Version 0.01

Document history

Date	Version	Description	Author
	0.01	Initial version	NAI

Distribution

Name	0.01													

Document approval

Nathan Goedeke	Erik Phillips
Devin Hill	Harryson Tun
…………………	

Dr. George Dimitoglou, Comupter Science Department of Hood College

………………….

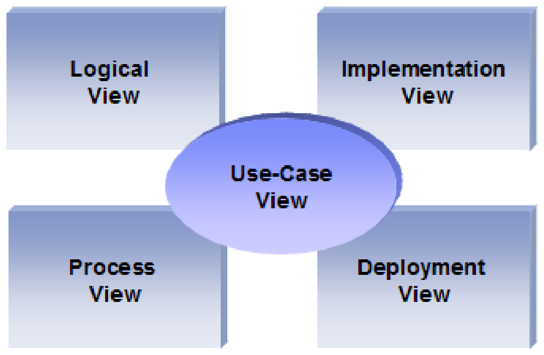
Contents

1. *Introduction*

1.1 Purpose of this document

The Software Architecture Document (SAD) contains the architectural description of Project Greed developed by New Age Infrastructures (NAI) consisting of developers Nathan Goedeke, Devin Hill, Harryson Tun and Erik Phillips. This description consists of various architectural views of the system, in order to highlight the different aspects of it.

The description makes use of the well-known 4+1 view model.



The 4+1 view model enables various stakeholders (client) to establish the impact of the chosen architecture from their own perspective. The Process View (communication of processes) is not a separate chapter but can be found in the chapters 3.3 and 5.

1.2 References

Title	Version	Author	Location
Vision			
Product Acceptance Plan			
Use Case Model			
SOLA Software Architecture Document	1.1	Andrew McDonald	https://blackboard.hood.edu/bbcswebdav/pid-123879-dt-content-rid-325964_1/courses/CS_324_01_LEC_FALL_2014/sola_software_architecture_document_v1.1_0.pdf

1.3 Document Overview

Chapter	Reader	Objective
Architectural	Software Architect	Overview of architecturally relevant requirements.
3 Logical View	Developer	Knowledge of the application's conceptual structure, as a basis for technical designs.
4 Implementation View	Developer	Knowledge of the application's technical structure.
5 Deployment View	System Administrator roles	Knowledge of the way in which the application is deployed and (internal and external) communication takes place.

2. ***Architectural requirements***

2.1 Non-functional requirements

Source	Name	Architectural relevance	Addressed in:
Dimitoglou	Web accessible	Project Greed is only web accessible. There is no option for mobile device accessibility.	3.1.3

Dimitoglou	Login/sign-up	The login engine allows users to create an account which will allow access to competitions and transactions. Project Greed will provide a GUI system that will take user login information via input and store it to the login repository. This will generate an instance of an account with a portfolio automatically provided.	3.1.1
Dimitoglou	Portfolio management	The portfolio engine will allow users to create multiple portfolios with which they can use to enter and create competitions. There is one main portfolio the user has to make transactions. This portfolio can also hold many portfolios which can be used in competitions. Project Greed provides a portfolio repository system that will store all information from the portfolio engine.	3.1.4
Dimitoglou	Search engine	This engine will allow users to search the various data bases for requested information.	3.1.1
Dimitoglou	Competition engine	This engine allows users to use a portfolio to create, participate in, or delete competitions within the simulated environment.	3.1.1
Dimitoglou	What if simulator engine	This engine allows users to create different scenarios based on stored historical data.	3.1.1
Dimitoglou	API generated stock index	This is an imported real time updated web graphic which lists the current value of stocks being bought and sold.	3.1.4
Dimitoglou	Ticker	The ticker is a GUI that provides relevant information to the user via a banner on the web page.	3.1.1
Dimitoglou	News-feed	The news-feed is a GUI which provides relevant financial news and other information not found in the ticker.	3.1.1
Dimitoglou	Repository	This is a module which takes information from it's corresponding engine and stores the information as well as making that information available to other sources within the simulated environment.	3.1.4

2.2 Use Case View (functional requirements)

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

Source	**Name**	**Architectural relevance**	**Addressed in:**
3.3.1	New User	Creates an new account with unique username and password, and a new portfolio with unique number.	

3.3.2	Buying Stock	Adds stock resources to currently-active portfolio and deducts the corresponding amount from remaining capital. Accessed via transaction engine in the portfolio view.	3.3.1
3.3.3	Selling Stock	Removes stock resources from currently-active portfolio and adds the corresponding amount to the current capital. Accessed via transaction engine in the portfolio view.	3.3.2
3.3.4	Create Competition	Creates a new competition with desired parameters such as duration and entrance fee. Accessed via registration engine on the competition page.	3.3.3
3.3.5	Join Competition	Generates a virtual portfolio with duplicate stock resources and capital. Deducts required fee from both. At the termination of competition, fee is refunded back to the original portfolio and the virtual instance is deleted.	3.3.3
3.3.6	What If? Simulator	Enters past transaction information and calculates how much would have been gained or lost (from currently-active portfolio) if this transaction had taken place.	3.3.4

3. *Logical View*

This section describes the architecturally-significant parts of the design.

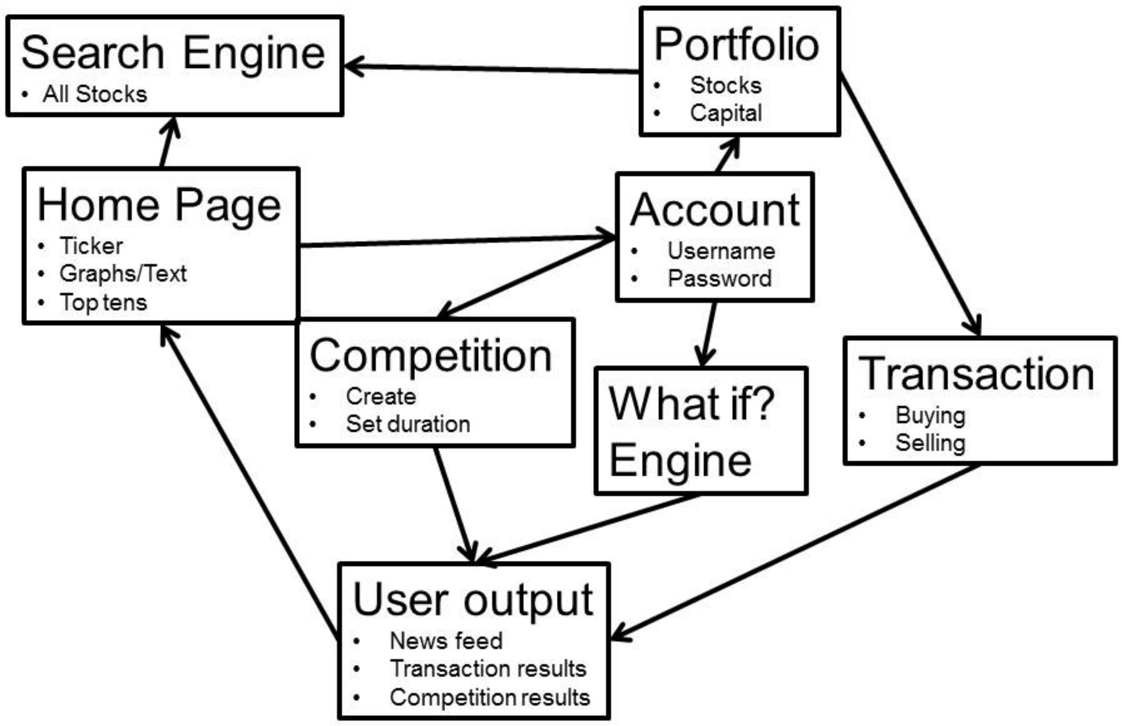
3.1 Tiers

The layers or tiers that make up this project are as follows:

- Presentation
- Service
- Domain
- Data

3.1.1 Presentation

The Presentation Layer encapsulates the components that end users will interact with, including account settings, portfolio and competition management.



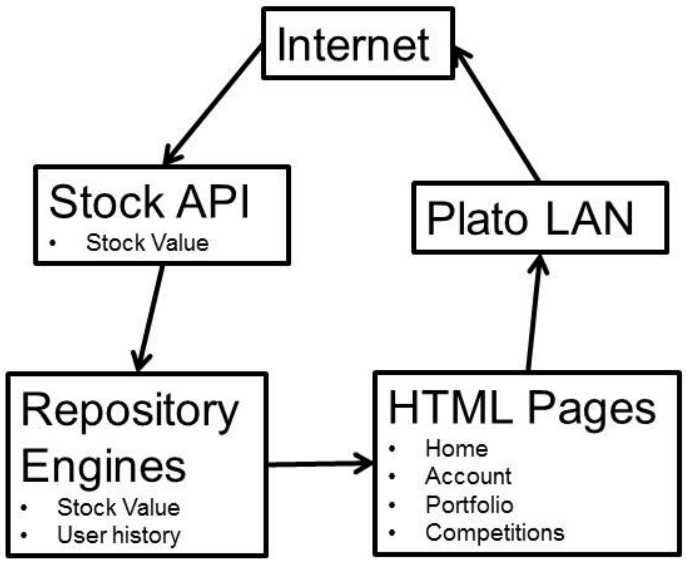
3.1.2 Service

Project Greed implements a web-based virtual financial service using a MySQL database, webpage designing languages such as PHP, HTML5, and JavaScript, algorithmic languages such as C++ and Java. Additionally, this Project will use the LAN protocols on Pluto.

Project Greed will be available across all operating systems and all platforms except for mobile devices. The individual modules will act independently and will be reusable in other projects.

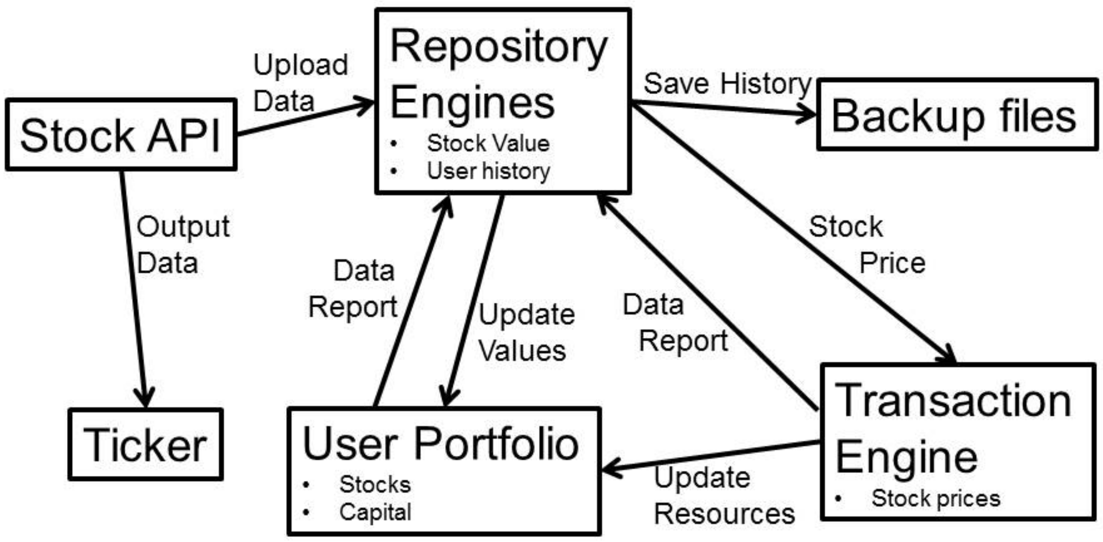
3.1.3 Domain

This project will be available on the internet via the Pluto server and can be accessed through any web browser.



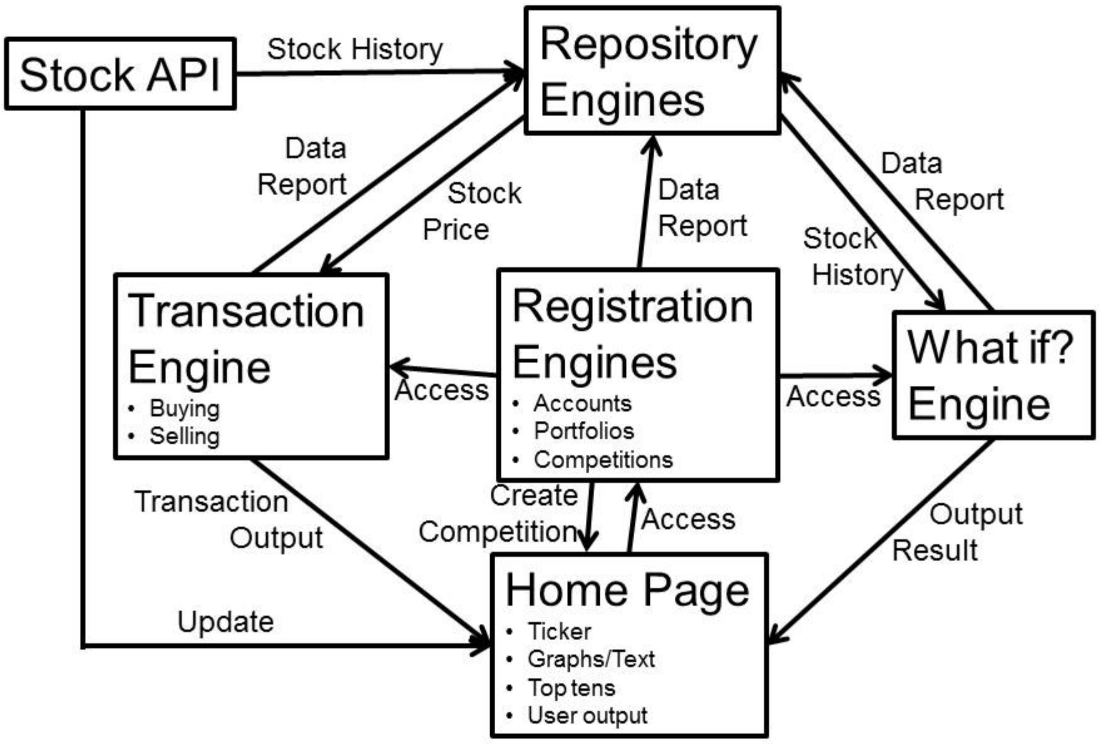
3.1.4 Data

Data gathered from user input and API will be maintained within several databases managed using MySQL, for example, individual module repositories.



Stock API
Upload Data
Repository Engines
• Stock Value
• User history
Save History
Backup files
Output Data
Ticker
Data Report
Update Values
Data Report
Stock Price
User Portfolio
• Stocks
• Capital
Update Resources
Transaction Engine
• Stock prices

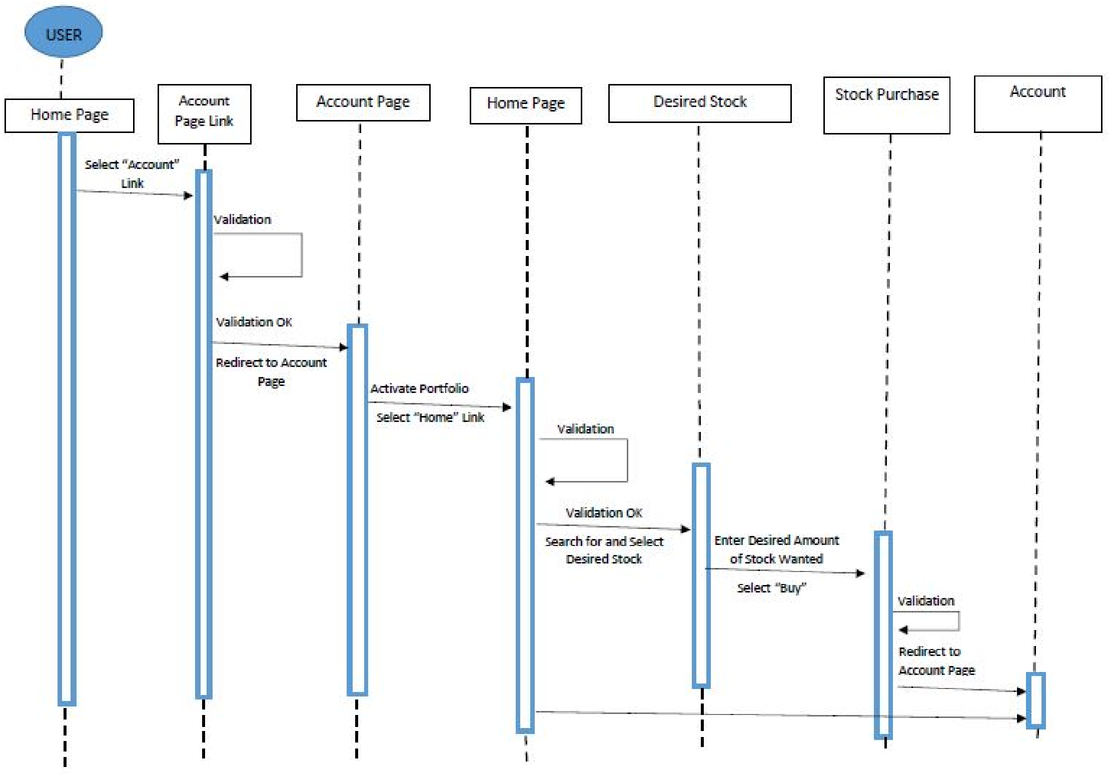
3.2 Subsystems



3.3 Use Case Realizations

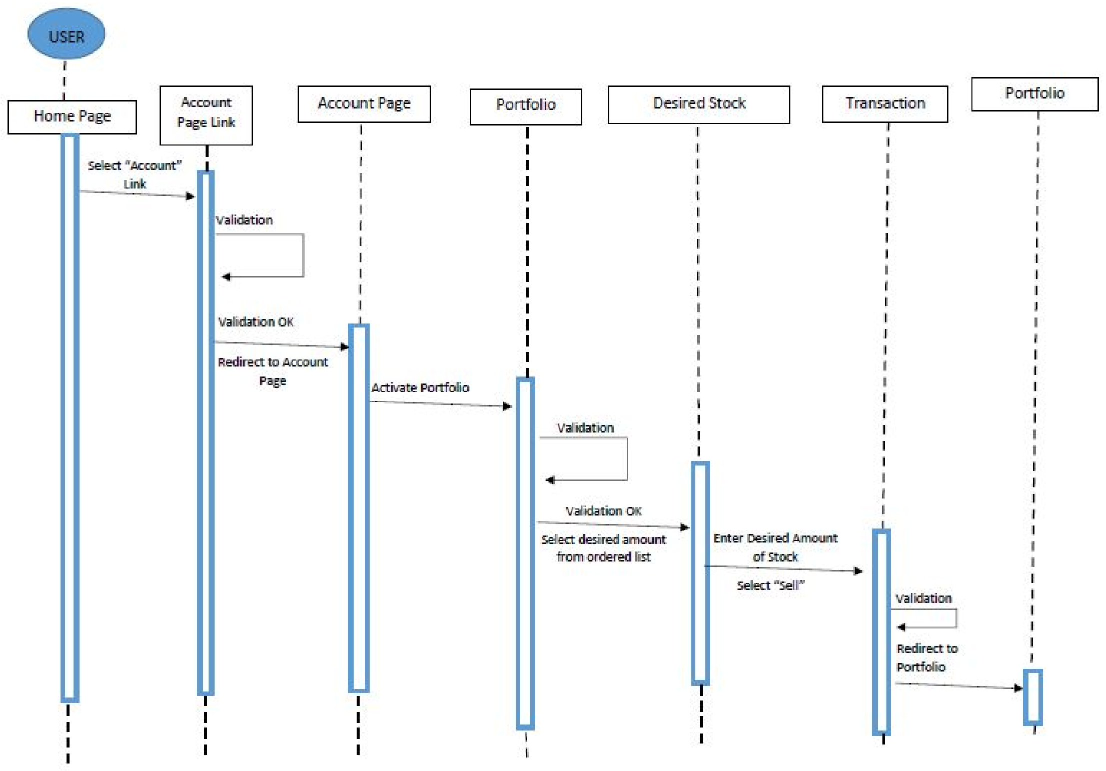
3.3.1 Buying Stock

The user accesses his/her account page, then activates a portfolio. At that point, the user may search for stock on the home page, enter desired amount, and select “buy”. The corresponding cost will be deducted from the portfolio’s current capital, and the user is redirected to the account page.



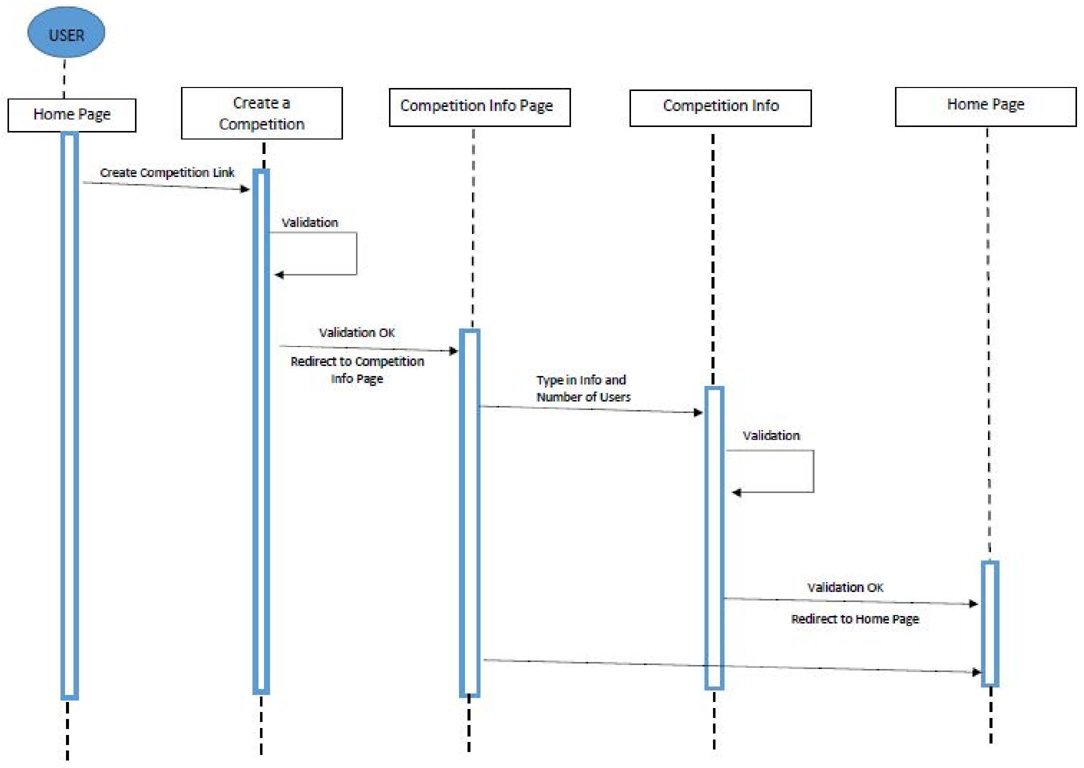
3.3.2 Selling Stock

The user accesses home page and opens the currently-active portfolio. He or she may then select the desired stock from the ordered list, enter the quantity, and select “sell”. The corresponding value of the stock is then returned to the capital, and the user is redirected to the portfolio view.



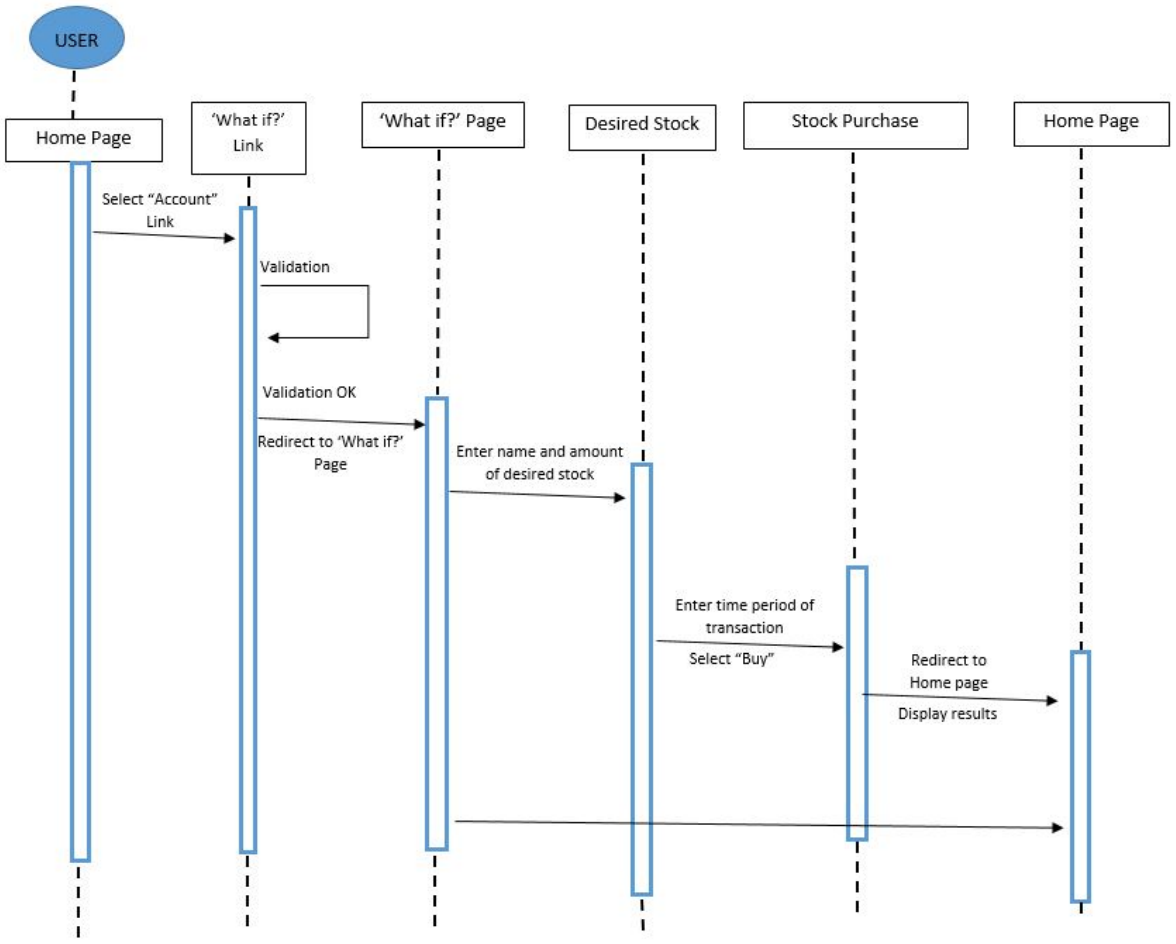
3.3.3 Create Competition

The user accesses the competition page and has the ability to enter necessary information for a competition. The user is then redirected to the home page.



3.3.4 What if? Simulator

The user accesses the 'what if?' page and has the ability to enter past stock information for transaction. The user is then shown what his/her net result would be if that transaction occurred at that time.



4. *Implementation View*

4.1 Structure of the packages

Project Greed will be presented in two deliverable packages.

4.1.1 The first package will include a semi-functional website with a complete authentication and user-management system. It will also include a static database that implements a few, simple, predefined transactions. It will be presented as a visual demonstration with a written account of our current progress.

4.1.2 The second package will be the final implementation. It will include a fully-functional website with complete user and portfolio management functions. It will also include a dynamic, self-updating database that utilizes real-time API information. Additionally, this package will include an installation guide, user guide, programmer's guide, source code documentation, and a final report.

4.2 Realization of tiers

4.2.1 Registration information

Project Greed will utilize three main objects: Accounts, Portfolios, and Competitions (See section 3.4 in SRS). All three of these will be created by user specifications via registration engines, and stored in a database system via repository engines. An account is created first and foremost by the user from the homepage. Once this is done, the user can then access his/her account page. It's from the account page that the user can generate portfolios and competitions; however, a new competition requires the user to have at least one portfolio, and will automatically be associated with the currently-active portfolio. Each portfolio automatically comes with $100K as capital, and no stock resources.

4.2.2 Stock information

Another object that Project Greed will utilize is Stock. Unlike the other three objects, stock is generated automatically from online API data on an hourly basis. This data is inserted into a database via the repository engines. Each stock carries an attribute that designates what user (if any) possesses that resource. If a user buys stock, that attribute is updated via the transaction engine (simultaneously updating the portfolio's current capital). The transaction engine is accessed by a user from the portfolio view on his/her account page. If the user enters a competition, a virtual portfolio is generated with a copy of the user's current capital and stock resources. This portfolio is deleted upon the termination of the competition, and the entrance fee is repaid to his/her current capital. Additionally, the user can access a What If? simulator engine from the account page. On this page, the user can enter stock information from some time in the past and compute what the net gain or loss would have been if he/she bought that stock at that time. All user output, news feed, competition results, and top ten stock lists are displayed on the home page by accessing respective information from the repository engines. Graphs and text files displaying the current status of each stock is accessed using a search engine from the home page.

4.3 (Re)use of components and frameworks

The only third-party component implemented in this product will be the interface with the online API information. This interface will download real-time stock information automatically, and distribute it to the appropriate databases.

5. *Deployment View*

This section describes the physical configuration on which the software is deployed and run. It shows the physical and virtual nodes that execute or interface to the system and their interconnections. In addition to the illustrating the physical nodes used for deployment, the deployment diagram that follow also illustrate the key communication paths between the various components along with the relevant communication protocols.

Name	Type	Description
Computer	Physical	User display and input device
HTTP	Virtual	Internet protocol
LAN	Physical	Pluto campus secure network
Point-to-point	Virtual	Authentication and other information transfer. Uses TCP/IP protocol to connect to Pluto.