**Software Requirements Specification**

**for**

InvestiMapp

**Version 1.0 *pending***

**Prepared by Team InvestiMapp**

**February 13th, 2018**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# **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.

## **Document Conventions**

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

## **Intended Audience and Reading Suggestions**

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

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**
     1. Great user interface and experience
     2. Security of personal information
     3. Integration with trading platform (Robinhood)

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

## **References**

* 1. [Style Guide](https://atlassian.design/) - Atlassian Design Guide
  2. [AWS Security](https://d0.awsstatic.com/whitepapers/aws-security-best-practices.pdf)
  3. [AWS WhitePapers](https://aws.amazon.com/whitepapers/)

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

# **Overall Description**

## **Product Perspective**

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.

2.2 - Major Components

Insert Drawing of components here

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

## **Product Functions**

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

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

## **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

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## **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. AWS Cognito Service (Login Service)
       4. AWS DynamoDB
       5. AWS CodeStar Deployment Manager
       6. AWS CloudFormation

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## **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.

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

## **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

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## **Assumptions and Dependencies**

* + 1. Dependencies
       1. Realtime API for Stock Prices - [API](https://www.alphavantage.co/documentation/)
       2. AWS Ecosystem
    2. Assumptions
       1. Assume the user has an investment portfolio

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

# **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

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

## **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.

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

## **Software Interfaces**

* + 1. API Call Lambda Function
       1. Out : JSON
       2. In: JSON
       3. Description : This interface will be between a Stocks API and a lambda function.

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

## **Communications Interfaces**

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

# **System Features**

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## **System Feature 1**

*<Don’t really say “System Feature 1.” State the feature name in just a few words.>*

4.1.1 Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

4.1.2 Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

4.1.3 Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

REQ-1:

REQ-2:

## **System Feature 2 (and so on)**

# **Other Nonfunctional Requirements**

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

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

## **Safety Requirements**

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

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

## **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.

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

## **Software Quality Attributes**

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

## **Business Rules**

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

# **Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*