Software Requirements Specification

for

InvestiMapp

**Version 1.3**

**Prepared by Team InvestiMapp**

**April 14th, 2018**

**Table of Contents**

[**Revision History**](#_x33wp8gkz48x) **3**

[**Introduction**](#_g296inlmrh13) **4**

[**Purpose**](#_3znysh7) **4**

[**Intended Audience and Reading Suggestions**](#_cpm6vqhc4hod) **4**

[**Product Scope**](#_s9laor9cnkhc) **4**

[**References**](#_erzmnx15xz5e) **5**

[**Overall Description**](#_ds304dd9adgv) **5**

[**Product Perspective**](#_54m0hfc2ocjx) **5**

[**Product Functions**](#_m7wgk2zcc8p0) **6**

[**User Classes and Characteristics**](#_co2d52oik0wz) **6**

[**Operating Environment**](#_1pyrmynvdnyy) **7**

[**Design and Implementation Constraints**](#_69h8itgu9t00) **7**

[**User Documentation**](#_qnqop13ws94j) **7**

[**Assumptions and Dependencies**](#_wmgk05k0rk3v) **7**

[**External Interface Requirements**](#_1uv3gqnf2sm0) **8**

[**User Interfaces**](#_827j90j7g97s) **8**

[**Hardware Interfaces**](#_fikwiybl0cl4) **8**

[**Software Interfaces**](#_xfbw6orzupyy) **8**

[**System Features**](#_cf5x82a3tlcf) **8**

[**Account Creation and Login**](#_li5jyccpc3b1) **8**

[**Stock Data Manipulation**](#_np363mj8k32q) **9**

[**Dashboard Updates**](#_tx5q3x21tqbs) **10**

[**Dashboard Graphs**](#_l7uoxf5fem2v) **11**

[**Notifications**](#_1khdx0qj73cm) **11**

[**Other Nonfunctional Requirements**](#_8otsignfb2n5) **12**

[**User Requirements**](#_73vm1xk2xf22) **12**

[**Performance Requirements**](#_fphjqdp2jyd5) **12**

[**Safety Requirements**](#_2rk3ogn1o3sw) **13**

[**Security Requirements**](#_341e5vke56bx) **13**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Darrel Donald | 2/12/2018 | Initial Document Creation | 0.1 |
| Ryan Hutchinson | 2/13/2018 | Additions to all sections | 0.2 |
| Ryan Hutchinson | 2/15/2018 | First Edition | 1.0 |
| Ryan Hutchinson | 3/8/2018 | Adding UML Use Case Diagrams | 1.1 |
| Ryan Hutchinson | 4/10/2018 | Fixing Past Mistakes | 1.2 |
| Ryan Hutchinson | 4/14/2018 | Privacy and Security (Auth0) | 1.3 |

# Introduction

## Purpose

InvesitMapp is an investment tracking web application. InvestiMapp will focus on the user’s bottom line. All of the user interface will be based on what the user paid for the the stock and all success or failure elements will be based on this.

## Intended Audience and Reading Suggestions

This document is meant for:

1. Developers of InvestiMapp
   1. Reading Sequence
      1. Entire Document
2. Product Owner of InvestiMapp
   1. Reading Sequence
      1. 1 - Introduction
      2. 2 - Overview
      3. 4 - Functionality
      4. 5.5 - Business Rules
3. Testers and Quality Control of InvestiMapp
   1. Reading Sequence
      1. 1 - Introduction
      2. 2 - Overview
      3. 4 - Functionality
4. Documentation Writers
   1. Reading Sequence
      1. Entire Document

## Product Scope

* + 1. **Short Description** - Investing app will be a web-based application that tracks users individual investment portfolios. The app will be used by any user who thinks their trading platform is complicated and is hard to use to track successes and failures
    2. **Purpose** - InvestiMapps’s purpose is based on giving the user the best possible display of their investment portfolios’ movements. Most all of the current publically available platforms to track your portfolio are integrated with the trading platform. Therefore they focus mainly on the success of a stock on any given day. This is great for day traders or for people who are looking to buy stock. If you just want to watch your current investment it can be confusing and lead to bad decisions.
    3. **Relevant Benefit** 
       1. Show the user their specific overall net gains or loss on all currently held stocks.
       2. Allow the user to be informed of the stock’s performance over different periods of time.
    4. **Goals**
    5. Great user interface and experience
    6. Security of personal information
    7. Integration with trading platform (Robinhood)

## References

[Style Guide](https://atlassian.design/) - Atlassian Design Guide

[AWS Security](https://d0.awsstatic.com/whitepapers/aws-security-best-practices.pdf)

[AWS WhitePapers](https://aws.amazon.com/whitepapers/)

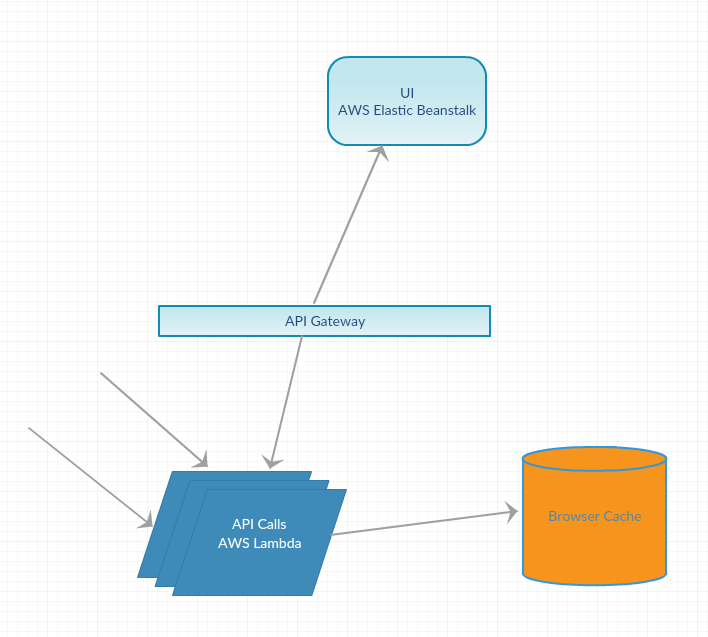
[Auth0 JS](https://auth0.com/docs/quickstart/spa/vanillajs)

# Overall Description

## Product Perspective

* + 1. This is a brand new product. InvestiMapp was a thought after a team member made a couple bad trades based on the markets that day and not on the overall net gain of the stock. This lead to selling the stock even though the overall net gain was still good and it was still a solid stock to hold. InvestiMapp looks to fix that. With the default setting comparing the stock price at which you purchased the stock to what the current price is. Not what that stock opened at and what it is at when you look at it.

Major Components - Diagram



## Product Functions

* + 1. **User Functions**
       1. Create an Account
       2. Input Investment Portfolio
       3. Set Notifications
       4. Review Net Gain and Loss across portfolio
    2. **System Functions**
       1. API calls to Markets
       2. Account Controls
       3. Responsive

## User Classes and Characteristics

* + 1. **Daily User (Robinhood Trader)**
       1. Frequency
          1. This user will most likely check InvestiMapp daily to see if any moves need to be made.
       2. Subset of Functions
          1. This user will use all functionality as the software is based off of this user
       3. Experience - Varies from experience day/long trader to someone who started using Robinhood yesterday
    2. **Weekly User**
       1. Frequency
          1. This user will most likely check InvestiMapp every week or so just to monitor their portfolio. These users do not check everyday because of either a large and stable portfolio or that they bought stock to hold for a while but still want to look at progress or lack of.
       2. Subset of Functions
          1. This user will use all functionality also
       3. Experience - Varies from experience day/long trader to someone who started using Robinhood yesterday

## Operating Environment

The System will be based on the AWS ecosystem. AWS Codestar setups up web applications and web services. We will be creating a web application and a web service using the frameworks provided. They will handle deployment, building, and testing. Github will handle our version control.

* + 1. **Server Infrastructure**
       1. Web Server (All Latest Versions: Automatically updated)
       2. AWS Lambda
       3. Auth0 Login and Token Service
       4. AWS DynamoDB
       5. AWS CodeStar Deployment Manager
       6. AWS CloudFormation

## Design and Implementation Constraints

Since this is a class project the limits we have are as follows:

* + 1. Time - Only semester long class
    2. Budget - We will not be looking to spend any money at all.

## User Documentation

* + 1. FAQ Page - We will supply an FAQ question and add to it as users have questions.
    2. User Video for tutorial usage

## Assumptions and Dependencies

* + 1. **Dependencies**
       1. Realtime API for Stock Prices - [API](https://api.robinhood.com)
       2. AWS Ecosystem
       3. Auth0 Reliability
    2. **Assumptions**
       1. Assume the user has an investment portfolio

# External Interface Requirements

## User Interfaces

* + 1. **Design**
       - 1. Top Navigation Bar

User Identification / Login

* + - * 1. [Semantic UI](https://semantic-ui.com/) - Cards
        2. Errors - Handled with UI modals (ui modal basic with error information)
    1. **Colors**
       1. Red and Black
    2. **Buttons**
       1. All **submit** buttons in blue
       2. All **add** buttons in green
       3. All **delete** buttons in red

## Hardware Interfaces

* + 1. All devices, including all of those in the server infrastructure, need to be able to connect to the internet. Whether that be wired or wireless.

## Software Interfaces

* + 1. All Communications - Via HTTP post
       1. Out : JSON
       2. In: JSON
       3. Description : This interface will be between all interfaces.

# System Features

## Account Creation and Login

4.1.1 ~~Description and Priority~~

~~This feature will allow unregistered users to registered for an account. This will allow for persistent data across many web sessions for a single user, increasing usability.~~

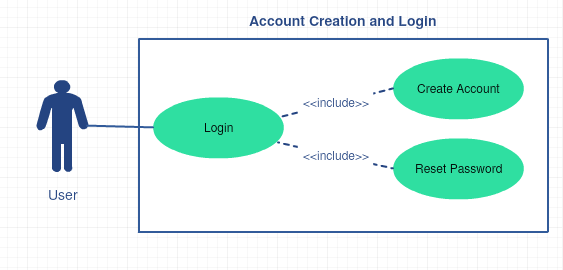
**~~Priority: low-~~** ~~This is an absolutely necessary functionality.~~

~~4.1.2 Stimulus/Response Sequences~~

**~~Stimulus~~**~~: User clicks “Create Account”~~

**~~Response~~**~~: A form is shown to the user to fill out information. After submitted an email will be sent to the the user to confirm their account creation.~~

~~4.1.3 Use Case Diagram~~



4.1.4 ~~Functional Requirements~~

~~REQ-1: The system shall allow for an unregistered user to create an account.~~

~~REQ-2: The system shall allow a user to login with credentials provided on account creation. (REQ - 1)~~

~~REQ-3: The system shall allow for a user to reset their password if they forget it.~~

## Stock Data Manipulation

4.2.1 Description and Priority

This feature will allow the users to add and delete stocks. This manipulation will be initiated by the user to allow them to take advantage of the dashboard and UI that InvestiMapp offers.

**Priority: HIGH -** This is an absolutely necessary functionality.

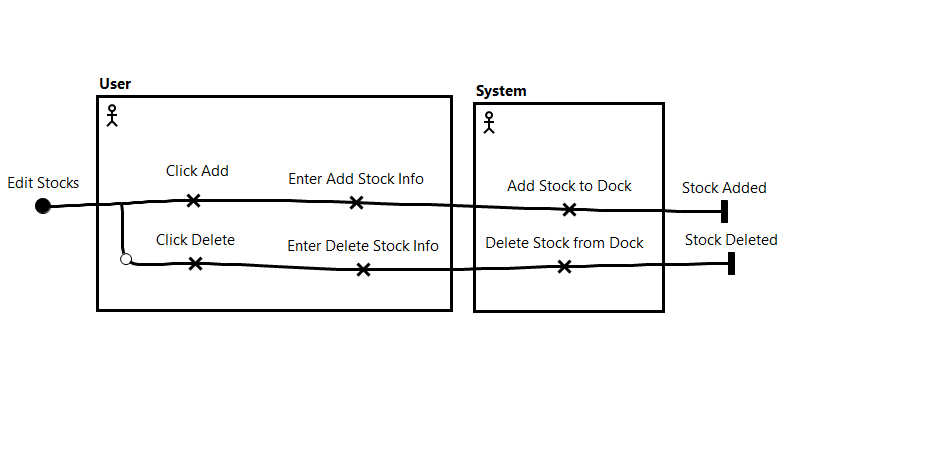
4.2.2 Stimulus/Response Sequences

**Stimulus**: User clicks “Add Stock” or “Remove Stock” and fills out the forms that display then submits.

**Response**: The page will redirect to their dashboard with the manipulation complete and stocks updated.

4.2.3 Use Case Diagram

## 



4.2.4 Functional Requirements

REQ-4: ~~The system shall allow for a registered and logged in user to add stocks and the stock info to their account portfolio.~~

The system shall allow for a user to add stocks and stock info to their browser’s cached portfolio.

REQ-5: ~~The system shall allow for a registered and logged in user to delete existing stocks from their account portfolio.~~

The system shall allow for a user to delete existing stocks form their browser’s cached portfolio.

REQ-6: The system shall validate all stock names against a list of supported stock markets.

REQ-7: The system shall support the New York Stock Exchange and the NASDAQ for users to claim stock on.

## Dashboard Updates

4.3.1 Description and Priority

This feature will be the comparison feature that will make InvestiMapp stand out from the competitors. We will compare the price at which the user bought the stock to the current stock price and show a net gain or loss.

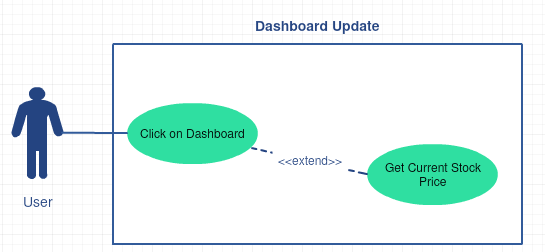
**Priority: HIGH -** This is an absolutely necessary functionality.

4.4.2 Stimulus/Response Sequences

**Stimulus**: User views the dashboard page.

**Response**: The page will be updated during the load with the latest comparison stock price information.

4.4.3 Use Case Diagram

4.4.4 Functional Requirements

REQ-8: The system will compare the current stock price on the stock market to the price at which the user bought the stock and show the net gain or loss.

REQ-9: The system shall show the current stock price of the user’s stocks.

## Dashboard Graphs

4.5.1 Description and Priority

This feature will be graph your net gain and loss on any given stock for different time intervals.

**Priority: LOW -** This feature is not necessary

4.5.2 Stimulus/Response Sequences

**Stimulus**: User views the dashboard page.

**Response**: The page will be updated during the load with the latest graphs of user stocks

4.5.3 Functional Requirements

REQ-10: The system will graph the output of REQ-8 over different time periods.

REQ-11: The system will graph the output of REQ-9 over different time periods.

## ~~Notifications~~

~~4.6.1 Description and Priority~~

~~This feature will enable users to set notifications to be sent to them if a certain stock price changes a set percentage.~~

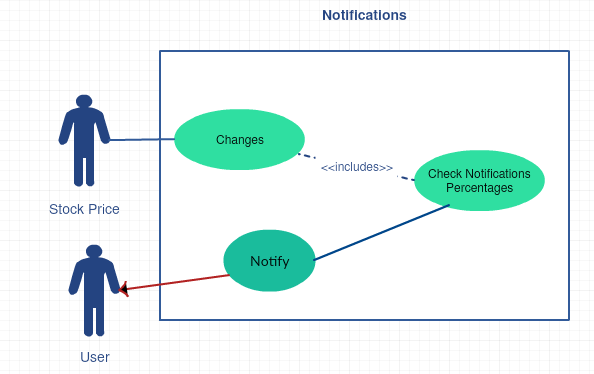
**~~Priority: LOW -~~** ~~This feature is not necessary~~

~~4.6.2 Stimulus/Response Sequences~~

**~~Stimulus~~**~~: A stock price changes a certain percentage.~~

**~~Response~~**~~: The system sends an email or text to the user.~~

~~4.6.3 Use Case Diagram~~

~~~~

~~4.6.4 Functional Requirements~~

~~REQ-12: The system will send notifications to specific user with the information on the stock that is changing in percentages.~~

\*because the application is ultimately a companion to Robinhood, the user would not want notifications from both applications, so this requirement has been deprecated.

# Other Nonfunctional Requirements

## User Requirements

* + 1. REQ -15: The user should have a basic understanding of certain terms in regards to trading. Such as equity, options trading, etc.

## Performance Requirements

* + 1. Stock Updates - Need new stock data at a minimum of 30 seconds per update. Need to make sure that users have the ability to see market fluctuations.

REQ-13: The system shall load new stock prices every 30 seconds for all users.

* + 1. Page load time - All page load time should be less than a second. This does not include all account info on the page.

REQ-14: The system shall always load page in less than a second 95% of the time.

## Safety Requirements

* + 1. There are no safety requirements
    2. If you are addicted to gambling call : 1-800-522-4700

## Security Requirements

* + 1. **~~User Information~~** ~~- AWS Cognito, if setup correctly, will be more secure than anything we could build. They allow login with many types of other platforms (Google, Amazon, Facebook). This means they can use two-factor authentication and password resetting through the other services.~~
    2. **Portfolio information** - This will need to be protected but will not be critical information because this is user inputted information and the initial scope of the project does not hold any bank account information or login for other services.
    3. **~~Token Authentication~~** ~~- This requirement will be for all of the HTTP requests that we will be making. These include all of the requests too and from our API Gateway that interface with the database and all other user data. These tokens will be sent from the client and then authenticated on the API side. Most of the authentication will be handled by Auth0, a third part service.~~
    4. **Encrypted Caching** - In place of user authentication and storing user information, the user’s browser will hold a constant session to the application in which their added stocks will be saved. Because there will be no user data saved on the application, there will be far less possibility for the user’s information to be at risk.