100/100

Good work!

Process MeNtOR 3.o

Uni-SEP

CoinMaster

Design Document

Version:	0.
Print Date:	Mar.16
Release Date:	
Release State:	Initial
Approval State:	Draft
Approved by:	
Prepared by:	
Reviewed by:	
Path Name:	
File Name:	Group11-SDD-CS2212B.doc
Document No:	

object
oriented pty. ltd.

Document Change Control

Version	Date	Authors	Summary of Changes
0.4	Mar 5	Ziyuan Li	Intro and References
0.5	Mar 9	Mingkai Yang, Sihui He, Yuhan Zhang	Activities Plan
0.6	Mar14	Yuhan Zhang, Ziyuan Li, Mingkai Yang	Design diagrams, Overview.
0.7	Mar 16	ALL	Final check and modify diagrams

Document Sign-Off

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

Contents

1 Introduction

1.1 Purpose

This document details the requirements of the system CoinMaster. CoinMaster is like coinbase but for broker managers and with human intervention in the process of selecting and trading. CoinMaster 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 decisionse 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

The SDD document contains the following information:

a. **Major Design Decisions** section contains significant design choices and the reasons supporting the decisions.
b. **Architecture** section contains the component diagram, which models the structure of the system as a collection of components.
c. **Activities Plan** section contains a list of product backlog items, which may later be considered for sprint backlog and the group meeting logs.
d. **Test Driven Development** contains a list of the test cases, which will be used to test against the system once coding is done.

1.3 Resources - References

https://www.coinbase.com/
https://www.coingecko.com/
https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency
https://www.uml-diagrams.org/component-diagrams.html
http://www.eclipse.org/downloads/index.php
https://www.section.io/engineering-education/how-to-create-a-user-login-page-using-java-gui/
https://medium.com/swlh/building-a-bitcoin-price-watcher-with-alerts-in-java-d52824e0631e

2 Major Design Decisions

Text describing significant design choices, and modularization criteria.

Write significant parts for designing the system.

We use the three-tier architecture; the entire system can be divided into three subsystems including “UI”, “CoinMaster Backend” and “Data Access” subsystems. The “UI” subsystem serves as the presentation tier which displays the information of user data and provides interfaces for users to interact with the systems. The “UI” subsystem contains three major interfaces: “Log-in service”, “Trading client actions management”, and “Trading action logs”. The “Log-in service” interface is used for users to submit their usernames and passwords; the “Trading client actions management" system displays the current status of trading brokers and allows users to update trading broker information, select strategy, and add the coin lists for each user. The "trading action log" interface displays the log of trading activity of each client is displayed in a histogram and a Table.

The "CoinMaster" subsystem serves as the application tier which fetches data from the "data access" subsystem and processes data according to the use cases. There are four components in the "CoinMaster" subsystem: "user authentication", "broker management", "perform trading", and "transaction log request handler". The "user authentication" component is used to verify the log-in information of users. The "broker management" component is used to update broker information by allowing users to add brokers, modify coin lists, select strategies, and remove brokers. The "perform trading" component is used to pass the appropriate prices to the right trading broker and perform the computation of the specific trading strategy associated with each broker. However, if the trading strategy is not applicable to apply, the component will send a failure message back to the main UI.

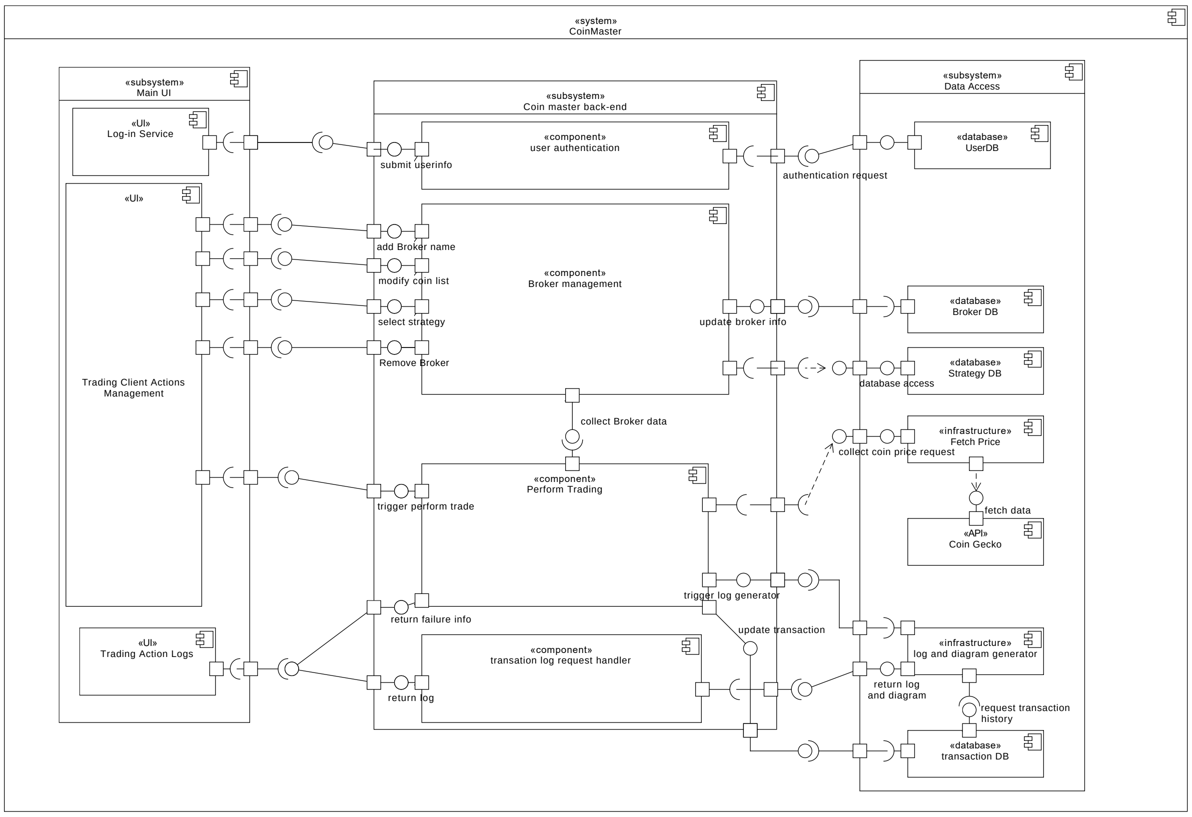
The "Data Access" subsystem serves as the data tier which Houses database servers where information is stored and retrieved. There are four databases in this subsystem: "user DB", "Broker DB", "Strategy DB", "Transaction DB"; meanwhile there are two infrastructure components: "Fetch Price" and "Log and Diagram Generator". The "Fetch Price" component fetches data from the API component "Coin Gecko".

3 Architecture

Provide the ***component diagram*** *of .ur system. If you need nested diagrams please use nesting levels. Include explanations on the functionality of each component. Provide the exposed interfaces of each component and list and briefly describe the functionality (one sentence) of the operations included in each such interface. Comment if you are using any specific architectural style or combinations of architectural styles.*

Once you provide your component diagram, you can use the following table structure to describe the operation for each exposed interface.

✓



1. Main UI: Main user interface for user to interact with the backend system
2. Log-in service: Serve as the verification portal to receive the input user ID and password and request the user authentication for verification.
3. Trading Client Actions Management: Act as the major user interface for verified users to modify the broker information and perform trades.
4. Trading Actions Logs: After the trading is performed, ask for the graphic result for the transaction records and failure information
5. CoinMaster back-end: Serve as the backend system to connect the user interface and the database.
6. Broker management: Provide the functionality-oriented method for user to update the broker information, and serve as the intermediate media to send user request to the database to access the broker information and predefined strategy.
7. Perform Trading: Serve as the intermedia processor to receive the trading perform command from the user and triggers the process for price access data and diagram generator. Also, it sends all relevant trading information to the transaction database.
8. Transaction log request handler: Request the diagram generator to return the trading records diagram and send the graphs to the main user interface.
9. User DB: The exogenous database storing the user credentials including username and password.
10. Broker DB: The exogenous database storing all information of brokers. Requests the back-end system to retrieve broker information.
11. Strategy DB: Store predefined strategies so that it can provide them when implementing trade performance.
12. Fetch price: fetch the price from coin Gecko and provide the price for trading perform system.

13. Log and diagram generator: Receive the request from trading perform system and generate trade table and histogram using transactions records received from the transaction DB.
14. Transaction DB: Receive the trading records from trading perform system and provide the transaction records to the generator

Subsystem Name	Component Name	Interface Name	Operation Signature	Description of the Operation
Main UI	Log-in Service	Verify status	+sendAuthenticationRequest (string userName, string password)	Send the authentication request to user authentication service in data access subsystem.
	Trading Action Logs	Display	+VisualizeResult()	Once the trading strategy has been invoked, display the histogram and table of trader actions.
CoinMaster backend	Broker Management	Add Broker Name	+AddBroker(string brokerName)	Add the broker with the given name to the broker database
			+displayStrategyList()	Display pre-defined strategies.
		Modify coin list	+AddInterestCoinListByBrokerName(Coin cp, string brokerName)	Set the broker's interested coin list and store it in the broker database

			+DeleteInterestCoinListByBrokerName(Coin cp, string brokerName)	delete the broker's interested coin list from the broker database
		Select strategy	+selecteStrategy(string strategyName)	Select the pre-defined strategy from database.
		Remove broker	+deleteBroker (string brokerName)	Delete the broker with the given name from the broker database
	Perform Trading	Trigger Perform trade	+triggerComputation()	Invoking the trading strategy associated with each client and perform the computation.
		Return failure info	+failureHandler()	Display an error message that the trading strategy can not be applied. Add a row on the table in the UI with the name of the broker, the name of the strategy, the indication "Fail", and the date of the failed transaction.
	User authentication	Submit user information	+validateUser (string userName,	The log-in system compares the input

			string password)	password with the password stored in the user database. If the input password doesn't match the one stored in the user database, return an error message.
			+verifyIfCoinExistsInTradingBroker(string strategyName, string BrokerNmae)	Verify if the cryptocoins used in the trading strategy are in the list of cryptocoins the trading broker declared interest in. If no, call failureHandler() method to handle the fail situation.
	Transaction log request handler	Return log	+transactionLog()	return the log of transaction history to trading actions logs UI component.
			+transactionTable()	Return the table of transaction history to trading actions logs UI component.
Data Access	Fetch Price	Fetch data	+fetchPrice(string[] coinList)	Fetch the price data from Gecko with the

				corresponding coins.
	Log and Diagram Generator	Return log and diagram	+logGenerator ()	Request the transcation history from trascation database, and return the log of transaction to transaction log request handler component to handle the log request.
			+tableGenerator	Request the transcation history from trascation database, and return the table of transaction history to transaction log request handler component to handle the log request.

4 Activities Plan

4.1 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.

Sprint goal: Complete and design SDD document, Initial coding the log-in System.

Backlog item	Estimate(hours)
SDD document and requirement analysis and initial design	6
Complete the Project backlog and Sprint backlog.	1.5
Revise UML Class Diagram	0.5

Architecture and component diagram design	10
Group meeting logs	0.5
Overview and intro	1
Initial coding design	3
Log-in system design and coding	10
Log-in system test	2
Code the middle tire	30
Test and debug	30

Sprint backlog

Tasks	Mar. 3	Mar. 5	Mar. 9	Mar. 10	Mar. 11	Mar. 12	Mar.13	Mar.14	Mar.15
SDD document and requirement analysis and initial design	2	4							
Complete the Project backlog and Sprint backlog.		0.5	1						
Revise UML Class Diagram		0.5							
Architecture and component diagram design			2	2	4		2		
Group meeting logs	0.5								
Overview and intro	0.5							0.5	

Initial coding design					3				
Log-in system design and coding						4	2	2	2
Log-in system test									2
Code the middle tire								2	2
Test and debug									

4.2 Group Meeting Logs

Present Group Members	**Meeting Date**	**Issues Discussed / Resolved**
All members	Mar. 5th (2 hours)	Assignment going through and tasks assigning
All members	Mar. 11th (6 hours)	Architecture and component diagram design, prepare the initial coding
All members	Mar. 14th (4 hours)	Work combining and revising
All members	Mar. 15th (4 hours)	Final checking and submitting

Sihui He: SH

Ziyuan Li: ZL

Yuhan Zhang: YZ

Mingkai Yang: MY

Backlog item	Estimate (hours)	Members
SDD document and requirement analysis and initial design	6	SH, ZL, YZ, MY

Complete the Project backlog and Sprint backlog.	1.5	YZ, MY
Revise UML Class Diagram	0.5	SH
Architecture and component diagram design	10	YZ, ZL, MY
Group meeting logs	0.5	ZL
Overview and intro	1	ZL
Initial coding design	3	YZ, MY
Log-in system design and coding	10	SH, YZ
Log-in system test	2	SH
Code the middle tire	30	SH, ZL, YZ, MY
Test and debug	30	SH, ZL, YZ, MY

5 Test Driven Development

Test ID	01
Category	Login Evaluation & Credential DB Connection
Requirements Coverage	UC1 – The user logs into the system
Initial Condition	The software has been initiated and the login interface has shown.
Procedure	1. The user enters a valid user name 2. The user enters the corresponding valid password 3. The user clicks *Submit!*
Expected Outcome	The software will connect the DB and successfully verify the credentials. The user can successfully log in and the main UI will show.
Notes	All upper-case characters will automatically be converted to lower-case characters when tested against the record in the DB.

Test ID	02
Category	Adding a Trading Broker
Requirements Coverage	UC2. Adding and Removing a Trading Broker
Initial Condition	The user has successfully logged in and the main UI is presented. The default row has been filled.
Procedure	1. The user clicks *Add Row*

	2. A second fillable row shows up. 3. The user enters a broker name that has not been previously entered. 3. The user enters a list of valid cryptocurrency abbreviations separated by commas. 4. The user selects a strategy from the drop-down list of strategies.
Expected Outcome	All verifications are successful. A new row with corresponding information shows up in the main UI.
Notes	Broker name is verified against DB and cryptocurrency’s names is verified against the list fetched from online.

Test ID	03
Category	Performing Trade
Requirements Coverage	UC3. Performing Trading
Initial Condition	The user has logged in. The main UI is in display. There is a least one broker entry with a not none strategy been selected.
Procedure	1. Select the broker. 2. Select the strategy in the drop-down list. 3. Click *Perform Trade*
Expected Outcome	The program can correctly evaluate the conditions in the selected strategy by fetching real-time prices. If the conditions were not met, the program will return a message indicating failed transaction. If the conditions were met, the program will return a message indicating the transaction has been successful.
Notes	If the program cannot successfully fetch the price from the internet (internet connection issue or the cryptocurrency website is offline), the program will return a connection error message.

Test ID	04
Category	Displaying the Trading Action for All Trading Clients
Requirements Coverage	UC3. Performing Trading UC4. Displaying the Trading Action for All Trading Clients
Initial Condition	The user has logged in. The main UI is in display. There is a least one transaction been successfully processed.
Procedure	1. Select the broker. 2. Select the strategy in the drop-down list. 3. Click *Perform Trade* 4. Observe how *Trade Actions* and *Actions Performed by Traders So Far* have changed.

Expected Outcome	The new transaction has been correctly shown in the *Trade Actions*, and the *Actions Performed by Traders So Far has changed accordingly*.
Notes	N/A