**Project Transition**

**Usage Instructions**

How to use Odin: For Users

**Menu**

The little 3 line in the top left is a menu. Click on that to open the menu on the side, which will give you navigation options.

**Registration**

1. Access the login page to register a new user.
2. Click on the sidebar menu.
3. Click on “Register” from the sidebar menu; it will take you to a new page.
4. Enter your Username, First Name, Last Name, Email, and Password.
5. Click “Register” and the page should link back to the login page with a confirmation message that you have successfully registered.

**Login**

1. Access the login page from the main website page.
2. Enter your username and password.
3. Press “Sign In” and the page should redirect to your Odin Portfolio Manager.

**Add a Portfolio**

1. Open the sidebar.
2. Click on “Add Portfolio”
3. Enter a Portfolio name and description.
4. Press “Add” and the page should redirect to the main Portfolio Manager with your newly added portfolio displayed on the page.

**Remove a Portfolio**

1. Open the sidebar.
2. Click on “Remove Portfolio”
3. Check the checkboxes of the portfolios you wish to remove.
4. Click on “Remove”.
5. You should be redirected to the main Portfolio Manager page with a confirmation message that you have successfully deleted the portfolio.

**Add a Stock**

1. Make sure you have at least one portfolio already created (you can only add stocks to existing portfolios).
2. Open the sidebar.
3. Click on “Add Stock”
4. Check the portfolio(s) in which you wish to add the stock to.
5. Enter the desired stock market Ticker and the Market (eg. “GOOG” for Google and “DJI” for Dow Jones Industrial).
6. Press “Add” and you should be redirected to a new page to input fields for your stock.
7. Select the desired fields using control click or shift click. (Select all by clicking on a field then pressing Ctrl+A).
8. Press “Add” and you should be redirected to the main Portfolio Manager page with the confirmation message “Fetch XML Created!”.
9. The feed handler takes up to 15 seconds to fetch the information from the stock feed. Refreshing the page after a short while will reload the fields you have selected for your stock and the respective values.

**Remove a Stock**

1. Open the sidebar.
2. Click on “Remove Stock
3. Check the checkboxes of the stocks you wish to remove.
4. Click on “Remove”.
5. You should be redirected to the main Portfolio Manager page with a confirmation message that you have successfully deleted the stock.

**Create a Graph of a Stock**

1. Open the sidebar.
2. Click on “Graphical View”.
3. Enter the ticker of the stock you wish to view.
4. Select the Time Span for the graph of the stock you wish to view using the dropdown menu when clicking on the default option of “1 day”.
5. Using another drop down menu, select the Chart Type. Line graph is the most typical and will display data in the form of averages per unit of time. Bar will show the range of stock traded in that unit of time. Candle displays a statistical box plot for each unit of time
6. Choose the chart scale, either “Arithmetic” or “Logarithmic”.
7. Set a Moving Average Indicator. This is basically a second line that averages the price over the specified period of time to view long term changes.
8. Press the “Generate” button and the Graph should appear on the bottom of the screen.
9. You may change any of the settings above and generate more graphs that will appear under the already generated graph(s) without having to reload the page.

**Refresh the Page**

1. Open the sidebar.
2. Click “Refresh”
3. The page should redirect to itself and reload.

**Log Out**

1. Open the sidebar.
2. Click “Logout”.
3. You should be logged out and redirected to the login page.

**Hack into Tony’s Account**

1. Sign out
2. Go to the Login page
3. For username enter: “' OR username IS NOT NULL OR username = '”
4. Enter anything for password
5. Press Sign In
6. You should have access to Tony Simoutttotut’s Odin profile.

If you experience any problems, please contact your local administrator.

**Server Setup Instructions**

Ubuntu 14.04 is the recommended Linux distribution for this application. Other distributions and versions may work, but have not been tested.

Next, you’ll need to install some dependencies. You’ll need the following:

-Apache2

-MySQL

-Python

-Python support for CGI

-Python support for MySQL

-PHP

To install the server, extract the contents of odin.tar.gz (included on LMS) into your main webserver directory, by default /var/www/html.

Once this is setup, configuration of the MySQL server is required. Create a user ‘odin’ with password ‘odintest123’ (configurable in config.php), with all permissions on the local server. Run database.py (located in the scripts directory) in debug mode to set up the database with tables and schema required for Odin to run. By default, the database name is ‘odindatabase’, but this is configurable in database.py and config.php. The server is now ready. To configure Yggdrasil, go into the yggdrasil directory (in scripts) and set the correct IP addresses for your setup in conf.py. For security purposes, make sure that the scripts directory is not readable from the web, otherwise the

To start the backend for the server, make sure that Apache is functioning, and then run ‘python yggdrasil.py’, ‘python bifrost.py’, and ‘python database.py’ (in that order). The output from each of these commands will provide useful debugging information for getting the configuration to work.

**Final Test Results**

Final Test Statement

The test plan specified the testing of functionality first. In order to pass, 100% of the test cases must pass for each of the testers that volunteered. The only issue that occurred was when one of our testers Chris Sacco ran Test Case 1 to register a new user. He attempted to register a username as chris, which was already in the database as Chris Weir’s account username. This caused Chris Weir’s account to be overridden by the new request. This bug was fixed quickly by simply implementing code to check for duplicate usernames and prevent the account creation from submitting if a duplicate username is detected. No other issues from the Test Cases occurred.

For the Interface portion of the test plan, the average of the individual scores for Usability resulted in a 9.125 out of 10. We found that 5 out of 8 testers praised the implementation of check boxes and selection boxes. The only complaints were around the time it takes for Bifrost to pull the stock information from the stock feed after the user has submitted a stock request. The reason for this is uncontrollable due to the supplied, free stock feed from Yahoo! that implements a 15 second delay. Because of this and the fact that the average Usability score was a 9.125 that surpassed our required score of 8.0, the team decided to not improve this implementation for the release.

The Appearance specification for the Interface portion of the test plan received an average of 9.5 out of 10. Testers complimented the “simplistic” look of the web interface and the “elegance” of the sidebar implementation. Some minor errors with spelling and formatting were spotted by some testers and fixed shortly after. Some of the users also did not feel displaying the exact database query number as confirmation messages was necessary and took away from the appearance and separation between the User and the backend of the product. Example: “Removed Portfolio[64]”. This was something that was kept for testing and can be easily removed. Because the high score of 9.5/10 surpasses the minimum set requirement of 8.0 out of 10 for appearance, the software passes the Interface portion of the test plan.

After some minor tweaks and debugging the application performs its core functionalities consistently and it’s graphical user interface passes the targeted value for Appearance and Usability for the majority of testers. Thus, it is safe to say this software is ready for production use based on the recorded results from the recorded results from the test plan drafted for this application.

**TESTING RESULTS**

|  |  |
| --- | --- |
| **Tester Number** | **Tester Name** |
| **1** | Andrew Groskopf |
| 2 | Nick Dvorak |
| 3 | Zach Hurley |
| 4 | Ben Hurley |
| 5 | Chris Sacco |
| 6 | Mike Belanger |
| 7 | Anton Nekhai |
| 8 | Jonah Duch |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Individual Results** | **Overall**  **Result** | **Comments** |
| 1. RegisterNewUser | 1. Pass 2. Pass 3. Pass 4. Pass 5. Fail 6. Pass 7. Pass 8. Pass | Fail | Multiple registration for Accounts with the same username overwrites the previous Account information. Simple fix to check for existing username is necessary.  “Shouldn’t it ask for my password twice?” |
| 1. UserLogin | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Standard” |
| 1. AddPortfolio | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Pretty Simple” |
| 1. AddStock | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Having two pages for this is kinda strange.”  “You should make it so I can go back and add fields to a stock I already added on the portfolio.” |
| 1. RemovePortfolio | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Okay.” |
| 1. RemoveStock | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Is that it?” |
| 1. Save/Load Portfolio | 1. Pass 2. Pass 3. Pass 4. Pass 5. Pass 6. Pass 7. Pass 8. Pass | Pass | “Looks like my stuff got saved.” |

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface** | **Individual Score** | **Overall**  **Result** | **Suggestions and Complaints** |
| Usability | 1. 10 2. 10 3. 8 4. 7 5. 10 6. 10 7. 9 8. 9 | 9.125 | 1. “Simple layout.” 2. “Pretty clear cut which is good I guess.” 3. “It should probably check the stock ticker and market ticker that I put or I can just put random ones that make no sense.” 4. “I don’t like that I have to wait a while for it to pull information. It should at least tell me that or have like a built in scroll thingy that refreshes the page when it’s done.” 5. “Yeah it’s very easy to use 10/10.” 6. “I’m not sure what I’m trying to do but it seems to work right.” 7. “Definitely useable.” 8. “Not gonna lie I would probably never use this, but if I did it seems pretty useable.” |
| Appearance | 1. 10 2. 10 3. 9 4. 9 5. 10 6. 8 7. 10 8. 10 | 9.5 | 1. “Looks pretty good.” 2. “Pretty sure it’s just cuz you used Twitter Bootstrap but okay.” 3. “Grahpical” Spelled Wrong 4. ”The graphs for the graphical plugin don’t move like the rest of the page when I the sidebar opens.” 5. “It’s nice. I like the sidebar.”  “Can you get rid of the ‘Successfully deleted portfolio[72]!’? The number is confusing since I don’t have that many portfolios so I don’t know what it represents.”  1. “Can I change the color scheme?” 2. “Looks like a phone app… Actually it looks pretty good on a phone.” |

**Status Report & Contribution Summary**

**Chris Weir**

Chris ensured code was packaged for installation, determined dependency list, wrote installation instructions. Chris also made sure the best practices were fulfilled and wrote the contribution summaries and status report.

**Tong “T0ng” Liu**

Tong summarized the testing data, wrote the testing section, and put the data in a table. Tong also wrote the user instructions and made them beautiful. Tong also made sure the best practices were fulfilled.

**Anthony “Tondiggidy” Simonutti**

Tony packaged the installation into a tar.gz file for upload.

**Cody “Codigail” Doyle**

Assisted in editing the user instructions and installation instructions

Since the last stakeholder review, we have gathered testing data, fixed outstanding bugs in our code, and fully cleaned up the code into an installable tar.gz. The code for the installation was put into a single directory, and compressed into a tar.gz archive, which can be easily distributed. Each member went out and got testing information from friends, and compiled it into a document. We also went through our code, and fixed some of the silly bugs on the front and backends as we found them. Only important, application breaking implementations were fixed. Some of the interface design and layout requests were ignored since the application passed by our Test Plan scoring method for Usability and Appearance. As the project comes to a close, we can look back and take away the value of planning and design, which can save huge amounts of time in the long run. It has been a long and memorable process. In the end, our goals were accomplished and we can proudly present the software we created as ready for production use.

**Best Practices Fulfilled**

First 4 are required, in addition to 3 others

1. Project Management Website ← Required (and fulfilled)

<https://code.google.com/p/sd-and-d-project/>

1. Code Repository ← Required (and fulfilled)

<https://code.google.com/p/sd-and-d-project/source/list>

1. Documented Coding Standards ← Required (and fulfilled)

<https://docs.google.com/document/d/1SCp_QcEjwyQlpdp3kNW5xzDY1HU6pKXWxiXItvmD8bQ/edit>

1. Use the basic concepts behind object oriented design ← Required (and fulfilled)

-We used classes for each object

-Plugins inherit their basic structure from the template class

1. Mock Objects For Unit Testing

-Yggdrasil, Bifrost, and Frigg can all run standalone, and since they talk to each other, we just use made up inputs and outputs to simulate the others while testing one at a time.

-Testing the web interface was done with old data generated by Yggdrasil, Bifrost and Frigg

1. Third Party Component or Tool ← We have Apache2, bootstrap, etc…

-Apache2 Webserver (for the web interface)

-PHP (Tony *loves* PHP)

-MySQL (for the databasing)

-Bootstrap (for making it pretty)

-jQuery (for more advanced bootstrcap functionality and javascript)

1. Design Patterns ← We probably used some of these or something

-Observer - The observer pattern was used as an event handling method for notifying Bifrost and Frigg of new stock requests. In essence, a stock object that contained a compact string in XML format was kept as a subject. Once the user fills out a stock exchange request form, the state of stock object will change along with the XML string it contains. An Observer notifies both Bifrost and Frigg in which both of them will pull the new data from the XML string and do their respective tasks with the new data. Bifrost will pull stock information from the stock feed and Frigg will create a new stock query based on the chosen stock and requested fields.

We used trello until we moved over to scrum meetings:

