DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Re-Accredited by NAAC with 'A' Grade
Accredited by NBA for AERO, BME, CSE, ECE, EEE, IT & MECH.
PERAMBALUR-621212, TAMILNADU, INDIA.
Website: www.dsengg.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

U23ITP43/ WEB TECHNOLOGY

LABORATORY

II YEAR / IV SEMESTER

REGUALTION: 2023

Prepared By

Mrs.S.Abirami

AP/IT

Approved By

HOD / IT

LIST OF EXPERIMENTS

1. Create a web page with the following using HTML.
    - To embed an image map in a web page.
    - To fix the hot spots.
    - Show all the related information when the hot spots are clicked
2. Create a web page with all types of Cascading style sheets.
3. Client-Side Scripts for Validating Web Form Controls using DHTML.
4. Installation of Apache Tomcat web server.
5. Write programs in Java using Servlets:
    - To invoke servlets from HTML forms.
    - Session Tracking.
6. Write programs in Java to create three-tier applications using JSP and Databases
    - For conducting on-line examination.
    - For displaying student mark list. Assume that student information is available in a database which has been stored in a database server.
7. Programs using XML – Schema – XSLT/XSL.
8. Programs using DOM and SAX parsers.
9. Programs using AJAX.
10. Consider a case where we have two web Services- an airline service and a travel agent and the travel agent is searching for an airline. Implement this scenario using Web Services and Database.

TOTAL: 60 PERIODS

Ex. No:1 **IMAGE MAPPING IN HTML**
Date :

AIM:

To create a web page which includes a map and display the related information when a hot spot is clicked in the map.

PROCEDURE:

1. Create a html file with map tag.
2. Set the source attribute of the img tag to the location of the image and also set the use map attribute.
3. Specify an area with name, shape and href set to the appropriate values.
4. Repeat step 3 as many hot spots you want to put in the map.
5. Create html files for each and every hot spot the user will select.

PROGRAM:

ImageMap.html

```
<HTML>
<HEAD>
<TITLE>Image Map</TITLE> </HEAD>
<BODY>
<img src="india_map.jpg" usemap="#metroid" ismap="ismap" > <map name="metroid"
id="metroid">
<area href="TamilNadu.html" shape="circle" coords="208,606,50" title="TamilNadu"/>
<area href="Karnataka.html" shape="rect" coords = "130,531,164,535" title ="Karnataka" />
<area href="AndhraPradesh.html" shape="poly" coords =
"227,490,238,511,230,536,198,535,202,503" title ="Andhra Pradesh" />
<area href="Kerala.html" shape="rect" coords = "154,606,166,621" title ="Kerala" /> </map>
</BODY>
</HTML>
```

TamilNadu.html

```
<HTML><HEAD>
<TITLE>About Tamil Nadu</TITLE>
</HEAD>
<BODY>
<CENTER><H1>Tamil Nadu</H1></CENTER> <HR>
<UL>
<LI>Area : 1,30,058 Sq. Kms.</LI>
<LI>Capital : Chennai</LI>
<LI>Language : Tamil</LI>
<LI>Population : 6,21,10,839</LI> </UL><hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

Karnataka.html

```
<HTML>
<HEAD>
<TITLE>About Karnataka</TITLE> </HEAD>
```

```
<BODY>
<CENTER><H1>Karnataka</H1></CENTER>
<HR> 5
<UL>
<LI>Area : 1,91,791 Sq. Kms</LI>
<LI>Capital : Bangalore</LI>
<LI>Language : Kannada</LI>
<LI>Population : 5,27,33,958</LI>
</UL>
<hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

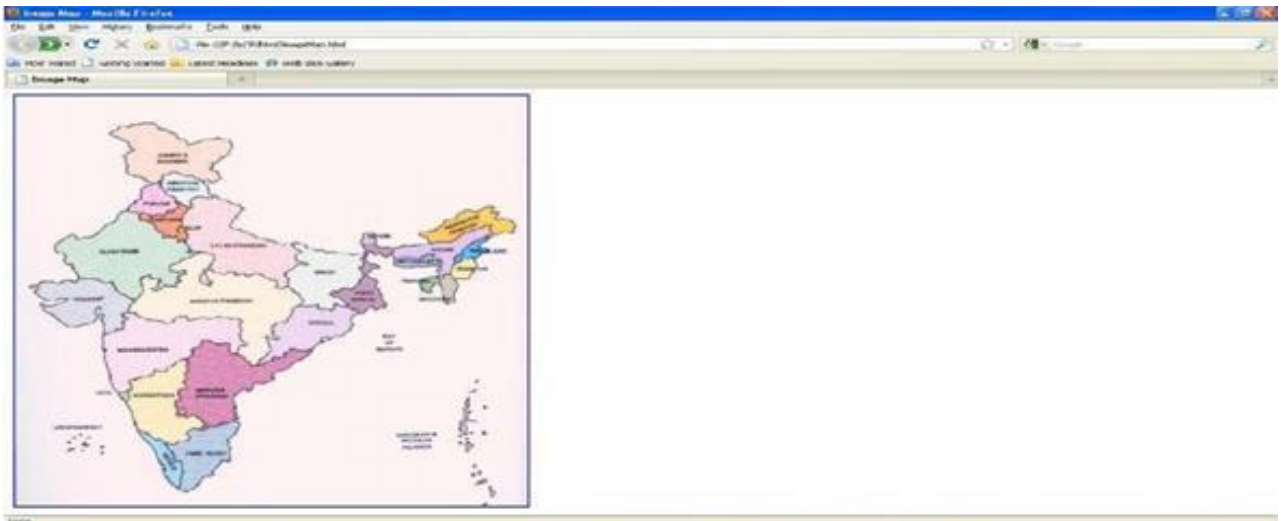
AndhraPradesh.html

```
<HTML>
<HEAD>
<TITLE>About Andhra Pradesh</TITLE> </HEAD>
<BODY>
<CENTER><H1>Andhra Pradesh</H1></CENTER> <HR>
<UL>
<LI>Area : 2,75,068 Sq. Kms</LI>
<LI>Capital : Hyderabad</LI>
<LI>Language : Telugu</LI>
<LI>Population : 7,57,27,541</LI>
</UL>
<hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

Kerala.html

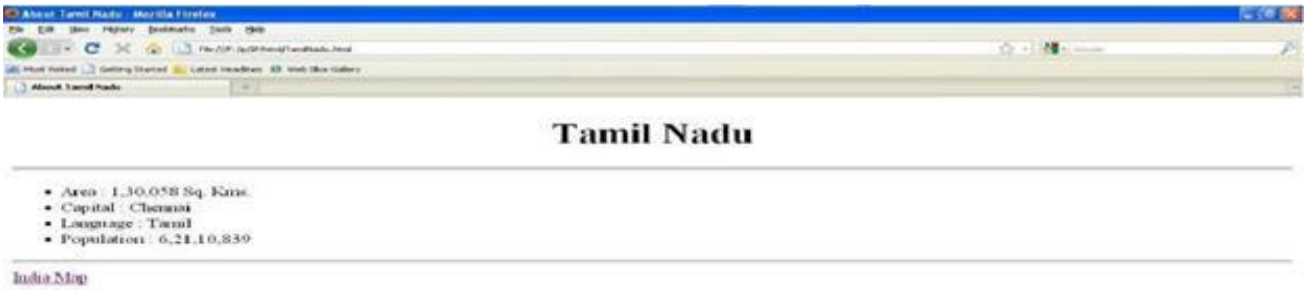
```
<HTML>
<HEAD>
<TITLE>About Kerala</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>Kerala</H1></CENTER>
<HR>
<UL>
<LI>Area : 38,863 Sq. Kms.</LI>
<LI>Capital : Thiruvananthapuram</LI>
<LI>Language : Malayalam</LI>
<LI>Population : 3,18,38,619</LI>
</UL>
<hr>
<a href='ImageMap.html'>India Map</a>
</BODY></HTML> 6
```

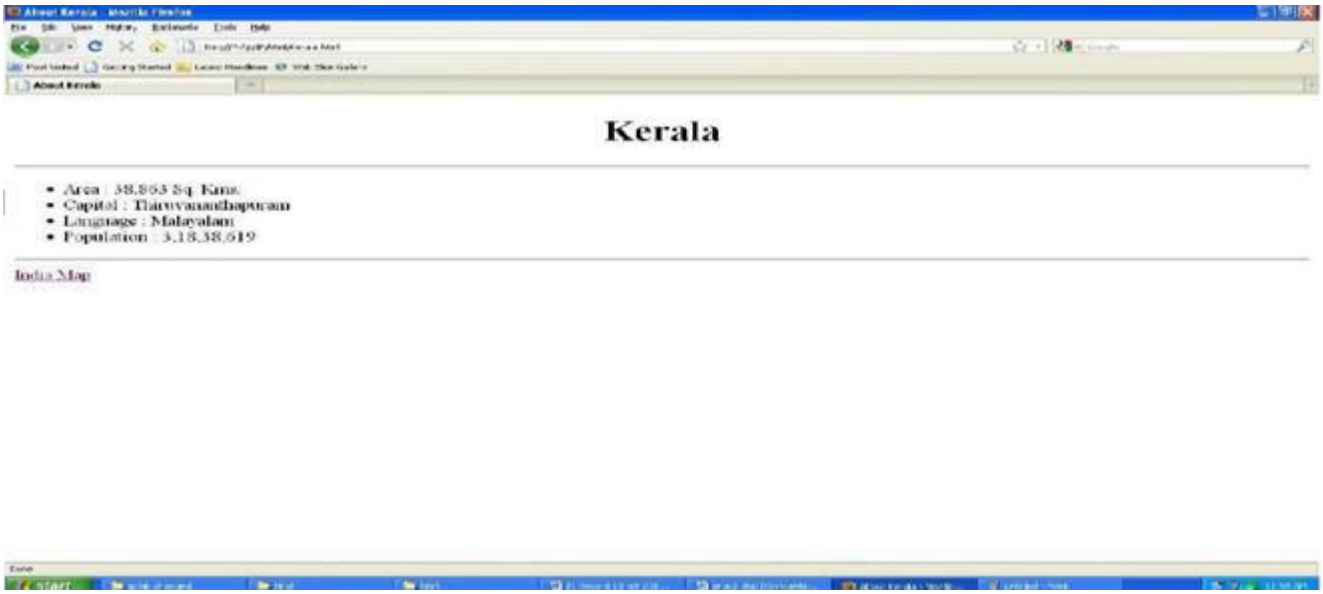
OUTPUT:




Andhra Pradesh
Area : 2,75,068 Sq Kms
Capital : Hyderabad
Language : Telugu
Population : 7,57,27,541
India Map


Karnataka
Area : 1,91,791 Sq Kms
Capital : Bangalore
Language : Kannada
Population : 5,27,33,958
India Map





RESULT:

Thus the creation of a web page includes a map and display the related information when a hot spot is clicked in the map was successfully executed and verified.

Ex. No: 2 **STYLE SHEETS**
Date :

AIM:

To create a web page that displays college information using various style sheet

PROCEDURE:

1. Create a web page with frame sets consisting two frames
2. In the first frame include the links
3. In the second frame set display the web page of the link
4. Create a external style sheets
5. Create a embedded style sheets
6. Create a inline and internal style sheets and make it link to the external style sheets

PROGRAM:

XYZ.CSS:

```
h3
{
font-family:arial;
font-size:20;
color:cyan
}
table{
border-color:green
}
td
{
font-size:20pt;
color:magenta
}
```

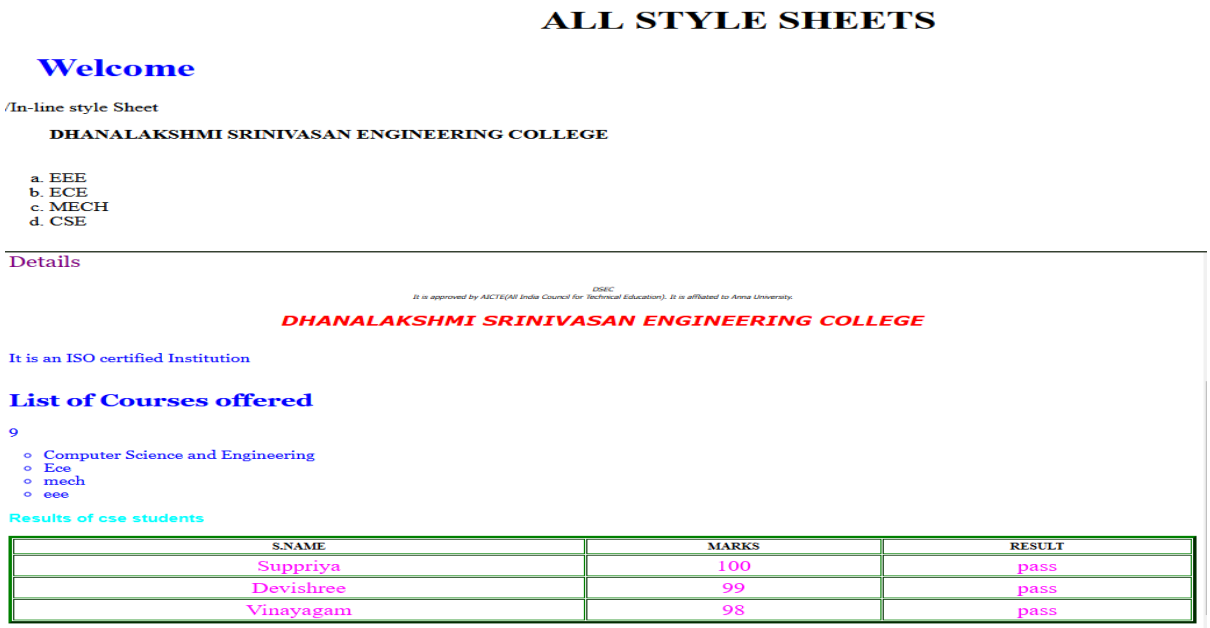
HTML CODE:

```
<html>
<head>
<h1>
<center>ALL STYLE SHEETS</center>
</h1>
<title>USE of INTERNAL and EXTERNAL STYLESHEETS </title>
<link rel="stylesheet" href="xyz.css" type="text/css">
<style type="text/css">
.vid
{
font-family:verdana;
font-style:italic;
color:red;
text-align:center
}
.ani
{
```

```
font-family:tahoma;
font-style:italic; 8
font-size:20;
text-align:center;
}
font
{
font-family:georgia;
color:blue;
font-size:20
}
ul
{
list-style-type:circle
}
p
{
font-family: georgia, serif;
font-size: x-small;
}
hr
{
color: #ff9900; height: 1px
}
a:hover
{
color: #ff0000;
text-decoration: none
}
</style>
</head>
<body>
<h1 style="color:blue;margin-left:30px;">Welcome</h1> //In-line style Sheet
<ol style="list-style-type:lower-alpha">
<b>St.Anne’s College of Engineering and Technology </b>
<br>
<br>
<br>
<li> EEE</li>
<li> ECE </li>
<li> MECH</li>
<li> CSE</li>
</ol>
<p style="font-size:20pt;color:purple">Details</p>
<p class="ani">St.Anne’s College <br>It is approved by AICTE(All India Council for Technical
Education). It is affliated to Anna University.<br></p>
<h2 class="vid"> St.Anne’s college of engineering </h2> <br>
<font>It is an ISO certified Institution
</font>
<br>
<br>
```

```
<font>
<h2>List of Courses offered</h2> 9
<ul>
<li>Computer Science and Engineering</li>
<li>Ece</li>
<li>mech</li>
<li>eee</li>
</ul>
</font>
<h3>Results of cse students</h3>
<table width="100%" cellspacing="2" cellpadding="2" border="5"> <tr>
<th>S.NAME</th> <th>MARKS</th> <th>RESULT</th>
</tr>
<tr>
<td align="center">Suppriya</td> <td align="center">100</td>
<td align="center">pass</td>
</tr>
<tr>
<td align="center">Devishree</td> <td align="center">99</td>
<td align="center">pass</td>
</tr>
<tr>
<td align="center">Vinayagam</td> <td align="center">98</td>
<td align="center">pass</td> </tr>
</table>
</body>
</html>
```

OUTPUT:



ALL STYLE SHEETS

Welcome

/In-line style Sheet

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

a. EEE
b. ECE
c. MECH
d. CSE

Details

DSEC

It is approved by AICTE(All India Council for Technical Education). It is affiliated to Anna University.

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

It is an ISO certified Institution

List of Courses offered

9

- Computer Science and Engineering
- Ece
- mech
- eee

Results of cse students

S.NAME	MARKS	RESULT
Suppriya	100	pass
Devishree	99	pass
Vinayagam	98	pass

RESULT:

Thus the creation of a web page that displays college information using various style sheet was successfully executed and verified.

EX.NO:3 CLIENT-SIDE SCRIPTS FOR VALIDATING WEB FORM CONTROLS USING DHTML

AIM:

To write Client Side Scripts for Validating Web Form Controls using DHTML

ALGORITHM

STEP 1: Draw Form design using <form> tag.
STEP 2: Write function program to validate all the fields in the form.
STEP 3: Call the function program in the required place

PROGRAM:

```
<html>
<head>
<title>A Simple Form with JavaScript Validation</title>
<script type="text/javascript">
<!--
function validate_form ( )
{
valid = true;
if ( document.contact_form.contact_name.value == "" )
{
alert ( "Please fill in the 'Your Name' box." );
valid = false;
}
return valid;
}
//-->
</script>
</head>
<body bgcolor="#FFFFFF">
<form name="contact_form" method="post"
action="/cgi-bin/articles/development/javascript/form-validation-with-javascript/contact_simple.cgi"
onSubmit="return validate_form ( );">
<h1>Please Enter Your Name</h1>
<p>Your Name: <input type="text" name="contact_name"></p>
<p><input type="submit" name="send" value="Send Details"></p>
</form>
</body>
</html>
(Or)
<form onsubmit="return validateForm ( this )" ...>
... form control definitions here ...
</form>
<script language="JavaSCript">
<!--
function validateForm ( form ) {
// check form fields and
```

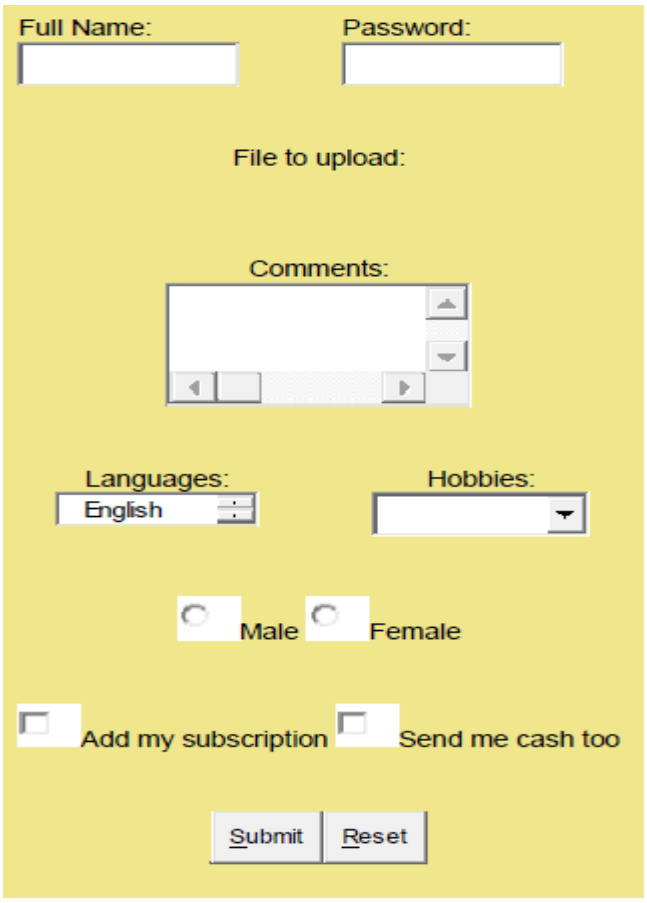
```
// return false if not validated
// return true otherwise
}
//-->
</script>
```

When the submit button is clicked, the function loops through the form elements collection and checks for:

```
<script language="JavaScript">
<!--
function validateForm ( form ) {
for ( var e = 0; e < form.elements.length; e ++ ) {
var el = form.elements [ e ];
if ( el.type == 'text' || el.type == 'textarea' ||
el.type == 'password' || el.type == 'file' ) {
// check text fields
if ( el.value == '' ) {
alert ( 'Please fill out the ' + el.name + ' field' );
el.focus ( );
return false;
}
}
else if ( el.type.indexOf ( 'select' ) != -1 ) {
// check selectable dropdown menus
if ( el.selectedIndex == -1 ) {
alert ( 'Please select a value for ' + el.name );
el.focus ( );
return false;
}
}
else if ( el.type == 'radio' ) {
// check radio button groups
var group = form [ el.name ];
var checked = false;
if ( !group.length ) checked = el.checked;
else
for ( var r = 0; r < group.length; r ++ )
if ( ( checked = group [ r ].checked ) )
break;
if ( !checked ) {
alert ( 'Please check one of the ' +
el.name + ' buttons' );
el.focus ( );
return false;
}
}
else if ( el.type == 'checkbox' ) {
// check checkbox groups
var group = form [ el.name ];
if ( group.length ) {
var checked = false;
for ( var r = 0; r < group.length; r ++ )
```

```
if ( ( checked = group [ r ].checked ) )
break;
if ( !checked ) {
alert ( 'Please check one of the ' +
el.name + ' checkboxes' );
el.focus ( );
return false;
}
}
}
}
return true;
}
//-->
</script>
```

OUTPUT:



RESULT:

The following example contains a generic function that may be used as a general handler for checking whether all the fields in a form have been filled out.

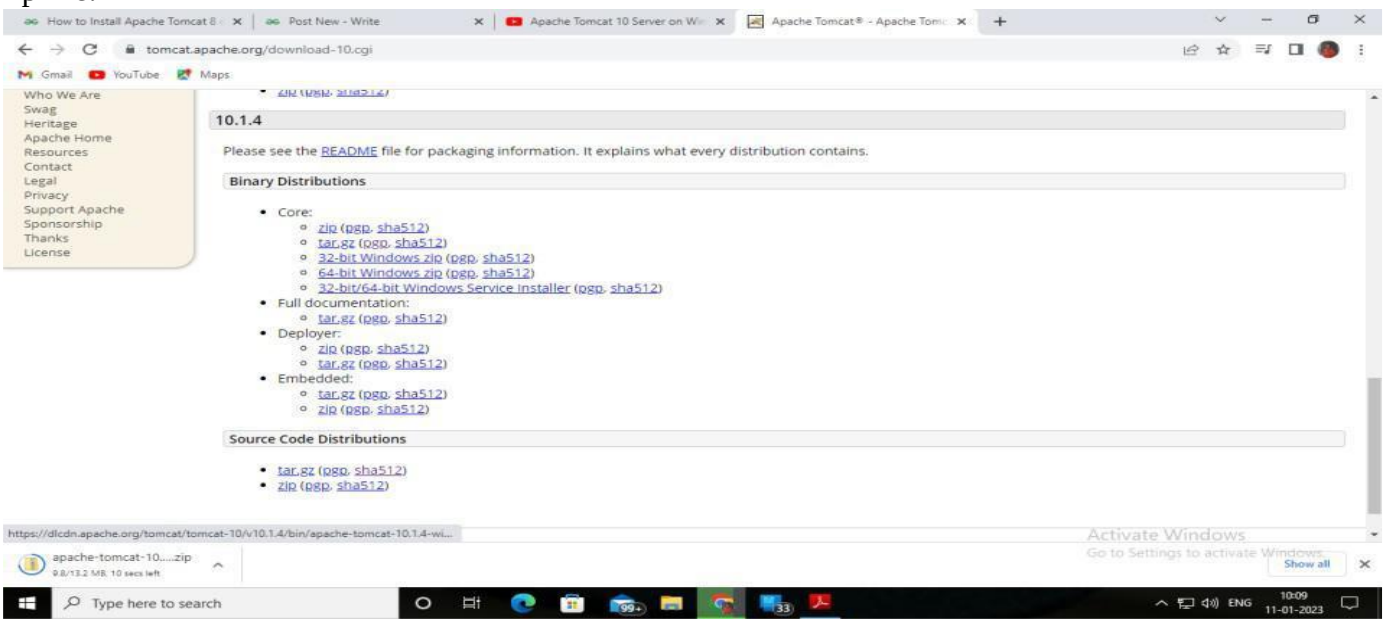
EX.NO:4 INSTALLATION OF APACHE TOMCAT WEB SERVER

AIM:

Installation of apache tomcat web server

ALGORITHM:

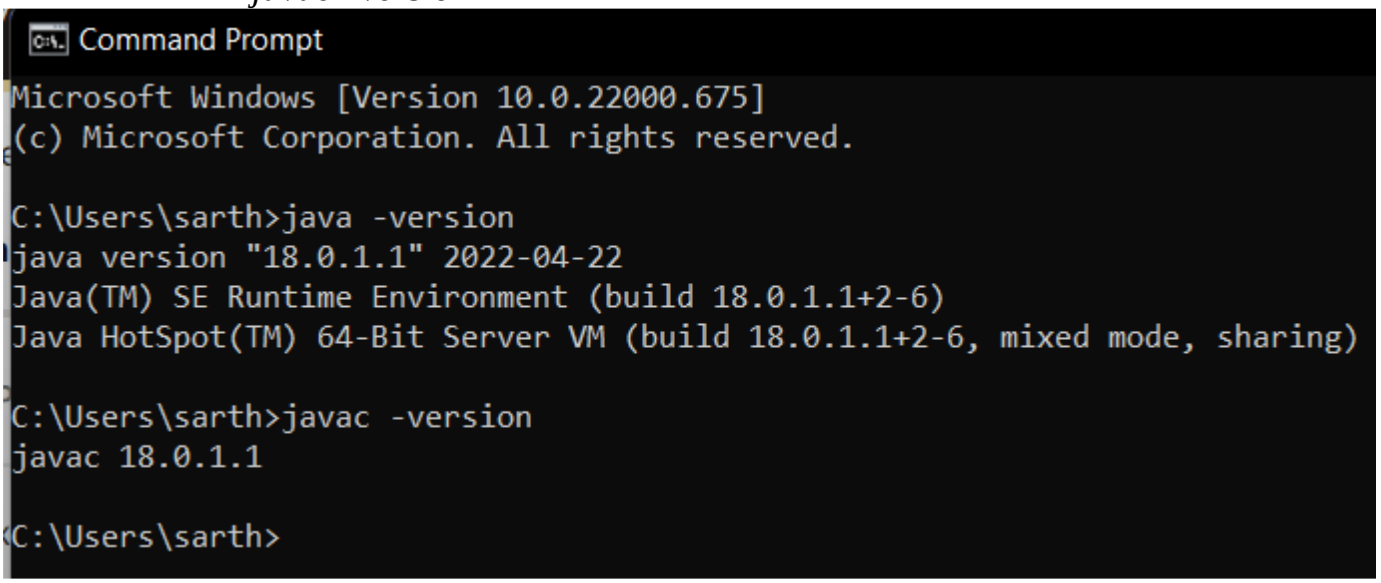
STEP 1: We need to first install the Tomcat 10 zip file from this website. On the website, select the 64-bit Windows zip (PGP, sha512) in the Core section to start the download process for the Tomcat zip file.



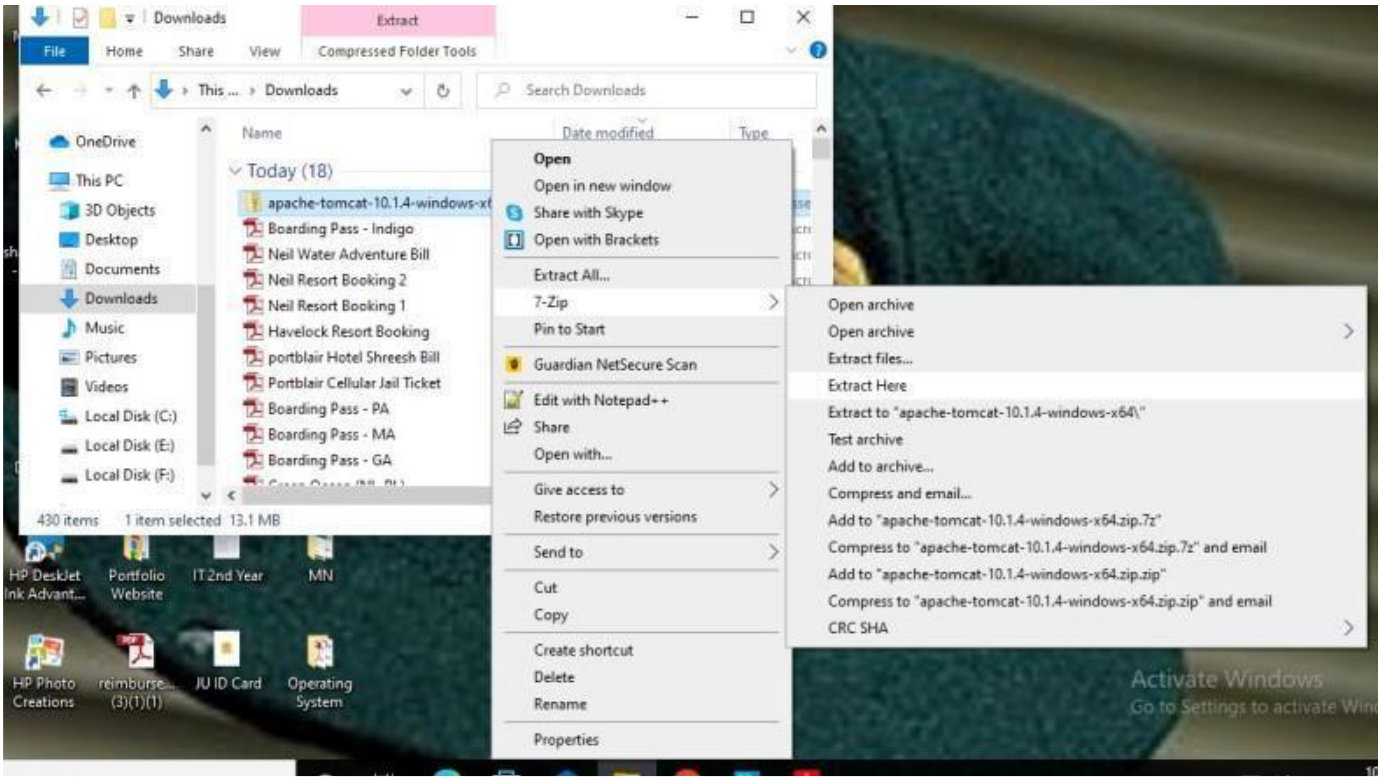
STEP 2: Check If JDK Is Installed**.** Open Command Prompt and enter the following commands

java-version

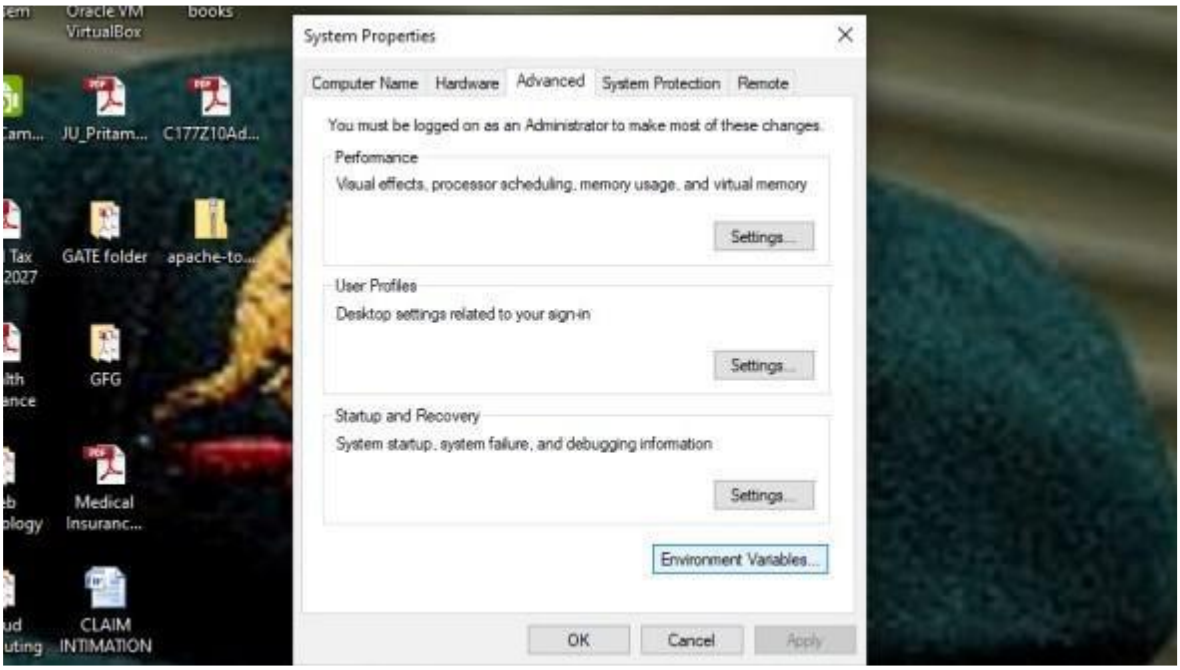
javac –version



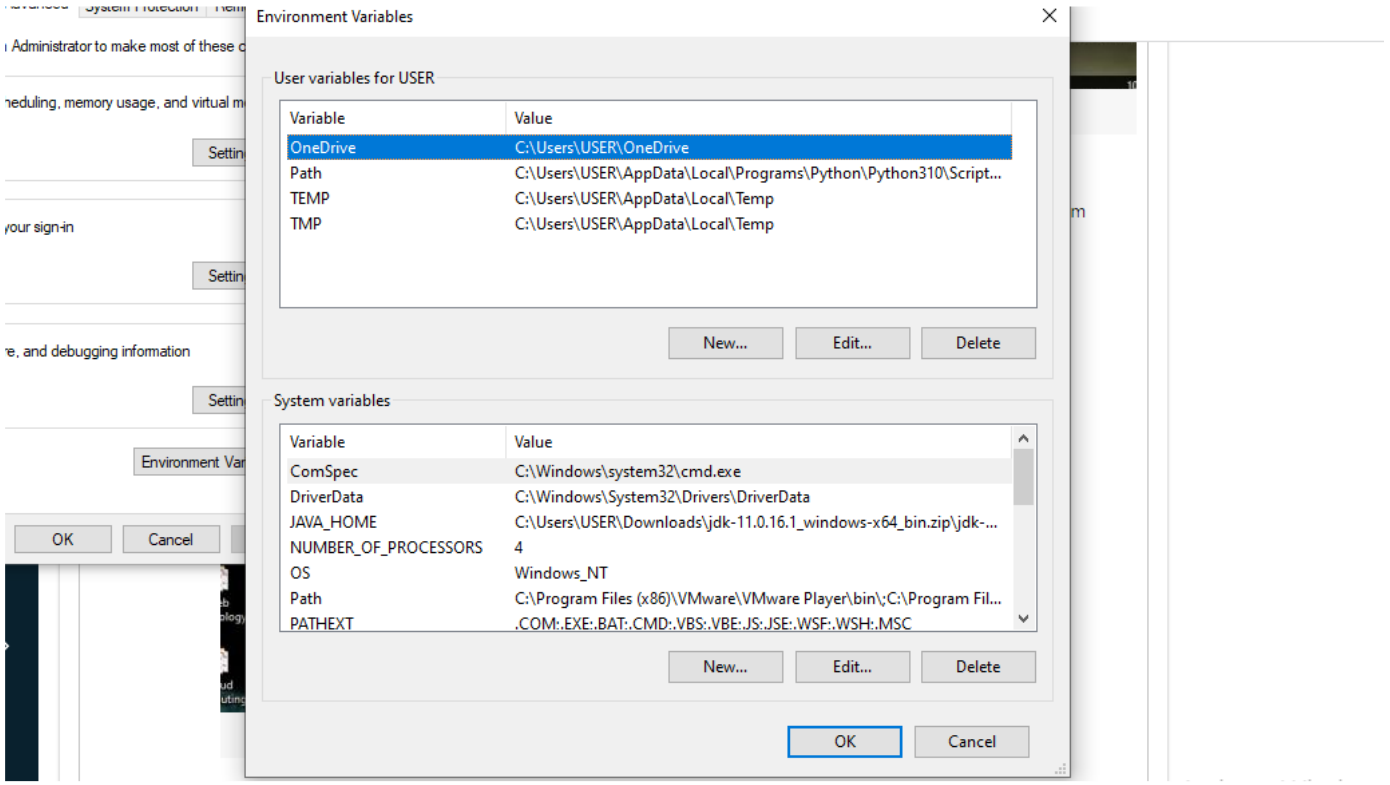
STEP 3: Unzip Tomcat 10 Zip File**.** Go to the location where you have downloaded the Tomcat 10 zip file. Right-click on the apache tomcat file place the cursor on 7-Zip and click on Extract Here to extract the folder.



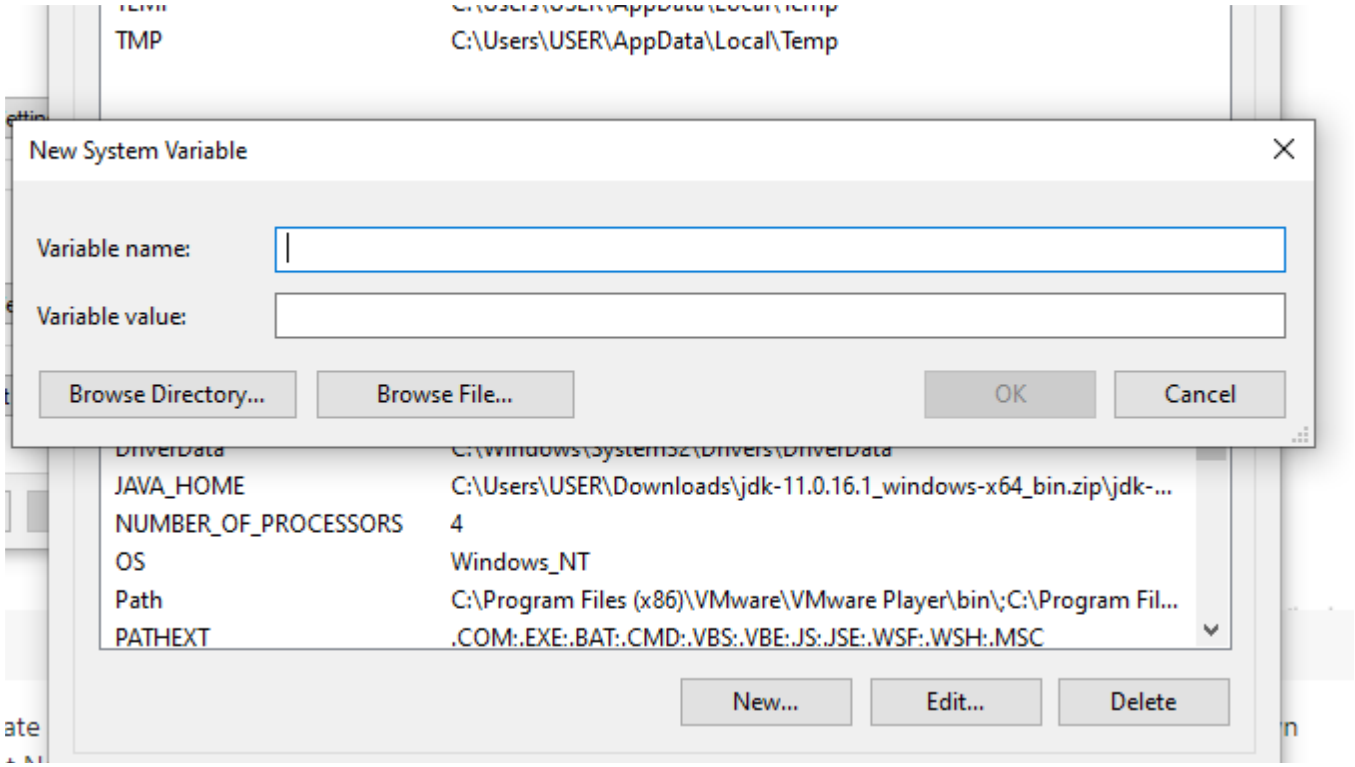
STEP 4: Creating JAVA_HOME Variable**.** Click Start then in the search bar, search for "Edit the system environment variables" and click on it. The following System Properties box will open. Select Environment Variables in the box.



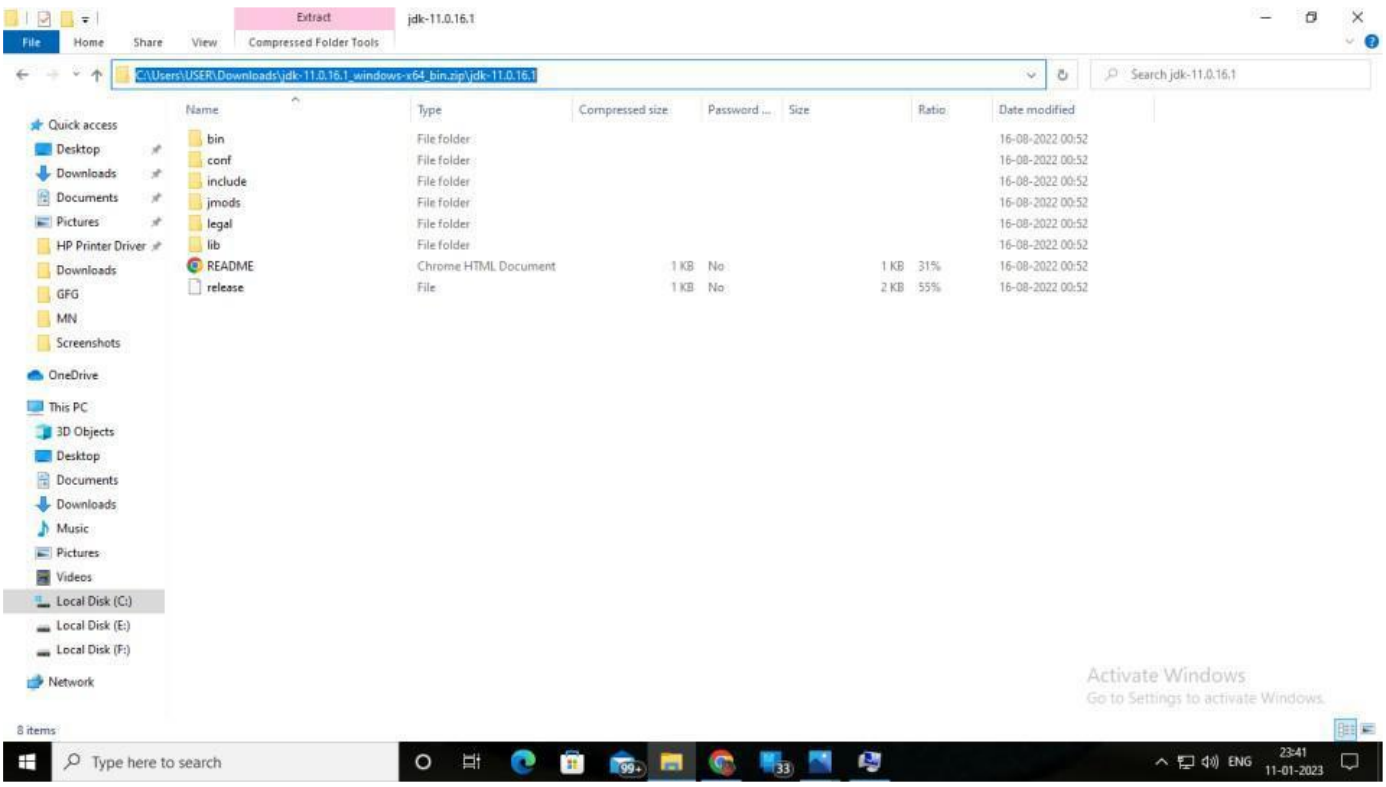
On clicking Environment Variables the following box will open.



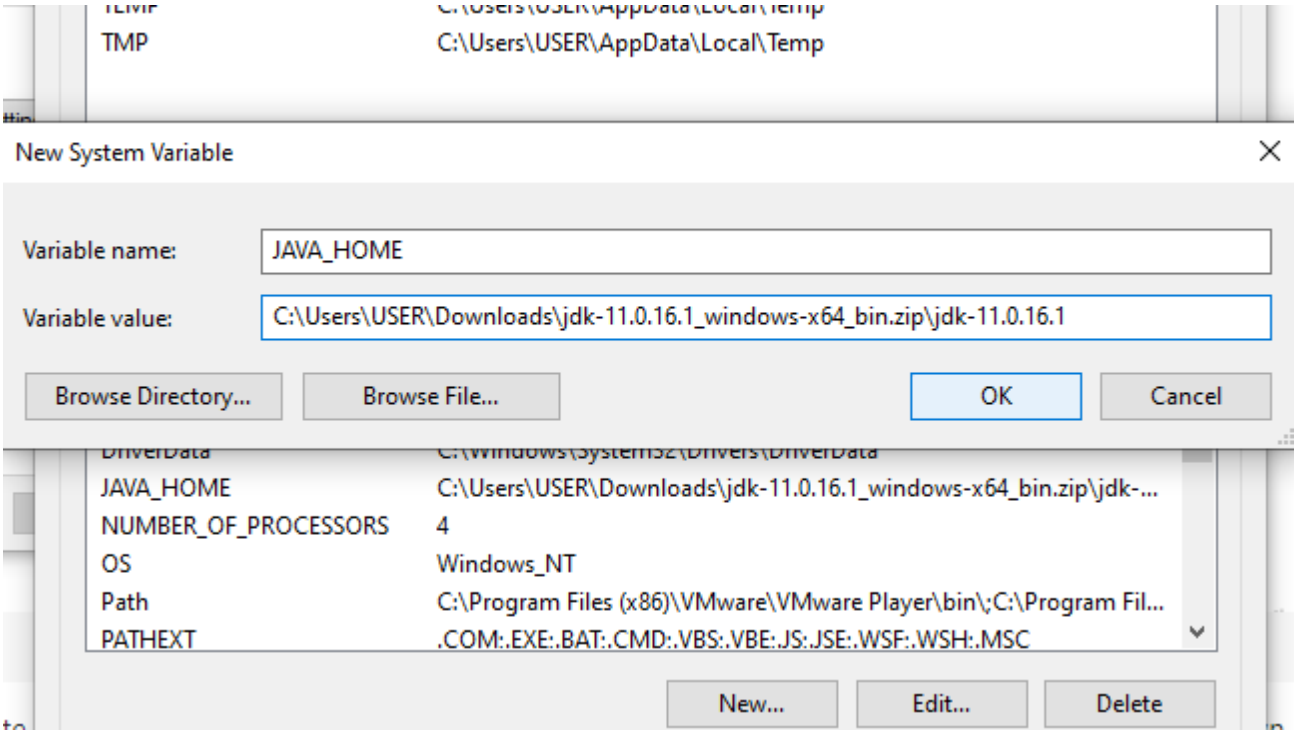
STEP 5: We have to create a JAVA_HOME variable and insert the path of the JDK file stored in our computer, which will be shown below. So select New from the System variables in the above picture. A New System Variable box will be opened where we will have to fill in the Variable name and Variable value.



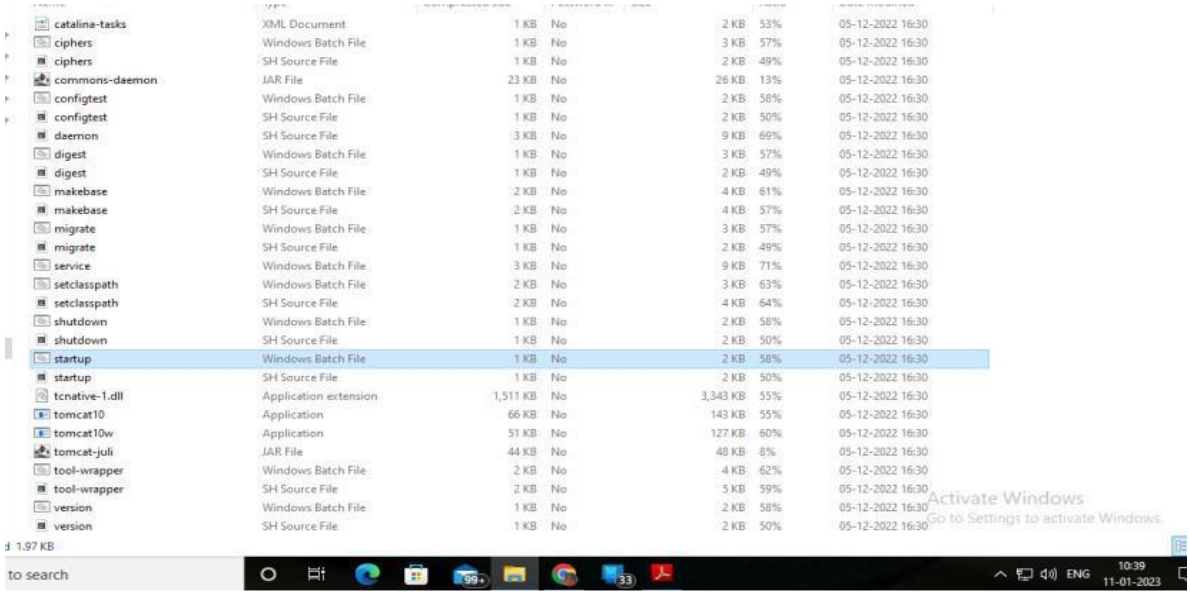
STEP 6: Go to the location where you have stored the contents of the JDK file in My Computer or PC. Copy the root path of the location of the JDK file as shown below.



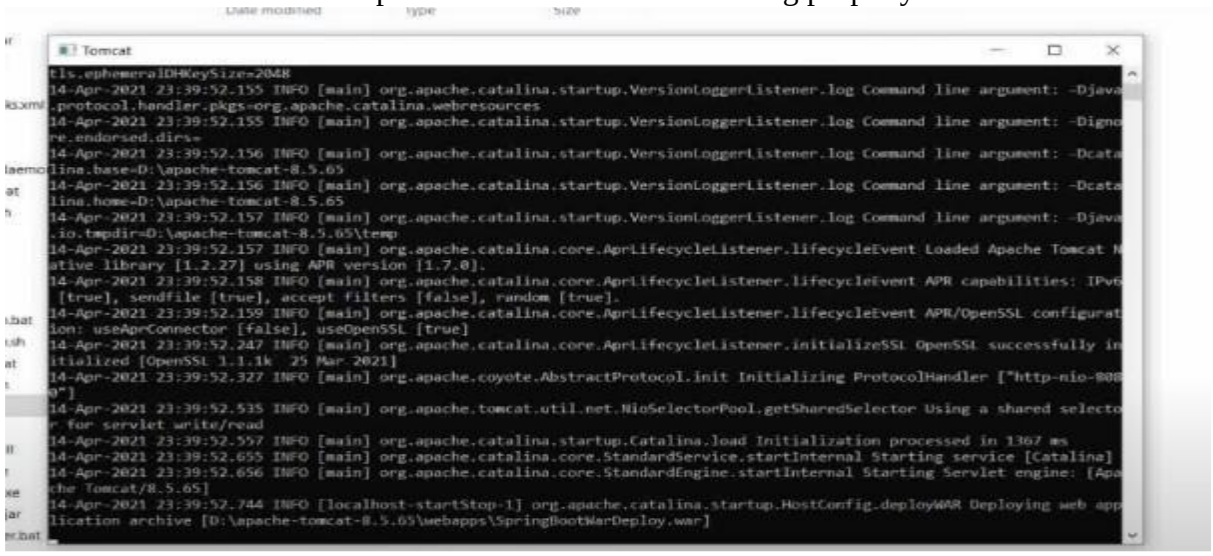
STEP 7: Paste the JDK path from the above picture into the Variable value field and in the Variable name field, give the name JAVA_HOME as shown below



STEP 8: Check the Working of Tomcat**.** Open the extracted apache tomcat file. We will see all the following files in them. Among them open the batch file named



STEP 9: Check if the Startup Windows Batch File is running properly



By default, the server port number is 8080.

STEP 10: Successful Installation of Tomcat**.** Our next step is to open an internet browser like Google and type in the address http://localhost:8080 Then click enter and the following browser will appear suggesting the successful installation of ApacheTomcat

.

RESULT:

Thus Apache Tomcat server were installed successfully**.**

Ex.No:5a **INVOKING SERVLETS FROM HTML FORM**
Date:

AIM:

To write a java program to invoke servlets from HTML form.

PROCEDURE:

client.html:

(1) Create a web page using HTML form that contains the fields such as text, password and one submit button.
(2) Set the URL of the server as the value of form's action attribute.
(3) Run the HTML program.
(4) Submit the form data to the server.

server.java:

(1) Define the class server that extends the property of the class HttpServlet
(2) Handle the request from the client by using the method service() of HttpServlet class.
(3) Get the parameter names from the HTML form by using the method getParameterNames().
(4) Get the parameter values from the HTML forms by using the method getParameter().
(5) Send the response to the client by using the method of PrintWriter class.

PROGRAM:

MySrv.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 MySrv extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
out.println("<HTML>");
out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");
out.println(" <BODY>");
//Getting HTML parameters from Servlet
String username=request.getParameter("uname");
String password=request.getParameter("pwd");
if((username.equals("user")) && (password.equals("pswd")))
{
out.println(" <h1> Welcome to "+username);
}
else
{
out.println(" <h1> Registration success ");
out.println(" <a href='./index.html'> Click for Home page </a>");
```

```
}
out.println(" </BODY>");
out.println("</HTML>");
out.close();
}
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
doPost( request,response);
}
}
```

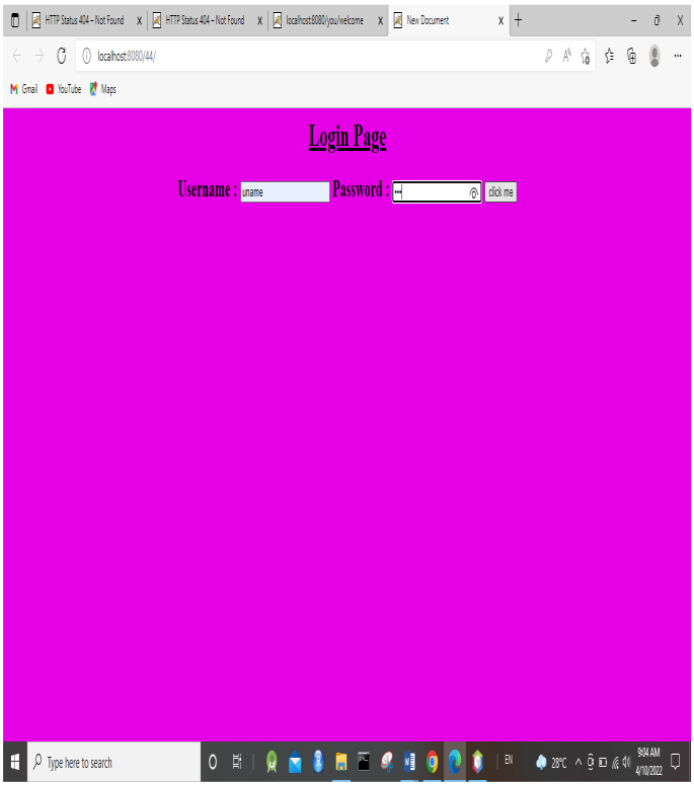
Registration.html:

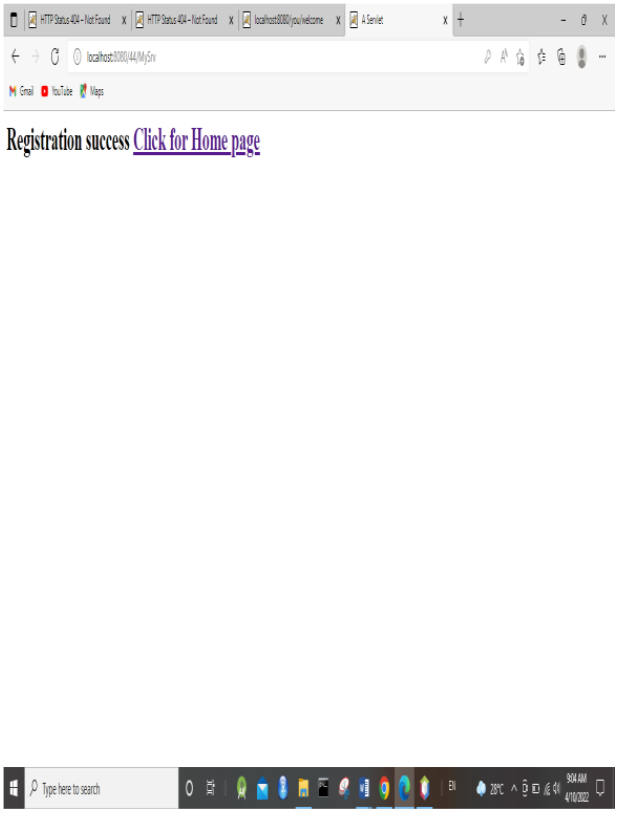
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY bgcolor='#e600e6'>
<form action='./MySrv' method="post">
<center> <h1> <u> Login Page </u></h1>
<h2> Username : <input type="text" name="uname"/>
Password : <input type="password" name="pwd"/>
<input type="submit" value="click me"/>
</center>
</form>
</body>
</HTML>
```

web.xml:

```
<web-app>
<servlet>
<servlet-name>MySrv</servlet-name>
<servlet-class>MySrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>MySrv</servlet-name>
<url-pattern>/MySrv</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>index.html</welcome-file>
</welcome-file-list>
</web-app>
```

OUTPUT:





RESULT:

Thus the java program to invoke servlets from HTML form has been executed successfully.

Ex.No:5b **SESSION TRACKING USING HIDDEN FORM FIELDS**
Date:

AIM:

To write a Java Program for Session Tracking Using Hidden Form Fields. This servlet demonstrates session tracking using hidden form fields by displaying the shopping cart for a bookworm. Note that, if you try this servlet, the buttons at the bottom of the page it generates don't take you anywhere real.

PROCEDURE:

1. Create a web page using HTML form that contains the fields such as text, password and one submit button.
2. Set the URL of the server as the value of form's action attribute.
3. Ask if the user wants to add more items or check out.
4. Include the current items as hidden fields so they'll be passed on and submit to self.

PROGRAM:

register.html:

```
<html>
<body bgcolor = "cyan">
<center>
<h1>WELCOME TO REGISTRATION PAGE</h1>
<form action="./registerone" METHOD="post">
Name: <input type="text" name = "name"><br><br>
Password: <input type="password" name="password"><br><br>
PROFESSION:
<select name="profession">
<option value="engineer">ENGINEER</option>
<option value="teacher">TEACHER</option>
<option value="businessman">BUSINESSMAN</option>
</select><br><br>
<input type="submit" value="REGISTER">
</form>
</center>
</body>
</html>
```

web.xml

```
<web-app>
<welcome-file-list>
<welcome-file>register.html</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>RegistrationServletOne</servlet-name>
<servlet-class>RegistrationServletOne</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>RegistrationServletOne</servlet-name>
<url-pattern>/registerone</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>RegistrationServletTwo</servlet-name>
```

```
<servlet-class>RegistrationServletTwo</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>RegistrationServletTwo</servlet-name>
<url-pattern>/registertwo</url-pattern>
</servlet-mapping>
</web-app>
```

RegistrationServletOne.java:

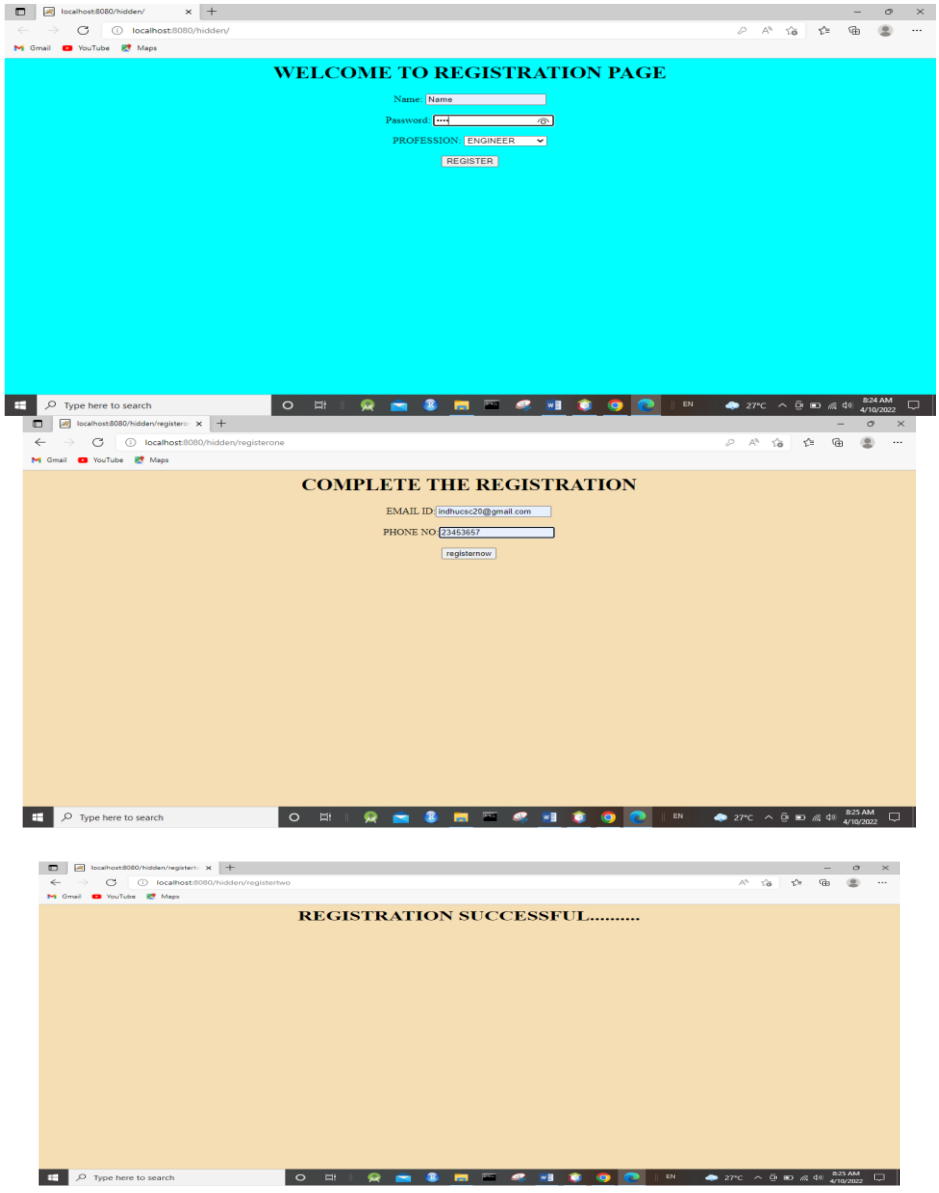
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class RegistrationServletOne extends HttpServlet
{
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
String name = request.getParameter("name");
String password = request.getParameter("password");
String profession = request.getParameter("profession");
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body bgcolor = wheat>");
out.println("<center>");
out.println("<h1>COMPLETE THE REGISTRATION</h1>");
out.println("<form action = ./registertwo method = post");
out.println("<input type = hidden name = name value =" + name + ">");
out.println("<input type = hidden name = password value =" + password + ">");
out.println("<input type = hidden name = profession value =" + profession + ">");
out.println("EMAIL ID:<input type =text name = email><br><br>");
out.println("PHONE NO:<input type =text name = cell><br><br>");
out.println("<input type =submit value=registernow>");
out.println("</center>");
out.println("</body></html>");
out.close();
}
}
```

RegistrationServletTwo.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 RegistrationServletTwo extends HttpServlet
{
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
String name = request.getParameter("name");
```

```
String password = request.getParameter("password");
String profession = request.getParameter("profession");
String email = request.getParameter("email");
String cell = request.getParameter("cell");
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body bgcolor = wheat>");
out.println("<center>");
out.println("<h1>REGISTRATION SUCCESSFUL.....</h1>");
out.println("</center>");
out.println("</body></html>");
out.close();
}
```

Output:



RESULT:

Thus the Java program for Session Tracking using hidden form fields has been executed successfully

EX.NO:6a THREE-TIER APPLICATIONS USING JSP AND DATABASES FOR CONDUCTING ON-LINE EXAMINATION

AIM:

To create a three-tier applications using JSP and Databases For conducting on-line examination.

ALGORITHM:

1. Create a database named examdb in mysql workbench
2. Inside examdb database create a table named examtable
3. Inside the examtable add column name as ans and set the datatype to be varchar(50)
4. Insert the coorect answer into the tables.
5. Design the HTML page (index.html) with the following
    a) Create a form to get the input from the user.
    b) Use radio buttons to make various options for the questions.
    c) Set the URL of the server (examserver.jsp) as the value of the action attribute.
    d) Use submit button to invoke the server and send the form data to the server.
6. Create the JSP file with the following
    a) Read the input from the client.
    b) Retrieve the answers from the database.
    c) Match the answers from the user with the correct answers from the database table.
    d) For each correct answer increment the mark by 5.
    e) Server displays the mark and result to the client as a response

PROGRAM:

Index.html

```
<html>
<head>
<title>Online Exam Client</title>
<style type="text/css">
body{
background-color:black;
```

```
color:blue;
}
</style>
</head>
<body>
<h2 style="text-align:center">ONLINE EXAMINATION</h2>
<h3>Answer the following questions (5 marks for each correct answer)</h3>
<hr/>
<form name="examForm" method="post" action="examserver.jsp">
1.C was developed by?<br/>
<input type="radio" name="ans1" value="Dennis Ritchie">Dennis Ritchie
<input type="radio" name="ans1" value="Stroustrup">Stroustrup
<input type="radio" name="ans1" value="David Ritchie">David Ritchie
<input type="radio" name="ans1" value="Charles Babbage">Charles Babbage
<br/><br/>
2. All computers must have <br/>
<input type="radio" name="ans2" value="Operating System">Operating System
<input type="radio" name="ans2" value="Application Software">Application
Software
<input type="radio" name="ans2" value="CD Drive">CD Drive
<input type="radio" name="ans2" value="Microsoft word">Microsoft word
<br/><br/>
3. The term PC means <br/>
<input type="radio" name="ans3" value="Private Computer">Private Computer
<input type="radio" name="ans3" value="Professional Computer">Professional
Computer
<input type="radio" name="ans3" value="Personal Computer">Personal
Computer
<input type="radio" name="ans3" value="Personal Calculator">Personal
Calculator
<br/><br/>
<input type="submit" value="Check Your Result"/>
</form>
</body>
</html>
```

EXAM.JSP:

```
<%@page contentType="text/html" language="java" import="java.sql.*"%>
<html>
<head>
<title>Online Exam Server</title>
<style type="text/css">
body{background-color:black;font-family:courier;color:blue}
</style>
```

```
</head>
<body>
<h2 style="text-align:center">ONLINE EXAMINATION</h2>
<p>
<a href="index.html">Back To Main Page</a>
</p>
<hr/>
<%
String str1=request.getParameter("ans1");
String str2=request.getParameter("ans2");
String str3=request.getParameter("ans3");
int mark=0;
String dbDriver = "com.mysql.jdbc.Driver";
String url = "jdbc:mysql:// localhost:3306/";
String dbname = "examdb";
String username = "root";
String password = "root";
Class.forName(dbDriver);
Connection con = DriverManager.getConnection(url +
dbname,username,password);
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT * FROM examtable");
int i=1;
while(rs.next())
{
if(i==1)
{
String dbans1=rs.getString(1);
if(str1.equals(dbans1))
{
mark=mark+5;
}
}
if(i==2)
{
String dbans2=rs.getString(1);
if(str2.equals(dbans2))
{
mark=mark+5;
}
}
if(i==3)
{
String dbans3=rs.getString(1);
```

```
if(str3.equals(dbans3))
{
mark=mark+5;
}
}
i++;
}
if(mark>=10)
{
out.println("<h4>Your Mark Is : "+mark+"</h4>");
out.println("<h3>Congratulations....! You Are Eligible For The Next
Round...</h3>");
}
else
{
out.println("<h4>Your Mark is : "+mark+"</h4>");
out.println("<h3>Sorry....!! You Are Not Eligible For The Next Round...</h3>");
}
%>
</form>
</body>
</html>
```

OUTPUT:

RESULT:

Thus the three-tier applications using JSP and Databases for conducting on-line examination has been completed and successfully verified.

Ex. No. 6b THREE-TIER APPLICATIONS USING JSP AND DATABASES FOR DISPLAYING STUDENTS MARKS

AIM:

To create a three tier application for displaying student marks using JSP and database**.**

ALGORITHM:

1. Create a database named studentdb2 in my SQL workbench
2. Inside studentdb2 database create a table named student
3. Inside the student table add column name as student id, student name, mark and set the data type to be int ,varchar (50),int respectively.
4. Design the HTML page (index.html) with the following
    a) Create a form to get the input from the user.
    b) Set the URL of the server (examserver.jsp) as the value of the action attribute.
    c) Use submit button to invoke the server and send the form data to the server.
5. Create the JSP (student.jsp) file with the following
    a) Read the input from the client.
    b) Retrieve the student details from the database.
    e) Server displays the id, name and mark to the client as a response.

PROGRAM:

Index.html

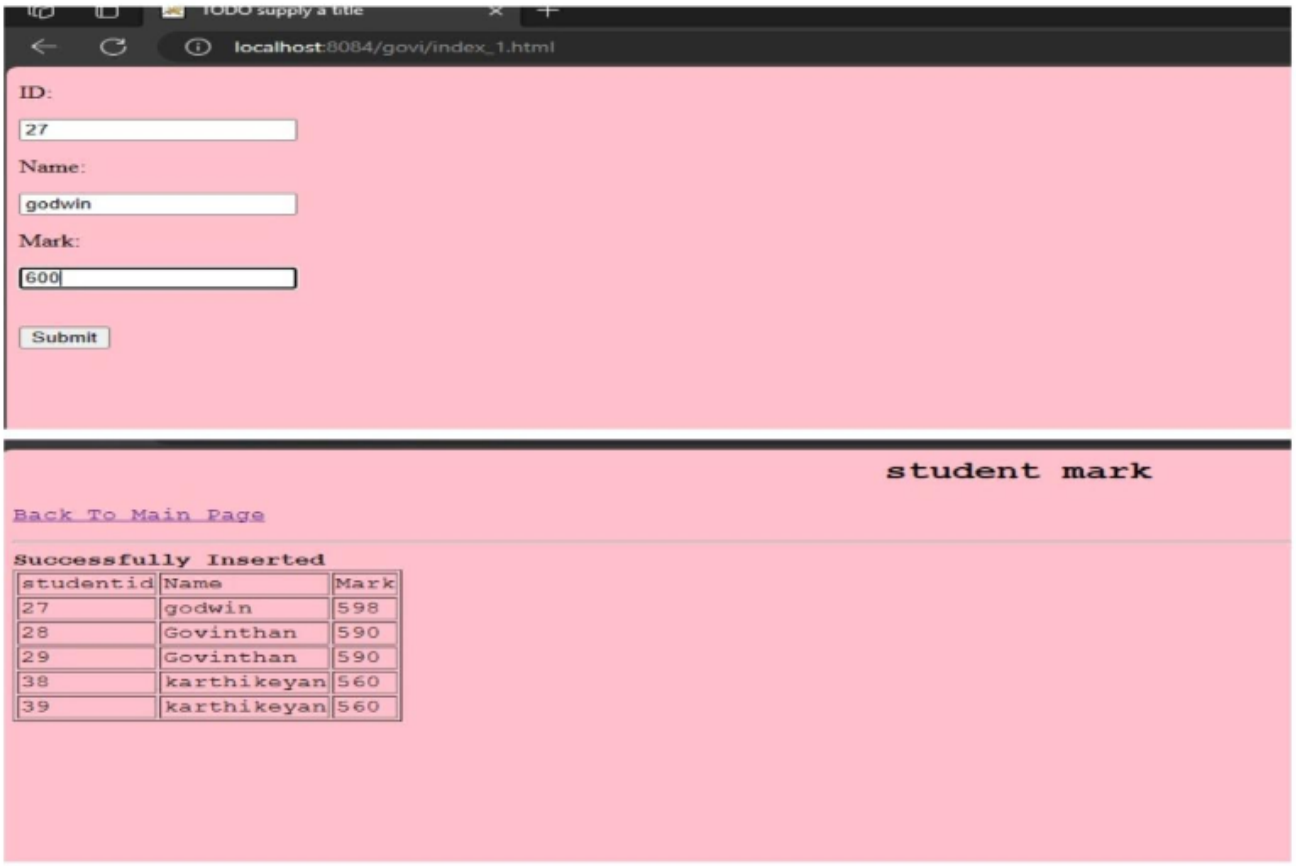
```
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
</head>

<body>
<form action="student.jsp" method="post">
<p>ID:</p>
<!-- Create an element with mandatory name attribute,
so that data can be transfer to the servlet using getParameter() -->
<input type="text" name="studentid"/>
<br/>
<p>Name:</p>
<input type="text" name="studentname"/>
<br/>
<p>Mark:</p>
<input type="text" name="mark"/>
<br/>
<br/><br/>
<input type="submit"/>
```

```
</form>
</body>
</html>
```
Student.jsp:
```
</html>
<%@page contentType="text/html" language="java" import="java.sql.*"%>
<html>
<head>
<title>Student mark display</title>
<style type="text/css">
body{background-color:black;font-family:courier;color:blue}
</style>
</head>
<body>
<h2 style="text-align:center">student mark</h2>
<p>
<a href="index.html">Back To Main Page</a>
</p>
<hr/>
<%
String dbDriver = "com.mysql.jdbc.Driver";
String url = "jdbc:mysql:// localhost:3306/";
String dbname = "studentdb2";
String username = "root";
String password = "root";
Class.forName(dbDriver);
Connection con = DriverManager.getConnection(url +
dbname,username,password);
PreparedStatement st;
st = con.prepareStatement("insert into student values(?, ?,?)");
st.setInt(1, Integer.valueOf(request.getParameter("studentid")));
st.setString(2, request.getParameter("studentname"));
st.setInt(3,Integer.valueOf(request.getParameter("mark")));
st.executeUpdate();
out.println("<html><body><b>Successfully Inserted"+
"</b></body></html>");
out.print("<table
border=1><tr><td>studentid</td><td>Name</td><td>Mark</td></tr>");
ResultSet rs=st.executeQuery("select * from student");
while(rs.next())
{
int id=rs.getInt("studentid");
String name=rs.getString("studentname");
int mark=rs.getInt("mark");
```

```
out.print("<tr><td>"+id+"</td><td>"+name+"</td><td>"+mark+"</td></tr>");
}
rs.close();
con.close();
st.close();
%>
</form>
</body>
</html>
```

OUTPUT:



RESULT:

Thus the three tier application for displaying student marks using JSP and database has been completed and successfully verified.

Ex. No. 7a **PROGRAMS USING XML – SCHEMA**

AIM:

To write Programs using XML – Schema.

ALGORITHM:

1. In Net Beans , point to New on the File menu, and then click Project.
2. Select the XML File type, and then click Open.
3. Save the file as book. xsd in the same folder as your XML document.
4. Save the modified XML document
5. Finally validate the xml file

PROGRAM:

XML.xml

```
<?xml version="1.0"?>
<bookstore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="book.xsd">
<book isbn="978-1234567890" edition="1st" language="English"
availability="in_stock">
<title> Web Technologies</title>
<author>Jeffrey Jackson </author>
<price>19.99</price>
<genre>Non-Fiction</genre>
<publish_date>2023-01-15</publish_date>
<description>A great book that you won't be able to put down.</description>
</book>
<book isbn="978-0987654321" language="Spanish">
<title> The Blue Umbrella </title>
<author>Ruskin BOnd </author>
<price>24.95</price>
<publish_date>2023-03-20</publish_date>
</book>
</bookstore>
```

book.xsd:

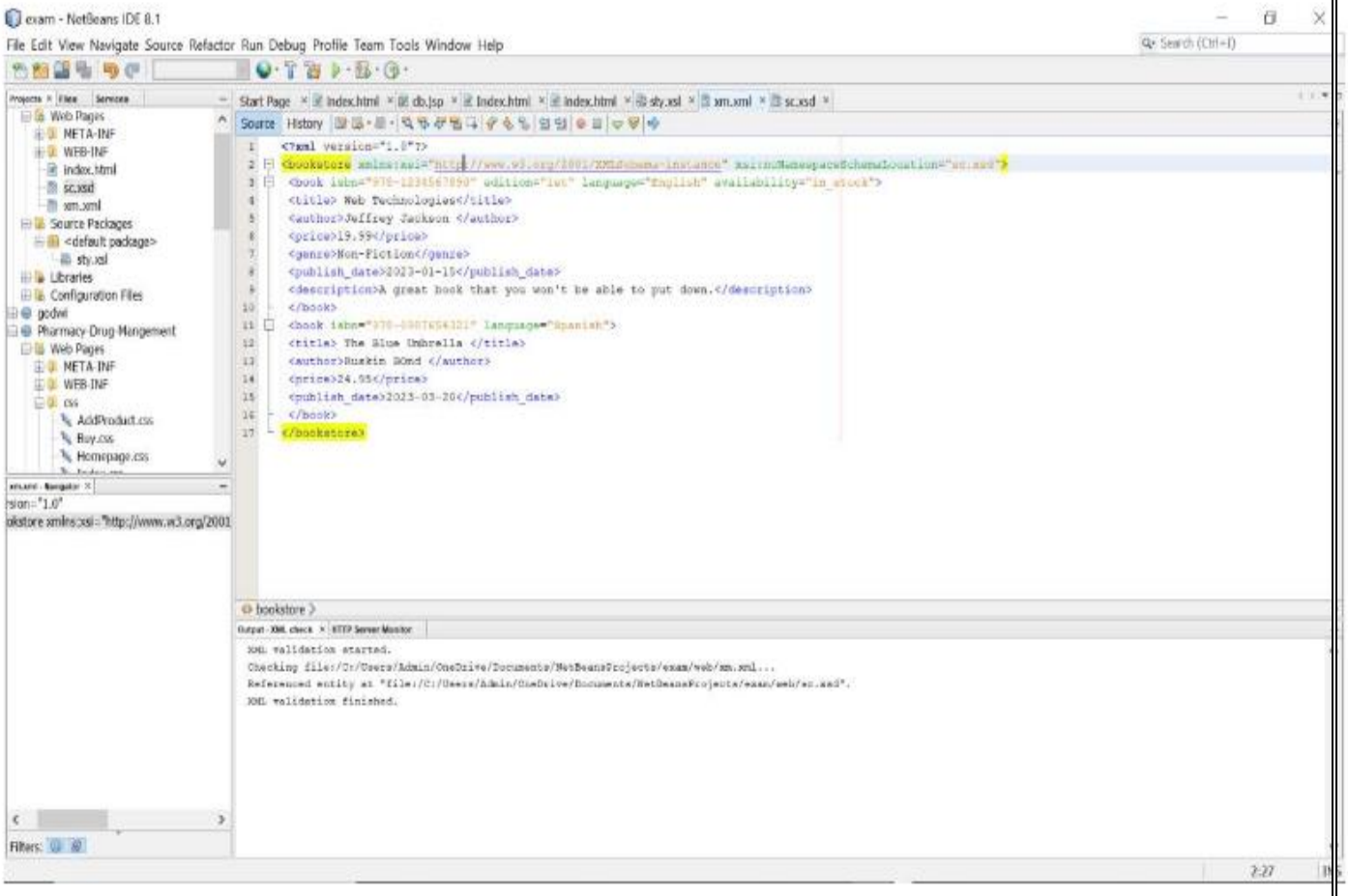
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="bookstore">
```

```
<xs:complexType>
<xs:sequence>
<xs:element name="book"
minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="title"
type="xs:string"/>
<xs:element name="author"
type="xs:string"/>
<xs:element name="price"
type="xs:decimal"/>
<xs:element name="genre"
type="xs:string" minOccurs="0"/>
<xs:element name="publish_date"
type="xs:date"/>
<xs:element name="description"
type="xs:string" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="isbn"
type="xs:string" use="required"/>
```

```
<xs:attribute name="edition"
type="xs:string" use="optional"/>
<xs:attribute name="language"
default="English">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration
value="English"/>
<xs:enumeration
value="Spanish"/>
<xs:enumeration
value="French"/>
<xs:enumeration
value="German"/>
<xs:enumeration
value="Other"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="availability"
default="in_stock ">
```

```
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration
value="in_stock"/>
<xs:enumeration
value="out_of_stock"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

OUTPUT:



RESULT:

Thus the Programs using XML – Schema has been completed and successfully verified.

Ex. No. 7b WRITE PROGRAMS USING XSL

Date :

AIM:

To write Programs using XSL.

ALGORITHM:

1. Select File > New > Other. In the New window, select XML > XSL. Click Next.
2. Select the My Project directory.
3. In the File name field, type Title. xsl and click Next.
4. In the Select XML file page, select the Title. xml file.
5. Click on xsl transformation and open the generated html file

PROGRAM:

XML.xml

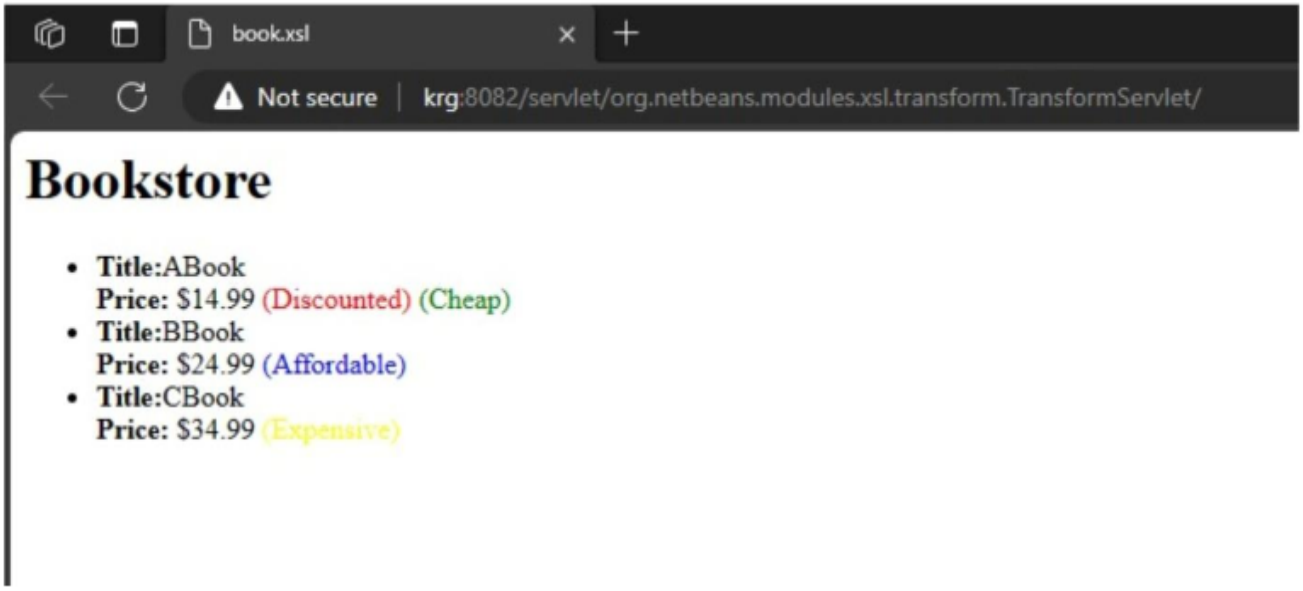
```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
<title>CBook </title>
<author>Author 3</author>
<price>34.99</price>
</book>
<book>
<title>BBook </title>
<author>Author 1</author>
<price>24.99</price>
</book>
<book>
<title>ABook </title>
<author>Author 2</author>
<price>14.99</price>
</book>
</bookstore>
```

XSL.xsl:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/T
ransform" version="1.0">
<xsl:output method="html"/>
<xsl:template match="/bookstore">
<html>
<head>
<title>book.xsl</title>
</head>
```

```
<body>
<h1>Bookstore</h1>
<ul>
<xsl:for-each select="book">
<xsl:sort select="title"
order="ascending" />
<li>
<strong>Title:</strong>
<xsl:value-of select="title" /><br />
<strong>Price:</strong>
$<xsl:value-of select="price" />
<xsl:if test="price &lt;
15.00">
<span style="color:
red;"> (Discounted)</span>
</xsl:if>
<xsl:choose>
<xsl:when test="price
&lt; 20.00">
<span style="color:
green;"> (Cheap)</span>
</xsl:when>
<xsl:when test="price
&gt; 20.00 and price &lt; 30.00">
<span style="color:
blue;"> (Affordable)</span>
</xsl:when>
<xsl:otherwise>
<span style="color:
yellow;"> (Expensive)</span>
</xsl:otherwise>
</xsl:choose>
</li>
</xsl:for-each>
</ul>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

OUTPUT:



RESULT:

Thus the Programs using XSL has been completed and successfully verified.

EX.NO:8a **XML DOM PARSER IN JAVA**

AIM

Implementation of XML DOM Parser in java.

PROCEDURE

Step 1: Set up your Eclipse Project

1. Open Eclipse and create a new Java project:

- Go to File > New > Java Project.
- Name your project (e.g., "XMLDOMParser").

2. Create a Java class:

- Right-click on the src folder in your project, then select New > Class.
- Name your class (e.g., "XMLParserExample") and include the public static void main(String[] args) method.

Step 2: Create an XML File

1. Right-click on your project and create a new file:

- Go to New > File.
- Name your file (e.g., "sample.xml") and add XML content inside the file. Here's an example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
<employee id="1">
<name>John Doe</name>
<department>IT</department>
<location>New York</location>
</employee>
<employee id="2">
<name>Jane Smith</name>
<department>HR</department>
<location>San Francisco</location>
</employee>
</employees>
```

Step 3: Write the XML DOM Parser Code Now, write code to parse this XML file using the DOM parser.

```
import org.w3c.dom.*;
```

```
import javax.xml.parsers.*;
import java.io.File;
public class XMLParserExample {
public static void main(String[] args) {
try {
// Initialize the parser
File xmlFile = new File("sample.xml");
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(xmlFile);
// Normalize the XML structure
doc.getDocumentElement().normalize();
System.out.println("Root element: " + doc.getDocumentElement().getNodeName());
// Get all employee nodes
NodeList nodeList = doc.getElementsByTagName("employee");
// Loop through each employee node
for (int i = 0; i < nodeList.getLength(); i++) {
Node node = nodeList.item(i);
if (node.getNodeType() == Node.ELEMENT_NODE) {
Element element = (Element) node;
// Extract data from each employee element
String id = element.getAttribute("id");
String name = element.getElementsByTagName("name").item(0).getTextContent();
String department =
element.getElementsByTagName("department").item(0).getTextContent();
String location =
element.getElementsByTagName("location").item(0).getTextContent();
// Print employee details
System.out.println("Employee ID: " + id);
System.out.println("Name: " + name);
System.out.println("Department: " + department);
System.out.println("Location: " + location);
System.out.println("--------------------");
}
}
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Step 4: Add Required Libraries. The code uses the org.w3c.dom and javax.xml.parsers packages, which are part of the JDK, so you don’t need to add any external libraries.

Step 5: Run the Program

Run your Java program by selecting the class and choosing Run > Run or by pressing Ctrl + F11.

OUTPUT:

Root element: employees
Employee ID: 1
Name: John Doe
Department: IT
Location: New York

Employee ID: 2
Name: Jane Smith
Department: HR
Location: San Francisco

RESULT

The XML file is parsed using DOM, which is implemented in java.

EX.NO:8b **XML SAX PARSER IN JAVA**

AIM

Implementation of XML SAX Parser in java.

PROCEDURE

STEP 1: Set Up Eclipse Project

1. Open Eclipse and create a new Java Project:
   - Go to File > New > Java Project.
   - Name your project (e.g., SAX Parser Example).
2. Create a new Java class:
   - Right-click on the src folder in the project.
   - Select New > Class.
   - Name the class (e.g., SAX Parser Example).

STEP 2: Create an XML File

1. Right-click on the project and create a new file:
   - Go to New > File.
   - Name it employees.xml and paste the following content inside:

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
<employee id="1">
<name>John Doe</name>
<department>IT</department>
<location>New York</location>
</employee>
<employee id="2">
<name>Jane Smith</name>
<department>HR</department>
<location>San Francisco</location>
</employee>
</employees>
```

STEP 3: Create a SAX Handler Class Next, to create a class that handles SAX events by extending Default Handler.

1. Right-click on the src folder, and create another Java class named Employee Handler:

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class EmployeeHandler extends DefaultHandler {
boolean bName = false;
boolean bDepartment = false;
boolean bLocation = false;
@Override
```

```
public void startElement(String uri, String localName, String qName, Attributes attributes)
throws SAXException {
// Detecting the start of the element
if (qName.equalsIgnoreCase("employee")) {
String id = attributes.getValue("id");
System.out.println("Employee ID: " + id);
} else if (qName.equalsIgnoreCase("name")) {
bName = true;
} else if (qName.equalsIgnoreCase("department")) {
bDepartment = true;
} else if (qName.equalsIgnoreCase("location")) {
bLocation = true;
}
}
@Override
public void endElement(String uri, String localName, String qName) throws SAXException {
// When the end of an employee element is reached
if (qName.equalsIgnoreCase("employee")) {
System.out.println("End of employee details.\n");
}
}
@Override
public void characters(char[] ch, int start, int length) throws SAXException {
// Handling the text between the XML tags
if (bName) {
System.out.println("Name: " + new String(ch, start, length));
bName = false;
} else if (bDepartment) {
System.out.println("Department: " + new String(ch, start, length));
bDepartment = false;
} else if (bLocation) {
System.out.println("Location: " + new String(ch, start, length));
bLocation = false;
}
}
@Override
public void startDocument() throws SAXException {
System.out.println("Start parsing document...\n");
}
@Override
public void endDocument() throws SAXException {
System.out.println("End parsing document.");
}
}
```

STEP 4: Write the Main Class to Parse the XML Now, go back to the SAXParserExample class (created earlier) and implement the code to parse the XML file using SAX

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.File;
public class SAXParserExample {
public static void main(String[] args) {
try {
// Create a SAXParserFactory instance
SAXParserFactory factory = SAXParserFactory.newInstance();
// Create a SAXParser instance
SAXParser saxParser = factory.newSAXParser();
// Create an instance of the EmployeeHandler
EmployeeHandler handler = new EmployeeHandler();
// Parse the XML file using the handler
File xmlFile = new File("employees.xml");
saxParser.parse(xmlFile, handler);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

STEP 5: Add Required Libraries (If Needed)

1. Ensure that the JRE System Library is set up correctly by right-clicking the project and selecting:

- Properties > Java Build Path > Libraries.
- Check if the JRE System Library is selected and configured.

STEP 6: Run the Program

1. Right-click on the SAXParserExample.java class and select Run As > Java Application.

OUTPUT:

```
Start parsing document...
Employee ID: 1
Name: John Doe
Department: IT
Location: New York
End of employee details.
Employee ID: 2
Name: Jane Smith
Department: HR
Location: San Francisco
End of employee details.
End parsing document.
```

RESULT: The XML file is parsed using SAX ,which is implemented in java.

EX.NO:9 **JAVA SCRIPT PROGRAM FOR A AJAX**

AIM:

To write a java script program for a AJAX.

ALGORITHM:

1. Start the program.
2. A scripting language that is commonly hosted in a browser to add Inter activity to HTML PAGES.
3. Defines the structure of a webpage as a set of programmable objects that can be accessed through java script.
4. Allows a client-side script to perform and http request.
5. AJAX applications use xml http request object to perform asynchronous requests to the server as opposed to performing a full page refresh.
6. Display the result.
7. Stop the program.

PROGRAM:

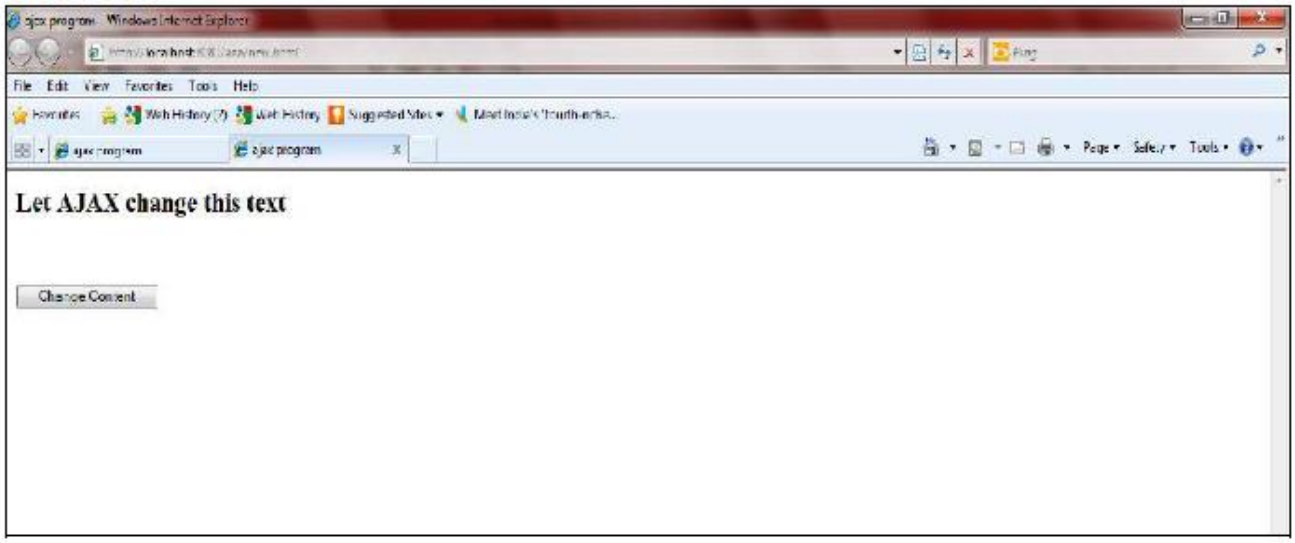
```
<html>
<head>
<script type="text/ javascript ">
Function load XML Doc()
{
if (window.XMLHttpRequest)
xmlhttp=new XMLHttpRequest();
else
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 &&xmlhttp.status==200)
{
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","new.txt",true);
xmlhttp.send();
}
</script>
<title>ajax program</title>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
```

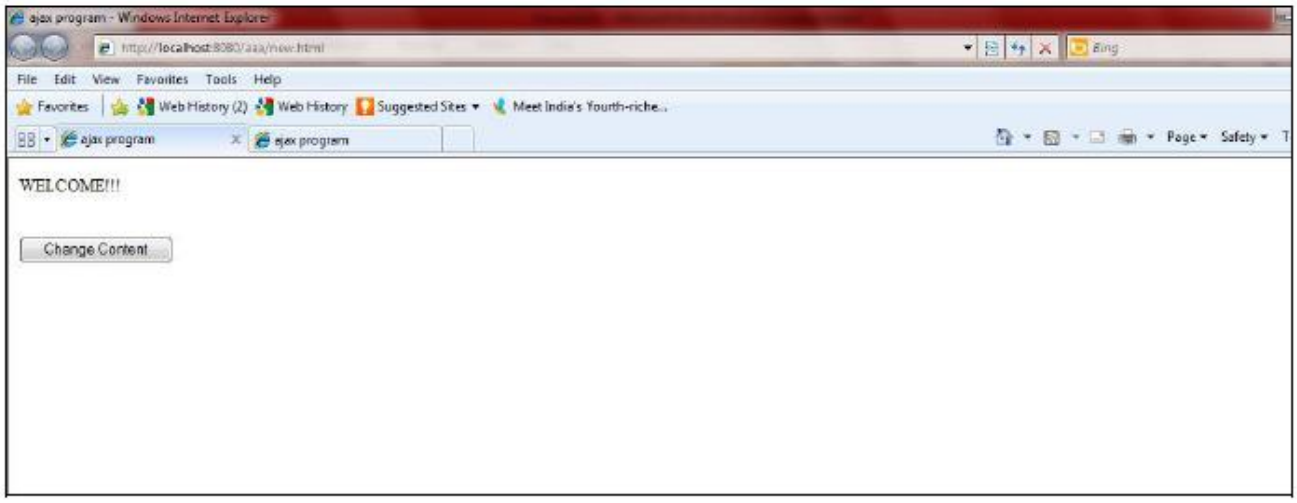
</body>
</html>

New.txt:

WELCOME!!!

OUTPUT:





RESULT:

Thus the program for AJAX was executed and the output was verified.

EX.NO: 10 IMPLEMENTING AN APPLICATION USING THE WEB SERVICES
DATE:

AIM:

To implement a application using the web services.

ALGORITHM:

1.Start the program
2.Create a root rocess for reservation
3.Create a service with focus on each item
4.Run the program, display the result
5.Stop the program.

SOURCE CODE:

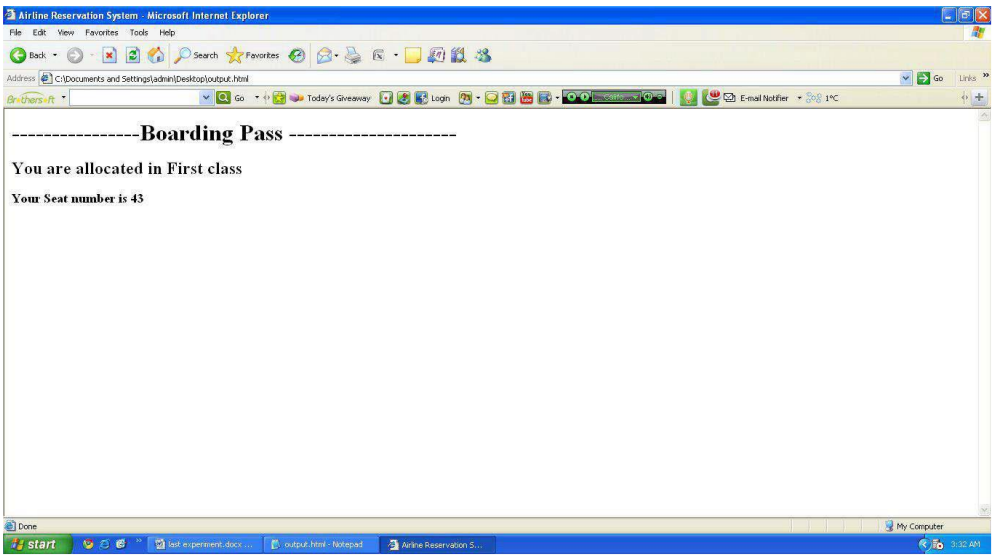
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Solution11.16 -->
<!-- Airline Reservation System-->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Airline Reservation System</title>
<script type = "text/javascript">
<!--
var input;
var secondInput;
var element;
var secondElement;
var firstCount = 0;
var economyCount = 0;
var seats = [ ,0,0,0,0,0,0,0,0,0,0]; //allocate 10-element Array
function startArray()
{
for(var i=0; i<11; i++)
{
input = window.prompt("Please type 1 for First Class and Please type 2 for Economy.","0");
if (input == 1 || input == 2)
{
element = linearSearch(seats);
if(element==-1&&input==1)
{
document.writeln("The First Class is already fully
booked<br/>");
```

```
secondQuestion(seats);
}
else if (element ==-1 && input == 2)
{
document.writeln("The Economy Class is already fully
booked<br/>");
secondQuestion(seats);
}
else
boarding Pass(input);
}
//to terminate the program
else
{
window.status="Bye-bye!";
System.exit(0);
}
}
}
function linear Search(the Array)
{
if (input == 1)
{
for (var n=0; n<6 ; n++)
if (the Array [n] == 0)
return n;
}
else if (input == 2)
{
for (var n=6; n<11 ; n++)
if (the Array [n] == 0)
return n;
}
return -1;
}
function boarding Pass(the Input)
{
if (input ==1)
{
document.writeln("----------BOARDING PASS----------<br/>");
document.writeln("You are allocated in the First Class<br/>");
document.writeln("Your seat number is "+ element+"<br/>");
document.writeln("----------<br/>");
seats[element]= 1;
```

```
firstCount++;
}
else if (input ==2)
{
document.writeln("----------BOARDING PASS----------< br />");
document.writeln("You are allocated in the Economy Class< br />"); document.writeln("Your seat
number is "+ element +"<br/>");
document.writeln("----------<br/>");
seats[element]= 1;
economy Count++;
}
}
Function second  Question(the Array)
{
if (input == 1)
{
for (var n=6; n<11 ;n++)
{
if (the Array [n] == 0)
{
second Input = window .prompt("Do you want to move to Economy Class?
(If YES, please press 1. If NO, please press 2)","0");
if ( second Input == 1)
{
input = 2;
element=linear Search(seats);
document.writeln("You have been allocated to Economy
Class<br/>");
boardingPass(input);
break;
}
else if (secondInput == 2)
{
document.writeln("Next flight leaves in 3 hours<br/>"); break;
}
}
}
}
else if (input == 2)
{
for (var n=0; n<6 ;n++)
{
if (theArray [n] == 0)
{
```

```
secondInput = window.prompt("Do you want to move to First Class? (If YES,
please press 1. If NO, please press 2)","0");
for (var n=0; n<6 ;n++)
{
if (theArray [n] == 0)
{
secondInput = window.prompt("Do you want to move to First Class? (If YES, please press 1. If NO,
please press 2)","0");
boarding Pass(input); break;
}
else if (secondInput == 2)
{
document.writeln("Next flight leaves in 3 hours<br/>");
break;
}
} }
}
}
//-->
</script>
</head>
<body onload = "startArray()"></body>
</html>
```

OUTPUT:



RESULT:

Thus the program is executed and verified successfully