**\*|∅ ELABORATION**

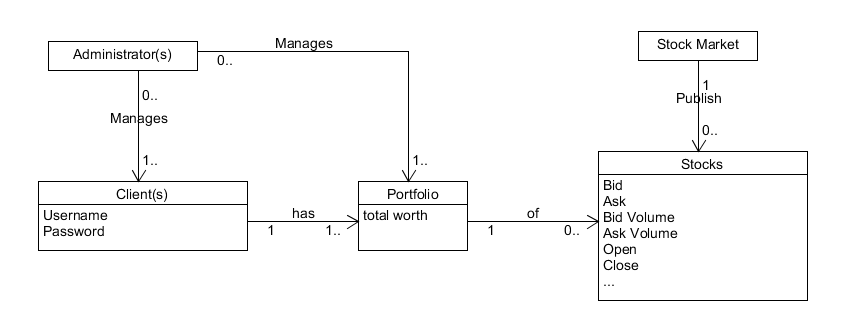
Date: 10/9/2014

Team Name: \*|∅

Team Members: Cody Doyle, Tong Liu, Tony Simonutti, Chris Weir

Product Name: Yggdrasil

**DOMAIN MODEL DIAGRAM**



**SUPPLEMENTAL SPECIFICATIONS**

MoSCoW

Must

1. Server fetches desired stock information from stocks API.
2. Working database that stock history can be stored in and accessed from MySQL.
3. Bifrost feed handler processes and delivers data to Yggdrasil main program for storage in Database every 15 minutes or on demand
4. Plugin can create Custom Portfolios containing a user’s designated stocks.
5. Graphical Plugin displays graphical representation of stock data across a user defined time scale.
6. Program runs on Chrome browser.

Should

1. Accounts can be created and saved for custom user log-in to access portfolios.
2. User can set and log in with a chosen username and password.
3. Plugin can load a stock price graph with the starting time equal to the date the stock was first added to a user’s portfolio.
4. Graphs can be refreshed to display
5. Consistently adhere to coding standards located here: <http://www.possibility.com/Cpp/CppCodingStandard.html>.
6. System can be installed onto a 2010 or later model standard personal computer within 15 minutes.

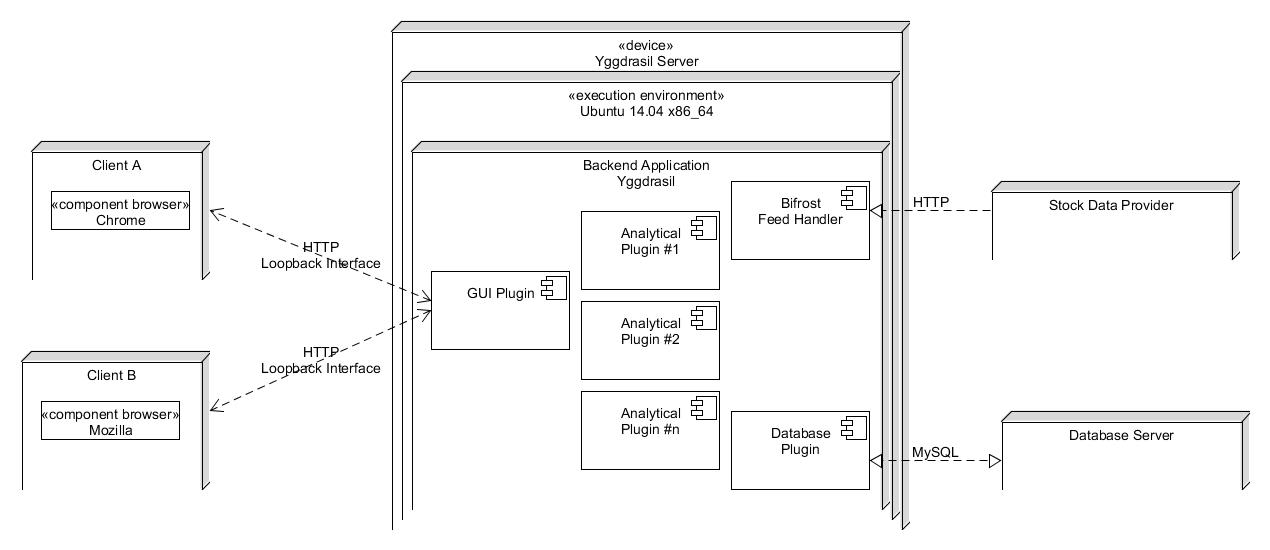
Could

1. User has the ability to change their password.
2. System requires no more than 30 seconds to pull stock data from API and store the information on the database.
3. Database storage efficiency: Each stock’s price history for the last 8 months takes up not more than 1GB.

Won’t

1. System displays notification in the event of a hardware failure.
2. Backup entire database to another hard drive.

**DEPLOYMENT DIAGRAM**



**USE CASES**

Use Case List:

|  |  |
| --- | --- |
| Yggdrasil  [X]Current stock information query  [X]Add symbol to subscription list  [X]Remove symbol from subscription list  [X]Retrieve subscription list  [X]Open Plugin  []Close Plugin  [X]Install Plugin | Graphical  [X]Load Graph  [X]Load General Graph  [X]Load Personal Graph  [X]Refresh Graph  [X]Export Graph |
| Portfolio Display Interface  [X]Save Portfolio  [X]Load Portfolio  [X]Export Portfolio  [X]Import Portfolio  [X]Add Ticker to Portfolio  [X]Remove Ticker from Portfolio | Database  []Query database for ticker history  []Create new database file  []Save database file (export database)  []Load database file (import database) |

**YGGDRASIL\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| **Use Case:** Current stock information query  **Description:** the system shows information on a stock that the user specifies  **Actors:** User  **Preconditions:** Software is installed and configured  **Flow of Events:**  Use Case starts when customer selects Check Stock Price.   1. user enters the ticker symbol that they want info on 2. user selects which fields they are interested in 3. user submits request 4. system displays “fetching data” 5. system displays data   **Postconditions:** System displays data | **Use Case:** Add symbol to subscription list  **Description:** the system adds a symbol that the user specifies to the subscription list  **Actors:** User  **Preconditions:**  Software is installed and configured  **Flow of Events:**  Use Case starts when customer selects Add Subscription.   1. User enters symbol they want to add 2. System displays subscription added   **Postconditions:** none |
| **Use Case:** Remove symbol to subscription list  **Description:** the system removes a symbol that the user specifies from the subscription list  **Actors:** User  **Preconditions:** Software is installed and configured  **Flow of Events:**  Use Case starts when customer selects Remove Subscription.   1. System displays current subscription list 2. User enters symbol they want to remove from the list 3. System displays subscription removed   **Postconditions:** none | **Use Case:** Retrieve subscription list  **Description:** The system displays all current subscriptions  **Actors:** User  **Preconditions:** Software is installed and configured  **Flow of Events:**  Use Case starts when customer selects Display Subscription.   1. System displays subscription list   **Postconditions:** none |
| **Use Case:** Open Plugin  **Description:** System opens interface for specific plugin.  **Actors:** User  **Preconditions:** Plugins are installed for the user.  **Flow of Events:**  Use Case starts when customer selects Plugins in the Menu.   1. System displays list of currently implemented plugins 2. User selects one of the plug ins   **Postconditions:** System opens plug in menu | **Use Case:** Install Plugin  **Description:** System installs plugin specified by the user.  **Actors:** User  **Preconditions:** User has the location of a Plugin that can be utilized by the system  **Flow of Events:**  Use Case starts when customer selects Install Plugin in the Menu.   1. System asks the user for the location of the plug in they want to install 2. User inputs the location of the plug in file 3. System displays “Checking file” 4. System displays “verifying that that the file can be installed” 5. System installs the plug in   **Postconditions:** System adds the plug in to the list of plug ins |

**PORTFOLIO DISPLAY INTERFACE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| **Use Case:** Open Portfolio  **Description:** System opens a previously portfolio of ticker symbols  **Actors:** Users  **Preconditions:** User previously saved a portfolio of ticker symbols  **Flow of Events:**  Use case starts when the user selects “open portfolio”  1. System displays all previously saved portfolios  2. User selects one  3. User selects open  4. System displays “opening…”  5. System displays ticker symbols in new window  7. System displays ticker updates in new window  **Postconditions:** Portfolio is open for viewing/editing | **Use Case:** Save Portfolio  **Description:** System saves portfolio of ticker symbols  **Actors:** User  **Preconditions:** User previously has open a portfolio of ticker symbols  **Flow of Events:**  Use case starts when the user selects “save portfolio”   1. System prompts user for a name for the portfolio if none chosen already 2. User enters a name if needed 3. User selects save 4. System displays “saving...”   **Postconditions:** Portfolio is saved |
| **Use Case:** Import Portfolio  **Description:** System imports a portfolio from a text file  **Actors:** Users  **Preconditions:** User has a text file in the proper format  **Flow of Events:**  Use case starts when the user selects “import portfolio”  1. System prompts user for file path  2. User enters file path  3. User selects import  4. System displays “importing..”  5. System displays imported portfolio  **Postconditions:** Portfolio is imported for viewing/editing | **Use Case:** Export Portfolio  **Description:** System exports portfolio to a text file  **Actors:** Users  **Preconditions:** User previously saved a portfolio of ticker symbols  **Flow of Events:**  Use case starts when the user selects “export portfolio”  1. System prompts user for a filename to export to  2. User enters file name  3. User selects export  4. System displays “exporting...”  **Postconditions:** The portfolio is exported |
| **Use Case:** Add Ticker to Portfolio  **Description:** System opens a previously portfolio of ticker symbols  **Actors:** Users  **Preconditions:** User previously saved a portfolio of ticker symbols  **Flow of Events:**  Use case starts when the user selects “open portfolio”  1. System displays all previously saved portfolios  2. User selects one  3. User selects open  4. System displays “opening…”  5. System displays ticker symbols in new window  7. System displays ticker updates in new window  **Postconditions:** Portfolio is open for viewing/editing | **Use Case:** Remove Ticker from Portfolio  **Description:** System removes a ticker from the portfolio  **Actors:** Users  **Preconditions:** User has open a portfolio  **Flow of Events:**  Use case starts when the user selects a ticker  1. User opens ticker menu  2. User selects delete  3. System displays “deleting…”  **Postconditions:** Ticker is deleted |

**GRAPHICAL DISPLAY INTERFACE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| **Use Case:** Load Graph  **Description:** System outputs graphical representation of stock data.  **Actors:** User  **Preconditions:** User has Graphical Plugin installed  **Flow of Events:**  Use case starts when the user selects “Graph Stock”   1. System prompts name of stock to display 2. User enters specific stock ticker   **Postconditions:**  System Displays “General” or “Personal” | **Use Case:** Load General Graph  **Description:** System outputs general graphical representation of stock data.  **Actors:** User  **Preconditions:**   1. User has Graphical Plugin installed 2. User has selected “General Graph”   **Flow of Events:**  Use case starts when the user selects “General”   1. System displays graph of desired stock price over the last 8 months.   **Postconditions:** System displays graph of stock information. |
| **Use Case:** Load Personal Graph  **Description:** System outputs graphical representation of stock data contained within the currently opened portfolio  **Actors:** User  **Preconditions:**  1. User has Graphical Plugin installed  2. User has opened at least one portfolio  3. User has selected “Personal Graph”  **Flow of Events:**  Use case starts when the user selects “Personal”   1. System displays graph of desired stock price since adding the stock to their portfolio   **Postconditions:**  System displays graph of stock information. | **Use Case:** Refresh Graph  **Description:** System re-outputs graphical representation of stock data including recently added data  **Actors:** User  **Preconditions:**  1. User has Graphical Plugin installed  2. User has either a graph displayed  **Flow of Events:**  Use case starts when the user selects “Refresh”   1. System displays graph of desired stock price with extended time line up to most recently available data   **Postconditions:** System displays updated graph of stock information. |
| **Use Case:** Export Graph  **Description:** System saves a picture of the desired stock graph to a specified location  **Actors:** User  **Preconditions:**  1. User has Graphical Plugin installed  2. User has a graph displayed  **Flow of Events:**  Use case starts when the user selects “Export”   1. System displays “Export file to location” 2. User inputs file directory 3. System displays: “Saved graph in png format in desired directory”.   **Postconditions:** System displays graph is saved. |  |

**WORK BREAKDOWN STRUCTURE**

Underlined sections within development represent a portion of the project and their framework, each framework is developed by two - three of the team members while documenting how the framework will interact with individual components. Each individual component can be developed by a single person after main framework is created.

Conceptual Stage is already complete; Developmental stage begins after Elaboration(Conceptual and Elaboration are equivalent).

Odin Analytical Suite

**1.0 - Conceptual**

1.1 - Define Requirements

1.1.1 - Define Feed Handler Requirements

1.1.2 - Define Yggdrasil Requirements

1.1.3 - Define Plugins and Requirements

1.2 - Define Functionality

1.2.1 - Define Feed Handler Functionality

1.2.2 - Define Yggdrasil Functionality

1.2.3 - Define Plugins Functionality

1.3 - Develop Project Plan

**2.0 - Development**

2.1 - Feed Handler

2.1.1 - Fetch Stock Data

2.1.2 - Repackage Stock Data

2.1.3 - Send Stock Data to Yggdrasil

2.2 - Yggdrasil

2.2.1 - Receive Stock Data

2.2.2 - Subscription List

2.2.2.1 - Query Stock information

2.2.2.2 - Retrieve Subscription List

2.2.2.3 - Add Stock to Subscription List

2.2.2.4 - Remove Stock from Subscription List

2.2.3 - Plugin Manager

2.2.3.1 - Install Plugin

2.2.3.2 - Open Plugin

2.2.3.3 - Close Plugin

2.3 - Portfolio Display Interface

2.3.1 - Portfolio List

2.3.1.1 - Add Ticker

2.3.1.2 - Remove Ticker

2.3.1.3 - Load Portfolio

2.3.1.4 - Save Portfolio

2.3.2 - Export/Import Module

2.3.2.1 - Import Portfolio

2.3.2.2 - Export Portfolio

2.4 - Graphical Display Interface

2.4.1 - Load Graph

2.4.1.1 - Load General Graph

2.4.1.2 - Load Personal Graph

2.4.2 - Refresh Graph

2.4.3 - Export Graph

2.5 - Database Plugin

2.5.1 - Query Database

2.5.2 - Create New Database File

2.5.3 - Import Database

2.5.4 - Export Database

**3.0 - Roll Out**

**4.0 - Support**

**UPDATED PROJECT SCHEDULE**

The structure of this project will focus primarily around weekly iterations to complete weekly functionalities.

**October 6th** - Elaboration Ends

**October 14th** - Create a feed handler that is able to pull information from either Yahoo! or Google stocks API. The primary function of iteration is to pull and display data from the stocks API in real time (or close to) as well as send the information for use through Yggdrasil.

**October 20th** - Make a database and create a plugin that is able to take the information pulled from the stocks API feed handler and store the information for historical reference. This would entail mostly databasing structures to obtain the best methods of storing data for easy retrieval.

**October 27th** - Create a simple user interface that is able to access the database and pull stock information. For example, a user would be able to input a stock and time, then the program will output the ask prices and traded shares within the last 10 minutes.

**October 27th** - Stakeholder Product Review #1 - For the product review, the objective is to show the functionality of the feed handler. The demonstration will present the program pulling a stock’s information from either Yahoo! or Google stocks API, storing it in the database, then the user retrieving the information back from the database through a user interface command.

**November 3rd** - User profile plugin; Stores user information such as favorite stocks for ease of use by users, allows persistent data storage across all users/platforms.

**November 10th** - Second plugin; Simple stock prediction calculations with associated graphs output to the user graphically. Built off of existing interface plugin.

**November 17th** - Prediction Algorithm Plugin; Looks at information from different stocks and implements a user defined algorithm to predict a reaction from a corresponding or related stock. The objective is to just to be able to implement user defined functions to receive a corresponding graph that can be compared

**November 20th** - Stakeholder Product Review #2 - Presentable mock analytical plugins shown, and demonstrate interaction with the Yggdrasil core application. This portion of the project will be undertaken by each member of the group, with major testing of all relevant plugins and debugging as needed.

**Contribution Summary**

**CODY**

Cody Doyle contributed to many subsections of the Elaboration deliverable, mainly: Domain Model, Yggdrasil Use Cases, Domain Model. Cody also edited and helped clarify other sections for a more accurate Elaboration.

**TONG**

Tong has contributed to various sections of the Elaboration including work cases and supplemental specifications. Tong helped review the other sections of the Elaboration to make sure they were up to par with the standards and fulfilled the rubric requirements.

**CHRIS**

Chris Weir contributed to multiple sections of the Elaboration deliverable, including the Deployment Diagram, Use Cases, and many of the status reports, which were summarized below. Chris reviewed the document for accuracy and completeness.

**TONY**

Tony Simonutti contributed several subsections of the Elaboration deliverable, including but not limited to: Use Cases, Work Breakdown Structure, and Tony’s Contribution Summary. In addition to writing these sections Tony had a hand in editing each section and preparing the final draft of the deliverable.

**Status Report**

Since our Inception we have made a lot of headway toward making the Odin Analytical Suite. We have done a lot of planning before we delve into writing our code. We have created a domain model to look at all of the real world actors that are involved in the environment that our program will interact with. We have specified all of the hardware and software required for our program in our domain model. We have clearly decided all of the work that has to be done going forward and decided what priority these should have to be worked on. We have redesigned our schedule to fit our current thoughts on what needs to be done. We have thought about all the interaction the user will have with our product and designed use cases to represent these.