STOCK MARKET SIMULATOR SYSTEM SPECIFICATION DOCUMENT

Group GWMF

Project Monitor: *Sebastian Coope*

Demonstration Reviewer: *Sebastian Coope*

Group Member: *Jin Zhang, Jingzhi Gong, Linhui Lu, Jinwei Zhang, Qihao Feng*

Introduction

This document is the final version of the Stock market simulator System requirement. The purpose of this document is to present the original specifications of the system, such as the mission statement, system boundary, use cases, function requirements and project plan; as well as the achieved requirements, the unachieved pointes and the additional features of the system.

Project Description

- **Potential customer**

The project is aimed at the customers who are interested in learning stock knowledge, or students who wish to practice stock trading skills in real life.

- **Mission statement**

The Stock market simulator System aims at crawling and storing real-time stock information with a database implementation, predicting the future trends of the stocks by applying machine learning algorithms, providing an User Interface and emulating the stock on-line transactions in real life to help customers who are interested in learning stock knowledge, or students who wish to practice stock trading skills in real life.

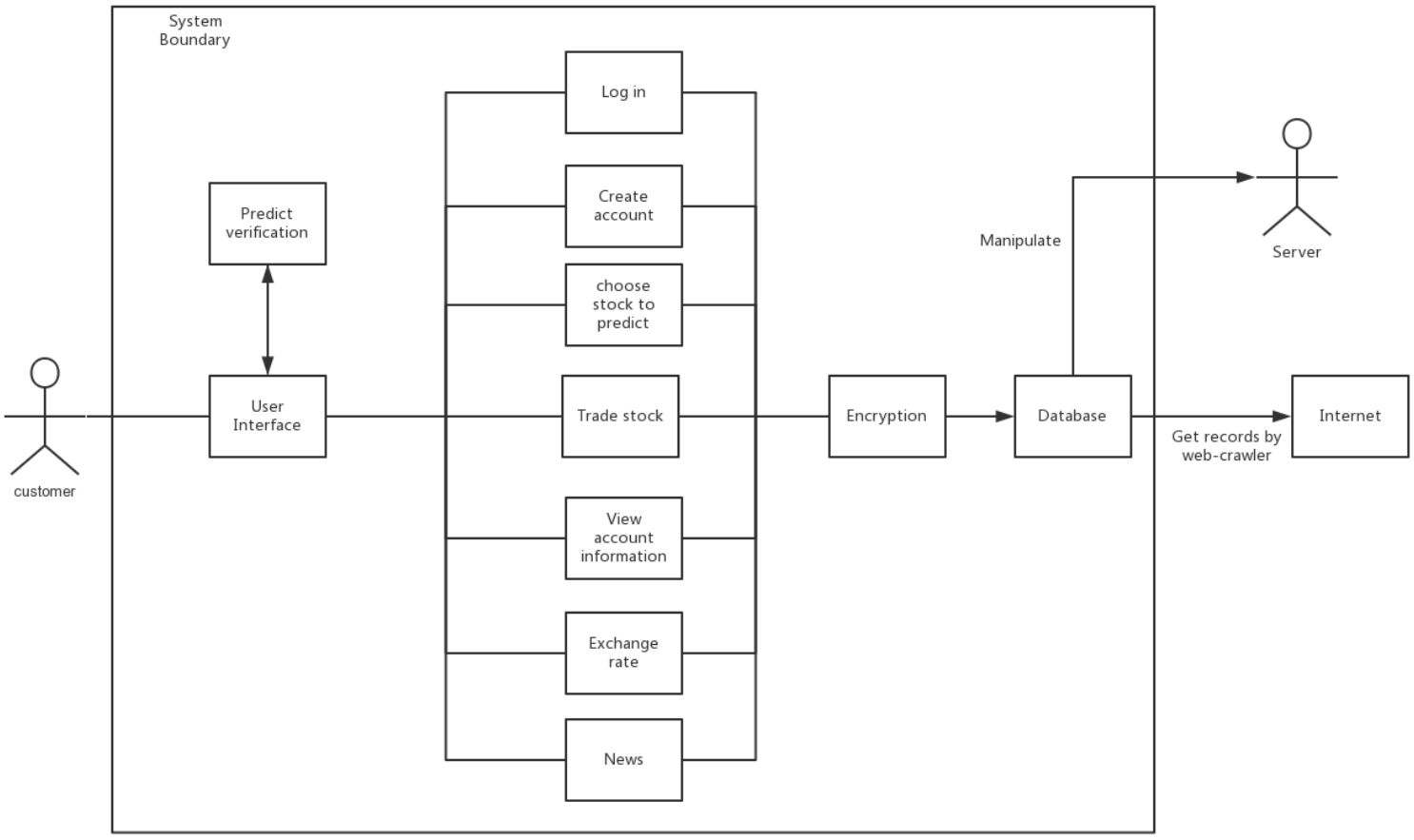
- **Mission objectives** [1]

1. To crawl stock information from Yahoo Finance website.
2. To crawl real-time update stock information.
3. To provide a sign up service for a new user of the system.
4. To let the users check their account information.
5. To encrypt the user information to maintain the privacy.
6. To implement a Database to store and manipulate stock and user data
7. To provide a fast system which respond critical transaction within a reasonable time limit.
8. To provide a simplified user interface so that a non-professional customer can use it.
9. To let the user top up using his/her bank account.
10. To provide a predicting service for users to predict the future trend of a stock.
11. To verify the prediction, which means to compare the prediction with the actual stock trend.
12. To provide extra functions such as News section, Exchange rate section etc.

Note that this section is constructed refer to Figure 6.8 - Mission objectives for stayHome database system of Connolly and Begg (2004).

Statement of deliverables:

- **System boundary diagram**



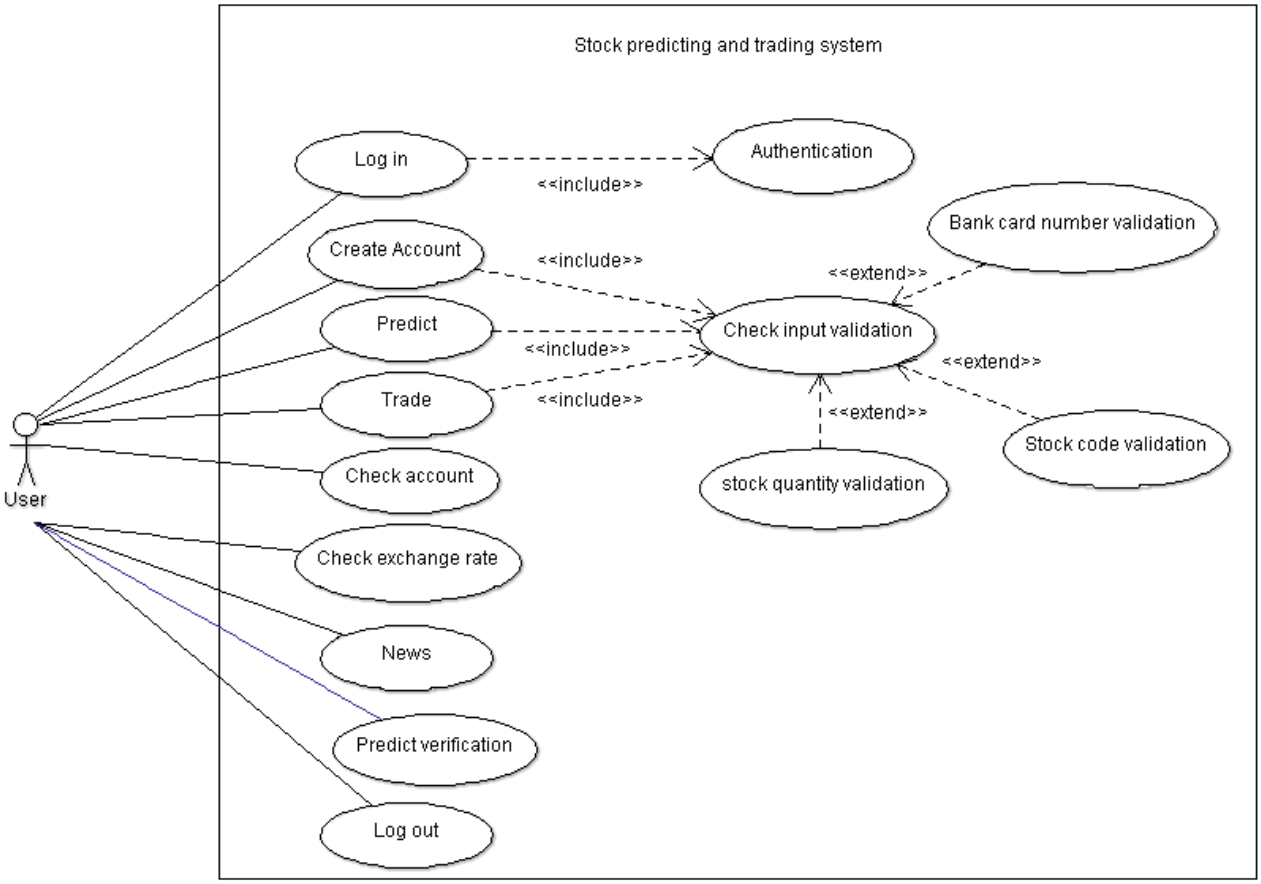
The major elements inside the system are the dynamic database and the main functional program. The major elements outside the system are customer who use the stimulator system, the server which manipulate the database and the internet which send record by the respond of the database. Customers may interacts with the system through user interface, all the function of the system can be used by customer through a particular interface. Database is manipulated inside server. As part of the stimulator system, database may get records from internet dynamically.

- **Different User views**

The Stock market simulator system has two kinds of users – the Customer and the System administrator. In the customers' view, there are nine functions, which are: Login, Create Account, Predict, Trade, Check Account, Check Exchange rate, News, Predict verification and Log out.

Moreover, the functions of the System administrator are: Delete User Information, Cancel Transaction, View User Account and Check Transaction.

Details of the user view can be described by Use case diagrams:



- **Use Case descriptions**

Name	Log in
Description	User input username and password to log in
Pre-condition	User have account of the stock simulator system
Event flow	User input username and password; System check input validation; User success logs in or wrong input appear.
Include	Authentication

Name	Authentication
Description	Get user name and password then match them with the record in database
Pre-condition	User input username and password
Event flow	Accept user's input Check it with database record Successful login
Extends	Authentication invalid

Name	Create account
Description	New user input personal information to register
Pre-condition	User want to use system but do not have account
Event flow	User input information which system ask for System accept and store that information Account created successfully
Include	Check input validation

Name	Predict
Description	Processors draw a future trend about a specific stock based on linear regression algorithm
Pre-condition	User log in
Event flow	User input the specific stock code System accept input and return the data user request
Include	Check input validation

Name	Trade
Description	User can buy stock or sell the owned stock
Pre-condition	User login and have enough money
Event flow	User input the specific stock code and quantity they need Processor check the valid transaction then respond to user. System accept input and return success or not
Include	Check input validation

Name	Check input validation
Description	System accept users' input and check whether the transaction is valid or not
Pre-condition	User case
Event flow	Accept input Match the input with records or check its format Return true or false
Extend	Bankcard validation, stock code validation, stock quantity validation

Name	Bankcard validation
Description	Check card number valid or not
Event flow	Check input is 16 digits Return result

Name	Stock code validation
Description	Check input code exist or not
Event flow	Check input stock code exist or not Return result

Name	Stock quantity validation
Description	Check user have enough money to buy stock
Event flow	Check whether user can buy this amount of stock Return result

Name	Check account
Description	Get account information from database and present it to user
Pre-condition	User log in
Event flow	User click view bottom System get personal information from database and present to user

Name	Check exchange rate
Description	Get exchange rate information from system
Pre-condition	User log in
Event flow	User click exchange rate bottom System get information from database and present to user

Name	News
Description	Provide news to user
Pre-condition	User log in
Event flow	User click News bottom System get News from database and present to user

Name	Log out
Description	Give a safety way for user to log out
Pre-condition	User log in
Event flow	User click log out bottom System provide the pop-up windows to inform user User log out

Name	Delete user information
Description	Choose a user and delete his/her information from the database.
Pre-condition	Server is running
Event flow	Administrator choose a particular user account Choose the clear function
Include	Use case: Administrator authentication

Name	Cancel transaction
Description	Administrator cancel the illegal transaction
Pre-condition	Server is running Malicious transaction is detected
Event flow	Administrator choose the transaction Re-check the invalid transaction Choose cancel function
Include	Use case: Administrator authentication

Name	View user account
Description	Administrator choose a particular user account to view
Pre-condition	Server is running
Event flow	Administrator search a particular user ID Choose view function to review the transaction detail.
Include	Use case: Administrator authentication

Name	Check transaction
Description	Administrator selects a particular transaction to check and review.
Pre-condition	Server is running
Event flow	Administrator search in the transaction database Select a particular transaction to view
Include	Use case: Administrator authentication

Name	Check transaction
Description	Administrator selects a particular transaction to check and review.
Pre-condition	Server is running
Event flow	Administrator search in the transaction database Select a particular transaction to view
Include	Use case: Administrator authentication

Name	Administrator authentication
Description	Check if the identity of the administrator is valid or not
Pre-condition	Server is running
Event flow	Administrator input username and password; System check input and match them with the record encrypted in database;
Extends	Use case: Authentication invalid

Name	Authentication invalid
Description	Administrator authentication failed
Pre-condition	Server is running
Event flow	Administrator entered the wrong username or password

Functional requirement

- **Log in**

  The users who have signed up an account can enter their user ID and password to use the system.

- **Sign up**

The user can register an account of the system by filling the required information, such as name, password, phone and etc. The account information should be stored in the database.

- **Mine**

  The user can see his/her information such as funds and stocks. The program should get all information from database and display it on the main menu.

- **Quotation**

  The user can view a list of stocks, and choose one to see its quotation details. The program should provide visualization of stock data, using graphics like Candlestick graph, moving average graph.

- **Predict**

  The program should use machine learning algorithms to predict the future trend of a stock, using the stock data from the database. The predicting algorithms should have a certain level of accuracy. The result should be visualized by graph.

- **Verify**

  The program should be able to verify the prediction, by comparing the predicting values and the actual values.

- **Top up**

  The user can top up some funds into the program to buy stocks.

- **News**

  The program should provide stock-relative news for the user to explore.

- **Exchange rate**

  The program should promote the user to view the current Exchange rate in the world.

- **Encryption**

  The user information should be encrypted and have a certain level of safety.

- **Web Crawling**

  The crawler should web crawl stock information from the Internet. There should be two kinds of data crawled. Firstly, the historical data of the stock. Secondly, the stock daily information in real time. The stock information data should be got by web-crawler automatically and transferred into the readable data and then stored in the database.

- **Server**

  The server should be deployed on a remote host so that the users can access it from a private computer.

- **Database**

  Database plays a significant role in this simulator system. It should be able to store and manipulate all the record that users need as well as information of each user.

- **Data visualization**

  The stock history data as well as the prediction data should be visualized as graphs, so that the user can easily read.

Requirements that have been implemented:

- **Log in**

```
public Main(){
        try {
                Registry registry = LocateRegistry.getRegistry(1099);
                queryPro = (queryProcessor) registry.lookup("queryProcessor");
        } catch (RemoteException | NotBoundException e) {
                e.printStackTrace();
        }
}
```

This will start client thread by creating instance of Main class using rmi remote procedure call.

```
public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("logInScreen.fxml"));

        Scene scene = new Scene(root);

        primaryStage.setScene(scene);
        primaryStage.show();
}
```

The start method will be executed after the launch(args). Strat method will load the FXML which have already designed by scene builder.

```
public static void main(String[] args) {
        //System.setProperty("java.rmi.server.hostname", "127.0.0.1");
        //new Main();
    System.out.println("connected");
    launch(args);
}
```

```css
.font{
    -fx-text-inner-color: white;
}

.button:hover{

    -fx-background-color: transparent;
}


.button{

    -fx-background-color:  #ea4443;

}

.container {
    -fx-background-image: url("SystemImage\bar.jpg");

    -fx-background-repeat: no-repeat;
}
```

In order to make the system more easy to use CSS file is used to beatify out system. The picture above is the CSS file for changing font and the cursor effect.

```
List<String> user_info =
Main.getQueryPro().viewPersonalInformation(Integer.valueOf(user_id));

        System.out.println(user_info);
        if(user_info.isEmpty())
                isValid = false;
        if(pw.equals(user_info.get(1)))
                isValid = true;

        if (isValid) {
        errorMessage.setText("welcome " + user_info.get(0));
        // e -> primaryStage.setScene(scene2)

        //close the log-in window
        Stage a = (Stage) login.getScene().getWindow();
        a.close();

        //open the main window
        Stage primaryStage = new Stage();
        Parent root2 = FXMLLoader.load(getClass().getResource("mainMenu.fxml"));
        Scene scene2 = new Scene(root2);
        primaryStage.setScene(scene2);
        primaryStage.show();
```

Once the main method is started it will load up a FXML file in a root node and show in a stage.

- **Sign up**

```
@FXML
  public void signinButtonClick() throws Exception {

     //close the log-in window
```

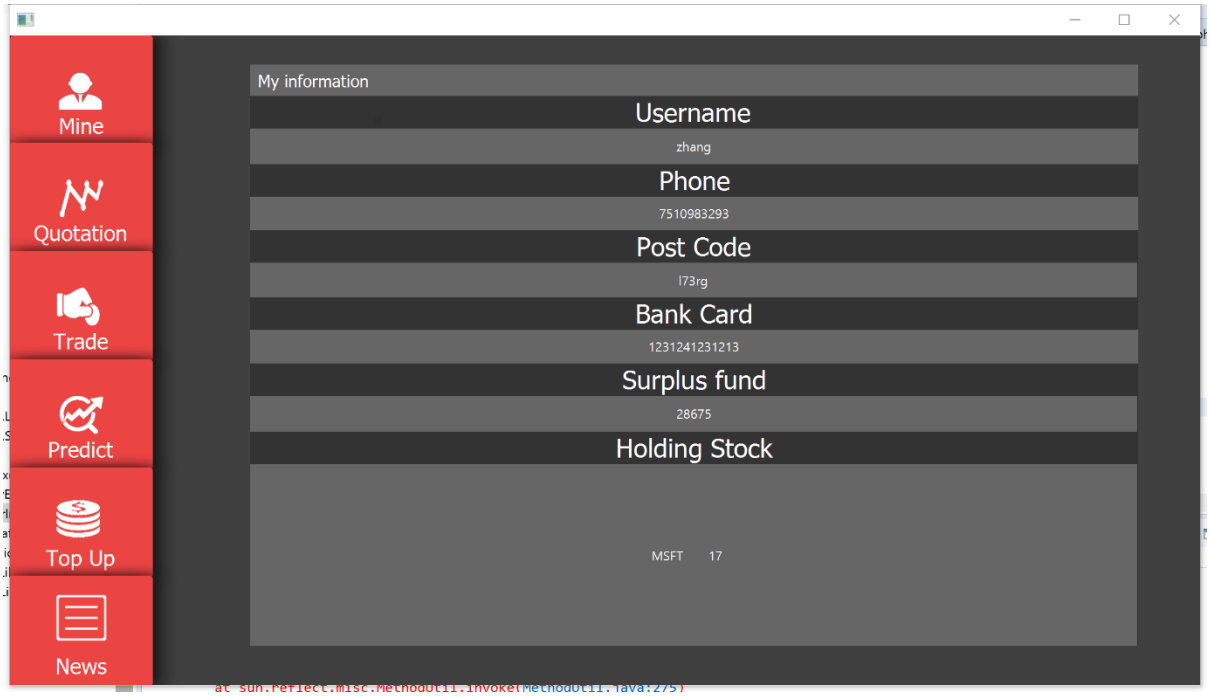
```
    Stage a = (Stage) login.getScene().getWindow();
    a.close();

    Stage primaryStage = new Stage();
    Parent root2 = FXMLLoader.load(getClass().getResource("profile.fxml"));
    Scene scene2 = new Scene(root2);
    primaryStage.setScene(scene2);
    primaryStage.show();

}
```

Once user finish logging in, and click the log in button, this event will trigger and generate another stage. A new fxml file is load into the new stage.

- **Mine**



Once user enter main menu, system will automatically get the userId and hold it. When user click mine button, system will use userId to match the information exist in database and extract them from database. Then display them in the window to the user which can help them better have knowledge about their account.

```
void mineInformation(ActionEvent event) throws NumberFormatException, RemoteException {

    information.setVisible(true);

    Trade.setVisible(false);

    highestRise.setVisible(false);

    lowestRises.setVisible(false);
```

```
        priceGraph.setVisible(false);
        stockDetail.setVisible(false);
        predictGraph.setVisible(false);
        topUpArea.setVisible(false);

        List<String>user_info=
Main.getQueryPro().viewPersonalInformation(Integer.valueOf(Main.getUser_id()));
        username.setText(user_info.get(1));
        fund.setText(user_info.get(3));
        phone.setText(user_info.get(4));
        postCode.setText(user_info.get(5));
        bankCard.setText(user_info.get(6));

        List<List<String>>user_stock=
Main.getQueryPro().viewPersonalStock(Integer.valueOf(Main.getUser_id()));
        String user_hold = "";
        for(int i=0; i<user_stock.size(); i++){
            user_hold += user_stock.get(i).get(1)+"\t"+user_stock.get(i).get(2)+"\t";
        }
        holding.setText(user_hold);
    }
```

- **Quotation**

  We implemented quotation function by using 4 part to display the information. They are stock list, graph container, confirm button and detail information display.

  First use enter quotation part by click on quotation button. Then user need to choose one stock from the list part and click the confirm button to submit the choice to the server. Once server get the data send from client. It will try to get price information from the database and use those data to generate a graph for user. Then, client will get the graph and put it to the graph container.

```
@FXML
void quotation(ActionEvent event) throws IOException {
    highestRise.setVisible(true);
    confirm.setVisible(true);
    priceGraph.setVisible(true);
    stockDetail.setVisible(true);
    information.setVisible(false);
    predictGraph.setVisible(false);
    topUpArea.setVisible(false);
    Trade.setVisible(false);

    List<String> allstock_id = Main.getQueryPro().getStockId();

    for(int i=0; i<allstock_id.size(); i++){
        stockListView.getItems().add(allstock_id.get(i));
    }

    byte[] readin = Main.getQueryPro().getStockHisChart("bac");
   // System.out.println("The stock id is: "+stockListView.getSelectionModel().getSelectedItem());
    InputStream in = new ByteArrayInputStream(readin);
    BufferedImage priceimage = ImageIO.read(in);

    ImageView img = new ImageView(SwingFXUtils.toFXImage(priceimage, null));
    img.setFitWidth(stockchart.getPrefWidth());
    img.setFitHeight(stockchart.getPrefHeight());
    stockchart.setGraphic(img);


}
```

- **Predictor**

  When a user use the prediction function, the program will create a prediction id that records the user id, stock id and date time in the database. We have set two algorithms to implement the prediction function, which are Linear Regression and Support Vector Machine.

  - **Linear Regression Algorithm**

  This algorithm needs to set a moving time window ($D_t = \{X_{t-q+1} .. X_t\}$) to predict the next chronological sequential stock prices X_{t+1}. We set **LinearDeviation** class to get the model deviation of the four input data (high, low, open, clost).

  Then the weight array will be trained by using the training data. Delta learning rule and gradient descending algorithm is used here.

```
for(int i=Constant.Window_Size;i<testData.length-Constant.Window_Size;i++){
    double predictData=0;

    for(int k=0;k<Constant.Window_Size;k++){
predictData=predictData+weights[k]*testData[i-Constant.Window_Size+k];
    }
    for(int k=0;k<Constant.Window_Size;k++){
weights[k]=weights[k]+Constant.Learning_rate*(testData[i-Constant.Window_Size+k]-
predictData)*testData[i-Constant.Window_Size+k];
    }
}
```

The specific calculating process is encapsulated in the class **Executor**. Finally, the prediction result will be write into a txt file for generating graphics.

- **Support Vector Machine**

The main body of the SVM prediction algorithm is written in Python. Therefore, we have set a **SVMPredictor** class to invoke the python program and transmit some parameters. Java.Runtime() is used here for running python file in the console.

```
Runtime r = Runtime.getRuntime();
String argv[] = {"cmd", "/c", "python", "SVMPredictor.py",String.valueOf(days),
String.valueOf(txtId), stockCode};
try {
    Process p = r.exec(argv);
} catch (IOException e) {
    ErrorExecutor.writeError(e.getMessage());
}
```

For implementing SVM algorithm, we need to import several packages, such as sklearn, numpy, MySQLdb, which are used for SVM calculating, read data from the database separately. Firstly, the SVM model should be trained by the stock data. And then the test data can be used to get the prediction result.

```
clf = svm.SVR();

clf.fit(high_train, high_y);

high_result = clf.predict(high_test);
```

Finally, the result should be written into a txt file for generating the graphic.

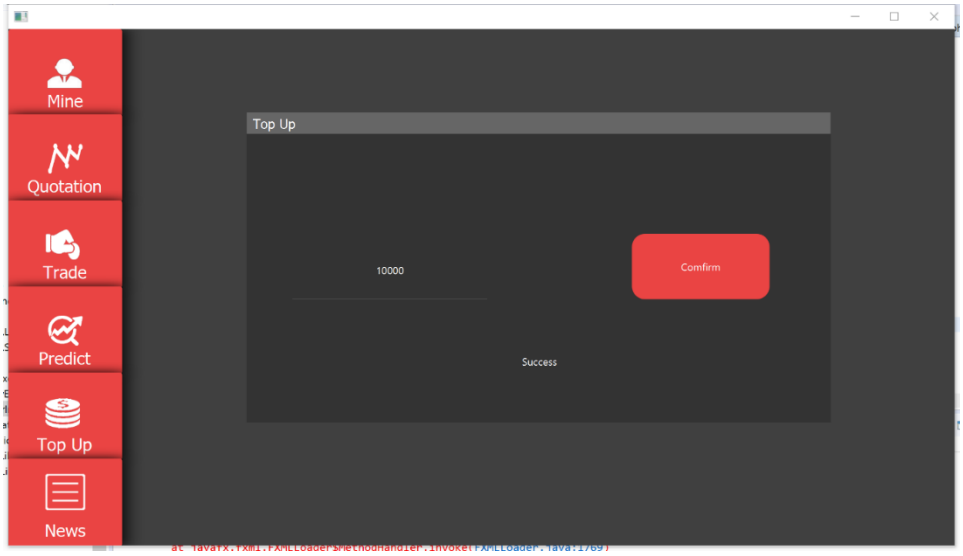
- **Verify**

As mentioned above, each prediction event will be given a unique predict id which has been recorded into the database with user_id, stock_id and date of prediction. Therefore, it is necessary for us to give a feedback about the prediction accuracy to the user after the prediction. For example, after ten days of prediction of “MSFT”(Microsoft), user will be able to view the verification diagram in the prediction panel. User should choose which verification to view, and the system will generate a diagram includes actual stock data and the prediction data.

- **Top up**

For top up function, user will use it by clicked the top up button. Once enter the window as shown below, input the amount of money that you need and confirm it. If the number is valid

then top up success and a message will show to user. Else, fail to top up and inform user to input number to try again.




```
@FXML
void confirmTopUp(ActionEvent event) {
    boolean d1 = topUpMoney.getText().isEmpty();
    if(d1){
        topUpError.setText("Please enter the number of money");
    }else{
        try{
            double money = Double.valueOf(topUpMoney.getText());
            boolean result = Main.getQueryPro().topup(Integer.valueOf(Main.getUser_id()), money);
            if(result){
                topUpError.setText("Success");
            }else{
                topUpError.setText("There's something wrong in the topup process");
            }
        } catch(Exception e){
            topUpError.setText("Please enter digits");
        }

    }
}
```

- **News and Exchange rate**

  The basic idea of News and Exchange rate are same. These two function will load a web engine and there is two different website been used. Which are https://uk.finance.yahoo.com for the news and http://www.xe.com/ for the exchange rate. The node is format by VBox and show by another stage.

```
@FXML
  void news(ActionEvent event) {
     WebView news=new WebView();
     WebEngine engine=news.getEngine();
     engine.load("https://uk.finance.yahoo.com");

     VBox root=new VBox();
```

```
        root.getChildren().add(news);
        Stage stage=new Stage();
        Scene scene=new Scene(root,800,500);
        stage.setScene(scene);

        stage.show();
    }
```

- **Encryption**

  There's seldom data that need to be encrypted before storing into the database except the usernames and the passwords. Therefore, BASE64 has been chosen as the encryption algorithm because it is encapsulated in the JAVA, and we can import it without import any other jar packages. The following codes show the encoding and decoding processes.

```
private String encode(String item){
byte[] tmp = item.getBytes();
            Base64 encode = new Base64();
            tmp = encode.encode(tmp);
            return new String(tmp);
}

private String decode(String item){
            byte[] tmp = item.getBytes();
            Base64 decode = new Base64();
            tmp = decode.decode(tmp);
            return new String(tmp);
}
```

- **Web Crawler**

  We need to crawl two kinds of stock data from the website. One of them is the historical data, and another one is the daily stock data. Therefore, we have set a thread-pool which contains two threads that initialize these two crawlers. The following paragraph will show the steps of setting two crawlers.

  Step1. Set up a thread pool

```
private static ExecutorService crawlers;

public static void startCrawler(){
                    crawlers = Executors.newFixedThreadPool(2);
```

  Java has several method to realize multi-threads. FixedThreadPool is chosen here due to the number of crawlers is fixed. And the thread pool will help us to manage the threads of running, waiting, sleep, invoke and other.

  step2. Execute two crawlers.

```
try {
        crawlers.execute(new HistoricalCrawler());
        crawlers.execute(new DailyCrawler());
} catch (SQLException e) {
        ErrorExecutor.writeError(e.getMessage());
}
```

Both **HistoricalCrawler** and **DailyCrawler** classes implement the interface **Runnable** to run the crawler. And the crawlers' objects are instantiated here.

Step3. Shut down the th read pool

```
crawlers.shutdown();
```

Finally, the thread pool should be shut down.

- **HistoricalStockCrawler**

This class is set for historical stock data. Yahoo historical data API is used here.

```
public static final String YAHOO_FINACE_URL = "https://table.finance.yahoo.com/table.csv?";
```

Parameters need to be added after the "csv?". For example, if we need to get IBM's stock data between 2008-01-01 and 2017-04-28, the parameters need to set as the following form:

Csv?s=IBM&a=2008&b=01&c=01&d=2017&e=04&f=28

IO stream is used here for sketching data from the respond information.

```
URL myURL = null;
URLConnection con = null;
InputStreamReader ins = null;
BufferedReader in = null;
try{
        myURL = new URL(url);
        con = myURL.openConnection();
        ins = new InputStreamReader(con.getInputStream(), "UTF-8");
        in = new BufferedReader(ins);
```

However, we can get "high, low, open, close, volume" from the respond information of the stock. The returned information should be stored in the database. Therefore, a connection with the database is set here. The form of respond information and the sql order is shown as the following:

```
2008-04-01,41.98,40.91,41.07,41.73,1272200

Insert into ibm values ('2008-04-01','41.98','40.91','41.07','41.73','1272200')
```

- **DailyCrawler**

This class is used for crawling real-time from the API. However, real-time, free and accurate three cannot get the same time. Therefore, we choose to get the data at a latency of 5 minutes compared with the actual stock market. The API provided by YAHOO is shown here:

```
String YAHOO_FINACE_URL_TODAY = "http://finance.yahoo.com/d/quotes.csv?";
```

As the historical crawler, the parameters need to be set here for specifying which stock and what information we need to get from the API.

```
Csv?&s=IBM &f=sd1l1yr
```

&s represents the corporation's stock code, and &f means the types of stock information.

S – stock code

D1 – last trade date

L1 – last trade price

Y – dividend yield

R – PE ratio

The respond information will get sketched by using IO stream, and we need set the crawler get the stock data every 5 minutes automatically.

```
try {
        Thread.sleep(1000*60*5);
} catch (InterruptedException e) {
        ErrorExecutor.writeError(e.getMessage());
}
```

- **Server**

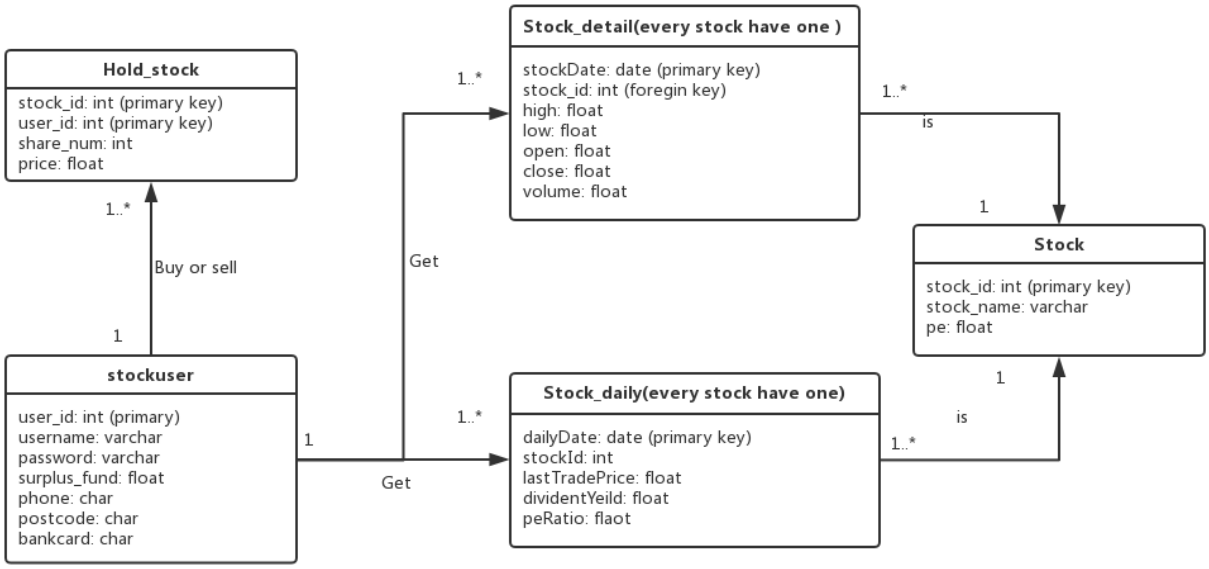
We intended to set our server on the Godaddy server. Unfortunately, Godaddy haven't support JVM since 2012. Then we try to set our server on the Amazon EC2. Because the communication skills involved in our program is java rmi, port 1099 and tcp protocol are quite significant for us.

```
LocateRegistry.createRegistry(1099);
ServerImpl remote = new ServerImpl();
```

Unfortunately again, we failed to telnet our Amazon EC2 server address on the port 1099. Hence, we decided to set our server on the school's server. However, it still failed due to the school's firewall. Finally, we decided that the local version of our program should be realized first. Therefore, our server is set on the local host.

```
Naming.rebind("rmi://localhost/queryProcessor", remote);
```

- **Database**



The ER-diagram describes the whole database table and their relations between each other. The user can buy or sell to modify the data in Hold_stock table. The user can get stock information from Stock_daily and Stock_detail. Records in Stcok_detail and Stock_daily its corresponding stock_id and name which store in the Stock table.

- **Data visualization**

The data visualization is implemented using Python 2.7. We applied Anaconda 2.4 for developing the graph generator. Furthermore, the plotting library Matplotlib and array processing package Numpy are used.

The data needed is loaded from txt file, which is generated by the web crawler. The plotted graph will be stored at local for the program to use.

```
#read the attributes
date, high, low, open, close, volume = np.loadtxt(
                stockFile,delimiter='\t',skiprows=1,usecols=(1,2,3,4,5,6),
                converters={1:mdates.strpdate2num('%Y-%m-%d')},unpack=True)
```

The graphs that can be generated are: Candlestick graph, moving average, prediction graph and volume graph. For example: the Candlestick graph is plotted using the Matplotlib.finance package:

```
#draw the candlestick graph
candlestick_ohlc(ax1,candleAr,width=1,colorup='#008B00',colordown='#FF3030')
```

The codes to calculate and draw the Moving Average chart for 10, 30 and 60 days:

```
def movingaverage(values,window): #the function to calculate moving average
    weights = np.repeat(1.0,window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas
```

```
#time periods of moving average
MA1 = 10
MA2 = 30
MA3 = 60
```

```
#draw the moving average graph
ax1.plot(date[-SP:],Av1[-SP:],label=label1,linewidth=1.5,color='#1C86EE')
ax1.plot(date[-SP:],Av2[-SP:],label=label2,linewidth=1.5,color='y')
ax1.plot(date[-SP:],Av3[-SP:],label=label3,linewidth=1.5,color='#D15FEE')
```

Project plan

The project will be developed according to a Project plan, which identifies the activities, internal and external stones and their dates. The Project plan contains three main stages – the preparation stage, the design stage, and the implementation stage.

- **Preparation stage**

In the Preparation stage, the activities aim to acquire background knowledge about the project and then form a Requirement document. The background knowledge involves 4 aspects. First, to understand the project better, we will learn basic information about stock market, such as the way to buy or sell stock in real life and the way to read stock quotations. Second, we have to learn about database programming skills, since this Stock trading and the predicting system will include a database implementation. A user interface is needed for the system, so we need to research GUI implementation. We should have clear knowledge about stock predicting algorithm for the predicting function of our system. After the research, we reach an internal milestone, which is the background information on the project. The next activity is to write a requirement report, to conclude our knowledge so far and plan for future moves. To clearly state the requirements, 3 important diagrams are needed, which are the Gantt chart, Use Case, and System Boundary diagram. The other parts will be discussed and finished by all group members. After the report milestone, we will prepare for a review meeting of this stage.

- **Data requirement** [2]**:**

The data required by the system is stock information. This will include: Open: High, Low, Market cap, PE ratio (TTM), Dividend & yield, Beta, stock graph, candy graph (Yahoo! Finance, 2010).

1. Open: the price at the beginning of market day
2. High: the highest price in market day
3. Low: the lowest price in market day
4. Close: the price at end of market day
5. PE (Price-Earnings Ratio): ratio of a company's stock price to the company's earnings per share.

6. Candlestick Charts: it is a style of financial chart used to describe price movements. Each "candlestick" typically shows one day price changing containing open, close, high and low.
7. Moving Average: A moving average (MA) is a trend-following or lagging indicator because it is based on past prices.

- **Design stage**

The second stage is Design. We have planned a sequence of activities to design the Stock predicting and trading system. To begin, we do the architectural design, in which an overall structure and subsystems will be produced so that the later designs can be consistent. For example, the basic structure of Database subsystem, the Stock predicting subsystem and the Stock trading subsystem etc. Then, we design the interfaces of the subsystems, which specifies how this subsystem will interact with others, such as how the Stock trading subsystem will operate data in the Database subsystem. We can also begin to design the GUI of the system at this point, since the basic functions are known. This process is expected to continue for a long period of time, to cater for any changes. After the Interface design, the subsystem components are designed, which corresponds to specific system functions. For instance, the stock predicting function is a component. The next phase is to design the detailed data structure to implement the functions, such as how to store and manipulate the stock information. After that, we design the specific algorithm to predict the future trend of the stock, as well as the other algorithms of the system. Once this is done, we reach an internal milestone, which indicates that all system design is finished. During the whole designing stage, a Design report will be documented. This is an external milestone. Also, a presentation should be prepared to demonstrate our design to the reviewer.

- **Implementation stage**

Finally, we implement all the design in the Implementation stage. The different system functions can be concurrently implemented since they are designed as separate components and can be grouped together later. This includes the basic functions, GUI and stock predicting algorithm implementation. Note that the basic functions includes log in function, log up function, user information viewing function etc. In case we don’t have enough hands, part of the team members will develop the basic functions first, and then focus on the database implementation. Once a component is finished, the Component testing and refinement phase can be started. This phase continues until all components are done and tested. Then, we shall group all components together into a Demo System, which can be recognized as an internal milestone. The next phase will be Integration testing and refinement using Eclipse built-in tool ANT and JUnit, in which we test the system as a whole and refine the functions further. After the Demo material submission milestone, we will give a presentation to the reviewer to demonstrate the use of our Demo system, and then do the final refinements. This is followed by the final milestone – the Portfolio submission deadline. The major challenges carrying out the project is that we need to learn how to nest a database inside the function code and how to build an elegant user interface using swing component. Therefore, the skill of programming in python

and testing skill using eclipse will be required. Besides that skill of analysing the stock information will also be needed by background research.

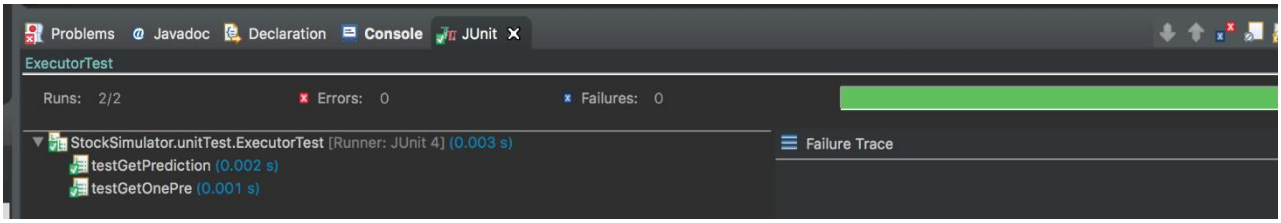
- **Testing requirements**

During the development, we implemented component testing to testify every part of code that we write and then assemble it. White box testing are used for the component test.

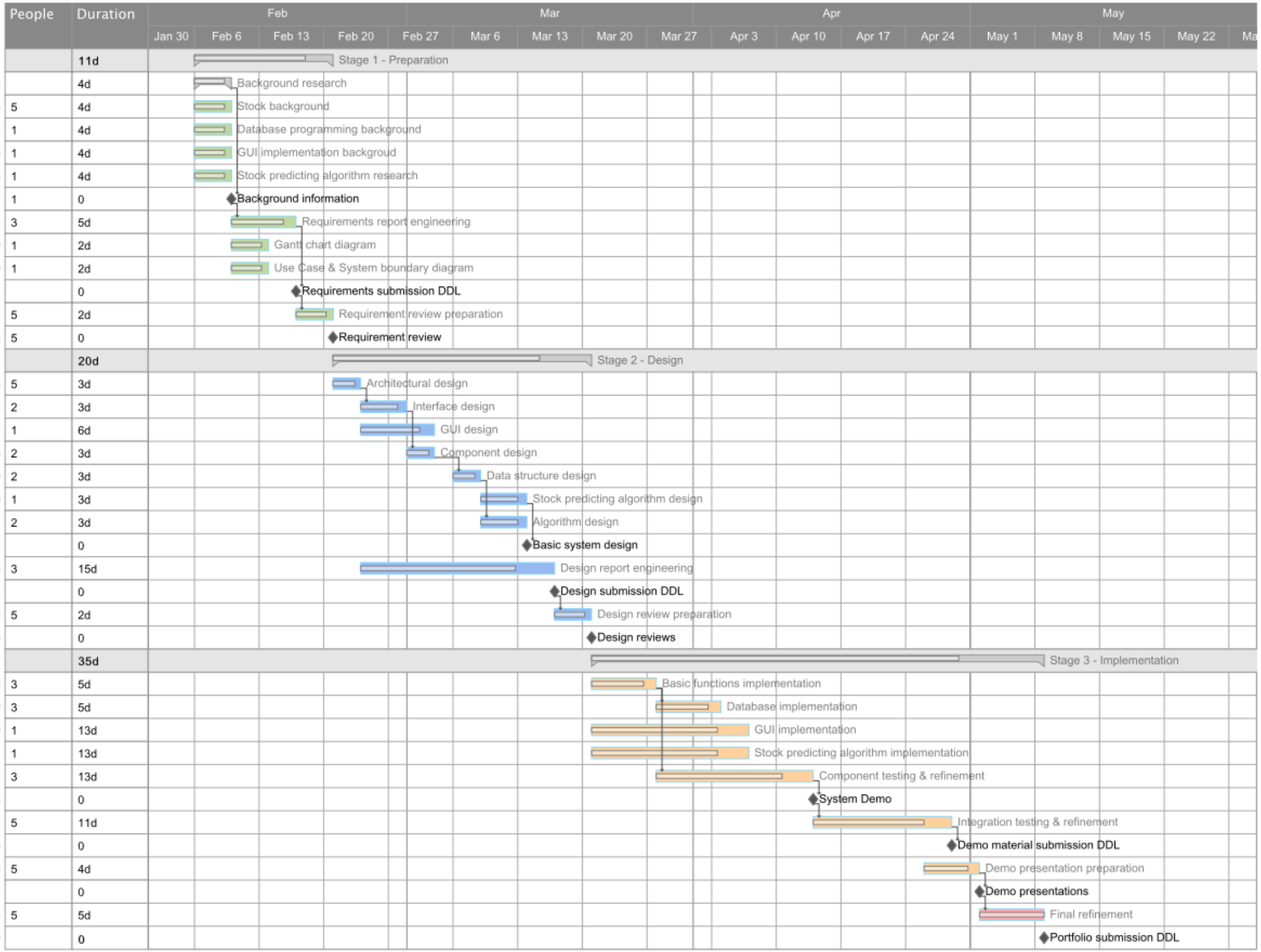
For example

```
1  package StockSimulator.unitTest;
2
3  import static org.junit.Assert.*;
8
9  public class ExecutorTest {
10     @Test
11     public void testGetOnePre() {
12         double doubleArray[] = new double[400];
13         Executor exe = new Executor(0, doubleArray, 0);
14         assertEquals(exe.getOnePre(), (double) 0,0.0);
15
16     }
17
18     @Test
19     public void testGetPrediction(){
20         double doubleArray[] = new double[400];
21         Executor exe = new Executor(0, doubleArray, 0);
22         Float[] pres = new Float[0];
23         assertEquals(exe.getPrediction(),pres);
24     }
25 }
26
```

testGetOnePre() is the method to test the result of the getOnePre method in linear regression algorithm. The test case is set as above the result of this test case is supposed to be 0. After testing, the result is same as the desired result, so that this test case succeed.



- **Gantt chart diagram**


People
Duration
Feb
Mar
Apr
May
Jan 30
Feb 6
Feb 13
Feb 20
Feb 27
Mar 6
Mar 13
Mar 20
Mar 27
Apr 3
Apr 10
Apr 17
Apr 24
May 1
May 8
May 15
May 22
1 11d Stage 1 - Preparation
2 4d Background research
3 5 4d Stock background
4 1 4d Database programming background
5 1 4d GUI implementation backgroud
6 1 4d Stock predicting algorithm research
7 1 0 Background information
8 3 5d Requirements report engineering
9 1 2d Gantt chart diagram
10 1 2d Use Case & System boundary diagram
11 0 Requirements submission DDL
12 5 2d Requirement review preparation
13 5 0 Requirement review
14 20d Stage 2 - Design
15 5 3d Architectural design
16 2 3d Interface design
17 1 6d GUI design
18 2 3d Component design
19 2 3d Data structure design
20 1 3d Stock predicting algorithm design
21 2 3d Algorithm design
22 0 Basic system design
23 3 15d Design report engineering
24 0 Design submission DDL
25 5 2d Design review preparation
26 0 Design reviews
27 35d Stage 3 - Implementation
28 3 5d Basic functions implementation
29 3 5d Database implementation
30 1 13d GUI implementation
31 1 13d Stock predicting algorithm implementation
32 3 13d Component testing & refinement
33 0 System Demo
34 5 11d Integration testing & refinement
35 0 Demo material submission DDL
36 5 4d Demo presentation preparation
37 0 Demo presentations
38 5 5d Final refinement
39 0 Portfolio submission DDL

In the Gantt chart diagram, the horizontal coordinates represent the timeline, while in the vertical coordinates, each row records a task, its start time and ends time, with a row number, the number of people involved in and the time duration etc. The row number is recorded on the table, the first attribute represents the number of people involved in the task, the second column is the time duration of the task, and then the right-hand side is the graphical representation of the task sequence and time duration. Note that each task bar has a smaller stick inside, which denotes the time slack of the task. Moreover, the black rhombus stands for milestones, and the sequence of tasks depends on the start time of the tasks. The black lines with an arrow connecting two tasks mean that one task must finish before another.

For instance, in the third row, the task name is 'Stock background'. From the diagram, we can see this task will last for approximately 4 days. There are 5 people involved in, which means all team members have to research about the stock background. The task starts on 6th of February and should end on 10th Feb. It has the same start time with tasks in the 4th, 5th, and 6th line. This means they can be executed concurrently.

Bibliography:

[1] T. M. Connolly and C. E. Begg, *Database solutions. [electronic book] : a step-by-step guide to building databases*(Online access with purchase: Dawsonera). Harlow : Pearson Education Ltd. / Addison-Wesley, 2004.

[2] Yahoo! Finance. *How to Read a Stock Quote.* [Online] Access: http://todayforward.typepad.com/todayforward/2010/04/how-to-read-a-stock-quote.html. Apr 28, 2010. Last-access: Fab 16, 2017

[3] Microsoft, "Model-View-Controller," 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/ff649643.aspx.