Web-Based Stock Forecaster

REPORT 2, PART 2

GROUP 9

KRISHA PAULA OLANDAY
VINAY PANJABI
VINAY SHIVAKUMAR
NEHA DESAI
JONATHAN HAAS
SIVARAMHARESH SIVA

DROPBOX: HTTPS://WWW.DROPBOX.COM/SH/58UOMDLU8CB6IPI/9FRUKZ7NXK
WEBSITE: HTTPS://BITBUCKET.ORG/VINKNEE/STOCKFORECASTER-GROUP-9

INSTRUCTOR: IVAN MARSIC

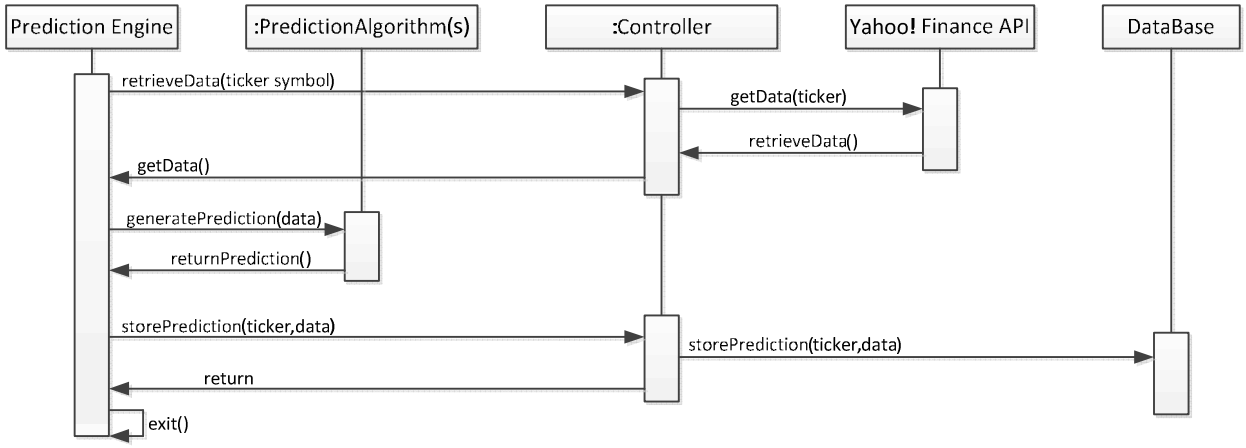
TABLE OF CONTENTS

1 CONTRIBUTION BREAKDOWN

All members contributed equally.

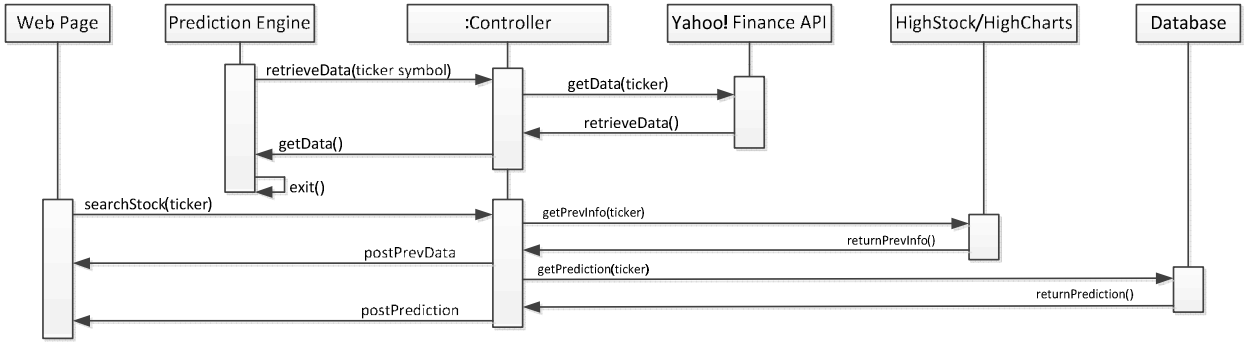
2 SEQUENCE DIAGRAM

2.1 OBTAIN PREDICTION



The design principle employed in this diagram is the Expert Doer Principle because there is a decent amount of communication in between the objects, but it is short and focused. Therefore, since the communication chains are shortened between objects, this principle works best for this design.

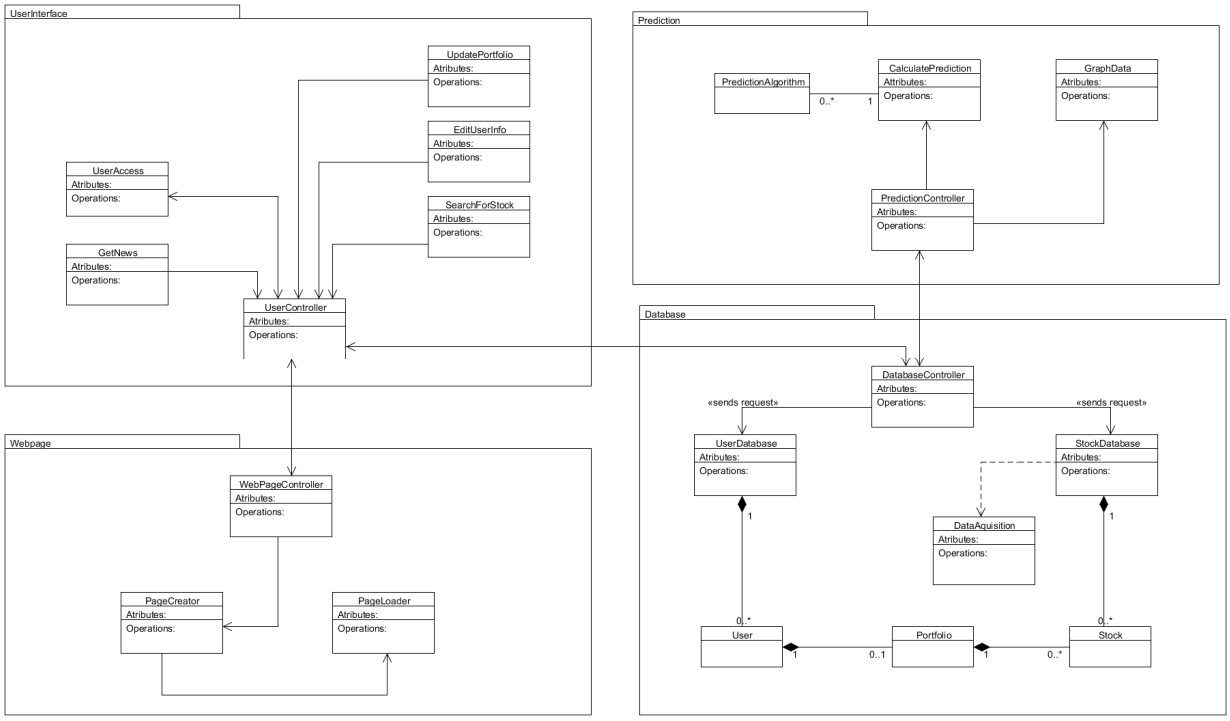
2.2 DATA ACQUISITION



The High Cohesion Principle is exemplified in this diagram since there is a lot of communication between the objects. All the data is being moved from location, to location to be stored. Therefore, there needs to be lots of communication.

3 CLASS DIAGRAM AND INTERFACE SPECIFICATION

3.1 CLASS DIAGRAM



3-1: Class Diagram

3.2 DATA TYPES AND OPERATION SIGNATURES

The detailed specifications of each individual class is show in in UML notation in Figures 3.2, Figures 3.3, Figures 3.4 and Figure 3.5. The classes are separated into four packages: UserInterface, Database, Prediction, and Webpage. We will explain each class's functions and operations in package order.

3.2.1 UserInterface Package

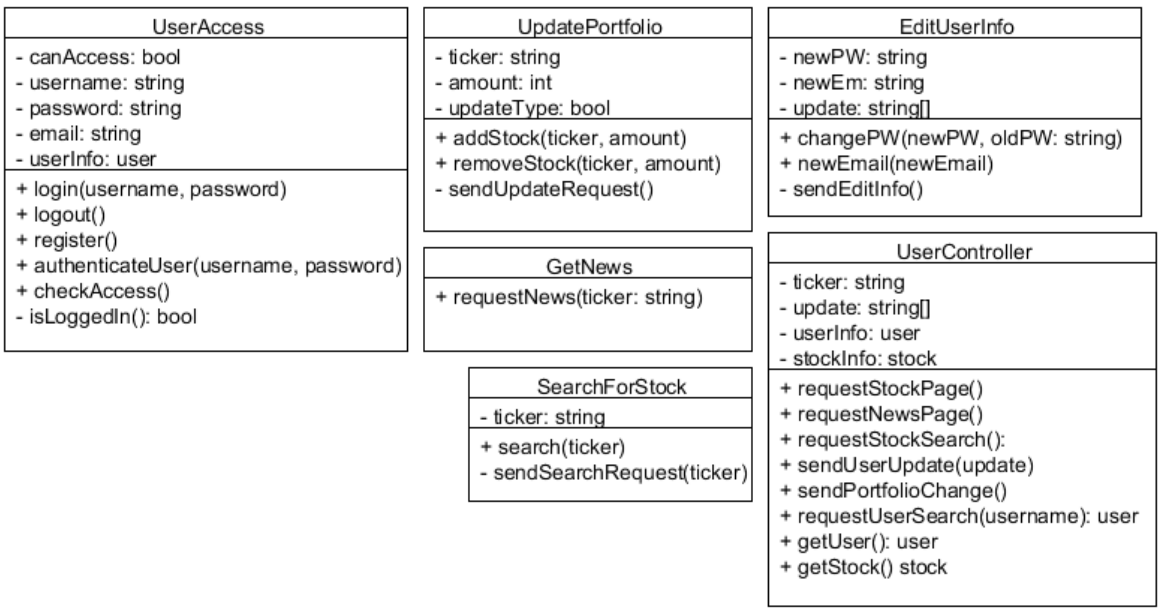
The classes Login and Register are very self-explanatory and do not need clarification. So we will not go in depth with these two classes because they are not unique to project itself.

1. UserAccess

   UserAccess controls when the user is able to access the system. The class will check if the user is currently logged in or logged out and will adjust the access accordingly. In order to achieve this, the class itself will have two parameters. One parameter will be a boolean flag and the other one is simply the username. Every time the user logs into the website (and is properly authenticated), the login function will call UserAccess and set the Boolean flag to 1, making it true. When that subsequent user logs out of the website, the Boolean flag will be set to 0, making it false. UserAccess will actually be linked to the UserDatabase and the Boolean flag will be stored with the user information. This is where the second parameter comes in; it is used to check if the user already has access to the web application (currently logged in) or they haven't

logged in yet. UserAccess looks for the username in the Database and checks if the Boolean flag has been asserted. The UserAccess class also handles the login and register requests of the user.

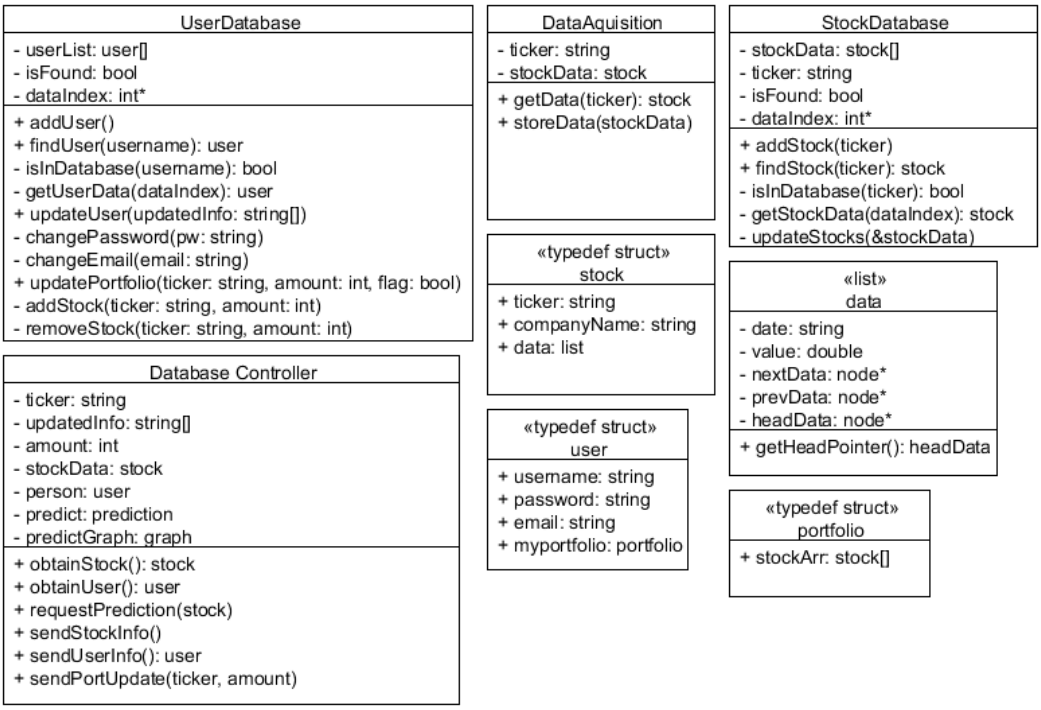
2. EditPortfolio
   EditPortfolio is a simple class that the user uses to update their portfolio of stocks. The class enables users to add a stock and remove a stock to their portfolio. In order to do this, the class has two attributes the stock ticker and an amount. This way the user can add or remove the desired number of a given stock.
3. EditUserInfo
   This is another class that is fairly simple. EditUserInfo enables users to change their password, update their email, and other information. If a user does change their information, the class will send this information to the Database with a Boolean flag that will indicate that information has been changed. The Database will then know that information has been changed and update it accordingly.
4. SearchForStock
   The class is the key feature of our system; it enables users to search the Database for a given stock using either the stock ticker or the actual company name. The class will send this request to the Database and have the stock's information, prediction, and graphs be returned. Once all the information is returned it will send the information to the Controller so that the stock's webpage may be created and loaded for the user.
5. GetNews
   This class obtains news articles using RSS feeds from reputable websites and Twitter to obtain news about a certain stock.
6. UserController
   As the name suggests, this is the controller for the user interface. The controller receives requests to the Database from the functions within the user interface and sends them to the Database. It is essentially the communicator between the interface, the database, and the webpage.



3.2.2 Database Package

The classes and objects in this package store, obtain, and maintain all of the user, portfolio, and stock data. First we will talk about the classes that define objects before delving into the classes that maintain functionality in the Database.

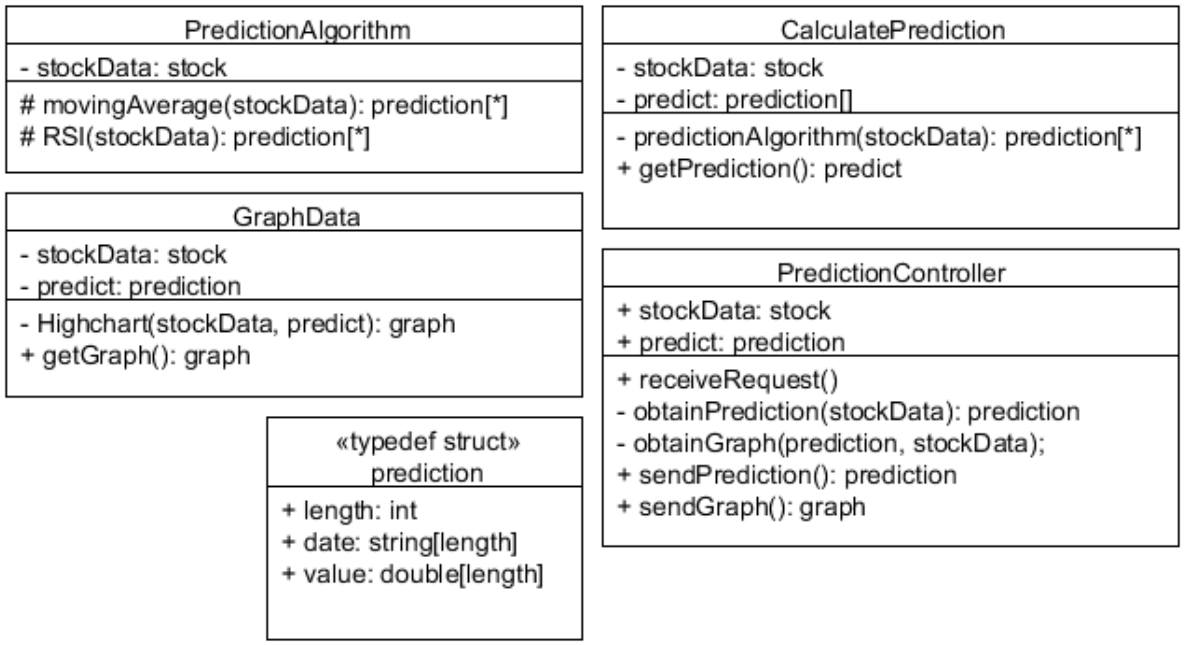
1. Stock
   This class defines and contains all the information that defines a stock. This information is: stock ticker, company name, a 2D array of the stock's historical data. This two-dimensional array will contain the stock values in a range of dates.
2. User
   User is a class that is similar to stock in that it defines and contains all the information that defines a user. It will hold the user's username, password, email, and portfolio. The portfolio is a separate class that will be defined below. Therefore the User class contains the Portfolio class.
3. Portfolio
   The Portfolio class holds the stocks that each user adds into their portfolio. The class itself has an array of stocks as its attribute. The class enables the User class and other classes in the Database to access and edit the portfolio.
4. UserDatabase
   This class contains a list of all the users that had registered in the system. It has operations such as addUser, findUser, updateUser, updatePortfolio and validateUser. These operations are self-explanatory in that it enables the class to add new users to the database, finds a user in the database, updates user and stock information,,and validates existing users when they log into the system.
5. StockDatabase
   The StockDatabase is similar to the UserDatabase. It is a class that contains a list of all the stocks acquired from the Yahoo! Finance API. It has functions such as findStock, addStock, updateStock. The StockDatabase uses the class DataAquisition to add new stocks and update current stocks in the database.
6. DataAquisition
   DataAquisition is the class that queries stock information from Yahoo! Finance API. It has 2 operations, aquireStock and queryNewData. The aquireStock operation obtains new stock data form Yahoo! Finance if the StockDatabase does not have that specific stock. The operation queryNewData, is what is used to update and add new stock values into the already stored stocks in the StockDatabase.
7. DatabaseController
   This class is similar to the UserController and is the controller for the database. It communicates with the UserController and ObtainPrediction to send and receive requests from the functions in the database.



3.2.3 Prediction Package

The prediction package holds all of the classes that are needed to obtain a prediction.

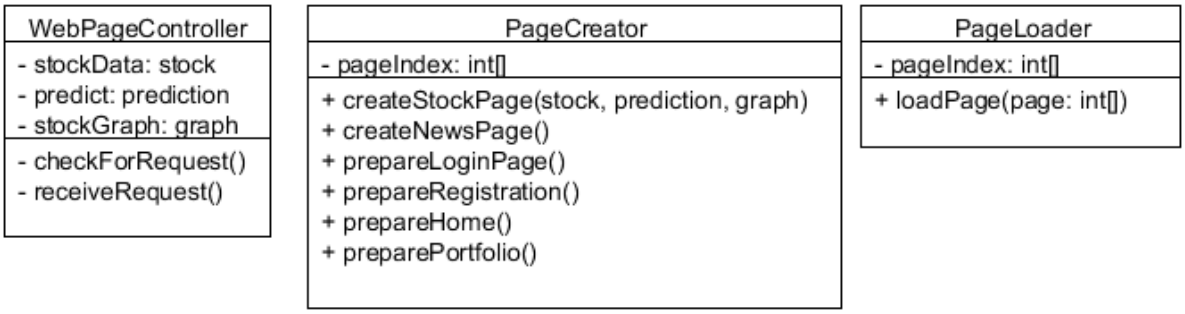
1. PredictionController
   This class is the controller for the Prediction package. It receives requests from the database to calculate a prediction and then conveys the request to CalculatePrediction.
2. CalculatePrediction
   This class obtains an algorithm from the Algorithm class to calculate the prediction.
3. PredictionAlgorithm
   Each operation in this class is a specific prediction algorithm.
4. GraphData
   This class calls Highchart to create a graph of the historical data. It also obtains the prediction from CalculatePrediction to include the prediction in the graph.



3.2.4 WebPage Package

The webpage package contains the classes the will ultimately prepare the requested webpage and then load that generated webpage.

1. PageCreator
   This class prepares the webpages that the user requests. It obtains all the parameters that will be on the webpage.
2. PageLoader
   After the webpage has been prepared by the PageCreator class, it tells the PageLoader class to load the prepared webpage.
3. WebPageController
   This class handles all the requests to prepare and load a webpage. It obtains the information to be placed on the webpage and then sends those attributes to the PageCreator.



3.3 TRACEABILITY MATRIX

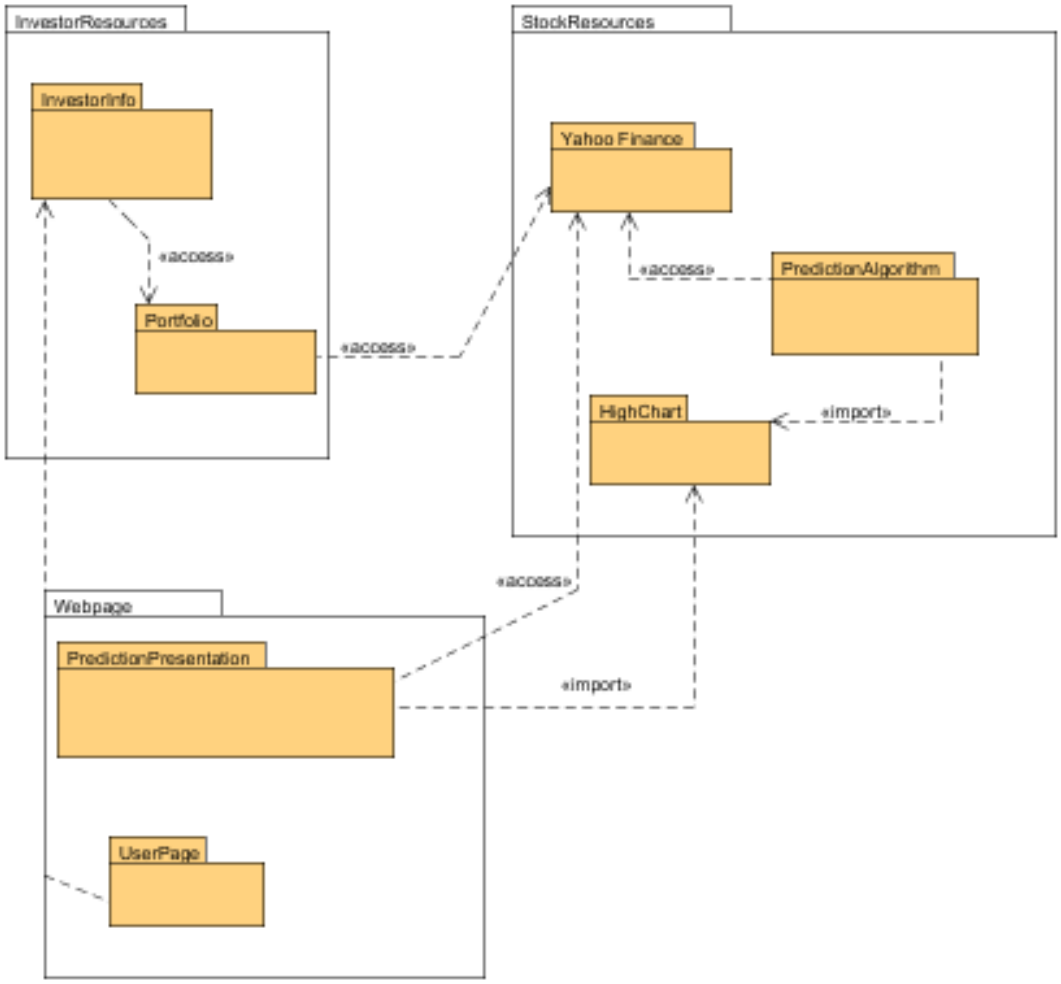
		CLASSES																		
CONCEPTS	PW	UserAccess	GetNews	UpdatePortfolio	EditUserInfo	SearchForStock	DatabaseController	UserDatabase	StockDatabase	DataAquisition	User	Portfolio	Stock	PredictionAlgorithm	CalculatePrediction	GraphData	PredictionController	WebPageController	PageCreator	PageLoader
USER DATABASE	7							X			X	X								
EDITUSER INFO	4			X	X			X												
MAINTAIN DATABASE	4						X	X	X											
UPDATE PORTFOLIO	12			X	X			X				X								
USER ACCESS	7	X																		
MANAGE USER	8	X		X	X			X				X								
STOCK DATABASE	33						X		X				X							
DATA AQUISITION	34								X	X			X							
SEARCHFOR STOCK	28					X			X				X							
OBTAIN PREDICTION	26													X	X	X	X			
WEBPAGE	28		X															X	X	X
MAX PW		8	28	12	12	28	33	12	34	34	7	12	34	26	26	26	26	28	28	28
TOTAL PW		15	28	24	24	28	41	35	99	34	7	27	95	26	26	26	26	28	28	28

4 System Architecture and System Design

4.1 Architectural Styles

This project will be deployed using a Client/Server model because the system is segregated into two applications. The client queries and submits data through the web site and on the server side the user and stock information is stored in the database. The predictions are also done on the server side, or on a seperate machine that will go through the tracked stocks and store the prediction information.

4.2 Identifying Subsystems



Our design will be structured similarly to the MVC model (Model, View, and Control). Model deals with storage and managing data, View is how the user interacts with our product, and Control handles the main algorithm. The subsystems that are shown in the UML package diagram are named Investor, Stock, and Webpage.

The Investor package has the Portfolio which holds all the stocks of a specified user and the personal information of the user. The Portfolio has a dependency on the Investor because each investor has its own portfolio. The portfolio also has a dependency on the information from Yahoo Finance since it needs that information to update the portfolio.

The Stock subsystem has a dependency on the Yahoo Finance subsystem. This is because the info in the Yahoo Finance subsystem needs to be used for all of the things the Stock subsystem uses. Highchart also needs this information to generate its charts. It also relies on the PredictonAlgorithm subsystem to show the prediction.

The different tags next to each line are for different reasons. <<Import>> is placed where information is shared from system to system publicly, and <<access>> is for when this data transfer is to be private, like when sharing passwords, or other sensitive information.

4.3 Mapping Subsystems to Hardware

Our system does need to be run on multiple computers. The subsystems are mapped in the following components:

- The StockResource package and the packages contained inside it are allocated in the central server.
- The WebPage package is allocated in the client's PC.
- The InvestorResource package is also contained insidethe central server.

4.4 Persistent Data Storage

For the stock forecaster the system does need to save data that will outlive a single execution of the system. A sample of the data that should be saved is stock tickers and their corresponding company names, user information, and stock data. The data will be stored using a relational database, specifically MySQL. MySQL was chosen for the job due to its open source nature and has all the functions required for the job. Also, several team members have experience with this database and feel that it is the best option. The data will be stored in three separate tables. The first table will hold the user data which includes username, password, and portfolio. The second table will have stock ticker, the corresponding company name, and information about the company such as industry and a small description. The third and final table will have the stock ticker and the moving average data computed for the stock.

4.4.1 Table 1: User

```
Create Table user_data (
      UserID  int not null auto_increment,
      Name varchar(100) not null,
      Username varchar(100) not null,
      Password varchar(100) not null
      Email varchar(100) not null
      Primary Key(user_id)
);
```

UserID	Name	Username	Password	Email	Portfolio

4.4.2 Table 2: Stock

```
Create Table stock_company(
            Ticker varchar(100),
      Company_Name varchar(255)
);
```

Ticker	Company_Name

4.4.3 Table 3: Stock Data

```
Create Table stock_data(
      ID int not null auto_increment,
      Ticker varchar(100),
      Prediction_Model varchar(255),
      D1 int,
      … D200 int
);
```

ID	Ticker	Prediction_Model	D1 ….	…. D200

4.4.4 Table 4: Portfolio

```
Create Table portfolio(
      Username varchar(100) Not Null,
      Stock1_in_Portfolio var(100) Not Null,
… StockN_in_Portfolio var(100) Not Null,
);
```

Username	Stock1_in_Portfolio ….	StockN_in_Portfolio

4.5 GLOBAL CONTROL FLOW

4.5.1 Execution Order

The system is event driven, where a user must send commands to the system to get their desired information. For example, if a user wished to view Google's stock chart, they must input the ticker into the search bar and search for the stock. To view predictions the user must prompt the system to show the prediction, as well.

4.5.2 Time Dependency

The system is event driven in regards to the prediction models, but the database containing the stock prices will update daily. The prediction will happen in real-time once the user has requested the information.

4.5.3 Concurrency

The system will use multiple threads in the prediction process. Once the time comes to make a prediction, the predictor will create threads to carry out the calculations involved in the prediction model. At the end of this process the threads will recombine and the prediction will be sent to the user.

4.6 HARDWARE REQUIREMENTS

Screen display: Since our Web application utilizes Twitter Bootstrap, this component allows any computer resolution starting from 1024 px X 768 px (portrait size) to 1920 px X 1080 px (large display).

4.6.1 Disk Storage

The Web Application isn't fully implemented yet, but here are some preliminary estimates:

1. User needs modern computer (preferably a processor of i5 or higher)
2. Approximate server space should be near 20 gigabytes (for SQL Database).
3. RAM: 2 GBs

4.6.2 Communication network

At the moment we are unsure of the exact bandwidth needed or used, but once it is known the document will be updated.

4.6.3 Device Flexibility

This web application should primarily be used for computers, however with the utilization of Twitter Bootstrap, we also have the capability of extending this application to Tablets and/or phones. If there is enough time and resources, the web application would be extended to other devices as well.

5 ALGORITHMS AND DATA STRUCTURES

5.1 MOVING AVERAGE MODEL

Moving average is a statistical process that creates a set of averages for many small subsets of the full dataset. By taking the averages of mini-subsets of the dataset, it helps smoothen out the trend of the data by showing the long-term lengths. It usually shows the price fluctuation of a stock for a certain period, but can also be used for longer periods.

Algorithm used for moving average:

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

$$Pm = prices\ of\ the\ stock\ at\ a\ certain\ point$$

$$n = period\ of\ days\ the\ data\ is\ across$$

Fig 5.1.1 Moving Average Model Implementation Example

5.2 Relative Strength Index

RSI, in practice, is an oscillator from 0-100 showing the momentum of a given stock. The equation for the RSI is calculated by the following formula:

$$RSI = 100 - 100/(1 + RS)$$

$$RS = Average\ of\ x\ days'\ up\ closes\ /\ Averages\ of\ x\ days'\ down\ closes$$

This formula will either be incorporated into the prediction algorithms, or we may pull the data from a website. When the oscillator goes below the 30 line, it is an indication that the stock has a good chance of going up. Conversely, when the oscillator climbs above 70, there is a high chance the stock price will fall soon. Using that information, the prediction algorithm will detect when a stock has passed either threshold, and adjust its prediction model accordingly.

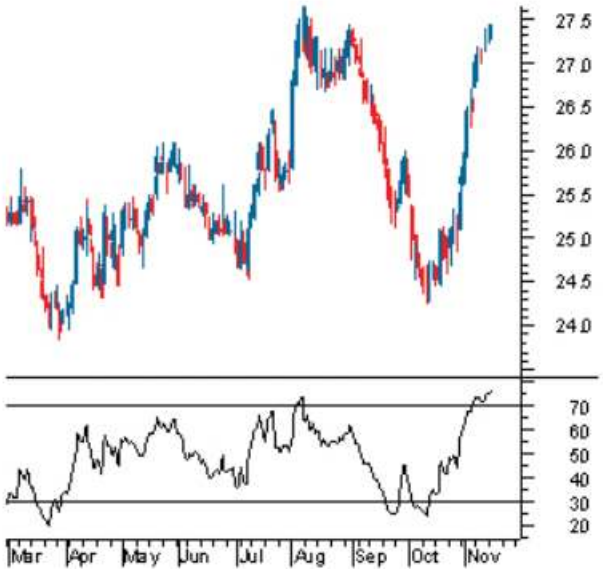


Fig. 5.2.1 Relative Strength Index Model

5.3 Data Structure Usage

The only data structure used in this system is the built-in data structure provided by the python package Pandas called DataFrame. The DataFrame is a tabular, spreadsheet-like data structure. It contains an ordered collection of columns, each of which can be a different type. This data structure was chosen since it has many built-in methods such as join and sort so; we did not to implement these functions. This saves us time and energy as well as allowing us to focus on creating the other parts of the system. Furthermore, this data structure is very flexible to work with since it can take multiple types and columns can be added as needed. For instance, the data structure internally handles missing data points and automatically creates an index to work with. Individual rows and columns can be accessed similar to accessing arrays in other programming languages. This allows us to easily work with the data we get from Yahoo's Finance API.

Fig 5.3.1: This is a sample DataFrame for the company Google.

GOOG

	Date	Open	High	Low	Close	Volume	Adj Close
0	2014-03-14	1181.99	1190.87	1172.53	1172.80	2293900	1172.80
1	2014-03-13	1207.95	1210.50	1184.76	1189.06	2339700	1189.06
2	2014-03-12	1196.40	1207.85	1184.19	1207.30	1964300	1207.30
3	2014-03-11	1213.77	1214.32	1196.64	1199.99	1713000	1199.99

4 2014-03-10 1215.69 1217.64 1204.09 1211.57 1214600 1211.57

Moving_Average_200_Day

0 403.93005

1 405.08390

2 406.20225

3 407.28745

4 408.44785

6 User Interface Design and Implementation

The team is currently in the process of implementing the user interface. We will upload the implementation once it is complete.

The user interfaces have not changed much from the initial mock ups created in Report 1. The only major changes to the user interface is that we will no longer be implementing the "Previously Searched Stock" list. This decision was made due to the time constraints and resources. We want to focus majority of our time making sure that the necessary interfaces are functioning correctly. Everything else stated in the User Interface Specification section of Report 1 will be implemented.

6.1 Login

The first page that users will encounter will be the login page. All users will be required to either login to an existing account or sign up for a new one. This page will show a simple box with two input areas where the user can enter their user name and password. Underneath this input are two buttons next to each other. The left button will be the LOGIN button and the button next to it is the SIGN UP button.

Clicking the LOGIN button will either bring you to the home page if you entered the correct username and password or it will just refresh the login page with an error stating that you have entered your credentials wrong.

6.2 Sign Up/Registration

On the login page, if a user is not already registered, they can sign up to create a new account by clicking on the SIGN UP button. When clicked, this button will bring them to the registration page where they will be able to create a username and password. They will also be required to enter an email address and optionally enter their name.

6.3 Navigation

We will have a static navigation bar at the top of the website. This ensures that no matter which page the user is on, the navigation bar will consistently be at the top of the page for the user's ease of

navigation. The navigation bar will consists of three tabs and a search bar. The three tabs will be on the left hand side of the navigation bar and they are HOME, PORTFOLIO, and NEWS. Next to these tabs will be the search bar. It will span the middle section of the navigation bar. Users will be able to search different stocks no matter which page of the site they are on. After the search bar, there will be a drop down ACCOUNT menu. This drop down menu is where users will be able to edit their account information and sign out of the system.

6.4 STOCK PAGE

Each stock in our database will have their own individual pages. The page itself will be divided into three distinct section. The main and largest section will be on the top left hand side and it will take up two thirds of the page. This section is where the graph of the stock's historical and prediction data will be displayed. The graph will have an indicator of the current data and which points on the graph represent the stock prediction. Underneath this section, will be a smaller section that will show the open, high, low, and current value of the stock for that given hour or however long we decide to refresh the data. The third section will be a column next to the graph. This is where you will see the stock ticker that identifies which stock you are currently viewing as well as the company name and a link to their website underneath it. Next to that ticker title will be a button that enables you to add that specific stock to your portfolio. Underneath this will be links to news articles about the stock. This way you can see relevant news information about the stock.

6.5 HOME

The home page will be the page that users see after they log into the site. It consists of two columns. The left column is a list of current news and events of the stocks that are in the user's portfolio. If they are a new user, it will invite them to add stocks to their portfolio. The right column will show a graph of all the stocks they have in their portfolio. This graph will comparatively show the performance of the individual stocks that the user owns. Underneath the graph, it will state which of their stocks has the worst and best performance. This way users can see how well their portfolio is doing and see which stocks are creating a profit and which stocks aren't.

6.6 PORTFOLIO

The portfolio page is where users can actively add and remove stocks to their portfolio. The stocks will be in a list form and next to each stock will be the current stock value as well as add and remove buttons. Once these respective buttons are clicked, a pop up with the stock ticker and a box will pop up asking how many shares they would like to add or remove from the stock they have in their portfolio. They will also be required to verify that they want to make these changes.

At the top and bottom of the Portfolio page, there will also be a button with an input field next to it that will enable users to add a stock to their portfolio. The user will enter the stock ticker to this input field and click "ADD STOCK". Once clicked, the stock's page will be loaded and if the user truly wants to add the stock they can click on the "ADD STOCK" button next to the Stock Ticker on the page. A pop up will then appear verifying that they truly want to add the stock to their portfolio.

6.7 NEWS

This news pages will be a stream of the current events that are happening about different companies. These events will be obtained from reliable news sources via their RSS Feeds. The news on this page will not be catered to the stocks in a user's portfolio but is more of general news feed. Users will be able to see and read current events even though they do not have those companies in their portfolio.

6.8 FAQ

This page will have a list of frequently asked questions that will explain to users what each part of the website is and teach them how to use the site including understanding various prediction models.

7 DESIGN OF TESTS

*A lot of the code has not yet been implemented, so specific testing cases that may be relevant to certain frameworks or specific code implementation may not be mentioned yet

7.1 USER INTERFACE TESTING

7.1.1 Use Case 1 – Search for Stock

This is an important concept in our domain analysis and it enables the user to search for a stock in our database. Once the stock is found the system will request the stock's prediction and load the stock page. The test for this use case is to primarily ensure that the search function is working and the input entered by the user is read by the system properly.

7.1.2 Test 1 – Controller Received and Sent Input Request

This test will see if the input entered by the user is received and sent by the controllers in the system. It will make sure that the user controller can read input entered by the user. If the user controller cannot obtain this input then it will not be able to convey this information to the database controller and an error will occur internally and the stock page will not load for the user. This test will also see if the database controller is able to obtain the input request from the user controller. This test is to see if the stock that is searched by the user is sent properly to the system.

7.1.3 Test 2 – Database Connection

This test will make sure that the database controller sends the request to the database properly. It will also make sure that the information that the database sends back is the same information that the controller requested.

7.2 STOCK PAGE

7.2.1 Test 1 – Load Page

This test will make sure that after the stock has been searched that the correct page is loaded. It will also make sure that the various components of the page is also loaded successfully. This will include the graphs

7.2.2 Test 2 – Graph

This ensures that the stock's graph on the page is correct and loads properly. There will be two graphs, one will have a dependency on yahoo finance data to populate and the other will have data that will be populated from our database. We will be able to identify issues due to the fact that we have two graphs pulling data from different sources. This could be useful because if one graph's data isn't loading properly we can pinpoint it on that graphs data source which will aid in debugging.

7.3 Web Pages

7.3.1 Test 1 – Load Page

Like the stock page, we will run a test for all the other pages on the website and make sure they load properly. These pages are Portfolio, Login, Registration, Logout, Home, News, and FAQ

7.4 Prediction Algorithm Testing

The prediction algorithms will be tested using previous data. This has not been implemented yet in terms of actual testing, however manual testing has begun that compares the result of the moving average based off of historical data. This testing can be automated using programs that will test for validity of both the algorithm itself as well as the validity of the code. We will implement these tests for all of the algorithms that we intend on implementing, at least the Moving average and RSI for now.

7.5 Integration Testing

Our integration testing will be done using big bang integration. Prior to compiling each individual sub part, we will conduct unit testing on individual pieces of software in order to validate methods and strategies so that we can eliminate logical errors and concentrate on technical errors that may occur as a result of integration. There aren't too many external dependencies (the only is yahoo! finance api) and everything is relatively modular since everything has its own purpose and is independent of other aspects of the website so pinpointing issues shouldn't be too much of an issue. We will design larger test cases once the program is complete that will allow the programmer to quickly identify any areas of concern, whether it be connection issues, algorithm issues or even API issues just as a few examples.

8 Plan of Work

Weekly Group Meetings: Tuesday at 4:30pm

8.1 Progress as of March 19, 2014

8.1.1.1 Database and Data Collection Progress (Sivaramharesh Siva and Neha Desai):

The tables for the SQL database have been diagramed and implemented. Data has been collected and analysis has begun, they are working closely with the Prediction, Research and Algorithm team in order to add the data into the database. This is on time relative to the time table that had originally been set.

8.1.1.2 Front-End Web Development Progress (Vinay Panjabi and Krisha Paula Olanday):

All of the website framework has been chosen and the design is finished. The pages have been mocked up on the front end and implementation of the charts has also begun. They will soon be ready to retrieve data from the database once we set up a proper environment that we can connect to and collect from. This is on time relative to the time table that has originally been set (see gantt chart).

8.1.1.3 Prediction Research and Algorithm Progress(Vinay Shivakumar and Jonathan Haas):

Siva has provided Vinay and Jon the information on the Moving Average Prediction Model. They have made the decision that the Moving Average Prediction Model is a good start. However it should not be the standalone model for prediction. They have found other models such as the MACD, RSI, and Exponential Moving Average and are currently doing more research on them. Siva has created the algorithm for the Moving Average Model. The main focus has been to expand predictions by using the RSI model. This will provide at least two predictions for the user to see and decide with. The moving average has been implemented and testing is occurring to see the validity of our implemented code.

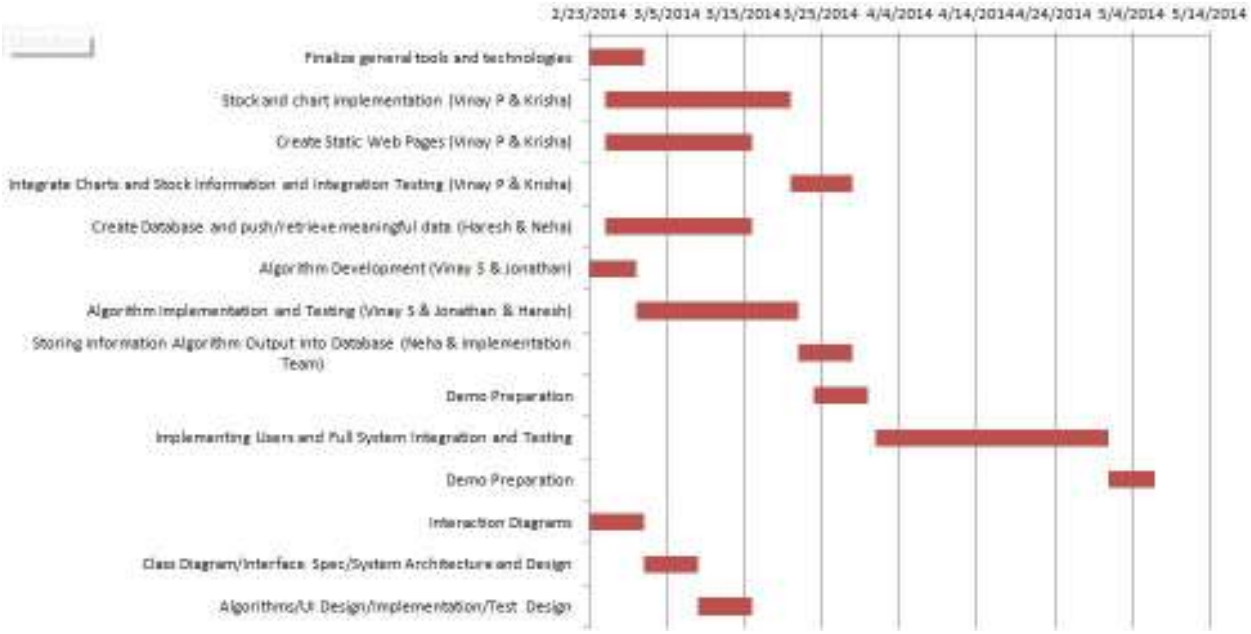


Figure 8-1: Future Plan of Work

The plan of work focuses mainly around the deliverables for this project. By demo number one we plan on having substantial progress the individual pieces developed by each sub group so the main focus for the second demo is creating users as well as full integration and testing. Part of what will make this plan of work successful is the specific roles that each sub team is being given and allowing each to master their own art by the time the first demo arrives, allowing for further development to occur thereafter as a larger group. It is important that we use the first week of finalizing general tools and techniques in order to ensure all planned technology will be able to be integrated into a working final product. This decision will be based off of research of previous projects that have been developed in the past as well

as our own personal experiences and intuition. This plan is currently being met with no delays or setbacks that have been observed thus far. This is still much coding to do and a lot of potential for technical setbacks, however we are confident that we can stick to this time table. This report was completed with some level of difficulty due to the coordination issue over spring break however, each sub team's individual component of this report were submitted early, the only set back was with testing. Most of the issues occurred due to the fact that we have not yet fully completed the code of our application so the exact implementation is still up in the air, but the concepts will still apply and that is what we have included in this report for the unit and integration testing section. Expect to see these further refined as we continue to develop our software.

9 REFERENCES

1. http://www.investopedia.com/terms
2. http://en.wikipedia.org/wiki/Stock_market_prediction
3. http://www.dummies.com/how-to/content/stock-investing-for-dummies-cheat-sheet.html
4. Sarkar, Asani. “Equity Market Valuation”.
5. *Master Trader – Top Strategies for Market Success.* Tigrent Learning. Print.
6. https://en.wikipedia.org/wiki/Stock
7. http://en.wikipedia.org/wiki/Moving_average
8. http://www.investopedia.com/terms/r/rsi.asp