Functional Requirements
for
"Capital Games"

Report 1: Part 2
Software Engineering
14:332:452

Team 2:

Jeff Adler
Eric Cuiffo
Nick Palumbo
Jeff Rabinowitz
Val Red
Dario Rethage

February 14, 2013
Version: 1

Contributions Breakdown

		Names					
Category	Points	Jeff A	Eric C	Nick P	Jeff R	Val R	Dario R
Project Management	10 Points	0%	0%	0%	0%	0%	0%
Customer Requirements	9 Points	0%	0%	0%	0%	0%	0%
System Requirements	6 Points	0%	0%	0%	0%	0%	0%
Functional Requirements	30 Points	0%	0%	0%	0%	0%	0%
User Interface Specifications	15 Points	0%	0%	0%	0%	0%	0%
Domain Analysis	25 Points	0%	0%	0%	0%	0%	0%
Plan of Work	5 Points	0%	0%	0%	0%	0%	0%

Contents

1 Functional Requirements Specification

1.1 Stakeholders

1.2 Actors and Goals

1.3 Use Cases

Before a user can participate in most of the functionality of our site, the user must first join or create a league. Ultimately, creating a league is very similar to joining a league, with the only exception being that the user becomes League Manager of a league that they create and that the new league must be added to the database. Therefore, the two will be represented as one use case with alternate scenarios. The responsibilities of the League Manager will be explored in a later use case. One relevant aspect of the responsibily of a League Manager to the use case though, is whether a league is made public or private; that is, whether it shows up in a public league listing page or can only be joined by direct invitiation from the League Manager. Thus, our first use case involves a business policy:

CG-BP01: So that a user may create a join leagues with only their friends, leagues marked as private will not show up on the league listings.

Thus, a user will only be able to browse listings of public leagues.

Use Case UC-1	**Join League**
Related Requirements:	ST-2, ST-21
Initiating Actor:	User
Actor's Goal:	To join or create a fantasy finance league
Participating Actors:	Database, Other Users, League Managers
Preconditions:	-A public league exists and has open positions
	-User is logged in
Postconditions:	-The **Database** is updated to reflect the user's participation in the lague
Flow of Events for Main Success Scenario:	

→	1. **User** navigates to leagues listing page
←	2. **System** displays public leagues available for the **User**, sorted by date created
→	3. **User** selects join on a league in which they are interesting in joining
←	4. **System** registers **User** into that league, notifying **Database** to update to reflect this change
	Flow of Events for Extensions (Alternate Scenarios):
3a. The **user** selects create league rather than join league	
→	1. **User** inputs desired league name and settings
←	2. **System** (a) creates the league and inputs it to the **Database** and (b) registers the **User** into that league as **League Manager**

It is important here to note another business policy of our site relevant to the user's experience:

CG-BP02: A user is able to join an unlimited number of leagues and become League Manager for as many leagues as the user wishes to create.

Though the settings are selected when creating the league, any League Manager can change the settings of their league at any time. These settings are comprehensive, including such items as name, privacy, number of spots, and rules which define how much money each league member receives to spend and other conditions under which all league members much operate for participation in that league. In addition, the League Manager can also manage members from the settings.

Use Case UC-2	**Change League Settings**
Related Requirements:	ST-22, ST-23
Initiating Actor:	League Manager
Actor's Goal:	To change the settings of a league and manage its members
Participating Actors:	Database, other Users
Preconditions:	-User is the League Manager of the league -User is logged in
Postconditions:	-The league settings have been changed as desired and the **Database** reflects the changes
	Flow of Events for Main Success Scenario:
→	1. **League Manager** selects the league settings option from the league page
←	2. **System** requests the current settings from the **Database** and presents them to the **League Manager** along with options to change the settings
→	3. **League Manager** updates the settings, such as privacy, league name, number of spots, rules, and managing users

←	4. **System** sends the updated settings to the **Database**
Flow of Events for Extensions (Alternate Scenarios):	
1a. The **User** selecting league settings is not the **League Manager**	
←	1. **System** requests the current settings from the **Database** and displays them, but does not provide ways to change them
4a. The **League Manager** has altered the status of a league member	
←	1. **System** will request the **Database** to update the **User**'s status in the league, be it becoming league manager or removing that **User**'s instance from this league (banned)

It is of some concern that League Managers may become abusive of their powers, and therefore it is important to create on a policy to explicitly state how this power is treated. In modern fantasy leagues (such as football, baseball, etc.), the League Manager does not typically have their power moderated, and this has not caused any problems in the success of these fantasy websites. The ability to leave a league and join another is left to the users if they feel that their league manager has become abusive. Their joining of the league acts as an implicit contract to accept of the League Manager's settings. However, if this League Manager becomes a problem and users bring it to an administrator's attention, disciplinary action may be taken. Thus we generate the next site policy:

CG-BP03: A League Manager is able to change the status of users in their league without moderation. However, if a League Manager is deemed abusive, a site administrator may take disciplinary action against them.

Core to our site is the ability of the user to have access to information about companies so that the user may make informed decisions on how he would like to invest. As this is so crucial to the functionality of this project, it is absolutely necessary to make information easily available to the user and presented in a way that is clear and easy to understand. Therefore, the search of companies as mentioned in ST-3 should be simple to use and intuitive and the display of company profiles as mentioned in ST-4 should be such that a user can easily access any desired information about the company's financial performance.

Use Case UC-3	**Browse Companies**
Related Requirements:	ST-3, ST-4
Initiating Actor:	User
Actor's Goal:	To bring up information on a desired company
Participating Actors:	Database, Financial API
Preconditions:	-Financial API is accepting inquiries
	-User is logged in
Postconditions:	-None worth mentioning
Flow of Events for Main Success Scenario:	

→	1. **User** begins entering a search term
←	2. **System** makes suggestions for companies in real-time
→	3. **User** enters search term or selects a suggestion
←	4. **System** (a) requests information from **Financial API** and (b) displays the information to the user in a clear and interactive manner
Flow of Events for Extensions (Alternate Scenarios):	
1a. The **User** selects a direct link to a company rather than enter a search term	
←	1. Same as step 4 above
3a. The search term is invalid, i.e. the company does not exist	
←	1. System informs user company does not exist and offers similarly titled companies as links

Note that the exact way in which the information requested from the Financial API is displayed to the user is not specified in this use case. This will be described instead in later sections about on-screen appearance requirements as to try to separate the functionality of the site and design of the site as separate as possible.

The goal of browsing companies ultimately is for the user to gain the knowledge needed to place market orders. Market orders are the atomic action of our site; i.e. the center point of every league is the user's ability to initiate transactions in an attempt to invest their money as best they can.

Use Case UC-4	**Place Market Order**
Related Requirements:	ST-5
Initiating Actor:	User
Actor's Goal:	To buy or sell stock, or to place a short, stop, or limit order
Participating Actors:	Database, Financial API
Preconditions:	-Financial API is accepting inquiries -User is logged in -User is a member of a league
Postconditions:	-**User**'s portfolio is updated to reflect change in position
Flow of Events for Main Success Scenario:	
→	1. **User** selects the league in which they would like to place the order
←	2. **System** displays prompt for market order, including type, amount, and company
→	3. **User** fills out form and requests the order be placed
←	4. **System** (a) requests market price from **Financial API** and (b) places the order into the **Database**

←	5. The order either resolves or expires, and the **System** updates the **User**'s position in the **Database** accordingly
Flow of Events for Extensions (Alternate Scenarios):	
1a. The **User** chooses to place a market order from a company's profile rather than from the league page	
→	1. The **User** selects which league in which to place the order
←	2. The **System** takes the **User** to league marker order prompt as described in Step 2 above, with the prompt for company already filled out
→	3. Go to Step 3 above
4a. The **User** does not have enough money or margin to place the order	
←	1. The **System** informs the **User** that they do not have enough money or margin to place the order and returns them to the market order prompt

The potential kinds of orders referenced in the above use case are defined in the glossary. The details on the necessary computations to enact these orders will be defined in a section later on.

In order to keep track of their own finances and any of their fellow league member's finances, a user must be able to view member portfolios. This keeps with the competitive nature of our site in addition to allowing the user to track their own progress.

Use Case UC-5	**View Portfolio**
Related Requirements:	ST-6, ST-7, ST-11
Initiating Actor:	User
Actor's Goal:	To view one's own finances or another's finances
Participating Actors:	Database, other Users
Preconditions:	-User is a member of a league -User is logged in -Database is tracking user's position
Postconditions:	-None worth mentioning
Flow of Events for Main Success Scenario:	
→	1. **User** selects a league member's profile
←	2. **System** requests that member's information from the **Database** and displays it in an organized and graphical manner to the **User**
Flow of Events for Extensions (Alternate Scenarios):	
2a. The **User** is viewing their own portfolio	
←	1. The **System** gives the **System** options to place market orders related to their existing positions

Once again, the exact display of information is not defined in the use case, but rather will be

explored further in the section about user interface specifications. Next to discuss is the tutorial as referenced in ST-8. We consider this to be one of the main aspects that separates us from previous iterations of fantasy stock leagues; our site will educate users new to finance and enable them to learn all about the world of finance and how to invest, in addition to how to these subjects relate to the use of our site.

Use Case UC-6	**Access Tutorials**
Related Requirements:	ST-8
Initiating Actor:	User
Actor's Goal:	To become educated in finance
Participating Actors:	None
Preconditions:	-User is logged in
Postconditions:	-None worth mentioning
	Flow of Events for Main Success Scenario:
→	1. **User** selects the tutorial option from the site's main page
←	2. **System** displays possible topics on which the **User** may be educated on
→	3. **User** selects topic
←	4. **System** presents an interactive tutorial to the **User**, which will be further elaborated upon in a later section

In order to maintain a clean fantasy finance experience for our regular users, site administrators will reserve the ability to moderate other users–issuing warnings, suspensions, or even bans for abusive activity. To put it explicitly:
CG-BP04: Site administrators will warn, suspend, or ban users for abusive activity–this includes aggressive behavior on league comments or user messages, spamming users, joining numerous leagues without active participation, and anything else that is deemed to harm the experience for other users.

Use Case UC-7	**Take Disciplinary Action**
Related Requirements:	ST-27
Initiating Actor:	Site Administrator
Actor's Goal:	To take action against an abusive **User**
Participating Actors:	Database, Users
Preconditions:	-Initiating actor is a **Site Administrator**
	-There are outstanding abuse reports
Postconditions:	-The **Database** is updated to reflect any actions taken against the **User**

	The abuse report shows that it has been resolved on the administration page
	Flow of Events for Main Success Scenario:
→	1. **Site Administrator** selects the site administration page option from the main screen (only viewable by **Site Administrators**)
←	2. **System** makes a request to the **Database** and displays all outstanding abuse reports
→	3. **Site Administrator** (a) selects an abuse report, (b) reviews the report, and (c) selects what action is to be taken (if any)
←	4. **System** implements the action selected by the **Site Administrator** and updates the **Database** accordingly

1.4 System Sequence Diagrams

Project Management

References