SOFTWARE REQUIREMENTS
SPECIFICATION

for

Algorithmic Trading Application

Version 1.0 approved

Prepared by Jacob Berwick, Francisco Bolanos,
Hae Won Cho, Dakota Cleeves, Alec Hayden,
Matthew Jordan, Jose Ramos, Shile Song

Oregon State University, CS 461, Fall 2019, CS10
## Contents

1 **Introduction** 4  
   1.1 Purpose ................................................. 4  
   1.2 Scope .................................................... 4  
   1.3 Product Overview ....................................... 4  
      1.3.1 Product Perspective ............................. 4  
      1.3.2 Product Functions ............................... 5  
      1.3.3 User Characteristics ............................ 5  
      1.3.4 Limitations ....................................... 5  
   1.4 Definitions ............................................ 6  
   1.5 References ............................................ 6  

2 **Specific Requirements** 7  
   2.1 External Interfaces ................................... 7  
   2.2 Functions ............................................. 7  
   2.3 Usability Requirements ................................ 7  
   2.4 Performance Requirements ........................... 8  
   2.5 Logical Database Requirements ...................... 8  
   2.6 Design Constraints .................................. 8  
   2.7 Software System Attributes ......................... 8  

3 **Gantt Chart** 10
1 Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for the Algorithmic Trading Application. Specific requirements and design concerns are addressed to ensure efficient and successful development.

1.2 Scope

The software product to be produced is a web-based application. The application will create investment portfolios for users, containing stocks, options, and equities. The investment portfolio will be dependent on the user’s risk tolerance and their account balance. The portfolio will be monitored and changed based on risk management and dynamic strategies. The benefit of using this application is that it does all of the research and trading involved in developing an investment portfolio. There is no need to gain an in-depth knowledge of the markets to use this application. Another benefit, as a result of the portfolios being tested for financial resiliency, is that customers won’t need to constantly check on their portfolios. They can purchase an investment portfolio and let it be, having the peace of mind that it was built for resiliency. The main objective of this application is to bring financial resilience to those who cannot effectively invest on their own. We want the portfolios created by the application to be able to withstand financial shocks so that the user never loses more than their tolerated amount. The main goal is to have an easy-to-use application that does all the analysis and trading involved in portfolio management so that a user’s investment portfolio can be financially resilient.

1.3 Product Overview

1.3.1 Product Perspective

The user interface will be a web-accessible application. Users will be able to login to view their portfolio and other market data. The graphical interface should be pleasant and easy to navigate. Charts should be displayed to show portfolio performance and stock prices over time. Other relevant data should be neatly organized into tables under the chart. The only hardware interface will be through the interaction the browser has with the user’s computer and will not be addressed in our application. This application will be built using a micro-service architecture so that each component can communicate freely and effectively. Our main application will have to interface and communicate with
the data interface in order to access market and user data. Additionally, the underlying machine learning processes will need to interface with the data interface as well in order to make decisions. Lastly, the resilience manager will need access to both the machine learning processes and the data interface to analyze the resilience. Our application will be capable of providing the user with messages and notifications about the status of their portfolio. This will allow for a trustworthy and accurate summary of the changes in portfolio being made by our application. This web application will be a standalone product. It is not an element to a larger system nor is it the parent product of a smaller system. It will later be ported to a mobile application as well.

1.3.2 Product Functions

This product will perform 5 major functions. The first function required is the development of investment portfolios taking into account the user’s risk tolerance and the amount of money they want to invest. The second function required is the management of investment portfolios through buying and selling equities for users. The third function is the development of an exit and rebalancing strategy that will be presented to the user alongside the proposed portfolio. The fourth function of the application is the execution of the exit and rebalancing strategy. Lastly, the fifth function will store user and portfolio data in a database for client use.

1.3.3 User Characteristics

While we certainly welcome anybody who would like to invest safely, we are aware that this product will cater towards those reaching retirement age and above. Education level of our users is expected to be between High School and college level. Our users will typically have minimal experience in investing and trading. This product targets older people, so it is not expected that they will be experts or advanced users of technological products such as web applications.

1.3.4 Limitations

Because this app will most likely be recognized as a securities brokerage app by the SEC, we will have to follow all policies and regulations set by the SEC\(^1\). As a result, audits may need to be performed so the application should be developed with the idea of making audits easy to perform. Since this app will allow users to buy expensive portfolios, robust handshake protocols will be needed to insure the clients are in communication with the correct servers, and that the servers are in communication with the correct users. Since buying and selling stocks, options, and other securities comes with commission and broker fees, it is important that the application behaves exactly the way it needs to. Or else we face profit losses paying fees in the case that the application behaves erroneously and performs an action not approved by the user. Lastly, in order to purchase an investment portfolio, we would need an extensive amount of personal information such as name, date of birth, address, bank account, and possibly Social Security Numbers. We need to be able to sacrifice some speed and efficiency for security.
1.4 Definitions

1. SEC - Securities and Exchange Commission

1.5 References

[2]. https://www.quantconnect.com/
2 Specific Requirements

2.1 External Interfaces

The main external interface is Interactive Brokers. Interactive Brokers is a US based brokerage firm which we will be using to access stock market data[1]. Another external interface we will be using is QuantConnect, an open-source cloud-based algorithmic trading platform[2]. QuantConnect will serve as the engine for our learning algorithm and act as the core for the program’s logic.

2.2 Functions

A basic outline of the application’s use is as follows: The user will enter their risk tolerance and the amount they want to invest. From there, the user can request a recommendation for a portfolio. The application will then collect historical data on the markets. Using the data, the application applies machine learning and resilience techniques to create an optimal portfolio for the user. If the user is happy with the recommendation, they can approve the portfolio. The application, upon approval, will purchase all of the assets found in the recommended portfolio. The application would then follow through with the exit strategy and rebalance strategy to handle the user’s old portfolio. The application would then display different statuses about the portfolio and other statistics on its performance.

2.3 Usability Requirements

The first usability requirement for the User Interface is learnability. Users should be able to easily accomplish basic tasks when they first come across the design. The second component is efficiency. Users should be able to quickly perform the tasks they want to perform. The third component is memorability. User’s should be able to reestablish proficiency in using a UI, after not having used it for some time. The fourth component is errors. Errors should be very minimal and user’s should be able to recover from them. Also, there will be an error prevention feature so users cannot make major portfolio mistakes. The last component is satisfaction. Users should feel satisfaction and be content when using our UI.
2.4 Performance Requirements

This application should be built to be scalable as the number of end users is unknown. As new users sign up, our infrastructure should be able to scale accordingly. In the future there is potential for thousands of users to be accessing the application simultaneously. 99% of requests should be processed in less than 1 second to ensure a satisfactory user experience. Additionally, market transactions should be prioritized to be executed as quickly as possible due to the fast rate of change in the market.

2.5 Logical Database Requirements

We will be using a structured database to store user and portfolio data. The types of information we are going to be storing is information on certain stock prices given a period of around a month. The second type of data that will be stored is the user data for the application. This includes account credentials as well as portfolio information. The various functions will be able to use this as training data to look for market patterns that our algorithm could potentially analyze and predict. This data will be used many times for the purpose of training our algorithm to recognize market trends. The user information will also be accessed whenever a user is trying to either login or check on their investments.

2.6 Design Constraints

One of the design constraints given the requirements is that the application must make investments within a certain window of profit. Our investment strategy follows a midterm approach. A generated portfolio should not be considered in day trading, but it should also not be overly long-term. Our client has specified that there should only occur a few trades a few times a month in order to keep user portfolios relatively stable. This means that our algorithm will most likely have to be trained on this time frame.

2.7 Software System Attributes

For our application to be reliable we will research portfolio optimization methods, individual stock trading methods, and Monte Carlo simulation methods to ensure consumers achieve financial resilience. Security involves protecting from SQL injection because we will be using a database to store user and market information, encouraging our users to use strong passwords, having validation on both the client and server side, and protect against Cross-site scripting. To help with code maintainability, we will be developing the application making sure to avoid certain code smells. These coding smells include bloaters such as code, methods and classes that have increased to a large size making them hard to work with. Object Orientation Abusers which is an incomplete or incorrect application of object oriented programming principles. Change preventers which are changes that requires you to make many changes in other places too. Dispensables
which are pointless objects whose absence would make the code better. Lastly, couplers which involves excessive coupling between classes.
3 Gantt Chart

https://tinyurl.com/y3cgbxoj