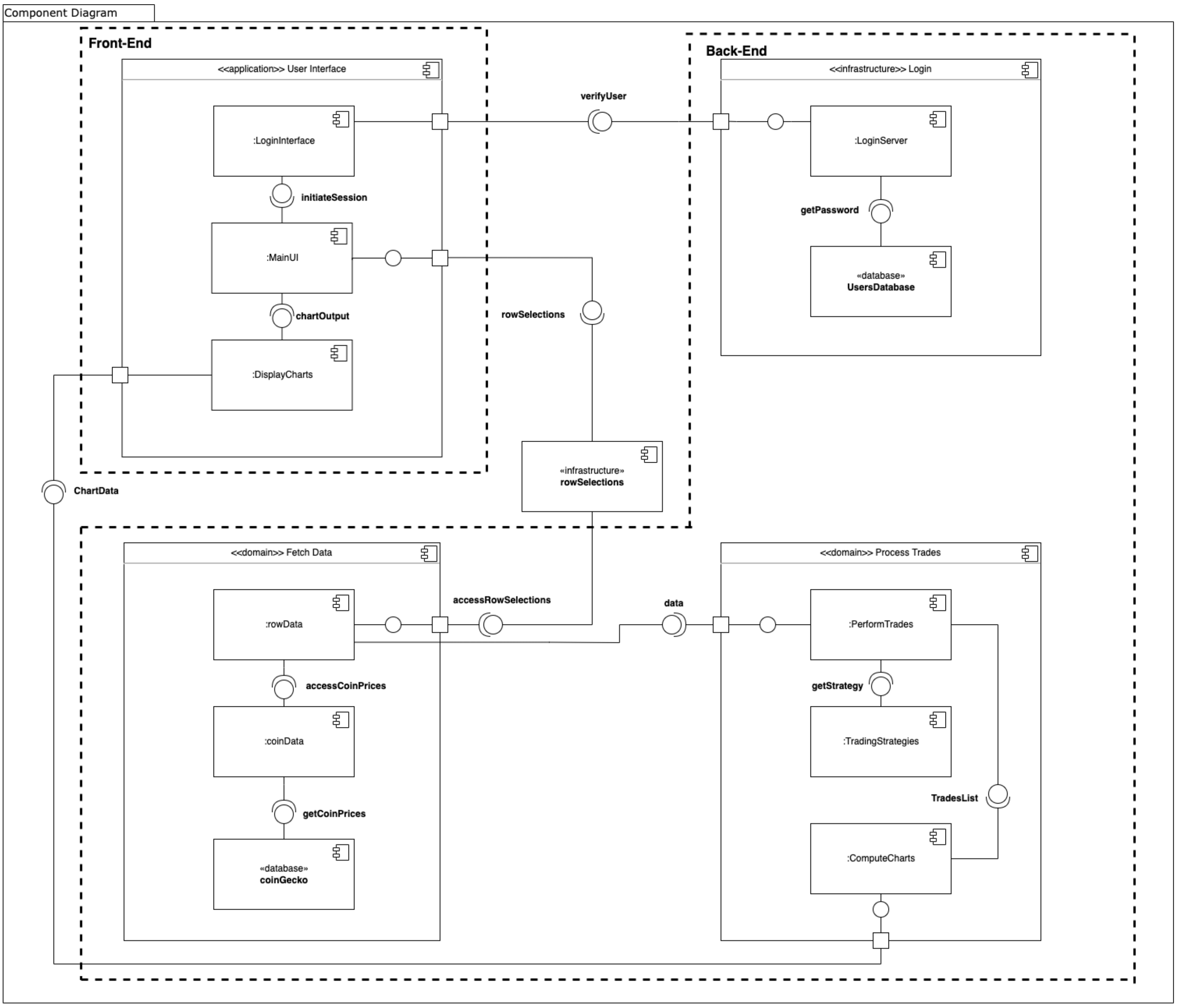
2 Major Design Decisions

The classes were organized into User, User_Database, User_Interface, CryptoCurrency, UI, Trading_Broker, Selector, Trade_Result and Trading_Strategy. This was done in order to have low coupling as the classes interact through passing data as a parameter between each other. For instance, Trading_Broker passes coinList to Trading_Strategy, for evaluation. This can be described as data coupling, as only the required data(coinList) is being passed as a parameter to Trading_Strategy.

All modules lead towards delivering a single output of the Crypto Trade Analysis in a data chart. This means the system is highly cohesive, as each class executes different, individual tasks while passing the output to other classes as inputs for the next tasks. This system can be described as a functional system as the same data type is being passed along, while the entire system contributes to one single output.

3 Architecture

Component Diagram



Classes and their Corresponding Methods

Component	Class	Methods
LogIn	LoginServer	getUser(): string
		getPass(): string
		validate(user): boolean
User Interface	Login	getInstance(): login
	MainUI	getInstance(): MainUI
	Session	getInstance(): session
	ChartType	getType(): string
	Table	setColumnNames(columnNames): void
		setData(data): void
	Histogram	setValues(strategy, amount): void
Fetch Data	DataFetcher	getDataForCrypto(id, date): object
		getPriceForCoin(id, date): double
		getMarketCapForCoin(id, date): double
		getVolumeForCoin(id, date): double
	Crypto	setCrypto(selectedCryptos): void
		getCryptos(): ArrayList
	TradeAmount	setAmount(selectedAmount): void
		getAmount(): int
	Strategy	setStrategy(selectedStrategy): void
		getStrategy(): string
Process Trades	PerformTrade	performTrade(Crypto, Strategy, TradeAmount): Result
	Result	getResult(): float

Interface Descriptions

Component Name	*Interface Name*	*Operation Signature*	*Description of the Operation*
LoginInterface	**initiateSession**	void startSession()	Called once the user's login credentials have been verified to initiate the main user interface of the trading program
MainUI	**rowSelections**	getRow(broker, coins[], strategy)	This function is called for each row in the table on the user interface. It passes the information in the row including the broker name, the coins list, and the strategy. The rowSelections is then used as a middle tier to facilitate getting data from the backend and processing trades.
		performTrades()	Calls this function to initiate the trading process
DisplayCharts	chartOutput	displayHistogram()	Called to render the histogram onto the main UI. Display strategies so that it outputs all the strategies and how many times they were used.
		displayTradeLogTable()	Called to render the trade log table onto the main UI.
LoginServer	validateUser	bool validateUser(string	Receives the username and password entered

		username, string password)	into the LoginInterface. Validates them and returns true if valid user and false if invalid user.
UsersDatabase (Note: this component is a database separate from main program)	getPassword	string getPassword(string username)	Receives the username, returns password associated with username if the user exists
rowSelections	accessRowSelections	void rowSelections(rows[broker, coins[], strategy])	Passes the rows entered by the user with the trading broker name, coins list, and strategy in a list
rowData	data	rowData(rows[broker, coins[], strategy])	Allows access to the user inputted data so that the trades can be performed on the data
		coinsData(coins[])	Gives the coins with the coin prices for the trade processing. coins[] will be a struct data structure with the name as one field and the price as a second field.
coinData	accessCoinPrices	getCoins(coinList[])	Allow for the coins to be inputted together as a list, in order for the coinData component to get the prices for the coins
coinGecko	getCoinPrices	float getPrice(coin)	Uses the coinGecko database to input the name of a cryptocoin and output the price

PerformTrades	TradesList	getTrades(TradesList)	The perform trade component performs trades based on the trading techniques. The getTrades function allows for these trades to be accessed in order to compute charts from the trades.
TradingStrategies	getStrategy	getStrategy(string strategyName)	Returns the trading strategy that is used to compute the trades made for each broker.
ComputeCharts		tableData()	Processes trade data so that it can be used to display a table containing the information regarding the broker name, strategy, buy/sell, and amount
		histogramData()	Processes data so that it is able to be visually displayed in a histogram.

Architectural Styles

For our system, we are using the following **three different architectural styles** to best set up our design:

1. Transactional Database Style

Transactional database-style structures a data store that resides at the center of this architecture and is accessed frequently by other components that create, read, update and/or delete data within the store. We are using a transactional database style for two components of our system. The first one will be the login database that our system uses to store the login information, such as the username and passwords, of all the users. Our user class will be interacting with this login database by only reading the data to see if the username-password combination that a user enters is valid.

In some cases, the software can access a central repository which is a passive data repository meaning that the software accesses the data independent of any changes to the data or action of the client software. Therefore, our second transactional database will be the Coin-Gecko repository. Our crypto coin class will be interacting with the Coin-Gecko database to read information about a specified coin such as the coin name and coin price. This will help determine whether trades can be made.

2. Model-View Controller

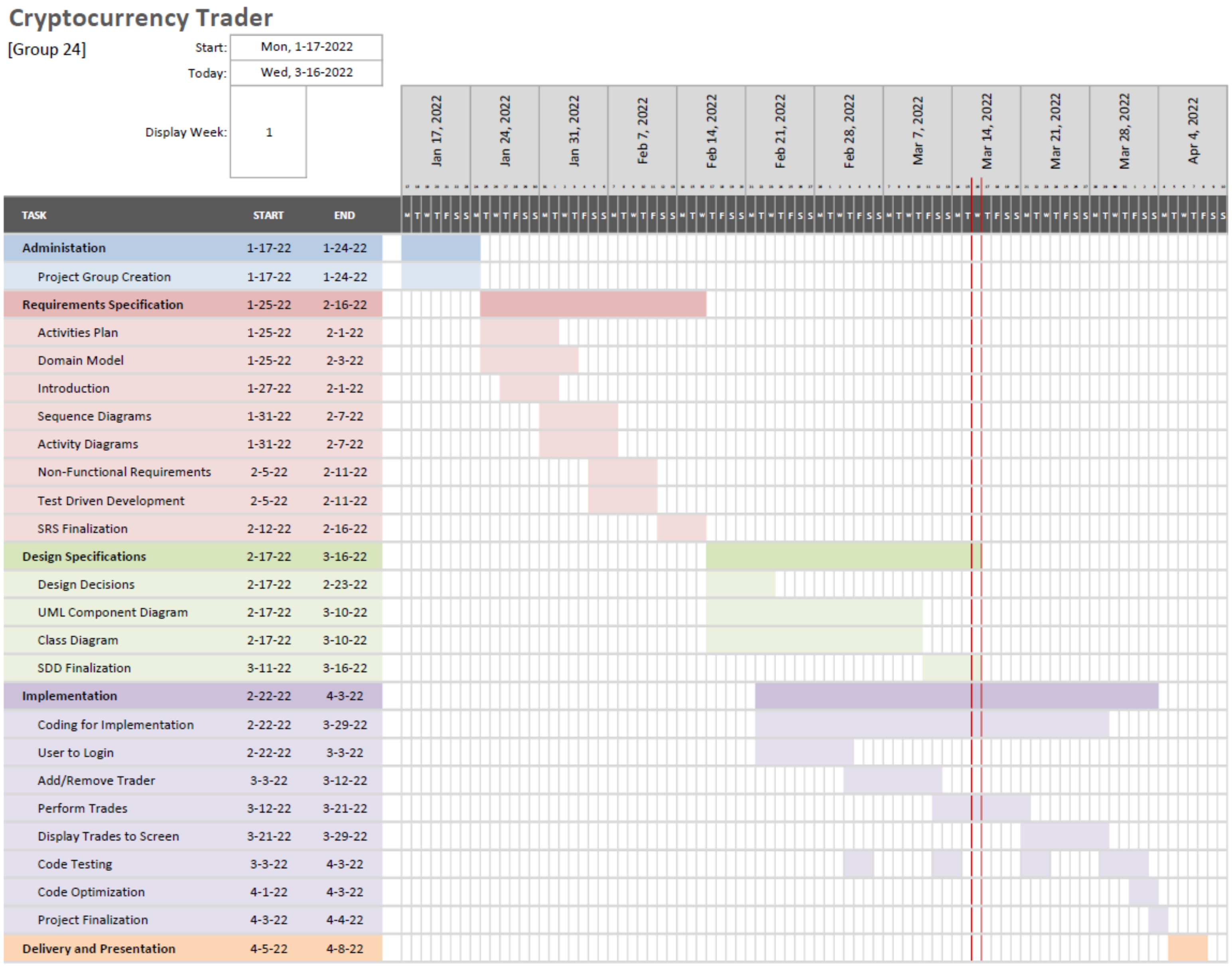
The model-view-controller is a decomposition of an interactive system broken down into three components: 1) a model containing the core functionality and data 2) one or more views displaying information to the user 3) one or more controllers that handle user input. The second architectural style in our software will be the model-view-controller style which will be used to notify our viewers (e.g. different displays). When a user presses the "Perform Trade" button on the main UI page, the analysis of the trades will be completed and the model will notify the viewers to display the information in two different ways: 1) in a trades table 2) in a histogram.

3. Tiered Hierarchy

The tiered architectures are a special kind of layered architecture for enterprise applications. Specifically, the three-tier client-server architecture design which includes 1) a user system interface 2) processing management 3) database management. The last architecture style that our software system is using is the three-tiered system. Our software system is designed in a tiered format to ensure that the user does not know what is happening in the back-end. The user will only have access to the front end which is the UI. The middle-tier will be our selector class that connects the selection of the user with the processor. The last tier is the back-end, the processor, which includes the perform trades and fetch data class.

4 Activities Plan

Gantt Chart



Cryptocurrency Trader

[Group 24]

Start:	Mon, 1-17-2022
Today:	Wed, 3-16-2022
Display Week:	1

TASK	START	END
Administation	1-17-22	1-24-22
Project Group Creation	1-17-22	1-24-22
Requirements Specification	1-25-22	2-16-22
Activities Plan	1-25-22	2-1-22
Domain Model	1-25-22	2-3-22
Introduction	1-27-22	2-1-22
Sequence Diagrams	1-31-22	2-7-22
Activity Diagrams	1-31-22	2-7-22
Non-Functional Requirements	2-5-22	2-11-22
Test Driven Development	2-5-22	2-11-22
SRS Finalization	2-12-22	2-16-22
Design Specifications	2-17-22	3-16-22
Design Decisions	2-17-22	2-23-22
UML Component Diagram	2-17-22	3-10-22
Class Diagram	2-17-22	3-10-22
SDD Finalization	3-11-22	3-16-22
Implementation	2-22-22	4-3-22
Coding for Implementation	2-22-22	3-29-22
User to Login	2-22-22	3-3-22
Add/Remove Trader	3-3-22	3-12-22
Perform Trades	3-12-22	3-21-22
Display Trades to Screen	3-21-22	3-29-22
Code Testing	3-3-22	4-3-22
Code Optimization	4-1-22	4-3-22
Project Finalization	4-3-22	4-4-22
Delivery and Presentation	4-5-22	4-8-22

Modification Date: 3/1/2022 4:08:00 PM

Project Backlog and Sprint Backlog

Backlog Item	Time (days) Estimate
~~Activities Plan~~	~~7~~
~~Domain Model~~	~~9~~
~~Introduction~~	~~5~~
~~Sequence Diagrams~~	~~7~~
~~Activity Diagrams~~	~~7~~
~~Non-Functional Requirements~~	~~6~~
~~Test Driven Development~~	~~14~~
~~Design Decisions~~	~~6~~
~~UML Component Diagram~~	~~21~~
~~Class Diagram~~	~~21~~
Creating Trading Strategies	5
Coding for Implementation	35
Allowing User to Login to System	9
Allowing User to Add/Remove a Trading Broker	9
Allow User to Perform Trades	9
Display Trades Made to Screen	8
Code Testing	14
Code Optimization	3
Project Finalization	2

Sprint Backlog

Create trading strategies, complete implementation of code (including allowing user to login to system, allowing user to add/remove rows in table, allowing user to perform trades and displaying trades on the screen)

Group Meeting Logs

Present Group Members	Meeting Date	Issues Discussed / Resolved
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	January 28th, 2022	First group meeting (1 hour) Introductions/getting to know each other/discussing strengths Sorting out initial logistics ● Creating shared folder on Google drive ● Communication channels ● How often to meet ● Timeline for when to finish each section by Went through project description and Deliverable 1 Discussed questions drawn from project description Decided order to complete tasks 1. Domain model first as a group 2. Sequence and activity diagrams 3. Remaining sections individually
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	January 31st, 2022	Group Meeting 2 (1 hour) Group research and study for how to make a domain model for a software system Brainstorming ideas for domain model
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	February 2nd, 2022	Group Meeting 3 (1 hour) Work session on domain model ● Vicky decided to meet with course TA to ensure our domain model was on the right track and ask clarification questions General discussion of progress made on project and how we will handle time before due date Divided sequence and activity diagrams by use case 1. Use case 1 - Ali 2. Use case 2 - Vicky 3. Use case 3 - Ian

		4. Use case 4 - Hala
Hala Elewa, Ian Guenther Green, Vicky Jiang	February 7th, 2022	Group Meeting 4 (1 hour) Discussion on progress of sequence and activity diagrams Vicky discussed information that TA provided regarding domain model Divided remaining tasks 1. Introduction - Hala 2. Non-Functional Requirements - Ali 3. Activities plan, Product Backlog and Sprint Backlog - Ian 4. Test Driven Development - Vicky
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	February 12th, 2022	Group Meeting 5 (1.5 hours) - discussion and work session Domain Model ● Vicky and Hala had worked on last parts of domain model prior to this meeting ● Full group performed final edits ● Vicky finished class descriptions Discussed where we are at with our respective sections ● Ali finished most of NFR - had some questions regarding system requirements etc. ● Hala finished introduction and rough draft of sequence diagram ● Ian has rough draft of sequence diagram and edits for domain model ● Vicky did domain model descriptions
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	February 13th, 2022	Group Meeting 6 (30 minutes) - brief section updates Update on where we are at with sections New section assignments ● Use case 4 sequence diagram - Vicky ● Use case 4 activity diagram - Ian ● Test driven development - Hala
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	February 14th, 2022	Group Meeting 7 (45 minutes) Group editing of entire document for submission

Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	February 26th, 2022	Group Meeting 8 (30 minutes) Discussed comments received on our first SRS submission
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	March 5th, 2022	Group Meeting 9 (1 hour) Looked into description for Deliverable 2 Brainstormed ideas for component diagram and major design decisions
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	March 8th, 2022	Group Meeting 10 (45 minutes) Split remaining parts and started to work on them ● Introduction - Hala ● Major Design Decisions - Ali ● Architecture - Ian and Vicky ● Activities Plan, Product Backlog, Sprint Backlog - Ian ● Test Driven Development - Hala
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	March 12th, 2022	Group Meeting 11 (2 hours) - discussion and work session Introduction completed Created system architecture Finished first version of component diagram Continue to work on test driven development
Hala Elewa, Ian Guenther Green, Vicky Jiang, Ali Mohamed	March 14th, 2022	Group Meeting 12 (1 hour, 45 minutes) Work session to finalize our parts ● Finalized architecture, component diagram, and interface descriptions - Ian and Vicky ● Finalized test driven development - Ali and Hala Started discussion on code implementation
Hala Elewa, Ian Guenther Green, Vicky Jiang	March 15th, 2022	Group Meeting 13 (1 hour 30 minutes) Group editing of entire SDD document for submission

5 Test-Driven Development

Test ID	UC1.1
Category	Username and password stored on file or DB
Requirements Coverage	UC1-Successful-User-Login
Initial Condition	void startSession(), The username and password are correct and working
Procedure	1. The user selects login. 2. The user provides a user name. 3. The user provides a password. 4. The user logs in to the system and is presented with the main UI window.
Expected Outcome	The login window closed and the main UI was displayed
Notes	The user should provide only alphanumeric characters for their username and password.

Test ID	UC1.2
Category	username and/or password entered not correct
Requirements Coverage	UC1-Unsuccessful-User-Login
Initial Condition	The username and password are invalid.
Procedure	1. The user provides a user name 2. The user provides a password 3. Notification that provided credentials are incorrect 4. Terminate program
Expected Outcome	Program termination
Notes	The user should provide only alphanumeric characters for their username and password.

Test ID	UC2.1
Category	Evaluation of displayed list of cryptos, correct trading name
Requirements Coverage	UC2.1-Crypto-List

Initial Condition	The system has been initiated and displayed on the UI
Procedure	1. The user inputs their name. 2. The user selects coins. 3. The user selects strategy. 4. The user clicks on it to perform trading. 5. Users can delete trading brokers by clicking 'remove row' 6. The list of available cryptos is displayed
Expected Outcome	The user is able to see the list
Notes	N/A

Test ID	UC2.2
Category	Evaluation of displayed list of cryptos
Requirements Coverage	UC1-Unsuccessful-trader-name
Initial Condition	Duplicate name not allowed.
Procedure	1. The user inputs their name. 2. The user selects coins. 3. The user selects strategy. 4. The user clicks on it to perform trading. 5. Users can delete trading brokers by clicking 'remove row'
Expected Outcome	An error message is displayed and the broker is not added.
Notes	

Test ID	UC3.1
Category	Evaluating trades
Requirements Coverage	UC3-CorrectCoins
Initial Condition	Selected coins and strategy
Procedure	1. User selects 'Preform Trade' 2. The server UI triggers the computation. 3. Get the prices for the selected coins. 4. Notify the client

Expected Outcome	updated information for each broker with the strategy that suits them. Trades made.
Notes	The broker should get only notifications about their chosen coins.

Test ID	UC3.2
Category	Evaluating trades
Requirements Coverage	UC3-IncorrectCoins
Initial Condition	The broker got notified about different coins other than the ones he chooses
Procedure	1. User selects 'Preform Trade' 2. The server UI triggers the computation. 3. Get the prices for the other coins. 4. Notify the client
Expected Outcome	Trade failed, Display a message to the client.
Notes	N/A

Test ID	UC3.3
Category	Evaluating trades
Requirements Coverage	UC3-EmptyRowstest, void preformTrade(rows[broker, coins, strategy])
Initial Condition	Rows are Empty.
Procedure	1. User selects 'Preform Trade' 2. The program checks for the data in the rows. 3. Error is displayed, prompting the user to fill the rows.
Expected Outcome	The user is prompted to fill the rows.
Notes	N/A

Test ID	UC4.1
Category	Creating a visual representation of trades(graph)
Requirements Coverage	Display the name, strategy, coins, action, quantity, unit-price, time-stamp in a histogram and a table.

Initial Condition	Trades have been executed, and are ready to be displayed.
Procedure	1. User selects 'Display Trades' 2. Successful trades for each strategy are counted. 3. The Histogram is created. 4. The histogram is displayed to main UI.
Expected Outcome	A histogram and a table are displayed in the main UI that has all of this information.
Notes	The table will consist of the most used strategy.