UNIVERSITÉ Concordia UNIVERSITY

SOEN 6461
SOFTWARE DESIGN METHODOLOGY
winter 2017

Deliverable 1

Declaration

We, the members of the team, have read and understood the Communal Work Protocol, and agree to abide by it, without any exception, under any circumstances, whatsoever.

Team H

Nikita Baranov
Gurpreet Raju
Kirtan solanki
Ajeeta kaveti
Navdeep
Jyotsna rana
Rajan Takiar

1 TABLE OF CONTENTS

2 INTRODUCTION

The purpose of the system is to provide bridge between client and stock market. Customer can view the chart of loser and gainer from stock market through system. Methodology of system allows the client to select specific time for stock movement with chart so client can conclude whether to purchase stocks from share market.

2.1 PROJECT OVERVIEW

The purpose of this document is to present the design of the stock market analysis, It will provide details on the architectural design, the software interface design, and the internal module design. The architectural design will describe the software architecture that was chosen for the system and a class diagram of this architecture. The software interface design will have screen shots of the graphical user interface and how the users interact with the system. Finally, the internal module design will describe in detail the different modules through the use of class diagrams.

2.2 PROJECT SCOPE

This project made up of creating moving average charts that depends on stock market by selecting specific time. The system modules include client's registration, edit profile of client, select moving average period of system and view the chart of the system. So, through the system client can see the moving average of stock market and make decision.

2.3 DOCUMENT PREVIEW

This document uses the IEEE standard Times New Roman size 12 fonts. Bold/Underline is used to convey important terms. Every requirement statement will have its own priority.

3 OVERALL DESCRIPTION

3.1 PRODUCT PERSPECTIVE:

The Stock market analysis application being developed for profit RUS Software Engineering Company is a new self-contained product. It contains the following features:

Moving averages: user can add simple moving averages on chart, and can customize the time frame for each one.

chart: Displays the stock analysis

indicator: Allows you to select an indicator.

3.2 PRODUCT FEATURES

3.2.1 Moving average:

1. Select moving average
2. Adjust time

3.2.2 Chart:

Create a new chart

3.2.3 Indicator:

Displays the moving average

3.2.4 User Classes and Characteristics:

The users of the stock market analysis application will vary from Software Engineering students to Software Engineering professionals. We will categorize the users of the stock market analysis application as Power Users, Intermediate Users and End Users.

3.2.5 Operating Environment

The operating system will be Microsoft Windows 7 running java virtual machine. Eclipse SDK Version 3.2.1 will be used. Other software components include Eclipse Java Development Tools, Eclipse Plug-In development environment, etc.

3.2.6 Design and Implementation Constraints

The programming language used to implement the stock market analysis will be Java. The user will have limited access to login and choose the indicator which includes: simple moving average will have limited access. There will also be some user restrictions on the charts. Future upgrades and maintenance of the software will be handled by profit RUS Software Company

3.2.7 User Documentation

The user documentation standard we will use is List or Reference. The commands will be listed alphabetically and indexed. This standard will appeal to the advanced user.

3.2.8 Assumptions and Dependencies:

The development requires the Microsoft Windows operating system. There will be 2 user types: Registered users and unregistered users There will be limited access to the system. At this point, it is not clear whether we will use software components from another project.

3.2.9 Budget

The objective of the project plan and measure is to give lucid view of the project which helps to increase the project planning and increase the visibility of the project. Plan and measurement also give the status of the project and inform the sponsor and senior about project.

Time schedule

Velocity of project = no of story point / no of story point completed by Iteration

No of story point = 8

No of story point completed by iteration = 2

Velocity = 8/2 = 4 iterations.

Budget schedule

Iteration	Number	Days	Hours	Men	Estimated Cost	Margin
Iteration	0	12	84	7	11760	+-10%
Iteration	1	10	70	7	9800	+-10%
Iteration	2	12	84	7	11760	+-10%
Iteration	3	12	84	7	11760	+-10%
Total	3	34	238	-	45080	

Detailed scheduling

1) Scope planning meeting

Project team member present the stories that require to be completed in that iteration. In this meeting project, brief introduction is given. Time for this meeting is 1 hour.

2) Stand-up meeting

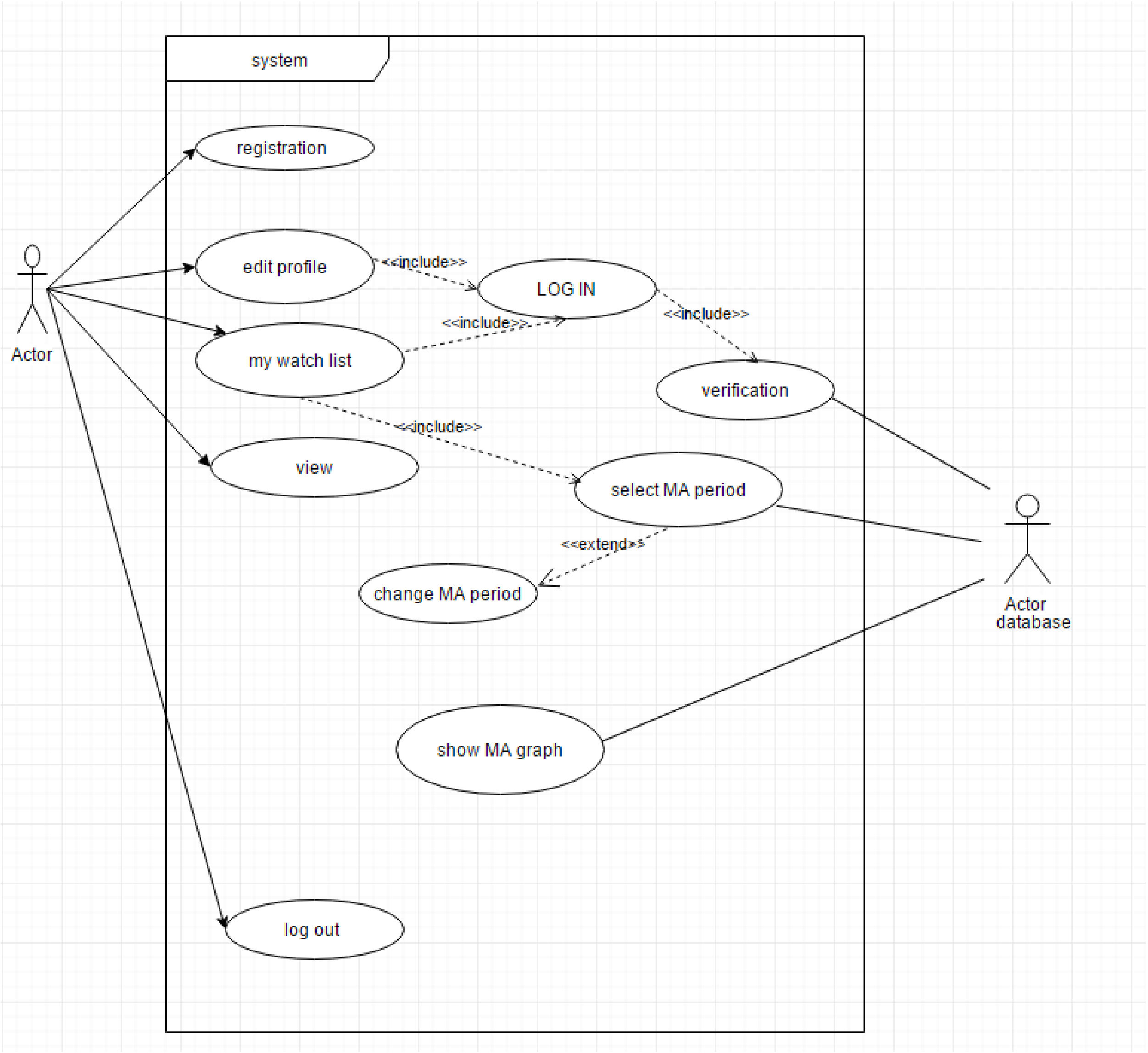
This meeting is held every day and discuss about previous day work and work detail about present day. Write down Limitation and new other obstacle will encountered by team. Time for this meeting is ten minutes.

3) Backlog grooming meeting

Purpose of this meeting is to identify new user stories that added by team. This meeting held in each iteration. Time limitation for this meeting is 1 hour.

4 SYSTEM FEATURES

4.1 USE CASE DIAGRAM



4.2 FULLY DRESSED SCENARIOS FOR ABOVE SYSTEM.

Number	1
Name	Registration
Priority	First

Primary actor	Client
Secondary actor	Database
Summary	Client able to register
Preconditions	Internet connection, email id
Postconditions	Client success fully registered
Trigger	Client
Scenarios	User will open ---------- User will register in the system. User provides email id and password. User will able to view his/her profile. User allow to edit profile.
Exceptions	3.1 customer registration is not valid 3.2 customer provides another email-id.

Number	2
Name	Log in
Priority	First
Primary actor	Client
Secondary actor	Database
Summary	Client able to login
Preconditions	Internet connection, email id, password
Postconditions	Client success fully logged in
Trigger	Client clicks on login button
Scenarios	User will login in the system. User provides email id and password. User will able to view his/her profile. User allow to edit profile.
Exceptions	2.1 customer login is not valid. 2.2 customer provides incorrect email-id. 2.3 customer provides incorrect password.

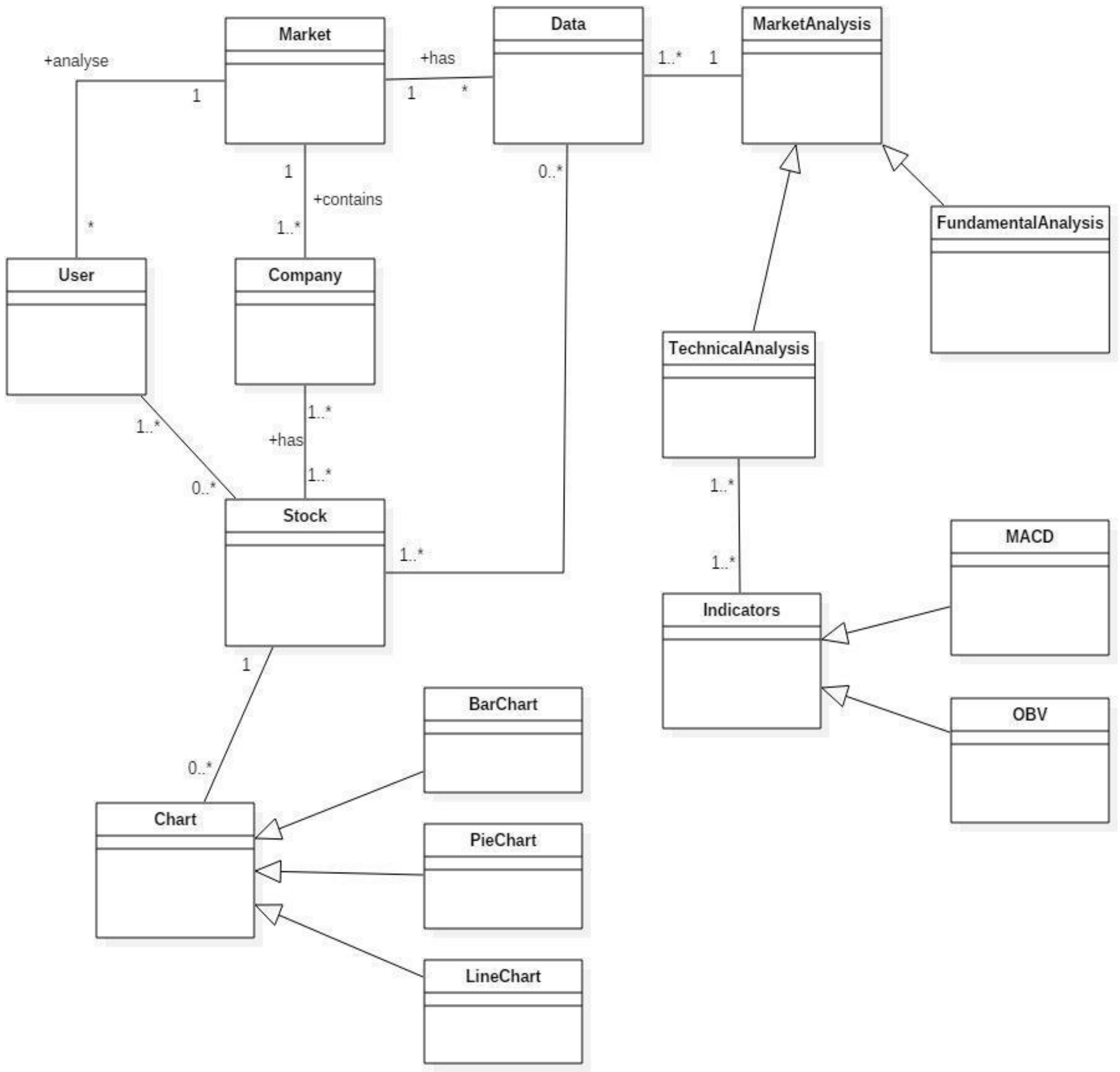
Number	3
Name	My watch list
Priority	Second
Primary actor	Client
Secondary actor	Database
Summary	Client see MA chart
Preconditions	Valid login
Postconditions	Client able to see price, MA chart
Trigger	Client write ma period
Scenarios	User will login in the system. User will add MA period. User can change MA period. User can see MA graph.
Exceptions	1.1 customer login is not valid. 2.2 customer provides incorrect MA period.

Number	4
Name	Change MA period
Priority	Third
Primary actor	Client
Secondary actor	Database
Summary	Client can change indicator of MA period.
Preconditions	Valid login
Postconditions	Client able to see price, MA chart
Trigger	Client write MA period.
Scenarios	User will login in the system. User will add MA period. User can change MA period. User can see MA graph.

Number	5
Name	Log out
Priority	First
Primary actor	Client
Secondary actor	Database
Summary	Client able to logout
Preconditions	Logged in
Postconditions	Client success fully logged out
Trigger	Client clicks on logout button
Scenarios	User will login in the system. User provides email id and password. User will able to view his/her profile. User allow to edit profile. User will add MA period. User can change MA period. User can see MA graph User log out from system.
Exceptions	customer unable to log out due to low internet connection.

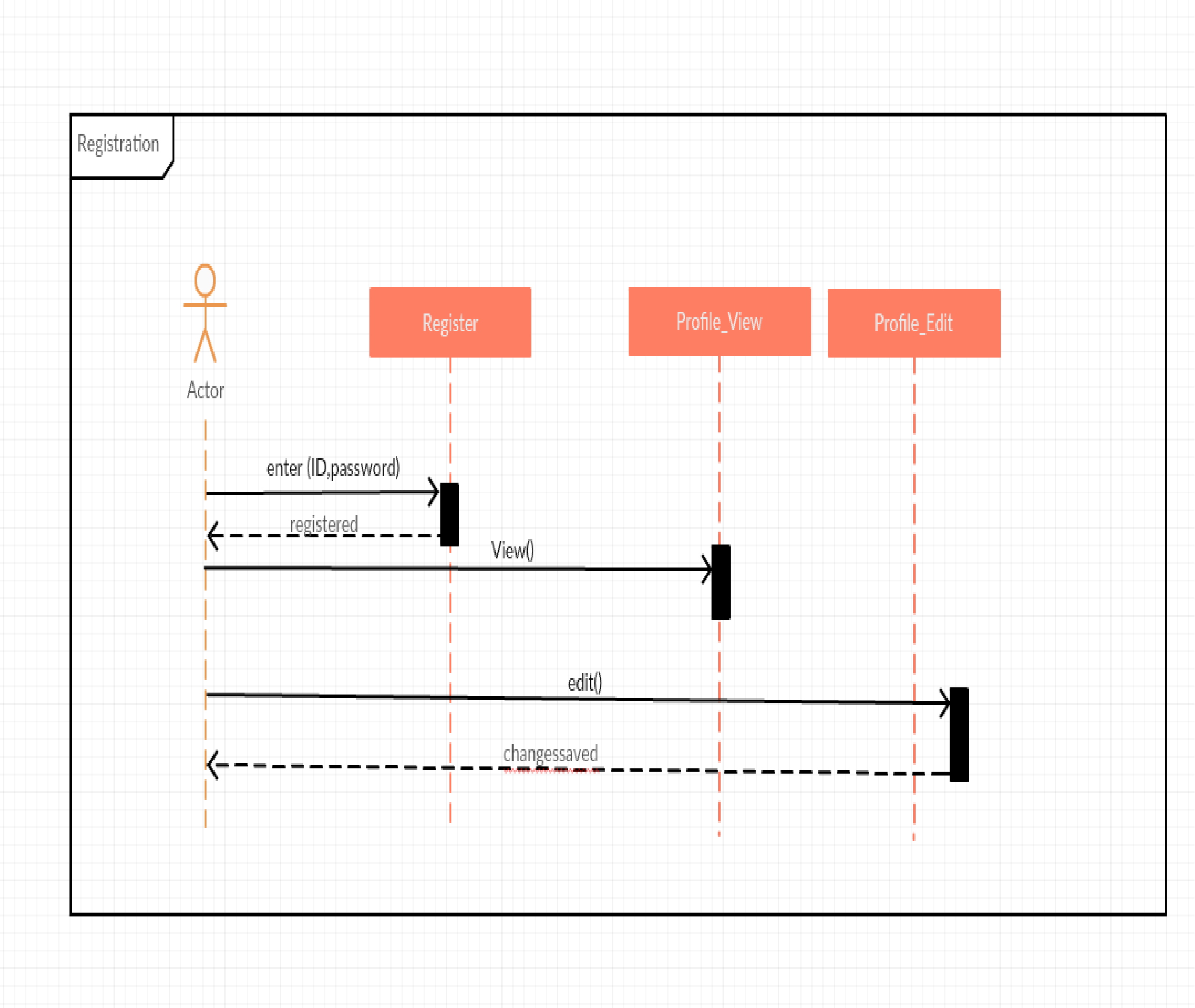
Priority 1> first =low 2> second =medium 3> third =high

4.3 DOMAIN MODEL DIAGRAM FOR SYSTEM: -

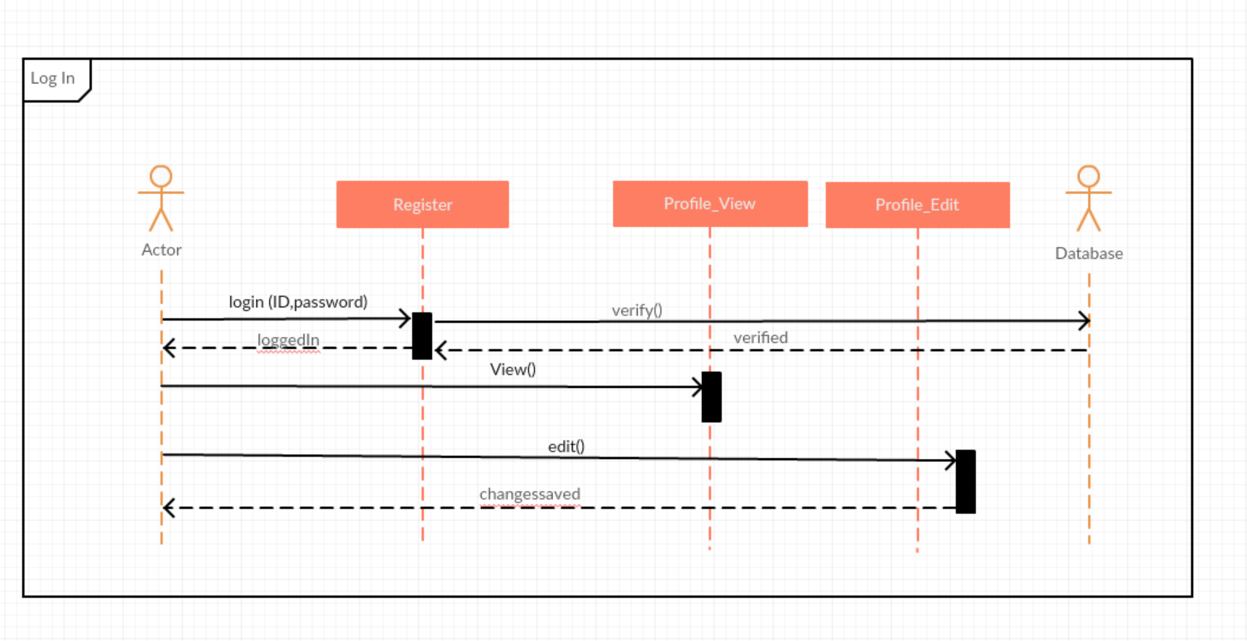

Market
Data
MarketAnalysis
+analyse
+has
1..*
1
1
1
*
1
0..*
+contains
*
1..*
FundamentalAnalysis
User
Company
TechnicalAnalysis
1..*
1..*
+has
0..*
1..*
Stock
1..*
1..*
1..*
MACD
Indicators
1
BarChart
OBV
0..*
Chart
PieChart
LineChart

4.4 SEQUENCE DIAGRAM

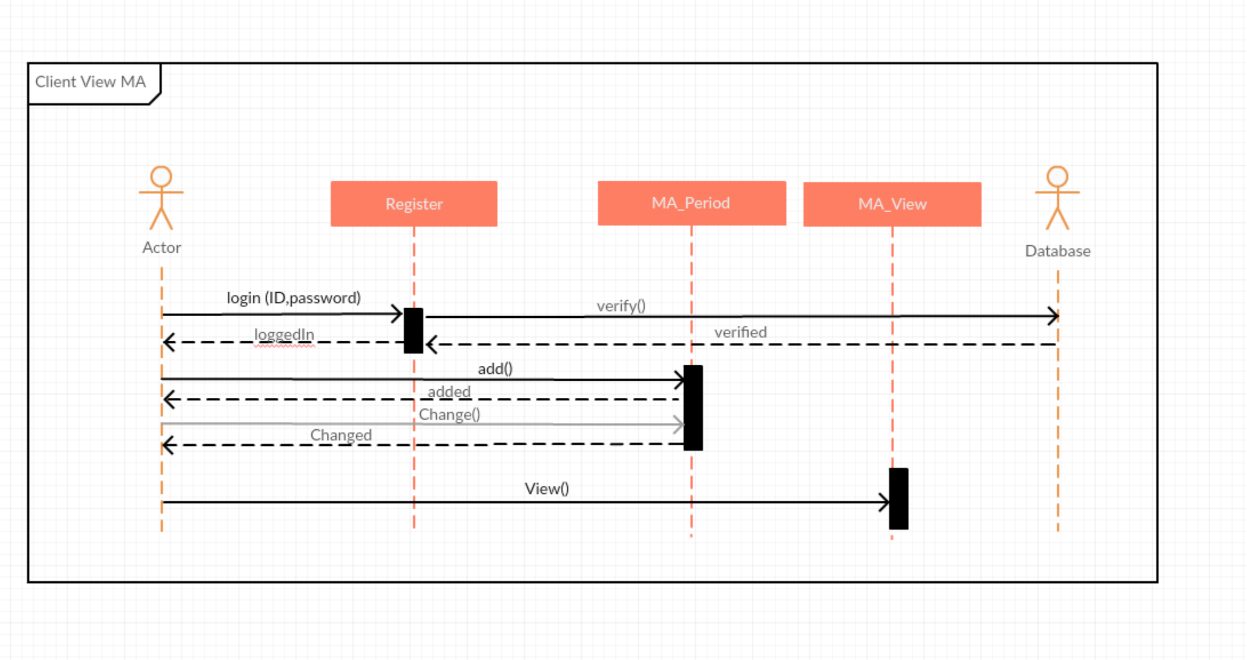
4.4.1 Registeration



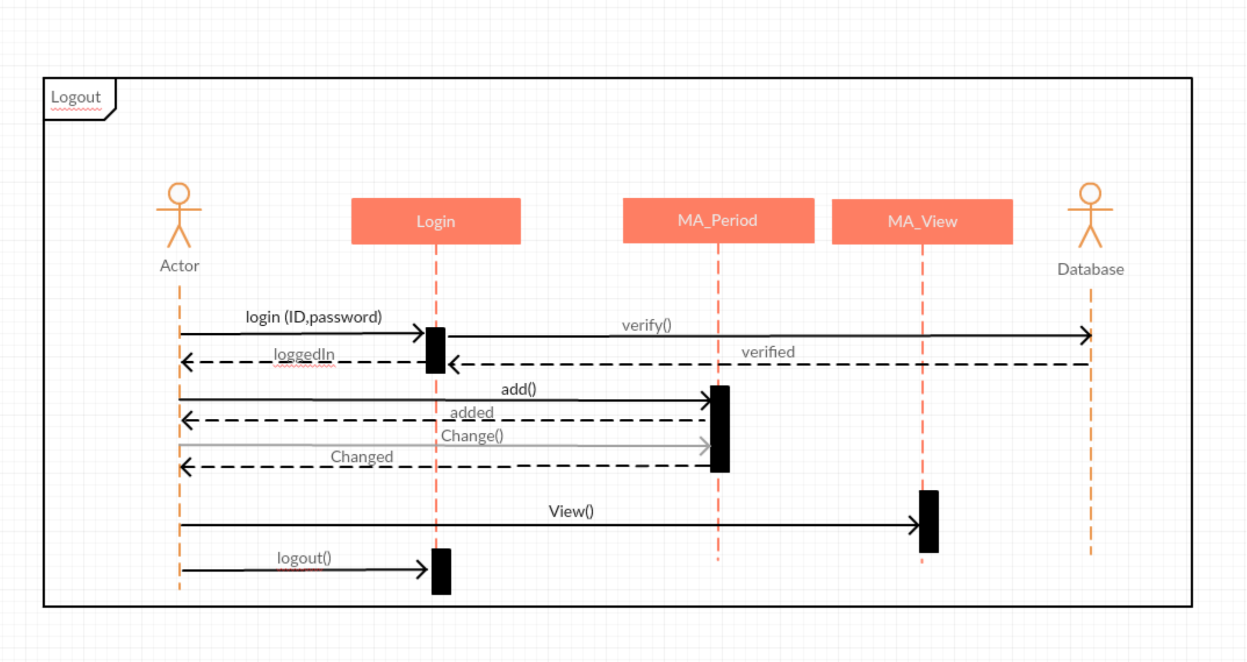
4.4.2 Login



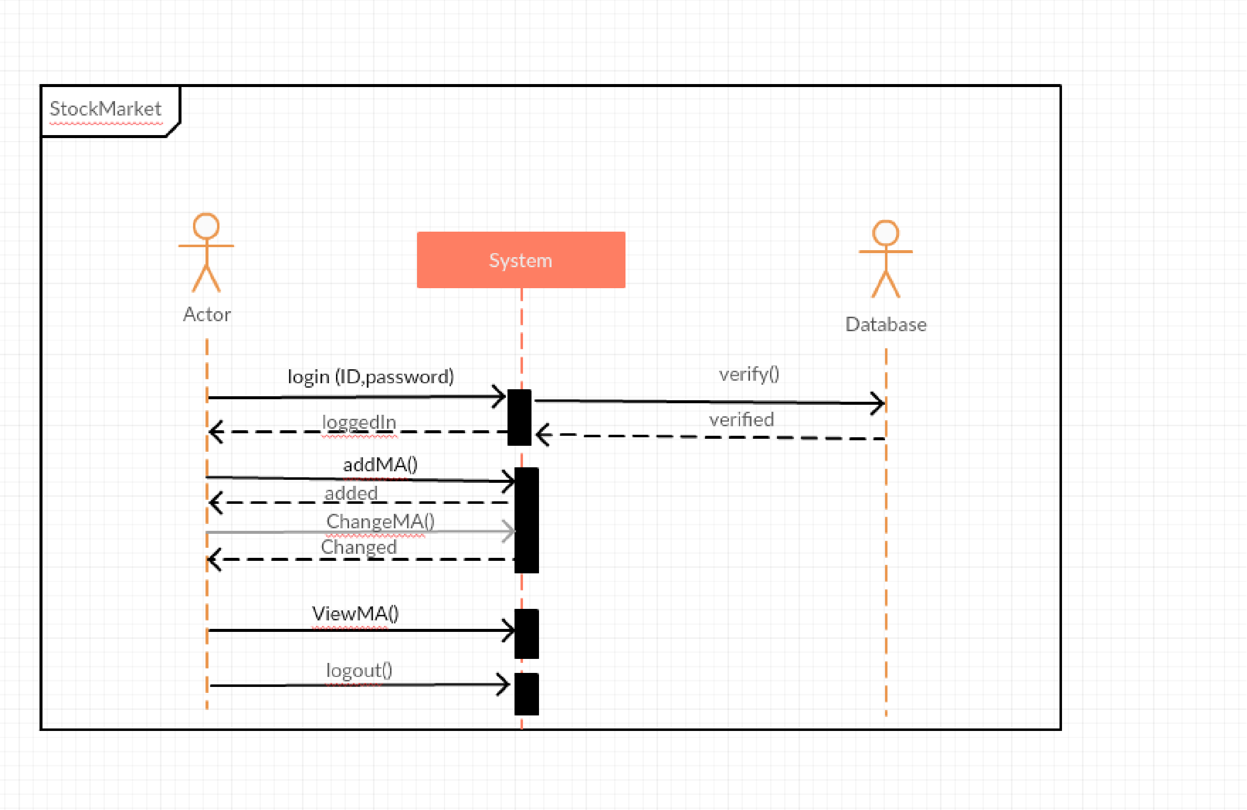
4.4.3 ClientViewMA



4.4.4 Logout



4.4.5 System Sequence Diagram:



5 DESIGN

5.1 ARCHITECTURAL DESIGN

5.1.1 Description

The system is divided into three modules model, view and controller. The modules interact together using their respective interfaces, which have been described in detail in section 3.2. Now, we turn our attention to each module, separately. In this section, we will describe the detailed design of the three modules. First, a textual description of the classes will be given. This will be followed by a class diagram, which will describe the relationships between the classes. Subsequently, a detailed description of each class, its methods and attributes will be given.

Before proceeding with the detailed internal module design, it is important to identify that we are going to have

1. View – fxml file and jfx library
2. Model – a class to store structure data in pairs date and also to value and calculate the moving averages, class to parse csv file it also
3. Controller – The system consists of three controller classes

5.1.2 General Constraints

Hardware and software environments: The system designed should support the java virtual machine and it is a standalone application independent of any operating systems

External data representations: The system should support to read the csv file

Network requirements: There should be a secured internet connection established to run the application.

5.1.3 Program Structure

The system is designed based on three components model view and controller the view the most important module in the stock market analysis design is the model module. This module is used to represent the data and line charts, this includes moving average indicators, stock and technical analysis that must occur in the chart. The model consists of several classes along with two interfaces, which are all contained within a container known as model. Each class contains the attributes and methods necessary to meet its requirements. The following sections will describe the classes, their relationships and will provide a detailed description of each method listed below.

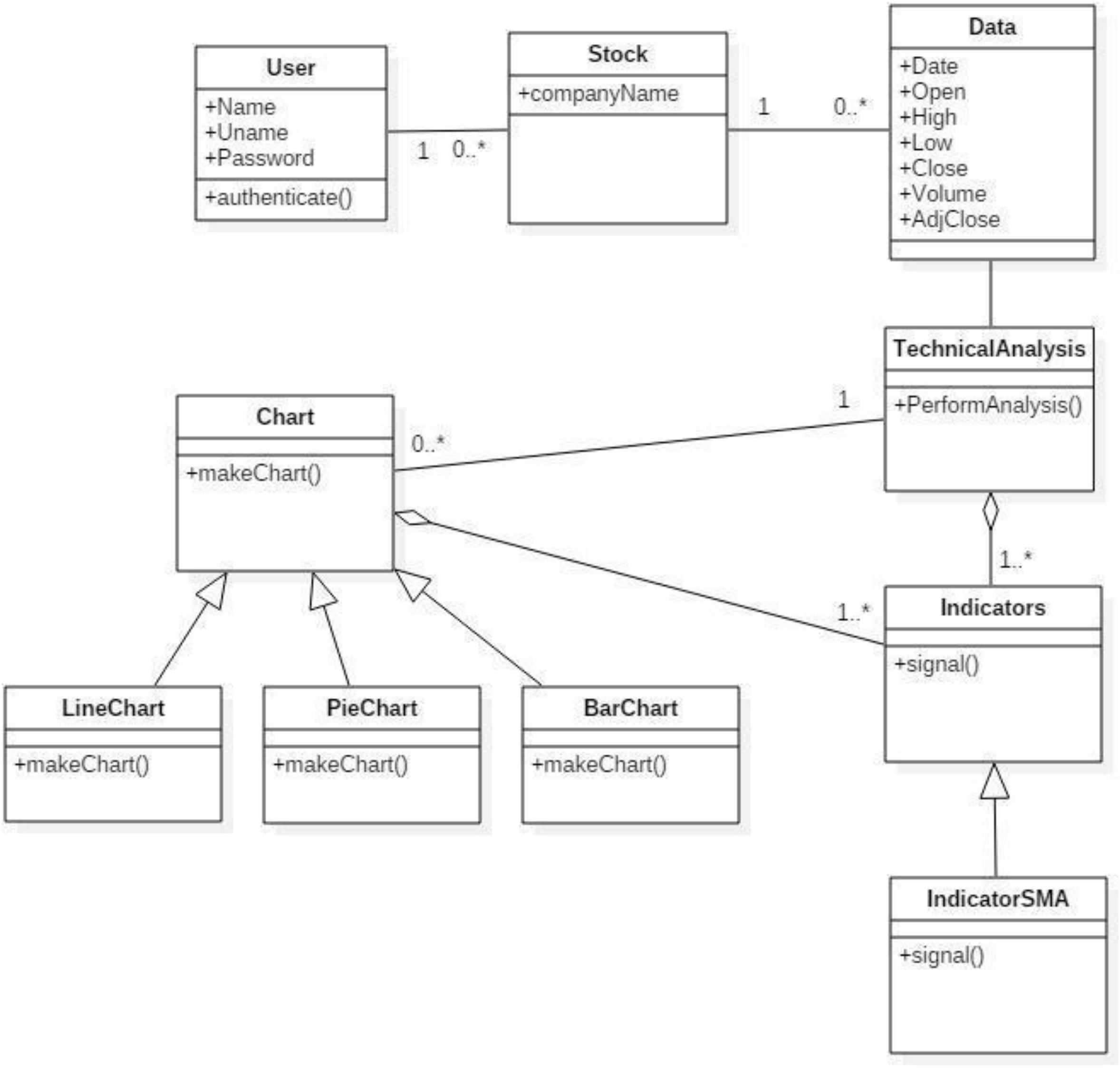
5.1.4 Alternatives Considered

The alternate architectural models that could possibly be included would be the call and return architecture and layered systems due to the interaction between the components. But still we choose the model view controller where each module consists of defined classes which enables high cohesion and low coupling among the classes.

Architectural views:

The logical view, use case view and development view are applicable for this design. The logical view is applicable from end users point of view to access the user interface of the system . The development view is applicable from developers point of view to develop the system based on java. The use case view is applicable with respect to the usecases in the design.

5.2 DESIGN OVERVIEW – CLASS MODEL



The class diagram depicts the classes of the system and their inter relationships. The class model consists of several classes ie., stock, user, data, technical analysis, chart, indicators. Each class contains the attributes and methods necessary to meet its requirements. The following sections will describe the classes, their relationships and will provide a detailed description of each method listed below.

5.2.1 General constraints:

Hardware and software environments: The system designed should support the java virtual machine and it is a standalone application independent of any operating systems

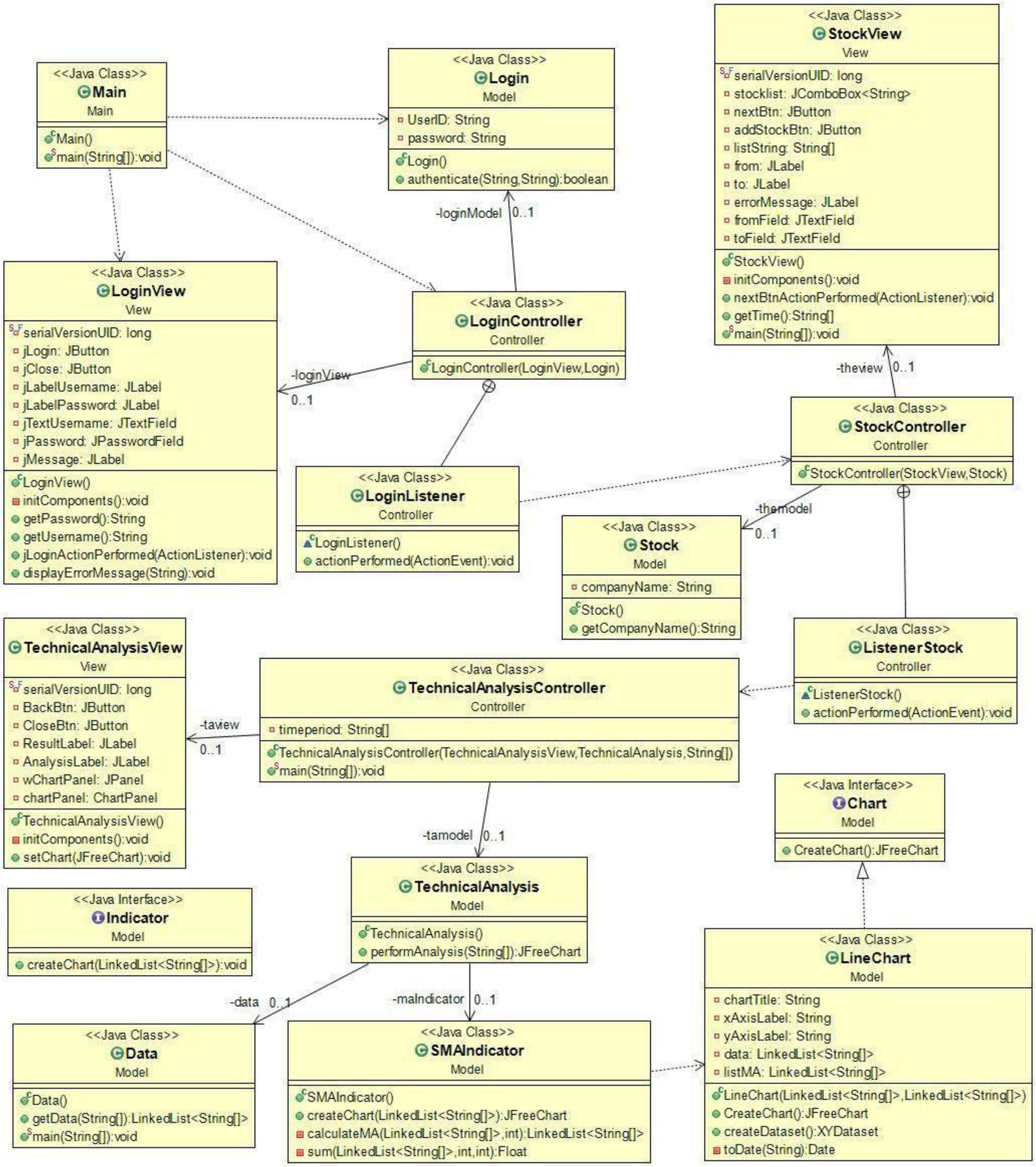
External data representations: The system should support to read the csv file

Network requirements: There should be a secured internet connection established to run the application.

5.2.2 ProgramStructure

The major components in the system are the model view and controller. The view module consists of three classes login, stock and technical analysis. The model module consists of six classes data, line chart, login, SMA indicator, stock, technical analysis and two interfaces chart and indicator. The interfaces in this module acts as adapter when new indicators are added to the system. The controller module consists of three classes login, stock and technical analysis.

6 DETAILED DESIGN



Method name	Authenticate()

Description	To validate the user credentials
Input	Strings
Output	N/A
Return type	Boolean

Method name	Setchart()
Description	To display chart in technical analysis view
Input	JFree chart
Output	Chart
Return type	Void

Method name	Perform analysis()
Description	It calls SMA Indicator and pass string to perform analysis.
Input	String
Output	N/A
Return type	JFree chart

Method name	get Data()
Description	To read data from csv file
Input	String
Output	N/A
Return type	Linked list

Method name	Create chart()
Description	It calls line chart for creating chart.
Input	Linked list
Output	N/A
Return type	JFree chart

Method name	Calculate MA()
Description	To calculate the moving averages

Input	Linked list and integer type
Output	N/A
Return type	Linked list
Method name	Calculatesum()
Description	It is a recursive function to add stock prices.
Input	Linked list and two integers
Output	N/A
Return type	Float

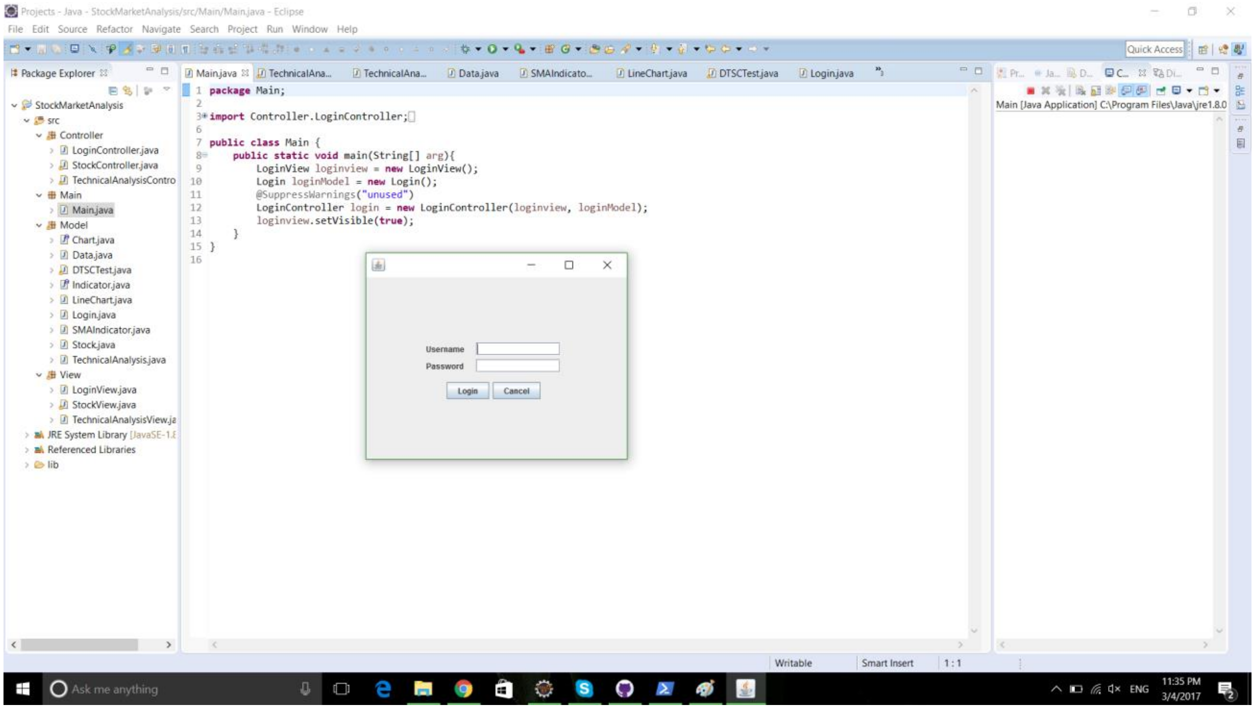
Method name	CreateChart()
Description	To create a chart
Input	N/A
Output	N/A
Return type	JFree Chart
Method name	CreateDataset()
Description	To create a data set for line chart
Input	N/A
Output	N/A
Return type	XY dataset

7 USER INTERFACE DESIGN

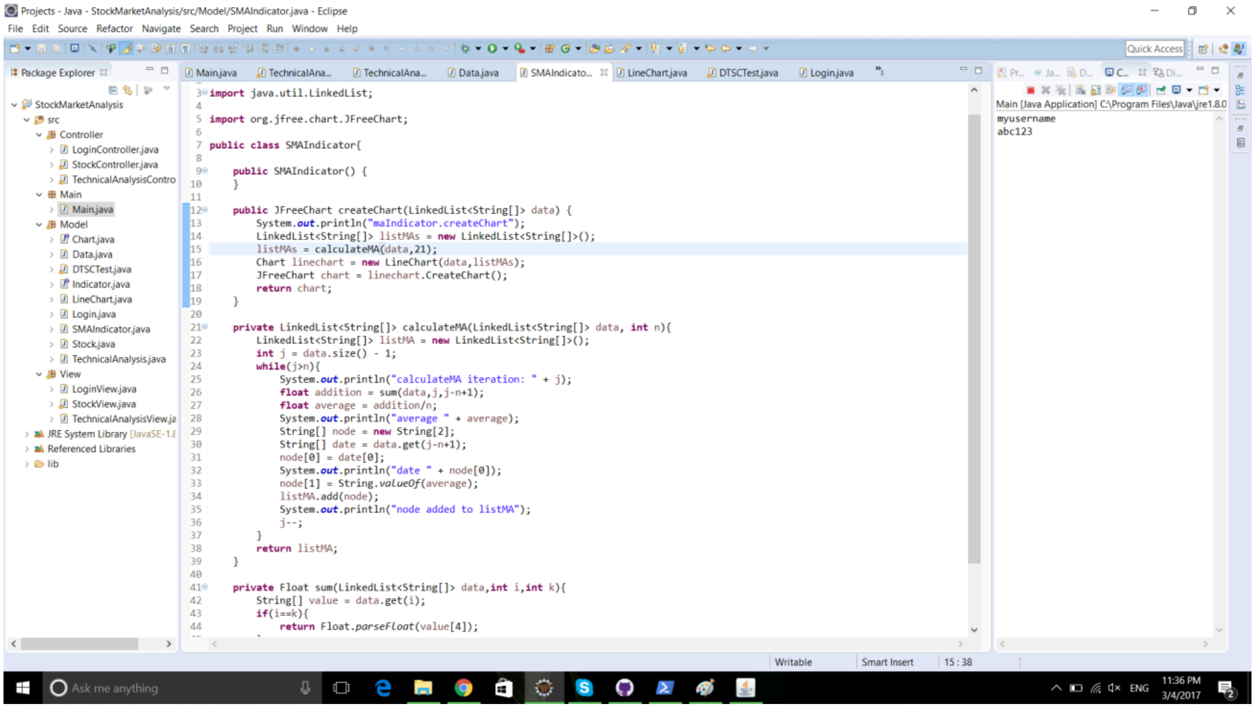
7.1 GUI COMPONENTS

For GUI components, we will use Java Swing Framework. For some components, AWT library can be used like action listeners. For showing charts, JFreeChart library will be used.

7.2 DETAILED DESCRIPTION

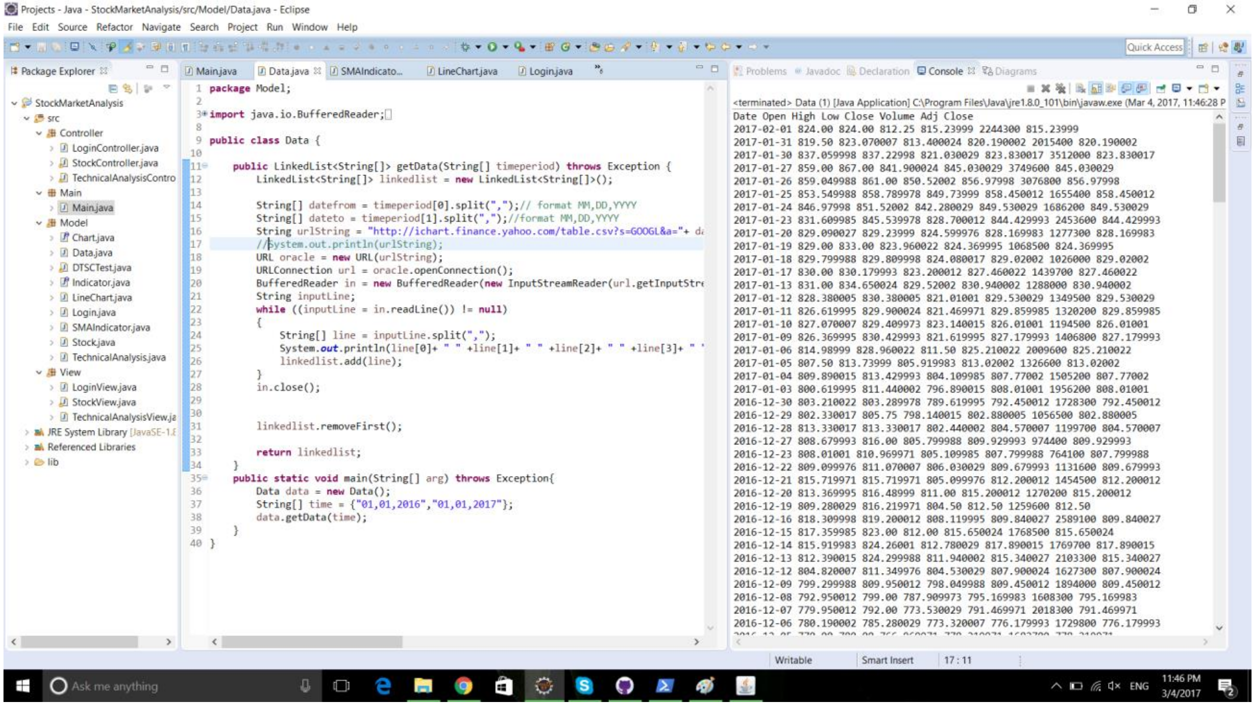


```java
package Main;

import Controller.LoginController;

public class Main {
    public static void main(String[] arg){
        LoginView loginview = new LoginView();
        Login loginModel = new Login();
        @SuppressWarnings("unused")
        LoginController login = new LoginController(loginview, loginModel);
        loginview.setVisible(true);
    }
}
```



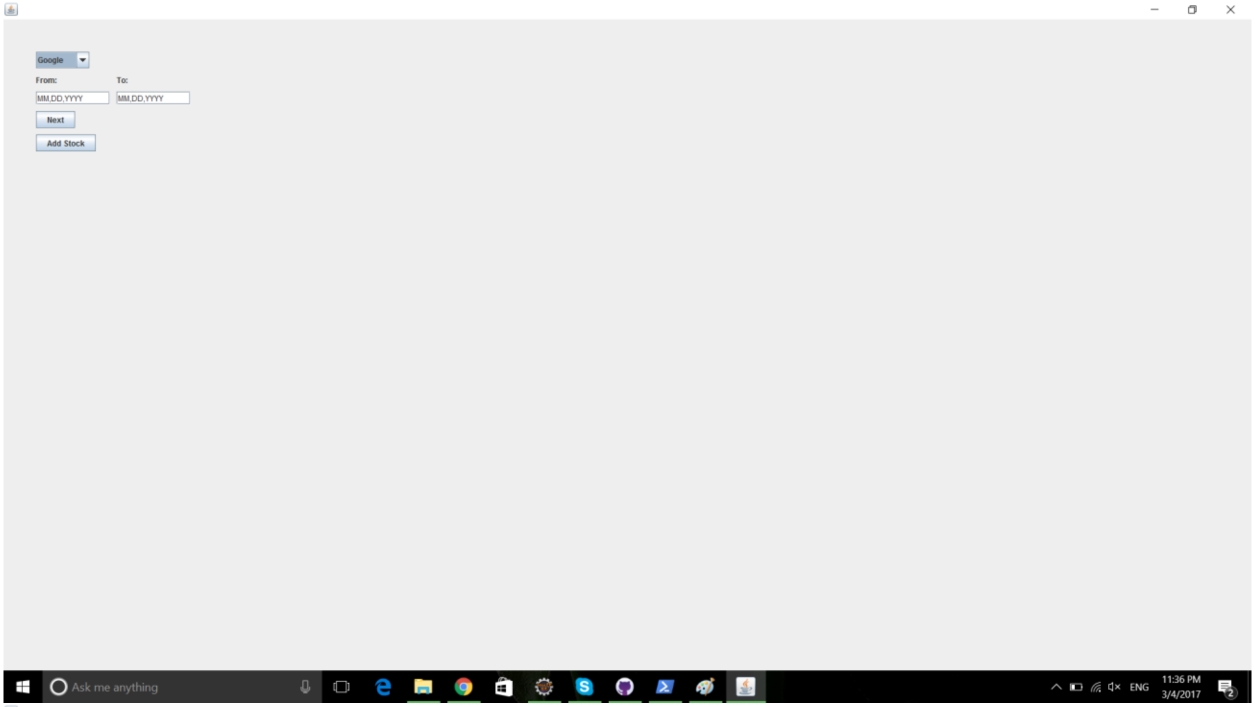
```java
import java.util.LinkedList;

import org.jfree.chart.JFreeChart;

public class SMAIndicator{

    public SMAIndicator() {
    }

    public JFreeChart createChart(LinkedList<String[]> data) {
        System.out.println("maIndicator.createChart");
        LinkedList<String[]> listMAs = new LinkedList<String[]>();
        listMAs = calculateMA(data,21);
        Chart linechart = new LineChart(data,listMAs);
        JFreeChart chart = linechart.CreateChart();
        return chart;
    }

    private LinkedList<String[]> calculateMA(LinkedList<String[]> data, int n){
        LinkedList<String[]> listMA = new LinkedList<String[]>();
        int j = data.size() - 1;
        while(j>n){
            System.out.println("calculateMA iteration: " + j);
            float addition = sum(data,j,j-n+1);
            float average = addition/n;
            System.out.println("average " + average);
            String[] node = new String[2];
            String[] date = data.get(j-n+1);
            node[0] = date[0];
            System.out.println("date " + node[0]);
            node[1] = String.valueOf(average);
            listMA.add(node);
            System.out.println("node added to listMA");
            j--;
        }
        return listMA;
    }

    private Float sum(LinkedList<String[]> data,int i,int k){
        String[] value = data.get(i);
        if(i==k){
            return Float.parseFloat(value[4]);
```

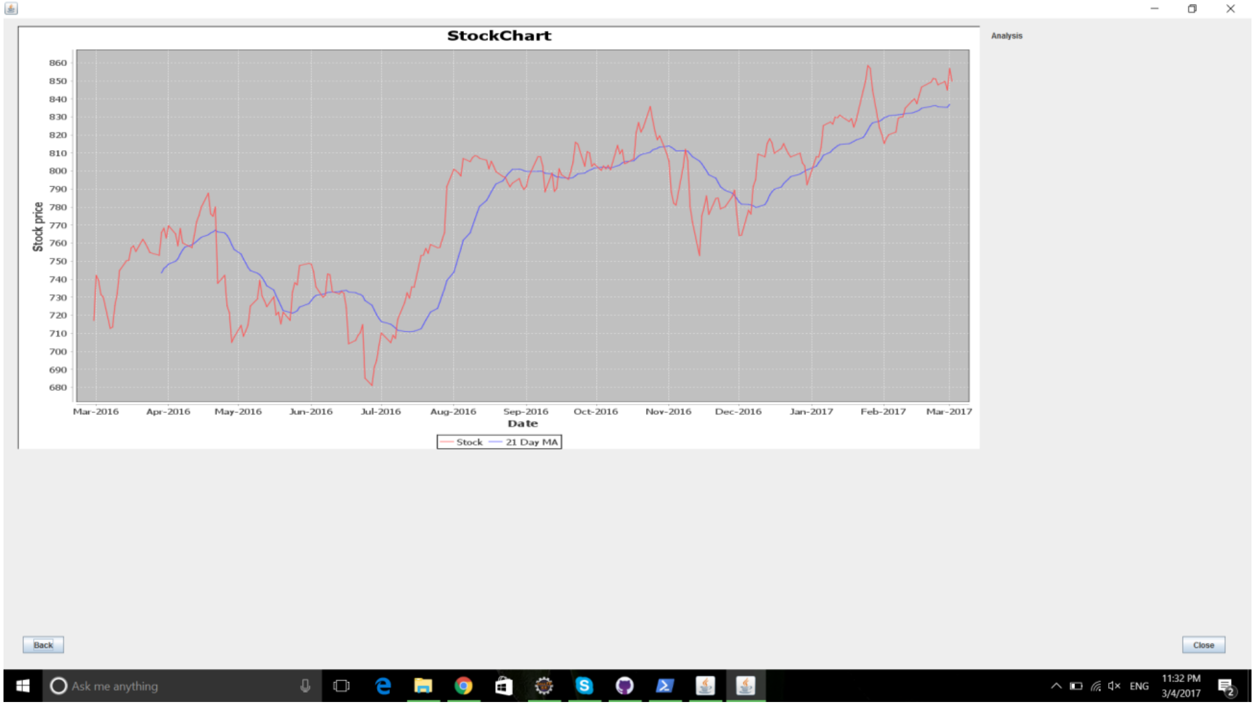
```
Main [Java Application] C:\Program Files\Java\jre1.8.0
myusername
abc123
```



```java
package Model;

import java.io.BufferedReader;

public class Data {

    public LinkedList<String[]> getData(String[] timeperiod) throws Exception {
        LinkedList<String[]> linkedlist = new LinkedList<String[]>();

        String[] datefrom = timeperiod[0].split(",");// format MM,DD,YYYY
        String[] dateto = timeperiod[1].split(",");//format MM,DD,YYYY
        String urlString = "http://ichart.finance.yahoo.com/table.csv?s=GOOGL&a="+ da
        //System.out.println(urlString);
        URL oracle = new URL(urlString);
        URLConnection url = oracle.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(url.getInputStre
        String inputLine;
        while ((inputLine = in.readLine()) != null)
        {
            String[] line = inputLine.split(",");
            System.out.println(line[0]+ " " +line[1]+ " " +line[2]+ " " +line[3]+ " '
            linkedlist.add(line);
        }
        in.close();


        linkedlist.removeFirst();

        return linkedlist;
    }
    public static void main(String[] arg) throws Exception{
        Data data = new Data();
        String[] time = {"01,01,2016","01,01,2017"};
        data.getData(time);
    }
}
```

```
<terminated> Data (1) [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (Mar 4, 2017, 11:46:28 P
Date Open High Low Close Volume Adj Close
2017-02-01 824.00 824.00 812.25 815.23999 2244300 815.23999
2017-01-31 819.50 823.070007 813.400024 820.190002 2015400 820.190002
2017-01-30 837.059998 837.22998 821.030029 823.830017 3512000 823.830017
2017-01-27 859.00 867.00 841.900024 845.030029 3749600 845.030029
2017-01-26 859.049988 861.00 850.52002 856.97998 3076800 856.97998
2017-01-25 853.549988 858.789978 849.73999 858.450012 1655400 858.450012
2017-01-24 846.97998 851.52002 842.280029 849.530029 1686200 849.530029
2017-01-23 831.609985 845.539978 828.700012 844.429993 2453600 844.429993
2017-01-20 829.090027 829.23999 824.599976 828.169983 1277300 828.169983
2017-01-19 829.00 833.00 823.960022 824.369995 1068500 824.369995
2017-01-18 829.799988 829.809998 824.080017 829.02002 1026000 829.02002
2017-01-17 830.00 830.179993 823.200012 827.460022 1439700 827.460022
2017-01-13 831.00 834.650024 829.52002 830.940002 1288000 830.940002
2017-01-12 828.380005 830.380005 821.01001 829.530029 1349500 829.530029
2017-01-11 826.619995 829.900024 821.469971 829.859985 1320200 829.859985
2017-01-10 827.070007 829.409973 823.140015 826.01001 1194500 826.01001
2017-01-09 826.369995 830.429993 821.619995 827.179993 1406800 827.179993
2017-01-06 814.98999 828.960022 811.50 825.210022 2009600 825.210022
2017-01-05 807.50 813.73999 805.919983 813.02002 1326600 813.02002
2017-01-04 809.890015 813.429993 804.109985 807.77002 1505200 807.77002
2017-01-03 800.619995 811.440002 796.890015 808.01001 1956200 808.01001
2016-12-30 803.210022 803.289978 789.619995 792.450012 1728300 792.450012
2016-12-29 802.330017 805.75 798.140015 802.880005 1056500 802.880005
2016-12-28 813.330017 813.330017 802.440002 804.570007 1199700 804.570007
2016-12-27 808.679993 816.00 805.799988 809.929993 974400 809.929993
2016-12-23 808.01001 810.969971 805.109985 807.799988 764100 807.799988
2016-12-22 809.099976 811.070007 806.030029 809.679993 1131600 809.679993
2016-12-21 815.719971 815.719971 805.099976 812.200012 1454500 812.200012
2016-12-20 813.369995 816.48999 811.00 815.200012 1270200 815.200012
2016-12-19 809.280029 816.219971 804.50 812.50 1259600 812.50
2016-12-16 818.309998 819.200012 808.119995 809.840027 2589100 809.840027
2016-12-15 817.359985 823.00 812.00 815.650024 1768500 815.650024
2016-12-14 815.919983 824.26001 812.780029 817.890015 1769700 817.890015
2016-12-13 812.390015 824.299988 811.940002 815.340027 2103300 815.340027
2016-12-12 804.820007 811.349976 804.530029 807.900024 1627300 807.900024
2016-12-09 799.299988 809.950012 798.049988 809.450012 1894000 809.450012
2016-12-08 792.950012 799.00 787.909973 795.169983 1608300 795.169983
2016-12-07 779.950012 792.00 773.530029 791.469971 2018300 791.469971
2016-12-06 780.190002 785.280029 773.320007 776.179993 1729800 776.179993
```


Google
From:
To:
MM,DD,YYYY
MM,DD,YYYY
Next
Add Stock
Ask me anything
ENG
11:36 PM
3/4/2017


StockChart
Analysis
Stock price
860
850
840
830
820
810
800
790
780
770
760
750
740
730
720
710
700
690
680
Mar-2016
Apr-2016
May-2016
Jun-2016
Jul-2016
Aug-2016
Sep-2016
Oct-2016
Nov-2016
Dec-2016
Jan-2017
Feb-2017
Mar-2017
Date
Stock
21 Day MA
Back
Close
Ask me anything
ENG
11:32 PM
3/4/2017

8 CONCLUSION

This design of the stock market analysis , will provide details on the architectural design, software interface design, and the internal module design. The architectural design will describe the software architecture that was chosen for the system and a class diagram of this architecture. The software interface design have screen shots of the graphical user interface and how the users interact with the system. Finally, the internal module design that describe different modules. This design is implemented on java virtual machine and model view controller model pattern is applicable to the system .

9 TEAM CONTRIBUTION

Team Member	Contributions
Gurpreet	3.2 UC3, 3.3, 3.4 4, 4, 5, 6,
Nikita	3.2 UC3, 3.3, 3.4 4, 4, 5
Kirtan	2, 3.1,3.2 UC2, 4, 5
Ajeeta	1, 3.1, 3.2 UC4, 3.4 1, 4, 5, 6, 7
Jyotsana	3.2 UC2, 3.4 5, 4
Navdeep	3.3
Rajan	3.2 UC1,3.4 2, 5