Interaction Specifications
for
"Capital Games"

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

Team 2:

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

March 3, 2013
Version: 1

Contents

1 System Interaction Diagrams

1.1 Financial Data Retrieval Subsystem

Enter the Capital Games Financial Adaptor

For the querying and retrieval of real time and historical financial data and stock quotes in a form that is both familiar and friendly to players of Capital Games, we will utilize the ***Yahoo! Finance*** Application Programming Interface (API), which allows for easy access of ***Yahoo! Finance*** stock data via data served via URLs that our system can retrieve, parse, and then translate for the use of Capital Games fantasy leagues platform. Since we will be drawing data from ***Yahoo! Finance***, it will be represented as external to the system of Capital Games. Internal to our system, however, will be the financial adaptor module that will automatically handle data retrieval from ***Yahoo! Finance*** based on user queries.

We chose this route over either option of having financial data querying and retrieval built-in to our system or taken from any other API because attempting to construct a built-in, live stock-querying system within Capital Games itself would have been both expensive and impractical — much akin to reinventing the wheel — and because ***Yahoo! Finance*** has proven itself as stable and reliable versus other available APIs. Thus, this section will explain our intended financial adaptor module for seamlessly delivering ***Yahoo! Finance*** data for use within Capital Games.

Essentially, by us deploying the a financial adaptor module into Capital Games, users will be able to easily search for stock data within our website and have it near-instantly displayed on the web page they are viewing without the user even being cognizant of all the work being done in the background via our financial adaptor module existing in our server. The financial adaptor module will have all the functionality for making requests for data from ***Yahoo! Finance*** based on user input and will actively draw and translate the raw data from ***Yahoo! Finance*** into a form that can be delivered within our own views ergo the data will be displayed on our web pages.

One consideration we need to take from our end for the building of our financial data is validating user queries for stock symbols. In other words, what would happen in the case that a user attempts to query a stock symbol, company name, industry, or sector that does not exist? To resolve such issues, our adaptor will also draw from our own database built into the website that keeps an updated list of valid stock symbols and names that is drawn from a source similar to ***Yahoo! Finance***, *EODData*. We are using *EODData* to supplement our use of the ***Yahoo! Finance*** API as *EODData* offers easy retrieval of all stock symbols and names in a method that is similar to ***Yahoo! Finance***. ***Yahoo! Finance*** unfortunately does not offer that particular feature, so we will be using *EODData* as a supplement to that, in that respect. We will essentially update our database via *EODData* and our financial adaptor module at each market opening and closure to account for any mergers, acquisitions, or any other major changes involving companies in the stock market.

Once user queries are validated by our financial adaptor module, our financial adaptor module will then parse the user query into a URL format that will allow for the retrieval of data via ***Yahoo! Finance***. Upon completing this, the URL will then be passed through our financial adaptor to ***Yahoo! Finance***, from which data will be returned to our financial adaptor module via a comma-separated values format (.csv, a container for easily passing volumes of data), which our financial adaptor will then translate into an arrangement that our views can utilize to deliver to the content to the webpage the user made the query from. From there, the user can then view the data and choose whether they would like to interact with the queried stock within Capital Games.

To elaborate on the technical specifications of our financial adaptor, the rest of this section will incorporate and explain interaction diagrams of methods used by our financial adaptor, illustrating the process I summarized regarding how our financial adaptor will go through interacting with ***Yahoo! Finance***, *EODData*, and the Capital Games platform.

All interaction diagrams will begin in the following page.

Financial Adaptor Interaction Diagrams

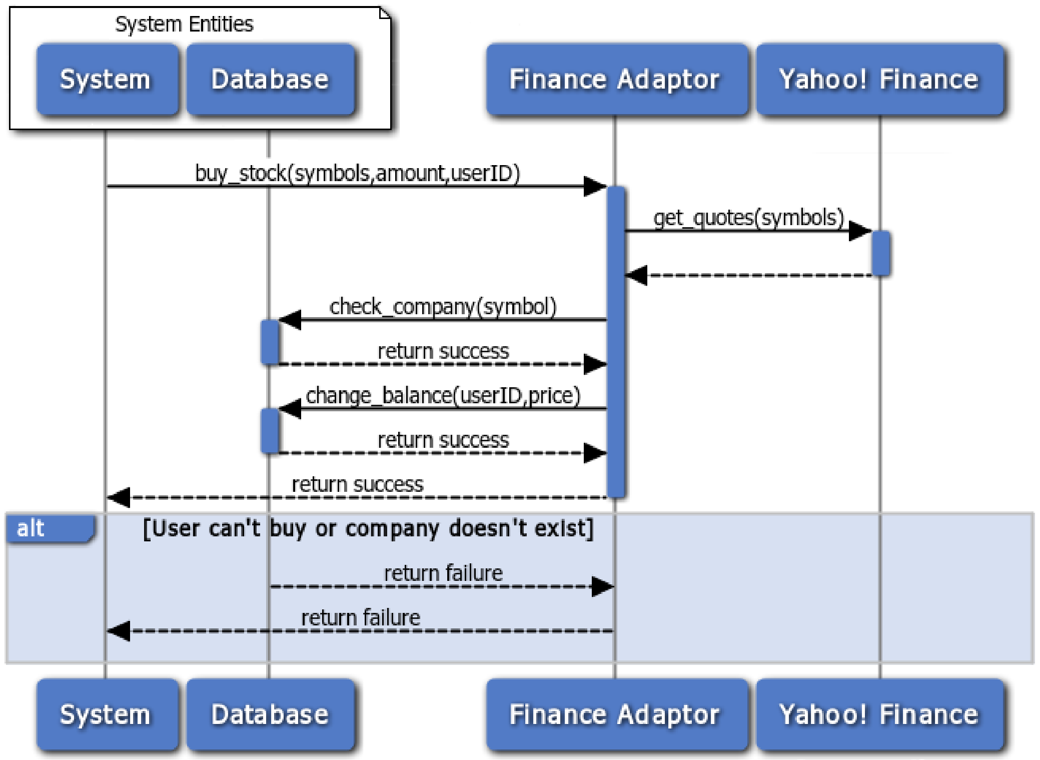

Figure 1.1: When a user buys a stock, the browser will inform the system of the transaction so that it can be approved. The system passes over the process to the finance adaptor who will check if the company is accepting trades, the current price from Yahoo! Finance, and if the user is able to afford the purchase from the database. If all goes well, the transaction will be recorded in the database and the balance will be changed. After all that is complete, the transaction will marked as a success and the system will be notified.

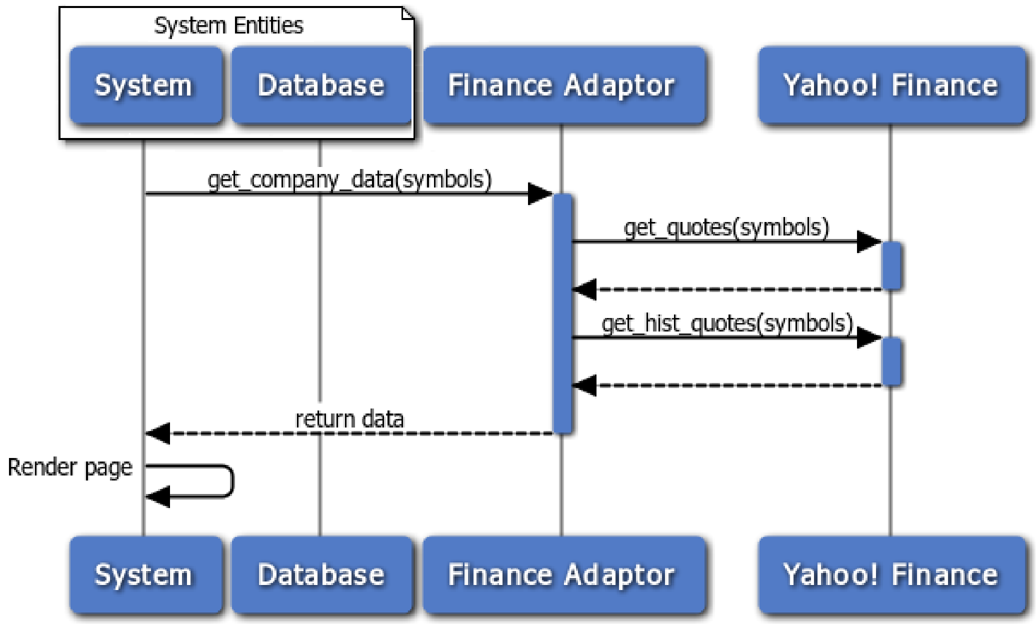

Figure 1.2: When a user wants to view a company page, the company data must be loaded from our finance API. Once the process is passed to the finance adaptor, the quotes and the historical quotes will be pulled from Yahoo! Finance and brought back to the system, who will prepare the page for the user.

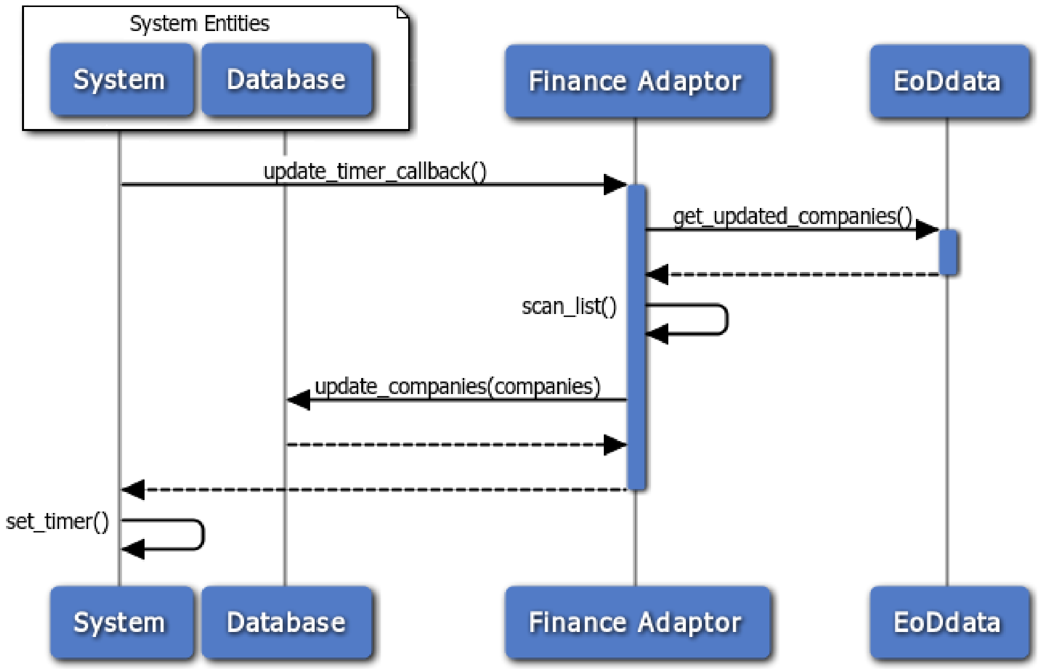

Figure 1.3: We need to keep a local copy of the current companies in our database so we can do rapid processes sing of all of the companies. In order to do this, there will be a timer that is set to update the database every once in a while. When the timer goes off, the system will pass the process onto the finance adaptor. The finance adaptor will then call data from EODData, who knows all of the current companies in the stock market. The finance adaptor will then scan the data for any new/deleted companies and change the database accordingly. After this is complete, the timer will start again so this process can loop.

1.2 Asynchronous Processing Subsystem

2 Plan of Work

2.1 Development and Report Milestones

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

Development milestones have been spread out following the completion of the first report on 22 February 2013, beginning with deploying our development environment and server through Heroku from which we continue to our next milestone of deploying Ruby on Rails as well as all the Gems and API packages we are incorporating into our project, most notably Yahoo! Finance.

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

Core goals leading up to Demo 1 include establishing all core functionality for Capital Games. This includes the following:

- **Rails framework-deployed core functionality :** This includes a working system for navigating the website, registering a new user account, logging in, and creating as well as participating in leagues.
- **Setting a foundation for the database:** On top of having the aforementioned core functionalities, they also must be able to pass data through a routed database.
- **Implementing the Yahoo! Finance API:**
- **A functional user interface:** Our website should be usable, and having a functional user interface from the start will give us a lot of room to expand and optimize the UI.

2.2 Breakdown of Responsibilities Introduciton

Contributions leading up to the completion of this report are covered in the "Contributions" table on the page following the gantt chart. For the future division of labor, we all plan on subdividing aspects of both the next reports as well as the development of the Capital Games Alpha.

2.3 Breakdown of Responsibilities

Responsibilities for server/development environment deployment and set-up will be shared between Val and both Jeffs, as all three equally have great experience in the subject. Meanwhile Nick, and Eric will work of sequence diagrams.

While all other diagrams on the report will be covered by both Jeffs, the User Interface Design and Implementation will be worked on by Val, Dario, and Eric. Nick and Dario will work on both data types and operations while Val and Eric also work on the traceability matrix.

System architecture and design will be covered by Nick and Val. Jeff R. and Dario will begin on the database structure and site routing via the Rails framework. Nick and Jeff A. will work on the implementation of users in the meantime.

From there we will further subdivide work on the final aspects of the website, likely sticking with our initial idea of splitting predominant responsibilities following the model, view, and controller (MVC) design pattern. Jeff R. and Jeff A. will lead work relating to model design, Dario and Eric will lead work relating to views and interface, and Nick and Val will lead work involving controllers. That being said, while the MVC pattern will model sub-component ownership among the team. Individual implementation responsibilities will be distributed a bit more evenly based on the particular strengths of team members.

In summary, Project Ownership will be based on the MVC architecture. To reiterate, Jeff R. and Jeff A. will have ownership over the Model portion, Dario and Eric will have ownership over the Views (user interface, etc.) portion, and Nick and Val will have ownership over the controllers portion. Even beyond Project Ownership, however, responsibility for the whole project will be shared and the success of our MVC architecture requires close coordination between all aspects.

Overall project success will be decided with how well the MVC component teams communicate and work with each other, as Capital Games will rely on the interactivity between the Model, Views, and Controller portions of the architecture.

2.4 Gantt Chart of Projected Milestones

Best viewed at 100% or greater:

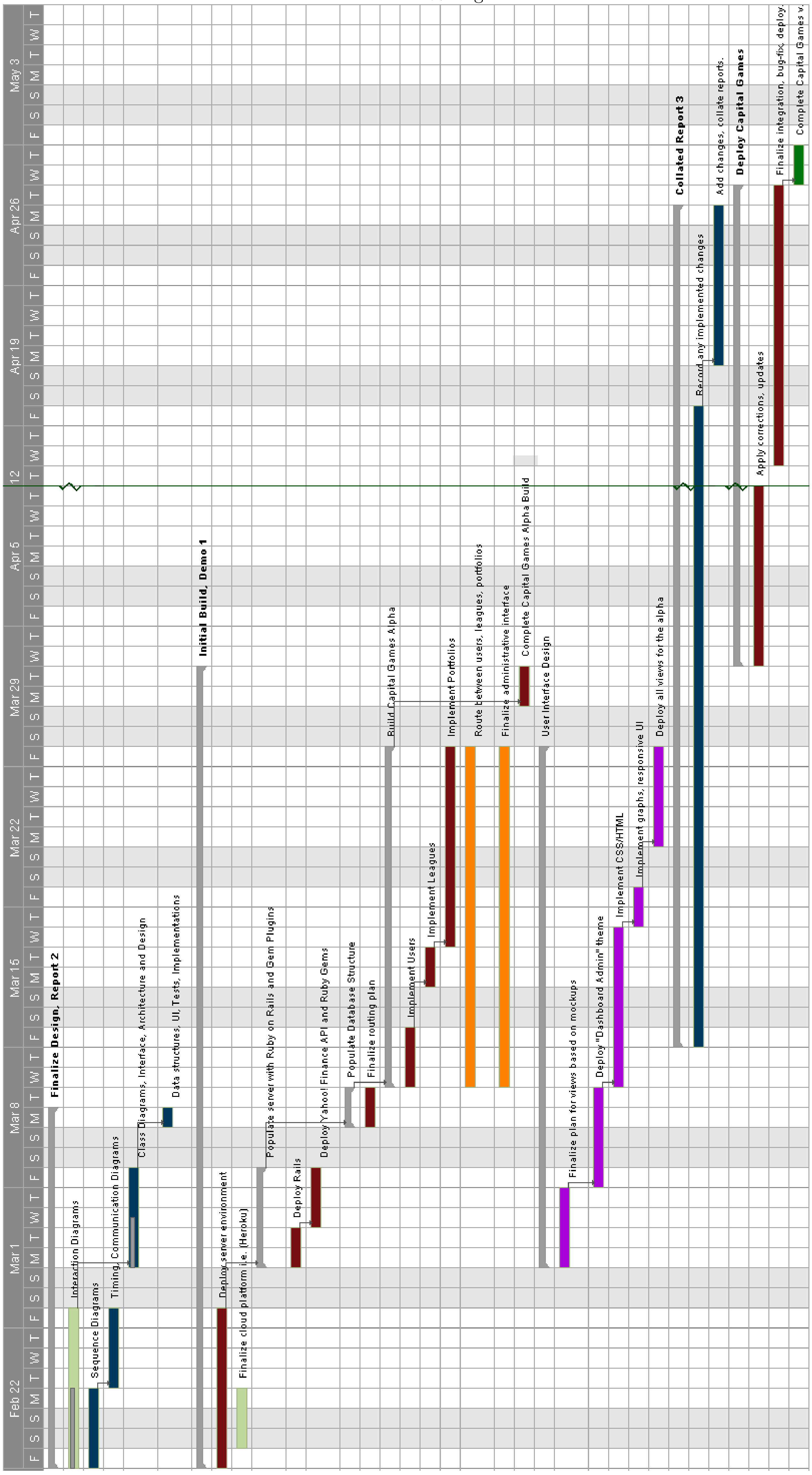

Figure 2.1: This gantt chart projects how we will concurrently work on the project. All blue items are report-related, red and orange relate to the core project development and purple illustrates UI milestones.

3 Project Management

3.1 Coordination and Division of Labor

3.2 User Story Matrix

3.3 Report 1 Contributions

		Names					
Category	Points	Jeff A	Eric C	Nick P	Jeff R	Val R	Dario R
Project Management	10 Points	0%	0%	40%	50%	0%	10%
Customer Requirements	9 Points	20%	0%	0%	80%	0%	0%
System Requirements	6 Points	0%	33%	0%	0%	33%	33%
Functional Requirements	30 Points	30%	10%	35%	0%	5%	20%
User Interface Specifications	15 Points	0%	33%	0%	0%	33%	33%
Domain Analysis	25 Points	25%	25%	10%	20%	20%	0%
Plan of Work	5 Points	0%	0%	0%	0%	50%	50%

References