**Software Design Description**

**For**

**Armadillo 4000**

**Version 1.1 approved**

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

**February 05, 2018**

Table of Contents

[1. Introduction 3](#_Toc510182148)

[1.1 Purpose 3](#_Toc510182149)

[1.2 Scope 3](#_Toc510182150)

[1.3 Intended Audience 3](#_Toc510182151)

[1.4 Reference(s) 3](#_Toc510182152)

[1.5 Summary 3](#_Toc510182153)

[2. Definitions 3](#_Toc510182154)

[3. Conceptual model for software design descriptions 4](#_Toc510182155)

[4. Design Viewpoints 4](#_Toc510182156)

[4.1 Composition Viewpoint 4](#_Toc510182157)

[4.2 Logical Viewpoint 5](#_Toc510182158)

[4.3 Information Viewpoint 5](#_Toc510182159)

[4.4 Algorithm Viewpoint 6](#_Toc510182160)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for changes | Version |
| Ryan Mock | March 30, 2018 | Revision of program functionalities. | 1.1 |

# 1. Introduction

## 1.1 Purpose

The intended purpose of this SDD is to provide the developers with a design overview with which to build the program.

## 1.2 Scope

The scope of this document is for documenting the fields, methods and interactions between classes.

## 1.3 Intended Audience

The intended audience for this design document is the developers and testers who will be programming and debugging this software.

## 1.4 Reference(s)

<https://www.investopedia.com/articles/technical/052201.asp>

<https://www.investopedia.com/articles/technical/071601.asp>

## 1.5 Summary

The rest of the SDD contains two more sections. The first breaks it down on a conceptual level, whereas the second contains more technical details, breaking it down into specific classes, methods and variables.

## 2. Definitions

|  |  |
| --- | --- |
| ***SDD*** | Software Design Document. |
| ***Fields*** | A field, in Java, is a variable which is located inside a class. |
| ***Methods*** | Chunk of code, used for a single task, which can be reused throughout the program. |
| ***Classes*** | A class is defined as being a template that objects are created from. |
| ***Object*** | An instance of a class. |
| ***Variables*** | A container that holds value(s) for use in a program. |
| ***API*** | Application Programming Interface. |
| ***Financial Asset*** | Used to describe the stock or currency chosen. |
| ***SMA*** | Simple Moving Average. The simple moving average is the average price over a specified period. |
| ***RSI*** | Relative Stock Index. The relative stock index indicates whether movement of the stock is caused by overbuying or overselling. |

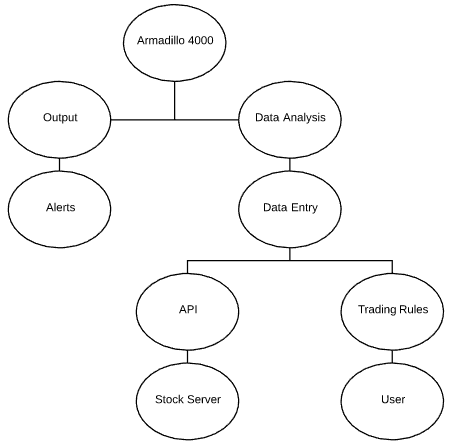
# 3. Conceptual model for software design descriptions

Users will first be asked to select stocks or cryptocurrency. Once the selection has been made, the user will enter the relevant symbol (Ex: Alphabet Inc. stock symbol would be *GOOGL)*. After the user has entered the symbol, they are then asked to select one trading rule for that specific stock from a pool of options. The options for trading rules are stock price, stock SMA, and stock RSI.  When the user executes the program, the API will start to pull information from the stock server. Once the current value from the stock server fulfills one of the user’s trade rules, 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.

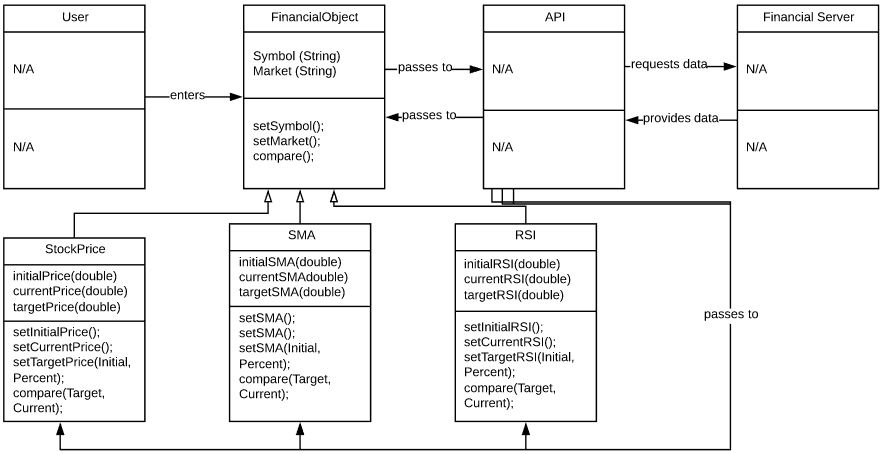
# 4. Design Viewpoints

## 4.1 Composition Viewpoint

The Armadillo 4000 is comprised of 2 primary functions: The Output function, which is used to facilitate the display of alerts to the user, and the Data Analysis, which is further broken down into 2 subsections. These 2 subsections contain the API and the Trading Rules. The API section queries and receives real-time stock data from the stock market server, whereas the Trading Rules section contains user-defined criteria which are used to evaluate stock market data and display alerts when specific conditions are met.



## 4.2 Logical Viewpoint



## 4.3Information Viewpoint

The user selects the type of market to monitor [Stock/Crypto]. They then enter the financial asset(s) [Stock/Crypto-currency] to be monitored. Then the user is asked to choose a trading rule to be linked to the asset, from the provided menu. The program will ask the user if they want to enter another financial asset. If the user chooses to do so, the program repeats the previous steps. Otherwise, the program sends a request to the API for the relevant data. The API then sends a request to the financial server, which in turn replies with the requested data. Then, the API passes the necessary data back into the program. The program then uses this data in conjunction with the trading rules to display alerts when conditions are met. If conditions are not met, the program continues querying the server, updating the data, and checking the data against the trading rule.

## 4.4 Algorithm Viewpoint

1. The program asks the user to enter their choice of market.
2. The program asks the user to enter their choice of stock/crypto-currency (Bitcoin).
3. The program asks the user to choose from a list of trading rules, based on the type of market chosen.
   1. If the user chooses to monitor their stock/currency based on price, the program asks the user to enter a percentage value, which will be used to determine the value the user will be alerted at.
   2. If the user chooses to monitor their stock/currency based on the SMA, the program asks the user to enter a time period value which will be used to determine the length (in days) the API will use in its call.
   3. If the user chooses to monitor the stocks RSI, the program asks the user to enter a time period which will be used to determine the time (in days) that will be used in the API call.
4. The user then enters a value into their chosen trading rule, for example, 10%.
5. If they choose to enter more, the program asks the user to select market and stock.
6. If the user chooses to stop entering stocks by entering the menu option “Start Monitoring”, the program will begin querying the server for the relevant data.
7. The program compares the users chosen values to the relevant data from the stock server.
8. If conditions are met, an alert is displayed to the user within ten seconds of the query being made.[[1]](#footnote-1)
9. If conditions are not met, the program continues to update values by querying the server and comparing the values that have been set.

1. The alert will be displayed via the compare method, as opposed to having its own method. [↑](#footnote-ref-1)