**Requirements Analysis Document**

Introduction

Stockestry is a stock market game that pulls stock information off of the Australian Stock Exchange every hour simulating the experience of a real stock broker. Players are given a lump sum of 1 million dollars to aid in their initial investments and are ranked based off of their profit made.

Purpose of the System

The system is made to mimic the Australian Stock Exchange; providing the correct stock price information on an hourly basis to give players a realistic experience.

Scope of the System

The system is set to be completed in 3 months; full functionality and a deliverable product to be ready by the deadline. A database in which all information is stored must be readily prepared to store relevant information in regards to users and stock information. The user-interface is to be kept simple and clean allowing users to maneuver around the website with ease. The website must also have a responsive design to suit any screen size; whether it be a computer monitor or a smartphone screen.

Proposed System

Functional Requirements

The current system is currently fully functional; the stock information is reliably being pulled by the API and users are able to buy and sell stocks for any company listed with the Australian Stock Exchange. After buying or selling a stock, the transaction history for the specific user will be updated. The profits made from buying and selling stocks is used to rank the user against everyone else and is reflected on the leaderboard. Users are able to place a company under their watch list so they can monitor the price of a company’s stocks and receive an alert when the price changes according to the percentage the user has specified. User’s are able to add other users to their friends list as well as communicate with them through the website in turn leaving a notification when they do receive a new message. Admins are able to view the portfolios of every user as well as ban them if there is sufficient reason to do so. Admins are also allowed to delete stocks from specific users and view their transaction history.

Nonfunctional Requirements

When a user is first directed to the website, they are taken to the register/login pages. The appropriate fields must be under a certain format (email having an @ sign followed by a .com). Once logged in, the user can navigate through each page via the navbar on the side of the screen. When the pages are viewed on a lower resolution, the elements of the page will shrink and adapt accordingly to suit the viewer's needs.

**Software Stack**

For the system we used Laravel as our framework with PHP Storm as our IDE (Integrated Development Environment). The plugins used within the framework were Excel and Carbon. We at first used SQLite as our original database to manage the database locally and swapped to MySQL when we moved the project online because the host of our website could not use SQLite. Amazon Web Servers was used to host the website. The API used to pull the relevant stock information is the Yahoo Finance API.

**Architecture Design**

For the architecture design we used a clean laravel layout ( which followed the MVC architectural pattern) to organise our models, views and controllers. Everything in Laravel has a designated section ie; routes are together as well as pages always being in resources/views. We used a master blade template for all of our page layouts which made creating new pages simpler where in which we just change the page to suit the need.