

Criterion B

Contents:

Design Approach	2
Design of solution:	2
Initial paper designs:	2
Use case diagram:	3
Explanations of Use Cases:	3
System flow charts:	5
UML class interaction diagram:	6
Algorithm Flow Charts:	7
Data structures:	11
Test Plan:	14
Bibliography:	17

Design Approach

The product will be designed in a bottom-up approach. Each part of the program will be designed using basic data structures and low complexity, to make sure that they have the primitive functionalities, and then they will be integrated together using more complex data structures. In order for the design to be up-to-date with the client, evolutionary programming will be used to constantly inform the client on new changes, where feedback can be provided.

Design of solution:

Initial paper designs:



Figure 1: Initial paper design of product

Use case diagram:

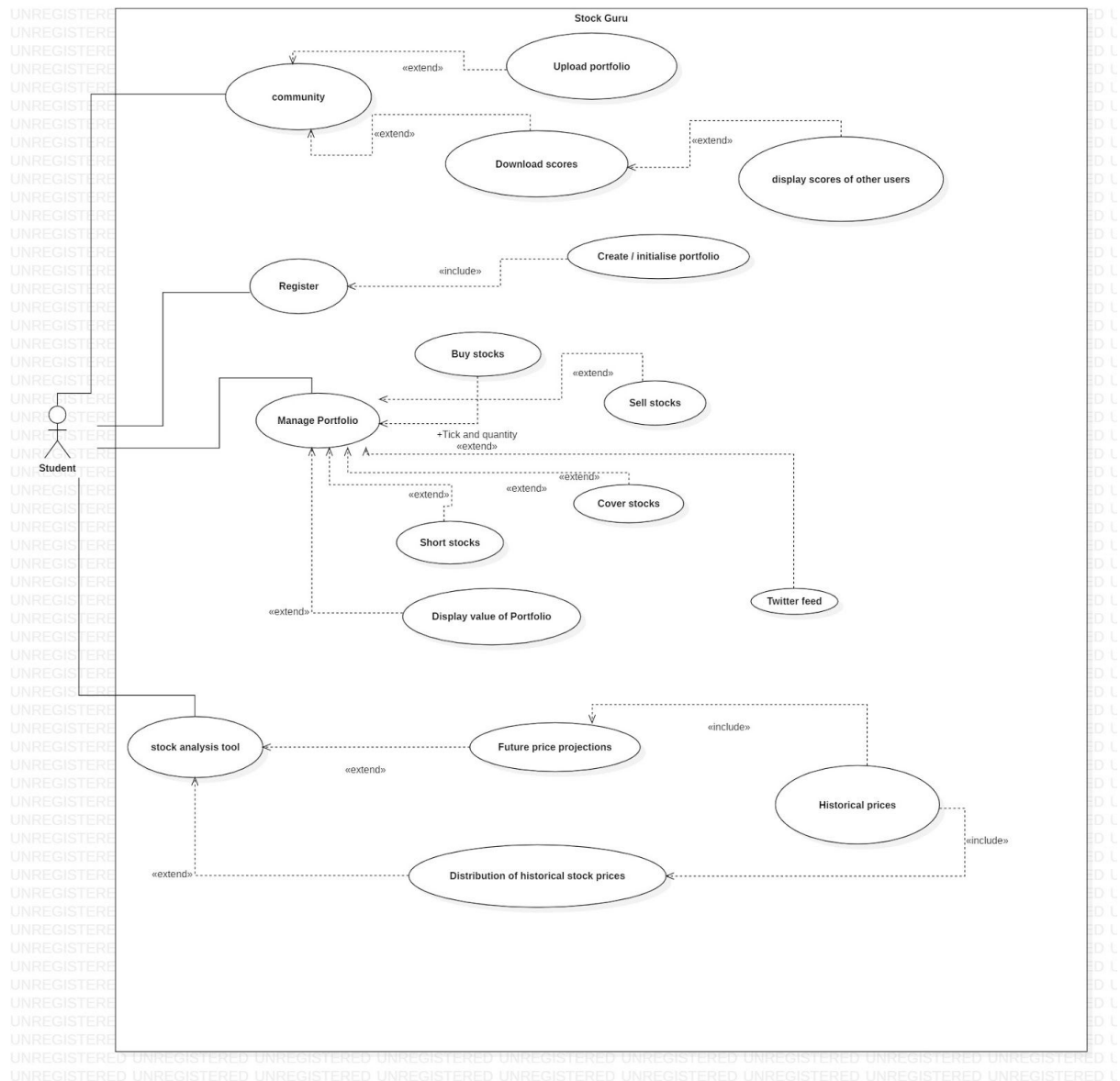


Figure 2: Use Case Diagram outlining the actions the student can perform

Explanation of Use Case:

Student program

Use Case Name: stock analysis tool	<ul style="list-style-type: none"> Student views statistics and analysis of stock market prices
---	--

Pre-conditions: presses button in menu interface; enter company name, start and end date, and depending on what option is selected (future predictions, historical distribution, distribution of prices), one inputs number of projections, how many days into the future to project, how many bins for the distribution.	Post-conditions: LineChart is produced providing visualisation for option selected by user
Main Path:	<ul style="list-style-type: none"> • Menu button pressed • One of three options are pressed (future predictions, historical distribution, distribution of prices) • Details are entered according to option selected in jdialog
Alternative Path:	<ul style="list-style-type: none"> • Syntax error due to incorrectly written data in jdialog • Line chart is not opened

Use Case Name: community	<ul style="list-style-type: none"> • Student views and compares the value of his portfolio to other local users
Pre-conditions: presses button in menu interface depending on available options	Post-conditions: jtable of values of portfolios of users with their usernames;
Main Path:	<ul style="list-style-type: none"> • Menu button pressed • One of three options are pressed
Alternative Path:	<ul style="list-style-type: none"> • No alternative path

Use Case Name: Manage Portfolio	<ul style="list-style-type: none"> • Student manages his portfolio and can make 'transactions' and view social media comments
Pre-conditions: presses button in menu interface; buy/sell stocks: enter company name and quantity of stocks to be bought or sold; twitter: enter keywords	Post-conditions: Buy/sell stocks: complete transaction and update user portfolio; view portfolio: jtable of company names and quantities of stock; twitter: jframe with jtextarea containing tweets depending on keywords
Main Path:	<ul style="list-style-type: none"> • Menu button pressed • One of three options are pressed (buy/sell stocks, view portfolio, twitter)

	<ul style="list-style-type: none"> Required details are entered according to option selected in jdialog or jframe/jtable appear
Alternative Path:	<ul style="list-style-type: none"> Error popup message saying that transaction could not be validated. Jframe does not display

System flow charts:

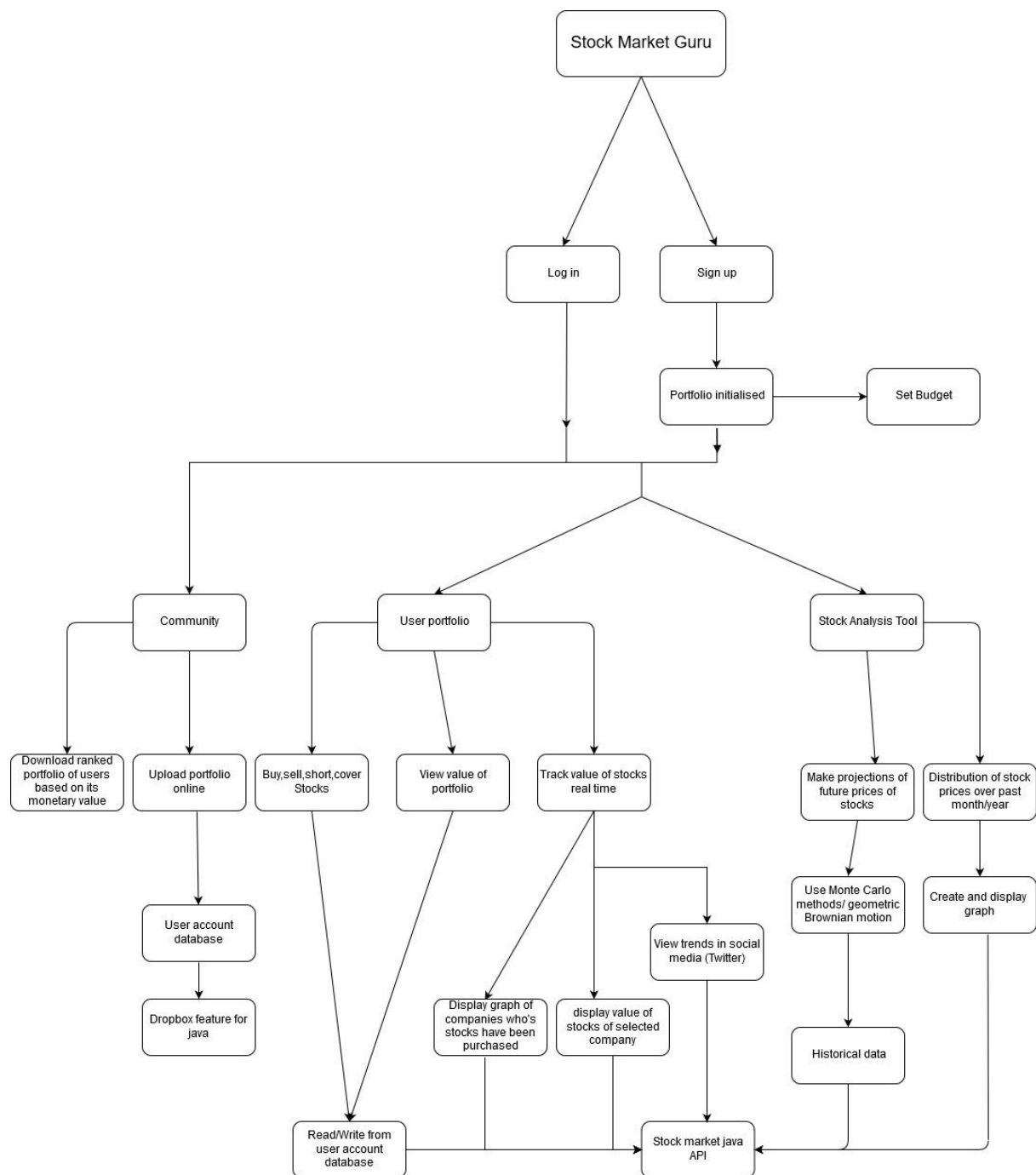


Figure 3: System Flowchart describing main functions of the program

UML class interaction diagram:

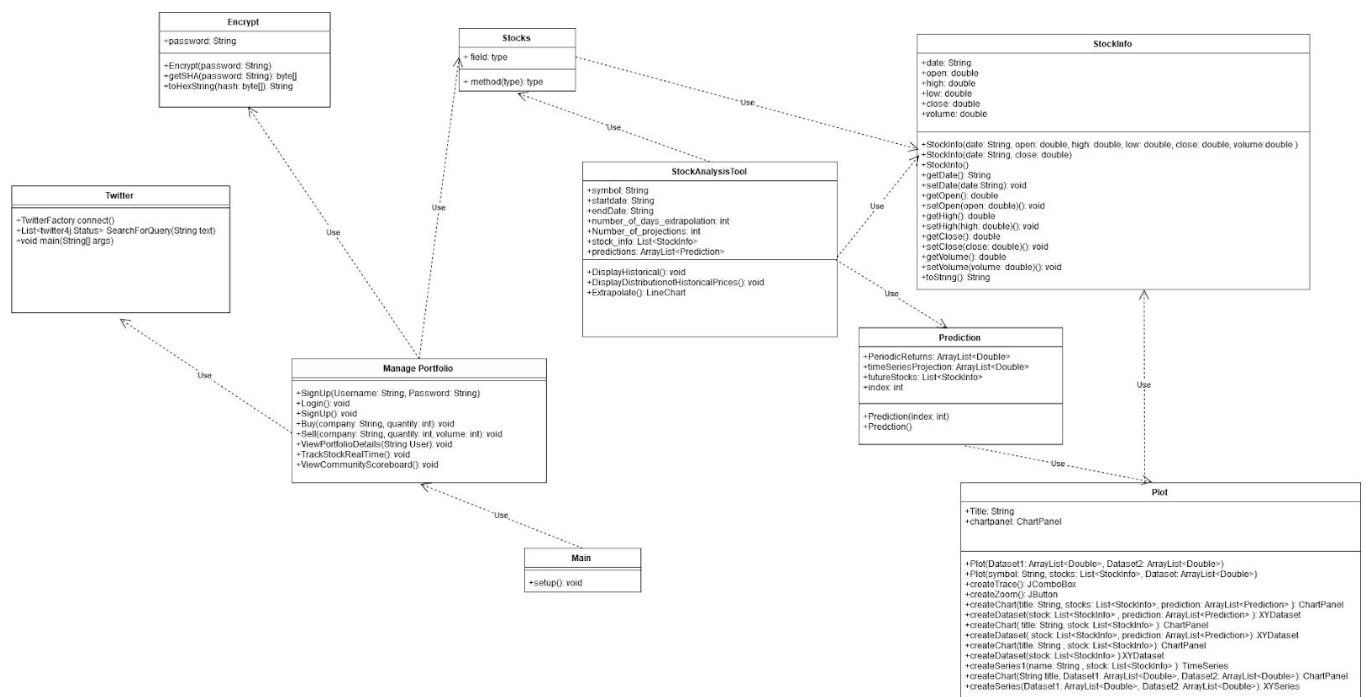


Figure 4: Class diagram displaying the structure of the programs system by showing all of the required classes, their variables and methods with their states, and the relationships amongst themselves.

Algorithm Flow Charts:

Pre-conditions	Post Conditions
Stock Symbol for Nasdaq listed securities	Update balance on user portfolio
Volume of stocks to be bought/sold	
Volume of stocks to be shorted	

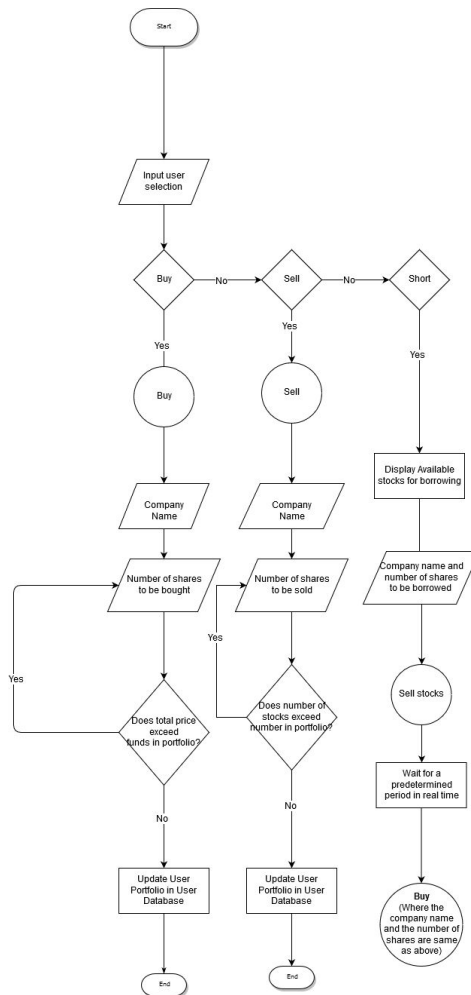


Figure 5: Flowchart describing the actions of buying, selling and shorting (covering) stocks

Pre-conditions	Post Conditions
Stock Symbol for Nasdaq listed securities	Output line graph of historical stock price time series
Start date in specific format ('yyyy-MM-dd')	

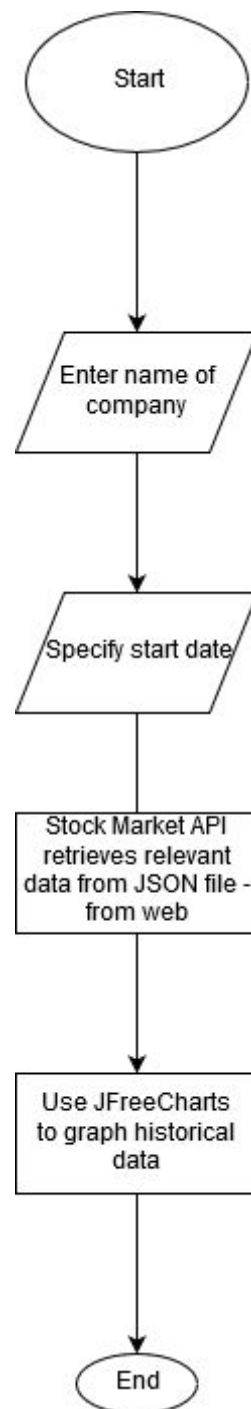


Figure 6: Flowchart describing plotting of historical data

Pre-conditions	Post Conditions
Stock Symbol for Nasdaq listed securities	Return time series of projected historical stock values
Number of days (integer) into future of stock price projection	
Time series (list of 'stock' objects) of historical values up to certain past date ('yyyy-MM-dd')	

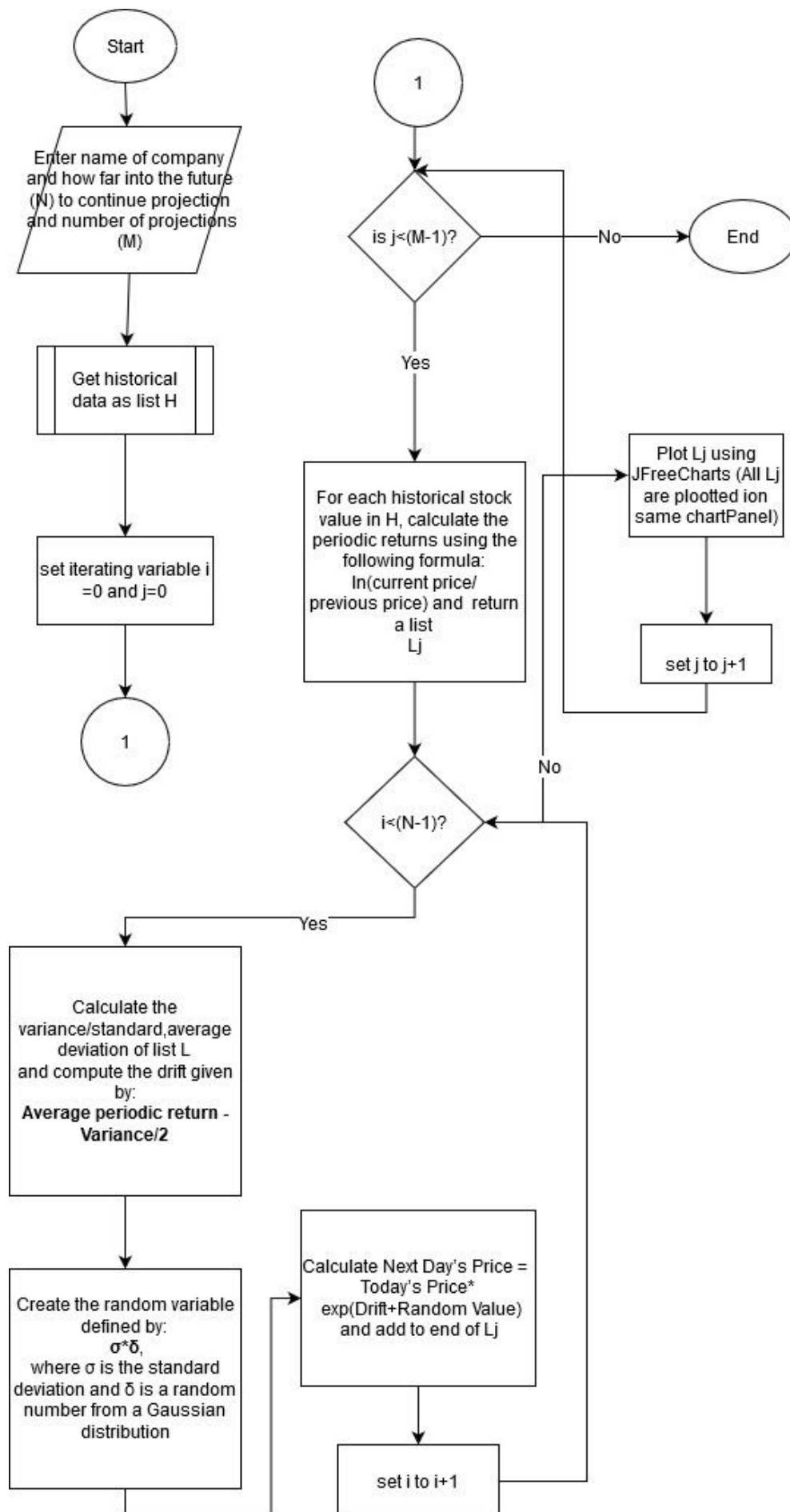


Figure 7: Flowchart describing implementation of geometric brownian motion algorithm for generation of future stock prices

Pre-conditions	Post Conditions
Username (string)	Proceed to main menu interface
Password (string)	Be denied access if the SHA 256 converted password and username are not found in the database of users

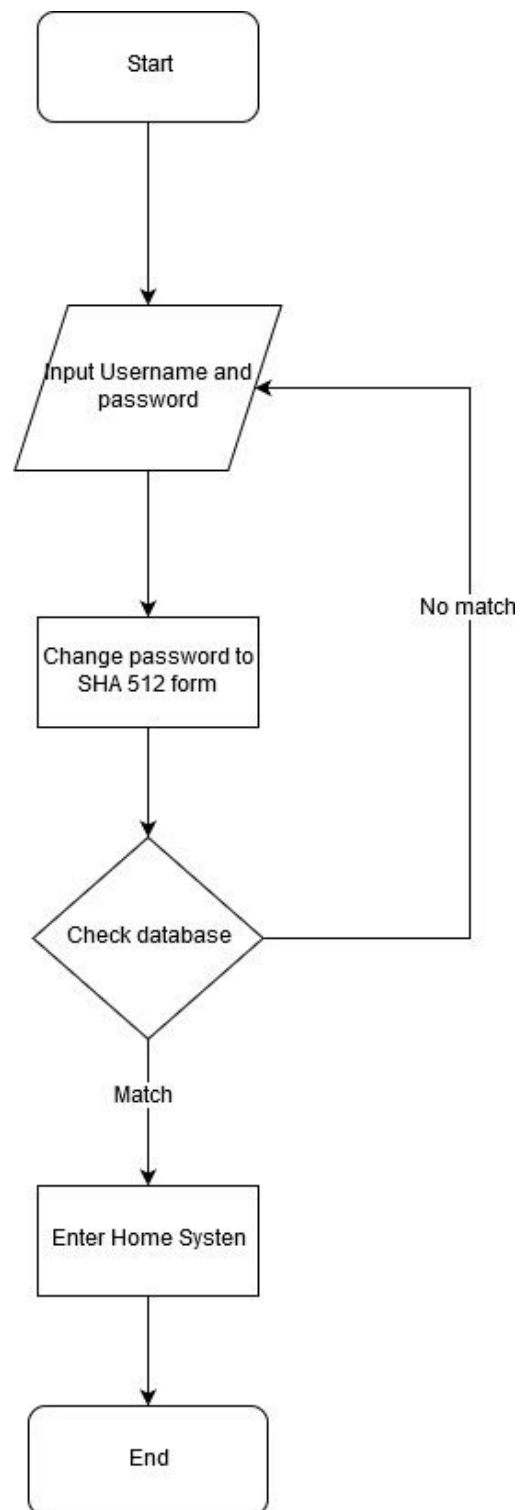


Figure 8: Flowchart describing the password has based encryption algorithm

Data structures:

Structure	Used for	Examples
ArrayList	To store closing market value of various stocks	futureStocks timeSeriesProjection Predictions Stock_info
StockInfo	Representing	Stock StockInfo(date, open, high, low, close, vol)
Random	To be used in geometric Brownian motion algorithm	nextGaussian
Date	Representing and manipulating dates from which stock values are to be obtained and added to create time series	Current finalDate
JPanel	To provide visualisations for statistics and historical stock values in the form of plots	chart
MYSQL Database	To store and manage user profiles and data	Users
Projection	Used to store a newly created time series of generated stock values	stockProjection
Timeseries	To create the time series dataset of stocks	HistoricalValues
XYSeries	To plot two datasets of real (double) numbers against each other; used in plotting	series = new XYSeries("Probability Distribution Function")
Binary Search Tree	To store user portfolio with company names and corresponding stock numbers owned by user	UserStockTree
byte[]	Encrypting user's password	byte[] Bytes

Table 1: Data structures that will be used in the product

Binary Tree:

Binary search trees will be used for storing an individual user's portfolio. Each node will include the total volume of stocks belonging to a particular company.

A node is entered based on string comparison of the company name, also known as symbol. A pseudo-code example of a recursive addition method is shown.

```
// recursive function for adding element to binary search tree
Function Node add(Node current, symbol, volume)
    if current == null then
        return new Node(symbol, volume);
    End if
    else if symbol.compareTo(current.symbol) > 0 then
        current.right = add(current.right, symbol, volume)
    End else if
    else if symbol.compareTo(current.symbol) < 0 then
        current.left = add(current.left, symbol, volume)
    else if
    else if symbol.compareTo(current.symbol) == 0 then
        current.volume += volume;
    End if

    return current;

Current = root; // current node is set to root of the tree
Symbol = "AMZN"; // company name set to google
Volume = 100; // volume of stocks to be bought set to 100

add(Current , Symbol , Volume ) // adds 100 Amazon stocks to user tree in example
```

Figure 9: Pseudo-code example of addition method for binary tree

Below is an example of a binary tree for a user where the node in figure 9 is added.

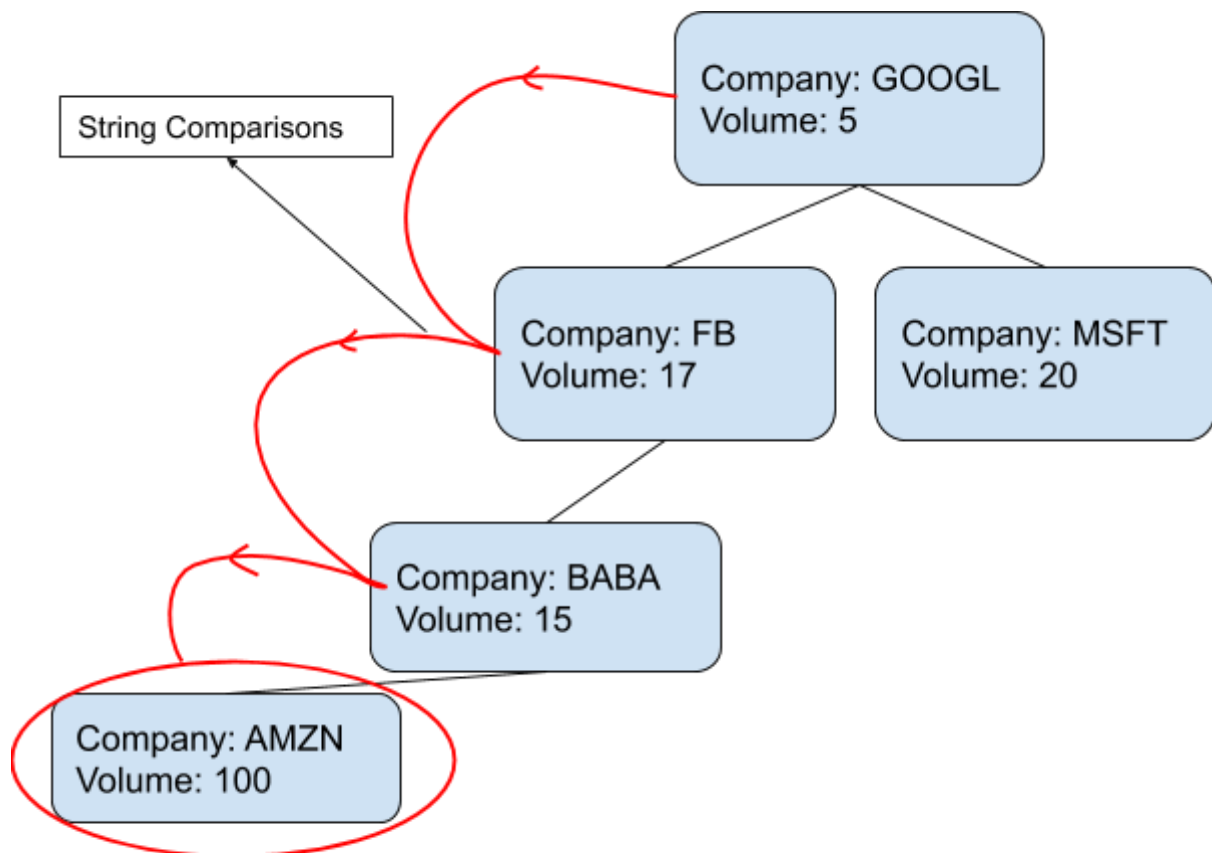


Figure 10: Example of binary search tree diagram for sample user

Date:

The date format used to access daily stock prices is: "yyy-mm-dd"

That is the precondition in any algorithm involving dates for the correct entry of date is the above format.

Below is an example of pseudocode in which the date is incremented

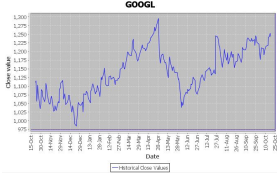
```
Set interval to 24 * 1000 * 60 * 60; // one day in milliseconds
Set date to "2019-10-23";
Set formatter to new SimpleDateFormat("yyyy-MM-dd"); //creating date
//format
Set current to (Date) formatter.parse(date);
Set finalTime to current.getTime() + val * interval;
Set finalDate to new Date(finalTime);
Set Final to formatter.format(finalDate);
Set iterateTime to current.getTime() + interval; // convert string date to
//number to which one day is added
Set iterateDate to new Date(iterateTime);
Set StartDateIterate to formatter.format(iterateDate);
//the above line would print "2019-10-24"
```

Figure 11: Date increment in java

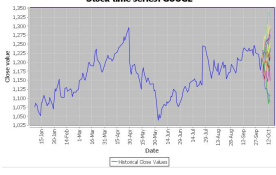
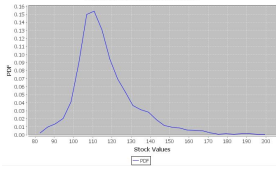
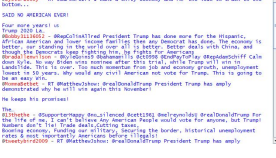
Test Plan:

Success Criterion	Description	Expected outcome	Actual outcome
1.1	The product will model a perfectly efficient market	Stocks are traded using their 'fair' value. Simply, the program reads historical quotes from online assuming the validity of the Efficient Market Hypothesis ¹	
1.2	The product will model market volatility	Erratic plot of future stock time series (Appendix A, Geometric Brownian Motion Illustration)	Product successfully models stock price volatility
2	The product will allow the students to trade virtual stocks using a fixed initial budget	-	
2.1	The product will enable the use to buy shares	The user should be able to enter company name and number of stocks to be bought (e.g., 'MSFT', 100)	
2.2	The product will enable the use to sell shares	The user should be able to enter company name and number of stocks to be sold. (e.g., 'MSFT', 75)	

¹ Kuepper, J. (2019, November 18).

2.3	The product will enable the user to short shares	The user should be able to borrow shares and sell them, hoping he or she can scoop them up later at a lower price, return them and keep the difference. ² (e.g., 'GOOGL', 30)	
3	The product will model the changes in the portfolio	Each time the user view their portfolio, the value of their stocks will be retrieved from an online source in real time	
4	Multiple portfolios - each for a user - will be stored in a database once initialised and their values tracked in real time	The program will store the user portfolios in a mysql database	
5	The Product will use historical data from companies to make predictions for their future price using monte carlo/stochastic methods.	The product should retrieve stock price data from an online source and form a time series that is to be plotted.	
6	The product will display historical data of companies on a graph.	It should plot the closing daily stock values against the date under consideration	

² <https://www.marketwatch.com/story/why-you-should-never-short-sell-stocks-2015-11-19> (Last accessed: December 3 2019).

7	The product will give projections of future price movements of a company's stock as line graphs using simple statistical procedures incorporating "drifts" and "shocks"	Stock projections will include a deterministic component using the moving average generated by given stock values and will incorporate a random shock using a gaussian random variable	 <p>Stock time series: GOOGL</p>
8	The product will produce and display a distribution (approximating a normal/ lognormal distribution) given the historical data	The distribution of generated stock values within a time frame should follow a lognormal distribution	 <p>Distribution of Stock Prices</p>
9	The product will use statistical methods and models. More specifically, random walks and Geometric Brownian Motion	The program will implement a stochastic algorithm using pseudo random methods to obtain and print future time series	
10	The product will include social media posts, more specifically, tweets generated from a simple query using an API.	The program should allow the user to enter keywords and print a list of relevant tweets (e.g., query: 'Trump, American economy')	<p>Result for query: 'Trump, American economy'</p> 
Miscellaneous (Normal)	Whether password encryption occurs correctly	To display SHA 256 string value of user password with no error	<p>"Password"</p> <p>=</p> <p>e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a</p>
Miscellaneous (Boundary)	Test the limits of the part of the product responsible for plotting	Generate an 'absurd' (500) number of simulations of future	The future paths were successfully generated with an

	and generating future stock time series	stock prices given a set of historical values	acceptable latency of 25 seconds.
Miscellaneous (Erroneous)	Enter string instead of date in stock analysis tool for obtaining historical stock quotes	Enter "I love computer science" Instead of date format "yyyy-mm-dd"	Error message displays violation of pre-conditions of algorithm

Table 2: Test plan showing which success criterion each test corresponds to, description of test, and the expected and actual outcomes of the test.

Word Count: 180

Bibliography:

- Kuepper, J. (2019, November 18). Efficient Market Hypothesis (EMH) Definition. Retrieved November 22, 2019, from <https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp>.