VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI-590018

A MINI PROJECT REPORT ON

STOCK MARKET SYSTEM

Submitted in partial fulfilment of the requirements

For the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

SHRAVAN BHAT

[1KS16CS092]

Under the guidance of

Mr. K Venkata Rao
Assoc. Prof, Dept. Of CSE

Mr. Kumar K
Asst. Prof, Dept. Of CSE

Mr. Roopesh Kumar B N
Asst. Prof, Dept. Of CSE

Department of Computer Science & Engineering

K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

2018-19

K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that mini project work entitled **"STOCK MARKET SYSTEM"** carried out by **Mr. SHRAVAN BHAT** bearing USN **1KS16CS092** bonafide student of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the year 2018. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini Project work prescribed for the said degree for the 5th semester.

Dr . Rekha. B. Venkatapur
Prof & HOD, CS & E Department

Dr . T.V. Govindaraju
Principal/Director, KSIT

Mr. K Venkata Rao
Assoc. Prof, Dept. Of CSE

Mr. Kumar K
Asst. Prof, Dept. Of CSE

Mr. Roopesh Kumar B N
Asst. Prof, Dept. Of CSE

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

I take this opportunity to thank one and all involved in building this project. Firstly I would like to thank the college for providing us an opportunity to work on the project.

I would also like to thank the management of **K. S. Institute of Technology** for providing all the resources required for the project.

I wish to acknowledge my sincere gratitude to our beloved Principal, **Dr. T.V. Govindaraju** for his encouragement and providing all facilities for the accomplishment of this project.

This project would not have been possible without the support of our beloved **Prof & HOD, Dr. Rekha.B.Venkatapur, Dept. of CSE.**

I am also highly grateful to my project guides, **Mr. K Venkata Rao, Mr. Kumar K**, and **Mr. Roopesh Kumar B N, Dept. of CSE** who have been very generous in assisting and supporting, to do this Project **"Stock Market System".**

I also would like to thank all other teaching and non-teaching staff members who have extended their support and co-operation while bringing up this project.

SHRAVAN BHAT

(1KS16CS092)

ABSTRACT

Stock Market System is a system which enables individuals and organizations to trade shares and company stocks in the stock markets. It enables everyone to invest their money and manage and maintain their individual portfolios. It is a user friendly system which makes it easy to trade, find and invest in new companies which may interest them. We can also see the historical prices through the system including the companies open, high, close etc. on the Nifty 50 stock market which I have used as a base for this project. The system also allows us to trade and sell different stocks.

CONTENTS

Chapter 1

INTRODUCTION

1.1 OVERVIEW

"Stock Market System" is designed to help invest and trade in stocks, shares and currencies on the international stock market system. It enables us to view all the historical data on the companies present in the stock market and make smart investments decisions.

It also enables us to view and trade on the current investments made by customer and thus help them decide whether to trade and sell those stocks or to buy any more stocks. It also allows them to view their total profits and loss made on the stock markets.

1.2 PROBLEM STATEMENT

The main aim of "Stock Market System" is to make and easy interface for bankers, investors and businesses to make systematic decisions on the stock market and help them invest in the stocks of their choice.

1.3 DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified ,and the database schema, which defines the

database's logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

1.4 SQL

SQL is a standard language for storing, manipulating and retrieving data in databases.

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.[13]Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

1.5 HTML / JavaScript

HTML is a markup language used for structuring and presenting content on the web and the fifth and current major version of the HTML standard.
HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

1.6 JAVA CONNECTIONS

To connect the database with the front end we use a java connector JDBC (Java Database Connectivity). JDBC is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

To achieve connectivity we use JSPs (Java Server Pages) in this project. Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP is similar to PHP and ASP, but it uses the Java programming language.

Chapter 2

REQUIREMENTS SPECIFICATION

A computerized way of handling information about property and users details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a database driven web application whose requirements are mentioned in this section.

2.1 OVERALL DESCRIPTION

A reliable and scalable database driven web application with security features that is easy to use and maintain is the requisite.

2.2 SPECIFIC REQUIREMENTS

The specific requirements of the Stock Market System are stated as follows:

2.2.1 SOFTWARE REQUIREMENTS

- IDE - NetBeans 8.2
- Web Browser – Firefox 50 or later, Google Chrome – 60 or later
- Database support - MySQL 5.7
  - MySQL Server 5.7
  - MySQL Shell 1.0.10
  - MySQL Workbench
- Operating system – Windows 7 / Ubuntu 16.04
- JDK 1.8
- Server deployment - Tomcat server / Glassfish Server

2.2.2 HARDWARE REQUIREMENTS

- Processor – Pentium IV or above
- RAM – 2 GB or more
- Hard disk – 3 GB or more
- Monitor – VGA of 1024x768 screen resolution
- Keyboard and Mouse

2.2.3 TECHNOLOGY

- HTML is used for the front end design. It provides a means to structure text based information in a document. It allows users to produce web pages that include text, graphics and hyperlinks.
- CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document.
- SQL is the language used to manipulate relational databases. It is tied closely with the relational model. It is issued for the purpose of data definition and data manipulation.
- Java Server pages is a simple yet powerful technology for creating and maintaining dynamic-content web pages. It is based on the Java programming language. It can be thought of as an extension to servlet because it provides more functionality than servlet A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development.
- We require a JDBC connection between the front end and back end components to write to the database and fetch required data.

Chapter 3

3. DETAILED DESIGN

3.1 SYSTEM DESIGN

The web server needs a JSP engine, i.e., a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs. This server will act as a mediator between the client browser and a database.

The following diagram shows the JSP architecture.

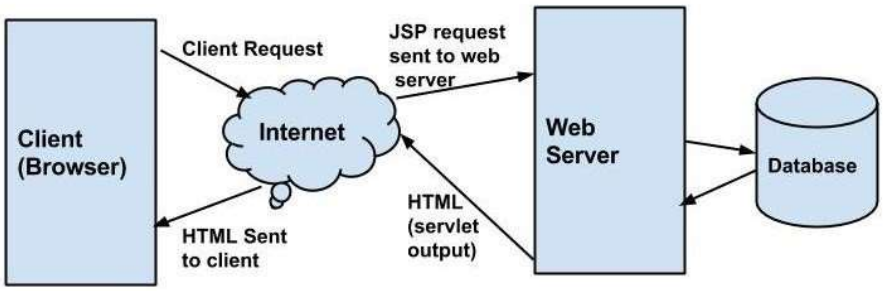


Fig. 3.1 JSP Architecture

Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client. Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users. Several types of databases, including relational or multimedia, may be created. Additionally, database architects may use one of several languages to create databases, such as structured query language.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business.

An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ER Diagram is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.

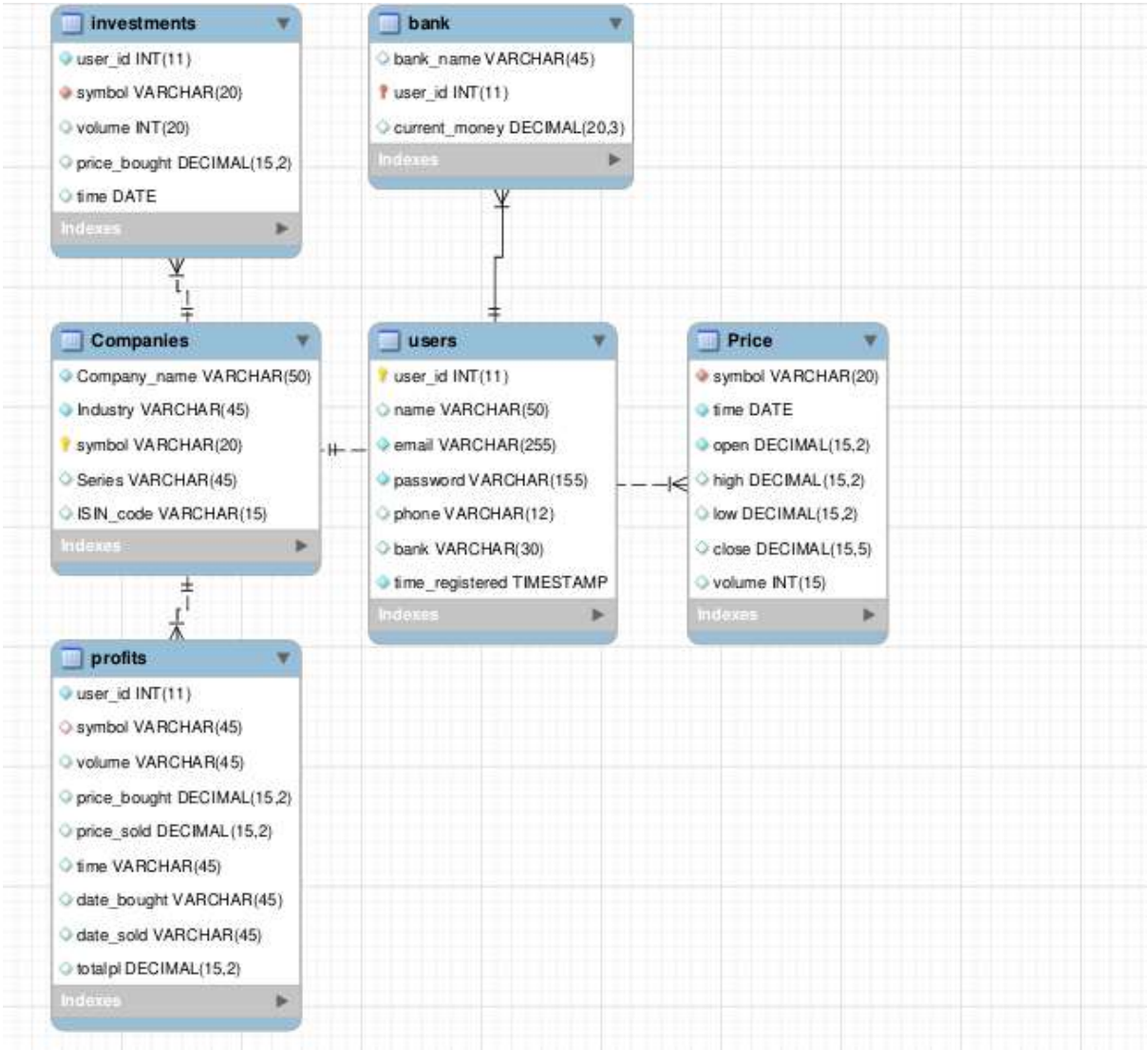


Fig. 3.2.1 Enhanced ER diagram of Stock Market System

- - - 1: n Non Identifying Relationship

1: n Identifying Relationship

1:1 Identifying Relationship

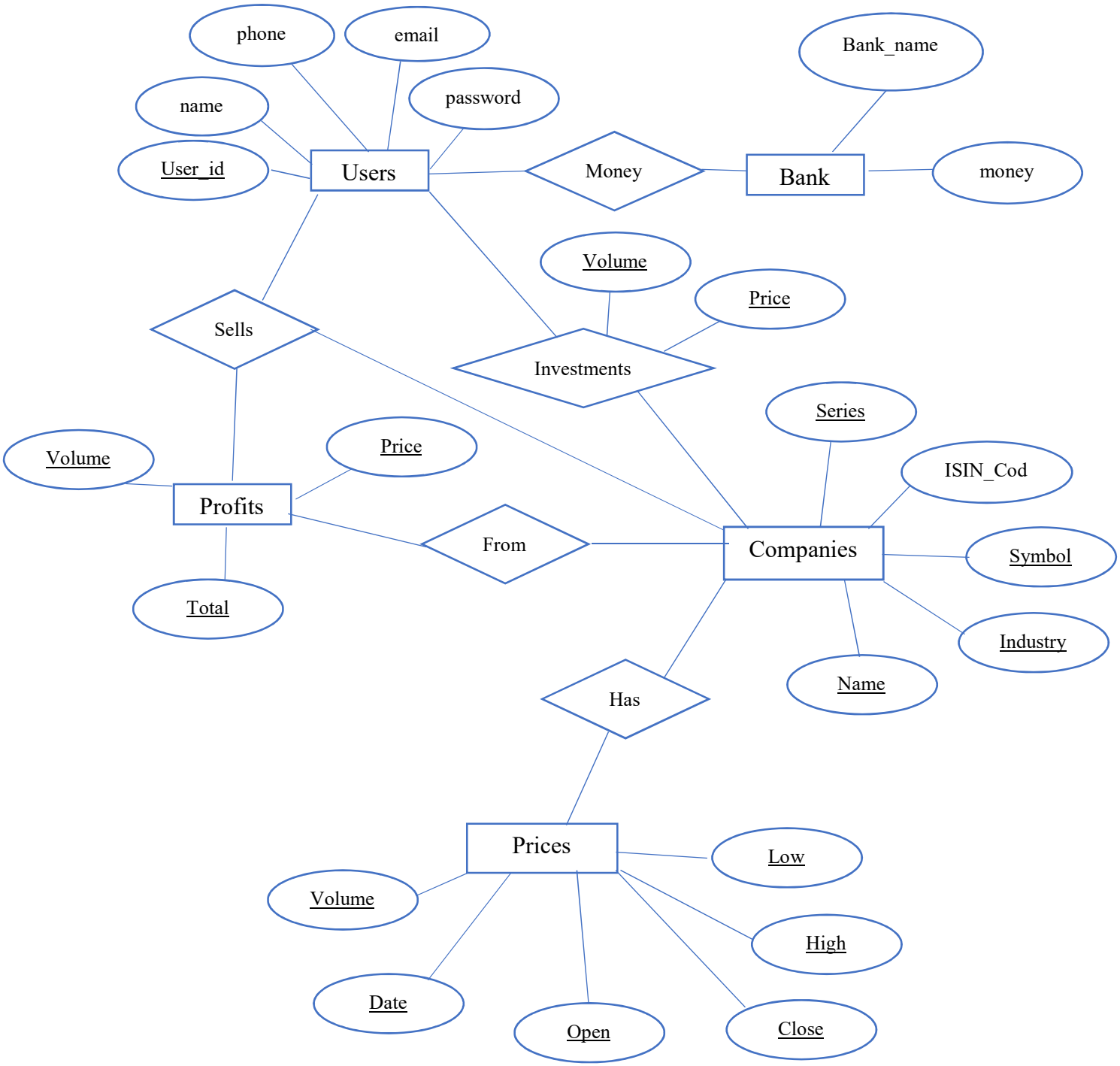


Fig. 3.2.2, ER diagram of Stock Market System

3.3 RELATIONAL SCHEMA

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

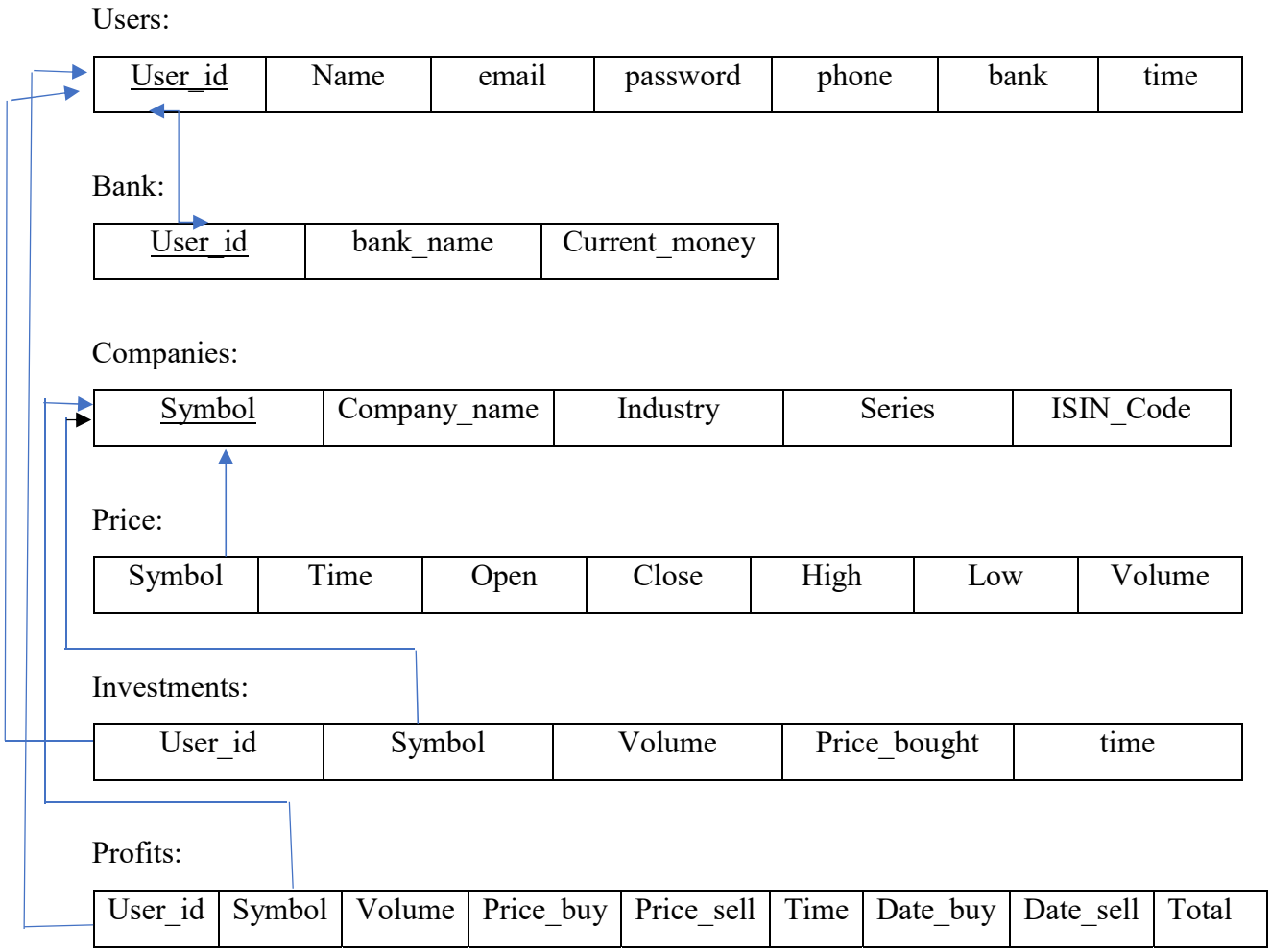


Fig. 3.3, Schema diagram

3.4 DESCRIPTION OF TABLES

The database consists of six tables:

1. Users: It stores the user details.
   - User_id: Unique user id done by auto increment.
   - Name: Name of the user.
   - Phone: Phone number of the user.
   - Email: Email id of the user.
   - Password: Password associated with user to login into system
   - Bank: Bank name associated with user

2. Bank: It stores the financial details of the users.
   - User_id: Foreign key of user associated with the bank
   - Bank Name Bank of the user.
   - Current_money: Current money held by user in the stock market platform.

3. Companies: Stores the companies registered with the stock market
   - Company_name: Name of the company.
   - Industry: Name of the seed.
   - Symbol: Unique symbol of the company used to trade shares.
   - Series: Whether it is equity or mutual fund etc.
   - ISIN_Code: International code of the company.

4. Price: It stores the prices of the company across various times.
   - Symbol: Unique Symbol of the company
   - Time: Date of the info
   - Open: The opening price on particular day.
   - Close: The closing price on particular day.
   - High: The highest price on particular day.
   - Low: The lowest price on particular day.
   - Volume: The total volume traded on the day

5. Investments: It stores the current investments held by the customer
   - User_id: Unique identification of user who bought the shares.
   - Symbol: Company of which shares were bought
   - Volume: Total volume of shares bought.
   - Price_bought: Price at which the stock was bought
   - Time: Time of the investment

6. Profits: It stores the total profit/loss data after stock sale.
    - User_id :Unique identification number given to the user.
    - Symbol: Company of which shares were bought
    - Volume: Total volume of shares bought.
    - Price_bought: Price at which the stock was bought
    - Price_sold: Price at which the stock was sold
    - Time: Time of the selling
    - Date_bought: Date at which stock was bought
    - Date_sold: Date at which stock was sold
    - Totalpl: Total profit/loss made from selling.

Chapter 4

IMPLEMENTATION

4.1 MODULES AND THEIR ROLES

4.1.1 Login: Login for the new user.

```
<%
        String userid = request.getParameter("email");
        String pwd = request.getParameter("pwd");
        String username;
        String query = "select * from users where email=?";
        PreparedStatement psm = con.prepareStatement(query);
        psm.setString(1,userid);
        ResultSet rs=psm.executeQuery();

        if (rs.next()) {
        String entpass = rs.getString("password");
        String cipher = entpass.substring(12);
        BASE64Decoder decoder = new BASE64Decoder();
        try {
        String decoded = new String(decoder.decodeBuffer(cipher));
        if (decoded.equals(pwd))
        {
           username = rs.getString(3);
           session.setAttribute("userid", rs.getString("user_id"));
           session.setAttribute("username", username);
           response.sendRedirect("stocks/index.jsp");
        }
        else {
           %>
           <p class="text-info text-center">Invalid Password, Go Back and try again!
              <br></p>                    <%                    }                    }%>
```

4.1.2 Companies List: List of all companies in nifty 50

```
<tbody class="table-big">
<%
    ResultSet rs;
    rs = st.executeQuery("select * from Companies");
    while(rs.next())
    {
        out.println("<tr>");
            out.println("<td><a
            href='http://www.google.com/search?q="+rs.getString("Company_name")+"'>
            <b>"+rs.getString("Company_name")+"</b></a></td>");
            out.println("<td>"+rs.getString("Industry")+"</td>");
            out.println("<td>"+rs.getString("symbol")+"</td>");
            out.println("<td>"+rs.getString("Series")+"</td>");
            out.println("<td>"+rs.getString("ISIN_code")+"</td>");
        out.println("</tr>");
    }
%>
</tbody>
```

4.1.3 Latest Price: List of latest price of companies in nifty 50

```
<tbody class="table-big">
<%
ResultSet rs;
rs = st.executeQuery("select P.* from Price P ORDER BY P.time DESC, P.symbol ASC LIMIT
48;");
    while(rs.next())
    {
    out.println("<tr>");
    out.println("<td><b>"+rs.getString("symbol")+"</b></td>");
    out.println("<td>"+rs.getString("open")+"</td>");
    out.println("<td>"+rs.getString("high")+"</td>");
    out.println("<td>"+rs.getString("low")+"</td>");
    out.println("<td>"+rs.getString("close")+"</td>");
```

```
    out.println("<td>"+rs.getString("volume")+"</td>");
    out.println("</tr>");
}
%>
</tbody>
```

4.1.4 Historical Price: Retrieve the price of stocks between 2 selected periods:

```
ResultSet rs;
        rs = st.executeQuery("SELECT * FROM Price WHERE time >= '"+from+"' AND time <=
'"+to+"' AND symbol ='"+stock+"' LIMIT 31");

        if(rs.next())
        {
                String ctime = rs.getString("time");
                String test[] = (ctime.split("-"));
                String test2 = "new Date("+test[0]+","+(Integer.parseInt(test[1])-1)+","+test[2]+" )";

                data+=rs.getString(type);
                data3+="{ x : "+test2+" ,y : "+rs.getString(type)+"}";
                data2+=rs.getString("time");
                out.println("<tr>");
                        out.println("<td>"+rs.getString("time")+"</td>");
                        out.println("<td>"+rs.getString("open")+"</td>");
                        out.println("<td>"+rs.getString("high")+"</td>");
                        out.println("<td>"+rs.getString("low")+"</td>");
                        out.println("<td>"+rs.getString("close")+"</td>");
                        out.println("<td>"+rs.getString("volume")+"</td>");
                out.println("</tr>");

                while(rs.next())
                {
                        data+=","+rs.getString(type);
```

```
                        ctime = rs.getString("time");
                        String test21[] = (ctime.split("-"));
                        String test22 = "new Date("+test21[0]+","+(Integer.parseInt(test21[1])-
1)+","+test21[2]+" )";


                        data3+=",{ x : "+test22+" ,y : "+rs.getString(type)+"}";

                        data2+=rs.getString("time");
                        out.println("<tr>");
                                out.println("<td>"+rs.getString("time")+"</td>");
                                out.println("<td>"+rs.getString("open")+"</td>");
                                out.println("<td>"+rs.getString("high")+"</td>");
                                out.println("<td>"+rs.getString("low")+"</td>");
                                out.println("<td>"+rs.getString("close")+"</td>");
                                out.println("<td>"+rs.getString("volume")+"</td>");
                        out.println("</tr>");
                }
        }
        else
        {
                out.println("<h2> No such stocks available between the dates you had entered
</h2>");
        }
```

4.1.5: Graphing: Graph for the same using a library canvas JS to graph all the data:

```
<script>

var chart = new CanvasJS.Chart("chartContainer", {
        animationEnabled: true,
        theme: "light2",
        title:{
                text: "Stock Prices"
        },
```

```
	axisX:{
		crosshair: {
			enabled: true,
			snapToDataPoint: true
		}},
	axisY:{
		title: "<% out.print(type.toUpperCase()); %>",
		crosshair: {		enabled: true		},
		includeZero: false
	},

	toolTip:{		shared:true		},
	legend:{
		cursor:"pointer",
		verticalAlign: "bottom",
		horizontalAlign: "left",
		dockInsidePlotArea: true,
		itemclick: toogleDataSeries
	},
	data: [{
		type: "line",

		name: "<% out.print(type.toUpperCase()); %>",
		markerType: "square",
		xValueFormatString: "DD MMM, YYYY",

		dataPoints: [
		<% out.print(data3); %>
		]		}]	});
chart.render();
function toogleDataSeries(e){
	if (typeof(e.dataSeries.visible) === "undefined" || e.dataSeries.visible) {
		e.dataSeries.visible = false;
	} else
```

```
                e.dataSeries.visible = true;
        chart.render();
}
</script>
```

4.1.6: Buying Stocks: JSP Code to buy any new stocks

```
String uid = (String)request.getSession().getAttribute("userid");

float money=0;
rs = st.executeQuery("select current_money from bank WHERE user_id="+uid+";");
if(rs.next())
        money = Float.valueOf(rs.getString("current_money"));

rs = st.executeQuery("select * from Price WHERE symbol = '"+ request.getParameter("stock") + "'
AND time >= '"+request.getParameter("date")+"' ORDER BY time ASC LIMIT 1;");
if(rs.next())
        {
                float prices = Float.valueOf(rs.getString("close"));
                int volume = Integer.parseInt(request.getParameter("volume"));
                String date = rs.getString("time");
                float sale = volume*prices;
                if(sale>money)
                {
                out.println("<h2> Sorry, you have insufficient balance to buy the stocks</h2>");
                }
                else
                {
                        out.println("Date is :"+date);
                        ResultSet sp = st.executeQuery("select * from investments WHERE
symbol='"+request.getParameter("stock")+"' AND user_id = '"+uid+"' AND time='"+date+"';");
                if(sp.next())
                        {
                                PreparedStatement ps = null;
```

```
                String sql="Update investments set volume = volume + "+volume+"
WHERE user_id='"+uid+"' AND time='"+date+"'";
                        ps = con.prepareStatement(sql);
                        int i = ps.executeUpdate();


                }
                else
                {
                CallableStatement cstat = con.prepareCall("{call buy (?,?,?,?,?)}");
                        cstat.setString(1, uid);
                        cstat.setString(2, request.getParameter("stock"));
                        cstat.setString(3, String.valueOf(volume));
                        cstat.setString(4, String.valueOf(prices));
                        cstat.setString(5, String.valueOf(date));
                ResultSet sp2 = cstat.executeQuery();
                }
                PreparedStatement ps = null;
            String sql="Update bank set current_money = current_money - "+sale+" WHERE
user_id='"+uid+"'";
                ps = con.prepareStatement(sql);
                int i = ps.executeUpdate();
                ResultSet rs2 = st.executeQuery("select current_money from bank WHERE
user_id="+uid+";");
                if(rs2.next())
                        out.println("<h2>Shares successfully added, New balance :<i
class='fa fa-rupee-sign'></i> "+rs2.getString("current_money"));


                }
            }


            else
                out.println("<h2> No such stock available from given date </h2>");
    %>
```

4.1.7: Selling Stocks: JSP Code to sell any stocks which user might have

```
ResultSet rs;
String uid = (String)request.getSession().getAttribute("userid");;
String buydate = request.getParameter("buydate");
String selldate = request.getParameter("selldate");
float money=0,sell_sale=0;
rs = st.executeQuery("select current_money from bank WHERE user_id="+uid+";");
rs.next();
money = Float.valueOf(rs.getString("current_money"));

ResultSet sellrs = st.executeQuery("select * from Price WHERE
symbol='"+request.getParameter("stock")+"' AND time >= '"+selldate+"' ORDER BY time ASC
LIMIT 1;");
if(sellrs.next())
{
float prices1 = Float.valueOf(sellrs.getString("close"));
int volume1 = Integer.parseInt(request.getParameter("volume"));
String sell_date = sellrs.getString("time");
sell_sale = volume1*prices1;

ResultSet buyrs = st.executeQuery("select * from investments WHERE
symbol='"+request.getParameter("stock")+"' AND time = '"+buydate+"' ORDER BY time ASC
LIMIT 1;");
if(buyrs.next())
{
        int volume2 = Integer.parseInt(buyrs.getString("volume"));
        String buy_date = buyrs.getString("time");
        float buy_Price = buyrs.getFloat("price_bought");
        float buy_sale = buyrs.getFloat("price_bought") * volume1;

        if(volume1 > volume2)
                out.println("<h2> You can't sell stocks more than the volume you have </h2>");
        else
        {
                float total_pl = sell_sale - buy_sale;
```

```
if(volume1 == volume2)
{
    PreparedStatement ps = null;
    //out.println("DELETE FROM investments WHERE user_id='"+uid+"' AND
symbol='"+request.getParameter("stock")+"' AND time = '"+buydate+"'");
    String sql="DELETE FROM investments WHERE
symbol='"+request.getParameter("stock")+"' AND time = '"+buydate+"'";
    ps = con.prepareStatement(sql);
    int i = ps.executeUpdate();
}
else
{
    PreparedStatement ps = null;
    String sql="Update investments set volume = volume - "+volume1+"
WHERE user_id='"+uid+"' AND time='"+buydate+"' AND symbol =
'"+request.getParameter("stock")+"'";
    ps = con.prepareStatement(sql);
    int i = ps.executeUpdate();
}

CallableStatement cstat = con.prepareCall("{call sell (?,?,?,?,?,?,?,?,?)}");
cstat.setString(1, uid);
cstat.setString(2, request.getParameter("stock"));
cstat.setString(3, String.valueOf(volume1));
cstat.setString(4, String.valueOf(buy_Price));
cstat.setString(5, String.valueOf(prices1));
cstat.setString(6, String.valueOf(java.time.LocalDate.now()));
cstat.setString(7, String.valueOf(buydate));
cstat.setString(8, String.valueOf(sell_date));
cstat.setString(9, String.valueOf(total_pl));
ResultSet sp2 = cstat.executeQuery();

PreparedStatement ps = null;
String sql="Update bank set current_money = current_money + "+sell_sale+"
WHERE user_id='"+uid+"'";
```

```
                ps = con.prepareStatement(sql);
                int i = ps.executeUpdate();


                ResultSet rs2 = st.executeQuery("select current_money from bank WHERE
user_id="+uid+";");
                if(rs2.next())
                {
                        out.println(sell_date);
                        out.println("<h2>Shares successfully sold worth "+sell_sale+" and total
profit/loss is "+total_pl+"<br/> New bank balance :<i class='fa fa-rupee-sign'></i>
"+rs2.getString("current_money"));


                }
        }
}


else
        out.println("<h2> No such stock available from given buying date </h2>");
}
else
        out.println("<h2> No such stock available from given selling date </h2>");
%>
```

4.2 TRIGGERS AND STORED PROCEDURES

Trigger is used in Stock market to initialize a new user into banks with an initial deposit of 1 lakh the moment a new user is registered.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `users_AFTER_INSERT` AFTER
INSERT ON `users` FOR EACH ROW
BEGIN
INSERT INTO `project`.`bank` (`bank_name`,`user_id`,`current_money`)
 VALUES (new.bank,new.user_id,100000);
END ;
```

Stored procedure is used in all the user forms to register the user.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `register`(in name char(50), in email char(255), in password char(155), in phone char(12), in bank varchar(30))
BEGIN
INSERT INTO users(`name`,`email`,`password`,`phone`,`bank`)
        VALUES( name,email,password,phone,bank);
END
```

Stored procedure is also used in buy stocks.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `buy in uid char(50), in symbol char(255), in volume char(155), in prices varchar(30),in time2 varchar(30))
BEGIN
INSERT INTO investments VALUES (uid,symbol,volume,prices,time2);
END;
```

4.3 RESULT

The resulting system is able to:

- Authenticate user credentials during login.
- Salted encryption for security of user passwords.
- Register new users and link to their banks.
- Allow users to view historical data of all the stocks.
- Allow user to see the companies present in the stock market.
- Ability to buy and trade shares in the companies they want to.
- Sell the stocks when they feel they have made a profit.

Chapter 5

TESTING

5.1 SOFTWARE TESTING

Testing is the process used to help identify correctness, completeness, security and quality of developed software. This includes executing a program with the intent of finding errors. It is important to distinguish between faults and failures. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. Most testing occurs after system requirements have been defined and then implemented in testable programs.

5.2 MODULE TESTING AND INTEGRATION

Module testing is a process of testing the individual subprograms, subroutines, classes, or procedures in a program. Instead of testing whole software program at once, module testing recommend testing the smaller building blocks of the program. It is largely white box oriented. The objective of doing Module testing is not to demonstrate proper functioning of the module but to demonstrate the presence of an error in the module. Module testing allows implementing of parallelism into the testing process by giving the opportunity to test multiple modules simultaneously.

The final integrated system too has been tested for various test cases such as duplicate entries and type mismatch.

5.3 LIMITATIONS

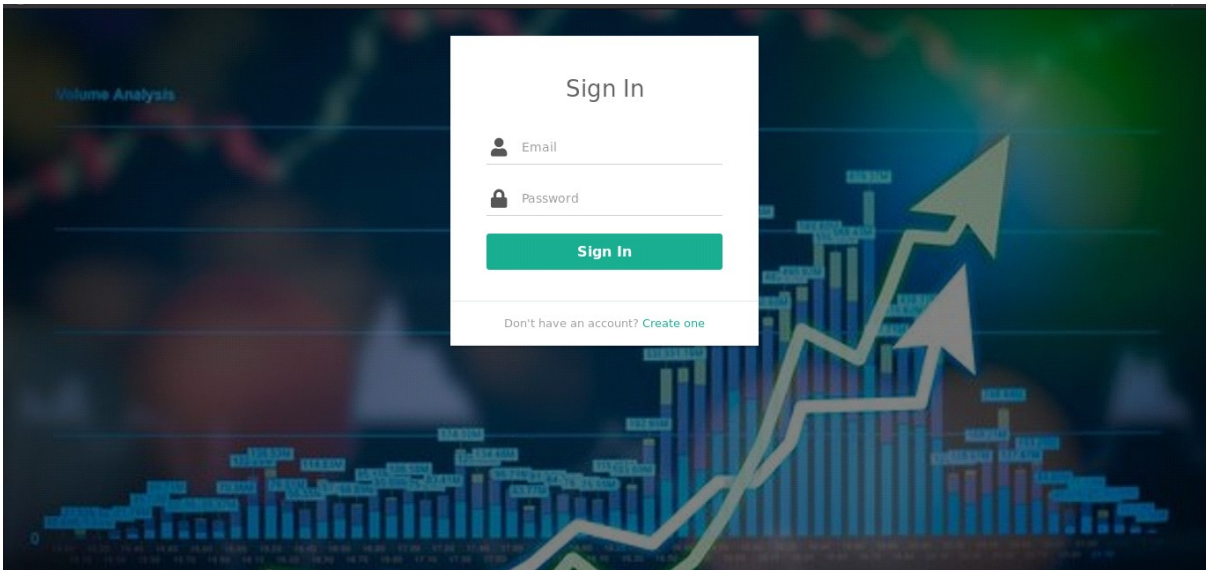
- Does not track markets at the live end and database is not fully up to date.
- User's session timing is not recorded
- Better secure interfaces needed for communication with the banks.
- Only restricted to Nifty 50 market currently, needs to be extended to more markets.

Chapter 6

SNAPSHOTS

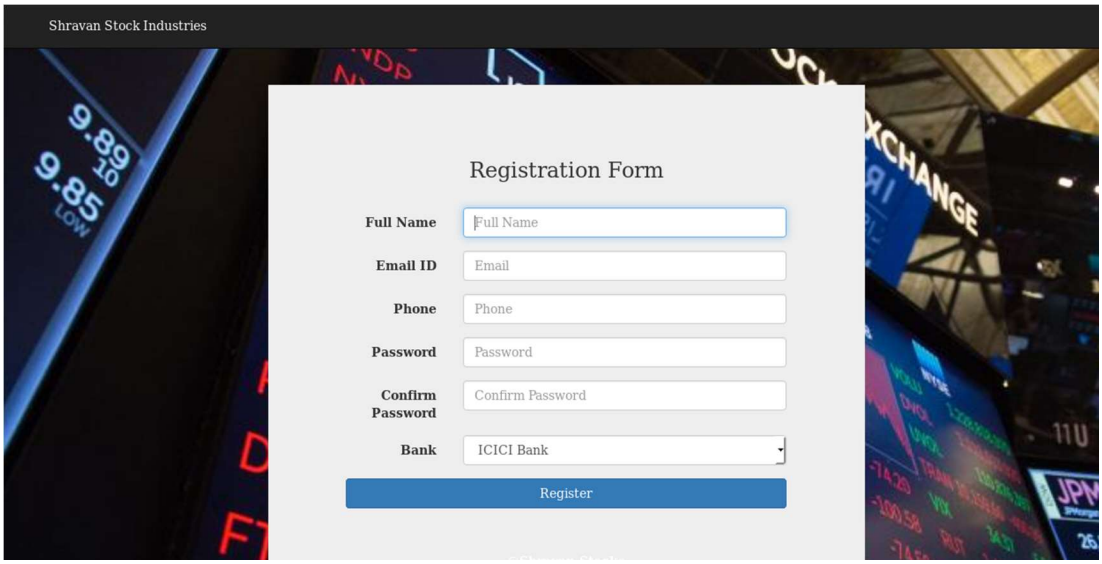
This chapter consists of working screenshots of the project.

6.1 LOGIN PAGE



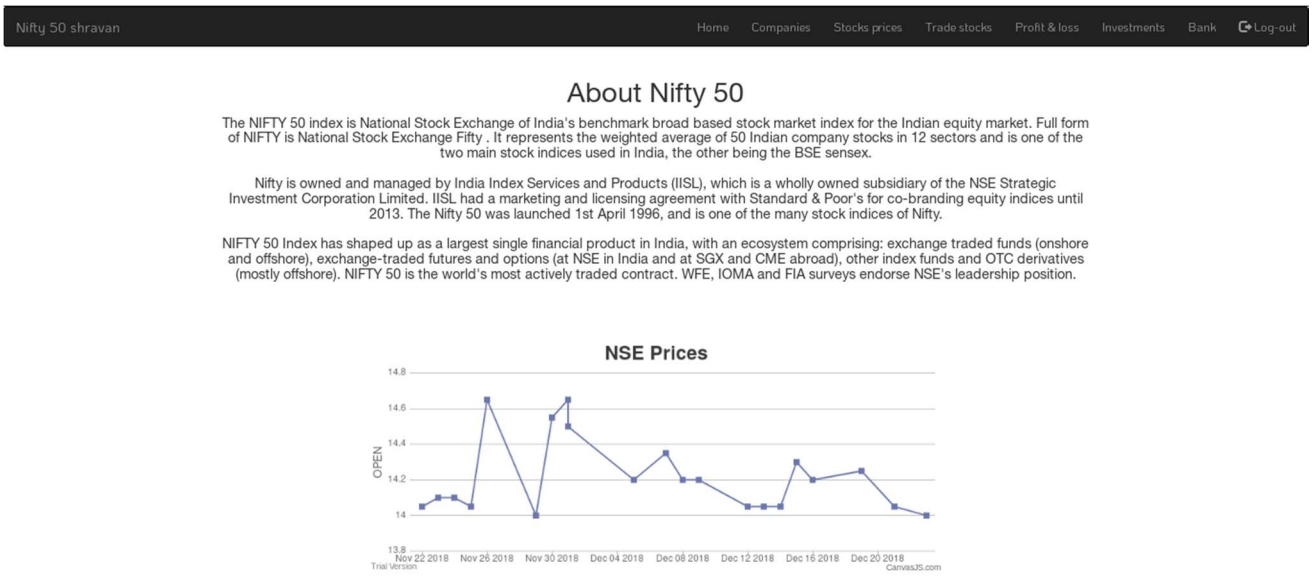
This is the login page for existing users and is the first page shown to any customer.

6.2 REGISTRATION PAGE



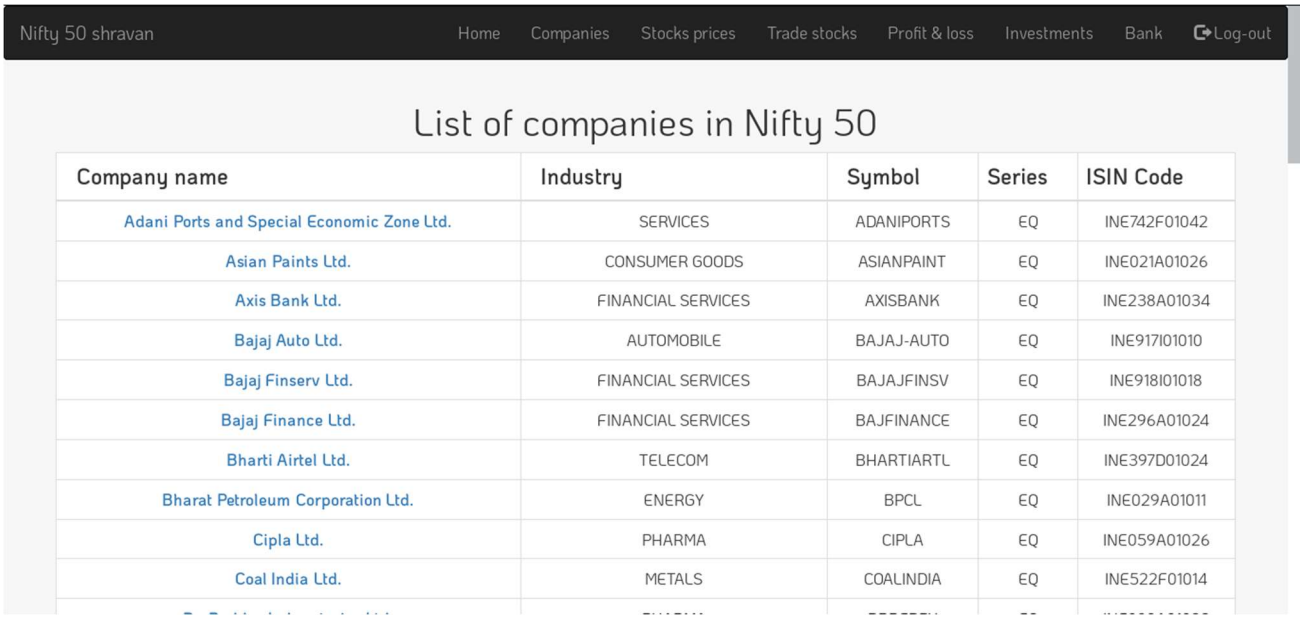
This is The registration page for any new users.

6.3 HOME PAGE



First home page shown to customers after login.

6.4 LIST OF COMPANIES



Company name	Industry	Symbol	Series	ISIN Code
Adani Ports and Special Economic Zone Ltd.	SERVICES	ADANIPORTS	EQ	INE742F01042
Asian Paints Ltd.	CONSUMER GOODS	ASIANPAINT	EQ	INE021A01026
Axis Bank Ltd.	FINANCIAL SERVICES	AXISBANK	EQ	INE238A01034
Bajaj Auto Ltd.	AUTOMOBILE	BAJAJ-AUTO	EQ	INE917I01010
Bajaj Finserv Ltd.	FINANCIAL SERVICES	BAJAJFINSV	EQ	INE918I01018
Bajaj Finance Ltd.	FINANCIAL SERVICES	BAJFINANCE	EQ	INE296A01024
Bharti Airtel Ltd.	TELECOM	BHARTIARTL	EQ	INE397D01024
Bharat Petroleum Corporation Ltd.	ENERGY	BPCL	EQ	INE029A01011
Cipla Ltd.	PHARMA	CIPLA	EQ	INE059A01026
Coal India Ltd.	METALS	COALINDIA	EQ	INE522F01014

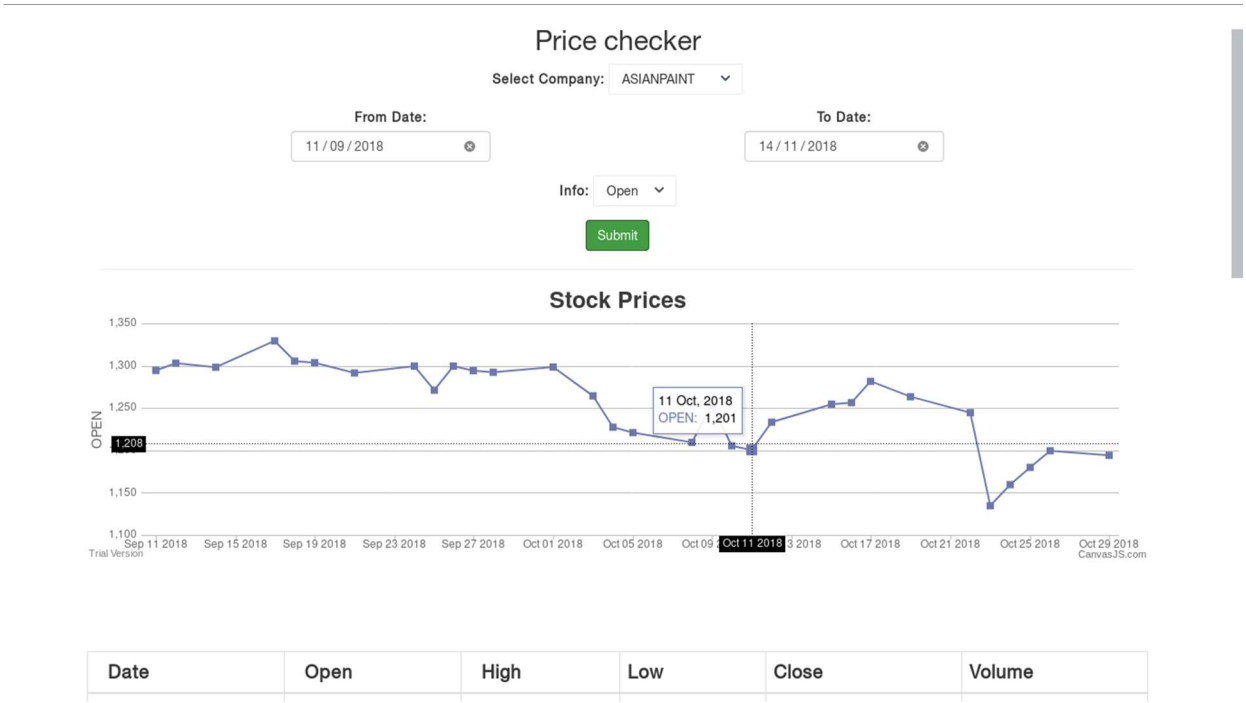
List of Nifty 50 Companies used in our system.

6.5 LATEST STOCK PRICE

Company name	Open	High	Low	Close	Volume
ADANIPORTS	368.45	368.50	356.00	360.05000	2294392
ASIANPAINT	1329.95	1349.50	1323.60	1348.00000	966456
AXISBANK	618.65	628.00	616.50	622.70000	4341256
BAJAJ-AUTO	2606.00	2616.10	2576.00	2585.14990	123296
BAJAJFINSV	5730.20	5824.80	5721.35	5746.79980	98365
BAJFINANCE	2400.00	2411.90	2346.95	2363.00000	722592
BHARTIARTL	328.05	338.90	328.05	336.60000	3350134
BPCL	330.10	332.90	323.45	324.65000	4360679
CIPLA	520.00	523.35	512.80	515.60000	974936
COALINDIA	256.95	258.50	251.00	251.55000	2218223
DRREDDY	2607.00	2637.90	2586.00	2592.45000	523247
EICHERMOT	24160.00	24488.00	23626.60	23779.94920	38445

This page shows the latest stock market prices for all companies from the database.

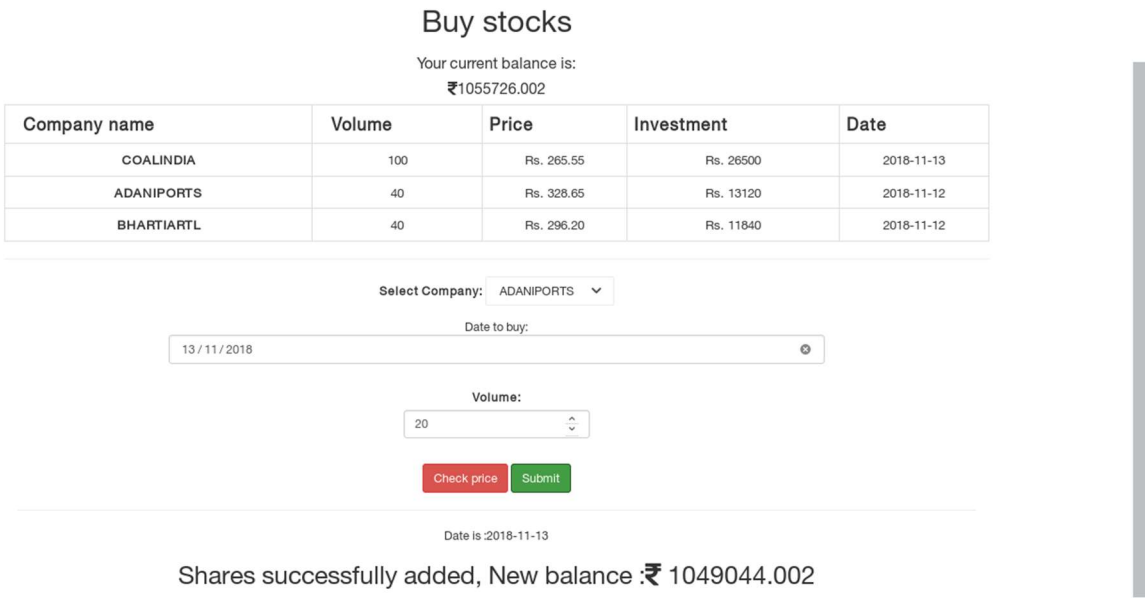
6.6 HISTORICAL STOCK PRICES



Date	Open	High	Low	Close	Volume

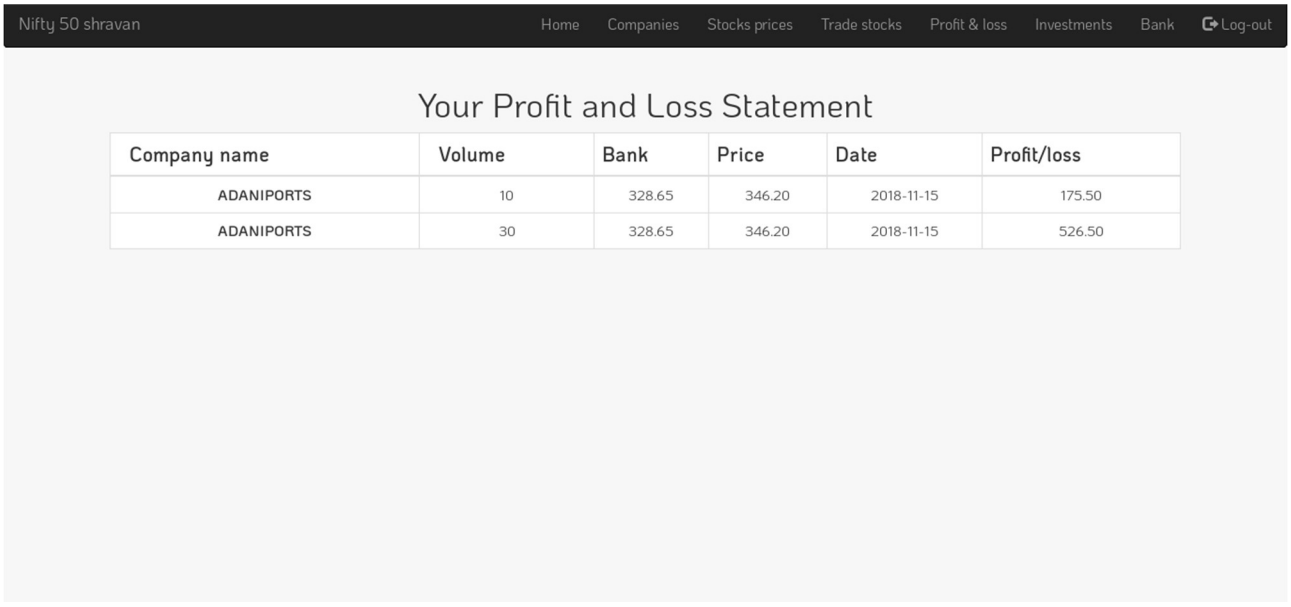
This allows users to see high, open, close and volume traded over the selected time range for any company they see.

6.7 BUY STOCKS PAGE



This page allows you to invest money in the stock of your choice.

6.8 PROFIT AND LOSS PAGE



We can see all investments made and their respective profit/loss made from the investments.

CONCLUSION

The Stock Market System provides easier maintenance of various stocks that person will invest in. It allows simplified operation and is a time saving platform with the ability to view historical data and thus invest easily and carefully on the various companies. The application has been completed successfully and tested with suitable test cases. It is user friendly and contains suitable options for users and shareholders. This is developed using HTML5, CSS, JavaScript, JSP and SQL. The goals achieved by this project are:

- Centralized database
- Easier buying, selling of various stocks.
- User friendly environment.
- Efficient management of stocks.
- Ability to view historical data and analyse them for better growth.
- View profits and loss statements, current investments in various companies.

FUTURE ENHANCEMENTS

Future upgrades to this project will implement:

- Better interfaces for the ability to view the stock prices of various companies including better analytics, more data across various companies, sectors and industries
- More stock market platforms including Sensex, Dow Jones etc.
- Ability to trade in forex exchanges and mutual funds.
- Better banking implementations between the customer and his bank.
- Ability to see and analyse the various companies customers tend to trade and analyse these for better info.
- Ability to view timely data across various years and months between various time ranges as required.

REFERENCES

1. Ramakrishnan, R., & Gehrke, J. (2011). Database management systems. Boston: McGraw-Hill.

2. Monson-Haefel, R. (2007). J2EE Web services. Boston, Mass: Addison-Wesley. Silberschatz A., Korth H. F., & Sudarshan S. (2011).

3. Database systems concepts. Estados Unidos: McGraw-Hill Companies, Inc.

4. Hanna P. (2002): JSP 2.0 The Complete Reference, Second Edition McGraw Hill Education.

5. David F. (2011). JavaScript: The Definitive Guide Sixth edition.

6. https://www.w3schools.com

7. https://www.canvasjs.com

8. https://getbootstrap.com/

9. https://fontawesome.com