CS CAPSTONE REQUIREMENTS DOCUMENT

NOVEMBER 28, 2017

INVESTMENT PERFORMANCE MOBILE APP

PREPARED FOR

HEDGESERV

EDISON TSAI _____ _____
Signature *Date*

PREPARED BY

GROUP 68

THE CLEVERLY NAMED TEAM

TYLER JONES _____ _____
Signature *Date*

SAMUEL COONEY _____ _____
Signature *Date*

AVIRAL SINHA _____ _____
Signature *Date*

CONTENTS

1 Abstract 2

2 Introduction 2

2.1 Purpose . . . 2

2.2 Scope . . . 2

2.3 Definitions, acronyms, and abbreviations . . . 2

2.4 Overview . . . 3

2.5 Product perspective . . . 3

2.6 Product Functions . . . 3

3 Specific requirements 3

3.1 Backend . . . 3

3.2 Frontend . . . 4

4 Timeline 4

1 ABSTRACT

This document will outlining what we are doing for the project and what our client will get once we are finished. This will be essentially a contract of our work showing in detail. We will be detailing different tasks at the individual level and some will detail requirements at the functional level. In addition to this we will also have a detailed Gantt chart showing our timeline of the work we have to do.

2 INTRODUCTION

2.1 Purpose

The purpose of this mobile application will be to provide it's users insight into the quality of their investments from a portfolio level. By using this application, a user will be able to track and gain valuable insight that wouldn't be possible otherwise.

2.2 Scope

The scope of this project is to provide users with a method to track investment portfolios containing stocks and stock options. Users will have the ability to enter in investments and recieve detailed tracking information on these assets. Data will be displayed including prices, purchase dates, volatility, and more for both portfolio wide and individual investments.

2.3 Definitions, acronyms, and abbreviations

- HedgeServ: A global, independent fund administration provider headquartered in New York with 10 offices and 195+ clients worldwide.
- Xamarin: A tool built into Microsoft Visual Studio that allows developers to deliver native C# applications to Android, iOS, and Windows devices.
- C#: (pronounced as *see sharp*) is a programming language developed by Microsoft. It encompasses strong typing, imperative, declarative, functional, generic, object-oriented, and component-oriented programming styles.
- Asset: A valuable property such as resources, estate, holdings, etc...
- Investments: An asset or item puchased with the hope that it will geenerate income and is not consumed today and used to create wealth in the future.
- Portfolio: A group of financial assets that are held by investors or managed by financial professionals.
- Options: Contracts that grant the right but not obligation to buy or sell an underlying asset at a set price before a certain date.
- Front End: The view of an application that provides the end user with a friendly interface to interact with.
- Back End: The behind the scenes of an application that typically handles business logic and the storage of data.
- Domain-Specific Lanuage: (Often refered to as *DSL*) A programming language that has been specialized to a particular application domain.
- SQL: A domain-specific programming language used for managing data held in a database management system.

2.4 Overview

The Investment Performance Mobile Application will allow users to view their investment portfolio on iOS and Android devices at a portfolio level. Users will recieve data at a portfolio level as well as drill down to any of their specific investments.

2.5 Product perspective

Currently, most mobile financial applications provide investment performance information on only one investment type at a time. This project seeks to solve the current lack of a tool that concurrently tracks a variety of different investment assets in one consolidated source. In doing so, this application will allow users to place and view all of their investments in a single location while simultaneously recieving performace updates for each individual investment and the portfolio as a whole.

2.6 Product Functions

This mobile app will provide users investment performance from a portfolio level. The user will be able to enter investments, either in bulk or individually, and have their portfolio be displayed all together. Then users will have the ability to view different data points about their portfolio as well as drill down into individual investments in order to get further data on a single asset.

While in the portfolio view, users will see a pie chart with their portfolio split up into percentages. For instance, if a user owns $1000 worth of assets, these may be across stocks or stock options. A user may have $500 invested in the stock market, and $500 in a call option. The pie chart would reflect this division. Additionally, the user will be able to retrieve benchmarks on their entire portfolio. The S&P 500, for example, would provide the user with a benchmark by which they could compare the performance of their entire portfolio against. Moreover, the user will be able to disect the peformance of their entire portfolio based off a variety of time intervals. The user will see the aggregate performance of their investments over time intervals ranging from a week all the way to multiple years+. Lastly, the user will be able to see their Sharpe ratio and volatility of their portfolio. The user will also be able to see all of this data at an individual investment level. This means that this app should provide the user time data, the Sharpe ratio, volatility, and benchmarks of individual investments, when relevant/available.

3 SPECIFIC REQUIREMENTS

3.1 Backend

- There must be no longer than a 3 second response time on any given query to the database. If queries fail, the data should be blank or missing, rather than inaccurate. Users should be able to rely on 100% accurate pricing data, even if not 100% up to date; inaccurate pricing is not an option. Crashing is better than showing incorrect data, however this crash rate should be low. Moreover, the crash rate will be measured as the number of crashes relative to the number of queries to the database, rather than crashes relative to unit time.
- Database must be able to accomodate users having more than one portfolio/account
- Database must be able to accomodate users having a joint portfolio/investment.
- All transactions must be stored in the database

- The final product must be secure. Usernames and passwords must be safely stored in the database, and prevent hacking attempts such as SQL injections.
- Investment prices stored in the database should be automatically updated daily and store old data so that portfolios can track historical data.

3.2 Frontend

- Front end development will be written in C#, and SQL will be used for the backend. Various API's can be used to recieve data from internet sources.
- Investments must be able to be removed from the portfolio, as the user should have complete control over what they track.
- Portfolio level will allow the user to see a pie chart of their total assets with each section being an individual investment.
- The portfolio will have relative benchmarks to industry standard sources like the S&P500.
- An option to view the portfolio's history in a graph must be displayed with a variety of time intervals ranging from a day to weeks to multiple years.
- The portfolio should display the current Sharpe Ratio as well as its volatility.
- Individual investments should also track time data, Sharpe Ratio, Volatility, and relative benchmarks when relevant/available separate from the portfolio benchmarks.
- Users must be able to enter data through the application in two formats. First, one investment at a time manually added in using the user-friendly interface. Second by adding a CSV file or EXCEL file containing the user's portfolio data to the application will automatically populate the user's profile based off the contents of said file.
- Investments should list purchase price and quantity, at a minimum.
- Both the portfolio and individual investments should show returns based off time and purchase price.
- User's portfolio should be cached onto the device when the user logs in.
- Stock splits should properly increase investment quantity and reduce price when such event occurs (investment value should remain unchanged).

4 TIMELINE

Attached below

HedgeServ Investment Port...	start	end	0h	0%
Requirements	**12/06/17**	**05/19/18**	**0h**	**0%**
C# front end development	01/02	03/22	0	0%
SQL Database development	01/01	03/23	0	0%
1 second response time for database...	01/27	04/20	0	0%
If queries fail, data should either be b...	01/01	05/19	0	0%
Investments must not be hard coded	01/03	03/29	0	0%
Investments must have ability to add...	01/06	03/31	0	0%
Investments can be added in through...	01/11	03/23	0	0%
Frequency waits then queries	01/12	03/22	0	0%
Quandl API	12/06	04/25	0	0%
Measure app performance through r...	01/25	04/28	0	0%
Have data adjust for stock splits	02/01	04/04	0	0%
Security:passwords, client privacy, ...	01/25	04/14	0	0%
Drop down search for investment fe...	01/11	03/14	0	0%
Portfolio Summary: Time series plot o...	01/05	03/17	0	0%
investment entries will have: invest...	01/12	03/03	0	0%
allow for manual entry of data if not ...	01/19	03/18	0	0%
purchase price for trades	02/01	03/03	0	0%
Main benchmark: Show how portfolio...	01/23	03/23	0	0%
Calculate volatility at both investmen...	02/04	03/15	0	0%
Get returns using sharp ratio	01/25	02/28	0	0%
Get database to update every 15 mi...	02/13	03/23	0	0%
User portfolio should be cached when...	01/18	03/16	0	0%
Stretch Goals	**03/01/18**	**03/23/18**	**0h**	**0%**
Show related investments and how t...	03/01	03/23	0	0%
Related articles based on the perfor...	03/01	03/23	0	0%