Project Greed

Software Requirements Specification

Version 1.0

October 3, 2014

New Age Infrastructures (NAI)
Erik Phillips, Nathan Goedeke, Devin Hill, Harryson Tun

Prepared for
Principles of Software Engineering
Fall 2014

• Revision History

Date	Description	Author	Comments
10/3/14	Version 1.0	NAI	Original Document

• Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of a web-based stock market simulator. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to user input. This document is intended for both the stakeholders and the developers of the system and will be proposed to Dr. Dimitiglou{ XE "Historical Society" } for its approval.

1.2 Scope

This software system will be a web-based stock market simulator for a team project, specifically, for CS 324 Principles of Software Engineering at Hood College in Frederick, Maryland. This system will be designed to simulate a real web site, which allows users to purchase/sell stocks and similar commodities, participate in competitions against other users, and monitor real time stock market activity, while remaining easy to understand and use. More specifically, this system is designed to allow a user to create a profile and act as his or her own stockbroker through the management of his or her own financial portfolio. This system will also allow users to monitor real time stock market activity by providing the user with continuous real time updates throughout the day. Furthermore this system will allow the user to either create a competition or simply join a competition. Finally, this software system will provide the user with relevant and current financial news via a news-feed ticker.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Software Requirements Specification (SRS)	*A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.*
User Interface	*Any external input device that interacts with the product.*
Engine	*Self-contained software module within the project that is modified by either a user or another engine.*
Repository	*A module that contains and returns historical information and handles error procedures.*
Portfolio	*Personal financial and stock information.*
TCP/IP Protocol	*Internet communication protocol*
HTTP Protocol	*Specific internet destination*
API	*Application programming interface, i.e. any external programs essential for running the website.*
Ticker	*Published stock information on a banner*
News-feed	*Miscellaneous published information provided to the user*

Dump file	*Information pulled from immediate user interaction for maintenance purposes.*
Input	*Any user interaction with the website, or information provided by the user.*

1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

"Software Requirements Specification." *Www.cse.msu.edu*. Michigan State University, 15 Apr. 2004. Web. 9 Oct. 2014.

1.5 Overview

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter. The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product. Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language. The fourth chapter shows the client diagrams of the work flow of the project, how each subsystem interact with one another and, the stages of development for the project from cradle to grave.

2. General Description

2.1 Product Perspective

This project is a new, self-contained, and stand-alone product. It is intended to simulate stock market scenarios that may be seen in similar web-based environments, i.e. J. P. Morgan and Chase Bank.

2.2 Product Functions

Refer to section 3.3 for specific product functionality.

2.3 User Characteristics

The User{ XE "Reader" } is expected to be reasonably Internet literate, be able understand how to create an account using simple site generated prompts, be able to use a search engine and have a reasonable understanding of the stock market. The home page of this website will have a search function and link to create a user(s) account.

2.4 General Constraints

This product is a simulation. It is not intended for commercial use, and all resources are virtual. This product is strictly web-based, so an internet connection is required. It requires a server with sufficient memory to run simulations. The Pluto environment will provide only minimal security to run secure scripts, so only generic information will be entered for simulation purposes. The individual engines must be interchangeable for reusability purposes.

2.5 Assumptions and Dependencies

- The user is internet literate
- The user can create an account
- Hourly stock information is available and accurate
- Users have direct control of their own resources within the simulated environment (i.e., the simulation requires external user interaction)

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- Web accessible
- Log in/ Sign-up
- Search engine
- Portfolio management
    - Create/Delete portfolios
    - Compare portfolios with other users
    - Portfolio transactions
        - Buying/selling stocks
        - What if? Simulator
    - Activate portfolios
- Competition interaction (management/sign-up)
    - Create competitions
        - Set duration/time-line
        - Max/min number of participants
    - Competition rules
        - Winner determined by greatest financial gain
        - Initial capital is set (constant)
        - Quit competition
    - Competition results
        - Check current status throughout competition
    - Delete competition

3.1.2 Hardware Interfaces

- Web based only (no mobile devices)
- Information transferred using TCP/IP and HTTP protocols.
- Accessible across all operating systems

3.1.3 Software Interfaces

- API generated graphs/charts
- Ticker (as a banner)
  - Hourly stock information
- News-feed
  - Top tens
  - Open competitions
  - Competition results

3.1.4 Communications Interfaces

- Available through social networks

3.2 Functional Requirements

3.2.1 Repository engine

3.2.1.1 Introduction

- Stores historical information and dump files
- Sends data at later points to other engines as requested, such as the ticker/newsfeed, transaction engine, and ‘what if?’ engine
- Any sort of data corruption, such as memory leaks and dangling pointers, will return error and dump file memory in order to prevent further damage

3.2.1.2 Inputs

- Gather information from corresponding engine

3.2.1.3 Processing

- Save information

3.2.1.4 Outputs

- Creates back-up file and stores information to it
- Send results to ticker/news-feed (if necessary), and to individual stocks

3.2.1.5 Error Handling

- In case of corrupt data collection (i.e. memory leaks, dangling pointers/modifiers), return error and dump file

3.2.2 Registration (with competition, portfolio, user) engine

3.2.2.1 Introduction

- Allows user input
- Stores input to create an account, competitions and portfolios

3.2.2.2 Inputs

- User-generated input

3.2.2.3 Processing

3.2.2.4 Outputs

- Sends information to respective repository

3.2.2.5 Error Handling

- Incorrect user input returns error

3.2.3 Search engine

3.2.3.1 Introduction
- Allows users to search for stock entries

3.2.3.2 Inputs
- User-generated input

3.2.3.3 Processing
- Search for entry

3.2.3.4 Outputs
- Returns list of stocks within search parameters

3.2.3.5 Error Handling
- Incorrect user input returns error

3.2.4 Transaction engine

3.2.4.1 Introduction
- Allows users to exchange stocks for money

3.2.4.2 Inputs
- User-generated input

3.2.4.3 Processing
- Buys and sells stock for portfolio capital

3.2.4.4 Outputs
- Changes stock ownership and updates capital
- Sends information to respective repository

3.2.4.5 Error Handling
- Incorrect user input returns error

3.2.5 'What if?' engine

3.2.5.1 Introduction
- Allows users to calculate what profit/losses could have occurred if he had purchased stock at some time in the past.

3.2.5.2 Inputs
- User-generated input

3.2.5.3 Processing
- Gathers information from repository as per user request
- Calculate hypothetical profit/losses

3.2.5.4 Outputs
- Sends information to respective repository
- Displays results in a GUI

3.2.5.5 Error Handling
- Incorrect user input returns error

3.3 Use Cases

3.3.1 New user

1. Selects "login/sign-up" on home page
2. Enters username and password
3. Selects "my account" to visit account page

4. Selects “new portfolio”
5. A new instance of portfolio is generated and appended to his account. The initial capital is $100K.

3.3.2 Buying stock

1. Accesses account page
2. Selects desired portfolio
3. Selects “activate”
4. Return to homepage
5. Search for desired stock symbol
6. Enter the desired amount of stock and select “buy”
7. Return to portfolio to view remaining capital

3.3.3 Selling stock

1. Access account page
2. Select desired portfolio
3. Activate if not already
4. Select desired stock from the ordered list
5. Enter the desired amount of stock and select “sell”
6. Return to portfolio to view current capital

3.3.4 Creating a competition*

1. Access competition page
2. Select “create”
3. Enter desired information
4. Return to home page to view competition status

3.3.5 Joining a competition*

1. Select competition from homepage news feed
2. Select “join”
3. Return to homepage to view competition status

3.3.6 Deleting a defunct competition

1. Select the competition from the homepage news feed
2. Select “delete”
3. You may only delete a competition that you created

*** Note: Competitions are automatically associated with the currently active portfolio. If you do not want your portfolio in the competition, do not activate it.**

3.4 Classes / Objects

3.4.1 Account

3.4.1.1 Attributes

- Username
- Password
- Portfolios
- Competitions started

- Competitions participating

3.4.1.2 Functions

- Accesses registration functions

3.4.2 Portfolio

3.4.2.1 Attributes

- Number
- Activation status (Boolean)
- Stocks
- Total capital

3.4.2.2 Functions

- Accesses registration functions
- Self-deletion

3.4.3 Stock

3.4.3.1 Attributes

- Company symbol
- Cost

3.4.3.2 Functions

- Used by registration functions
- Accesses repository function

3.4.4 Competition

3.4.4.1 Attributes

- User created
- Users enrolled
- Opening status (Boolean)
- Duration
- Winner status

3.4.4.2 Functions

- Accesses registration functions
- Self-deletion

3.5 Non-Functional Requirements

3.5.1 Performance

- All transactions should be processed in less than a second

3.5.2 Reliability

- Should not experience a technical error more than thrice a week
- User errors should be clearly defined

3.5.3 Availability

- Web-accessible only (not mobile)

3.5.4 Security

- None

3.5.5 Maintainability

- Updated automatically every hour
- Checked for errors once a week

3.5.6 Portability

- None

3.6 Inverse Requirements

- Cannot exchange stocks for real money
- The amount of buying and selling does not affect the stock's price
- Cannot delete an account or a stock

3.7 Design Constraints

- Framework implemented in HTML
- GUI interface generated using JavaScript and PHP
- Database created in MySQL
- Miscellaneous logic (repositories, etc.) implemented in C++

3.8 Logical Database Requirements

A database is needed to keep track of stock information, user portfolios, and user accounts. Every user has a set of portfolios, which in turn has a set of stocks. Each company has a number of stocks that can be bought by users.

4. Analysis Models

4.1 Data Flow Diagrams (DFD)

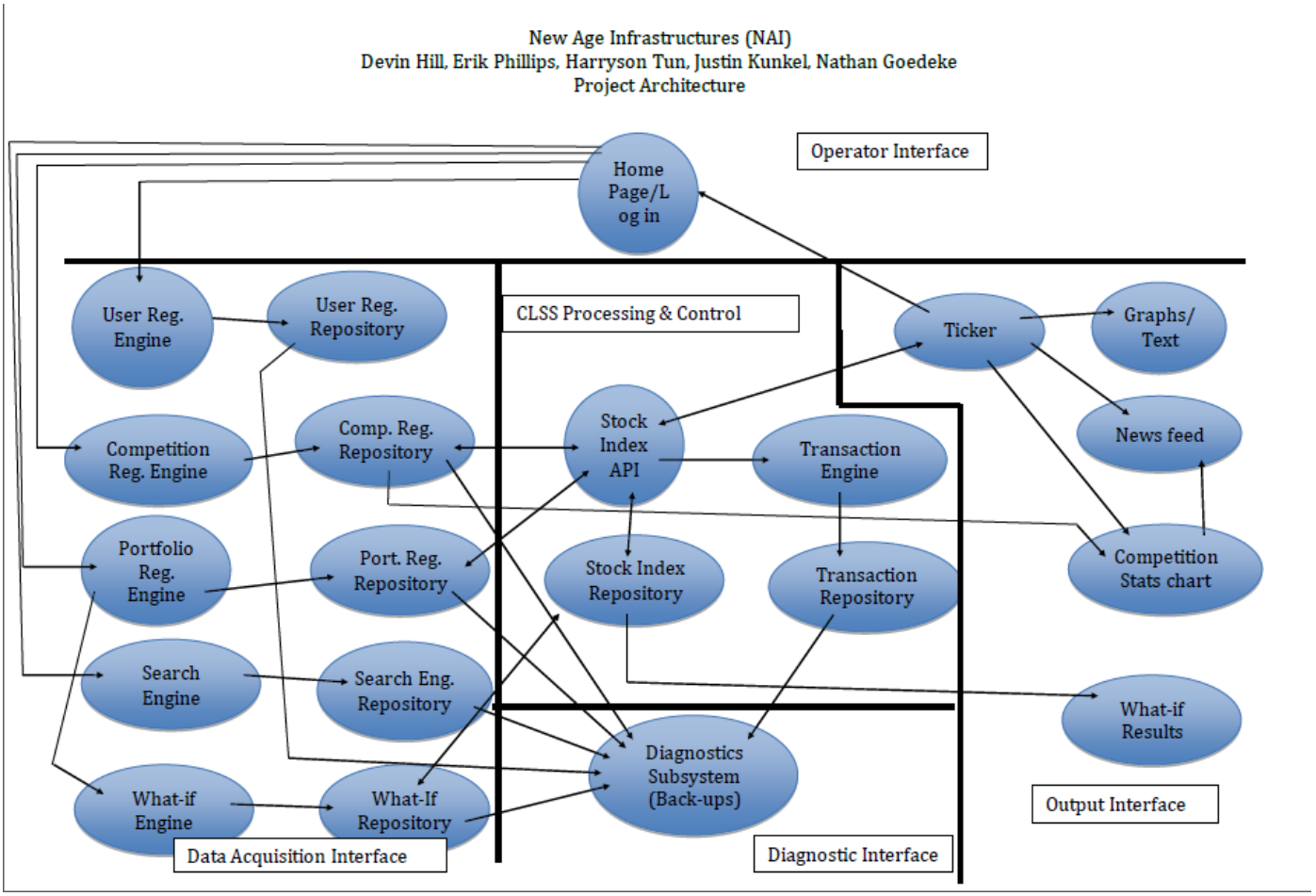

Figure 1: Architectural design and data flow for Project Greed

4.2 State-Transition Diagrams (STD)

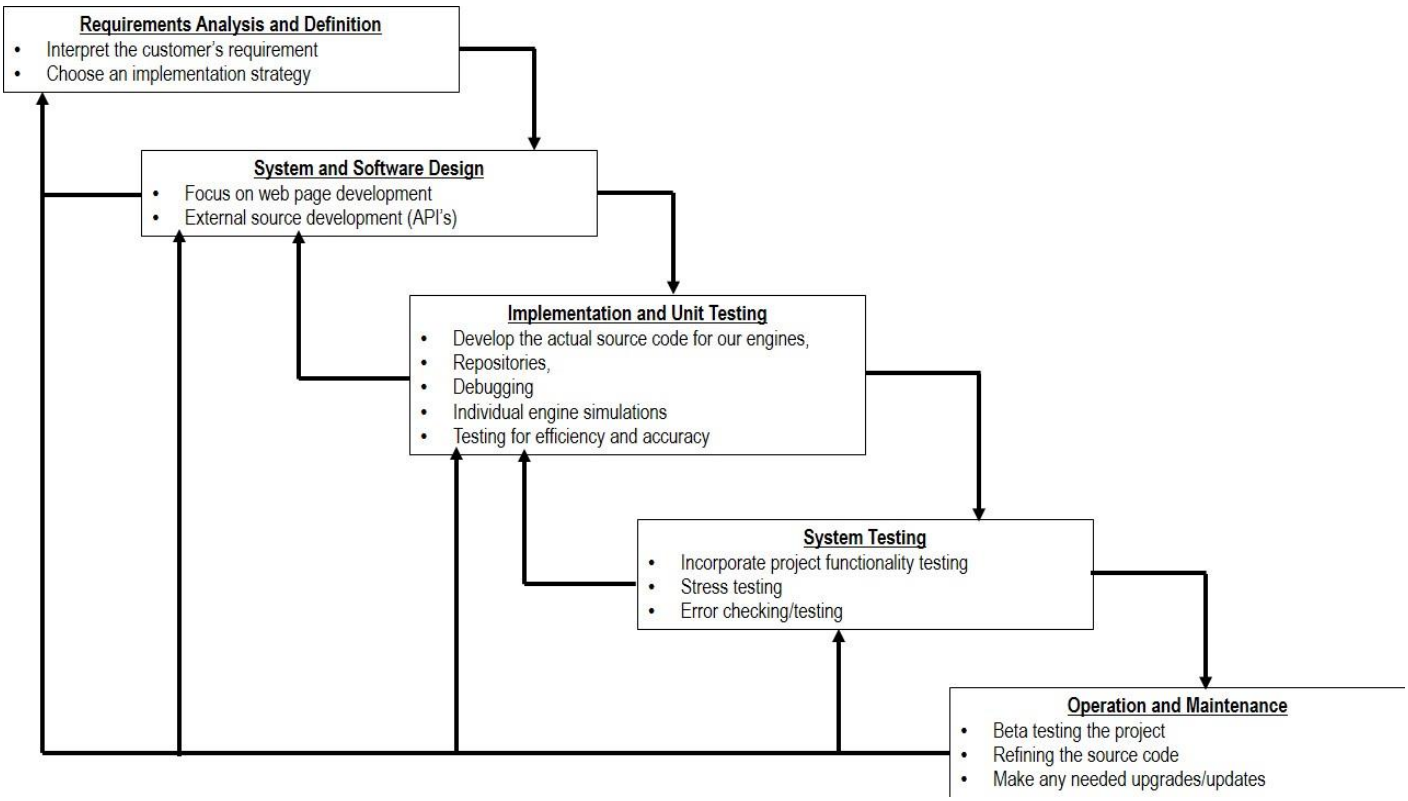

Figure 2: The extended Waterfall Model as applied to Project Greed

5. Change Management Process

Any alteration to this document shall be submitted by the liaison after unanimous agreement from project members, and attaining approval from client.