The Paramount Investments League

Report 1
Software Engineering
14:332:452

Team 1:

David Patrzeba
Eric Jacob
Evan Arbeitman
Christopher Mancuso
David Karivalis
Jesse Ziegler

February 23, 2014

Hyperlinks:

Webapp Link
Project Repository
Reports Repository

Revision History:

Version No.	Date of Revision
v.1.1	2/7/2014
v.1.2	2/16/2014
v.1.3	2/23/2014

Contents

1 Customer Statement of Requirements

1.1 Problem Statement

The stock market, more specifically the New York Stock Exchange(NYSE) and the Nasdaq play a pivotal role in the American economy today. Both are signals of the strength of the private sector and consumer confidence. It is thus no surprise that more and more people want to be involved in these markets and attempt to increase their own wealth.

There is however a barrier to entry for many people, both young and old in participating. That is why with Paramount Investments League we are interested in a platform for interacting with these markets and providing educational interfaces for breaking down these barriers. Users should be able to easily register with the system and begin participating immediately. They should be given an imaginary cash portfolio where they can perform basic market orders such as buy and sell. These orders to should mimic real market orders as closely as possible and should include a brokers fee. More sophisticated market maneuvers should be unlocked as the user progresses through an achievements ladder.

Paramount Investments League is geared towards a wide array of audiences and expects a variety of users with varying knowledge levels to participate. In order to maintain appeal amongst these users the platform should provide rewards to users for acheiving particular goals. We would like to replicate the idea of achievements or trophies similar to the Microsoft xBox and Sony Playstation family of systems. These achievements can award users with new abilities or additional cash to their portfolio as they rise up the achievements ladder. Users should also be able to create leagues to help further enhance the competitiveness of the game.

Leagues exist to allow multiple users to compete against a subset of the global user base with individual league rules. This allows leagues to set particular goals in order to be declared the winner. Leagues will require a cash buy-in that will be pooled together and distributed to the winner(s) as seen fit by the league creator. To help facilitate these leagues, a leader board will be created for each individual league such that users can see their progress. In addition to league leader boards, mutliple global leaderboards will be available providing specific metrics of comparison.

To help facilitate a better understanding of markets, market metrics should be available to the user through news feeds of companies in their portfolio, interactive charts, and a live ticker of current trades happening on our platform. Users should be able to have granular control of email and social media updates.

The entire experience should be unified across mobile, tablet, and the desktop and combined with the above features provide an enthralling core experience for users to learn about the stock market.

1.2 Glossary of Terms

Achievement – Any set goal reached by an investor. Achievement rewards can be managed by a league manager and may include badges, capital, equity, etc.

Transaction Ticker – Constantly updating scroll of most recent trades across the market. Users can observe market trends from global equities which may or may not already be in their portfolio.

Leaderboard – Global or league based ranking system determined by overall net worth of player.

Security – A tradable asset of any kind. Can include debts, equities, or derivatives. For the purpose of this game, we will be dealing primarily with equities.

Dividend – A payment made by a corporation to its shareholders, generally as a distribution of profit. It is usually distributed as a fixed percent of shareholder value.

Derivative – Any financial contract which derives its value from another asset or index.

- **Option** – Gives the user the option to buy or sell an asset at a specified price on or before a given date. The buyer and seller are both obligated to fulfill the transaction on the given date if the option is taken.

- **Future** - Allows the buyer to buy an asset at its current price and pay for it at that price in the future. A future is generally exchange traded. The buyer and seller are both obligated to fulfill the transaction on the given date if the future is taken.

- **Forward** – Allows the buyer to buy an asset at its current price and pay for it at that price in the future. A forward is a private agreement between buyer and seller not necessarily based around market equity. The buyer and seller are both obligated to fulfill the transaction on the given date if the future is taken.

League – A market simulation with a pre-determined rule set and several investors with a common goal to determine a winner. Goals can vary across leagues as determined by league managers. Investors can choose to opt into a private league, public league, or no league at all.

Portfolio – A detailed account of assets associated with a particular investor in a given league. Portfolios are unique to each user and will contain specific details such as earnings, losses, performance, averages, as well as detailed asset performances of equities within the given portfolio.

League manager – The league manager will have the responsibility of adding and/or removing investors from the league. League managers control settings, and victory conditions for a particular league. League managers maintain their manager status only for the league in which they have created.

Order – An investor must place an order for the purchase or sale of an asset.

Stock – A type of asset that represents equity in a company.

- **Ask Price** – The price at which a trader is willing to sell a stock.
- **Bid Price** – The price a trader is willing to pay for a stock.
- **Bid-Ask Spread** – The bid-ask spread describes the difference in price between the bid and the ask. These two prices are marginally different, but always with the ask being the more expensive of the two. It represents the friction inherent in trading a stock.[1]

Ticker Symbol – an abbreviation used to uniquely identify publicly traded shares of a particular stock on a particular stock market.

Symbol List – a list of a market/several market's ticker symbols.

Market Order – Any order placed for immediate market transaction.

- **Buy** – User has elected to purchase a particular stock and has placed a bid for that stock.
- **Sell** – User has elected to sell a particular stock and has posted an ask price for it.
- **Short** – Typically used by an investor who expects the value of a stock to decrease. The investors borrows shares of a particular stock and sells them at market price. The investor is responsible for the increased value as well should the stock's value increase.[2]

Limit - An investment which will only take place at a given price. An investor placing a buy limit will place a maximum amount the pay and an investor place a sell limit will place a minimum value for which the stock be sold. Limit orders are not guaranteed to ever process, and only do when the particular limit is reached.[3]

Stop – Orders which are activated if a particular stop falls below or rises above a particular price. It is used to minimize gains and losses for the investor.[4]

Share – A small percentage of a given company which can purchased or sold from other traders.

2 System Requirements

2.1 User Stories

The user stories written and elaborated below demonstrate several particular instances and requirements for program functionality, as well as a weight to measure relative importance of each requirement. In particular these functions are not necessarily written in order of particular weight or functional precedence but are simply a list of end user story requirements and relative weighted importance. It is important to observe that these cases will be elaborated on and referenced in further sections of this document. The following are told from the perspective of the user from his or her view with the intention of fully encapsulating what he or she should expect to be able to see or do upon entering and regularly using the referenced software.

Identifier	User Story	Weight
ST-1	As a user, I can create an account without registering with the website in order to participate in Paramount Investment League.	10 pts
ST-2	As a user, I can access the application across multiple platform paradigms so that I may continue to participate when I don't have access to a desktop computer.	10 pts
ST-3	As a user, I can join or create leagues with self-selected goals so that I may compete with others in a simulated stock market environment based on near real-time stock data.	10 pts
ST-4	As a user, I can search for companies by stock symbol and be presented with their current financial information so that I may research future investments.	6 pts
ST-5	As a user, I can browse a companies profile and view the performance data over a configurable span of time so that I may determine whether or not I want to invest in them.	6 pts
ST-6	As a user, I can buy or sell stocks so I may build my portfolio.	10 pts

ST-7	As a user, I can earn badges(achievements) that reward me with additional capital or new features for accomplishing predefined tasks.	10 pts
ST-8	As a user, I can manage my portfolio within a league to track my investments.	8 pts
ST-9	As a user, I can visually track my finances via graphs and charts so I may more easily manage my portfolio.	4 pts
ST-10	As a user new to the stock market, I will have access to an educational interface that teaches me about the stock market via pop-up dialogues.	6 pts
ST-11	As a user, I can see trades being made by all other users in real-time via a stock-ticker like marquee so I may have a quick overview of current trends.	3 pts
ST-12	As a user, I can see the performance of other users' portfolios so I may observe the investment habits of others.	2 pts
ST-13	As a user, I can view a portfolio leader board so I may have a summary of relative performance between users in my league.	1 pt
ST-14	As a user, I can opt to receive periodic e-mail notifications of my stock performance or trades so I may be kept up to date even when not actively viewing the site.	3 pts
ST-15	As a user, I can additionally link my account with social media sites so I may share my fantasy league experience with friends.	1 pt
ST-16	As a league manager, I can add league rules, a league name, and a league logo to personalize my league.	8 pts
ST-17	As a league manager, I may invit who I want to join.	8 pts
ST-18	As a league manager, I can create league announcements.	4 pts
ST-19	As a site administrator, I can view reports of and delete leagues that are inactive.	2 pts
ST-20	As a site administrator, I may post front page news or announcements.	3 pts
ST-21	As a site administrator, I may have access to a user count, number of active leagues, total leagues, quantity of daily transactions, the most/least popular stocks, and newly created so I may have reliable site statistics.	9 pts

ST-22	As a league manager, I can choose the specific victory conditions for a particular game (eg: first to a certain capital, net gain, or overall gain within a time). As a user I can view this condition and my progress toward victory.	5 pts

The above requirements outline a general list of requirements which we expect to reflect the core functionality of our software (with higher weighted items acting as higher priority and being implemented first). The ultimate goal of the software is to simulate that of a real world stock market with users having the options to perform and carry out the important and basic trading actions (see ST-6). We plan to add increased functionality when compared to years prior, however. With the addition of achievements, varied victory conditions, as well as increased leaderboard functionality Paramount Investments will appeal to a larger audience than that of years past (see ST-7, ST-13, ST-22). Notice that items such as administrative privileges as well as league creation and stock execution are prioritized with substantially higher priority with relation to our newly added functionality. This is because the core functionality of the software is absolutely crucial to it working. We will expand on the core as well Supplement requirements.

Core requirements

These requirements are absolutely crucial to the viability and progression of the software. That is the user can create and log into an account on a daily basis. We will use a basic authentication system to implement this. Importantly the user will be able to access this UI on multiple different platforms to ensure complete and smooth transitional access to the system with zero down time. (ST-1, ST-2)

The user will be able to access his or portfolio. (ST-6) From this portfolio, they can view their currently owned stocks as well as monitor the performance of their portfolio. They can view progress toward goal requirements and badges. (ST-8) From this location they can take action to buy and sell stock or perform short, stop, limits, etc.

League managers will have access to a specific configuration setup where they can choose victory conditions, league settings, and monitor progress of investors within the league. This functionality is core to the formation of leagues within the game. (ST-3, ST-16)

This project will NOT be its own market. In order to maintain the idea of perfect competition and unbiased market prices, all data will be taken from Yahoo! Finance to submit data and trades will be taken from here. This software is not intended to be a way for people to trade actual stock, rather just a resource for learning the market and tools of trade.

Supplemental Requirements

The user will be able to access social media integrated applications, and decide whether or not to keep their social media profile updated and informed with updates on progress from their fantasy league. (ST-15) They may also receive email updates with various progressions in the game (ST-14)

The user will be kept updated on the progress of other users to view their trades as well as recent market trades and trends. (ST-12, ST-11)

Users will also have access to on-site term explanation similar to that seen on Wikipedia. That is, they may scroll over an underlined term to find a brief definition and additional resources. (ST-10)

2.2 Nonfunctional Requirements

Functionality

Additional features for security will be enabled through the use of a OpenID and OAuth through a third-party library. There exists several packages for the purpose of authentication and authorization of users. Key authentication features are the ability to encrypt and store passwords, provide recovery options for users that have forgotten their password, and store a cookie to validate the session.

Usability

A key point in the design of this application is ease of use and appeal to the users. The application should be interactive, informative and consistent across multiple platform paradigms. Additionally the application will be used to provide the educational interfaces noted in ST-9 which should be able to be toggled on and off so that users can always view the information again.

Reliability

In order to ensure that there is no confusion to the user in the case of the internet or server failure, all transactions end with a final confirmation, and no changes to the account are made until after this confirmation. The user's portfolio will thus always be in a consistent state and will be restored when the user is able to log back in. A user that leaves the application and returns later will still be logged in. Server failure should also be dealt with by keeping backups of user data. Proper care should also be taken to handle a situation where a particular stock source is not available (i.e. Yahoo Finance).

Performance

In order to have a great performance, the website should be as lightweight as possible by keeping hardware demands to a minimum on both the client and server sides. For it to be efficient, any task initiated by the user should be completed in a timely manner. The web server should be able to serve concurrent requests especially when a large number of users are logged in. Any frameworks used should be lightweight but consideration should be taken not to prematurely optimize.

Supportability

It should be feasible to extend pr update any server components and include improved versions of modules which can be installed only by administrators. For scaling purposes, it should be made easy to include an additional number of servers to achieve load balancing. The system should be platform independent so that it is easy to move to newer technologies or the next versions of web server. The system itself should also be backed up to a remote server for the sole purpose of extending functionality and testing new features in a controlled environment.

2.3 On-Screen Appearance Requirements

There are a few on screen requirements that will be universal to the entire site:

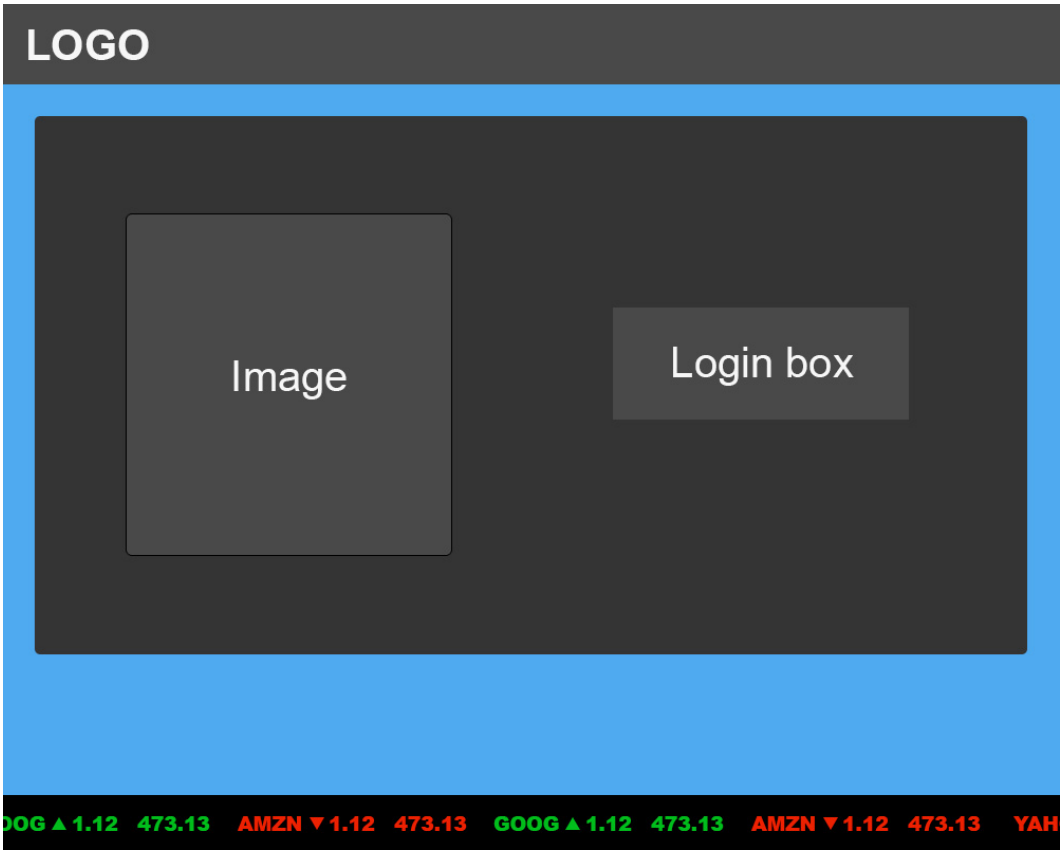

Figure 2.1: Basic on screen requirements of login page

Identifier	Requirement
OSR-1	Every page has a scrolling ticker across the bottom of the page to update the user on stock movement.
OSR-2	Every page, with the exception of the login page, will have navigational links across the top, the user's username and their current position in the leaderboard.

There are also the following requirements for specific pages:

Identifier	Requirement
OSR-3	A custom 404 not found page will be displayed to a user when they try to access a URL/URI that doesn't exist or is not designed for them to be accessing.
OSR-4	On the portfolio page users will find currently owned stocks, charts and graphs, trade transcations, and a news feed.
OSR-5	The leaderboard view will contain users ranked by the top networth from their respective portfolios.

3 Functional Requirements Specification

3.1 Stakeholders

The target demographic for the software described tends to be centered on students and first time investors. That being said, it is likely to see the software expand to take a large role in both university and pre-university classrooms, as a means of teaching general financial concepts. It would not be unlikely to see the game further expand to a larger range of users than other similar software due to increased functionality, addition of achievement and leaderboards, and ability to join with or without league functionality. Specifically, the addition of achievements leaves the user with the desire to return and spend additional time trading on the software.

The league will be a free service with the intention of eventually moving to a subtle-advertisement platform which will have no impact on the user. Once a substantial enough user-base is generated, it will not be unlikely to see advertisements begin to commence in order to bring revenue to the company. As a free service (with eventual advertisements) we expect the platform to attract the greatest number of users, and due to increased functionality, keep said users on the platform for the greatest amount of time. The software is targeted not only at students and potential investors, but at nearly everyone who desires to gain a greater understanding of the financial industry as well as those who would simply like to practice trading before executing in the real market.

3.2 Actors and Goals

Guest

A visitor to the website who has either not logged in or just a simple visitor

- Register and create an account using OpenID/OAuth2
- View the latest trades

Investor

A user who has an account in our servers and is logged in to their account

- Research the latest updates in the market
- View their portfolio

- Execute orders of any kind
- Join/create a league
- Take part in competitions

League Administrator

Manages the leagues that they have created

- Can set league to be public/private
- Set the rules for the league

Database System

Holds the information for the accounts of all users

- Insert information as accounts are created
- Push data back to views about users/events
- Store new data about about users/events

Financial API

Gives the stocks in our database up to date prices

- Fetch real world information and update our database accordingly

Site Administrator

Manages the overall website

- Ensure fair competition between leagues/players

Browser

The middleman between user and system

- Present data to the user
- Retrieve data from the user

Yahoo! Finance

The unit that knows about current financial statistics

- Retrieve data about stocks

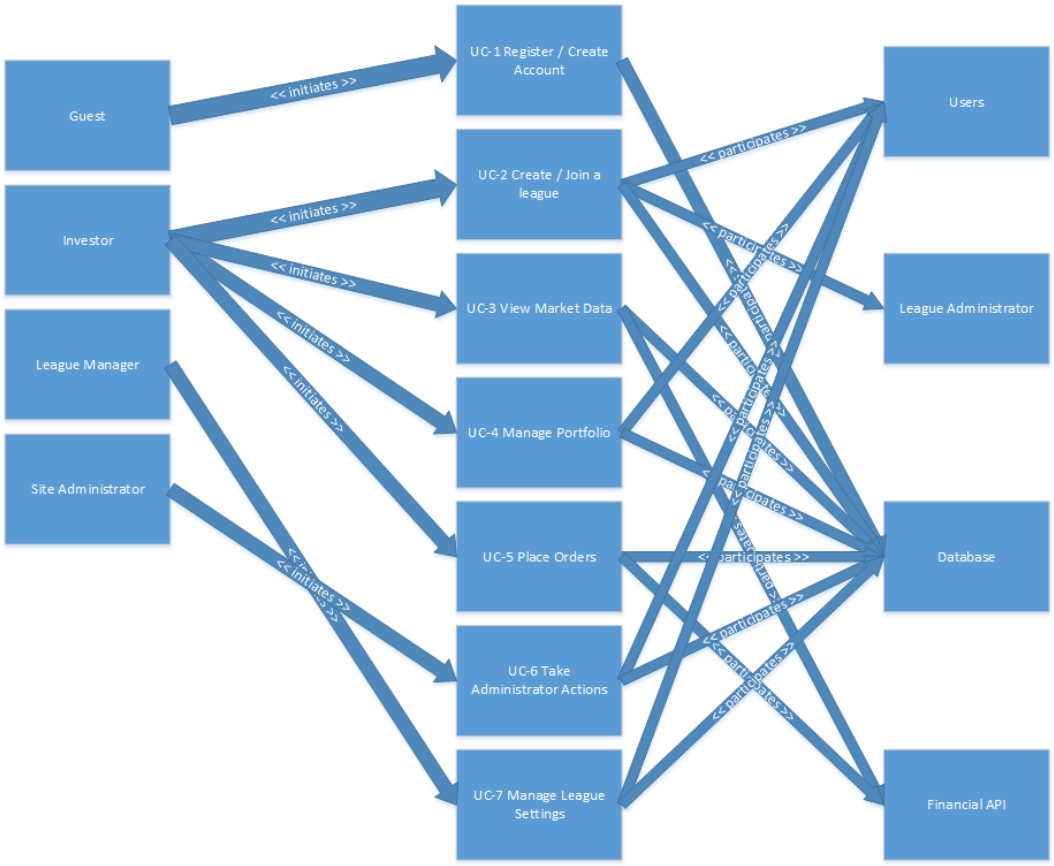

Figure 3.1: This graphic illustrates the relationships between the core actors of our platform.

Queueing System

A subsystem for scheduling orders so as not to block user interactions.

- Place orders to be executed or canceled asynchronously
- Schedule events and mailings for system

3.3 Use Cases

Preface

Users will have instant access to the functionality of the site as soon as they have created an account and logged in. That is, they can perform all of the functions that an investor can perform. They will also have the ability to choose to create or join a league, though they are not required to in order to experience the full functionality of the software. This will give our software a broader demographic when compared to that of years prior. That being said, league creation and administrative user delegation will be an important part of the functionality of the program and will be described its own use case. Core functionalities of respect user types will be elaborated below.

Fully-Dressed Use Cases

Having an account within our database is necessary for the user to experience functionality within the software. That being said, the user can create an account in different ways. They can choose to have the information imported by logging in through any OpenID/OAuth service service (eg: google, facebook, twitter, etc.) This will populate the database fields automatically with data driven from the external resources. Once the user has created an account, they can log in through OpenID/OAuth and will generally remain logged into the system as long as they remain logged into their service of their choice.

CG-BP01 A user can create an account such to access the full features of the game. They can return and log in with minimal burden and access all of the information previously stored.

Use Case UC-1	**Register/Create an account using OpenID/OAuth2**
Related Requirements:	ST-1, ST-2
Initiating Actor:	Guest
Actor's Goal:	Register with our servers
Participating Actors:	Guest, Database
Preconditions:	-Guest must not be a registered user
Postconditions:	-The **Database** is updated with guests information and logs the guess in as an **Investor**
	Flow of Events for Main Success Scenario:
→	1. **Guest** navigates to Paramount Investment League and logs in
←	2. **System** checks **database** for **investor** and isn't found
←	3. **System** retrieves OpenID/OAuth info and registers guest in **database** as an **investor**
→	4. **System** sends out confimation to user and displays starter portfolio
	Flow of Events for Alternate Scenarios:
→	1. **Investor** attempts to login
←	2. **System** checks **Database** and finds **investor**
←	3. **System** loads **investor** info from **database**
→	4. **System** displays users portfolio

Any user has the option at any time to create or join a league. The user who has requested to create a league will have elevated privileges versus a standard user. The league manager will be prompted to make a league with various setting options for victory condition, badges, achievements, etc. The league manager can also choose whether or not to make the league private. A private league will restrict users to those invited by the league manager, and will require a password to join. After initial setup, league managers will have minimal access to settings. That is, halfway through a league, the manager cannot decide to change the victory conditions. This prevents the

league manager from abusing power to tip the scales in his or her favor.

CG-BP02 A user can create a league or join a league if he or she desires. Note that users may trade without a league, but has access to create or join a new league at any time. Leagues may be private or public.

Use Case UC-2	**Create/Join a League**
Related Requirements:	ST-3, ST-8, ST-16, ST-17, ST-18
Initiating Actor:	Investor
Actor's Goal:	Create or join a league to compete in
Participating Actors:	Database, other Investors
Preconditions:	-Investor is logged in
	-league is not created or user hasn't joined league
Postconditions:	-The league is created with the appropriate settings or
	-The **Investor** has joined the league
	-The **Database** has been updated
	Flow of Events for Main Success Scenario:
→	1. **Investor** navigates to and clicks on the create league dialogue.
→	2. **System** displays to the **Investor** the available options for creating a league.
→	3. **Investor** updates the settings, such as privacy, league name, number of spots, and managing users
←	4. **System** sends the updated settings to the **Database**
→	5. **System** sends confimation to the **Investor**
	Flow of Events for Alternate Scenarios:
3a. The **Investor** selects league settings that are disallowed, such as a league name that already exists.	
→	4. **System** informs user what settings are incorrect.
Investor wishes to join a league	
→	1. **Investor** navigates to league listing.
←	2. **System** updates **database** with **investors** info.
→	3. **System** confirms **investor** as part of league and displays league site.

Users can view raw market data which will be pulled from Yahoo Finance in near real time. Users can view company data either on their own portfolio page or through the companys specific info page. That is, the user can view detailed information of each stock or company before committing to a trade from a variety of sources. Users will be able to compare their portfolios performance to typical market trends from the Nasdaq, S&P 500, and DJIA. There will also be a

stock ticker ribbon on the bottom of the screen for users to receive constant real time feeds of most recent trades happening within the market.

CG-BP03 A user can view market data in near-real time. They will have access to data taken from the Yahoo! Finance API and can view this data in their portfolio or by searching for stocks using tickers.

Use Case UC-3	**View Market Data**
Related Requirements:	ST-4, ST-5, ST-10, ST-11
Initiating Actor:	Investor
Actor's Goal:	View the latest information about stocks, companies, and trades
Participating Actors:	Database, Yahoo! Finance
Preconditions:	-Yahoo! Finance is accepting inquiries
	-Investor is logged in
Postconditions:	-None worth mentioning
	Flow of Events for Main Success Scenario:
→	1. **Investor** searches for a market term
←	2. **System** sends request to **database**
→	3. **System** returns suggested terms
→	4. **Investor** selects a term from suggested terms list and sends request
←	5. **System** sends request to **Yahoo! Finance**
←	6. **database** is updated.
	Flow of Events for Alternate Scenarios:
Search Fails	
←	6. **Yahoo! Finance** returns no results
→	7. **System** informs **investor** of search failure

User will be able to view all major items within their portfolio as well as place trades from their portfolio page. From this page, a user can view detailed analysis and graphs of each of their respective stocks as well as their current rank within their league (if applicable) and globally. Users will also be able to place trades for respective companies through their portfolio page. Users can buy, sell, short, or carry out any additional action on any stock or security within the limits of their finances and league settings through this page (See UC-5). Users will also be able to customize and change views as well as add stock index comparisons to monitor their success vs market success.

CG-BP04 All users will have a portfolio which will house detailed information regarding all of their stocks and performance. They will be able to perform trades from this location and will also have access to the core functionality of the site through this page. It will essentially be the users home page.

Use Case UC-4	Manage Portfolio
Related Requirements:	ST-8, ST-9, ST-10, ST-12, ST-13, ST-14
Initiating Actor:	Investor
Actor's Goal:	Manage portfolio by viewing current standings/stocks/securities
Participating Actors:	Database, Yahoo! Finance
Preconditions:	-Yahoo! Finance is accepting inquiries
	-User is logged in
Postconditions:	-**Investor**'s portfolio is updated to reflect change in position
	Flow of Events for Main Success Scenario:
→	1. **Investor** navigates to their portfolio
←	2. **System** requests users portfolio from **database**
→	3. **System** displays portfolio to **investor**
→	4. **Investor** adjusts their portfolio
←	5. **System** updates the **database**
→	6. **System** displays confirmation of portfolio update

User should be able to place trades from various locations. That is, they may place it through their portfolio by typing in the ticker and quantity of shares. They may also navigate to a certain companys page and elect to purchase shares there. Selling shares should be done through the users portfolio where they may see the exact quantity of shares of each respective companies they own. Error messages will be thrown and orders not processed should a user request to buy more shares of a company the he or she can afford or the user attempts to sell more than he or she has. Main transactions will occur through the users portfolio.

CG-BP05 A user can place any market order he or she wishes to place (I.E. buy, sell, short, limit, stop, etc). These trades can be executed from multiple locations and at the prohibitions of the league rules.

Use Case UC-5	Place a Market Order
Related Requirements:	ST-6, ST-11
Initiating Actor:	Investor
Actor's Goal:	Place orders to buy/sell/short stocks, or place a stop/limit order
Participating Actors:	Database, Yahoo! Finance API
Preconditions:	-Investor is logged in
	-Yahoo! Finance is accepting inquiries

Postconditions:	-**Database** us updated with the users position
	Flow of Events for Main Success Scenario:
→	1. **Investor** enters a market order
←	2. **System** attempts to get update from **Yahoo! Finance**
←	3. **System** receives information back from **Yahoo! Finance**
←	4. **System** validates and records trade in **database**
→	5. **System** confirms trades and displays changes in **investor** portfolio
	Flow of Events for Alternate Scenarios:
3a. **System** doesn't recieve information back from **Yahoo! Finance**	
←	4. **System** notifies **investor** of failed request

Of the 3 user types, administrator is the highest and reserved only for developers and administrators of the software. Administrators have the ability to modify or delete leagues or specific users if the administrator feels that power is being abused. The administrator will also have elevated privileges to makes changes to the site. Their main purpose will be to suspend or ban users or leagues and ensure that the site is not being abused. This includes but is not limited to robot or AI users or user account spamming or advertising rather than trading properly.

CG-BP06 Administrators will have access to suspend, ban, or remove leagues or players completely. Administrators will have elevated privileges relative to other users and will use these privileges with the sole intention of maintaining the integrity of the software.

Use Case UC-6	**Take Administrative Actions**
Related Requirements:	ST-19, ST-20, ST-21
Initiating Actor:	Site Administrator
Actor's Goal:	Perform administrative work for the website, manage database
Participating Actors:	Database, Investors, League Manager
Preconditions:	-User is the site Administrator
	-Administrative actions need to be taken
Postconditions:	-Conflicts/Issues have been resolved
	Flow of Events for Main Success Scenario:
→	1. **Site Administrator** requests logs from **system**
←	2. **System** closes saves log file and returns log report

Depending on the specific role of the user. (I.e. investor, league manager), users should be able to customize several different items. That is, the league manager will have ultimate customization of the rules of his or her league, including but not limited to: victory conditions, achievements, and starting capital. These rules are to be established prior to the beginning of the league and not

touched for the duration. League manager will also have a few other elevated privileges depending on whether he or she also falls into the investor role as well. Investors will be allowed to manage their own personal settings including but not limited to: email alerts, social media integration, and notifications.

CG-BP07 League managers will have access to a list of settings which they may select before the initiation of the league competition. They will have slightly elevated privileges from basic investors, but not enough to abuse power.

Use Case UC-7	**Manage League Settings**
Related Requirements:	ST-16, ST-17, ST-18
Initiating Actor:	League Manager
Actor's Goal:	Change league settings to the League Mangers preference
Participating Actors:	Database, other Investors
Preconditions:	-Initiating actor is the **League Manager**
	-There are outstanding abuse reports
Postconditions:	-The **Database** is updated to reflect the chagnes made.
	The abuse report shows that it has been resolved on the administration page
	Flow of Events for Main Success Scenario:
→	1. **League Manager** makes adjustment to league settings
←	2. **System** makes a update to the **Database**
→	3. **System** displays updated settings to the **league manager**

Traceability Matrix

The traceability matrix presented in *Figure 3.2* is based on only the full dressed use cases above and thus is only a partial representation of the complete project.

Requirements	Priority Weight	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7
ST-1	10	X						
ST-2	10	X						
ST-3	10		X					
ST-4	6			X				
ST-5	6			X				
ST-6	10					X		
ST-7	10							
ST-8	8		X		X			
ST-9	4				X			
ST-10	6			X	X			
ST-11	3			X		X		
ST-12	2				X			
ST-13	1				X			
ST-14	3				X			
ST-15	1							
ST-16	8		X					X
ST-17	8		X					X
ST-18	4		X					X
ST-19	2						X	
ST-20	3						X	
ST-21	9						X	
Total Priority		20	38	21	24	13	14	20

Figure 3.2: The traceability matrix for the given use cases.

3.4 System Sequence Diagrams

In the following sequence diagrams, we describe exactly the interactions between the key actors our system. It is important to note that most of the interaction between the user and system is facilitated by the browser. The user, through filling forms and button clicks, instructs the browser which requests to make to the system. In turn, the system communicates with the database to request the desired data, takes any required actions, and delivers the data to the browser for presentation to the user.

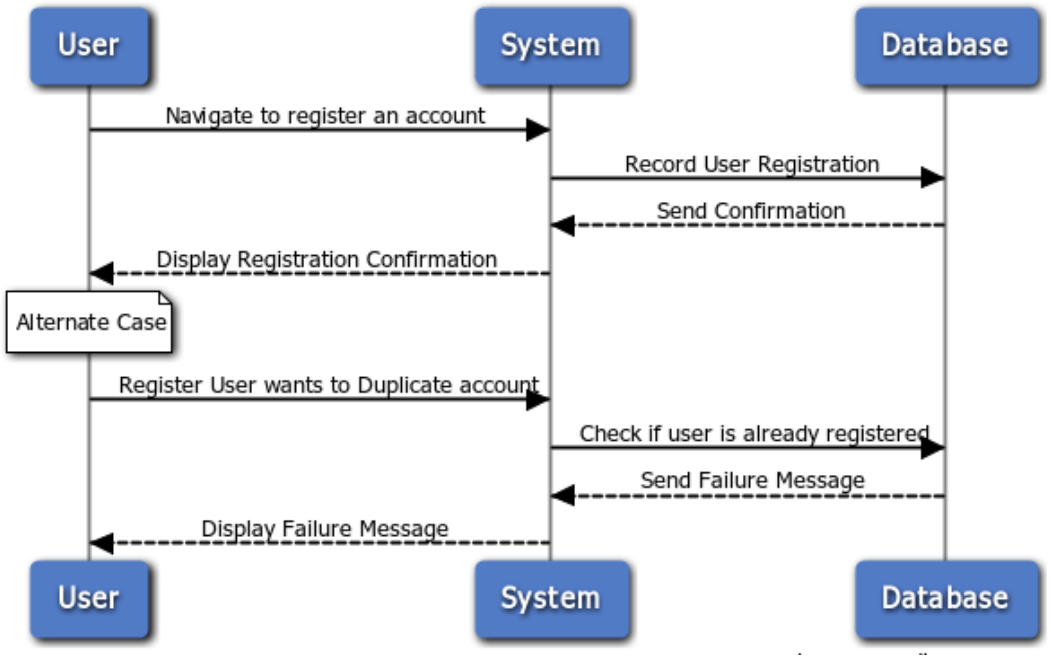

Figure 3.3: See UC-1 on page 19. When the user navigates to the login/register accounts page, this use case is triggered. The system makes it necessary to have an account before using Paramount Investments leagues. The System then takes the information that the user has input and sends them to the database, which then sends it back to the system to display on the screen to the user. If the system finds that the user that is registering with the same credentials as an existing account, the system throws up an error appropriately.

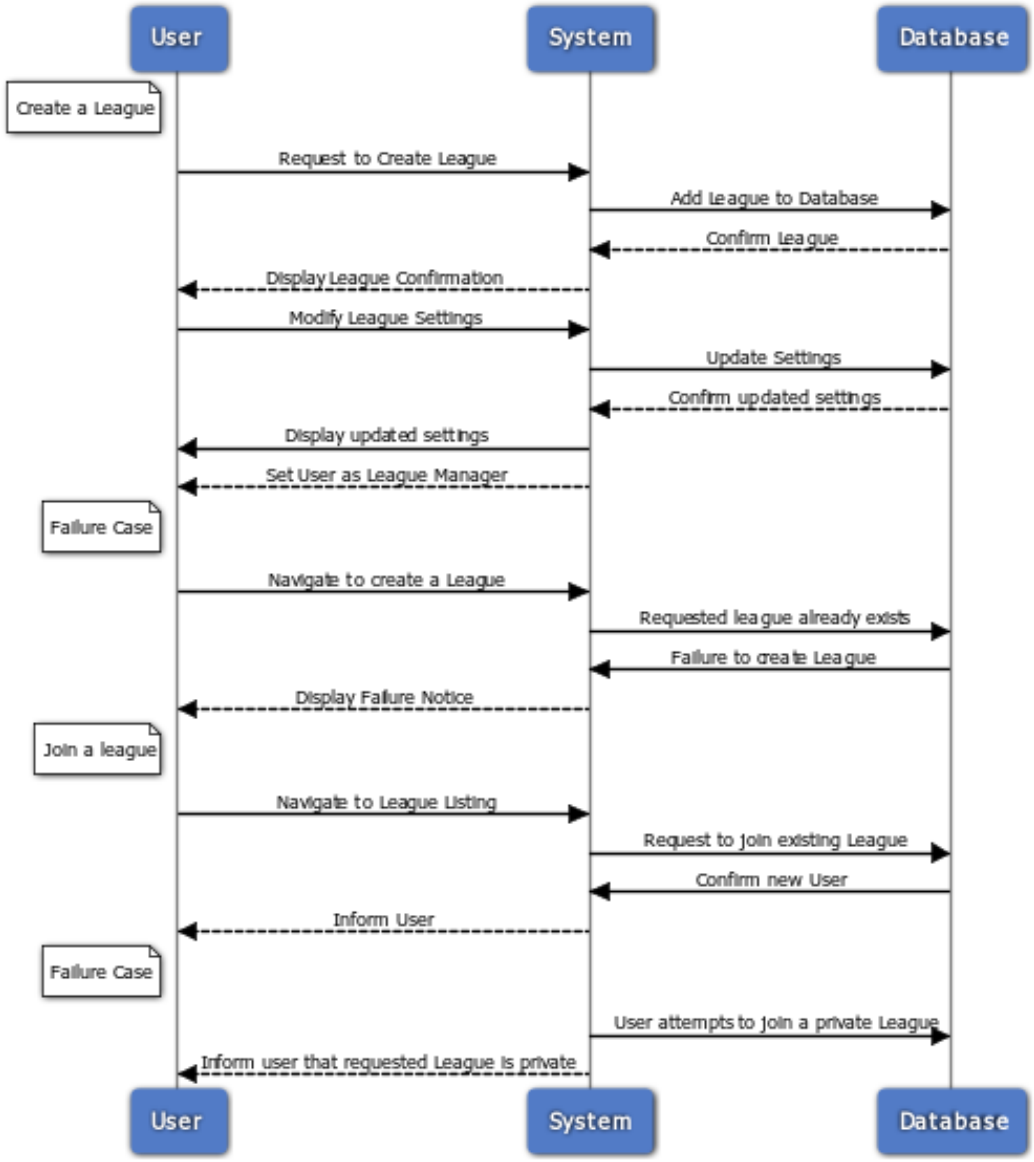

Figure 3.4: See UC-2 on page 20. This use case is triggered when the user navigates to the create league page. The user requests the system to create a league, which then sends appropriate data to the database. Once the data is stored successfully, the database sends a confirmation back to the system, which then displays an appropriate message to the user. If the user wants to join a league, the user requests the system appropriately, which then sends the data to the database regarding the right league and if successful sends the confirmation to the user.

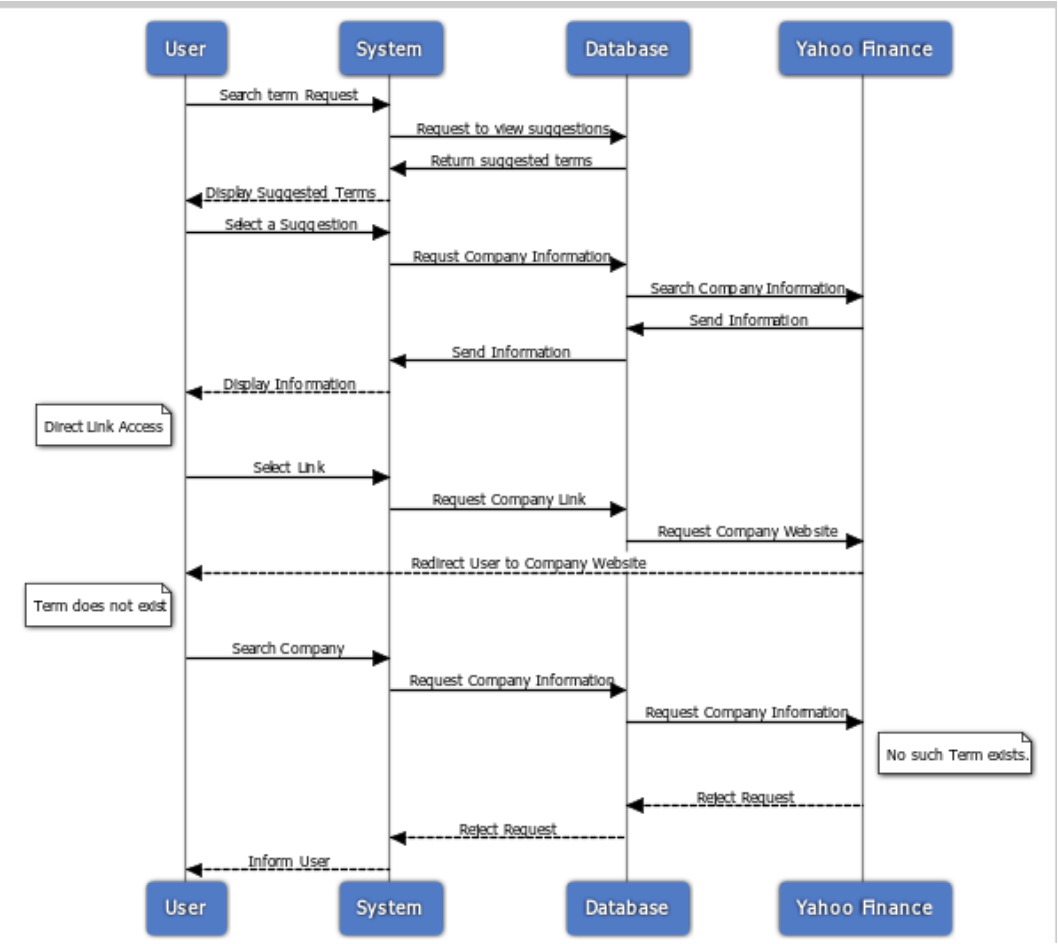

Figure 3.5: See UC-3 on page 45. When the user navigates to the research stock page, this use case is triggered. The user specifies to the system exactly what market data they would like to view. If the user wants to research off the companys website, then user will click on the hyperlink present in the system. If the user wants to view the data through the interface that Paramount Investment Provides, the system then pulls information from the Yahoo Finance API and then sends it back to the system to display to the user.

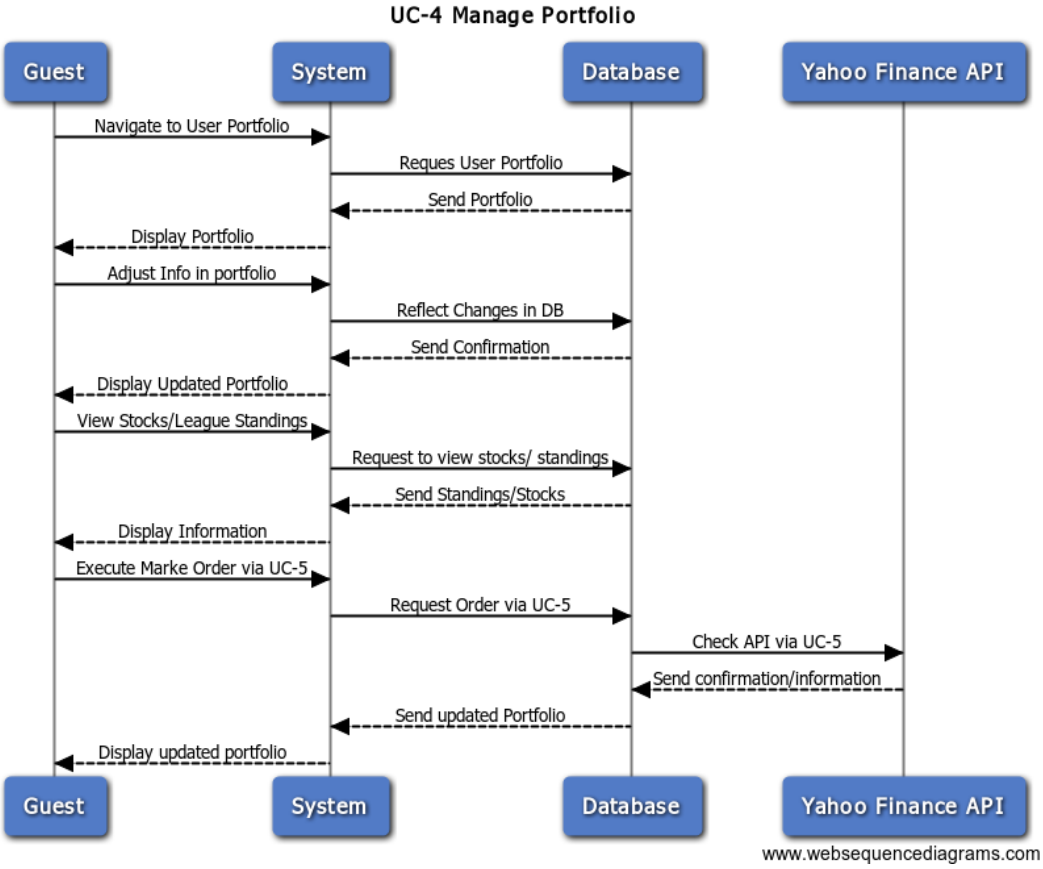

Figure 3.6: See UC-4 on page 22. This use case is triggered when the user goes to his/her own portfolio page. The user navigates to the view portfolio page. The system then requests the database to retrieve the users information. Once the database sends this data back to the system, the system displays it to the user. The user is now free to modify aspects of his/her portfolio. Once the user is finished modifying/updating their portfolio, the system will send the changes to the database, which will then store the information and save it.

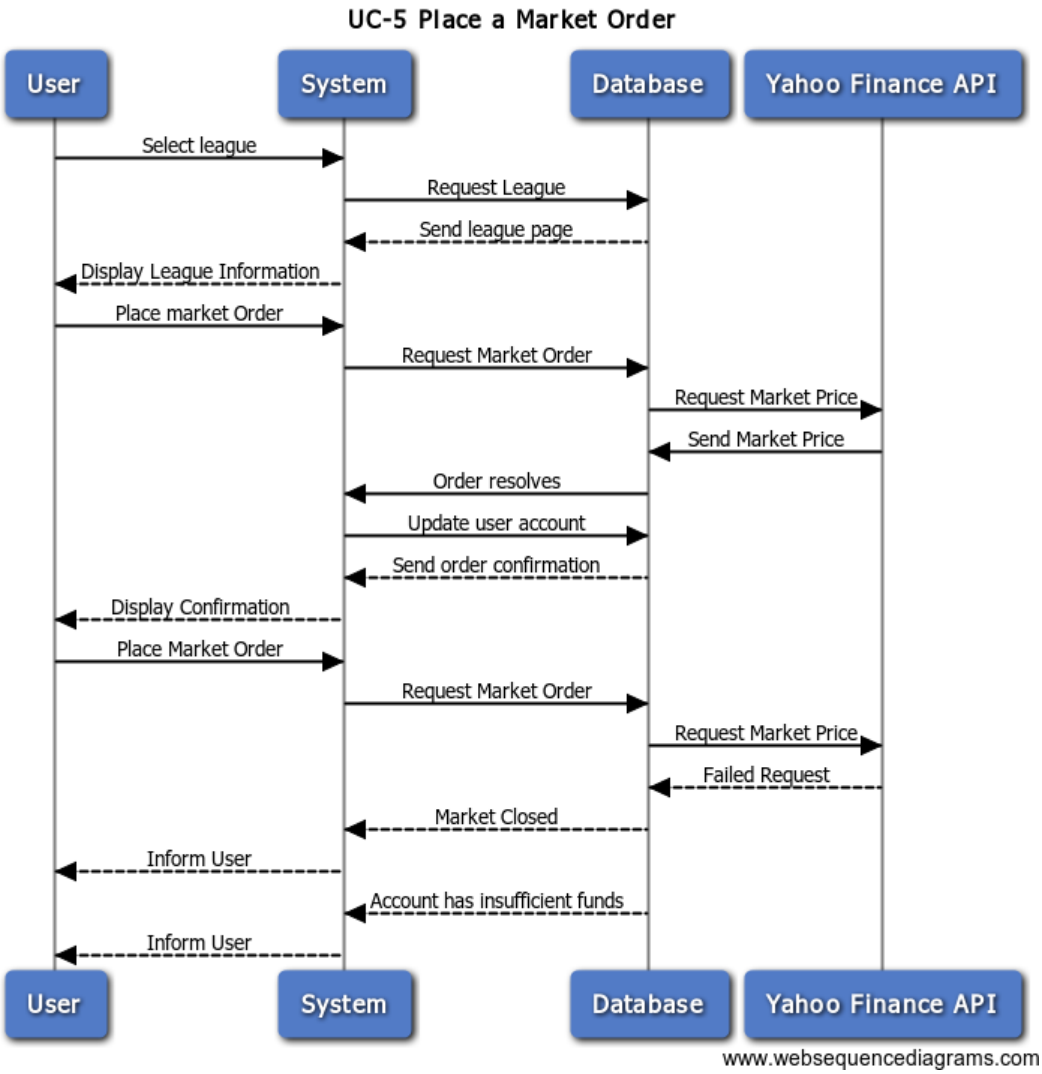

Figure 3.7: See UC-5 on page 22. This use case is triggered when the user goes to place a market order in the place order page. The user selects a league to place the order into. The system then displays the information to the user, who then requests to place a market order. The system then queries to Yahoo Finance API to retrieve information about the stock prices. The system then takes that information and processes it with what the user wants to do. If the order is completed successfully, it is appropriately displayed.

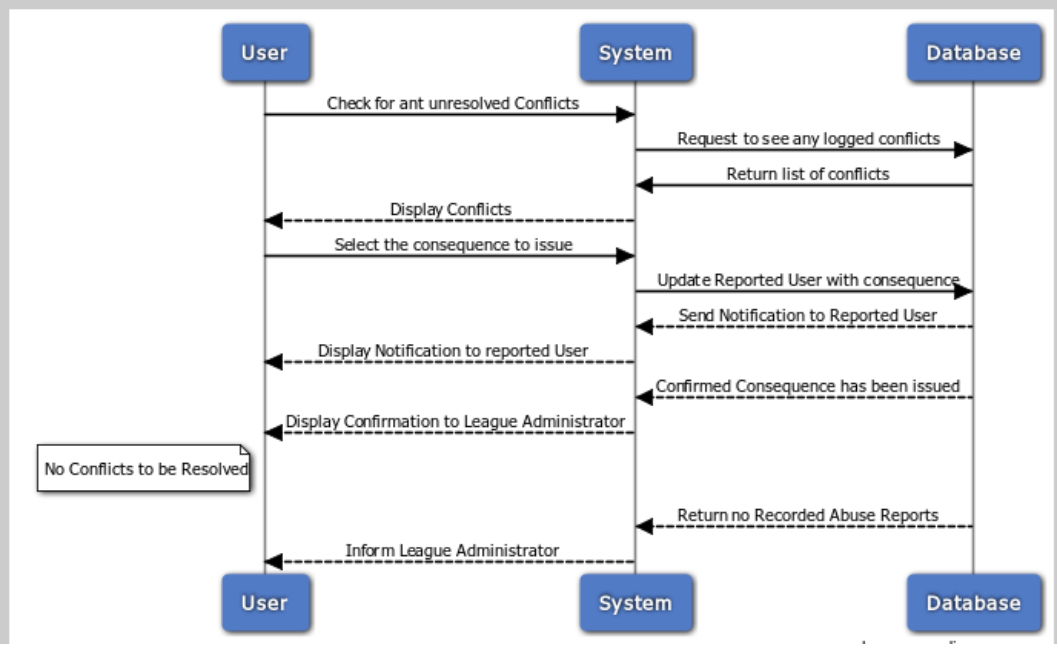

Figure 3.8: See UC-6 on page 23. This use case is triggered when the administrator of the website/ league wants to take action. The system first checks if the user logging in has administrative privileges in their respective group. If so, the system then looks in the database to check for any logged conflicts. If there are unresolved conflicts, the database returns them and then user can then view the conflicts. If there are no conflicts to be resolved, then display so appropriately to the administrator/logged in user.

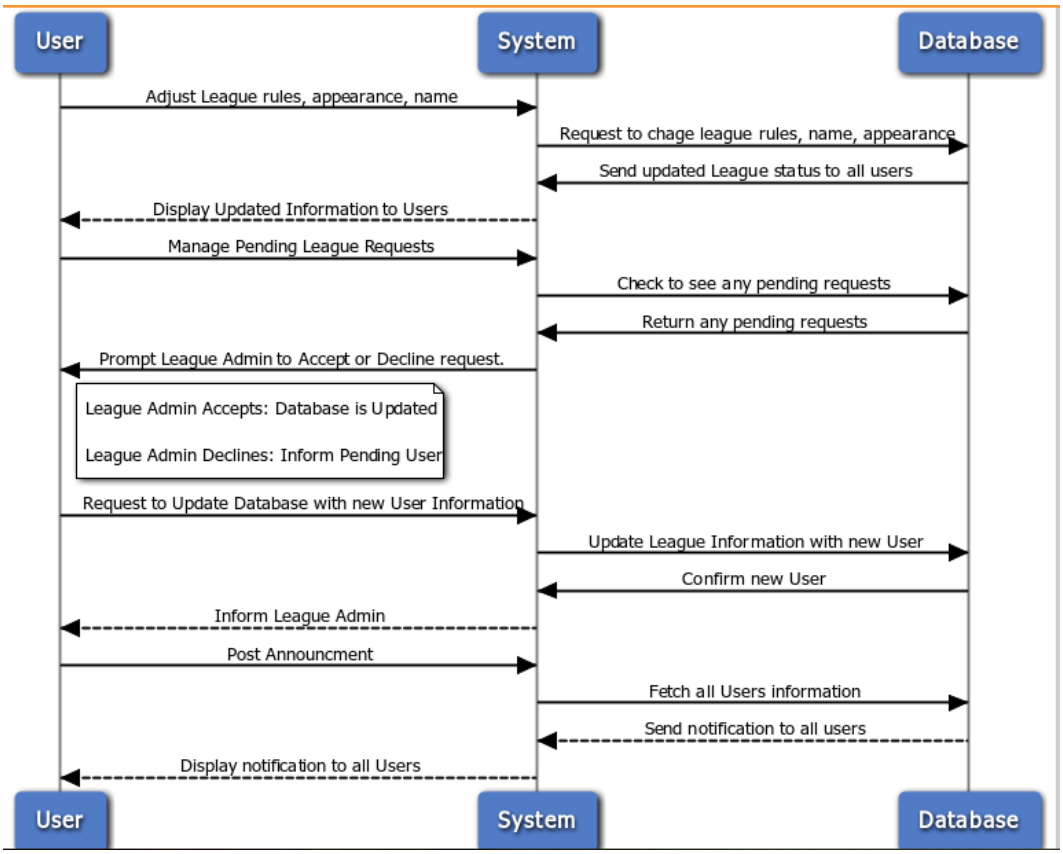

Figure 3.9: See UC-7 on page 24. This use case is triggered when the manager of a league wants to change the league settings. The system first checks to see if the user that is logged in, is the league manager and if so, the grants the user privilege to change league settings. The user then requests to change the league settings. The system retrieves the league data and modifies it as the user has requested.

4 User Interface Specification

4.1 Preliminary Design

The user interface (UI) for Paramount Invesments Leagues will act as a command center for users to interact with their portfolio, leagues they are a part of, and conduct research on potential orders. More specifically, the command center will act as the primary; but not the only; view for users to interact with the system. The command center will provide a snapshot of the users current portfolio and its value, their global rank, a dash to perform market orders, a news feed, and a graphing dash in order to quick analysis of stock performance. The UI will persist a users global rank across all views as well as a ticker of current trades being placed through the Paramount Investment League.

The UI should be lightweight so as not to burden our more restrictive target platforms of mobile and tablet. The colorscheme will be chosen to be easy on the viewer, though this is subjective, the colorscheme will be a basic pallet of grey/black/white/blue, tending toward pastel and web supported colors.

The UI will be built on top of Twitter's open source Bootstrap CSS[5] framework to help facilitate deleriving content to the three target platforms, desktop, mobile, and tablet. Bootstrap provides a mobile first design philosophy, but can be customized to target specific platforms.

Landing Page and Login

Paramount Invesment League is designed around allowing users to easily begin using the service, also know as "zero effort" resgistration. In order to accomplish this, the system does not require the user to register a new user name/account with our system, but instead piggybacks on OpenID[**?**] and OAuth[6] allowing users to use their Google, Facebook, Twitter, and other OpenID/OAuth accounts to login. You'll also notice that upon initial visit, the header is empty providing no navigation, this may be relaxed in the future to allow the user to explore some of the features of the website that don't require user authentication such as stock research. (*See figure 3.1*)

Global Header

The header (*see Figure 3.2*)across the website will remain persistant across the website once the user is logged into the system. Navigation is done between essentially 4 views in the following order, My Portfolio, Stock, League, Leaderboard,. These names are placeholders and will most likely be

Figure 4.1: First iteration of Landing/Login page.

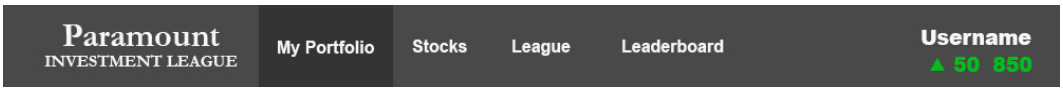

Figure 4.2: Preliminary design for a global header. This users is up 50 spots for the day.

My Portfolio, My Leauges, Leaderboards, Analyze Assets. The 'My Leagues' and 'Leaderboards' will be turned into drop downs as users expand into leagues to allow quick navigation.

The website name will also navigate to My Portfolio. The username will be replaced by the users actual username, and below it will be the users global rank. The rank will be highlighted in red or green depending on whether they have improved their position on the day, or it has declined. It will also indicate how many spots they have moved.

Global Ticker

One interesting feature of Paramount Investment Leagues will be its active ticker at the bottom of the website. This ticker will be seen in all views, including the Landing Page once there is enough volume to keep the ticker full. The ticker serves two goals, one for new users, and one for existing users. The first goal is to entice new users to participate by demonstrating that the app is being widely used. The second goal is to give a snapshot to existing users of assets that are "on

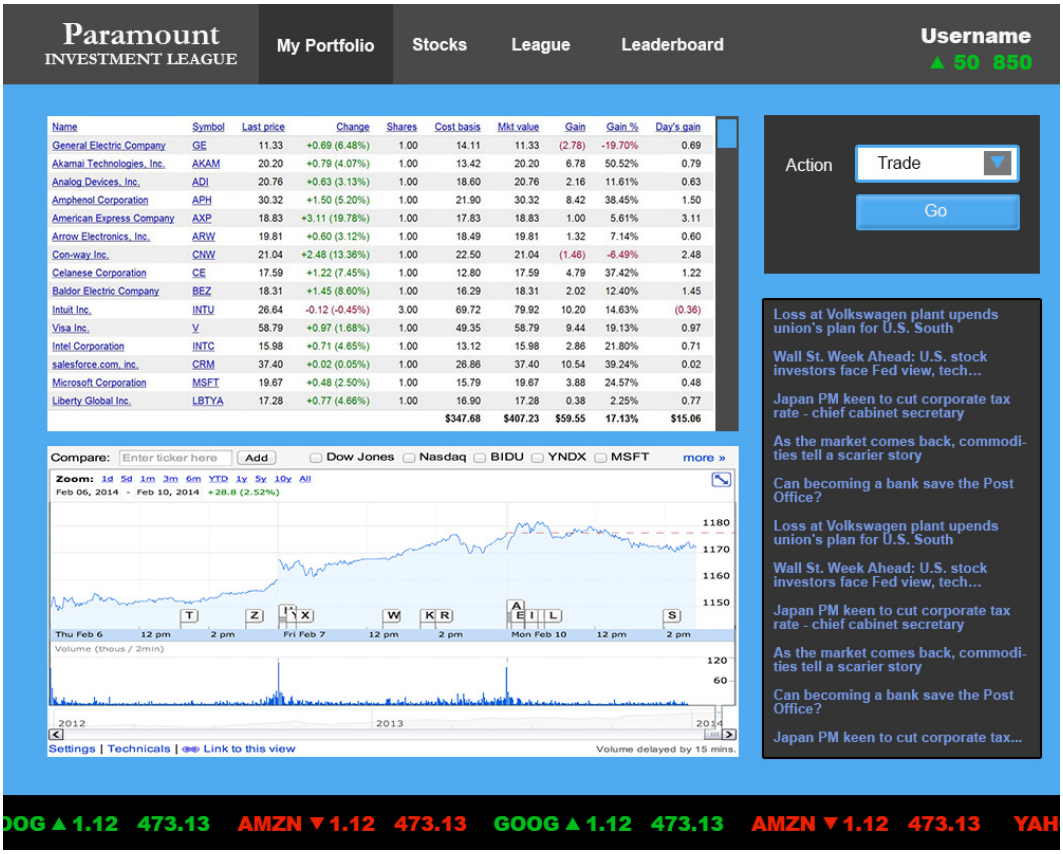

Figure 4.3: The preliminary design of the 'My Portfolio' view.

the move" so that they can attempt to remain competetive. The ticker can be seen at the bottom of all the figures.

My Portfolio

The 'My Portfolio' (*see Figure 3.3*) view of the website will act as the command center for a user wanting to get news about companies/assets in their portfolio, perform an order, or conduct quick graphical anaylsis of assets in their portfolio and compare them to any other asset available for trade through the platform.

More importantly, it provides a snapshot of the users portfolio including a scrollable list of all the assets inside the portfolio and a summary of said assets. In the future, assets will be 'clickable' and will take the user to a summary page of that asset, but that is not planned for the initial 2 iterations.

Leagues

The 'League' (*see Figure 3.4*) view will present a user that isn't a part of a league the ability to create a new league of join an existing league. Not shown in *Figure 3.4* is the view that a user

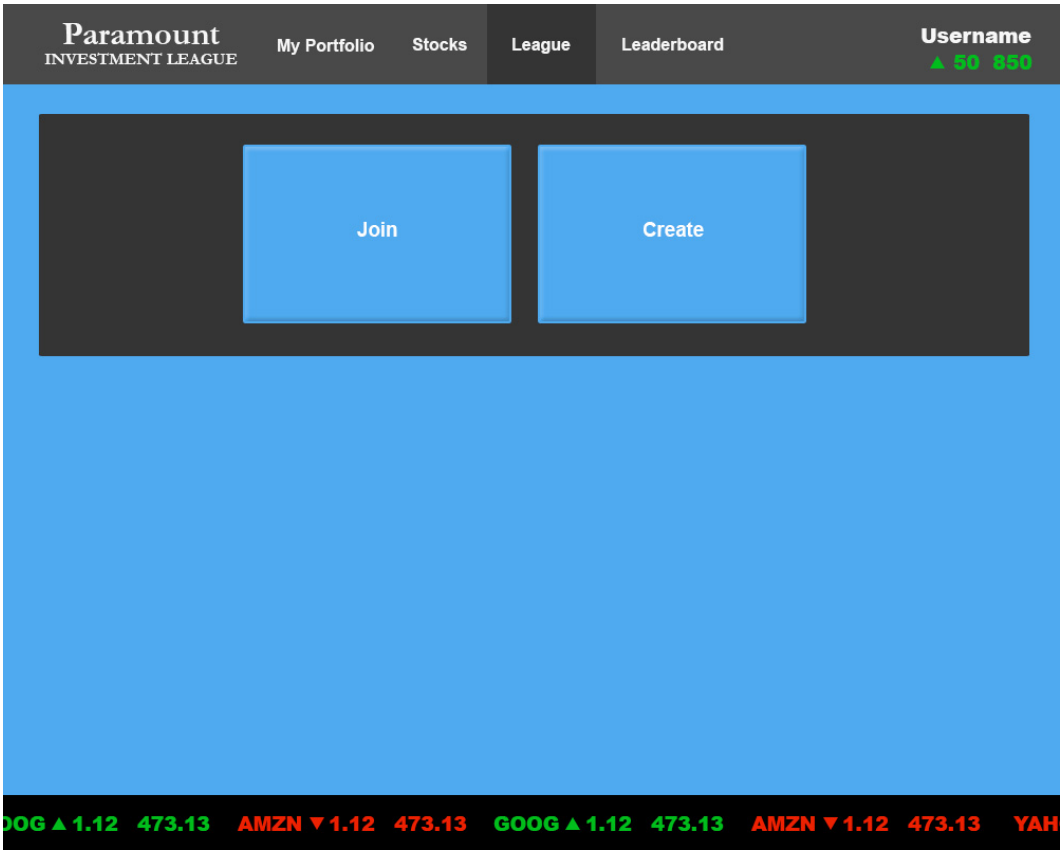

Figure 4.4: This is the league creation/join view. This would be the view presented to a user that is a part of no league yet.

who is a part of a league. This view will still persist the join/create dialogues, but will also present a list of all the leagues that user is a part of, their rank within said league, and their movement within said league.

Leaderboards

The 'Leaderboards' (*see Figure 3.5*) view will present the user with a partial view of the full leaderboard for a given league, or for every user. It will show their rank, their movement, the value of their portfolio as well as the same stats for all other users around them. The view will be scrollable if there are more records then can be displayed, and will center the user in the middle of the view unless they are at the top or bottom of the board.

Asset Analysis

The 'Stocks' view (*see Figure 3.5*) will be renamed to more align its function with its name, which is to analyze assets. It will a more in depth way of anaylzing an asset versus what is available in the 'My Portfolio' view. There will be a news feed at the bottom of assets that you are searching for. There will also be a more formal analysis of asset data presented including P/E ratio, 52 week

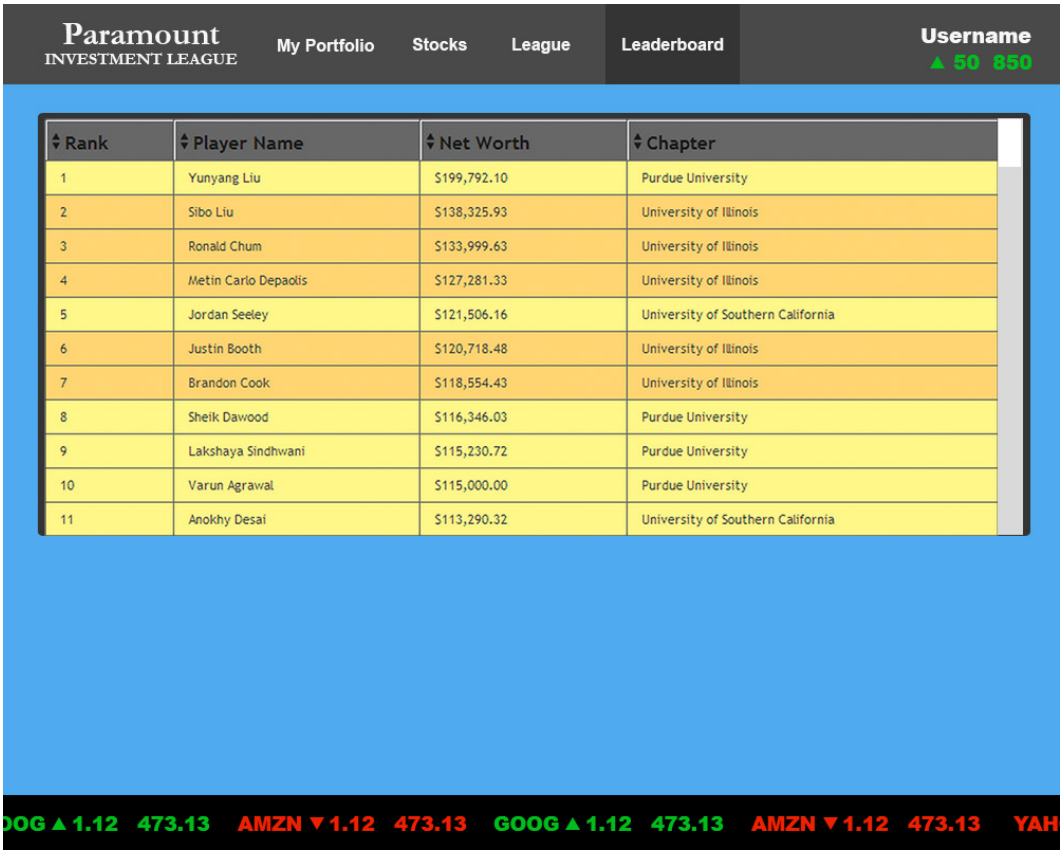

Figure 4.5: Here is the leaderboard view which will be the same for both leagues and global leaderboards. This view represents a global leader board. The colorscheme of this view here is incomplete and will fall inline with the remainder of the site.

range, Volume, EPS, etc. This isn't shown in the figure, but will one-half to two-thirds of the space that has been set aside for the news feed.

This is also one of the views and functionalities that has been identified to not require the user to be logged in. While it will not be availble to non-users in the intial product, it can be made available in future releases.

4.2 User Effort Estimation

Several of the most common usage scenarios for Paramount Investment Leagues:

Figure 4.6: The preliminary view for asset anaylsis.

Usage Scenario	Clicks	Keystrokes
Login & Register	2-3	0-1
Place an Order	4-6	2-12
Join a League	3-4	0-50
Create a new League	6-7	11-100
Analyze Asset	2	2-5
View Leaderboard	2	0

Login & Register

Assume the user has come to the domain and wishes to Login if already registered, or register if already a user:

- **Navigation:**
    1. Click on OpenID icon (Google, Facebook, Twitter, etc).
    2. Click on your account (optional for multiaccounts).
    3. Click on login, or hit enter.

Place an Order

Assume the user has already logged in and they wish to place an order:

- **Navigation:**
    1. Navigate to 'My Portfolio', 0-1 clicks.
- **Data Entry:**
    1. Select order type from drop down, 2 clicks
    2. Click textbox to enter asset name. 1 click
    3. Enter assets name eg: 'G', 'O', 'O', 'G', 1-4 keystrokes
    4. Press tab to specify number of shares, 1 keystroke (user could also execute 1 click)
    5. Enter the number of shares, 1-7 keystrokes
    6. Click execute, 1 click

Join a League

Assume that the user wishes to join a league and is logged in:

- **Navigation:**
    1. Click on League, 1 click
    2. Click on Join, 1 click
- **Data Entry:**
    1. Click on a League, or enter its name, 1 click or up to 50 keystrokes
    2. Click on confirmation dialogue, 1 click

Create a League

Assume that the user wishes to create a league and is logged in:

- **Navigation:**
    1. Click on League, 1 click
    2. Click on Create, 1 click

- **Data Entry:**
    1. Enter its name, 1-50 keystrokes
    2. Select ruleset from dropdown, 2 clicks
    3. Fill in parameters, 1-2 clicks and 10-50 keystrokes
    4. Click on confirmation dialogue, 1 click

Analyze an Asset

Assume that the user is logged in and they want to start an in depth analysis of an asset:

- **Navigation:**
    1. Click on Stock, 1 click
- **Data Entry:**
    1. Click on the textbox for entering an asset name, 1 click
    2. Enter an asset name, 1-4 keystrokes
    3. Hit enter, 1 keystroke

View Leaderboard

Assume that the user has logged in and wants to veiw a leaderboard:

- **Navigation:**
    1. Click on Leaderboard, 1 click
    2. Click on Select Legue/Global, 1 click

5 Domain Model

5.1 Concept Definitions

The Domain Model Concepts are derived from responsibilities contained in the Use Cases from Chapter 3.

Responsibility	Type	Concept
R1:Allow new user to create an account to partake in stock trading game as an Investor. And Login existing user.	D	Account Controller
R2: Initialize accounts with fixed amount of capital.	D	Account Controller
R3: Check if Investor is in a league or not.	K	League Controller
R4: Update new leagues at set interval and display to correct Investors.	K	League Controller
R5: Retrieve information about Stocks, Trades, and Companies.	K	Yahoo! Finanace Adapter
R6: Display information about users Stocks, Trades and Leagues	K	Player Profile View
R7: Record and Execute Buy/Sell/Short Stock trades	D	Order System Controller
R8: Display Welcome screen to create an account or log-in.	K	Login View

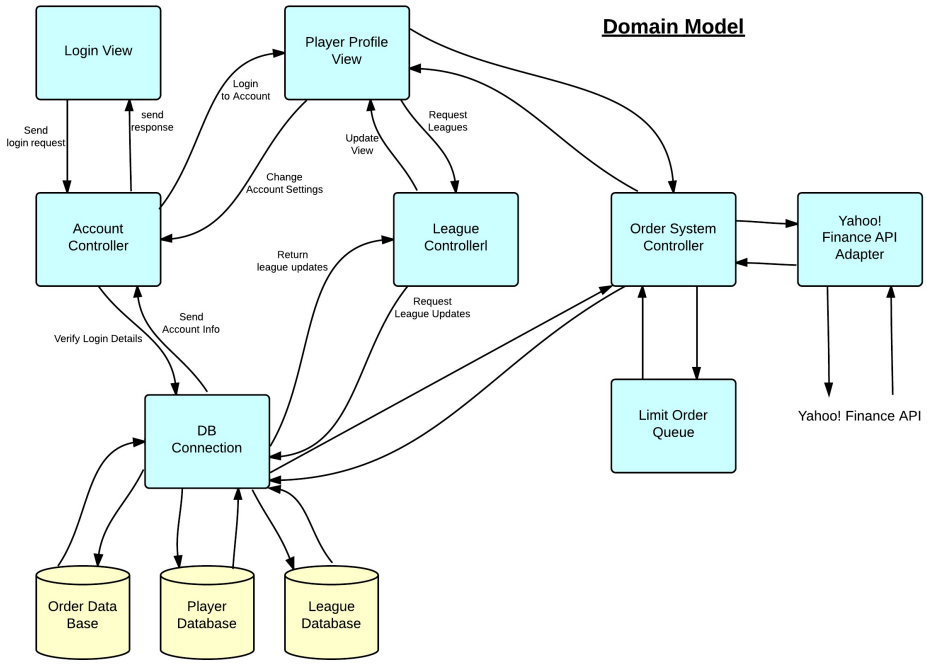

Figure 5.1: The domain model.

Account Controller

The first step for anyone using this system is to gain access by creating an Account. Account Creator is an interface that allows a player to create a new Investor account. It will check with the database for an account with the same details, and if not found it will proceed to create the account and store the details into the database.

Player Profile View

The Player Profile displays statistics and saved settings for the user that logs into The Paramount Investments League. Player Profile should query Yahoo! Finance Adaptor API to get data about Watchlist stocks, and any searched Target stock.

Login View

The Login View displays a UI to allow the user to login with OpenID. This will send a request to the account controller. If it fails the Login View will be updated to reflect this. If this succeeds the user will now see the Player Profile View.

League Controller

League Manager will proceed to keep up to date display of rankings of every player in each league. These will be fetched from a database and displayed in the Player Profile. League Manager will allow Investors to join open leagues that they are not already in. This will require querying the database and updating if necessary. League manager may give our achievements based on certain accomplishments within the leagues.

Yahoo! Finance API Adapter

Yahoo! Finance API provides the almost real time stock data that our application is dependent on. Any downtime Yahoo! Finance experiences will affect our application. Yahoo! Finance API Adaptor serves as a translation between the CSV file that Yahoo! Finance produces through their API, and our application. Our Adaptor will take the spreadsheet Yahoo! Produces and convert the data into syntax that our application can understand. This adaptor is modular in order to allow multiple subsystems to make queries for live stock updates.

Order System Controller

Any order placed by an investor will go through the Order System which will sum the cost of the transaction and check that the account balance is satisfied. This will require communication with the Yahoo! Finance API Adaptor to get the current price of the target stock to be purchased/sold. If the order is a limit trade, we must define a way to check if the current price of the stock matches the limit price and execute the order.

Database Connection

We want to have a single subsystem maintain control of accessing the database to make retrieving information modular. This will allow expansion of the application to add additional functionality later which may also need access to the database. This should provide a layer of security so each subsystem does not access the database directly.

5.2 Association Definitions

Concept Pair	Association description	Association name
Login View <> Account Controller	Login View sends a login request. Account Controller can respond with Success or Failure.	sends
Account Controller <> DB Connection	Account Controller sends user login details. DB Connection sends account info or failure.	sends
Account Controller <> Player Profile View	Account Controller updates view to be the Player Profile View. (Profile View may also allow user to change some account settings).	updates
Player Profile View <> League Controller	User may send request to League Controller to update, join, or leave a league. League controller can update view with information.	sends, updates
League Controller <> DB Connection	League controller request league statistics from DB Connection. DB Connection sends statistics	sends
Player Profile View <> Order System Controller	Player Profile View sends requests to the Order System Controller. Order System Controller updates the Profile View.	sends, updates
Order System Controller <> Yahoo! Finance API Adaptor	Order System Controller sends requests to API Adaptor about stock prices. API Adaptor returns information about the stock info.	sends
Order System Controller <> Limit Order Queue	Order System Controller creates the limit order queue when a limit order is placed but cannot be executed immediately.	creates, updates, closes

The associations of domain concepts are derived from the table above. The Account Controller takes information that a user enters into the Login View. This information is sent to check with the Database. The Account Controller can change the view to the Player Profile based on what information is stored in the database. From the Player Profile View a user can access account settings, leagues, and portfolio details. The league details are managed by the League Controller, which can allow requests to create/join a certain league. The league controller should also periodically update the Player Profile Views statistics of the leagues they are in. The Order System Controller allows the user to search market data, and attempt buy/sell/short trades. The Order System Controller must communicate to see if the user has sufficient funds. Limit trade may not be executed until the stock price reaches a certain value. If the limit trade cannot be executed immediately a Limit Order Queue is created and these orders will be placed in here. The Order System Controller communicates with the Yahoo! Finance API Adaptor to retrieve current quotes for stocks. Yahoo! Finance Servers must be online for this to work correctly.

5.3 Attributes Definitions

Responsibility	Attribute	Concept
R9: Know if user login failed	LoginFailed	Login View
R10: Player's name and OpenID	Name/OpenID	Player Profile View
R11: Players Account Balances (Cash Balance, Money Invested, Daily Change).	Account Summary	Player Profile View
R12: Stocks User added to WatchList.	WatchList	Player Profile View
R13: Stocks User owns.	Owned Stocks	Player Profile View
R14: Leagues and Rankings	leagueID/rank	Player Profile View
R15: Know if user is logged in	isLoggedIn	Account Controller
R16: Know which leagues user is in.	isInLeague	League Controller
R17: Know if user is creating a league	isCreatingLeague	League Controller

5.4 Traceability Matrix

		DOMAIN CONCEPTS						
Use Case	PW	Account Controller	League Controller	Order System Controller	Login View	Player Profile View	Yahoo! API Adaptor	DB Connection
UC1	20	X			X			X
UC2	38	X	X			X		X
UC3	21			X		X	X	
UC4	24	X		X		X	X	X
UC5	13	X		X		X	X	X
UC6	14	X	X	X		X		X
UC7	20	X	X	X		X		X
Max PW		38	38	24	20	38	24	38
Total PW		129	72	92	20	130	58	129

Figure 5.2: The traceability matrix.

5.5 Economic and Mathematical Models

Perfect Competition

One of the prevalent concepts in the stock market is the economic concept of perfect competition, which says that not any single participant has enough resources/power to control the market. To apply the concept of perfect competition to our project we will need the following requirements:

- Not one person can control the market or industries, segment, etc.
- Users can feel free to execute trades at their convenience without having to worry about extra costs
- Every individual has access to same stock information as other investors
- The selling price is the same as the buying price.

In the real world, none of these requirements can be met, as there is always some problem that prevents the market from being in perfect competition. The following are just some of the problems:

- There are high net worth individuals/companies who have enough capital to change the tide of a certain sector of the market. If one of these individuals suddenly decides to leave a particular market, the move may suddenly shift the market and effect other investors in that market.
- In the real world, users typically dont have direct access to stocks. They have a broker (electronic or human) who they interact with, who then have direct access to stocks. Users cant usually execute trades/buy stocks without worrying about extra costs because of the commissions charged by brokers when trading stocks.
- The world is not a fair place, and neither is the stock market. There are individuals who because of the field that they work in, have much more insight into a particular industry/stock. These individuals then sell this information to potential buyers in hopes that it gives them an edge in trading. This gives a huge disadvantage to those that dont have access to more information bout stocks.
- Lastly, in the real world, the selling price is never usually the same as the bid price. The Bid-Ask spread, the difference between the buying and selling price tends to be greater than 0.

All these factors lead the stock market away from perfect competition.

How do we plan to fix these issues to ensure a near-perfect competition?

- All investors start with the same amount of money, this way no one person by default has more power than anyone else
- No commission will be charged when the trades are executed for any investor
- Insider trading will be avoided by standardizing the stock information across the board

- The ask-bid spread will be 0, so the selling price is the same as the buying price

Mathematical Model:

- Stock Prices
  - There are no complicated mathematical models behind how the stock prices are determined in our platform. The market prices that are retrieved from Yahoo Finance are the prices that are available to users in Paramount Investments
- Achievements
  - Achievements in Paramount Investments each have their own mathematical model. There are no complicated algorithms behind how these achievements are attained. If the user has met the required conditions for a certain achievement, then they will be given that specific award.
  - For example: Buy stocks whose P/E Ratio > 1

6 Plan of Work

6.1 Development and Report Milestones

Illustrated on the next page is a chart reflecting our goals relative to the project dead-lines. It incorporates both core development and report items. For our initial stages we focus on environment and platform set-up (eg: deploying a development webserver) and the initial, core code implementation. At the same time we will finalize the details of our final product via the report milestones.

Development milestones have been spread out following the completion of Report 1 on 23 February 2014. It begins with deploying our development environment and server through Digital Ocean[7]. We concurrently will roll out developer images, the Play Framework[8], and develop database schema. Implementing user registration/login will follow shortly along with deploying a solution to use the Yahoo! Finance API. The development milestone finishes up with the implementation of user portfolios along with basic market operations and basic achievements.

Report milestones are also set concurrently. As we begin to initialize our development environment, we will also build on top of and expand on previous reports to expand upon and fully realize the details of *P*aramount Investment League.

Core goals leading up to Demo 1 include establishing all core functionality for *P*aramount Investment League. This includes the following:

- **Play Framework deployment :** This includes basic site navigation, user login/registration, and Twitter Bootstrap deployment.
- **Setting a foundation for the database:** Schema should be built to be extensible to support future enhancements.
- **Implement the Yahoo! Finance API**
- **A functional user interface:** The user interface should function across multiple platforms with a focus on experience and expectations.

6.2 Breakdown of Responsibilities Introduciton

Contributions leading up to the completion of this report are covered in the "Contributions" in Chapter 7. For the future division of labor, we all plan on subdividing aspects of both the next reports as well as the development of the *P*aramount Investment League Demo 1.

6.3 Breakdown of Responsibilities

Core server deployment will be the repsonsibility of David Patrzeba. Eric Jacob will be responsible for the database rollout. David Patrzeba will also be responsible for the core software rollout on the server including git, Play Framework, nginx, and other core libraries and software. David Karivalis will be repsonsible for integrating Twitter Bootstrap into Play Framework.

Routing will be headed by Eric Jacob and assisted by Chris Mancuso and Evan Arbeitman.

User Interface will be done by David Karivalis and Jesse Ziegler and they will integrate the REST API[9] to facilitate dynamic views.

The rest of the development workload will be divied up based around the Model, View, Controller design pattern. David Patrzeba and Eric Jacob will focus on the controllers, David Karivalis and Jesse Ziegler will focus on the Views, and Evan Arbeitman and Chris Mancuso will focus on models. David P., David K., and Eric will be made available for technical advising.

David Patrzeba will be responsible for formatting the report. David Karivalis will be responsible for digitization of paper diagramming for all reports. Report duties will be divied up based on percieved strengths of the team and availability.

Overall project success will be decided with how well the MVC[10] component teams communicate and work with each other, as *P*aramount Investment League will rely on the interactivity between the Model, Views, and Controller portions of the architecture.

6.4 Projected Milestones

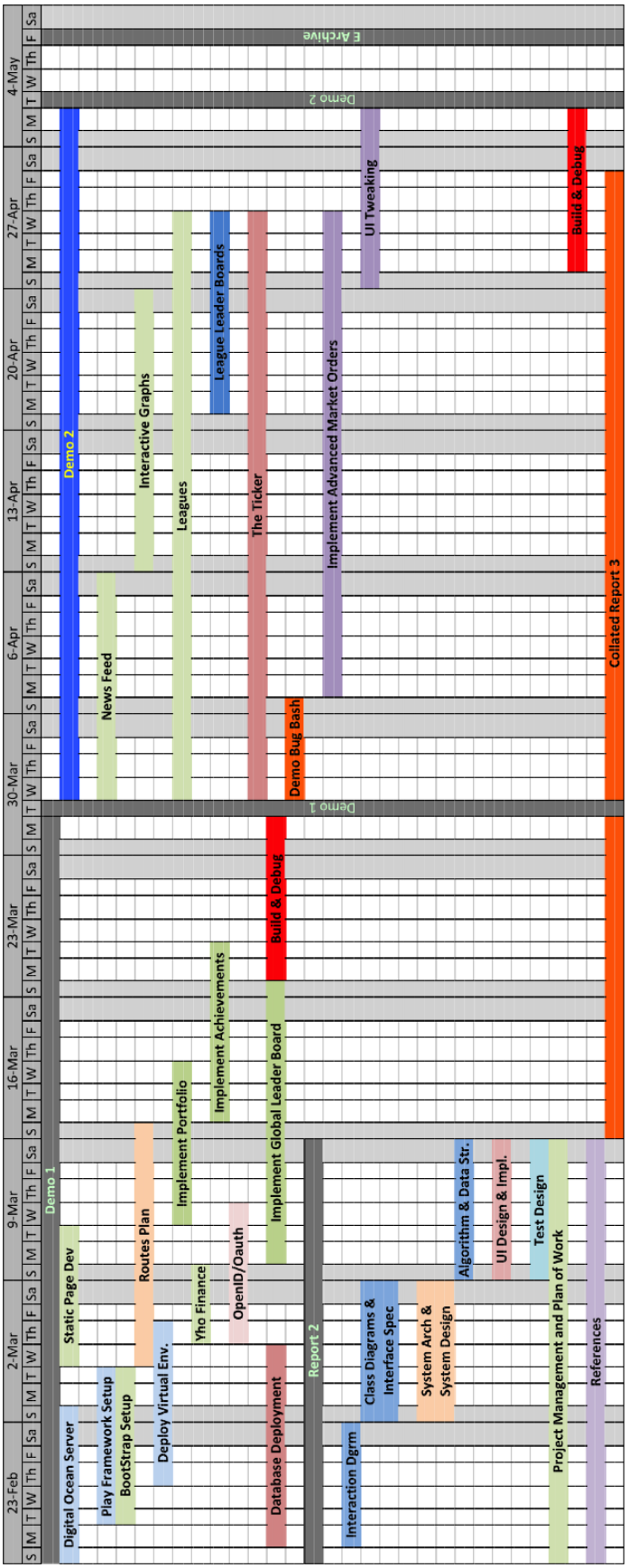

Figure 6.1: This chart is the roadmap to meeting all our milestones.

7 Project Management

7.1 Report 1 Contributions

		Names					
Category	Points	David P	David K	Jesse Z	Evan A	Eric J	Chris M
Project Management	10 Points	100%	0%	0%	0%	0%	0%
Customer Requirements	9 Points	50%	0%	0%	0%	0%	50%
System Requirements	6 Points	20%	20%	0%	0%	30%	30%
Functional Requirements	30 Points	0%	0%	0%	33%	33%	33%
User Interface Specifications	15 Points	50%	50%	0%	0%	0%	0%
Domain Analysis	25 Points	0%	0%	100%	0%	0%	0%
Plan of Work	5 Points	100%	0%	0%	0%	0%	0%

References

[1] Investopedia, "Bid-ask spread — Investopedia." http://www.investopedia.com/terms/b/bid-askspread.asp. [Online; accessed 18 February 2013].

[2] Investopedia, "Short (or Short Position) definition — Investopedia." http://www.investopedia.com/terms/s/short.asp. [Online; accessed 22 February 2013].

[3] Investopedia, "Limit order definition — Investopedia." http://www.investopedia.com/terms/l/limitorder.asp. [Online; accessed 23 February 2013].

[4] Investopedia, "Stop order definition — Investopedia." http://www.investopedia.com/terms/s/stoporder.asp. [Online; accessed 22 Febrauary 2013].

[5] Wikipedia, "Bootstrap (front-end frameworl)." http://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Online; accessed 23 February 2014].

[6] Wikipedia, "Oauth." http://en.wikipedia.org/wiki/OAuth. [Online; accessed 23 February 2014].

[7] Wikipedia, "Digital ocean." http://en.wikipedia.org/wiki/DigitalOcean. [Online; accessed 23 February 2014].

[8] Wikipedia, "Play framework." http://en.wikipedia.org/wiki/Play_Framework. [Online; accessed 23 February 2014].

[9] Wikipedia, "Representational state transfer - Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Representational_state_transfer. [Online; accessed 23 February 2013].

[10] Wikipedia, "Model-view-controller - Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Model-view-controller. [Online; accessed 23 February 2014].