Pooled Investment Portfolio Management Simulation
Project Specification

Project Repository

<https://github.com/CSC207-UofT/course-project-codemonkeys>

CRC Representation

CRC Cards.pptx

Contributors

Langson Zhang <https://github.com/langsonzhang>
Tammy Yujin Liu <https://github.com/tammyliuu>
Jiamu Sun <https://github.com/JackSunjm>
Andrew Hanzhuo Zhang <https://github.com/a663E-36z1120>
Peng Du <https://github.com/Vim0315>
Raymond Zhang <https://github.com/RaymondZhang24>
Edward Li <https://github.com/Edward11235>
Zixin (Charlie) Guo <https://github.com/charlieguo2021>

1. Domain

This project is a management tool for a simulated pooled investment portfolio, where multiple parties pool their funds together to form a single investment portfolio in the financial market. The project enables all invested parties to democratically make investment decisions for the portfolio through calling and casting votes. The voting power of each party varies according to the profitability of their past votes.

Different users interact with the system with a command prompt.

2. Entity Specification

2.1. Assets and Portfolio

Assets are either liquid or non-liquid. The only liquid asset in this simulation is USD, while non liquid assets include stocks, cryptocurrencies, precious metals, etc. All liquid

assets form the liquidity pool and non-liquid assets the investment pool. Non-liquid assets will record when it was bought and at what price it was bought.

The portfolio is a collection of assets. There is a global portfolio representing the pooled investments. Every user also has a shadow portfolio to keep track of their past investment performances.

2.2. Transaction

Each vote consists of 4 main pieces of information: what asset to buy/sell, volume to be bought/sold, when it was casted, and for or against.

2.3. Users

There are 2 classes of users: User and Administrators.

Users are the most rudimentary class of users. The voting power of a user is measured in USD, indicating how much liquidity/asset they have the power to make investment decisions for.

If a user wishes to make a transaction with a volume smaller than or equal to their voting power, they may do so without calling a vote. After successful registration, the regular investor brings in a variable amount of liquidity (USD) to the liquidity pool. The amount of liquidity brought in will be their initial voting power. The regular investor's voting power will be adjusted based on the profitability of their 10 most recent votes/transactions.

Administrators are regular investors with additional privileges. Administrators can directly veto or pass a vote, or ban other non-administrator users from the simulation.

Bots are investors that do not have voting power. A bot can also be owned by a regular investor, which means that the user will automatically make every investment decision made by the bot.

3. Use Case Specification

3.1. dataAccessInterface

Fetches and updates the price of an asset. All asset prices are obtained in real-time from Yahoo finance Java API, from which the profitability of the investment portfolio is calculated.

3.2. transactionManager

Delete, add, get, set, and store historical and pending transactions.

3.3. portfolioManager

Updates a portfolio if a vote is successful by carrying out the transaction in simulation.

Determines the profitability of a portfolio based on real-time prices.

Provide a summary of the status of the pooled portfolio including current levels of profit and growth and how much of what assets are held each in the liquidity and investment pool.

3.4. userManager

Calculates and updates the voting power of a user. Voting power is calculated based on the profitability of their 10 most recent votes/transactions. For each vote/transaction that is profitable, the user's voting power increases by 10%, and for each vote/transaction that is not, the user's voting power decreases by 10%.

Ban, promote, or accept an existing non-administrator user or users.

3.5. voteManager

Decides the result of votes. A successful vote will be turned into a transaction, which the portfolio will be updated accordingly. A vote is considered successful if either of the following requirements are met:

1. The total weighted voting power for a transaction subtracting the total weighted voting power against a transaction exceeds the volume of the transaction.

2. 24 hours after the vote is cast, the total weighted votes for a transaction subtracting is greater than the total weighted votes against a transaction.

Otherwise the vote is considered a failure and the transaction will not proceed.

3.5. commandInterface and command objects

command objects are objects implementing the commandInterface each representing an input command that the user can input from the CLI. commandInterface enforces that each of the methods must have a execute() method to execute the specified command functions. For the specific commands, see section 5.1.

4. Controller Specifications

4.1. commandParser and commandExecuter

commandParser parses the input command from the CLI and returns a command object to the commandExecuter, which executes the command.

4.2. graphicsPresenter

Generates graphs representing the current status of the portfolio.

5. Interface Specifications

5.1 CLI

The following commands each represents a command use case object that implements the commandInterface:

`Help`
Get information on all available commands.

`buy` <asset symbol> <volume in USD>

Initiate a buy transaction.

`sell` <asset symbol> <volume in USD>

Initiate a sell transaction.

`upvote` <vote ID> <optional admin flag '-admin'>

Vote in favour of a transaction.

`downvote` <vote ID> <optional admin flag '-admin'>

Vote against a transaction.

`checkPrice` <asset symbol>

Get the current price of an asset.

`createUser` <user ID>

Register a new user into the system.

`kick` <user ID>

Kick a user out of the system.

`leave` <user ID>

Let the user leave the system.

5.2 GUI

GUI includes a suite of graphs and charts to indicate how the pooled portfolio is performing.

5.3 YahooFinance

Third-party library that fetches real-time financial data from Yahoo Finance.