Kristu Jayanti College

AUTONOMOUS Bengaluru

Reaccredited 'A' Grade by NAAC | Affiliated to Bengaluru North University

PROJECT REPORT ON

"E-Banking System"

Submitted in partial fulfillment of the requirements for Enterprise Computing Project Lab for 4th Semester

Master of Computer Applications

Submitted by

Ashwini G ***19CS801038***

Jinu John ***19CS801019***

Under the guidance of

Dr. Velmurugan R

CERTIFICATE

This is to certify that the project titled ***"E-Banking System"*** has been satisfactorily completed by **Ms. Ashwini G** with **19CS801038**, in partial fulfillment of the requirements for ***Enterprise Computing Project Lab*** with course code **MCA2P2A41,** for the 4th Semester MCA course during the academic semester from February 2021 to April 2021 as prescribed by Bangalore University.

Faculty In-charge **Head of the Department**

Valued by

Examiner 1:_____ Date :

Examiner 2:_____ Centre: Kristu Jayanti College

ACKNOWLEDGEMENT

First of all, we would like to thank the God Almighty for all the blessings he has showered on us. Our spiritual quotient gave us more strength and motivation that helped immensely.

We would like to thank **Rev. Dr. Augustine George,** our Principal, for providing us his constant guidance and support. I would also like to thank **Rev. Fr. Lijo P Thomas,** our Financial Administrator, for providing us with the best facilities.

We are extremely thankful to our **Dr. R. Kumar**, Head, Department of Computer Science (PG) for giving us the essential support in the form of allocating comfortable lab hours and necessary resources.

We would like to extend our heartfelt thanks to **Dr. Velmurugan R**, our project guide for providing us the necessary details related to project development and process identification enabling us to finish the project within the stipulated time.

We thank all other faculty members who helped us a lot in completing this project. The computer lab was always open for us. We thank the lab administrator and other technical staff for their help and support.

We thank our class mates, who have pointed out errors and guided us a lot and we thank each and every one who has helped us.

Synopsis

"E-Banking System" is an enterprise project for managing their accounts at anywhere, anytime. This project enables the customer to perform the basic transactions through their PC or laptop. The customer can access the bank's website for viewing their account details and perform their transactions on account as per their requirements. With E-Banking, the brick and the mortar structure of the traditional banking gets converted into click and portal model.

E-Banking enables all the processes to be done online. This system is meant to overcome the drawbacks of the manual system. As the records are been maintained by the system, it is easy for the admin who is responsible for the system to update, retrieve, insert data successfully.

Modules:

1. **Registration** : In this, the customer will give his/her details and register their names.

2. **Account Status Check** : Here, the customer can check whether his/her status is active or inactive after the registration.

3. **Create Net Banking** : Here, the customer can create net banking through which they will be provided with a secret pin for further login.

4. **Customer** : Here, the customer can view their profile, edit and transfer money and receive e-statement.

5. **Admin** : Here, the admin can approve customer's details, create admin, create account, generate account number, block and unblock and finally, see all the transactions.

Software Specification:

Front-end : Java Enterprise, HTML, CSS, JavaScript

Back-end : Workbench

CONTENTS

1. INTRODUCTION

1.1 Problem Definition:

Back then, when we used to visit the bank for any financial purposes, the records were been entered manually and so there was always a possibility of losing the data and difficulty in finding out when required. Therefore, as the technology is improving, it is always a good practice and safer to maintain the records in the system where we can retrieve data when required. As the records are being maintained, it makes it easy for the Admin who is responsible for the system to update, retrieve, insert and delete data successfully.

E-Banking system is an enterprise project for managing their accounts at anywhere, anytime. This project enables the customer to perform the basic transactions through their PC or laptop. The customer can access the bank's website for viewing their account details and perform their transactions on account as per their requirements.This system is meant to overcome the drawbacks of the manual system. As the records are been maintained by the system, it is easy for the admin who is responsible for the system to update, retrieve, insert data successfully.

This application is developed by using Java Enterprise, HTML, JavaScript, CSS as front end and Workbench as back end.

1.2 Project Description:

The E-Banking System is built using Java Enterprise, HTML, CSS & JavaScript as a front end and Workbench as a back end . This system enables the customer to perform the basic transactions through their PC or laptop. The customer can access the bank's website for viewing their account details and perform their transactions on account as per their requirements. E-Banking enables all the processes to be done online. And this system is meant to overcome the drawbacks of the manual system. As

the records are been maintained by the system, it is easy for the admin to update, retrieve, insert data successfully.

*** User** - Admin, Customer

-Create Account

-Account Status check

-Create Net banking

*** Customer Module** -Money transfer

-Edit Profile

-Transaction Report

-Print Passbook

*** Admin** - Approve Account create customer

-Create Admin

-Create Account

-Show Transaction

-Edit Customer Information

2. SYSTEM STUDY

2.1 Existing System:

Existing system refers to the system that is being followed till now. The existing system requires more computational time, more manual calculations, and the complexity involved in selection of features is high. The other disadvantages are lack of security of data, deficiency of data accuracy, time consuming etc. To avoid all these limitations and make the working more accurately the system needs to be computerized. Here, in E-Banking System, a detailed study of existing system is carried along with all

the steps in system analysis. An idea for creating a better project was carried and the next steps were followed.

- Lack of security of data.
- More man power.
- Time consuming.
- Needs manual calculations.
- Necessity of the skilled persons.
- Excessive paper work.

2.2 Proposed System:

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work. The existing system has several disadvantages and many more difficulties to work well. The proposed system tries to eliminate or reduce these difficulties up to some extent. The proposed system will help the user to reduce the workload and mental conflict. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging.

❖ The system is very simple in design and to implement.

❖ The system requires very low system resources and the system will work in almost all configurations. It has got following features :-

- Ensure data accuracy.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better Service.

- This would help the corporation prepare and organize its schedules more efficiently on the basis of traffic demand.

- It would provide data on concessions given to various sections.

2.3 Use Case Diagram:

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

Purpose of Use Case Diagrams

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose.

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

Notations:

Actor:

An actor represents a role that an outsider takes on when interacting with the business system. For instance, an actor can be a customer, a business partner, a supplier, or another business system.

Every actor has a name:

Instead of a stick figure, other symbols can be used as well, if they fit the characteristics of the actor and lead to practical, easy-to-read diagrams.

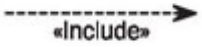
Association:

An association is the relationship between an actor and a business use case. It indicates that an actor can use a certain functionality of the business system ; the business use case:

Include Relationship:

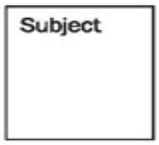
The include relationship is a relationship between two business use cases that signifies that the business use case on the side to which the arrow points is included in the use case on the other side of the arrow. This means that for one functionality that the business system provides, another functionality of the business system is accessed.

In this way, functionalities that are accessed repeatedly can be depicted as individual business use cases, which can be used in multiple ways:

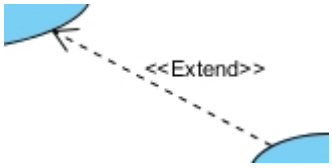


Subject:

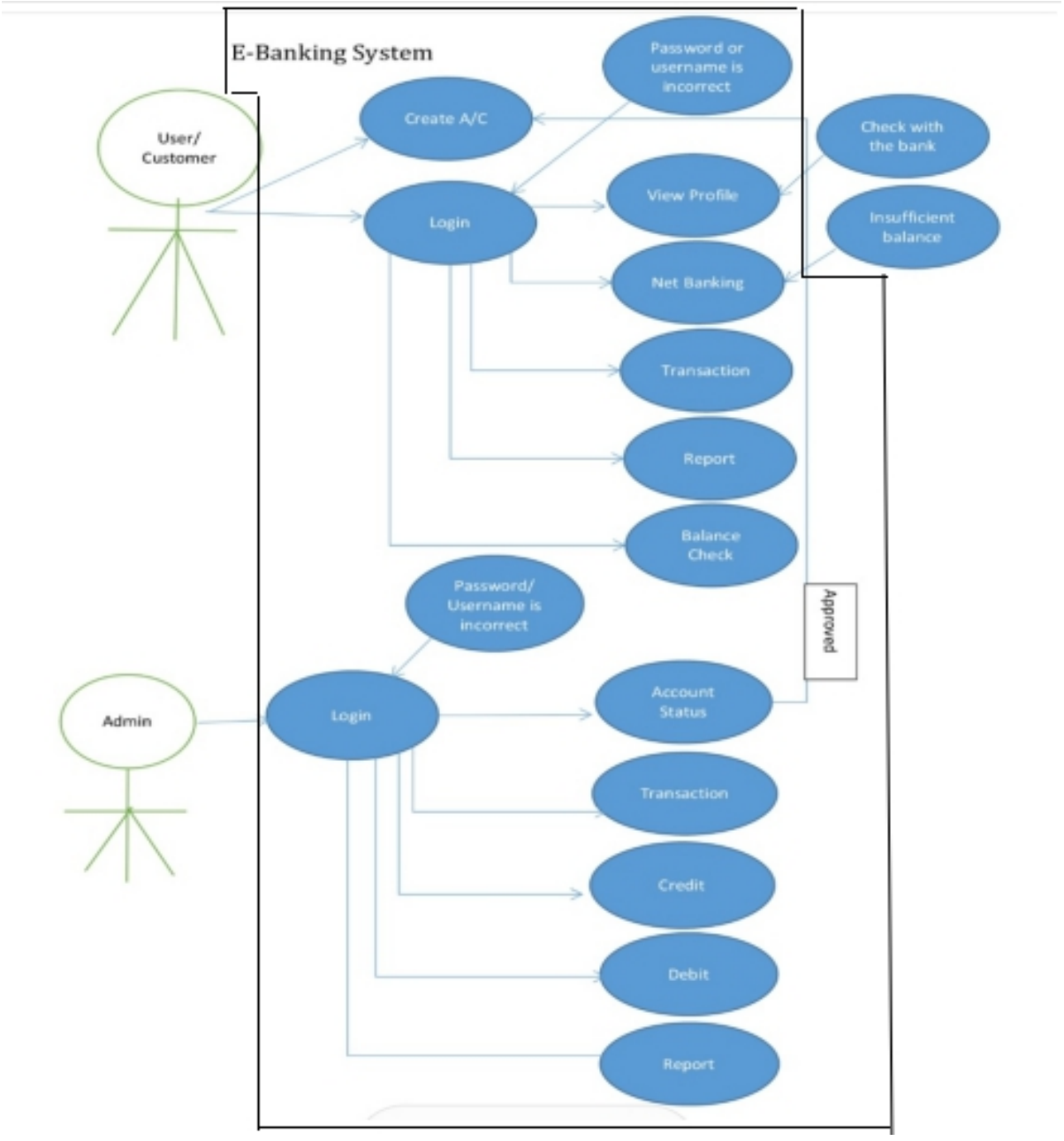
A subject describes a business system that has one or more business use cases attached to it. A subject is represented by a rectangle that surrounds attached business use cases and is tagged with a name:



Extend:



An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.



2.4 Activity diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The control flow is

drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Basic Activity Diagram Notations and Symbols

Initial State or Start Point:
A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.

Start Point/Initial State

Activity or Action State:
An action state represents the non-interruptible action of objects. You can draw an action state in SmartDraw using a rectangle with rounded corners.

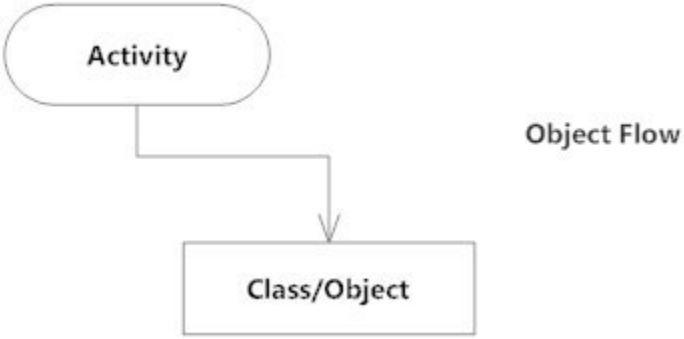
Activity

Action Flow:

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

Object Flow:

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



Decisions and Branching:

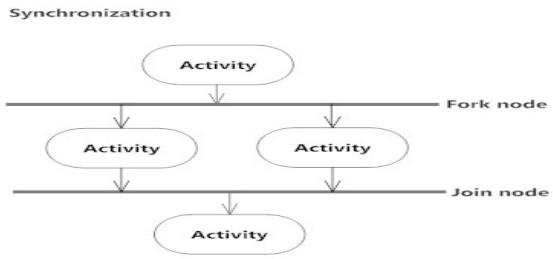
A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

Synchronization:

A fork node is used to split a single incoming flow into multiple concurrent flows. It is represented as a straight, slightly thicker line in an activity diagram.

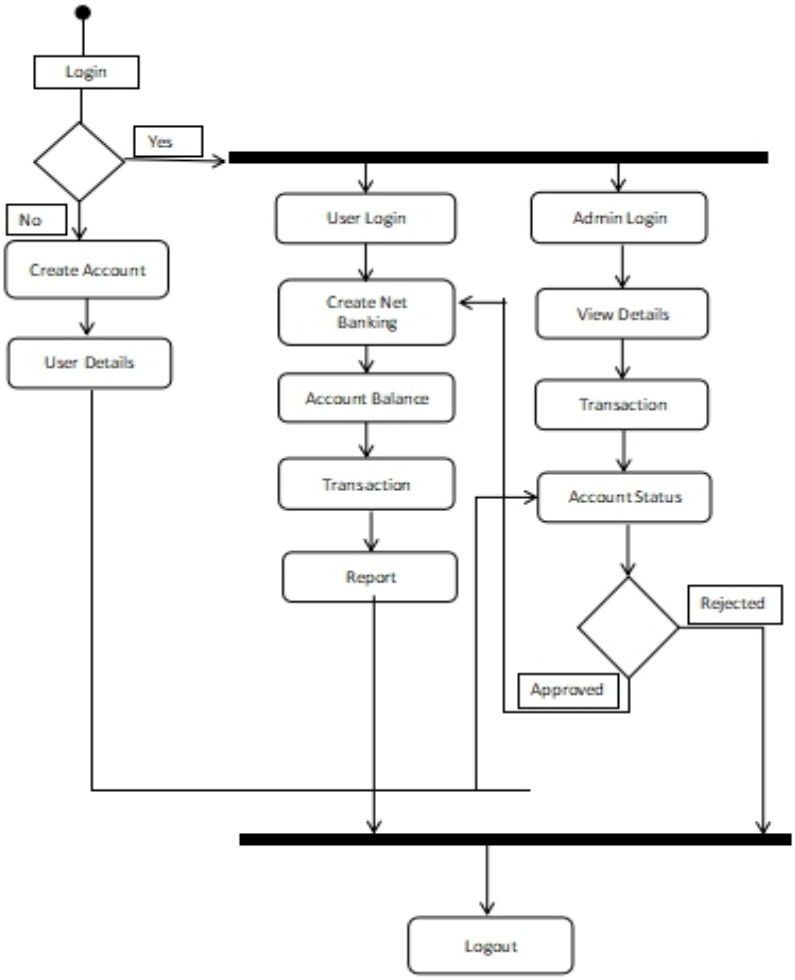
A join node joins multiple concurrent flows back into a single outgoing flow.

A fork and join mode used together are often referred to as synchronization.



Final State or End Point:

An arrow pointing to a filled circle nested inside another circle represents the final action state.



3. SYSTEM CONFIGURATION

3.1 Hardware Configuration:

HARDWARE COMPONENTS	REQUIREMENTS
Processor	2.8GHz Processor and Above
RAM	2 GB
Memory	500 GB

3.2 Software Configuration:

Front-End	HTML, JAVASCRIPT, CSS (Eclipse IDE for Enterprise Java Developers)
Back-End	Workbench
Operating System	Windows 10

4. DETAILS OF THE SOFTWARE

4.1 Overview of Front-End:

Java Enterprise Edition:

The Java EE stands for Java Enterprise Edition, which was earlier known as J2EE and is currently known as Jakarta EE. It is a set of specifications wrapping around Java SE (Standard Edition). The Java EE provides a platform for developers with enterprise features such as distributed computing and web services. Java EE applications are usually run on reference run times such as micro servers or application servers. Examples of some contexts where Java EE is used are e-commerce, accounting, banking information systems. Java EE has several specifications which are useful in making web pages, reading and writing from database

in a transactional way, managing distributed queues. The Java EE contains several APIs which have the functionalities of base Java SE APIs such as Enterprise JavaBeans, connectors, Servlets, Java Server Pages and several web service technologies.

JavaScript:

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

It is an interpreted programming language with object-oriented capabilities.

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

Advantages of JavaScript:

The merits of using JavaScript are −

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript:

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

HTML:

Hypertext Markup Language (**HTML**) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

CSS (Cascading Style Sheets):

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

Advantages of CSS:

- **CSS saves time** − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- **Easy maintenance** − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** − Now HTML attributes are being deprecated and it is being recommended to use CSS. So, it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

4.2 Overview of Back-end:

Workbench Database:

Workbench is a software application for development, administration of multiple relational databases using SQL, with inter operationality between different database systems, developed by Upscene Productions.

Because Databases Workbench supports multiple database systems, it can provide software developers with the same interface and development environment for these otherwise different database systems and also includes cross database tools

Workbench can be used to view, create and edit tables, indexes, stored procedures and other database meta data objects. It also supports-

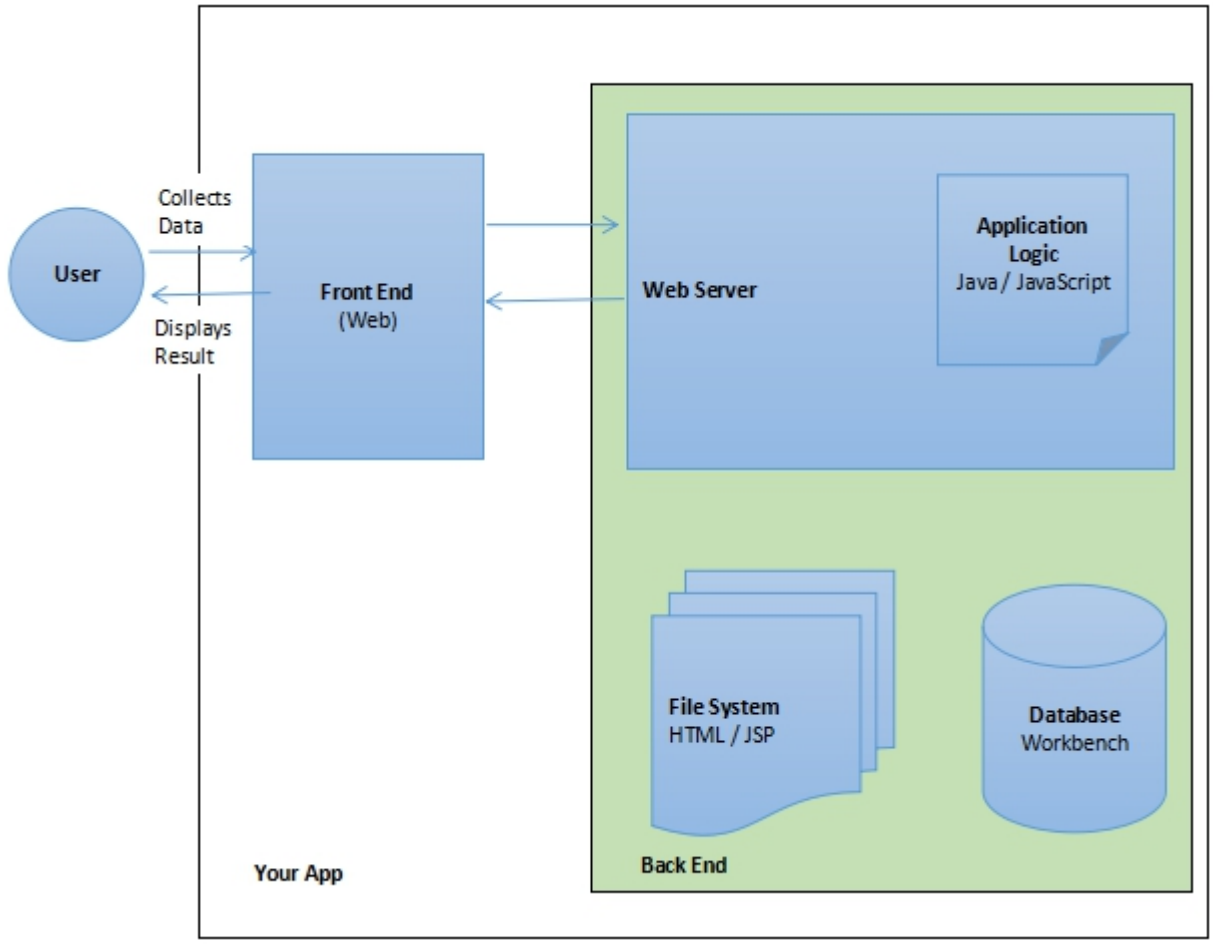
- testing SQL queries and viewing query plans.
- step-by-step debugging of stored routines.
- generating test data.
- transferring data between database systems (DataPump).
- import and export of data.
- database schema compare and change script creation.
- database schema migration, also from one database system to another.
- open ODBC or ADO data sources and MS Access databases.
- manage security items like users, groups and roles.
- create custom reports based on database queries.
- print database schema, source code, lists of objects or query result sets.

➢ **It includes several productivity features:**

- SQL Insight including "Join Completion"
- Parameter Insight
- Code Templates
- Object Templates
- Name Templates
- Two-way Visual Query Builder

5. SYSTEM DESIGN

5.1 Architectural Design:



5.2. Input Design:

The design of input also includes specifying the means by which end-users and system operators direct the system in which action to take. Input design consists of developing specification procedures for data preparation. These steps are necessary to put transaction into such usable form for processing and entry, the activity of putting the data into the computer processing. The objectives guiding the design of input focus on controlling the amount of input required, avoiding the delay, controlling the errors and

keeping the steps in simple. Input design is the process of converting user-oriented inputs to such computer based format. System analysts decide the input design as:

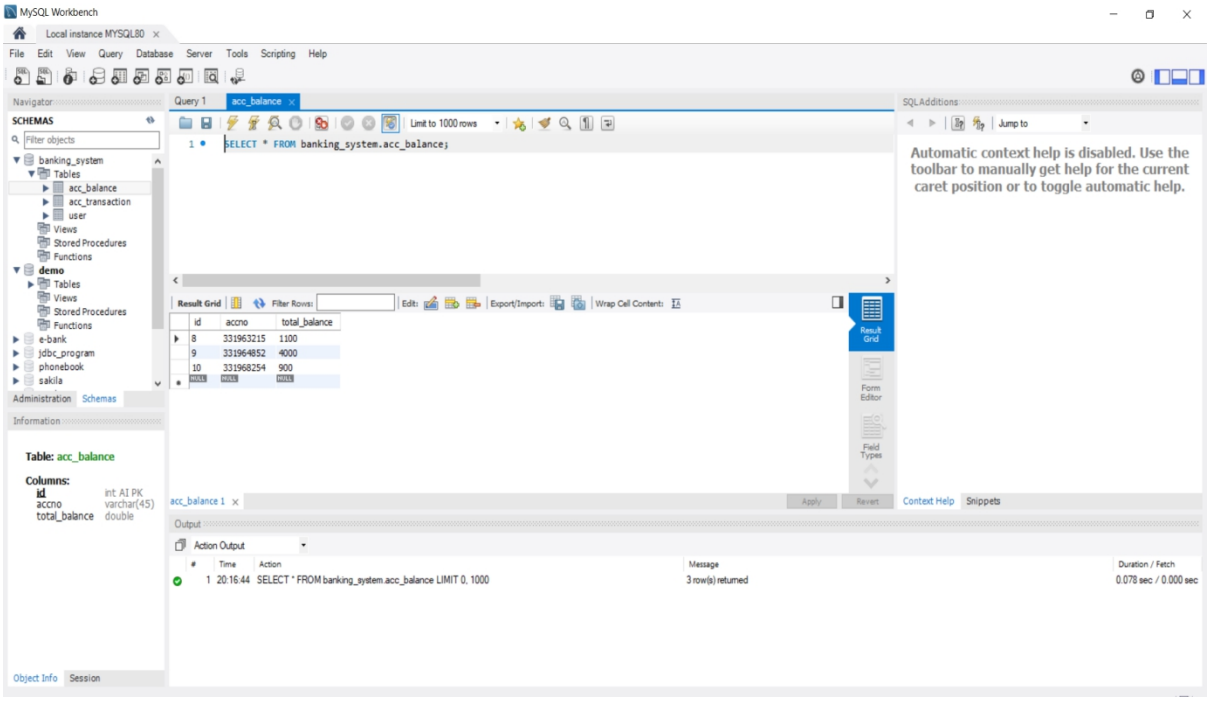
- What data to input?
- What medium to use?
- How the data should arrange or coded?

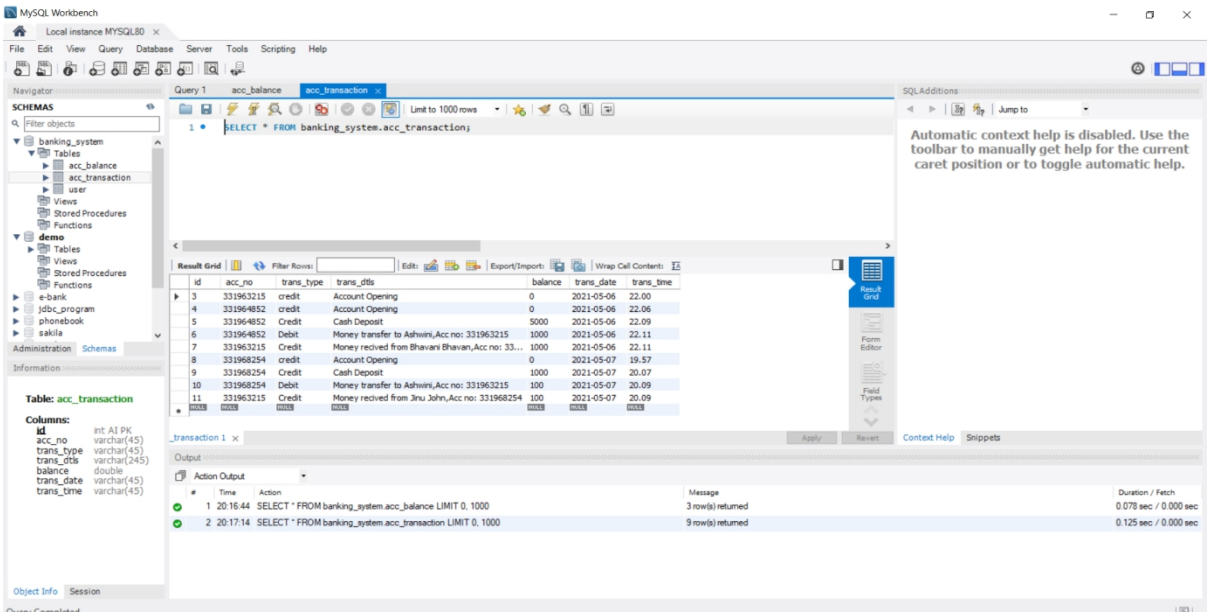
5.3. Output Design:

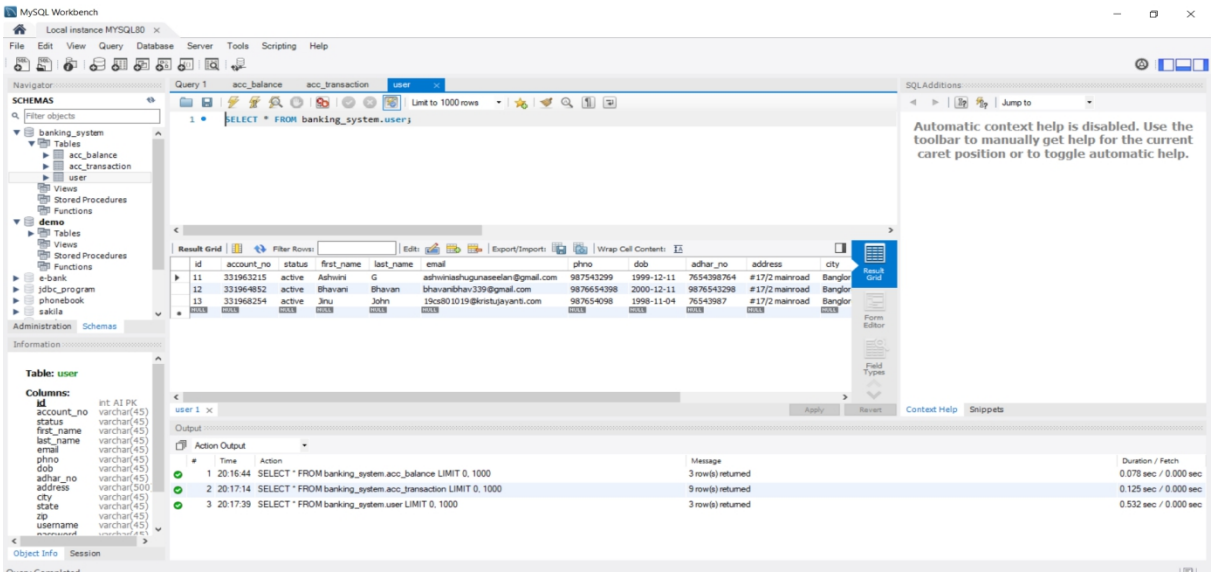
The output is the most important and direct source of the information to user. It is used to view the result of each operator we make. Efficient, intelligible output design should improve system‘s relationships with the user and help in decision making. System outputs are of three types, which are a report, a document and a message. When designing output, the systems analyst must accomplish:

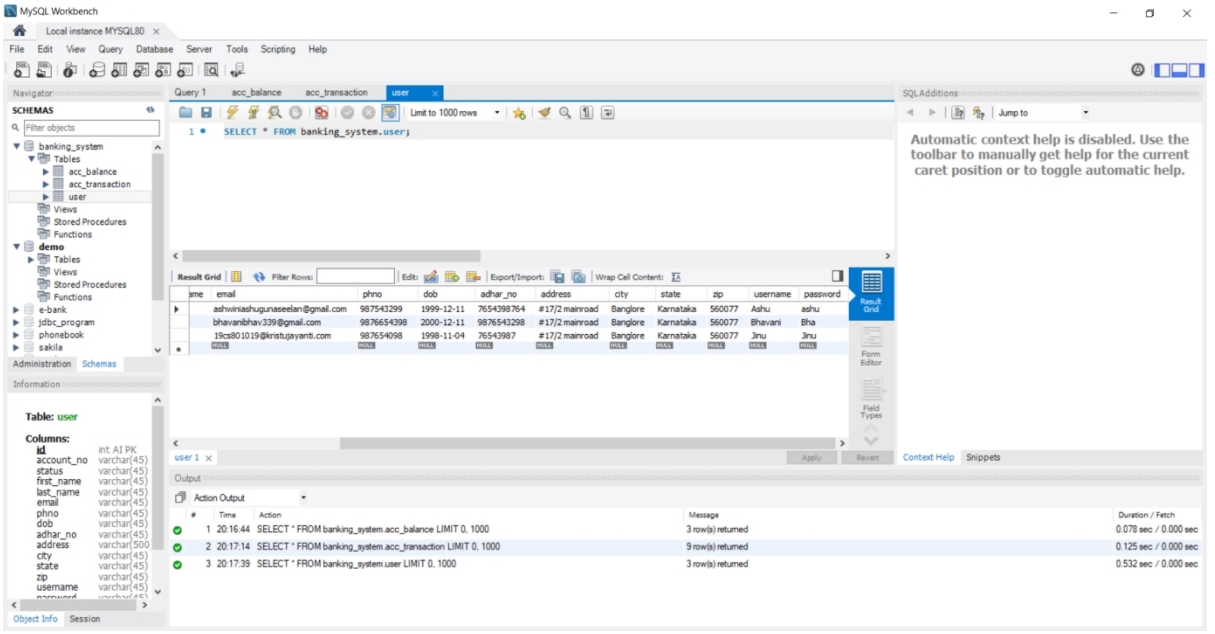
- Determine what information to present.
- Decide whether to display or print the information
- Arrange the presentation of information in an acceptable format.
- Decide how to distribute the output to intended receipts.

5.4. Database Design:






MySQL Workbench
Local instance MYSQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
banking_system
Tables
acc_balance
acc_transaction
user
Views
Stored Procedures
Functions
demo
e-bank
jdbc_program
phonebook
sakila
Administration Schemas
Information
Table: user
Columns:
id int AI PK
account_no varchar(45)
status varchar(45)
first_name varchar(45)
last_name varchar(45)
email varchar(45)
phno varchar(45)
dob varchar(45)
adhar_no varchar(45)
address varchar(500)
city varchar(45)
state varchar(45)
zip varchar(45)
username varchar(45)
Object Info Session
Query 1 acc_balance acc_transaction user
Limit to 1000 rows
1 SELECT * FROM banking_system.user;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
id account_no status first_name last_name email phno dob adhar_no address city
11 331963215 active Ashwini G ashwiniashugunaseelan@gmail.com 987543299 1999-12-11 7654398764 #17/2 mainroad Banglor
12 33196-4852 active Bhavani Bhavan bhavanibhav339@gmail.com 9876654398 2000-12-11 9876543298 #17/2 mainroad Banglor
13 331968254 active Jinu John 19cs801019@kristujayanti.com 987654098 1998-11-04 76543987 #17/2 mainroad Banglor
Result Grid Form Editor Field Types
user 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 20:16:44 SELECT * FROM banking_system.acc_balance LIMIT 0, 1000 3 row(s) returned 0.078 sec / 0.000 sec
2 20:17:14 SELECT * FROM banking_system.acc_transaction LIMIT 0, 1000 9 row(s) returned 0.125 sec / 0.000 sec
3 20:17:39 SELECT * FROM banking_system.user LIMIT 0, 1000 3 row(s) returned 0.532 sec / 0.000 sec


MySQL Workbench
Local instance MYSQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
banking_system
Tables
acc_balance
acc_transaction
user
Views
Stored Procedures
Functions
demo
e-bank
jdbc_program
phonebook
sakila
Administration Schemas
Information
Table: user
Columns:
id int AI PK
account_no varchar(45)
status varchar(45)
first_name varchar(45)
last_name varchar(45)
email varchar(45)
phno varchar(45)
dob varchar(45)
adhar_no varchar(45)
address varchar(500)
city varchar(45)
state varchar(45)
zip varchar(45)
username varchar(45)
Object Info Session
Query 1 acc_balance acc_transaction user
Limit to 1000 rows
1 SELECT * FROM banking_system.user;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
email phno dob adhar_no address city state zip username password
ashwiniashugunaseelan@gmail.com 987543299 1999-12-11 7654398764 #17/2 mainroad Banglore Karnataka 560077 Ashu ashu
bhavanibhav339@gmail.com 9876654398 2000-12-11 9876543298 #17/2 mainroad Banglore Karnataka 560077 Bhavani Bha
19cs801019@kristujayanti.com 987654098 1998-11-04 76543987 #17/2 mainroad Banglore Karnataka 560077 Jinu Jinu
Result Grid Form Editor Field Types
user 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 20:16:44 SELECT * FROM banking_system.acc_balance LIMIT 0, 1000 3 row(s) returned 0.078 sec / 0.000 sec
2 20:17:14 SELECT * FROM banking_system.acc_transaction LIMIT 0, 1000 9 row(s) returned 0.125 sec / 0.000 sec
3 20:17:39 SELECT * FROM banking_system.user LIMIT 0, 1000 3 row(s) returned 0.532 sec / 0.000 sec

6.SOURCE CODE

```
<%@page import="com.entity.AccountTransaction"%>
<%@page import="java.util.List"%>
<%@page import="com.db.DbConnect"%>
<%@page import="com.dao.UserDAOImpl"%>
<%@page import="com.entity.User"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: All Transaction</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body>
    <%@include file="all_component/navbar.jsp"%>
    <c:if test="${empty userobj}">
      <c:set value="Please Login" var="failedmsg" scope="session"></c:set>
    <c:redirect url="login.jsp"></c:redirect>
    </c:if>

    <div class="container-fluid">
    <div class="row">
    <div class="col-md-12">
    <h4 class="text-center">All
Transaction</h4>

<hr>
<table class="table table-striped mt-5">
<thead class="bg-primary text-light"

<tr>
```

```
<th scope="col">Account No</th>

<th scope="col">Credit / Debit</th>
<th scope="col">Transaction Details</th>

<th scope="col">Date</th>
<th scope="col">Time</th>
<th scope="col">Amount</th>

</tr>
</thead>
<tbody>
<%
User us = (User) session.getAttribute("userobj");
UserDAOImpl dao = new UserDAOImpl(DbConnect.getConn());
List<AccountTransaction> trans = dao.getAllTrans(us.getAccountNo());
for (AccountTransaction tr : trans) {
%>
<tr>
<th scope="row"><%=tr.getAccno()%></th>
<td><%=tr.getTransType()%></td>
<td><%=tr.getTransDtls()%></td>
<td><%=tr.getTransDate()%></td>
<td><%=tr.getTransTime()%></td>
<td><%=tr.getBalance()%></td>
</tr>
<%
}
%>
</tbody>
</table>

</div>
</div>
</div>

</body>
</html>
```

```
<%@page import="com.entity.User"%>
<%@page import="com.db.DbConnect"%>
<%@page import="com.dao.UserDAOImpl"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Balance</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body style="background-color: #f0f1f2;">
	<%@include file="all_component/navbar.jsp"%>
	<c:if test="${ empty userobj}">
	<c:set value="Please Login" scope="session" var="failedmsg"></c:set>
	<c:redirect url="login.jsp" />
	</c:if>
	<div class="container">
		<div class="row p-5">
			<div class="col-md-4 offset-md-4">
				<div class="card">
					<div class="card-body">
						<%
						User us = (User)
session.getAttribute("userobj");

UserDAOImpl dao = new UserDAOImpl(DbConnect.getConn());
Double bal = dao.checkBalance(us.getAccountNo());
						%>
<h5>Name: <%=us.getFirstName()+ us.getLastName()%></h5>
<h5>Account No: <%=us.getAccountNo()%></h5>
<h5>Balance: <%=bal%> <i class="fas fa-rupee-sign"></i></h5>
					</div>
```

```
                        </div>
                    </div>
                </div>
            </div>
</body>
</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Change Password</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body style="background-color: #f0f1f2;">
        <c:if test="${empty userobj}">
<c:set value="Please Login" var="failedmsg" scope="session"></c:set>
            <c:redirect url="login.jsp" />
        </c:if>

        <%@include file="/all_component/navbar.jsp"%>
        <div class="container">
            <div class="row m-3">
                <div class="col-md-4 offset-md-4">
                    <div class="card">
                        <div class="card-body">
<h5 class="text-center text-primary">Change Password</h5>
<c:if test="${not empty sucessmsg}">
<p class="text-center text-success">${sessionScope.sucessmsg}</p>
<c:remove var="sucessmsg" scope="session" />
                                    </c:if>
```

```
<c:if test="${not empty failedmsg}">
<p class="text-center text-danger">${sessionScope.failedmsg}</p>
<c:remove var="failedmsg" scope="session" />
                                        </c:if>

<form action="changepassword"
method="post"oninput='cp.setCustomValidity(cp.value != psw.value ?
"Passwords do not match." : "")'>

<c:if test="${not empty userobj}">
<input type="hidden" value="${userobj.accountNo}" name="accno">
                                        </c:if>

                                <div class="form-group">
<label for="exampleInputPassword1">Enter Old Password</label> <input
type="text" class="form-control" name="oldPwd" required="required">
                                </div>

<div class="form-group">
<label for="exampleInputPassword1">Enter New Password</label> <input
type="text" class="form-control" name=psw required="required">
</div>

<div class="form-group">
<label for="exampleInputPassword1">Confirm Password</label>
<inputtype="password" class="form-control" name=cp
required="required">
</div>

<div class="text-center">
<button type="submit" class="btn btn primary">Submit</button>
</div>
      </form>
      </div>
      </div>
      </div>
      </div>
      </div>
```

```
</body>
</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Open Account</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body style="background-color: #DCDCDC">
	<%@include file="all_component/navbar.jsp"%>

	<c:if test="${not empty SucessMsg}">
		<div class="alert alert-warning text-center" role="alert">
			<i class="fas fa-clock "></i>${SucessMsg}</div>
			<c:remove var="SucessMsg" scope="session"/>
	</c:if>
	<c:if test="${not empty ErrorMsg}">
		<div class="alert alert-warning text-center" role="alert">
			<i class="fas fa-clock "></i>${ErrorMsg}</div>
			<c:remove var="ErrorMsg" scope="session"/>
	</c:if>

	 <div class="container-fluid">
		<div class="row mt-2">
		<div class="col-md-8 offset-md-2">
		<div class="card">
		<div class="card-body">
		<form action="create_account" method="post">
		<div class="form-row">
		<div class="form-group col-md-6">
```

```
            <label for="inputEmail4">First Name</label> <input name="fn"
type="text" class="form-control" id="inputEmail4"
placeholder="First Name">
        </div>
        <div class="form-group col-md-6">
        <label for="inputPassword4">Last Name</label> <input name="ln"
type="text" class="form-control" id="inputPassword4"
placeholder="Last Name">
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<label for="inputEmail4">Email</label> <input name="em"
type="email" class="form-control" id="inputEmail4"
placeholder="Email">
</div>
<div class="form-group col-md-6">
<label for="Phonenumber">Phone Number</label> <input type="number"
class="form-control" name="ph">
</div>
</div>

<div class="form-row">
<div class="form-group col-md-6">
<label for="inputEmail4">DOB</label> <input name="dob"
type="date" class="form-control" id="inputEmail4"
placeholder="dd-mm-yyyy">
</div>
<div class="form-group col-md-6">
<label for="inputPassword4">Adhar Number</label> <input
name="adh" type="number" class="form-control" id="inputPassword4"
placeholder="" value="0000-0000-0000-0000">
</div>
</div>

<div class="form-group">
<label for="inputAddress">Address</label> <input name="add"
type="text" class="form-control" id="inputAddress"
```

```
placeholder="1234 Main St">
</div>

<div class="form-row">
<div class="form-group col-md-6">
<label for="inputCity">City</label> <input name="city"
type="text" class="form-control" id="inputCity">
</div>
<div class="form-group col-md-4">
<label for="inputCity">State</label> <input name="st"
type="text" class="form-control" id="inputState">
</div>
<div class="form-group col-md-2">
<label for="inputZip">Zip</label> <input type="number"
class="form-control" name="zip">
</div>
</div>
<div class="form-group">
<div class="form-check">
<input class="form-check-input" type="checkbox" id="gridCheck"
name="check"> <label class="form-check-label"
for="gridCheck"> Agree Terms & Condition </label>
</div>
</div>
<button type="submit" class="btn btn-primary">Sign in</button>
</form>
</div>
</div>
</div>
</div>
</div>

<div style="margin-top: 100px;">
<%@include file="/all_component/footer.jsp"%>
</div>
</body>
</html>
```

```
<%@page import="com.entity.AccountTransaction"%>
<%@page import="java.util.List"%>
<%@page import="com.db.DbConnect"%>
<%@page import="com.dao.UserDAOImpl"%>
<%@page import="com.entity.User"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Transaction Credit</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body>
    <%@include file="all_component/navbar.jsp"%>
    <c:if test="${empty userobj}">
    <c:set value="Please Login" var="failedmsg" scope="session"></c:set>
    <c:redirect url="login.jsp"></c:redirect>
    </c:if>

    <div class="container-fluid">
        <div class="row">
            <div class="col-md-12">
                <h4 class="text-center">All Credit Transaction</h4>

    <hr>
<table class="table table-striped mt-5">
    <thead class="bg-primary text-light">
    <tr>
    <th scope="col">Account No</th>
    <th scope="col">Credit / Debit</th>
    <th scope="col">Transaction Details</th>
    <th scope="col">Date</th>
    <th scope="col">Time</th>
```

```
        <th scope="col">Amount</th>

        </tr>
        </thead>
        <tbody>
        <%
        User us = (User) session.getAttribute("userobj");
        UserDAOImpl dao = new UserDAOImpl(DbConnect.getConn());
        List<AccountTransaction> trans =
dao.getAllTransByCredit(us.getAccountNo());
        for (AccountTransaction tr : trans) {
        %>
<tr>
<th scope="row"><%=tr.getAccno()%></th>
<td><%=tr.getTransType()%></td>
<td><%=tr.getTransDtls()%></td>
<td><%=tr.getTransDate()%></td>
<td><%=tr.getTransTime()%></td>
<td><%=tr.getBalance()%></td>
        </tr>
        <%
        }
        %>
        </tbody>
        </table>

</div>
        </div>
        </div>

</body>
</html>

<%@page import="com.entity.AccountTransaction"%>
<%@page import="java.util.List"%>
<%@page import="com.db.DbConnect"%>
<%@page import="com.dao.UserDAOImpl"%>
<%@page import="com.entity.User"%>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Transaction Debit</title>
<%@include file="all_component/allCss_file.jsp"%>
</head>
<body>
    <%@include file="all_component/navbar.jsp"%>
    <c:if test="${empty userobj}">
        <c:set value="Please Login" var="failedmsg"
scope="session"></c:set>
        <c:redirect url="login.jsp"></c:redirect>
    </c:if>

    <div class="container-fluid">
        <div class="row">
            <div class="col-md-12">
                <h4 class="text-center">All Debit Transaction</h4>

                <hr>
                <table class="table table-striped mt-5">
                    <thead class="bg-primary text-light">
                        <tr>
                  <th scope="col">Account No</th>
                 <th scope="col">Credit / Debit</th>
                <th scope="col">Transaction Details</th>
                <th scope="col">Date</th>
                <th scope="col">Time</th>
                <th scope="col">Amount</th>

                </tr>
                </thead>
                <tbody>
```

```
                        <%
        User us = (User) session.getAttribute("userobj");
        UserDAOImpl dao = new UserDAOImpl(DbConnect.getConn());
List<AccountTransaction> trans =
dao.getAllTransByDebit(us.getAccountNo());
for (AccountTransaction tr : trans) {
%>
<tr>
<th scope="row"><%=tr.getAccno()%></th>
<td><%=tr.getTransType()%></td>
<td><%=tr.getTransDtls()%></td>
<td><%=tr.getTransDate()%></td>
<td><%=tr.getTransTime()%></td>
<td><%=tr.getBalance()%></td>
</tr>
<%
}
%>
</tbody>
</table>

</div>
</div>
        </div>

</body>
</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Bank: Forgot Password</title>
<%@include file="all_component/allCss_file.jsp"%>
```

```
</head>
<body style="background-color: #DCDCDC;">
	<%@include file="all_component/navbar.jsp"%>

	<div class="container-fluid">
		<div class="row p-2">
			<div class="col-md-4 offset-md-4">
				<div class="card">
					<div class="card-body">
						<form class="needs-validation"
novalidate method="post"
action="forgot">
<h3 class="text-center">Forgot
Password</h3>

<c:if test="${not empty failedmsg}">
<p class="text-center text-danger">${failedmsg}</p>
<c:remove var="failedmsg" scope="session" />
</c:if>

<c:if test="${not empty succmsg}">
<p class="text-center text-success">${succmsg}</p>
<c:remove var="succmsg" scope="session" />
</c:if>
<div class="form-group">
<label for="uname">Account No</label> <input type="number"
class="form-control" id="accno"name="accno" required>
</div>
<div class="form-group">
<label for="pwd">Username</label> <input type="text"class="form-
control" id="pwd"name="uname" required>
</div>
<div class="text-center">
<button type="submit" class="btn btn-primary btn-lg">Submit</button>
		</div>
	</form>

</div>
    </div>
```

```
            </div>
        </div>
        </div>

        <div style="margin-top: 60px;">
            <%@include file="all_component/footer.jsp"%>
        </div>
</body>
</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
<%@include file="all_component/allCss_file.jsp"%>
<style type="text/css">
a {
    text-decoration: none;
    color: black
}

a:hover {
    text-decoration: none;
}

.back-img {
    background: url("img/b3.jpg");
    width: 100%;
    height: 600px;
}
</style>

</head>
```

```
<body>
	<%@include file="all_component/navbar.jsp"%>
	<c:if test="${ empty userobj}">
		<c:set value="Please Login" scope="session"
var="failedmsg"></c:set>
		<c:redirect url="login.jsp" />
	</c:if>

	<div class="container-fluid text-center">
		<img src="img/b4.jpeg" style="width: 300px; height: 200px;">
		<h1 class="text-center text-primary">Welcome to My Bank</h1>
	</div>

	<div style="margin-top: 120px;">
		<%@include file="all_component/footer.jsp"%>
	</div>
</body>
</html>

%-- <%@page import="java.sql.Connection"%>
<%@page import="com.db.DbConnect"%> --%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
.carousel-item:after {
	content: "";
	display: block;
	position: absolute;
	top: 0;
	bottom: 0;
	left: 0;
	right: 0;
	background: rgba(0, 0, 0, 0.4);
}
```

```
.carousel-caption {
	color: #fff;
	top: 50%;
}
</style>
<%@include file="all_component/allCss_file.jsp"%>
<meta charset="ISO-8859-1">
<title>Bank: Home</title>
</head>
<body style="background-color: #f0f1f2;">
	<%@include file="all_component/navbar.jsp"%>
	<!-- <div class="container-fluid back-img"></div> -->

	<!-- <div id="carouselExampleControls" class="carousel slide"
		data-ride="carousel">

		<a class="carousel-control-prev"
href="#carouselExampleControls"
			role="button" data-slide="prev"> <span
			class="carousel-control-prev-icon" aria-
hidden="true"></span> <span
			class="sr-only">Previous</span>
		</a> <a class="carousel-control-next"
href="#carouselExampleControls"
			role="button" data-slide="next"> <span
			class="carousel-control-next-icon" aria-
hidden="true"></span> <span
			class="sr-only">Next</span>
		</a>
	</div> -->


	<div id="carouselExampleIndicators" class="carousel slide"
		data-ride="carousel">
		<ol class="carousel-indicators">
			<li data-target="#carouselExampleIndicators" data-slide-
to="0"
				class="active"></li>
```

```
                    <li data-target="#carouselExampleIndicators" data-slide-
to="1"></li>
                    <li data-target="#carouselExampleIndicators" data-slide-
to="2"></li>
                </ol>
                <div class="carousel-inner">
                    <div class="carousel-item active">
                        <div class="carousel-caption">
                            <h1 class=" text-white">
                                Welcome to <br>Banking Management
System
                            </h1>
                        </div>
                        <img class="d-block w-100" src="img/p2.jpg"
alt="First slide"
                            style="width: 100%; height: 350px;">
                    </div>
                    <div class="carousel-item">
                        <img class="d-block w-100" src="img/p3.jpg"
alt="Second slide"
                            style="width: 100%; height: 350px;">
                    </div>
                    <div class="carousel-item">
                        <img class="d-block w-100" src="img/b3.jpg"
alt="Third slide"
                            style="width: 100%; height: 350px;">
                    </div>
                </div>
                <a class="carousel-control-prev"
href="#carouselExampleIndicators"
                    role="button" data-slide="prev"> <span
                    class="carousel-control-prev-icon" aria-
hidden="true"></span> <span
                    class="sr-only">Previous</span>
                </a> <a class="carousel-control-next"
href="#carouselExampleIndicators"
                    role="button" data-slide="next"> <span
```

```
                    class="carousel-control-next-icon" aria-
hidden="true"></span> <span
                    class="sr-only">Next</span>
            </a>
      </div>


      <div class="container p-3">
            <div class="row">
                  <div class="col-md-3">
                        <div class="card" style="height: 230px;">
                              <div class="card-body ">
                                    <h5 class="text-center">Banking</h5>
<ul>
<li>Savings and Deposits</li>
<li>Consultancy Services</li>
<li>Accounts and Deposits</li>
<li>Card Services</li>
<li>Digital Products</li>
<li>Online Banking</li>
</ul>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card" style="height: 230px;">
<div class="card-body ">
<h5 class="text-center">Credit Facilites</h5>
<ul>
<li>Loans & Advances</li>
<li>Apply for Retail Loans</li>
<li>DRI Loans</li>
<li>Educational Loan</li>
</ul>
</div>
</div>
</div>
<div class="col-md-3">
```

```
<div class="card" style="height: 230px;">
<div class="card-body ">
<h5 class="text-center">Invest and Insurance</h5>
<ul>
<li>Life Insurance</li>
<li>Health Insurance</li>
<li>General Insurance</li>
<li>Mutual Funds</li>
<li>Depository Services</li>
</ul>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card" style="height: 230px;">
<div class="card-body ">
<h5 class="text-center">Offers</h5>
<ul>
<li>Emergency Credit Line</li>
<li>Guarantee Scheme</li>
</ul>
</div>
</div>
</div>
</div>
</div>

<%@include file="all_component/footer.jsp"%>
</body>
</html>
```

Output code:

```
package com.admin.servlet;

import java.io.IOException;

import java.time.LocalDate;
```

```
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.dao.AdminDAOImpl;
import com.dao.UserDAOImpl;
import com.db.DbConnect;
import com.entity.AccountTransaction;

@WebServlet("/add_transaction")
public class AddTransaction extends HttpServlet {
	@Override
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
		try {
			String accno = req.getParameter("accno");
			Double amt =
Double.parseDouble(req.getParameter("amt"));
```

```
                String transType = req.getParameter("transtype");
                UserDAOImpl dao = new
UserDAOImpl(DbConnect.getConn());
                Double totalBal = dao.checkBalance(accno);
                AdminDAOImpl dao2 = new
AdminDAOImpl(DbConnect.getConn());
                AccountTransaction trans = new AccountTransaction();
                trans.setAccno(accno);
                trans.setTransType(transType);
                trans.setBalance(amt);
                trans.setTransDate(LocalDate.now().toString());
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH.mm");
                trans.setTransTime(LocalTime.now().format(formatter));
                HttpSession session = req.getSession();
                if ("Credit".equals(transType)) {
                        totalBal += amt;
                        boolean f = dao2.updateBalance(accno, totalBal);
                        trans.setTransDtls("Cash Deposit");
                        dao2.addTransaction(trans);

                        session.setAttribute("msg", "Cash Deposit
Sucessfully");
```

```
                    resp.sendRedirect("admin/new_transaction.jsp");
                } else if ("Debit".equals(transType)) {
                    totalBal -= amt;
                    dao2.updateBalance(accno, totalBal);
                    trans.setTransDtls("Cash withdraw from bank");
                    dao2.addTransaction(trans);
                    session.setAttribute("msg", "Cash Withdraw
Sucessfully");
                    resp.sendRedirect("admin/new_transaction.jsp");
                } else {
                    session.setAttribute("failedMsg", "Select transaction
type..");
                    resp.sendRedirect("admin/new_transaction.jsp");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
package com.admin.servlet;
import java.io.IOException;
import java.time.*;
```

```
import java.time.format.DateTimeFormatter;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.dao.AdminDAOImpl;
import com.db.DbConnect;
import com.entity.AccountTransaction;
@WebServlet("/admin/approve")
public class ApproveStatus extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        try {
            String accno = req.getParameter("accno");
            String email = req.getParameter("email");
            String name = req.getParameter("uname");
            AccountTransaction actrns = new AccountTransaction();
            actrns.setAccno(accno);
            actrns.setTransType("credit");
            actrns.setTransDtls("Account Opening");
            actrns.setBalance(0.00);
            actrns.setTransDate(LocalDate.now().toString());
            DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH.mm");
            actrns.setTransTime(LocalTime.now().format(formatter));
            String msg = "Hello " + name + ",\n Your Account Open
Sucssfully. Your Account No:" + accno
                    + ".\n Now You Can Create Netbanking.";
            String sub = "Account Opening Confirmation";
            String from = "19CS801038@kristujayanti.com";
            AdminDAOImpl dao = new
AdminDAOImpl(DbConnect.getConn());
```

```
                boolean f = dao.updateStatus(accno);
                HttpSession session = req.getSession();
                if (f) {
                        dao.setBalance(accno, 0.00);
                        dao.addTransaction(actrns);
                         SendEmail(msg, sub, email, from);
                        session.setAttribute("sussMsg", "Account Opening
Sucessfully");
                        resp.sendRedirect("acc_status.jsp");
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
    }
    private static void SendEmail(String msg, String subject, String to,
String from) {
        // TODO Auto-generated method stub
        // variable gmail host
        String host = "smtp.gmail.com";
        // get system properties
        Properties properties = System.getProperties();
```

```
        // setting important information to properties object
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");
        // step-1 get session
        Session session = Session.getInstance(properties, new
Authenticator() {
            @Override
            protected PasswordAuthentication
getPasswordAuthentication() {
                // TODO Auto-generated method stub
                return new
PasswordAuthentication("19CS801038@kristujayanti.com",
"19CS801038@kristujayanti.com");
            }
        });
        // step-2 compose msg
        MimeMessage m = new MimeMessage(session);
        try {
            // from email
            m.setFrom(from);
            // add reciepnt
```

```
                m.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));


                // add subject
                m.setSubject(subject);
                m.setText(msg);


                // step-3 send messgae using transport class
                Transport.send(m);


            } catch (MessagingException e) {
                e.printStackTrace();
            }
        }
}
package com.admin.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import com.dao.AdminDAOImpl;
import com.db.DbConnect;
```

```
import com.entity.User;
@WebServlet("/create_accnt")
public class CreateAccnt extends HttpServlet {
	@Override
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
		try {
			String fn = req.getParameter("fn");
			String ln = req.getParameter("ln");
			String em = req.getParameter("em");
			String ph = req.getParameter("ph");
			String dob = req.getParameter("dob");
			String adh = req.getParameter("adh");
			String add = req.getParameter("add");
			String city = req.getParameter("city");
			String st = req.getParameter("st");
			String zip = req.getParameter("zip");
			String check = req.getParameter("check");
			User us = new User(fn, ln, em, ph, dob, adh, add, city, st,
zip);
			HttpSession session = req.getSession();;
			if (check == null) {
```

```
                    session.setAttribute("ErrorMsg", "Please check Agree
Terms & Condition");
                    resp.sendRedirect("admin/open_acc.jsp");
                    } else {
                        AdminDAOImpl dao = new
AdminDAOImpl(DbConnect.getConn());
                        boolean f = dao.createAccnt(us);
                        if (f) {
                            session = req.getSession();
                            session.setAttribute("SucessMsg", "Account
Open Sucessfully..");
                             resp.sendRedirect("admin/open_acc.jsp");
                        }else {
                    session.setAttribute("ErrorMsg", "Something wrong on
Server.");
                    resp.sendRedirect("admin/open_acc.jsp");
                        }
                    }
                } catch (Exception e) {
                e.printStackTrace();
                }
        }
}
```

```
package com.admin.servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.dao.AdminDAOImpl;

import com.db.DbConnect;

import com.entity.User;

@WebServlet("/update")

public class EditServlet extends HttpServlet {

	@Override

	protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {

		try {

			String accno=req.getParameter("accno");

			String uname = req.getParameter("uname");

			String fn = req.getParameter("fname");

			String ln = req.getParameter("lname");

			String em = req.getParameter("email");
```

```
String ph = req.getParameter("phno");
String dob = req.getParameter("dob");
String adh = req.getParameter("adhno");
String add = req.getParameter("address");
String city = req.getParameter("city");
String st = req.getParameter("state");
String status = req.getParameter("sts");
User us=new User();
us.setAccountNo(accno);
us.setUserid(uname);
us.setFirstName(fn);
us.setLastName(ln);
us.setEmail(em);
us.setNumber(ph);
us.setDob(dob);
us.setAdhar(adh);
us.setAddress(add);
us.setCity(city);
us.setState(st);
us.setStatus(status);
AdminDAOImpl dao=new
AdminDAOImpl(DbConnect.getConn());
```

```
                boolean f=dao.editUserAccount(us);
                HttpSession session=req.getSession();
                if(f)
                {
                        session.setAttribute("succmsg", "Account update
Sucessfully");
                        resp.sendRedirect("admin/all_user.jsp");
                }else {
                        session.setAttribute("failedmsg", "Something wrong
on server ");
                        resp.sendRedirect("admin/all_user.jsp");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
package com.admin.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.dao.AdminDAOImpl;
import com.dao.UserDAOImpl;
import com.db.DbConnect;
import com.entity.User;
@WebServlet("/edit_view")
public class EditViewServlet extends HttpServlet {
	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
		try {
			String accno = req.getParameter("accno");
			AdminDAOImpl dao = new
AdminDAOImpl(DbConnect.getConn());
			User us = dao.getUserByAccont(accno);
			HttpSession session = req.getSession();
			if (us != null) {
				session.setAttribute("user", us);
				resp.sendRedirect("admin/edit_acc.jsp");
			}
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

7. TESTING

Software Testing-Levels:

Black-Box Testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White-Box Testing:

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Unit Testing:

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the

test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Integration Testing:

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

System Testing:

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

8. IMPLEMENTATION

Our experience in implementing the project "E-Banking System" started up with collecting requirements regarding the complete flow on how an e-banking system is actually been done, details about the workers and their interaction with the customers, how the payment can be done online etc. Soon after collecting the requirements that is needed for a fully pledged project and on how to implement our project, then we started implementing the Admin side and then moved on to the user side we created the UI and then started with the coding part and then linked it to the back end using Workbench database.

Implementing "E-Banking System" would not be possible without a developing tool for providing a user interface for the user to interact as well as storing the inputted data from the user. The tools which are used for

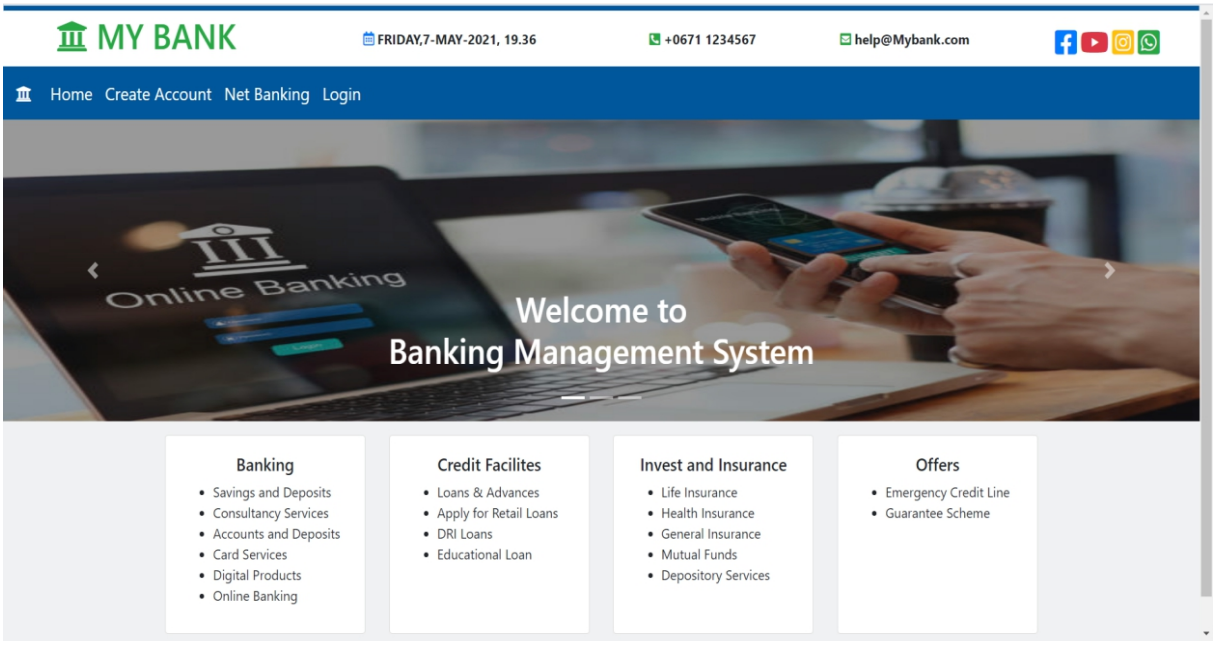
developing the “E-Banking System” are Eclipse IDE for Enterprise Java Developer(using HTML, JAVASCRIPT, CSS) as a front-end and Workbench Database as a back end.

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage the main work load, the greatest upheaval and the major impact on the system shifts to the user department. If the implementation is not carefully planned and controlled, it can cause confusion.

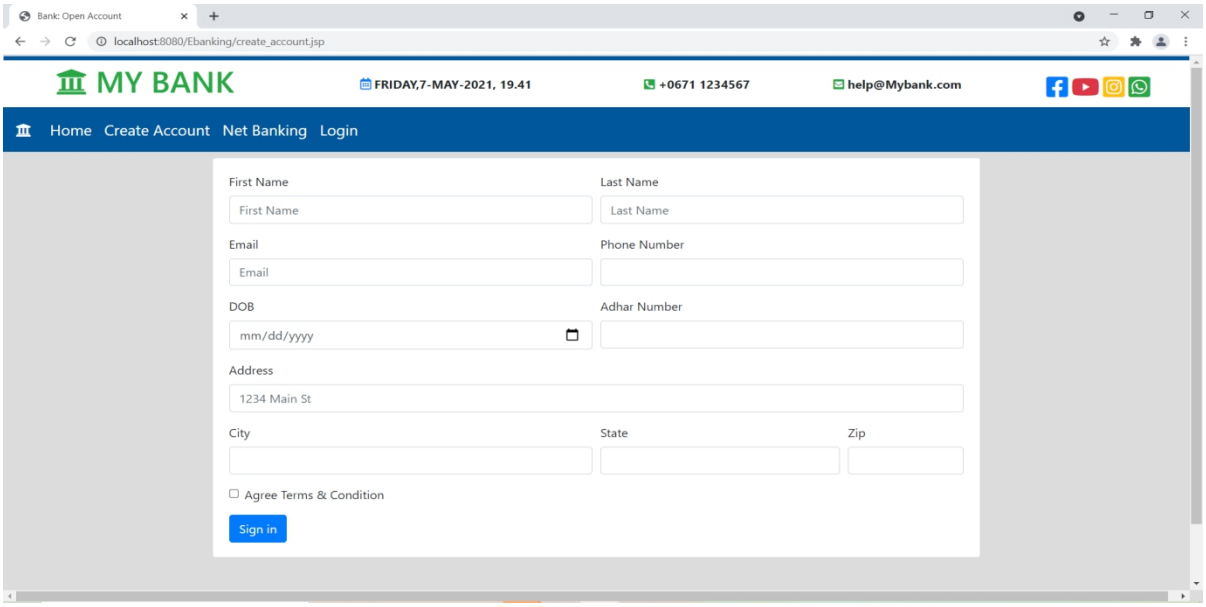
Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the company requirement. Successful implementation may not guarantee improvement in the company using the new system, but improper installation will prevent it.

9.SCREENSHOTS

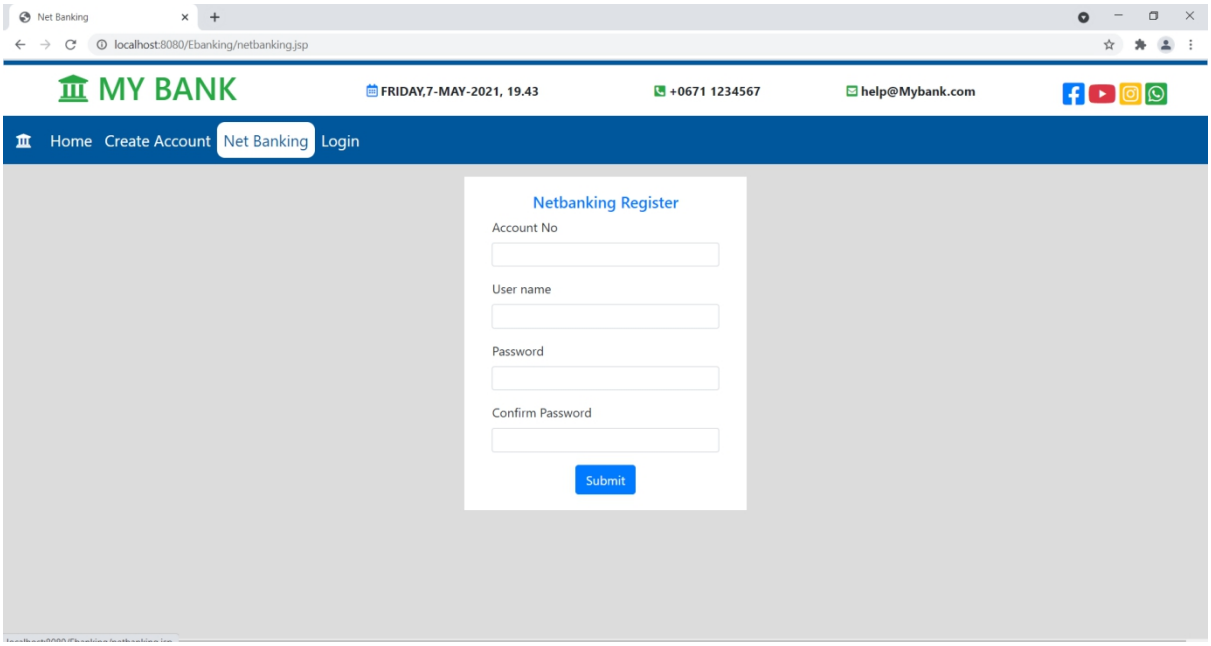
Home page:



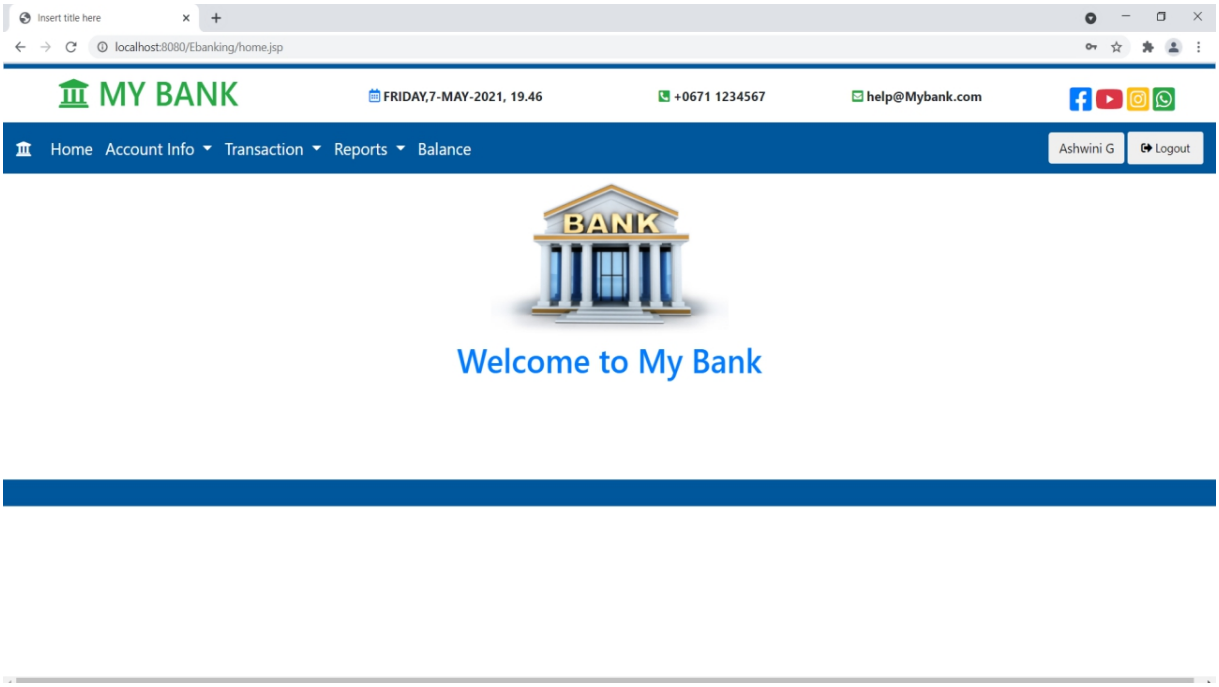
Create account:



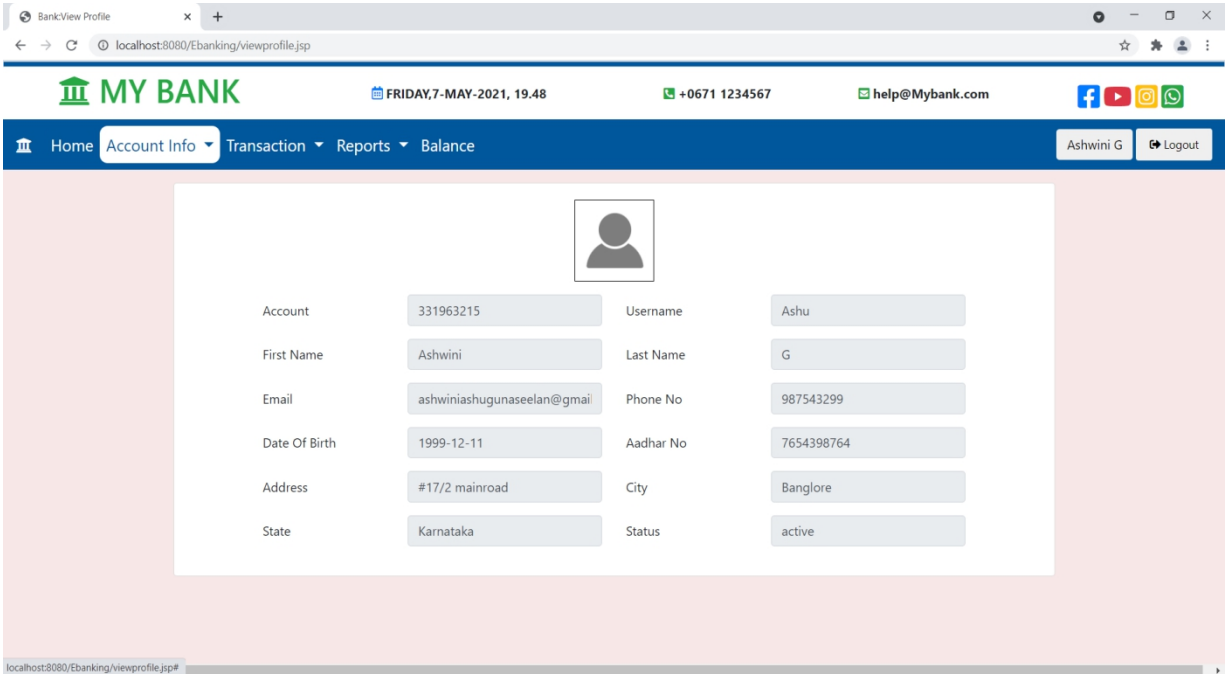
Net banking:



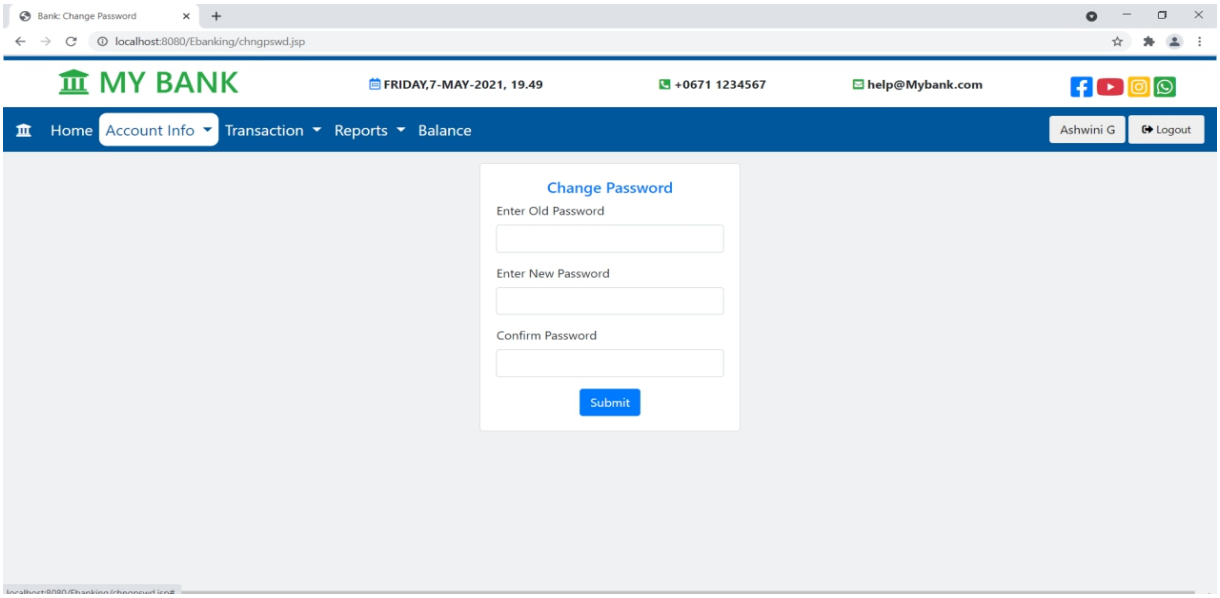
Login for use:



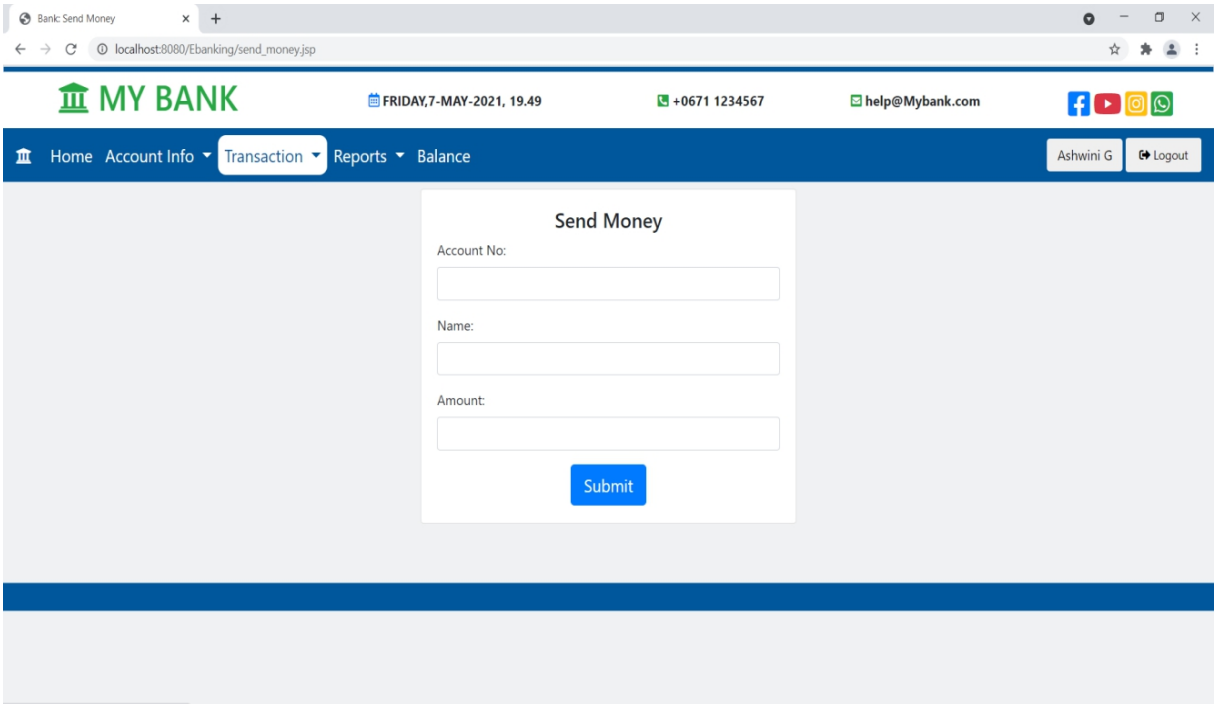
Account info:



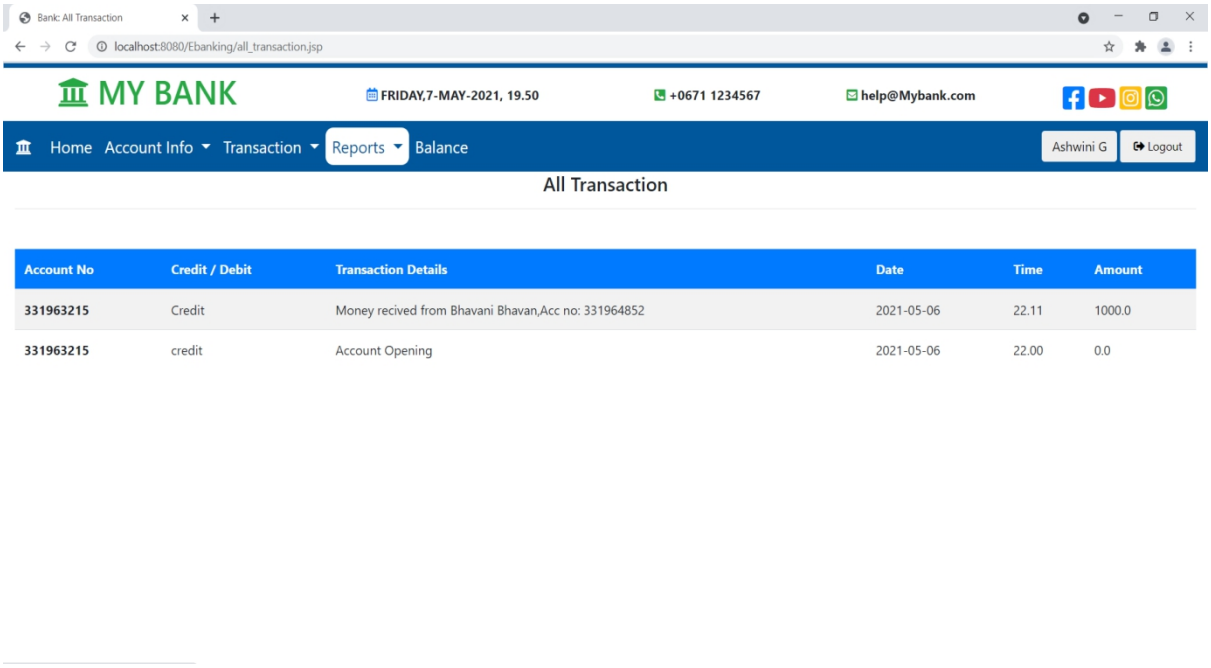
Change password:



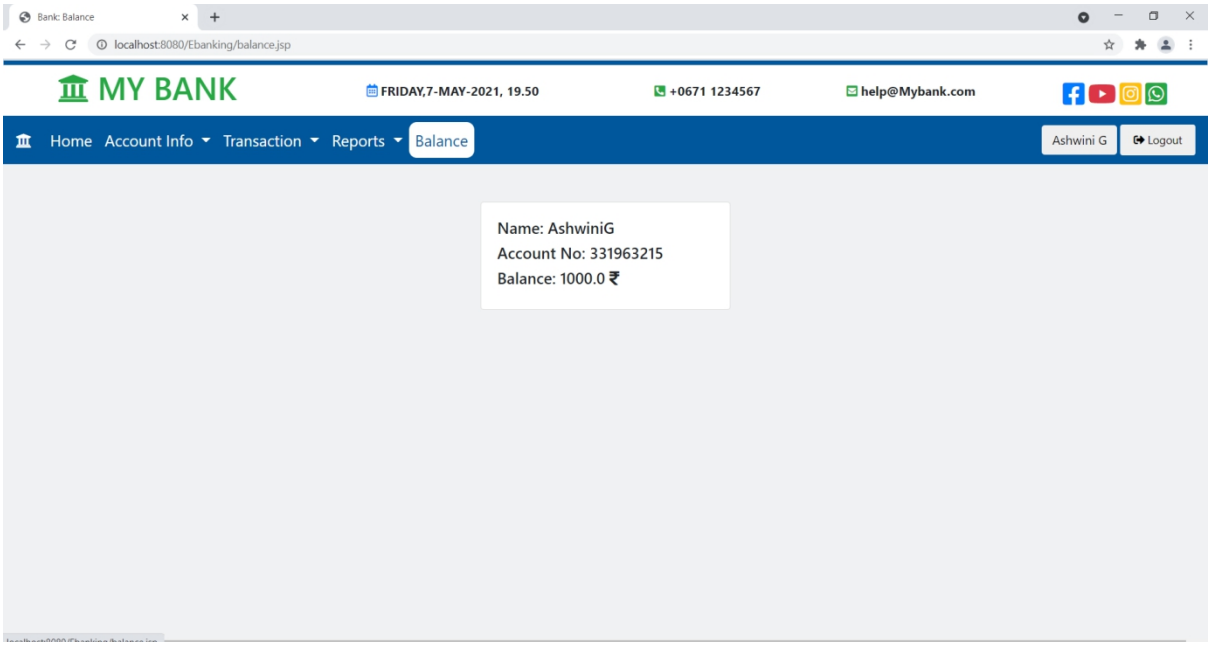
Transaction:



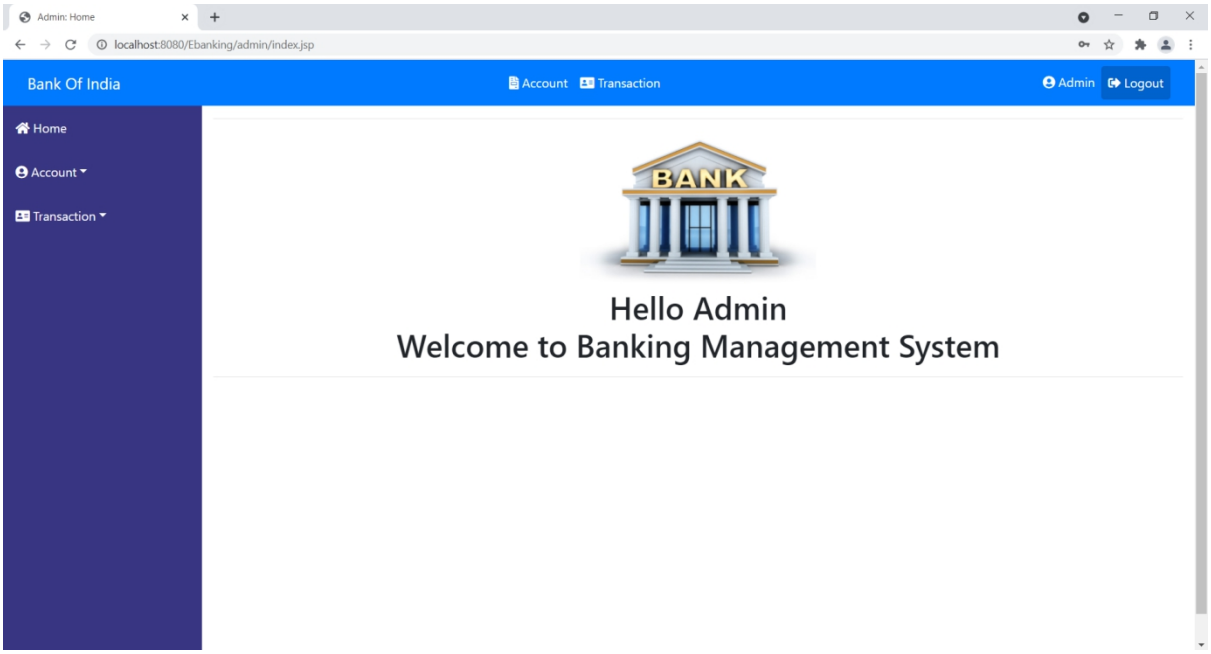
Report:



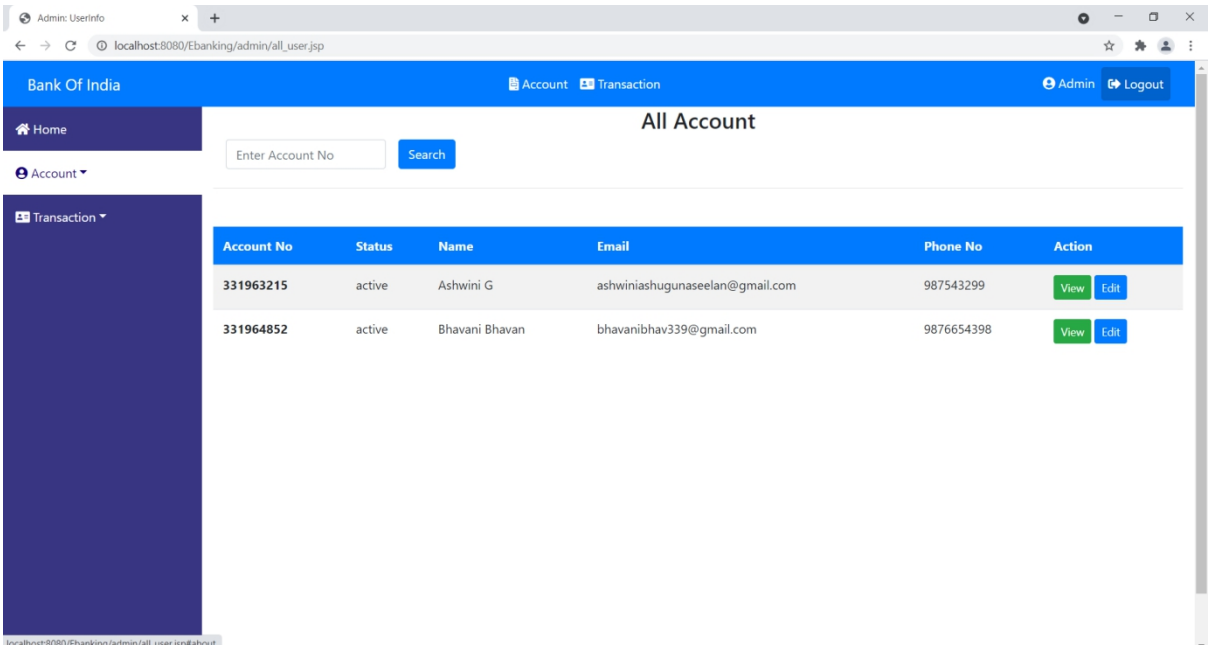
Balance:



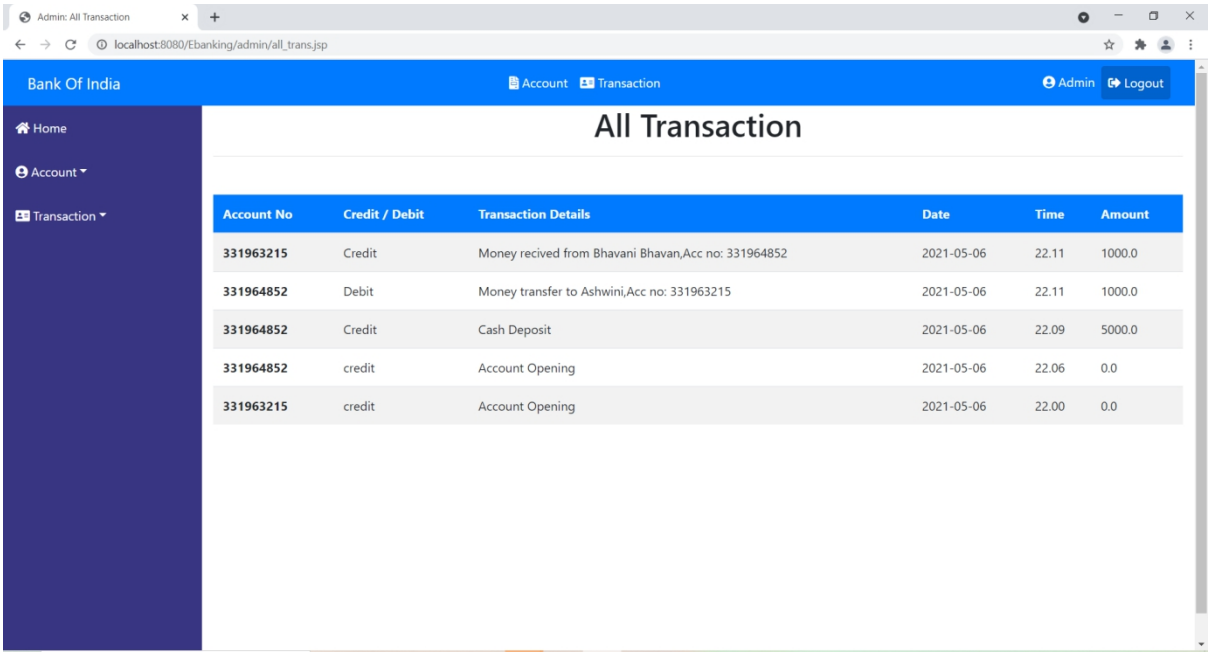
Admin login:



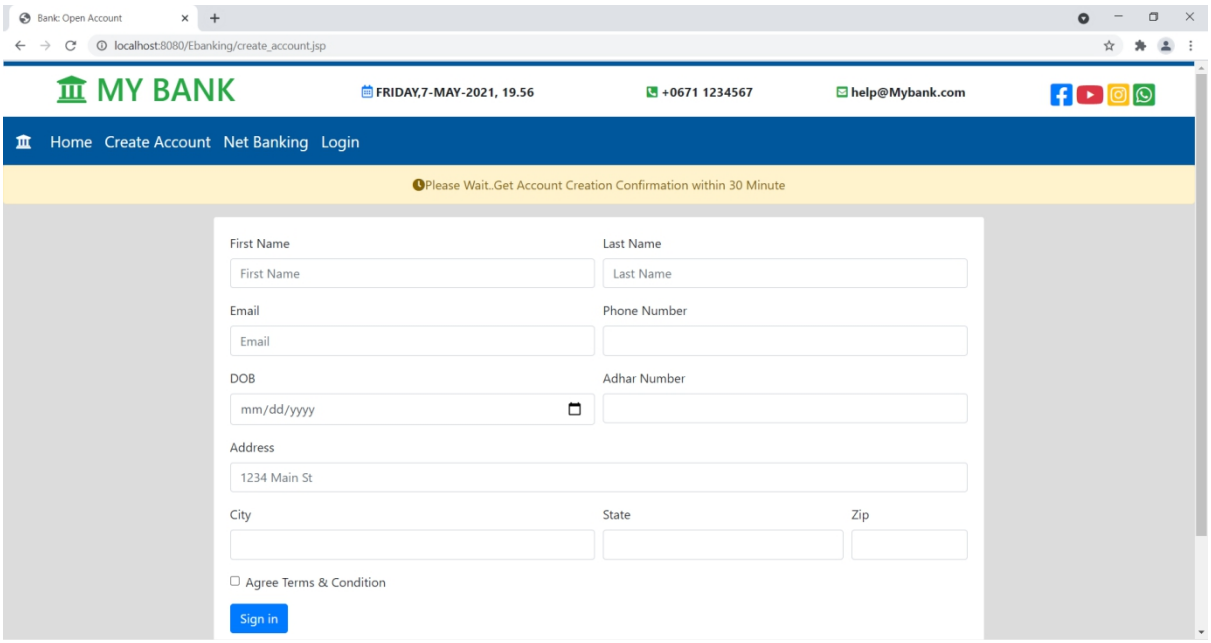
Account details:



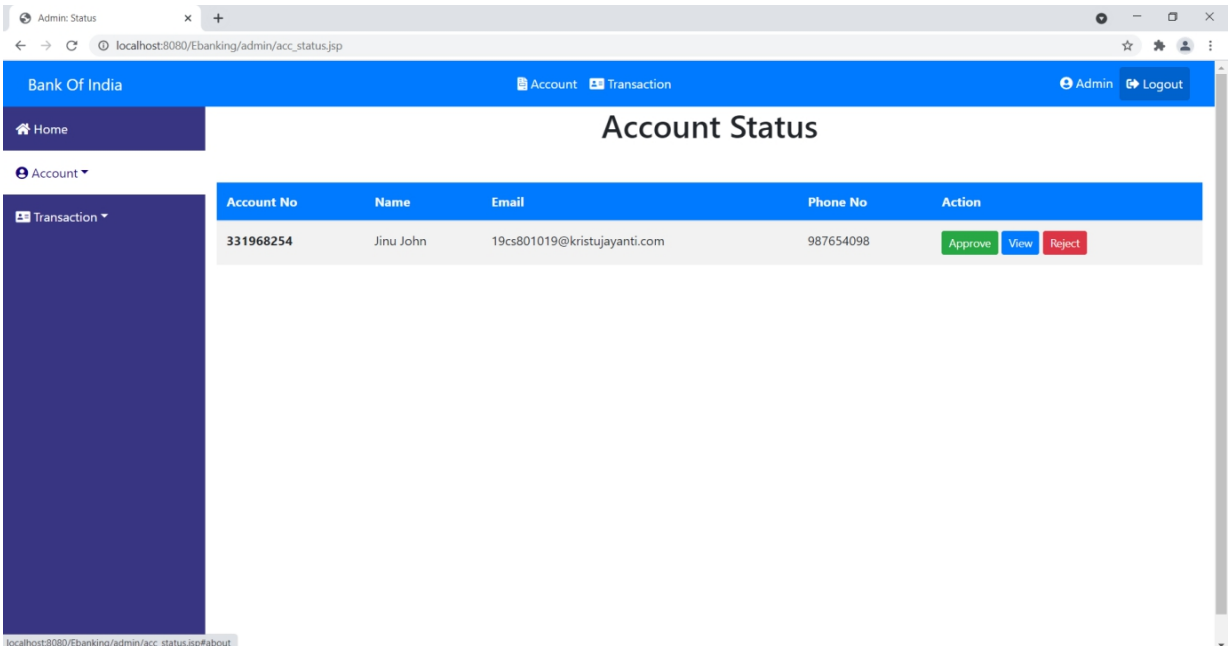
Transaction:

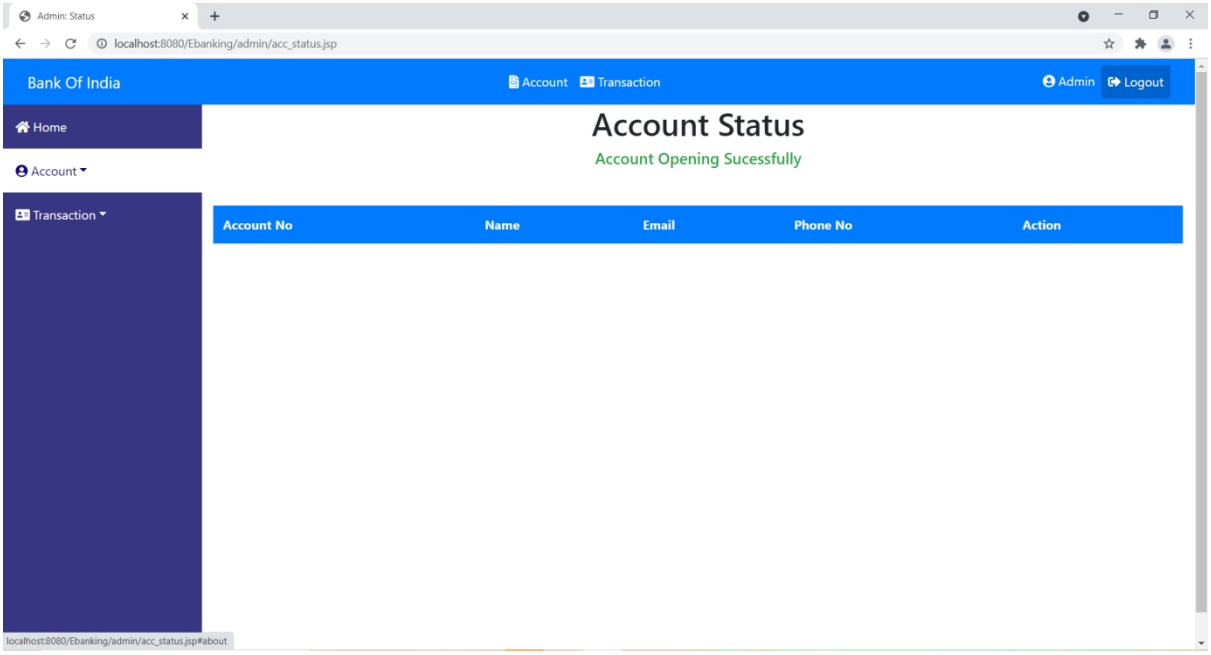


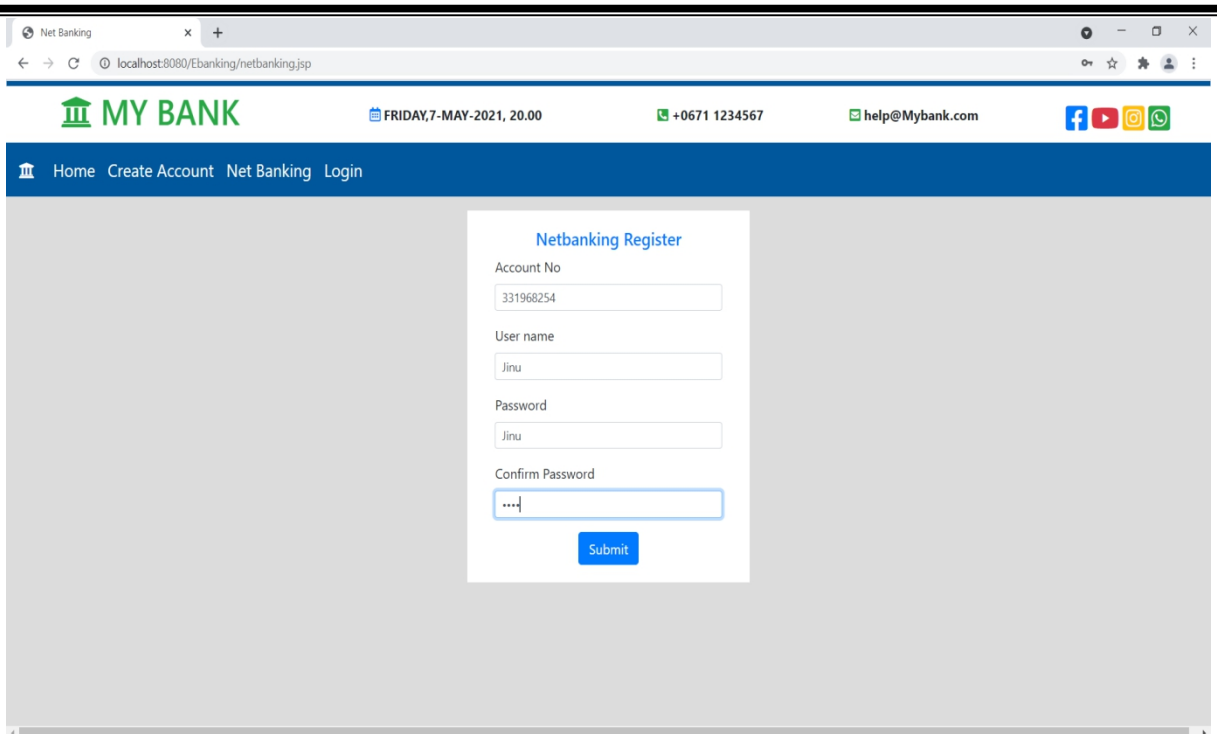
Output:

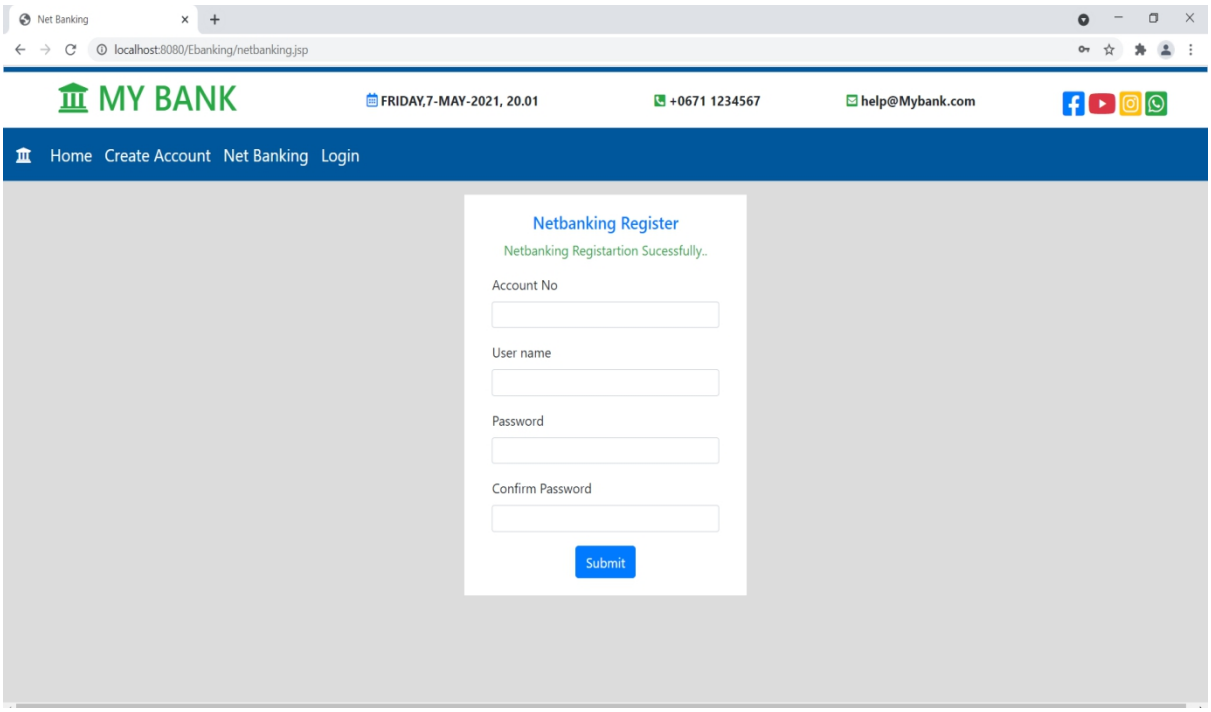


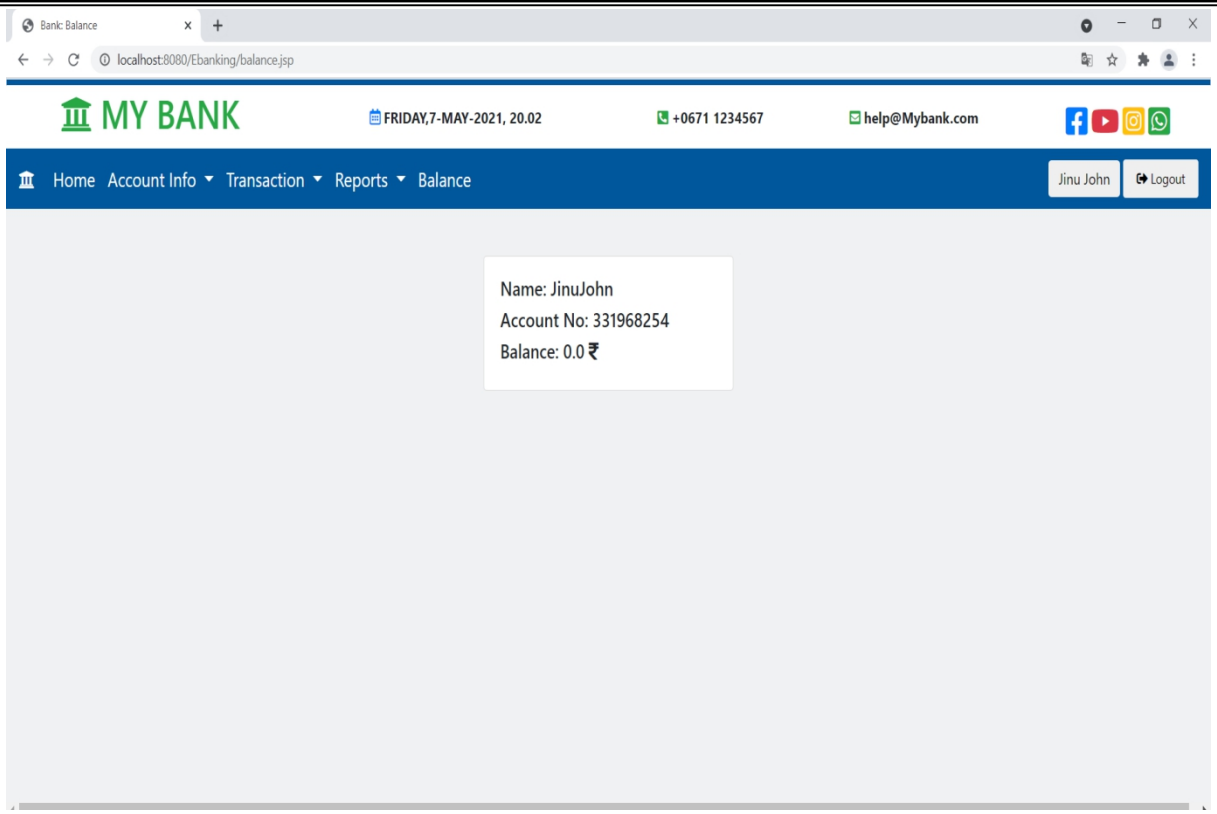
Account status:

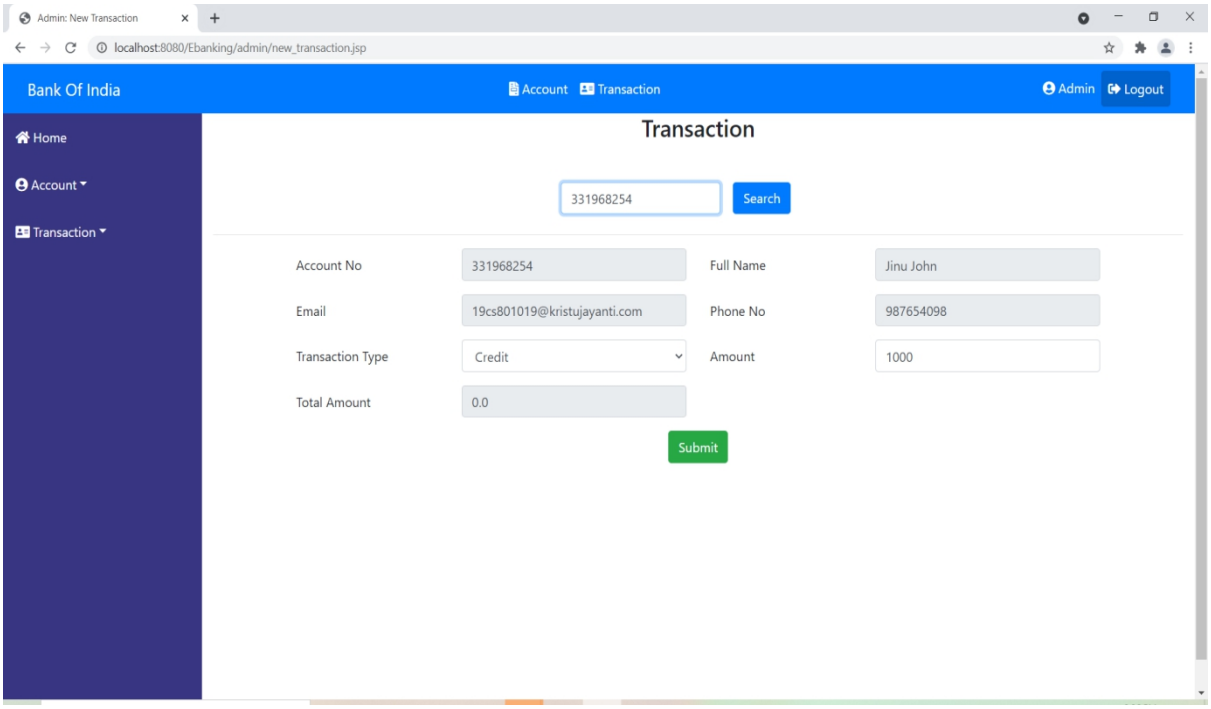


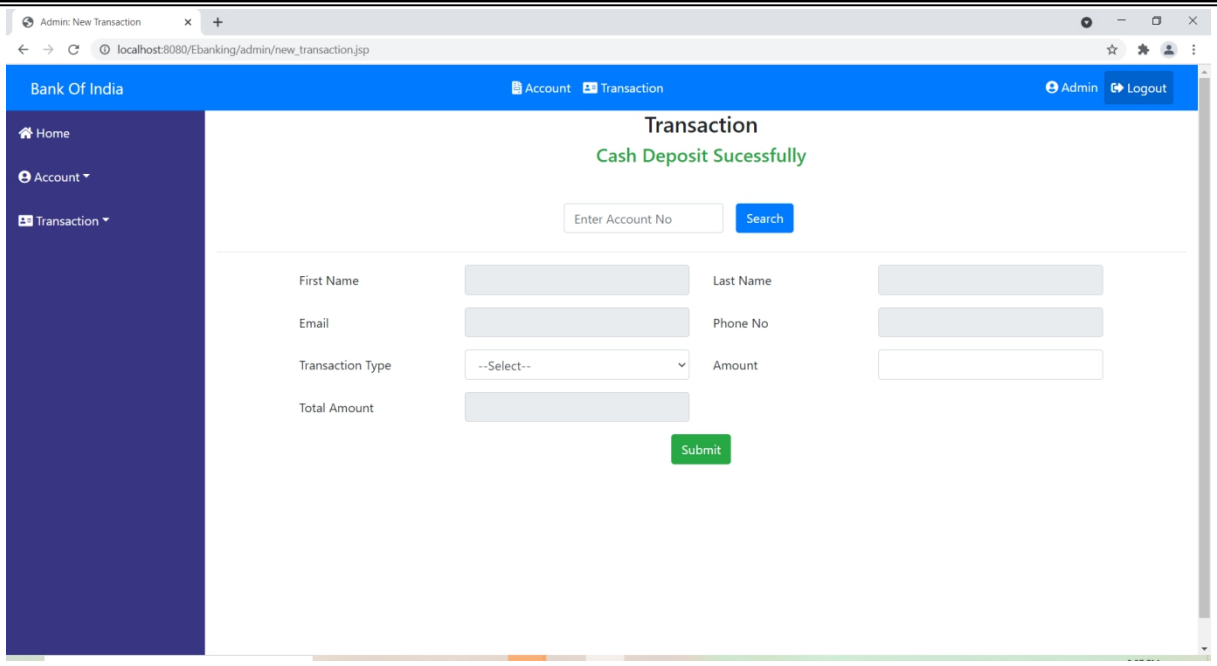


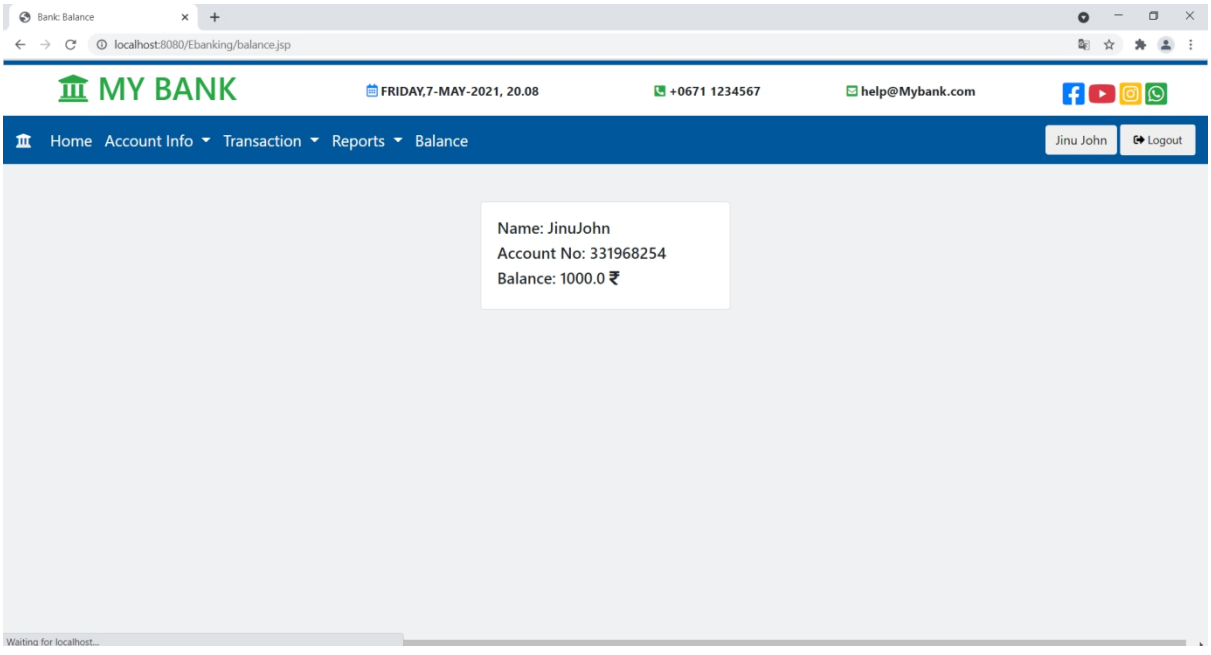

Net Banking
localhost:8080/Ebanking/netbanking.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.00
+0671 1234567
help@Mybank.com
Home Create Account Net Banking Login
Netbanking Register
Account No
331968254
User name
Jinu
Password
Jinu
Confirm Password
Submit

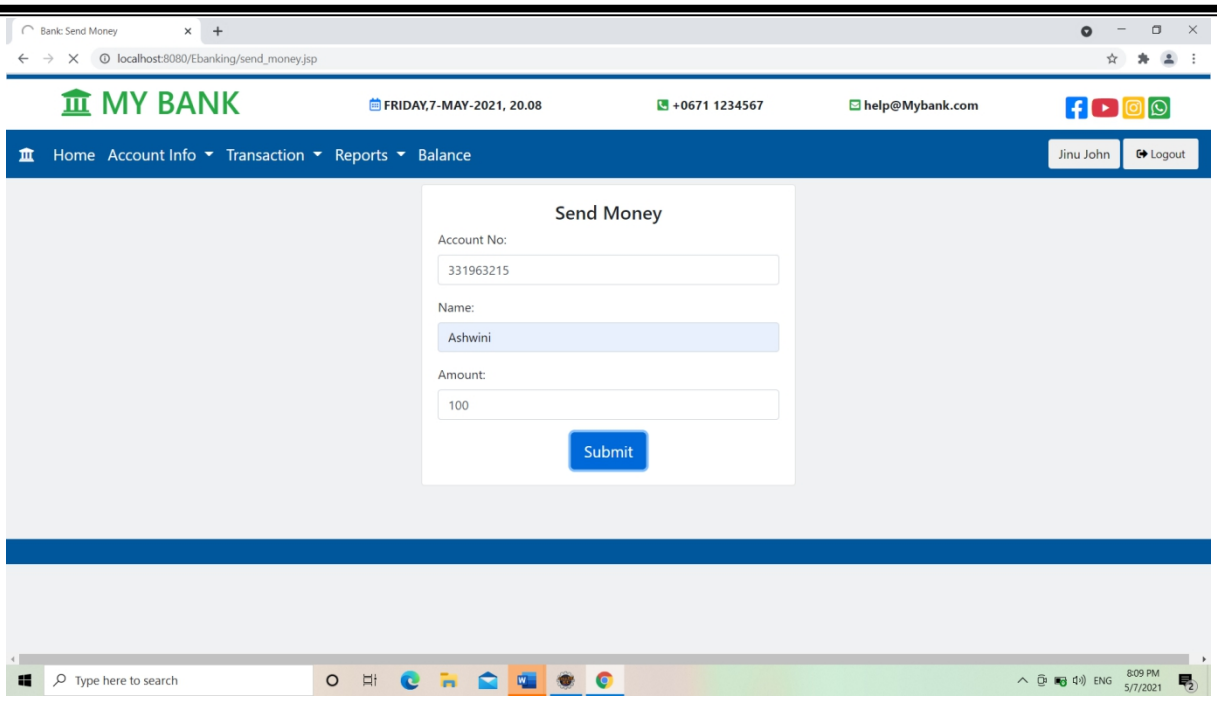

Net Banking
localhost:8080/Ebanking/netbanking.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.01
+0671 1234567
help@Mybank.com
Home Create Account Net Banking Login
Netbanking Register
Netbanking Registartion Sucessfully..
Account No
User name
Password
Confirm Password
Submit

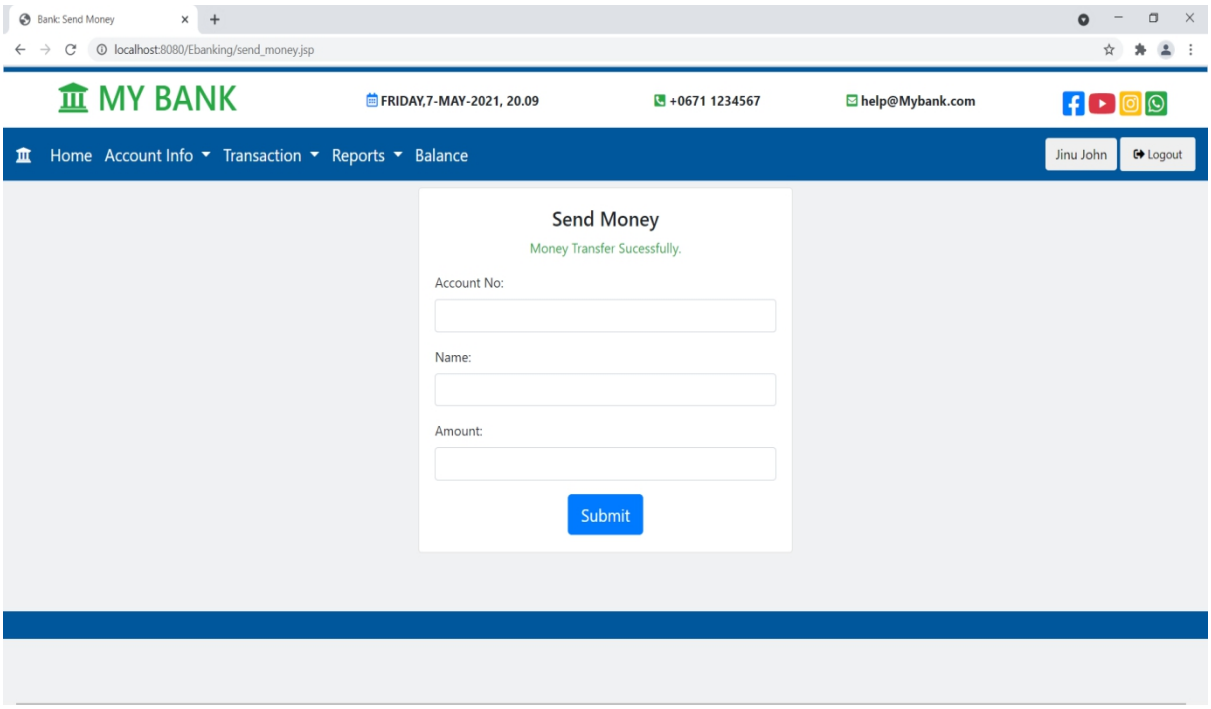

Bank: Balance
localhost:8080/Ebanking/balance.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.02
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
Name: JinuJohn
Account No: 331968254
Balance: 0.0 ₹

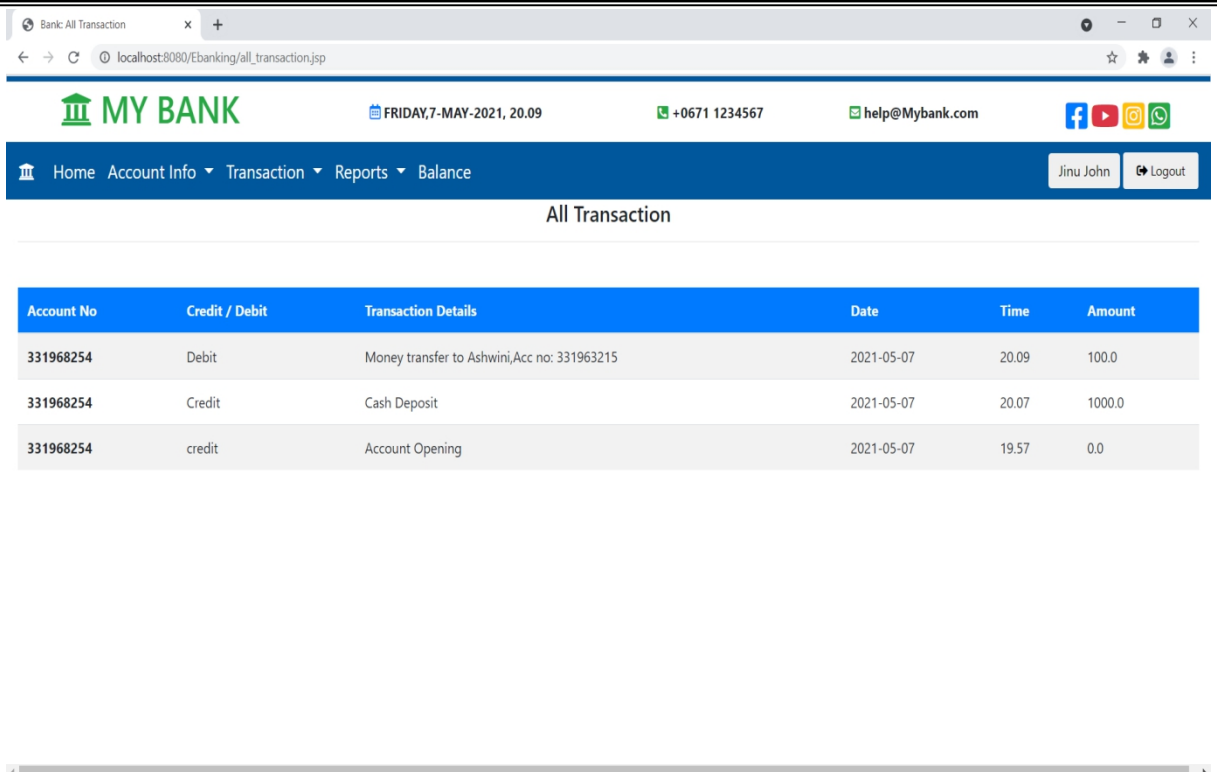

Admin: New Transaction
localhost:8080/Ebanking/admin/new_transaction.jsp
Bank Of India
Account
Transaction
Admin
Logout
Home
Account
Transaction
Transaction
331968254
Search
Account No
331968254
Full Name
Jinu John
Email
19cs801019@kristujayanti.com
Phone No
987654098
Transaction Type
Credit
Amount
1000
Total Amount
0.0
Submit


Admin: New Transaction
localhost:8080/Ebanking/admin/new_transaction.jsp
Bank Of India
Account
Transaction
Admin
Logout
Home
Account
Transaction
Transaction
Cash Deposit Sucessfully
Enter Account No
Search
First Name
Last Name
Email
Phone No
Transaction Type
--Select--
Amount
Total Amount
Submit

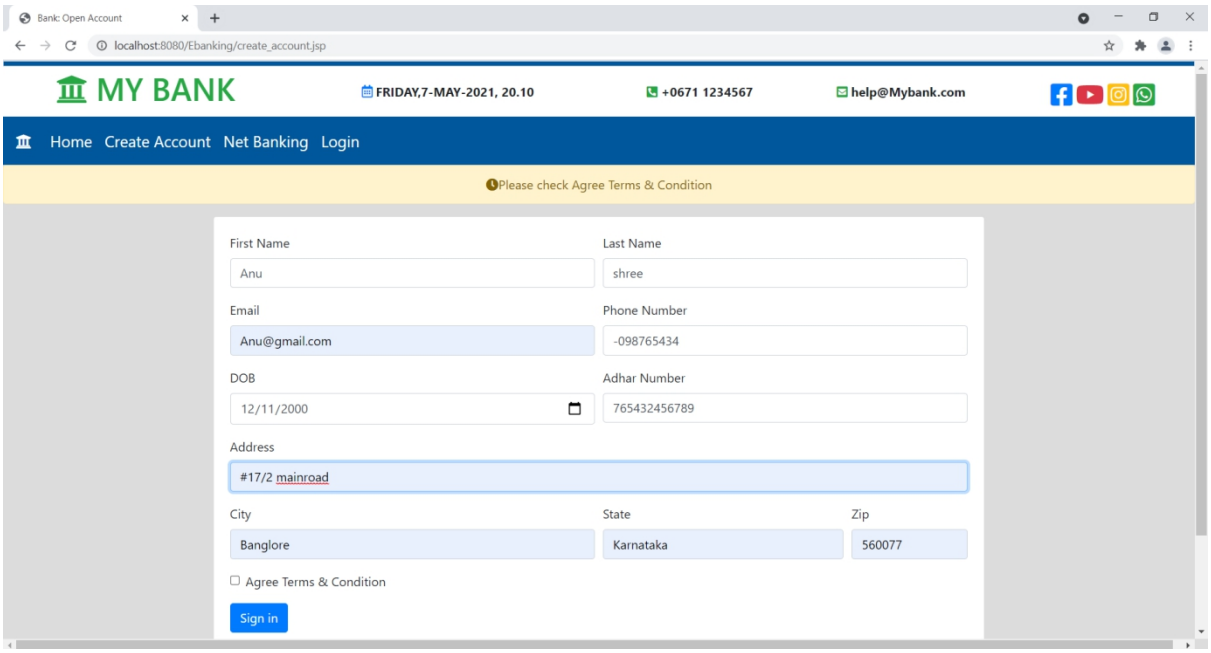

Bank: Balance
localhost:8080/Ebanking/balance.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.08
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
Name: JinuJohn
Account No: 331968254
Balance: 1000.0 ₹
Waiting for localhost...

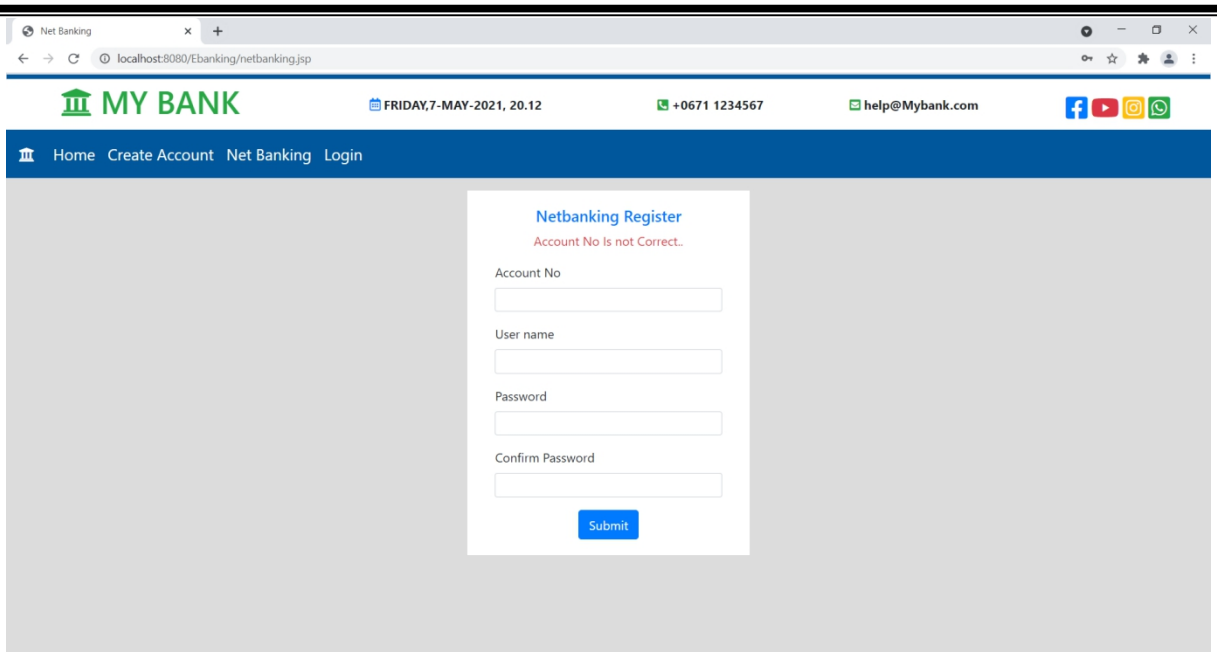

Bank: Send Money
localhost:8080/Ebanking/send_money.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.08
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
Send Money
Account No:
331963215
Name:
Ashwini
Amount:
100
Submit
Type here to search
ENG
8:09 PM
5/7/2021

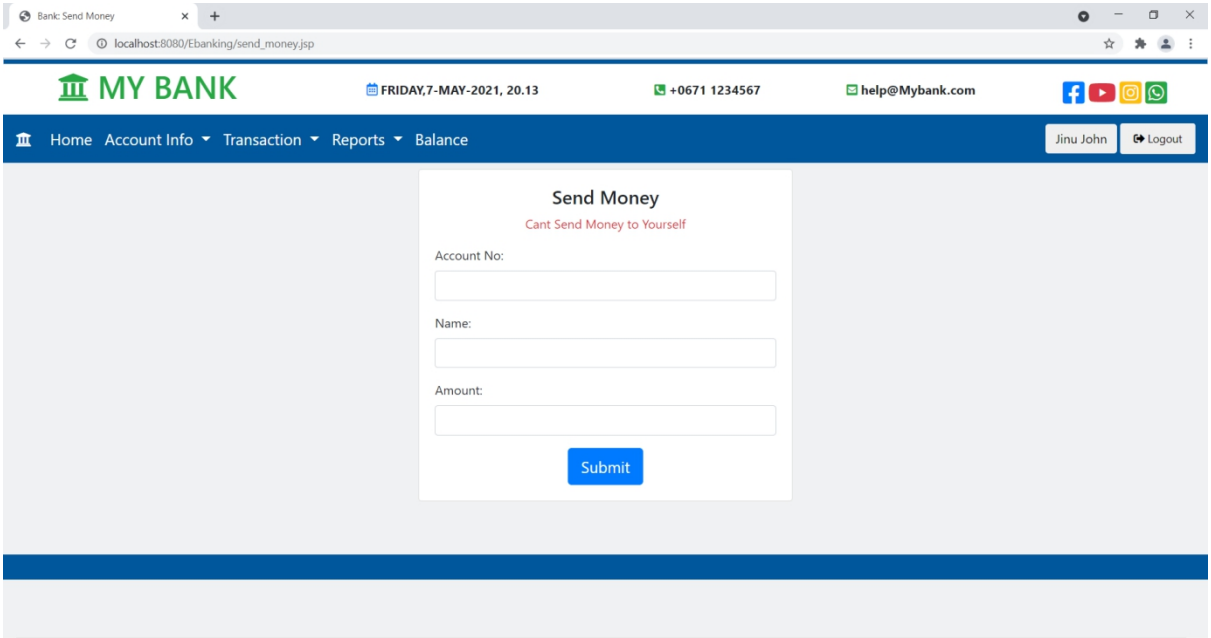

Bank: Send Money
localhost:8080/Ebanking/send_money.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.09
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
Send Money
Money Transfer Sucessfully.
Account No:
Name:
Amount:
Submit

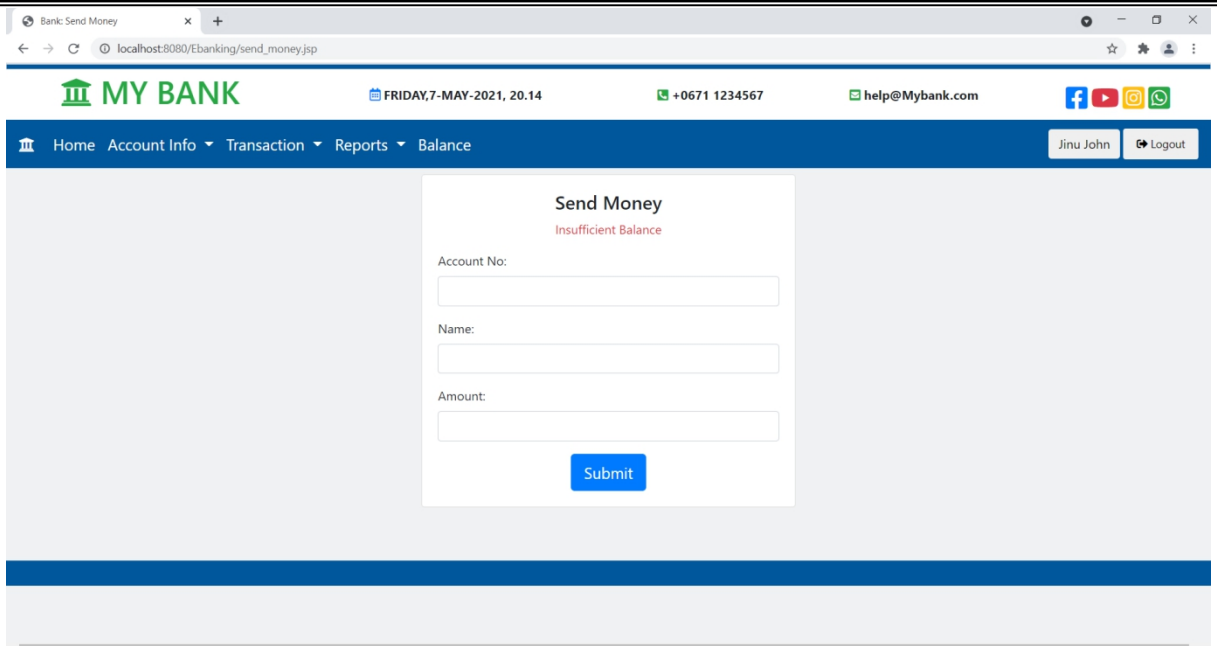

Bank: All Transaction
localhost:8080/Ebanking/all_transaction.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.09
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
All Transaction
Account No
Credit / Debit
Transaction Details
Date
Time
Amount
331968254
Debit
Money transfer to Ashwini,Acc no: 331963215
2021-05-07
20.09
100.0
331968254
Credit
Cash Deposit
2021-05-07
20.07
1000.0
331968254
credit
Account Opening
2021-05-07
19.57
0.0

Validation:


Bank: Open Account
localhost:8080/Ebanking/create_account.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.10
+0671 1234567
help@Mybank.com
Home
Create Account
Net Banking
Login
Please check Agree Terms & Condition
First Name
Anu
Last Name
shree
Email
Anu@gmail.com
Phone Number
-098765434
DOB
12/11/2000
Adhar Number
765432456789
Address
#17/2 mainroad
City
Banglore
State
Karnataka
Zip
560077
Agree Terms & Condition
Sign in


Net Banking
localhost:8080/Ebanking/netbanking.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.12
+0671 1234567
help@Mybank.com
Home
Create Account
Net Banking
Login
Netbanking Register
Account No Is not Correct..
Account No
User name
Password
Confirm Password
Submit


Bank: Send Money
localhost:8080/Ebanking/send_money.jsp
MY BANK
FRIDAY,7-MAY-2021, 20.13
+0671 1234567
help@Mybank.com
Home
Account Info
Transaction
Reports
Balance
Jinu John
Logout
Send Money
Cant Send Money to Yourself
Account No:
Name:
Amount:
Submit



10. CONCLUSION

The project entitled "E-Banking System" was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. we started implementing the Admin side and then moved on to the user side, we created the User interface and then started with the coding part and then linked it to the back end using Workbench database. The purpose of this project is enabling the customer to perform the basic transactions through their PC or laptop. The customer can access the bank's website for viewing their account details and perform their transactions on account as per their requirements. With E-Banking, the brick and the mortar structure of the traditional banking gets converted into click and portal model.

In abstract, this system is perceived to the user as a reliable and user-friendly manner.

11.BIBLIOGRAPHY

BOOKS/REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, "*System Analysis and Design*", 2009.
- Pankaj Jalote, "So*ftware Engineering*: a precise approach", 2006.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.
- Software Engineering: A Practitioner's Approach by Roger S. Pressman

WEBSITES:

- **HTML Design tutorial:** https://www.w3schools.com/html/
- **CSS Design tutorial :** https://www.w3schools.com/css/
- **JAVASCRIPT:** https://www.w3schools.com/js/