INDEX

S.No	Date	List of Experiments	Page No	Signature of Faculty In-Charge
1.		Create distributed applications using RMI		
2.		Create applications which can demonstrate the use of JDBC for Database Connectivity		
3.		Create Student applications using JDBC Database Connectivity		
4.		Develop web applications using Servlet.		
5.		Develop web applications using Servlet Request , Servlet Response		
6.		Program that demonstrates the use of session management in Servlet		
7.		Develop web Applications using JSP		
8.		Create applications using Include Directive JSP Include Action		
9.		Create a JSP based web application which allows the user to edit his/her database Information		
10.		An EJB application that demonstrates Session Bean. - Stateless Bean		
11.		An EJB application that demonstrates Session Bean. – Stateful Bean		
12.		An EJB application that demonstrates Entity Bean.		
13.		MVC Architecture Implementing MVC with Request Dispatcher Data Sharing Approaches		
14.		Build a web application that collects the user's name and displays "Hello World" followed by the user name		
15.		Create our view which will be required to browse and upload a selected file.		

EX.NO: 1 **REG.NO :**

DATE: **NAME:**

ADDITION OF TWO NUMBERS USING RMI

Aim:

To add two Numbers using RMI concepts

Procedure :

1. Create the remote interface
2. Provide the implementation class of the remote interface
3. Create the remote (server) application
4. Create client application

Program :

1. **Create the remote interface(Adder.java)**

```
import java.rmi.*;
public interface Adder extends Remote
{
public int add(int x,int y)throws RemoteException;
}
```

2. **Provide the implementation class of the remote interface(AdderRemote.java)**

```
import java.rmi.*;
import java.rmi.server.*;
public class AdderRemote extends UnicastRemoteObject implements Adder{
AdderRemote()throws RemoteException{
```

```
super();
}
public int add(int x,int y)
{
return x+y;}
}
```

3. Create the remote(server) application(MyServer.java)

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer{
public static void main(String args[]){
try{
Adder stub=new AdderRemote();
Naming.rebind("rmi://localhost:5000/sonoo",stub); }catch(Exception e)
{System.out.println(e);}
}
}
```

4. Create client application(MyClient.java)

```
import java.rmi.*;
public class MyClient{
public static void main(String args[]){
try{
Adder stub=(Adder)Naming.lookup("rmi://localhost:/sonoo");
System.out.println(stub.add(34,4));
}catch(Exception e){}
} }
```

For running this rmi example

1) compile all the java files

```
javac *.java
```

2) create stub and skeleton object by rmic tool

rmic AdderRemote

3) start rmi registry in one command prompt

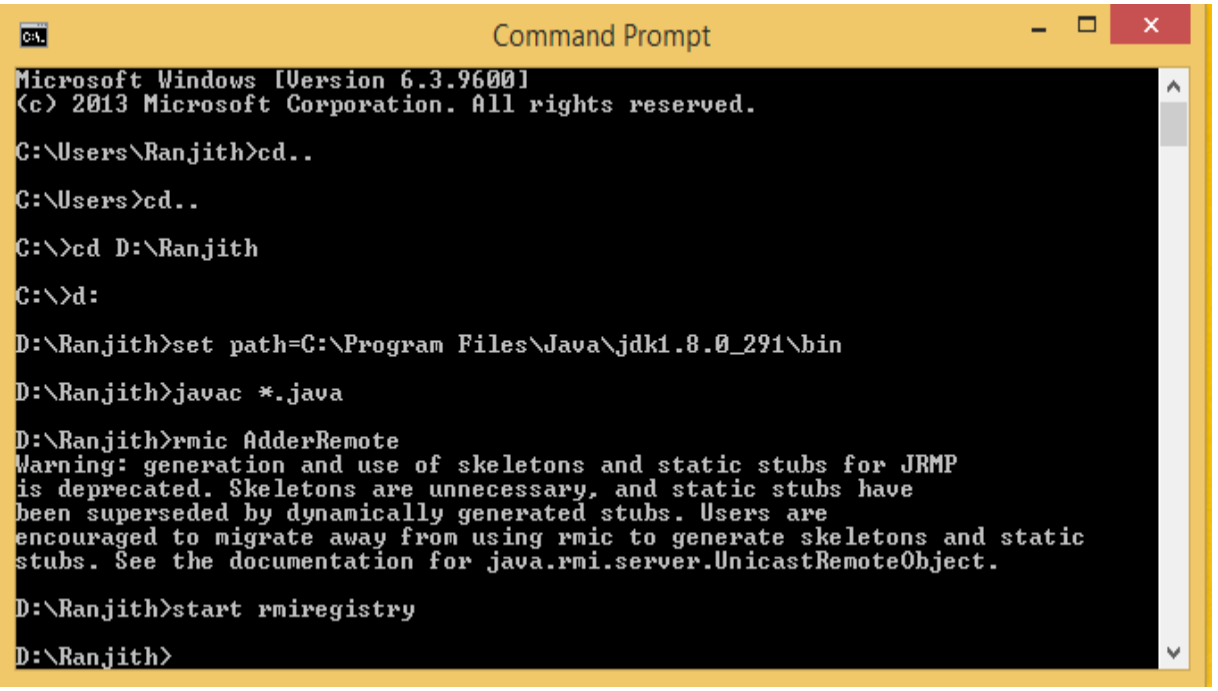
start rmiregistry

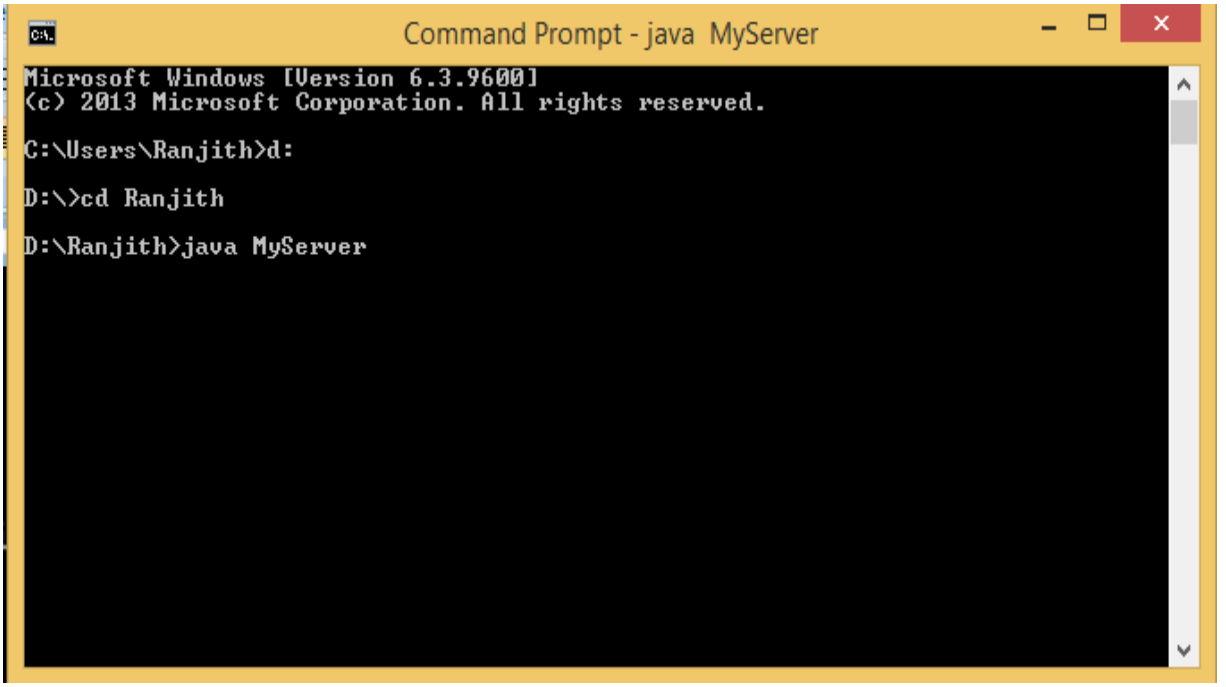
4) start the server in another command prompt

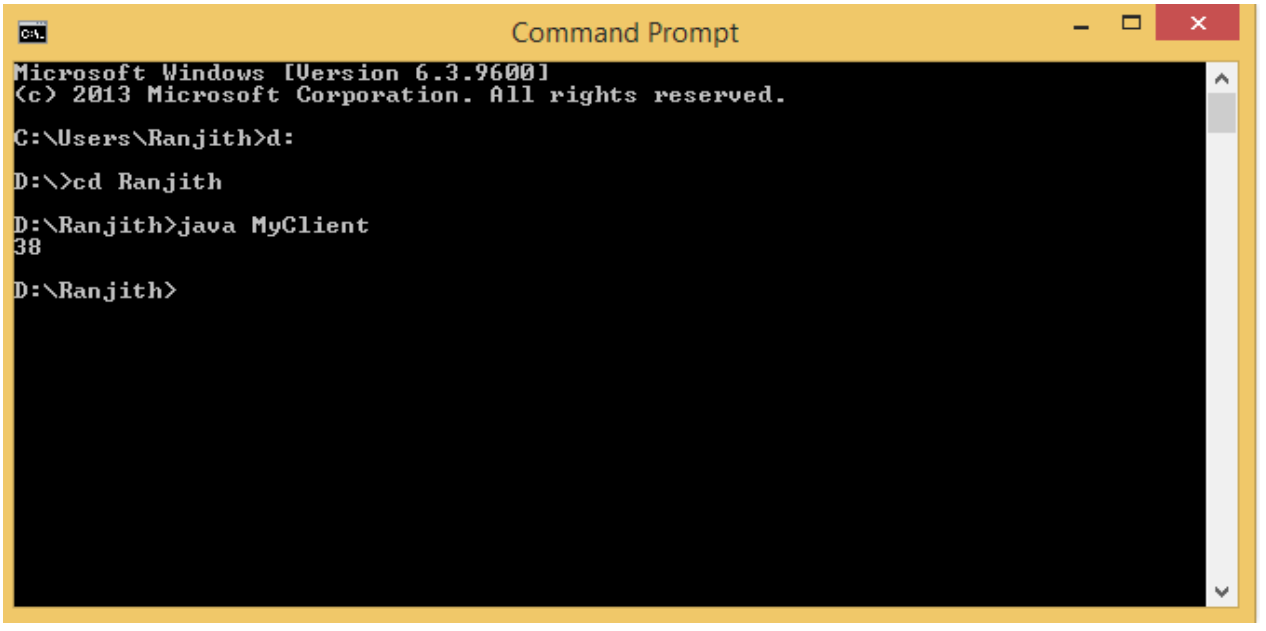
java MyServer

5) start the client application in another command prompt

java MyClient







RESULT: Thus the program has been executed successfully and output verified.

EX.NO: 2 **REG.NO :**

DATE: **NAME:**

CREATE APPLICATIONS USING JDBC

Aim: To retrieve the records from employee table using Mysql Database and JDBC**.**

Procedure :

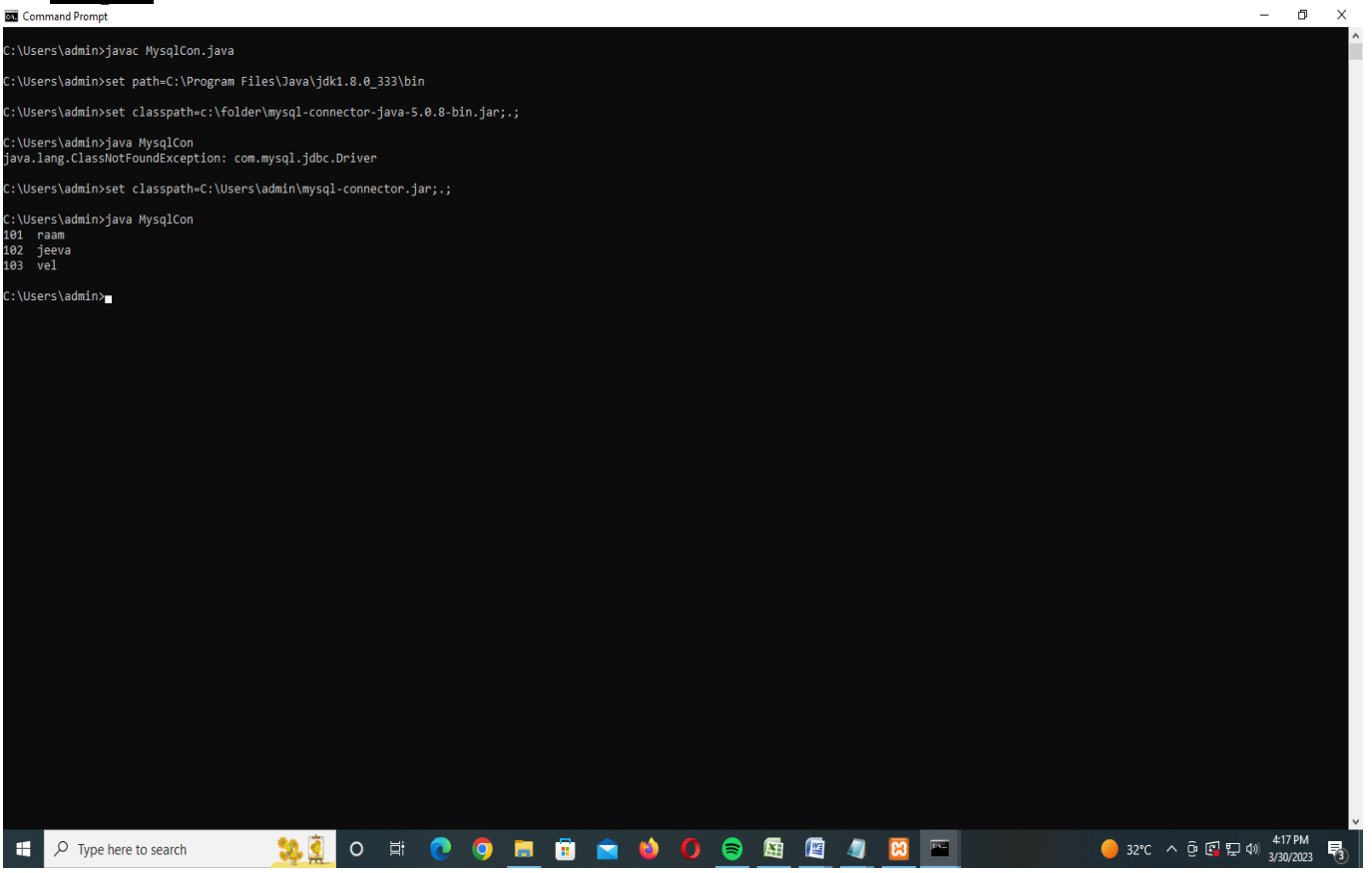
There are 5 steps to connect any java application with the database using JDBC. These steps areas follows:

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

Program(MysqlDB)

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/emp","root","");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from employee");
while(rs.next())
{
System.out.println(rs.getInt(1)+"  "+rs.getString(2));
}
con.close();
}catch(Exception e)
{ System.out.println(e);
}
}
}
```

Output:



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 3 **REG.NO :**

DATE: **NAME:**

CREATE STUDENT APPLICATIONS USING JDBC

Aim:

To create a student application using Mysql Database and JDBC**.**

Procedure :

There are 5 steps to connect any java application with the database using JDBC. These steps areas follows:

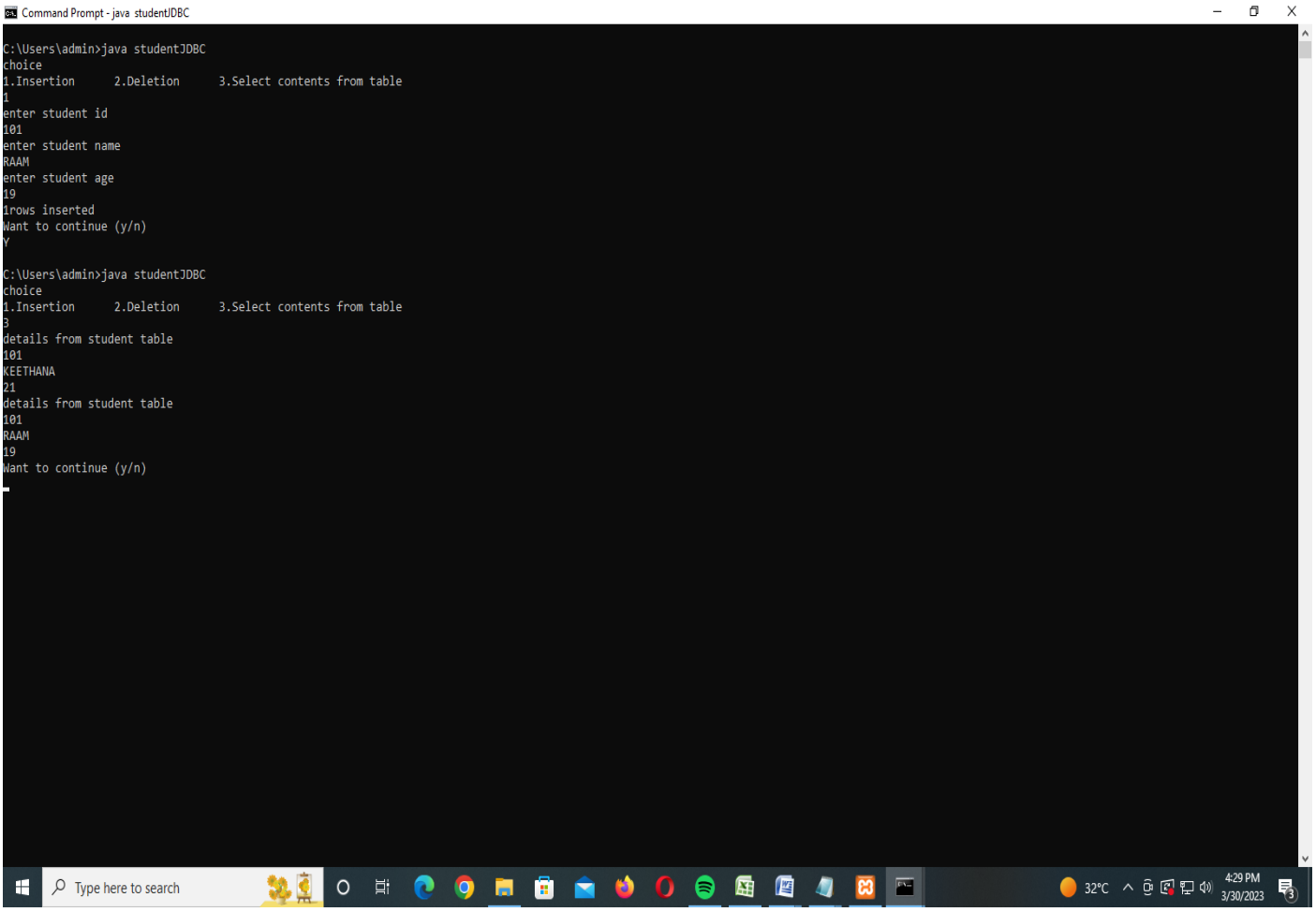
1. Register the Driver class
2. Create connection
3. Create statement interface
4. Execute queries
5. Close connection

Program(MysqlDB)

```
import java.sql.*;
import java.io.*;
public class studentJDBC
{
public static void main(String args[])
{
int choice;
String n,ch;
int id,age;
int rows;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/bsc","root","");
Statement s=con.createStatement();
do
{
System.out.println("choice");
System.out.println("1.Insertion\t 2.Deletion \t 3.Select contents from table");
choice=Integer.parseInt(br.readLine());
switch(choice)
{
case 1:
{
String ss="insert into student(sid,sname,age) values(?,?,?)";
```

```
PreparedStatement ps=con.prepareStatement(ss);
System.out.println("enter student id");
id=Integer.parseInt(br.readLine());
System.out.println("enter student name");
n=br.readLine();
System.out.println("enter student age");
age=Integer.parseInt(br.readLine());
ps.setInt(1,id);
ps.setString(2,n);
ps.setInt(3,age);
rows=ps.executeUpdate();
System.out.println(rows+"rows inserted");
break;
}
case 2:
{
String ss="delete from student where id=?";
PreparedStatement ps=con.prepareStatement(ss);
System.out.println("enter student id to delete");
id=Integer.parseInt(br.readLine());
ps.setInt(1,id);
rows=ps.executeUpdate();
System.out.println(rows+"rows deleted");
break;
}
case 3:
{
ResultSet rs=s.executeQuery("select * from student");
while(rs.next())
{
System.out.println("details from student table");
System.out.println(rs.getInt(1));
System.out.println(rs.getString(2));
System.out.println(rs.getInt(3));
}
break;
}
default:
System.out.println("choose between 1 to 3");
}
System.out.println("Want to continue (y/n)");
ch=br.readLine();
}while(ch.equals("y"));
s.close();
con.close();
}
catch(Exception e)
{
System.out.println("Exception caught"+e);
}
}
}
```

Output:



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 4 **REG.NO :**

DATE: **NAME:**

DEVELOP WEB APPLICATIONS USING SERVLET

Aim: Develop simple Login Application using Servlet.

Procedure :

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

Index.html

```
<form method="post" action="LoginServlet"> Email
ID:<input type="text" name="email" ><br/>
Password:<input type="password" name="pass" /><br/>
<input type="submit" value="login" />
</form>
```

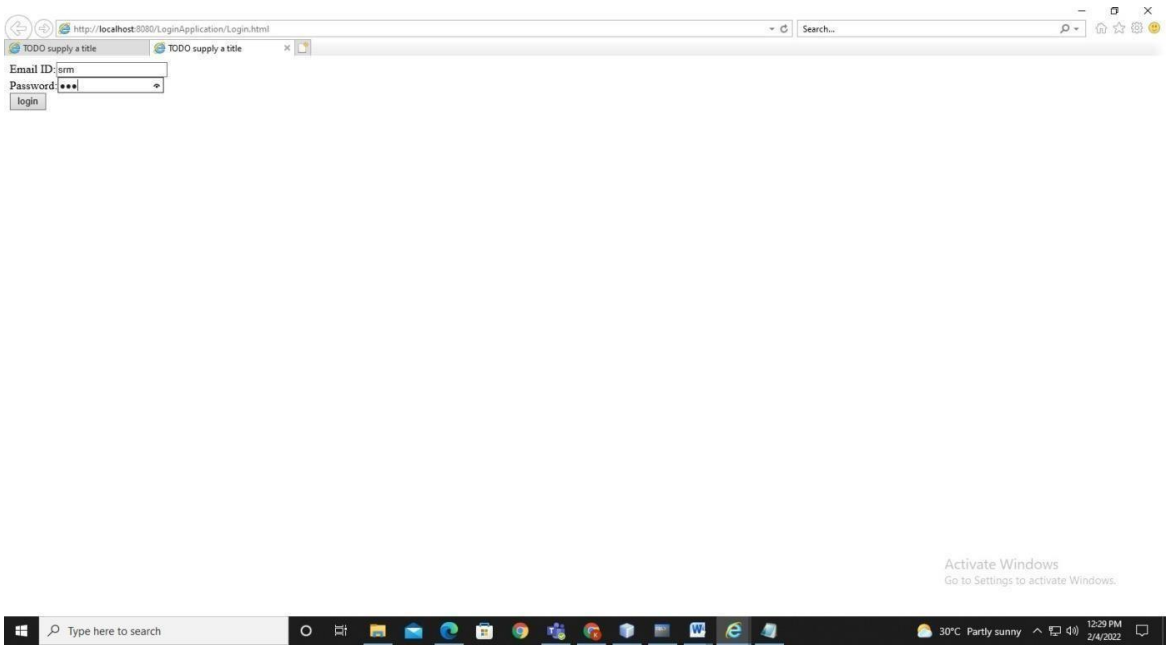
LoginServlet.java

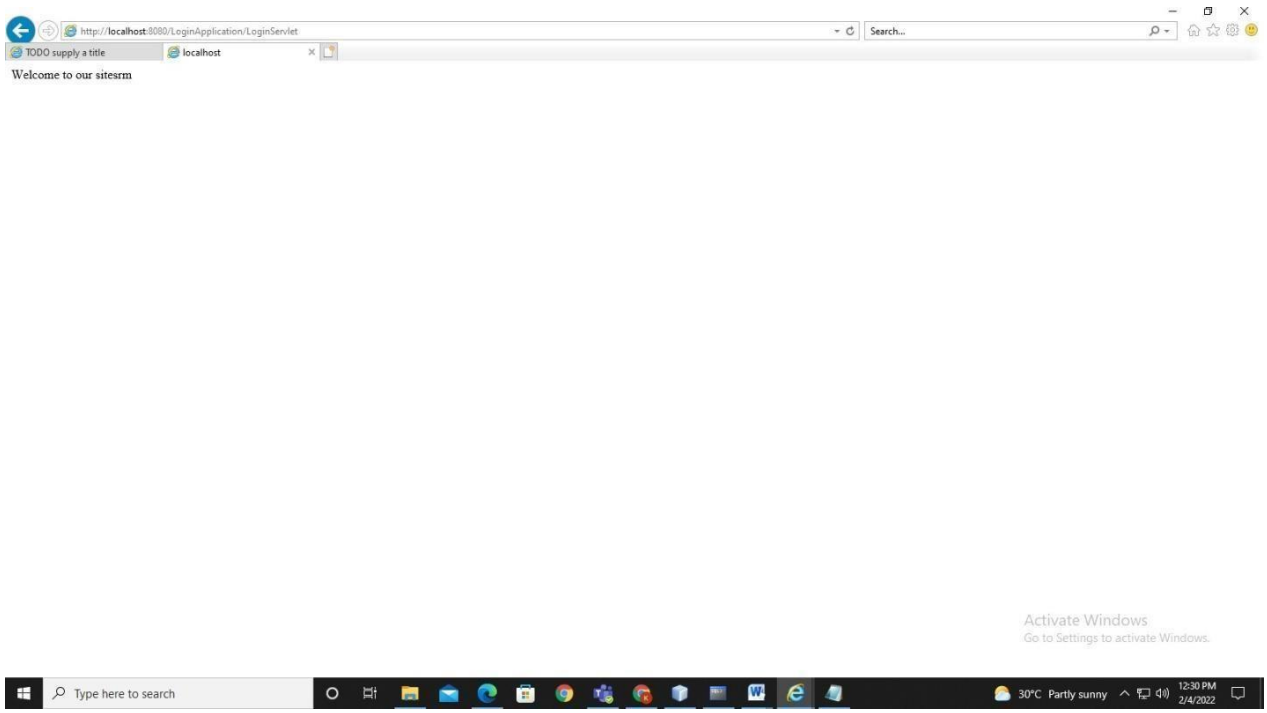
```
import java.io.*; import
javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet

{

  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException

   {

      response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();
 String user = request.getParameter("email"); String
 pass = request.getParameter("pass");
 if(user.equals("srm")&& pass.equals("123"))
 {

 out.println("Welcome to our site" +  user);

 }

 else

 {

 out.println("Username or Password incorrect!! Try Again");

 }

 }
```

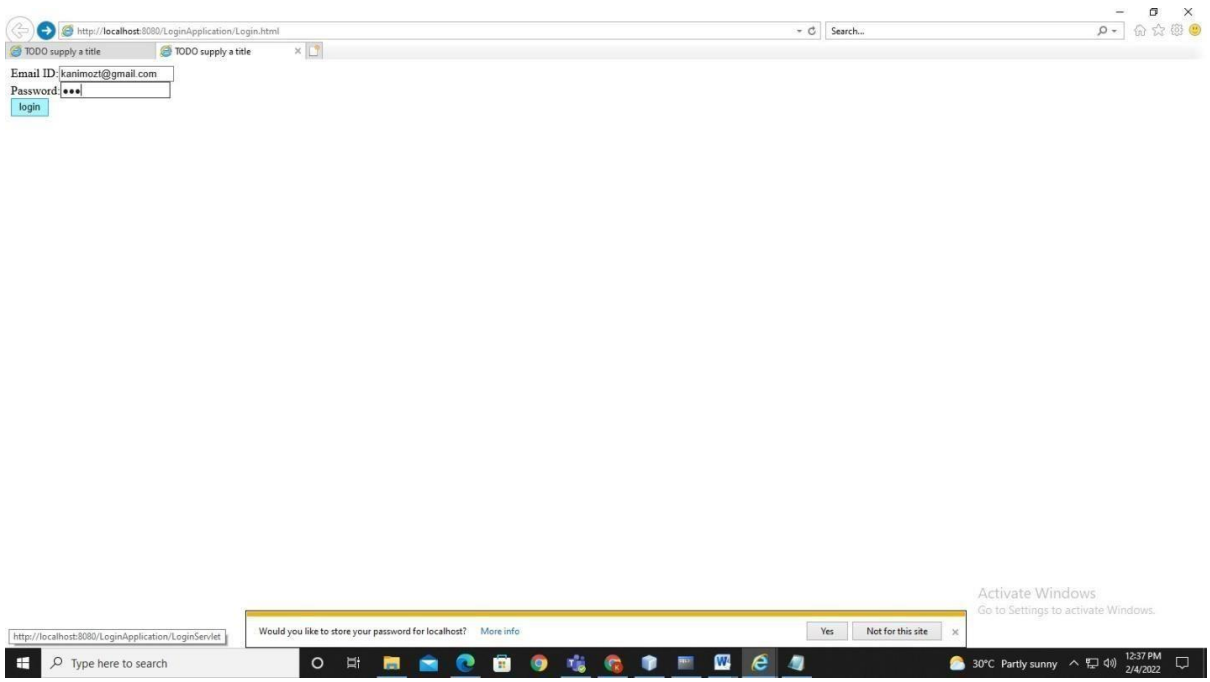
Web.xml

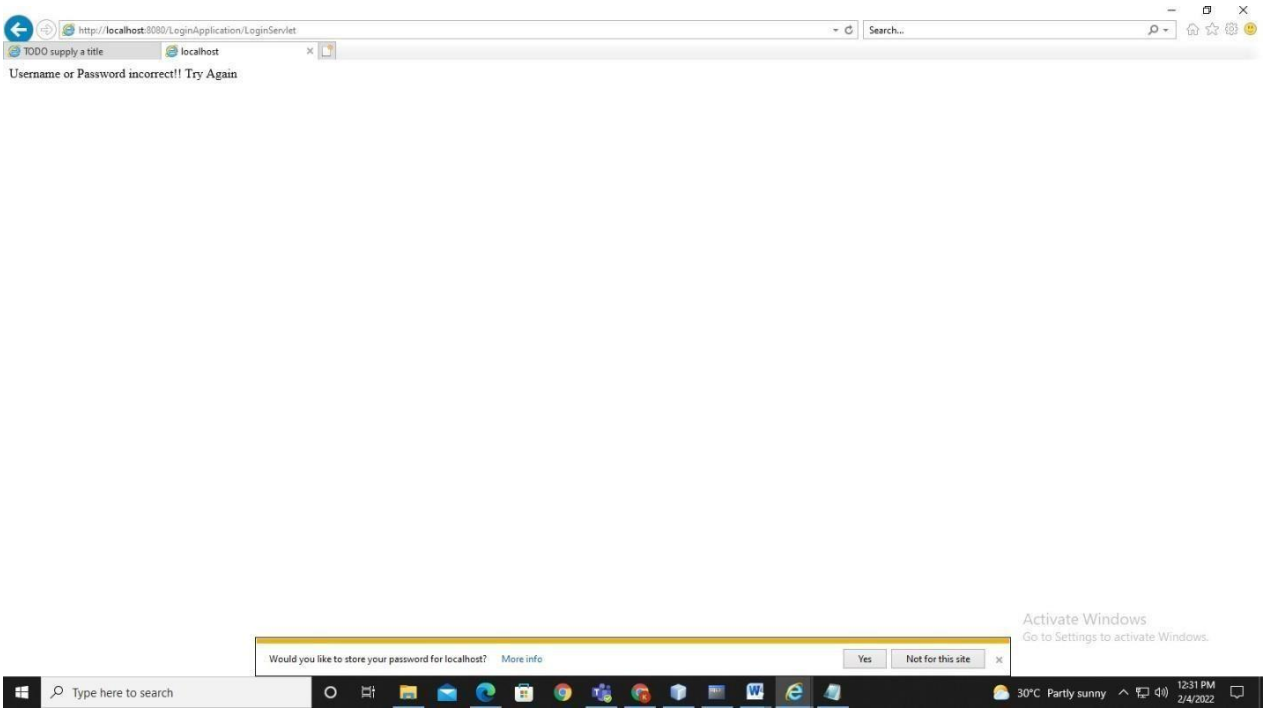
```
<?xml version="1.0" encoding="UTF-8"?>

<servlet>

<servlet-name>Login</servlet-name>

<servlet-class>Login</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Login</servlet-name>

<url-pattern>/login</url-pattern>

</servlet-mapping>

</web-app>
```

OUTPUT:




http://localhost:8080/LoginApplication/LoginServlet
Search...
TODO supply a title
localhost
Welcome to our sitesrm
Activate Windows
Go to Settings to activate Windows.
Type here to search
30°C Partly sunny
12:30 PM
2/4/2022


http://localhost:8080/LoginApplication/Login.html
Search...
TODO supply a title
TODO supply a title
Email ID: kanimozt@gmail.com
Password:
login
Activate Windows
Go to Settings to activate Windows.
http://localhost:8080/LoginApplication/LoginServlet
Would you like to store your password for localhost? More info
Yes
Not for this site
Type here to search
30°C Partly sunny
12:37 PM
2/4/2022



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 5 **REG.NO :**

DATE: **NAME:**

TO CREATE USER LOGIN FORM USING SERVLET

Aim:

To create the user Login page and to Read the Form Data using Servlet

Procedure :

The steps are as follows:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

Program:

Get.html

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

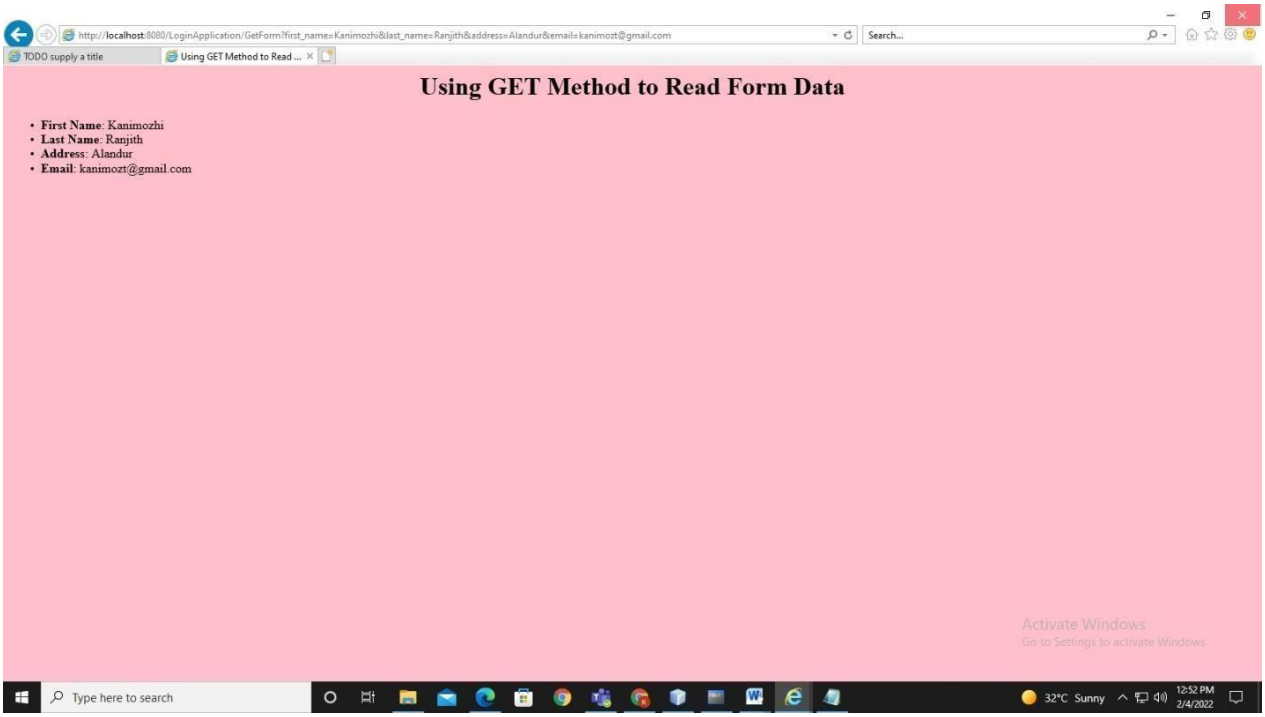
```
  </head>
    <body>
      <form action = "GetForm" method = "GET">
         First Name:  <input type = "text" name = "first_name">
         <br />
         Last Name:       <input type = "text" name = "last_name" />
         <br />
         Address:         <input type = "text" name = "address"/>
         <br />
         Email:           <input type = "text" name = "email"/>
         <br />
         <input type = "submit" value = "Submit" />
       </form>
     </body>
</html>
```

GetForm.java

```
import java.io.*; import
javax.servlet.*;
import javax.servlet.http.*;
public class GetForm extends HttpServlet
{
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
{
  response.setContentType("text/html;charset=UTF-8"); PrintWriter
  out = response.getWriter();
String title = "Using GET Method to Read Form Data"; String
docType =
  "<!doctype html public \"-//w3c//dtd html 4.0 " + "transitional//en\">\n"; out.println(docType +
"<html>\n"
    + "<head><title>" + title + "</title></head>\n" + "<body bgcolor
    = \"pink\">\n" +
      "<h1 align = \"center\">" + title + "</h1>\n" + "<ul>\n" +
        "  <li><b>First Name</b>: "
        + request.getParameter("first_name") + "\n" + "
        <li><b>Last Name</b>: "
        + request.getParameter("last_name") + "\n" + "
        <li><b>Address</b>: "
        + request.getParameter("address") + "\n" + "
        <li><b>Email</b>: "
        + request.getParameter("email") + "\n" + "</ul>\n" +
    "</body>" +
  "</html>"
);}}
```

Output:



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 6 **REG.NO :**

DATE: **NAME:**

SESSION MANAGEMENT IN SERVLET

AIM:

Servlet application for session management using Cookies.

PROGRAM:

FirstServlet.java

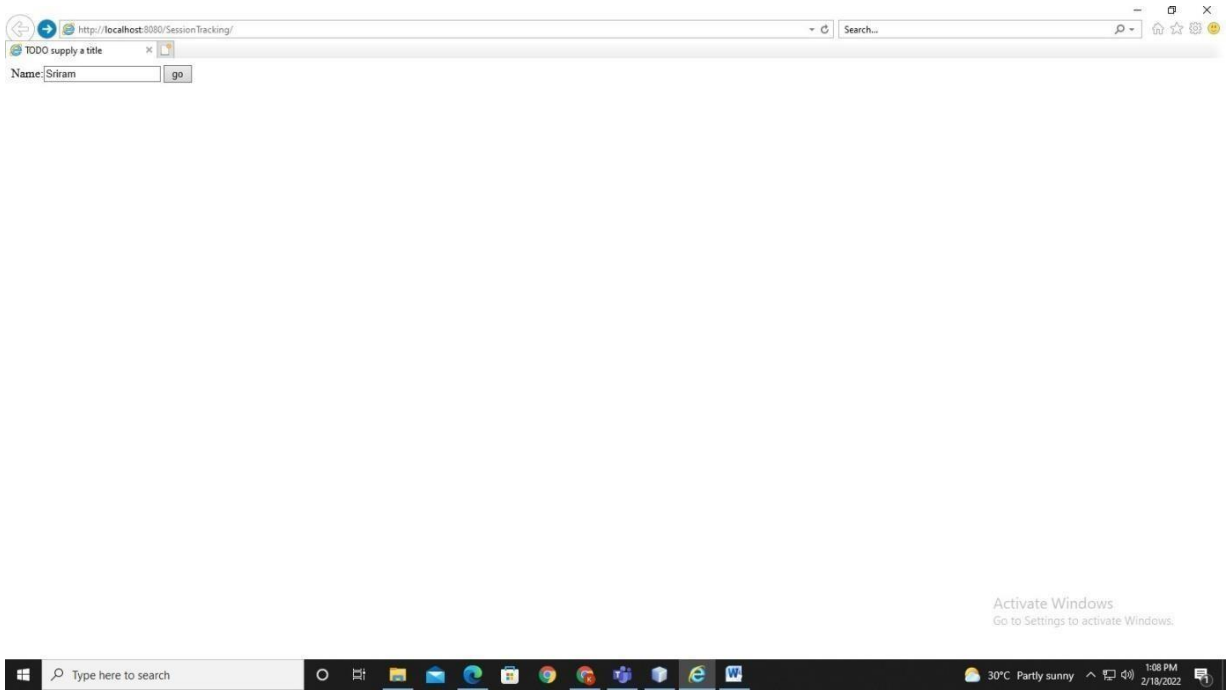
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class FirstServlet extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws
ServletException, IOException
{
 response.setContentType("text/html");

  PrintWriter out = response.getWriter();

  String n=request.getParameter("userName");
 out.print("Welcome "+n);
 out.print("<a href='SecondServlet?uname="+n+"'>visit</a>");
 out.close();
 }

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
  {

    processRequest(request, response);

    }

}
```

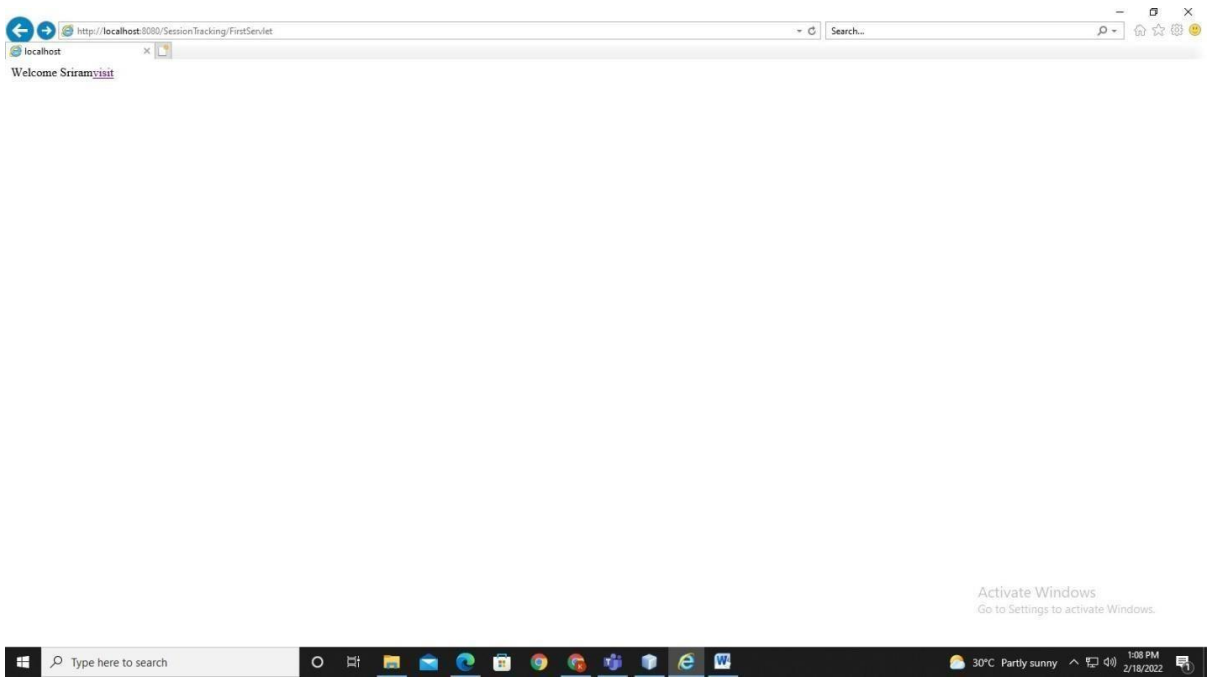
Second Servlet.java

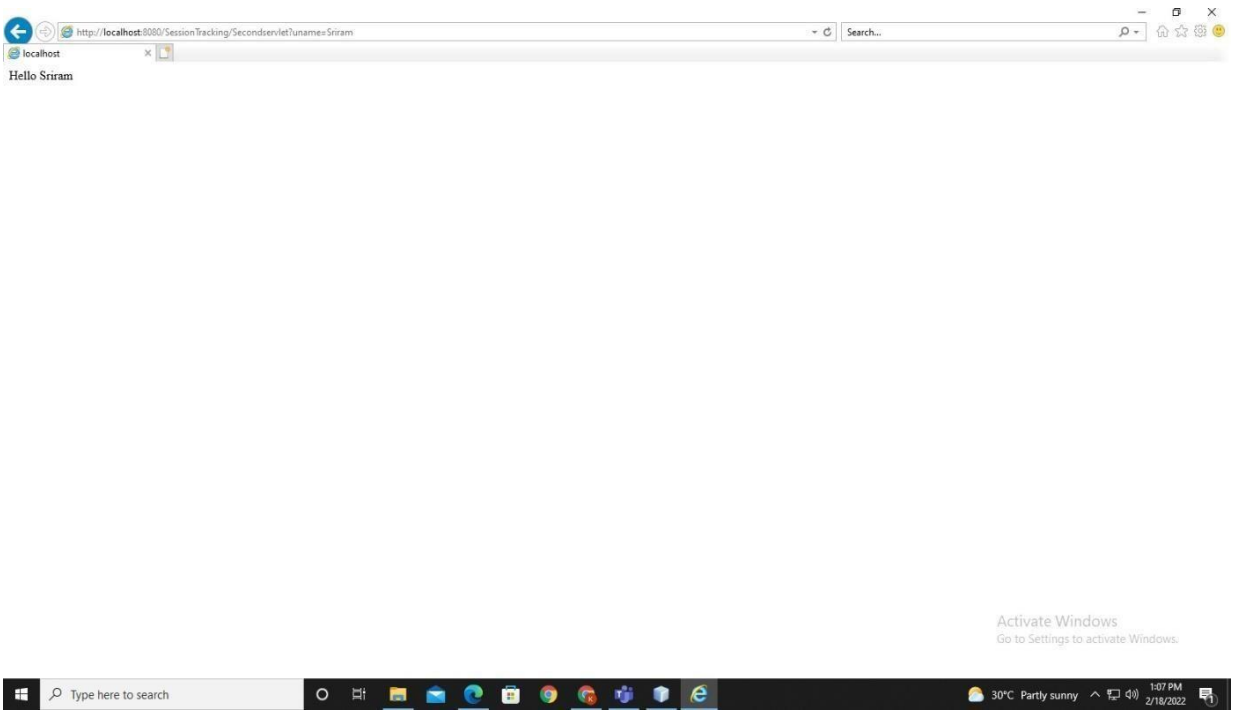
```
import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SecondServlet extends HttpServlet

{

 protected void processRequest(HttpServletRequest request,
          HttpServletResponse response)throws ServletException, IOException
 {

 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 String n=request.getParameter("uname");
 out.print("Hello "+n);
 out.close();
  }

  protected void doPost (HttpServletRequest request, HttpServletResponse response)throws
  ServletException, IOException
    {

        processRequest(request, response);
     }

     }
```

Index.html

```
   <html>

   <body>

   <form action="FirstServlet" method="post">
   Name:<input type="text" name="userName"/><br/>
   <input type="submit" value="go"/>

   </form>

   </body>

   </html>
```


http://localhost:8080/SessionTracking/
Search...
TODO supply a title
Name: Sriram
go
Activate Windows
Go to Settings to activate Windows.
Type here to search
30°C Partly sunny
1:08 PM
2/18/2022


http://localhost:8080/SessionTracking/FirstServlet
Search...
localhost
Welcome Sriramvisit
Activate Windows
Go to Settings to activate Windows.
Type here to search
30°C Partly sunny
1:08 PM
2/18/2022



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 7 **REG.NO :**

DATE: **NAME:**

:

WEB APPLICATIONS USING JSP

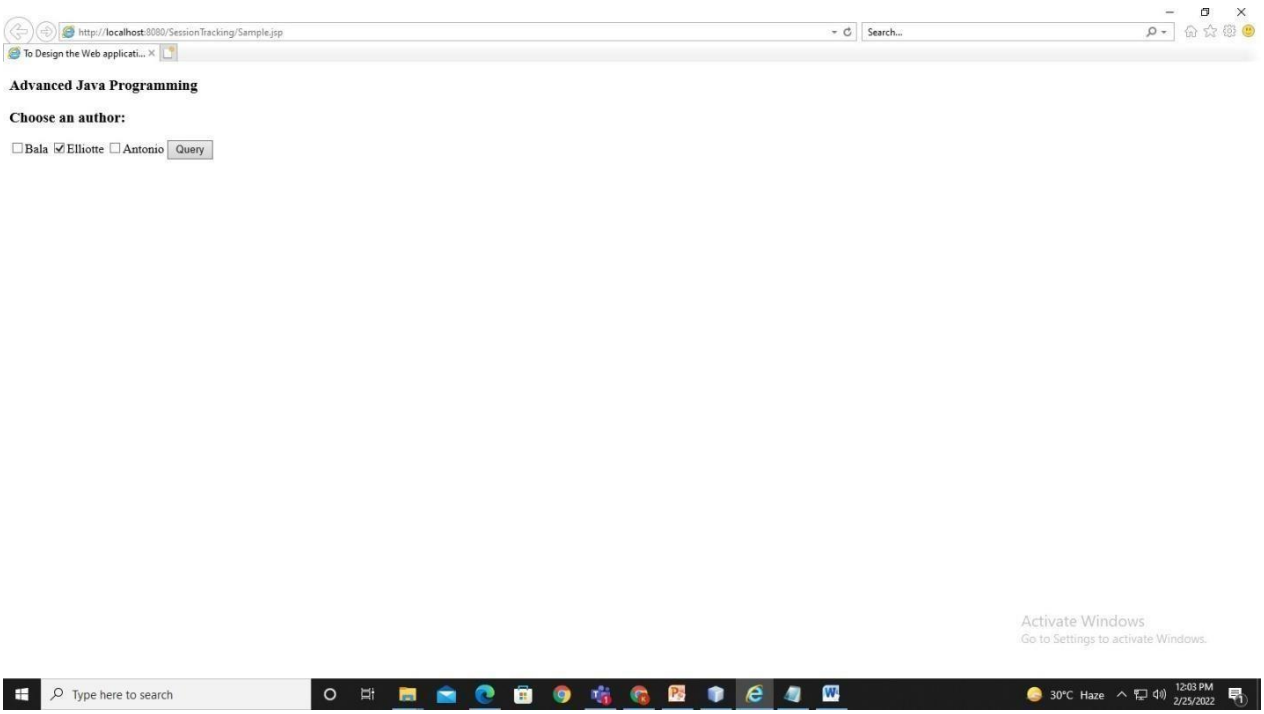
Aim : To Develop a web application to select the author of a Text Book using JSP.

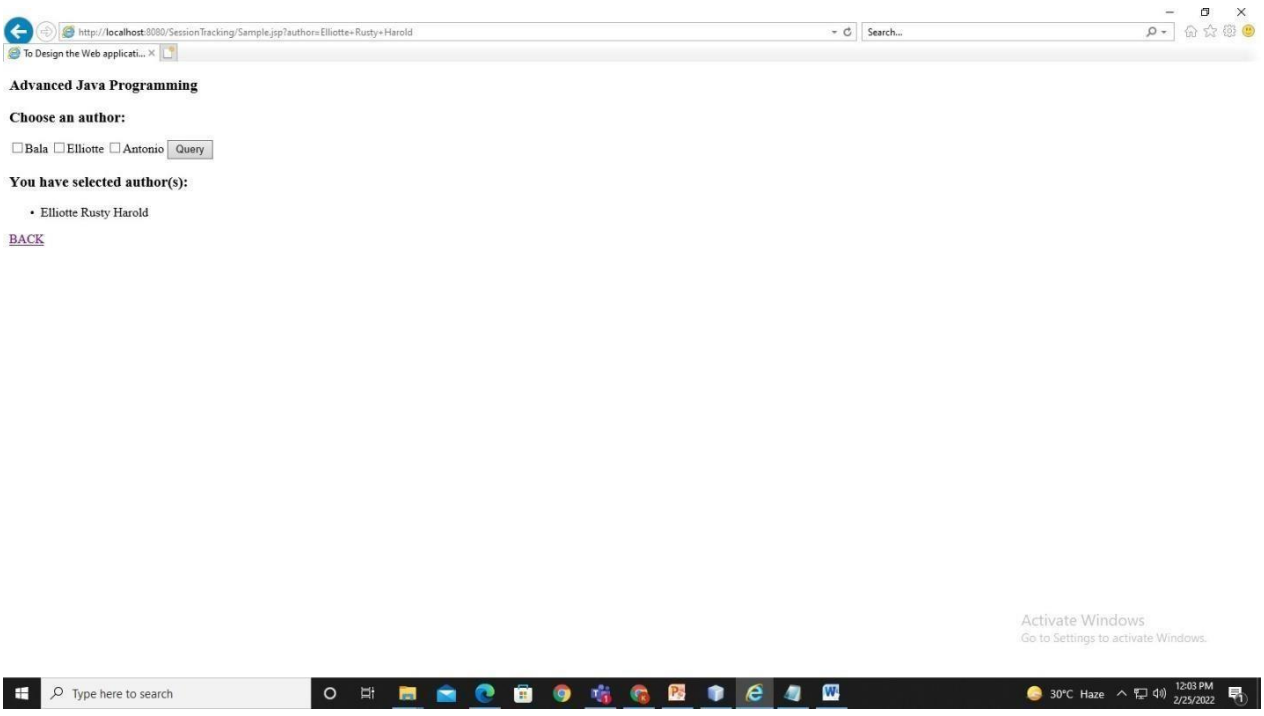
Program:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>To Design the Web application using JSP </title>
</head>
<body>
<h3>Advanced Java Programming</h3>
<h3>Choose an author:</h3>
<form method="get">
<input type="checkbox" name="author" value="Balaguruswamy">Bala
<input type="checkbox" name="author" value="Elliotte Rusty Harold">Elliotte
<input type="checkbox" name="author" value="Antonio Goncalves">Antonio
<input type="submit" value="Query">
</form>
```

```
<%
String[] authors = request.getParameterValues("author"); if (authors
!= null) {
%>
<h3>You have selected author(s):</h3>
<ul>
<%
for (int i = 0; i < authors.length; ++i) {
%>
<li><%= authors[i] %></li>
<%
}
%>
</ul>
<a href="<%= request.getRequestURI() %>">BACK</a>
<%
}
%>
</body>
</html>
```

Output:





RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 8 **REG.NO :**

DATE: **NAME:**

CREATE APPLICATIONS USING INCLUDE DIRECTIVE

AIM:

To create web applications using Include Directive JSP Include Action

Procedure :

1. Create a directory structure
2. Create a JSP file
3. Compile the file
4. Create a deployment descriptor
5. Start the server and deploy the project

Program:

Date.jsp

```
<html>
<head>
<title>Date Display</title>
</head>
<body>
<%-- JSP comments --%>
<%@page import="java.util.Date"%>
<%! Date date; %>
<% date = new Date(); %>
<b> System date and time: </b> <%= date %>
```

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

Order.jsp

```
<HTML>
<HEAD>
    <TITLE>A Catalog Order Form</TITLE>
</HEAD>
<BODY>
<H1> Invoice Form</H1>
<%!
    String item[] = {"Pen", "Sketch", "Marker"}; double
    price[] = {20.50, 10.50, 15.50};
    int quantity[] = {10, 15, 25};
%>
<TABLE ALIGN="center" BGCOLOR="blue" BORDER="1" WIDTH="75%">
<TR><TD>Item</TD>
<TD>Price</TD>
<TD>Quantity</TD>
<TD>Total Price</TD>
</TR>
<% for (int i=0; i<3; i++) { %>
      <TR><TD><%= item[i] %></TD>
      <TD><%= price[i] %></TD>
      <TD><%= quantity[i] %></TD>
      <TD><%= price[i] * quantity[i] %></TD>
      </TR>
<% }
//end for loop %>
</TABLE>
</BODY>
</HTML>
```

Includedir.jsp

```
<html>
<head>
<title>JSP Program using Include Directive</title>
</head>
<body>
<h1>Include the First File:</h1>
<br><br>
<%@ include file="Date.jsp"%>
<br><br><br>
<h2>Include the Second File:</h2>
<br>
<%@ include file="Order.jsp"%>
</body>
</html>
```

Output:

RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 9 **REG.NO :**

DATE: **NAME:**

CREATE A JSP BASED WEB APPLICATION USING DATABASE

Aim :

To write a JSP Program for Storing the Employee Details through JDBC Connectivity andPrepare the PayRoll.

Procedure :

Index (HTML)

Step 1 : Create New Web Application

Step 2 : Open New HTML file

Step 3 : Design the Employee Details Page using Text Boxes and Submit Buttons

Step 4 : Give the JSP Name In the form action to Invoke the Server and Validate

Server (JSP)

Step 1 : Create new JSP Program

Step 2 : Get the Values from Employee Details Form

Step 3 : Create a new Database

Step 4 : Create a new Table

Step 5 : Process Pay Roll and Store all in the Table

Step 6 : Print the PayRoll

Step 7 : End the program

Program:

/* Implementation of JSP Program for Storing the EmployeeDetails through JDBC and Prepare the PayRoll */

index.html *(HTML File)*

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Employee Details HTML Page</title>
    </head>
```

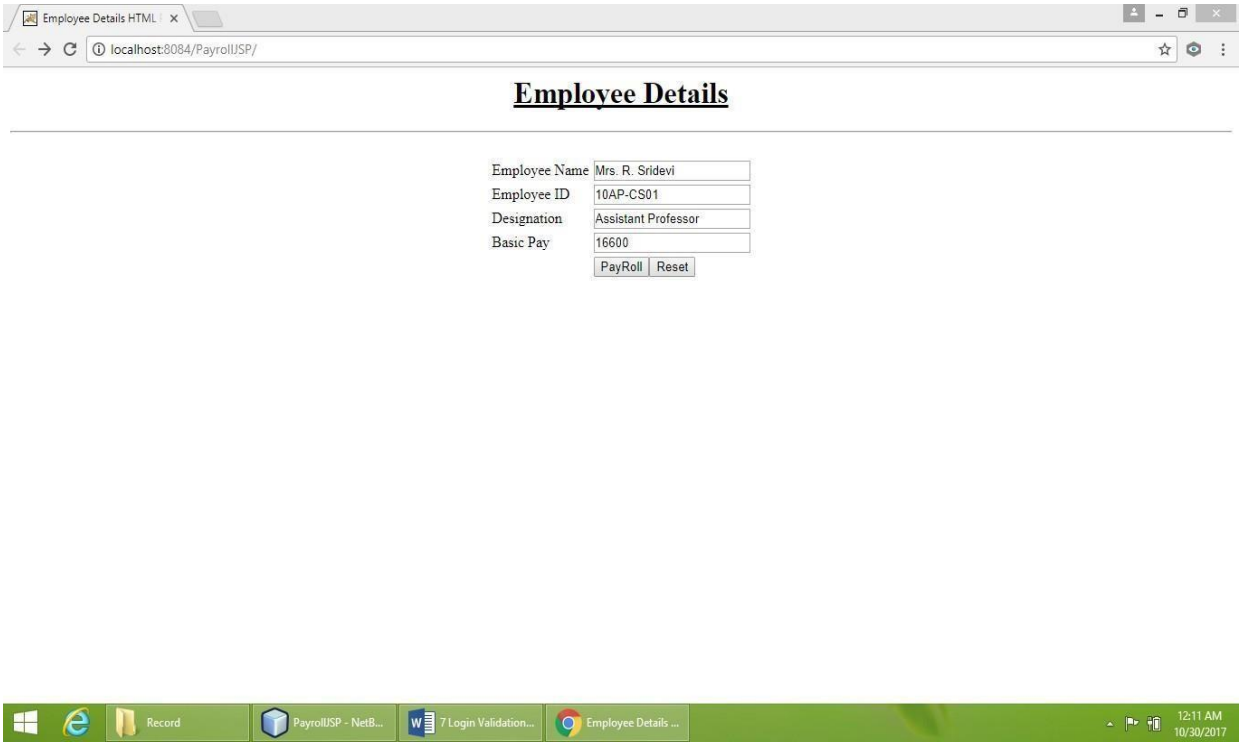
```
<body>
    <center><h1> <b><u>Employee Details</u></b></h1><hr>
        <form action="Employee.jsp">
        <table>
        <tr><td>Employee Name</td><td><input type="text"
                name="ename"></td></tr>
        <tr><td>Employee ID</td><td><input type="text"
                name="eid"></td></tr>
        <tr><td>Designation</td><td><input type="text"
                name="desig"></td></tr>
        <tr><td>Basic Pay</td><td><input type="text"
                name="bp"></td></tr>
        <br>
        <tr><td></td>
        <td><input type="submit" value="PayRoll"><input type ="reset"
                value="Reset"</td></tr></b>
        </table>
        </form>
    </center>
</body>
</html>
```

Employee.jsp *(Java Server Page)*

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Store the Employee Details</title>
    </head>
    <body>
        <%@ page language="java" %>
        <%@ page import="java.sql.*" %>
        <%@ page import="java.sql.DriverManager.*" %>

    <%
        String ename = request.getParameter("ename");String eid =
        request.getParameter("eid"); String desig =
        request.getParameter("desig");String bp =
        request.getParameter("bp");
```

```
double bp1 = Integer.parseInt(bp);
double  pf = (bp1/100)*12;
double da = (bp1/100)*70; double hra =
(bp1/100)*10; double gp = bp1 + da +
hra;double np = gp - pf;
Class.forName("org.apache.derby.jdbc.ClientDriver");
String conn = "jdbc:derby://localhost:1527/Employee;create = true;user=Muthamizh;
                password=periyar";
Connection con = DriverManager.getConnection(conn);
 PreparedStatement ps = con.prepareStatement("insert into Employee1
                                (Name, ID, Desig, BP, DA, HRA, GP, PF, NP)
                                values(?,?,?,?,?,?,?,?,?)");

    ps.setString(1, ename);
    ps.setString(2, eid);
    ps.setString(3, desig);
    ps.setDouble(4, bp1);
    ps.setDouble(5, da);
    ps.setDouble(6, hra);
    ps.setDouble(7, gp);
    ps.setDouble(8, pf);
    ps.setDouble(9, np);
    ps.executeUpdate();
    out.println("Employee Name                    : <b>" + ename);
    out.println("</b><br>");
    out.println("Employee ID                      : <b>" + eid);
    out.println("</b><br>"); out.println("Basic    : <b>" + bp);
    Pay out.println("</b><br>");
    out.println("Dearness Allowance               : <b>" + da);
    out.println("</b><br>"); out.println("House
    Rent Allowanceout.println("</b><br>");        : <b>" + hra);
    out.println("Gross Pay                        : <b>" + gp);
    out.println("</b><br>"); out.println("Emp.
    Provident Fund out.println("</b><br>");
    out.println("Net Pay out.println("</b><br>");  : <b>" + pf);
    ps.close();                                   : <b>" + np);
    con.close();
%>
</body>
</html>
```

Output



Result:

Thus the program has been executed successfully and output verified

EX.NO: 10 **REG.NO :**

DATE: **NAME:**

AN EJB APPLICATION THAT DEMONSTRATES SESSION BEAN - STATELESS BEAN

AIM:

To develop an EJB application that demonstrates Stateless Session Bean for addition of Numbers**.**

PROCEDURE:

1. Select file -> new project -> java EE -> Enterprise Application -> next -> give project name and project location -> next -> select glassFishServer -> Select all EJBModule,WebApplicationModule and finish.

2. Right click on project-ejb and select new-> session bean -> give EJBName, location,package And session type (stateless)& select local interface. It will create java files like projectname.java, local.java.

3. Right click inside projectname.java page -> click insert code -> add business method -> and Define business methods.method : add returnType:int param1:int –a param2:int-b (Press OK).

4. change in add method return (a + b)

5. Create a servlet by right clicking the war file and select -> new servlet, give name and location

-> next -> accept the servletname and URL pattern(s) fields and click finish.

6. Right click on the content of servlet session and select insert code then select callEnterprise Bean -> Select sessionBean -> click ok.

7. Add the following line in the end of try block
out.println("Addition :" + sessionBeandemo.add(10, 20));

8. Set the relative URL of the project -> right click the project folder -> select properties -> run -

>select display browser on run relative URL /MyServlet -> OK.

9. Build and run the project

PROGRAM:

AddSessionBean.java

```
package bsc.cs.ejb;
 import javax.ejb.Stateless;
@Stateless

public class AddSessionBean implements AddSessionBeanLocal {
    @Override
    public int add(int a, int b) {
        return (a+b);
    }

    // Add business logic below. (Right-click in editor and choose

    // "Insert Code > Add Business Method")

}
```

AddSessionBeanLocal.java

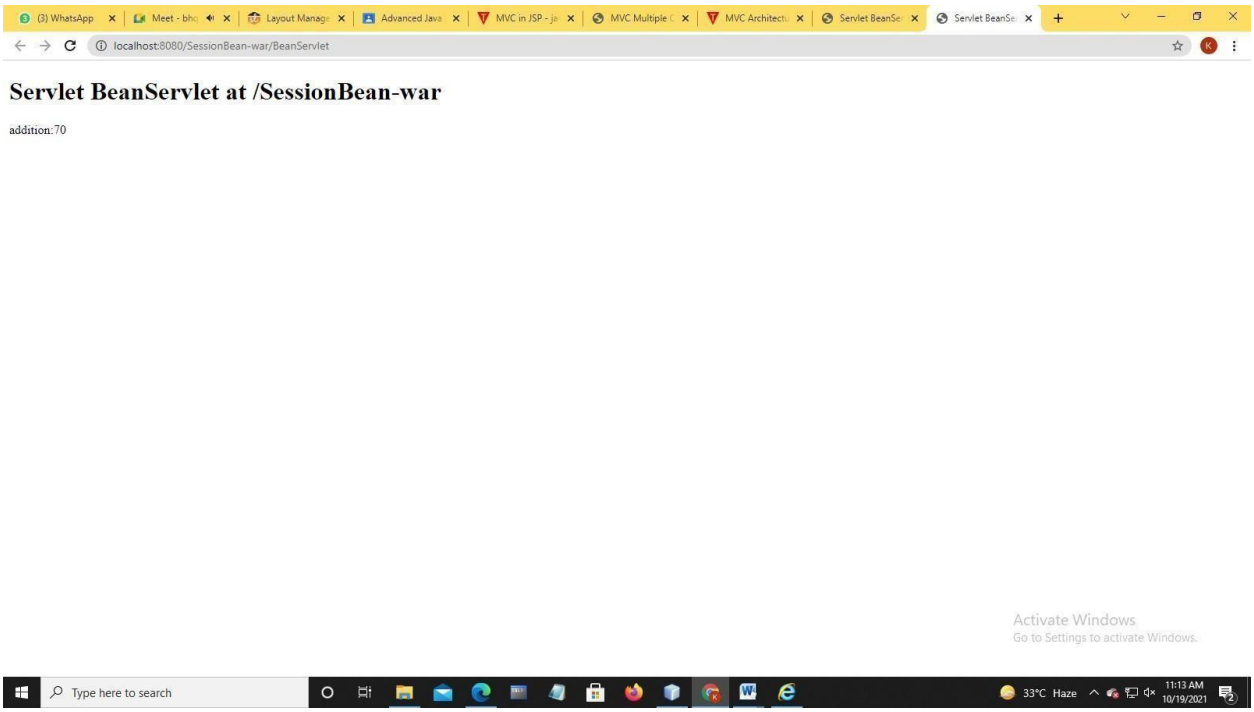
```
package bsc.cs.ejb;
import javax.ejb.Local;
@Local
public interface AddSessionBeanLocal {
     int add(int a, int b);
    }
```

AddServlet.java

```
package cs.web;
import bsc.cs.ejb.AddSessionBeanLocal;
import java.io.IOException;
import java.io.PrintWriter;
 import javax.ejb.EJB;
 import avax.servlet.ServletException;
 import javax.servlet.http.HttpServlet;
 import javax.servlet.http.HttpServletRequest;
 import javax.servlet.http.HttpServletResponse;
 @WebServlet(urlPatterns ={"/BeanServlet"})
 public class BeanServletextends HttpServlet {
    @EJB
 private Bean2Local bean2;
 protected void processRequest(HttpServletRequest request,HttpServletResponse
 response)throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
                /* TODO output your page here.You may
        use following sample code. */
                out.println("<!DOCTYPEhtml>");
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Servlet BeanServlet</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Servlet BeanServlet at " + request.getContextPath() +"</h1>");
                out.println("</body>");
                out.println("</html>");
                out.println("addition:"+bean2.Add(50,20));
            }
    }
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
            response)throws ServletException, IOException {
        processRequest(request, response);

    }
    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
            ServletException,IOException
    {
        processRequest(request, response);
    }

    public String getServletInfo()

     {
    return "Short description";

    }// </editor-fold>

}
```

Output:



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 11 **REG.NO :**

DATE: **NAME:**

AN EJB APPLICATION THAT DEMONSTRATES SESSION BEAN - STATEFULL BEAN

Aim:

To develop an EJB application that demonstrates Statefull Session Bean.

Procedure:

Step 1 : Create enterprise application project Step 2: Create EJB
Step 3 : Put Business login into EJB
Step 4: Create Servlet which call your business logic Step 5 : Define dependency between Servlet and EJB Step 6: Put controller logic into your servlet
Step 7 : Create jsp file
Step 8 : Fill JSP file with user interface code Step 9 : Deploy and run EJB war file.

Program:

Index.html

```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div>TODO write content</div>
    <form action="calcservlet" method="post">
        <input type="text" name="t1"></br>
      <input type="text" name="t2"></br>
      <input type="submit" value="go">
    </form>
  </body>
</html>
```

calcservlet.java

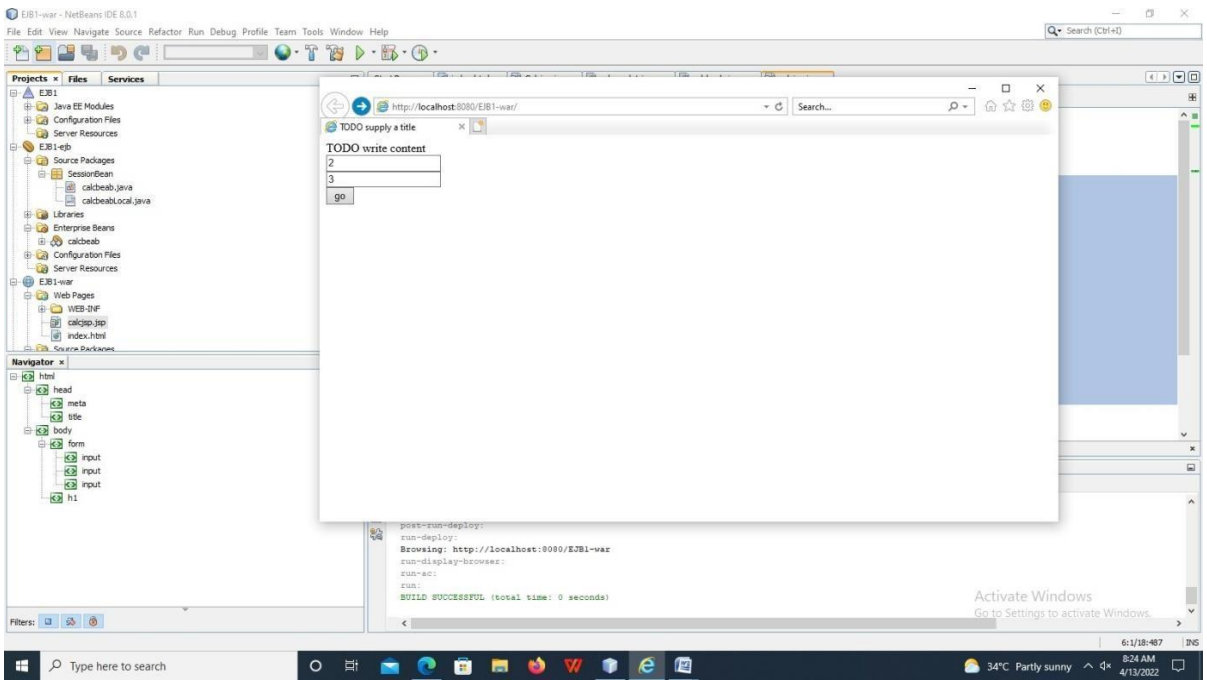
```
package SessionBean;
import
java.io.IOException;
import
java.io.PrintWriter;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class calcservlet extends HttpServlet {
   @EJB
   private calcbeabLocal calcbeab;
   protected void processRequest(HttpServletRequest request, HttpServletResponse
         response) throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-
      8"); try (PrintWriter out = response.getWriter()) {
         /* TODO output your page here. You may use following sample code. */
         out.println("<!DOCTYPE html>");
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Servlet calcservlet</title>");
         out.println("</head>");
         out.println("<body>");
         int
         a=Integer.parseInt(request.getParameter("t1")
         ); int
         b=Integer.parseInt(request.getParameter("t2"))
         ;
         out.println("<h1>SUM = "+calcbeab.addition(a,b)+"</h1>");
         out.println("</body>");
         out.println("</html>");
      }
   }
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse
         response) throws ServletException, IOException {
      processRequest(request, response);
   }
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse
         response) throws ServletException, IOException {
      processRequest(request, response);
   }
   @Override
   public String
      getServletInfo() { return
      "Short description";
   }}
```

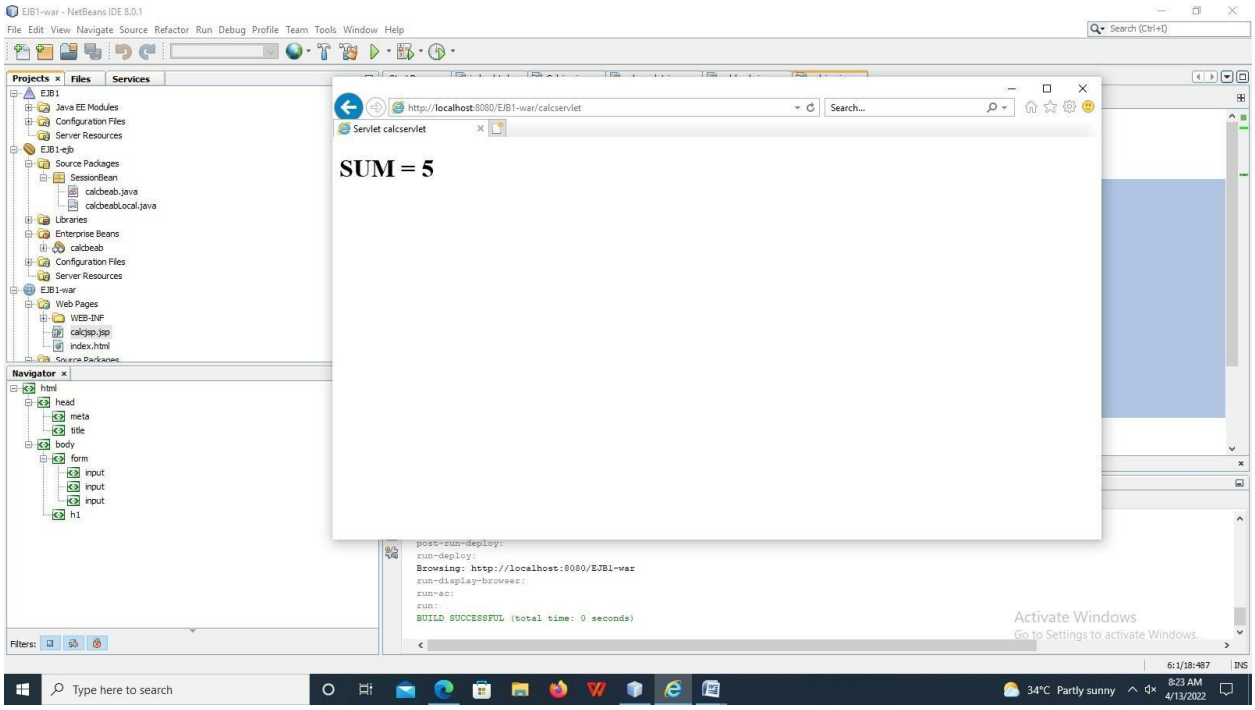
calcbeab.java

```
package SessionBean;
import javax.ejb.Stateful;
@Stateless
public class calcbeab implements
   calcbeabLocal { @Override
   public Integer addition(int a, int b) {
      return (a+b);
   }
}
```

calcjsp.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <form action="calcservlet" method="post">
        <input type="text" name="t1">
        <input type="text" name="t2">
              <input type="submit" value="ADD">
       </form>
      <h1>Hello World!</h1>
   </body>
</html>
```





RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 12 **REG.NO :**

DATE: **NAME:**

AN EJB APPLICATION THAT DEMONSTRATES ENTITY BEAN

Aim:

Develop an EJB application that demonstrates Entity Bean to store objects in an ObjectDB database using JPA**.**

Procedure :

Step 1: Create a Java EE Web Project Select **Java Web > Web Application** and click **Next**.Choose a Project Name (e.g. **Guestbook**) and click **Next**.

Step 2 ; To add ObjectDB/JPA support for the project **Right click** the Libraries folder in the [Projects] window and select **Add Jar/Folder...**Select the **objectdb.jar** file from the **bin** subdirectory of the ObjectDB installation directory.

Step 3: Entity Class and Persistence Unit To store objects in an ObjectDB database using JPA we need to define an entity class:Open the [New Entity Class] dialog box, e.g. by right clicking the project node (in the [Projects] window) and selecting **New > Entity Class**... (or **New > Other... > Persistence > Entity Class** and clicking **Next**). click **Finish** to generate a default persistence.xml file with a default persistence unit.

Step 4: : Define an EJB Session Bean Open the [New Session Bean] dialog box by right clicking the guest package node (in the [Projects] window), selecting **New > Other... > Java EE** (or **Enterprise JavaBeans) > Session Bean** and clicking **Next**.

Step 5 : Add a Servlet Class Open the [New Servlet] dialog box by right clicking the guest package node (in the [Projects] window) and selecting **New > Servlet...**Enter **GuestServlet** as the class name - use **exactly** that case sensitive class name.

Step 6: Add a JSP Page Open the [New JSP File] dialog box by right clicking the Web Pages node (in the [Projects] window) and selecting **New > JSP**...Enter **guest** as the jsp file name - use **exactly** that case sensitive class name.Click **Finish** to create the new JSP file.

Step 7 : Run the Java EE 6 Application You can run the application now by right clicking the **GuestServlet** node (in the [Projects] window), selecting **Run File**, and then clicking **OK** (no need to change the servlet execution URI).

Program :

Guest.java

```
package guest;

import java.io.Serializable;

import java.sql.Date;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;

@Entity
public class Guest implements Serializable { private
    static final long serialVersionUID = 1L;
    // Persistent Fields:
    @Id @GeneratedValueLong
    id;
    private String name; private
    Date signingDate;
    // Constructors:
    public Guest() {
    }
    public Guest(String name) {
        this.name = name;
        this.signingDate = new Date(System.currentTimeMillis());
    }
    // String Representation:
    @Override
    public String toString() {
        return name + " (signed on " + signingDate + ")";
    }
}
```

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="GuestbookPU" transaction-type="JTA">
    <provider>com.objectdb.jpa.Provider</provider>
```

```
<properties>
 <property name="javax.persistence.jdbc.url"  value="$objectdb/db/guests.odb"/>
 <property name="javax.persistence.jdbc.user"  value="admin"/>
      <property name="javax.persistence.jdbc.password"  value="admin"/>
   </properties>
  </persistence-unit>
 </persistence>
```

GuestDao.java

```
package guest;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
@Stateless
public class GuestDao {
    // Injected database connection:
    @PersistenceContext private EntityManager em;
    // Stores a new guest:
    public void persist(Guest guest) {
        em.persist(guest);
    }
    // Retrieves all the guests:
    public List<Guest> getAllGuests() {
        TypedQuery<Guest> query = em.createQuery(
            "SELECT g FROM Guest g ORDER BY g.id", Guest.class);
        return query.getResultList();
    }
}
```

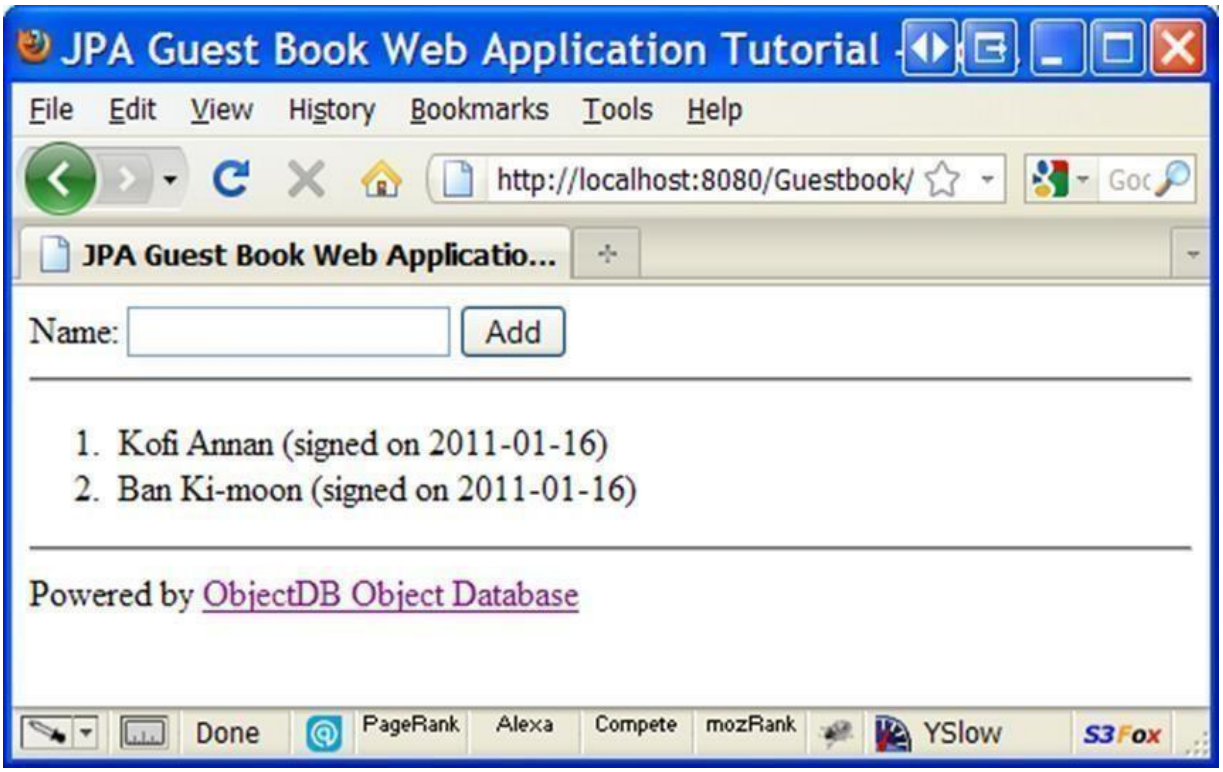
GuestServlet.java

```
package guest;
import java.io.IOException;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
 @WebServlet(name="GuestServlet", urlPatterns={"/guest"})
public class GuestServlet extends HttpServlet {
 private static final long serialVersionUID = 1L;
    // Injected DAO EJB:
    @EJB GuestDao guestDao;
    @Override
    protected void doGet(
        HttpServletRequest request, HttpServletResponse response)
            throws ServletException,  IOException {
         // Display the list of guests:
        request.setAttribute("guests", guestDao.getAllGuests());
        request.getRequestDispatcher("/guest.jsp").forward(request, response);
    }
    @Override
    protected void doPost(
        HttpServletRequest request, HttpServletResponse response)
            throws ServletException,  IOException {
        // Handle a new guest:
        String name = request.getParameter("name");
        if (name != null)
            guestDao.persist(new Guest(name));
        // Displaythe list of guests:
        doGet(request, response);
    }
}
```

Guest.jsp

```
<%@page contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@page import="java.util.*,guest.Guest"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>JPA Guest Book Web Application Tutorial</title>
  </head>
  <body>
    <form method="POST" action="guest"> Name:
      <input type="text" name="name" />
      <input type="submit" value="Add" />
    </form>
    <hr><ol> <%
      @SuppressWarnings("unchecked")
      List<Guest> guests = (List<Guest>)request.getAttribute("guests");if
      (guests != null) {
            for (Guest guest : guests) { %>
              <li> <%= guest %> </li> <%
          }
        } %>
      </ol><hr>
     <iframe src="http://www.objectdb.com/pw.html?jee-netbeans"
      frameborder="0" scrolling="no" width="100%" height="30"> </iframe>
  </body>
</html>
```

OUTPUT :



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 13 **REG.NO :**

DATE: **NAME:**

IMPLEMENTING MVC WITH REQUEST DISPATCHER

AIM:

Develop Login application to implement MVC with Request Dispatcher.

PROCEDURE:

Step 1: Create a Java EE Web Project Select **Java Web > Web Application** and click **Next**.Choose a Project Name (e.g. **EmailLogin**) and click **Next**.

Step 2 : Add a JSP Page Open the [New JSP File] dialog box by right clicking the Web Pages node (in the [Projects] window) and selecting **New > JSP**... for obtaining email, and password and clicks on submit then the action is passed in mvc_servlet where email and password are passed.

Step 3 : Add a Servlet Class Open the [New Servlet] dialog box selecting **New > Servlet...** (e.g. mvc_servlet is controller layer, the request is sent to the bean object which act as model layer.

Step 4: Create a java bean (e.g LoginBean). The email and password values are set into the bean and stored for further purpose.

Step 5 : Add a another JSP Page Open the [New JSP File] dialog box by right clicking the Web Pages node (in the [Projects] window) and selecting **New > JSP**(e.g Success.jsp) From the bean, the value is fetched and shown in the view layer.

PROGRAM:

Index.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Start Page</title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   </head>
```

```
   <body>
      <h1>Hello World!</h1>
 <form action="ControllerServlet" method="post">
Name:<input type="text" name="name"><br>
Password:<input type="password" name="password"><br>
<input type="submit" value="login">
</form></body>
</html>
```

ControllerSerlvet.java

```
package
com.mycompany.mvc1; import
java.io.IOException; import
java.io.PrintWriter;
import
javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ControllerServlet extends
HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse
         response) throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter  out=response.getWriter();
      String
      name=request.getParameter("name");
      String password=request.getParameter("password");
      LoginBean bean=new LoginBean();
      bean.setName(name);
      bean.setPassword(password);
      request.setAttribute("bean",bean);
      boolean status=bean.validate();
```

```
        if(status){
            RequestDispatcher rd=request.getRequestDispatcher("success.jsp");
            rd.forward(request, response);
        }
        else{
            RequestDispatcher rd=request.getRequestDispatcher("error.jsp");
            rd.forward(request, response);
        }
      }
   @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
            resp) throws ServletException, IOException {
        doPost(req, resp);
    }
}
```

LoginBean.java

```
package com.mycompany.mvc1;
public class LoginBean {
private String
name,password; public
String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String
    getPassword() { return
    password;
}
public void setPassword(String password)
    { this.password = password;
```

```
}
public boolean validate(){
   if(password.equals("admin")){
      return true;
   }
   else{
      return false;
   } }  }
```

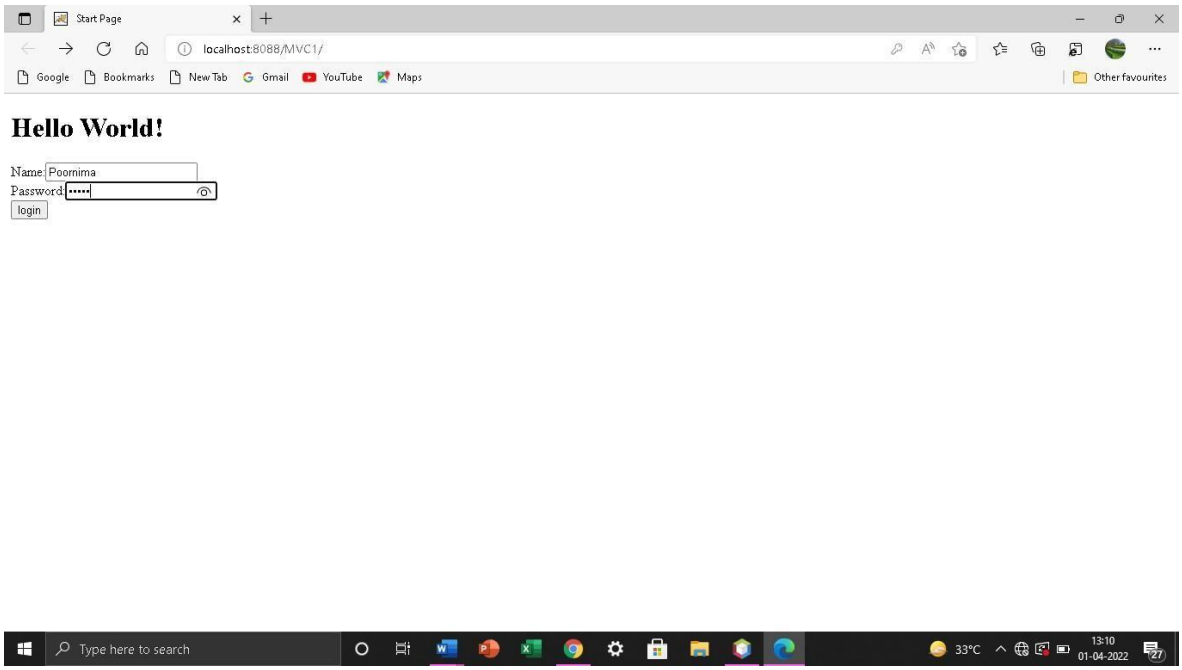
Success.jsp

```
<%@page import="com.mycompany.mvc1.LoginBean"%>
<p>You are successfully logged in!</p>
<%
LoginBean bean=(LoginBean)request.getAttribute("bean");
out.print("Welcome, "+bean.getName());
%>
      <h1>Hello World!</h1>
   </body>
</html>
```

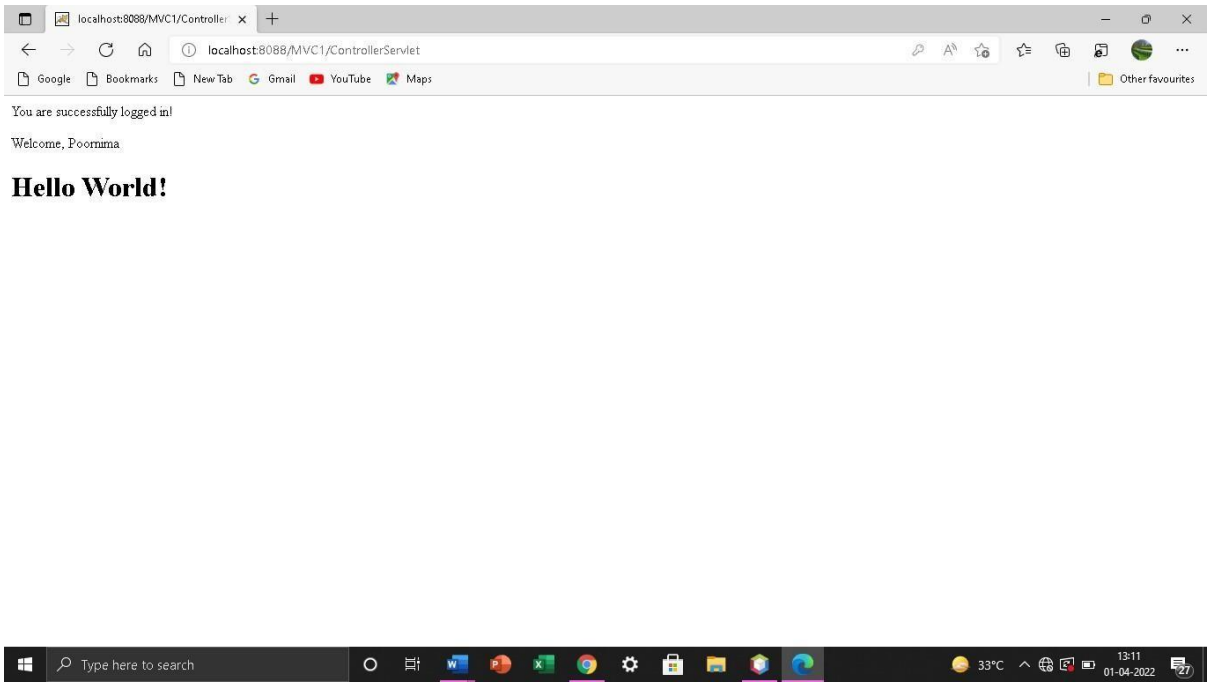
Error.jsp

```
<p>Sorry! username or password error</p>
<%@ include file="index.jsp" %>
```

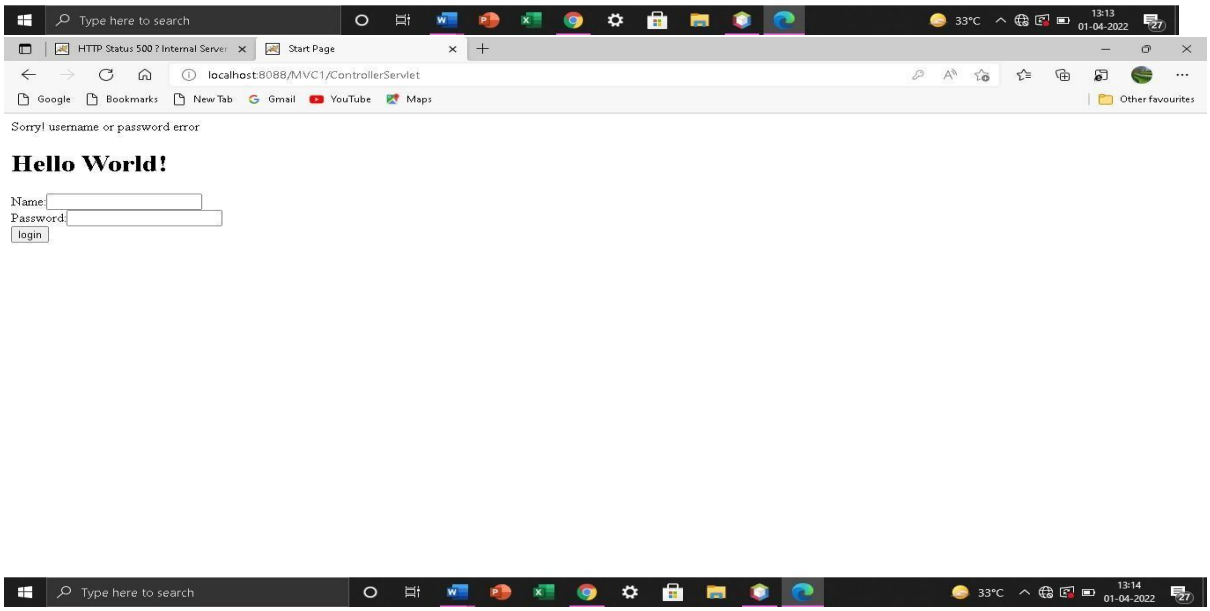
Output:


Start Page
localhost:8088/MVC1/
Google
Bookmarks
New Tab
Gmail
YouTube
Maps
Other favourites
Hello World!
Name: Poornima
Password:
login

Type here to search
33°C
13:10
01-04-2022


localhost:8088/MVC1/Controller
localhost:8088/MVC1/ControllerServlet
Google
Bookmarks
New Tab
Gmail
YouTube
Maps
Other favourites
You are successfully logged in!
Welcome, Poornima
Hello World!

Type here to search
33°C
13:11
01-04-2022



RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 14 **REG.NO :**

DATE: **NAME:**

TO BUILD A WEB APPLICATION USING STRUCTS

Aim:

To build a web application that collects the user's name and displays "Hello World" followed by the user name

Procedure:

Step 1 : Create The Model Class MessageStore.java

Step 2 : Create The Action Class HelloWorldAction.java

Step 3 - Create The View HelloWorld.jsp

Step 4 - Add The Struts Configuration In struts.xml

Step 5 - Create The URL Action

Step 6 - Build the WAR File and Run The Application

Program:

MessageStore.java

```
package

org.apache.struts.helloworld.model; public

class MessageStore {
   private String message;

   public MessageStore() {
      message = "Hello Struts User";
   }

   public String getMessage()
      { return message;
   }
}
```

HelloWorldAction.java

```
package org.apache.struts.helloworld.action;

import org.apache.struts.helloworld.model.MessageStore;

import com.opensymphony.xwork2.ActionSupport;
```

```
public class HelloWorldAction extends
   ActionSupport { private MessageStore
   messageStore;

   public String execute() {
      messageStore = new MessageStore() ;


      return SUCCESS;
   }


   public MessageStore getMessageStore()
      { return messageStore;
   }
}
```

HelloWorld.jsp

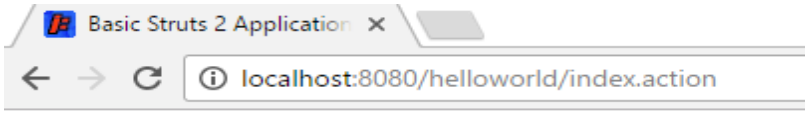
```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Hello World!</title>
 </head>
 <body>
   <h2><s:property value="messageStore.message" /></h2>
 </body>
</html>
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
                    "-//Apache Software Foundation//DTD Struts Configuration
                    2.5//EN" "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
   <constant name="struts.devMode" value="true" />

   <package name="basicstruts2" extends="struts-default">
      <action name="index">
         <result>/index.jsp</result>
      </action>

      <action name="hello" class="org.apache.struts.helloworld.action.HelloWorldAction"
      method="execute">
         <result name="success">/HelloWorld.jsp</result>
     </action>
  </package>
</struts>
```

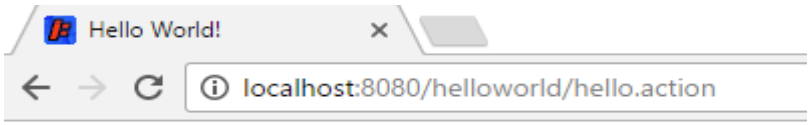
index.jsp

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Basic Struts 2 Application - Welcome</title>
   </head>
   <body>
      <h1>Welcome To Struts 2!</h1>
      <p><a href="<s:url action='hello'/>">Hello World</a></p>
   </body>
</html>
```



Welcome To Struts 2!

Hello World



Hello Struts User

RESULT:

Thus the program has been executed successfully and output verified.

EX.NO: 15 **REG.NO :**

DATE: **NAME:**

TO UPLOAD A SELECTED FILE USING STRUCTS

Aim:

Create our view which will be required to browse and upload a selected file.

Procedure:

Step 1: Create an index.jsp with plain HTML upload form that allows the user to upload a file

Step 2: Create a simple jsp file success.jsp to display the outcome of our file upload in case it becomes success.

Step 3: Create error.jsp in case there is some error in uploading the file

Step 4: Create a Java class called uploadFile.java which will take care of uploading file and storing that file at a secure location

Step 5: Add The Struts Configuration in struts.xml

Step 6: Add <interceptor-ref> tag inside <action> in struts.xml

Step 7: Create The URL Action

Step 8: Build the WAR File and Run The Application

PROGRAM:

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
  pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>File Upload</title>
  </head>
```

```
    <body>
    <form action = "upload" method = "post" enctype = "multipart/form-data">
      <label for = "myFile">Upload your file</label>
      <input type = "file" name = "myFile" />
      <input type = "submit" value = "Upload"/>
    </form>
  </body>
</html>
```

success.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
  <head>
    <title>File Upload Success</title>
  </head>
    <body>
    You have successfully uploaded <s:property value = "myFileFileName"/>
  </body>
</html>
```

error.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
  <head>
    <title>File Upload Error</title>
  </head>
    <body>
    There has been an error in uploading the file.
  </body>
</html>
```

uploadFile.java

```
package com.srm.struts2;
import java.io.File;
import org.apache.commons.io.FileUtils;
import java.io.IOException;
import com.opensymphony.xwork2.ActionSupport;
public class uploadFile extends ActionSupport {
  private File myFile;
  private String myFileContentType;
  private String myFileFileName;
  private String destPath;
  public String execute() {
    /* Copy file to a safe location */
    destPath = "C:/apache-tomcat-6.0.33/work/";
    try {
      System.out.println("Src File name: " + myFile);
      System.out.println("Dst File name: " + myFileFileName);

      File destFile  = new File(destPath, myFileFileName);
      FileUtils.copyFile(myFile, destFile);
     }
       catch(IOException e) {
      e.printStackTrace();
      return ERROR;
    }
    return SUCCESS;
  }
  public File getMyFile() {
    return myFile;
  }
```

```
  public void setMyFile(File myFile) {
    this.myFile = myFile;
  }
    public String getMyFileContentType() {
    return myFileContentType;
  }
    public void setMyFileContentType(String myFileContentType) {
    this.myFileContentType = myFileContentType;
  }
  public String getMyFileFileName() {
    return myFileFileName;
  }
    public void setMyFileFileName(String myFileFileName) {
    this.myFileFileName = myFileFileName;
  }
}
```

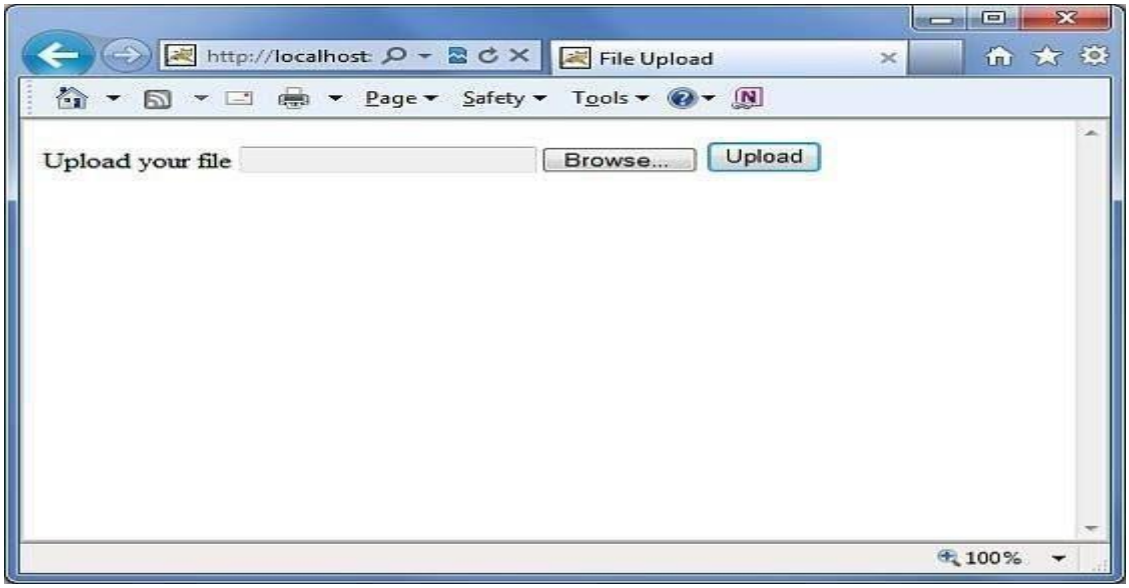
struts.xml

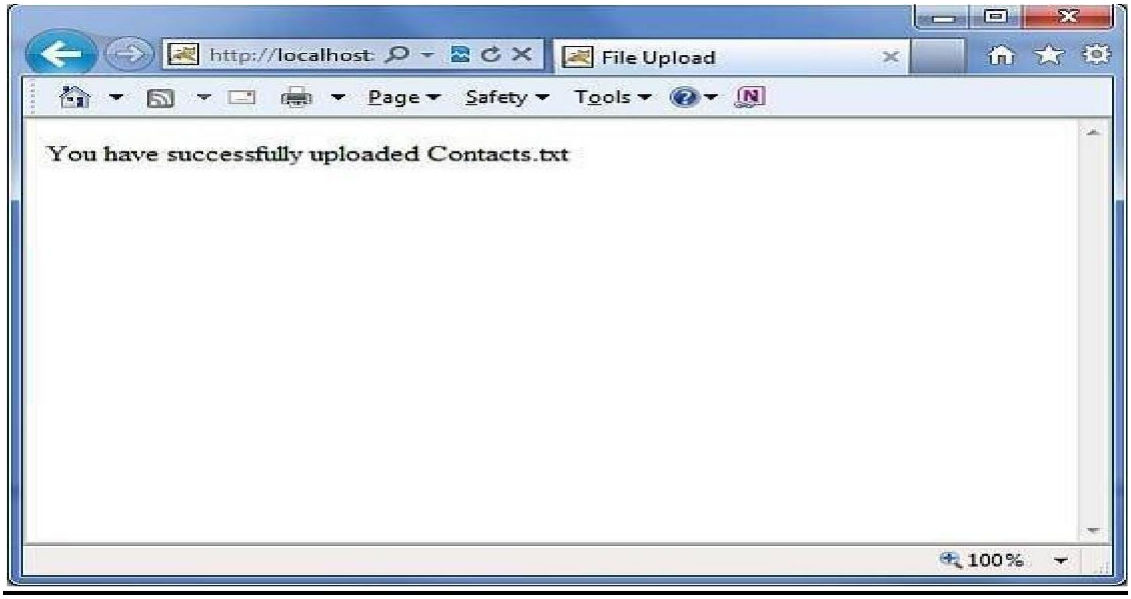
```
<?xml version = "1.0" Encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name = "struts.devMode" value = "true" />
  <constant name = "struts.multipart.maxSize" value = "1000000" />
  <package name = "helloworld" extends = "struts-default">
    <action name = "upload" class = "com.srm.struts2.uploadFile">
      <result name = "success">/success.jsp</result>
      <result name = "error">/error.jsp</result>
    </action>
  </package>
</struts>
```

Web.xml

```
<?xml version = "1.0" Encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns = "http://java.sun.com/xml/ns/javaee"
  xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id = "WebApp_ID" version = "3.0">
  <display-name>Struts  2</display-name>
    <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Output:





RESULT:

Thus the program has been executed successfully and output verified.