**\*|∅ INCEPTION**

Date: 09/22/2014

Team Name: \*|∅

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

Product Name: Yggdrasil

**VISION STATEMENT**

**EXECUTIVE SUMMARY**

Yggdrasil is a cost-efficient modular plugin based application, intended to fulfill the stock market data requirements of hedge funds and investment banks that want to build a customized solution for their individual needs. The core set of features of the Yggdrasil application include support for pre-built or custom stock feed handlers, databasing of feed history and analytical data, and support for analytic engine plugins.

**BUSINESS CASE**

Many of the current solutions on the market are very expensive and limit how much customization can be made without making changes to the application’s source code. By providing a plugin based application to consumers, they can create anything they may need.

Currently, two of the major players in the financial software industry are Bloomberg L.P. and Thomson Reuters Corporation. Both companies provide a finished product, which both have similar functionalities, but for a steep price. Bloomberg L.P.’s “Bloomberg Terminal” software costs at least $20000 [(source)](http://qz.com/84961/this-is-how-much-a-bloomberg-terminal-costs/) per person for one year, and Thomson Eikon ranges from $3600-$21000 per person for one year [(source)](http://blogs.cfainstitute.org/insideinvesting/2013/12/10/the-secret-revolution-in-finance/). Unfortunately for the companies purchasing this software, their developers who are not using the applications for trading are charged the same amount as traders. This leads to either very high development costs for customized solutions, or slower development speed due to the lack of availability of the software to the developers.

The Yggdrasil software is designed to be an alternative to the “Bloomberg Terminal” with the added bonus of having support for modular development. Our software may not necessarily include all the same benefits as the bloomberg terminal, but the modular design allows for users to tailor our product to their specific needs. The Yggdrasil product is designed to be the cheaper alternative marketed towards technology forward companies lacking the capital needed to use the current market standard. The main focus is the modularity, and constant support for our users and their needs.

**STAKEHOLDER DESCRIPTION**

According to the Wikipedia article for Bloomberg L.P., they hold approximately one third of the $16 billion financial data industry. With many smaller firms looking to get into the financial market, a lower cost, more developer friendly software suite would pose to gain a measurable portion of this market, and expand it to a wider audience. \*|∅ Software has the potential to increase profits significantly with our Yggdrasil platform, boosting dividends for investors. In addition, \*|∅ Software can offer services developing further plugins on a contract basis with various investment firms, boosting revenue.

**INTENDED FEATURES**

Yggdrasil will provide a modular platform, with the feature set only limited by the user’s imagination and skill. \*|∅ Software will also provide basic starter plugins so development teams can hit the ground running, and create useful customized plugins. Basic plugins will include a proof of concept feed handler, a historical databasing plugin, and example analytical plugins. Yggdrasil will also include basic documentation intended to help user’s build simple plugins easily.

**POTENTIAL RISKS**

The Yggdrasil platform has several risks associated with its design. One major risk is in the proof of concept feed handlers. As we do not have access to actual stock market feeds, we are using the Google Stocks API and Yahoo Finance API as makeshift feeds to prove that the feed handler concept is possible. An additional concern is the legality of the proof of concept feed handlers. Yahoo Finance and Google stocks APIs allow free use as long as the software using the API is not used to make money. However our program is only being used as a proof of concept, showing that our modular program can fetch feeds from stock feeders and manipulate that data.

**USER SCENARIOS**

**JOHN’S PERSONA**

John is an unmarried 45 year graduate of University of Pennsylvania’s Wharton School of Business with a business degree and economics minor. Now a small business owner taking in annual profits in excess of $800,000, John lives a comfortable life spending his leisure time camping and weekending at his lake house and fishing on his yacht. John originally worked at various large investment banks, and left Goldman Sachs after 12 years to start his own hedge fund. He also holds a large share of several small companies, and was fined $15,000 for violating a minor SEC regulation in his personal trading. He currently only has a few employees, and is looking to expand his business.

**JOHN’S USER INTERFACE**

John looks into a new, multi-featured, multi-platformed stock market analyzer that updates him with stock market information and trends both on his desktop PC at work and his tablet while he is sunbathing at home. Currently a small business owner, John does not have the resources or volume to participate in real time algorithmic trading. Luckily, John can choose the features he needs and opts out of the algorithmic trading software to save money. Nonetheless, as John’s business expands, the modular design of the stock market analyzer allows John to continue to purchase new modules he requires that can be seamlessly incorporated into the existing framework.

John wants to see how his company’s stocks in Chipotle are doing. He goes on his computer and pulls up Yggdrasil and select Chipotle. The screen displays the summary of all the data for Chipotle’s stock in the last hour. John sees that Chipotle’s stock rose 20% in the last 5 minutes. He uses his custom plug-in to predict if the stock will keep going up and it returns yes. Great, John has his traders double their investment in Chipotle’s stock. As the stock keeps going up, John watches happily as Yggdrasil shows his stocks predicted increase in profit.

**JOHN’S GOALS**

John’s has several goals when looking for a software solution, but they all feed into the same main desire; to increase the scope and reach of his company. Since licensing for professional grade stock market analyzers is extremely expensive, John would like to cut back on his licensing expenses while increasing the number of licences he has access to. John would also like to increase the number of traders he has doing trades at any given time, and by increasing the amount of licenses, he is able to increase traders without paying unnecessarily large monetary costs. John would also like more control over what his software does and outputs, the current software he uses forces his traders to work in a less than ideal environment. John would also like to increase the wealth of himself and his company, and he would like to gain more respect within industry. Finally John would like to increase the number of customers he is able to service at any given time. Each of these goals stems from having an inexpensive modular product to replace the expensive and archaic product he currently uses, by switching to the Yggdrasil base system he is able to accomplish his goals.

**JANE’S PERSONA**

Jane is a 37 year old married, working mother of two. She is currently the manager of a market data team for a software company which develops analytical and trading software for large banks. As a manager, Jane maintains a stable six figure yearly salary. Much of the software her team is working on requires a stock data feed from an exchange or third party provider in order to perform tests on their platforms. The cost of a developer subscription is outside her team’s budget, and she has been having trouble keeping developers on staff since the documentation available for feeds is poorly written and often incorrect. With all of these concerns, Jane leads a very high stress life. Last week she was unable to attend her daughter’s dance recital, as she was stuck in a meeting with Bloomberg’s development support team until 7pm.

**JANE’S USER INTERFACE**

Jane is now using Yggdrasil. One of her clients is interested in software that will use a new algorithm to predict stock information on the currently rising Chipotle stock. Jane and her employees use Yggdrasil’s API to make a plugin that used a NASDAQ feed to monitor a portfolio of stocks and predicts the outcome of Chipotle’s stock based on the research done by the company’s research analysts. Now using a custom algorithm that Jane’s quants team has developed, her client is now able to watch the stock and receive real-time information on bid-offer spreads, current orders, and liquidity of the stocks along with the prediction based. Jane’s clients learn that Chipotle’s stock value is inflated and should no longer invest in it as the price will likely fall out in the upcoming weeks.

**JANE’S GOALS**

Jane’s goals revolve around making her team able to develop new algorithms on the fly that can be conveniently implemented by her clients. In this scenario, her clients are the traders at her bank who implement the algorithms to find trading patterns. Because every second counts, it is Jane’s priority to minimize the time between release of information and the corresponding trading decisions. This means everything inbetween market data feed, algorithm development, and implementation is potentially unnecessarily wasted time. With a shortage of developers on the team, Jane’s main objective comes down to finding a cost efficient, simple method of allowing her team to develop algorithms that can be easily and quickly implemented by the traders in terms of cross-compatibility and flexibility. By using Yggdrasil’s affordable plugins, Jane was able to achieve her goal by providing her team with a simple interface for designing algorithms based on stock market feeds that could be easily transferred and implemented for the traders due to the compatibility and simplicity of the Yggdrasil platform.

**PROJECT SCHEDULE**

**COMPLETE DATES**

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

October 6th - Elaboration Ends

October 14 - 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 20 - 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 27 - 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 27 - 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 3 - 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 10 - Second plugin; Simple stock prediction calculations with associated graphs output to the user graphically. Built off of existing interface plugin.

November 17 - 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**

**CHRIS**

Chris Weir contributed several sections, such as the Executive Summary, Business Case, Stakeholder Description, Intended Features, John’s persona and goals, and Jane’s persona and goals. In addition to writing Chris provided guidance and vision to the team in terms of project planning and goal setting.

**CODY**

Cody Doyle contributed to many subsections of the Inception deliverable, mainly: John’s and Jane’s user scenarios and the schedule. Cody also edited and helped clarify other sections for a more accurate inception.

**TONG**

Tong was involved with general decision making and setting the scopes of the project. He has contributed to various components of the Inception deliverables, most notably the user personas, scenarios, and goals for John and Jane. Tong was involved with the final rendition of the Inception to make sure all the criteria of the Inception rubric were filled.

**TONY**

Tony Simonutti contributed several subsections of the Inception deliverable, including but not limited to: John’s Goals, Intended Features, Potential Risks, Business Case, Stakeholder's Description, and his own 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**

All or Nothing started as a group of four computer science students with little knowledge of software development design and probably even less about documentation. Yet, a shield was created to remind us of our skills, weaknesses, and ultimately our dedication to learning the techniques to proper software development and apply that knowledge to our own project. By September 8th, the end of the first week, we have already begun setting up the foundations of organizing the project with tools such as Trello, Google Groups, and an SVN repository. Our project included risks such as general unfamiliarity with market concepts, Yahoo/Google stocks API, and Subversion. Using our designated, weekly meeting time, we helped each other setup Subversion folders and discussed the stock market as well as our project’s scope, thus mitigating our risks.

Over the course of the next few weeks, we set the vision for our project, Yggdrasil, and clarified the objectives and outcomes. User scenarios and personas were created to help us better understand how we want our project to function as a product. Coding standards defined, UML tool chosen, our team now has just about all the tools necessary to form a strong foundation and begin working on Yggdrasil.