Project 2

Report 3

December 7

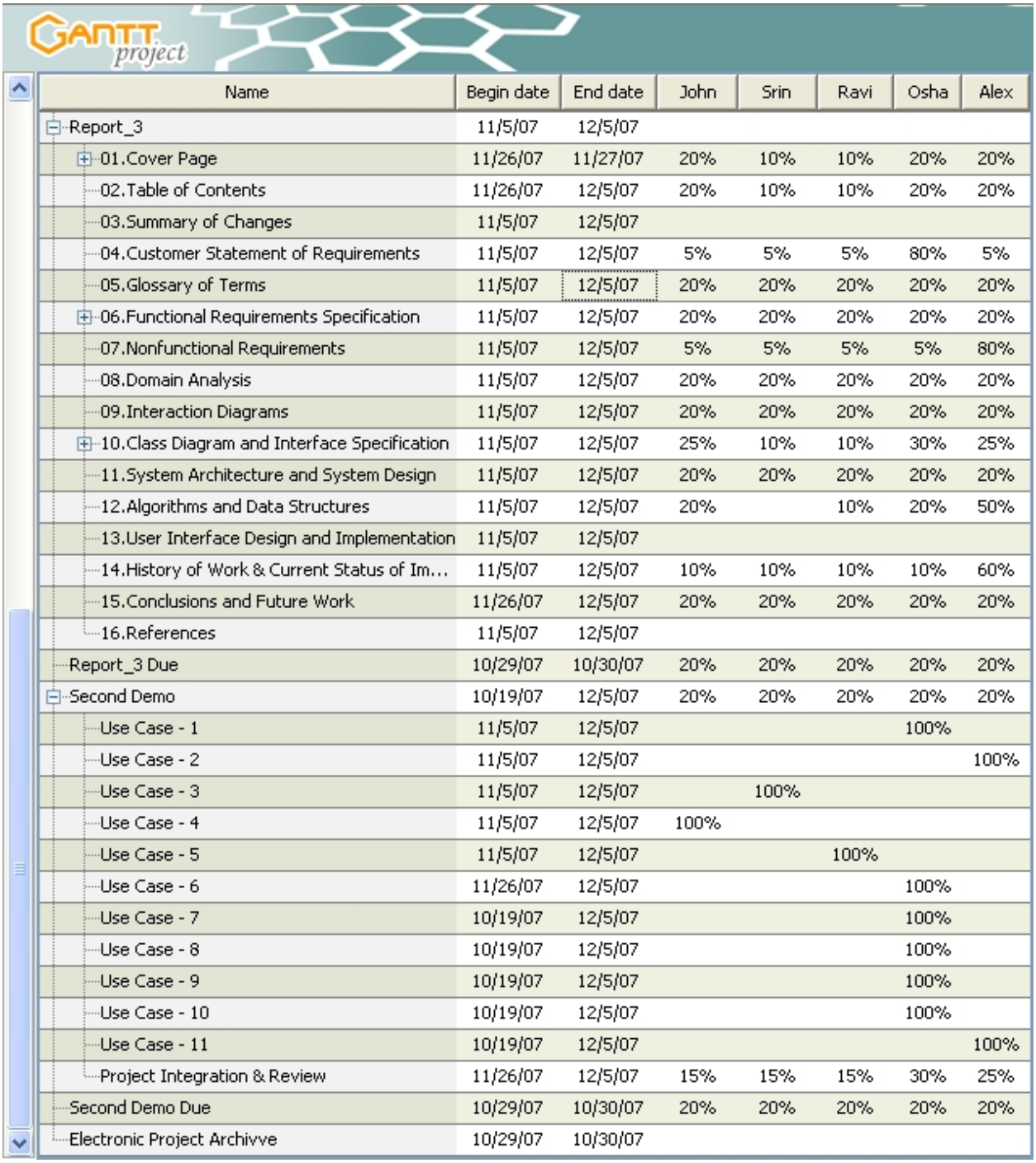
2007

This document summarizes the use cases, system sequence diagrams, domain models, class diagram, interaction diagram, hardware requirements, algorithms, historical work, and other implementation details of Project 2 for the Fall 2007 ECE567 - Stock Web Application back end.

By Osha Fuangkasae
Zong-Zhi 'Alex' Lin
John Paul Varkey
Srinivas Mudireddy
Ravi Gudur

Breakdown of responsibilities for Report 3

Figure 1: Provides the breakdown of responsibilities for Report 3



Name	Begin date	End date	John	Srin	Ravi	Osha	Alex
Report_3	11/5/07	12/5/07					
01.Cover Page	11/26/07	11/27/07	20%	10%	10%	20%	20%
02.Table of Contents	11/26/07	12/5/07	20%	10%	10%	20%	20%
03.Summary of Changes	11/5/07	12/5/07					
04.Customer Statement of Requirements	11/5/07	12/5/07	5%	5%	5%	80%	5%
05.Glossary of Terms	11/5/07	12/5/07	20%	20%	20%	20%	20%
06.Functional Requirements Specification	11/5/07	12/5/07	20%	20%	20%	20%	20%
07.Nonfunctional Requirements	11/5/07	12/5/07	5%	5%	5%	5%	80%
08.Domain Analysis	11/5/07	12/5/07	20%	20%	20%	20%	20%
09.Interaction Diagrams	11/5/07	12/5/07	20%	20%	20%	20%	20%
10.Class Diagram and Interface Specification	11/5/07	12/5/07	25%	10%	10%	30%	25%
11.System Architecture and System Design	11/5/07	12/5/07	20%	20%	20%	20%	20%
12.Algorithms and Data Structures	11/5/07	12/5/07	20%		10%	20%	50%
13.User Interface Design and Implementation	11/5/07	12/5/07					
14.History of Work & Current Status of Im...	11/5/07	12/5/07	10%	10%	10%	10%	60%
15.Conclusions and Future Work	11/26/07	12/5/07	20%	20%	20%	20%	20%
16.References	11/5/07	12/5/07					
Report_3 Due	10/29/07	10/30/07	20%	20%	20%	20%	20%
Second Demo	10/19/07	12/5/07	20%	20%	20%	20%	20%
Use Case - 1	11/5/07	12/5/07				100%	
Use Case - 2	11/5/07	12/5/07					100%
Use Case - 3	11/5/07	12/5/07		100%			
Use Case - 4	11/5/07	12/5/07	100%				
Use Case - 5	11/5/07	12/5/07			100%		
Use Case - 6	11/26/07	12/5/07				100%	
Use Case - 7	10/19/07	12/5/07				100%	
Use Case - 8	10/19/07	12/5/07				100%	
Use Case - 9	10/19/07	12/5/07				100%	
Use Case - 10	10/19/07	12/5/07				100%	
Use Case - 11	10/19/07	12/5/07					100%
Project Integration & Review	11/26/07	12/5/07	15%	15%	15%	30%	25%
Second Demo Due	10/29/07	10/30/07	20%	20%	20%	20%	20%
Electronic Project Archivve	10/29/07	10/30/07					

Notes: We placed this Figure in the beginning of the Report because the Report 3 guidelines suggested that it should be presented *before* the Table of Contents.

Responsibilities from Perspective of Developing and Implementing Classes

List the names of modules and classes that each team member is responsible for developing, coding, and testing:

- Osha Fuangkasae : UC1Controller, UC6Controller, DatabaseCommunicator, PerlCommunicator, XMLParser, SUDTimer, ECE567, SystemMenu, Observer, Subject
- Zong-Zhi 'Alex' Lin: UC2Controller, MarketEvalulator, InvestorController, PortfolioEvaluator, StockEvaluator, SingleIndexModel, EvalutatedStockSet, BaseStats, ValueAtRisk
- Srinivas Mudireddy: UC3Controller
- John Paul Varkey: UC4Controller, RankEstimator, PlayerInfo
- Ravi Gudur: UC5Controller

John Paul Varkey will coordinate the integration.

Osha Fuangkasae will perform the testing of the integrated system.

Zong-Zhi Lin and Osha Fuangkasae are responsible for most of the documentation compilation and formatting.

Breakdowns of responsibilities for Report 1 and 2 are included in History of Work section.

Table of Contents

Table of Figures

Summary of Changes

The key revision since the previous two reports are the additions of new use cases

1. **Use Case 7 – Test Portfolio Limits:** The SUD performs checks on current stock values to detect whether or not they cross Investor-defined Portfolio Limits. The SUD e-mails each Investor whose Portfolio Limit has been passed. (By Osha Fuangkasae)

2. **Use Case 8 (UC8) – Match Pending Transactions:** The SUD performs matches on current stock values to detect whether or not they meet the ask price or bid price of pending transactions. The SUD completes and e-mails each Investor whose pending transactions have been completed. (By Osha Fuangkasae)

3. **Use Case 9 (UC9) – Reset Timer:** The SUD performs to set the timer to a different interval and starts its period. (By Osha Fuangkasae)

4. **Use Case 10 (UC10) – Perform Market Orders:** The SUD executes the valid Market Orders, and updates the Investor Database to reflect any necessary changes. Note that the buy and sell cases are so similar that they are both placed in this Use Case. (By Osha Fuangkasae)

5. **Use Case 11 (UC11) - Portfolio risk analysis – Value at Risk:** The system offers portfolio risk analysis to generate relevant measures to allow players to measure their current risk position in their portfolio via Value at Risk metric. (By Zong-Zhi 'Alex' Lin)

Customer Statement of Requirements (narrative):

John Connor
Shaba-Doo Inc.

Dear Project 2 Team:

Congratulations on being chosen for a rewarding and exciting new work opportunity! We are a startup company named Shaba-Doo, and we'd like to make a global footprint for ourselves by producing a web application that allows people to test their stock investing skills. Our idea is for prospective “Investors” to register for our web application, and enter them into a risk free investing tournament. Each Investor to start is given the same amount of money to use to buy stocks. They are then let loose into the real life stock market to buy, sell, and modify their stock portfolios as they see fit. Our web application keeps track of each Investor's portfolio, and uses actual data from Yahoo! Finance to regularly determine the worth of a particular Investor. Through profits gained from advertisements, we intend to award the best Investors on a regular basis, in order to provide an additional incentive to play the game and perform well. Furthermore, we would also like to trace a successful Investor's history, to determine what decisions put her ahead of everyone else. As Shaba-Do, our company goals are to educate ourselves and our Investors on stocks and investing, make some money from advertisements, and set a standard of quality for our future projects by providing a robust, reliable system for our Investors.

We have fortunately divided this large task into three different modules. We currently have two teams working on the Investor GUI (HTML and PHP) and the Administrator interface, but we need a project group to gather the stock data and perform calculations on it so that our users can

achieve what we have outlined above. In other words, we need your team to work on the back end of this project. As stated, we need you to think of a way to access and glean information from Yahoo! Finance. We currently do not know how much processing power our servers are capable of, so we would like to be able to dynamically set the time intervals between database updates. We need to somehow store past success information, in order to provide the success strategies that successful Investors have employed in the past. It might make sense to put this Stock History into a location other than the Stock Database, since the database schema has yet to be created by our user interface team. Moreover, the back end needs to communicate seamlessly with this Stock Database so that the GUI team can display updated data. We only expect to offer around hundred stocks to choose from, but we would like to make modifications to the stock name list on an intermittent basis.

As another challenge, we need your team to perform current status evaluation on our Investors' portfolios on a regular basis, and e-mail Investors with that information. This is to keep them informed of how they are currently performing, and remind them to make changes to their portfolio, if necessary. At the same time, we would like you to compare the Investor's portfolios relative to the whole groups', so that a Website Manager can at any time see the current "winner". Your team must decide on the format of this e-mail, since Shaba-Doo does not know anything about investing and cannot provide any useful suggestions regarding such things. We also need you to research what mathematical models can indicate the risk level of an investment portfolio and summarize this, perhaps in a number that beginning Investors can quickly understand.

Finally, we would like your team to access a particular successful Investor's history to determine what buys and sells have awarded him the most value. Our intent is for Investors to be able to identify practices of the current tournament winner to attempt to emulate her styles.

Please keep us informed of your progress. Let us know if you have any questions, and again, congratulations on accepting such a wonderful opportunity!

Sincerely,

John Connor
Shaba-Doo Inc.

(PS - I've included a list of requirements as a checklist for your team.)

Written by Osha Fuangkasae

Requirements

1. The system should track the stock movements of the stocks that are on “watch-lists,” i.e., they may not be owned currently by players but are considered for purchasing when certain conditions are met. Stock prices can be retrieved from http://finance.google.com/finance, http://finance.yahoo.com/, http://www.freerealtime.com/, etc. The frequency of tracking is limited by the system processing power.
2. The system should periodically perform evaluation of player portfolios. The summary report is generated and emailed to the player. At this time the system should also perform player ranking and notify the system administrator, who may consider awarding the best players.
3. The system should analyze player performance over time and try to identify the “best practices,” or trading strategies pursued by the best-performing players. The analysis could include the parameters such as which stocks they hold (by industry type, etc.), how frequently they trade, and correlate their trading behavior with market indices, such as Dow Jones Industrial Average (http://www.djindexes.com/), Standard & Poor’s 500 Index (http://www2.standardandpoors.com/portal/site/sp/en/us/page.topic/indices_500/2,3,2,2,0,0,0,0,0,0,0,0,0,0,0,0.html), etc.
4. The system should offer fundamental and technical analysis of selected stocks. http://en.wikipedia.org/wiki/Fundamental_analysis, http://en.wikipedia.org/wiki/Technical_analysis.
5. The system should offer portfolio risk analysis to generate Sharpe Ratio, and other relevant measures if time permits to allow players to measure their current or expected position risk in their trading, portfolio or stocks that are on "watch-lists". (Markowitz Portfolio Analysis for the Individual Investor - http://faculty.washington.edu/rons/articles/mark.ps, Individual Investor Risk Aversion and Investment Portfolio Composition - http://www.jstor.org/view/00221082/di991947/99p07387/0).

Glossary of Terms

Controller – This is the entity that manages communication between the other entities.

Investor Database – This is the database that contains the information about all of the registered Investors.

Portfolio Limit – These are monetary thresholds imposed by the Investors on their own Stock Portfolios. It is expected that if these thresholds are crossed, the corresponding Investor will be notified.

Server Administrator – This is the server side user who starts the server and is in charge of changing update intervals, stock watch lists, database information, and any other administrative issues.

Stock Attribute File – This is the list of stock attributes that the SUD is expected to track.

Stock History File - This log contains stock information for a specified time window that is used to calculate performance and graphs. It also contains the names of the stocks to keep track of.

Stock Database – This is the database that contains the current information about stocks being tracked by the web application.

Stock List – This is the list of stocks that the SUD is expected to track.

Stock Portfolio – This is the current set of stocks that an Investor owns.

System Menu – This is the text based menu that the Server Administrator is shown once the server is running.

Timer – This is an abstract concept referring to an entity that notifies listeners on timeout.

Timer Interval – This is the amount of time that occurs between recurrent use cases.

Yahoo! Finance Library – This is a library whose capabilities include parsing Yahoo! Finance.

Functional Requirements Specification

Stakeholders

We only considered the stakeholders for our (Project 2's) system. In other words, because we are only developing the back end of an investing game web application, we decided to keep the stakeholder scope limited to those interested in such a back end project.

Our possible stakeholders include Investors interested in learning and experimenting with investing. They are open to the idea of an online game intended to help them do so at a merely virtual risk. The Server Administrator of this web application is a stakeholder since she needs the back end functionality of our server in order for the entire application to function properly. Finally, Future webmasters interested in implementing a similar system could use and build on modules from ours, so they can be considered stakeholders.

Actors and Goals

Format - [Actor : Goal]

- *Investor* : Risk-free experimentation in the world of stock investing
- *Investor Database* : Provide relevant information about investor and relevant information about his or her portfolio
- *Server Administrator* : Keep the Stock Database updated when necessary
- *Website Manager* : Use data and reports created by the SUD to approve and award the superior investors to increase the usage of the SUD.
- *Stock Database* : Provide information about current stock values
- *Timer* : Notify listeners when time interval runs out
- *Yahoo! Finance* : Provide current (or 10 minutes delayed) stock prices updates

Use Cases

Casual Descriptions

Use Case 1 (UC1) - Get online stock quotes: UC1 is used to keep the Stock Database updated. This needed for Project 1 to display the data in their GUI. It is also necessary to perform the calculations done in the following other use cases.

Use Case 2 (UC2) - Portfolio risk analysis – Sharpe Ratio: The system offers portfolio risk analysis to generate relevant measures to allow players to measure their current risk position in their portfolio via Sharpe Ratio metric.

Use Case 3 (UC3) - E-mail current status of Investor portfolios: The system periodically performs evaluations of all Investors' portfolios. The summary report is generated and emailed to the player.

Use Case 4 (UC4) – Ranking the players: The system generates Investor rankings and notifies the Server Administrator, who may consider awarding the best player.

Use Case 5 (UC5) - Determine the best practices: The system analyzes a particular Investor's performance over time to try to identify her "best practices" or trading strategies. The analysis could include the parameters such as which stocks they hold (by industry type, etc.), how frequently they trade, and correlate this trading behavior with market indices, such as Dow Jones Industrial Average (http://www.djindexes.com/), or Standard & Poor's 500 Index.

Use Case 6 (UC6) - Set Timer Interval: The Server Administrator sets the time frame between updates to the Stock Database.

Use Case 7 (UC7) – Test Portfolio Limits: The SUD performs checks on current stock values to detect whether or not they cross Investor-defined Portfolio Limits. The SUD e-mails each Investor whose Portfolio Limit has been passed.

Use Case 8 (UC8) – Match Pending Transactions: The SUD performs matches on current stock values to detect whether or not they meet the ask price or bid price of pending transactions. The SUD completes and e-mails each Investor whose pending transactions have been completed.

Use Case 9 (UC9) – Reset Timer: The SUD performs to set the timer to a different interval and starts its period.

Use Case 10 (UC10) – Perform Market Orders: The SUD executes the valid Market Orders, and updates the Investor Database to reflect any necessary changes. Note that the buy and sell cases are so similar that they are both placed in this Use Case.

Use Case 11 (UC11) - Portfolio risk analysis – Value at Risk: The system offers portfolio risk analysis to generate relevant measures to allow players to measure their current risk position in their portfolio via Value at Risk metric.

Fully-Dressed Use Case Descriptions

Use Case 1 (UC1): Get online stock quotes

- Initiating Actor: The initiating actor is the Timer, which is set in UC6 to initiate UC1 on a regular basis.
- Actor's Goal: To gather stock quote information from online and store the results in both a Stock History File and Stock Database on the local machine.
- Participating actors: Stock List, Stock Attribute File, Stock Database, Stock History File, Yahoo! Finance Library
- Preconditions: The SUD is currently within the System Menu (not currently running any other use cases), the Timer Interval has been set, and the Stock List and the Stock Attribute File have been defined. The Stock Database exists and implements the expected schema.
- Trigger: The Timer times out.
- Post conditions: The Stock Database is updated, as is the Stock History File. The Timer is reset. The System Menu is shown.
- Main Success Scenario:

  → 1. Timer initiates Stock Database update.

  ← 2. System reads Stock List.

  ← 3. System reads Stock Attribute File.

  ← 4. System calls Yahoo! Finance Library.

  ← 5. System copies the information returned by the Yahoo! Finance Library into the Stock Database.

  ← 6. System gathers stock quotes and stores the information into the Stock History File.

  ← 7. System resets Timer.

  ← 6. System returns to System Menu.

- Author and Date: Osha Fuangkasae on September 26th, 2007, update on October 9th, 2007

Use Case 2 (UC2): Portfolio risk analysis – Sharpe Ratio

- Initiating Actor: Timer.
- Actor's Goal: To understand the expected risk position in his/her portfolio or "watch-list" stocks via the magnitude of Value at Risk metric.
- Participating actors: Stock Database, Mathematical Model
- Preconditions: The User is properly authenticated, the list of stocks to risk-analysis are known, and a Stock History File update has occurred.
- Trigger: The User asks for an updated risk analysis.
- Post conditions: The Portfolio Risk Analysis subsystem generates a risk & reward positions of various combinations of stocks from User's portfolio.
- Main Success Scenario:

  → 1. Server Admin requests the system to perform portfolio risk analysis.

  ← 2. System (a) retrieves the stocks in Investor's portfolio, (b) retrieves stock's risk and reward information from Stock Database, (c) apply mathematical model to derive the risk/reward evaluation of Investor's portfolio, and (d) updates Risk Information in Investor Portfolio.

  ← 3. System (a) notifies Server Admin that the analysis process is complete and (b) returns to System Menu.

- Flow of Events for Failure Scenario:

  System fails to retrieve investor and/or his/her portfolio information. A potential example is that database system is down.

  → 1. Server Admin triggers the system to obtain all the required information for performing portfolio risk analysis.

  ← 2.System (a) detects error(s) in retrieving the information (b) marks a failed attempt, and (c) notifies the system to retry.

  ← 3.After a fixed number of failed attempts, the system notifies the Server Admin.

- Author and Date: Zong-Zhi Lin, September 26th, 2007

Use Case (UC3): E-mail current status of investor portfolios

(Sri) Use Case UC-3: Evaluate each portfolio and email it to the corresponding investor

1. **Initiating Actor:** Timer
2. **Actor's Goal:** To perform the evaluation of investor portfolios, summarize the information and email each investor and also notify the system administrator.
3. **Participating Actors:** Timer, System Administrator and Investors
4. **Preconditions:** The system contains the information of all the Investors.
5. **Post conditions:** Each player will be emailed a summary of portfolios.
6. **Flow of Events for Main Success Scenario:**

   → 1.**Timer** triggers the system to obtain the information of all the investors, summarize them. Investor information is obtained from the "Investor" table and with the help of investor_id, the stock symbol, number of shares and the buy_price are obtained from "Portfolio" table and the current price of each stock is obtained from the "Stock" table. Then the total gain for each investor is calculated. The portfolio includes the total gain and current available funds (obtained from "Investor" table) for each investor.
   ← 2. **System** then emails the investors, the summary of their portfolios.
   ← 3. **System** signals the administrator that the **Investors** have been provided with the summaries of individual portfolios

7. **Flow of Events for Failure Scenario:**

   Internet fails to email the investors

   → 1.**Timer** triggers the system to obtain the information of all the Investors, summarise them.
   ← 2.**System** (a) detects error and (b) marks a failed attempt.

8. **Author and Date:** Sri, September 26th, 2007

❖ Author and Date: Sri, September 26th, 2007

Use Case 4 (UC4): Ranking the Players

- Initiating Actor: Timer
- Actor's Goal: To periodically rank the players
- Participating Actors: Investor Database
- Preconditions: The system contains the information of all the players
- Trigger: The Timer runs out chooses the corresponding menu item to rank the players
- Post conditions: The players will be ranked based on their performance
- Main Success Scenario:

  → 1. Timer request the system to rank the players

  ← 2. System (a) retrieves the portfolios from the investor data base (b find out the rank of the players

  ← 3. System updates the rank of the players in Investor Database
- Author and Date: John Paul, September 26th, 2007

Use Case 5 (UC5): Identify the best practices

- Initiating Actor: Timer
- Actor's Goal: Tracking the trading values of the customers, compare them to grade them along with their strategies.
- Participating actors: Timer, Investor Table, Stock History File and Investor
- Preconditions: The Updated ranking is stored in a table named Investor.
- Post conditions: A table "Best Strategy is updated with the strategy of the new Rank 1 player, with his portfolio items such as Buy Price, Symbol, Number of shares.
- Main success scenario:

  → 1.Stock vales of the each trader is identified and their values (along with the company name, stock value and the company recent trading history has been copied.
- Author and Date: Ravi, September 26th, 2007

Use Case 6 (UC6): Set Timer Interval

- Initiating Actor: Server Administrator
- Actor's Goal: Get the SUD running to regularly update the Stock Database
- Participating Actors: Timer
- Preconditions: None.
- Trigger: The Server Administrator starts the SUD.
- Post conditions: The Timer Interval has been set and the Timer has started counting down to the next Database update. The System Menu is shown.
- Main Success Scenario:

  → 1. Server Administrator starts SUD.

  ← 2. SUD prompts Server Administrator for the Timer update interval.

  → 3. Server Administrator inputs setting.

  ← 4. SUD sets Timer.

  ← 5. SUD starts Timer.

  ← 6. SUD displays System Menu.
- Author and Date: Osha Fuangkasae on September 26th, 2007

Use Case 7 (UC7): Test Portfolio Limits

- Initiating Actor: Timer
- Actor's Goal: Alert an Investor if her Portfolio Limit has been broken.
- Participating Actors: Internet, Investor Database, Stock Database
- Preconditions: The Stock and Investor Databases exist and are populated, and are connected to the SUD. The SUD can communicate with Investors through the Internet.
- Trigger: The Timer times out.
- Post conditions: All Investors whose Portfolio Thresholds have been surpassed have been notified.
- Main Success Scenario:

  → 1. Timer notifies SUD of timeout.

  ← 2. System performs Portfolio Threshold calculations over all Stock Portfolios using the following process:

    a. System requests Portfolio Threshold from Investor Database.
    b. System requests current stock information from Stock Database.
    c. System calculates total Stock Portfolio value using the current stock prices from the Stock Database.
    d. System compares the two data and finds that the Portfolio Threshold has been surpassed.
    e. System sends a notification using the Internet to the Investor.
    f. System moves on to the next Investor's Portfolio Threshold.
- Alternate Scenarios:

  ← 1. System performs Portfolio Threshold calculations over all Stock Portfolios, which are outlined below:

    a. System compares the two data and finds that Portfolio Threshold still holds.
    b. GOTO Main Success Scenario 2 f
- Author and Date: Osha Fuangkasae on October 9, 2007

Use Case 8 (UC8): Match Pending Transactions

- Initiating Actor: Timer
- Participating Actors: Investor Database, Stock Database
- Preconditions: The Stock and Investor Databases are properly initialized, populated, and connected to the SUD.
- Trigger: The Timer runs out.
- Post conditions: The valid Stop Orders are transmuted into Market Orders. The outdated Stop Orders are considered Failed Orders.
- Main Success Scenario:
  1. → Timer times out.
  2. ← System performs Stop Order calculations, iterating over all Stop Orders owned by each Investor as outlined below:
     a. System requests Investor Stop Orders from Investor Database.
     b. System requests Stop Order from Investor Database.
     c. System checks duration of Stop Order and finds that it is still valid
     d. System requests current stock information from Stock Database.
     e. System compares the two values and finds that the Stop Order price has been reached.
     f. System changes Stop Order to Market Order.
     g. System moves on to next Stop Order. If the Investor has no more Stop Orders, System moves on to next Investor.
- Alternate Scenarios:
  2. (1) ← System performs Stop Order calculations, iterating over all Stop Orders owned by each Investor as outlined below:
     a. System checks duration of Stop Order and finds that it is invalid.
     b. System changes Stop Order to Failed Order
  2. (2) ← System performs Stop Order calculations, iterating over all Stop Orders owned by each Investor as outlined below:
     a. System compares the two values and finds that the Stop Order price has not been reached.
     b. GOTO Main Success Scenario 2 f
- Author and Date: Osha Fuangkasae on October 9, 2007

Use Case 9 (UC9): Reset Timer

- Initiating Actor: Server Administrator
- Participating Actors: Timer
- Preconditions: The Timer has been previously set.
- Trigger: The Server Administrator sends the request to reset the Timer
- Post conditions: The Timer is set to a different interval and starts its period.
- Main Success Scenario:
  1. → Server Administrator chooses to reset the Timer.
  2. ← System performs the following actions:
     a. System stops current Timer.
     b. Include::UC6
- Author and Date: Osha Fuangkasae

Use Case 10 (UC10): Perform Market Orders

- Preliminary Notes: The buy and sell cases are so similar that they are both placed in this Use Case. Their differences are presented using the notation '/'.
- Initiating Actor: Timer
- Participating Actors: Stock Database, Investor Database
- Preconditions: Stock and Investor Database have been properly initialized and populated.
- Trigger: Timer times out.
- Post conditions: The valid Market Orders have been executed, and the Investor Database has been updated to reflect any necessary changes.
- Main Success Scenario:
  1. → Timer runs out.
  2. ← System performs Market Order calculations, iterating over all Market Orders owned by each Investor as outlined below:
     a. System gets Investor's Current Funds and Stock Portfolio from Investor Database.
     b. System gets current Market Order from Investor Database.
     c. System gets current stock information from Stock Database.
     d. System checks Market Order to detect that the order is a buy/sell.
     e. System compares Stock Limit value to current price and finds that the Stock Limit has not been surpassed.
     f. System finds that Investor has enough Current Funds/Current Shares to purchase/sell the number of shares requested.
     g. System inserts/deletes the stock shares in the Investor's Stock Portfolio.
     h. System changes Market Order into Successful Order
     i. System decrements/increments Current Funds from Investor.
     j. If another Investor Market Order exists, GOTO Main Success Scenario 2 b. Otherwise go to 2 a for the next Investor.

- Alternate Scenarios:
  2. (1) ← System performs Market Order calculations, described below:
     a. System compares Stock Limit value to current stock value and finds that the Stock Limit has been reached.
     b. System changes Market Order to Failed Order.
     c. System moves on to next Market Order.
  2. (2) ← System performs Market Order calculations, described below:
     a. System finds that Investor does not have enough Current Funds/Shares to purchase/sell the number of shares requested.
     b. GOTO Alternate Scenario 2 (1) f
- Author and Date: Osha Fuangkasae

Use Case 11 (UC11): Portfolio risk analysis – Value at Risk

- Initiating Actor: Timer.
- Actor's Goal: To understand the expected risk position in his/her portfolio via the magnitude of Value at Risk metric.
- Participating actors: Investor Database, Mathematical Model
- Preconditions: The User is properly authenticated, the list of stocks to risk-analysis are known, and a Stock History File update has occurred, i.e., Use Case 2 needs to be executed before this use case.
- Trigger: The User asks for an updated portfolio risk analysis.
- Post conditions: The Portfolio Risk Analysis subsystem generates Value at Risk positions of portfolios.
- Main Success Scenario:

  → 1. Server Admin requests the system to perform portfolio risk analysis – Value at Risk.

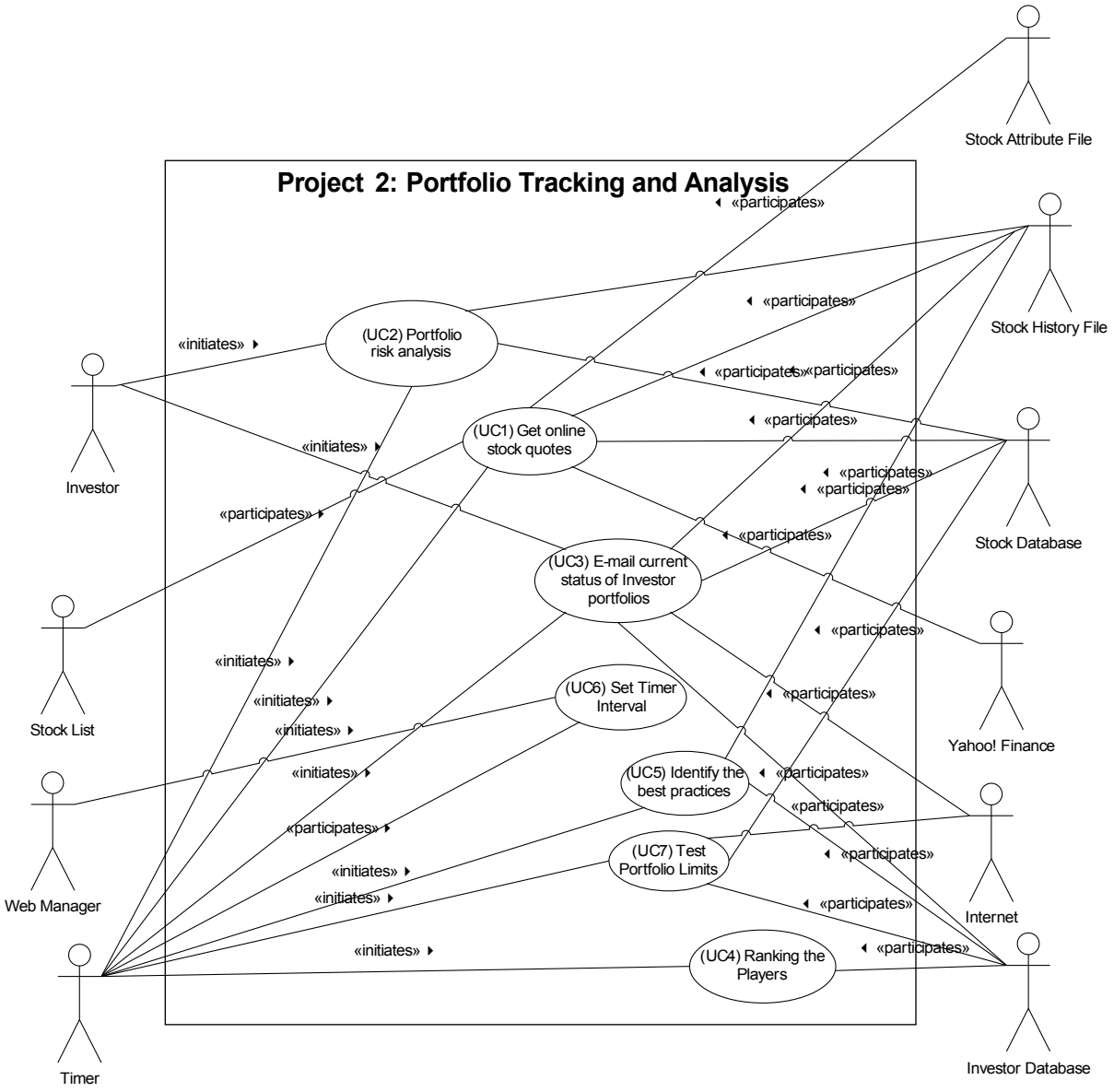
  ← 2. System (a) retrieves investors and his or her portfolio's risk and reward information from Investor Database, (b) apply mathematical model to derive the Value at Risk evaluation of Investor's portfolio, and (c) updates Value at Risk Information in Investor Portfolio.

  ← 3. System (a) notifies Server Admin that the analysis process is complete and (b) returns to System Menu.

- Author and Date: Zong-Zhi 'Alex' Lin, December 6, 2007

Use Case Diagram

Figure 2: Use Case Diagram



System Sequence Diagrams

Figure 3: UC1 System Sequence Diagram

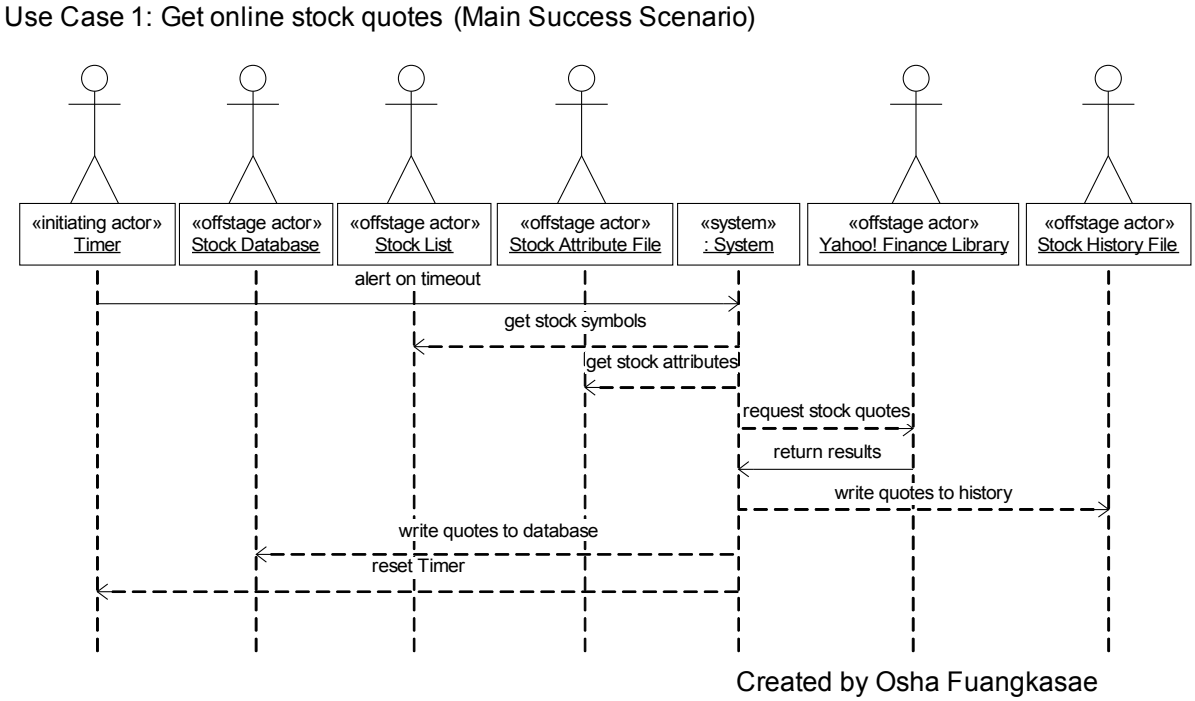


Figure 4: UC2 System Sequence Diagram

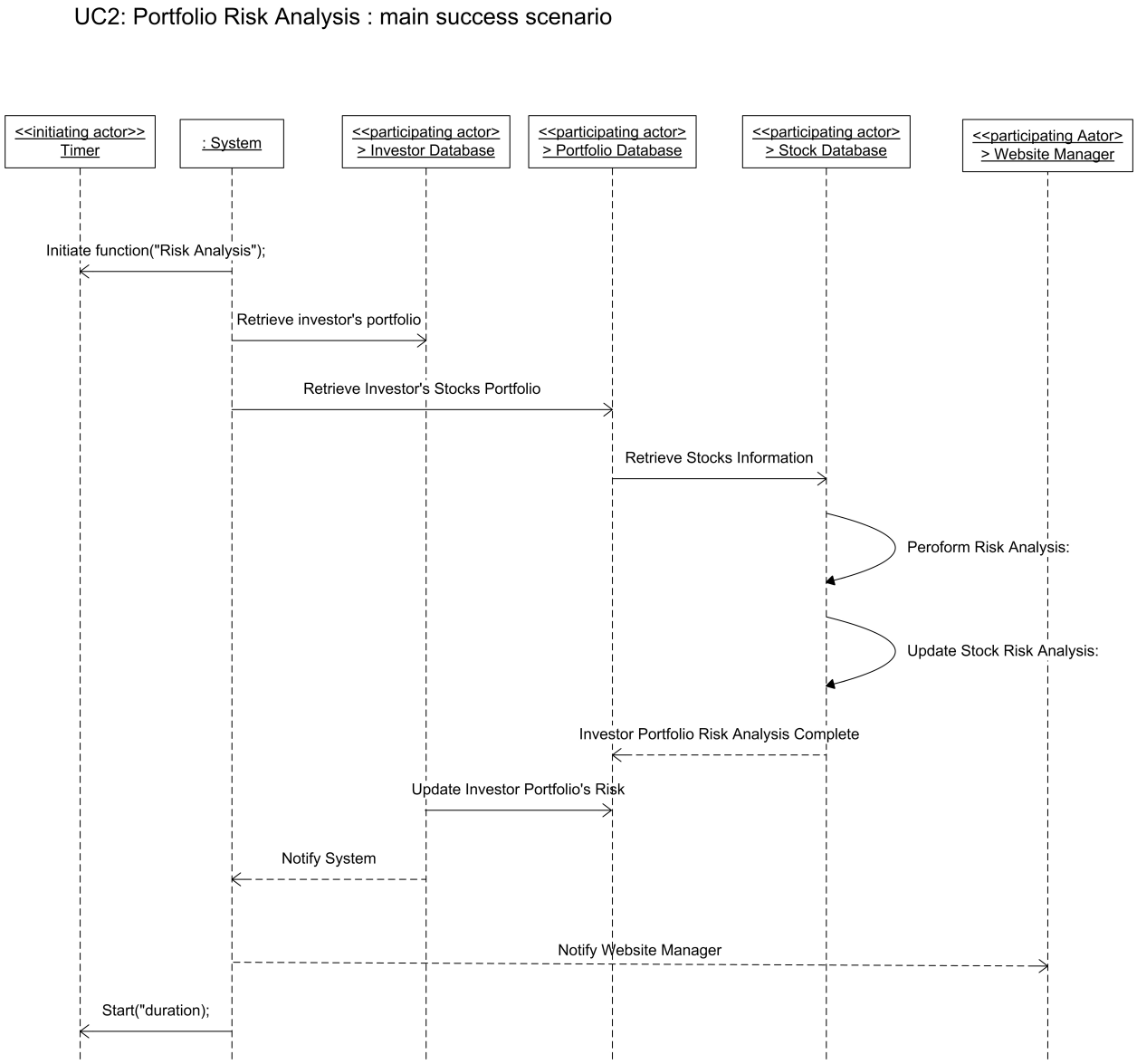


Figure 5: UC3 System Sequence Diagram

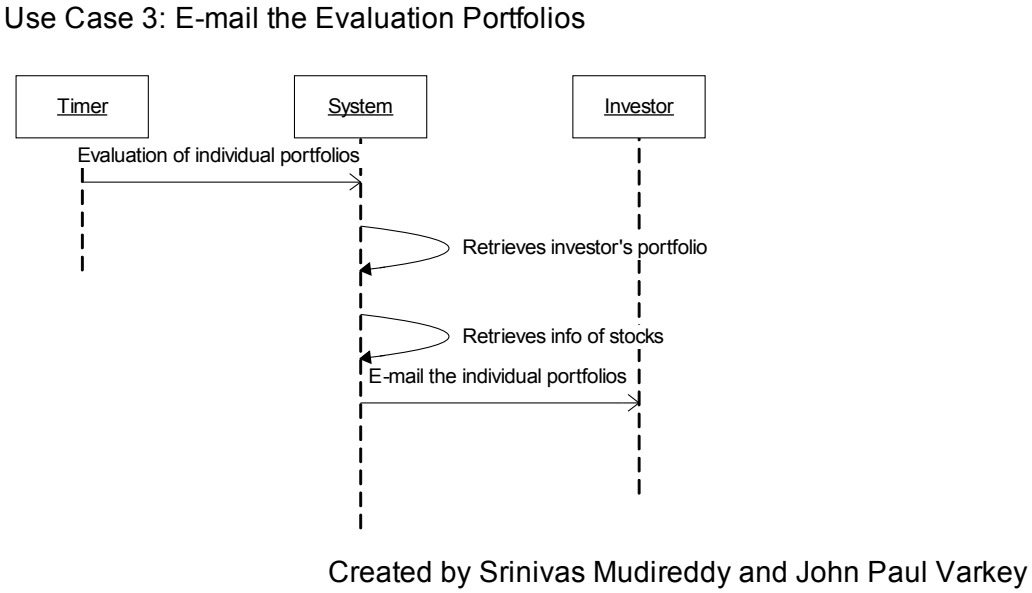


Figure 6: UC 4 System Sequence Diagram

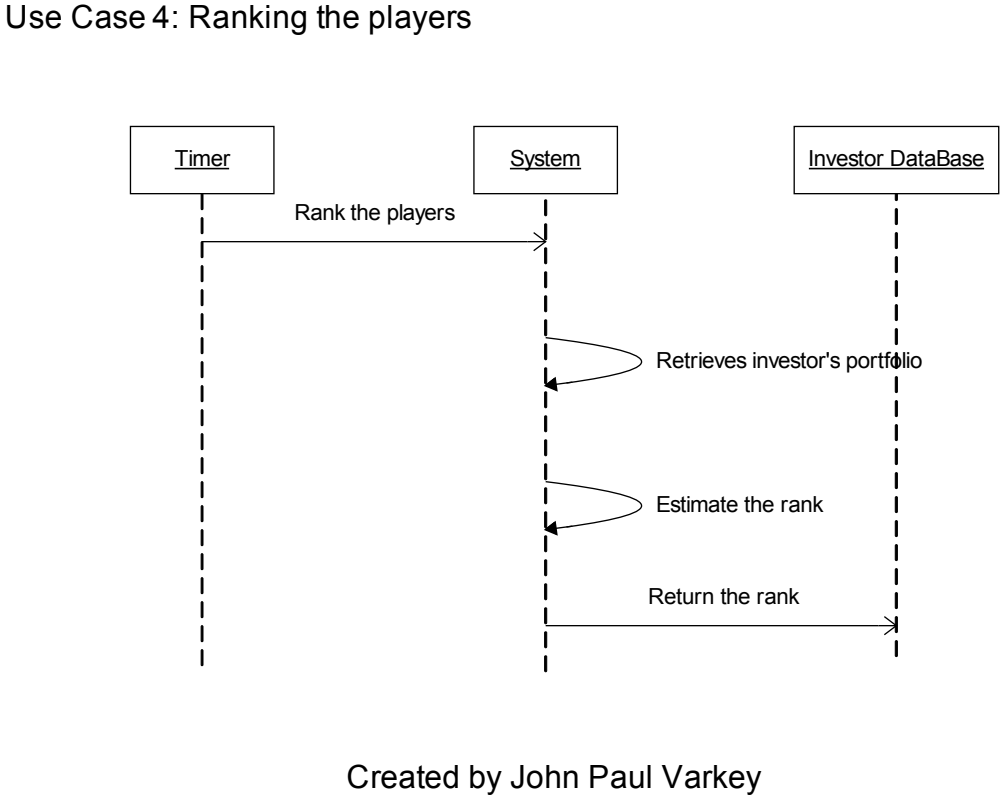
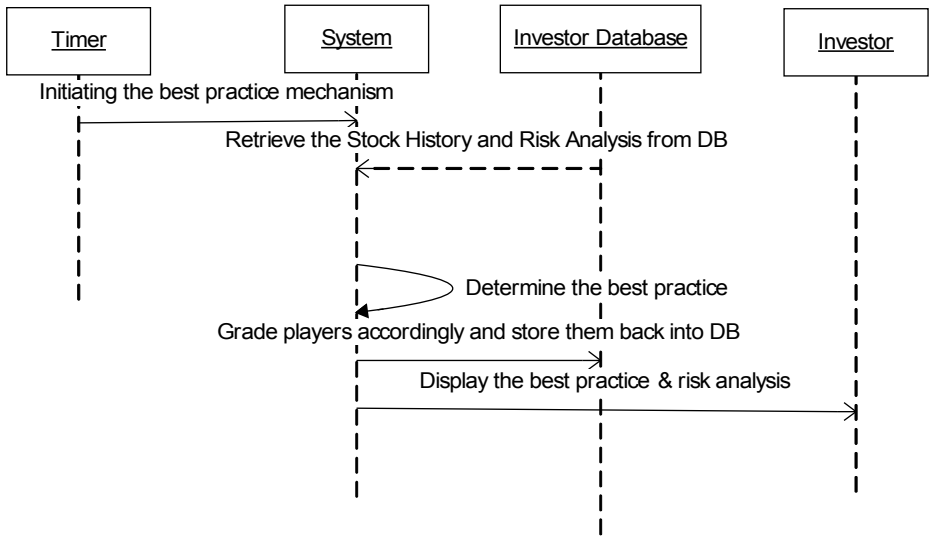


Figure 7: UC5 System Sequence Diagram

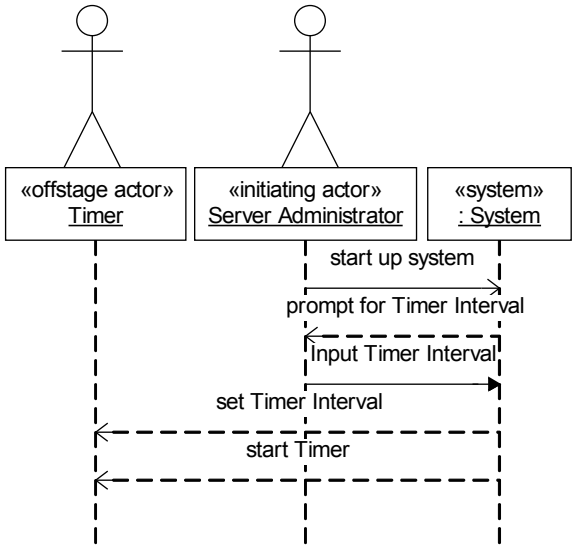
UC5: Identify the best practices



Created by Ravi Gudur

Figure 8: UC6 System Sequence Diagram

Use Case 6: Set Timer Interval (Main Success Scenario)



Created by Osha Fuangkasae

Notes: The Timer is set to constantly update the Stock Database. If there are currently no registered Investors, the Timer will still run for UC1, but the SUD will not perform any Investor related functions once it sees there are no Investors in the Investor Database.

Figure 9: UC7 System Sequence Diagram

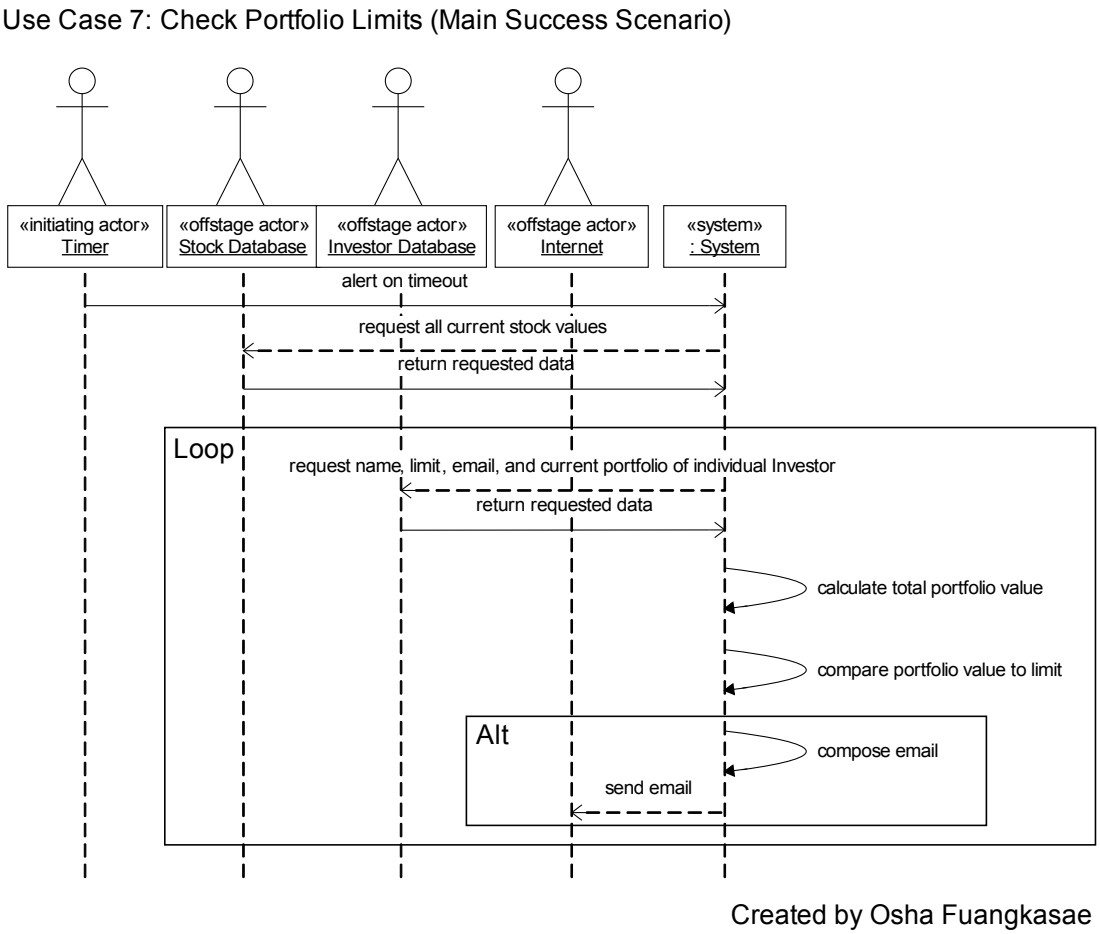
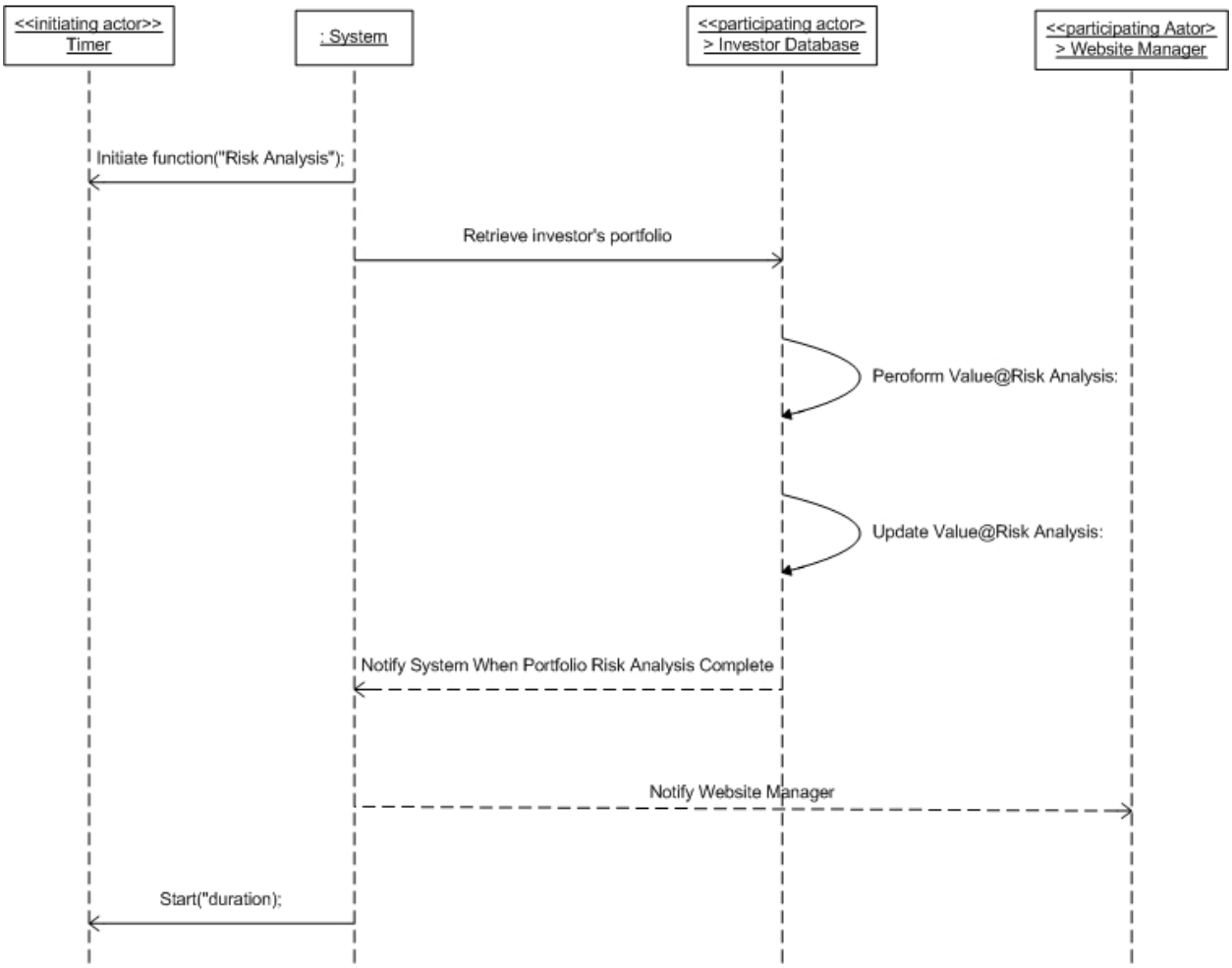


Figure 10 UC11 System Sequence Diagram

UC11: Portfolio Risk Analysis – Value at Risk



Created by Zong-Zhi Lin

Nonfunctional Requirements

FURPS+

F(unctionalities): Expressed with use cases.

U(sability): All of the operations performed by our SUD execute and complete in 10 seconds or less, not counting requests to outside entities.

R(eliability): If stock request times out, notify the Website Manager and use past stock information. System must be up 365 days x 24 hours and break down time < 1 hour per year. Currently, we do not have any statistics on how frequently this happens for Yahoo!Finance. With more testing, we will develop monitor this statistics with more testing. If the time-out request frequency is high, we may need to suggest to Website Manager a switch to Google!Finance.

P(erformance): User requested services response time must be under 10 seconds via broadband connection and less than 1 minute via dial-up connection.

S(upportability):

+(Other):

Domain Analysis

Domain Tables

Table 1: Concept Definitions

Concept Description	Type	Concept Name	Used in
It is the Timer Controller's responsibility to interface between the Controller and the Timer.	D	Timer Controller	UC1 - UC11
The Controller coordinates all of the messages between concepts and knows how to perform each use case.	D	Controller, UC2 Controller, UC5 Controller	UC1 - UC11
The Database Communicator interfaces between the Portfolio Database, Stock Database, Investor Database, and the Controller	D	Database Communicator	UC1 – UC11
The System Menu displays the available system functions.	D	System Menu	UC1 - UC11
The Yahoo! Finance Library Communicator communicates with the Yahoo! Finance Library.	D	Yahoo! Finance Library Communicator	UC1
E-mails the Investor with the current status of her portfolio.	D	E-Mailer	UC3
Provides the profit value of each portfolio.	D	Evaluation of Portfolios	UC3
Provides the risk analysis of each Investor's portfolio	D	Risk Analysis Function	UC2
Provides the top Investors with the largest profits.	D	Best Investor	UC5

Table 2: Association Definitions

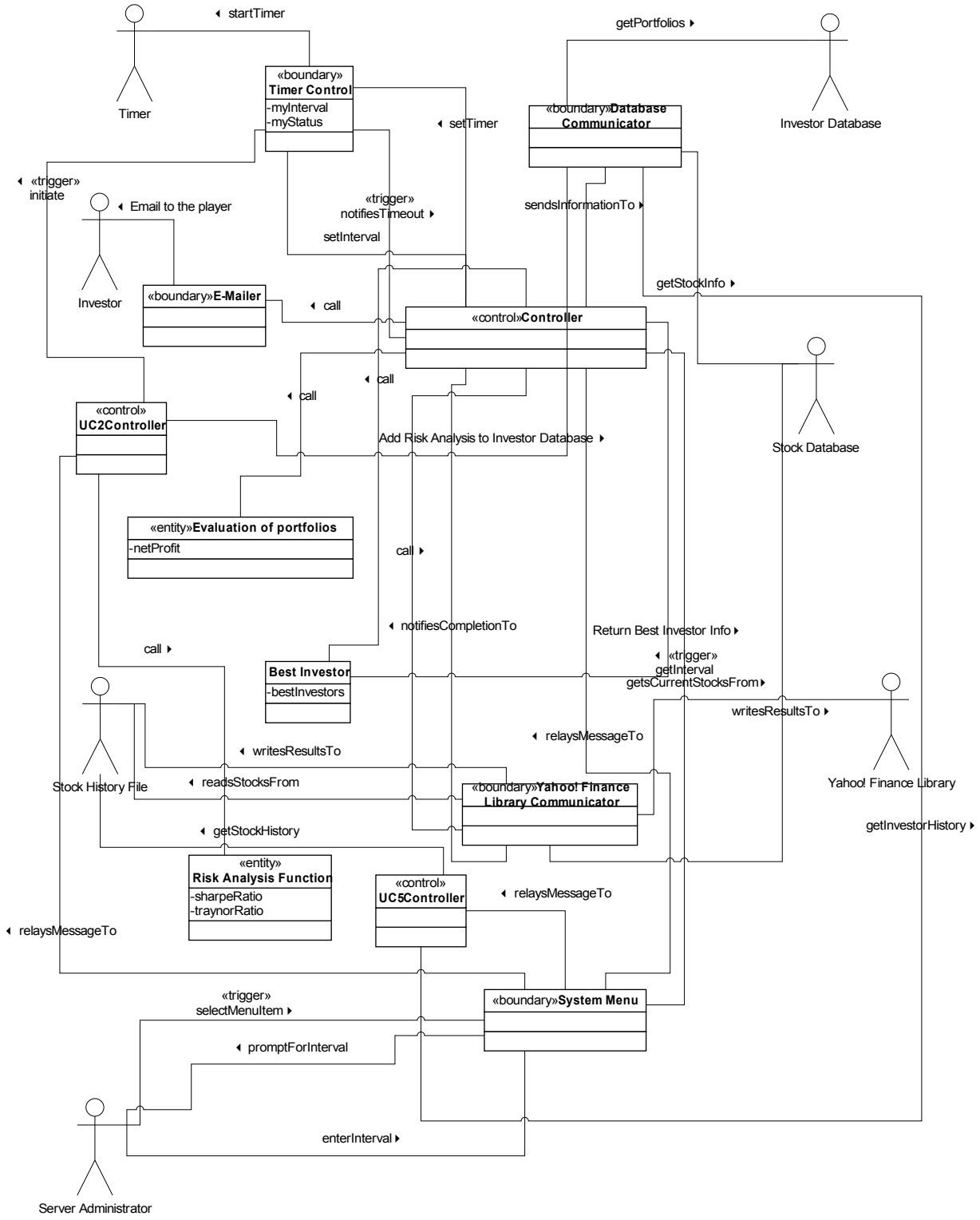
Association Description	Association Name	Used in
Writes any results to the connected concept	writesResultsTo	UC1
Gathers stock quotes	getCurrentStocksFrom	UC1
Return success on task completion	notifiesCompletionTo	UC1
Calls a system script	calls	UC1
Sets the Timer myInterval field	setTimer	UC6
Displays prompt for Timer interval	promptForInterval	UC6
User input of Timer interval	enterInterval	UC6
Reset and starts the Timer	startTimer	UC1, UC6
Alert listeners on timeout	notifiesTimeout	UC1
Reads the stock names to track	readsStocksFrom	UC1
Select the relevant option from System Menu	selectMenuItem	UC1, UC6
Relay Server Administrator's request	relaysMessageTo	UC1, UC6
Requests for Timer interval	getInterval	UC6
Activates main function of connected concept	call	UC2 – UC11
Adds the Risk Analysis data to the Investor Database	Add Risk Analysis to Investor Database	UC2
Returns who is the best player	Return Best Investor Info	UC4
E-mails the player with the portfolio evaluation	Email to the player	UC3
Gets the history of the Investor's portfolio	getInvestorHistory	UC5
Gets the history of all Stocks	getStockHistory	UC5

Table 3: Attribute Definitions

Attribute Description	Attribute Name	Used in
Provides the Stock Portfolio's marginal gain Vs total volatility	sharpeRatio	UC2
Provides the investor Portfolio's maximum loss not exceeded with a given probability defined as the confidence level, over a given period of time.	valueAtRisk	UC11
Net profit for each Investor	netProfit	UC4
Timer interval	myInterval	UC6
Current Timer status (timed out or ticking)	myStatus	UC6

Domain Model Diagrams

Figure 11: Domain Model



Mathematical Models Used in SUD

We found measures to determine Risk Analyses of Stock Portfolios in the book *Investing* (Ref. 2). The authors refer to two commonly used and known measures (See Algorithm section in this report for more technical details):

The *Shapre Ratio Measure* divides average portfolio excess return over the sample period by the standard deviation of returns over that period. It measures the reward to total volatility trade-off.

The *Value at Risk (VaR) Measure* is the maximum loss not exceeded with a given probability defined as the confidence level, over a given period of time. It is commonly used by security houses or investment banks to measure the market risk of their asset portfolios (**market value at risk**), although VaR is a very general concept that has broad applications. VaR is widely applied in finance for quantitative risk management for many types of risk. VaR does not give any information about the severity of loss by which it is exceeded.

Written by Zong-Zhi Lin

User Interface Design

Our SUD does not require a UI beyond the text based interaction with the Web Manager. As Ivan Marsic indicates, “I suggest you just ignore the UI section and assume that the rest of your report is worth up to 100 %.”

Interaction Diagrams

Do interaction diagrams for the use cases you elaborated (fully described) in Report #1

Mention explicitly what design principles you employ in the process of assigning responsibilities to objects. This can be done either as comment "bubbles" in the diagram, or in the caption of the diagram.

Figure 12: Use Case 1 Interaction Diagram (Created by Osha Fuangkasae)

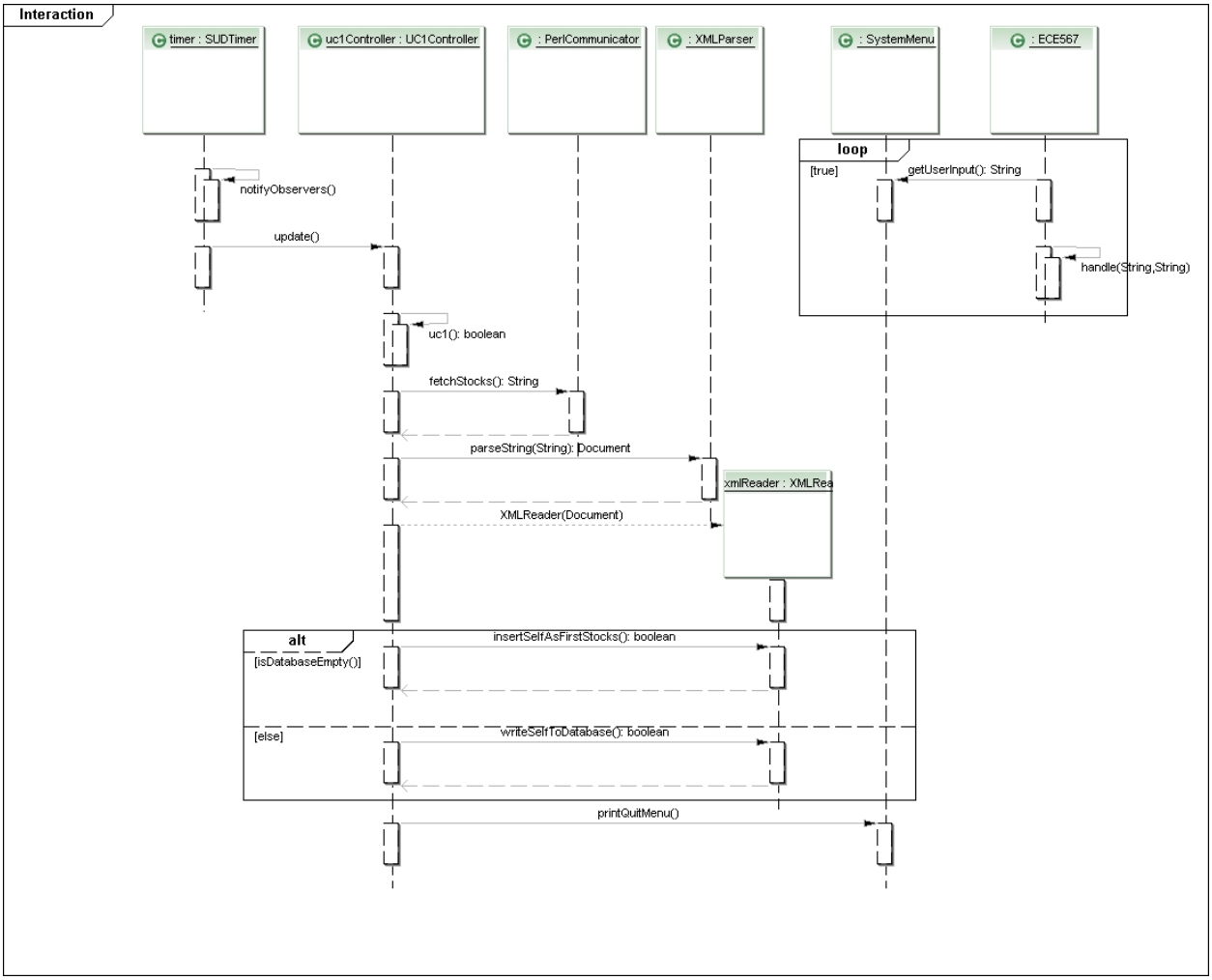
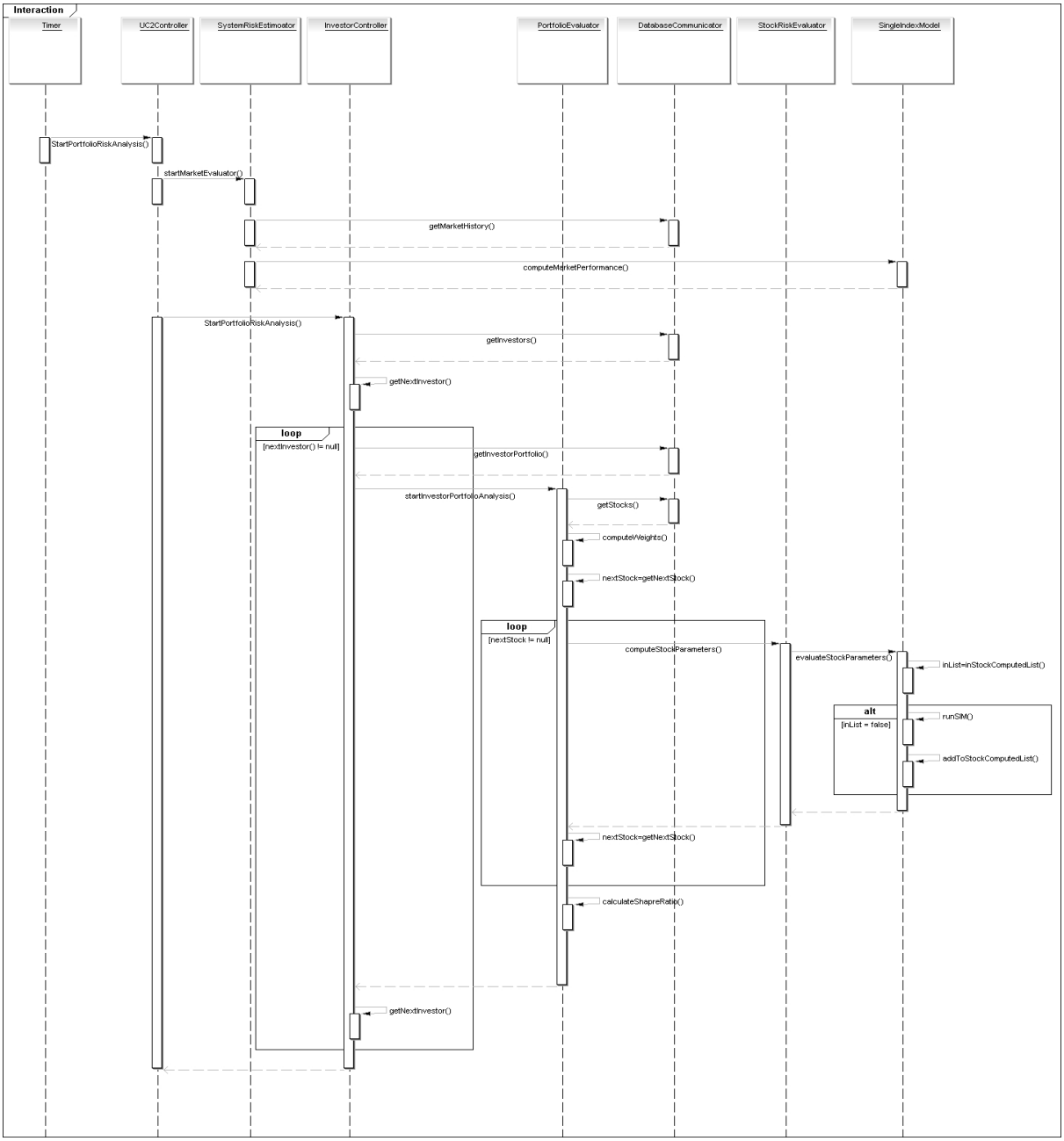


Figure 13: Use Case 2 Interaction Diagram (Created by Zong-Zhi Lin)



Notes: The diagram starts with calculating the performance of the market index (such as Dow Jones Industrial Average), which will be used as the reference to understand and calculate the performance of a stock. It then follows with two loops; the outside loop iterates over investors and the inside loop iterates all stocks of an investor's portfolio. Single Index Model is used to calculate the performance of a stock (such as return and risk/variance) relative to a market index or benchmark index that represents the performance of the macroeconomics. The information of the stocks in each investor's portfolio will be used to calculate his or her portfolio's Sharpe-ratio.

Notes (cont.): Design principles used for assigning responsibilities are Expert Doer principle (SingleIndexModel class), High Cohesion principle (added InvestorController class to reduce the amount of responsibilities that could be taken on by UC2Controller class), and Low Coupling principle (classes are not over-burdened).

Figure 14: Use case 3 interaction diagram (Created by Srinivas Mudireddy)

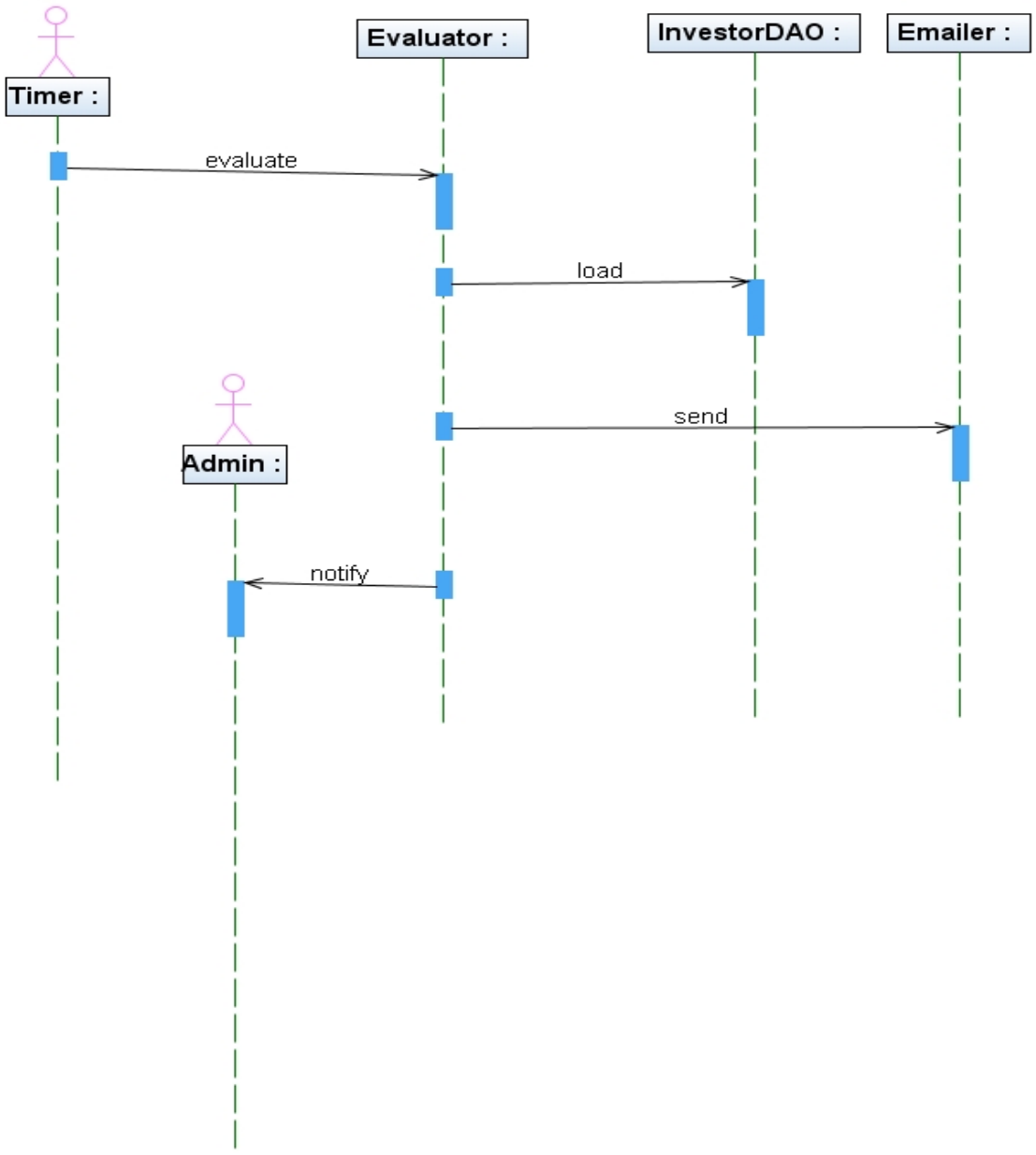


Figure 15: Use case 4 interaction diagram (Created by John Paul Varkey)

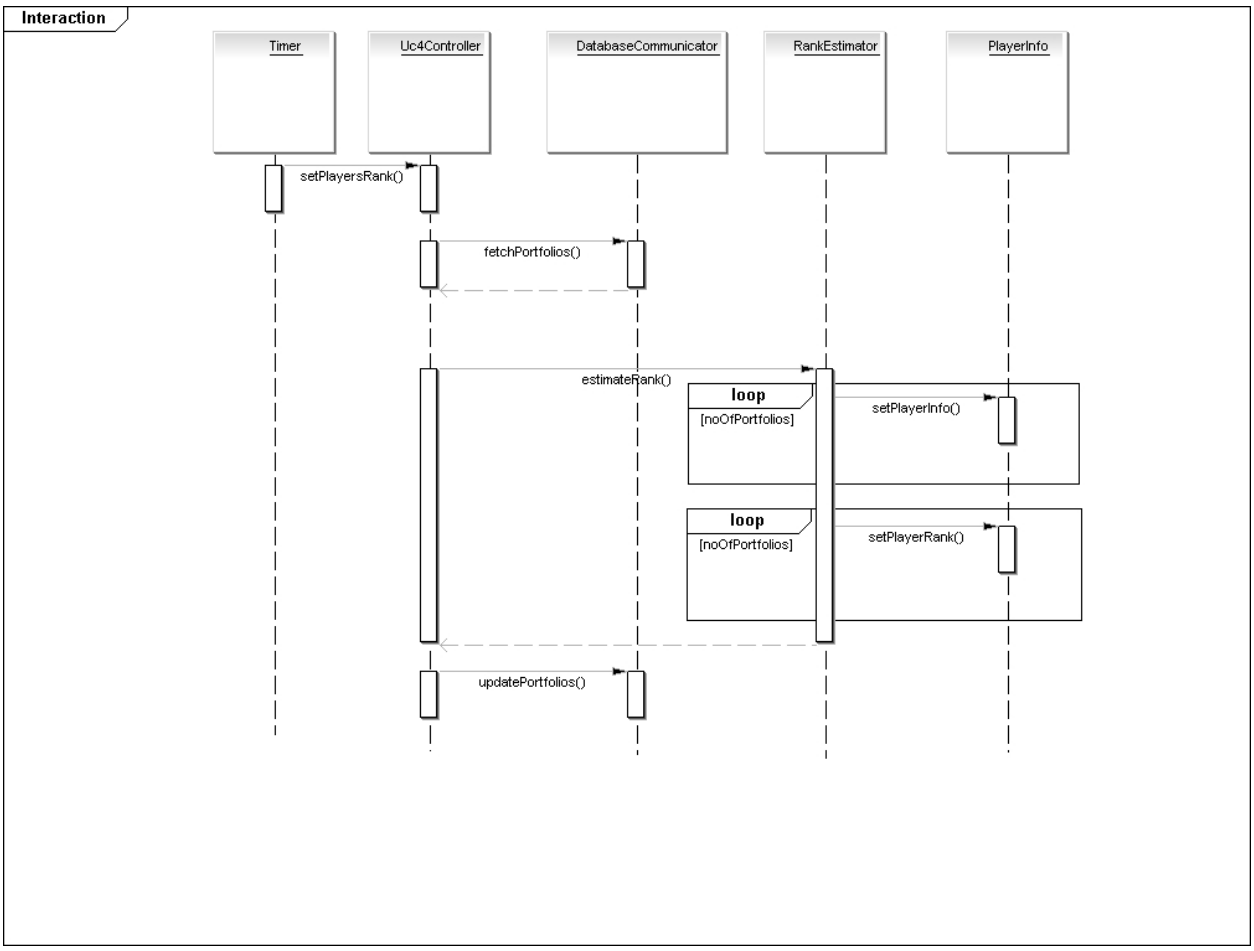


Figure 16: Use case 5 interaction diagram (Created by Ravi Gudur)

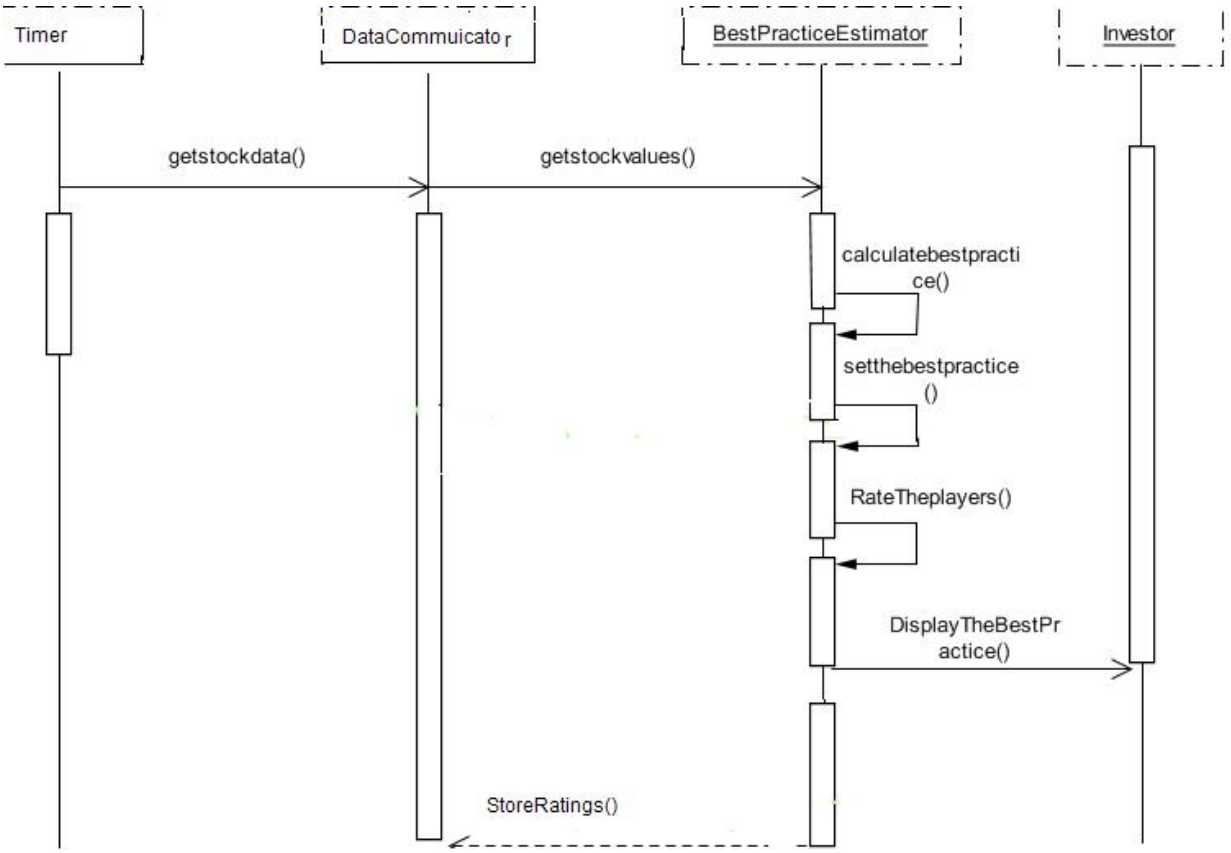


Figure 17: Use Case 6 Interaction Diagram (Created by Osha Fuangkasae)

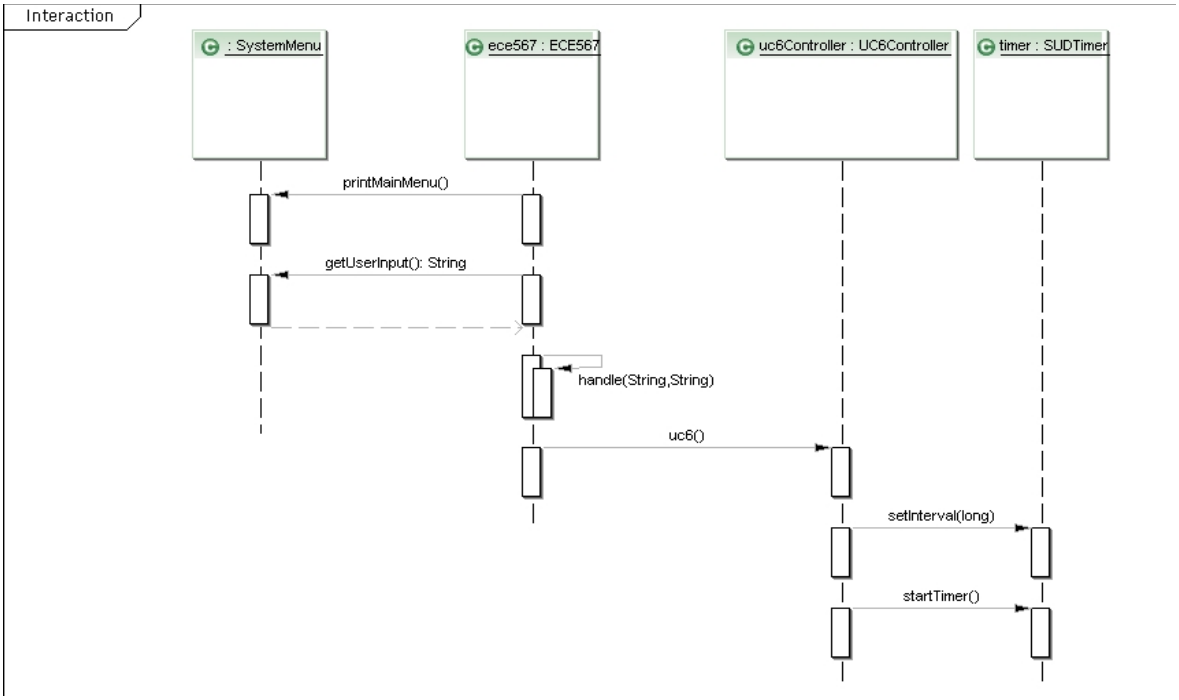


Figure 18: Use Case 7 Interaction Diagram (Created by Osha Fuangkasae)

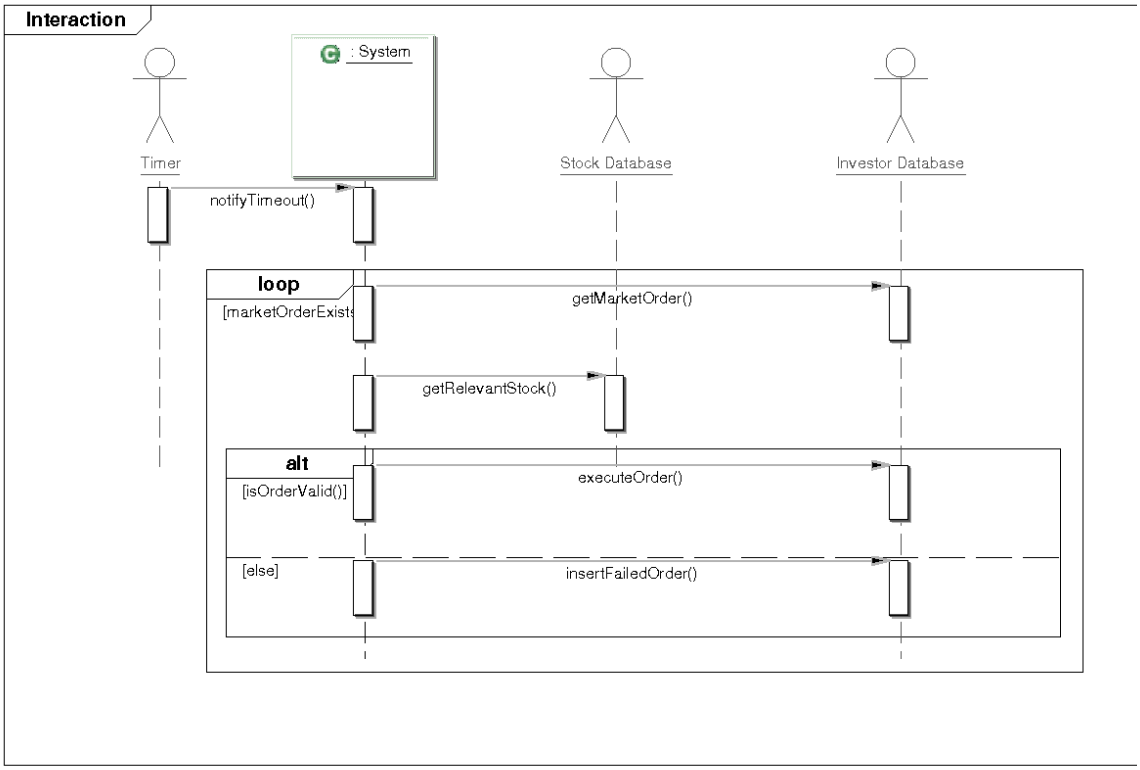


Figure 19: Use Case 8 Interaction Diagram (Created by Osha Fuangkasae)

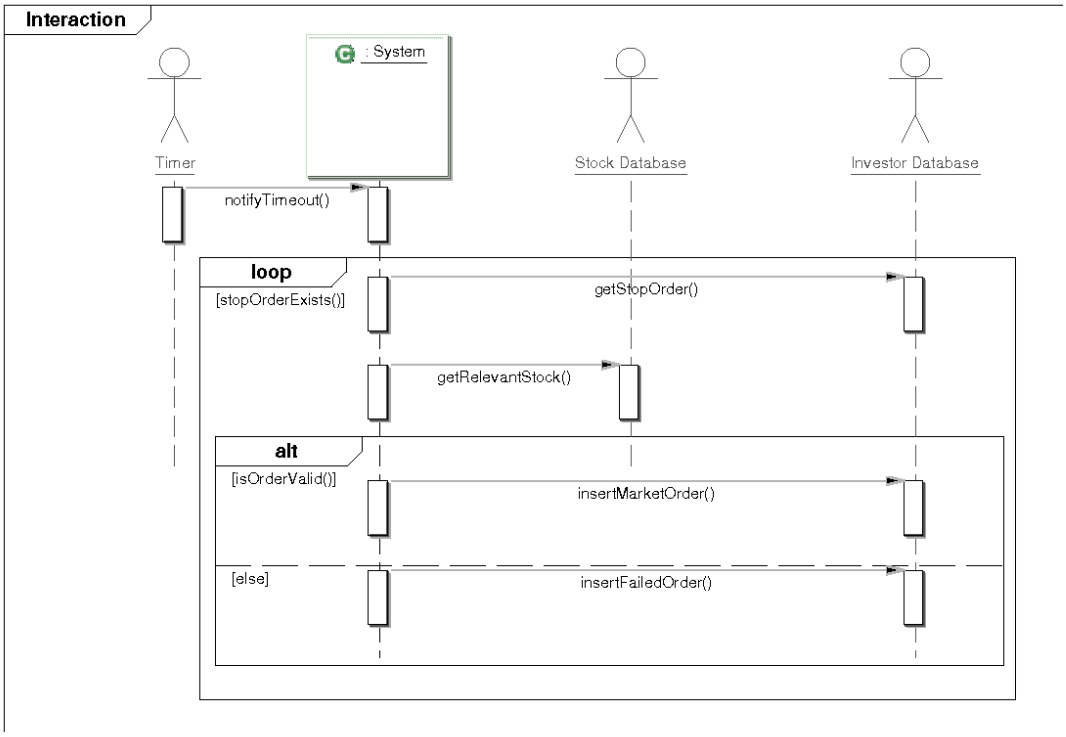
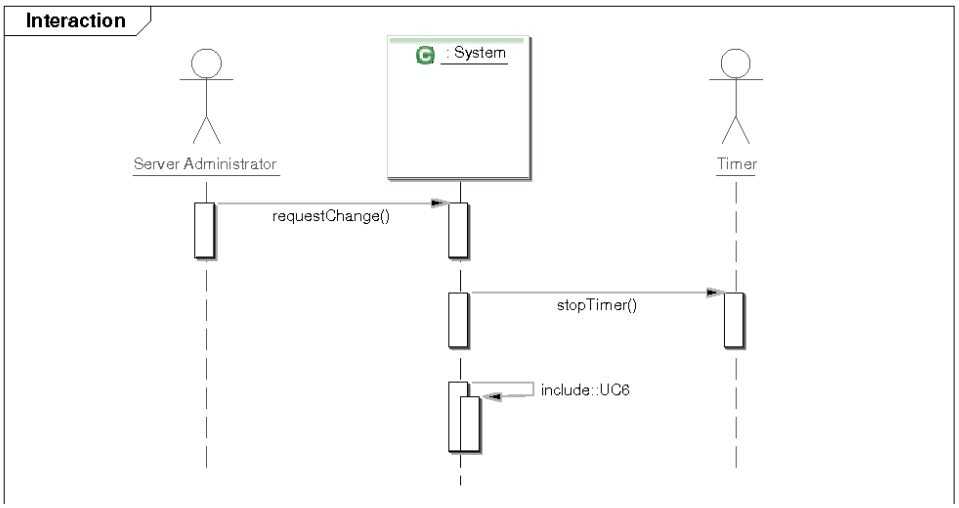


Figure 20: Use Case 9 Interaction Diagram (Created by Osha Fuangkasae)



Class Diagram and Interface Specification

Class Diagram

Show all classes and their associations. Only indicate visibilities of attributes and operations; full details about the types and signatures should be provided in the next item.

Notes: Our class diagram is so large that we split it into four quadrants and print them all here.

Figure 21: Complete Class Diagram

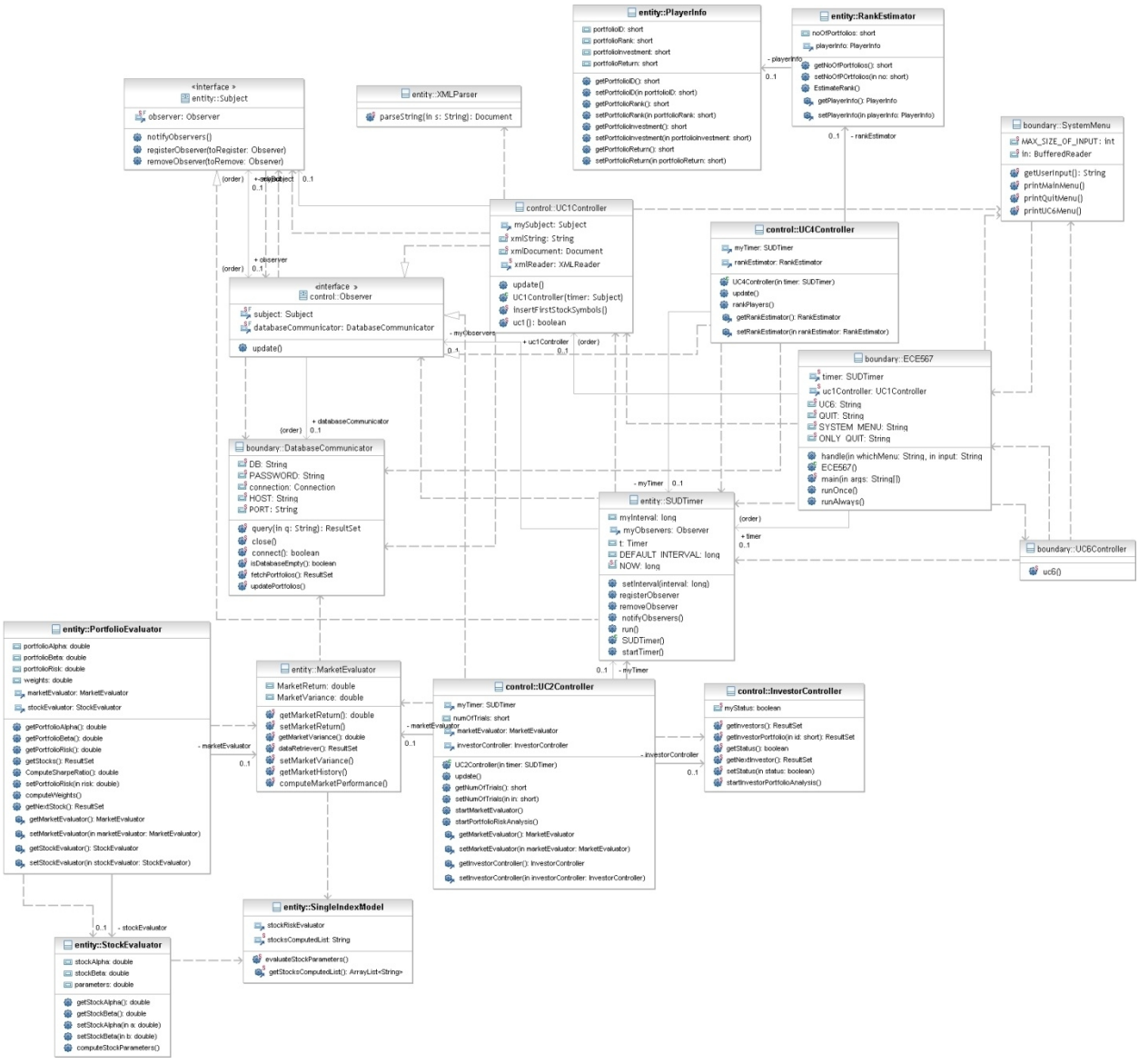


Figure 22: Zoom in of top left

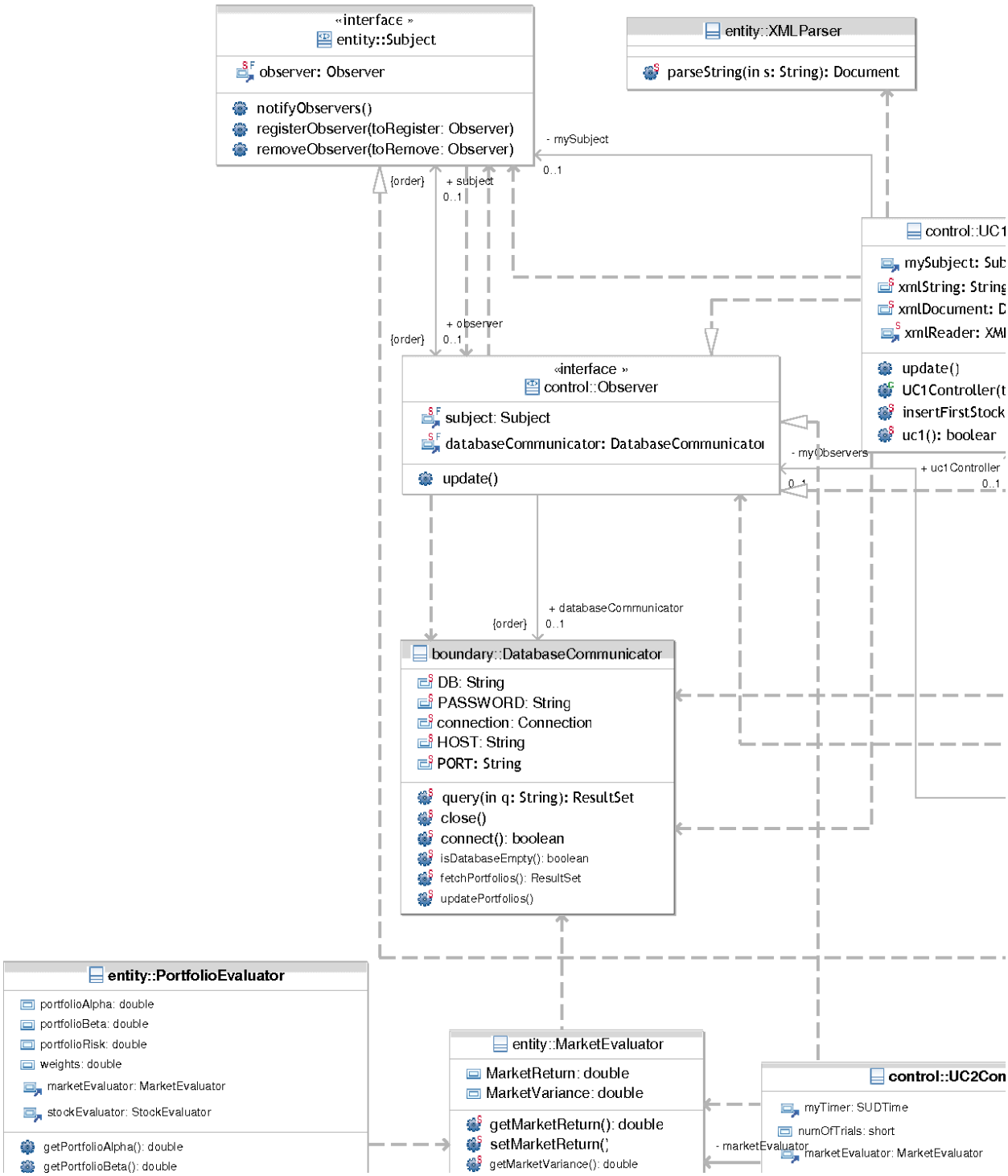


Figure 23: Zoom in of top right

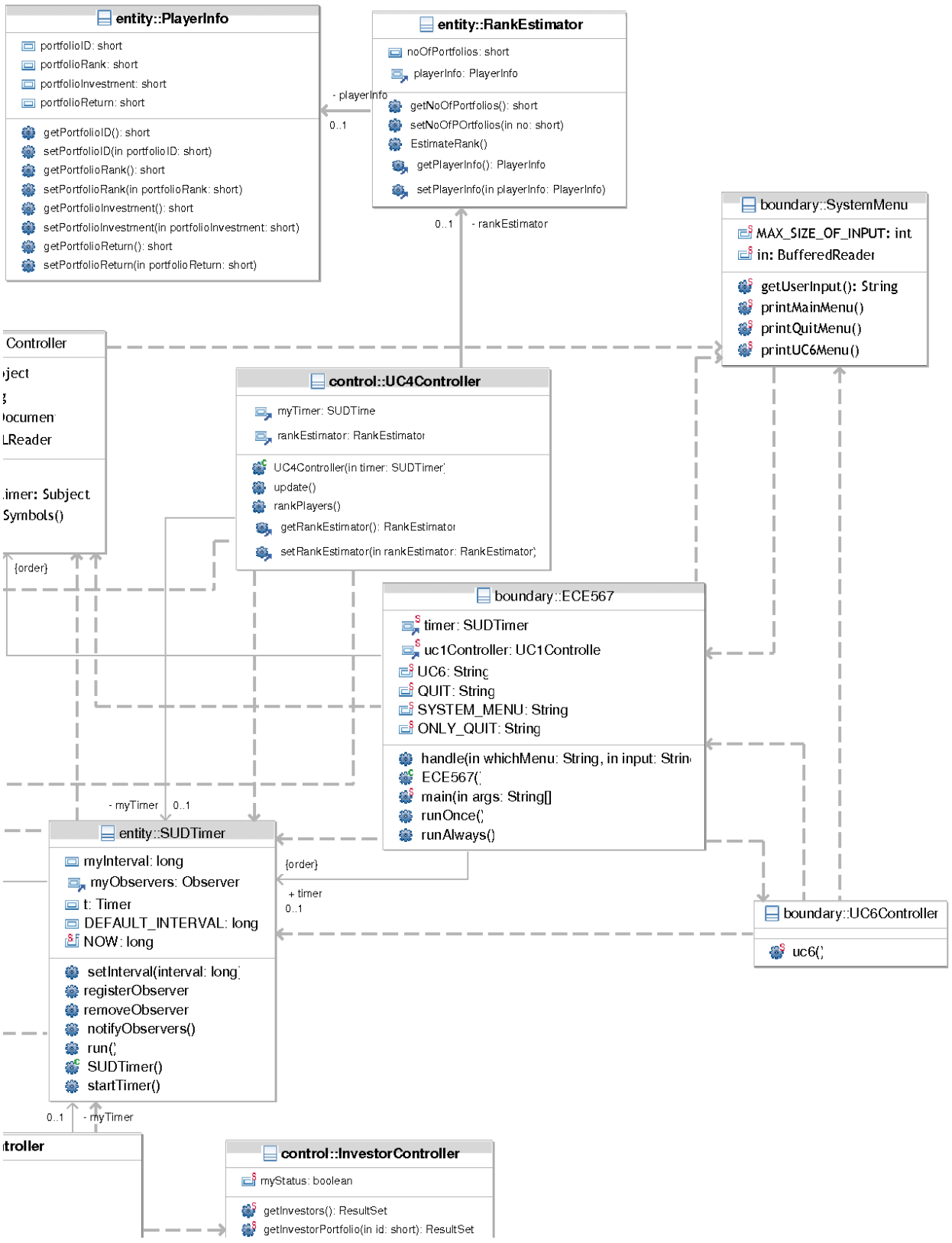
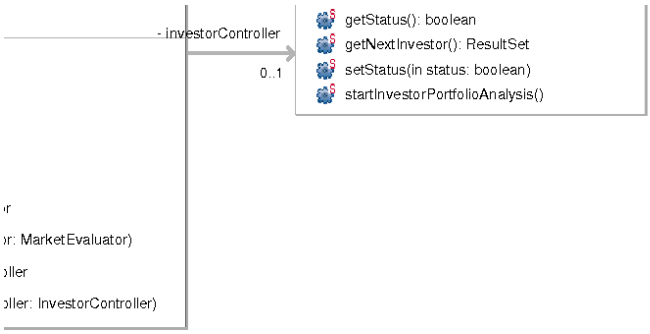


Figure 24: Zoom in of bottom left

Figure 25: Zoom in of bottom right

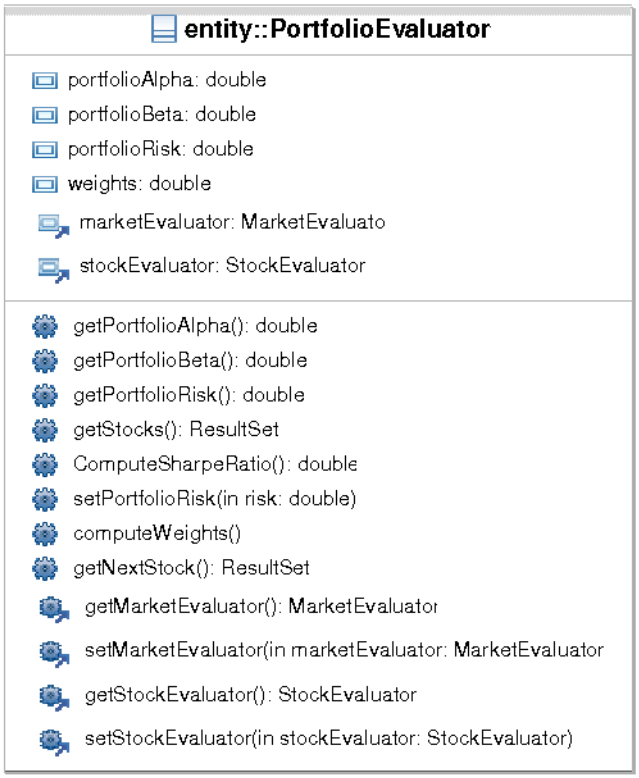


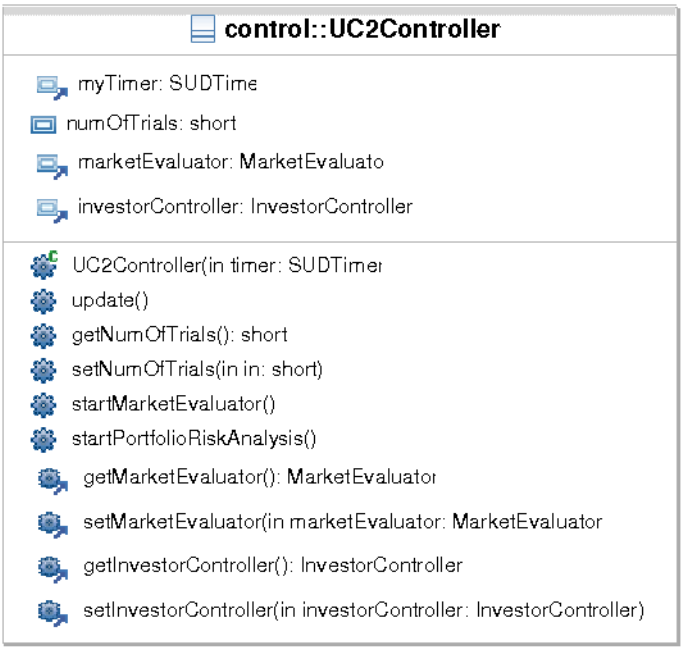
Data Types and Operation Signatures

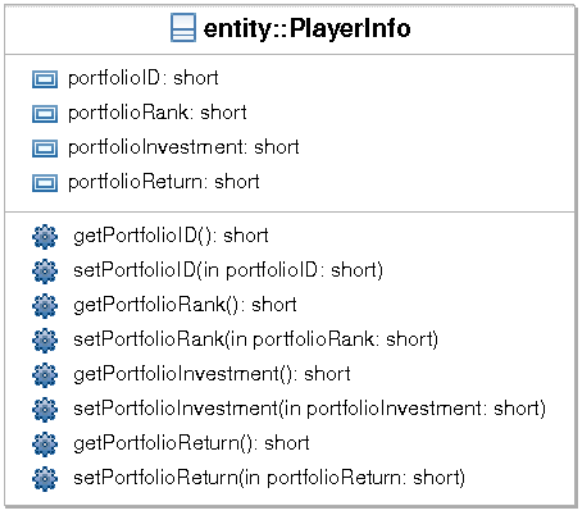
Independently of the class diagram, write down class specification in UML notation. For every class, specify data types of all attributes and operation signatures.

Notes: For unknown reasons, some of our type descriptions are cut off at the end. We believe that this is due to an issue caused by the program that we are using to draw the diagrams. We are using Eclipse-UML, an extension found at http://www.omondo.com/ (ref: Omondo)

Figure 26: Datatype definitions 1


entity::PortfolioEvaluator
portfolioAlpha: double
portfolioBeta: double
portfolioRisk: double
weights: double
marketEvaluator: MarketEvaluato
stockEvaluator: StockEvaluator
getPortfolioAlpha(): double
getPortfolioBeta(): double
getPortfolioRisk(): double
getStocks(): ResultSet
ComputeSharpeRatio(): double
setPortfolioRisk(in risk: double)
computeWeights()
getNextStock(): ResultSet
getMarketEvaluator(): MarketEvaluator
setMarketEvaluator(in marketEvaluator: MarketEvaluator
getStockEvaluator(): StockEvaluator
setStockEvaluator(in stockEvaluator: StockEvaluator)


control::UC2Controller
myTimer: SUDTime
numOfTrials: short
marketEvaluator: MarketEvaluato
investorController: InvestorController
UC2Controller(in timer: SUDTimer
update()
getNumOfTrials(): short
setNumOfTrials(in in: short)
startMarketEvaluator()
startPortfolioRiskAnalysis()
getMarketEvaluator(): MarketEvaluator
setMarketEvaluator(in marketEvaluator: MarketEvaluator
getInvestorController(): InvestorController
setInvestorController(in investorController: InvestorController)


entity::PlayerInfo
portfolioID: short
portfolioRank: short
portfolioInvestment: short
portfolioReturn: short
getPortfolioID(): short
setPortfolioID(in portfolioID: short)
getPortfolioRank(): short
setPortfolioRank(in portfolioRank: short)
getPortfolioInvestment(): short
setPortfolioInvestment(in portfolioInvestment: short)
getPortfolioReturn(): short
setPortfolioReturn(in portfolioReturn: short)

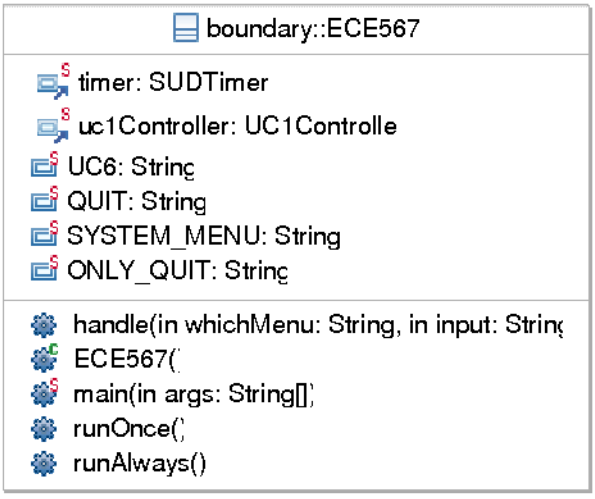

boundary::ECE567
timer: SUDTimer
uc1Controller: UC1Controlle
UC6: String
QUIT: String
SYSTEM_MENU: String
ONLY_QUIT: String
handle(in whichMenu: String, in input: Strin
ECE567(
main(in args: String[])
runOnce()
runAlways()

Figure 27: Datatype definitions 2

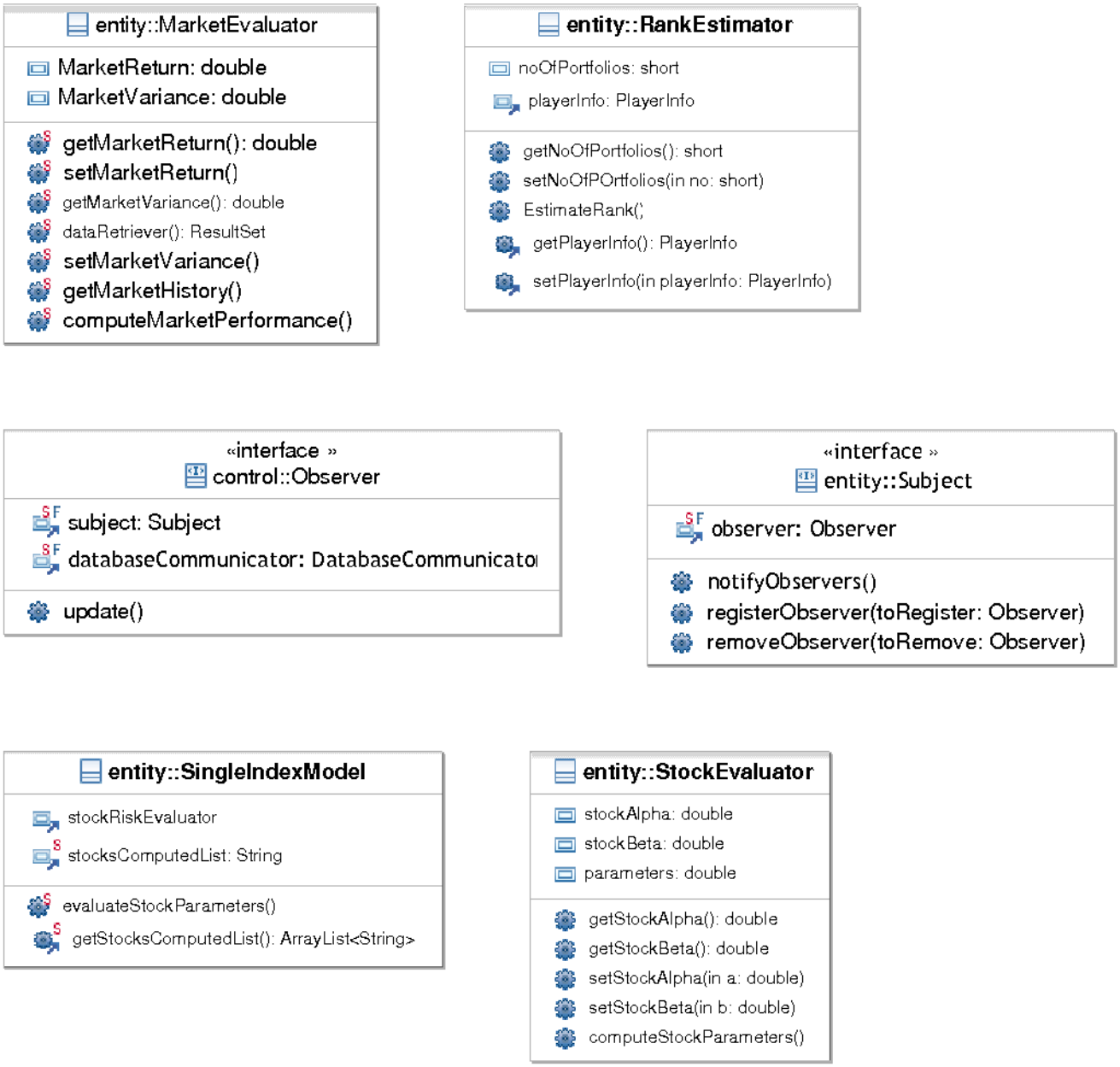
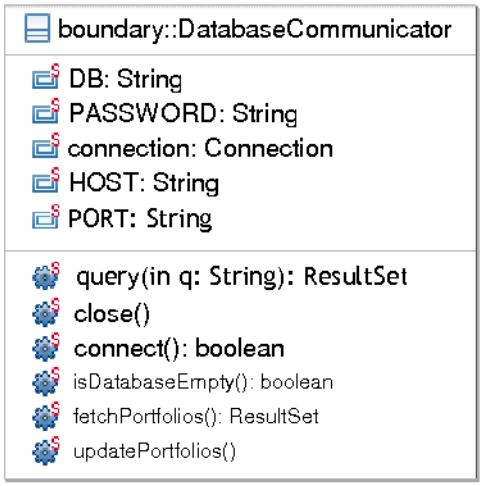
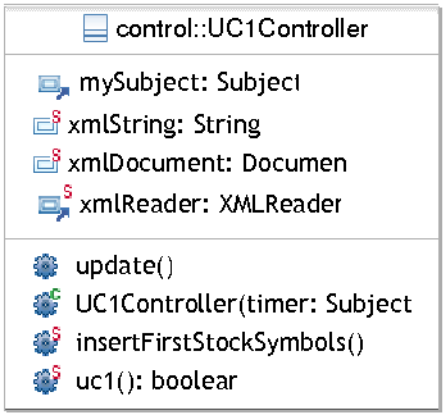
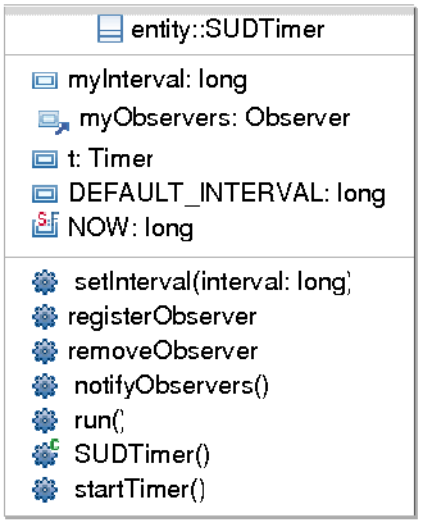


Figure 28: Datatype definitions 3







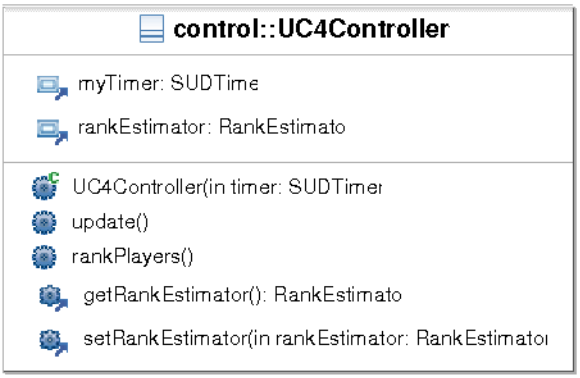
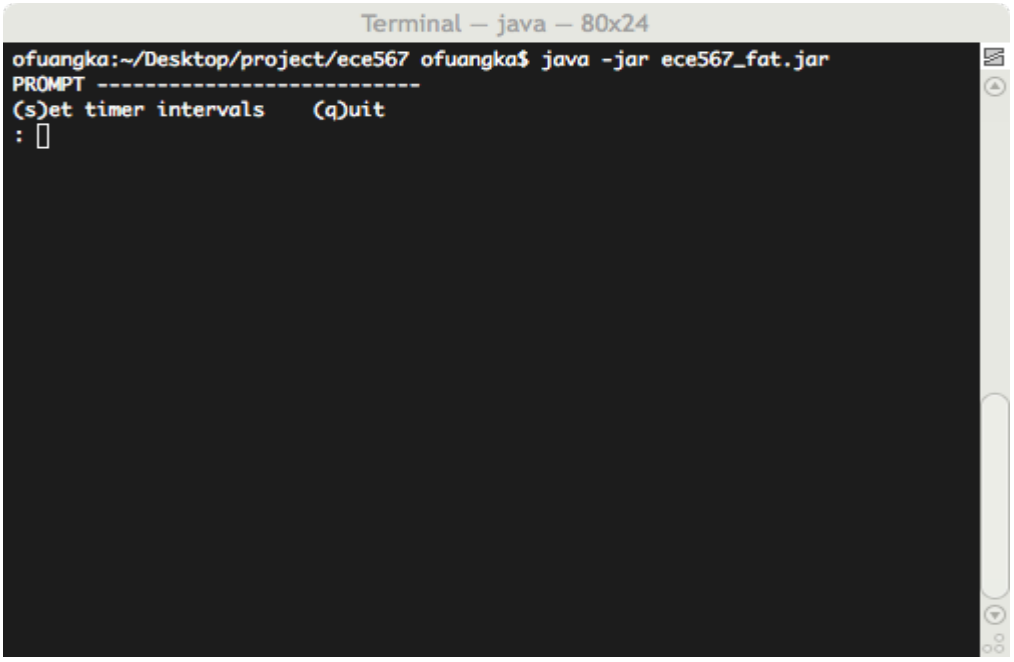


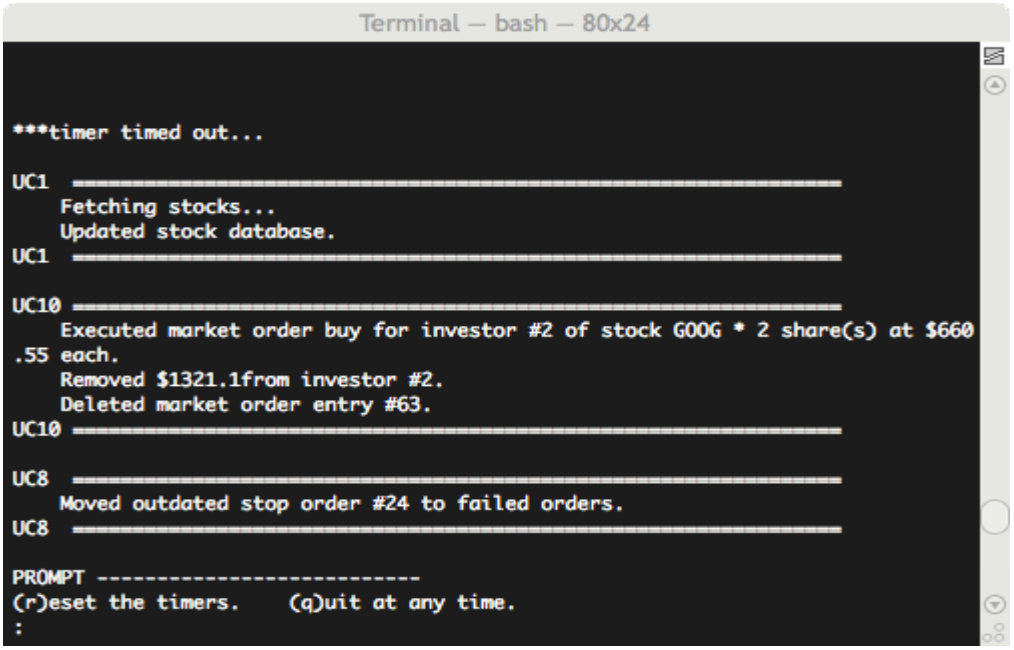
Figure 29: Datatype definitions 4

System Screen Shots of Use Cases

Main Menu (By Osha Fuangkasae)

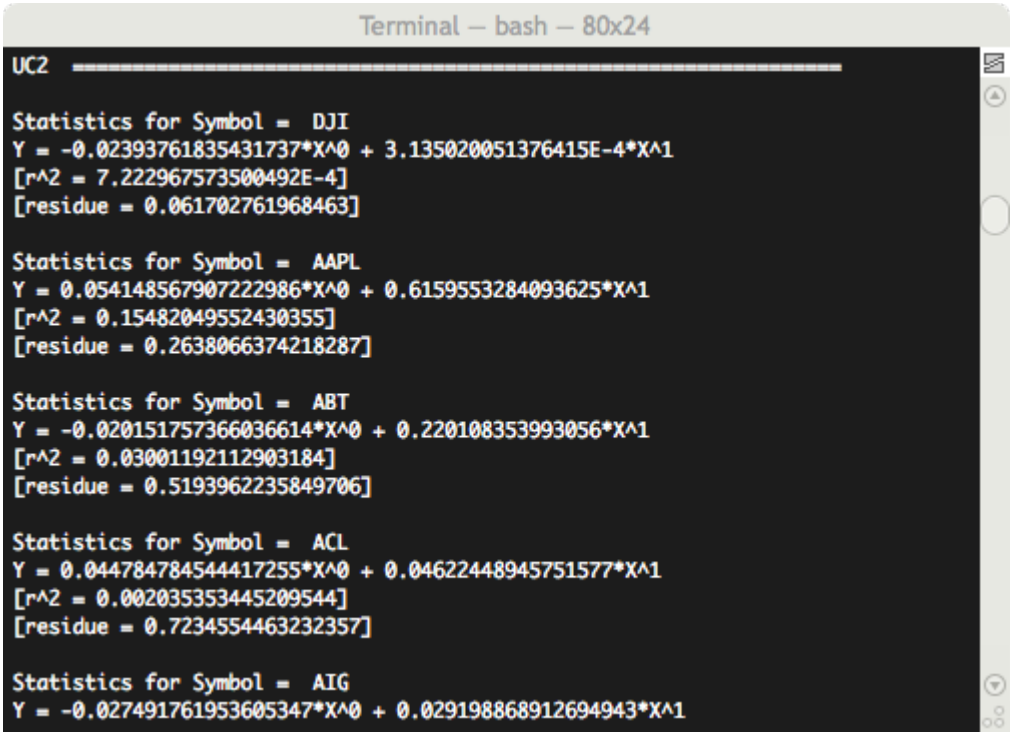


Use Case 1, 8, and 10 (By Osha Fuangkasae)



Use Case 2 – Stock Performances (By Zong-Zhi 'Alex' Lin)

This table summarizes the curve fitting of stocks available in Table stock. Except market benchmark, DJI (Dow Jones Industry Average), we use least square linear regression to draw the performance relation of each stock to the performance of Market Benchmark, DJI.



Statistics for Symbol = DJI Y = -0.02393761835431737*X^0 + 3.135020051376415E-4*X^1 [r^2 = 7.222967573500492E-4] [residue = 0.061702761968463]	Statistics for Symbol = CAJ Y = 0.017881965952352 45*X^0 + 0.5980464591451112*X^1 [r^2 = 0.3577883986665908] [residue = 1.0031721928998762]
Statistics for Symbol = AAPL Y = 0.05414856790722298 6*X^0 + 0.6159553284093625*X^1 [r^2 = 0.15482049552430355] [residue = 0.2638066374218287]	Statistics for Symbol = CHL Y = -0.016649655262964497*X^0 + 0.09668522133920528*X^1 [r^2 = 0.0072374862714493565] [residue = 1.0040207172616458]
Statistics for Symbol = ABT Y = -0.020151757366036614*X^0 +	Statistics for Symbol = CSCO Y = 0.008259416616828994*X^0 +

0.220108353993056*X^1 [r^2 = 0.03001192112903184] [residue = 0.5193962235849706]	-0.47786981945269036*X^1 [r^2 = 0.5049727927876295] [residue = 1.0024338519121434]
Statistics for Symbol = ACL Y = 0.044784784544417255*X^0 + 0.04622448945751577*X^1 [r^2 = 0.002035353445209544] [residue = 0.7234554463232357]	Statistics for Symbol = CVX Y = 0.03169044584554275*X^0 + -0.3165394790240628*X^1 [r^2 = 0.12733591659016075] [residue = 1.0025218581807864]
Statistics for Symbol = AIG Y = -0.027491761953605347*X^0 + 0.029198868912694943*X^1 [r^2 = 0.0012313860775926236] [residue = 0.8521092143774384]	Statistics for Symbol = DELL Y = 0.004098551105218 3425*X^0 + 0.22641378073637802*X^1 [r^2 = 0.10774865684887827] [residue = 1.0020674669883531]
Statistics for Symbol = AMD Y = -0.017194024983920374*X^0 + -0.8913706513534689*X^1 [r^2 = 0.5663141418094028] [residue = 0.9243524488771816]	Statistics for Symbol = DNA Y = -0.010335905085862419*X^0 + -0.6106756454151807*X^1 [r^2 = 0.8614577086807889] [residue = 1.001147327001419]
Statistics for Symbol = AMGN Y = 0.05176948339545441*X^0 + 0.7840007161256178*X^1 [r^2 = 0.3028674457271967] [residue = 0.9642317254697648]	Statistics for Symbol = EMC Y = -0.008339218327061 73*X^0 + -0.05268273096631594*X^1 [r^2 = 0.015208827203458267] [residue = 1.0009156003811592]
Statistics for Symbol = AXA Y = 0.01054121740254555*X^0 + -0.21711610736252435*X^1 [r^2 = 0.044167951266627925] [residue = 0.9839300936370189]	Statistics for Symbol = GME Y = -0.02981791762572406*X^0 + 0.007179071696481186*X^1 [r^2 = 2.2254968506229528E-5] [residue = 1.0048575698963698]
Statistics for Symbol = AZ	Statistics for Symbol = GOOG

Y = 0.024738184245307 18*X^0 + -0.8957856099765871*X^1 [r^2 = 0.5303400841042966] [residue = 0.9932963013229478]	Y = -0.023432978362554265*X^0 + -0.3702407799997411*X^1 [r^2 = 0.10570469701843901] [residue = 1.004627345678593]
Statistics for Symbol = AZN Y = 0.005880388064968891*X^0 + -0.12246355413404778*X^1 [r^2 = 0.02768254932056927] [residue = 0.9976488667751887]	Statistics for Symbol = GS Y = -0.0015614207280170 83*X^0 + -0.39048827512240936*X^1 [r^2 = 0.2689682571519976] [residue = 1.003098358537627]
Statistics for Symbol = BAC Y = -0.020831316776144453*X^0 + -0.8626850039670807*X^1 [r^2 = 0.6993191340708272] [residue = 0.9994338491736612]	Statistics for Symbol = GSK Y = 0.0012047423888962 75*X^0 + -0.49976088762316434*X^1 [r^2 = 0.11434523434974272] [residue = 1.005220775436628 7]
Statistics for Symbol = BCS Y = -0.026518249352869176*X^0 + 0.1618921685659959 5*X^1 [r^2 = 0.029646394769078873] [residue = 1.0013502041343116]	Statistics for Symbol = HBC Y = 0.006613632115260977*X^0 + 0.4834971950281958*X^1 [r^2 = 0.41191234088722434] [residue = 1.003240932489706]
Statistics for Symbol = BP Y = -0.008222199492962318*X^0 + -0.661903632182613*X^1 [r^2 = 0.3947129736510434] [residue = 1.0019530549816849]	Statistics for Symbol = HPQ Y = -0.014219307772506212*X^0 + -0.23792664682564643*X^1 [r^2 = 0.05894346890396082] [residue = 1.0033366066188902]
Statistics for Symbol = BRK-A Y = -0.013644302678795319*X^0 + -0.004156224198481908*X^1 [r^2 = 1.166112690246142E-5] [residue = 1.0037912649728225]	Statistics for Symbol = IMO Y = -0.006405979821814 96*X^0 + -0.7635185577904207*X^1 [r^2 = 0.45571429006047365] [residue = 1.0029902081513735]
Statistics for Symbol = C	Statistics for Symbol = INTC

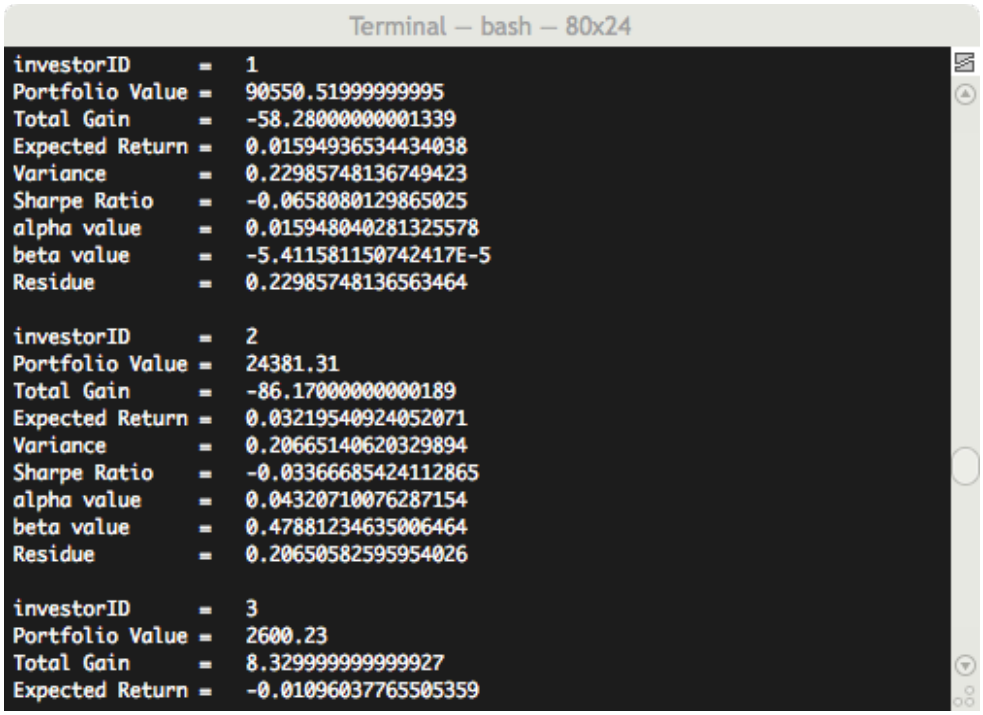
Y = 0.01606460240813701*X^0 + -0.34900669245733823*X^1 [r^2 = 0.1030111763612623 1] [residue = 1.0039085145118494]	Y = -0.025253592661503795*X^0 + 0.07135403043658345*X^1 [r^2 = 0.003056954243903724] [residue = 1.0046473941796081]
Statistics for Symbol = JNJ Y = 0.011112375713548553*X^0 + 0.06763389984925398*X^1 [r^2 = 0.005651130390075637] [residue = 1.0038495810874013]	Statistics for Symbol = JPM Y = 0.03350660691481476*X^0 + 0.11149235939222771*X^1 [r^2 = 0.006762393348985433] [residue = 1.0053883245901394]
Statistics for Symbol = LFC Y = -0.03934681452957427*X^0 + 0.218804650673966*X^1 [r^2 = 0.045196108324379536] [residue = 1.0046102579293863]	Statistics for Symbol = MOT Y = -0.020043168973000673*X^0 + 0.5319892513530128*X^1 [r^2 = 0.2686095296760603] [residue = 1.0037660393211651]
Statistics for Symbol = MRK Y = -0.02560645056941166*X^0 + -1.2804672567897453*X^1 [r^2 = 0.6523868338856713] [residue = 1.0035410096178785]	Statistics for Symbol = MSFT Y = 0.02175506167282722*X^0 +0.18614030885060254*X^1 [r^2 = 0.017084288919266666] [residue = 1.0055525497393845]
Statistics for Symbol = NOC Y = -0.006884542250660682*X^0 + -0.2643911170822473*X^1 [r^2 = 0.12908690436132034] [residue = 1.0036679677918725]	Statistics for Symbol = NOK Y = 0.014107379439503225*X^0 + -0.3169472434435743*X^1 [r^2 = 0.0599833752322308] [residue = 1.0048213277577218]
Statistics for Symbol = NVS Y = -0.014942304652452372*X^0 + 0.03828044901413923*X^1 [r^2 = 0.0030187343750072227]	Statistics for Symbol = ORCL Y = 0.009380509197227 33*X^0 + -1.021643815210312*X^1 [r^2 = 0.494463374802358 5]

[residue = 1.0033270829414205]	[residue = 1.0036896091979506]
Statistics for Symbol = OXY Y = -0.036875344343975915*X^0 + 0.3124806649205539*X^1 [r^2 = 0.06806814393701083] [residue = 1.0043819144322494]	Statistics for Symbol = PFE Y = -0.014517651992897007*X^0 + 0.9833909468919476*X^1 [r^2 = 0.5363672771048464] [residue = 1.0037762457727983]
Statistics for Symbol = SNY Y = -0.0043493753723333195*X^0 + -0.44774158942609066*X^1 [r^2 = 0.12068178065466902] [residue = 1.0046598750932578]	Statistics for Symbol = T Y = -0.04221581205441499*X^0 + -0.044719455431109736*X^1 [r^2 = 0.0018031804990814879] [residue = 1.0044290633437933]
Statistics for Symbol = TXN Y = -0.027568272723382893*X^0 + 0.13628902869089174*X^1 [r^2 = 0.014650290815183766] [residue = 1.0045839268334988]	Statistics for Symbol = VOD Y = -0.013253290113428929*X^0 + 0.52237624904956*X^1 [r^2 = 0.5076272669390988] [residue = 1.002792272075047]
Statistics for Symbol = VZ Y = 0.002587534300670883*X^0 + 0.10092476801348907*X^1 [r^2 = 0.007896837911641391] [residue = 1.0038265814063725]	Statistics for Symbol = WB Y = -0.011321125502099265*X^0 + -0.18640872320574361*X^1 [r^2 = 0.019692725161162824] [residue = 1.0051949975132506]
Statistics for Symbol = WMT Y = -0.020203197114290018*X^0 + 0.5561815233044046*X^1 [r^2 = 0.2086149184752754] [residue = 1.0048213449476813]	Statistics for Symbol = WYE Y = 0.012222811510098483*X^0 + -0.15860729445970903*X^1 [r^2 = 0.06107748116048412] [residue = 1.0031424304543026]
Statistics for Symbol = XOM Y = -0.019766835385171905*X^0 + -0.02177471862914535*X^1	

[r^2 = 3.2809077353146E-4] [residue = 1.0043139933597007]	

Use Case 2 - Portfolio's Performances (By Zong-Zhi 'Alex' Lin)

This table summarizes the performance and risk position of each investor's portfolio, which is derived from the information on stocks obtained in Use Case 2.

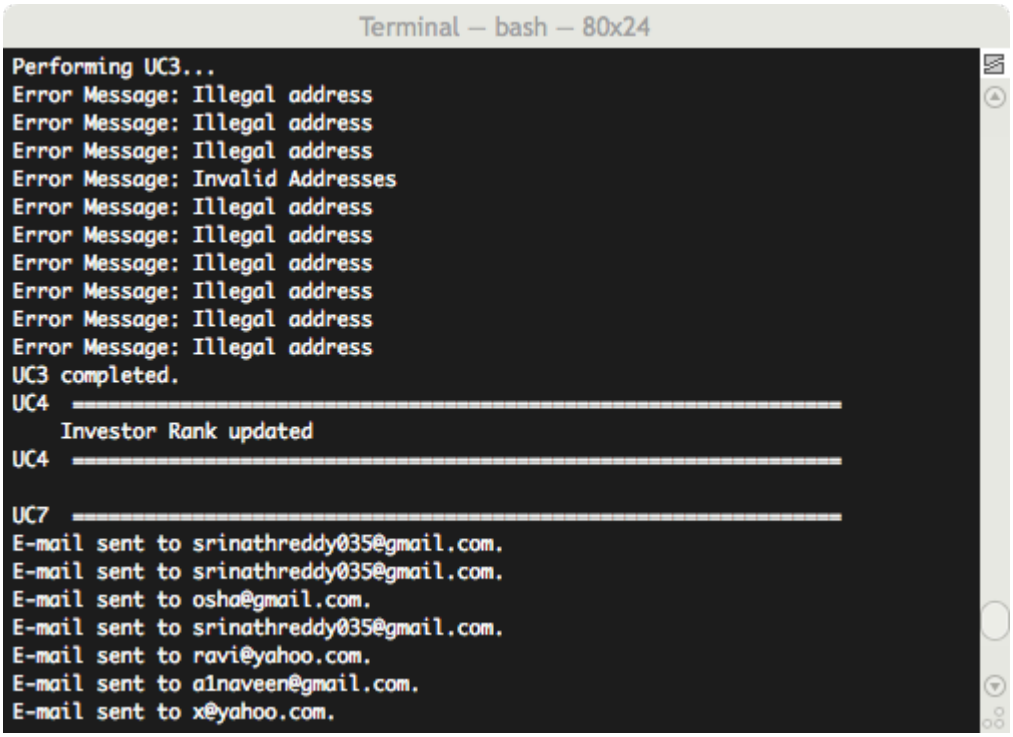


investorID = 1 Portfolio Value = 90550.51999999995 Total Gain = -58.28000000001339 Expected Return = 0.01594936534434038 Variance = 0.22985748136749423 Sharpe Ratio = -0.0658080129865025 alpha value = 0.015948040281325578 beta value = -5.411581150742417E-5 Residue = 0.22985748136563464	investorID = 2 Portfolio Value = 24381.31 Total Gain = -86.17000000000189 Expected Return = 0.03219540924052071 Variance = 0.20665140620329894 Sharpe Ratio = -0.03366685424112865 alpha value = 0.04320710076287154 beta value = 0.47881234635006464 Residue = 0.20650582595954026
investorID = 3	investorID = 4

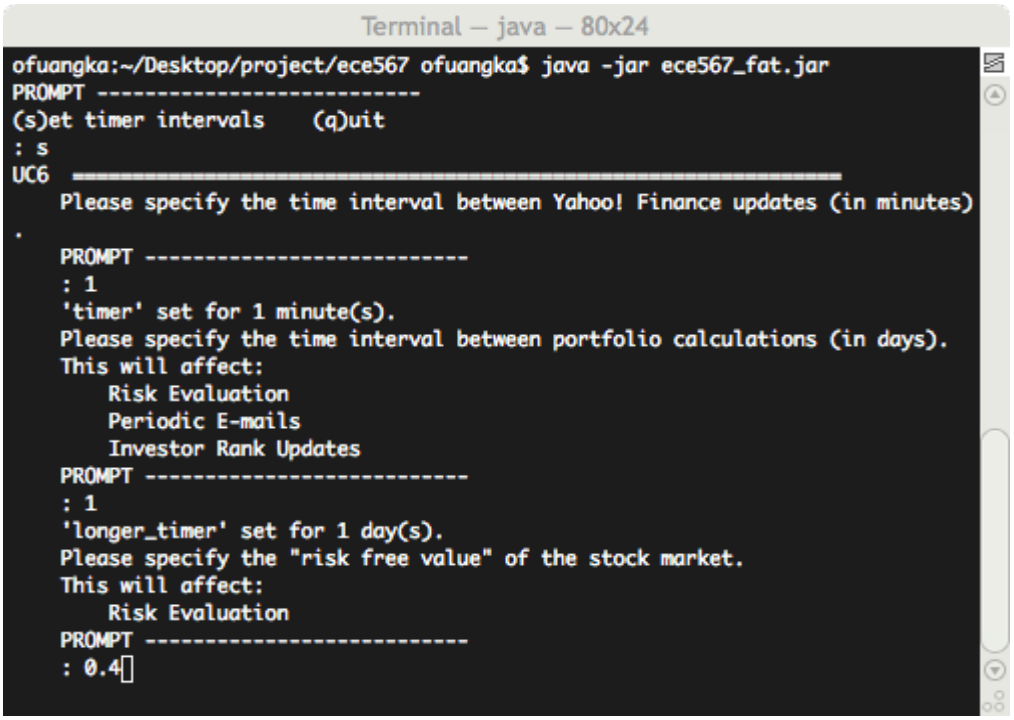
Portfolio Value = 2600.23 Total Gain = 8.329999999999927 Expected Return = -0.01096037765505359 Variance = 0.6072042735385016 Sharpe Ratio = -0.07502296203388843 alpha value = -0.020594881317831253 beta value = -0.41895841956152713 Residue = 0.6070928149325749	Portfolio Value = 4386.49 Total Gain = -62.710000000000036 Expected Return = -0.013123100959993068 Variance = 0.8236474743623753 Sharpe Ratio = -0.06679861560762161 alpha value = -0.02177601163670837 beta value = -0.3762803318529635 Residue = 0.8235575670949324
investorID = 5 Portfolio Value = 2316.1699999999996 Total Gain = -76.43000000000029 Expected Return = -0.01359705945159466 Variance = 0.7474535883123996 Sharpe Ratio = -0.0706688774817628 alpha value = -0.023730959555552694 beta value = -0.4406771155119911 Residue = 0.7473302742066917	investorID = 63100 Portfolio Value = 607.2 Total Gain = 0.0 Expected Return = 0.012698 Variance = 1.0053793729194604 Sharpe Ratio = -0.034708769504864095 alpha value = 0.00120474238889627 beta value = -0.499760887623164 Residue = 1.00522077543663
investorID = 90525 Portfolio Value = 1147.7 Total Gain = 0.0 Expected Return = 0.004237996166245534 Variance = 0.6139990619897668 Sharpe Ratio = -0.05521063950845179 alpha value = 0.005191245071585379 beta value = 0.04146032853023954 Residue = 0.6139979704558378	investorID = 97527 Portfolio Value = 14342.94 Total Gain = 0.0 Expected Return = -0.01123970353219075 Variance = 0.8539712868615617 Sharpe Ratio = -0.06356383855642077 alpha value = -0.018756728570933853 beta value = -0.32688868666293786 Residue = 0.8539034334728159
investorID = 99675 Portfolio Value = 3909.0800000000004 Total Gain = 0.0 Expected Return = 0.039983 Variance = 0.2640475559462775	

Sharpe Ratio = -0.014628623895417336 alpha value = 0.054148567907223 beta value = 0.615955328409363 Residue = 0.263806637421829	

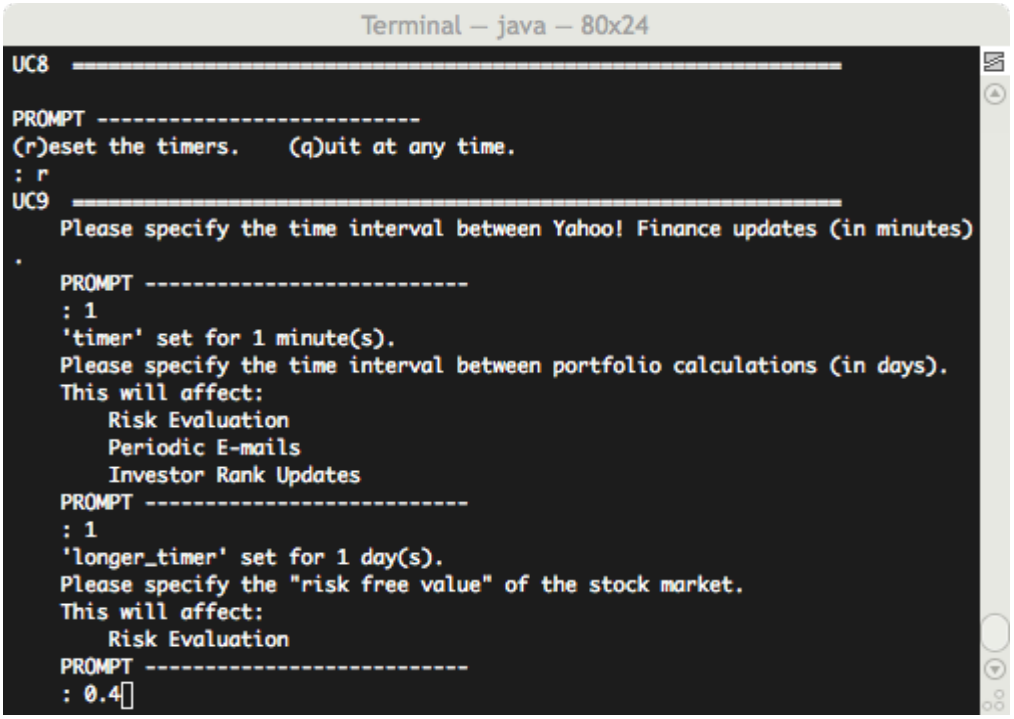
Use Case 3 (By Srinivas Mudireddy), 4 (By John Paul Varkey) and 7 (By Osha Fuangkasae)



Use Case 6 (By Osha Fuangkasae)

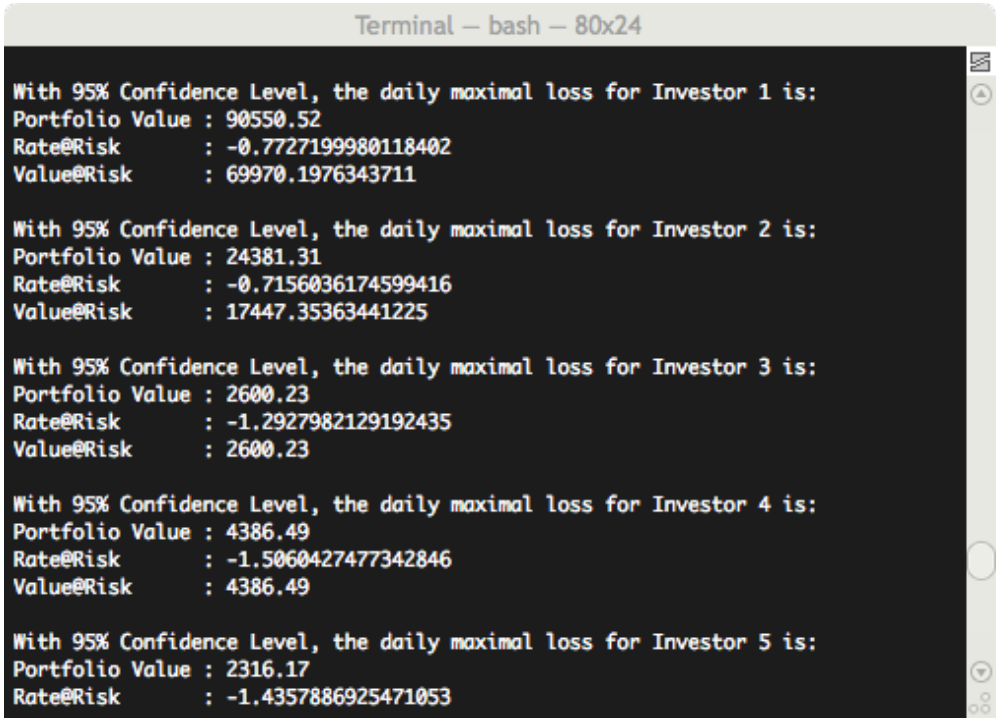


Use Case 9 (By Osha Fuangkasae)



Use Case 11 – Value at Risk for Portfolios (By Zong-Zhi 'Alex' Lin)

This table summarizes the value at risk information of each investor's portfolio, which is using from the information on stocks obtained in Use Case 2 and formula .



With 95% Confidence Level, the daily maximal loss for Investor 1 is: Portfolio Value : 90550.52 Rate@Risk : -0.7727199980118402 Value@Risk : 69970.1976343711
With 95% Confidence Level, the daily maximal loss for Investor 2 is: Portfolio Value : 24381.31 Rate@Risk : -0.7156036174599416 Value@Risk : 17447.35363441225
With 95% Confidence Level, the daily maximal loss for Investor 3 is: Portfolio Value : 2600.23 Rate@Risk : -1.2927982129192435 Value@Risk : 2600.23

With 95% Confidence Level, the daily maximal loss for Investor 4 is: Portfolio Value : 4386.49 Rate@Risk : -1.5060427477342846 Value@Risk : 4386.49
With 95% Confidence Level, the daily maximal loss for Investor 5 is: Portfolio Value : 2316.17 Rate@Risk : -1.4357886925471053 Value@Risk : 2316.17
With 95% Confidence Level, the daily maximal loss for Investor 90525 is: Portfolio Value : 1147.7 Rate@Risk : -1.2847525523218548 Value@Risk : 1147.7
With 95% Confidence Level, the daily maximal loss for Investor 97527 is: Portfolio Value : 14342.94 Rate@Risk : -1.531393569635318 Value@Risk : 14342.94
With 95% Confidence Level, the daily maximal loss for Investor 63100 is: Portfolio Value : 607.2 Rate@Risk : -1.6367202939676035 Value@Risk : 607.2
With 95% Confidence Level, the daily maximal loss for Investor 99675 is: Portfolio Value : 3909.08 Rate@Risk : -0.8053101380296424 Value@Risk : 3148.021754368914

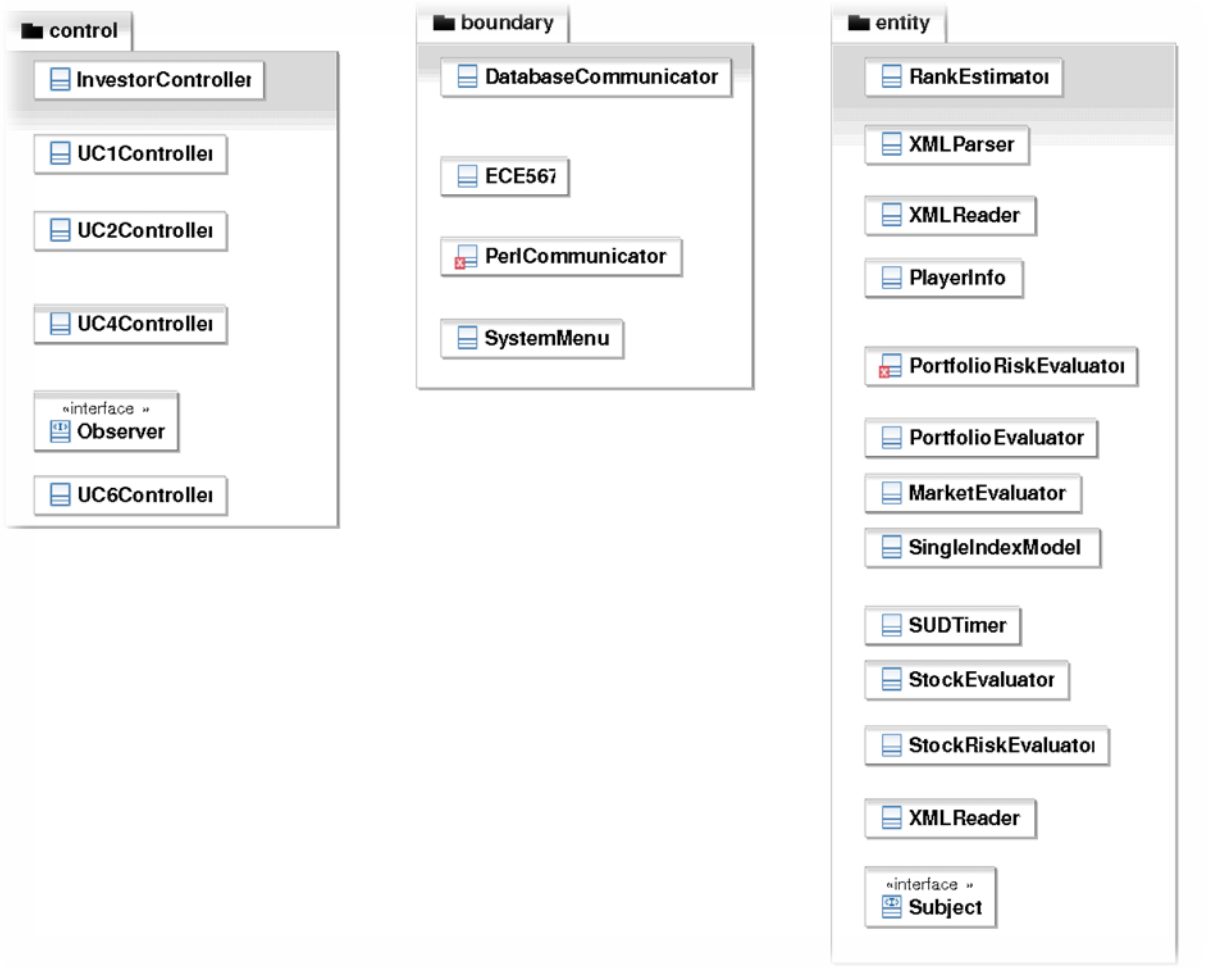
System Architecture and System Design

Architectural Styles - Describe the architectural styles used in your design

We are using **Component-based software engineering**. This architecture is described on Wikipedia as *"...a branch of the software engineering discipline, with emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components. Components are considered to be a higher level of abstraction than objects and as such they do not share state and communicate by exchanging messages carrying data"* (ref: Wikipedia). This architecture fits our style of programming, which is an attempt at "top down programming" where our major task is split into smaller modules to facilitate testability, extensibility, and maintenance.

Identifying Subsystems

Figure 30: Package diagram



Mapping Subsystems to Hardware

Does your system need to run on multiple computers? For example, you may have client (web browser) and server (web server) processes, running on different machines.

YES: Although our system was developed on a single computer, the Perl and Java components can be split across different computers. Because we did not anticipate this functionality, we chose not to allow the user to specify a URL for the Perl script. We considered creating as extensible and maintainable a system as possible, but made the decision to focus instead on correctness and prompt release. However, adding user defined URL's is relatively easy, and we may do so if we have time after project completion. In other words, our program is currently hard coded to work on a single machine due to time constraints.

Persistent Data Storage

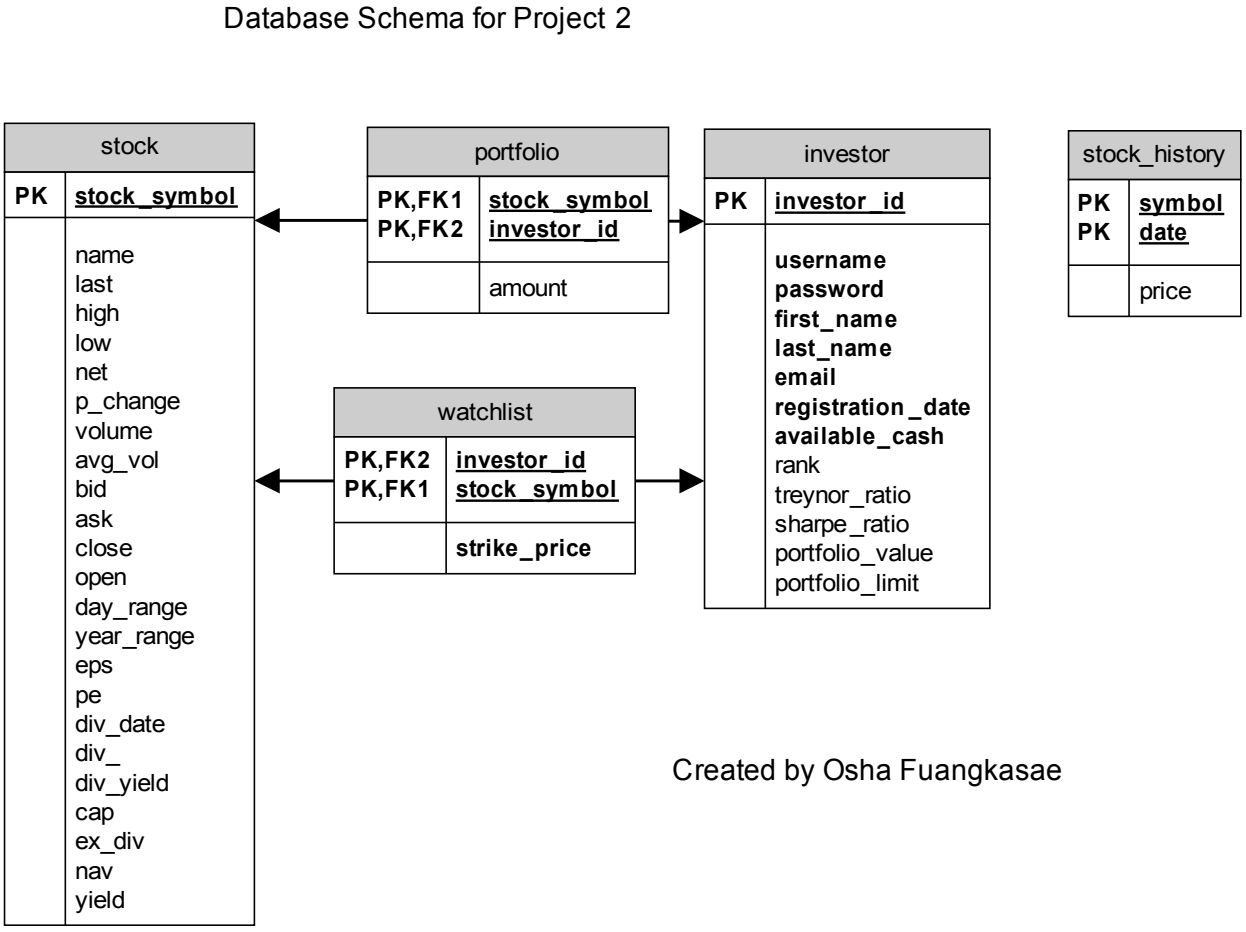
Does your system need to save data that need to outlive a single execution of the system?

YES: We require both a persistent database and files indicating which stocks and stock attributes to monitor.

Identify the persistent objects and select the storage management strategy, e.g., flat files, relational database, etc.

Again, our stock list and stock attribute files are in XML format. The following relational database schema outlines what Project 2 outputs and needs.

Figure 31: Database Schema



Network Protocol

Which communication protocol do you use, e.g., plain Java sockets, Java RMI, Java JDBC, HTTP, etc? Explain why you made your specific choice.

We are using XMLRPC to communicate to Perl from Java. We had the choice of using XMLRPC or SOAP. XMLRPC was chosen since documentation online outlines how to use this protocol to call Perl from Java (ref: Call Perl routines from Java). Furthermore, although SOAP is much more powerful, it is much more complex to use. We also use Java JDBC to communicate to the MySQL database. Again, this is due to the wealth of documentation online for JDBC (ref: The Java Tutorials).

Global Control Flow

Execution Order: Is your system procedure-driven and executes in a "linear" fashion, where every user every time has to go through the same steps, or is it an event-driven system that waits in a loop for events, and every user can generate the actions in a different order?

Our system is event driven. We wait in a loop for user input. We also perform actions whenever the Timer runs out.

Time dependency: Is there any timers in your system?

Yes. There is a main Timer (called SUDTimer) that all use case controllers observe.

Is your system of event-response type, with no concern for real time, or is it a real-time system? If it's real-time, is it periodic, and what are the time constraints for each period?

Our SUD is both event-response and periodic. We wait for input from the Server Administrator, which can come at any time. We also perform periodic updates to the Stock Database. The period of these updates is specified by the Server Administrator when she first starts the program.

Concurrency: Does your system use multiple threads?

YES. We use two threads of execution.

Identify the objects that have separate threads of control and describe any synchronization between the threads.

Our Timer runs in its own thread. We do not utilize any synchronization (mutexes or semaphores) between the main thread and the Timer thread since they are meant to run at the same time and do not share resources.

Hardware Requirements

What system resources your system depend upon? Examples are, screen display, disk storage, communication network, or you may be accessing some special sensor/instrument.

Describe exact requirements for these resources in order for your system to run. For example, you need color display, with minimum resolution of 640 × 480 pixels; minimum of 2 Gbytes hard disk space; minimum network bandwidth 56 Kbps.

Our SUD requires a fast Internet connection, and requires a server that performs well with Perl, MySQL, Java, and Apache HTTP Server with CGI scripts installed. The SUD also requires software components—the following files that we provide must be placed in the “cgi-bin” folder of the Apache installation:"server.pl", "PerlFunctions.pm", and "PerlSystemFacade.pm" Perl scripts. Furthermore, the directory "xml/" containing the files "stock_attributes.xml" and "stock_list.xml" also be located there. The Perl libraries "Finance::Quote", "XMLRPC::Lite", and "XML::Simple" must be installed. The MySQL database ece567 must exist on the server, with update and insert privileges granted to "webuser"@'%' identified by "webpass". The relational table "stock" must be defined, with the following columns:

1. symbol varchar(5)
2. name varchar(40)
3. last varchar(20)

4. high varchar(20)
5. low varchar(20)
6. net varchar(20)
7. p_change varchar(20)
8. volume varchar(20)
9. avg_vol varchar(20)
10. bid varchar(20)
11. ask varchar(20)
12. close varchar(20)
13. open varchar(20)
14. day_range varchar(40)
15. year_range varchar(40)
16. eps varchar(20)
17. pe varchar(20)
18. div_date varchar(20)
19. div_ varchar(20)
20. div_yield varchar(20)
21. cap varchar(20)
22. ex_div varchar(20)
23. nav varchar(20)
24. yield varchar(20)

Algorithms and Data Structures

Algorithms

We use complex algorithms to determine the risk value of each stock, each individual Investor Portfolio, and to identify best investment strategy.

Algorithm to evaluate investor's portfolio's risk and Sharpe-ratio

We use Single Index Mode to derive the expected return and variance of each investor's portfolio. The results of portfolio's expected return and variance are then used to calculate Sharpe Ratio of each investor's portfolio. We start our discussion by starting with traditional approach to serve as an introduction of this subject.

Traditional Approach

A portfolio's performance, i.e., portfolio's expected return, is the result of the performance of its components. Thus, the return realized on a portfolio is a linear combination of the returns on the individual investments. Mathematically, the expected return of a portfolio is a weighted average of the expected returns of the components, where x_i = proportion of portfolio invested in security i or stock i, and R_i = the returns realized on the individual investment of security i or stock i:

Figure 32: Traditional expected return

$$E(R_p) = \sum_{i=1}^{n} \left[x_i E(R_i) \right] \quad \text{and} \quad \sum_{i=1}^{n} x_i = 1$$

The variance of the portfolio is not a linear combination of component variances. For an n-security portfolio, the portfolio variance is defined as follows. Note that the equation above includes the correlation coefficient (or covariance) between all pairs of securities in the portfolio. A covariance matrix is a tabular presentation of the pairwise combinations of all portfolio components. The required number of covariances to compute a portfolio variance is $(n^2 - n)/2$. This portfolio construction technique using the full covariance matrix is called a Markowitz model.

Figure 33: Traditional variance

$$\sigma_p^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \rho_{ij} \sigma_i \sigma_j$$

where x_i = proportion of total investment in Security i

ρ_{ij} = correlation coefficient between Security i and Security j

Single-Index Model (SIM)

The Single-Index Model (SIM) compares all securities to a single benchmark, which is an alternative to comparing a security to each of the others. The basic idea is that by observing how two independent securities behave relative to the common macroeconomic factor of a market value, we learn something about how the securities are likely to behave relative to each other. Note that the single index model drastically reduces the number of computations needed to determine portfolio variance.

We are assuming that returns on a security/stock come from 2 sources: Common macro-economic factor and firm specific events. For example, Dow Jones Industrial Average is often considered as a valid proxy for the common macro-economic factor. Mathematically, the expected return of a security is defined as follows:

Figure 34: SIM expected return

$$R_i = E(r_i) + \beta_i\, m + e_i$$

- R_i = Return for security i
- β_i = Factor sensitivity or factor loading or factor β
- m = Surprise in macro-economic factor
- e_i = Firm specific events

The expected return of a portfolio is a weighted average (w_i's) of the expected returns of the components:

From the formula in 3., it is easy to see that the variance of a security i can be derived as follows depending on its degree of sensitivity to macro-economic factor:

Figure 35: SIM variance

$$\sigma_i^2 = \beta_i^2\, \sigma_m^2 + \sigma^2(\mathbf{e}_i) \sim \text{diagonal elements}$$

- σ_i^2 = total variance of security i
- $\beta_i^2\, \sigma_m^2$ = systematic variance
- $\sigma^2(\mathbf{e}_i)$ = unsystematic variance

The covariance between a pair of security r_i and security r_j in the portfolio is:

Figure 36: Covariance

$$Cov(r_i, r_j) = Cov(\beta_i m+e_i, \beta_i m+e_j) = \beta_i\beta_j\sigma_m^2$$

The correlation coefficient between a pair of security r_i and security r_j in the portfolio equals the product of correlations with the macro-economic (or market) index:

Figure 37: Correlation coefficient

$$\begin{aligned} Corr(r_i, r_j) &= \beta_i\beta_j\sigma_m^2 / \sigma_i\sigma_j \\ &= [\beta_i \sigma_m^2 \beta_j\sigma_m^2] / [\sigma_i \sigma_m \sigma_j \sigma_m] \\ &= Corr(r_i, r_m)\, Corr(r_i, r_m) \end{aligned}$$

The total risk/variance of a portfolio comes from two components: Market or systematic risk (risk related to the macroeconomic factor or market index) and unsystematic or firm specific risk (risk not related to the macro factor or market index).

Figure 38: Total risk/variance

$$\sigma_P^2 = \beta_P^2 \sigma_m^2 + \sigma^2(e_P)$$

$$\sigma_P = \sqrt{\beta_P^2\sigma_m^2 + \sigma^2(e_P)} = \sqrt{\sigma_m^2\left(\sum_{i=1}^{N} w_i\beta_i\right)^2 + \sum_{i=1}^{N} w_i^2\sigma^2(e_i)}$$

Calculation of Sharpe-Ratio:

Figure 39: Sharpe-Ratio

$$S_P = \frac{E(R_P) - r_f}{\sigma_P}$$

where r_f is risk free interest rate.

Written by Zong-Zhi Lin

Algorithm to evaluate investor's portfolio's value at risk

The most popular and traditional measure of risk is ***volatility***. The main problem with volatility, however, is that it does not care about the direction of an investment's movement: a stock can be volatile because it suddenly jumps higher. For investors, risk is about the odds of losing ***money***, and Value-at-Risk (VaR) is based on that common-sense fact. By assuming investors care about the odds of a really big loss, VaR answers the question, "What is my worst-case scenario?" or "How much could I lose in a really bad period ?"

Value-at-Risk, VaR, is a risk measure that enables investors to determine how much the cash value of a portfolio could decline over a defined time horizon with a given probability as a result of adverse changes in market conditions.

A VaR statistic has three components: a time period, a confidence level and a loss amount (or loss percentage). Keep these three parts in mind as we give some examples of variations of the question that VaR answers: What is the most I can - with a 95% or 99% level of confidence - expect to lose in dollars over the next period of time?

Methods of Calculating VaR - The Variance-Covariance Method

This method assumes that portfolio's returns are normally distributed. In other words, it requires that we estimate only two factors - an ***expected*** (or average) return and a ***standard deviation*** - which allow us to plot a normal distribution curve.

The idea behind the variance-covariance is to take advantage of the normal curve – i.e., we automatically know where the worst 5% and 1% lie on the curve. They are a function of our desired confidence and the standard deviation (σ):

Confidence	# of Standard Deviations (σ)
95% (high)	$-1.65 \times \sigma$
99% (really high)	$-2.33 \times \sigma$

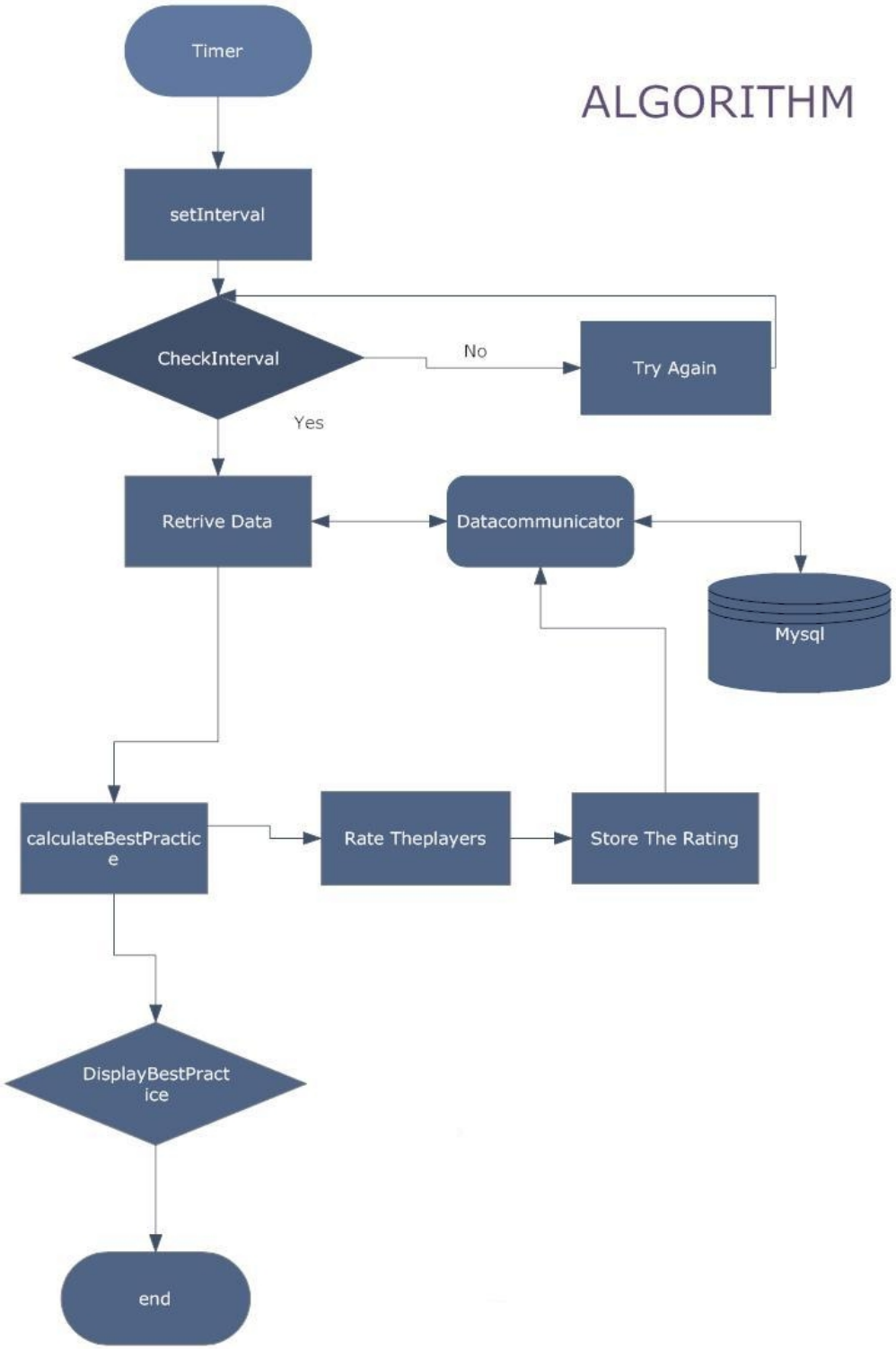
In this project, 95% level of confidence is used to calculate how much could an investor lose in a really bad period.

Summarized by Zong-Zhi Lin

Sources: Value at Risk - http://en.wikipedia.org/wiki/Value_at_risk, Introduction to Value at Risk (VAR) – Part 1, http://www.investopedia.com/articles/04/092904.asp

Algorithm for Best Strategies

Figure 40: Algorithm for best strategies (Created by Ravi Gudur)



Data Structures

Does your system use any complex data structures, such as arrays, linked lists, hash tables, or trees?

YES. We use XML to communicate updated stock information between Perl and Java. We also use `Vector`'s and `ArrayList`'s.

What criteria you used in deciding what data structure to use, e.g., performance vs. flexibility?

We chose XML because of its flexibility. We knew that we would be communicating across languages, and XML has major support in most major languages. Furthermore, node access is logical and easy to comprehend, and we have experience using XML when creating web pages in HTML.

We used `Vector`s in passing parameters in XMLRPC for no particular reason. Our Perl functions do not take parameters, so this was an arbitrary choice, made because it was what the example that we looked at chose (ref: Call Perl routines from Java).

We chose `ArrayList`'s for our Timer Observers, our stockComputedList and as a medium to be used in sorting Investors for our ranking system. `ArrayList`'s provide ease of access and flexibility because they can contain any Object. Because we focused on correctness Vs performance, we chose not to implement our own customized List.

History of Work

The Gantt charts documents how the actual milestones and deadlines evolved, both planning and scheduling our project, and the distribution of responsibilities among team members. They are attached here:

Figure 41: Lists the projected milestones and dates by which we plan to accomplish – Part I

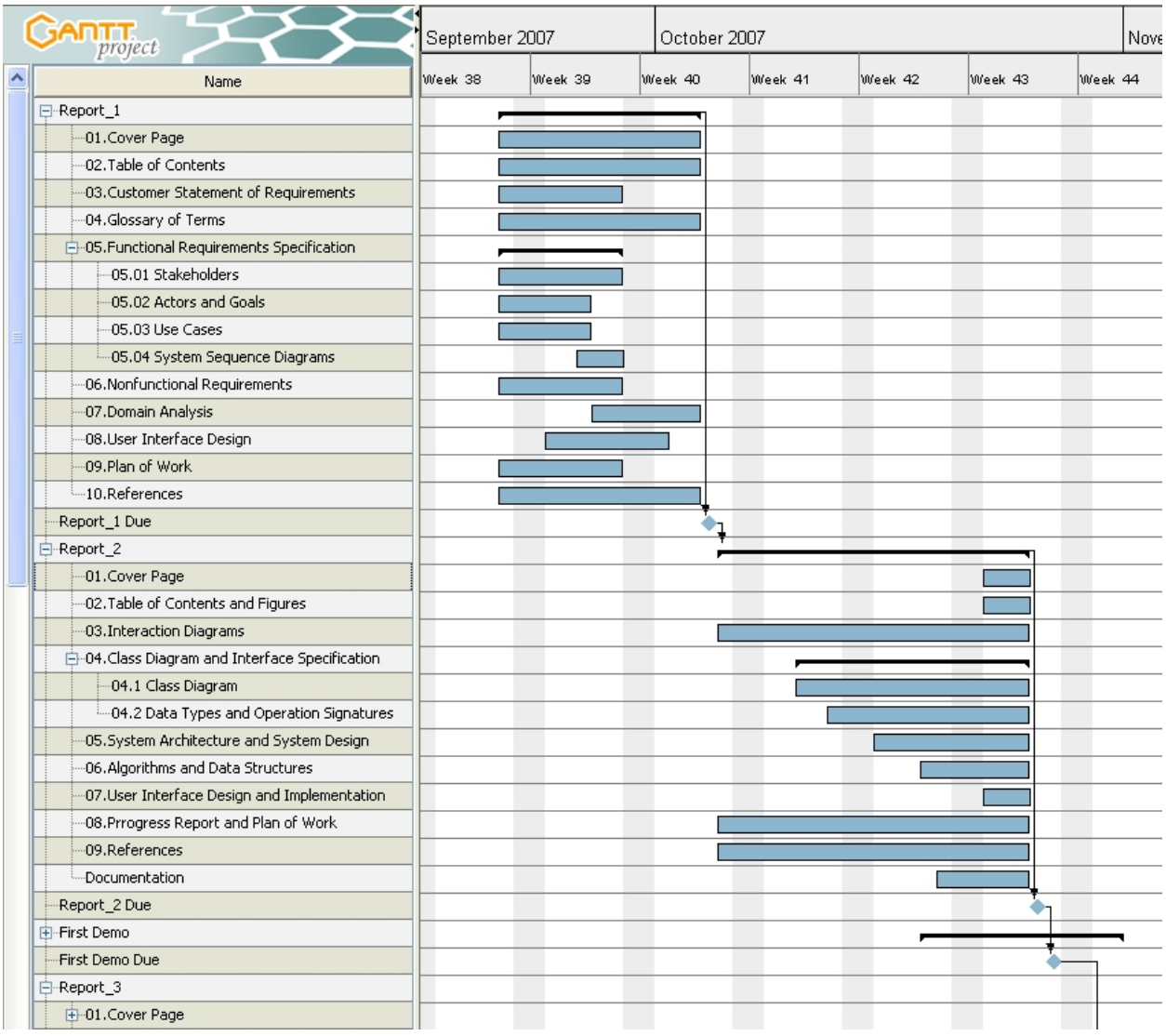
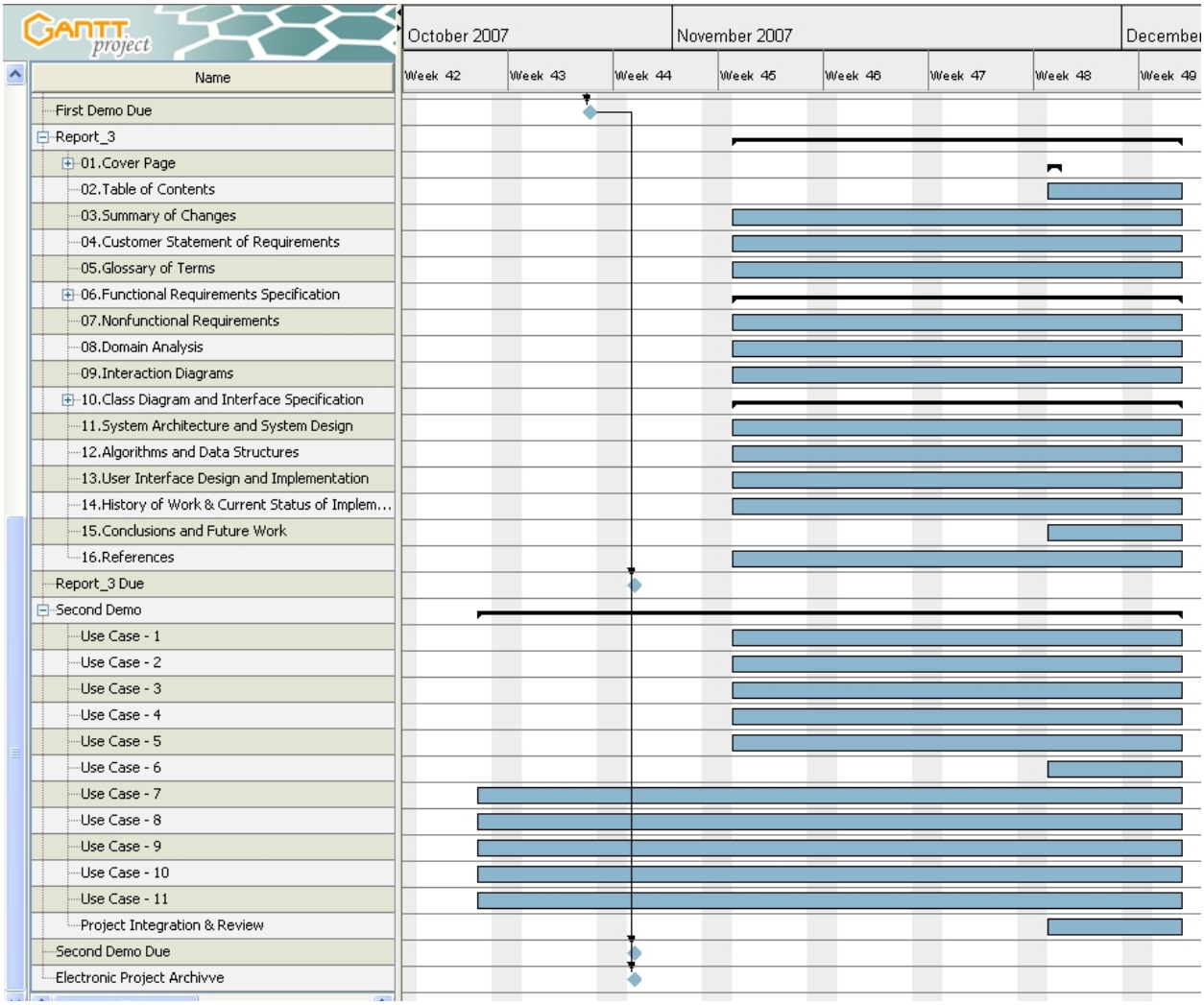
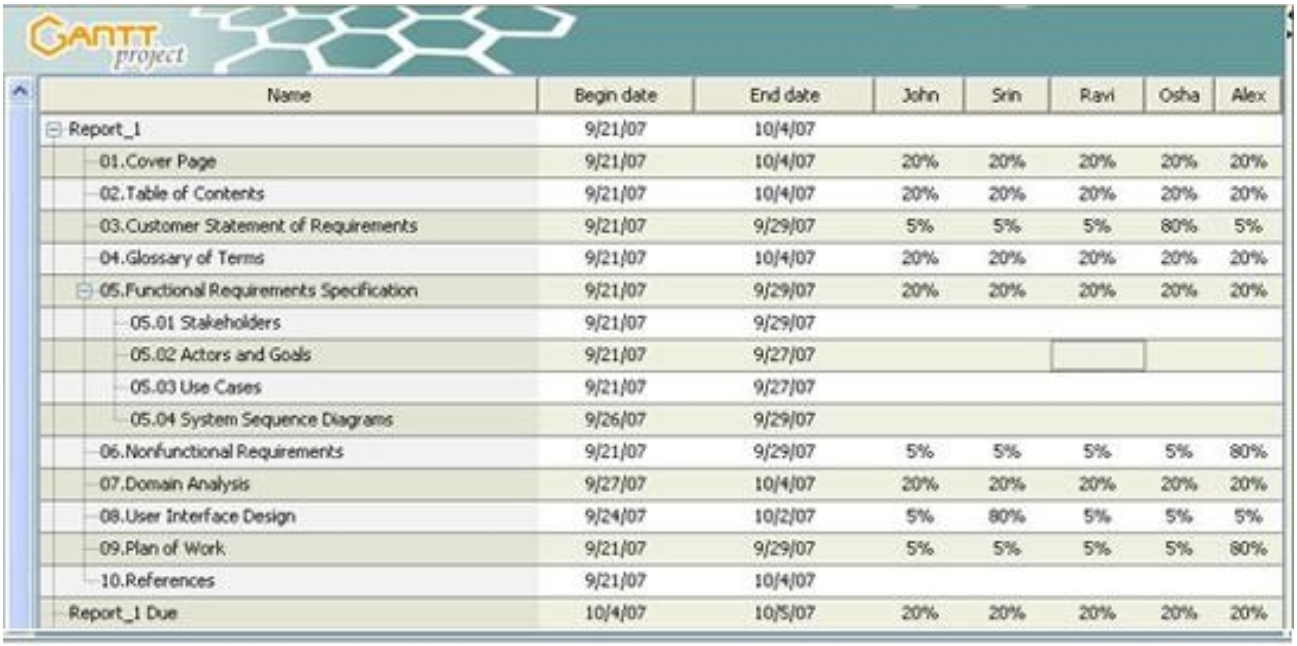


Figure 42: Lists the projected milestones and dates by which we plan to accomplish – Part II



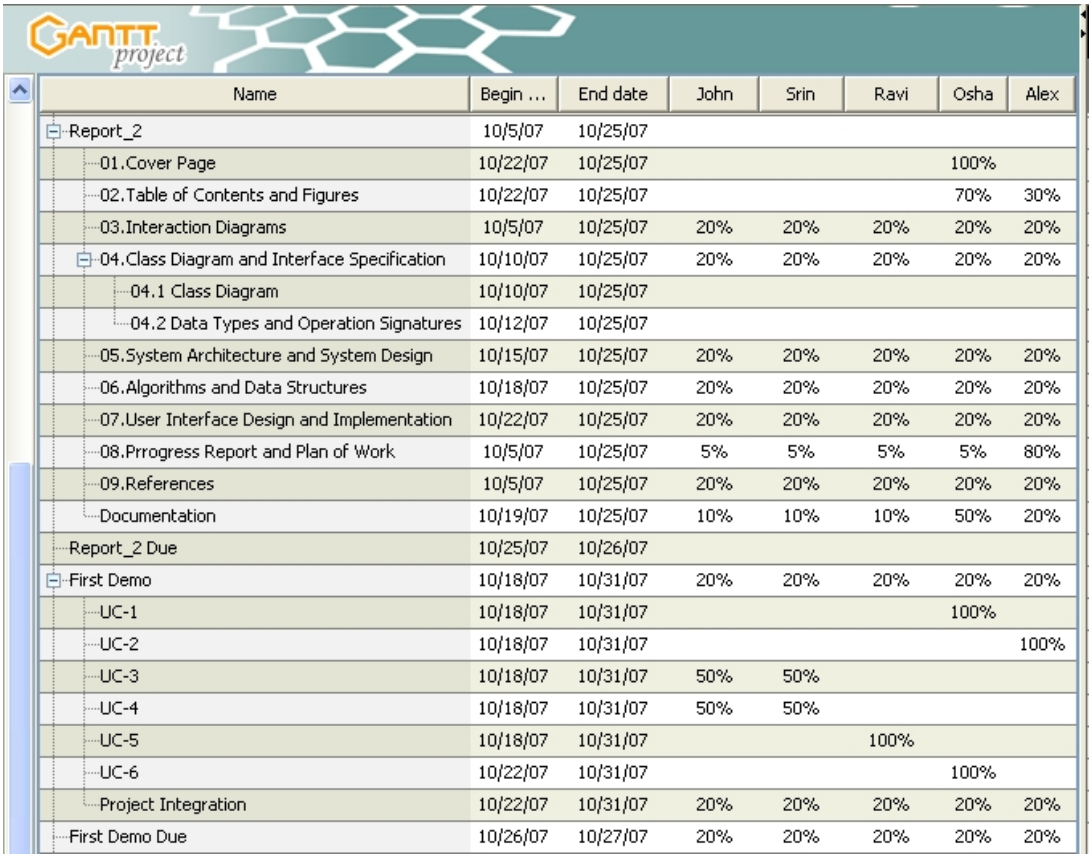
Breakdown of Responsibilities

Figure 43: Provides the breakdown of responsibilities for Report 1 of Project 2



Name	Begin date	End date	John	Srin	Ravi	Osha	Alex
Report_1	9/21/07	10/4/07					
01.Cover Page	9/21/07	10/4/07	20%	20%	20%	20%	20%
02.Table of Contents	9/21/07	10/4/07	20%	20%	20%	20%	20%
03.Customer Statement of Requirements	9/21/07	9/29/07	5%	5%	5%	80%	5%
04.Glossary of Terms	9/21/07	10/4/07	20%	20%	20%	20%	20%
05.Functional Requirements Specification	9/21/07	9/29/07	20%	20%	20%	20%	20%
05.01 Stakeholders	9/21/07	9/29/07					
05.02 Actors and Goals	9/21/07	9/27/07					
05.03 Use Cases	9/21/07	9/27/07					
05.04 System Sequence Diagrams	9/26/07	9/29/07					
06.Nonfunctional Requirements	9/21/07	9/29/07	5%	5%	5%	5%	80%
07.Domain Analysis	9/27/07	10/4/07	20%	20%	20%	20%	20%
08.User Interface Design	9/24/07	10/2/07	5%	80%	5%	5%	5%
09.Plan of Work	9/21/07	9/29/07	5%	5%	5%	5%	80%
10.References	9/21/07	10/4/07					
Report_1 Due	10/4/07	10/5/07	20%	20%	20%	20%	20%

Figure 44: Provides the breakdown of responsibilities for Report 2 of Project 2



Name	Begin ...	End date	John	Srin	Ravi	Osha	Alex
Report_2	10/5/07	10/25/07					
01.Cover Page	10/22/07	10/25/07				100%	
02.Table of Contents and Figures	10/22/07	10/25/07				70%	30%
03.Interaction Diagrams	10/5/07	10/25/07	20%	20%	20%	20%	20%
04.Class Diagram and Interface Specification	10/10/07	10/25/07	20%	20%	20%	20%	20%
04.1 Class Diagram	10/10/07	10/25/07					
04.2 Data Types and Operation Signatures	10/12/07	10/25/07					
05.System Architecture and System Design	10/15/07	10/25/07	20%	20%	20%	20%	20%
06.Algorithms and Data Structures	10/18/07	10/25/07	20%	20%	20%	20%	20%
07.User Interface Design and Implementation	10/22/07	10/25/07	20%	20%	20%	20%	20%
08.Prrogress Report and Plan of Work	10/5/07	10/25/07	5%	5%	5%	5%	80%
09.References	10/5/07	10/25/07	20%	20%	20%	20%	20%
Documentation	10/19/07	10/25/07	10%	10%	10%	50%	20%
Report_2 Due	10/25/07	10/26/07					
First Demo	10/18/07	10/31/07	20%	20%	20%	20%	20%
UC-1	10/18/07	10/31/07				100%	
UC-2	10/18/07	10/31/07					100%
UC-3	10/18/07	10/31/07	50%	50%			
UC-4	10/18/07	10/31/07	50%	50%			
UC-5	10/18/07	10/31/07			100%		
UC-6	10/22/07	10/31/07				100%	
Project Integration	10/22/07	10/31/07	20%	20%	20%	20%	20%
First Demo Due	10/26/07	10/27/07	20%	20%	20%	20%	20%

Conclusions and Future Work

Conclusions

In developing Use Case 2 and 11, the first technical challenge I encountered is that the calculation of Sharpe ratio and Value@Risk metrics involves several mathematical formulas, models and their concomitant data sets. I was not so clear at the beginning the project on how to lay the calculation out in a nice and neat style. The incrementally progressing steps of software engineering development style have helped me to build these use cases from an concept entity to a realized module.

In developing Use Case 4 the challenge I faced was on what basis the ranking should be done on the players, as it was not clear to me in the beginning. But giving all the players initial fund to investment that made the calculations much easier.

The course Software Engineering has been of great help to me in terms of developing knowledge of languages like UML and strengthens my knowledge about JAVA. It has also taught me to work on a project and to coordinate with team members.

One of the technical challenges for me was the email part of UC-3. After researching on this for a while, I understood that JAVA offers mail Apache to serve this purpose. I also had a few problems with developing the logical part of my code. But I could resolve this problem with the help of the course. The course also helped me understand how to go about working on a project and develop efficient and effective code. It would have been helpful had we been given a detailed description of the general problems that occur in developing projects and the best practices to handle them.

Project challenges from Osha Fuangkasae:

• Communication between all languages.

We had to use many different communication protocols that we were unfamiliar with, including XMLRPC, JavaMail, and JDBC. Because none of us had much experience in these communication protocols, development time was longer, and development itself included the frustrating bugs due to protocol inexperience. However, this was to be expected, and the protocols we used are fairly ubiquitous, meaning our future projects will be less frustrating.

• Project and Team management

In graduate school, students have multiple responsibilities, which can include social relationships, jobs, other class work, and many more. Because of this, it was difficult to get everyone in our team synchronized. Willing team members would miss important meetings and would then be lost in future discussions. At the same time, those who have kept abreast of the

informational sessions had their own responsibilities, which prevented them from keeping the others updated.

• Coding

We did not have many difficulties in cross code modification. Team members would have self-contained isolated assignments and did not often modify each other's code, which is often a source of frustration in other software projects. This luck is probably due to the well-developed use cases, which form well-defined divisions between concept responsibilities. Possible improvements can still be made in our collaborative code however, as there was some replicated code segments dealing with database communication across developers. Furthermore, since we do not expect others to reuse any of our code, and because a lot of trial and error was necessary to get the protocols working, it lacks commenting and may be sloppier than otherwise.

Future Work

(1) Clean up of code. If we were to continue this project, we would want to make sure that the proper Perl libraries are installed on the EE server. Once everything is installed properly, we finally load our program on the server fix any lingering issues.

(2) Metrics of Sharpe ratio and Value@Risk implemented in Use Case 2 and 11 are generic metrics to help investors understand the risk position of their portfolios or stocks. However, financial market is very complicated and with the limited information conveyed in only one or two metrics, one can only get a partial picture of how his or her portfolio stands in the market. Thus, more risk metrics such as volatility/standard deviation, semi-variance (or downside risk) and expected shortfall, can be further explored to provide investors more information specific to the composition of their portfolio.

(3) Developing HTML script containing the portfolio to be emailed to the investors.

(4) Alerting the investor to deposit funds into their accounts whenever the available funds go below a certain limit.

(5) Following the practices adopted by the brokerage companies around the world

(6) Developing interfaces to other brokerage companies around the world to facilitate and coordinate the flow of stocks

(7) Improving the refreshing rate of the stock database

(8) Executing investors' requests as quickly as possible and

(9) Trying to maintain the integrity of the database.

(10)Along with the strategies of the best players, the new players could be benefited if some fundamental information about the strategies is displayed. for ex:

i. Scalping: Scalping is one of the most popular strategies, and it involves selling almost immediately after a trade becomes profitable. Here the price target is obviously just after profitability is attained.

ii. Fading: Fading involves shorting stocks after rapid moves upwards. This is based on the assumption that (1) they are overbought, (2) early buyers are ready to begin taking profits and (3) existing buyers may be scared out. Although risky, this strategy can be extremely rewarding. Here the price target is when buyers begin stepping in again.

iii. Daily Pivots: This strategy involves profiting from a stock's daily volatility. This is done by attempting to buy at the low of the day (LOD) and sell at the high of the day (HOD). Here the price target is simply at the next sign of a reversal, using the same patterns as above.

iv. Momentum: This strategy usually involves trading on news releases or finding strong trending moves supported by high volume. One type of momentum trader will buy on news releases and ride a trend until it exhibits signs of reversal. The other type will fade the price surge. Here the price target is when volume begins to decrease and bearish candles start appearing.

(11)Web links like this can be displayed to educate the new traders.
http://www.investopedia.com/articles/trading/06/DayTradingRetail.asp

References

The list of references should contain exact references and URLs of any material that is used in the project and doesn't come from the textbook.

1. Software Architecture (Component-based software engineering) Wiki: http://en.wikipedia.org/wiki/Software_componentry
2. Call Perl routines from Java: http://www.javaworld.com/javaworld/jw-10-2004/jw-1011-xmlrpc.html
3. The Java Tutorials (JDBC Database Access): http://java.sun.com/docs/books/tutorial/jdbc/basics/connecting.html
4. Omondo: http://www.omondo.com/
5. Z. Bodie, A. Kane, and A. Marcus, Investment, 7th ed., 2006
6. Markowitz Portfolio Analysis for the Individual Investor - http://faculty.washington.edu/rons/articles/mark.ps
7. Individual Investor Risk Aversion and Investment Portfolio Composition - http://www.jstor.org/view/00221082/di991947/99p07387/0
8. Value at Risk - http://en.wikipedia.org/wiki/Value_at_risk
9. Introduction to Value at Risk (VAR) – Part 1, http://www.investopedia.com/articles/04/092904.asp
10. Single Index Model - http://en.wikipedia.org/wiki/Single_Index_Model
11. Correspondence with Ivan Marsic. *Re: ece567 - project questions*. Sept 22, 2007. 8:21 AM
12. Perl Finance::Quote package documentation - http://sourceforge.net/docman/display_doc.php?docid=168&group_id=4232
13. UML standards-

    http://www.omg.org/gettingstarted/what_is_uml.htm

Appendix

For Use Case 2 and 11, Written by Zong-Zhi 'Alex' Lin)

There are total 9 classes for Use Case 2 and 11.

BaseStats.java

-- computeExpectedReturn() : Calculate Expected Return of a set of data.

-- computeVariance() : Calculate Variance of a data set.

EvaluatedStockSet.java

-- notInStockComputedSet() : Check if a stock has been analyzed and evaluated.

InvestorController.java -- Retrieve all the active investors, store them in an ArrayList,
and then evaluate and update their portfolios one by one.

-- InvestorController() : constructor

-- getMaxNumOfTrial() : Retrieve the maximal number of allowed trials for attempting to connect to database.

-- getNumOfTrial() : Retrieve the number of attempted trials for connecting to database.

-- incrementFailCounts() : Increment the number of trial after a failed attmpt to connect to network.

-- getInvestors() : Retrieve all investors and store them in an ArrayList.

-- updateInvestorsPortfolioParameters() : Update parameters assoicated with investor's portfolio after the evaluation

-- printPortfolioParameters() : Print parameters assoicated with investor's portfolio after the evaluation

-- portfolioRiskAnalysis() : Peroform risk analysis of each investor's portfolio one by one.

UC2Controller.java

-- UC2Controller() : Execute Use Case 2 and Use Case 11.

MarketEvaluator.java

-- getMarketHistory() : Retrieve all the market price history (Dow Jones Industrial Average) to prepare for doing standard linear regression

-- computeMarketParameters() : Perform standard linear regression

-- updateMarketParameters() : Update parameters of the market after standard linear regression

-- updateMarketSharpeRatio() : Update Market's Sharpe Ratio

-- moveMarketHistoryData() : Move market history price data from attribute roi to attribute ticker to prepare for evaluating the stocks and investor's portfolio against market history data.

-- updateMarketDispersion() : Update Market Dispersion such as expected return and variance.

-- refreshMarketParameters() : a routine to perform all the tasks in this class.

PortfolioEvaluator.java

-- getStocks() : Use investor unique id to get all the stocks owned by the investor and store those stocks in an ArrayList.

-- calculatePortfolioValue() : calculate Portfolio Value (use current stock price)

-- calculatePortfolioTotalGain() : calculate Portfolio Total Gain, equal the difference portfolio value obtained from the method of calculatePortfolioValue() and the portfolio value at the time of the purchase.

-- calculateStocksWeightsInPortfolio() : Calculate the weight of each stock in a portfolio

-- updateStocksWeightsInPortfolio() : Update the weight of each stock in a portfolio to portfolio table.

-- evaluateStocksInPortfolio() : Perform standard linear regression on each stock.

-- calculatePortfolioParameters() : calculate Portfolio Parameters basing on the computational results obtained from the method of evaluateStocksInPortfolio().

-- updatePortfolioParameters() : update Portfolio Parameters to table investor.

-- computeSharpeRatio() : Compute Sharpe ratio of a portfolio

-- evaluatePortfolioParameters() : a method to perform all the tasks in this class.

SingleIndexModel.java

-- linear_equation() : Apply least squares to raw data to determine the coefficients for an n-order equation: y = a0*X^0 + a1*X^1 + ... + an*X^n. Returns the coefficients for the solved equation, given a number of y and x data points.

-- gauss() : This is a standard gaussian technique for solving simultaneous equations of the form: |A| = |B| * |C| where we know the values of |A| and |B|, and we are solving for the coefficients in |C|.

-- print_equation() : simple routine to print the equation for inspection.

-- marketMatrix() : Read data points in ArrayList into double data matrix which can directly feed into the routine to least square linear regression.

StockEvaluator.java

-- getAllStocks() : Retrieve all the stocks of an investor's portfolio to prepare for doing standard linear regression

-- getStockHistory() : Retrieve all the stock price history too prepare for doing standard linear regression against all the market price history

-- computeStockParameters() : Perform standard linear regression between stock performance and market performance.

-- updateStockParameters() : Update stock parameters obtained by the method computeStockParameters() to table stock.

-- updateAllAvailableStocks() : Compute and update stock parameters obtained by the method computeStockParameters() to table stock.

ValueAtRisk.java

-- doesInvestorOwnStocks() : Check if an investor owns any stock at all.

-- printValueAtRisk() : Print the values obtained from the method calculateValueAtRisk().

-- calculateValueAtRisk() : Calculate the maximum lost value within a specified level of confidence level.

Written by Osha Fuangkasae

DatabaseCommunicator.java
connect and query the database

ECE567.java
contains main() function, receives input from the user

Mailer.java
creates and sends e-mail

PerlCommunicator.java
calls the relevant perl functions

SystemMenu.java
stores the strings that the Main Menu uses

Observer.java

interface for all controllers to listen to the timer

UC1Controller.java, UC7Controller.java, UC8Controller.java, UC10Controller.java
executes their respective use cases

Lock.java
required to keep the two timers synchronized without interleaving

Queries.java
contains commonly used SQL queries

Subject.java
interface of timer that allows it to register observers

XMLParser.java
parses an XMLReader to yield the relevant data to submit to the database

XMLReader.java
contains an xml string that can be parsed by XMLParser.java