UNIVERSITY COLLEGE OF ENGINEERING KANCHEEPURAM

(A Constituent college of Anna University Chennai)

KANCHIPURAM - 631 552

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS375 - WEB TECHNOLOGIES

Name: ______________________________

Register no: ______________________________

Year/Semester:___________**Branch:** ____________

UNIVERSITY COLLEGE OF ENGINEERING KANCHEEPURAM

(A Constituent college of Anna University Chennai)

KANCHIPURAM - 631 552

BONAFIDE CERTIFICATE

REGISTER NO

Certified that this is the bonafide record of work done by Mr/Ms... of ...... semester B.E. Computer Science and Engineering Branch / Batch during the academic year 20...... to 20….. in the **CCS375 - WEB TECHNOLOGIES.**

Staff In-Charge **Head of the Department**

Submitted for the University Practical examination held on ....

Internal Examiner **External Examiner**

TABLE OF CONTENTS

Ex.No: 1 **Date:**	**Web Page with Image Map**

Aim:

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

Procedure:

1. Create an HTML file (e.g., index.html).
2. Insert an image into the web page using the <img> tag.
3. Define an image map using the <map> and <area> tags to create clickable hot spots.
4. Specify coordinates for each hot spot and link them to specific actions (e.g., displaying alerts or navigating to URLs).
5. Test the web page in a browser to ensure hot spots display the intended information when clicked.

Program:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Image Map</title>
    <script>
      function showInfo(area) { alert("You clicked on: " + area); }
    </script>
    <style>
      body {
        background-color: antiquewhite;
      }
      .page {
        color: coral;
        font-family: "Courier New", Courier, monospace;
        display: flex;
        flex-direction: column;
        align-items: center;
      }
    </style>
  </head>
  <body>
    <div class="page">
      <div>
        <h1>Image Map</h1>
      </div>
      <div class="image">
        <img src="Image.jpg" style="width: 500px; height: 300px; border-radius: 2%"
          alt="Sample Image" usemap="#imageMap"
        />
      </div>
    </div>
    <map name="imageMap">
      <area shape="rect" coords="50,50,150,150" href="#"
        onclick="showInfo('Top Left Rectangle'); return false;" alt="Top Left"
      />
      <area shape="circle" coords="300,100,50" href="#"
        onclick="showInfo('Right Circle'); return false;" alt="Right Circle"
      />
      <area shape="poly" coords="100,200,200,200,150,300" href="#"
        onclick="showInfo('Bottom Triangle'); return false;" alt="Bottom Triangle"
      />
    </map>
  </body>
</html>
```

Output:

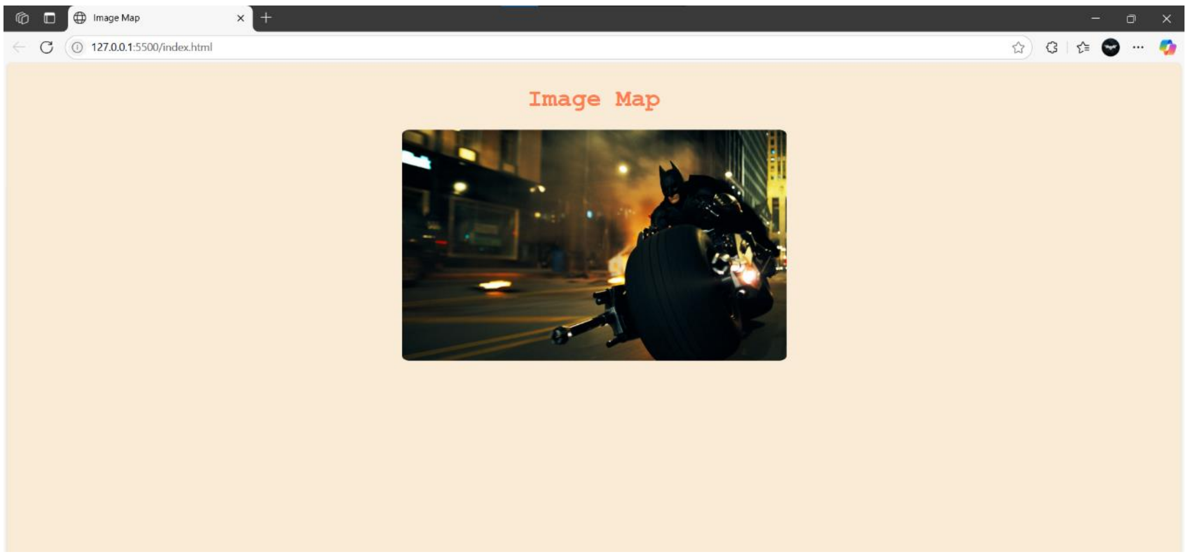
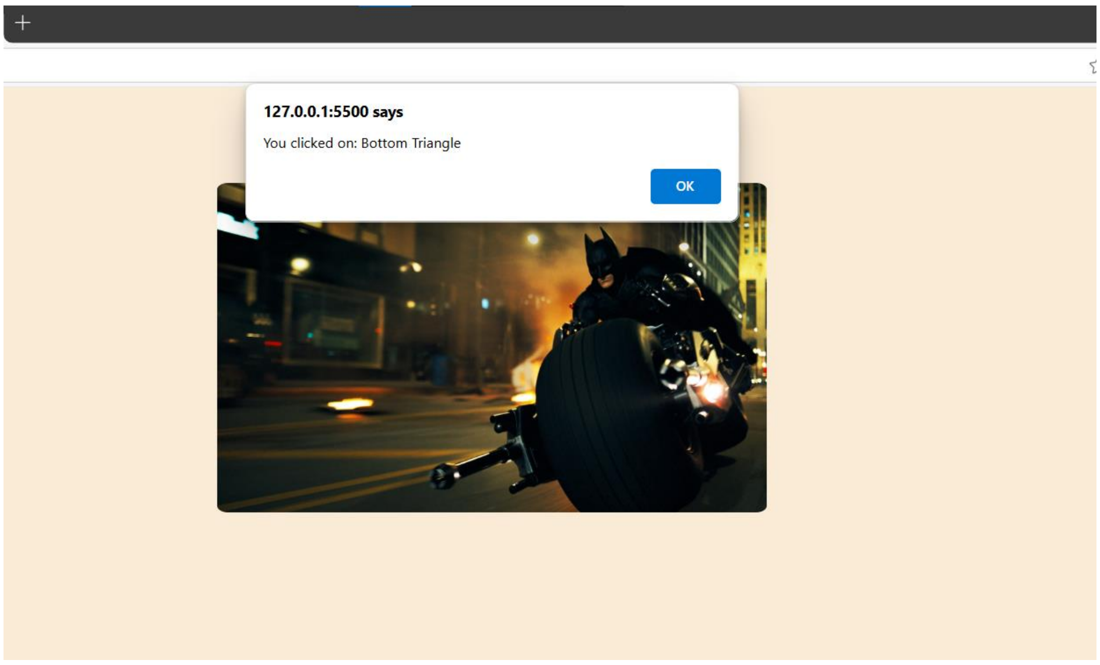
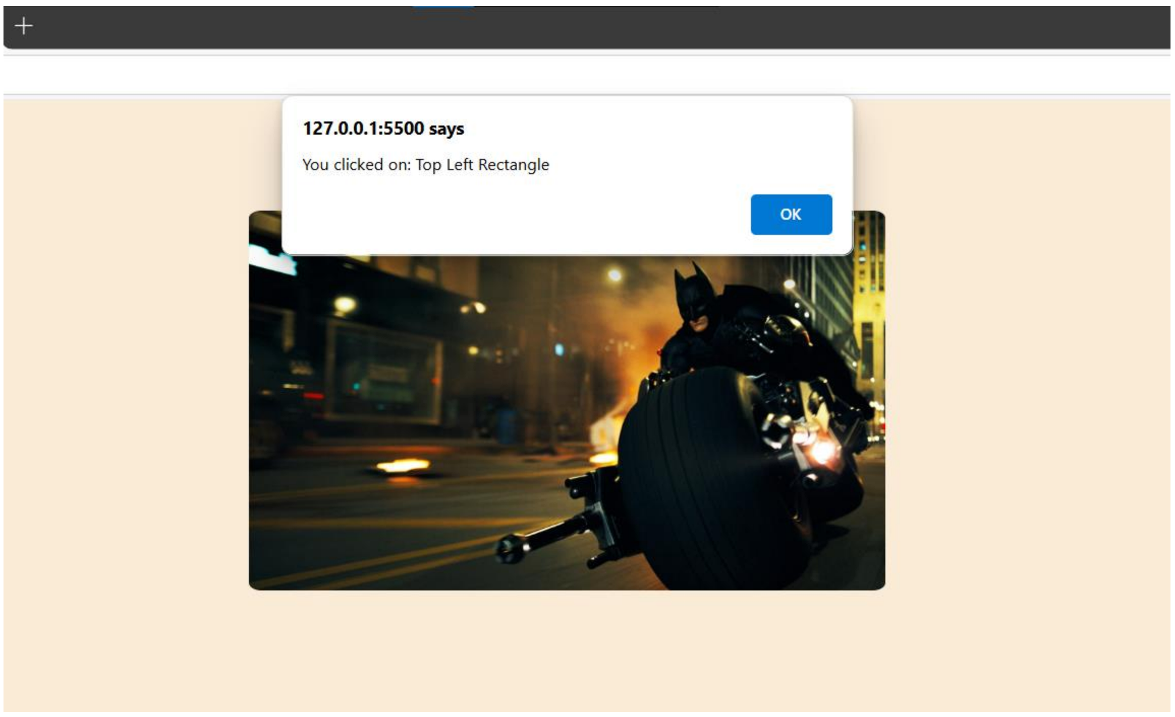


Image Map



127.0.0.1:5500 says

You clicked on: Bottom Triangle



127.0.0.1:5500 says

You clicked on: Top Left Rectangle

Result:

Thus, to create a web page using HTML that embeds an image map, defines hot spots, and displays related information when the hot spots are clicked is successfully executed.

Ex.No: 2 **Date:**	**Web Page with Cascading Style Sheets**

Aim:

To create a web page that demonstrates inline, internal, and external Cascading Style Sheets (CSS).

Procedure:

1. Create an HTML file (e.g., styles.html) and a separate CSS file (e.g., styles.css).
2. Apply inline CSS to an element using the style attribute.
3. Include internal CSS within a <style> tag in the HTML <head>.
4. Link the external CSS file using the <link> tag.
5. Design the web page to showcase different styling for text, backgrounds, and layouts.
6. Test the web page in a browser to verify all CSS types are applied correctly.

Program:

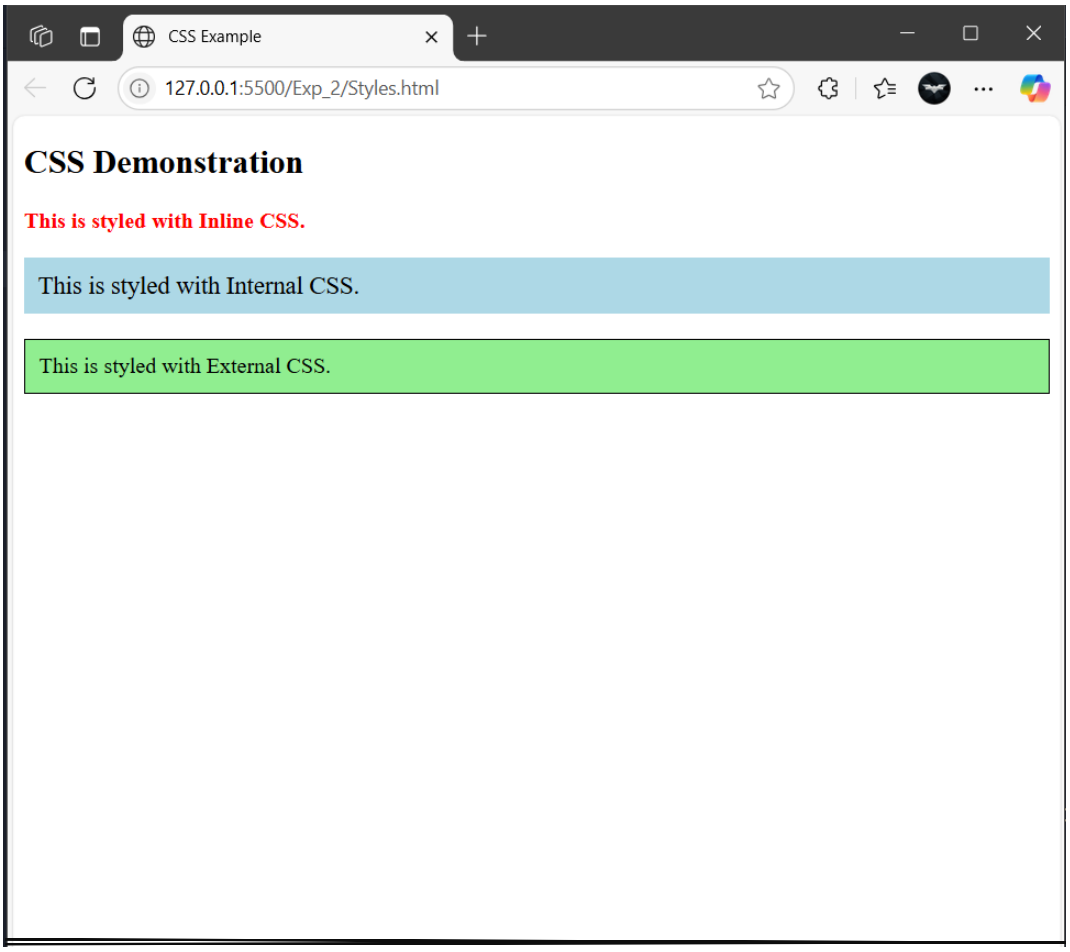
Styles.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>CSS Example</title>
    <link rel="stylesheet" href="styles.css" />
    <style>
      /* Internal CSS */
      .internal {
        background-color: lightblue;
        padding: 10px;
        font-size: 18px;
      }
    </style>
  </head>
  <body>
    <h2>CSS Demonstration</h2>
    <!-- Inline CSS -->
    <p style="color: red; font-weight: bold">This is styled with Inline CSS.</p>
    <!-- Internal CSS -->
    <p class="internal">This is styled with Internal CSS.</p>
    <!-- External CSS -->
    <p class="external">This is styled with External CSS.</p>
  </body>
</html>
```

Styles.css

```
/* External CSS */
.external {
  background-color: lightgreen;
  padding: 10px;
  font-size: 16px;
  border: 1px solid black;
}
```

Output:


CSS Example

127.0.0.1:5500/Exp_2/Styles.html

CSS Demonstration

This is styled with Inline CSS.

This is styled with Internal CSS.

This is styled with External CSS.

Result:

Thus, to create a web page that demonstrates inline, internal, and external Cascading Style Sheets (CSS) is successfully executed.

Ex.No: 3 **Date:**	**Client-Side Form Validation using DHTML**

Aim:

To create a web page with client-side form validation using Dynamic HTML (DHTML) and JavaScript.

Procedure:

1. Create an HTML file (e.g., form.html) with a form containing input fields (e.g., name, email, password).
2. Add a submit button and a <div> for displaying validation messages.
3. Write JavaScript to validate form inputs (e.g., check for empty fields, valid email format).
4. Use DHTML to dynamically update the page with validation messages without reloading.
5. Test the form in a browser to ensure validations work and messages display correctly.

Program:

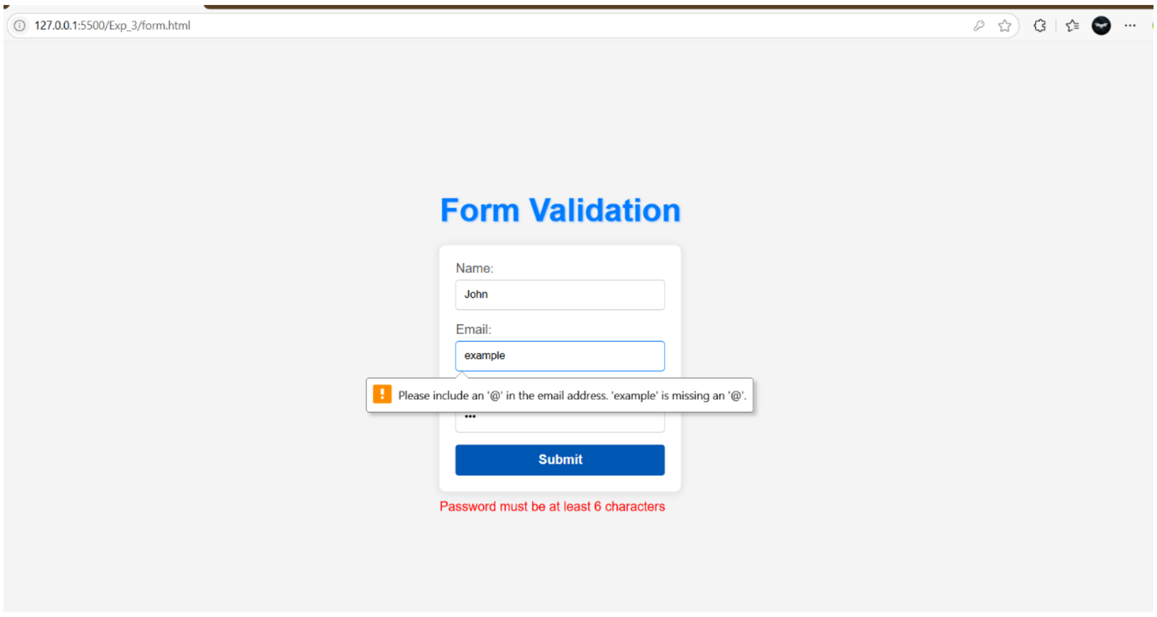
Form.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <title>Form Validation</title>
   <style>
     body {
       font-family: Arial, sans-serif;
       background-color: #f4f4f4;
       display: flex;
       justify-content: center;
```

```
  align-items: center;
  height: 100vh;
  margin: 0;
}
h2.title {
  color: #007bff;
  font-size: 2.5rem;
  text-align: center;
  margin-bottom: 20px;
  text-shadow: 1px 1px 2px rgba(0, 123, 255, 0.5);
  font-weight: 700;
}
form {
  background: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  width: 300px;
  box-sizing: border-box;
}
label {
  display: block;
  margin-bottom: 5px;
  color: #555;
}
input[type="text"],
input[type="email"],
input[type="password"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
  transition: border-color 0.3s;
  box-sizing: border-box;
```

```
      }
      input[type="text"]:focus,
      input[type="email"]:focus,
      input[type="password"]:focus {
        border-color: #007bff;
        outline: none;
      }
      input[type="submit"] {
        background-color: #007bff;
        color: white;
        border: none;
        padding: 10px;
        border-radius: 4px;
        cursor: pointer;
        width: 100%;
        transition: background-color 0.3s;
        font-size: 1rem;
        font-weight: 600;
      }
      input[type="submit"]:hover {
        background-color: #0056b3;
      }
      .error {
        color: red;
        margin-top: 10px;
      }
      .success {
        color: green;
        margin-top: 10px;
        font-weight: 600;
      }
    </style>
  </head>
  <body>
    <div>
  <h2 class="title">Form Validation</h2>
  <form id="myForm" onsubmit="return validateForm(event)">
    <label for="name">Name:</label>
    <input type="text" id="name" placeholder="Enter your name" />
    <label for="email">Email:</label>
    <input type="email" id="email" placeholder="Enter your email" />
    <label for="password">Password:</label>
    <input
      type="password"
      id="password"
      placeholder="Enter your password"
    />
    <input type="submit" value="Submit" />
  </form>
  <div id="errorMsg"></div>
</div>
<script>
  function validateForm(event) {
    event.preventDefault();
    let name = document.getElementById("name").value;
    let email = document.getElementById("email").value;
    let password = document.getElementById("password").value;
    let errorMsg = document.getElementById("errorMsg");
    errorMsg.innerHTML = "";
    if (name === "") {
      errorMsg.innerHTML += '<p class="error">Name is required</p>';
    }
    if (!email.includes("@")) {
      errorMsg.innerHTML += '<p class="error">Invalid email format</p>';
    }
    if (password.length < 6) {
      errorMsg.innerHTML +=
        '<p class="error">Password must be at least 6 characters</p>';
    }
    if (errorMsg.innerHTML === "") {
          errorMsg.innerHTML =
            '<p class="success">Form submitted successfully!</p>';
        }
        return false;
      }
    </script>
  </body>
</html>
```

Output:



127.0.0.1:5500/Exp_3/form.html

Form Validation

Name: John

Email: example

Please include an '@' in the email address. 'example' is missing an '@'.

Submit

Password must be at least 6 characters

Result:

Thus, to create a web page with client-side form validation using Dynamic HTML (DHTML) and JavaScript.

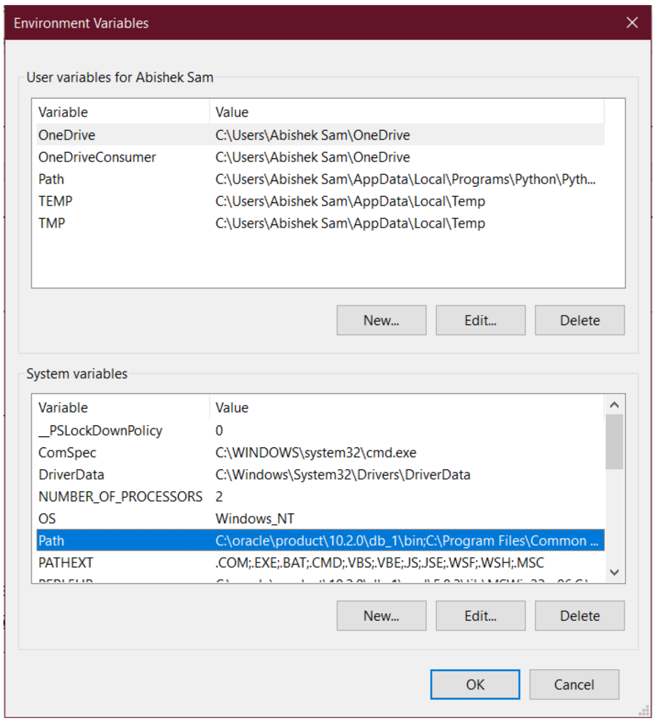
Ex.No: 4 **Date:**	**Installation of Apache Tomcat Web Server**

Aim:

To install and configure the Apache Tomcat web server on a local system**.**

Procedure:

Step-1: Download and install JDK (e.g., OpenJDK or Oracle JDK) from *https://openjdk.java.net/* or *https://www.oracle.com/java/*. Set the JAVA_HOME environment variable:



Environment Variables

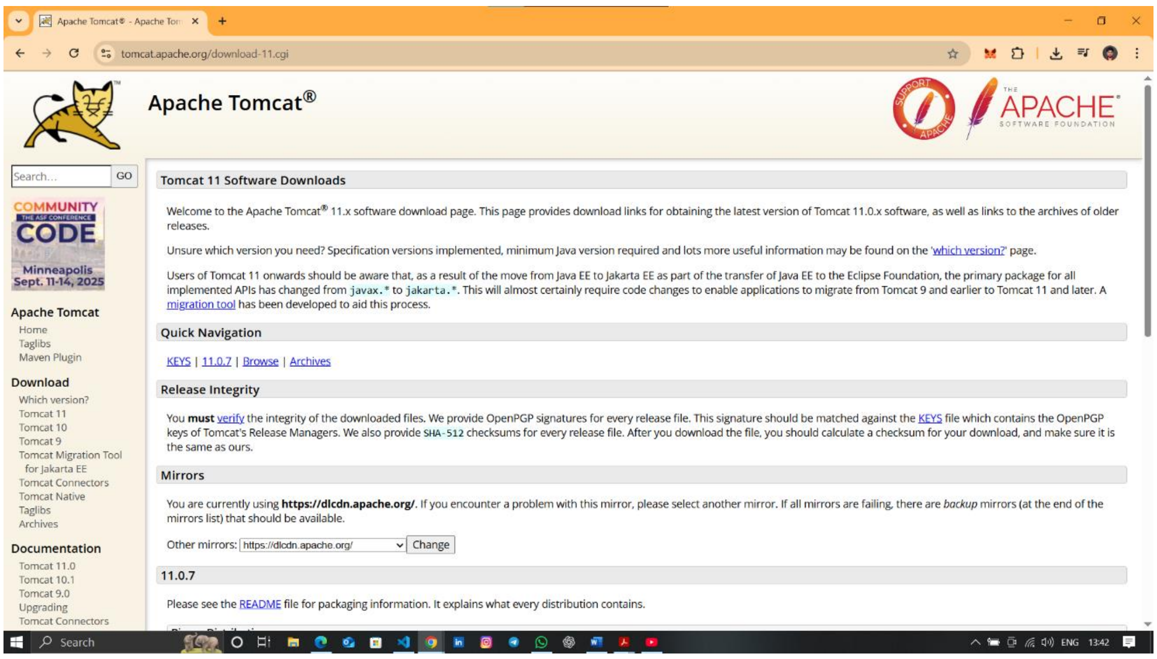
User variables for Abishek Sam

Variable	Value
OneDrive	C:\Users\Abishek Sam\OneDrive
OneDriveConsumer	C:\Users\Abishek Sam\OneDrive
Path	C:\Users\Abishek Sam\AppData\Local\Programs\Python\Pyth...
TEMP	C:\Users\Abishek Sam\AppData\Local\Temp
TMP	C:\Users\Abishek Sam\AppData\Local\Temp

System variables

Variable	Value
__PSLockDownPolicy	0
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\oracle\product\10.2.0\db_1\bin;C:\Program Files\Common ...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

Step-2: Download Tomcat (e.g., apache-tomcat-10.1.x.tar.gz or .zip) from https://tomcat.apache.org/download-11.cgi . Extract to a directory (e.g., C:\Tomcat or /opt/tomcat)



Apache Tomcat®

Tomcat 11 Software Downloads

Welcome to the Apache Tomcat® 11.x software download page. This page provides download links for obtaining the latest version of Tomcat 11.0.x software, as well as links to the archives of older releases.

Unsure which version you need? Specification versions implemented, minimum Java version required and lots more useful information may be found on the 'which version?' page.

Users of Tomcat 11 onwards should be aware that, as a result of the move from Java EE to Jakarta EE as part of the transfer of Java EE to the Eclipse Foundation, the primary package for all implemented APIs has changed from `javax.*` to `jakarta.*`. This will almost certainly require code changes to enable applications to migrate from Tomcat 9 and earlier to Tomcat 11 and later. A migration tool has been developed to aid this process.

Quick Navigation

KEYS | 11.0.7 | Browse | Archives

Release Integrity

You **must** verify the integrity of the downloaded files. We provide OpenPGP signatures for every release file. This signature should be matched against the KEYS file which contains the OpenPGP keys of Tomcat's Release Managers. We also provide `SHA-512` checksums for every release file. After you download the file, you should calculate a checksum for your download, and make sure it is the same as ours.

Mirrors

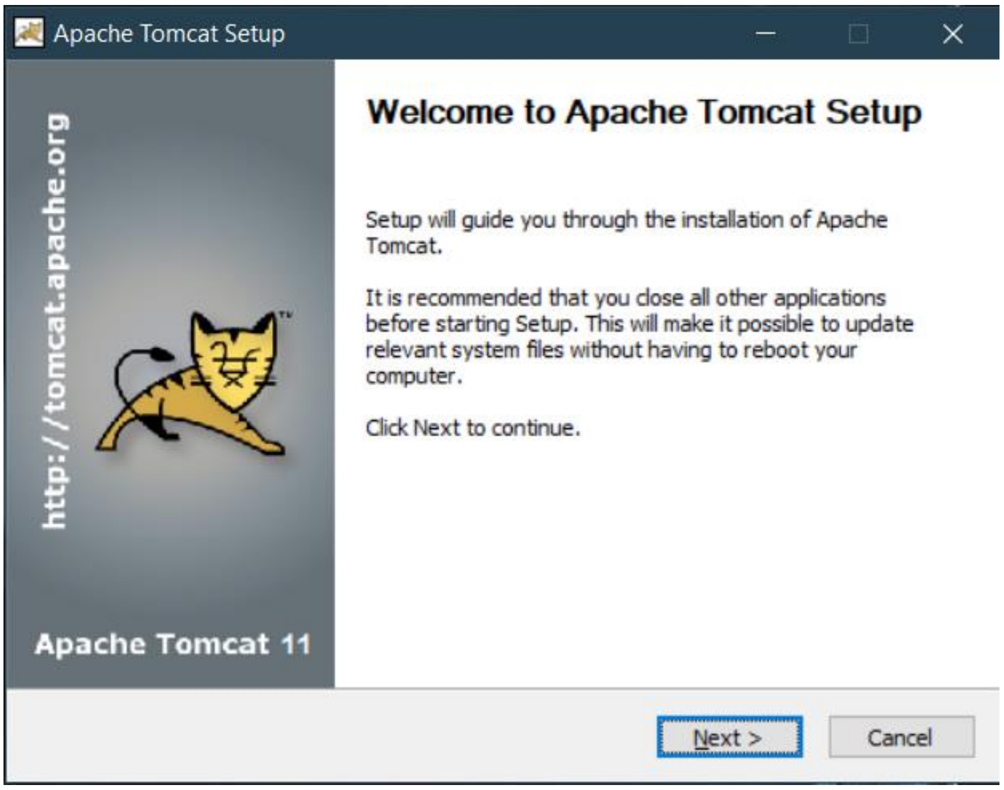
You are currently using **https://dlcdn.apache.org/**. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors: https://dlcdn.apache.org/ Change

11.0.7

Please see the README file for packaging information. It explains what every distribution contains.

Step-3: Run the *Apache-tomacat.exe* file and accept the user agreement in the installation wizard.



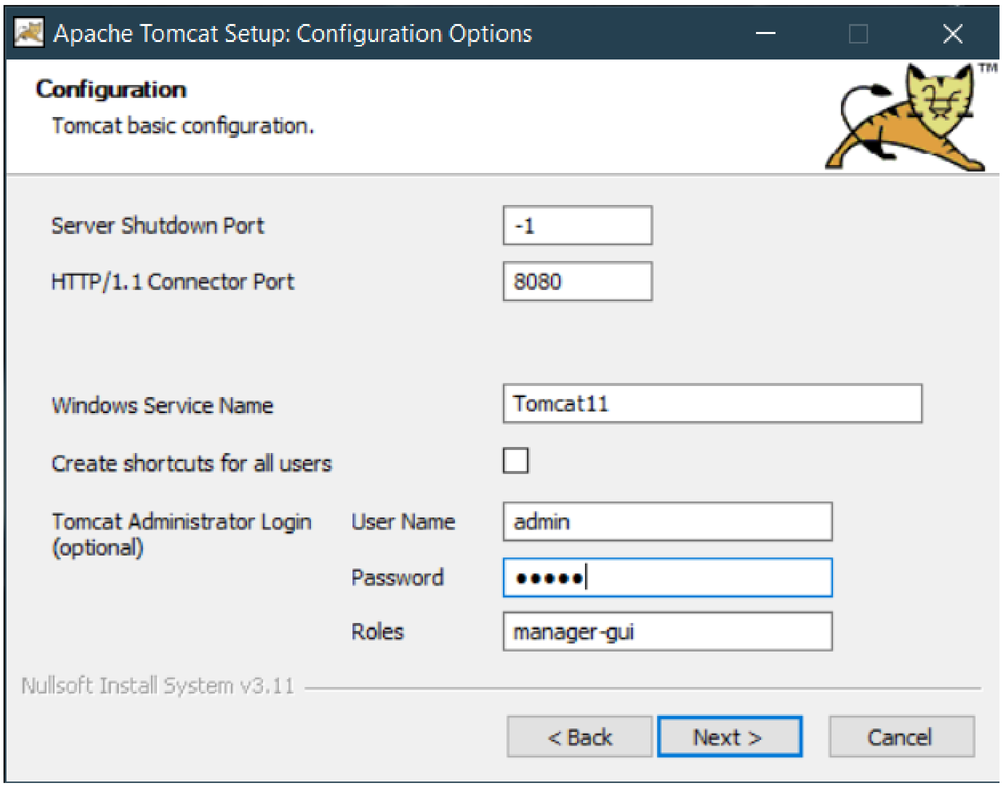
Welcome to Apache Tomcat Setup

Setup will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Step-4: provide username and password.

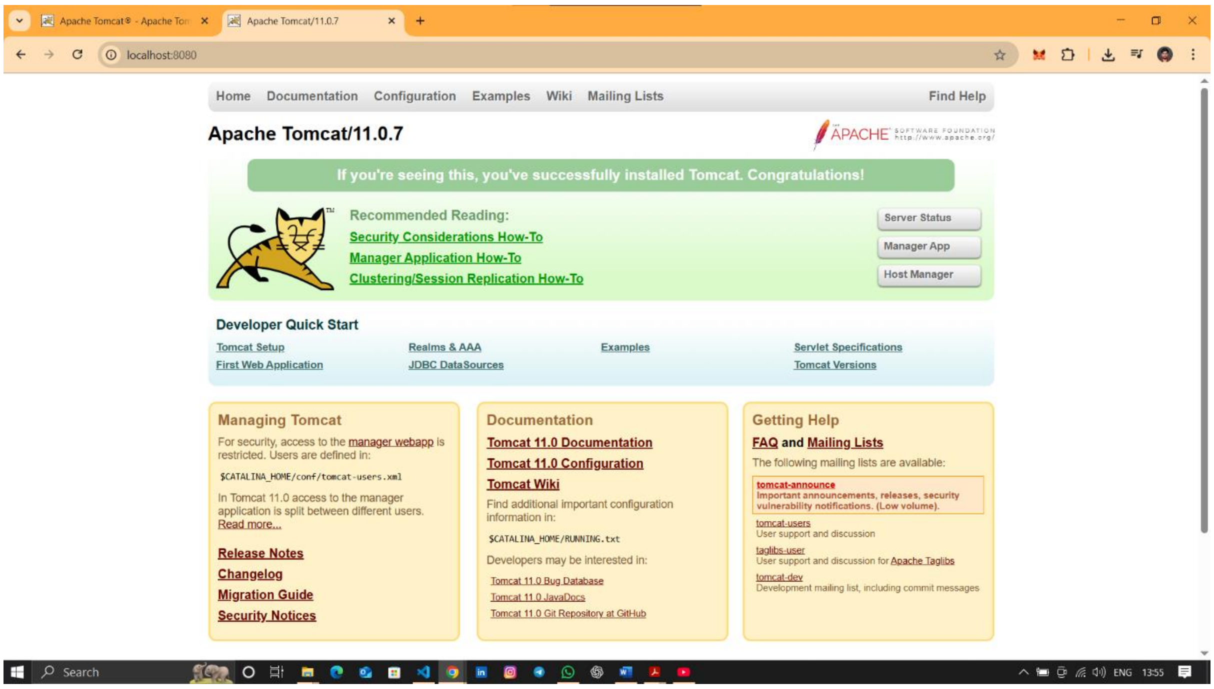

Apache Tomcat Setup: Configuration Options
Configuration
Tomcat basic configuration.

Server Shutdown Port: -1
HTTP/1.1 Connector Port: 8080
Windows Service Name: Tomcat11
Create shortcuts for all users: ☐
Tomcat Administrator Login (optional) User Name: admin
Password: •••••
Roles: manager-gui

Nullsoft Install System v3.11
< Back
Next >
Cancel

Step-5: To verify installation, open a browser and go to *http://localhost:8080*

Confirm the Tomcat welcome page appears.



Apache Tomcat/11.0.7

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

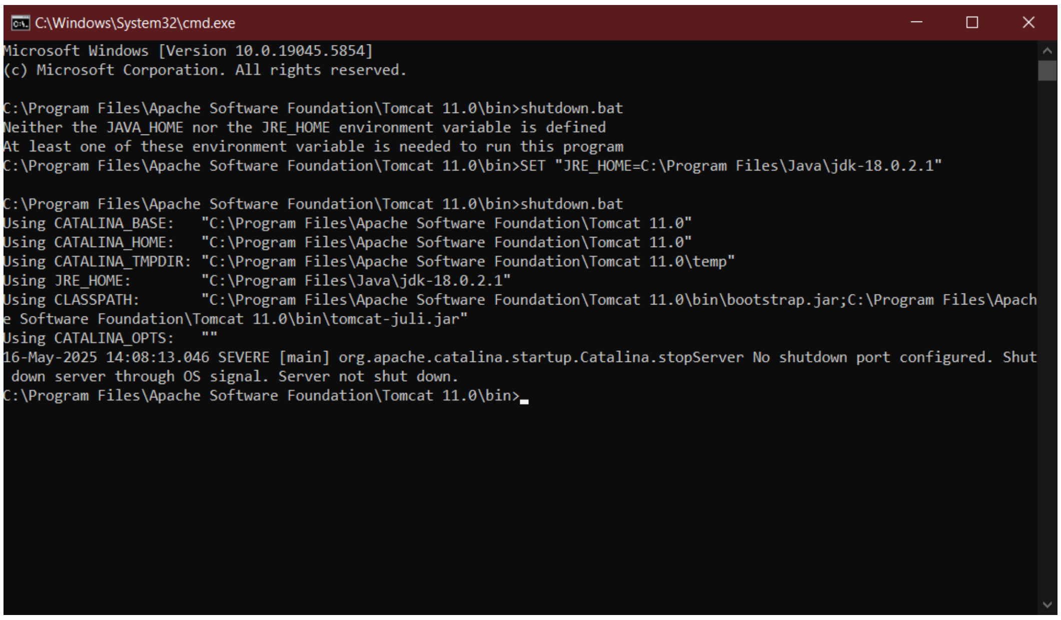
Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
```
$CATALINA_HOME/conf/tomcat-users.xml
```
In Tomcat 11.0 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation
Tomcat 11.0 Documentation
Tomcat 11.0 Configuration
Tomcat Wiki
Find additional important configuration information in:
```
$CATALINA_HOME/RUNNING.txt
```
Developers may be interested in:
Tomcat 11.0 Bug Database
Tomcat 11.0 JavaDocs
Tomcat 11.0 Git Repository at GitHub

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

Step-6: To stop tomcat, open the terminal go to the **bin** directory of tomcat and run **shutdown.bat**



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>shutdown.bat
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program
C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>SET "JRE_HOME=C:\Program Files\Java\jdk-18.0.2.1"

C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>shutdown.bat
Using CATALINA_BASE:   "C:\Program Files\Apache Software Foundation\Tomcat 11.0"
Using CATALINA_HOME:   "C:\Program Files\Apache Software Foundation\Tomcat 11.0"
Using CATALINA_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 11.0\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-18.0.2.1"
Using CLASSPATH:       "C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin\bootstrap.jar;C:\Program Files\Apach
e Software Foundation\Tomcat 11.0\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
16-May-2025 14:08:13.046 SEVERE [main] org.apache.catalina.startup.Catalina.stopServer No shutdown port configured. Shut
 down server through OS signal. Server not shut down.
C:\Program Files\Apache Software Foundation\Tomcat 11.0\bin>
```

Result:

Thus, to install and configure the Apache Tomcat web server on a local system is successfully executed.

Ex.No: 5 Date:	**Java Servlets for Form Invocation and Session Tracking**

Aim:

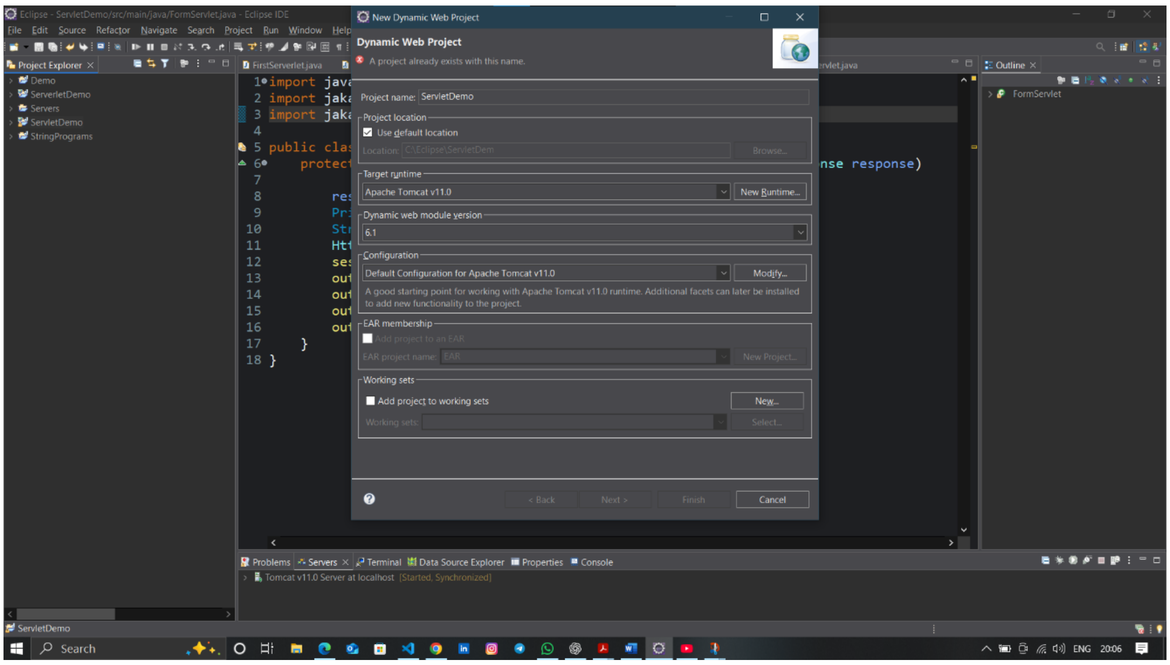
To write Java servlet programs to invoke servlets from HTML forms and implement session tracking.

Procedure:

Step-1: Install JDK and Apache Tomcat and Install Eclipse IDE with the Tomcat plugin (e.g., Eclipse IDE for Enterprise Java Developers).

Step-2: In Eclipse, create a new **Dynamic Web Project** (e.g., ServletDemo):

- File → New → Dynamic Web Project.
- Set Tomcat as the target runtime.
- Ensure the **Dynamic Web Module** version is compatible (e.g., 4.0).



New Dynamic Web Project

Dynamic Web Project

A project already exists with this name.

Project name: ServletDemo

Project location

[✓] Use default location

Location: C:\Eclipse\ServletDem

Target runtime: Apache Tomcat v11.0

Dynamic web module version: 6.1

Configuration: Default Configuration for Apache Tomcat v11.0

A good starting point for working with Apache Tomcat v11.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

[ ] Add project to an EAR

EAR project name: EAR

Working sets

[ ] Add project to working sets

Tomcat v11.0 Server at localhost [Started, Synchronized]

Step-3: Place index.html in the web root directory:

- If *src/main/webapp* exists or was created, put **index.html** in *src/main/webapp/*.
- If using a custom web folder (e.g., WebContent), create it manually and place index.html there.

- Create *src/main/webapp/WEB-INF/* and place **web.xml** in it.
- Place **FormServlet.java** and **SessionServlet.java** in *src/main/java/* .

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Form Input</title>

</head>

<body>

  <h2>Enter Details</h2>

  <form action="formServlet" method="post">

    <label>Name:</label>

    <input type="text" name="username"><br>

    <input type="submit" value="Submit">

  </form>

</body>

</html>
```

web.xml

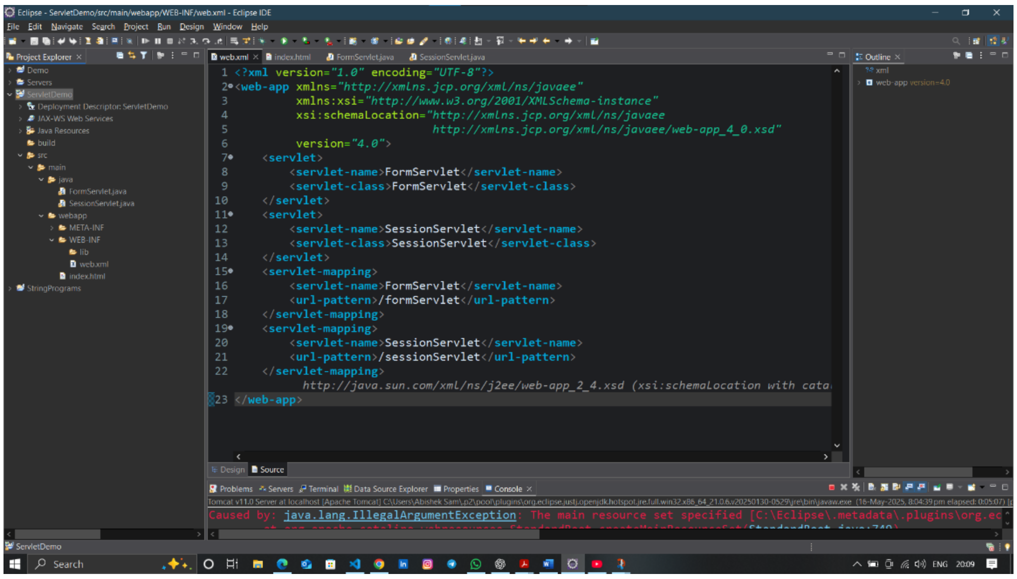
```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

               http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"

     version="4.0">

  <servlet>

    <servlet-name>FormServlet</servlet-name>

    <servlet-class>FormServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>SessionServlet</servlet-name>
        <servlet-class>SessionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FormServlet</servlet-name>
        <url-pattern>/formServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SessionServlet</servlet-name>
        <url-pattern>/sessionServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

FormServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FormServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String username = request.getParameter("username");
        HttpSession session = request.getSession();
        session.setAttribute("username", username);
        out.println("<html><body>");
        out.println("<h2>Form Data Stored in Session</h2>");
        out.println("<a href='sessionServlet'>View Session Data</a>");
        out.println("</body></html>");
    }
}
```

SessionServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SessionServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(false);
        out.println("<html><body>");
        if (session != null && session.getAttribute("username") != null) {
            String username = (String) session.getAttribute("username");
            out.println("<h2>Session Data</h2>");
            out.println("Username: " + username);
        } else {
            out.println("<h2>No Session Data Found</h2>");
        }
        out.println("</body></html>");
    }
}
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>FormServlet</servlet-name>
        <servlet-class>FormServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>SessionServlet</servlet-name>
        <servlet-class>SessionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FormServlet</servlet-name>
        <url-pattern>/formServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SessionServlet</servlet-name>
        <url-pattern>/sessionServlet</url-pattern>
    </servlet-mapping>
         http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd (xsi:schemaLocation with cata
</web-app>
```

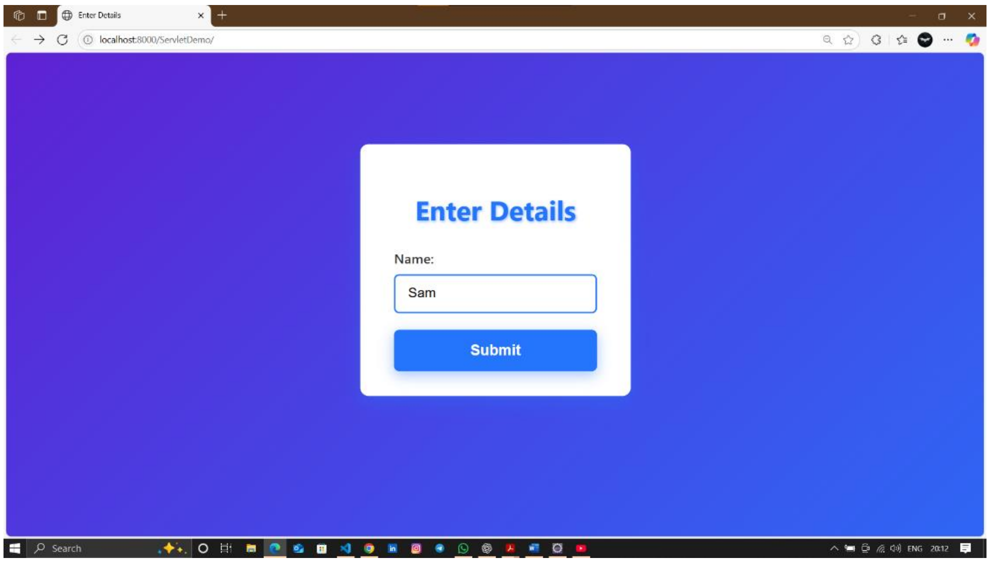
```
Tomcat v11.0 Server at localhost [Apache Tomcat] C:\Users\Abishek Sam\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (16-May-2025, 8:04:39 pm elapsed: 0:05:07)
Caused by: java.lang.IllegalArgumentException: The main resource set specified [C:\Eclipse\.metadata\.plugins\org.ec
```

Step-4: Compile and Deploy:

- Right-click the project → Run As → Run on Server → Select Tomcat.
- Eclipse will compile the Java files, package the web resources, and deploy to Tomcat.

Step-5: Test:

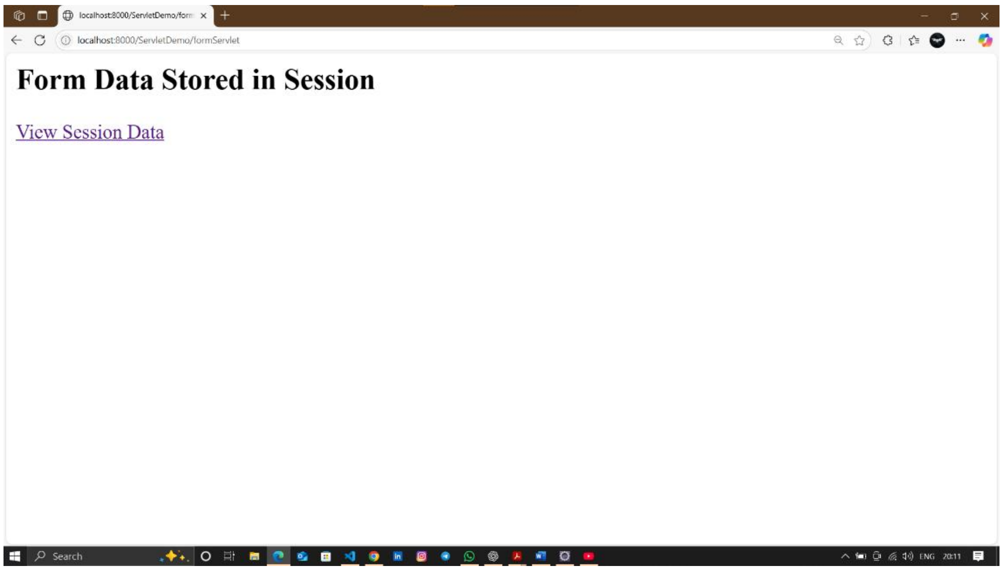
- Open a browser and go to *http://localhost:8080/ServletDemo/*.
- Enter a username (e.g., "Sam") in the form and click Submit.
- Click the "View Session Data" link to verify the stored username is displayed.



localhost:8080/ServletDemo/

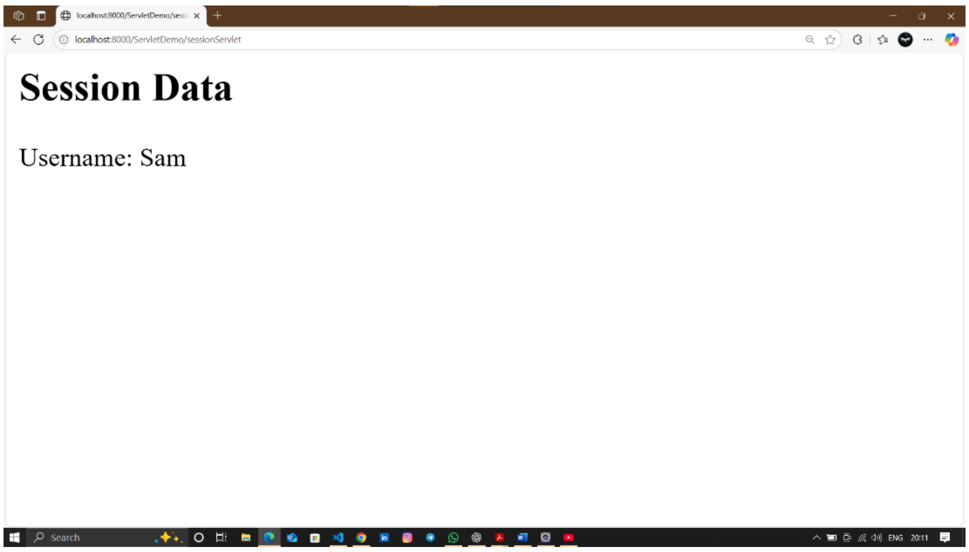
Enter Details

Name: Sam


localhost:8000/ServletDemo/formServlet

Form Data Stored in Session

View Session Data


localhost:8000/ServletDemo/sessionServlet

Session Data

Username: Sam

Result:

Thus, to write Java servlet programs to invoke servlets from HTML forms and implement session tracking is successfully executed.

Ex.No: 6 **Date:**	**Java Servlets for Form Invocation and Session Tracking**

Aim:

To develop Java programs using JSP and databases for conducting an online examination and displaying a student mark list.

Prerequisites:

- Java Development Kit (JDK) installed (version 8 or above).
- Apache Tomcat server installed.
- MySQL server installed.
- Eclipse IDE (with Dynamic Web Project support).
- MySQL Connector/J (JDBC driver for MySQL).

Procedure:

Step 1: Set Up MySQL Database

- Open MySQL Client MySQL client to connect to MySQL server.
- Create Database and Tables

```
CREATE DATABASE StudentDB;

USE StudentDB;



CREATE TABLE Students (

   id INT AUTO_INCREMENT PRIMARY KEY,

   name VARCHAR(100),

   marks INT

);

INSERT INTO Students (name, marks) VALUES

('Alice', 85),

('Bob', 90),

('Charlie', 78);
```

Step 2: Create Dynamic Web Project in Eclipse

1. Open Eclipse IDE.
2. Create a New Dynamic Web Project:
    - Go to *File > New > Dynamic Web Project*.
    - Name the project **OnlineExamApp**.
    - Select Apache Tomcat as the target runtime.
    - Click Finish.

Step 3: Add MySQL Connector/J

1. Download MySQL Connector/J:
    - Download the MySQL Connector/J (JDBC driver) from the official MySQL website.
2. Add Connector to Project:
    - Copy the **mysql-connector-java-9.3.0.jar** file to the *webapp/WEB-INF/lib* folder of your project.
    - Right-click the jar file in *Eclipse > Build Path > Add to Build Path.*

Step 4: Add Java Classes

1. Create Package:
    - Inside the src folder, right-click and select New > Package.
    - Name the package com.lab.
2. Add Java Classes:
    - Right-click the com.lab package and select New > Class.
    - Create the following classes:
        - **DBConnection.java**
        - **SubmitExamServlet.java**

<u>DBConnection.java</u>

```
package com.lab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnection {

    public static Connection getConnection() throws SQLException {

        String url = "jdbc:mysql://localhost:3306/StudentDB?useSSL=false";

        String user = "root";

        String password = "Ragnarok@2468";  // Change this to your MySQL password

        try {
            Class.forName("com.mysql.cj.jdbc.Driver"); // Register MySQL JDBC driver
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return DriverManager.getConnection(url, user, password);
    }
}
```

SubmitExamServlet.java

```
package com.lab;
import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
@WebServlet("/SubmitExamServlet")
public class SubmitExamServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    // Process POST request and respond with JSON status as plain string
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Simulate exam submission process - always success here
        boolean submissionSuccess = true;
        // Manually build JSON string
        String jsonResponse = "{\"success\": " + submissionSuccess + "}";
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        response.getWriter().write(jsonResponse);
        response.getWriter().flush();

    }

}
```

Step 5: Add JSP Files

1. Add JSP Files:
    - Inside the WebContent folder, create the following JSP files:
        - **exam.jsp**
        - **marks.jsp**

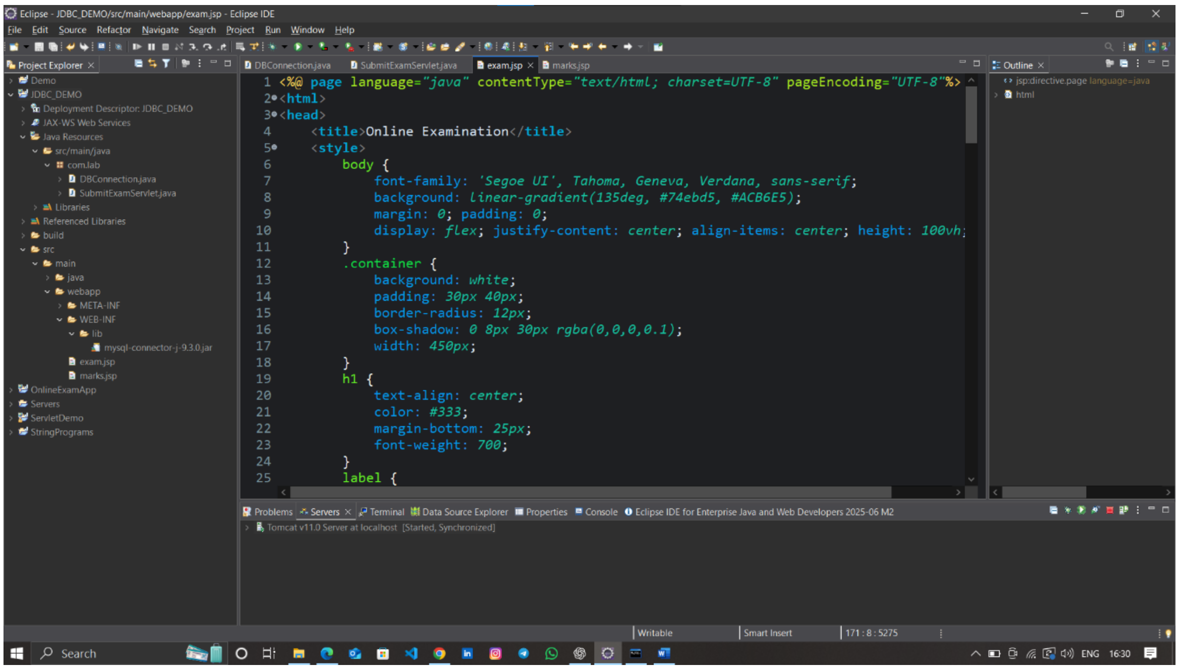
exam.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<html>

<head>

    <title>Online Examination</title>

    <style>

        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            background: linear-gradient(135deg, #74ebd5, #ACB6E5);

            margin: 0; padding: 0;

            display: flex; justify-content: center; align-items: center; height: 100vh;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Online Examination</h1>

        <form id="examForm">

            <label for="q1">Question 1: What is Java?</label>

            <input type="text" id="q1" name="q1" required>

            <label for="q2">Question 2: Explain JSP.</label>
            <input type="text" id="q2" name="q2" required>
            <button type="submit">Submit Exam</button>
        </form>
    </div>
    <div id="submissionModal" class="modal">
        <div class="modal-content">
            <div id="modalMessage"></div>
            <button class="close-btn" onclick="closeModal()">Close</button>
        </div>
    </div>
    <script>
        const form = document.getElementById('examForm');
        const modal = document.getElementById('submissionModal');
        const modalMessage = document.getElementById('modalMessage');
        form.addEventListener('submit', function(event) {
            event.preventDefault();
            const formData = new FormData(form);
            fetch('SubmitExamServlet', {
                method: 'POST',
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                if(data.success) {
                    modalMessage.textContent = '✅ Your exam has been submitted successfully!';
                    openModal();
                } else {
                    modalMessage.textContent = '⚠️ Submission failed. Please try again.';
                    openModal();
                }
            })
            .catch(() => {
                modalMessage.textContent = '⚠ An error occurred submitting the exam.';
                openModal();
            });
        });
        function openModal() {
            modal.style.display = 'block';
        }
        function closeModal() {
            modal.style.display = 'none';
            form.reset();
            // Optionally redirect after closing modal:
            // window.location.href = 'marks.jsp';
        }
        // Close modal if clicked outside content
        window.onclick = function(event) {
            if (event.target == modal) {
                closeModal();
            }
        }
    </script>
  </body>
  </html>
```

marks.jsp

```
<%@ page import="java.sql.*"%>

<%@ page import="com.lab.DBConnection"%>

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<html>

<head>

   <title>Student Marks List</title>

   <style>

     body { font-family: Arial, sans-serif; background: #fff9f9; margin: 40px; }

     h1 { color: #333; }

     table { border-collapse: collapse; width: 60%; margin-top: 20px; }

     th, td { border: 1px solid #ddd; padding: 8px; text-align: center; }

     th { background-color: #007bff; color: white; }

   </style>

</head>

<body>

   <h1>Student Marks List</h1>

   <table>

       <tr>

           <th>Student ID</th><th>Name</th><th>Marks</th>

       </tr>

       <%

           Connection conn = null;

           Statement stmt = null;

           ResultSet rs = null;

           try {

               conn = DBConnection.getConnection();

               stmt = conn.createStatement();
```

```
            rs = stmt.executeQuery("SELECT * FROM Students");
            while (rs.next()) {
        %>
            <tr>
                <td><%= rs.getInt("id") %></td>
                <td><%= rs.getString("name") %></td>
                <td><%= rs.getInt("marks") %></td>
            </tr>
        <%
            }
          } catch(Exception e) {
            out.println("Database error: " + e.getMessage());
          } finally {
            try { if(rs != null) rs.close(); } catch(Exception ignore) {}
            try { if(stmt != null) stmt.close(); } catch(Exception ignore) {}
            try { if(conn != null) conn.close(); } catch(Exception ignore) {}
          }
        %>
    </table>
</body>
</html>
```

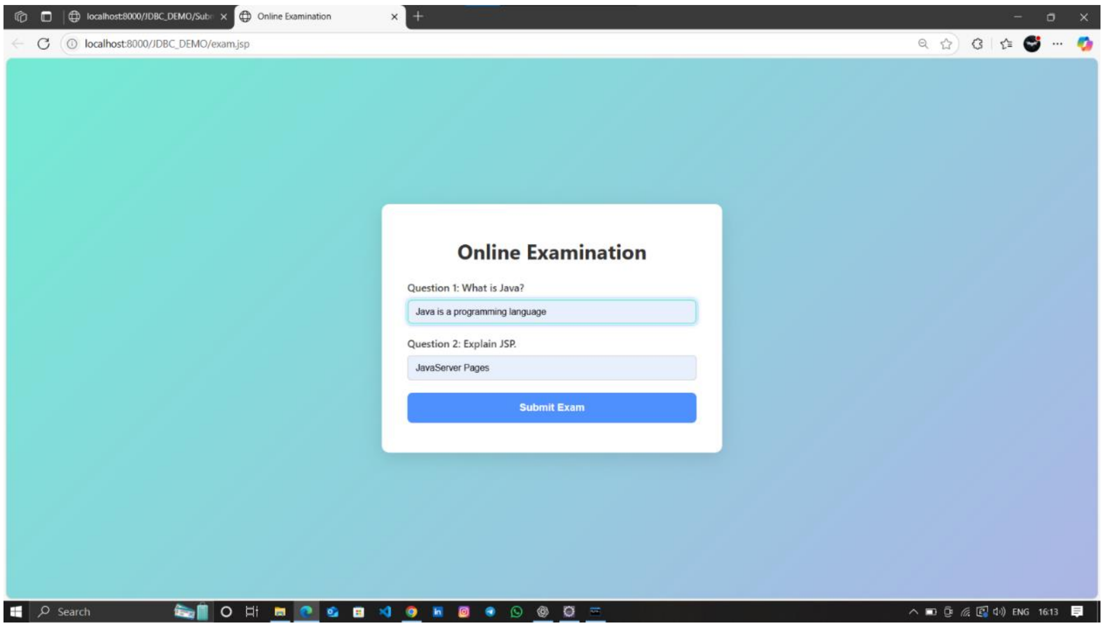


```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
    <title>Online Examination</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #74ebd5, #ACB6E5);
            margin: 0; padding: 0;
            display: flex; justify-content: center; align-items: center; height: 100vh;
        }
        .container {
            background: white;
            padding: 30px 40px;
            border-radius: 12px;
            box-shadow: 0 8px 30px rgba(0,0,0,0.1);
            width: 450px;
        }
        h1 {
            text-align: center;
            color: #333;
            margin-bottom: 25px;
            font-weight: 700;
        }
        label {
```

```
Tomcat v11.0 Server at localhost  [Started, Synchronized]
```

Step 6: Run the Application

1. Run on Server:
   - Right-click the project > Run As > Run on Server.
   - Select Apache Tomcat and click Finish.
2. Access the Application:
   - Open a web browser and navigate to *http://localhost:8080/OnlineExamApp/exam.jsp*.



localhost:8000/JDBC_DEMO/exam.jsp

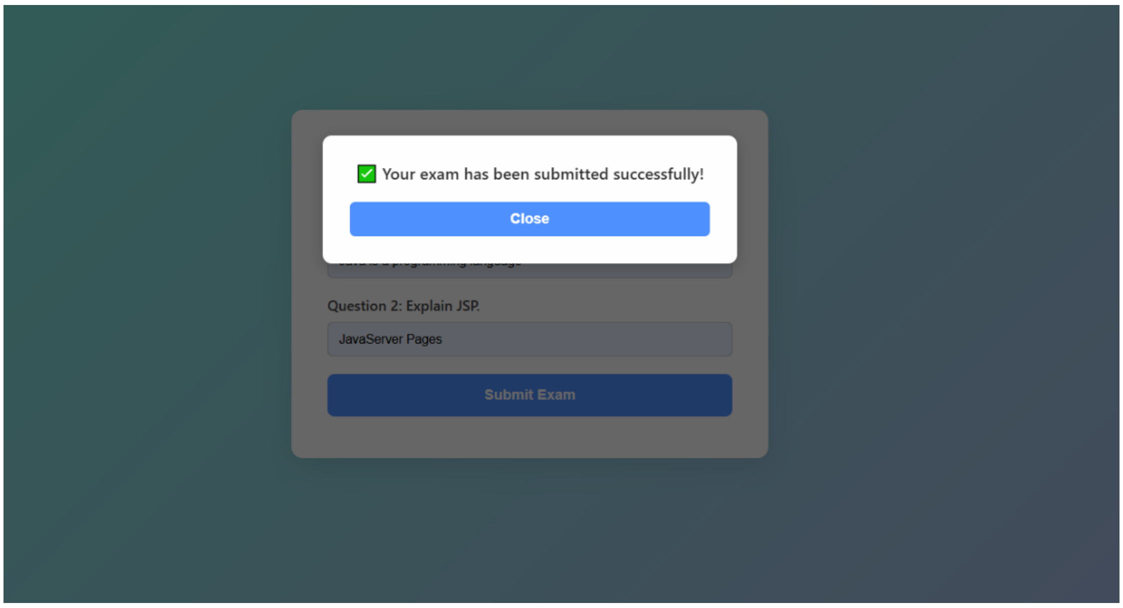
Online Examination

Question 1: What is Java?

Java is a programming language

Question 2: Explain JSP.

JavaServer Pages


✅ Your exam has been submitted successfully!

Java is a programming language

Question 2: Explain JSP.

JavaServer Pages

Step 7: Verify Database Entries

1. Check Submissions:
    - You can verify the entries in the Students table using your MySQL client to ensure the application is functioning correctly.

Result:

Thus, to develop Java programs using JSP and databases for conducting an online examination and displaying a student mark list is successfully executed.

Ex.No: 7 **Date:**	**Programs using XML, Schema, and XSLT/XSL**

Aim:

To create an XML document with a schema for validation and transform it into HTML using XSLT.

Prerequisites

- Java Development Kit (JDK) installed.
- Eclipse IDE
- A modern web browser (e.g., Chrome, Firefox, Edge) with support for XSLT transformation.

Procedure:

Step 1: Prepare XML, XSD, and XSLT Files

1. Create XML File:
    - Create a file named **students.xml.**
    - Student data with a processing instruction linking to the XSLT.

<u>students.xml</u>

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

<?xml-stylesheet type="text/xsl" href="transform.xsl"?>

<students>

   <student>

      <id>1</id>

      <name>Alice</name>

      <marks>85</marks>

   </student>

   <student>

      <id>2</id>

      <name>Bob</name>

      <marks>90</marks>
```

```
    </student>

    <student>

        <id>3</id>

        <name>Charlie</name>

        <marks>78</marks>

    </student>

</students>
```

2. Create XML Schema File:
    - Create a file named **students.xsd.**
    - XML Schema content defining the structure and constraints for the XML data.

students.xsd

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="students">

    <xs:complexType>

      <xs:sequence>

        <xs:element name="student" maxOccurs="unbounded">

          <xs:complexType>

            <xs:sequence>

              <xs:element name="id" type="xs:int"/>

              <xs:element name="name" type="xs:string"/>

              <xs:element name="marks" type="xs:int"/>

            </xs:sequence>

          </xs:complexType>

        </xs:element>

      </xs:sequence>

    </xs:complexType>
  </xs:element>

</xs:schema>
```

3. Create XSLT File:
   - Create a file named **transform.xsl.**
   - XSLT content which will transform the XML data into HTML.

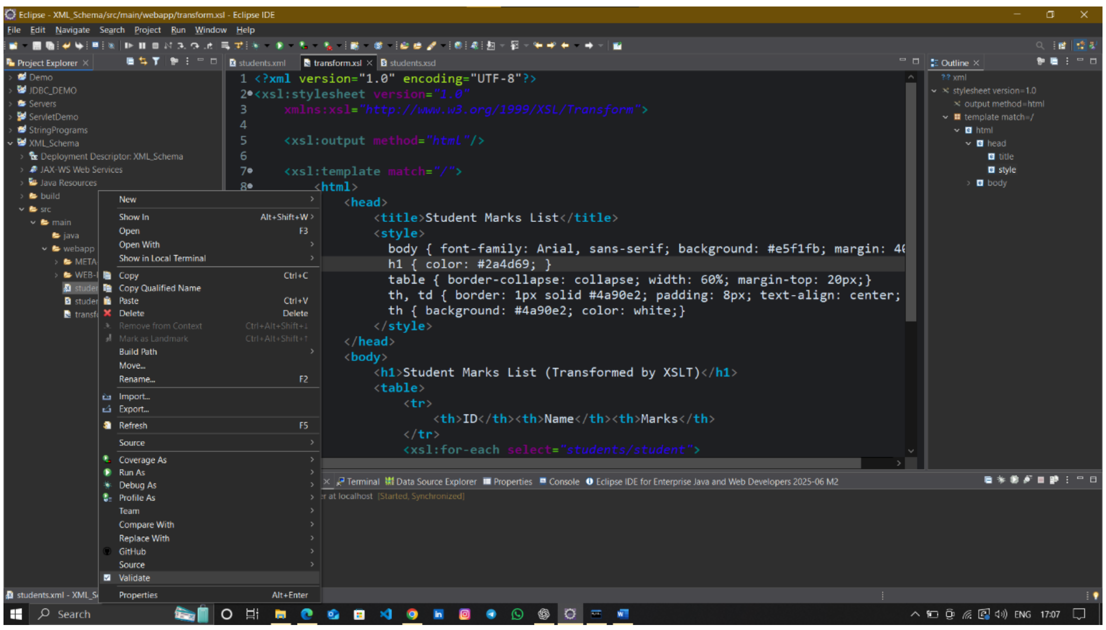
transform.xsl

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

<xsl:stylesheet version="1.0"

   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:output method="html"/>

   <xsl:template match="/">

      <html>

         <head>

            <title>Student Marks List</title>

            <style>

              body { font-family: Arial, sans-serif; background: #e5f1fb; margin: 40px;}

              h1 { color: #2a4d69; }

              table { border-collapse: collapse; width: 60%; margin-top: 20px;}

              th, td { border: 1px solid #4a90e2; padding: 8px; text-align: center; }

              th { background: #4a90e2; color: white;}

            </style>

         </head>

         <body>

            <h1>Student Marks List (Transformed by XSLT)</h1>

            <table>

               <tr>

                  <th>ID</th><th>Name</th><th>Marks</th>
```

```
                </tr>

                <xsl:for-each select="students/student">

                  <tr>

                    <td><xsl:value-of select="id"/></td>

                    <td><xsl:value-of select="name"/></td>

                    <td><xsl:value-of select="marks"/></td>

                  </tr>

                </xsl:for-each>

              </table>

            </body>

          </html>

    </xsl:template>

</xsl:stylesheet>
```

Step 2: Validate XML against Schema

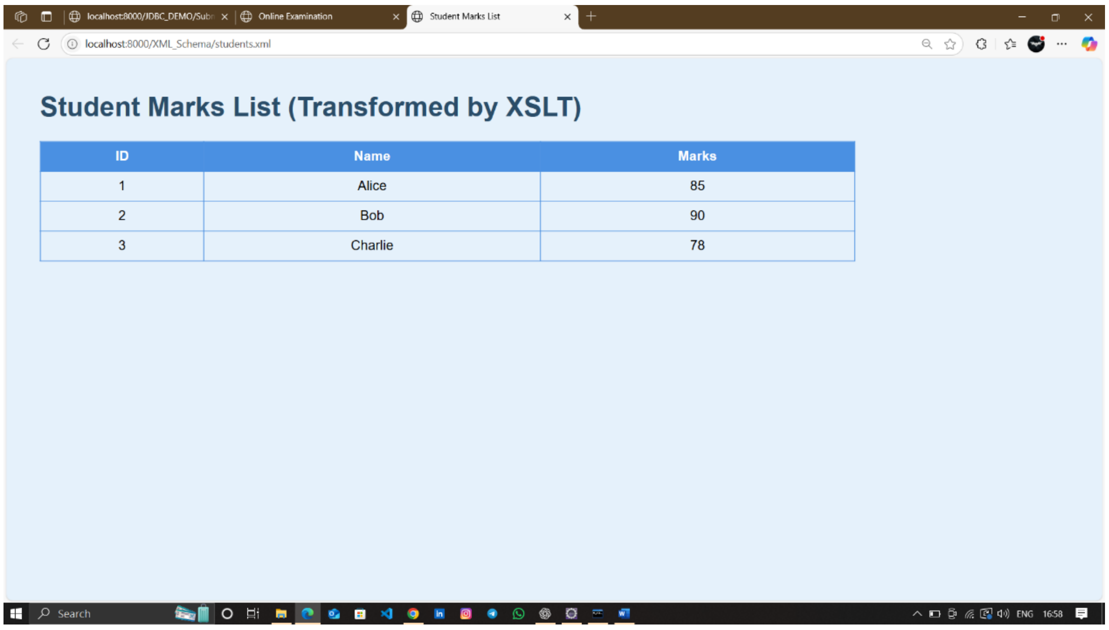
1. In Eclipse:
   - Right-click *students.xml* > *Validate.*
   - Eclipse will check the XML content against the students.xsd schema.
   - Confirm there are no validation errors.



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:output method="html"/>

    <xsl:template match="/">
        <html>
            <head>
                <title>Student Marks List</title>
                <style>
                  body { font-family: Arial, sans-serif; background: #e5f1fb; margin: 4
                  h1 { color: #2a4d69; }
                  table { border-collapse: collapse; width: 60%; margin-top: 20px;}
                  th, td { border: 1px solid #4a90e2; padding: 8px; text-align: center;
                  th { background: #4a90e2; color: white;}
                </style>
            </head>
            <body>
                <h1>Student Marks List (Transformed by XSLT)</h1>
                <table>
                    <tr>
                        <th>ID</th><th>Name</th><th>Marks</th>
                    </tr>
                    <xsl:for-each select="students/student">
```

Step 3: Run the Application

1. **Run on Server:**
    - Right-click the *project* > *Run As* > *Run on Server*.
    - Select Apache Tomcat and click Finish.
2. **Access the Application:**
    - Open a web browser and navigate to *http://localhost:8080/ XML_Schema/students.xml*.


localhost:8000/JDBC_DEMO/Sub
Online Examination
Student Marks List
localhost:8000/XML_Schema/students.xml
Search
ENG
16:58

Student Marks List (Transformed by XSLT)

ID	Name	Marks
1	Alice	85
2	Bob	90
3	Charlie	78

Result:

Thus, to create an XML document with a schema for validation and transform it into HTML using XSLT.