Demonstration Specification
for
"Capital Games"

Demonstration 1
Software Engineering
14:332:452

Team 2:

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

April 7, 2013
Version: 1

Contents

		Names					
Category	Points	Jeff A	Eric C	Nick P	Jeff R	Val R	Dario R
Coding	30 Points	15%	20%	20%	15%	15%	15%
Unit Testing	10 Points	5%	15%	20%	30%	10%	20%
Integration Testing	10 Points	10%	10%	20%	10%	20%	30%
Web Design	10 Points	0%	50%	0%	0%	25%	25%
Debugging	5 Points	20%	20%	10%	20%	15%	15%
Documentation	5 Points	25%	25%	0%	25%	0%	25%
Data Collection	8 Points	0%	0%	25%	10%	50%	15%
Database Design	5 Points	10%	15%	25%	30%	5%	15%
Flyers	2 Points	75%	0%	0%	25%	0%	0%
Slides	3 Points	0%	0%	25%	25%	25%	25%
Activity Coordination	3 Points	15%	5%	20%	30%	30%	0%
Meeting Organization	3 Points	0%	0%	50%	50%	0%	0%
Website Administration	3 Points	100%	0%	0%	0%	0%	0%
Production Database Administration	3 Points	100%	0%	0%	0%	0%	0%

0.1 Itemized Contributions

- Jeff Adler
  - Provisioned web server and domain names
  - Authorized developer accounts on web server for team members
  - Initialized and administrated database, web server stack, and config files
- Eric Cuiffo
  - Bootstrapped CSS theme
  - Implemented UI specifications
  - Designed Comments system
  - Invented Activity Stream system

- Nick Palumbo
  - Coordinated Activities and Meetings
  - Designed Message system
  - Designed Research system
  - Assisted with Activity Stream system
- Jeff Rabinowitz
  - Coordinated Activities and Meetings
  - Designed RESTful Routes
  - Implemented Controllers and Models
  - Authored Unit and Integration Tests
  - Created Brochures
- Val A. Red
  - Installed and customized Financial Adaptor unit
  - Developed Research News system
  - Collected Data
  - Created Slides
- Dario Rethage
  - Developed Orders system
  - Designed logo
  - Authored Unit and Integration Tests
  - Designed database Models

1 Project Program Code Writing

Most programming was shared between all members of the group. This was coordinated during group sessions held during the week of 22-26 March 2013. Specific areas of focus that particular members were involved in included the Financial API (yfinadaptor.rb) reworked mostly by Val Red and integrated into the Rails' application code by Nick Palumbo. Most of the controllers, routing, and chart generation were shared between Nick Palumbo and Jeff Rabinowitz. Dario Rethage headed orders development with Eric Cuiffo.

Yahoo! Finance API (lib/assets/yfinadaptor.rb)

While Val A. Red did much of the development on the Yahoo! Finance API, he mainly worked with an open-source ruby iteration that received stock quotes from Yahoo! Finance as a hash and returned strings containing the respective data. Much of the work that had to be done for the purposes of Capital Games was the rewriting of the code to repurpose the string back into a hash that could be invoked by our rails application. In addition, values had to be returned in the hash converted to cents and returned as an integer to be compatible with our monetary value parser. Percentages and ratios, however, were left as float values. Essentially, all financial data drawn by Capital Games was due to this code being called upon by other parts of the rails application.

2 Unit & Integration Testing

Testing is a critical part of software engineering – just as important, if not more so, than the actual programming. Without tests, how can one know that anything works as intended?

There are two types of testing we intend to perform on our project: unit and integration testing. Both live within the "spec" folder of our directory.

2.1 Unit Testing

Because we programmed our application using Ruby, we programmed our unit tests using the RSpec programming suite. RSpec provides a natural and domain-specific language for testing individual suites of functionality, both within generic Ruby, and especially within Ruby on Rails web frameworks such as ours. Our tests check the responsiveness of both individual attributes of different modules, and even that they behave in controlled manners.

Unit Tests were written jointly by Jeff Rabinowitz and Dario Rethage on the data models used by the application.

2.2 Integration Testing

Integration testing is always more difficult than unit testing, because of the need to test as many interactions as possible between units. To simplify this, we will leverage the Capybara testing suite, which utilizes the Selenium Web Driver to simulate end-user interactions and test the expected output against the actual results of the interactions. Due to time constraints, we were unable to author integration tests for the application, which is something we intend to remediate promptly.

Integration Tests will be written jointly by Jeff Rabinowitz and Dario Rethage on the behavioral usage of the site.

3 Webpage Design

Our web page design adapated bootstrap as well as elements of the template "Dashboard Admin". Eric Cuiffo worked tirelessly in order to integrate both to make our website as responsive as possible. Dario Rethage contributed to the graphic design of the logo and the modification of certain AJAX elements.

4 Requirements Testing & Debugging

Most testing was done in-house as a shared responsbility by all group members. Particularly, debugging had to be done with the Yahoo! Finance Adaptor, the controller, as well as the server. Jeff Adler predominantly dealt with the inital server debugging of Capital Games. All application side debugging was done by the respective authors.

5 Program Documentation

Dario Rethage authored end-user documentation, included in the docs folder. Jeff Rabinowitz used RDoc to generate HTML pages of the technical documentations, compiled from source code. It is also included in the docs folder. Finally, the README's for bootstrapping are, again, in the docs folder.

6 Database Design & Maintenance

The database models were jointly developed by Jeff Adler, Nick Palumbo, Jeff Rabionwitz, and Dario Rethage. The production database is maintained by Jeff Adler.

7 Data Collection

8 Brochures & Slides Preparation

All members contributed to the writing, publishing and distribution of brochures. A copy is attached in the following pages.

Slides were prepared by Val A. Red on Microsoft Powerpoint based on input by Jeff Rabinowitz. These slides are also attached in the following pages. Images were obtained over the internet and photoshpped to make the presentation look professional; an image of Professor Marsic included among them in the slide explaining who "Capital Games" is for in order to stress how universally approachable our web application is supposed to be. Essentially, for the powerpoint, we focused on the educational and responsive aspects of Capital Games to make our audience more familiar with what makes our web application a unique and attractive option.

Brochures were prepared by Jeff Rabinowitz and Jeff Adler using basic principles of our site as well as some of the mockups of our application from the initial specifications.

9 Project Management

9.1 Activity Coordination

Most group activity coordination was done either through a Facebook Group including all members of the group or by SMS text via a smart phone application called "GroupMe." We coordinated assignments for each group member based on their skills and/or volunteering to take responsibility for a certain aspect of the project. Typically, Jeff Adler took on the lower level server-side responsibilities, Eric Cuiffo took on broadly implementing the user interface, Val Red adapted external modules for our website (such as the news feed and financial data from Yahoo!), Jeff Rabinowitz fleshed out the database and underlying structure/configuration of the website, Dario Rethage developed and implemented the logic and forms for transactions, and Nick Palumbo on tying individual elements together–such as plugging in Yahoo! Finance or Highcharts for UI. In order to ensure our code was managed and merged smoothly, we used a Git repository, which is a form of version control. It features simple public or private file management for any number of users, automatic merging, issue generation, and other features, making it incredibly useful for our purposes.

9.2 Organizing Meetings

Meetings were predominantly organized by Nick Palumbo and Jeff Rabinowitz. We met mostly at the Hill Center's iLabs. Meeting times and responsibilities were designated or confirmed through either Facebook or GroupMe, and both proved very effective. In general, meetings were scheduled weekly (typically on Mondays) to discuss and divide responsibilities. In this way, we were able to keep everyone up-to-date on what was expected on them and to meet deadlines with regularity. In case that someone was unable to make one of the meetings, Val Red kept minutes of each meeting to keep everyone current on what was discussed. Much of our coding and report work was facilitated during these meetings, and the meetings typically resulted in our most productive cycles of work as we were able to motivate one another in an in-person environment. All in all, our meetings were easy to coordinate, went smoothly, and were very successful.

9.3 Web Server Administration

In order to achieve full site functionality with Capitalgam.es much configuration was necessary to ensure that each individual part integrates successfully into one cohesive unit. We chose to host our website using a VPS(Virtual Private Server) hosted by ChicagoVPS, purchased the domain name Capitalgam.es from en.gandi.net, and used Amazon's Route 53 as our Domain Name System. Our

host came preconfigured with Debian 6(Squeeze). Having years of experience using Ubuntu and other user level Linux distro's it was an easy transition to learn the ins and outs of Debian as it is the parent of Ubuntu. I will now walk through the configuration process that was necessary to get Capitalgam.es to successfully deploy in production mode.

The first things necessary install were apache2, and ruby. While apache2 can easily be installed using apt-get, our group is working with a very specific version of ruby, by first installing RVM (Ruby Version Manager) we were able to select the specific version of ruby (1.9.3) that we developed in. After installing ruby, the next installation was RubyGems, which handles all required Ruby package installations. Rails does not naturally integrate into apache2, in order to facilitate this integration Phusion Passenger is needed. After installing the Phusion Passenger gem using RubyGems, the apache2 configuration file needed to be modified to specify the passenger module locations for when apache2 looks for passenger after reading the Virtual Host file.

For any site running on apache2, a virtual host file is required in order to direct apache2 to the correct directory based on the port the server is being accessed from (port 80 by default). With the Phusion Passenger configuration set for passenger module use, it was just necessary to add the app path to the virtual host file and then use the standard apache2 command a2ensite then service restart apache2 to enable it. By using github, our team was easily able to update our webapp locally on our own computers, push it to github, and then pulls it back down onto the server. By having the git repository in the same directory that the virtual host is pointed to, all that was needed at this point was having mysql installed, properly configured, and pointed to by the /config/database.yml file in our webapp. Using the command passenger start e production d started our app in production mode detached as a daemon which allowed us to have the webapp run 24/7 as a background process unowned by a user which is vital to ensuring the process doesnt die on user logout.

A large problem faced while detaching processes with passenger is the way the daemons are tracked. When a detached passenger session is launched a .pid file is generated in the current directory. This .pid file contains the process ID of the daemon. Without this file passenger will not be able to stop the server as it does not know the process ID of the server. When first working with passenger I had been launching the server from locations that did not have write access. This caused the PID files never to generate and created a large obstacle when we wanted to take down the server. The solution to our problem came out of using the ps aux command to manually locate the daemons needed to be killed then using the kill -9 system call to kill them.

9.4 Production Database Administration

In production we used MySQL2 along with the MySQL gem, as previously stated this was specified in /config/database.yml of our webapp. We have our login defaulted to Username: root Password: none due to us not having front end access directly to the server, the Database can only be accessed after first authenticating through SSH which is secure.