Rutgers University Investing Report #1

Group Members:

Daniel Su	Thanh Do Huu	Micah Moore
Jason Scatena	Boris Hazanov	Tam Duong

Table of Contents

	Team Members					
	Daniel Su	Micah Moore	Jason Scatena	Boris Hazanov	Tam Duong	Thanh Do Huu
Project Management (10 pts)	50%	25%		25%		
Customer Statement of Requirements (9 pts)	50%	5.6%	27.8%	5.6%	5.6%	5.6%
System Requirements (6 pts)	50%			16.7%	16.7%	16.7%
Functional Requirements Specification (30 pts)	50%		43.3%	6.7%		
User Interface Specs (15 pts)		36.7%			26.7%	36.7%
Domain Analysis (25 pts)		40%			20%	40%
Plan of Work (5 pts)			100%			

Point Allocation:
Boris Hazanov: (2.5+.5+1+2) = 6

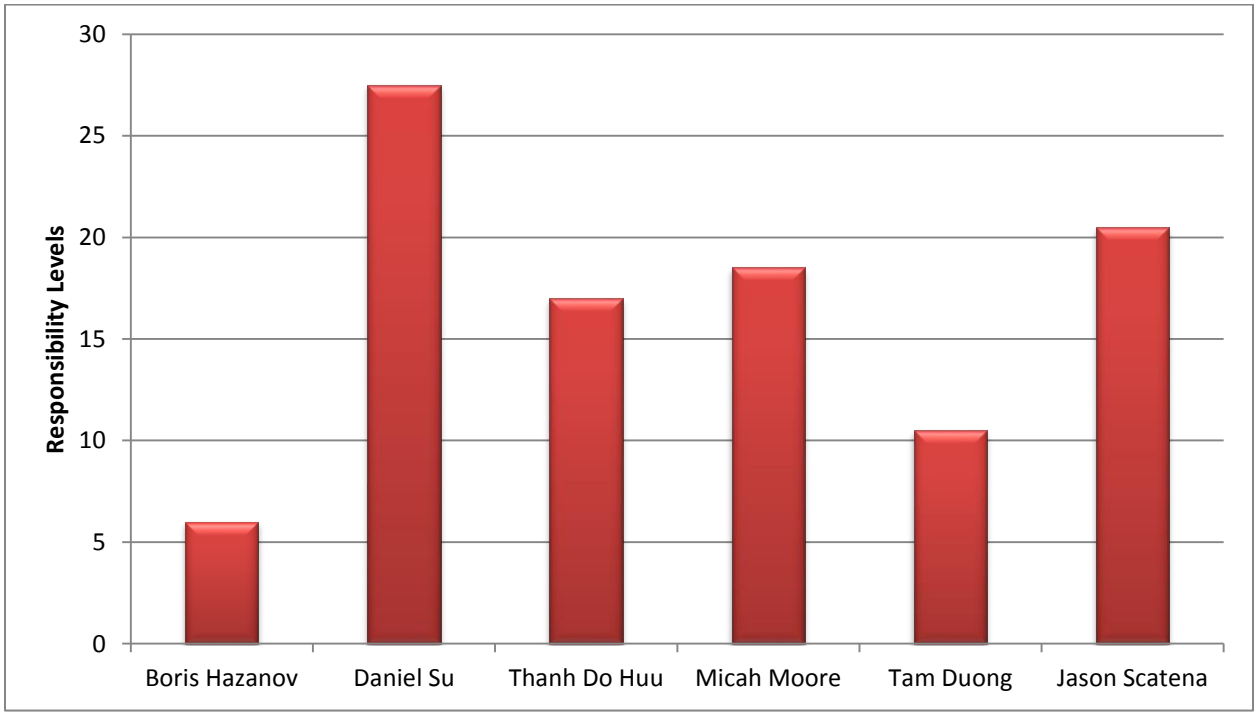
Daniel Su: (5+0.5+4+1+1+1+5+5+5) = 27.5

Thanh Do Huu: (.5+1+5.5+5+2.5+2.5) = 17

Micah Moore: (2.5+.5+5.5+5+2.5+2.5) = 18.5

Tam Duong : (.5+1+4+5) = 10.5

Jason Scatena: (2.5+8+5+5) = 20.5



Customer Statement of Requirements

Problem Statement

To Whom It May Concern,

As you no doubt know already, your team has been contracted to develop a stock market fantasy league software system for the Rutgers School of Business. Our department has decided to pursue this venture as a means of furthering our mission: "Rutgers Business School serves New Jersey and the world by cultivating business knowledge, ethical judgment, and global perspective in our students, scholarship, and society through innovative research and teaching and robust partnerships with businesses across our diverse communities." We have decided that the best way to "cultivate business knowledge…and global perspective" in our students is to employ a stock market game.

There is great value in experience, especially in a topic such as investing in the stock market. The complexity of reading the market and making wise investments makes first-hand experience an extremely valuable learning tool. Students often times don't have disposable income that they can afford to risk playing the market. In order to address this, the concept of a stock market game has come about, in which contestants are given a set amount of fictional money to "invest" in real stocks. At the end of the game, the participant with the most money wins. While disguised as a game, this activity provides students of finance an opportunity to make use of concepts learned in class and hone their abilities. The financial industry has changed considerably in recent times. While a trader a mere 20 years ago would rely on television, newspapers and telephones to gather market information traders today have the entire internet at their disposal. The birth of the modern web has allowed the populace to become generators of media in addition to consumers. It is now more possible than ever to gauge public opinion on companies and products. A myriad of online services exist that will allow a user to set up a stock market game at no cost. None of these other services, however, stress the importance of using the modern internet and social media as an information gathering tool. A stock market game that addresses this can better prepare students for real life trading. This can be done by building a stock market game service that integrates, at it's very core, tools to make sense of the flood of information being posted to social media at all times. To not train students to use such a readily available source of pertinent information, would put our them at a major disadvantage.

In order for a stock market game to be usable, certain key functionalities will need to be included. There are three main roles users of our system will take: system administrator, league administrator, and player. The role of system administrator should be designed so that it can be undertaken by a professor or a member of the department who is assigned this extra responsibility. Whomever this role falls upon will, obviously, need a working understanding of the system at a fairly low-level since they will be the first resource in the event of a system malfunction. We understand that this role, in it's nature, requires a fairly large base of knowledge. We expect your team to either train personally, or provide documentation for the benefit of that individual. The system administrator will manage day to day activities relating to the hosting of the system, updates to the system, adding and removing league administrators, and monitoring system resource usage.

Each professor, that teaches one or more sections of a relevant course, will take on the role of league administrator. Extensive computer literacy cannot be assumed, and as such the user

interface must put a heavy focus on being intuitive and easy to use. From this interface the professor must be able to control the entire game for his class sections. He or she will be able to manage player interaction, help players troubleshoot issues they may encounter, create a new league, and manually add or remove player accounts. The professor will have tools at his or her disposal that allow him or her to gain a big picture overview of the league. For example leaderboards and other league statistics. The league administrator will also be able to adjust rules and policies for his or her specific league so as to help it align more closely with the course syllabus.

Students will take on the role of system player. To avoid unnecessary work on the part of the league administrator, the system should allow players to register a new account and join the appropriate league with no administrator interaction. When a player accesses the web interface they will be presented with the functionality needed to research, buy, and sell stocks. The players will need to have options for data visualization and statistics tracking. A host of options for alerting the player of pertinent information using products they are already comfortable with should be explored, such as: twitter, SMS, and email.

As primarily an education tool, the game must attempt to train student players to best make use of the resources available to them. The market value of a company isn't exclusively based on how they are doing economically; public opinion greatly influences stock prices. As of last summer, twitter had over 210 million active users. The flood of tweets posted daily act as a free source of market research giving a survey of public opinion. Using this freely available market information, we want students, using our system, to be able to make use of functionality unavailable in previous stock market simulations and become more broadly informed investors.

We would like you to build a system that gives players a twitter-sourced sentiment score for all the stocks in their portfolio. In addition, players should be able to manually ask the system to watch the sentiment of prospective stocks. To further improve the player's experience, the game should also allow players to follow certain organizations, groups or fields of interest using public twitter accounts. The goal of this feature is to quickly and accurately inform players of recent changes in stocks that they own or are researching. This information will help players make informed decisions, using the public opinion of a company's success, as opposed to solely the news and financial data available through more traditional media.

In order to make the system easily accessible, we would like to make the game a web application, not requiring professors or students to download and install anything to their personal devices. Professors and students should interact with the game through a pragmatic and clean web interface that looks good and is easily usable across a host of devices. We would like this interface to be hosted on a server owned by the department and handled by the system administrator. We expect the interface to provide access to all the functionality of the system in a clear and intuitive way.

The functionality we expect from this system can be summarized as follows:

- For every user
  - Pre-login info page
  - Login
  - Registration
  - Account management
    - Change password
    - Change contact information
- For the system administrator
  - Add or remove professors as league administrators
  - View system usage statistics
  - Manually reset user passwords
- For the League administrators
  - Start, manage and end a league
  - Set league rules and policies
    - starting capital
    - start date
    - end date
    - league name
  - Manually add or remove players
  - View league statistics
- For the Player
  - View their portfolio
  - View competitor portfolios through a profile page
  - Initiate market orders
    - Short
    - Cover
  - Initiate conditional orders
    - Stop orders
    - Limit Orders
  - View twitter feeds of relevant organizations
  - View twitter sentiment scores of various organizations
  - View stock price history of any public company
  - view leaderboard and other league statistics
  - Communicate with league administrator
  - Access public league chat room

We are looking forward to seeing the system that your team develops and encourage you to contact us if you have any questions or updates.

Regards,
The Rutgers School of Business Development Board

Glossary of Terms

Stock Market League - A market simulation that allows users to practice trading and learn how the market works

Stock - A type of security that represents a claim on part of corporation's assets and earnings

Ask Price - The minimum price that a seller or sellers are willing to receive for the security

Bid Price - The maximum price that a buyer or buyers are willing to pay for a security

Ticker Symbol - A stock symbol or ticker symbol is an abbreviation used to uniquely identify publicly traded shares of a particular stock on a particular stock market

User Groups:

Player - A standard user that participates in the leagues and has control over their personal profile and settings

League Administrator - The league administrator manages leagues that they have created and the players that participate in those leagues

System Administrator - The system administrator is the super user who has the highest privileges and can manage all other users as well as system settings

Orders:
Market Orders (Immediately executed):

Buy Order - An order to purchase a specific amount of stock

Sell Order - An order to sell a specific amount of stock

Short Order - An order where a sell is performed using borrowed stocks. The trader then expects the value to decrease and to profit by performing a cover order to return the loaned stocks at a lower price

Cover Order - An order where a buy is performed in order to cover / return stocks that were previously loaned to the trader

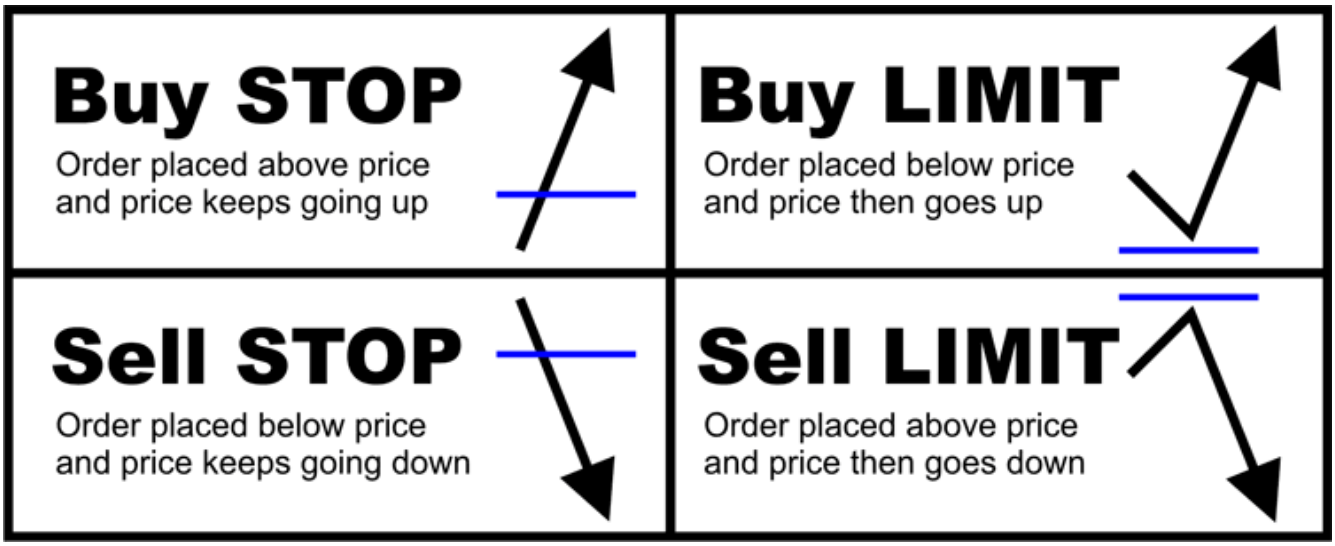
Conditional Orders (Executed on met condition):

Stop Order - An order that activates only when the security you want to buy or sell reaches the stop price

Limit Order - An order that sets the maximum or minimum at which to buy or sell a stock

*Limit orders guarantee the trade will be made at particular price while a stop order does not

Below is an image that allows better understanding of these terms



Additional information can be found at http://www.investopedia.com/

System Requirements

Functional Requirements

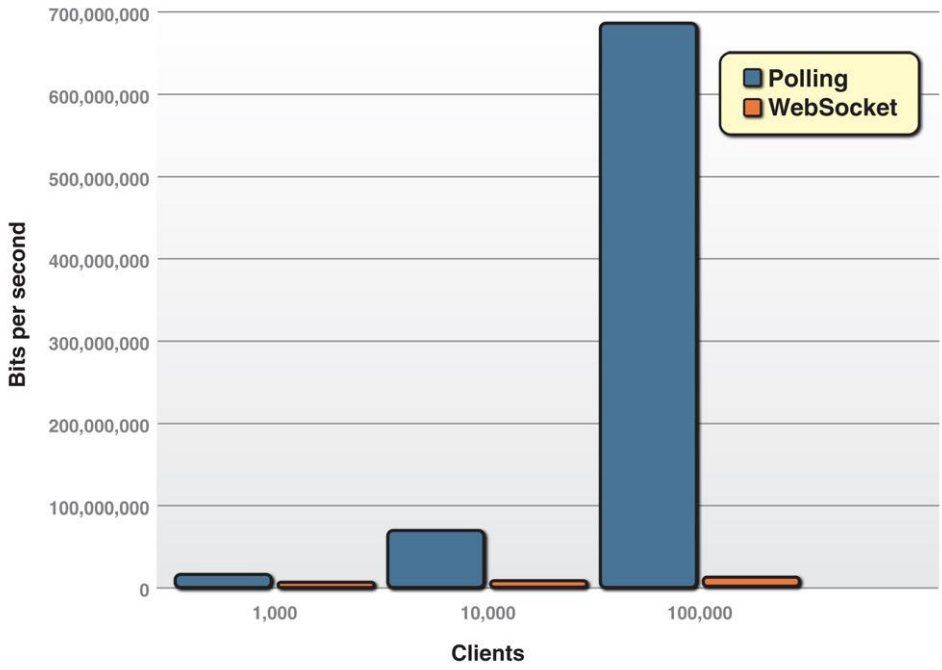
PW = Priority Weight (from 1 to 5)

ID	PW	Requirement
REQ-1	5	The system will provide an information page pre-login.
REQ-2	5	The system will allow users to log-in.
REQ-3	5	The system will provide a registration page for new users.
REQ-4	5	The system will allow users to change their password.
REQ-5	4	The system will allow users to change their contact information.
REQ-6	5	The system will allow the system administrator to add or remove professors as league administrators.
REQ-7	5	The system will allow the system administrator to view system usage statistics.
REQ-8	5	The system will allow systems administrator to manually change a user's password.
REQ-9	5	The system will allow to the league administrator to start, manage and end a league.
REQ-10	5	The system will allow to the league administrator to set league rules, including: • Starting capital. • Start date. • End date. • League name.
REQ-11	4	System will allow league administrator to add/remove players within their league manually.
REQ-12	3	System will allow league administrator to view leagues statistics.
REQ-13	3	System will allow to each user view his statistics.
REQ-14	3	System will allow to each user View competitor portfolios through a profile page

REQ-15	3	System will allow to each user Initiate market orders such as: • Short • Cover
REQ-16	2	System will allow to each user Initiate market orders such as: • Stop orders • Limit Orders
REQ-17	3	The system will allow users to View twitter feeds of relevant organizations
REQ-18	2	The system will allow users to View twitter sentiment scores of various organizations
REQ-19	3	The system will allow users to View stock price history of any public company
REQ-20	2	The system will allow users to View leaderboard and other league statistics
REQ-21	5	The system will allow for low latency real-time trades
REQ-22	5	The system will allow for scalability

Analysis on REQ-22

Stock market simulation systems done by previous groups all had one thing in common. They utilized the HTTP protocol to accomplish market orders. While this works fine in small scale deployments, when these systems are rolled into high volume environments, they will start to require massive costs to maintain. What we are proposing is to use a WebSocket implementation that reduces the overhead for each trade from up to kilobytes of data, down to just a few bytes. With a high enough volume of trading, this implementation can save a very significant amount of money.



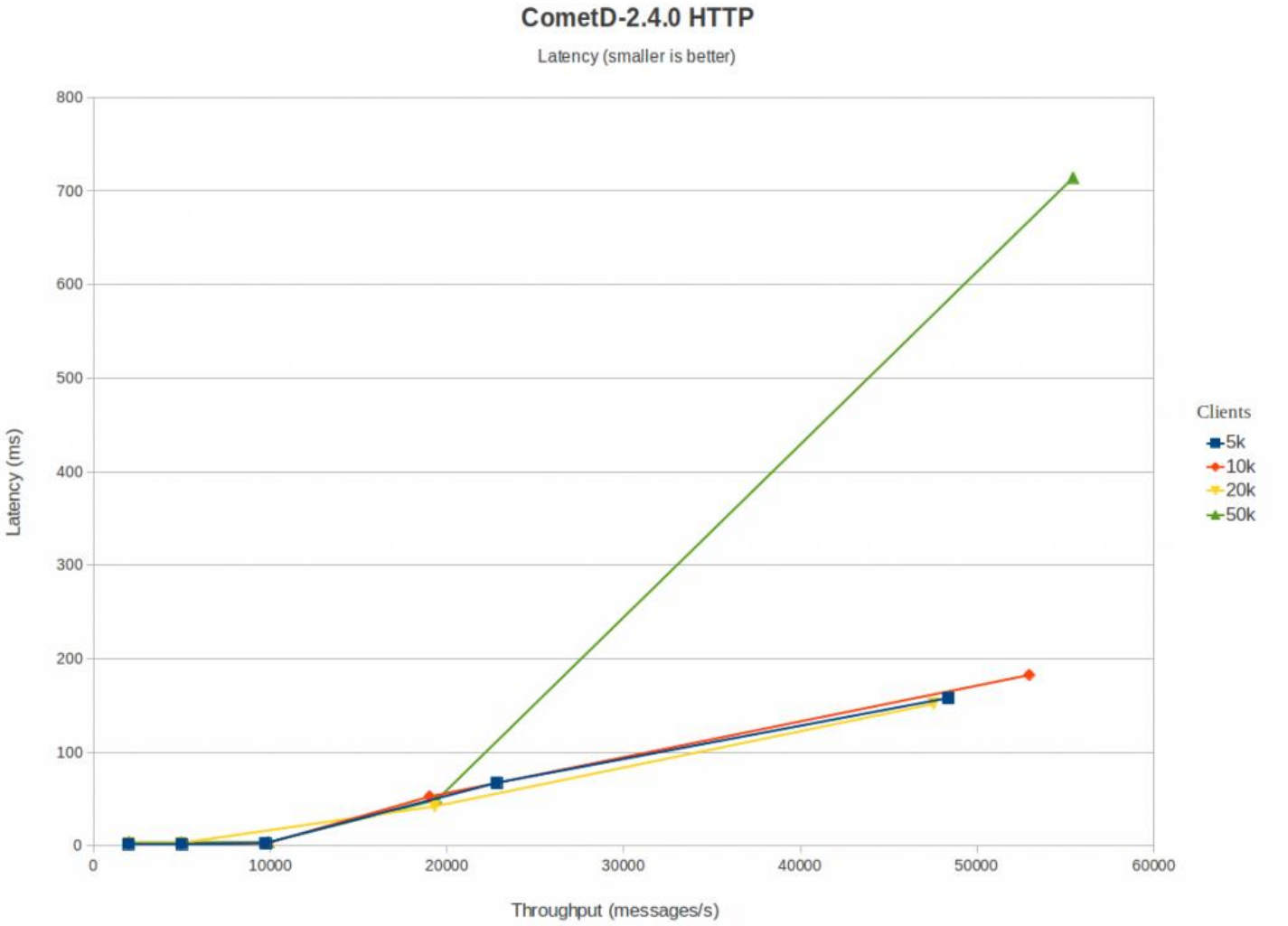
http://blog.kaazing.com/2010/02/24/5-signs-you-need-html5-web-sockets-part-2/

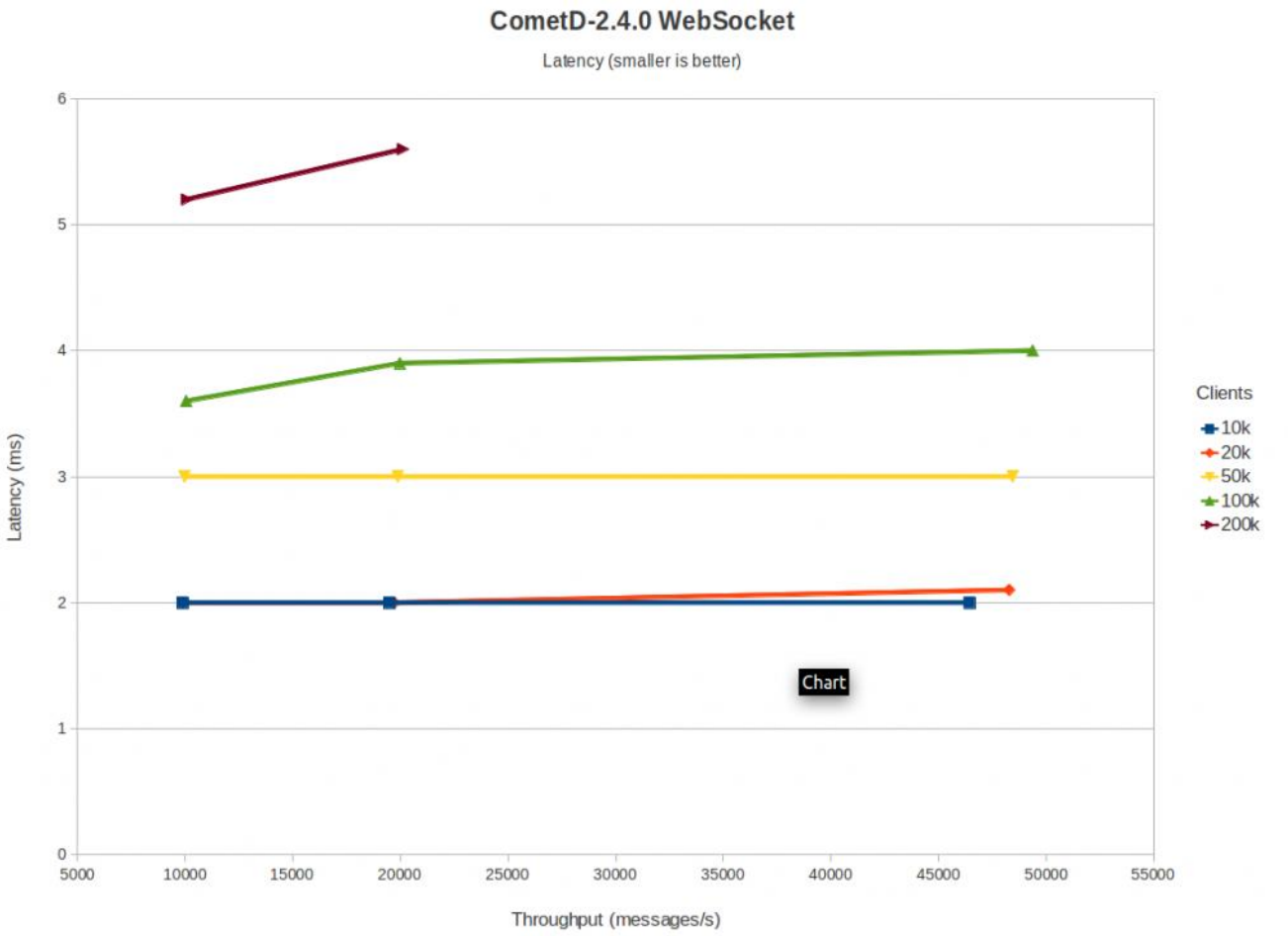
Analysis on REQ-21

By using the WebSocket implementation, we also reduce the latency response down. This can have huge ramifications for a high frequency trading simulation. InformationWeek (http://www.informationweek.com/wall-streets-quest-to-process-data-at-the-speed-of-light/d/d-id/1054287) estimates that 1ms of latency can be worth up to $100 million per year to a major brokerage firm. The quest for low latency is so high, that firms are willing to lay down private fiber lines to improve latency response by just microseconds. (http://spreadnetworks.com/press-releases/10-04-2012-latency-improvements/)

While our implementation is not to conduct real world market trades, a low latency response does allow for more simulation options.

Below are simulations on the two different protocols that demonstrates the advantage of WebSocket


CometD-2.4.0 HTTP
Latency (smaller is better)
Latency (ms)
800
700
600
500
400
300
200
100
0
0
10000
20000
30000
40000
50000
60000
Throughput (messages/s)
Clients
5k
10k
20k
50k


CometD-2.4.0 WebSocket
Latency (smaller is better)
Latency (ms)
Throughput (messages/s)
Clients
10k
20k
50k
100k
200k
Chart
0
1
2
3
4
5
6
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
55000

Non Functional Requirements

Functional
The big three web browsers will be supported so compatibility with a majority of users will be achieved. User security will be a major priority with features for authentication and encryption for sensitive data such as passkeys.

Usability
We will focus on providing a clean and consistent interface through CSS that will appeal to the user. AJAX will also be used to allow for immediate responses for user actions.

Reliability
Users will be given a confirmation message for sales transactions to allow for detection of user error. Error messages will be displayed to the user to notify failure of completing the proposed action.

Performance
There will be a focus on scalability and an efficient system for passing message between the client and the server (ie. for transactions).

Supportability
The front end will support access from mobile devices or devices with smaller resolutions. Maintenance support is also included through the admin control system.

PW = Priority Weight (from 1 to 5)

ID	PW	Requirement
REQ-23	3	Inter browser compatibility
REQ-24	5	User Security
REQ-25	2	Graphical Design
REQ-26	4	Responsiveness
REQ-27	3	Error Response
REQ-28	4	Scaling and Efficiency
REQ-29	3	Front End Interface
REQ-30	4	Maintenance Control

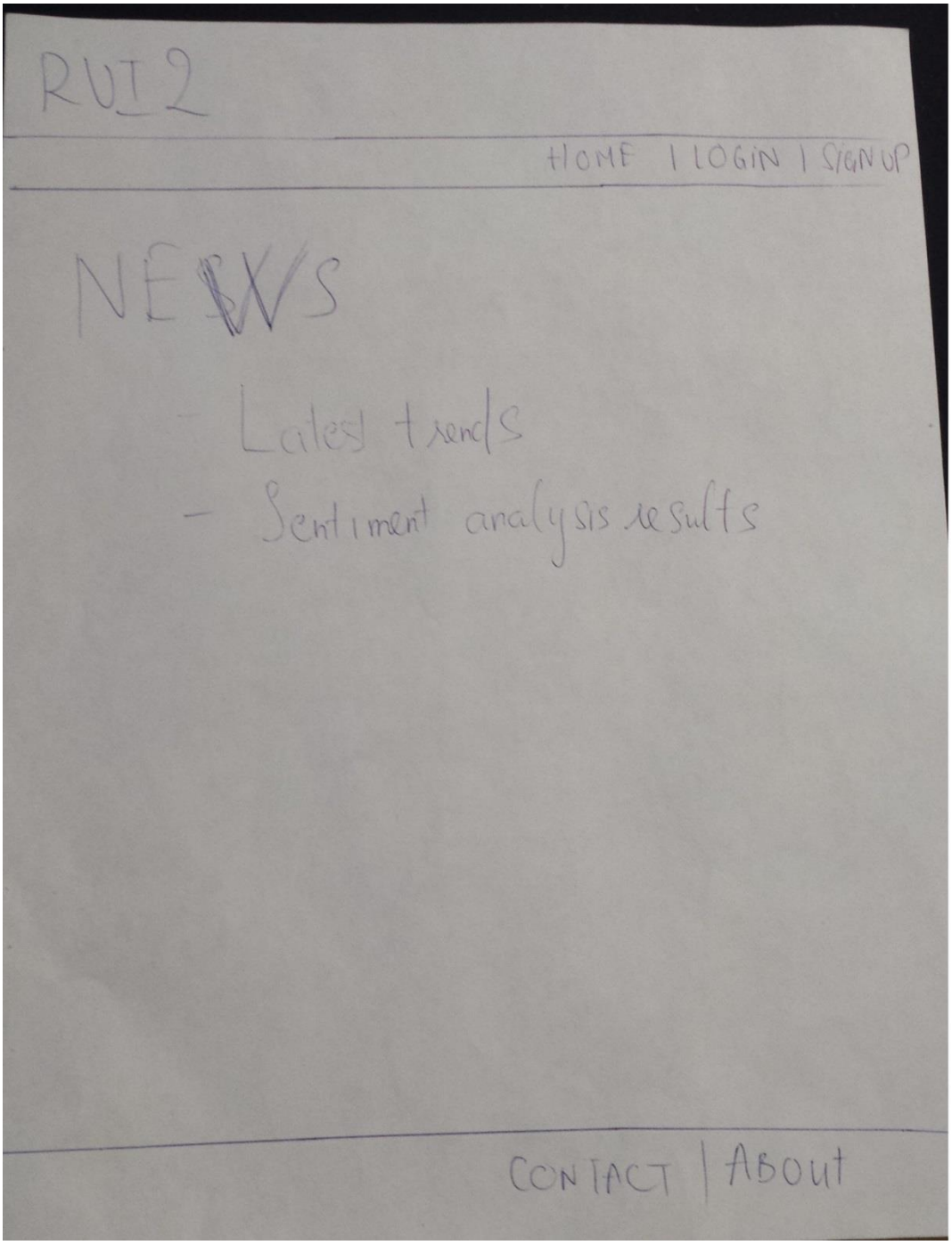
On-Screen Appearance Requirements

The on screen appearance design will primarily cater to those with a laptop and desktop system, for resolutions of 720p and greater. There will also be support for handheld and tablet devices through the use of responsive CSS in order provide these users with a functional interface. Dynamic data loading through JavaScript will also be used in order to minimize wait time for the user.

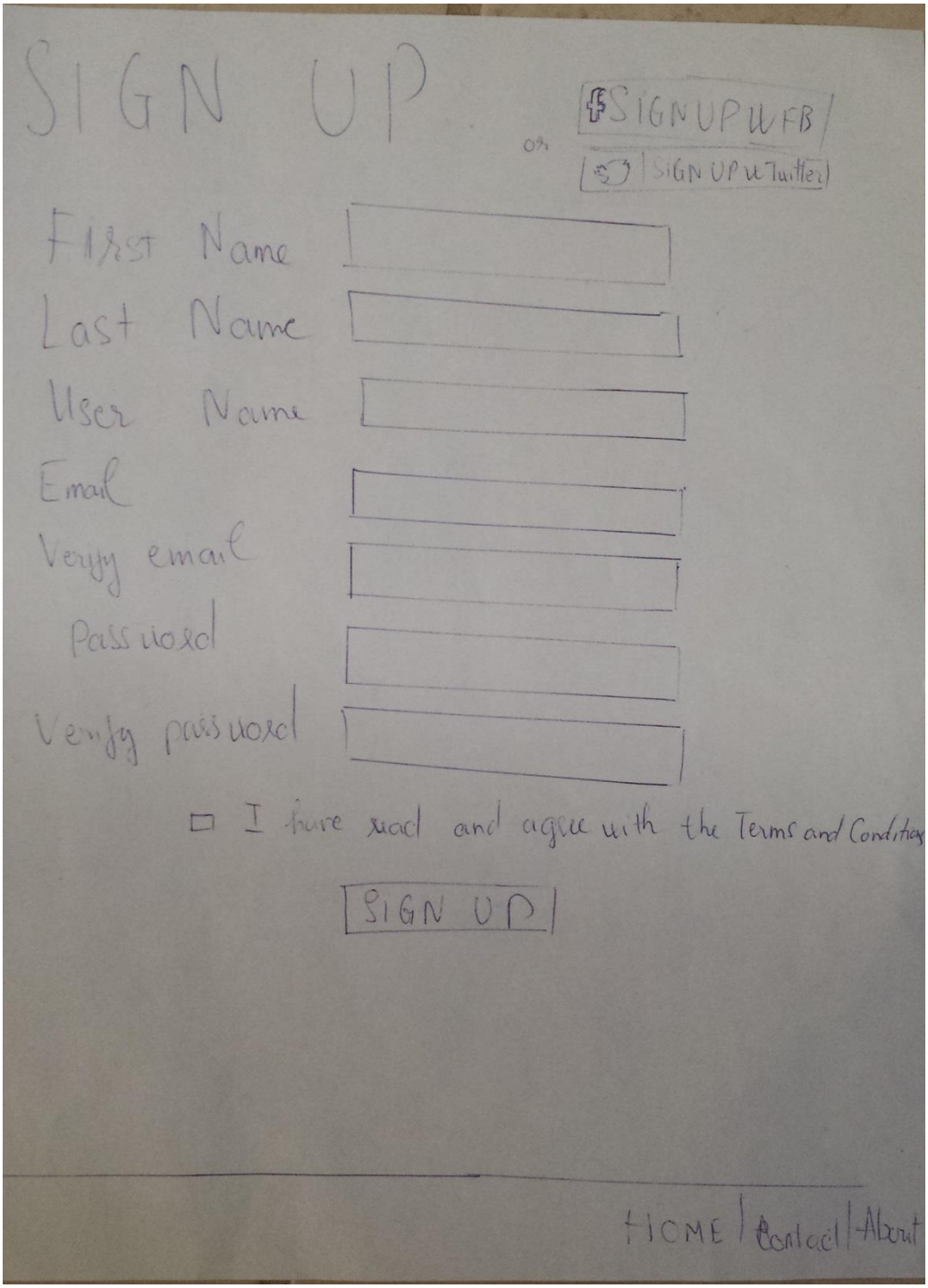
Technologies used will be restricted to those that are universally compatible. Flash and Java will not be utilized due to their limited compatibility and massive resource drain.

ID	PW	Requirement
REQ-31	3	Responsive CSS / Cross Device Compatibility
REQ-32	4	Rapid dynamic data updates

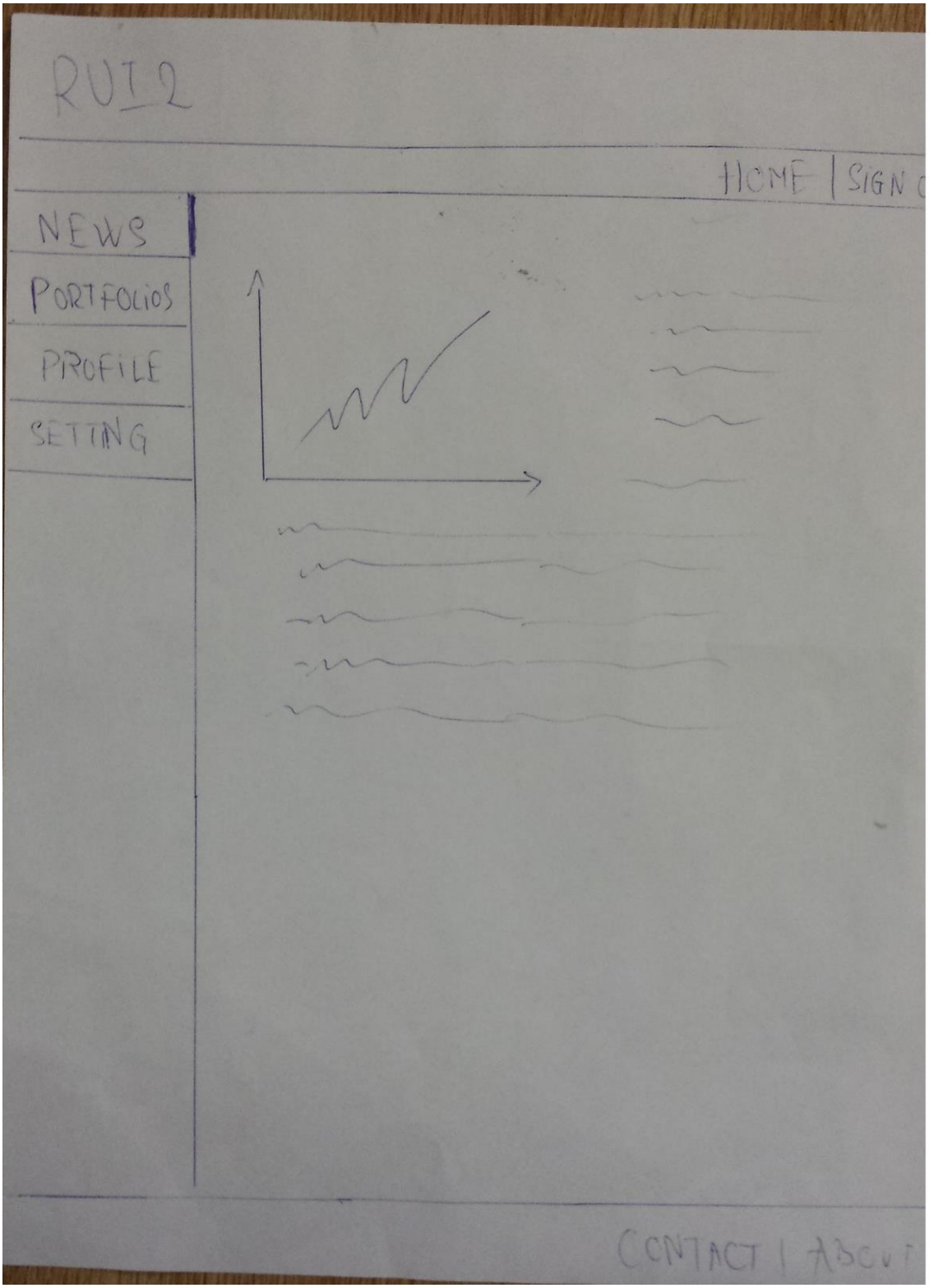
Home Page


RUI 2
HOME | LOGIN | SIGN UP
NEWS
- Latest trends
- Sentiment analysis results
CONTACT | ABOUT

Signup Page


SIGN UP
or
SIGN UP W FB
SIGN UP W Twitter
First Name
Last Name
User Name
Email
Verify email
Password
Verify password
I have read and agree with the Terms and Conditions
SIGN UP
HOME | Contact | About

User's Page


RUI 2
HOME | SIGN
NEWS
PORTFOLIOS
PROFILE
SETTING
CONTACT | ABOUT

Functional Requirements Specification

Stakeholders:

Internal stakeholders:

- Owners: people who legally have the right to possess the web application. Owners' goals is contributing the software to multiple universities across the world, keep low cost maintenance and customer satisfaction.

- Managers: person who is in charge of affairs, , resources and expenditures of the web application. They interests include performance, growth, customer satisfaction, profit, cost, employees, and demand.

- Employees: person who is in charge of maintaining the web application, they will provide support to system administrators (university professors) , they will perform software updates to minimize troubleshooting. Their interests are reliability, working conditions, salary, working hours, job security, and benefits.

External stakeholders:

- Customers (Universities, students and professors): A person who is registered in the web application and creates his own stock portfolio, a system administrator(professor) can register without having stock portfolio and universities that will host the web application on their web page. . The customer's interests include software value, quality, reliability and service.

- Advertisers: An academic institute who calls the attention of the students to participate in daily activities. Finance companies who want to address students.
  Advertiser's interests are number of customers, detecting leading players.

- Ministry of Education- The executive policy making body in the united states. Ministry of education goal is to provide better finance studies platform in universities without putting in risk student money.
- Competitors: A company providing the identical products to universities. Their interests include profit, demand, customers and quality.

- Stock Researcher: An individual who researches the human behavior in the field of investment in stocks. His or her interests include investors, human behavior, and investing strategies.

Actors and Goals:

- User: Any student which enrolled to Finance class.
  Type: Initiating
  Goal: Create an account

- Investor: A student who is authenticated using the login system and is interacting with the system, portfolio.
  Type: Initiating
  Goal: Login,join a league, monitor stocks and portfolio, buy and sell.

- League Administrator: Class professor who is authenticated using the login system and is interacting with the system.
  Type: Initiating.
  Goal: Login,create new league, monitor students portfolios, create and delete accounts.

- Yahoo Finance: The external source where real-world stock quotes are obtained at periodic time intervals.
  Type: Participating
  Goals: None.

- Database: A place where information about the various stocks such as price quotes,ticker symbols, and market name, are stored. Also, it stores a list of investors currently part of the system and their settings such as user id, passwords, email address, and other personal details.
  Type: Participating
  Goal: None.

- Email Server: A machine responsible for sending messages to investors via E-mail and SMS.
  Type: Participating
  Goal: None.

- Advertiser: An individual who interacts with the system through a user account and posts university related activities. Finance companies who interacts with the system through a user account and posts job openings.
  Type: Initiating
  Goals: Login. Post Advertisements. Remove Advertisements.

Use Cases

UC-1: Register -- Allows a student to register an account and enter a game by filling out a form and entering a class code given out by a professor/league administrator
Derived from REQ-3

UC-2: Make Trades -- Allows a player to initiate trade orders, the system will then respond appropriately based on market conditions and status
Derived from REQ-15 & REQ-16

UC-3: Setup League -- Allow a league administrator to start a game and initialize settings
Derived from REQ-9 & REQ-10

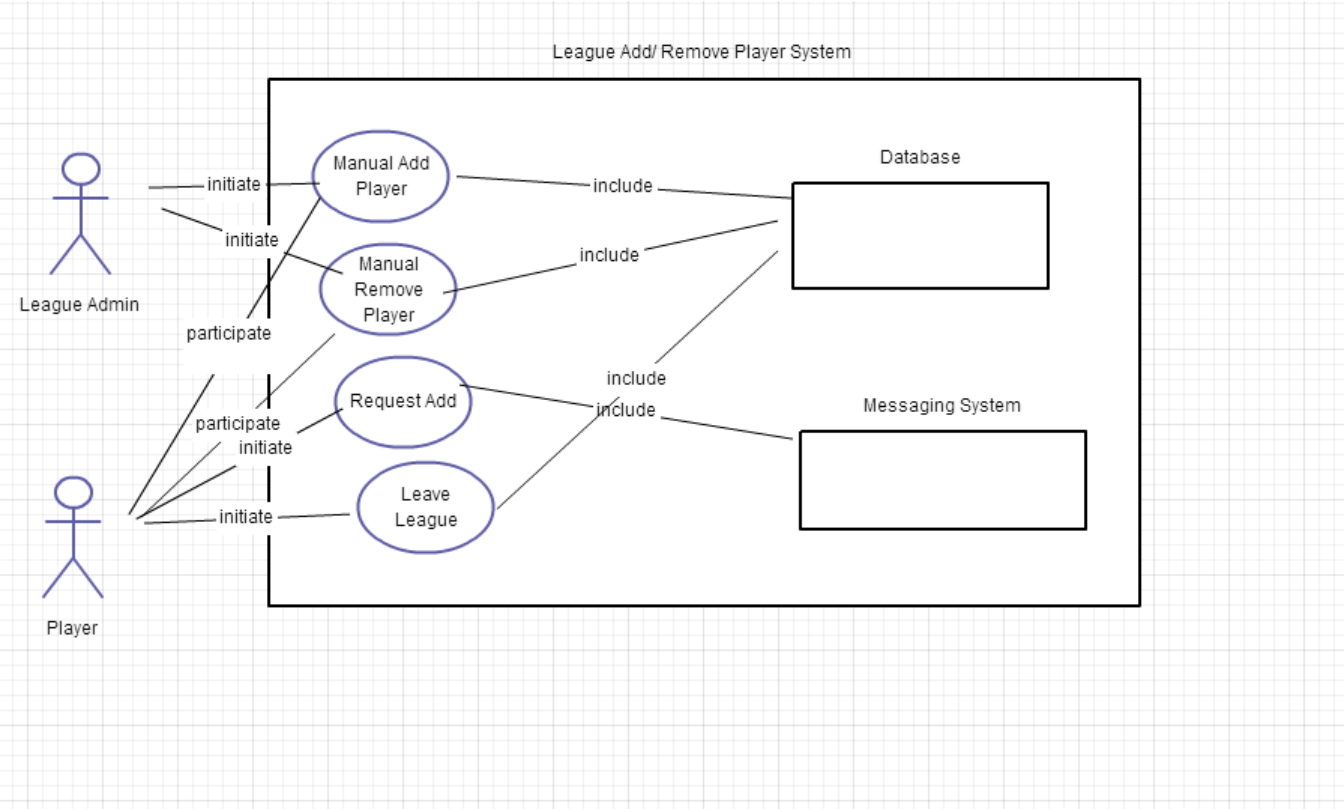
UC-4: View Profile -- Allows all users in a league to view the profiles of all other players in the league.
Derived from REQ-13 & REQ-14

UC-5: Manually add player to Leauge -- Allow a league administrator to manually add a student's account.
Derived from REQ-11

UC-6: Player Joining a League -- Allow a student account to join a league using a password supplied to them by their teacher, the league administrator.
Derived from REQ-11

UC-7: View League Stats -- Allow a player to exam a statistics and leaderboard page for the league they are in.
Derived from REQ-20

UC-8: Twitter Research -- Allows a user to make use of twitter streams for market research through a custom tool.
Derived from REQ-17 and REQ-18


League Add/ Remove Player System
League Admin
Player
initiate
initiate
participate
participate
initiate
initiate
Manual Add Player
Manual Remove Player
Request Add
Leave League
include
include
include
include
Database
Messaging System

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
REQ1	5	X							
REQ2	5	X							
REQ3	5	X	X						
REQ4	5	X					X		
REQ5	4	X							
REQ6	5								
REQ7	5								
REQ8	5								
REQ9	5								
REQ10	5			X					
REQ11	4		X	X					
REQ12	3		X			X			
REQ13	3						X	X	
REQ14	3				X			X	
REQ15	3				X				
REQ16	2								
REQ17	3								
REQ18	2		X						X
REQ19	3		X						X
REQ20	2								
REQ21	2								
REQ22	1							X	
MAX PW		5	5	5	3	4	5	3	3
Total PW		25	20	10	6	4	9	8	5

Use Case UC-2:	Make Trades

Related Requirements: REQ-10, REQ-15, REQ-16
Initiating Actor: Player
Actor's Goal: Initiate a trade of some sort in a simulated stock market
Participating Actors: Database, Finance API
Preconditions: Player is a registered student
Successful End Condition: Database is updated to reflect portfolio and capital changes
Failed End Condition: Player is notified their trade could not be made and is not charged
Extension Points:

Flow of Events for Main Success Scenario
include::Login (UC-XX)

← 1. System displays an interface for the player to enter a stock ticker, share amount and select a market order type
→ 2. Player makes their selections on the market order screen
← 3. System polls the Finance API for market status
→ 4. Finance API indicates last known market price for ticker and market status
← 5. System writes trade to database, completing it if market is open and user has enough money (charging users account and adding to their portfolio). If market is closed it adds it to a pending trades list, to be completed when the market is open.
← 6. System outputs to display indicating to player the status of their trade.

Flow of Events for Extensions (Alternate Scenarios):

step 5a: User didn't have enough money
← 6. System indicates to user that they had insufficient funds through main display and informs them the market order could not be made.

step 4a: Finance API couldn't be reached
← 6. System emails system administrator informing them of the error. Then indicates to user that stock prices are temporarily unavailable, and ask if they want to leave order pending or cancel it.
→ 7. User chooses one of two options.
← 8. System either cancels order or adds it to pending list on server.
← 9. System indicates to user that it has completed their choice succesfully

Use Case UC-5:	Administration- Manually Add Player to League

Related Requirements: REQ-11
Initiating Actor: League Administrator (LA)
Actor's Goal: Add a player to the league
Participating Actors: Database, Player
Preconditions: Player exists
Player is not in league database
Successful End Condition: Player is added to league database

Flow of Events for Main Success Scenario:
← 1. System notifies league admin of add request and provides the player's profile and ID
→ 2. LA goes to the admin panel of the requested league, inputs player ID into "ADD PLAYER" functionality and submits
← 3. System verifies player ID is valid and sends player a join league request.
→ 4. Player verifies request
← 5. System enrolls player in the specified league, and notifies LA of confirmation

Flow of Events for Extensions (Alternate Scenarios):
2a. LA inputs invalid player ID
← 1. System detects invalid ID and displays an error

2b. LA inputs valid player ID that is already enrolled
← 1. System detects player is already enrolled and displays an error

4a. Player denies request
← 1. System notifies LA of rejected request

Use Case UC-6:	Player requests to join league

Related Requirements: REQ-21
Initiating Actor: Player
Actor's Goal: Automatically enroll in specified league
Participating Actors: Database, League Admin
Preconditions: Player is logged in
Failed password attempts is zero
Successful End Condition: Player is added to league database

Flow of Events for Main Success Scenario:
← 1. System queries user for league ID
→ 2. Player inputs a league ID
← 3. System verifies league ID and queries user for league password
→ 4. Player inputs a league password and confirms action
← 5. System verifies password and enrolls player in league

Flow of Events for Extensions (Alternate Scenarios):
2a. Player enters invalid ID
← 1. System detects invalid ID, displays error, and prompts for re-entry of league ID

2b. Player enters valid ID, however is already enrolled in league
← 1. System detects valid ID and erroneous request. Error message is displayed

4a. Player enters invalid league password
← 1. System detects invalid password and gives the player the following options
- Re-enter league password
- Message League Admin for Administrative Manual Add

← 1a. System detects failed password attempts has reached limit.
- The event and offending player info is logged.

→ 1b. Player re-enters league password, return to main success state 3

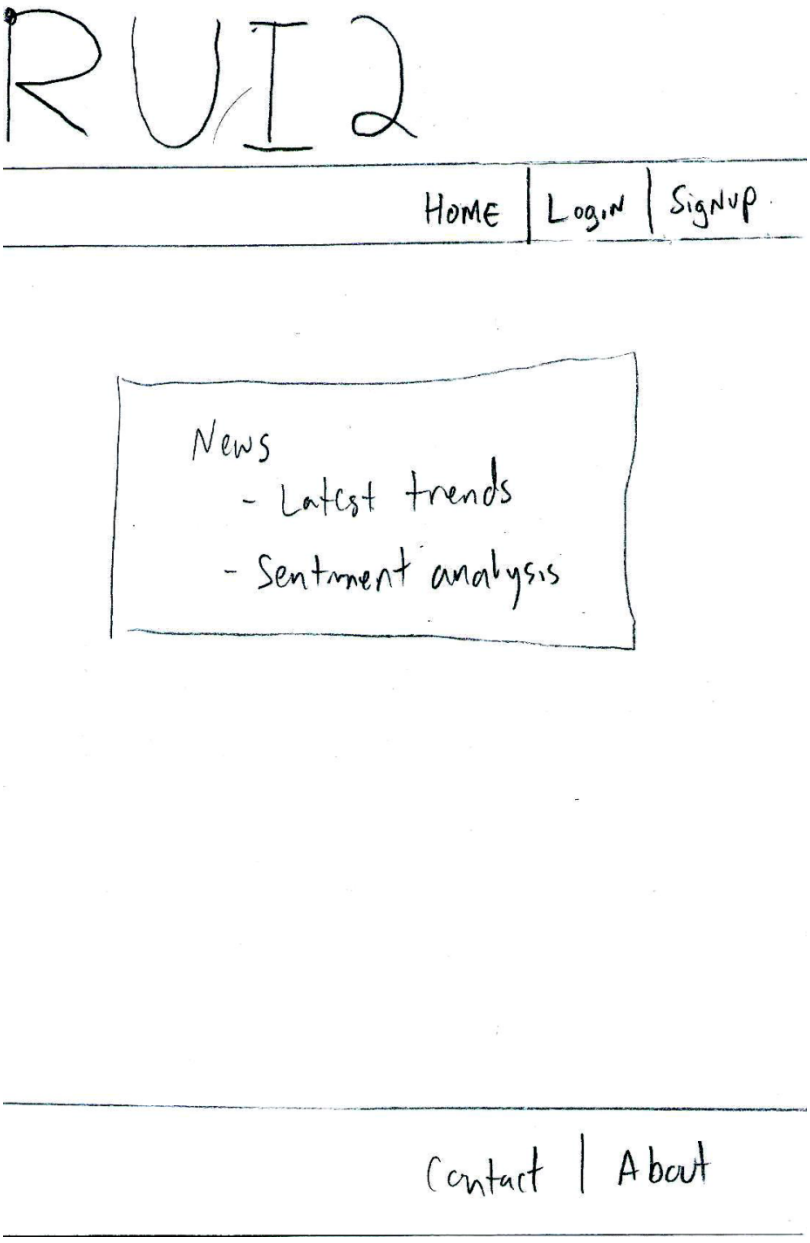
→ 1c. Player request to contact league Admin
← a. System provides player with request form
→ b. Player writes and submits message

c. Refer to Use Case 5

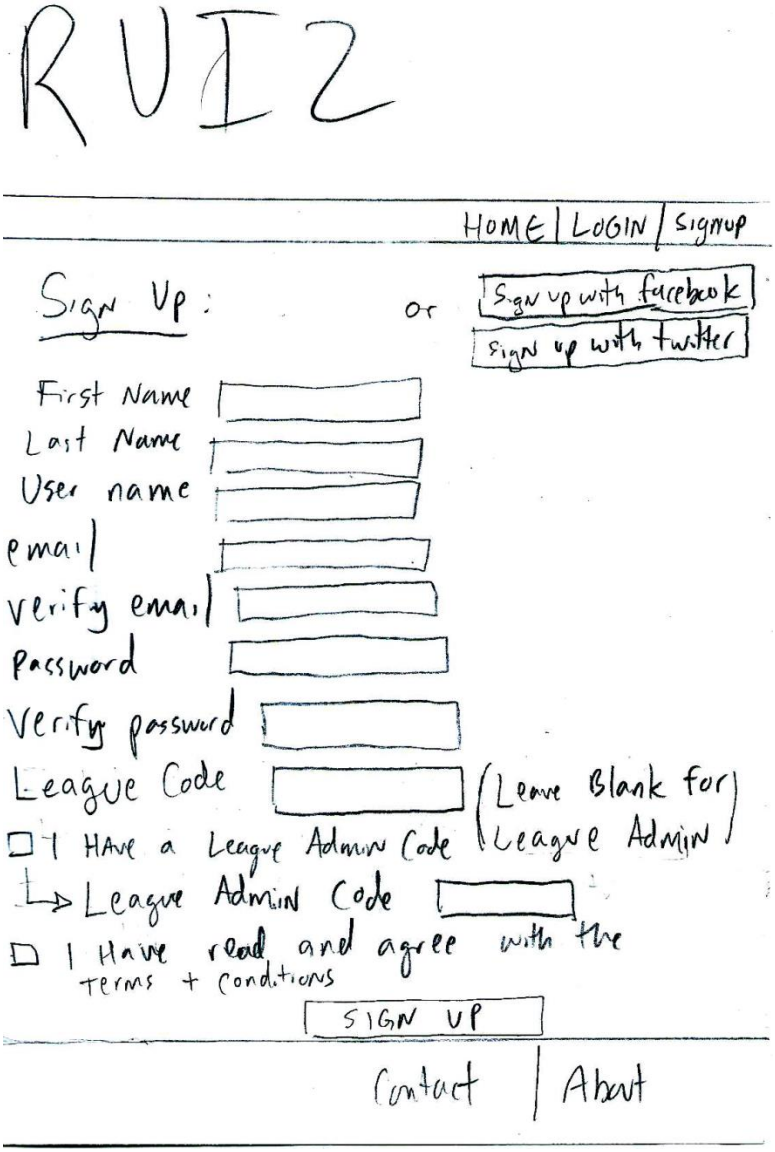
User Interface Specification

Preliminary Design

UC1 - User Registration:



For the first time user, they can click Signup on the welcome page which redirects him/her to the signup page.

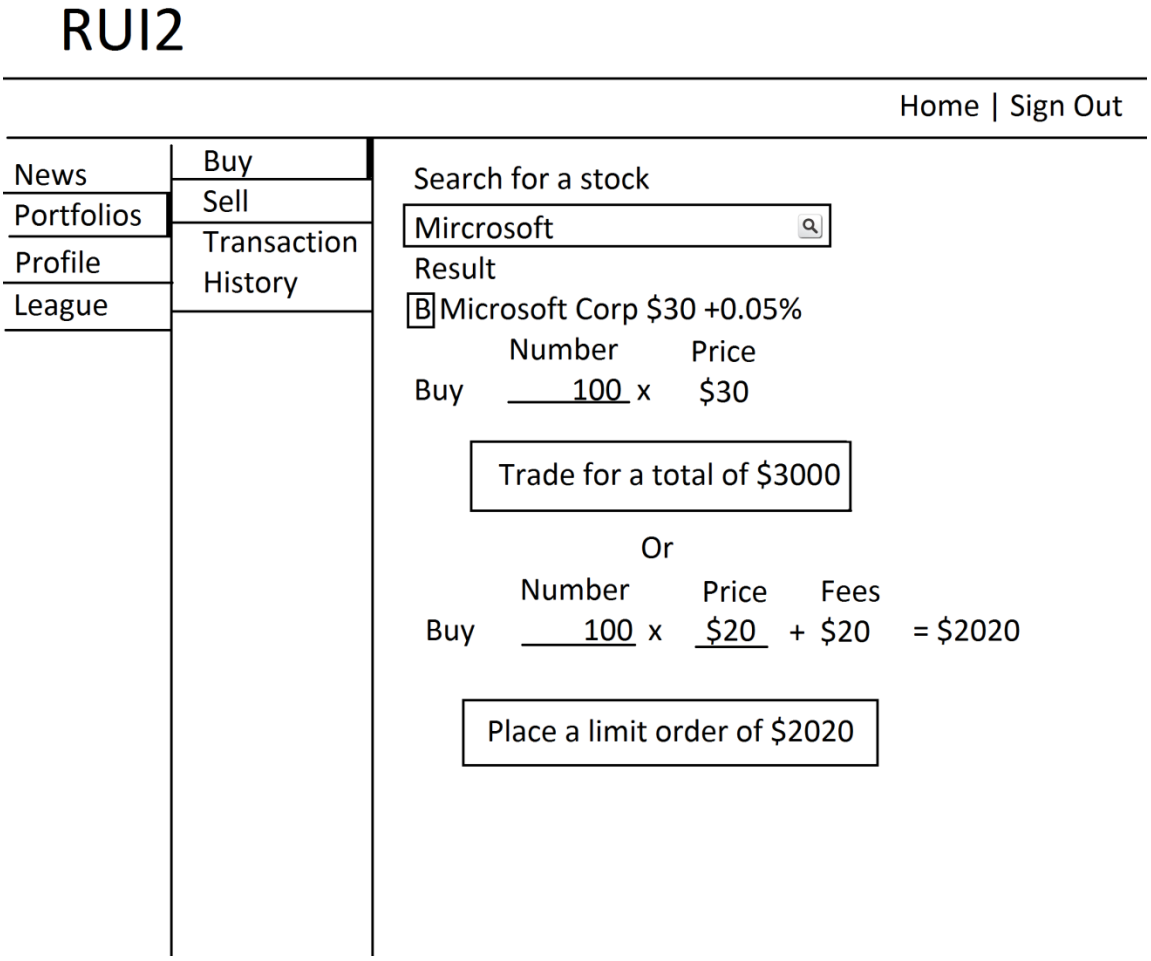


The player enters his/her first and last name, desired user name, email and password (and verify). The player will also be given a league code by the league administrator. The player must click the: I have read and agree with the terms and conditions before enabling the signup button. Alternatively the player can click signup with Facebook or signup with twitter, and the first 7 fields will be swapped with just their Facebook login and password.
When the league administrator signs up, he/she will enter the identical information to a player, but leave the league code blank, and click the box “I have a league admin code”, which will bring up

another text box field ‘League Admin Code’, in which he/she will enter the league administrator code given to them by the system administrator. This will entitle the league administrator to be a super user and have access to the extra administrative options.

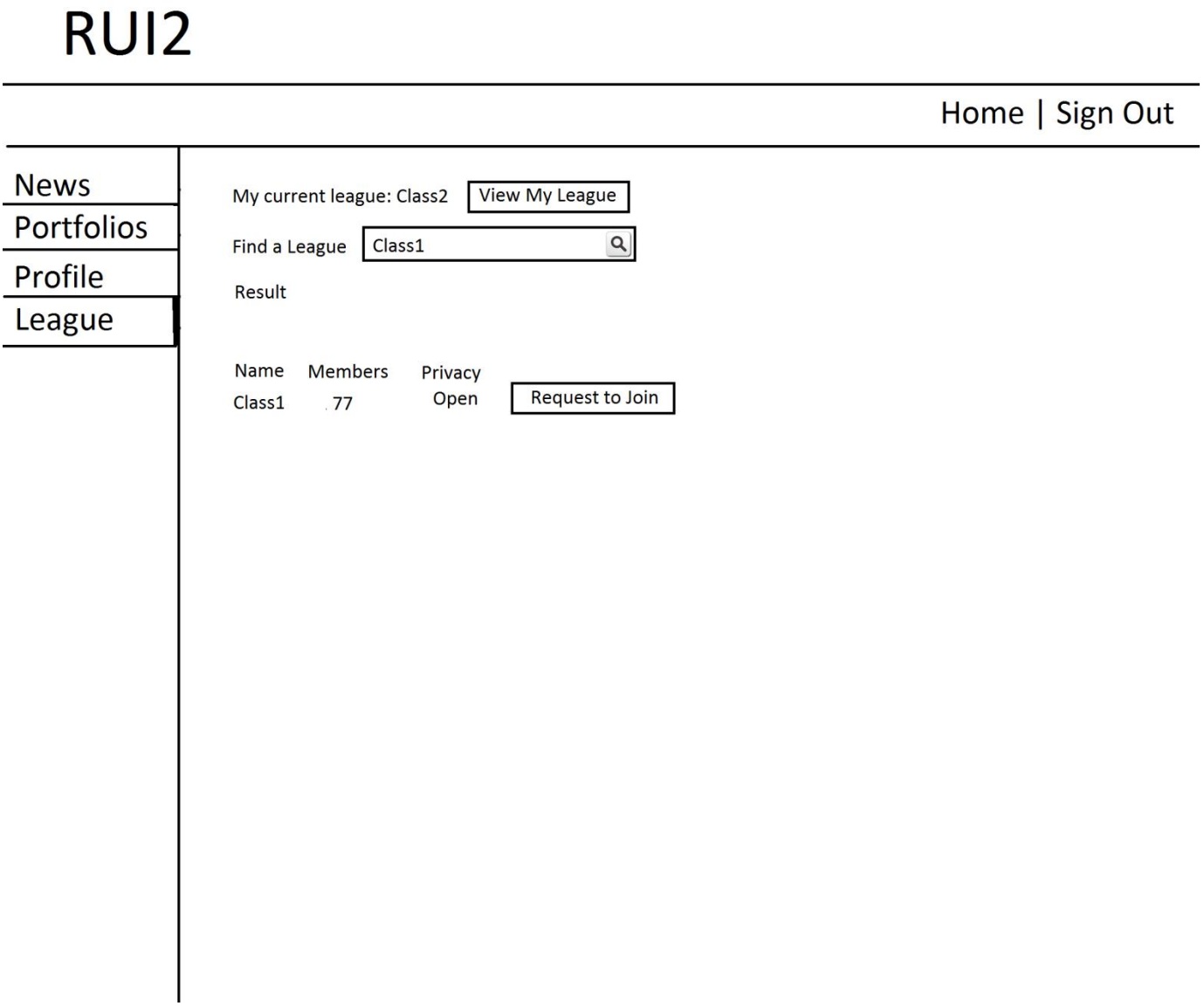
UC-2: Make Trades:

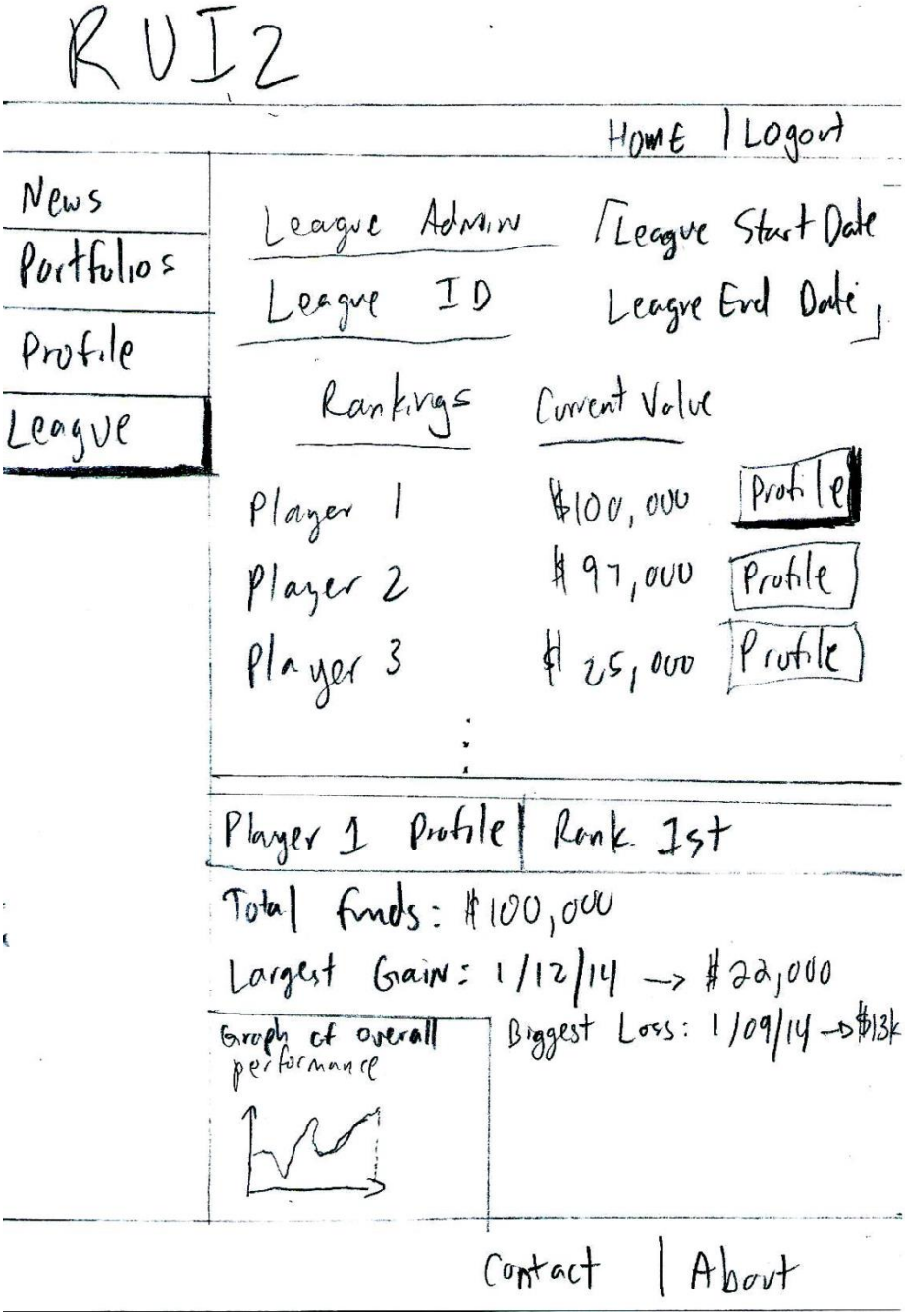
The player can make purchases by selecting PORTFOLIOS/ BUY tab. In the Buying page, they can search for a stock by typing its name into the search box. Clicking the B button on the left of the result will make the program receive the information of the stock. Player will then proceed with the transaction by typing in the amount they want to buy. Finally they can either make an immediate order by clicking “Trade for a total of __” button or select “Place this order as a Stop/Limit order” which they can later view in transaction history.



Similarly for Selling, players can click the “S” button left to an owned stock to sell them. They can trade it immediately with the market price or place a Stop/Limit order with an additional fee

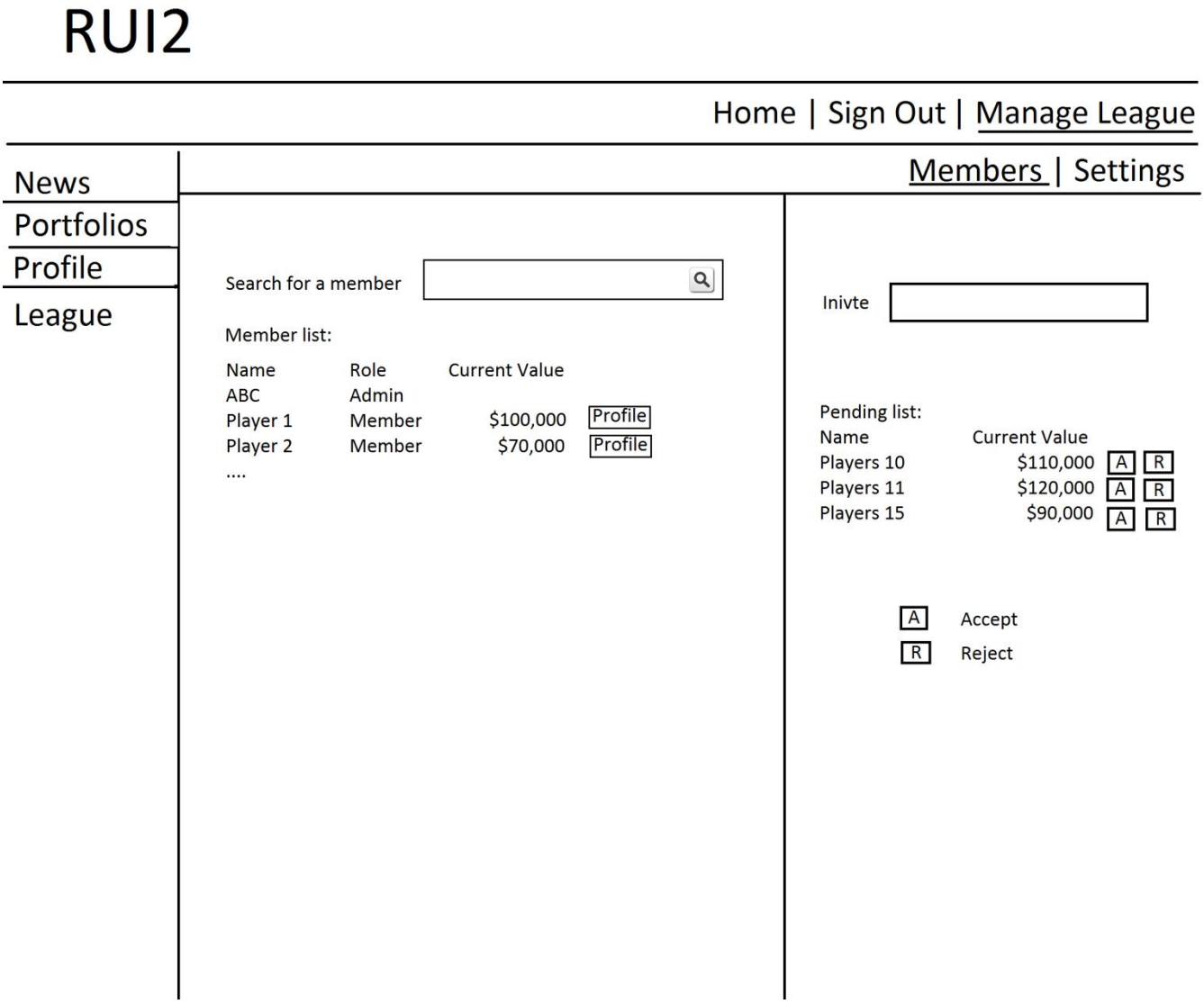
UC-4: Allows all users in a league to view the profiles of all other players in the league.



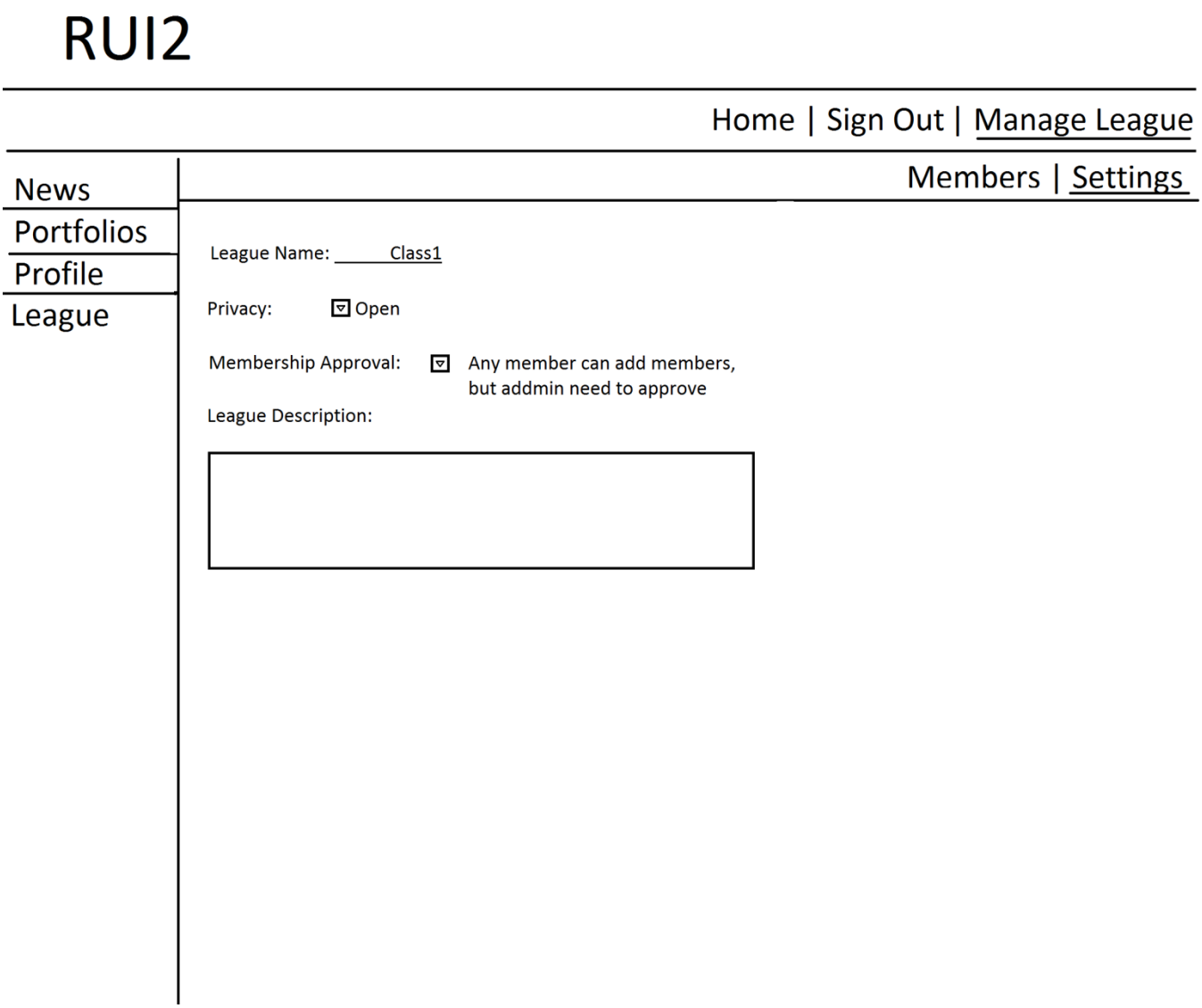


Once the player has logged in they can click the league tab on the left then select “View My League” button, which will bring up the current league admin’s name, the league id, the start and end date of the league, and rankings of all the players in the league. Aligned with each of these players will be a button to view their individual profile. Whichever profile button is selected, that players profile will be displayed underneath the rankings. This profile will contain the players ranking, their total funds, their largest day for gain/loss and a graph of their overall performance for the whole league period. By default the league leader’s profile will be displayed when the league page is first loaded.

UC-3: Setup League -- Allow a league administrator to start a game and initialize settings
Since a league can have a large number of member, say 100 for a class, having a member's profile displayed at the bottom of the page may give the admin a frustrating experience, especially moving back and forth viewing different users' profile(player 30 and player 70 for example). Instead, i suggest that the profile button will lead the user to the player's profile page in which they can see the mentioned profile in details

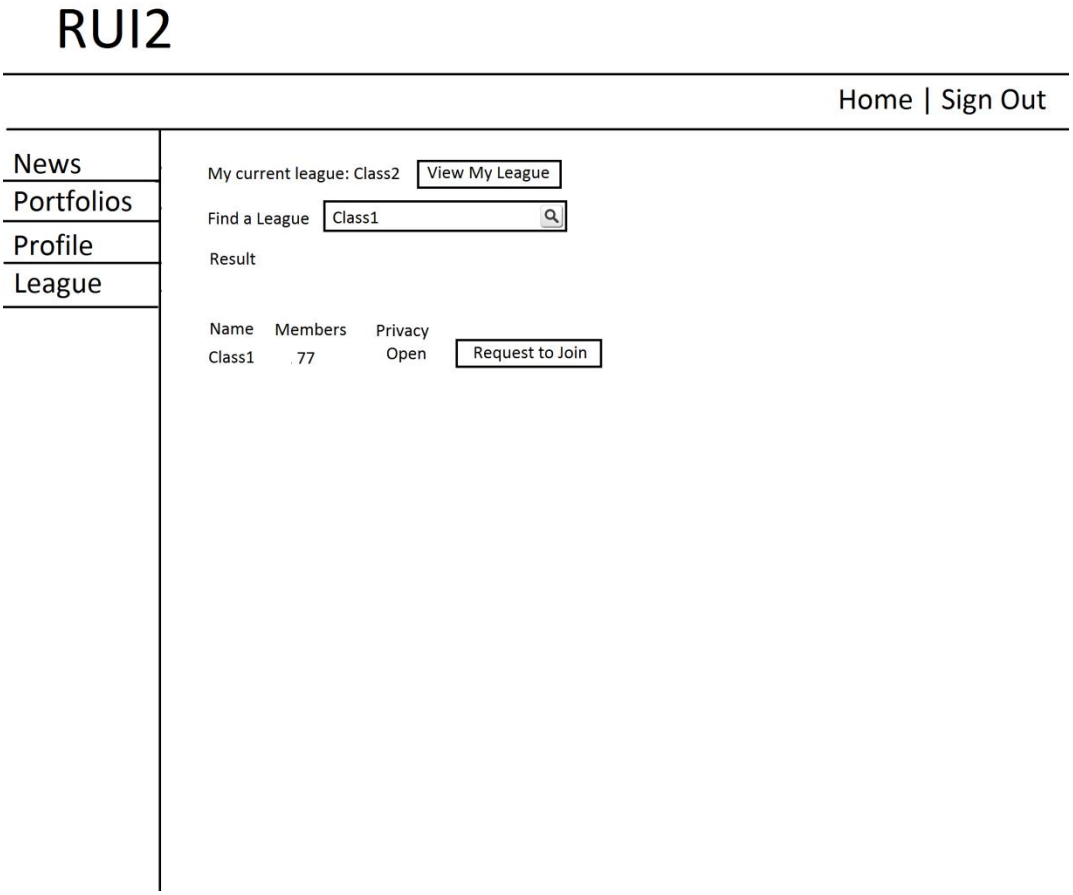


Selecting Manage League/Members , the league admin can get an overview of the league's members, he/she can view any member's profile by click the profile button next to the member on the member's list. The admin can also find a specific player quickly by typing the name into the search box.
The league's admin can also invite people to join the league by typing the player's username into the box next to invite. There is a pending list which list the players that want to join the league, the league admin can either accept or reject the request by click the corresponding button. (this seems to meet UC-5)



And if the league admin wants to change the league's name, privacy, membership approval setting and write a description for the league, he/she can choose the setting tab.

Use Case UC-6: Player requests to join league



Players can also request to join a league by first searching for the league using its name then click the “Request to join” button next to the result to send the request

User Effort Estimation

1. Sign Up: 4 mouse clicks, 86 keystrokes
a. Click Sign Up on the right corner of header
b. Data of Users : 20 keystrokes of first name and last name, 10 keystrokes of user name, 15 keystrokes of email,15 keystrokes of verify email, 11 keystrokes of password, 11 keystrokes of verify password.
c. Click League admin code
d. Input the league admin code, 4 keystrokes
e. Click to agree with terms and conditions
f. Click sign up to be done.

2. Login: 1 click, 21 keystrokes
a. Click Login on the right corner of header
b. 10 keystrokes of user name, 11 keystrokes of password.

3. League Portfolio: 2 clicks, 10 keystrokes
a. Click Portfolio at home page on left side.
b. put amount of shares want to sell/buy
c. click confirm what we did.

4. Setting: 2 clicks,
a. Click Setting at home page on left side
b. Click parts you want to setting.

5. Trade: 2 clicks, 10 keystrokes
a. click the company
b. choose buy/sell
c. put amount of shares you want to buy/sell, 10 keystrokes
d. click confirm

6. Profile: 1 click
a. click profile to see information
b. Edit information

7. Create new League: 2 clicks, 15 keystrokes
a. Click League tab
b. Enter league name
c. Click confirm

Domain Model

Domain Analysis

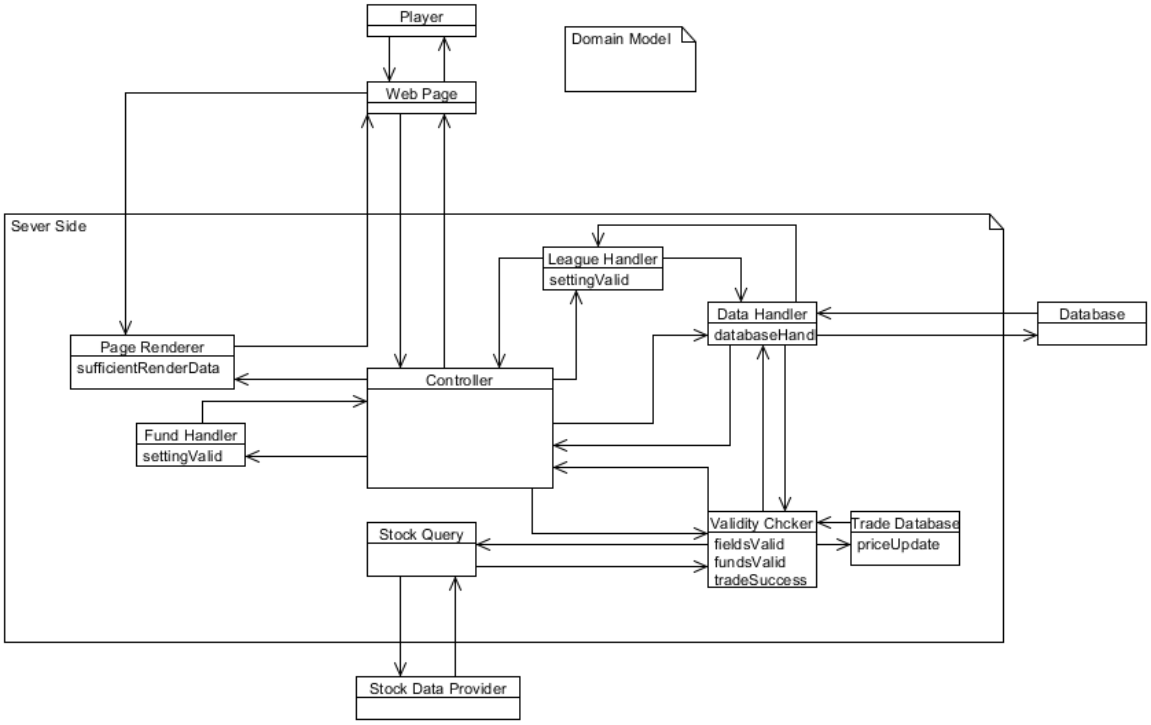


Figure 1: Domain Model

This is our general domain model which shows important objects and their interactions with others.

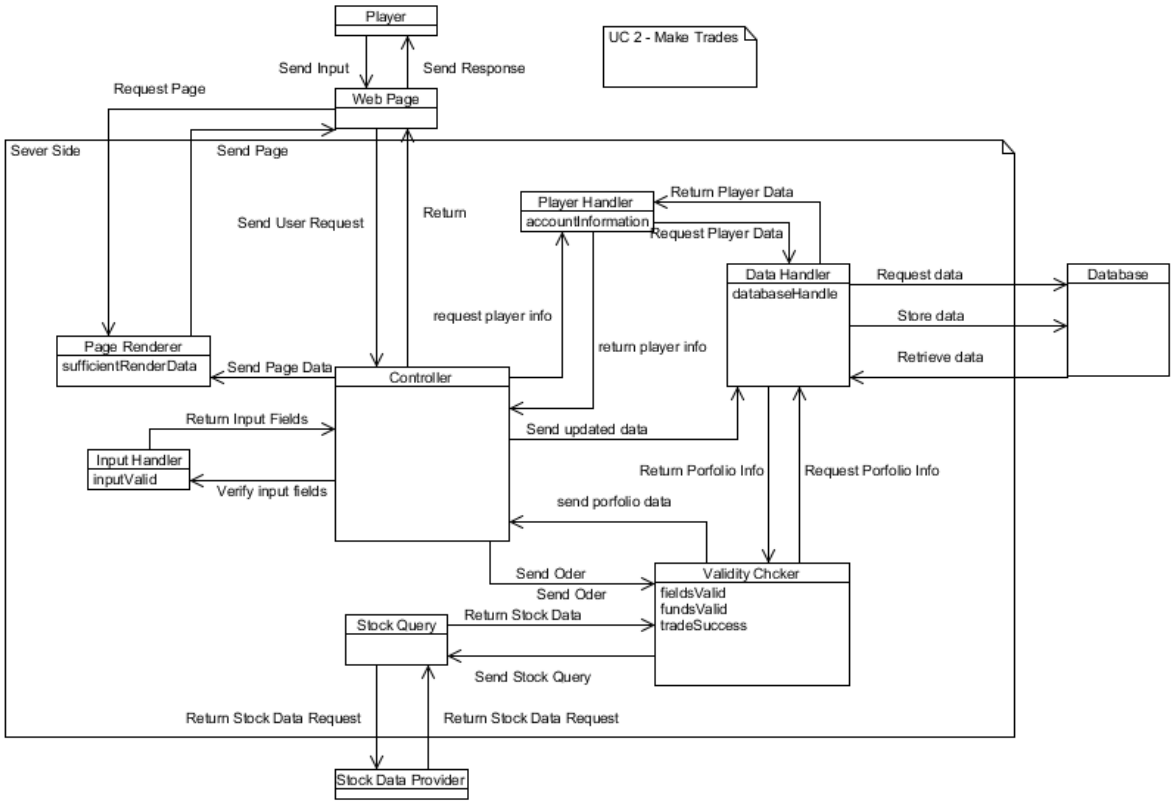


Figure 2 - Make Trade Model

Figure 2 represents our UC-2 Make Trades. When a player make a trade order on the web page, the order is sent to the controller, which directs the request to Validity Checker to check whether the request is valid and the player has enough funds or has enough stocks to sell. If all of above conditions are met, the validity sends a request to Data Handler to make the transaction by updating the player's data. The controller notified by Validity Checker that the transactions are made successfully, then send the updated data about player to Page Renderer to generate updated info displayed on user's web page.

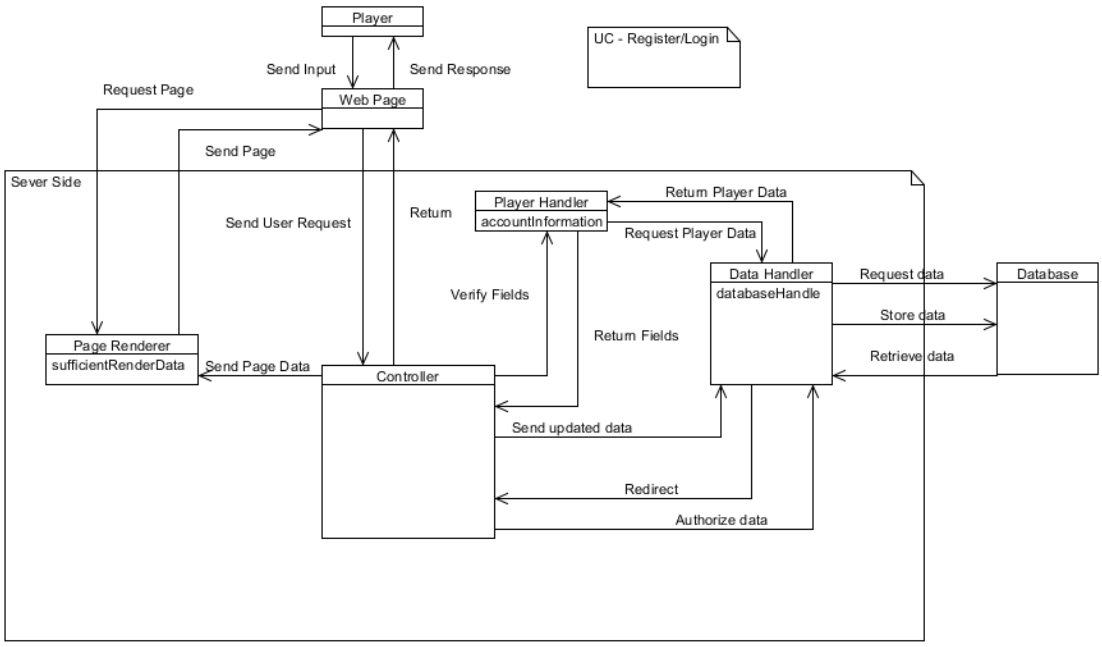


Figure 3 - Register/Login Model

Figure 3 represents UC-1 Register/Login
Register: Player fill in the form and send a register request through web page to the Controller. The Controller conveys the request to Player Handler, whom first verifies player's data validity and availability before sending creating new profile request to data handler. After the data processing processes are finished, the result is sent to the controllers whom relay the data to Page Renderer to generate a page displayed later on Player's web page.

Login: Similar to Register, player fills in form and sends a login request to controller. Controller relay the request to Player Handler to verify username and password. If the player typed in the correct combination, the player is granted access to the system.

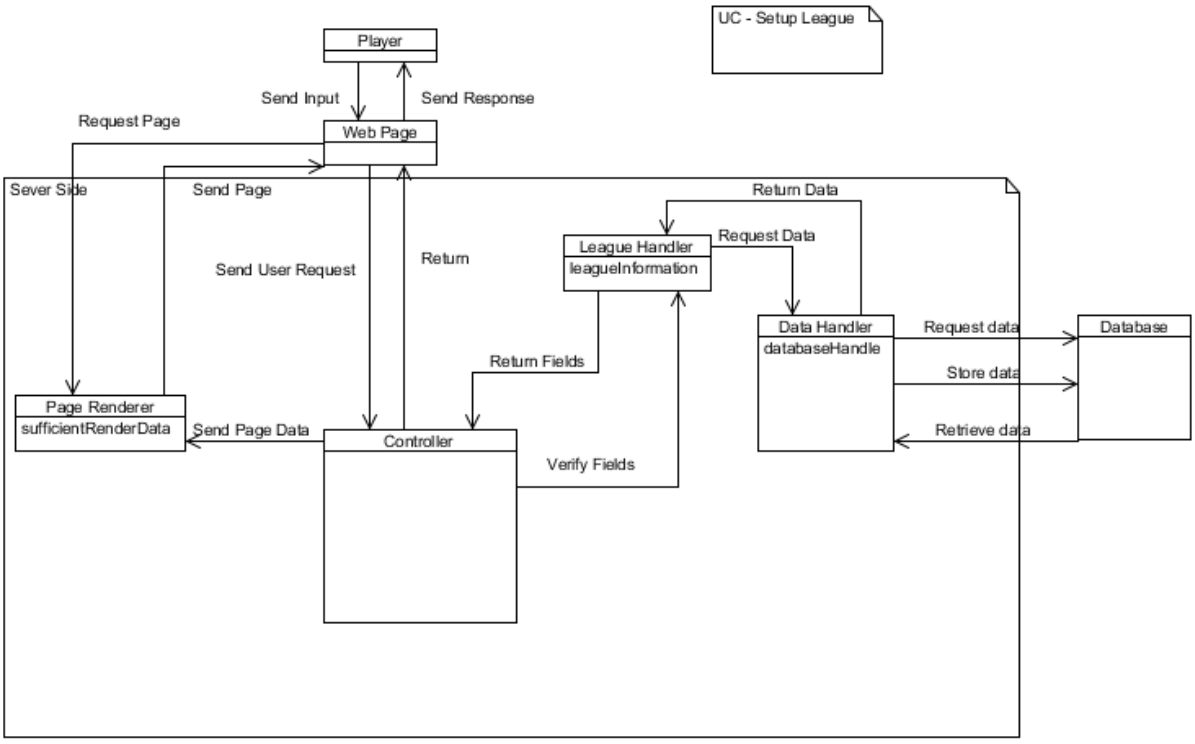


Figure 4 - League Model

Figure 4 represents UC-3 Setup League. To create a new league, the administrator (user) fill out a form then send the form via web page to controller. The controller replays the request to create a new league to League Handler, whom checks the validity) of the info (by comparing with data from database with help of Data Handler, then request an update in data. If the creating new league process is successful, the administrator will receive the updated info on web page with the data generated using page renderer

.

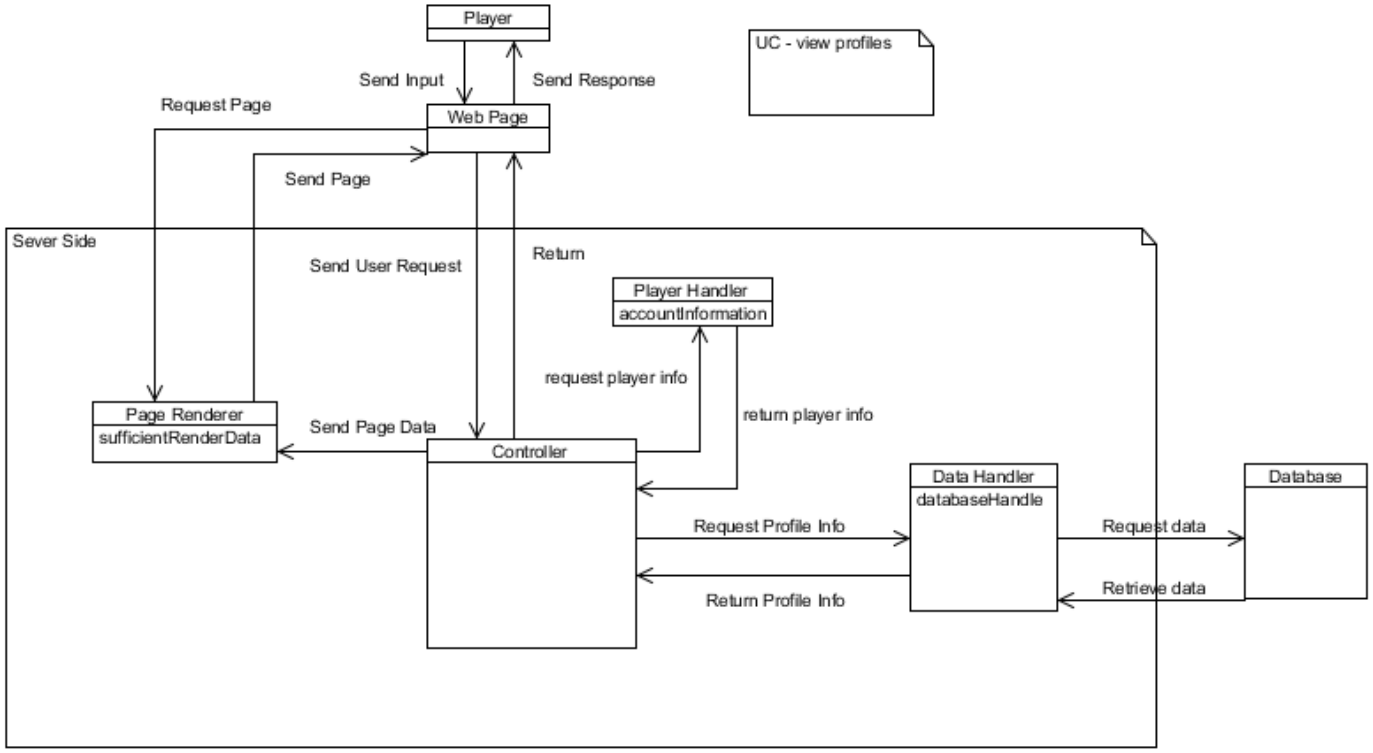


Figure 5 - View Profile Model

Figure 5 represents UC-4: Viewing player's profile. The player first sends the request to the system through Web Page. The request is directed to Controller. The Controller then request profile info from Data Handler. Data Handler conveys the request to Database and receive data back. The data is later sent to Page Renderer to Render the web pages contain for the player.

Concept Definitions

Player:
Definition: Somebody who wants to interact with the system
Responsibilities:

- Research stocks
- Request trades
- Manage Portfolios
- Manage League (League Administrator only)
- Navigate through website
- Request League info/ Player profile

Web Page:
Definition: a web document that is suitable for the World Wide Web and the web browsers.
Responsibilities:

- Receive player requests
- Send requests to Controller
- Send request for pages to PageRenderer
- Receive pages from the PageRenderer
- Pass data to the browsers to display to players

Page Renderer:
Definition: objects that process player's requests and render web pages from data accordingly
Responsibilities:

- Receive page data from the controller
- Process the data into easily viewable format
- Send the results to the Web Page

Controller
Definition: Objects that control the operations of the system base on the player's requests
Responsibilities:

- Receive player's requests from the web page
- Send the page data to the Page Renderer
- Request/Receive Player Data from the Player Handler
- Send/Receive updated data to the Data Handler
- Send the player's input to the Input Handler to verify

Validity Checker
Definition: Objects that test whether the inputs/ requests are valid before send the requests to other objects
Responsibilities:

- Receive requests/ orders from Controller and Verify them
- Send the requests/ orders to Stock query or Database handler depend on the requests/orders

Stock Query
Definition: Objects that fetch real-time stock info along with sentiment results
Responsibilities:

- Receive stock data request from Validity Checker
- Request/Receive stock data from stock data provider

Data Handler
Definition: Objects that deal with data related processes
Responsibilities:

- Receive requests and send data to controllers, League Handler and Player Handler
- Send request, retrieve and update data in database

League Handler
Definition: Objects that handle league related processes
Responsibilities:

- Receive league related requests and verify them
- Send requests to database handler
- Receive data from data handler

Stock Data Provider
Definition: Sources of real-time stock data that is accessible by the system
Responsibilities:

- Receive requests from Stock Query
- Send Data to Stock Query

Association Definitions

Concept Pair	Association Description	Association Name
Web Page ↔ Controller	Web page sends the user request to the controller to be processed and distributed. Controller sends return to web page to signify completion of process.	Send user request, return
Web Page ↔ Page Renderer	When the web page is signalled by the controller that the controller is finished processing the user request, the web page signals the page renderer to request the page. The page renderer sends the rendered page to the web page to be displayed.	Request page, send page
Controller→Page Renderer	Controller sends requests to Page Renderer	Convey requests
Controller ↔ League Handler	The controller sends the data inputted by the user (in this case a league admin) to the league handler to verify the validity (Correct data types, valid dates etc.) and checks that it doesn't conflict with other previous leagues in the database (conflicting league names for example).	Verify fields, Return fields
Controller ↔ Player Handler	controller sends requests for player data,	send request, receive data
Controller ↔ Data Handler	controller sends data requests from players to data handler and receive data back from data handler	send request, receive data
Data Handler ↔ Database	Request and retrieve data from database, store data	Convey requests
Validity Checker ↔ Stock Query	Validity Checker send validated requests to Stock Query and receive stock data back	Request data
Stock Query ↔ Stock Data Provider	Stock Query passess request to Stock Data Provider and receive back stocks data	Convey requests

Attribute Definitions

Concept	**Attribute**	**Meaning**
PageRenderer	sufficientRenderData	Used to determine whether the data retrieved is sufficient to generate web pages
InputHandler	inputValid	Used to determine whether the player's input is valid(no special characters, appropriate length)
DataHandler	databaseHandle	Communicates back and forth between the database to handle requests.
Validity Checker	fieldsValid,fundsValid,tradeSuccess	Used to determine if the input fields are valid, if the player has enough funds to make the purchase and whether the trade is a success.
PlayerHandler	accountInformation	Player name, Role(whether the user is a league administrator), funds, transaction history, performance
LeagueHandler	leagueInformation	League's name, members, status

Traceability Matrix

Use Case	PW	Player	Webpage	Controller	Page Renderer	Input Handler	Stock Query	Stock Data Provider	Player Handler	League Handler	Validity Checker	Data Handler	Database
UC-1	25	X	X	X	X	X			X			X	X
UC-2	20	X	X	X	X	X	X	X	X		X	X	X
UC-3	10	X	X	X	X					X		X	X
UC-4	6	X	X	X	X				X			X	X
UC-5	4	X	X	X	X					X		X	X
UC-6	9	X	X	X	X	X			X			X	X
UC-7	8	X	X	X	X				X			X	X
UC-8	5	X	X	X	X	X	X	X					
Max PW		25	25	25	25	25	20	20	25	10	20	25	25
Total PW		87	87	87	87	59	25	25	68	14	20	82	82

System Operation Contracts:

UC-1: Register:

- Preconditions:

-Players who have code given out by professor/league administration

- postconditions:

- Users have portfolio which contains users' information in the database

Operation: Joining a league

Precondition:

- The league exists
- The player has the league's password

Postcondition:

- Player successfully join in the league

UC-2: Make Trades:

- **Buy stocks**
- Preconditions:

- Users must have enough money to purchase stocks and bonds.
- There are Stocks that is available to purchase
- Transaction date is valid

- postconditions:

- Database has been updated with these changes

- Update Stocks inventory in database.

- **Sell Stocks**
- Preconditions:

- Users must have stocks that are available to sell
-

- postconditions:

- Database is updated with these changes
- Transaction date is valid

UC-3: Setup League :

- Preconditions:

- Input settings are valid

- postconditions:

- league informations are stored in database

Operation: manage league

- Preconditions:

- Users can control league privileges.

- Postconditions:

- League informations are valid in the database.

Operation: invite to League:

- Preconditions:

- Invitee must have account (users ID, league ID)
- invitee has valid portfolio

- Postconditions:

- None

Operation: View transaction:

- Preconditions:

- Initiating players is logged into the system

- Postconditions:

- Display information of players transaction

Operation: Administration and maintain website:

- preconditions:

- Player is league administration
- System administration is log in often

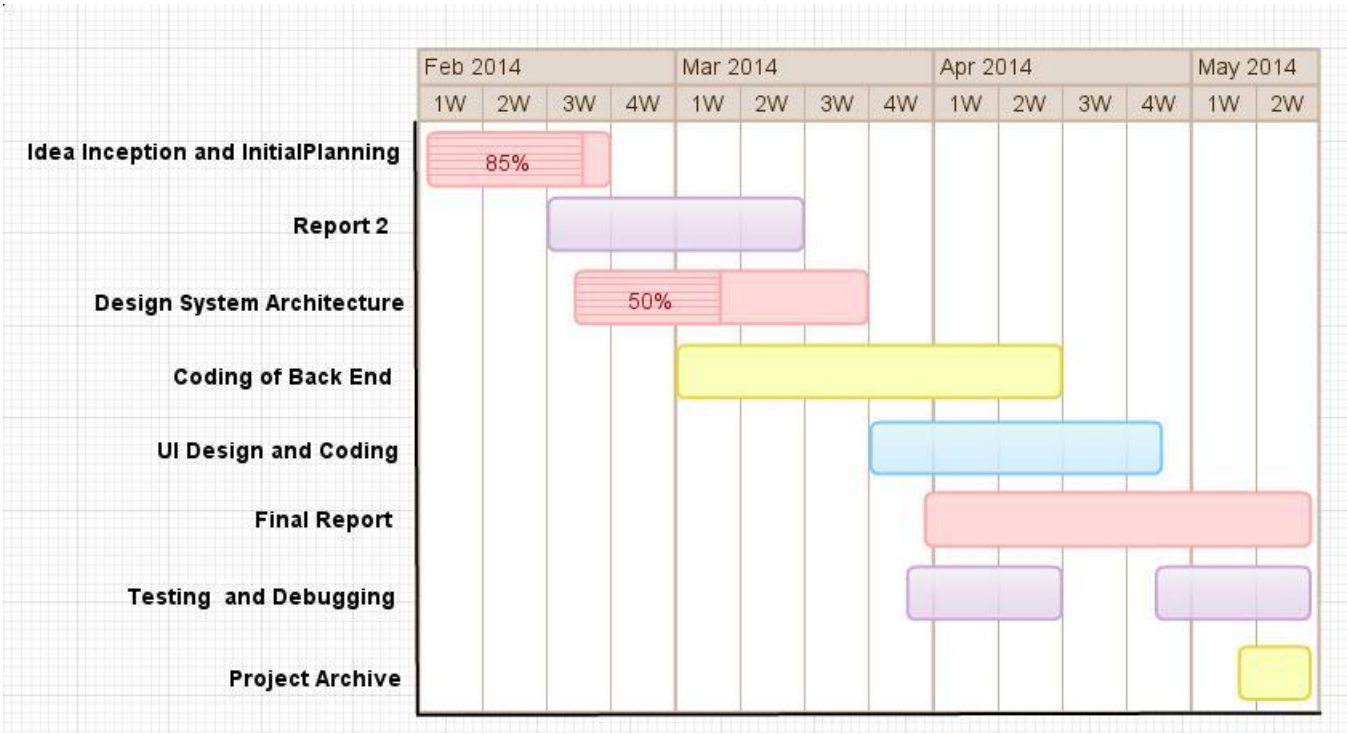
- -Postconditions:

- system administration control what website changing.

UC-4: View other player's profile

- preconditions:
  - Player is logged in
  -The profile of the other player is available for viewing
- postconditions:
  - Display the profile of the other player

Plan of Work



As can be seen in the Gantt chart we have decided that we will begin coding the core backend of our project well before working on the UI. We have made this decision because a pretty UI is useless if a robust system isn't backing it up. If time becomes an issue this will allow us to write a more basic pragmatic UI, as opposed to not having a working system. We have already begun the process of designing the system architecture, as shown in the chart, but will continue to iron out details in the design leading into coding and while coding. The timing we have laid out focus heavily on agile software development, where coding and design are practiced at the same time.

References

http://blog.kaazing.com/2010/02/24/5-signs-you-need-html5-web-sockets-part-2/

http://www.informationweek.com/wall-streets-quest-to-process-data-at-the-speed-of-light/d/d-id/1054287

https://webtide.intalio.com/2011/09/cometd-2-4-0-websocket-benchmarks/

http://spreadnetworks.com/press-releases/10-04-2012-latency-improvements/

http://www.investopedia.com/ - Used for financial term definition

http://www.umlet.com/