Ex No: 1	Interactive Image Map with Clickable Hotspots in HTML

AIM:

To create an interactive image map in a web page using HTML, define clickable hotspots, and display relevant information when the hotspots are clicked.

TOOLD REQUIRED:

1. Text Editor - Visual Studio Code
2. Web Browser
3. HTML & CSS

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>
```

RESULT:

Ex No: 2	Implementing All Types of Cascading Style Sheets in a Web Page

AIM:

To create a web page that demonstrates the use of all three types of Cascading Style Sheets (CSS): Inline CSS, Internal CSS, and External CSS.

TOOLD REQUIRED:

1. Text Editor - Visual Studio Code
2. Web Browser
3. HTML & CSS

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 an external style sheets

5. Create an embedded style sheets

6. Create an 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;
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>Sri Krishna College of 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">Sri Krishna College of Technologe <br>It is approved by AICTE(All India
Council for Technical Education). It is affiliated to Anna University.<br></p>
<h2 class="vid"> Sri Krishna College of Technology </h2> <br>
<font>It is an Autonomous 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>
```

RESULT:

Ex No: 3	Client Side Scripts for Validating Web Form Controls using DHTML

AIM:

To develop a web form with client-side validation using Dynamic HTML (DHTML), incorporating JavaScript, HTML, and CSS to ensure proper data entry before submission.

TOOLD REQUIRED:

1. Text Editor - Visual Studio Code
2. Web Browser
3. HTML & CSS
4. JavaScript

PROCEDURE:

1. Static web pages of an online Book store is developed with following pages.

 Home page

 Registration and user Login

 User profile page

 Books catalog

 Payment by credit card

2. Each input box in webpage is validated using java script code using <script> tag in html file.

3. Designed output is displayed.

PROGRAM:

Main.html:

```
<html>

<head>

<title> ONLINE BOOK STORES</title>

</head>

<body bgcolor="pink">

<marquee><h1 align=”center”><b><u><font color="white">

ONLINE BOOK STORAGE</u></font>

</b></h1></marquee>
```

```
<H2 ALIGN="CENTER">
<b><p><U><FONT COLOR="PURPLE">Welcome to online book storage.
Press login if you are having id otherwise press registration.</U></FONT></p></b></H2>
<H2> <FONT COLOR="WHITE"></FONT></H2>
<H3 ALIGN="CENTER">
<A HREF="reg.html"><BR><BR><FONT COLOR="black"><ITALIC>REGISTRATION
FORM</FONT></ITALIC><BR><BR>
<BR><BR><A HREF="profile.html"><FONT COLOR="black"><ITALIC>USER
PROFILE</FONT></ITALIC><BR>
<BR><BR><A HREF="login.html"><FONT COLOR="black"><ITALIC>USER
LOGIN</FONT></ITALIC><BR>
<BR><BR><A HREF="catalog.html"><FONT COLOR="black"><ITALIC>BOOKS
CATALOG</FONT></ITALIC><BR>
<BR><BR><A HREF="payment.html"><FONT
COLOR="black"><ITALIC>PAYMENT</FONT></ITALIC><BR>
<BR><BR><A HREF="Order.html"><FONT COLOR="black"><ITALIC>ORDER
CONFIRMATION</H3></FONT></ITALIC><BR>
</body>
</html>
```

Login.html:

```
<html>
<body bgcolor=”blue”><br><br><br>
<script language=”javascript”>
function validate()
{
var flag=1;
if(document.myform.id.value==""||document.myform.pwd.value=="")
{
alert("LoginId and Password must be filled")
 flag=0;
}
if(flag==1)
{
alert(“VALID INPUT”);
 window.open("catalog.html","right");
}
else
{
alert(“INVALID INPUT”);
//document.myform.focus();
}
}
</script>
<form name=”myform”>
<div align=”center”><pre>
LOGIN ID:<input type=”text” name=”id”><br>
PASSWORD:<input type=”password” name=”pwd”>
<br><br>
</pre>
<input type=”button” value=”ok” onClick=”validate()”>
<input type=”reset” value=”clear”>
</div>
</form>
</body>
</html>
```

Reg.html:

```
<html>
<body bgcolor=”blue”><br><br>
<script language=”javascript”>
var str=document.myform.phno.value;
var x;
if(flag==1)
{
alert("VALID INPUT");
}
else
{
alert("INVALID INPUT");
document.myform.focus();
}
}
</script>
<form name="myform">
<div align="center"><pre>
NAME :<input type="text" name="name"><br>
ADDRESS :<input type="type" name="addr"><br>
CONTACT NUMBER:<iput type="text" name="phno"><br>
LOGINID :<input type="text" name="id"><br>
PASSWORD :<input type="password" name="pwd"></pre><br><br>
</div>
<br><br>
<div align="center">
<input type="submit" value="ok" onClick="validate()">&nbsp;&nbsp;&nbsp;
<input type="reset" value="clear">
</form></body></html>
```

Catalog.html:

```
<html>
<body bgcolor="pink"><br><br><br>
<div align="center">
<pre>
```

```
BOOK TITLE:<input type="text" name="title"><br>
</pre><br><br>
</div>
<br><br>
<div align="center">
<input type="submit" value="ok" name="button1">
<input type="reset" value="clear" name="button2">
</body>
</html>
```

Order.html:

```
<html>
<body bgcolor="pink"><br><br><br>
<div align="center"><pre>
LOGIN ID :<input type="text" name="id"><br>
TITLE :<input type="text" name="title"><br>
NO.OF BOOKS :<input type="text" name="no"><br>
COST OF BOOK:<input type="text"name="cost"><br>
DATE :<input tpe="text" name="date"><br></pre><br><br>
</div>
<br><br>
<div align="center">
<input type="submit" value="ok" name="button1">
<input type="reset" value="clear" name="button2">
</body>
</html>
```

Payment.html:

```
<html>
<body bgcolor="pink"><br><br><br>
<script language="javascript">
function validate()
```

```
{
var flag=1;
if(document.myform.id.value==""||
document.myform.pwd.value==""||
document.myform.amount.value==""||
document.myform.num.value=="")
{
flag=0;
}
var str=document.myform.amount.value;
var x;
for(var i=0;i<str.length;i++)
{
x=str.substr(i,1);
if(!(x<=9))
{
flag=0;
break;
}
}
str=document.myform.num.value;
for(var i=0;i<str.lenght;i++)
{
x=str.substr(i,1);
if(!(x<=9))
{
flag=0;
break;
}
}
if(flag==1)
{
alert("VALID INPUT");
}
else
{
alert("INVALID INPUT");
document.myform.focus();
}
}
</script>
<form name="myform">
<div align="center"><pre>
LOGIN ID :<input type="text" name="id"><br>
PASSWORD :<input type="password" name="pwd"><br>
AMOUNT :<input type="text" name="amount"><br>
CREDITCARDNUMBER:<input type="PASSWORD" name="num+"><br></pre><br><br>
</div>
<br><br><div align="center">
<input type="submit" value="ok" onClick="validate()">
<input type="reset" value="clear">
</form>
</body>
</html>
```

Profile.html:

```
<html>
<body bgcolor="pink"><br><br><br>
<script type="text/javascript">
function validate()
{
```

```
var flag=1;
if(document.myform.id.value==""||
document.myform.pwd.value=="")
{
flag=0;
}
if(flag==1)
{
alert("VALID INPUT");
}
else
{
alert("INVALID INPUT");
document.myform.focus();
}
}
</script>
<form name="myform">
<div align="center"><pre>
LOGIN ID :<input type="text" name="id"><br>
PASSWORD:<input type="password" name="pwd"></pre><br><br>
</div>
<br><br>
<div align="center">
<input type="submit" value="ok" onClick="validate()">
<input type="reset" value="clear" >
</form>
</body></html>
```

RESULT:

Ex No: 4	Developing Java Servlets for Handling HTML Form Submission and Session Tracking

AIM:

To develop a web form with client-side validation using Dynamic HTML (DHTML), incorporating JavaScript, HTML, and CSS to ensure proper data entry before submission.

TOOLD REQUIRED:

1. Java Development Kit(JDK)
2. Integrated Development Environment (IDE)
3. Apache Tomcat Server
4. Servlet API
5. Web Browser
6. HTML & CSS
7. JavaScript

A) To invoke servlets from HTML forms:

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>
```

B) Session Tracking

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();
}
```

RESULT:

Ex No: 5	**Developing a Three-Tier Web Application using JSP and Databases for Online Examination and Student Mark List Management**

AIM:

To design and implement a three-tier web application using JSP, Servlets, and a database for conducting an online examination, where users can attempt questions and receive immediate evaluation.

To develop a student mark list management system that retrieves and displays student marks from a database using JSP and JDBC.

TOOLD REQUIRED:

1. Java Development Kit(JDK)
2. Integrated Development Environment (IDE)
3. Apache Tomcat Server
4. Servlet API
5. Web Browser
6. HTML & CSS
7. JavaScript

PROCEDURE:

Client:

1. In index.html on the client side declare the contents that you like to transfer to the server using html form and input type tags.

2. Create a submit button and close all the included tags.

Server:

1. Import all necessary packages

2. Define a class that extends servlet

3. In the doPost() method, do the following:

i) Set the content type of the response to "text/html"

ii) Create a writer to the response

iii) Get a paratmeter from the request

iv) If its value is equal to right answer then add 5 to mark variable

v) Similarly repeat step

vi) for all parameters

vii)Display the result in an html format using the writer

Student Mark List Database:

1. Import necessary to java packages and javax packages and classes
2. Create a class that extends HttpServlet and implements ServletException
3. and IOException
4. In the doGet() method, do the following:

    i) Create a PrintWriter object

    ii) Open a connection with the data source name

    iii) Write a sql query and execute to get the resultset

    iv) Display the resultset information in html form

PROGRAM:

Servlet Code:

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class StudentServlet3 extends HttpServlet
{
String message,Seat_no,Name,ans1,ans2,ans3,ans4,ans5; int Total=0;
Connection connect; Statement stmt=null; ResultSet rs=null;
public void doPost(HttpServletRequest request,HttpServletResponse response) throws
ServletException,IOException
{
try
{
String url="jdbc:odbc:NEO"; Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
connect=DriverManager.getConnection(url," "," "); message="Thank you for participating in
online
Exam";
}
```

```
catch(ClassNotFoundException cnfex){ cnfex.printStackTrace();

}

catch(SQLException sqlex){ sqlex.printStackTrace();

}

catch(Exception excp){ excp.printStackTrace();

}

Seat_no=request.getParameter("Seat_no"); Name=request.getParameter("Name");

ans1=request.getParameter("group1"); ans2=request.getParameter("group2");

ans3=request.getParameter("group3"); ans4=request.getParameter("group4");

ans5=request.getParameter("group5"); if(ans1.equals("True"))

Total+=2;

if(ans2.equals("False"))

Total+=2;

if(ans3.equals("True"))

Total+=2;

if(ans4.equals("False"))

Total+=2;

if(ans5.equals("False"))

Total+=2; try

{

Statement stmt=connect.createStatement();

String query="INSERT INTO student("+"Seat_no,Name,Total"+")
VALUES('"+Seat_no+"','"+Name+"','"+Total+"')";

int result=stmt.executeUpdate(query); stmt.close();

}catch(SQLException ex){

}

response.setContentType("text/html"); PrintWriter out=response.getWriter();
out.println("<html>");

out.println("<head>"); out.println("</head>"); out.println("<body bgcolor=cyan>");

out.println("<center>"); out.println("<h1>"+message+"</h1>\n");

out.println("<h3>Yours results stored in our database</h3>"); out.print("<br><br>");
out.println("<b>"+"Participants and their Marks"+"</b>"); out.println("<table border=5>");
try
{
Statement stmt=connect.createStatement(); String query="SELECT * FROM student";
rs=stmt.executeQuery(query); out.println("<th>"+"Seat_no"+"</th>");
out.println("<th>"+"Name"+"</th>"); out.println("<th>"+"Marks"+"</th>"); while(rs.next())
{
out.println("<tr>");
out.print("<td>"+rs.getInt(1)+"</td>");
out.print("<td>"+rs.getString(2)+"</td>");
out.print("<td>"+rs.getString(3)+"</td>");
out.println("</tr>");
}
out.println("</table>");
}
catch(SQLException ex){ } finally
{
try
{
if(rs!=null)
rs.close();
if(stmt!=null)
stmt.close();
if(connect!=null)
connect.close();
}
catch(SQLException e){ }
}
out.println("</center>");
out.println("</body></html>");
```

```
Total=0;
} }
```

HTML Code:

```
<html><head><title>Database Test</title></head> <body>
<center><h1>Online Examination</h1> </center>
<form action="StudentServlet3.view" method="POST"> <div align="left"><br></div>
<b>Seat Number:</b> <input type="text" name="Seat_no"> <div align="Right">
<b>Name:</b> <input type="text" name="Name" size="50"><br> </div>
<br><br>
<b>1. Every host implements transport layer.</b><br/> <input type="radio" name="group1"
value="True">True <input type="radio" name="group1" value="False">False<br>
<b>2. It is a network layer's responsibility to forward packets reliably from source to
destination</b><br/>
<input type="radio" name="group2" value="True">True
<input type="radio" name="group2" value="False">False<br>
<b>3. Packet switching is more useful in bursty traffic</b><br/> <input type="radio"
name="group3"
value="True">True<input type="radio" name="group3" value="False">False<br> <b>4. A
phone
network uses packet switching</b><br/> <input type="radio" name="group4"
value="True">True
<input type="radio" name="group4" value="False">False<br>
<b>5. HTML is a Protocol for describing web contents</b><br/> <input type="radio"
name="group5"
value="True">True
<input type="radio" name="group5" value="False">False<br> <br><br><br>
<center>
<input type="submit" value="Submit"><br><br> </center>
</form></body></html>
```

RESULT:

Ex No: 6	Programs using XML – Schema – XSLT/XSL

AIM:

To create and save an XML document at the server, which contain some users information. To develop Java Program takes user id as an input and returns the user details by taking the user information from the XML document.

TOOLD REQUIRED:

1. Java Development Kit(JDK)
2. Integrated Development Environment (IDE)
3. Apache Tomcat Server
4. Servlet API
5. Web Browser
6. HTML & CSS
7. JavaScript

PROCEDURE:

1. Save Students information in the XML file on the specific location.

2. Create and establish the server connection between html file and XML file in the host

3. Get the user ID as input

4. Display the user's information.

PROGRAM:

index.html:

```
<!DOCTYPE html>

<HTML>

<HEAD>

<TITLE>Searching for XML Elements </TITLE>

<SCRIPT>

function readXMLData()

{

var xmlDocumentObject, id , name , addr, phone, email;

xmlDocumentObject=new XMLHttpRequest();
xmlDocumentObject.open("GET","userlist.xml",false);

xmlDocumentObject.send();

xmlDocumentObject=xmlDocumentObject.responseXML;

id = xmlDocumentObject.getElementsByTagName("userid");

name = xmlDocumentObject.getElementsByTagName("username");

address = xmlDocumentObject.getElementsByTagName("address");

phone = xmlDocumentObject.getElementsByTagName("phone");

email = xmlDocumentObject.getElementsByTagName("email");

for (i = 0; i < id.length; i++)

{

output=id[i].firstChild.nodeValue;

if (output == document.getElementById("myText").value)

{displayDIV.innerHTML = id[i].firstChild.nodeValue + "<br> " +
name[i].firstChild.nodeValue

+"<br> " +address[i].firstChild.nodeValue + "<br> " +

phone[i].firstChild.nodeValue+"<br>"+email[i].firstChild.nodeValue;

} } }

</SCRIPT>

</HEAD>

<BODY>

<H1>Search User</H1>

<input type="text" id="myText" value="">

<input type="BUTTON" VALUE="Get User Details" ONCLICK="readXMLData()">

<P>

<DIV ID="displayDIV"> </DIV>

</BODY>

</HTML>
```

userlist.xml:

```
<userlist>

<userid>usr01</userid>

<username>xxxx</username>
```

```
<address>coimbatore</address>
<phone>9123456789</phone>
<email>xxxx@gmail.com</email>
<userid>usr02</userid>
<username>yyyy</username>
<address>chennai</address>
<phone>9111111111</phone>
<email>yyyy@gmail.com</email>
<userid>usr03</userid>
<username>zzzz</username>
<address>telugana</address>
<phone>9222222222</phone>
<email>zzzz@gmail.com</email>
<userid>usr04</userid>
<username>abcd</username>
<address>bangalore</address>
<phone>9555555555</phone>
<email>abcd@gmail.com</email>
<userid>usr05</userid>
<username>efgh</username>
<address>Perambalur</address>
<phone>9666666666</phone>
<email>efgh@gmail.com</email>
</userlist>
```

RESULT:

Ex No: 7	Programs using NoSQL

AIM:

To create and save an XML document at the server, which contain some users information. To develop Java Program takes user id as an input and returns the user details by taking the user information from the XML document.

TOOLD REQUIRED:

1. Java Development Kit(JDK)
2. Integrated Development Environment (IDE)
3. NoSQL Database

PROCEDURE:

1. Install Java Development Kit (JDK 8 or later).
2. Install an IDE like Eclipse, IntelliJ IDEA, or NetBeans.
3. Install MongoDB Community Edition (if using MongoDB as the NoSQL database).
4. Download and configure the MongoDB Java Driver (or Firebase SDK if using Firebase).

Set Up MongoDB

1. Start the MongoDB server (mongod).
2. Open MongoDB shell (mongo) and create a database:
   `use studentDB`
3. Create a collection for storing student details:

```
db.students.insertMany([

   { "studentId": 1, "name": "Alice", "marks": 85 },

   { "studentId": 2, "name": "Bob", "marks": 90 },

   { "studentId": 3, "name": "Charlie", "marks": 75 }

])
```

4. Verify the data
   `db.students.find().pretty()`
5. Create a Java Project in Eclipse/IntelliJ.
6. Add MongoDB Java Driver (mongodb-driver-sync) to the project via Maven or manually.
7. Write Java Code for CRUD Operations
8. Compile the Java program.
9. Run the program in the IDE or terminal.

10. Verify the changes in MongoDB using:
    `db.students.find().pretty()`

PROGRAM:

```
import com.mongodb.client.*;
import org.bson.Document;

public class MongoDBExample {
   public static void main(String[] args) {
      // Connect to MongoDB
      MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
      MongoDatabase database = mongoClient.getDatabase("studentDB");
      MongoCollection<Document> collection = database.getCollection("students");

      // CREATE - Insert a new student record
      Document newStudent = new Document("studentId", 4)
            .append("name", "David")
            .append("marks", 88);
      collection.insertOne(newStudent);
      System.out.println("New student inserted!");

      // READ - Retrieve and display all students
      System.out.println("Student Records:");
      FindIterable<Document> students = collection.find();
      for (Document student : students) {
         System.out.println(student.toJson());
      }

      // UPDATE - Modify a student's marks
      collection.updateOne(new Document("studentId", 2),
            new Document("$set", new Document("marks", 95)));
```

```
        System.out.println("Updated student marks!");

        // DELETE - Remove a student record
        collection.deleteOne(new Document("studentId", 3));
        System.out.println("Deleted student record!");

        // Close the MongoDB connection
        mongoClient.close();
    }
}
```

RESULT: