Money Machine

Report #1

Group No. 6

Team Members

Name	**Email**
Rylan Uherek	rylan@scarletmail.rutgers.edu
Avinash Oza	avioza@scarletmail.rutgers.edu
Aakash Patel	Aak4shpatel@gmail.com
Mozam Todiwala	tmozam@scarletmail.rutgers.edu
Mandeep Desai	Mandeep.desai111@gmail.com
Pintu Patel	Php28@scarletmail.rutgers.edu

Instructor: Prof. Ivan Marsic

Project URL: https://sites.google.com/site/sespring13/

Revision History:

Version No.	**Date of Revision**
v.1 – Part #1	2/11/2013
v.2 ** -- Part #1 + TOC (see below)	2/14/2013
v.3 – Part #2 (inc. Part #1, as per submit instructions)	2/18/2013
v. 4 – Final Report #1	**2/22/2013**

** We didn't realize that Part #1 of Report #1 required a TOC, Breakdown, & References. We are adding these in this revision. <u>However, we did submit the required documents on time</u>.

Individual Contributions Breakdown

Task/Group Member	Rylan	Avinash	Aakash	Mozam	Mandeep	Pintu
Project Management (10 points)	25%	15%	15%	15%	15%	15%
Sec 1: Customer Statement of Requirements (9 points)	100%					
Sec 2: System Requirements (6 Points)					50%	50%
Sec 3: Functional Requirements Specification (30 points)		50%		50%		
Sec 4: User Interface Specs (15 Points)			100%			
Sec 5: Domain Analysis (25 points)					50%	50%
Sec 6: Plan of Work (5 points)	100%					

** *Underlined & Italicized Percentages* indicate that the team member will in the future produce the specified work indicated in the box. Boxes which are *not* italicized or underlined indicate that the team member has already completed the specified work.

Individual Point Allocation

Team Member	Points / Estimated Points
Rylan	16
Avinash	16
Aakash	17
Mozam	17
Mandeep	17
Pintu	17

Individual Work Description, Project Management, & Notes

The following is a brief description of what each team member completed for Report #1:

Rylan:

- Wrote Customer Statement of Requirements & Glossary
- Gantt Charts / Timeline for Plan of Work (Report #2 + Coding Up To Demo #1)
- Project Management
    - Coordinated meetings / meeting times
    - Collated reports, documents, etc.
    - Represented group / contact point with TA & Dr. Marsic
    - Edited, modified styling, etc. on submitted documents

Avinash:

- Worked with Mozam to develop all use-cases, fully dressed by working with the previously created System Requirements Spec.
- Developed the use case diagrams needed to meet project requirements
- Editing of documents, meeting participation, commenting, suggestions, and document editing as needed
- Developed team Google Group / mailing list. Managed Dropbox share

Aakash:

- Developed comprehensive UI from mesh framework, and counted keystrokes / mouse clicks needed to complete tasks as defined in certain use cases
- Editing of documents, meeting participation, commenting, suggestions, and document editing as needed
- Created team website, handled updates to website

Mozam:

- Worked with Avinash to develop all use-cases, fully dressed by working with the previously created System Requirements Spec.
- Developed the use case diagrams needed to meet project requirements
- Editing of documents, meeting participation, commenting, suggestions, and document editing as needed

Mandeep:

- Worked with Pintu on developing System Requirements Spec. from the project proposal
- Co-Developed Domain Model
    - Attribute definitions, System operation contracts, Tracebility matrix, and concept definitions
- Editing of documents, meeting participation, commenting, suggestions, and document editing as needed
- Created, and updated References document for team

Pintu:

- Worked with Mandeep on developing System Requirements Spec. from the project proposal
- Co-Developed Domain Model
    - Domain model, Association definitions.
- Editing of documents, meeting participation, commenting, suggestions, and document editing as needed

NOTES:

- Although not credited, each team member did write at least one use case. However, all editing, and collating of use-cases is credited to Avinash and Mozam.

- Although not required (this work was complete prior to the Report #1 Requirements Change), Mozam created the System Sequence Diagrams for the Functional Requirements Spec. of the report.

- The use cases include 'mis-use cases'. Fully dressed use cases cover misuse of the system. The UI Specification also includes error handling, as developed by Aakash.

Table of Contents

1.0 Customer Statement of Requirements

Pig E-Bank
115 W 42nd Street
New York, NY 10036

Mr. Money Nickel
CEO, Pig E-Bank

The Virtual Stock Market Project, Group #6
Rutgers University

RE: Virtual Stock Market

Dear Project Group #6,

It is my pleasure to let you know that your team has been awarded a contract to develop a virtual stock market application for our bank. As your customer, we have a few requirements for the software which we would like to detail to better aid you in understanding our business requirements for the software.

As commonly known in our industry, there are a variety of virtual stock market applications available for use and purchase. However, each of these systems lack critical components which we, as a bank, need to use in order to better train our associates. Many associates who join our company out of college, understand basic market concepts, but lack the understanding of more complicated market products (mutual funds, options, etc.). These associates tend to lack the knowledge of related stocks. For example, they know about Apple (AAPL), but don't know about the background of similar companies, such as FoxConn, or Motorola who are major players in the Apple supply-chain. Finally, associates fail to know any basic investment strategies. Many simply know how to buy and sell stock, but cannot develop a portfolio which will provide them with a necessary rate of return to plan for their retirement in 20 – 40 years. Finally, some trainees do not understand how the market works. We need to be able to provide a colorful, interactive tutorial on how buying and selling stock works, trading, etc. Possibly in the future, we need the ability to add additional tutorials. We need the developed software to be able to answer those 3 major challenges. At some point, we plan on releasing this software to our customers. Customers generally don't know how to manage a 401k, or invest on their own. We want this software to be able to teach them to invest. At a high level, the system should have these basic features:

- Allow the buying and selling of multiple market products (including derivatives, stocks, and bonds)
- The ability to suggest users a stock / bond / investment strategy based on their portfolio holdings and using an analytic algorithm (such as found on Yahoo or Google Finance), or by taking a short survey which indicates their investment plans
- A fun & interactive tutorial system which can teach players about basic market operation, and can later be expanded to include new tutorials in the future
- The system should be 'easy-to-use', colorful, and fun

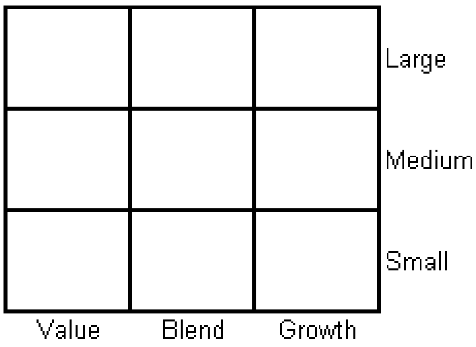
There are some general system requirements which we need:

- Ability to run on a web platform. We need to be able to access the game worldwide, without installing software.
- Host 'Investment Games', which are virtual games where users can play against each other by developing simple portfolios, and using buying and selling strategies to make money
- Provide 4 user roles (detailed later), 'Player', 'League Administrator', 'Administrator', 'Advertiser'
- A real-time trading system which gets market-prices within 5 seconds of actual accuracy (in the event there is a network outage, the system should be able to account for this by caching of prices)
- Mobile applications for BlackBerry, iOS, and Android which the user can use to play
- Simple registration system to make an account, and start a league
- Other virtual investment platforms are complex, and require multiple clicks to find simple data (e.g. a player's portfolio). The platform should allow all users (regardless of role) to be able to access critical data with minimal clicks.
- Connectivity to social media platforms. The platform should allow users to talk about it on Facebook or Twitter to develop an online presence
- Live ticket system to broker stocks, and perform complex orders such as limit orders
- The system should be able to support advertiser presence

For each user, the system should provide the following functional requirements:

- **Player:**
  - Ability to join multiple leagues at once, and participate in each game individually
  - The player should be able to view the league standings, see how other users are performing, view league settings (the start / end dates, the amount of start money, the portfolios of other users)
  - Develop (view, and modify) multiple portfolios (for each league), via the buying and selling of stock (via normal, shorted orders), options (put and call options), mutual funds, ETFs, and bonds
  - The portfolio should be clearly defined and have the player's holdings, ticker symbol, company name, shares / contracts owned, market value, and total portfolio value
  - Chat feature to talk to other users within the league
  - Access a tool to suggest a stock / portfolio (as discussed prior). The player should have the suggestions based on their portfolio, and based on taking a short survey with their investment goals (by time, and required return)
  - Check the price of a market product
  - Learn about how to invest via fun & interactive tutorials. The tutorials should be fun, colorful, and interactive
  - View a Morningstar™ report of their portfolio which reports risk vs. market capitalization (see the picture below)

Morningstar Style Box™



    - Access live finance news which may impact market prices
    - Ability to invite a friend to join their league
    - Functionality (which may be enabled / disabled / configured) to send a (daily / weekly) summary of the user's portfolio to their e-mail
    - Set their skill level, and change it as necessary. The skill level should change the layout of the page. The page layout (for an expert), should contain minimal direction. A beginner, however, should have pointers and guides on different parts of the page. If possible, an automated system to detect the user's skill level (preferably by a short quiz) would be preferable.
    - Access to a message board / portal where the user could ask questions which are not available for answer elsewhere
    - Challenge feature. The user should be able to lookup another user, and compare their portfolio (by terms of growth, stock, and returns). The user should be able to challenge them to a 1-on-1 game, where they can play each other in their own competitive league.
    - The player should be able to set a watch-list of stocks they may want to purchase. The watch-list should alert the user to the current market price of the stock, the day they added the stock, and a small note which the user can append to the stock.
- **League Administrator:**
    - Functionality to setup league settings (start and end date, initial funds), and league rules (which finance products can be bought – all, stocks only, etc. This should be accomplished by a check-box system or something similar).
    - A simple league management page where the users and settings can be managed.
    - Ability to invite users to play in a game
    - Option to kick a user out of a game, in the event they are being unruly
    - Message feature to message admins in the event of a problem
    - League comparison feature. The league administrator should be able to compare his league (total gain), vs. another league.
    - League challenge. The league administrator should be able to 'challenge' another league to a game.
    - Game type. The administrator should be able to set the game type. A normal game-type should end at the end-date of the game with the player with the most gains winning. A reverse game should give the player who loses the most money the win. This will teach users what are poor market strategies, and how difficult it can be to lose money.
    - The league administrator should have all the functionality of a player, and have all of the abilities of a player.
    - Leagues will all be set to public visibility (anyone, including non-league participants) should be able to see a league, members, and their portfolios

- **Administrator:**
    - The ability to delete a user, league, or advertiser
    - Functionality to disable / enable the platform for maintenance
    - A backup functionality to backup and restore the site
    - Functionality to disable the trading system in the event of a network issue, and switch to cached values if needed
    - Ability to check advertiser earnings, impressions, etc.
    - Enable / disable the advertisements on the website
- **Advertiser:**
    - Ability to upload a banner (image file), and a price per impressions value
    - Can check on how many impressions an ad has, and how many times each ad was clicked on
    - View their current advertisement bill owed to the site administrators

We are looking forward to seeing your development of these features and functionalities. If you have any questions about our requirements, feel free to contact at us at my above address.

Regards,

Money Nickel

Mr. Money Nickel
CEO, Pig E-Bank

1.1 Glossary of Terms

Bonds – A collective debt sold to investors in shares. Depending on safety of the debt, it provides a relatively low / medium rate of return on investment.

Derivatives – Market by-products (not stocks), but contracts such as Options which can also be traded in a market.

ETF – Exact duplicate of a mutual fund, and can be traded during investment hours.

Impression – When a user is shown an advertisement.

Investment – The process of buying securities, in hopes of growing the invested money for the future.

League – A group of investors who play against each other. They are ranked by the growth of their individual portfolios.

Market – An interactive forum for buying and selling financial products.

Market Capitalization – How much the market values a company. The market cap is defined by the share price times the number of outstanding shares.

Message Board – A portal where users can ask questions and message each other.

Morningstar™ - A company which specializes in market news.

Mutual Fund – A fund which takes an investor's money, and invests it collectively, providing an equal return to each investor. A mutual fund cannot be traded during market hours.

Options – A contract which allows you to buy / sell a set amount of shares in the market at some point in the future, at a set price. The option is simply a contract, and is bought in 100 share increments.

Order / Limit Order – *See Trade*. A Limit Order is a trade set to execute when the market price of a security reaches a specified price.

Portfolio – A collection of stocks, bonds, derivatives, and mutual funds owned by a player. The value of the portfolio is the sum value of its contents.

Risk – The qualitative property of a security with respect to how probable it may or may not grow money over time. Typically, stocks are considered to have more risk than certain bonds. Whereas options are even more risky.

Security – A market product such as a stock, bond, ETF, Mutual Fund, Option, etc. which has some monetary value.

Share – A fraction of a publicly owned company which may be traded in a market.

Stocks – A share in a publicly owned company. The share can be bought by a player, and put into their portfolio.

Ticket / Trading System – A system which takes a user's trades and processes them. It exchanges the user's money in the portfolio for a security. The system is able to lookup the value of a security at a given time.

Trade – A transaction where a user exchanges funds (money) for a security.

2.0 System Requirements

2.1 Functional Requirements

ID	PW	Requirement
REQ-1	5	The system shall allow new Players to register an account with their email, which should be external to our website. Required information shall include a unique username, password that meets the guidelines, as well as Player's first and last name, birth date and gender. Upon completion of successful registration, the Player account balance shall be decided by Game Administrator.
REQ-2	5	The system shall support placement of order by filling out an order ticket. The order ticket should contain client's information, order type, quantity, price and additional instructions. The system shall periodically review the queued orders process them when conditions are met.
REQ-3	5	The system shall review the order queue periodically and: 1. If all the conditions are matched, convert order into a market order and execute. 2. Else if, the order is expired or cancelled, remove from the queue and mark it failed. 3. Else, none of above, leave untouched. If either 1 or 2 is executed, the system shall record the transaction and notify the Player by sending a confirmation message.
REQ-4	5	The system shall maintain a database of Player portfolios and transactions. The database will also include league status for each player.
REQ-5	4	The system shall support creation of new leagues or entry to existing leagues. Players shall be allowed to create leagues and specify duration, capital limits, allowed sectors and entrance fees. The system shall also keep track of leagues' status based on investment returns.
REQ-6	4	The system shall provide market data (price data, bid/ask sizes, volume and news feed of relevant articles) for set of companies.
REQ-7	4	The system shall allow users to create and manage Funds. The rules of a Fund are specified when the Fund is created. These rules include the types of trades they are allowed to do and the types of assets they are allowed to hold.
REQ-8	3	The system shall contain learning tutorials based on Player's skill levels.

REQ-9	2	The system shall allow players to share their status on social media.
REQ-10	1	The system shall allow Players to submit technical problems and comments to the system administrator.
REQ-11	2	The system shall allow current players to refer friends.
REQ-12	2	The system shall allow advertisements of different organizations and companies.
REQ-13	4	The system shall suggest different securities and stocks based on the Player's portfolio.

2.2 Nonfunctional Requirements

ID	**PW**	**Requirement**
REQ-14	5	The system shall be simple to understand and use with minimal knowledge of a Player's learning curve. The layout of the page should be simple and easy to understand, and contain most of the contents on fewer pages.
REQ-15	5	The system shall maintain and store all the data and information on the system's database and not allow any data or information to be stored on Player's device. The system shall not allow Player to directly modify any data. Two copies of any record shall be kept in case of a failure.
REQ-16	3	The system shall be able to run on different platforms such as Windows, Unix, or Mac. It should the same theme and consistency between different browsers.
REQ-17	4	The system shall be efficient as possible, allowing Players to start a game within 5 clicks, buy a stock within 3 clicks, and view a portfolio in 2 clicks.

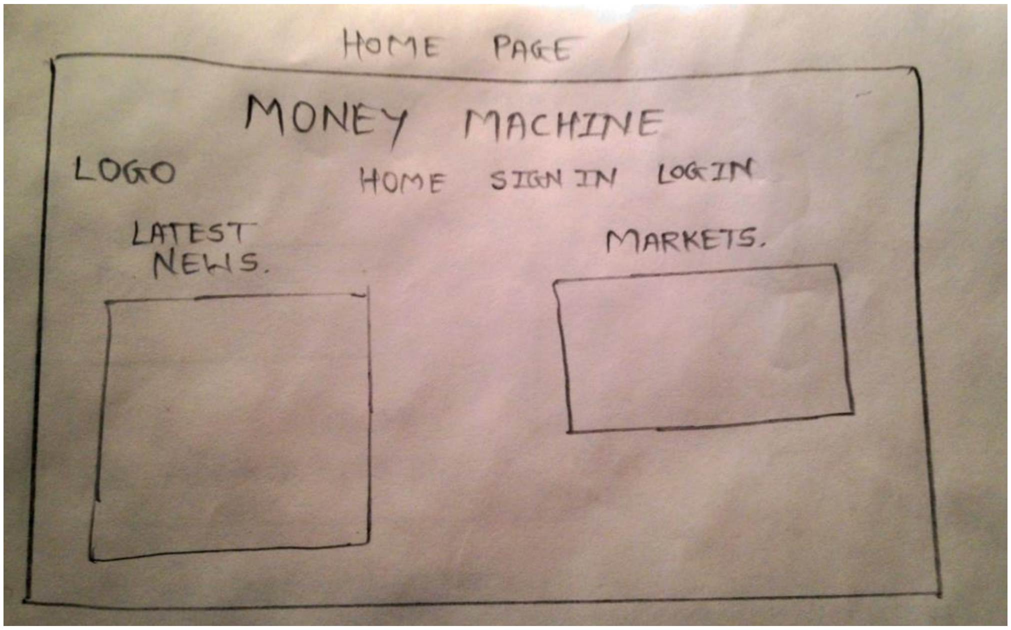
2.3 On-Screen Appearance Requirements

ID	**PW**	**Requirement**
REQ-18	5	The system must fit within a browser window of any browser.
REQ-19	3	The system must have a consistent look across different browsers and screen resolutions.
REQ-20	3	Advertisement should be adhere to the system administrator guidelines.

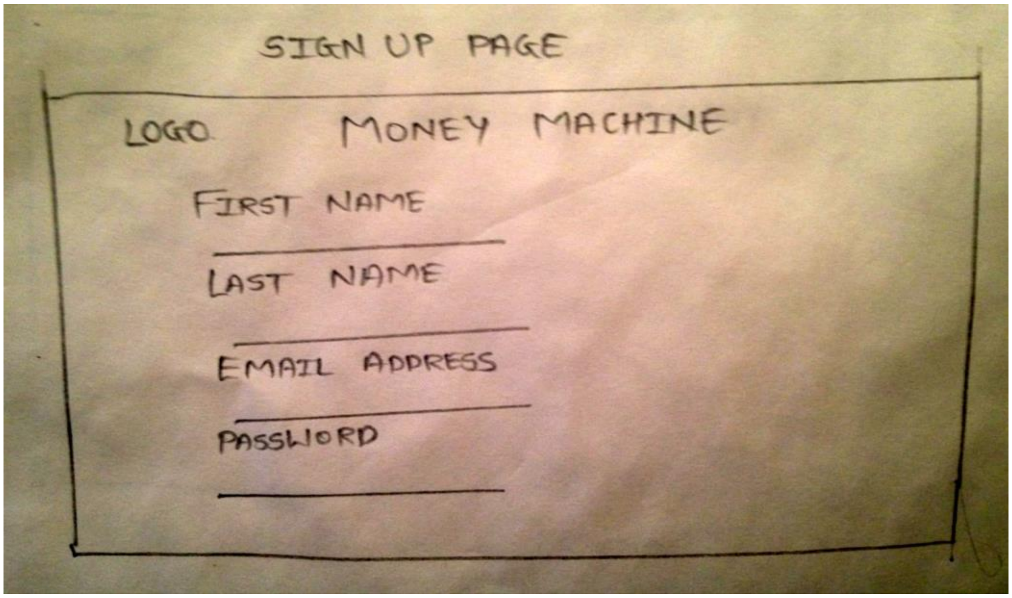
2.3.1 Screen Mockups

The following are mockups of specific pages from the project. They provide a rough idea of how specific, important pages of the project will look.

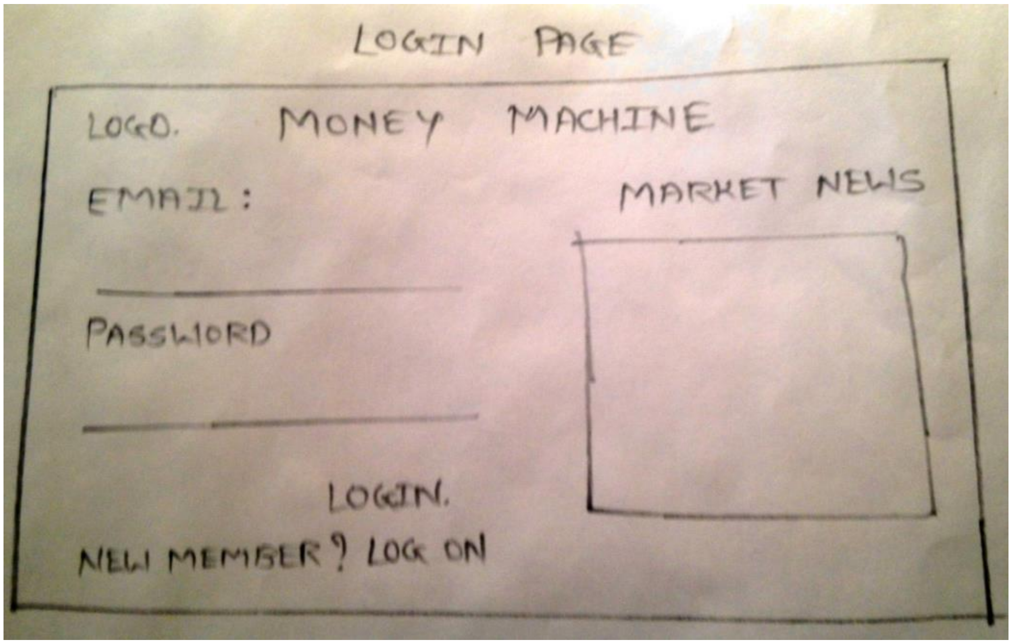
<u>Home Page:</u>



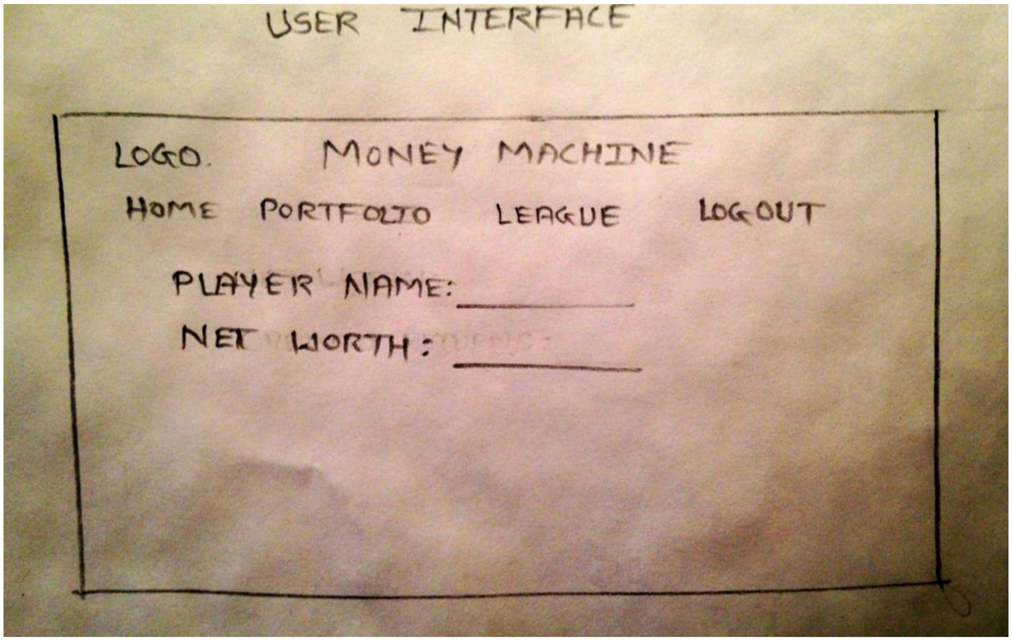
<u>Sign Up Page:</u>



Login Page:



League Interface Page (Dashboard):



3.0 Functional Requirements Specification

3.1 Stakeholders

- Potential Investors
- System Administrators
- Advertisers

3.2 Actors & Goals

- **Player**:
  Type - Initiating Actor, Participating Actor
  Goals - Access security information, buy and sell securities, create investment games, and view watch lists

- **Visitor**:
  Type - Initiating Actor
  Goal - To register for full access to the system.

- **Game Administrator**:
  Type - Initiating Actor, Participating Actor
  Goals - Manage an investment game. Start or end an investment game.

- **Advertiser**:
  Type - Initiating Actor, Participating Actor
  Goals - Add/Remove an advertisement, rotate advertisements, View advertising statistics

- **System Administrator**:
  Type - Initiating Actor, Participating Actor
  Goals - Maintain web presence, view suggestions from players, and provide strategic enhancements to website operations.

- **Trade Database**:
  Type - Participating Actor

- **Player Database**:
  Type - Participating Actor

- **Security Data Provider**:
  Type - Participating Actor
  Goals - Provide information in relation to securities. Handle trade creation and modification.

- **Web Server**:
  Type - Participating Actor

3.3 Use Cases

3.3.1 Casual Description

Use Case UC-1: Register

Actor: Visitor (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To register for a new account. A new player account will be created based on information provided from the visitor.

Use Case UC-2: Research Security

Actor: Player (Initiating), Security Data Provider (Participating), Web Server (Participating)
Goal: To provide information such as last price, bid/ask prices, fundamentals, charts, news, etc. Such information will be provided mainly from the Security Data Provider.

Use Case UC-3: Buy Security

Actor: Player (Initiating), Security Data Provider (Participating), Trade Database (Participating), Player Database (Participating), Web Server (Participating)
Goal: To purchase a security such as a bond, stock, option, etc. This will generate an order ticket which will contain order type (market, limit, buy to close, etc.) , security name/ ID, execution price, and time to expiry (Good Until Cancelled or Day Order). Prices will be provided from the Security Data Provider.

Use Case UC-4: Sell Security

Actor: Player (Initiating), Security Data Provider (Participating), Trade Database (Participating), Web Server (Participating)
Goal: To sell a security such as a bond, stock, option, etc. This will generate an order ticket which will contain order type (market, limit, sell to open, etc.), security name/ID, execution price, and time to expiry (Good Until Cancelled or Day Order). Prices will be provided from the Security Data Provider.

Use Case UC-5: View Portfolio

Actor: Player (Initiating), Security Data Provider (Participating), Trade Database (Participating), Player Database (Participating), Web Server (Participating)
Goal: To view current securities held, as well as available cash to withdraw/invest. This will be displayed for each league the player is a part of. Will also display current value of portfolios.

Use Case UC-6: View Transactions

Actor: Player (Initiating), Trade Database (Participating), Web Server (Participating)
Goal: To show pending, filled and cancelled transactions for the player.

Use Case UC-7: Create Investment Game

Actor: Player (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To create games where an initiating player becomes the game administrator of the created game.

Use Case UC-8: Join Investment Game

Actor: Player (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To join an investment game.

Use Case UC-9: Invite into Investment Game

Actor: Game Administrator (Initiating), Player Database (Participating), Web Server (Participating)

Goal: Invite players to join the investment game.

Use Case UC-10: Manage Investment Game
Actor: Game Administrator (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To add/remove players from the game as well as accept/decline requests to join game.

Use Case UC-11: Manage Portfolio
Actor: Player (Initiating), Player Database (Participating), Trade Database (Participating), Web Server (Participating)
Goal: To buy/sell & research securities.

Use Case UC-12: Manage Advertisers
Actor: System Administrator (Initiating), Advertiser (Participating), Web Server (Participating)
Goal: To manage and authorize advertisers.

Use Case UC-13: Manage Advertisements
Actor: Advertiser (Initiating), System Administrator (Participating), Web Server (Participating)
Goal: To change layout and frequency of the advertisements.

Use Case UC-14: Suggest Security
Actor: Player (Initiating), Trade Database (Participating), Player Database (Participating), Web Server (Participating)
Goal: To suggest a security to the player based on current market information as well as news.

Use Case UC-15: Challenge Player
Actor: Player (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To create a closed investment game where two players can go head to head against each other to build the most valuable/ highest growth portfolio.

Use Case UC-16: View Watch List
Actor: Player (Initiating), Trade Database (Participating), Player Database (Participating), Web Server (Participating)
Goal: To watch and track various security prices for securities which they may/may not have in their portfolio.

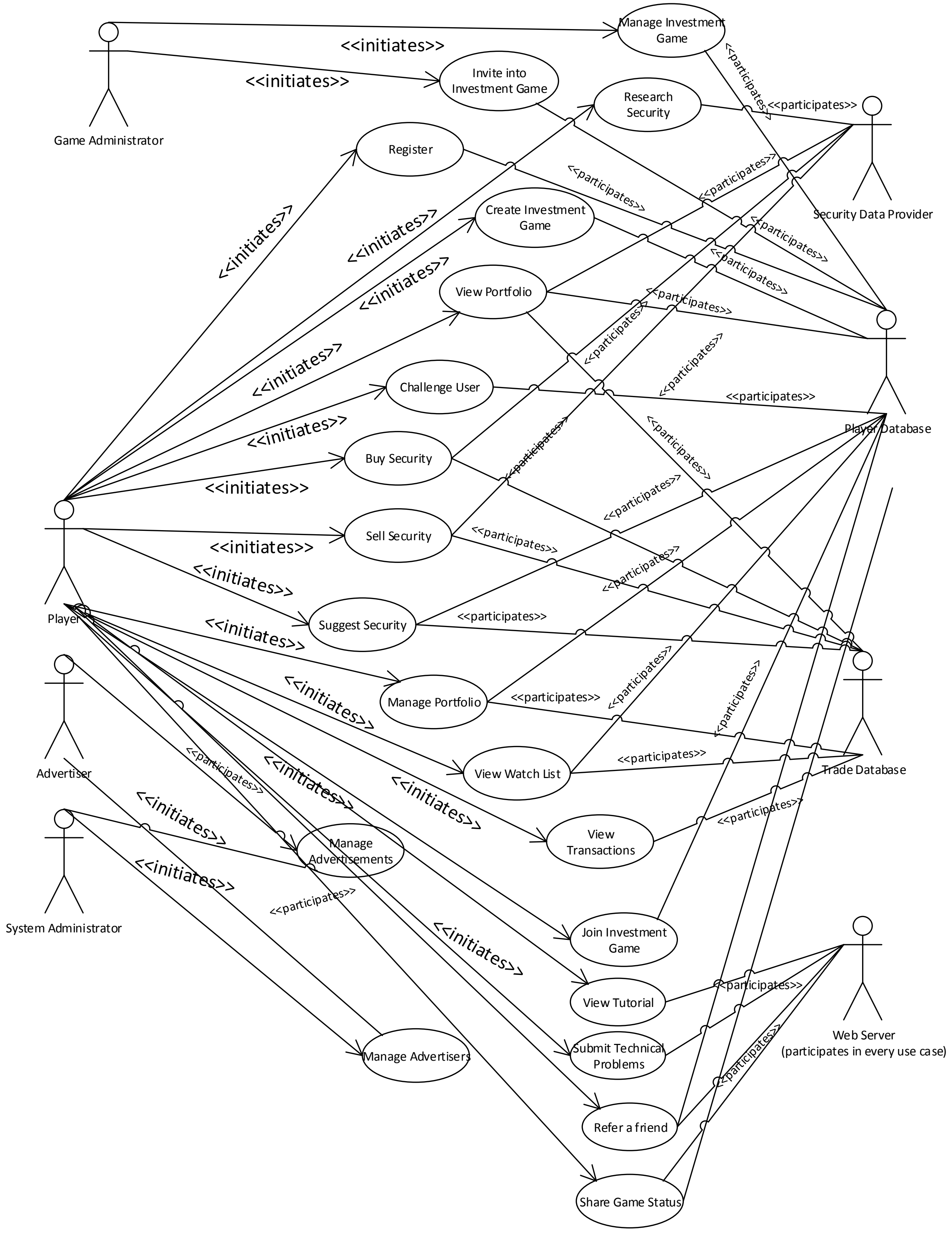
Use Case UC-17: View Tutorial
Actor: Player (Initiating), Web Server (Participating)
Goal: To enhance the player's trading knowledge in regards to their inputted skill level.

Use Case UC-18: Submit Technical Problems/Suggestions
Actor: Player (Initiating), Web Server (Participating)
Goal: To allow a Player to submit technical issues and suggestions about the system.

Use Case UC-19: Refer a Friend
Actor: Player (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To allow a Player to refer a friend.

Use Case UC-20: Share Game Status
Actor: Player (Initiating), Player Database (Participating), Web Server (Participating)
Goal: To allow a Player to share their game status with anyone on social media.

3.3.2 Use Case Diagram



3.3.3 Traceability Matrix

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13		
PW	**5**	**5**	**5**	**5**	**4**	**4**	**4**	**3**	**2**	**1**	**2**	**2**	**4**	Max	Total
UC-01	x			x										5	10
UC-02						x								4	4
UC-03		x	x	x		x								5	19
UC-04		x	x	x		x								5	19
UC-05				x										5	5
UC-06			x	x										5	10
UC-07				x	x		x							5	13
UC-08				x	x									5	9
UC-09				x	x									5	9
UC-10				x	x									5	9
UC-11				x										5	5
UC-12												x		2	2
UC-13												x		2	2
UC-14				x		x							x	5	13
UC-15	x			x	x		x							5	18
UC-16				x		x								5	9
UC-17								x						3	3
UC-18										x				1	1
UC-19											x			2	2
UC-20									x		x			5	5

3.3.4 Fully-Dressed Description

Use Case UC-1: Register

Related Requirements: REQ-1

Initiating Actor: Visitor

Initiating Actor's Goal: To register for a new account. A new player account will be created based on information provided from the visitor.

Participating Actors: Player Database, Web Server

Precondition: The visitor does not already have an account in the system.

Postcondition: The visitor successfully creates a new player profile and an appropriate entry is created in the Player database.

Flow of Events for Main Success Scenario:

1 → The visitor clicks the "Register" button or the visitor attempts to access a feature that is only for members.
2 ← The system provides the visitor with the registration page.
3 → The visitor submits the information to the system.
4 ← The system verifies the visitor's information and inserts this information into the Player Database.
5 ← The system provides confirmation to the visitor that their information was valid and a new profile was created successfully.

Flow of Events for Username/Email already in use:

1 → The visitor clicks the "Register" button or the visitor attempts to access a feature that is only for Players.
2 ← The system provides the visitor with the registration page.
3 → The visitor submits the information to the system.
4 ← The system attempts to verify the information. It finds that the username or email address is

already in use.
5 ← The system generates an error and presents the registration page back to the user for editing.

Use Case UC-3: Buy Security

Related Requirements: REQ-2, REQ-3, REQ-4, REQ-6, REQ-7
Initiating Actor: Player
Initiating Actor's Goal: To buy to close or buy to open a position.
Participating Actors: Player Database, Trade Database, Security Data Provider, Web Server
Precondition: Player must have enough balance to purchase the security.
Postcondition: The cost of the order is debited from the player's total balance and an order ticket is generated.

Flow of Events for Success Scenario

1 → The player chooses a security from their portfolio or from one of the supported markets.
2 ← The Security Data Provider retrieves information for the chosen security such as bid/ask spread, last price, volume traded, and other metrics for the security.
3 → The player then fills out information such as order type, expiry date and number of securities to buy.
4 ← An order ticket is generated and forwarded to the trade database for processing and order confirmation is provided to the player as well as a temporary hold is placed on the player's account for the cost of the order.
5 ← The player is notified when their order is filled. The cost of the order is debited from the user's portfolio and corresponding security is added to the portfolio.

Flow of Events for Insufficient Funds

1 → The player chooses a security from their portfolio or from one of the supported markets.
2 ← The Security Data Provider retrieves information for the chosen security such as bid/ask spread, last price, volume traded, and other metrics for the security.
3 → The player then fills out information such as order type, expiry date and number of securities to buy.
4← The system determines that the player has insufficient funds to buy security. The order ticket is destroyed and order is re-forwarded to the player for editing.

Use Case UC-4: Sell Security

Related Requirements: REQ-2, REQ-3, REQ-4, REQ-6, REQ-7
Initiating Actor: Player
Initiating Actor's Goal: To sell a security such as a bond, stock, option, etc to close or open a position.
Participating Actors: Security Data Provider, Trade Database, Web Server
Precondition: The player must either own the security, or must have enough money to put into a margin account.
Postcondition: The cost of the order is credited to the player's total balance and an order ticket is generated.

Flow of Events for Main Success Scenario:

1 → The player chooses a security from their portfolio or from one of the supported markets.
2 ← The Security Data Provider retrieves information for the chosen security such as bid/ask spread, last price, volume traded, and other metrics from the security.
3 → The player then fills out information such as order type, expiry date and number of securities to sell.
4 ← An order ticket is generated and inserted into the trade database. The player is provided with an order confirmation that confirms their order has been placed.

5 ← The player is notified when their order is filled. The cost of the order is credited to the player's portfolio, and the corresponding security (if the player owned it originally) is removed.

Flow of Events For Not Enough Margin:

1 → The player chooses a security from their portfolio or from one of the supported markets.
2 ← The Security Data Provider retrieves information for the chosen security such as bid/ask spread, last price, volume traded, and other metrics from the security.
3 → The player then fills out information such as order type, expiry date and number of securities to sell.
4 ← The system determines that the player does not have enough shares and does not have a high enough balance in their margin account. The order ticket is not placed for processing and is presented to the player for editing.

Flow of Events for Not Enough Stock (with no margin account):

1 → The player chooses a security from their portfolio or from one of the supported markets.
2 ← The Security Data Provider retrieves information for the chosen security such as bid/ask spread, last price, volume traded, and other metrics from the security.
3 → The player then fills out information such as order type, expiry date and number of securities to sell.
4 ← The system determines that the player does not have enough shares and does not have a margin account. The order ticket is not placed for processing and is presented to the player for editing.

Use Case UC-7: Create Investment Games

Related Requirements: REQ-5
Initiating Actor: Player
Initiating Actor's Goal: To initiate an investment game
Participating Actors: Player Database, Web Server
Precondition: Initiating player must be a registered user
Postcondition: The initiating player must become a game coordinator for the specific game and the game should be created.

Flow of Events for Main Success Scenario:

1 → The initiating player clicks on the tab "Create Game" and gets prompted to fill out a form which includes title of the game, a web url which directly links to the game, comment block (optional) and an expiry date .
2→ The game has a unique title and above field data is forwarded to the player database.
3 ← New data would be added to the player database and the player becomes the game administrator for the initiated game.
4 ← Player is notified that the game has been created.

Flow of Events for Duplicate Game Title

1 → The initiating player clicks on the tab "Create Game" and gets prompted to fill out a form which includes title of the game, a web url which directly links to the game, comment block (optional) and an expiry date .
2 ←The game title already exists in the player database and the initiating player is notified an "Invalid Name" error.
3 ← The Player is redirected to create game.

Use Case UC-14: Suggest Security

Related Requirements: REQ-13
Initiating Actor: Player
Initiating Actor's Goal: To obtain a list of suggested securities based on portfolio, risk appetite, and current conditions.
Participating Actors: Trade Database, Player Database, Web Server

Precondition: None
Postcondition: The player is provided with a list of suggested securities.
Flow of Events for Main Success Scenario:
1 → The player provides their risk appetite to the system.
2 → The Security Data Provider generates a list of securities which are to be suggested based on the current market conditions and the player's portfolio.
3 ← The system reads this result, and filters the returned list based on the risk appetite provided by the player.
4 ← The player is provided the list of securities that meet their goals.

Use Case UC-15: Challenge Player
Related Requirements: REQ-4, REQ-5
Initiating Actor: Player
Initiating Actor's Goal: To create a closed investment game where two players can go head to head against each other to build the most valuable/highest growth portfolio.
Participating Actors: Player Database, Web Server
Precondition: The challenger and challenged must both be in the Player Database
Postcondition: A new closed investment game consisting of both players is created.
Flow of Events for Main Success Scenario:
1 → The player requests to challenge another player.
2 ← The system requests the username of the user to challenge.
3 → The system queries the Player Database to see if there is a player with the username.
4 ← The Player Database confirms that the player exists. A new closed investment game is created, and the challenged player receives an invitation to the closed investment game.

Flow of Events for Non-Existent username:
1 → The player requests to challenge another player.
2 ← The system requests the username of the user to challenge.
3 → The system queries the Player Database to see if there is a player with the username.
4 ← The Player Database gives an error "The username does not exist". The system returns to Step 2.

3.4 System Sequence Diagrams

UC-1: Register

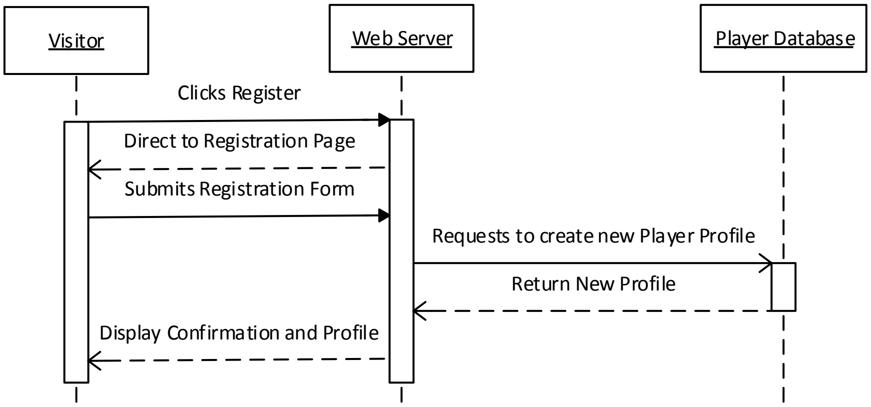
In this system sequence diagram, the visitor first navigates to the website. After reaching the website, the visitor clicks “Register”. After this, the visitor is presented with the registration page.

Once the user has submitted the registration page, the information provided is validated and is sent to the Player Database. The system then requests for a new player profile to be created for the visitor. The system then returns to the visitor that their profile creation was complete, and that they are now logged into the system.

The only alternate scenario to the main success scenario would be if any of the information entered by the user was invalid. In this situation, the system would return an error to the Visitor letting them know that there was an error in their submission. It would give the user another chance to submit the registration form.

As of now, no other implementations have been discussed, as the current one seems to be the most logical flow of events.

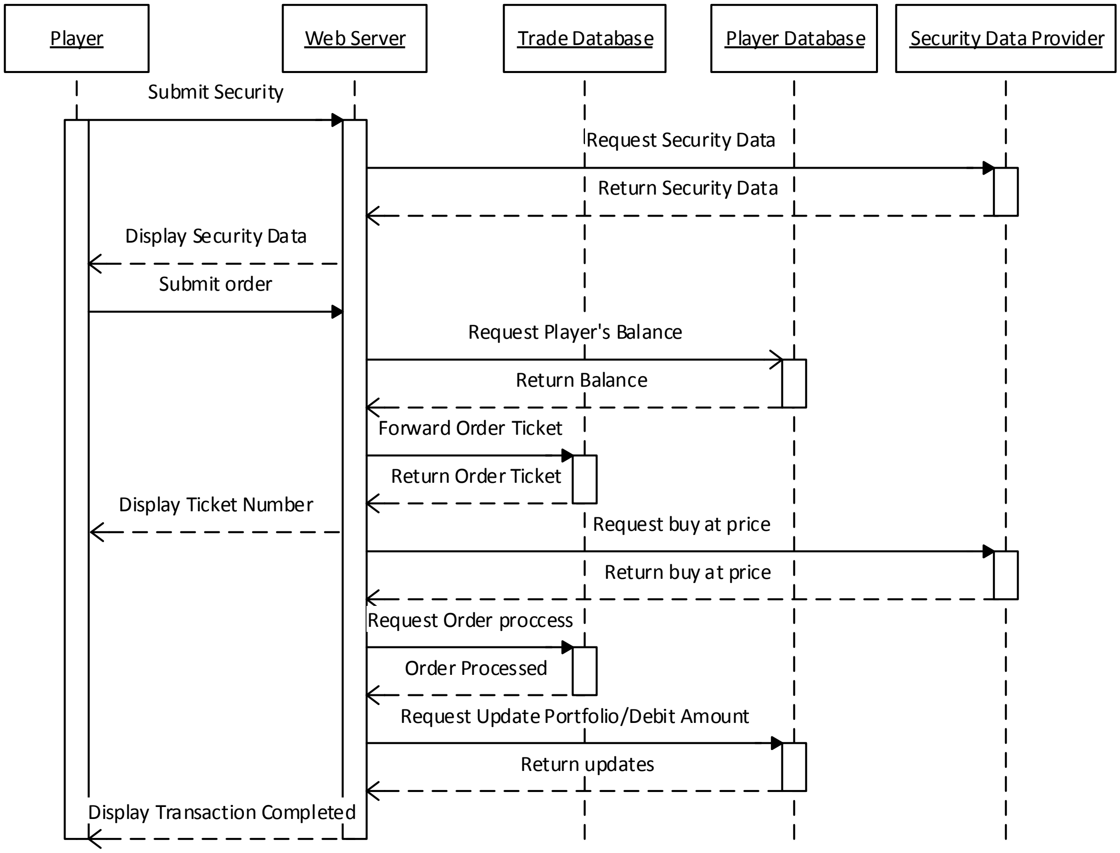
UC-1 System Sequence Diagram:


Visitor
Web Server
Player Database
Clicks Register
Direct to Registration Page
Submits Registration Form
Requests to create new Player Profile
Return New Profile
Display Confirmation and Profile

UC-3: Buy Security

In this system, the player selects a security of interest and in return the system should display the latest data of the security. This is done when the web server connects to the security data provider to read the data. The data is then displayed to the player. Then, the player must fill out order form which most importantly includes the buying price of the security. Upon clicking submit, the web server verifies if the player has enough balance to purchase. If the player has enough balance then the system requests an order ticket from the trade database for the particular security (securities). The player is displayed with a confirmation of order and order ticket number. The Web Server constantly reads the data of the particular security through the security data provider. Once, the parameters of the player match the current data, the system requests the trade database to process the order. Order is then processed and player's portfolio is updated. A notification is also sent to the player informing that the transaction is completed.

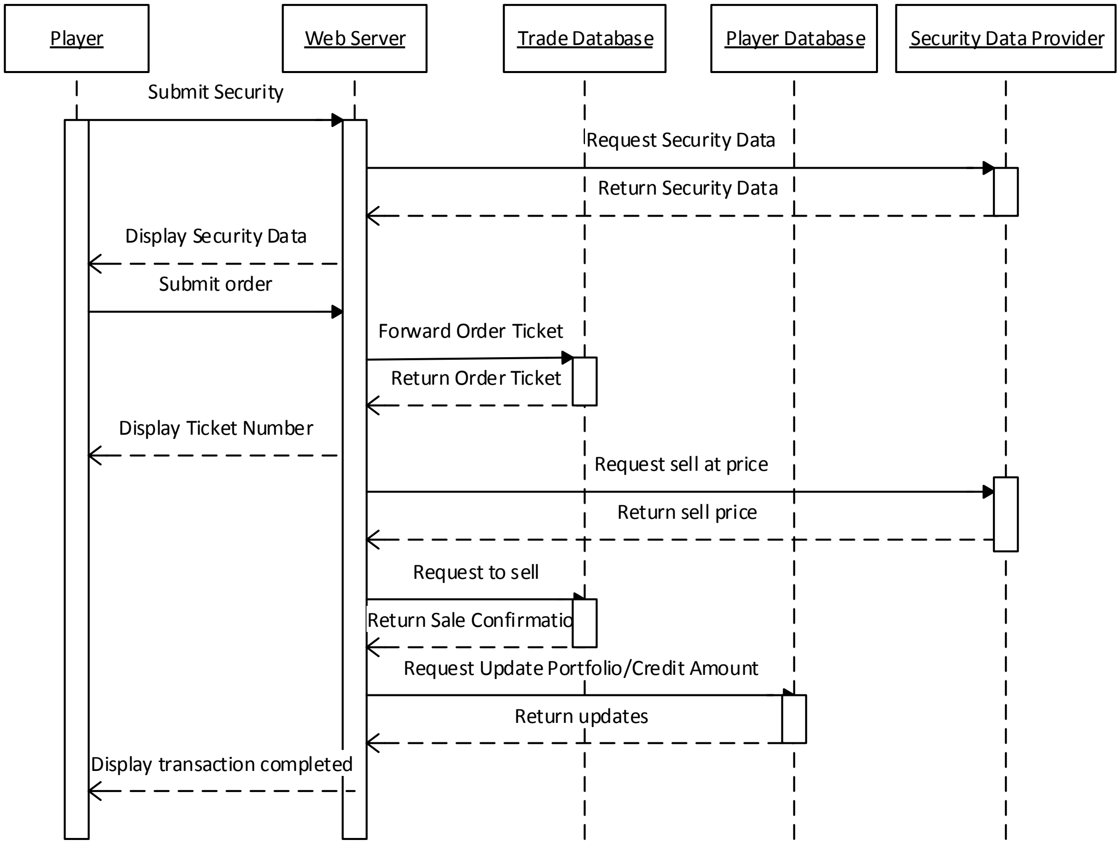
UC-3 System Sequence Diagram:



UC-4: Sell Security

In this system, the player selects a security of interest and in return the system should display the latest data of the security. This is done when the web server connects to the security data provider to read the data. The data is displayed to the player. Then, the player must fill out order form which most importantly includes the selling price of the security. A request for generating an order ticket is then sent out to the trade database. Once the order ticket has been generated the player is displayed confirmation of the order and the ticket number. The system constantly reads the data through the security data provider and once the price to sell his matched with the user's parameters the order is sent to be processed to the trade database.
Order is then processed and player's portfolio is updated. A notification is also sent to the player informing that the transaction is completed.

UC-4 System Sequence Diagram:

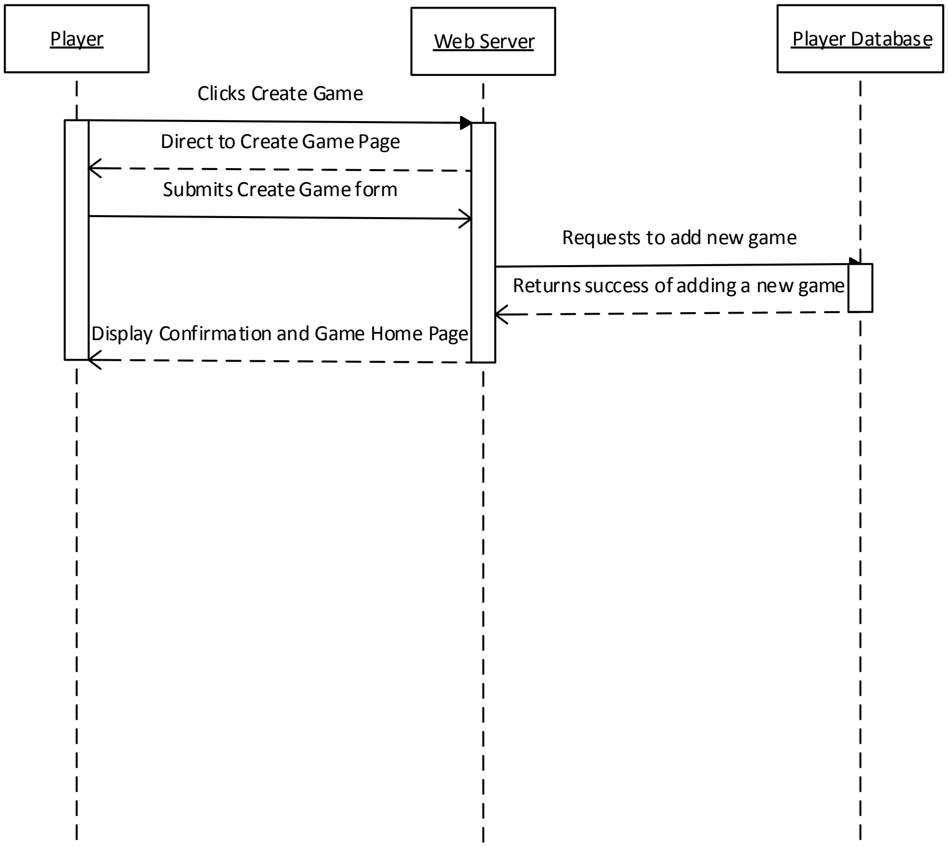


UC-7: Create Game

In this system sequence diagram, the Player requests for a new game to be created. The system then presents the user with the Create Game page. One the user has submitted the registration page, the information is validated and a request to create a new game is sent to the Player Database. The system then updated required fields in the Player Database and signals a success to the Web Server. The Web Server signals this back to the user with a confirmation that their game has successfully been create.

The only alternate scenario to the main success scenario would be if the game name the Player is trying to make is already taken. In this situation, the system would return an error after form submission letting the user know that their game name is already taken. It would ask the Player to choose another game name and go through the same process of revalidating.
As of now, no other implementations have been discussed, as the current one seems to be the most logical flow of events.

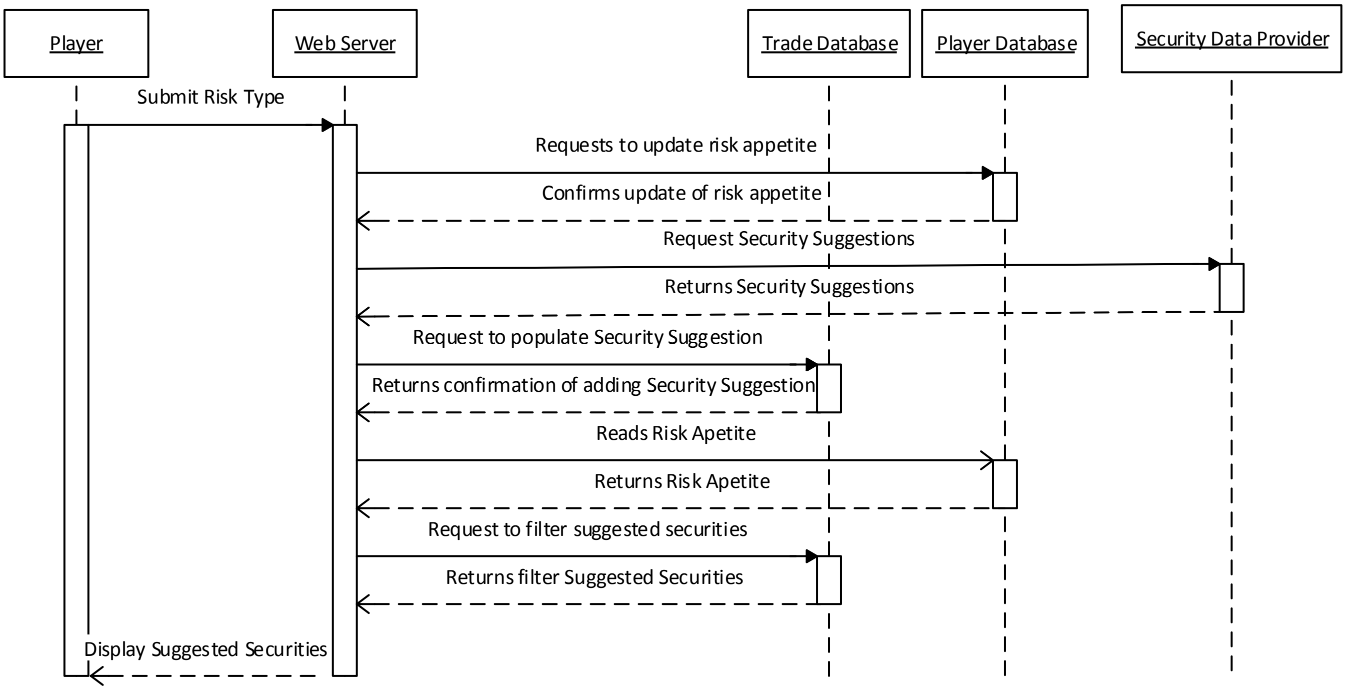
UC-7 System Sequence Diagram:


Player
Web Server
Player Database
Clicks Create Game
Direct to Create Game Page
Submits Create Game form
Requests to add new game
Returns success of adding a new game
Display Confirmation and Game Home Page

UC-14: Suggest Security

Each player has to select a risk appetite and submit to the user. This risk appetite is added to the player's database. Then, the system requests for security suggestions depending on the latest news, technical and fundamental analysis etc. through the Security Data Provider. A collection of the data is fed to the trade database. The system then reaches out to the player database to read the risk appetite and requests the trade database to filter out suggestions according to priority and the risk type requested. Finally, the suggestions are displayed to the user.

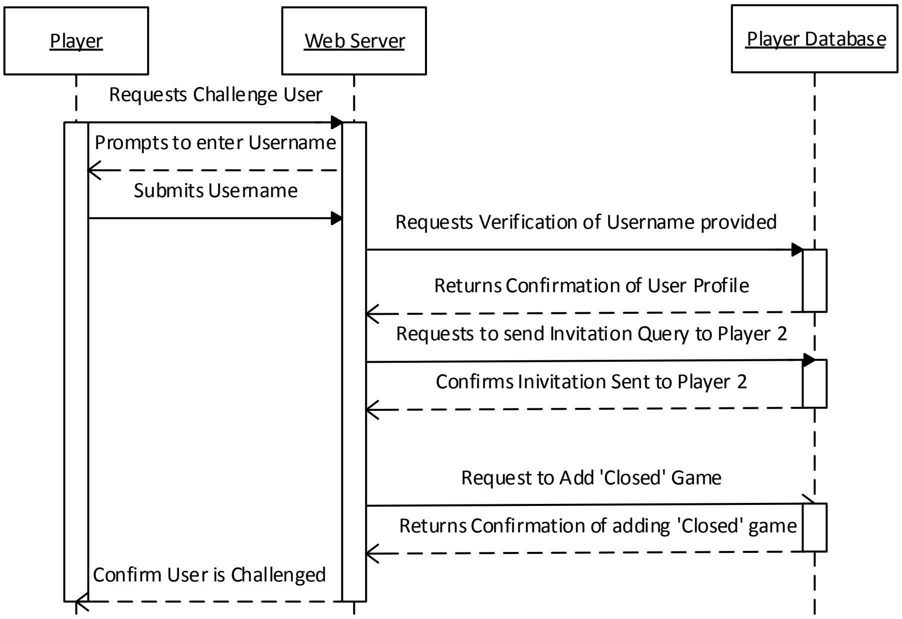
UC-14 System Sequence Diagram:


Player
Web Server
Trade Database
Player Database
Security Data Provider
Submit Risk Type
Requests to update risk appetite
Confirms update of risk appetite
Request Security Suggestions
Returns Security Suggestions
Request to populate Security Suggestion
Returns confirmation of adding Security Suggestion
Reads Risk Apetite
Returns Risk Apetite
Request to filter suggested securities
Returns filter Suggested Securities
Display Suggested Securities

UC-15: Challenge User

A player should be able to challenge other player(s). When the player requests to challenge another player, the system prompts the username of the player. The system verifies if the player exists in the player database. Once the verification is done the system sends a notification of challenge to Player 2. The portfolio of the two players get updated to yet another inner game (Closed Game) between the two. A notification is sent out to the initiating player saying that the challenge has started.

UC-15 System Sequence Diagram:


Player
Web Server
Player Database
Requests Challenge User
Prompts to enter Username
Submits Username
Requests Verification of Username provided
Returns Confirmation of User Profile
Requests to send Invitation Query to Player 2
Confirms Inivitation Sent to Player 2
Request to Add 'Closed' Game
Returns Confirmation of adding 'Closed' game
Confirm User is Challenged

4.0 User Interface Specification

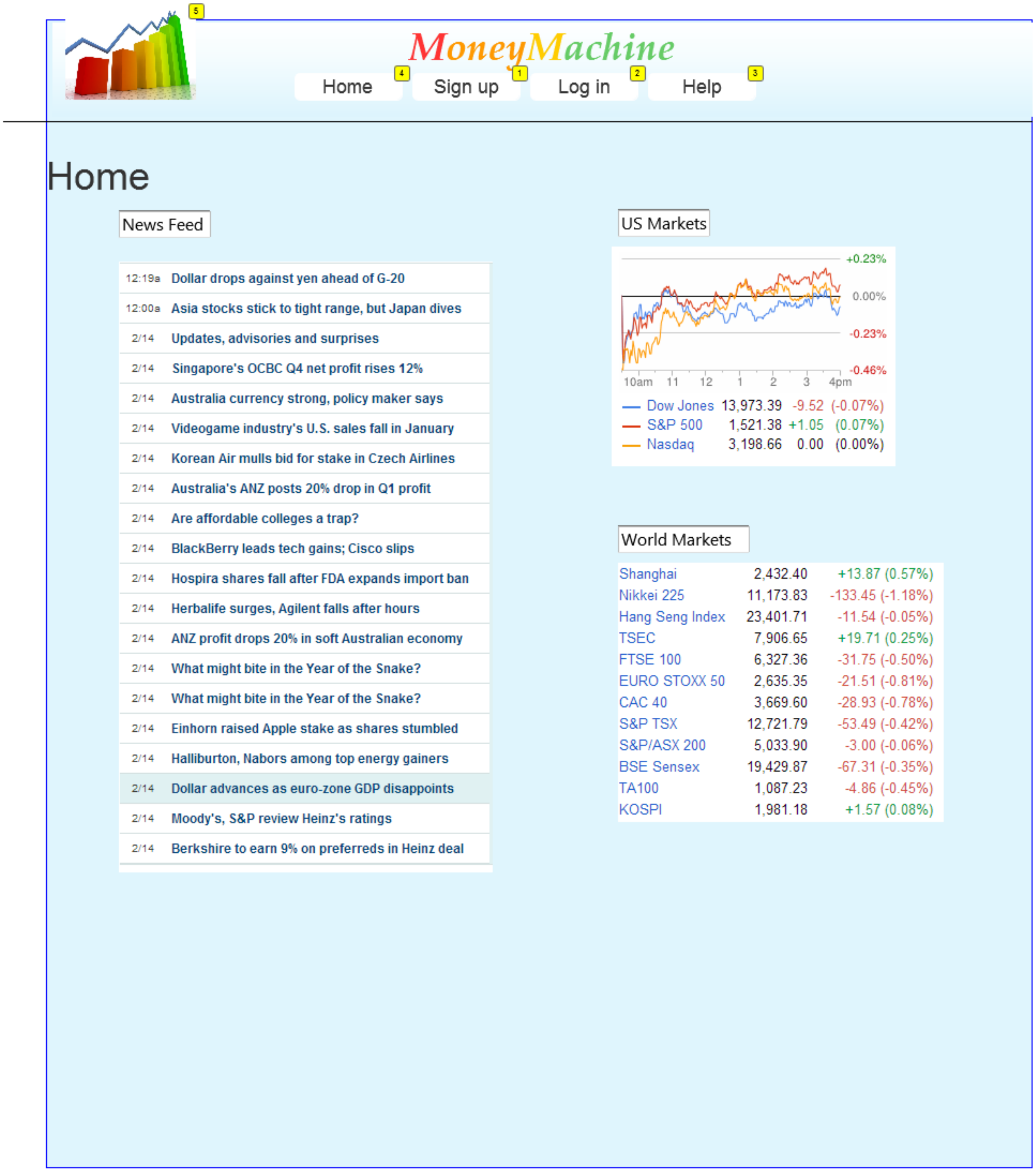
The following are the pages a UI Specification was developed for:

- Home
- Sign Up
- Login Page
- Help Page
- Tutorial Page
- After Logon Page
- League Page

Each page consists of a diagram, tabs (if applies), a click table, and errors (as needed).

4.1 Home

4.1.1 User Interface



Home

News Feed

12:19a	Dollar drops against yen ahead of G-20
12:00a	Asia stocks stick to tight range, but Japan dives
2/14	Updates, advisories and surprises
2/14	Singapore's OCBC Q4 net profit rises 12%
2/14	Australia currency strong, policy maker says
2/14	Videogame industry's U.S. sales fall in January
2/14	Korean Air mulls bid for stake in Czech Airlines
2/14	Australia's ANZ posts 20% drop in Q1 profit
2/14	Are affordable colleges a trap?
2/14	BlackBerry leads tech gains; Cisco slips
2/14	Hospira shares fall after FDA expands import ban
2/14	Herbalife surges, Agilent falls after hours
2/14	ANZ profit drops 20% in soft Australian economy
2/14	What might bite in the Year of the Snake?
2/14	What might bite in the Year of the Snake?
2/14	Einhorn raised Apple stake as shares stumbled
2/14	Halliburton, Nabors among top energy gainers
2/14	Dollar advances as euro-zone GDP disappoints
2/14	Moody's, S&P review Heinz's ratings
2/14	Berkshire to earn 9% on preferreds in Heinz deal

US Markets

Dow Jones	13,973.39	-9.52	(-0.07%)
S&P 500	1,521.38	+1.05	(0.07%)
Nasdaq	3,198.66	0.00	(0.00%)

World Markets

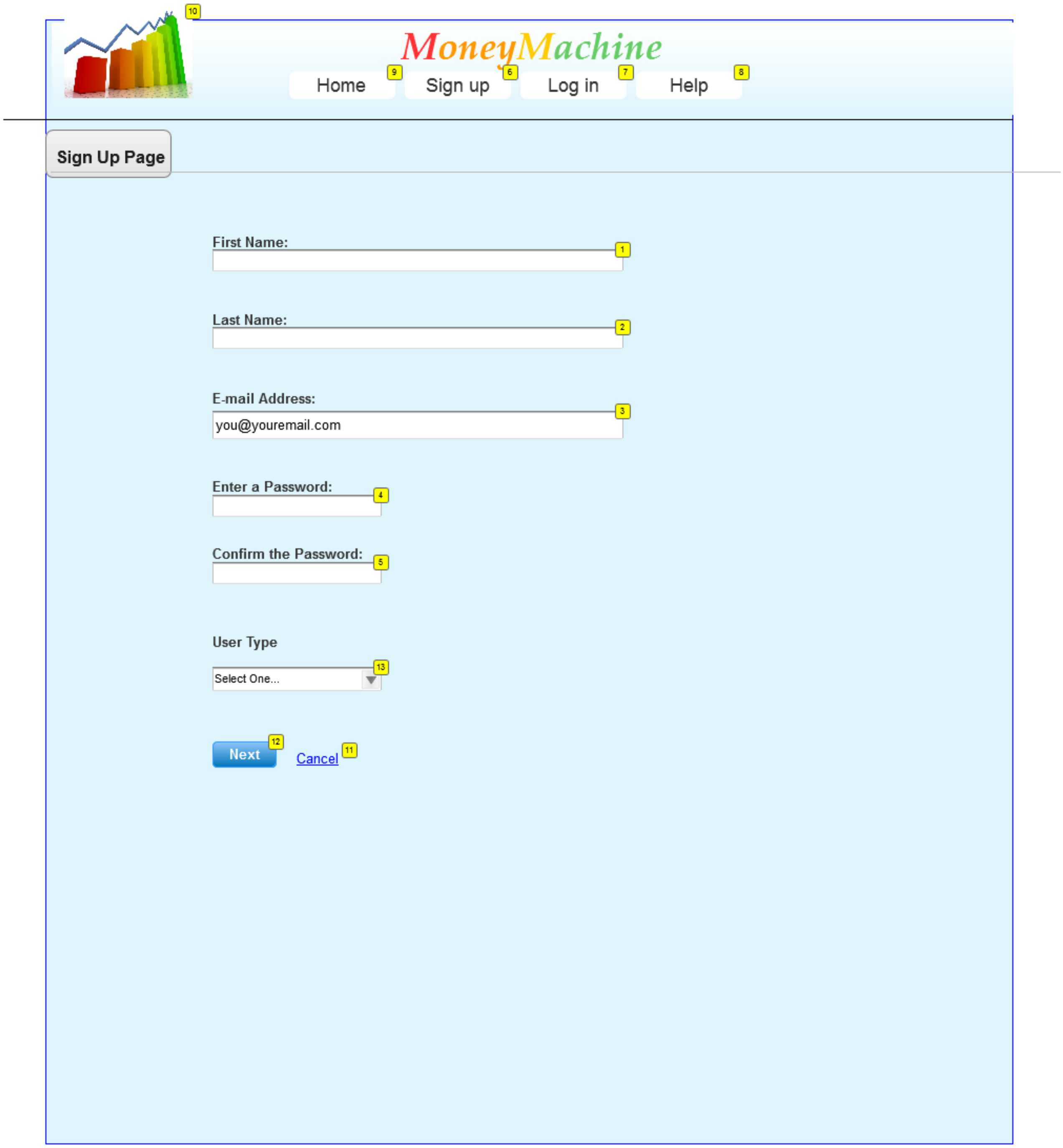
Shanghai	2,432.40	+13.87 (0.57%)
Nikkei 225	11,173.83	-133.45 (-1.18%)
Hang Seng Index	23,401.71	-11.54 (-0.05%)
TSEC	7,906.65	+19.71 (0.25%)
FTSE 100	6,327.36	-31.75 (-0.50%)
EURO STOXX 50	2,635.35	-21.51 (-0.81%)
CAC 40	3,669.60	-28.93 (-0.78%)
S&P TSX	12,721.79	-53.49 (-0.42%)
S&P/ASX 200	5,033.90	-3.00 (-0.06%)
BSE Sensex	19,429.87	-67.31 (-0.35%)
TA100	1,087.23	-4.86 (-0.45%)
KOSPI	1,981.18	+1.57 (0.08%)

4.1.1 Click Table

Footnote	Interactions
1	OnClick: Case 1: Open Sign Up in Current Window
2	OnClick: Case 1: Open Login Page in Current Window
3	OnClick: Case 1: Open Help Page in Current Window
4	OnClick: Case 1: Open Home in Current Window
5	OnClick: Case 1: Open Home in Current Window

4.2 Sign Up

4.2.1 User Interface



4.2.1 Form Table

Footnote	Label	Interactions
1	First Name	OnLostFocus: Case 1 (If length of value of widget Input Field is greater than "128"): Set Error Panels state to Too Long fade in and out Bring Error Panels to Front Case 2 (Else If text on widget Input Field equals ""): Set Error Panels state to Required fade in and out Bring Error Panels to Front Case 3 (Else If text on widget Input Field is not alpha-numeric): Set Error Panels state to Non-Alpha fade in and out Bring Error Panels to Front Case 4 (Else If True): Hide Error Panels
2	Last Name	OnLostFocus: Case 1 (If length of value of widget Input Field is greater than "128"): Set Error Panels state to Too Long fade in and out Bring Error Panels to Front Case 2 (Else If text on widget Input Field equals ""): Set Error Panels state to Required fade in and out Bring Error Panels to Front Case 3 (Else If text on widget Input Field is not alpha-numeric): Set Error Panels state to Non-Alpha fade in and out Bring Error Panels to Front Case 4 (Else If True): Hide Error Panels
3	Email Input Field	OnClick: Case 1: Set text on widget Email Input Field equal to "" OnKeyUp: Case 1: Set text on widget Email Input Field equal to "[[LVAR1.toLowerCase()]]" OnLostFocus: Case 1 (If length of value of widget Email Input Field is greater than "128"): Set Error Panels state to Too Long fade in and out Bring Error Panels to Front Case 2 (Else If text on widget Email Input Field equals ""): Set Error Panels state to Required fade in and out Bring Error Panels to Front Case 3 (Else If text on widget Email Input Field does not contain "@" and text on widget Email Input Field does not equal "."):


```
   Set Error Panels state to Invalid Format - Email fade in and out
   Bring Error Panels to Front
  Case 4
 (Else If True):
   Hide Error Panels
```

4	Password Field	

```
OnKeyUp:
  Case 1
 (If length of value of widget Password Field is greater than "4" and length of value of widget Password Field is less than "6"):
   Set Password Strength Meter Panels state to Poor
  Case 2
 (Else If length of value of widget Password Field is greater than "6" and length of value of widget Password Field is less or equals "8"):
   Set Password Strength Meter Panels state to Fair
  Case 3
 (Else If length of value of widget Password Field is greater than "8" and length of value of widget Password Field is less than "12"):
   Set Password Strength Meter Panels state to Good
  Case 4
 (Else If length of value of widget Password Field is greater than "12" and length of value of widget Password Field is less or equals "20"):
   Set Password Strength Meter Panels state to Very Good



OnLostFocus:
  Case 1
 (If text on widget Password Field equals ""):
   Hide Error Panels - Confirm fade 100ms
   Set Error Panels - Password state to Required fade in and out
   Bring Error Panels - Password to Front
  Case 2
 (Else If length of value of widget Password Field is less than "4"):
   Hide Error Panels - Confirm fade 100ms
   Set Error Panels - Password state to Too Short fade in and out
   Bring Error Panels - Password to Front
  Case 3
 (Else If True):
   Hide Error Panels - Password,
Error Panels - Confirm
```

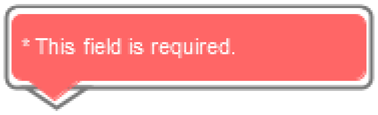
5	Confirm Password Field	

```
OnLostFocus:
  Case 1
 (If text on widget Confirm Password Field equals ""):
   Hide Error Panels - Password fade 100ms
   Set Error Panels - Confirm state to Required fade in and out
  Case 2
 (Else If length of value of widget Confirm Password Field is less than "4"):
   Hide Error Panels - Password
   Set Error Panels - Confirm state to Too Short fade in and out
  Case 3
 (Else If text on widget Confirm Password Field does not equal text on widget Password Field):
   Hide Error Panels - Password
   Set Error Panels - Confirm state to No Match fade in and out
  Case 4
```

		(Else If True): Hide Error Panels - Password, Error Panels - Confirm
6	Sign Up	OnClick: Case 1: Open Sign Up in Current Window
7	Login	OnClick: Case 1: Open Login Page in Current Window
8	Help	OnClick: Case 1: Open Help Page in Current Window
9	Home	OnClick: Case 1: Open Home in Current Window
10		OnClick: Case 1: Open Home in Current Window
11	Cancel Link	OnClick: Case 1: Open Home in Current Window
12	Next Button	OnClick: Case 1: Open Tutorial Page in Current Window
13	Dropdown Menu	OnChange: Case 1 (If selected option of Dropdown Menu equals "Select One..."): Set Error Panels state to Required fade in and out Bring Error Panels to Front Includes User type such as: -Player -Visitor -Advertiser OnLostFocus: Case 1 (If selected option of Dropdown Menu equals "Select One..."): Set Error Panels state to Required fade in and out Bring Error Panels to Front Case 2 (Else If True): Hide Error Panels fade 100ms

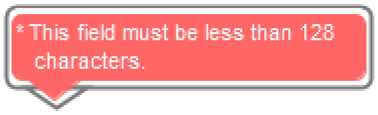
4.2.2 Error Panels

Required:



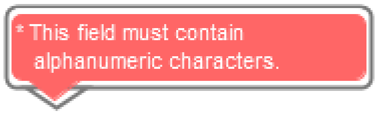
* This field is required.

Entered Text Too Long:



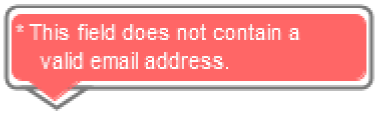
* This field must be less than 128 characters.

Non-Alpha:



* This field must contain alphanumeric characters.

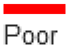
Invalid E-Mail:



* This field does not contain a valid email address.

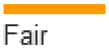
4.2.3 Password Strength Meter Panels

Poor:



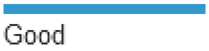
Poor

Fair:



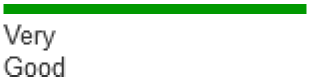
Fair

Good:



Good

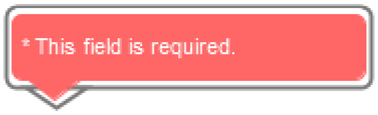
Very Good:



Very Good

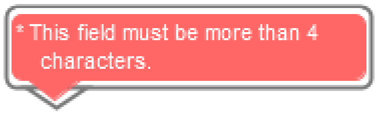
4.2.4 Error Panels - Password

Required:



* This field is required.

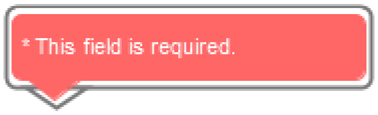
Too Short:



* This field must be more than 4 characters.

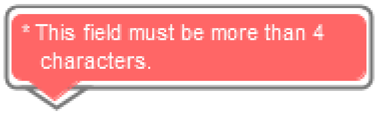
4.2.5 Error Panels - Confirm

Required:



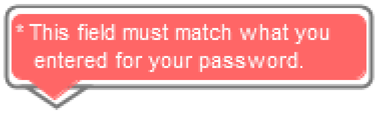
* This field is required.

Too Short:



* This field must be more than 4 characters.

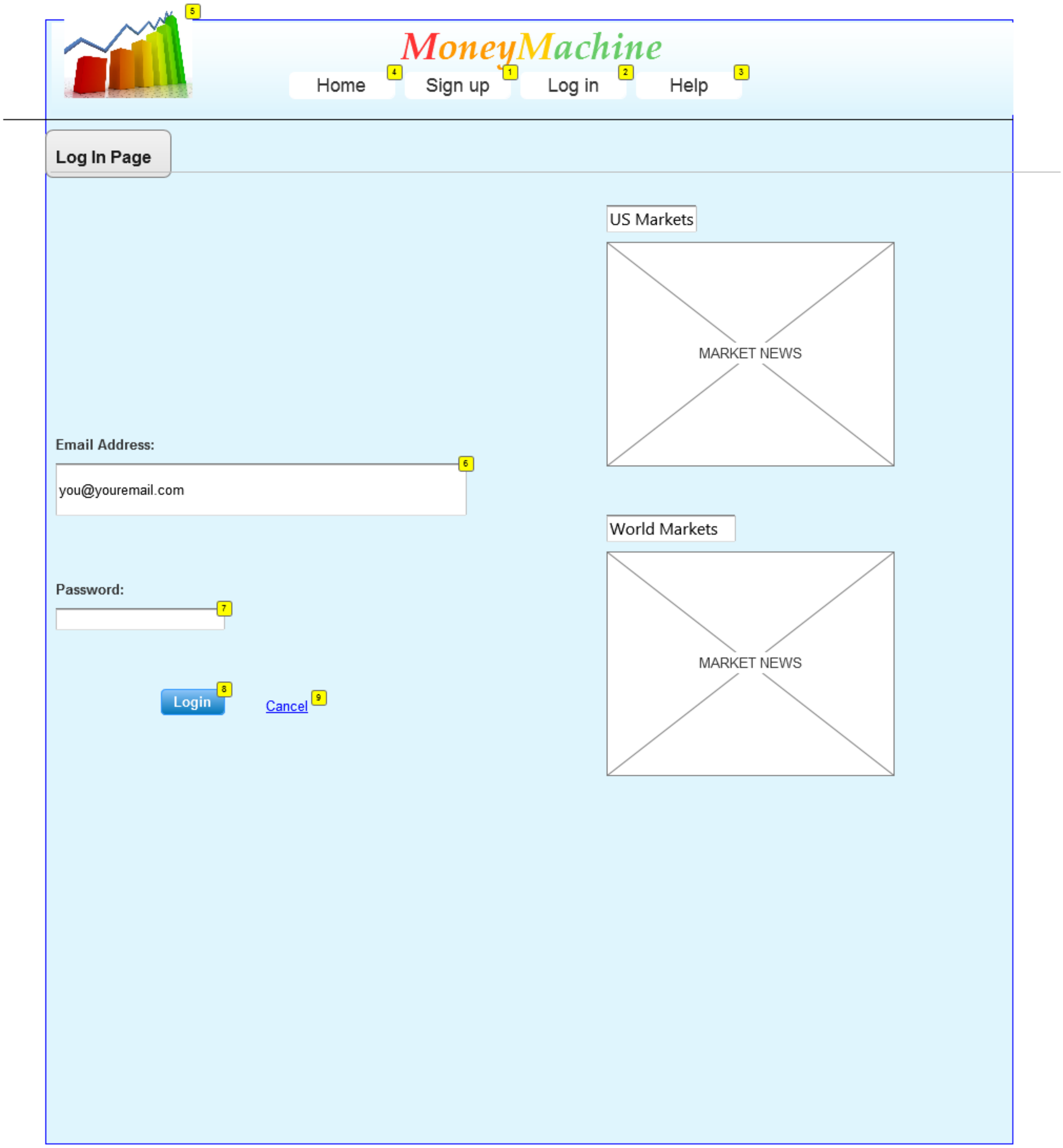
No Match:



* This field must match what you entered for your password.

4.3 Login Page

4.3.1 User Interface



Email Address: you@youremail.com

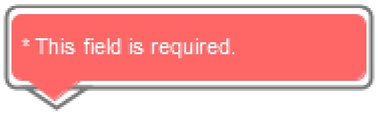
4.3.2 Form Table

Footnote	Label	Interactions
1	Sign Up	OnClick: Case 1: Open Sign Up in Current Window
2	Login	OnClick: Case 1: Open Login Page in Current Window
3	Help	OnClick: Case 1: Open Help Page in Current Window
4	Home	OnClick: Case 1: Open Home in Current Window
5	Logo	OnClick: Case 1: Open Home in Current Window
6	Email Input Field	OnClick: Case 1: Set text on widget Email Input Field equal to "" OnKeyUp: Case 1: Set text on widget Email Input Field equal to "[[LVAR1.toLowerCase()]]" OnLostFocus: Case 1 (If length of value of widget Email Input Field is greater than "128"): Set Error Panels state to Too Long fade in and out Bring Error Panels to Front Case 2 (Else If text on widget Email Input Field equals ""): Set Error Panels state to Required fade in and out Bring Error Panels to Front Case 3 (Else If text on widget Email Input Field does not contain "@" and text on widget Email Input Field does not equal "."): Set Error Panels state to Invalid Format - Email fade in and out Bring Error Panels to Front Case 4 (Else If True): Hide Error Panels fade 100ms
7	Password Field	OnKeyUp: Case 1 (If length of value of widget Password Field is greater than "4" and length of value of widget Password Field is less than "6"): Set Panel state to State Case 2 (Else If length of value of widget Password Field is greater than "6" and length of value of

		widget Password Field is less or equals "8"): Set Panel state to State Case 3 (Else If length of value of widget Password Field is greater than "8" and length of value of widget Password Field is less than "12"): Set Panel state to State Case 4 (Else If length of value of widget Password Field is greater than "12" and length of value of widget Password Field is less or equals "20"): Set Panel state to State OnLostFocus: Case 1 (If text on widget Password Field equals ""): Hide Panel Set Error Panels - Password state to Required fade in and out Bring Error Panels - Password to Front Case 2 (Else If length of value of widget Password Field is less than "4"): Hide Panel Set Error Panels - Password state to Too Short fade in and out Bring Error Panels - Password to Front Case 3 (Else If True): Hide Error Panels - Password
8	Login Button	OnClick: Case 1: Open After Logon Page in Current Window
9	Cancel Link	OnClick: Case 1: Open Home in Current Window

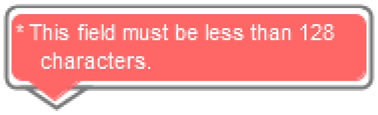
4.3.3 Error Panels

Required:



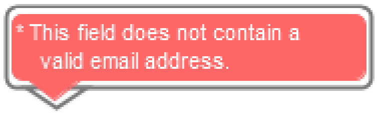
* This field is required.

Too Long:



* This field must be less than 128 characters.

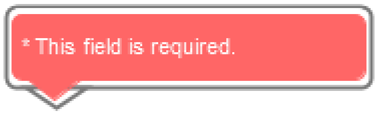
Invalid Format – Email:



* This field does not contain a valid email address.

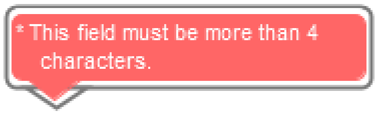
4.3.4 Error Panels - Password

Required



* This field is required.

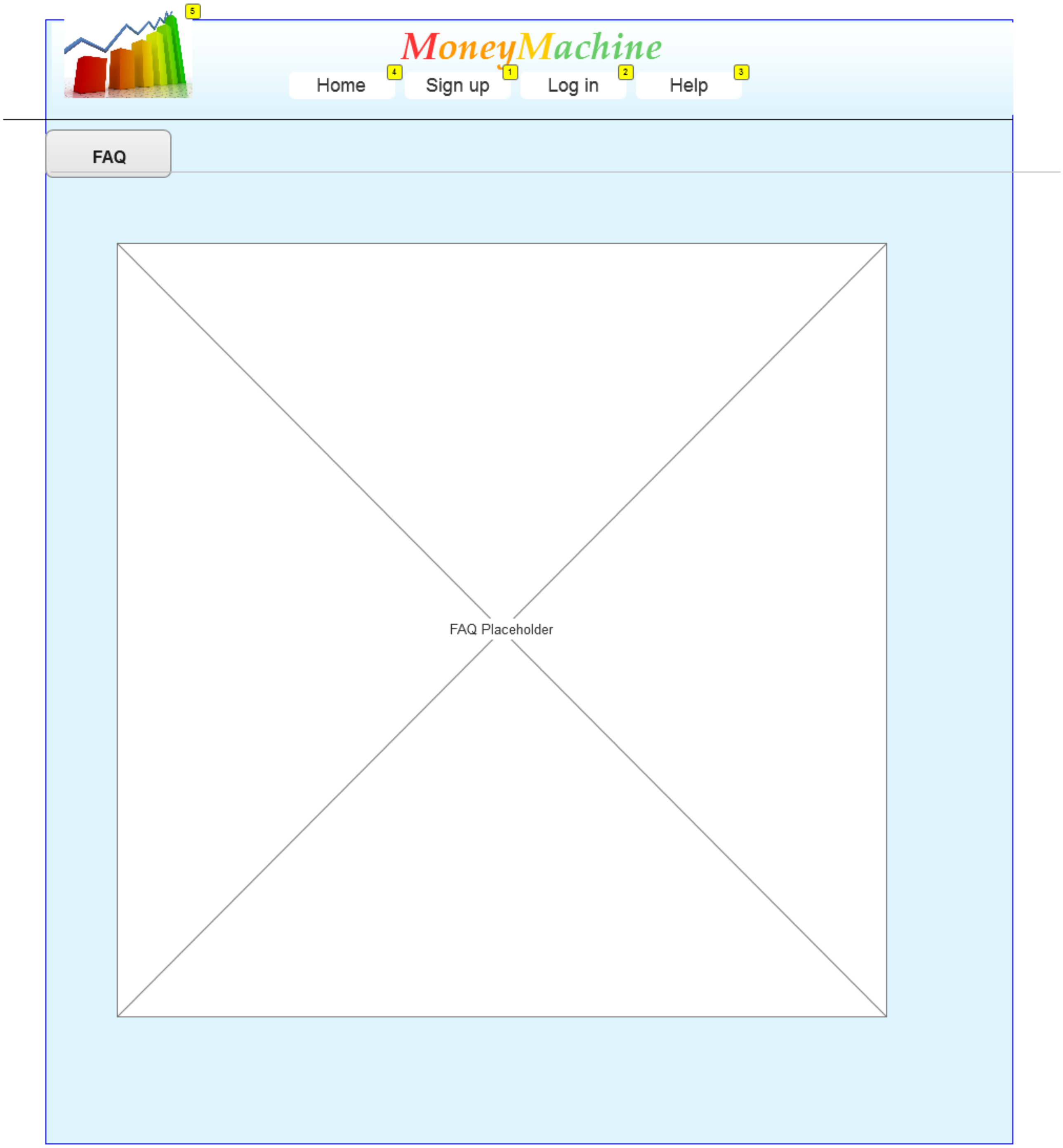
Too Short



* This field must be more than 4 characters.

4.4 Help Page

4.4.1 User Interface



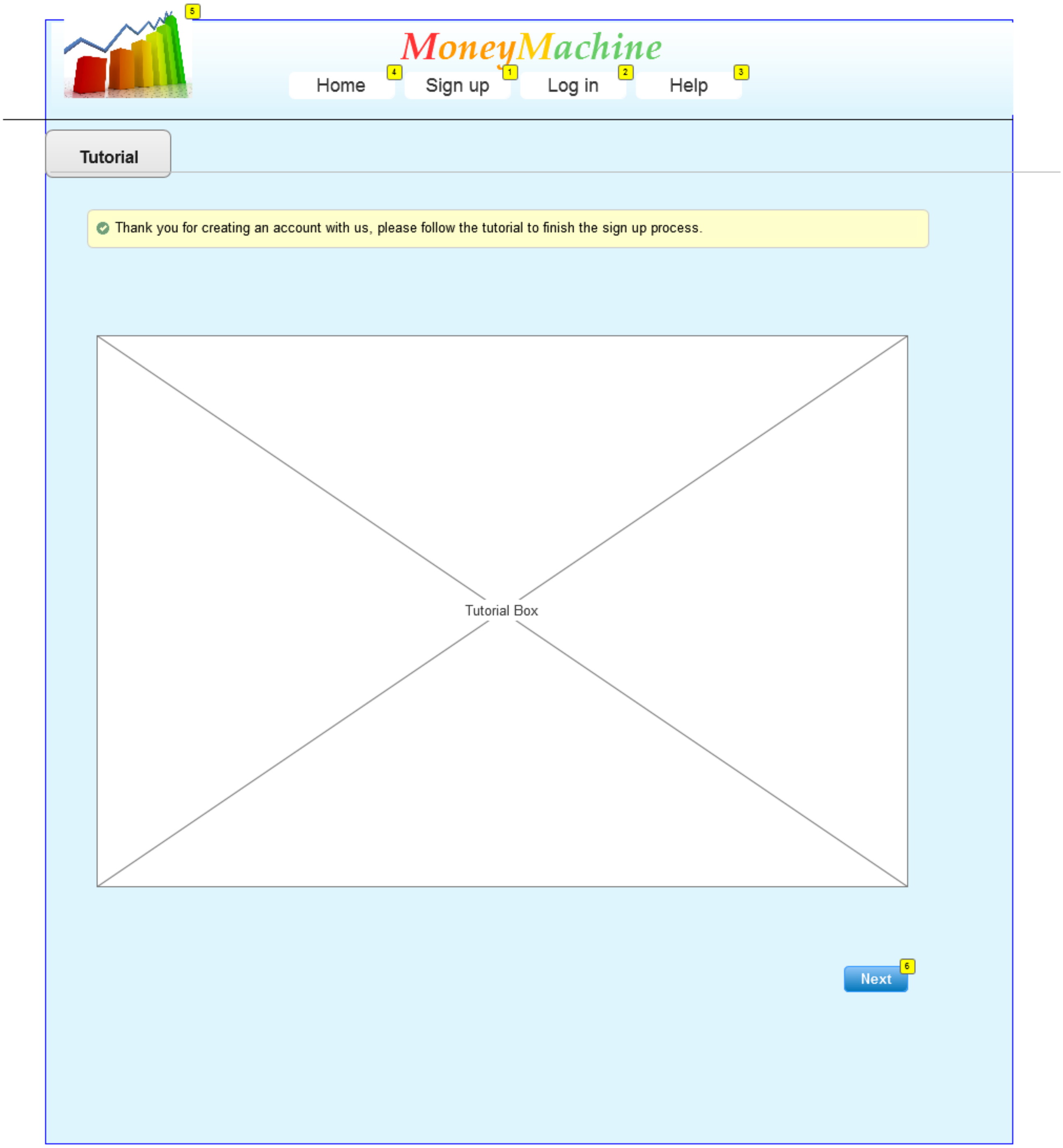
4.4.2 Click Table

Footnote	Interactions

1	OnClick: Case 1: Open Sign Up in Current Window
2	OnClick: Case 1: Open Login Page in Current Window
3	OnClick: Case 1: Open Help Page in Current Window
4	OnClick: Case 1: Open Home in Current Window
5	OnClick: Case 1: Open Home in Current Window

4.5 Tutorial Page

4.5.1 User Interface



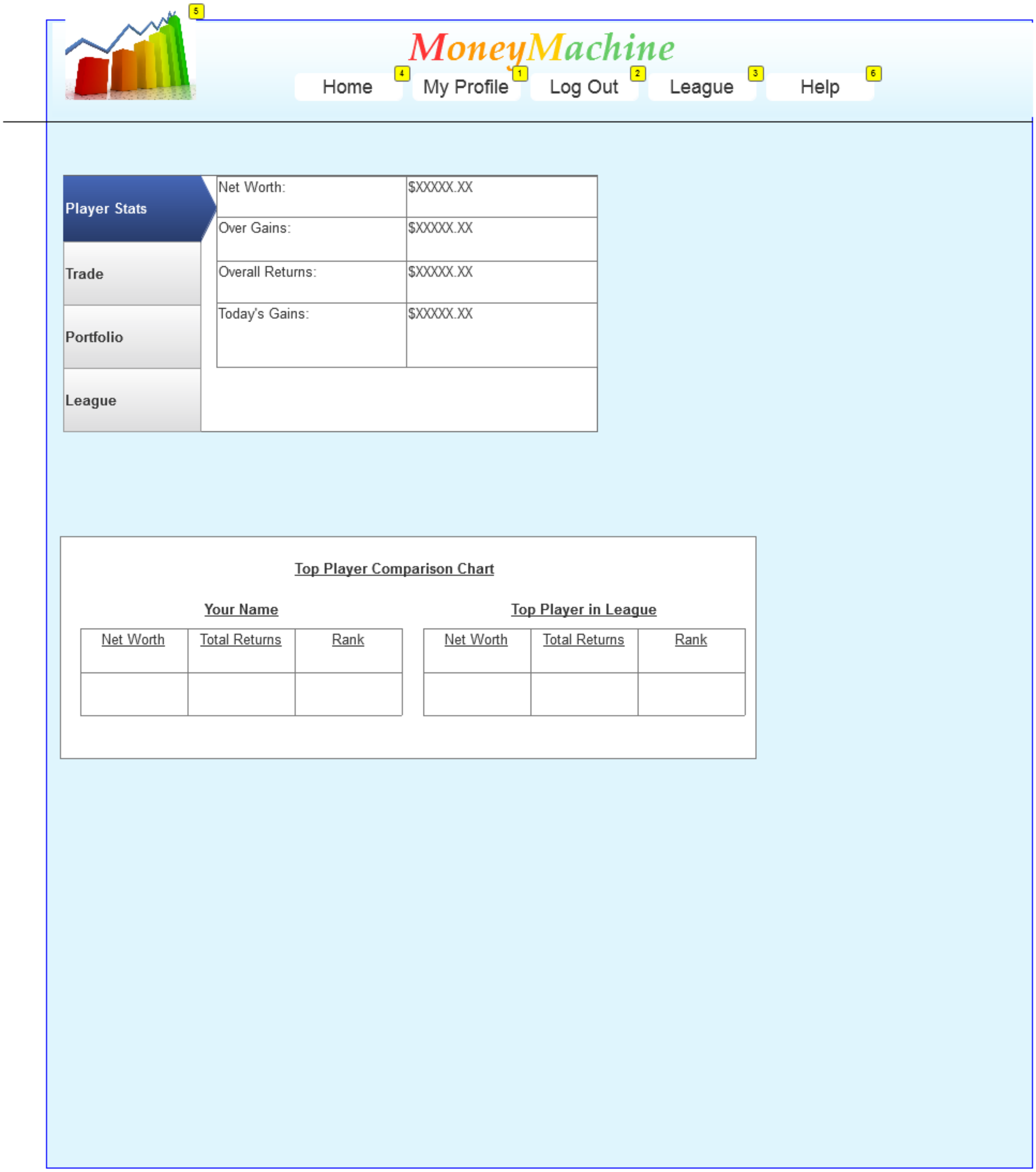
Thank you for creating an account with us, please follow the tutorial to finish the sign up process.

4.5.2 Click Table

Footnote	Label	Interactions
1	Sign Up	OnClick: Case 1: Open Sign Up in Current Window
2	Login	OnClick: Case 1: Open Login Page in Current Window
3	Help	OnClick: Case 1: Open Help Page in Current Window
4	Home	OnClick: Case 1: Open Home in Current Window
5	Logo	OnClick: Case 1: Open Home in Current Window
6	Next Button	OnClick: Case 1: Open After Logon Page in Current Window

4.6 After Logon Page

4.6.1 User Interface



Net Worth:	$XXXXX.XX
Over Gains:	$XXXXX.XX
Overall Returns:	$XXXXX.XX
Today's Gains:	$XXXXX.XX

Top Player Comparison Chart

Your Name

Net Worth	Total Returns	Rank

Top Player in League

Net Worth	Total Returns	Rank

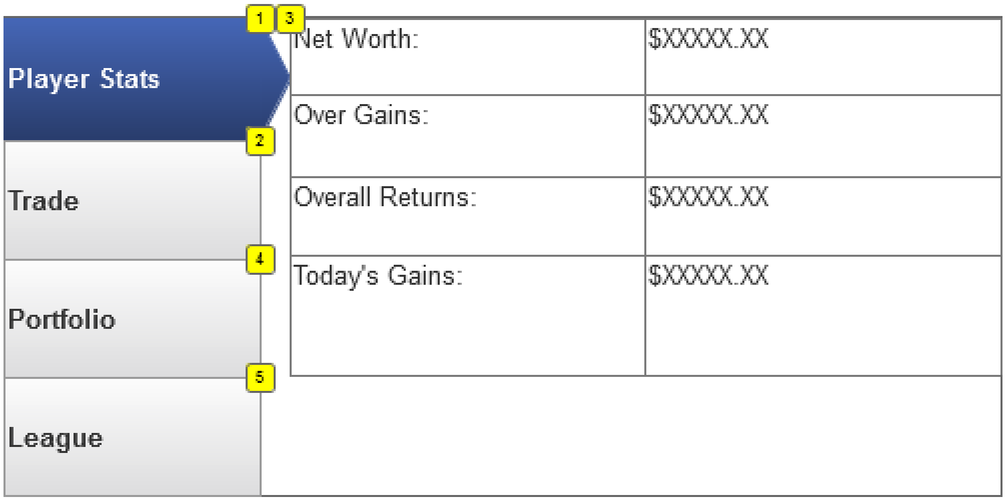
4.6.2 Click Table

Footnote	Label	Interactions
1	My Profile	OnClick: Case 1: Open After Logon Page in Current Window
2	Logout	OnClick: Case 1: Open Login Page in Current Window
3	League	OnClick: Case 1: Open League Page in Current Window
4	Home	OnClick: Case 1: Open Home in Current Window
5	Logo	OnClick: Case 1: Open Home in Current Window
6	Help	OnClick: Case 1: Open Help Page in Current Window

4.6.3 Tab Panel

4.6.3.1 Player Stats Tab

4.6.3.2 User Interface



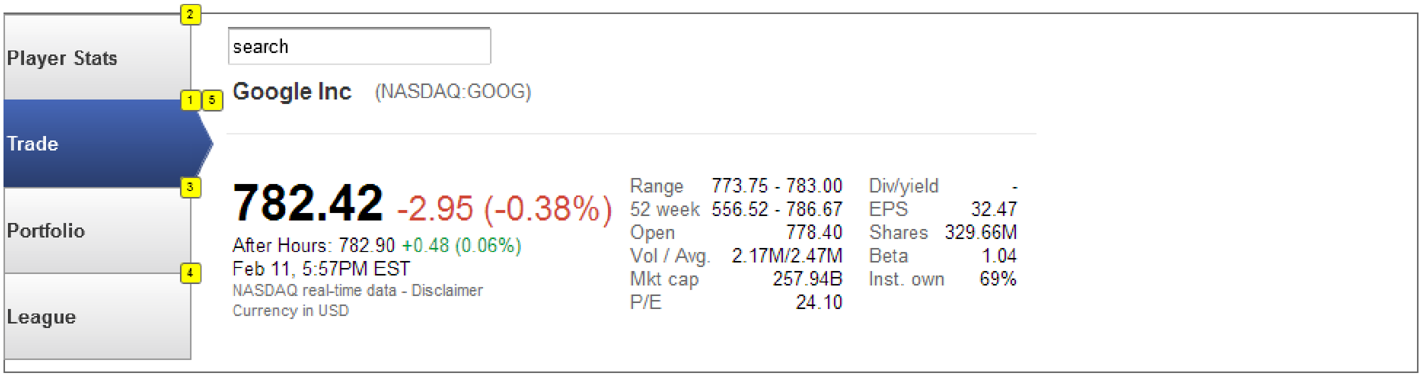
Net Worth:	$XXXXX.XX
Over Gains:	$XXXXX.XX
Overall Returns:	$XXXXX.XX
Today's Gains:	$XXXXX.XX

4.6.3.3 Click Table

Footnote	Interactions
1[1]	OnClick: Case 1: Set Tab Panel state to Player Stats
2	OnClick: Case 1: Set Tab Panel state to Trade
3	OnClick: Case 1: Set Panel state to Player Stats
4	OnClick: Case 1: Set Tab Panel state to Portfolio
5	OnClick: Case 1: Set Tab Panel state to League

4.6.3.4 Trade Tab

4.6.3.5 User Interface



Google Inc (NASDAQ:GOOG)

782.42 -2.95 (-0.38%)
After Hours: 782.90 +0.48 (0.06%)
Feb 11, 5:57PM EST
NASDAQ real-time data - Disclaimer
Currency in USD

Range	773.75 - 783.00	Div/yield	-
52 week	556.52 - 786.67	EPS	32.47
Open	778.40	Shares	329.66M
Vol / Avg.	2.17M/2.47M	Beta	1.04
Mkt cap	257.94B	Inst. own	69%
P/E	24.10		

4.6.3.6 Click Table

Footnote	Interactions
1	OnClick: Case 1: Set Tab Panel state to Trade
2	OnClick: Case 1: Set Tab Panel state to Player Stats

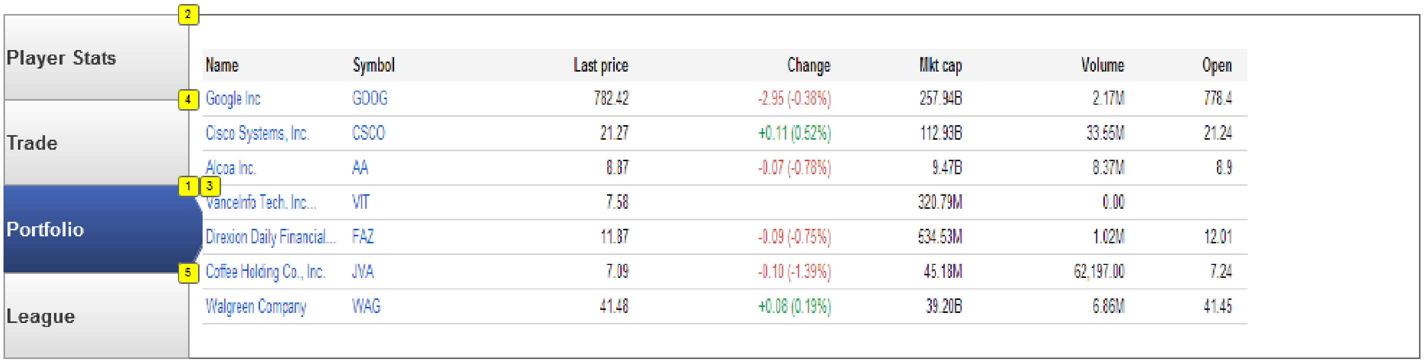
---

[1] The reason there are 2 footnotes (footnote 1 and 3 in this case) that are the same is to represent the 2 possibilities when the tab will be accessed the first is when that tab is initially loaded and the second time is when the tab is clicked on again after going to a different tab.

3	OnClick: Case 1: Set Tab Panel state to Portfolio
4	OnClick: Case 1: Set Tab Panel state to League
5	OnClick: Case 1: Set Panel state to State

4.6.3.7 Portfolio Tab

4.6.3.8 User Interface



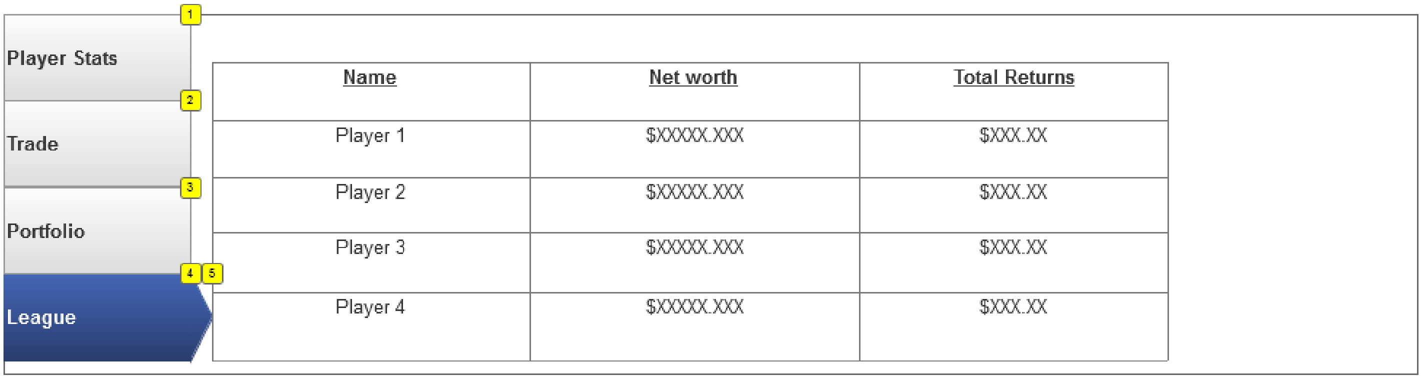
Name	Symbol	Last price	Change	Mkt cap	Volume	Open
Google Inc	GOOG	782.42	-2.95 (-0.38%)	257.94B	2.17M	778.4
Cisco Systems, Inc.	CSCO	21.27	+0.11 (0.52%)	112.93B	33.55M	21.24
Alcoa Inc.	AA	8.87	-0.07 (-0.78%)	9.47B	8.37M	8.9
VanceInfo Tech. Inc...	VIT	7.58		320.79M	0.00	
Direxion Daily Financial...	FAZ	11.87	-0.09 (-0.75%)	534.53M	1.02M	12.01
Coffee Holding Co., Inc.	JVA	7.09	-0.10 (-1.39%)	45.18M	62,197.00	7.24
Walgreen Company	WAG	41.48	+0.08 (0.19%)	39.20B	6.86M	41.45

4.6.3.9 Click Table

Footnote	Interactions
1	OnClick: Case 1: Set Tab Panel state to Portfolio
2	OnClick: Case 1: Set Tab Panel state to Player Stats
3	OnClick: Case 1: Set Panel state to Portfolio
4	OnClick: Case 1: Set Tab Panel state to Trade
5	OnClick: Case 1: Set Tab Panel state to League

4.6.3.10 League Tab

4.6.3.11 User Interface



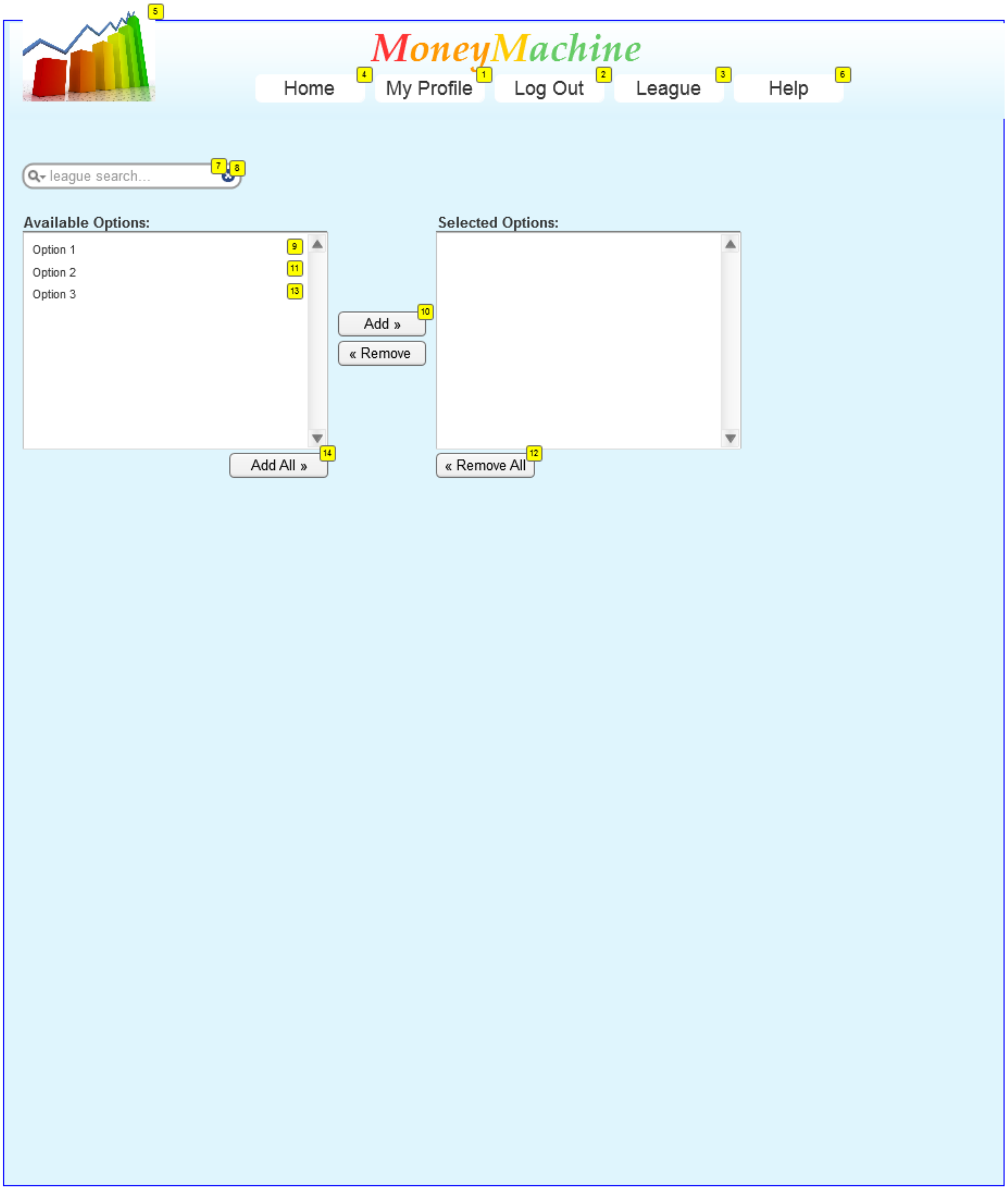
Name	Net worth	Total Returns
Player 1	$XXXXX.XXX	$XXX.XX
Player 2	$XXXXX.XXX	$XXX.XX
Player 3	$XXXXX.XXX	$XXX.XX
Player 4	$XXXXX.XXX	$XXX.XX

4.6.3.12 Click Table

Footnote	Interactions
1	OnClick: Case 1: Set Tab Panel state to Player Stats
2	OnClick: Case 1: Set Tab Panel state to Trade
3	OnClick: Case 1: Set Tab Panel state to Portfolio
4	OnClick: Case 1: Set Tab Panel state to League
5	OnClick: Case 1: Set Panel state to League

4.7 League Page

4.7.1 User Interface



4.7.2 Click Table

Footnote	Label	Interactions
1	My Profile	OnClick: Case 1: Open After Logon Page in Current Window
2	Log out	OnClick: Case 1: Open Login Page in Current Window
3	League	OnClick: Case 1: Open League Page in Current Window
4	Help	OnClick: Case 1: Open Home in Current Window
5	Logo	OnClick: Case 1: Open Home in Current Window
6	Help	OnClick: Case 1: Open Help Page in Current Window
7	Search Field	OnClick: Case 1: Set text on widget Search Field equal to "" OnLostFocus: Case 1: Set text on widget Search Field equal to "Search..."
8	Search Button	OnClick: Case 1: Set text on widget Search Field equal to ""
9	Option 1	OnClick: Case 1: Set Option 1 to Selected, Set value of variable SelectedItem equal to "Option 1"
10	Add Button	OnClick: Case 1 (If value of variable SelectedItem equals "Option 1"): Set text on widget Destination 1 equal to value of variable SelectedItem Case 2 (Else If value of variable SelectedItem equals "Option 2"): Set text on widget Destination 2 equal to value of variable SelectedItem Case 3 (Else If value of variable SelectedItem equals "Option 3"): Set text on widget Destination 3 equal to value of variable SelectedItem
11	Option 2	OnClick: Case 1: Option 2 to Selected, Set value of variable SelectedItem equal to "Option 2"

12	Remove All Button	OnClick: Case 1 (If text on widget Destination 1 does not equal "" or text on widget Destination 2 does not equal "" or text on widget Destination 3 does not equal ""): Set text on widget Destination 1 equal to "", and text on widget Destination 2 equal to "", and text on widget Destination 3 equal to "", and value of variable SelectedItem equal to ""
13	Option 3	OnClick: Case 1: Option 3 to Selected Set value of variable SelectedItem equal to "Option 3"
14	Add All Button	OnClick: Case 1 (If value of variable SelectedItem does not equal ""): Set text on widget Destination 1 equal to "Option 1", and text on widget Destination 2 equal to "Option 2", and text on widget Destination 3 equal to "Option 3"

5.0 Domain Analysis

5.1 Domain Model

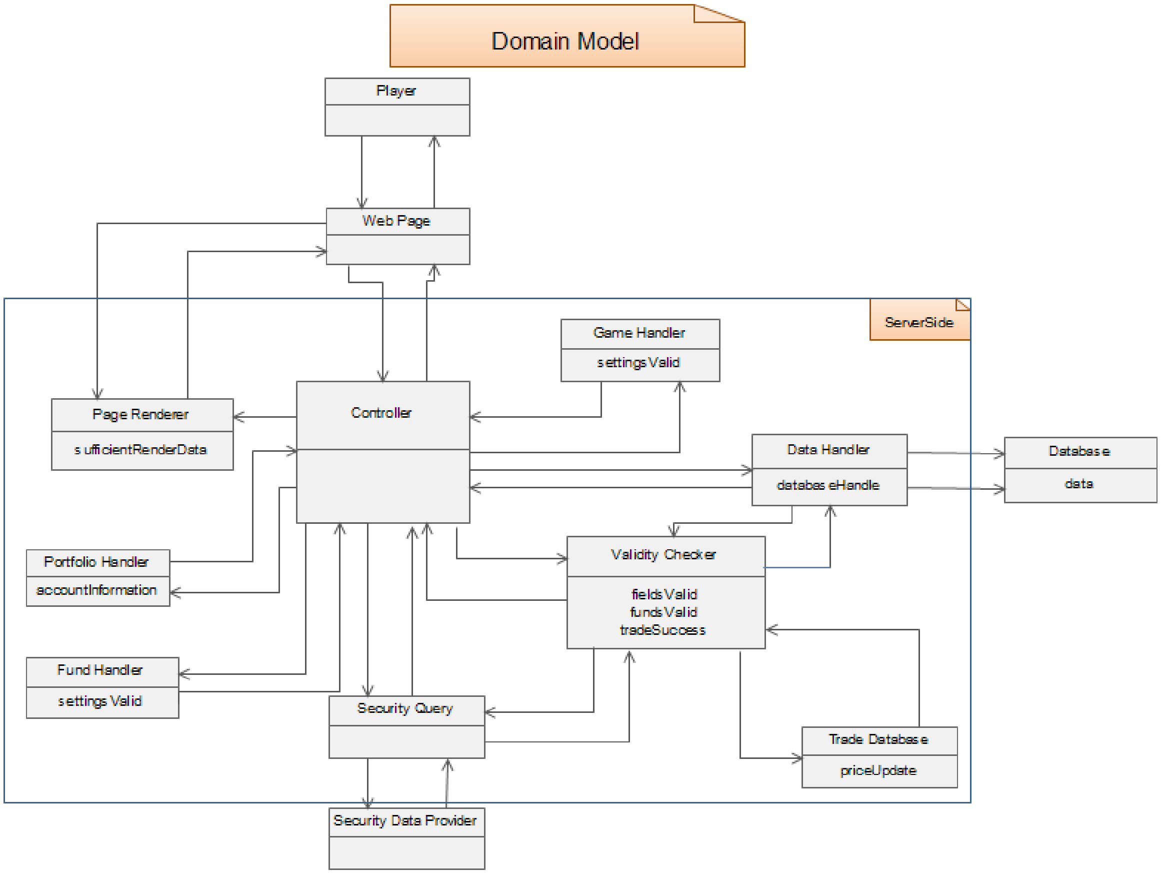


Figure 1: Domain Model

Figure 1, shows Money Machine's new, updated Domain Model. The subsequent diagrams give insight into how the concepts work to satisfy the key use cases of our updated website. The old domain model contained a Facebook concept, which is not in this Domain Model. The domain model is similar to the old domain model, but there are some changes to how the concepts will interact. In addition, the web server, web browser, and web framework is replaced with just web page. There are new concepts names added which are related to the new use case names.

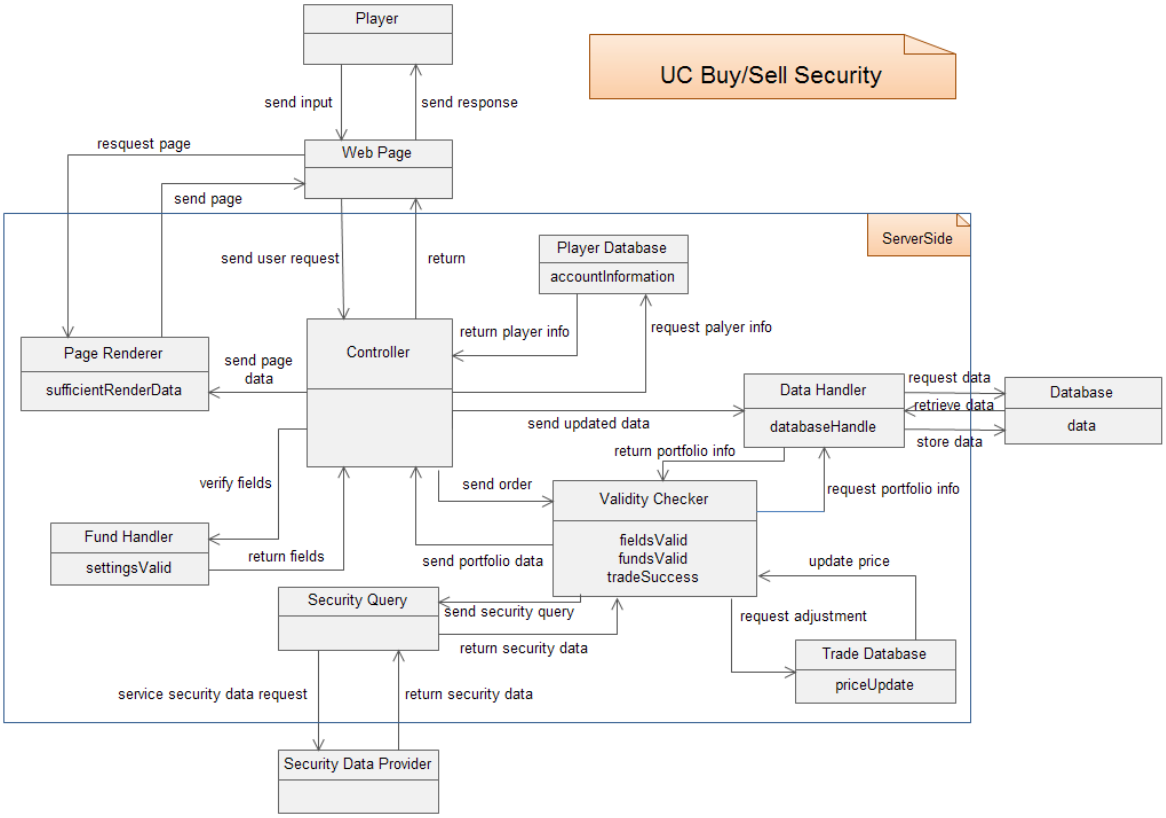


Figure 2: Buy/Sell Security

Figure 2, represent both our buy (UC-3) and sell (UC-4) use cases since they behave in the same way. The User fills out order information on the web page, and sends to request to order to the Controller. The controller relays the order to the Validity Checker so that it can send the corresponding security query to the Security Query concept, which fetches the necessary information from the remote Security Info Provider. The Validity Checker then sends a request to the Trade Database to adjust the stock price based on our current trade. Now the Validity Checker must retrieve the User's balance in order to verify the transaction is valid; it requests for the Data Handler to get this information from the Database. If the transaction is successful, the Controller tells the Data Handler to update the User's portfolio. Then the Controller will let the Page Renderer know what page to generate and pass necessary data. The Web Page is informed of the completion of the order and knows to request the page to be viewed from the Page Renderer.

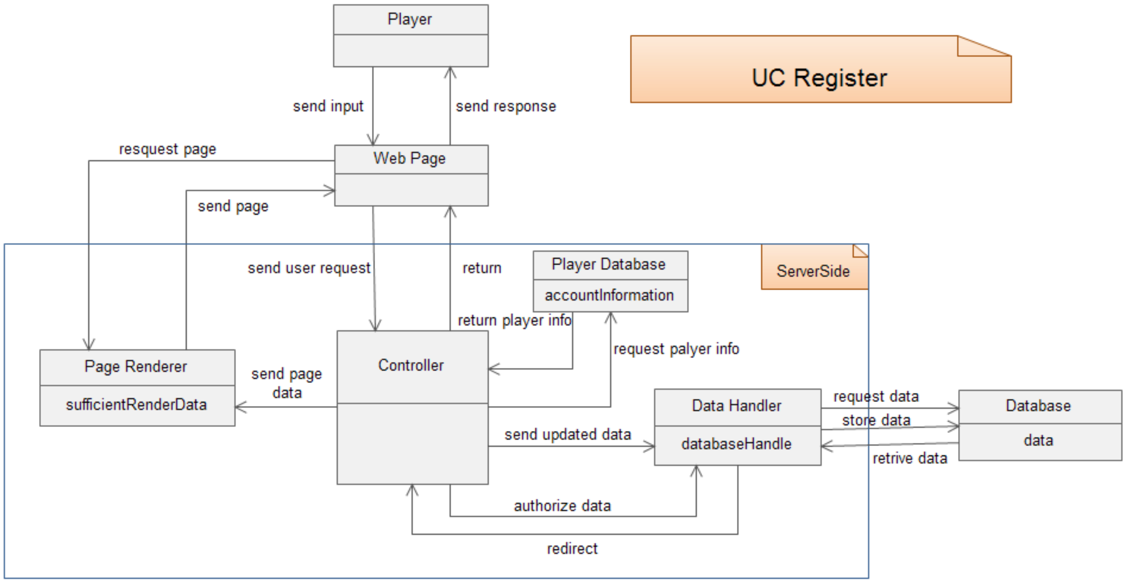


Figure 3: Register

The UC-6 Register is represented in Figure 3. First, the User tries to access the Player Portfolio, but he is not registered, so the web page tells the Controller to render the new registration page. The Controller will then send instructions to the Data Handler to a new account in our Database. The Controller also notifies to create a new Player Portfolio. The Controller then passes necessary data to the Page Renderer and informs Web Page that the process is complete. The Web Page will call for the Page Renderer to generate Player Portfolio page to be viewed.

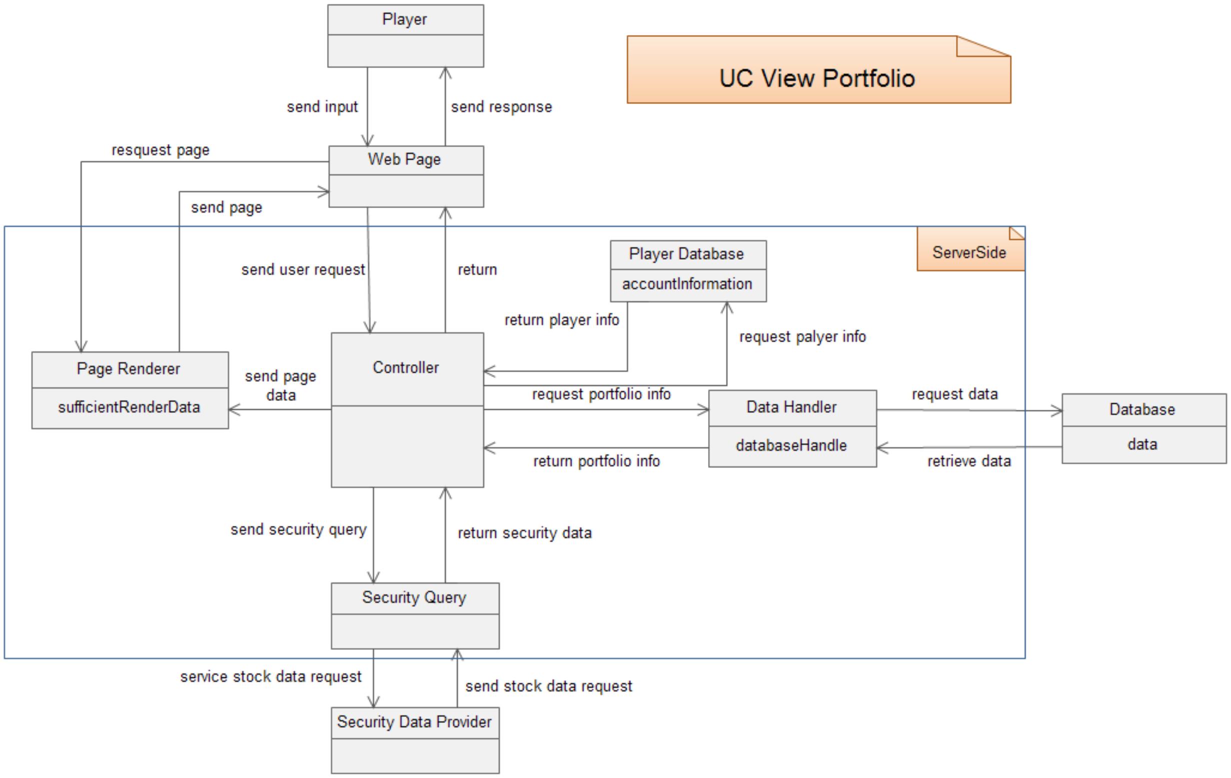


Figure 4: View Portfolio

Figure 4, shows the UC-5, View Portfolio. The Player queries about the portfolio from the Web Page, and this request gets sent to the Controller. To get the necessary data, the Controller will send a request for the portfolio info to the Data Handler, which obtains this data from the Database. The Controller will then query Security Query for each security held by the Player, which will obtain the necessary information from Security Info Provider. The portfolio is now ready to be viewed, so Controller gives the Page Renderer all necessary data and then lets the Web Page know the process has been finished. The Web Page requests the Page Renderer to create the required page to be viewed.

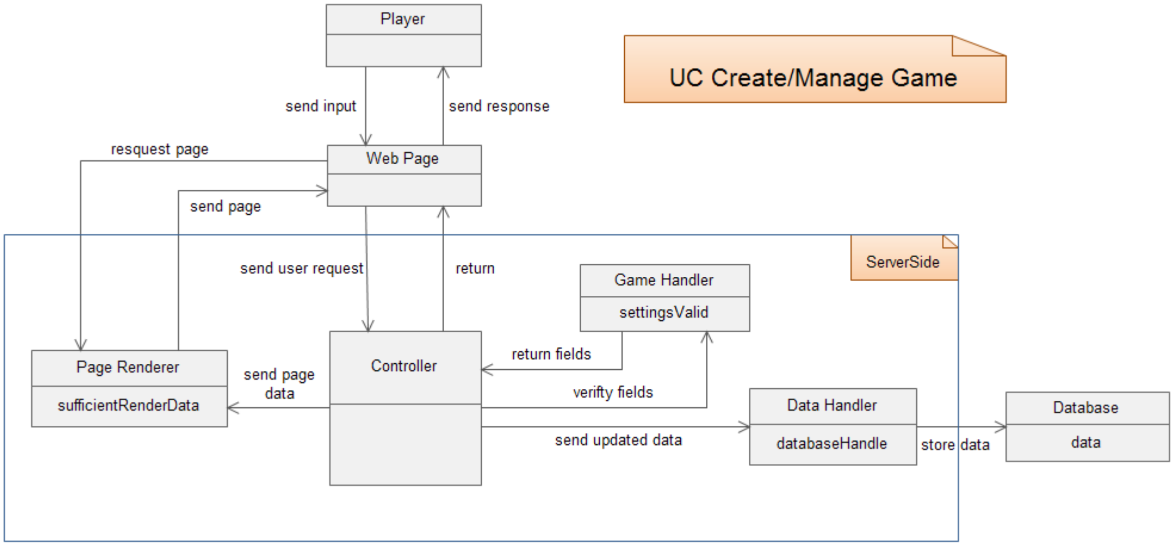


Figure 5: Create/Manage Game

Figure 5, represents the UC-7 and UC-10, the creation and management of Investment Games. The User fills in the necessary fields in order to create or change a Investment Game then the Web Page submits this info. The Controller will receive the request and call on the Game Handler to verify the validity of the fields. If there are no errors, the Controller will inform the Data Handler to store the new game or its new settings. Then (regardless of the validity of the fields), the Controller provides the necessary page data to the Page Renderer and informs the Web Page of the completion of the process. The Web Page calls for the Page Render to create the necessary page to be viewed.

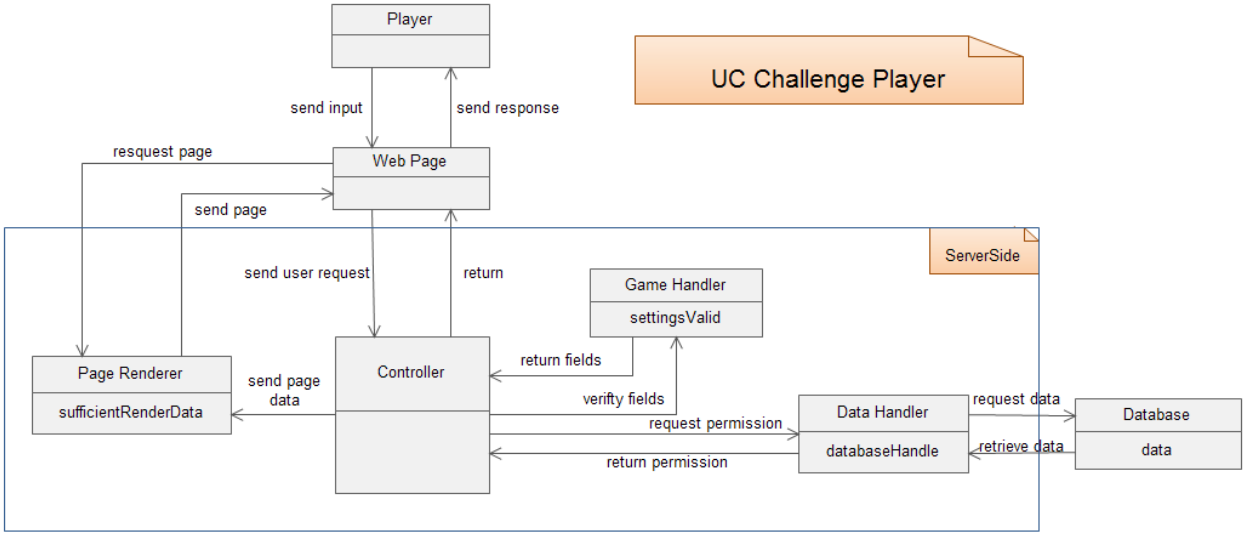


Figure 6: Challenge Player

Figure 6, represents the domain model for US-15, Challenge Player. A player should be able to challenge other player(s). The player sends a request to challenge other player by requesting in the Web Page. The Web Page sends a request to the Controller, the Controller then sends a request to Data Handler to verify that the player is in the database. Then the controller Data Handler sends a notification to Player 2 to accept the challenge. The player's database is updated with the new information. The Controller now sends the data to be rendered to the Page Renderer and then notifies the Web Page that the process is complete. The Web Page knows to request the page from the Page Renderer, which then services the request and generates the correct page to be viewed by the User.

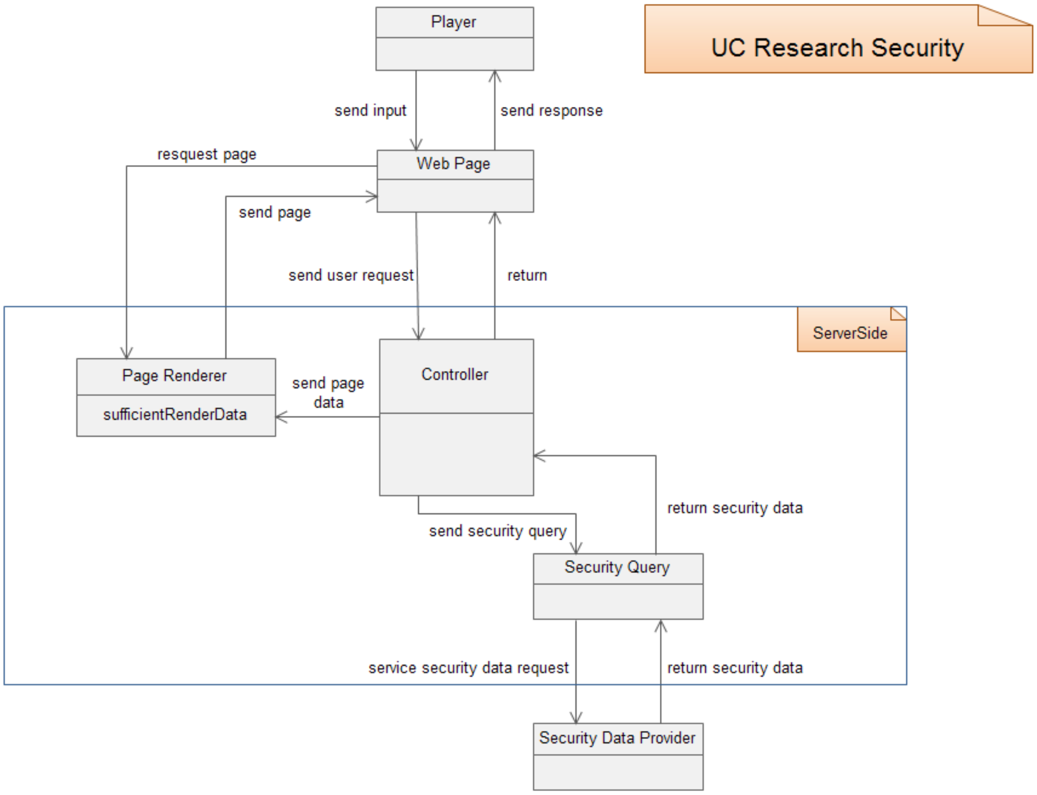


Figure 7: Research Security

Figure 7 shows the UC-3 Research Security. The security is requested through the Web Page by the User, which tells Controller to inform the Security Query to fetch the correct security data from Security Info Provider. Note that even an invalid ticker symbol will go through the same steps, the Security Query will just return N/A or 0 for all the fields. The Controller now sends the data to be rendered to the Page Renderer and then notifies the Web Page that the process is complete. The Web Page knows to request the page from the Page Renderer, which then services the request and generates the correct page to be viewed by the User.

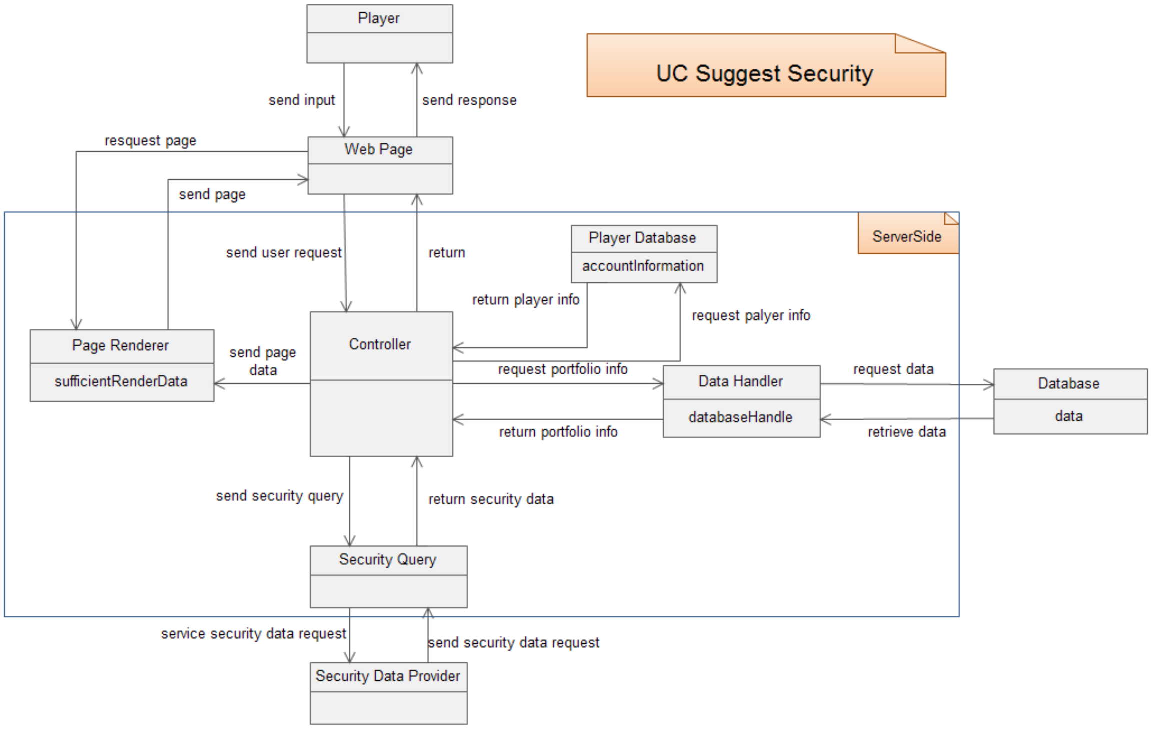


Figure 8: Suggest Security

Figure 8 shows the new use case UC-14, Suggest Security. The Player sends a risk appetite and submits it to through the Web Page to the Controller. The Controller sends a request to Data Handler to read the Player's Portfolio. The Data Handler retrieves the Player's Portfolio. The Player's Portfolio is analyzed and the controller sends a request to Security Query to query for the security returned by the Data Handler. The Security Data Provider, returns the requested security. The Controller, then renders the page to be rendered. The Page Renderer, sends the page to be viewed by the Player with the requested information.

5.1.1 Concept Definitions

"D" - *Doing* responsibilities.

"K" - *Knowing* responsibilities.

Player:
Definition: A person who uses or operates something
Responsibilities:

- Research stocks (D)
- Make requests for trades (D)
- Manage portfolio (K)
- Navigate through website (D)
- Manage Leagues (K)
- Manage Funds (K)
- Watch Learning tutorial if necessary (D)

Web Page:
Definition: A document connected to the World Wide Web and viewable by anyone connected to the internet who has a web browser.
Responsibilities:

- Take requests from the Player (K)
- Send requests to the Controller (D)
- Send page requests to the Page Renderer (D)
- Update new webpage to be displayed when new page is rendered (K)

Page Renderer:
Definition: Page rendering is the process of generating a page from the database
Responsibilities:

- Receive the required data to generate new page (K)
- Convert the data into user-friendly format (D)
- Send rendered pages to the Web Page (D)

Controller:
Definition: Takes user request and creates a web page that is user-friendly.
Responsibilities:

- Request account creation (D)
- Receive Player requests from the Web Page (D)
- Request an order (K)
- Request stock queries (K)
- Send League and Fund settings to be validated (K)
- Inform Web Page when process is complete (D)
- Send page data to be rendered (D)

Stock Query:
Definition: Fetch Real time stock prices.
Responsibilities:

- Receive requests from the Player for stock data (K)
- Request information from Stock Info Provider (D)
- Send updated stock data to Player (D)

Validity Checker:
Definition: Routines in a data entry program that test the input is correct or not.
Responsibilities:

- Determine if sufficient funds are available for the transaction (K)
- Request updated stock price based on liquidity model (D)
- Request and receive portfolio data (K)
- Send queries for stock data (D)

Liquidity Manager:
Definition: Manipulates the price to realistic real world price for slippage
Responsibilities:

- Determine new price (K)
- Send out updated stock information (D)
- Utilize algorithm to reflect realistic trades in the market (K)

Data Handler:
Definition: Communicates with database to service data requests
Responsibilities:

- Receive and send every kind of data used in system (D)
- Request data from Database (D)
- Send data to be stored in Database (K)

League Handler:
Definition: A Player who is allowed to create as well as participate in the Leagues.
Responsibilities:

- Receive initial or existing league requests (D)
- Determine if the requests are valid (K)
- Upon successful completion of Player's request, update database (K)
- Create a new league or let the Player participate in the other League (D)

Fund Handler:
Definition: A Player who handles his resources
Responsibilities:

- Receive requests for available Funds (D)
- Determine if requests are valid (K)
- Upon successful completion of Player's request, update the database (D)

Advertiser:
Definition: someone who is interested in advertising their business
Responsibilities:

- View their advertisement bill (D)
- Pay the bill to the Administrator (K)
- Upload a banner to the webpage (D)

5.1.2 Association Definitions

Concept Pair	Association Description	Association Name
Web Page ↔ Page Renderer	Request to visit page, sends rendered page	request page, send page
Web Page ↔ Controller	Passes the user's desired action, informs of process completion	send user request, return
Controller ↔ Page Renderer	Passes necessary data for page rendering	send page data
Controller ↔ Security Query	Asks for data on specific security, send data on specific security	send security query, return security data
Controller ↔ Validity Checker	Requests order to be carried out, passes new portfolio data	send order, send portfolio data
Controller ↔ Game Handler	Passes updated Game settings, validates updated settings	verify fields, return fields
Controller ↔ Player Database	Passes updated settings, validates updated settings	verify fields, return fields
Controller ↔ Data Handler	Passes updated data, ask for portfolio data to perform process, return altered portfolio data	send updated data, request portfolio info, return portfolio info
Security Query ↔ Security Info Provider	Asks for security data, return security data	send security data request, service security data request
Security Query ↔ Validity Checker	Asks for to query specific security, return security data	send security query, return security data
Validity Checker ↔ Data Handler	Asks for Player's portfolio information for validity purposes, passes user's portfolio information	request portfolio data, return portfolio data
Validity Checker ↔ Trade Database	Sends order information to determine adjusted price, return updated price	request adjustment, update price
Data Handler ↔ Database	Stores incoming data, request certain data, retrieve needed data	store data, request data, retrieve data

5.1.3 Attribute Definitions

Concept	**Attribute**	**Meaning**
Data Handler	databaseHandle	Interacts with the database to service data requests.
Database	data	It includes all data used in the system, which includes League information, Player information, stock prices, fund settings, and transaction history etc.
Page Renderer	sufficientRenderData	Generates a page from database with updated information.
Trade Database	priceUpdate	Generates new price for the future orders.
Game Handler	settingsValid	Decides whether the Player's requests are valid for the given League.
Fund Handler	settingsValid	Decides whether the Player's requests are valid for the given Fund.
Validity Checker	fieldsValid, fundsValid, tradeSuccess	Compares funds and prices to make sure a Player's request is valid.
Advertiser	SufficientBalance,,advertiseValid	Upon successful completion of payment to Administrator, upload a banner on webpage

5.1.4 Traceability Matrix

		Domain Concepts										
Use Case	**PW**	**Player**	**Webpage**	**Page Rendere**	**Controller**	**Stock Query**	**Validity Checker**	**Liquidity Manager**	**Data Handler**	**League Handler**	**Fund Handler**	**Advertiser**
UC-01	10				X				X			
UC-02	4	X	X	X		X		X				
UC-03	19	X			X		X	X	X		X	
UC-04	19	X			X		X	X	X		X	
UC-05	5	X	X	X								
UC-06	10	X		X								
UC-07	13	X	X	X	X				X	X		
UC-08	9	X			X		X		X			
UC-09	9				X							
UC-10	9				X		X			X		
UC-11	5							X		X	X	
UC-12	2											X
UC-13	2	X	X	X	X				X			X
UC-14	13	X										
UC-15	18	X										
UC-16	9	X	X	X		X						
UC-17	3	X										
UC-18	1	X										
UC-19	2	X										
UC-20	5	X	X									
Max PW		19	13	13	19	9	19	19	19	13	19	2
Total PW		123	38	43	88	13	56	47	72	27	43	4

5.2 System Operation Contracts

Operation*:* Register Player
Preconditions:
- None

Postconditions:
- Player's background (name, age, gender, skill level, etc.) information is stored in database

Operation*:* Login
Preconditions:
- Provide correct username and password

Postconditions:
- None

Operation: Learning Tutorial
Preconditions:
- Player has valid portfolio in database
- Player has selected their skills level based on the their knowledge

Postconditions:
- Store Player's skill level in database

Operation: Buying Securities
Preconditions:
- Player has enough cash available to purchase different securities
- Enough Stocks available in the market to be purchased
- Transaction data is valid

Postconditions:
- Update database with Player's new stock holdings
- Update the Stock the inventory in database

Operation: Selling Securities
Preconditions:
- Player has the enough securities to sell
- Transaction data is valid

Postconditions:
- Update the database with Player's stock holdings
- Update the Stock inventory in database

Operation: Allow Advertisement
Preconditions:
- Administrator has allowed the Advertisements on web page
- Advertiser has their own account
- Advertiser has paid their past due bill

Postconditions:
- Update the webpage based on Advertiser's request

Operation: Challenge Player
Preconditions:
- Player has valid portfolio in the database

Postconditions:
- Create a championship between Players

Operation: Query Stocks
Preconditions:
- Player has valid portfolio in the database

Postconditions:
- None

Operation: View Portfolio
Preconditions:
- Player has valid portfolio in the database

Postconditions:
- Let the Player change his portfolio and don't allow a player to make changes in another player's portfolio

Operation: Create Fund
Preconditions:
- Player has valid portfolio in the database

Postconditions:
- Database is updated with the new Fund's information

Operation: Invite to League
Preconditions:
- Player has valid portfolio in database
- Player(Invitee) is already in the League

Postconditions:
- None

Operation: Create League
Preconditions:
- Player has valid portfolio in database

Postconditions:
- Database is updated with the new League information

Operation: Manage League
Preconditions:
- Player has access to the league privileges
  Player's request are valid

Postconditions:
- Upon successful completion of Player's request update database

Operation: Manage Fund

Preconditions:

- Player is a Fund manager
- Player's requests are valid

Postconditions:

- Information is updated in the database

Operation: View League Standings

Preconditions:

- There exists Leagues

Postconditions:

- None

Operation: Submit technical problems to Administrator

Preconditions:

- Player has valid portfolio in database

Postconditions:

- Send an Email to Administrator regarding the problem

Operation: Friend Referral

Preconditions:

- Player has valid portfolio in database

Postconditions:

- An invitation to join a game is sent

Operation: Share Score/Portfolio on social network

Preconditions:

- Player has valid portfolio in database
- Player has provided correct username and password for the social media network the Player is trying to share a score

Postconditions:

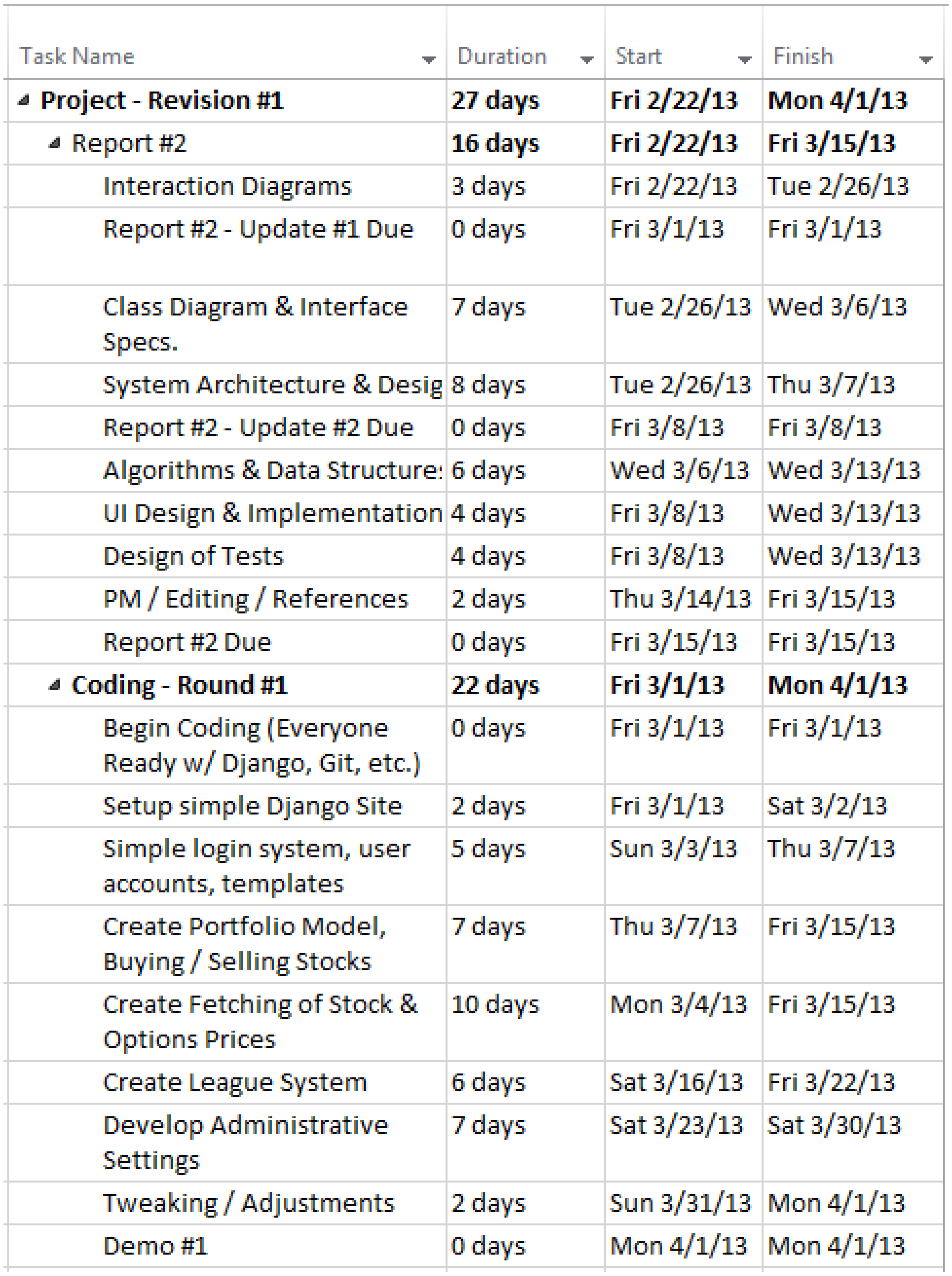
- None

6.0 Plan of Work

The following is our plan of work, up to & including the first demo. Within Report #2, we plan on including the updated roadmap for our project. As our project completes at the end of the semester, we plan on comparing our projected deadlines with the ones we have laid out. We will comment on this in Report #3.

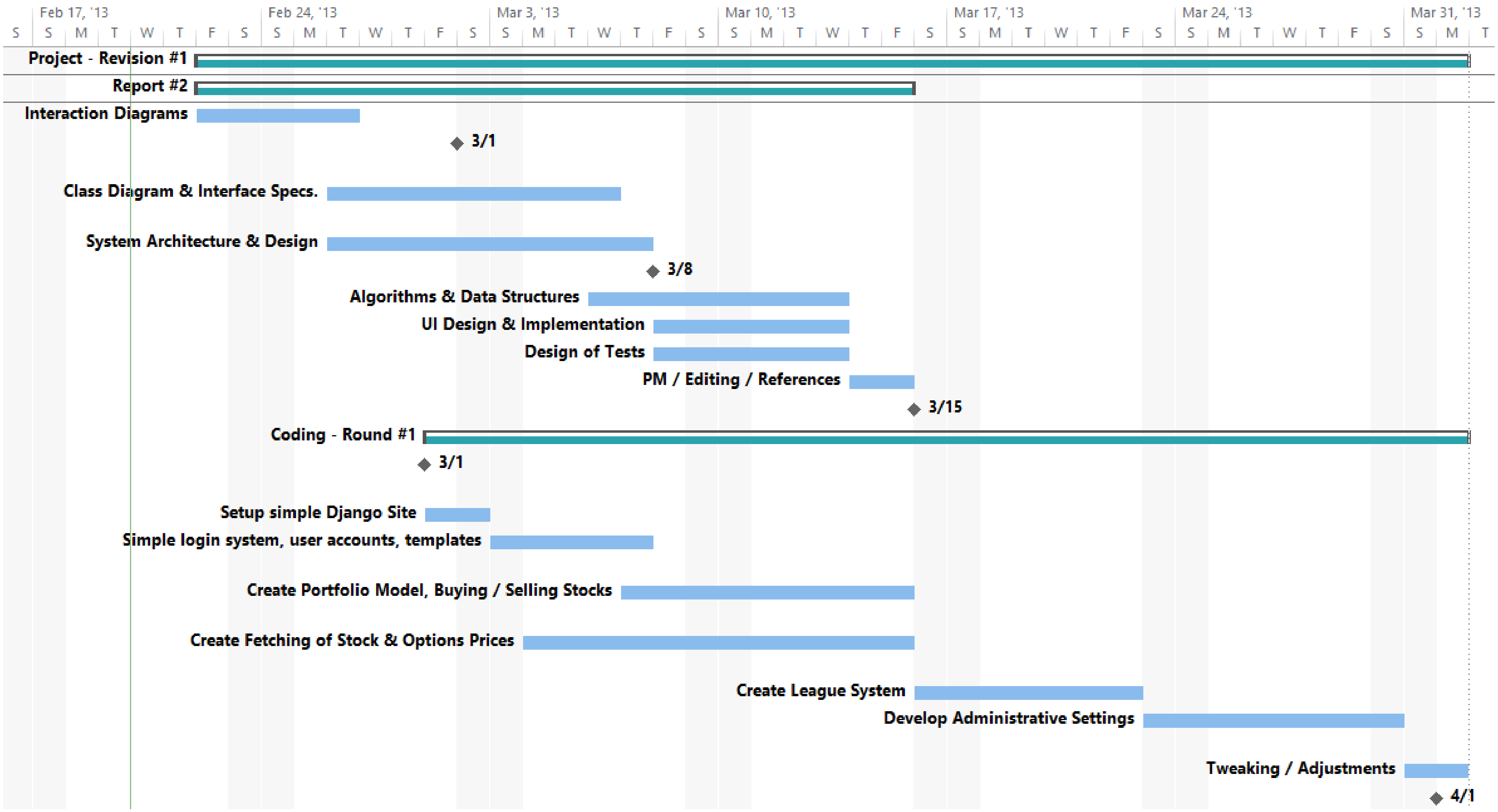
6.1 Project – Revision #1 Roadmap (Report #2 & Coding Round #1) - Task List

Task List generated via Microsoft Project:



Task Name	Duration	Start	Finish
Project - Revision #1	**27 days**	**Fri 2/22/13**	**Mon 4/1/13**
Report #2	**16 days**	**Fri 2/22/13**	**Fri 3/15/13**
Interaction Diagrams	3 days	Fri 2/22/13	Tue 2/26/13
Report #2 - Update #1 Due	0 days	Fri 3/1/13	Fri 3/1/13
Class Diagram & Interface Specs.	7 days	Tue 2/26/13	Wed 3/6/13
System Architecture & Desig	8 days	Tue 2/26/13	Thu 3/7/13
Report #2 - Update #2 Due	0 days	Fri 3/8/13	Fri 3/8/13
Algorithms & Data Structure:	6 days	Wed 3/6/13	Wed 3/13/13
UI Design & Implementation	4 days	Fri 3/8/13	Wed 3/13/13
Design of Tests	4 days	Fri 3/8/13	Wed 3/13/13
PM / Editing / References	2 days	Thu 3/14/13	Fri 3/15/13
Report #2 Due	0 days	Fri 3/15/13	Fri 3/15/13
Coding - Round #1	**22 days**	**Fri 3/1/13**	**Mon 4/1/13**
Begin Coding (Everyone Ready w/ Django, Git, etc.)	0 days	Fri 3/1/13	Fri 3/1/13
Setup simple Django Site	2 days	Fri 3/1/13	Sat 3/2/13
Simple login system, user accounts, templates	5 days	Sun 3/3/13	Thu 3/7/13
Create Portfolio Model, Buying / Selling Stocks	7 days	Thu 3/7/13	Fri 3/15/13
Create Fetching of Stock & Options Prices	10 days	Mon 3/4/13	Fri 3/15/13
Create League System	6 days	Sat 3/16/13	Fri 3/22/13
Develop Administrative Settings	7 days	Sat 3/23/13	Sat 3/30/13
Tweaking / Adjustments	2 days	Sun 3/31/13	Mon 4/1/13
Demo #1	0 days	Mon 4/1/13	Mon 4/1/13

6.2 Project – Revision #1 Roadmap (Report #2 & Coding Round #1) – Gantt Chart



7.0 References

1. Marsic, Ivan. Software Engineering. Rutgers University, unpublished. 2012. Web
2. “UML Class Diagram Help”, Class Draw. Macrospark Solutions, n. d. Web. 03 Feb. 2013.
3. MarketWatch. MarketWatch, 18 Oct. 2011. Web. 29 Jan. 2013.
4. Group 6. Bears & Bulls. 03 May. 2012. PDF file
5. Group 2. Stockhop: The Stock Market fantasy League Game. n. d. PDF file