**Software Requirements Specification**

**For**

**Armadillo 4000**

**Version 1.0 approved**

**Prepared by Juergen Kriz, Ryan Mock, Sean Dunbar, Sunveer Sandhu, Zach Cockle**

**January 18, 2018**

Table of Contents

[1. Introduction 3](#_Toc510180991)

[1.1 Purpose 3](#_Toc510180992)

[1.2 Scope 3](#_Toc510180993)

[1.3 Definitions, Acronyms and Abbreviations 4](#_Toc510180994)

[1.4 Reference(s) 4](#_Toc510180995)

[1.5 Overview 4](#_Toc510180996)

[2. Overall Description 4](#_Toc510180997)

[2.1 Product Perspective 4](#_Toc510180998)

[2.2 Product Functions 5](#_Toc510180999)

[2.3 User Characteristics 7](#_Toc510181000)

[2.4 Constraints 7](#_Toc510181001)

[2.5 Assumptions and Dependencies 7](#_Toc510181002)

[2.6 Apportioning of Requirements 8](#_Toc510181003)

[3. Specific Requirements 8](#_Toc510181004)

[3.1 External Interfaces 8](#_Toc510181005)

[3.2 Functional requirements 9](#_Toc510181006)

[3.3 Non-functional requirements 10](#_Toc510181007)

[3.3.1 Performance requirements 10](#_Toc510181008)

[3.3.2 Other non-functional requirements 10](#_Toc510181009)

[3.4 Logical Database Requirements 10](#_Toc510181010)

[3.5 Design compliance 10](#_Toc510181011)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for Changes | Version |
| Group | March 30, 2018 | Revised program functionalities. | 1.1 |

# 1. Introduction

## 1.1 Purpose

**1.1.a.**

The intended purpose of this SRS is to analyze and collect all ideas that we have brainstormed to define the system. It will also explain the purpose and features of the program, what the program will do, and what constraints the system must deal with, as the customers have instructed us to do. This document is intended for customer review and reference.

**1.1.b.**

The intended audience for this program would be the staff of the company that we are selling this program to, due to the fact they will be using this program frequently to check stocks.

## 1.2 Scope

**1.2.a.**

We have chosen the name of the program to be the Armadillo 4000, based off the company's name, as the company name is *ZooTech.* Since we are not marketing this application to other companies we can avoid applying for a trademark.

**1.2.b.**

Once operational, the program will be able to retrieve specific stock info entered by the users and suggest trades based on user trading rules for the given stocks that can be entered up to a maximum of five stocks. The program only returns an alert when rules have been achieved. The program does not sell stocks on behalf of the user, it only will notify them that their rule has been achieved. Also, the program will allow the user to choose whether to show any data on the stock entered, or only when the rule has been met. This will allow the software to remain uncomplicated.

**1.2.c.**

The user will provide chosen stocks, as well as trading rules. The program will then retrieve stock data in real-time and provide alerts based on the trading rules. Benefits allow the user to receive real-time notifications regarding stock market events, allowing the user to make better use of their time. Objectives include displaying real-time stock market data from specified stocks; goals include helping the user to maximize profits and minimize losses.

## 1.3 Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Meaning | Definition |
| NYSE | New York Stock Exchange |
| NASDAQ | American Stock Exchange |
| API | Application Program Interface, used to interact with an external application to pull data or other variables. |
| GUI | Graphical User Interface, the window through which user interacts with the software. |
| RSI | Relative Strength Index, a value used to determine a stock’s momentum (up/down). |
| SMA | Simple Moving Average, a value used to identify a stock’s current trend. |

## 1.4 Reference(s)

<https://simple.wikipedia.org/wiki/Application_programming_interface>

<https://www.investopedia.com/terms/s/sma.asp>

<https://www.investopedia.com/terms/r/rsi.asp>

## 1.5 Overview

The rest of the SRS contains two more sections describing the product perspective and the functions that the product would provide. The second chapter provides a brief description about the product and how it will react to other users when they are using the application.

The third chapter goes over the specific program requirements, going into detail regarding how the system acts with different interfaces.

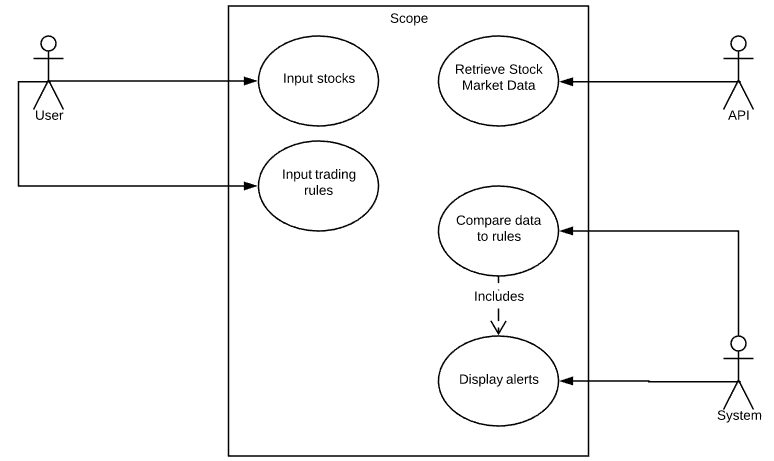
Finally, the document is organized by using headers with a Table of Contents allowing quick access to the different sections of the SRS.

# 2. Overall Description

## 2.1 Product Perspective

Our stock market application is independent of other programs, but it is not self-contained, as it relies on an API to retrieve data from the stock market.

## 2.2 Product Functions

The user inputs chosen stocks, as well as their custom trading rules. The program will then retrieve stock data via the API, and display alerts based on the given trading rules.

|  |  |
| --- | --- |
| Name: | Input Stocks |
| Actor: | User |
| Description: | This use case describes how a user would input their stocks of choice. |
| Successful Completion: | 1. User selects stock market, as opposed to cryptocurrency.  2. User enters the stock symbol of a chosen stock.  3. The program notifies user that the input has been accepted.  4. Program proceeds to "Input Trading Rules" Use Case.  5. Program asks user to enter another stock, or finalize list.  If the user enters another stock, repeat from step 3.  If the user finalizes the list, program notifies user of selection acceptance. |
| Alternative: | 1. User selects which market to monitor.  2. The user enters the stock symbol of a chosen stock.  3. Program notifies user that stock is not available on the chosen market.  4. Program asks user to re-enter stock symbol. |
| Precondition: | User wants to select stocks. |
| Post-condition: | Program monitors chosen stocks. |
| Assumptions: | None |

|  |  |
| --- | --- |
| Name: | Customize Trading Rules |
| Actor: | User |
| Description: | This use case describes how a user would input their custom trading values. |
| Successful Completion: | 1. Program asks user to set values in trading rules  2. User inputs high and low notification values. |
| Alternative: | 1. Program asks user for trading rule values.  2. User enters incorrect value. |
| Precondition: | User has entered an accepted stock. |
| Post-condition: | Program integrates custom values. |
| Assumptions: | User enters realistic values. |

|  |  |
| --- | --- |
| Name: | Retrieve Stock Market Data |
| Actor: | API |
| Description: | This use case describes how the API retrieves the market data. |
| Successful Completion: | 1. API sends a GET request to the stock market server.  2. Stock market server replies with requested data.  3. API receives requested data.  4. API passes data to program. |
| Alternative: | 1. API sends a GET request to the stock market server.  2. Stock market server is non-responsive. |
| Precondition: | User has selected stocks and defined rules. |
| Post-condition: | Program has retrieved data. |
| Assumptions: | None |

|  |  |
| --- | --- |
| Name: | Compare Data to Rules |
| Actor: | Armadillo 4000 |
| Description: | This use case describes how the program compares data to the defined rules. |
| Successful Completion: | 1. Program checks stock values.  2. Program checks defined rule values.  3. Program compares values. |
| Alternative: | N/A (unless program critically fails). |
| Precondition: | User has selected stocks, defined rules and API has retrieved data. |
| Post-condition: | Program has compared values to rules. |
| Assumptions: | None |

|  |  |
| --- | --- |
| Name: | Display Alerts |
| Actor: | Armadillo 4000 |
| Description: | This use case describes how the program would display alerts when rules are met. |
| Successful Completion: | 1. Program compares stock market data to defined rules.  2. Defined rule is met.  3. Program displays an alert. |
| Alternative: | 1. Program compares stock market data to defined rules.  2. No defined rule is met.  3. No alert is displayed. |
| Precondition: | User has selected stocks and defined rules. |
| Post-condition: | User is alerted. |
| Assumptions: | None |

## 2.3 User Characteristics

Armadillo 4000’s users are assumed to have basic computer skills, minimal experience using software, and little to no technical expertise. Users are expected to have a level of familiarity with the NYSE stock exchange that will allow them to identify the associated stock ticker symbols with the stocks they wish to select for monitoring. Users will also be required to enter input to specify the trading rule values such as stock price dollar/percentage increases or decreases that must be satisfied to trigger an alert from the system.

## 2.4 Constraints

**2.4.b**

Hardware limitations - Internet connection. Windows OS or MAC OSX.

**2.4.c**

No other applications.

**2.4.d**

No parallel operation, only a single user/instance at a time.

**2.4.e**

No audit functions, as state is not saved, and no records are kept.

## 2.5 Assumptions and Dependencies

If a user did not meet the limitations provided, the program would cease to be operable/functional.

## 2.6 Apportioning of Requirements

Some features that future versions could, but do not currently, include:

* **Saved profiles of chosen stocks and trade rules**
  + Would require the program to save data to a text file, so that it could be reused by the program without the user having to re-enter the stocks and rules.
* **GUI (Graphical User Interface)**
  + Would require the developers to design and connect a GUI. This GUI would make it much simpler and cleaner for the user to select their stocks and to create their trading rules, as most users don't deal with a command line environment often.
* **Stock categories/Codes**
  + Would require the program to be continuously pulling updated stock market data to keep current. This would allow the user to select, for example, "Technology Stocks", which would have the program select stocks automatically that are based upon technology.
* **Web integration**
  + Would require a website to be developed to house the application, to make it usable via a browser only. This would allow users to simply open the webpage and run the program from anywhere, instead of having to be on the machine the software is housed on.
* · **Option to change monitored stocks**
  + Would require the developers to code the stocks using instance variables, to allow the user to reset the values without restarting the program.

# 3. Specific Requirements

## 3.1 External Interfaces

Minimum interface hardware requirements will be a monitor and a keyboard.

**Inputs**

Application user input will be via command line, using the keyboard, in response to text-based prompts displayed on the monitor. The user will first be prompted to choose the stock exchange or cryptocurrency. Upon receiving a valid selection, the system will prompt the user to enter the stock/crypto symbol to be monitored and again await user input. Once the symbol is entered, the user will be asked to choose a rule from the options provided. Upon receiving this user input, the system will prompt the user to enter the appropriate values, using the current market value as a base price. Subsequent numeric user input will conclude information gathering from the user for this specific stock and trigger a prompt asking the user to indicate with a ‘Y’ or an ‘N’ if they wish to enter any further stocks. A ‘Y’ user input will repeat the stock entry process described above. An ‘N’ user input will end the stock entry process and trigger a confirmation message displayed on the monitor.

**Outputs**

The system will monitor the price of user-selected stocks in real-time and compare their values to user-defined notification values. Once the criteria for triggering a user notification is met, the system will notify the user of the event via a text-based message displayed on the monitor. This notification will include the stock symbol of the stock whose change in value triggered the notification and the current price of that stock. This must occur within a maximum of ten seconds from the time that the system receives the stock valuation data that triggers the notification.

## 3.2 Functional requirements

**3.2.1 Selecting Stocks:**

The user will be able to choose which stocks they wish to set rules for. This will be done by typing the stock symbol of the desired stock. The program will then verify that the stock exists, and a message will be displayed that the stock was successfully selected.

**3.2.2 Input Trading Rules:**

After successfully selecting a stock for the program to monitor, the user will be asked by the system to input values for when the system should alert the user. These are referred to as the trading rules. The value of each stock will be compared to the trading rule(s) chosen. For percentage, they will be set as an increase and decrease between the first value of the stock retrieved compared to the most recent value. RSI will look for the stocks Relative Strength Index to go above 70 or below 30, and SMA will look for the long-range and short-term values to cross.

**3.2.3 Retrieve Data in Real-Time:**

Once the user has determined the stocks they wish to monitor and the trading rules by which the system will alert them, the program will begin pulling information from the stock market using an API.

**3.2.4 Compare Data to Trade Rules:**

As the program receives new data, it will compare it to the original value it retrieved when it was started. The value gained from the comparison will be checked against the user’s trade rules for the stock in question.

**3.2.5 Display alerts:**

Once a user’s trade rule has been met, it will provide the user with a notification. The user must then decide whether or not to sell or buy the stock, however the actual trade will take place outside the system; this app will not support stock trading. This notification must occur within a maximum of ten seconds from the time that the system receives the stock valuation data that triggers the notification.

## 3.3 Non-functional requirements

Ability of user to operate program, without assistance, within 5 minutes. (Easy to use).

### 3.3.1 Performance requirements

**3.3.1.a**

Number of terminals to be supported: 1, as each user will be running a separate instance of the program.

**3.3.1.b**

Number of simultaneous users to be supported: 1, as each user will be running a separate instance of the program.

**3.3.1.c**

Type of info to be handled: Identified by stock symbols in the form of Strings. 3 trading rules. Type: Numeric (double). Stock symbols (Strings).

### 3.3.2 Other non-functional requirements

N/A

## 3.4 Logical Database Requirements

N/A

## 3.5 Design compliance

The program's response time is limited by the web server the program is pulling data from.