UNIVERSITY COLLEGE OF ENGINEERING NAGERCOIL

(ANNA UNIVERSITY CONSTITUENT COLLEGE)

KONAM, NAGERCOIL – 629 004

RECORD NOTE BOOK

CCS375-WEB TECHNOLOGIES LABORATORY

Register No :

Name :

Year/Semester :

Department :

UNIVERSITY COLLEGE OF ENGINEERING NAGERCOIL

(ANNA UNIVERSITY CONSTITUENT COLLEGE)

KONAM, NAGERCOIL – 629 004

Register No:

Certified that, this is the bonafide record of work done by Mr/Ms. …… of V Semester in Computer Science and Engineering of this college, in the Web Technologies Laboratory (CCS375) during academic year 2025-2026 in partial fulfillment of the requirements of the B.E Degree course of the Anna University Chennai.

Staff-in-charge **Head of the Department**

This record is submitted for the University Practical Examination held on ……

Internal Examiner **External Examiner**

INDEX

Exp No	Date	Title	Page	Sign

Ex No : 01	Creation of Image Map to Fix Hotspots	Page no :
Date :		

Aim:

To create a web page using HTML that embeds an image map, defines hotspots, and displays related information when the hotspots are clicked.

Description:

This experiment demonstrates how to create an interactive image map using HTML. An image map allows specific regions of an image—called hotspots—to be clickable, each linking to a different HTML page with relevant information. In this example, the map of India is used, and hotspots are defined for Tamil Nadu and Karnataka using the and tags. Each hotspot is assigned coordinates and a shape (circle or rectangle), and clicking on them navigates to a separate page displaying details about the selected state. This technique is useful for building visually engaging navigation systems in web applications.

Algorithm:

Step 1: Start
Step 2: Create an HTML file and load the image of the India map
Step 3: Use the tag to define a named image map
Step 4: Use tags to define clickable hotspots with shapes and coordinates
Step 5: Link each hotspot to a corresponding HTML page
Step 6: Create separate HTML pages for each state with relevant information
Step 7: Save and run the HTML file in a browser to test hotspot functionality
Step 8: Stop

Program:

Main.html:

```
<head>
<BODY bgcolor="#gop6876cgdt5564ss">
<img src="https://www.mapsofindia.com/images2/india-map-2019.jpg"
usemap="#indiamap"/>
<map name=indiamap>
<area shape="circle" coords="340,971,107,180" href="tamilnadu.html" alt="Tamilnadu">
<area shape="rect" coords="247,827,107,180" href="karnataka.html" alt="Karnataka">
</map>
</head>
</BODY> </html>
```

Karnataka.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Karnataka - India</title>
</head>
<body style="background-color: wheat;">
  <h1 style="text-align: center;">Karnataka</h1>
  <p>
    Karnataka is a southern Indian state known for its tech capital, Bengaluru. It has a rich history with UNESCO sites like Hampi and vibrant cultural traditions. The state also features lush landscapes, from the Western Ghats to coastal beaches.
  </p>
  <ul>
    <li>Districts: 30</li>
    <li>Capital City: Bangalore</li>
```

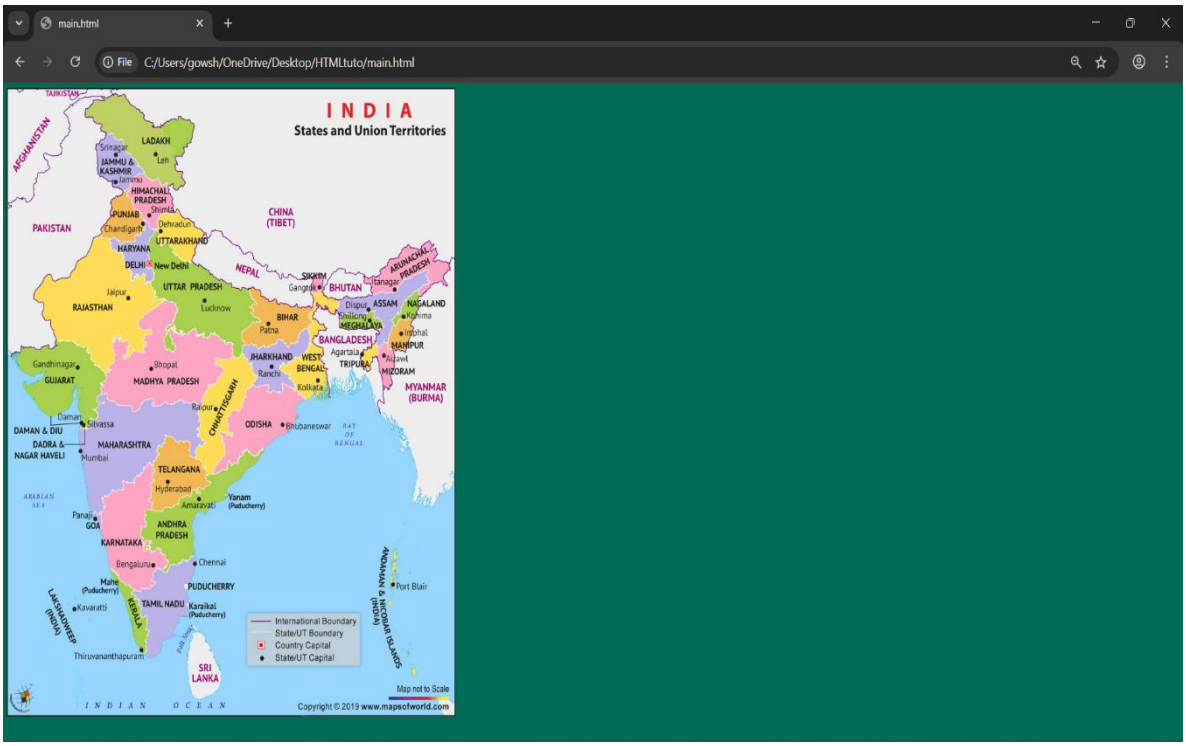
```
    <li>Governor: Thawar Chand Gehlot</li>
    <li>Chief Minister: Siddaramaiah</li>
    <li>Population: 61,130,704</li>
    <li>Tourist Spots: Gol Gumbaz, Mysore Palace, Keshava Temple</li>
  </ul>
  <a href="main.html">Back</a>
</body>
</html>
```

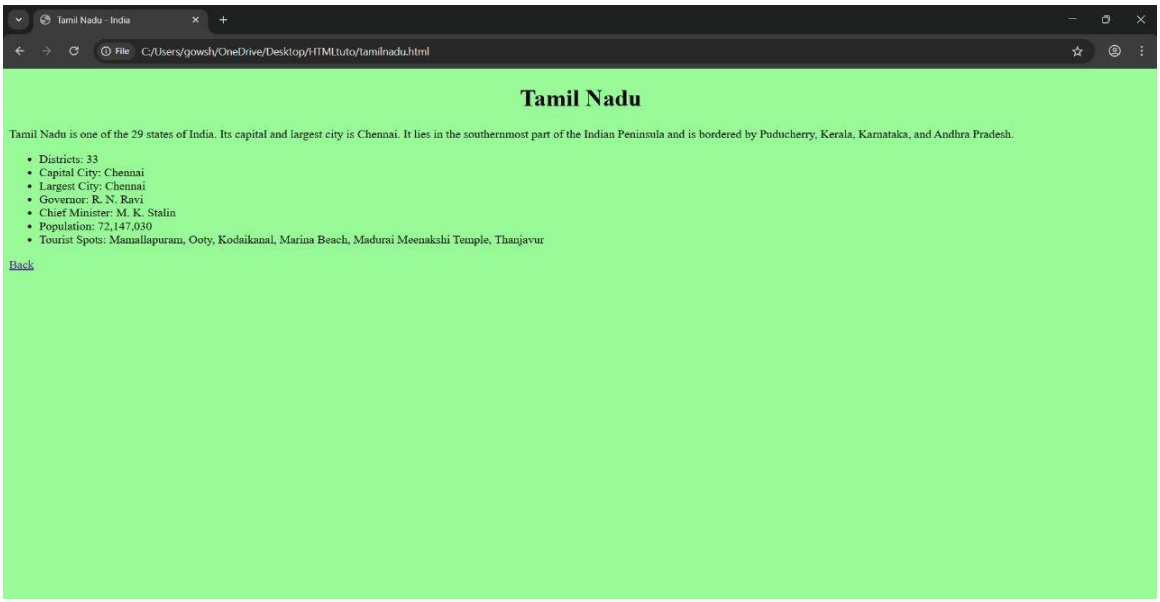
Tamilnadu.html

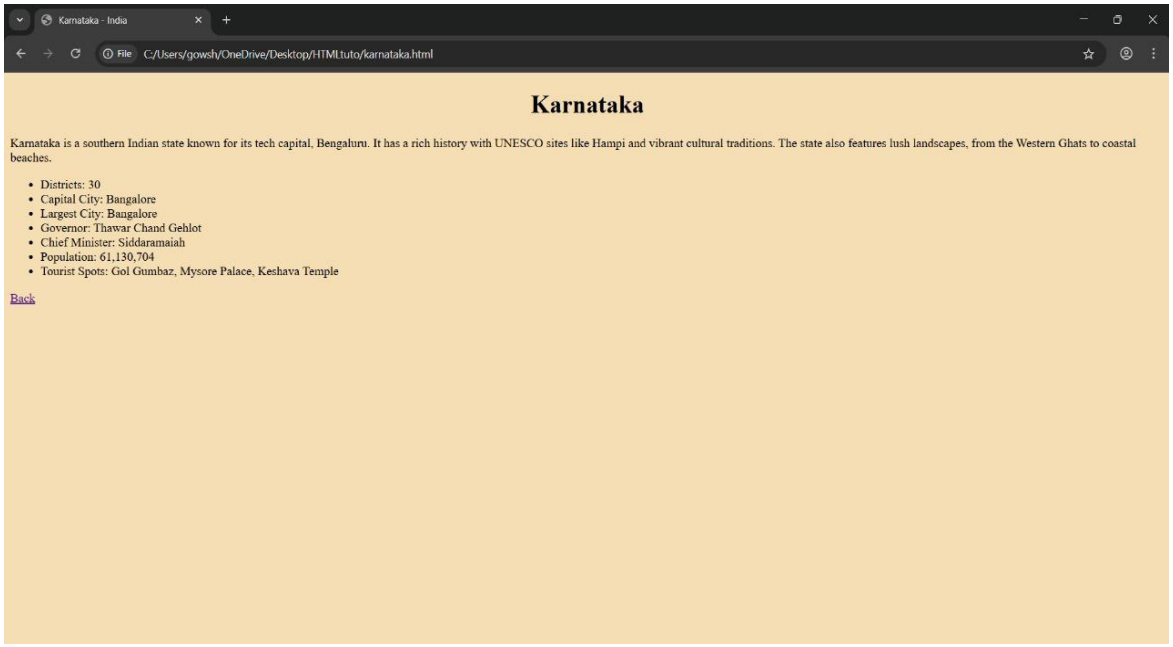
```
<!DOCTYPE html>
<html>
<head>
  <title>Tamil Nadu - India</title>
</head>
<body style="background-color: palegreen;">
  <h1 style="text-align: center;">Tamil Nadu</h1>
  <p>
    Tamil Nadu is one of the 29 states of India. Its capital and largest city is Chennai. It lies in the southernmost part of the Indian Peninsula and is bordered by Puducherry, Kerala, Karnataka, and Andhra Pradesh.
  </p>
  <ul>
    <li>Districts: 33</li>
    <li>Capital City: Chennai</li>
    <li>Largest City: Chennai</li>
    <li>Governor: R. N. Ravi</li>
    <li>Chief Minister: M. K. Stalin</li>
    <li>Population: 72,147,030</li>
```

```
    <li>Tourist Spots: Mamallapuram, Ooty, Kodaikanal, Marina Beach, Madurai Meenakshi Temple, Thanjavur</li>
  </ul>
  <a href="main.html">Back</a>
</body>
</html>
```

OUTPUT:







Result:

The web page was successfully created using HTML with an embedded image map. Hotspots were defined for Tamil Nadu and Karnataka, and clicking on each hotspot correctly navigated to a separate page displaying detailed information about the respective state.

Ex No : 02	Creation of Web Page Using All Types of Cascading Style Sheets	Page no :
Date :		

Aim:

To design a web page using all three types of Cascading Style Sheets (CSS): Inline, Internal, and External, to demonstrate styling techniques in web development.

Description:

This experiment demonstrates the use of all three types of Cascading Style Sheets (CSS) in a single web page: **Inline**, **Internal**, and **External**. CSS is used to control the presentation and layout of HTML elements.

- **Inline CSS** is applied directly within an HTML tag using the style attribute.
- **Internal CSS** is defined within the tag inside the section of the HTML document.
- **External CSS** is written in a separate .css file and linked to the HTML using the tag.

In this experiment, a web page about flowers is styled using all three methods. The external stylesheet (style.css) controls the overall layout and link behavior, the internal stylesheet styles the paragraph, and inline styles are used for specific elements like the table and span text. This approach helps understand how different CSS types interact and affect the final appearance of a web page.

Algorithm:

Step 1: Start

Step 2: Open a text editor and create a new HTML file named style.html.

Step 3: Define the basic HTML structure using:

- to declare the document type.
- to begin the HTML document.
- to include metadata and style references.
- to contain the visible content of the webpage.

Step 4: Inside the section:

- Step 4.1: Set the title using .
- Step 4.2: Add a comment indicating the use of an external style sheet.
- Step 4.3: Link the external CSS file using .

- Step 4.4: Add a comment indicating the use of an embedded style sheet.
- Step 4.5: Define embedded CSS rules using the tag
- Style the <p>tag with:
  * background-color: lightgrey
  * text-align: justify
  * margin: 2em 7em

Step 5: Inside the <body>section:

Step 5.1: Assign id="body" to the tag to apply external styles.

Step 5.2: Add a heading with the text "FLOWER".

Step 5.3: Insert a paragraph describing flowers.

- Step 5.3.1: Use an inline style to format the word “Flower” with:
- font-weight: 200
- font-size: x-large
- font-family: fantasy

Step 5.4: Add a comment indicating the use of inline style sheet.

Step 5.5: Create a <table> with inline styles

1. margin: auto
2. text-align: center
3. padding: 3px

Step 5.5.1: Inside the table, insert a to apply external styles.

Step 5.5.2: Add an unordered list with flower names as list items

Step 5.5.3: Each contains a hyperlink to an image of the flower.

Step 6: Create a new file named style.css and define the following external styles:

- Step 6.1: Style h1, h2 with:
  - text-decoration: underline
  - font-style: italic
  - text-align: center
- Step 6.2: Style #body with:
  - background-color: tan
  - border: 2px dotted red
  - text-align: center
- Step 6.3: Style .div with:
  - border: 2px solid peru
  - padding: 10px
- Step 6.4: Use universal selector * to apply:
  - letter-spacing: 1px

- Step 6.5: Define link states:
  - a:link → color: black
  - a:visited → color: yellow
  - a:hover → color: green
  - a:active → color: blue
- Step 6.6: Style list items ul li with:
- font-size: small

Step 8 : Save both style.html and style.css files.

Step 9 : Open style.html in a web browser to view the styled webpage

Step10: Verify that all three types of CSS—external, embedded, and inline—are applied

Step 11: Stop

Program:

Style.html

```
<!DOCTYPE html>
<html>
<head>
<title>FLOWERS</title>
<!-- External Style Sheet -->
<link rel="stylesheet" type="text/css" href="style.css">
<!-- Embedded Style Sheet -->
<style type="text/css">
p {
background-color: lightgrey;
text-align: justify;
margin: 2em 7em;
}
</style>
</head>
<body id="body">
<h1>FLOWER</h1>
```

Flower sometimes known as a bloom or blossom, is the reproductive structure found in flowering plants.
The flower is God's finest workmanship in the world. It is his finest gift to mankind.
We have seen flowers of many kinds and many colors. In India we see flowers like:

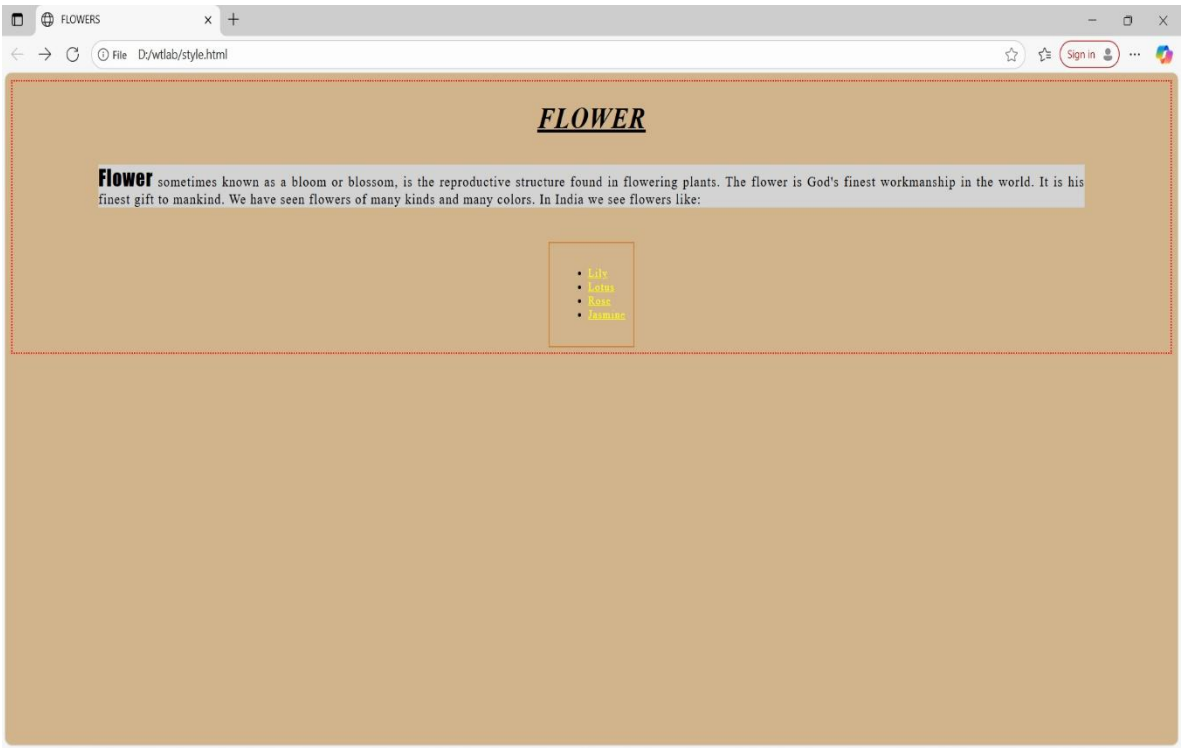
<!-- Inline Style Sheet -->
<table style="margin: auto; text-align: center; padding: 3px;">
<tr>
<td align="left">

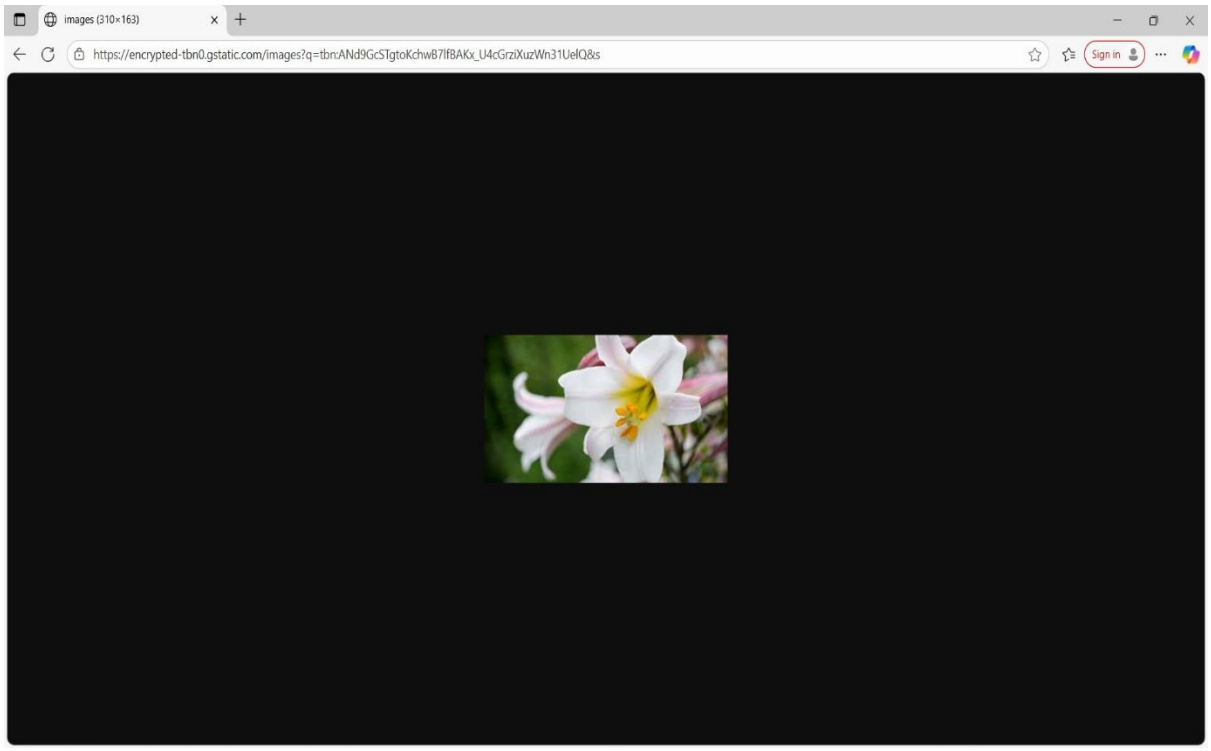
Lily
Lotus
Rose
Jasmine
</td> </tr>
</table>
</body>
</html>

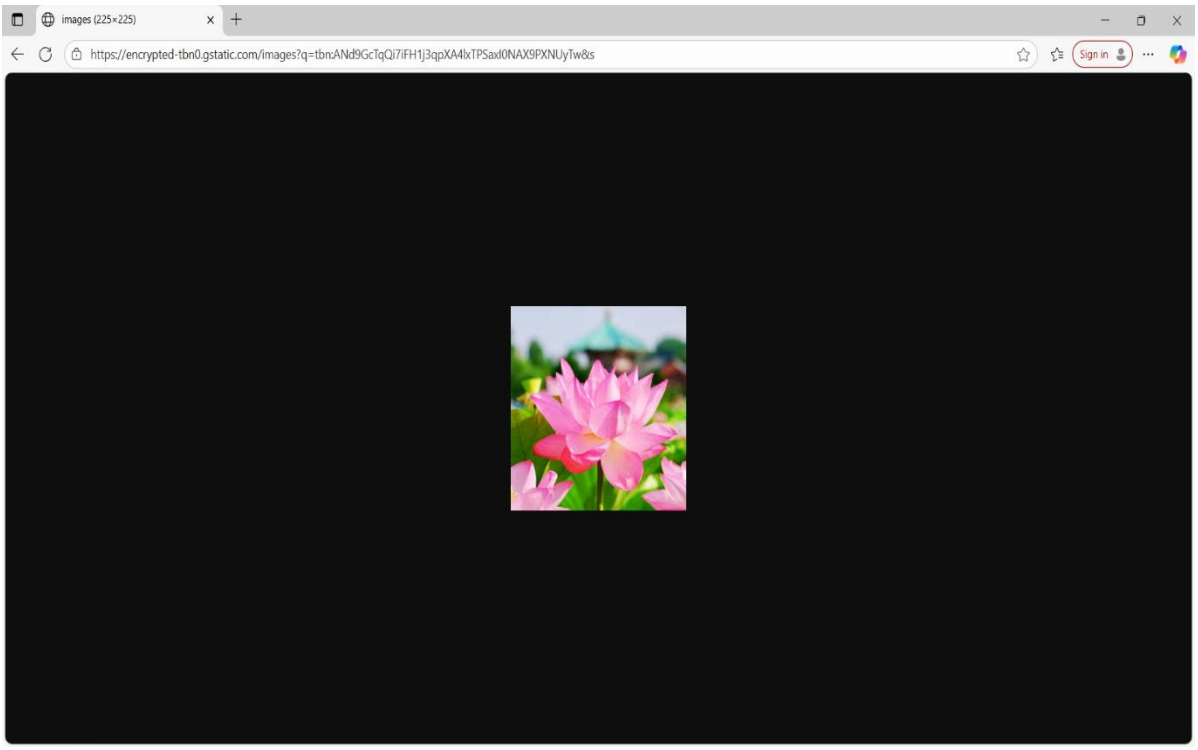
Style.css

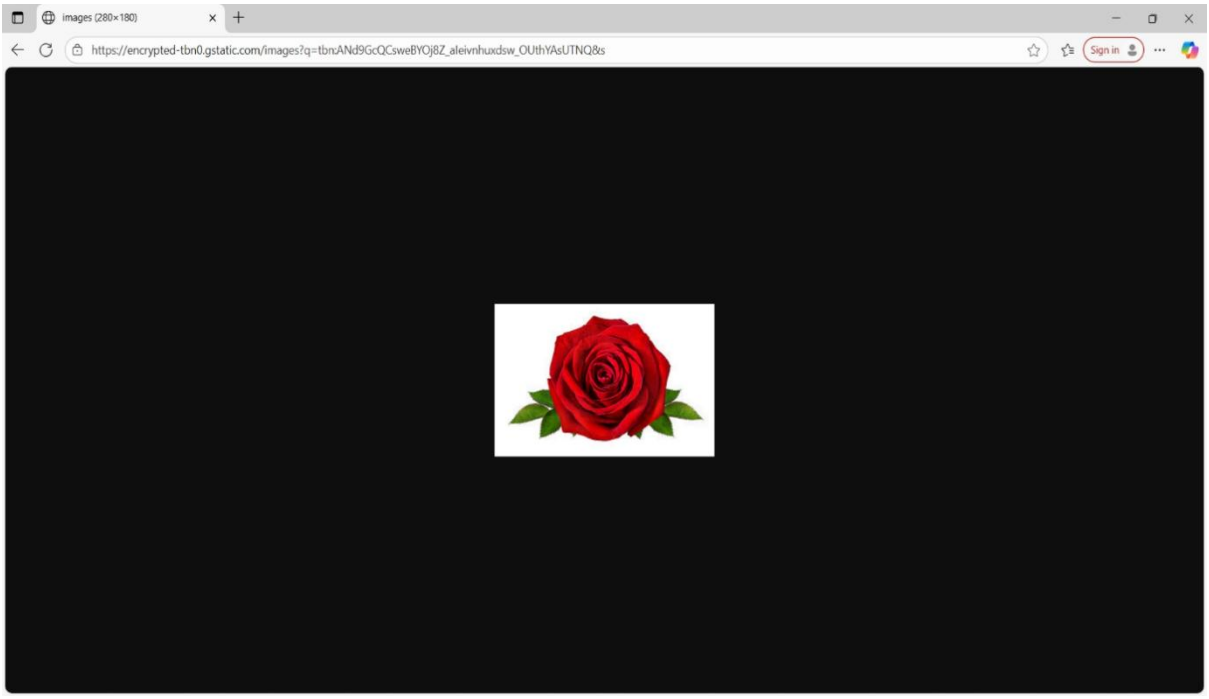
```
H1, h2 {
Text-decoration: underline;
Font-style: italic;
Text-align: center; }
#body {
Background-color: tan;
Border: 2px dotted red;
Text-align: center; }
.div {
Border: 2px solid peru;
Padding: 10px; }
{
Letter-spacing: 1px; }
A:link {
Color: black; }
A:visited {
Color: yellow;
}
A:hover {
Color: green;
}
A:active {
Color: blue;
}
Ul li {
Font-size: small;
}
```

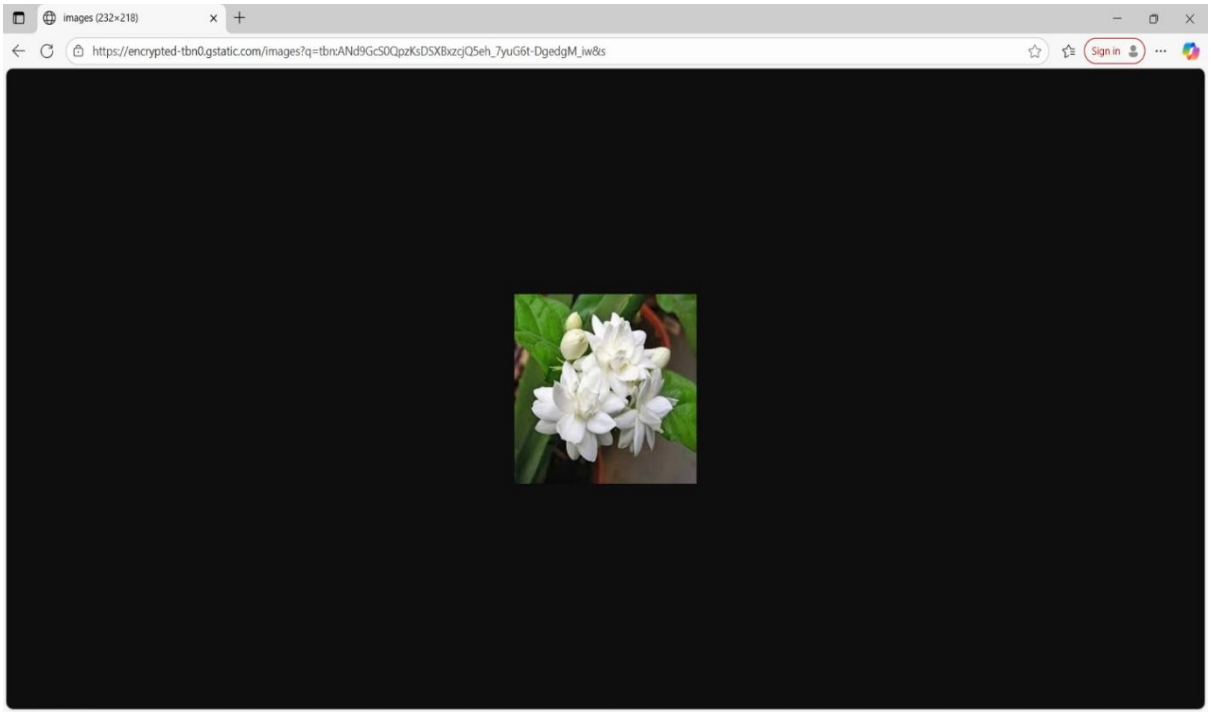
OUTPUT:


FLOWERS
File D:/wtlab/style.html
Sign in
FLOWER
Flower sometimes known as a bloom or blossom, is the reproductive structure found in flowering plants. The flower is God's finest workmanship in the world. It is his finest gift to mankind. We have seen flowers of many kinds and many colors. In India we see flowers like:
Lily
Lotus
Rose
Jasmine


images (310×163)
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSTgtoKchwB7IfBAKx_U4cGrziXuzWn31UelQ&s
Sign in


images (225×225)
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTqQi7iFH1j3qpXA4lxTPSaxl0NAX9PXNUyTw&s
Sign in


images (280×180)
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQCsweBYOj8Z_aleivnhuxdsw_OUthYAsUTNQ&s
Sign in



Result:

The web page was successfully created using all three types of Cascading Style Sheets—Inline, Internal, and External. Each style type was applied correctly to demonstrate its effect on the layout, formatting, and visual presentation of the content.

Ex No : 03	Client-Side Validation of Web Form Controls Using DHTML	Page no :
Date :		

Aim:

To develop a web page using Dynamic HTML (DHTML) that performs client-side validation of form controls such as text fields, radio buttons, checkboxes, and dropdowns before submission.

Description:

This experiment demonstrates how to perform client-side validation of web form controls using Dynamic HTML (DHTML), which combines HTML, CSS, and JavaScript. The validation ensures that required fields are filled before the form is submitted to the server. In this example, JavaScript is used to check whether the "Name" field is empty. If the field is left blank, an alert message is displayed and the form submission is prevented. This technique improves user experience by providing immediate feedback and reducing unnecessary server requests.

Algorithm:

Step 1: Start

Step 2: Create an HTML form with input fields (e.g., Name)

Step 3: Write a JavaScript function to validate the form inputs

Step 4: In the function, check if the required field is empty

Step 5: If the field is empty, display an alert message and return false to prevent form submission

Step 6: Link the validation function to the form's onsubmit event

Step 7: Create a separate HTML page (wlcm.html) to display a welcome message uponsuccessful submission

Step 8: Save and run the HTML file in a browser to test the validation

Step 9: Stop

Program:

control.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation</title>
  <script>
    function validateForm() {
      var x = document.forms["myForm"]["fname"].value;
      if (x == null || x == "") {
        alert("Name must be filled out");
        return false;
      } }
  </script>
</head>
<body>
  <form name="myForm" action="wlcm.html" onsubmit="return validateForm()"
method="post">
    Name: <input type="text" name="fname">
    <input type="submit" value="Submit">
  </form>
</body></html>
```

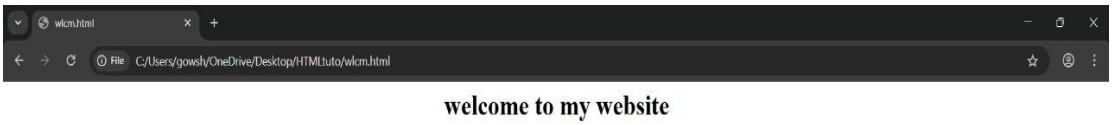
wlcm.html

```
<html>
<body>
<center> <h1>welcome to my website</h1> </center>
</body>
</html>
```

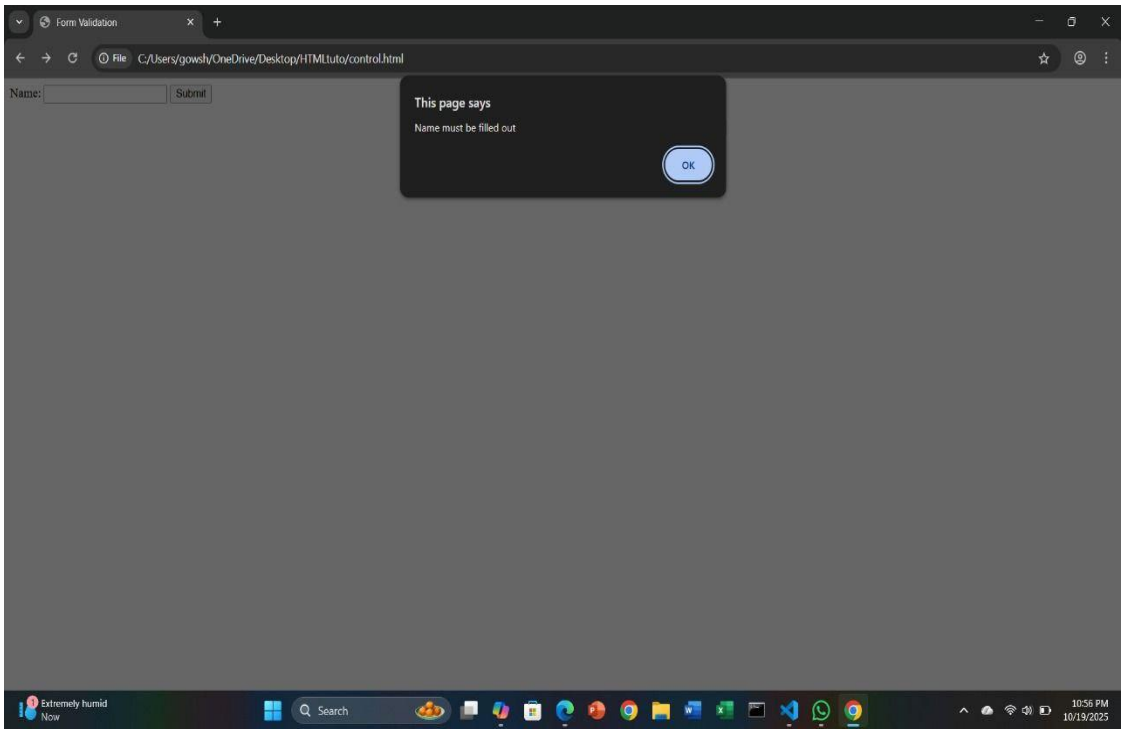
OUTPUT:


Form Validation
File C:/Users/gowsh/OneDrive/Desktop/HTMLtuto/control.html
Name: Jackie
Submit


Rain coming
11:18 pm
Search
10:55 PM
10/19/2025


wlcm.html
File C:/Users/gowsh/OneDrive/Desktop/HTMLtuto/wlcm.html
welcome to my website


Extremely humid
Now
Search
10:55 PM
10/19/2025



Result:

The web page was successfully developed using DHTML to perform client-side validation.

Ex No : 04	Installation of Apache Tomcat Web Server	Page no :
Date :		

Aim:

To write the steps for installing the Apache Tomcat web server in the system.

Apache Tomcat HTTP Server:

Apache Tomcat is a Java-capable HTTP server, which could execute special Java programs known as "Java Servlet" and "Java Server Pages (JSP)". Tomcat is an open- source project, under the "Apache Software Foundation" (which also provides the most used, open-source, industrial-strength Apache HTTP Server). The mother site for Tomcat is http://tomcat.apache.org. Alternatively, you can find Tomcat via the Apache mother site @http://www.apache.org. Tomcat was originally written by James Duncan Davison (then working in Sun) in 1998, based on an earlier Sun's server called Java Web Server (JWS). It began at version 3.0 after JWS 2.1 it was replaced. Sun subsequently made Tomcat open-source and gave it to Apache.

The various Tomcat releases are:

- Tomcat 3.0 (1999): Reference Implementation (RI) for Servlet 2.2 and JSP 1.1.
- Tomcat 4.1 (Sep 2002): RI for Servlet 2.3 and JSP 1.2.
- Tomcat 5.0 (Dec 2003): RI for Servlet 2.4 and JSP 2.0.
- Tomcat 6.0 (Feb 2007): RI for Servlet 2.5 and JSP 2.1.
- Tomcat 7.0 (Jan 2011): RI for Servlet 3.0, JSP 2.2 and EL. 2.2.
- Tomcat 8.0 (Jun 2014): RI for Servlet 3.1, JSP 2.3, EL 3.0 and WebSocket 1.0. Tomcat8.5
- (June 2016) supports HTTP/2, OpenSSL, TLS virtual hosting and JASPIC 1.1.
- Tomcat 9.0 (Jan 2018): RI for Servlet 4.0, JSP 2.3, EL 3.0, WebSocket 1.0, JASPIC 1.1.
- Tomcat 10.0 (???):

How to Install Tomcat and Get Started with Java Servlet Programming

STEP 1: Create a Directory to Keep all your Works

I shall assume that you have created a directory called "c:\my WebProject" (for Windows) or "-\my WebProject" (for Mac OS X) in your earlier exercises. Do it otherwise. This step is important; otherwise, you will be out-of-sync with this article and will not be able to find your files later.

STEP 2: Download and Install Tomcat For Windows

Goto http://tomcat.apache.org Under "Tomcat 9.0. (xx) Released", where {xx) is the latest update number =>Click "Download" Under "9.0. (xx)"=> Binary Distributions =>Core=> "zip" (e.g., "apache-tomcat-9.0.(xx).zip", about 11 MB).

UNZIP the downloaded file into your project directory "c:\my WebProject". Tomcat shall be unzipped into the directory "c:\my WebProject\apache-tomcat-9.0. (xx)".
For EASE OF USE, "c:\my WebProject\tomcat". we shall shorten and rename this directory to “<c:\myWebProject\tomcat”

Take note of Your Tomcat Installed Directory. Hereafter, I shall refer to the Tomcat installed directory as <TOMCAT_HOME>.

Tomcat's Sub-Directories

Take a quick look at the Tomcat installed directory. It contains the these sub-directories:

bin: contains the binaries and scripts (e.g., startup.bat and shutdown.bat for Windows; startup.sh and shutdown.sh for Unixes and Mac OS X).

conf: contains the system-wide configuration files, such as server.xml, web.xml, and context.xml.

webapps: contains the webapps to be deployed. You can also place the WAR (Webapp Archive) file for deployment here.

lib: contains the Tomcat's system-wide library JAR files, accessible by all web apps. You could also place external JAR file (such as MySQL JDBC Driver) here.

logs: contains Tomcat's log files. You may need to check for error messages here.

work: Tomcat's working directory used by JSP, for JSP-to-Servlet conversion.

STEP 3: Create an Environment Variable JAVA HOME (For Windows)

You need to create an environment variable (system variable available to all applications) called "JAVA_HOME", and set it to your JDK installed directory.

Many Java applications (such as Tomcat) require the environment variable JAVA HOME to be set to the JDK installed directory.

To set the JAVA_HOME environment variable:

First, find your JDK installed directory. For JDK 11, the default is "c:\Program Files\Java\jdk-11.0.{x)", where "{x} is the update number. Use your "File Explorer" to find this directory and take note of your update number {x}.

Check if JAVA_HOME is already set. Start a CMD and issue:

set JAVA_HOME

If you get a message "Environment variable JAVA_HOME not defined", proceed to the next step.

If you get "JAVA_HOME=C:\Program Files\Java\jdk-11.0.{x}", verify that it is set correctly to your JDK directory. If not, proceed to the next step.

To set the environment variable JAVA HOME in Windows 10:

Launch "Control Panel"=> (Optional) "System and Security"=> "System" =>Click "Advanced system settings" on the left pane.

Switch to "Advanced" tab=> Click "Environment Variables"

Under "System Variables" (the bottom pane)=> Click "New" (or Look for "JAVA HOME" and "Edit" if it is already set) => In "Variable Name", enter "JAVA_HOME" In "Variable Value", enter your JDK installed directory you noted in Step 1. (In the latest Windows 10: you can push the "Browse Directory" button and navigate to the JDK installed directory to avoid typo error.)

To verify, RE-START a CMD (restart is needed to refresh the environment variables) and issue:

set JAVA_HOME

JAVA_HOME=c:\Program Files\Java\jdk-11.0. (x) < Verify that this is YOUR JDK installed directory

Notes: Windows' environment variables (such as JAVA_HOME, PATH) are NOT case-sensitive.

STEP 4: Configure the Tomcat Server

The Tomcat configuration files, in XML format, are located in the "conf" sub-directory of your Tomcat installed directory, e.g. "c:\my WebProject\tomcat\conf" (for Windows) or "~/my Web Project/tomcat/conf" (for Mac OS X). The important configuration files are:

server.xml web.xml

context.xml

Make a BACKUP of the configuration files before you proceed!!!

Step 4(a) "conf\server.xml" - Set the TCP Port Number

Use a programming text editor (e.g., Sublime Text, Atom) to open the configuration file "server.xml".

The default TCP port number configured in Tomcat is 8080, you may choose any number between 1024 and 65535, which is not used by existing applications. We shall choose 9999 in this article. (For production server, you should use port 80, which is pre-assigned to HTTP server as the default port number.) Locate the following lines (around Line 69) that define the HTTP connector, and change port="8080" to port="9999".

<!-- A "Connector" represents an endpoint by which requests are received and responses are returned. Documentation at:

Java HTTP Connector: /docs/config/http.html

Java AJP Connector: /docs/config/ajp.html APR

(HTTP/AJP) Connector: /docs/apr.html

Define a non-SSL HTTP/1.1 Connector on port 8080

<Connector port="9999" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443"/>

Step 4(b) "conf\web.xml" - Enable Directory Listing

Again, use a programming text editor to open the configuration file "web.xml".

We shall enable directory listing by changing "listings" from "false" to "true" for the "default" servlet. This is handy for test system, but not for production system for security.

Locate the following lines (around Line 108) that define the "default" servlet; and change the "listings" from "false" to "true".

<servlet>

<servlet-name>default</servlet-name>

<servlet-class>org.apache.catalina.servlets. DefaultServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>(0)</param-value>

</init-param>

<init-param>

<param-name>listings</param-name>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

Step 4(c) "conf\context.xml" – Enabling Automatic Reload

We shall add the attribute reloadable="true" to the <Context> element to enable automatic reload after code changes. Again, this is handy for test system but not recommended for production, due to the overhead of detecting changes.

Locate the <Context> start element (around Line 19), and change it to <Context reloadable="true">.

<Context reloadable="true">

……

……

</Context>

STEP 5: Start Tomcat Server

The Tomcat's executable/programs and scripts are kept in the "bin" sub-directory of the Tomcat installed directory.

Step 5(a) Start Server

I shall assume that Tomcat is installed in "c:\my WebProject\tomcat". Launch a CMD shelland issue:

C: // Change drive

cd \my WebProject\tomcat\bin // Change directory to your Tomcat's binary directory startup

// Run startup.bat to start tomcat server

Step 5(b) Start a Client to Access the Server

Start a browser (Firefox, Chrome) as an HTTP client. Issue URL "http://localhost:9999"

toaccess the Tomcat server's welcome page. The hostname "localhost" (with IP address of

127.0.0.1) is meant for local loop-back testing within the same machine. For users on the other machines over the net, they have to use the server's IP address or DNS domain name in the form of "http://server Hostname OrIPAddress:9999".

(Optional) Try issuing URL http://localhost:9999/examples to view the servlet and JSP

examples. Try running some of the servlet examples.

Step 5(c) Shutdown Server

You can shutdown the Tomcat server by either:

Press Ctrl-C on the Tomcat console; OR

Run the "<TOMCAT_HOME>\bin\shutdown.bat" script. Open a new "cmd" and issue:

e:// Change the current drive

cd \myWebProject\tomcat\bin// Change the directory to your Tomcat's binary directory

shutdown// Run shutdown.bat to shut down the server

Result:

Thus, the installation of the Apache Tomcat web server was done successfully.

Ex No : 05	Java Servlet Programs for Form Handling and Session Tracking	Page no :
Date :		

Aim:

To develop Java servlet programs that demonstrate form handling by invoking servlets from HTML forms and implement session tracking to maintain user-specific data across multiple requests.

Description:

This experiment demonstrates how to build a simple login application using Java Servlets and cookies for session tracking. The system includes three servlets—LoginServlet, ProfileServlet, and LogoutServlet—along with supporting HTML pages.

- The **LoginServlet** handles form submission, validates credentials, and sets a cookie to track the user session.
- The **ProfileServlet** retrieves the cookie to display personalized content if the user is logged in.
- The **LogoutServlet** clears the cookie to log the user out and redirects to the login page.

The web.xml deployment descriptor maps servlet names to their respective URLs and sets the welcome page. This setup illustrates how servlets interact with HTML forms and manage user sessions using cookies.

Algorithm:

Step 1: Start

Step 2: Create HTML pages (index.html, login.html, link.html) for navigation and form input

Step 3: Configure web.xml to map servlets and define the welcome page

Step 4: Implement LoginServlet to:

- Read form data
- Validate credentials
- Set a cookie with the username

- Redirect to ProfileServlet on success

Step 5: Implement ProfileServlet to:

- Read cookies from the request
- Display personalized content if the user is logged in
- Redirect to login page if no valid cookie is found

Step 6: Implement LogoutServlet to:

- Clear the cookie by setting its max age to zero
- Redirect to the login page

Step 7: Deploy the application on Apache Tomcat and test the login flow

Step 8: Stop

Program:

Index.html

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="ISO-8859-1">
<title>Servlet Login Example</title>
</head>
<body>
<h1>Welcome to Login App by Cookie</h1>
<a href="login.html">Login</a>|
<a href="LogoutServlet">Logout</a>|
<a href="ProfileServlet">Profile</a>
</body>
</html>
```

Link.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Links</title>
</head>
<body>
    <a href="login.html">Login</a> |
    <a href="LogoutServlet">Logout</a> |
    <a href="ProfileServlet">Profile</a>
     <hr>
     <p>... other content goes here ...</p>
</body> </html>
```

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
</head>
<body>
    <form action="LoginServlet" method="post">
        Name: <input type="text" name="name"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="login">
    </form>
</body> </html>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        version="2.5">
    <!-- LoginServlet Configuration -->
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>com.javatpoint.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/LoginServlet</url-pattern>
    </servlet-mapping>
    <!-- ProfileServlet Configuration -->
    <servlet>
        <servlet-name>ProfileServlet</servlet-name>
        <servlet-class>com.javatpoint.ProfileServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ProfileServlet</servlet-name>
        <url-pattern>/ProfileServlet</url-pattern>
    </servlet-mapping>
    <!-- LogoutServlet Configuration -->
    <servlet>
        <servlet-name>LogoutServlet</servlet-name>
       <servlet-class>com.javatpoint.LogoutServlet</servlet-class>
```

```
    </servlet>
    <servlet-mapping>
        <servlet-name>LogoutServlet</servlet-name>
        <url-pattern>/LogoutServlet</url-pattern>
    </servlet-mapping>

    <!-- Add this section to define your default page -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

LoginServlet.java

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LoginServlet extends HttpServlet {
      protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          request.getRequestDispatcher("link.html").include(request, response);
          String name = request.getParameter("name");
```

```
        String password = request.getParameter("password");
         if (name == null || password == null) {
            out.print("Error: Please provide both username and password.");
            return;
        }
         if ("admin123".equals(password)) {
            out.print("You are successfully logged in!");
            out.print("<br>Welcome, " + name);
            Cookie ck = new Cookie("name", name);
            ck.setMaxAge(3600);
            response.addCookie(ck);
            response.sendRedirect("ProfileServlet");
         } else {
            out.print("Sorry, username or password error!");
            response.sendRedirect("login.html");
        }
       out.close();
    } }
```

LogoutServlet.java

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

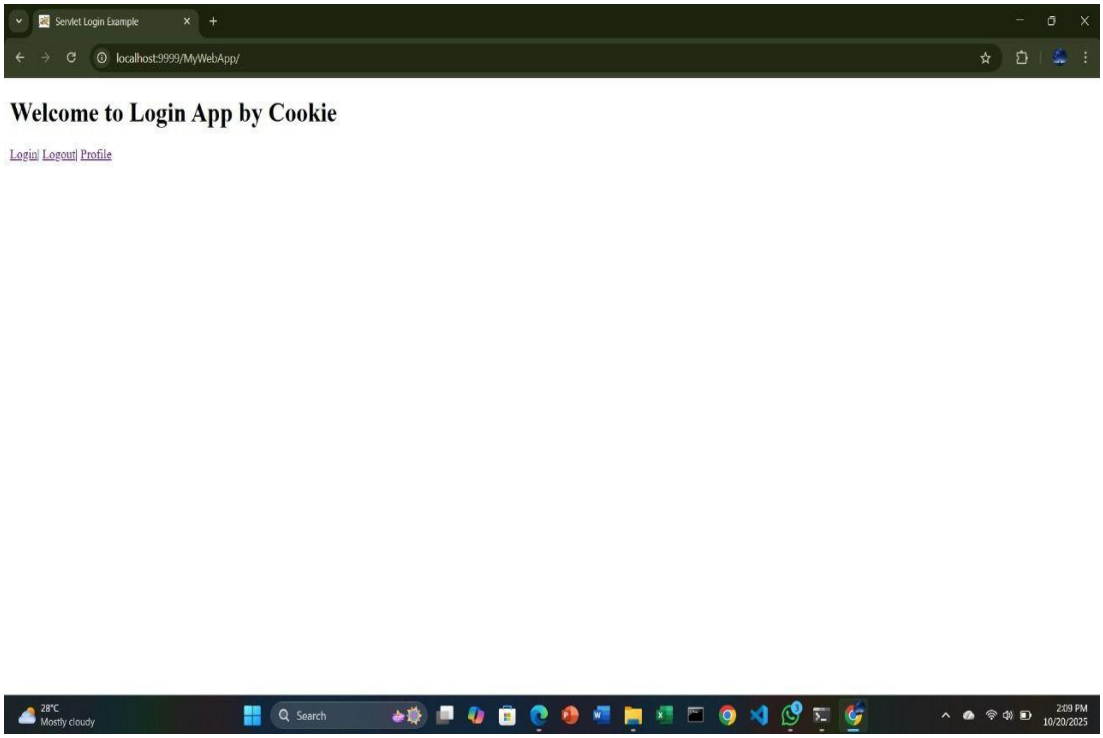
```
public class LogoutServlet extends HttpServlet {
      protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
       request.getRequestDispatcher("link.html").include(request, response);
       Cookie ck = new Cookie("name", "");
       ck.setMaxAge(0);
       response.addCookie(ck);
       out.print("You are successfully logged out!");
       response.sendRedirect("login.html");
   }
}
```

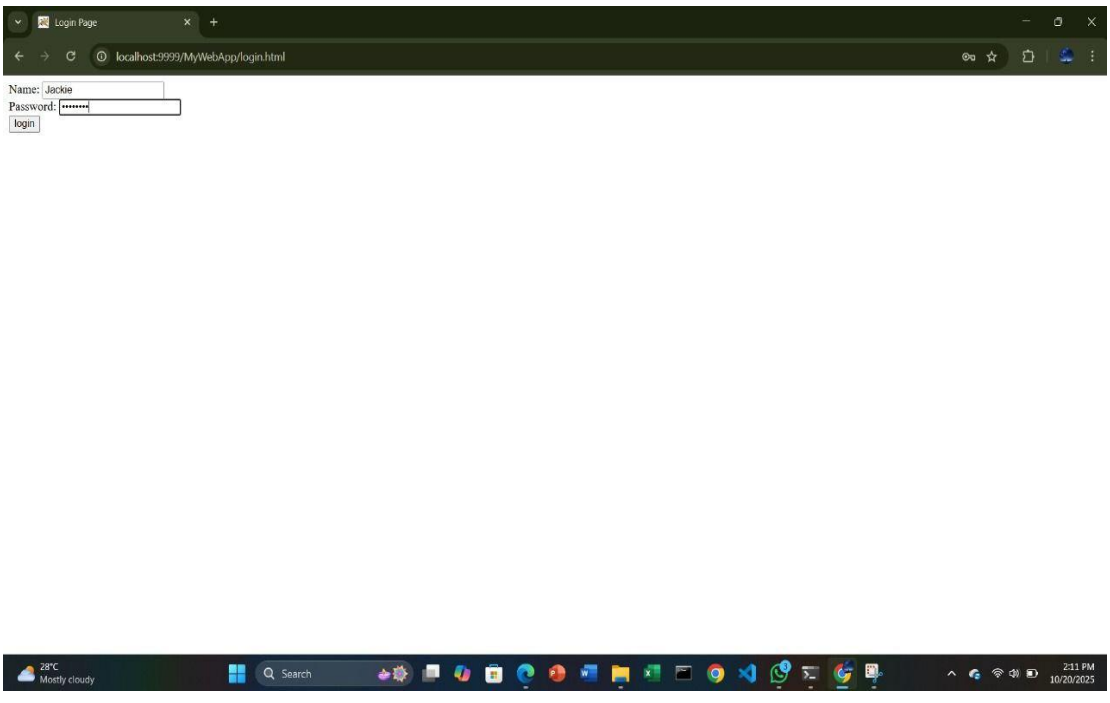
ProfileServlet.java

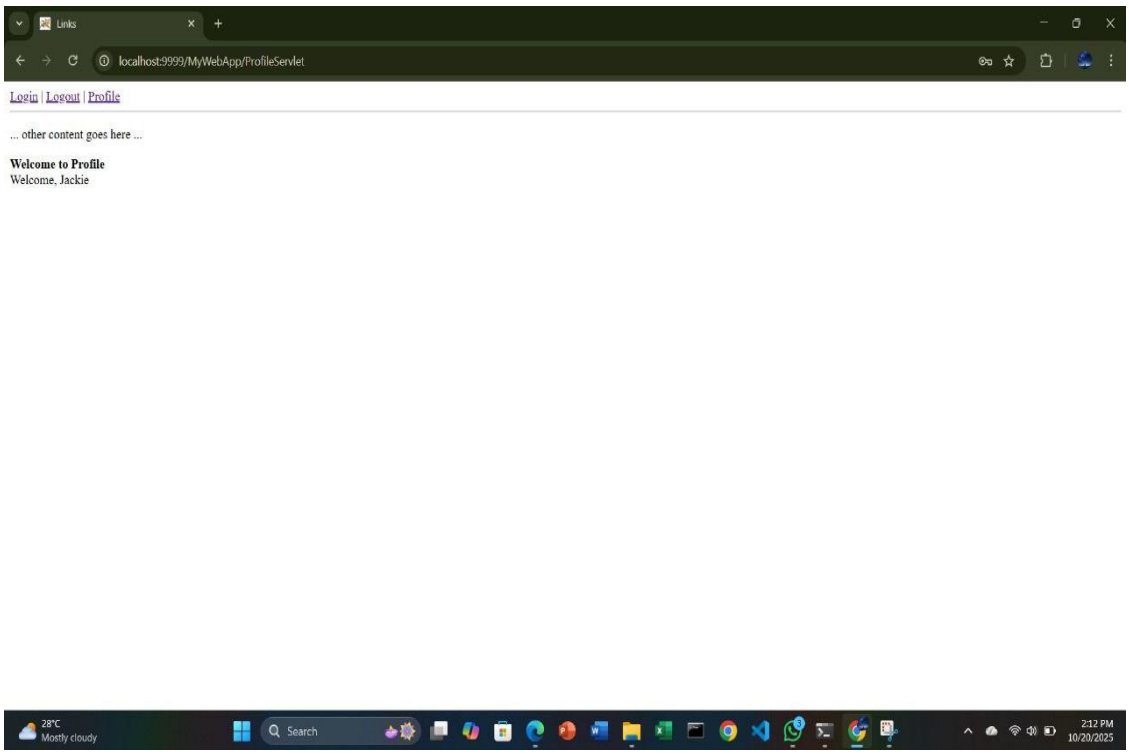
```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ProfileServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
      request.getRequestDispatcher("link.html").include(request, response);
```

```
    Cookie[] cookies = request.getCookies();
    String name = null;
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if ("name".equals(cookie.getName())) {
                name = cookie.getValue();
                break;
            }
        }
    }
    if (name != null && !name.equals("")) {
        out.print("<b>Welcome to Profile</b>");
        out.print("<br>Welcome, " + name);
    } else {
        out.print("Please login first");
        request.getRequestDispatcher("login.html").include(request, response);
    }
    out.close();
} }
```

OUTPUT:


Servlet Login Example
localhost:9999/MyWebApp/
Welcome to Login App by Cookie
Login| Logout| Profile
28°C
Mostly cloudy
Search
2:09 PM
10/20/2025


Login Page
localhost:9999/MyWebApp/login.html
Name: Jackie
Password:
login
28°C
Mostly cloudy
Search
2:11 PM
10/20/2025



Result:

The login application was successfully developed using Java Servlets and cookies. The servlets correctly handled form submission, maintained user sessions through cookies, and enabled personalized profile access and secure logout functionality.

Ex No : 6a	Creation of Three-Tier Applications Using JSP and Databases For conducting Online Examination	Page no :
Date :		

Aim:

To develop a three-tier web application using JSP and Oracle database that conducts an online examination, evaluates answers submitted by the user, and displays the result dynamically.

Description:

This experiment demonstrates how to build a simple online examination system using JavaServer Pages (JSP) and Oracle database.

- The front-end (ExamClient.html) presents multiple-choice questions to the user.
- The middle-tier (ExamServer.jsp) processes the submitted answers, compares them with correct answers stored in the database, calculates the total marks, and displays feedback.
- The back-end is an Oracle database table (examTab) that stores correct answers for each question.

The application uses JDBC to connect to the database and retrieve correct answers. It evaluates the user's responses and provides personalized feedback based on the score. The web.xml file configures the welcome page for the application.

Algorithm:

Step 1: Start

Step 2: Create an HTML form (ExamClient.html) with multiple-choice questions using radio buttons

Step 3: Set the form's action to ExamServer.jsp and method to POST

Step 4: In ExamServer.jsp, retrieve submitted answers using request.getParameter()

Step 5: Establish JDBC connection to Oracle database using DriverManager.getConnection()

Step 6: Execute SQL query to fetch correct answers from examTab

Step 7: Compare each submitted answer with the correct answer

Step 8: For each correct match, increment the score by 5 marks

Step 9: Display total marks and feedback message based on score

Step 10: Stop

Program:

ExamClient.html

```
<html>
<head>
<title>Online Exam Client</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>
<h3>Answer the following questions (5 marks for each correct answer)</h3>
<hr/>
<form name="examForm" method="post" action="ExamServer.jsp">
1. All computers must have <br/>
<input type="radio" name="ans1" value="Operating System">Operating System
<input type="radio" name="ans1" value="Application Software">Application Software
<input type="radio" name="ans1" value="CD Drive">CD Drive
<input type="radio" name="ans1" value="Microsoft word">Microsoft word
<br/><br/>
2. The term PC means <br/>
<input type="radio" name="ans2" value="Private Computer">Private Computer
<input type="radio" name="ans2" value="Professional Computer">Professional Computer
<input type="radio" name="ans2" value="Personal Computer">Personal Computer
<input type="radio" name="ans2" value="Personal Calculator">Personal Calculator
<br/><br/>
3.C was developed by?<br/>
<input type="radio" name="ans3" value="Dennis Ritchie">Dennis Ritchie
```

```
<input type="radio" name="ans3" value="Dennis Ritchie">Dennis Ritchie
<input type="radio" name="ans3" value="Stroustrup">Stroustrup
<input type="radio" name="ans3" value="David Ritchie">David Ritchie
<input type="radio" name="ans3" value="Charles Babbage">Charles Babbage
<br/><br/>
<input type="submit" value="Check Your Result"/>
</form>
</body>
</html>
```

ExamServer.jsp

```
<%@ page import="java.sql.*" %>
<html>
<head>
<title>Online Exam Result</title>
<style>
   body {
      background-color: black;
      color: cyan;
      font-family: Courier, monospace;
      text-align: center;
      padding-top: 50px;
   }
h2 { color: yellow; }
</style>
</head>
<body>
<h2>ONLINE EXAM RESULT</h2>
<hr/>
```

```
<%
String str1 = request.getParameter("ans1");
String str2 = request.getParameter("ans2");
String str3 = request.getParameter("ans3");
int mark = 0;
try {
   Class.forName("oracle.jdbc.driver.OracleDriver");
   Connection con = DriverManager.getConnection(
      "jdbc:oracle:thin:@localhost:1521/XEPDB1", "exam", "gk123");
   // Retrieve correct answers from table
   Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery("SELECT answer FROM examTab ORDER BY qid");
   int i = 1;
   while (rs.next()) {
      String correct = rs.getString("answer");
      if (i == 1 && str1 != null && str1.equalsIgnoreCase(correct)) mark += 5;
      if (i == 2 && str2 != null && str2.equalsIgnoreCase(correct)) mark += 5;
      if (i == 3 && str3 != null && str3.equalsIgnoreCase(correct)) mark += 5;
      i++;
   }
   con.close();
%>
<h3>Your Total Marks: <%= mark %></h3>
<%
   if (mark >= 10) {
%>
<p style="color:lime;">Fantastic! You have done really well ...keep learning and moving forward!</p>
```

```
<%
   } else {
%>   <p style="color:red;">Dont worry! Every mistake is a step toward mastery. Try again and you will shine next time!</p>
<%
   }
} catch (Exception e) {
   out.println("<h3 style='color:red;'>Error: " + e.getMessage() + "</h3>");
}
%>
</body>
</html>
```

Web.xml

```
<web-app>
  <display-name>OnlineExam</display-name>
  <welcome-file-list>
    <welcome-file>ExamClient.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Creating Table:

```
CREATE TABLE examTab (
  qid NUMBER PRIMARY KEY,
  question VARCHAR2(200),
  option1 VARCHAR2(100),
  option2 VARCHAR2(100),
  option3 VARCHAR2(100),
  option4 VARCHAR2(100),
  answer VARCHAR2(100));
```

Inserting Records:

INSERT INTO examTab VALUES (1, 'All computers must have', 'Operating System', 'Application Software', 'CD Drive', 'Microsoft Word', 'Operating System');

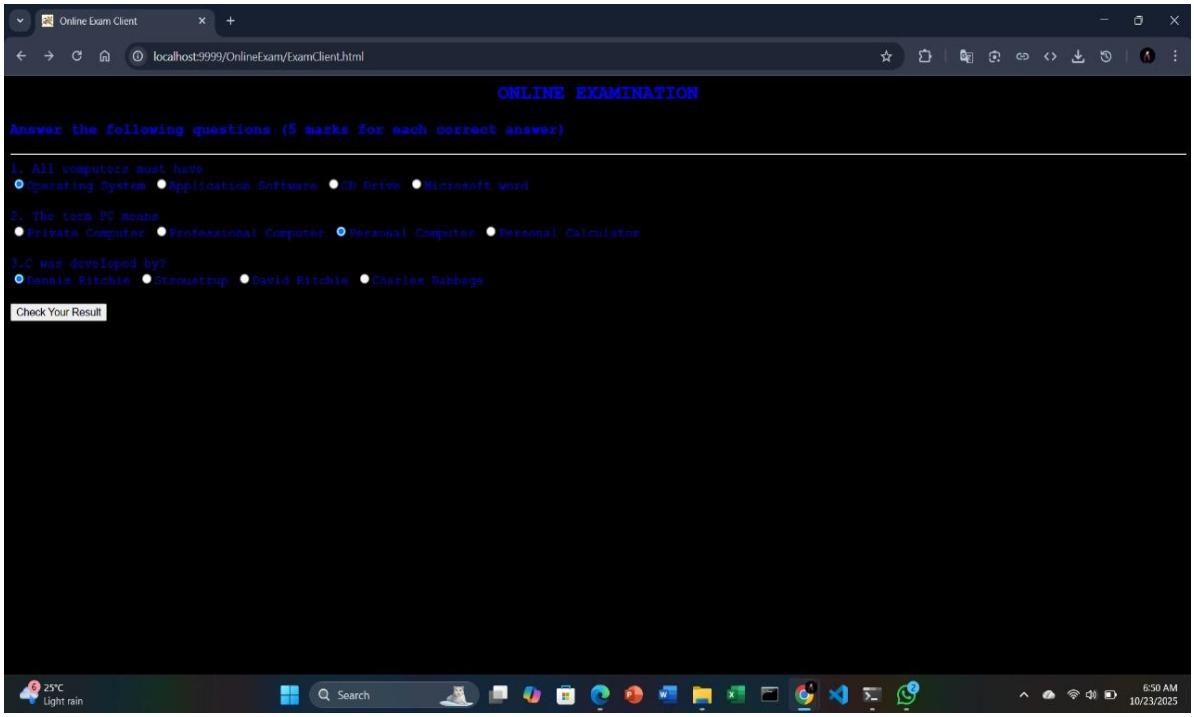
INSERT INTO examTab VALUES (2, 'The term PC means', 'Private Computer', 'Professional Computer', 'Personal Computer', 'Personal Calculator', 'Personal Computer');

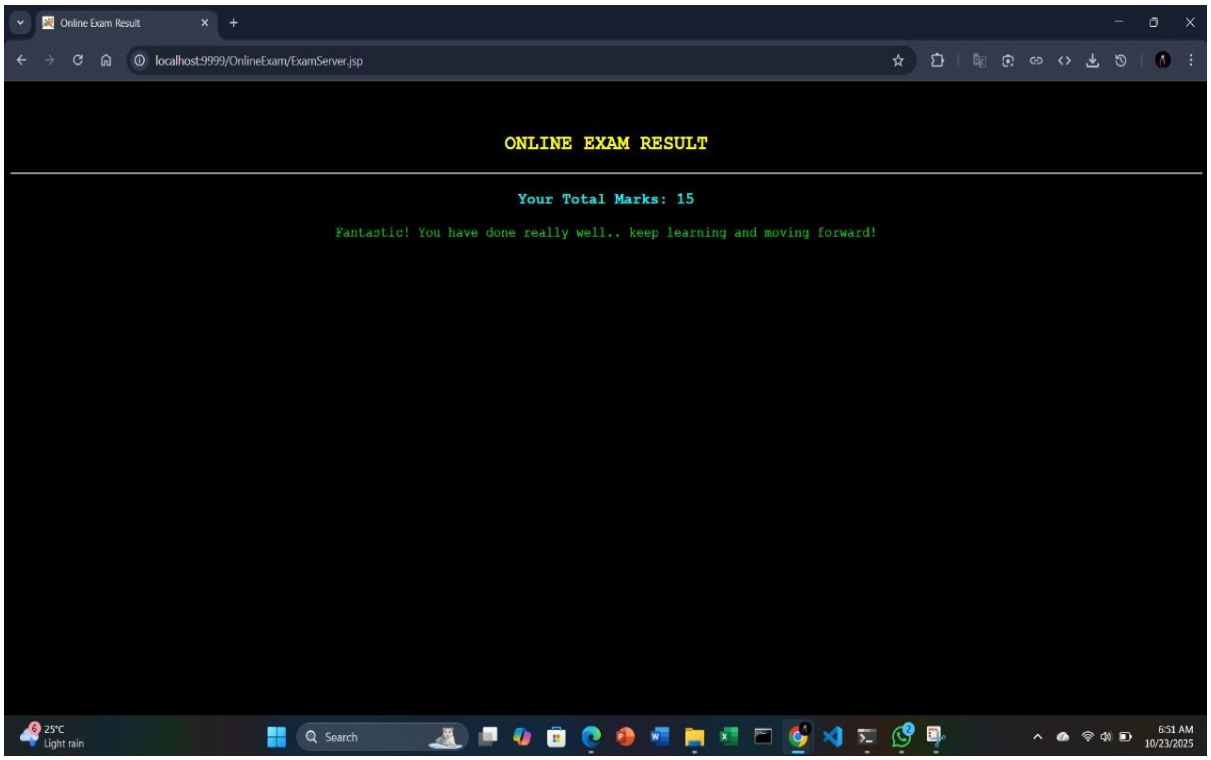
INSERT INTO examTab VALUES (3, 'C was developed by?', 'Dennis Ritchie', 'Stroustrup', 'David Ritchie', 'Charles Babbage', 'Dennis Ritchie');

COMMIT;

```
SQL> CREATE TABLE examTab (
  2    qid NUMBER PRIMARY KEY,
  3    question VARCHAR2(200),
  4    option1 VARCHAR2(100),
  5    option2 VARCHAR2(100),
  6    option3 VARCHAR2(100),
  7    option4 VARCHAR2(100),
  8    answer VARCHAR2(100)
  9  );

Table created.

SQL> INSERT INTO examTab VALUES (1, 'All computers must have', 'Operating System', 'Application Software', 'CD Drive', 'Microsoft Word', 'Operating System')
;

1 row created.

SQL> INSERT INTO examTab VALUES (2, 'The term PC means', 'Private Computer', 'Professional Computer', 'Personal Computer', 'Personal Calculator', 'Personal
Computer');

1 row created.

SQL> INSERT INTO examTab VALUES (3, 'C was developed by?', 'Dennis Ritchie', 'Stroustrup', 'David Ritchie', 'Charles Babbage', 'Dennis Ritchie');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

OUTPUT:





Result:

Thus,the three-tier applications for conducting online examination and displaying student marklist were successfully developed using JSP and database.

Ex No : 6b	Creation of three tiar applications using JSP and Databases for displaying Student Marklist	Page no :
Date :		

Aim:

To design and develop a three-tier web application using HTML, JSP and Database that displays a student's marklist based on the registration number entered by the user.

Description:

This experiment demonstrates how to build a dynamic web application using JSP and JDBC to fetch and display student examination results.

- The front-end () contains a form where the user enters their registration number.
- The middle-tier () processes the input, connects to the Oracle database, retrieves the student's marks, and displays them in a formatted table.
- The back-end is an Oracle database table () that stores student details and subject-wise marks.

The application uses a to securely query the database and display results only if matching record is found. This setup illustrates the three-tier architecture involving presentation, business logic, and data layers.

Algorithm:

Step 1: Start
Step 2: Create an HTML form to accept the register number (regno) from the user
Step 3: Submit the form data to marklist.jsp using the GET method
Step 4: In marklist.jsp, retrieve the regno parameter from the request
Step 5: Load the Oracle JDBC driver and establish a database connection
Step 6: Prepare and execute a SQL query to fetch student details from the database
Step 7: If a matching record is found, display the register number, name, and subject-wise marks in a table
Step 8: If no record is found, display an appropriate message
Step 9: Handle any exceptions and display error messages if needed
Step 10: Provide a link to return to the input form
Step 11: Stop

Program:

stud.html

```
<html>
<head>
<title>Three Tier Application</title>
<style>
  body { color:blue; font-family:courier; text-align:center; }
</style>
</head>
<body>
<h2>EXAMINATION RESULT</h2><hr/>
<form name="f1" method="GET" action="marklist.jsp">
    Enter Your Reg.No: <input type="text" name="regno"/><br/><br/>
    <input type="submit" value="SUBMIT"/>
</form>
</body>
</html>
```

marklist.jsp

```
<%@ page import="java.sql.*" %>
<html>
<head>
<title>Three Tier Application</title>
<style>
  body { color:blue; font-family:courier; text-align:center; }
  table { margin:auto; border-collapse:collapse; }
  th, td { border:1px solid blue; padding:8px; }
</style>
</head>
```

```
<body>
<h2>EXAMINATION RESULT</h2><hr/>
<%
String regno = request.getParameter("regno");
if (regno != null && !regno.trim().isEmpty()) {
   try {
      Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection(
         "jdbc:oracle:thin:@localhost:1521/XEPDB1", "exam", "gk123");
      PreparedStatement ps = con.prepareStatement(
         "SELECT * FROM markTab WHERE rno = ?");
      ps.setInt(1, Integer.parseInt(regno));
      ResultSet rs = ps.executeQuery();
      if (rs.next()) {
%>
         <h3>Register No: <%= rs.getInt("rno") %></h3>
         <h3>Name: <%= rs.getString("name") %></h3>
         <table border="1">
         <tr><th>SUBJECT</th><th>MARK</th></tr>
         <tr><td>Web Technology</td><td><%= rs.getInt("webtech") %></td></tr>
         <tr><td>Computer Networks</td><td><%= rs.getInt("compnet") %></td></tr>
         <tr><td>Compiler Design</td><td><%= rs.getInt("compdesign") %></td></tr>
         <tr><td>Distributed Computing</td><td><%= rs.getInt("distcomp") %></td></tr>
         <tr><td>Artificial Intelligence</td><td><%= rs.getInt("ai") %></td></tr>
</table>
<%
      } else {
         out.println("<h3>No record found for Register No: " + regno + "</h3>");
      }
```

%>

Back

</body>

</html>

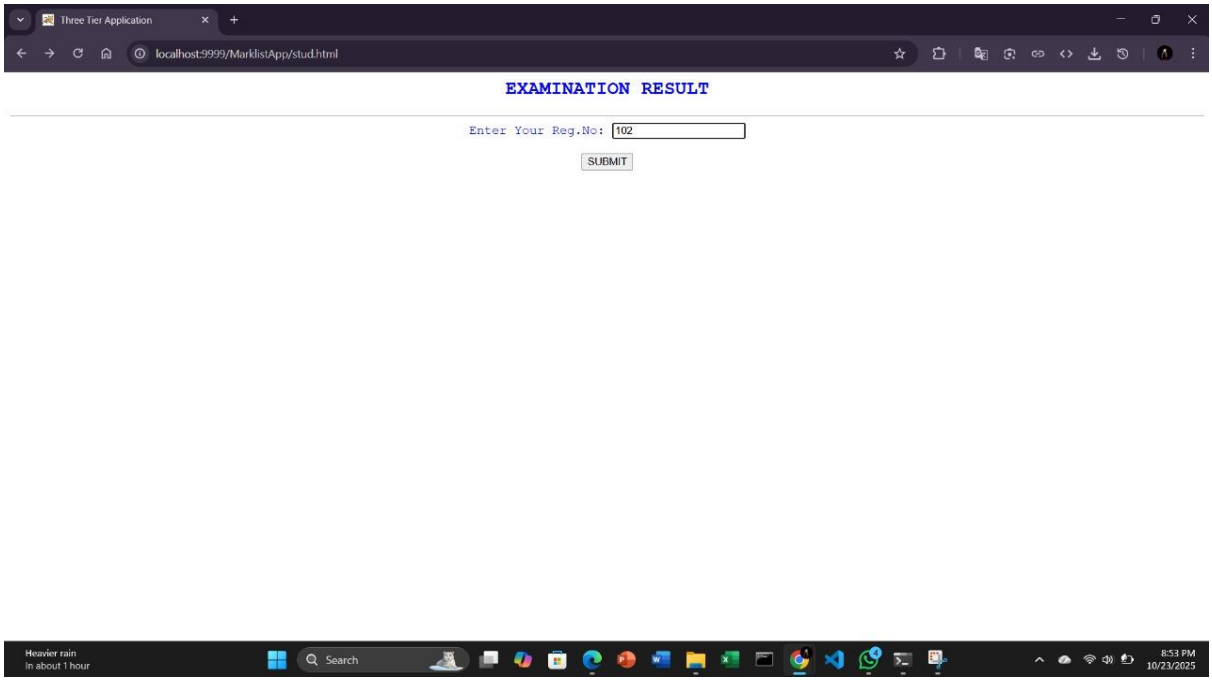
Creating Table:

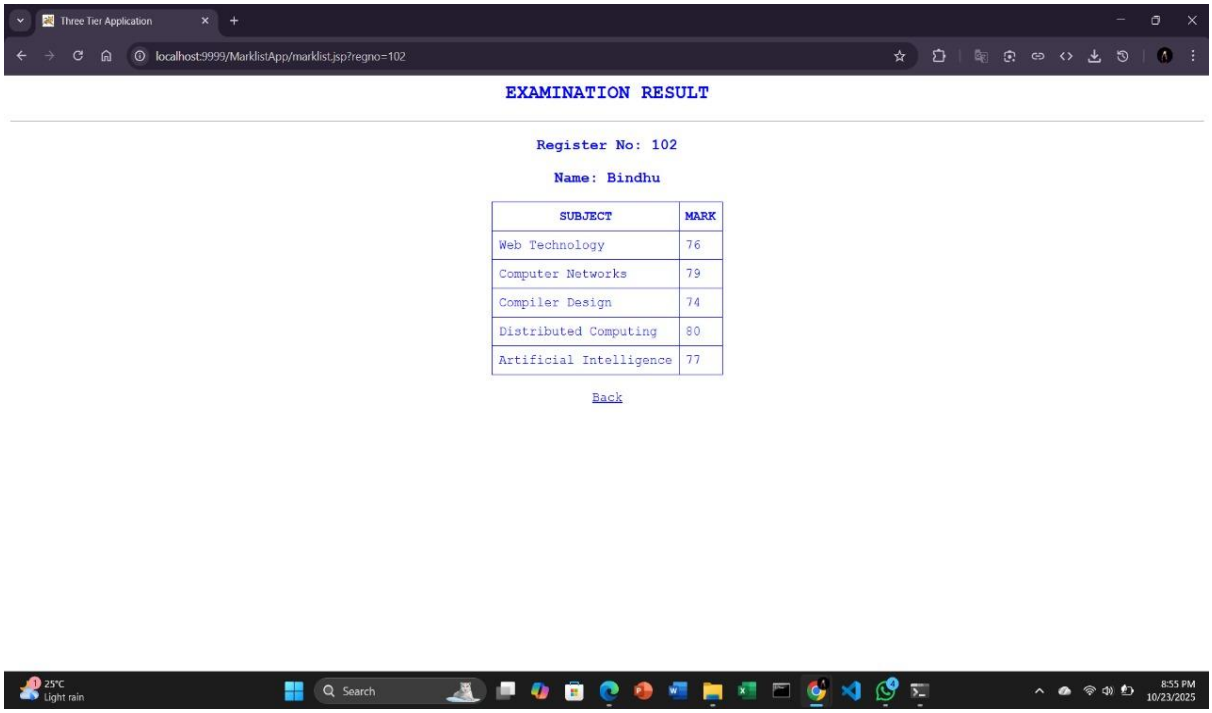
```
SQL> CREATE TABLE markTab (
  2      rno NUMBER PRIMARY KEY,
  3      name VARCHAR2(50),
  4      webtech NUMBER,
  5      compnet NUMBER,
  6      compdesign NUMBER,
  7      distcomp NUMBER,
  8      ai NUMBER
  9  );

Table created.

SQL> desc marktab;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RNO                                       NOT NULL NUMBER
 NAME                                               VARCHAR2(50)
 WEBTECH                                            NUMBER
 COMPNET                                            NUMBER
 COMPDESIGN                                         NUMBER
 DISTCOMP                                           NUMBER
 AI                                                 NUMBER
```

Inserting Records:

```
SQL> INSERT INTO markTab VALUES (101, 'Sidharth', 89, 85, 88, 90, 92);

1 row created.

SQL> INSERT INTO markTab VALUES (102, 'Bindhu',   76, 79, 74, 80, 77);

1 row created.

SQL> INSERT INTO markTab VALUES (103, 'Sujitha',  91, 88, 92, 89, 93);

1 row created.

SQL> INSERT INTO markTab VALUES (104, 'Aswin',    82, 84, 80, 83, 81);

1 row created.

SQL> INSERT INTO markTab VALUES (105, 'Samar',    78, 76, 75, 79, 77);

1 row created.

SQL>
SQL> COMMIT;

Commit complete.
```

OUTPUT:





Result:

Thus the three-tier web application was successfully developed. It retrieved and displayed student marks from the database based on the entered register number using HTML and JSP.

Ex No : 07	Programs using XML – Schema – XSLT/XSL	Page no :
Date :		

Aim:

To write programs using XML and XSLT to display CD catalog data in a formatted HTML table using transformation techniques.

Description:

This experiment demonstrates how to use XML for structured data representation and XSLT for transforming XML into a styled HTML output. The XML file (CDCatalog.xml) contains details of music CDs such as title, artist, country, company, price, and year. The XSL file (CDCatalog.xsl) defines a template to transform the XML data into an HTML table. It uses to loop through each CD entry and to organize the output by artist name. This technique is useful for presenting structured data in a readable format on web pages.

Algorithm:

Step 1: Start

Step 2: Create an XML file with CD catalog data

Step 3: Create an XSL file to define the transformation rules

Step 4: Link the XSL file to the XML using

Step 5: In the XSL file, use to match the root

Step 6: Use to iterate over CD entries

Step 7: Use to extract and display title and artist

Step 8: Apply to sort CDs by artist

Step 9: Save and open the XML file in a browser to view the transformed output

Step 10: Stop

Program:

CDCatalog.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="CDCatalog.xsl"?>  <catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
    <year>1982</year>
  </cd> </catalog>
```

CDCatalog.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Title</th>
            <th>Artist</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <xsl:sort select="artist"/>
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

OUTPUT:

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton

Result:

The XML and XSLT programs were successfully executed. The CD catalog data was displayed in a well-formatted HTML table, sorted by artist name using XSLT transformation.