



University of Glasgow | School of
Computing Science

Algorithmic Trading Tutor : Teaching the General Public about Financial Trading Algorithms.

James Michael Matthew Gallagher

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March 28, 2014

Abstract

This project focuses on creating a web application, AlgorithmicTradingTutor.com, which helps in teaching the general public about financial trading algorithms. The web application has been split into two segments : a learning centre where site visitors can access multiple forms of learning material, and an interactive game where user can pick from a selection of trading algorithms.. and match them against each other to see how they perform against historical market data. The web application also exposes an API which allows visitors to the site to upload their own algorithms, and compare their performance against others on the site.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Contents

1	Introduction and Motivations	1
1.1	Motivation	1
1.1.1	Aims	2
2	Literary Review and Background Research	3
2.0.2	An Introduction to of Financial Markets	3
2.0.3	Challenges in price discovery	4
2.0.4	Impact driven	4
2.1	Introduction to Automated Trading	4
2.1.1	Time Weighted Average Price Algorithm	5
2.1.2	Percentage of Volume Algorithm	5
2.2	Disasters and Regulation	5
2.2.1	Black Monday	5
2.2.2	Current Regulatory Status	5
2.3	Related Work	5
2.3.1	Investopdeia	5
2.3.2	Quantopian	6
2.3.3	QuantConnect	6
2.4	Web Based Learning	7
2.5	Problems with Gathering Materials	7
3	System Proposal	8
3.0.1	High Level Concept	8

3.0.2	Stakeholders	8
3.0.3	Functional Requirements	8
3.0.4	System Scope	9
3.0.5	System Actors and Use Cases	9
4	Implementation	10
4.1	Project Management	10
4.2	Tools	11
4.3	Development Cycles One : The TWAP and POV Trading Algorithms :	11
4.3.1	Google Finance	11
4.3.2	TimePoint Class and the Stock Class	13
4.3.3	Parser	13
4.3.4	Time Weighted Average Price Algorithm	14
4.3.5	buyTimes	14
4.3.6	The Time Weighted Average Price Algorithm	14
4.3.7	The Percentage of Volume Algorithm	15
4.4	Development Cycle Two : Implementing a Web Server	15
4.4.1	Prototype One: Spring MVC Web Framework	15
4.4.2	Prototype Two : Play Web Framework	16
4.4.3	how it was implemented	16
4.4.4	Displaying Algorithmic output.	17
4.4.5	Highstock.js	17
4.4.6	The Plays Template Engine.	18
4.4.7	Getting Data input from User's	18
4.5	Development Cycle Three : Upload Users Trading Algorithms	19
4.5.1	providing user to the site with the required files	19
4.5.2	displaying output of the uploaded algorithm	21
4.5.3	Retrospective	21
4.6	Development Cycle Four : Adding a Persistence layer	21

4.6.1	Implementation	22
4.6.2	Retrospective	25
4.7	Development Cycle Five : Designig a User Interface	25
4.7.1	Retrospective	28
5	Evaluation	29
5.1	Usability Evaluation	29
5.1.1	Think Aloud	30
5.1.2	Questionnaire	30
5.2	Changes Made to the Design of the Systems User Interface	32
5.2.1	Neilsens Heuristics	32
5.3	Unit Tests	32
6	Conclusion	33
6.1	Future Work	33
6.2	Conclusion	33
	Appendices	35
A	Evaluation Participant Materials	37
B	Generating Random Graphs	38

Chapter 1

Introduction and Motivations

1.1 Motivation

The rise of algorithmic trading over the last decade has become increasingly difficult for the general public to ignore.

Numerous high profile news stories, have captured the attention and interest of the general public. Stories such as the flash crash of 2010 where the Dow Jones Industrial average lost and regained (-dollar amount)9 percent of its total value in a single days trading, or the high profile losses of Knight Capital who lost 460 Million dollars over a period of 46 minutes, due to an error in their automated trading system.

Stories of huge losses, lighting fast trading and economic instability attract the worlds mainstream media to create attention grabbing headlines in the hope of selling papers.

However, a basic google search clearly shows the rise in prominence of the industry.

Statistic such as that between the period of 2005 and 2009 the average trade execution time on the New York Stock Exchange dropped from over ten seconds, to under one second*. Or a near threefold increase in the daily trading volume on New York Stock Exchange in the same period*, displays the increasing rise of automation in the worlds financial markets.

The rise of algorithmic trading in the last ten years has also captured the attention of the worlds largest governments.

The growing industry remains a hot topic within the European and British government, who continue to debate the costs and benefits of the practise, while trying to create appropriate legislation to encourage economic growth, while reducing market uncertainty.

The Challenging of European Unions draft law (MiFID II)* created to introduce tougher regulation of automated trading technologies, by the British governments FORSESIGHT* study which criticised the legislation, can only add to the publics curiosity in this subject.

Importantly, the role of the legislation created by these governing authorities has direct knock on effect to the quality of life of the general public. Any regions economic stability, has a direct impact on the population's employment rate, amount that governing bodies spends on public services and rate at which the general public can borrow money.

The shock of some of the worlds largest financial institutions teetering on bankruptcy during the 2008 Global financial crisis, and the shock waves which have been felt from this crisis in numerous first world countries such as Ireland, Greece and Spain has awoke the general public's interest in the need to firmly regulate the global financial markets, and the automated platforms that trade on them. (restructure keep content.)

Unfortunately, there is a distinct lack of accessible online teaching resources to learn about financial trading algorithms. This is partly due, to the multi paradigm nature of the field. However, there is also a clear lack of incentives for financial institutions to share their knowledge on the subject. Fears of disrupting profit margins, and giving away slight trading advantages in an ultra competitive trading environment, ensure the latest advances in the field are kept secret and unpublished.

AlgorithmTradingTutor.com has been designed to fill the void in accessible online teaching material in the field. The developed web app will present site visitors with accessible learning material on the subject, and take into account users preferred style of learning when presenting this multidisciplinary area of study.

The site is designed to teach the relevant information regarding computation, finance and market forces to site visitors with little or no previous experience in any of these topics. The web application also exposes an API which allows visitors to the site to upload their own algorithms, to provide interactive and dynamic learning environment for site visitors to interact with pre and custom made automated trading strategies.

1.1.1 Aims

The aim of this project is to develop a web application which improve the experience of learning about financial trading algorithms. The application focuses on harnessing well documented web based learning techniques to present this multidisciplinary area of study in an accessible and understandable manor.

The system can teach . . . in understanding the ...what they are trying to do .. and hoe they are implemented . The system also aims to harness the knowledge of those with experience in this interdisciplinary filed by allowing experts to upload trading alorithms to the site teach those no knowdge .

Research has been conducted in order to better understand the best way to present his interdiciplinary field to novices via a web based format. Research has also been carried out into the study of the potential threat computatinal finance posses to the worlds global financeial markets when they misbehave.

Allow users to test the knowledge they have gained based on the provided assessable learning material.

Chapter 2

Literary Review and Background Research

2.0.2 An Introduction to of Financial Markets

Financial markets facilitate the process of moving monetary funds between those who want to lend (invest), to those who want to borrow. Markets work by gathering many interested borrowers and investors in a single location, to make it easy for one to find the other, and assist in channelling funds to where they are likely to be most productive. A financial market is a place where borrowers and investors can trade funds -in the form of financial instruments- at a low transaction cost, and at a price which reflects supply and demand.

financial instrument are generally referred to as securities. Securities can differ in the legal obligations that are associated with them, in whether they can be sold in a secondary market and in the timings in which they pay out funds. Several different financials market exist, each of which are used to trade a specific type of security and each play an important role in the assistance of several every day business operations.

Capital markets allow business to raise money by issuing stock, or bonds, to willing investors. Broadly speaking the capital markers cans be categorised into the primary and secondary markets, based on the two stages of a securities life cycle. The primary market deals with the issuing of new financial instruemtnesn and subsequent trading of these securitirs takes places on the secondary markets.

Stocks (which represent a partial ownership in a corporation, and entitle the owner to a partial claim of a corporation assets and earnings.) and bonds (which represent a fixed interest loan between two parties. The full amount of the loan has to be paid back on a set date, and interest is paid out to the bond holder periodically.) are to of the most widely known financial instruments. The issuing of stocks and bonds provide a vital tool which allow business to raise money to expand their operations.

The futures markets allow business to transfer risk by issuing forwards contracts to willing investors. A forward contract is a contract between two parties to buy or to sell an asset at a specified future time at a price agreed upon on the date of the contracts creation. Future contracts allow sellers of goods to lock in profits, by agreeing today on a price that consumers will buy their product for in the future, irrespective of the future market price.

Financial regulators, such as the UK's Bank of England or the U.S Securities and Exchange Commission oversee the financial markets in their jurisdictions, and protect investors against potential fraudulent activity.

Several other types of financial markets exist which, again, are used to trade their own specific type of security(forwards/options/ swaps) .The previous metioned finacia instruments are out with the scope of this document due to the financial trading algorthms which are being taught by the developed web app being primarily focused on the equities market.

A financial market is a place where entities can trade financial assets at a low transaction cost, and at a price which reflects supply and demand. Markets work by gathering many interested buyer and sellers in a single location, and making it easier for one to find the other. Finance markets play an important role in the assistance of several every day business operations.

Capital markets allow business to raise money by issuing stock, or bonds, to willing investors.

The issuing of Stocks (which represent a partial ownership in a corporation, and entitle the owner to a partial claim of a corporation assets and earnings.) and Bonds (which represent a fixed interest loan between two parties. The full amount of the loan has to be paid back on a set date, and interest is paid out to the bond holder periodically.) provide a vital tool on allowing business to raise money to expand their operations.

The futures markets allow business to transfer risk by issuing forwards contracts to willing investors. A forward contract is a contract between two parties to buy or to sell an asset at a specified future time at a price agreed upon on the date of the contracts creation. Future contracts allow sellers of goods to lock in profits, by agreeing today on a price that consumers will buy their product for in the future, irrespective of the future market price.

Money markets allow business to increase or decrease their liquidity () by borrowing or loaning out money to other organisations. The money markets allow organisations with surplus funds to loan out capital, using a variety of financial instruments , to others in need of short term capital. a variety of different financial instruments exist bearing different maturities , levels of risk and structures to accommodate willing potential lenders

The commodities markets are used for the trade of numerous different types of primary, or non manufactured, materials. Commodities which are grown, (such as sugar, corn, coffee, coca) are commonly known soft commodities, and those which are mined (such as oil and gold) are called Hard commodities

2.0.3 Challenges in price discovery

2.0.4 Impact driven

2.1 Introduction to Automated Trading

The first generation of financial trading algorithms were natural evolutions of simple order slicing. These algorithms focused on meeting specific bench marks, and where designed to reduce the market impact of large orders. Algorithms such as such as time-weighted-average-price(twap) and vwap tended to be based on historic data, so as soon as an order was received a trading schedule could be determined. For example, for vwap this was based on a stocks historical market volume profile. Statically traded trading patterns increasingly proved to be vulnerable as other market participants could easily spot and take advantage of regular trading patterns.

To combat predatory trading algorithms started to incorporate elements of randomisation.

Consequently the natural progression form statically driven algorithms to more dynamic strategy. For example Pov bases its execution based on live market volume onstead of basing its execution on historical voulme profiling.

Due to first generation of algs not being designed to be price or risk sensitive, the seond gen of algs where designed to take this into account. These algs where designed to tackle and issued discussed in the . . . paper by . . .called the traders dilemma.. the dilma is : tradin to fast brings high martet impact, and trading too slow brings

2.1.1 Time Weighted Average Price Algorithm

2.1.2 Percentage of Volume Algorithm

2.2 Disasters and Regulation

2.2.1 Black Monday

2.2.2 Current Regulatory Status

2.3 Related Work

2.3.1 Investopdeia

Investopdeia.com is a web site devoted to providing free educational material on investing strategies, fundamental economics, personal finance and market analysis. The site sources articles from nearly 200 writers[1] and has extensive wiki and video based learning material and tutorials.

Investopdeia has over 6,300 articles and 750 pages of tutorials[1] covering trading fundamentals, technical indicators, trading strategies, popular trading software systems and numerous other aspects of finance and investing.

The site includes a comprehensive financial dictionary containing over 13,000 entries[1] to assist new users in understanding related technical financial terminology.

The site features a free multi user online stock simulator. The stock simulator provides an ideal learning environment for risk free participation in the stock market, giving users the ability to buy and sell equities and options at real market prices. The stock simulator is an effective tool for providing real-world experience to beginners learning fundamental trading strategies.

Investopedia have also released a mobile Application for the Phone, which is free to download . The App gives users access to Investopedia's full financial dictionary, site articles and other content. Investopedia's Iphone App has over 211,000 downloads and is the 25th free Finance App in the Itunes store.

The site articles and tutorials are an accessible and comprehensive entry point for the general public to learn about financial trading. The site maintains a strong focus on the field of finance, and avoids any form of educational material on computer programming.

Although the sites online stock simulator, allows users the ability to buy and sell equities in a safe environment, it does not allow for the creation or use of any basic automated trading strategys. This means visitors cannot learn about financial trading algorithms using the sites interactive trading environment.

The site provides great content on the general topic of finance, however a search of sites database of 6,300 articles for algorithmic trading returned only 24 articles, with only 8 articles being on the topic of algorithmic trading itself.

The site fails to present the subject in an accessible and un-intimidating manor, with phrases such as : system that utilizes very advanced mathematical models , Complex algorithms and used by large institutional investors being used in the site definition of the subject of algorithmic trading[2]

Unfortunately the site does not provide any place, such as a community forum, for visitors to the site to communicate and share ideas or knowledge with each other.

2.3.2 Quantopian

Quantopian is a browser based algorithmic trading platform. The site allows visitors to create, test, and trade algorithmic strategies with other user who visit the site.

The main way news user to the site are encourage to learn about financial trading algorithms, is by joining the community discussions and exploring algorithms that other members have shared on the site.

The site allows users to create algorithms in python, by providing a simple web editor for user to enter code into there browser. The site also allows user to copy or clone algorithms, which other members of the site have shared with the community.

Quantopian also allows users to back test their created algorithm against 11-year history of minute-level stock data, and provides a user interface for reviewing the performance of your algorithm against selected risk metrics - Maybe mention a few metrics ?

Once tested, the site provides a live trading platform to allow the use of created algorithms with real money. The live trading platform, which is currently in the process of being beta tested by members of the community, is set up by linking a brokerage account, to a Quantopian account on the site.

The site is feature rich, and has an active community of user who are open to helping new members who wish to learn. However, the site presumes a high level background knowledge in the fields of both finance and computer programming.

The site lacks a central repository of introductory material to enable new users to build up a basic knowledge on either of these fields. Unfortunately, the high level background knowledge required and high level level of technical language in use by the sites community does not make Quantopian easily accessible to the general public.

2.3.3 QuantConnect

Quantconnect is an alternative browser based algorithmic trading platform, which offers several of the same feature as the previously mentioned Quantopian. Quantconnect, again enables site visitors to create, test, and trade algorithmic strategies with other site members.

Quantconnect allows users to create algorithms in cSharp, by providing a feature rich IDE in the browser. The feature rich IDE allows user to create new projects from template algorithms, manage the directory structure of code by adding new files and folders, error syntax highlighting and autocompletion.

Quantconnect also enables users to back test their created algorithm against US equity data for over 16,000 stocks from January 1998 onwards. The site updates daily, to provide the previous days latest market data to all members.

Again the web app provides a user interface for reviewing the performance of your algorithm against selected risk metrics, and highlights key performance indicators. The site also provides a cSharp based library which allows a connection to be made to the Quantconnect severs, from you local machine. This lets user develop algorithms locally on their home computer, and then upload their compiled work to their QuantConnect account to be run and tested.

Quantconnect also provides crowd sentiment data to user through the use of the Estimote and Stockpulse services. The Estimote platform aggregates the financial analysis of a community 13,000 traders and investors to provide crowd sentiment data on possibilities of under/over pricing on US equities. The Stockpulse service provides twitter sentiment analysis, to identify moods, rumors, and market-moving trends and provide valuable trading ideas and signals on numerous globally traded financial assets.

The web application currently does not provide a live trading environment to allow users to use their created algorithms with real money. However, Quantconnect does advertise that such functionality is currently in active development.

Quantconnect is a feature rich browser based algorithmic trading platform. New users to the site are encouraged to learn about financial trading algorithms, by joining the community discussions and exploring algorithms that other members have shared. Quantconnects active community is however considerably smaller than the previously mentioned Quantopian.

Again, the site presumes a high level background knowledge in the fields of finance and programming in the cSharp language. Quantconnect lacks a central repository of introductory material to allow new users to build up a basic knowledge on either of these fields.

Unfortunately, the high level background knowledge required and the sites sparse active community does not make Quantconnect easily accessible to those lacking considerable background knowledge.

2.4 Web Based Learning

2.5 Problems with Gathering Materials

Chapter 3

System Proposal

This chapter outlines the concept of the proposed system and provides details of the proposed systems functional and non functional requirements.

3.0.1 High Level Concept

The systems concept is based on the identified need to provide a easily accessible tool to aid in teaching the general public about financial trading algorithms. This need has been identified via three two main resources.

Analysis of other tools : There are a limited amount of tool available aimed at aiding in the learning of financial trading algorithms. The tools which do exist are aimed at those who are knowledgeable in both the fields of computer science and finance. A application to aid in the teaching of financial trading algorithms to those with little background knowledge in the previously mentioned fields would be desirable.

Background research : background research mentioned in the previous section suggest that use of computer algorithms to trade financial securities is only going to increase in the coming years. With the increased automation of trading on the words. and the dangers that it can cause it is important the . in order to prevent future market crashes it is important to educate politicians / / those involved in both the financial and non-financial sectors and voters the majority on these systems

Communication with specialists: a final prototype of the web application was demod to two software engineering of the investment bank Jp Moragn , who expressed interest in the system.

3.0.2 Stakeholders

3.0.3 Functional Requirements

The functional requirements for the web application were captured using MoSCoW analysis. This software development technique provides a simple way of prioritizing system requirements and allows developers to focus on high priority requirements first. The method separates the software products system requirements into four distinct categories, each of which are explained below.

- **Must Have** : System Requirements critical to the success of the project. if one must have system requirements is not met at the end of the product developemt cycle the success of the project will be called

into question.

- Should Have : System Requirements important to the projects successs , however not completely critical . if one of these requiremetns are no the met the success of the project would not be called into question
- Could Have : requirements not necessary for the project to be a success, however the end product would benefit from their
- Won't have : Requirements which are out of the scope of the system during this phase of development. These requirements may be implemented in later future revision of the applications

Must Have

1 View algorithms : allow user to view and navigate through the algorithm available nio the web application

2 Run the TWAP algorithm : let users input their own chosen parameters into TWAP algorithm and view the results of the of algorithms calculation

2 Run the POV algorithm : allow users input their own chosen parameters into TWAP algorithm and view the results of the of algorithms calculation

3. View Stocks : view a selection of stock for the previously mentioned algorithms to run on

Should Have

1 Compare algorithms: select two available agroithms to be run in parallel so their out can be compared

2 upload algorithm : allows users to upload their own self defined algorithm, and for their output to be displayed on the site.

3 Run the VWAP algorithm : let users input their own chosen parameters into VWAP algorithm and view the results of the of algorithms calculation

3. search Stocks : Search selection of stock for the previously mentioned algorithms to run on

Could Have

1 Site Guide : User interface providing new visitors information on the systems features.

2. Web Portal: indexing useful information and relevant further reading for site visitors.

3.

Wont Have

1. In browser Algorithm Edited : The site does not allow hosted algorithms to be edited in browser, for them to be then be run with reevent changes ten shown in the browser window. 2. Algorithms operating on live data : . . . Hosted algorithms can not be used on live data streams

3.0.4 System Scope

3.0.5 System Actors and Use Cases

Chapter 4

Implementation

This chapter describes the implementation process and the software development methodology used during the implementation of the projects software product.

4.1 Project Management

Agile

The Agile Software Development Methodology was used to manage the code implantation stage of the project. The agile Methodology was chosen due to its emphasis on iterative and incremental software development cycles and the benefits associated with the method of project management. Although Agile Software Development practises come in several different varieties, the management of the project was carried out in accordance with the 12 principles behind the agile manifesto.

Information on where details of the 12 principles behind the agile manifesto can be found i the appendix [1]

The implementation of the project was broken down into six individual cycles :

- The first development cycles focused on implementing two of the financial trading algorithms that the web application make use of.
- The purpose of the second development cycles was to implement a suitable web server to host the two implemented financial trading algorithms.
- The third development cycles concentrated on implementing the "must have" system requirement which would site visitors to upload their own trading algorithms to be displayed on the system.
- The fifth development cycles aimed to create a suitable web based user interface for the web application to show all the functionality of the site to site visitors
- The purpose of the sixth development cycle was the implementation of a database back end to the web application. This persistence layer would store a selection of stocks, which s ite visitors could select for the sites implemented trading algorithms to operate on .
- The final development cycles occurred after feedback had been gathered from the first round of system usability user evaluations. The feedback gained from the first round user evaluations was used to make

design improvements to the systems user interface and Suggestions made by evaluation test subjects were incorporated into the web application user interface. The details of these changes will be covered in the following chapter, along information on the process of the systems evaluation.

Details of each development cycle follow bellow.

4.2 Tools

Version Control and ticket Engine

4.3 Development Cycles One : The TWAP and POV Trading Algorithms :

The aim of the first development cycle was the implementation of the two "must have" trading algorithms specified in the systems requirements specification: the time weighted average price algorithm, and the percentage of volume algorithm.

Before the implemented could start a decision had to be made regarding which programming language the algorithms should be implemented in.

The Java programming language was selected due to its type safety, large active community, extensive documentation, high quality development tools and previous background knowledge being held in the technology.

After an appropriate language had been selected, a data source of historic high quality stock data had to be found.

4.3.1 Google Finance

Finding a high quality data source of historic stock data turned out to be one of the more challenging aspects of the project. Data providing historic time, price and volume information on a stock is known as "tick" data. The required "tick" data which would provide information on stocks to the site would be judged on three requirements: resolution, correctness and price.

The most important requirement was resolution. Resolution refers to the time frequency that data is made available on a stock throughout the duration of the trading day. Common frequency include 5 minutes, 10 minute and 15 minutes intervals from market opens, until market close. It was decided "tick" data of a 1 minute time interval was required. This decision was based on two main factors

"Tick" data of a 1 minute time intervals would give visitors to the site added flexibility when selecting the starting and end times for operating algorithms. It was hoped that this added flexibility would increase interest in the system.

If "tick data of 5 minutes , 10 minute or 15 minutes was used, rounding would be required from the system if user selected a starting or end time for an algorithm which did not fall one of these specified time intervals. A user entering 9:47a.m as a start time for an algorithm into the system would have to be rounded up or down depending if 5 minutes or 10 minute tick data was being used. This functionality was also seen as a risk for potential off by one and rounding errors to enter the system.

Other requirements for the sites data source were that it must be error free, and free to access.

Several paid for high quality data sources of minute by minute resolution time are available online. [11] [10] [7] [?]. These data source unfortunately fall short of the third system data source requirement. Several free data sources such as . . . also exist which however do not offer tick data at the required minute by minute resolution required by the system.

Google Finance was selected due to it providing free, high quality 1 minute time interval tick data, and the service providing an accessible interface to interact with.

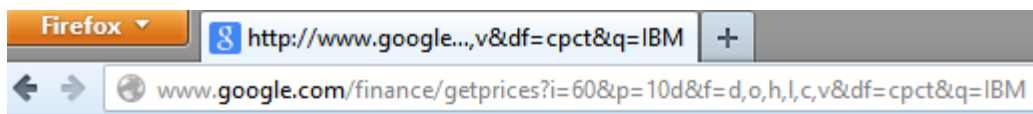
The service works by sending a URL request query to the service contain stock, date and frequency information of the stock required. The service then print out the requested information in a .csv file like format in the browser.

The URL format is:

```
http://www.google.com/finance/getprices?i=[PERIOD]&p=[DAYS]&d&f=d,o,h,l,c,v&df=cpct&q=[TICKER].
```

Where [PERIOD] specifies the resolution of the data to be returned in seconds. [DAYS] corresponds to amount of days, from today moving backwards that the data is needed for, and [TICKER] being the symbol of the stock requested.

An example url request query for 1 minute interval data on IBM for the last 10 days, and the returned .csv file are show below.



```

EXCHANGE%3DNYSE
MARKET_OPEN_MINUTE=570
MARKET_CLOSE_MINUTE=960
INTERVAL=60
COLUMNS=DATE,CLOSE,HIGH,LOW,OPEN,VOLUME
DATA=
TIMEZONE_OFFSET=-240
a1394458260,187.49,187.65,187.49,187.55,68613
1,187.61,187.62,187.47,187.5,13814
2,187.68,187.77,187.62,187.62,14239
3,187.76,187.78,187.65,187.68,9033
4,187.95,187.98,187.78,187.79,12257
5,188.054,188.091,187.99,187.99,12984
  
```

information gathered from the google finance service was gathered and stored in a .csv file.

After an appropriate data source had been found, a suitable data structure had to be created to store the data that the algorithm would operate on.

4.3.2 TimePoint Class and the Stock Class

A suitable data structure was required in order to store the time , volume and price information on an individual stock. A Class called TimePoint was created with these three instance variable, each to storing the previously mentioed values.

Once a class had been created to store the time , volume and price information of a stock at a particular point in time, a new class was created to store this set of information for the whole of the trading day.

The Stock class was created which contains 3 instance variables. The first a simple string instance variable to contain a stocks name, and the second and third as arraylists of the reviously mentioned timepoints class. Two arraylist were required.

The first named timeandprice, stores timepoint information for a stock for the entirety of a days trading. This array will act as the main data source for the sites trading algorithms. The sites algorithms will iterate over this arrayList, which will provided the input information to aid the individual calculations . . *****.

The second ArrayList of TimePoint values in the stock class was named purchased. The purchase arraylist is initially initilised as an empty arraylist. As each trading algorithm of the site operates , it role is to fill this array wich timepoint whihc represent the time , volume and price of when the algorithms specifics a purchase of the chosen stocks hould be made.

a uml diagram of the system follows below :

4.3.3 Parser

The next step in the implemmentation of the system was the creation of a parser. The systems parser reads in information from a csv file, created from google finance data, builds a stock class for the system algorithms to operate on.

A breif descriptionnn of the parser operation are as follows : The parser operates by creatig a scannner to readinput from file. A dummy date object is then created. The date object is set to the time of market open and a empty stock object is then instanciased . The createdscanner s then used to iterate over the input file one line at a time with price and volume information being extracted from it. This information is then used to instanciate a timepointclass, which is then added to the stocks classes timeandpricearray.

a code snippet from the systems parser can been viewed below :

```
public static Stock Parser(String file,String name) {
    Stock stock = null;
    try {
        Scanner fileIn = new Scanner(new File(file));
        fileIn.useDelimiter(",");
        DateFormat dateFormat = new SimpleDateFormat("HH:mm");
        Calendar cal = Calendar.getInstance();
        cal.set(Calendar.HOUR_OF_DAY,9); cal.set(Calendar.MINUTE,30);
        cal.set(Calendar.SECOND,0); cal.set(Calendar.MILLISECOND,0);

        stock = new Stock(name);
```

```

while (fileIn.hasNextLine() == true){
    String singleLine = fileIn.nextLine();
    String[] lineArray = singleLine.split(",");
    Float price = Float.parseFloat(lineArray[2]);
    Float volume = Float.parseFloat(lineArray[5]);
    TimePoint ts = new TimePoint(cal.getTime(),price,volume);
    cal.add(Calendar.MINUTE, 1);
    stock.setTimeandprice(ts);
}
} catch (FileNotFoundException e) {
    e.printStackTrace(); }
return stock;
}

```

4.3.4 Time Weighted Average Price Algorithm

The first algorithm to be implemented for the system to use was the time weighted average price algorithm. To explain the operation of the time weighted average price algorithm correctly, first two supporting methods to the algorithm must be introduced.

4.3.5 buyTimes

```
public static ArrayList<Date> buyTimes(Date s, Date e, int period);
```

The buyTimes method takes as input two Java Dates objects and an integer value. The first date object being the time selected by a user to start purchasing a stock, and the integer is the frequency specified by the user of how often they wish to make a purchase. These two values are used, along with the stock object, to create an ArrayList of date values representing a series of time points at which stock purchases are to be made. This ArrayList of purchase points is returned by the method.

4.3.6 The Time Weighted Average Price Algorithm

```
public static Stock TWAP(Stock s, Date start, Date end, int amount, int period)
```

The TWAP function takes as input two Java Dates, two integer values and one of the system's stock classes. The two Date objects are used as input to the previously mentioned buyTimes method, which returns an ArrayList of date values representing a series of time points at which stock purchases are to be made. The size of this array representing the total number of orders to be made is then used to divide the total amount of stock to be purchased, to find calculate the size of each individual order that the algorithm has to make. Once this has been calculated the price and time array of the stock object taken as input is iterated over to find price information for each time point matching times points specified in the ArrayList created by the buyTimes method. If a match exists between a time in the timeandprice array of the stock object and the buyTimes ArrayList, a TimePoint object is created containing Amount of stocks Purchased and the price paid for each and the time of purchase.

This process continues until the end of the buyTimes ArrayList has been reached. When the end of the buyTimes array list is hit, the algorithm returns the stock object it received as input, with a newly filled purchased array containing time, quantity and price information for all time points that the user has specified.

4.3.7 The Percentage of Volume Algorithm

```
public static Stock POV(Stock s, Date start, Date end, int amount, int volume)
```

The POV function takes as input two integer, one Stock object and a Java date Object . Again the two Date objects are used as input to the buyTimes method.

The Percentage of Volume Algorithm follows the market volume activity, until the specified quantity of stock has been purchased by the algorithm. The algorithm purchases a specific volume of market activity every minute until the transaction is complete. Therefore the input period of the used buyTimes function is a value of one.

Again the priceandtime array of the input stock object is iterated over until the first time held in the buyTimes array is found.

from this point onwards a percentage of that minute's market volume activity is purchased ,and used to create a timepoint object which will fill the input stock object's purchased array.

This process is continued until the amount of shares wish to be purchased is met.

The input Stock object with a filled purchased array instance variable is then returned by the algorithm .

how it effected the system : reflection

4.4 Development Cycle Two : Implementing a Web Server

The aim of second development cycle was to set up access to one of the implemented trading algorithms onto a suitable web server. In order for this to happen a suitable java based web framework had to be selected.

Two different java based web framework were selected for a prototype web server to be implemented. The pros and cons of the implementations of each framework were assessed in order to find out which framework would be most suitable to host the designed system . Details of each the prototypes follow below.

4.4.1 Prototype One: Spring MVC Web Framework

The Spring MVC web framework is currently the most widely used web framework in the java ecosystem. [5] Advocates of the framework state its frequent use of the inversion of control design pattern to reduced coupling and dependencies between system components as one of the framework main attractions [4] .

However critics of the framework cite complex systems configuration and excessive decoupling resulting in a lack of system coherence as downfalls of the framework . [3] It was selected to implement the first prototype of the systems web server due to wealth of available documentation and the framework being supported by a large active community. [6]

Other factors supporting this decision was the potential opportunity to use the frameworks high quality development tools such as the Spring Tool Suite which offered auto configuration of system dependencies via the dependency management tool maven, git version control integration and many other practical development functionality. [9]

A simple prototype system was built on top of the web frameworks Dispatcher -Servlet web Architecture

Further details of the first prototype system can be found in the appendix.

- The configuration of the web framework is heavily coupled with the popular dependency management and build automation tool Maven. When adding a new dependency to the project, Maven operates by fetching a secure and up to date version of the dependency from the build tools central repo. Significant difficulties experienced getting Maven to connect to the system central repository when inside the Glasgow University Eduroam network. This slowed down the development process of the first prototype significantly.
- The Spring MVC framework does not follow a standard 3 part model-view-controller system architecture pattern. The framework uses its own dispatcher-servlet system architecture pattern consisting of five components: a dispatcherServlet, HandlerMapper, View resolver view and controller. The additional components and added level of decoupling hindered comprehension of the system being built during the first prototype development.
- As mentioned above the majority of streamline Spring MVC project workflows are tightly coupled to the Maven build automation tool. This technology has a moderate learning curve, and learning to interact with this technology effectively was not seen as a priority to moving forward the progression of the project's development.
- IOC
- The base Spring framework gives a solid foundation for over 22 subprojects and

Due to the difficulties mentioned above, the Spring framework was rejected as a suitable choice for the system to be continued to be implemented upon.

4.4.2 Prototype Two : Play Web Framework

After the issues which arose during the creation of the first prototype of the system, a new Java based web framework had to be chosen as a suitable alternative for the system to be built upon. The Play web framework was selected as a suitable alternative.

The Play framework grew from Java developers seeking a web framework which was capable of rapid development cycles, similar to that of Ruby on Rails web framework. [8]

Advocates state the framework's low learning curve, minimal configuration and rapid development times as the framework's main appeals [4]. However, critics note the framework's immaturity and the fragmented nature of its development as negative aspects of the system. [2]

It was selected to implement a second prototype of the system's web server due to the following positive differences between it and the Spring framework :

removed the compile-deploy-retest cycle. You refresh your browser and the code is automatically compiled on the fly

Similar to the Spring framework Play also has a large resource of available documentation and is supported by a active community of developers

4.4.3 how it was implemented

User interface

4.4.4 Displaying Algorithmic output.

Once the both the TWAP and the POV algorithm had been successfully hosted on the play web server a front end to the web server had to be implemented to display each of the algorithm output.

Several different methods of creating a front end user interface to the algorithms output were looked into , including Apache Flex, d3.js and Google Charts.

JavaScript based options were given preference due to previous domain knowledge in the technology

After further assessment the Highstock.js JavaScript library was selected over the potential implementation options for the following reasons :

- Unlike Apache Flex, Highstock.js JavaScript library does not require users to have a Flash Player plugin installed in their browser. Apache Flex was also seen as an attractive option due to it be written in a superset of JavaScript, ActionScript 3 .
- despite The D3.js JavaScript library being a powerful and extendible visualisation tool, functionality to display both price and volume data on a time series graph could not . . without significant exertions .
- The Highstock.js JavaScript library was chosen over Google Charts due to the library having higher quality developer documentation and a more pleasing visual aesthetic.
- Highstock.js also offers additional integrated printing and download chart as a .jpg functionality unavailable in either d3.js or Google Charts

4.4.5 Highstock.js

Highstock.js operates on html content when html pages are loaded in the browser .

The Highstock.js library is initialised by defining a html element for the library to operate on .

```
<div id="container" style="height: 500px; min-width: 600px"></div>
```

The defined element can then be selected for library actions to be preformed on it . Actions include defining a input data set, labelling graph axis and specifying points of interest to be marked on the time series.

An example of the first two follow below :

```
<script>
$(function () {
  var inputData = [
    [1.3956534E12, 524.40],
    [1.39565346E12, 524.0],
    [1.39565352E12, 523.64]
  ];
  $('#container').highcharts('StockChart', {
    title: { text : 'TWAP Algorithm' },
    yAxis:[{ title :{text: 'Price' },
      height : 200,
      lineWidth: 2 },
    { title : { text: 'Volume' },
      top : 300,
```

```

        height : 100,
      }],
      series: [{ name: 'stock1', data: inputData } ]
    });
  });
</script>

```

As shown above the library takes a javascript array of time/price tuples as input.

The sample above showing the librarys input being hard coded into an html exists only to aid the explanation of the operation of the library. In the implemented system the highstock.js library is given its input by using plays scala-based template engine.

4.4.6 The Plays Template Engine.

A Play Scala template is a simple text file, that contains small blocks of Scala code. Templates are compiled as standard Scala functions, following a simple naming convention. The Scala template uses @ as the single special character. Every time this character is encountered, it indicates the begining of a Scala statement. A template can be though of as simply a function, so it requires parameters, which are declared on the first line of the template file.

The stock obejct provided as output from either of the systems trading algorithms , was passed as an input parameter into a scala template.

Inside the scala template, a function was created to iterate over the stock objects Array of time/price values which were inserted into the file. It was these values which were then used input to the highstock.js library.

An example of the created scala template use to provide input to the systems highstock.js library is shown below :

```

@(Stock: models.Stock )
@for(p <- Stock.getDataArray){
    [ @p.getMiliTime , @p.getPrice],
}

```

4.4.7 Getting Data input from User's

The Play framework contains several helper packages to handle HTTP submission and form validation . Form are created submissions are handled by defining a play form structure, and a method to be calleed upon the forms submission is specified in the form structures definition.

```

@form(action = routes.Application.TWAPForm, args = 'id -> "twap") {
  @inputText(
    field = TWAPForm("startHour"),
    args = ' __label -> "Start Time Hour?", 'placeholder -> "10" )
    <p class="buttons"> <input type="submit"> <p>
  }
}

```

4.5 Development Cycle Three : Upload Users Trading Algorithms

Allowing site visitors to upload their own trading algorithm to be displayed by the system, was the aim of the third development cycle. Numerous different implementation methods were considered when deciding the most viable way to implement this piece of system functionality.

The decided workflow for this piece of site functionality is as follows:

- Site visitors are given a copy of the two main classes that the site operates on: the Stock and the timepoint class.
- a working implementation of the sites Time weighted average price algorithm is provided inside the stock class. This algorithm is given to teach users the way algorithms running on the site take as input, and provide as output, a copy of the Stock Class. User are free to edit the inner workings of this method however they please.
- Once visitor have edited the operation of the provide example algorithm they are required to compile the two provide class in to Java .Jar file.
- Once in possession of the required .jar file, users can navigate to the upload algorithm interface provide by the system. Here user can submit their algorithm to be displayed on the system.

The implementation of this must have” system requirement was broken into three smaller subtasks: Providing user with the required system files/ the creation of an upload algorithm user interface on the web server, and displaying the output of the user uploaded algorithm. Implementation details for each follow below.

4.5.1 providing user to the site with the required files

In order to upload their own algorithm to be displayed by the site, users require a copy of the timepoint and stock classes used by the web application. A number of possible options were considered when deciding the best way to provide user with these file , including : letting site visitors download the files as a compressed package/ displaying the files as an html page to be copied / or displaying the files in a web based editor.

The browser based editor Ace was chosen over the other alternatives for the follow reasons:

- At first providing the needed files as a compressed file format (ZIP, RAR) was seen as an attractive option, due to this method reducing the total amount of bandwidth needed by visitors to gain access to these the files. This could be seen to be preferred by user who have access to limited bandwidth (connect via 3G) or who have low quality internet connections. This method does however require users to download a directory to gain access to the files. This could be as seen as an unattractive option to users who access the site on hardware limited devices (Mobile Phones/ Tablets). The other two options mentioned do not require visitors to download a directory to access the required files.
- This method above was eventual rejected however due to it requiring users to have additional external decompression software installed to access the required file. Again, the other two options mentioned do not have this drawback.
- A browser based editor was given preference over displaying the files as an html page due to this method allowing user to edit and change the displayed code which is one screen. This was hoped to increase users interest and comprehension of the required code...

- A browser based editor also offered automatic syntax highlighting, which again was hoped to increase visitors comprehension of the code.

The browser based JavaScript editor ace operates in an identical manor to the previously mentioned high-stock.js. The library is initialised by defining a html element for the library to operate on. The defined element is then selected for library actions to be performed on it. An example of the created user interface is shown below:



```

1
2 public class Stock {
3
4     private String name;
5     private ArrayList timeandprice;
6     private ArrayList purchased;
7
8     public Stock(String s ){
9         this.name = s;
10        this.timeandprice = new ArrayList();
11        this.purchased = new ArrayList();
12    }
13
14    public String getName() {
15        return name;
16    }
17

```

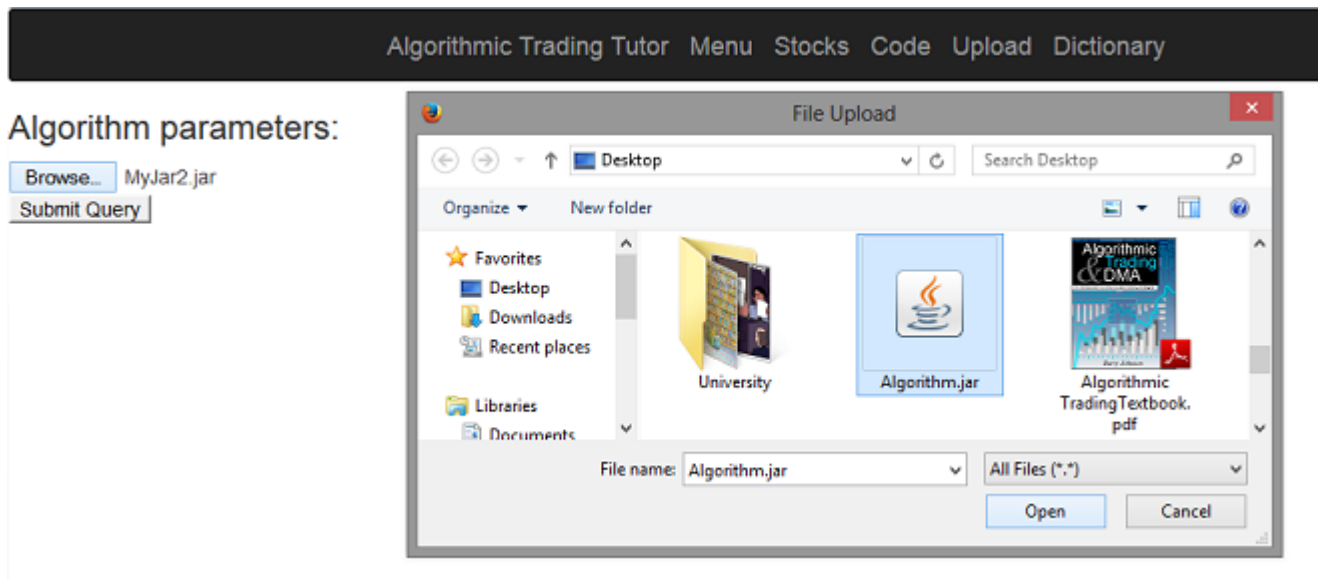
User are encourage to take a copy of this code, and to edit it in an IDE or text editor of their preference.

the creation of an upload interface on the web serve

The creation of an upload interface to allow user to submit their modified algorithm to the site was the simplest section in implementing this peace of functionality for the system

A minimal HTML page was created contain two action buttons. The first labelled "browse" opens a window displaying the site visitors systems desktop directory. From here user can navigate to the location of their compiled jar file.

The created user interface is shown below :



Once the Jar file is found the file can be submitted to the system in a single click.

4.5.2 displaying output of the uploaded algorithm

Once the users Jar file has been uploaded , it needed to be added the the systems class path

an out line of the this process is given below :

- A java ClassLoader is used to load the stock and timepoint class contained in the uploaded jar file onto the systems class path.
- an instance of the newly loaded time point class is instantiated. this is used with a date object, set to the time of market open , and a default input file to instantiated a copy of the newly loaded stock class.
- Once the newly loaded in stock class has been created, the classes . .
- Once the user modified algorithm has been run , the loaded in stock class contains a filled purchased array containing time and quantity information of when the user defined algorithm wishes to make purchases. The loaded in stock class however cannot be used to interact with any of the systems defined functions due to possible security conflicts . A stock object which is instantiated form the running web application must be instantiated , and the values from the two timepoint arrays of the JAR originated stock class are copied into this newly instantiated identical class.
- The purchased array of the newly instantiated stock class, which originates from the running web application is then used as output information to be input into a Scala play template and displayed on the system via highstocks.js

4.5.3 Retrospective

4.6 Developent Cycle Four : Adding a Persistance layer

The aim of the fourth development cycle was the addition of a database to the system. This persistence layer would store stock information,for the sites trading algorithms to take as input.

Visitors to the site are encourage to select a stock they are interest in for the site trading algorithms to operate on. It was hope that giving site visitors a range of possible stock to pick from would encourage interest in the system.

Multiple different implementation options were available to complete the implementation of this task. After an analysis of numerous available technoloies the H2 relational database management system (RDMS) in combination with an Ebean object relational mapper (ORM) were selected.

H2 is a java based, open source RDMS which offers in-memory and disk-based persistance functionality. The application also offers both embedded and server based deployment functionality.

Ebean ORM was designed to be simpler to use and understand than the Java Persistence API (JPI). It achieves this through its 'Session Less' architecture. Ebean does not require a JPA EntityManager or JDO PersistenceManager which removes the concepts of detached and attached java beans. Advocates of the tehcnology report the systems 'Session Less' architecture make Ebean much easier to learn understand and use than JPI.

The reasoning behind these two technologies been chosen to implementat the system persistance layer were as follows :

- An in-memory, embedded deployment of the h2 database was seen to provided numerous technical advantages over alternative implemetation methods. An embedded deployment allows minimal dependeces to be created, as this method does not attach the web application persistance layer to any external systems.
- An in memory deployment of the H2 DBMS offers system performance gains assosiated with the removal of repetitive system disk access's .
- Simplified Ebean ORM architecture was hoped to decrease development time and increase system comprehension.
- Play web framework comes out of the box with H2 DBMS and the Ebean ORM drivers installed. The minimal configuration requiredwas again hoped to decrease the length of the development cycle.
- The documentation for integration the H2 DBMS and the Ebean ORM with the Play web framework was extensive and detailed.

4.6.1 Implementation

The implementation of the system persistance layer was broken into four smaller subtasks : Defining a database schema and creating a population script/Defining an eBean ORM model/Creating a user interface to the implemented database. Implementation details for each follow below :

Defining the H2 database schema and creating a database population script

The Play Framework creates and popultes an embedded in memory H2 database by using a sql "evolution" script. These scripts are used to define the databases schema and then populate it with temporary data values. The sql "evolution" script used for defining the systems databse schema , and an example of the sql script used to insert a temporary data point are shown below :

```
//Database schema definition script
```

```

create table SystemTable (
  id                bigint not null,
  stock_name        varchar(255),
  stock_ticker       varchar(255),
  market_cap        varchar(255),
  stock_sector       varchar(255),
  data_date          varchar(255),

  constraint pk_dbrow primary key (id)
);

```

```
//Database population script
```

```

insert into SystemTable
  (id,stock_name,stock_ticker,market_cap,stock_sector,data_date) values (
  1,'Coca-Cola','KO', '$169,190 Million', 'Consumer Goods','10th December
  2013');

```

As can be seen above the schema definition script is used to define a database table name, the tables collums name and types and a table primary key.

Several addition columns were created to display additional information about stocks stored in the system, again to promote interest in this peice of system funcationality. seing as these cums The were not going to be used for calculation at any point durring the system operation it was decided that impleanted these collums as types varchar was acceptable .

Defining an eBean ORM model

An ebean model was created to map the table in the H2 database to a java class, which coul then be instanciaded and made use of by the system. The Play framework provides a convenient superclass `play.db.ebean.Model`, that tables can extended.

```

import java.util.*;
import javax.persistence.*;
import play.db.ebean.*;
import play.data.validation.*;
import com.avaje.ebean.*;

public class SystemTable extends Model {

  private static final long serialVersionUID = 1L;

  @Id
  public Long id;

  @Constraints.Required
  public String stock_name;

  @Constraints.Required
  public String stock_ticker;

```

```

    @Constraints.Required
    public String market_cap;

    @Constraints.Required
    public String stock_sector;

    @Constraints.Required
    public String data_date;

    /**
     * Generic query helper for entity SystemTable with id Long
     */
    public static Model.Finder<Long, SystemTable> find = new
        Model.Finder<Long, SystemTable>(Long.class, SystemTable.class);

    public static SystemTable findById(long id) {
        return find.byId(id);
    }

    public static List<SystemTable> findAll() {
        return find.all();
    }
}

```

As seen above a static field called find is added, taking input of type SystemTable and a Long identifier. This helper field is used to query the model which has been defined.

Creating a User Interface

The open source javascript library DataTables was used to create a user interface to the web application database. DataTables offers several pieces of additional interactive functionality to plain HTML tables. This functionality includes: Variable table length pagination/On-the-fly filtering, Multi-column sorting and auto handling column widths.

The javascript DataTables library operates in an identical manner to the previously mentioned highstock.js. The library is initialised by defining a html table for the library to operate on. The defined table is then selected for library actions to be performed on it. An example of the created user interface is shown below:

Step 1. : Pick a Stock of Interest.

Select a company of interest from one on the sites database.

Show entries Search:

Company	Ticker	Market Capitalization	Business Sector	Date
Coca-Cola	KO	\$169,190 Million	Consumer Goods	10th December 2013
Microsoft	MSFT	\$325,940 Million	Technology	11th December 2013
MorganStanley	MS	\$219,860 Million	Financial Services	12th December 2013

Showing 1 to 3 of 3 entries ◀ Previous Next ▶

4.6.2 Retrospective

4.7 Development Cycle Five : Designing a User Interface

The aim of the fifth development cycle was to create a suitable user interface (UI) for the system. The system user interface had to take into account the skills, experience and expectations of anticipated users.

The development cycle was seen as incredibly important as research has shown users often judge a system by its interface rather than its functionality [1]. Due to the system being a teaching tool it was important that the site's interface as easy to learn and could provide a level of familiarity after a period of minimal interaction.

The Twitter Bootstrap front-end web framework was chosen to assist in the creation of the system's UI. The web framework provides HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components to assist in the creation of web-based UI's. Twitter Bootstrap was selected over other alternative front-end web frameworks due to previous domain knowledge in the technology.

The creation of the web application's UI was broken into two smaller subtasks: Creating a welcome screen /creating a Menu screen/ and creating a dictionary. Implementation details for each follow below:

Welcome Screen

The welcome screen's main requirement was to showcase all of the functionality offered by the web application to site visitors. Due to the system being a teaching tool, it was understood that users may not be aware of the actual functionality offered by the system.

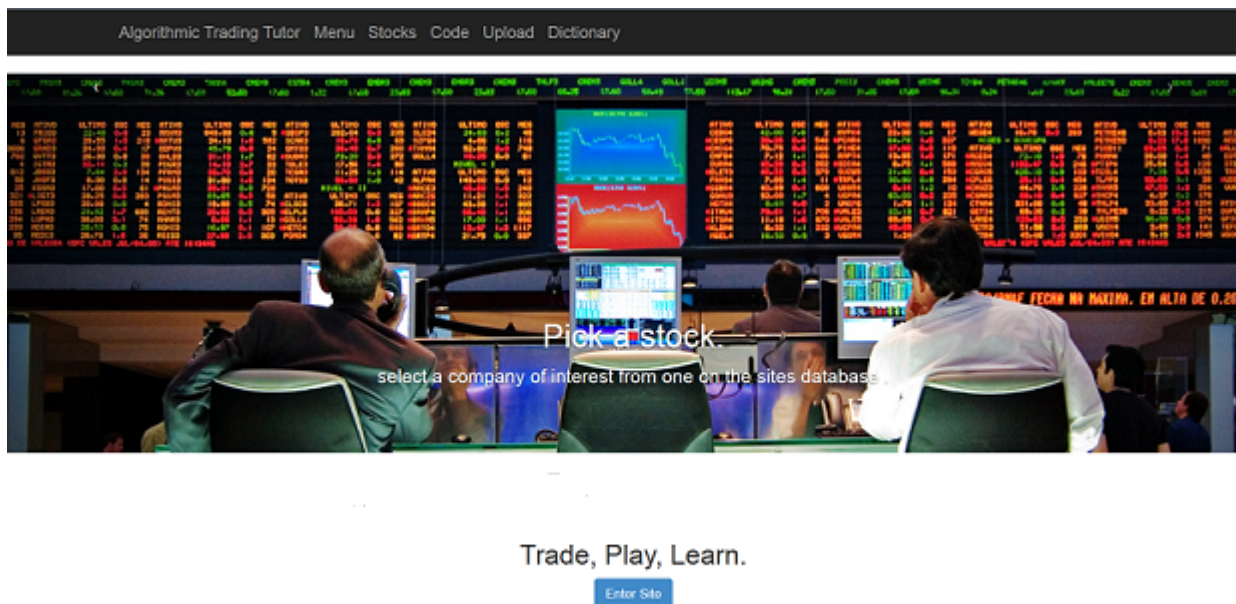
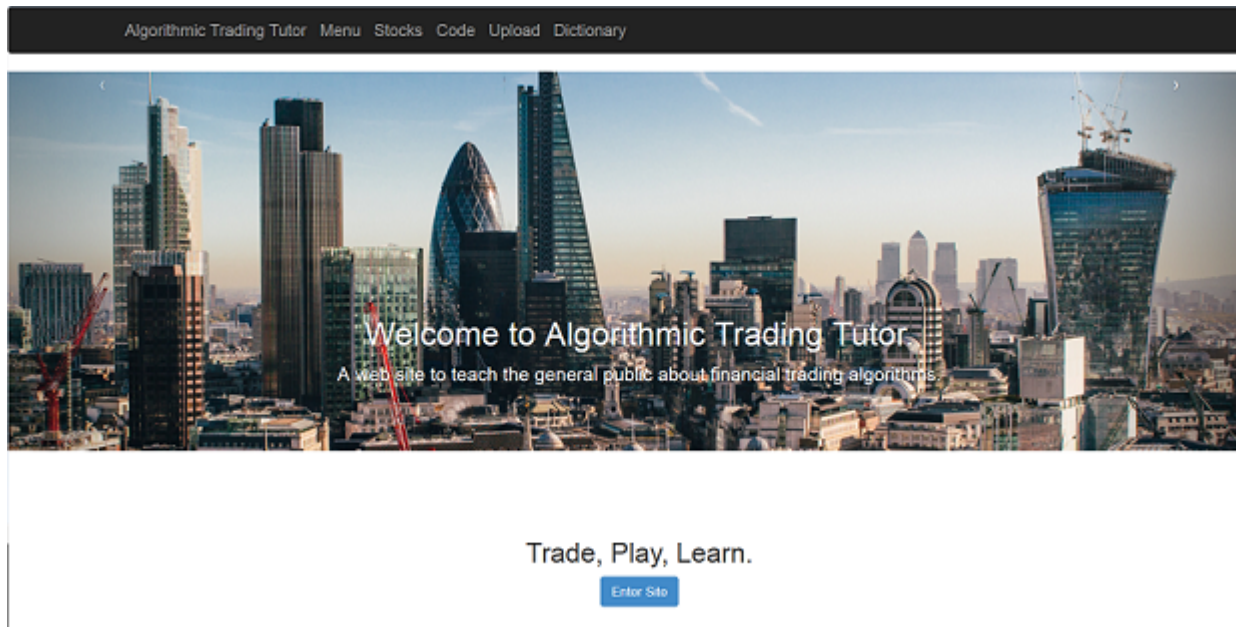
It was decided that large images were used to draw users' attention to the functionality of the system.

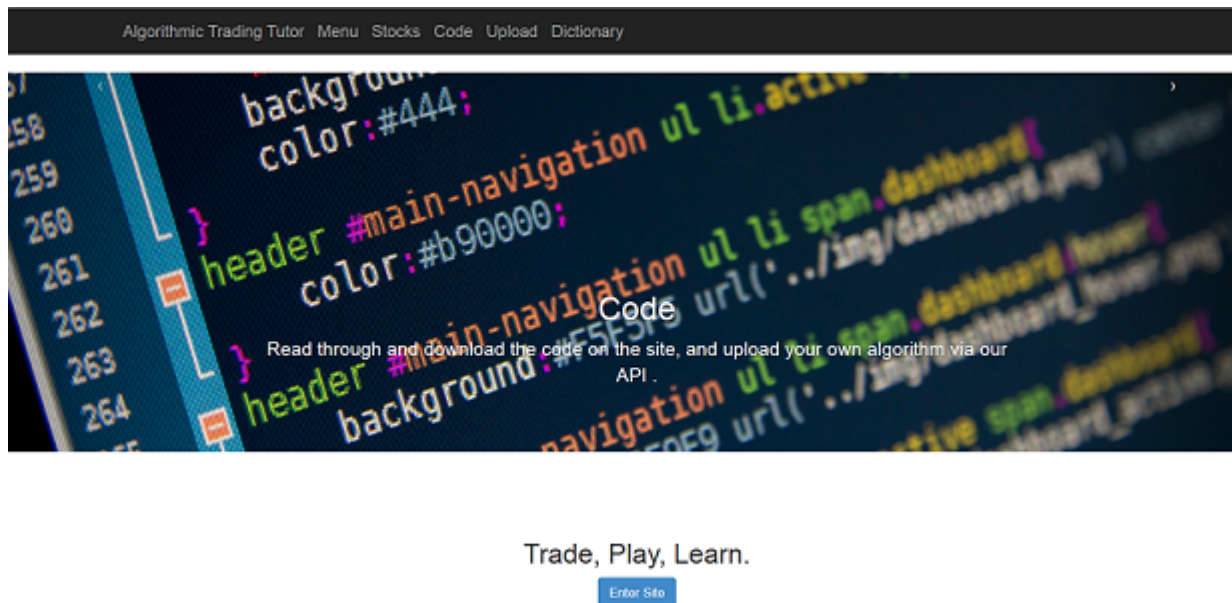
Bootstrap's carousel template was used to provide a slide show, displaying each piece of the web application's functionality supported by an explanatory image. The slide show was set to autoplay, which provided a great way to display all of the site's features with minimal input from users. Each slide features a brief description of the

current piece of functionality being shown. the use of technical language in each slide was kept to a minimum in order to not intimidate site visitors.

It was important that site would also accommodate more experienced users. a menu bar was created at the top of this screen to offer quick access to key site features, and to decrease the navigation time of experienced users around the site

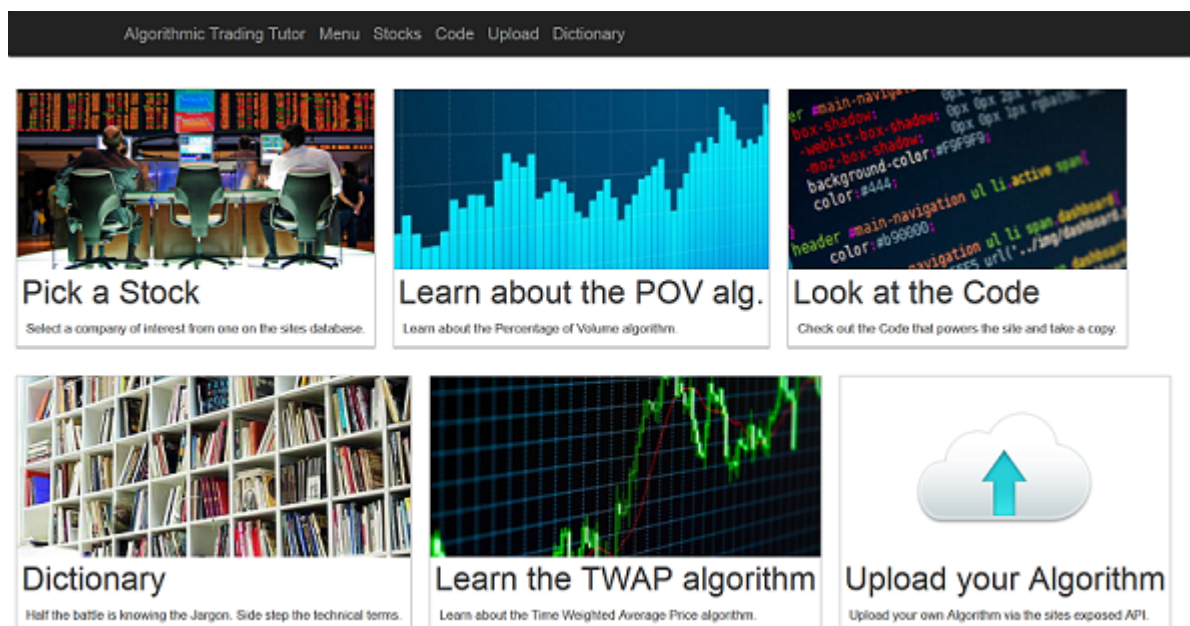
A selection of example slides used in the welcome screen can be seen below :





Menu Screen

The menu screen again is required to display all of the functionality that the web application offers. Due to the system being a teaching tool it was understood that user may not be aware of what functionality is offered by the system. Again, large images were used to capture the users attention, and the use of technical language was kept to a minimum in order to not confuse or intimidate site visitors. The same image used in the sites welcome screen were used again in this screen to follow user interface best practices of Consistency, and to increase site visitors levels of familiarity with individual images and pieces of the site functionality. Each image operates as a large button which takes users to the displayed piece of the site functionality. The image like buttons were chosen to increase visibility of the site's functionality, and to decrease possible selection errors (miss-clicks). The menu screen again features a top of the screen menu bar to accommodate for more experienced site visitors.

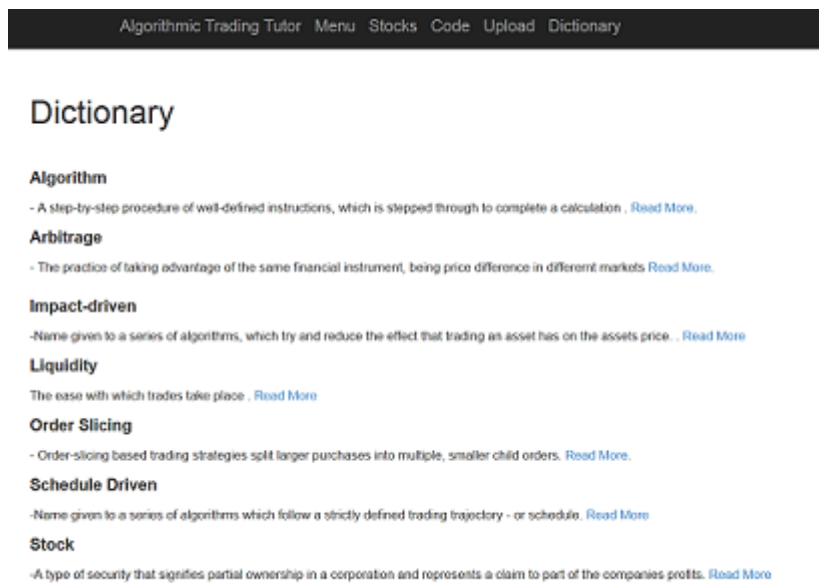


Dictionary Screen

A Dictionary of technical terms is used to increase the site's usefulness as a teaching tool. This requirement was inspired by the dictionary of financial terms offered by Investopedia seen in the documents related work section. The dictionary of algorithmic trading technical terms is another piece of functionality that makes this project an original piece of work. No other example of a dictionary of algorithmic trading terms aimed at novices has been able to be found in the public domain.

The screen was presented in a minimalist fashion in order to not distract users for the information being presented. The language used to explain the technical terms of the site's problem domain was aimed to be concise in order to aid comprehension. It was also ensured that the language used was consistent with terminology used in the rest of the web application to again aid comprehension.

an example of this screen is shown below :



4.7.1 Retrospective

.. .

The seven development cycles occurred after the first round of user evaluations. The feedback gained from the first round of user evaluations was used to make design improvements to the system's user interface.

Chapter 5

Evaluation

5.1 Usability Evaluation

After the implementation of the system has been completed a thorough user evaluation was conducted. The user evaluation functioned as a reliable way to test the usability and likeability of the system that had been created.

All usability testing performed by the team was done in accordance with the University of Glasgow ethics procedures. A copy of the University of Glasgow School of Computer Science Ethics check-list can be found at ++FILL IN ++

Two separate iterations of the evaluation stage were performed. This was so that feedback gained from the first round of the system evaluation could be incorporated into the web application design, and so the changes made could be then test on gauge improvement with regards to the usability of the system.

Each iteration of the evaluation process consisted of three parts: participant brief; think-aloud, and questionnaire. Each of which will be discussed below.

Each iteration of the evaluation process was conducted with a data set of 4 test subjects. The small data is a reflection of the time required to complete each evaluation, and each evaluation prioritised focusing on testing the depth of the knowledge that each test subject gained when interacting with the system.

Participant Brief

Each user evaluation started with an introductory briefing.

The conductor of the evaluation introduced each test participant to each of the three stages of the user evaluation which they were about to complete.

Test subject were then provided with a test number, a copy of the evaluation materials which they would require, and were then given a short introduction to the web application which described the aims and motivations behind the system that had been created.

During this stage the participant was asked to answer a few simple questions to gauge their competency using web based applications, and to gauge their personal interest and background experience in the systems domain.

During the introductory brief, it was made clear to the participant that no personal or identifying information would be collected from them during the user evaluation. It was hoped that this would hopefully decrease the number of participants who would not complete the evaluation fully, and to hopefully ease the process of gaining ethical permissions from the University. Due to the fact that no test participants decided to stop the evaluation half way through, and gaining ethical approval for the user evaluation was a simple process, it was felt like this was a beneficial decision.

In accordance with the University's ethical procedures, during the introductory briefing the test participants were reminded of their right to stop the evaluation at any time with no requirement to give reason. The participant was then further reminded that it was not them, but the system that was under evaluation. The participant was also provided with the contact details of the conductor of the evaluation, to allow them to contact the evaluator to answer any questions or give any thoughts that they had about the system or the user evaluation.

5.1.1 Think Aloud

The evaluation was performed using the Think-Aloud technique. The test participants were encouraged to talk out loud as they performed a series of tasks, designed to provide a full overview of the complete functionality provided by the system.

At this stage the evaluator conducting the evaluation observed each test candidates interaction with the system, and took note of any hesitation, possible confusion, or errors encountered when using the web application. The reactions shown by the test participant when interacting with the web application clearly highlighted usability problems which went unnoticed in the initial system design.

A copy of the Task List which was given to each test participant can be found in Appendix.

5.1.2 Questionnaire

The final part of the user evaluation asked the test candidate to complete a feedback questionnaire. This document asked them to rate their interest in the applications after their initial experience using the system. At this stage the test candidates were also given the opportunity to ask any further questions about the each of the systems. After being thanked for their time and made aware of the tests completion, every participant was encouraged to get in contact if they had any further thoughts they wished to add on the system, after having some time to think about the evaluation process.

A copy of the Questionnaire can be found in Appendix

Evaluation Results

Each user evaluation effectively communicated numerous positive aspects of the design of the web Application, and also shed light on aspects of the system which needed revising.

The feedback gained from the evaluation was mostly positive, with many test subjects commenting on the applications pleasant interface and minimal aesthetic. Numerous test subjects did however leave constructive criticisms on aspects of the system, mainly concentrating on how the application summarized output information for each of the systems algorithms .

Usability issues gained from the first iteration of the evaluation process were :

- When asked to explain technical terminology relating to the TWAP algorithm, 3 test subjects failed to notice the dictionary functionality located in the top of the screen menu bar. Test subjects however had no problem locating "upload" and "pick stock" functionality of the web application located in the same location when required to. When asked to answer question 2.b on the evaluation task all 4 test subjects based their answers on information displayed only on the TWAP algorithms page, and all subjects showed hesitation answering this question.
- Subjects responded positively to provided summary output of the TWAP algorithm. After being given a summary of the TWAP algorithm it was presumed subjects could extrapolate the required information necessary to answer questions 5.a and 5.b of the task sheet when using the sites POV algorithm. Initially no summary information was provided for the POV algorithms output. After being provided with summary information for the TWAP algorithm, user responded negatively when no summary information was provided for the output of the POV algorithm. It appeared that when users were provided with this initial functionality, they expected it to be replicated across all algorithms on the site.
- Subjects failed to make use of the provided zoom in functionality when viewing the displayed output of either the TWAP or POV algorithms. This functionality is of a high importance when aiming to understand the operation of the POV and relating the quantities of stock purchased by the algorithm to the overall volume of market activity of the stock being purchased.
- Subjects hesitated and displayed confusion over the use of ambiguous language when asked to provide input to one of the systems algorithms. On the completion of task 5.a (use the POV algorithm to make a purchase) the description of one of the algorithms input fields was thought to be ambiguous by two test subjects. The label "enter min left test subjects unsure if the field required a set time value of minutes , or a minimum value of shares to purchase. Subjects displayed confusion upon the completion of question 5(- make use the POv algorithm)
- Subjects hesitated and displayed confusion over the use of alternative, equivalent meaning, technical terminology used to describe the functionality of the TWAP algorithm. On the completion of task 2.a (explain technical terminology relating to the TWAP algorithm) the description of the operation of the algorithm used two equivalent meaning technical terms to describe the algorithms operation. This led to the confusion of two test subjects, and this led to this question being the most frequently incorrectly answered question on the task sheet.
- Even with a small sample size of just four test participants, users attention span varied wildly. One user gave up on one of the tasks (Task 2.b - explain terminology related to the TWAP algorithm .) within 20 seconds, after failing to complete the task on their first attempt. Another, spent over 90 seconds and trying to answer this question.
- Only two of the four test subjects partaking in the user evaluation managed to complete all five test tasks correctly. However, all five test users rated themselves either 'Very Confident' or 'Confident' when asked to rate competency using web applications, on a five point Likert scale.

A number of small bugs were also found during the first iteration of the usability evaluations. These system bugs were corrected before the start of the second round of evaluations.

The bugs were :

- An unused action button had been left in on the third slide, in the slide show of the sites welcome page. This button was removed.
- The menu bar at the top of the screen had not been added to the sites "select stock" page. The menu bar was added to this screen.

- The sizing of the background image of the third slide in the sites welcome page slide show was being displayed at an incorrect ratio in relation to the rest of the screen. The ratio of the displayed image was corrected.
- A typo in the description of functionality of the TWAP algorithm was found. This was removed.

5.2 Changes Made to the Design of the Systems User Interface

The feedback gained from the first round of user evaluations were used to make design improvements to the systems user interface .

Suggestions made by the evaluation test subjects were incorporated into the application user interface.

The Changes made to User Interface based on the feednack gained were as follows :

- A summary output of the operation of the POV algorithm
After test subjects responded negatively to one of the sites algorithm which provided a short summaryof its output output of the TWAP algorithm.
- embedded links
- coockie trail
- A typo in the description of functionality of the TWAP algorithm was found. This was removed.

A second round of usability test took place to gauge the effectiveness of these suggested improvement on the usability of the system.

5.2.1 Neilsens Heuristics

5.3 Unit Tests

Chapter 6

Conclusion

6.1 Future Work

The web application offers a number of potential opportunities for extension as part of future work.

Allow Site visitors to upload stock information to the web applications database

Display a Live Data feed for price information of stock on the web applications database

The Creation of Site Community Forum

joining the community discussions and exploring algorithms that other members have shared on the site.

Editing Site Hosted Algorithms in the Browser

The Use of Site Hosted Algorithms on Live Data Streams

6.2 Conclusion

```
> pdflatex example0
> bibtex example0
> pdflatex example0
> pdflatex example0
```

Bibliography

- [1] Agile Manifesto <http://agilemanifesto.org/principles.html>.
- [2] Brikman <http://www.quora.com/play-framework/what-are-the-pros-and-cons-of-play-framework-2-for-a-scala-developer>.
- [3] BRODWALL <http://java.dzone.com/articles/why-i-stopped-using-spring>.
- [4] FOWLER <http://martinfowler.com/articles/injection.html>.
- [5] MAPLE <http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/>.
- [6] MUGGA <http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/5>.
- [7] olsendata <http://www.olsendata.com>.
- [8] Play Documentation <http://www.playframework.com/documentation/1.1.1/faq>.
- [9] Spring Tool Suite <http://spring.io/tools/sts>.
- [10] tickdata <http://www.tickdata.com/>.
- [11] tickdatamarket <http://www.tickdatamarket.com/>.

Appendices

user interested in precise information or data relationships? Volume Info ! POV

Appendix A

Evaluation Participant Materials

An example of running from the command line is as follows:

```
> java MaxClique BBMC1 brock200_1.clq 14400
```

This will apply *BBMC* with *style* = 1 to the first brock200 DIMACS instance allowing 14400 seconds of cpu time.

Appendix B

Generating Random Graphs

We generate Erdős-Rényi random graphs $G(n, p)$ where n is the number of vertices and each edge is included in the graph with probability p independent from every other edge. It produces a random graph in DIMACS format with vertices numbered 1 to n inclusive. It can be run from the command line as follows to produce a clq file

```
> java RandomGraph 100 0.9 > 100-90-00.clq
```