Vel Tech Multi Tech

Dr.Rangarajan Dr.Sakunthala Engineering College

An Autonomous Institution

(Approved by AICTE,New Delhi & Affiliated to Anna University,Chennai)

ISO 9001:2015 Certified Institution,
Accredited by NBA (BME, CSE, ECE, EEE, IT & MECH),
Accredited by NAAC with 'A' Grade with CGPA of 3.49,
#42, Avadi-Vel Tech Road, Avadi, Chennai-600062, Tamilnadu, India

231ITV67 - WEB TECHNOLOGIES LAB INTEGRATED

NAME :

REGISTER NO :

ROLL NO :

BRANCH :B.TECH ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

YEAR : III

SEMESTER : V

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VISION

To promote Centre of excellence through effectual Teaching and Learning, imparting the contemporary knowledge centric education through innovative research in multidisciplinary fields.

MISION

- To impart quality technical skills through practicing, knowledge updating in recent technology and produce professionals with multidisciplinary and leadership skills.
- To promote innovative thinking for design and development of software products of varying complexity with intelligence to fulfil the global standards and demands.
- To inculcate professional ethics among the graduates and to adapt the changing technologies through lifelong learning.

Vel Tech Multi Tech

Dr.Rangarajan Dr.Sakunthala Engineering College

An Autonomous Institution

(Approved by AICTE,New Delhi & Affiliated to Anna University,Chennai)

ISO 9001:2015 Certified Institution,
Accredited by NBA (BME, CSE, ECE, EEE, IT & MECH),
Accredited by NAAC with 'A' Grade with CGPA of 3.49,
#42, Avadi-Vel Tech Road, Avadi, Chennai-600062, Tamilnadu, India

CERTIFICATE

Name: _____

Year: ___ **Semester:** ___ **Programme**: B.Tech – Artificial Intelligence and Data Science,

University Register No: _____**College Roll No:** _____Certified that this is the bonafide record of work done by the above student in **231AIV16 – WEB TECHNOLOGIES LAB INTEGRATED** during the academic year 2025-26.

Signature of Course In charge **Signature of Head of the Department**

Submitted for the University Practical Examination held on .....at VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE, #42, AVADI- VEL TECH ROAD, AVADI, CHENNAI-600062.

Signature of Examiners

Internal Examiner…….. **Examiner……**

Date…….

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEOs	PROGRAMME EDUCATIONAL OBJECTIVES
PEO1	Train the graduates with the potential of strong knowledge in the respective field and to create innovative multidisciplinary solutions for challenges in the society
PEO2	Groom the engineers to understand, analyse different nature of data and use Machine Learning techniques to develop software systems with varying complexity for data intensive applications
PEO3	To practice professionalism among the graduates and reflect good leadership skills with ethical standards and continued professional development through lifelong learning.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

PSOs	PROGRAMME SPECIFIC OUTCOMES
PSO1	To impart theoretical knowledge in the respective field along with recent industrial tools and techniques to solve societal problems
PSO2	Apply the core competency obtained in the field of Machine Learning for analysis, design and development of computing systems for multi-disciplinary problems
PSO3	Acquire knowledge in the field of intelligence, deep learning and develop software solutions for security and analytics of large volume of data.

191IT52A-WEB TECHNOLOGY

LABORATORY COURSE OBJECTIVES:

- To design web pages using Scripting languages.
- To learn server side programming using servlets and JSP.
- To develop web pages using XML/XSLT.

COURSE OUTCOMES:

COURSE OUTCOMES On completion of the course, students will be able to	
CO1	Design web pages using markup languages like HTML and XHTML.
CO2	Create dynamic web pages using DHTML and java script.
CO3	Implementation of client side web pages and server side web pages.
CO4	Represent web data using XML and develop web pages using JSP.
CO5	Implement various web services and their communication with other web services.

CORRELATION BETWEEN CO-PSO

CO	**PSO1**	**PSO2**	**PSO3**
CO1	3	3	2
CO2	3	3	2
CO3	3	3	2
CO4	3	3	2
CO5	3	3	2
CO	**3**	**3**	**2**

S.NO	DATE	LIST OF EXPERIMENTS	PAGE NO	SIGN
1		Create a web page with the following using HTML: i) To embed an image map in a web page. ii) To fix the hot spots. iii) Show all the related information when the hot spots are clicked.		
2		Create a web page with all types of Cascading style sheets.		
3		Create a web page with all types of Cascading style sheets.		
4		Write programs in Java using Servlets: a. To invoke servlets from HTML forms. b. Session Tracking.		
5		Write programs in Java to create three-tier applications using JSP and Databases: a. For conducting on- line examination b. For displaying student mark list. Assume that student information is available in a database which has been stored in a database server.		
6		Programs using XML –Schema –XSLT/XSL.		
7		Programs using Angular JS.		

SUPPORTING MARKS (75)					
Attendance and Punctuality	**Pre-Lab Preparation**	**Practical Skills and Experimentation**	**Lab Record(s)**	**Post-Lab Quiz**	**TOTAL**
10	15	25	15	10	**75**

Ex No: 1

FIX CREATION OF IMAGE MAP TO HOTSPOTS

AIM

To create a web page with image map to fix the hotspots and show its related information.

SOFTWARE USED

Dreamweaver or Notepad and browser.

DESCRIPTION

In Web page development, an image map is a graphic image defined so that a user can click on different areas of the image and be linked to different destinations. You make an image map by defining each of the sensitive areas in terms of their x and y coordinates (that is, a certain horizontal distance and a certain vertical distance from the left-hand corner of the image). With each set of coordinates, you specify a Uniform Resource Locator or Web address that will be linked to when the user clicks on that area.
There are three HTML elements used to create image maps:
<img> specifies the location of the image to be included in the map.
<map> is used to create the map of clickable areas.
<area> is used within the map element to define the clickable areas.

CODE main.html <html>

```
<head>
<BODY bgcolor="#gop6876cgdt5564ss">
<img src ="indiamap.jpeg" usemap="#indiamap" />
<map name=indiamap>
<area shape="circle" coords="500,1092,15" href="tamilnadu.html" alt="Tamilnadu">
<area shape="rect" coords="376,1076,406,1100" href="karnataka.html" alt="Karnataka">
</map>
</head>
</BODY>
</html>
```

tamilnadu.html

```
<html>
<head><title>Tanil Nadu - India</title></head>
<body bgcolor="palegreen">
<h1><center>Tamil Nadu</center></h1>
<h3>is one of the 29 states of India. Its capital and largest city is Chennai.
Tamil Nadu lies in the southernmost part of the Indian Peninsula and is
bordered by the States of puducherry, Kerala, Karnataka, Andhra Pradesh.
</h3>
<h3>
<ul>
<li>Districts<i> - 33</i>
<li>Capital City<i> - Chennai</i>
<li>Largest City<i> - Chennai</i>
<li>Governor<i> - Banwarilal Purohit</i>
<li>Chief Minister<i> - Edappadi K. Palaniswami </i>
<li>Population<i> - 72,147,030</i>
```

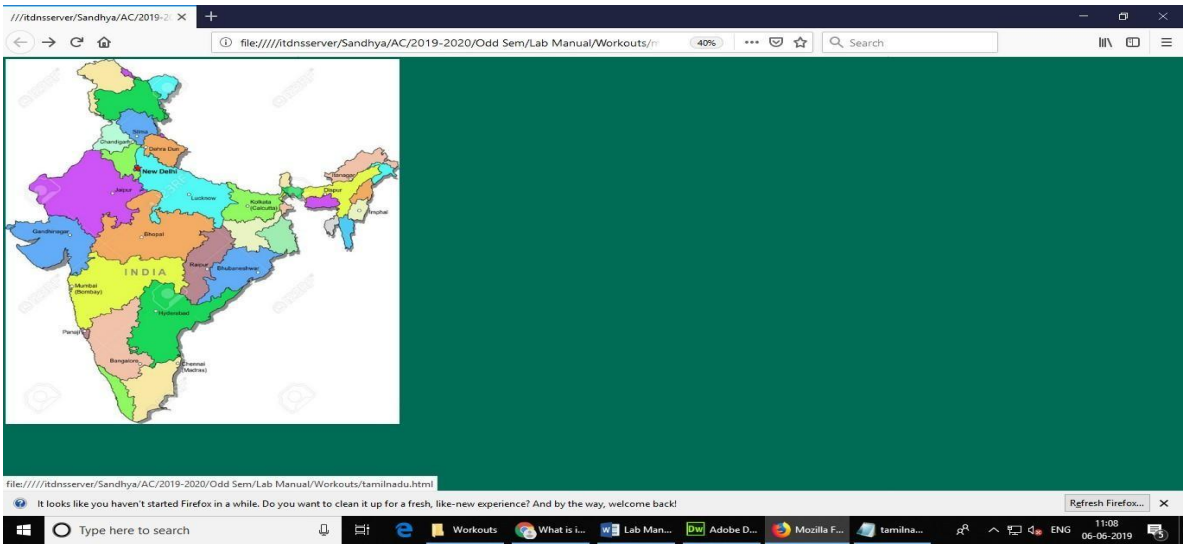
Tourist spots - Mamallapuran, Ooty, Kodaikanal, Marina,
Mudurai Meenakshi Amman Temple, Thanjavur etc.,

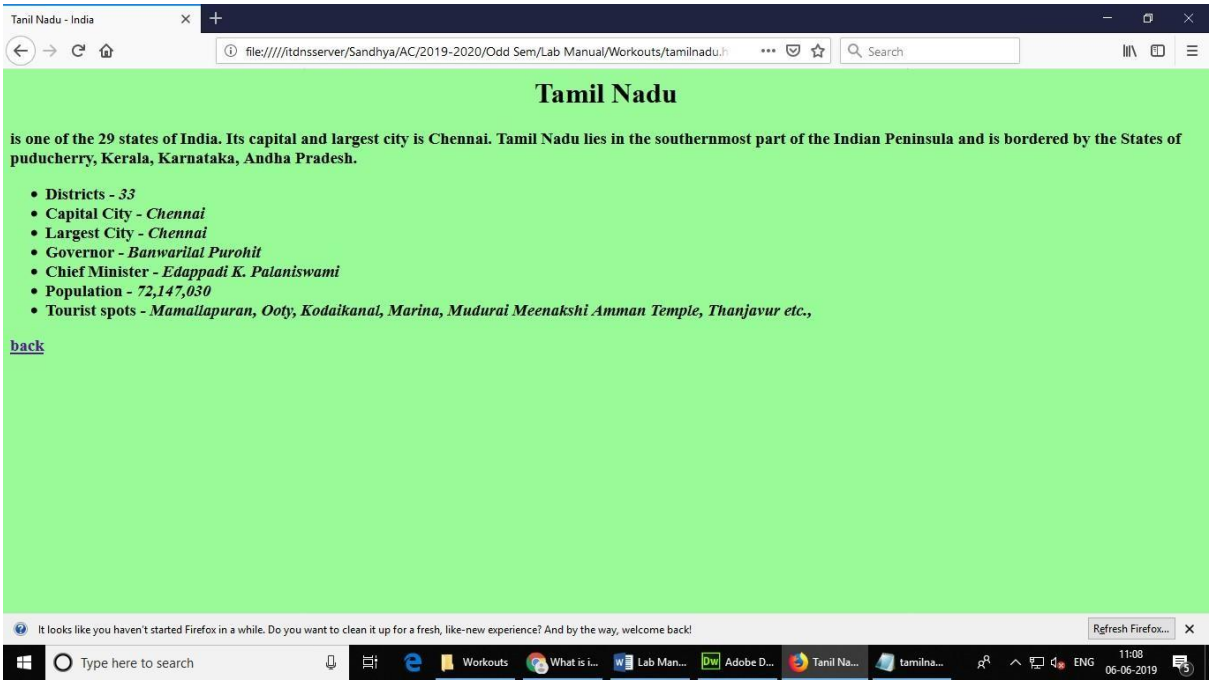
back
</body> </html>

karnataka.html

```
<html>
<head><title>Karnataka - India</title></head>
<body bgcolor="wheat">
<h1><center>Karnataka</center></h1>
<h3>
<ul>
<li>Districts<i> - 30</i>
<li>Capital City<i> - Bangalore</i>
<li>Largest City<i> - Bangalore</i>
<li>Governor<i>- Vajubhai Vala</i>
<li>Chief Minister<i> - H. D. Kumaraswamy </i>
<li>Population<i> - 61,130,704</i>
<li>Tourist spots<i> - Gol Gumbaz, Mysore Palace, Keshava Temple etc.,</i> </ul>
</h3>
<a href="main.html">back</a>
</body>
</html>
```

OUTPUT





RESULT

Thus the Web Page had been created using image map to fix the hotspots and relevant information had been displayed.

Ex No: 2

CREATION OF A WEB PAGE WITH ALL TYPES OF CASCADING STYLE SHEETS

AIM

To create a webpage with all types of Cascading Style Sheets.

SOFTARE USED

Dreamweaver or Notepad and browser.

DESCRIPTION

Cascading Style Sheet (CSS) is used to set the style in web pages which contain HTML elements. It sets the background color, font-size, font-family, color etc property of elements in a web pages.

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

ALGORITHM

Internal CSS:

STEP 1: Create a HTML program with <style> tag.
STEP 2: Inside the <style> tag, specify the format required for that web page.
STEP 3: Run the program with a web browser.

External CSS:

STEP 4: Open a notepad, type the needed CSS in it and save it with .css extension.
STEP5: Refer this .css file in the HTML using the tag <link>.
STEP 6: Run the program with a web browser.

CODE style.html <html>

```
<head>
<title>FLOWERS</title>
<!--Extended Style Sheet -->
<link rel="stylesheet" type="text/css" href="style.css">
<!-- Embed Style Sheet-->
<style type="text/css"> p{
background-color:
lightgrey;
text-align: justify;
margin: 2em 7em;
}
</style>
</head>
<body id="body">
<h1>FLOWER</h1>
<p>
<span style="font: 200 x-large fantasy">Flower</span>
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 the mankind.
We have seen the flowers of many kinds and to many colors.
In India we see the flowers like
```

```
</p>
<!-- Inline Sytle Sheet-->
<table style="background-position: center;text-align: center;padding: 3px;">
<tr>
<td align="left">
<div class="div">
<ul>
<li><a href="">Lily</a></li>
<li><a href="">Lotus</a></li>
<li><a href="">Rose</a></li>
<li><a href="">Jasmine</a></li>
</ul>
</div>
</td>
</tr>
</table>
</body> </html>
```

style.css

```
h1,h2{ text-decoration:
underline; font-style:
italic; text-align: center;
}
#body{ background-color:
tan; border: red dotted;
text-align: center;
} .div{
border: peru solid ;
} *{
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

RESULT

Thus a web page has been created with all types of CSS.

Ex No: 3

INSTALLATION OF APACHE TOMCAT WEB SERVER

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 use, 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 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. Tomcat 8.5 (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 0: Create a Directory to Keep all your Works

I shall assume that you have created a directory called "c:\myWebProject" (for Windows) or "~\myWebProject" (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 1: 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:\myWebProject". Tomcat shall be unzipped into directory "c:\myWebProject\apache-tomcat-9.0.{xx}".

For EASE OF USE, 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 webapps. 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 2: 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\jdk11.0.{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 3: 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:\myWebProject\tomcat\conf" (for Windows) or "~/myWebProject/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 3(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 3(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>
<param-value>true</param-value>
</init-param>
```

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

</servlet>

Step 3(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 4: Start Tomcat Server

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

Step 4(a) Start Server

For Windows

I shall assume that Tomcat is installed in "c:\myWebProject\tomcat". Launch a CMD shell and issue:

c: // Change drive

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

// Run startup.bat to start tomcat server

Step 4(b) Start a Client to Access the Server

Start a browser (Firefox, Chrome) as an HTTP client. Issue URL "http://localhost:9999" to access 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://*serverHostnameOrIPAddress*:9999".
(Optional) Try issuing URL http://localhost:9999/examples to view the servlet and JSP examples. Try running some of the servlet examples.

Step 4(c) Shutdown Server

For Windows

You can shutdown the tomcat server by either:

Press Ctrl-C on the Tomcat console; OR

Run "<TOMCAT_HOME>\bin**shutdown.bat**" script. Open a new "cmd" and issue: **c:**// Change the current drive

cd \myWebProject\tomcat\bin// Change directory to your Tomcat's binary directory **shutdown**// Run shutdown.bat to shutdown the server

RESULT:

Thus, the experiment “Installation of Apache Tomcat Web Server” was successfully completed. The Tomcat server was installed, configured, and verified to run correctly.

Ex No: 4

INVOKING SERVLETS FROM HTML FORMS-SESSION TRACKING

AIM

To create a simple application to perform session tracking using servlet.

SOFTWARE REQUIRED

Tomcat Server

DESCRIPTION

Session simply means a particular interval of time. Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet.

Session Tracking Techniques

There are four techniques used in Session tracking:

- **Cookies • Hidden Form Field • URL Rewriting • HttpSession**

ALGORITHM

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 GenericServlet.
2. Handle the request from the client by using the method service() of GenericServlet 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.

CODE

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
 <a href="login.html">Login</a> |
 <a href="LogoutServlet">Logout</a> |
 <a href="ProfileServlet">Profile</a>
 <hr> login.html
<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>  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(password.equals("admin123")){
         out.print("You are successfully logged in!");
         out.print("<br>Welcome, "+name);

         Cookie ck=new Cookie("name",name);
         response.addCookie(ck);
      }else{ out.print("sorry, username or password
         error!");
         request.getRequestDispatcher("login.html").include(request, response); }

      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!"); }
}
```

File: 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 ck[]=request.getCookies(); if(ck!=null){
       String name=ck[0].getValue();
      if(!name.equals("")||name!=null){
         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();
   }
}
```

web.xml

```
   <?xml version="1.0" encoding="UTF-8"?>
   <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml
   /ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

     <servlet>
      <description></description>
```

```
  <display-name>LoginServlet</display-name>
  <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>
 <servlet>
  <description></description>
  <display-name>ProfileServlet</display-name>
  <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>
 <servlet>
  <description></description>
  <display-name>LogoutServlet</display-name>
  <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>
</web-app>
```

OUTPUT


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Please login first
Name:
Password:
login
You can't visit profile page directly


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Please login first
Name:
Password:
login
You can't visit profile page directly


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Welcome to Profile
Welcome, sonoo
This is Profile Page

RESULT

Thus a simple application has been created for invoking servlets from HTML forms.

Ex No: 5a

CREATION OF THREE-TIER APPLICATIONS USING JSP AND DATABASES - For conducting on-line examination

AIM

To create a three-tier application using JSP and Databases for the conduction of online-examination.

SOFTWARE REQUIRED

Tomcat Server

DESCRIPTION

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

ALGORITHM

1. Design the HTML page (ExamClient.html) with the following
   a. Create a form to get the input from the user.
   b. Use radio buttons to make various options for the questions.
   c. Set the URL of the server (ExamServer.jsp) as the value of the action attribute.
   d. Use submit button to invoke the server and send the form data to the server.
2. Create the JSP file with the following
   a. Read the input from the client.
   b. Retrieve the answers from the database.
   c. Match the answers from the user with the correct answers from the database table.
   d. For each correct answer increment the mark by 5.
   e. Server displays the mark and result to the client as a response.

CODE **ExamServer.jsp:**

```
<%@page contentType="text/html" language="java" import="java.sql.*"%>
<html>
<head>
<title>Online Exam Server</title>
<style type="text/css"> body{background-color:black;font-family:courier;color:blue}
</style>
</head>
<body>
<h2 style="text-align:center">ONLINE EXAMINATION</h2>
<p>
<a href="ExamClient.html">Back To Main Page</a>
</p>
<hr/>
<%
String str1=request.getParameter("ans1");
String str2=request.getParameter("ans2");
String str3=request.getParameter("ans3");
int mark=0;
```

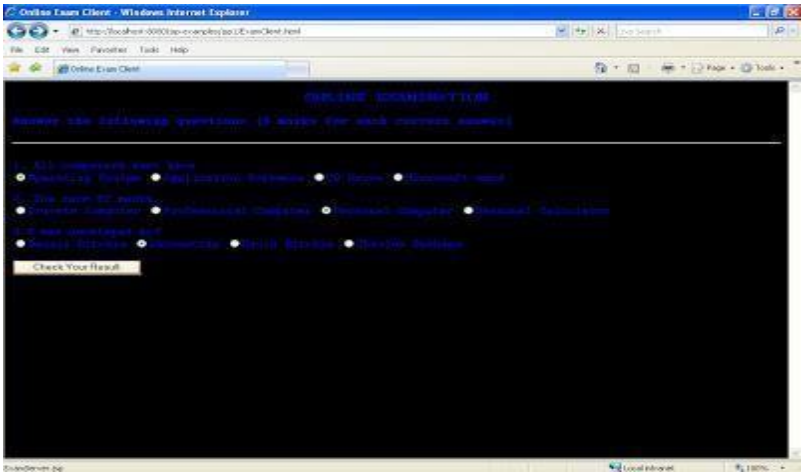
```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:examDS");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT * FROM examTab");
int i=1; while(rs.next())
{ if(i==1)
{
  String dbans1=rs.getString(1); if(str1.equals(dbans1))
  {
   mark=mark+5;
  }
} if(i==2)
{
  String dbans2=rs.getString(1); if(str2.equals(dbans2))
  {
   mark=mark+5;
  }
} if(i==3)
{
  String dbans3=rs.getString(1); if(str3.equals(dbans3))
  {
   mark=mark+5;
  }
 } i++;
}
if(mark>=10)
{          out.println("<h4>Your          Mark          Is          :          "+mark+"</h4>");
 out.println("<h3>Congratulations....! You Are Eligible For The Next Round...</h3>");
} else
{ out.println("<h4>Your Mark is : "+mark+"</h4>"); out.println("<h3>Sorry....!!
You Are Not Eligible For  The Next Round...</h3>"); }
%>
</form>
</body>
</html>
```

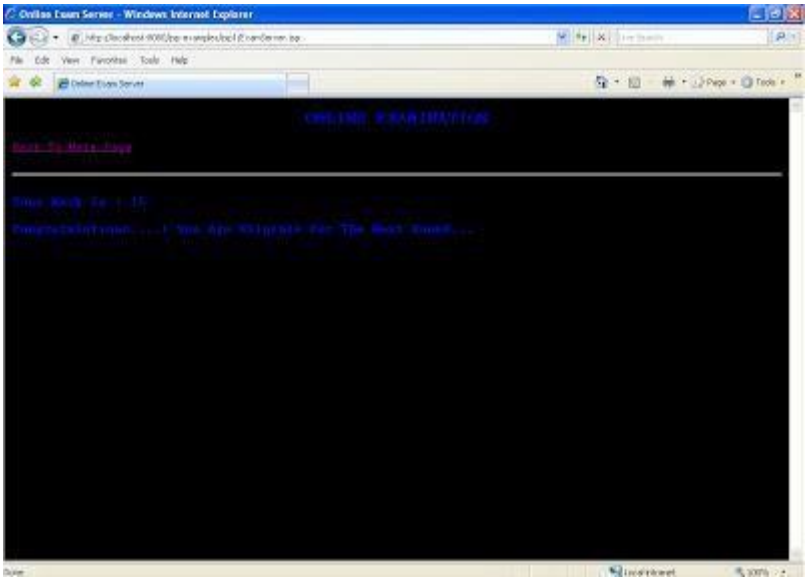
ExamClient.HTML:

```
<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/>
C. was developed by?<br/>
<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>
```

OUTPUT





RESULT

Thus a simple web application using JSP and Database had been created for the conduction of online examination.

Ex No: 5b

CREATION OF THREE-TIER APPLICATIONS USING JSP AND DATABASES - For displaying student mark list

AIM

To create a three-tier application using JSP and Databases for displaying the student mark list.

SOFTWARE REQUIRED

Tomcat Server

DESCRIPTION

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

ALGORITHM

1. Design the HTML page (stud.html) with the following
   a. Create a form to get the input (Register Number) from the user.
   b. Set the URL of the server (marklist.jsp) as the value of the action attribute.
   c. Use submit button to invoke the server and send the form data to the server.
2. Create the JSP file with the following
   a. Read the parameter value (Register Number) from the form by using the method getParameter().
   b. Server retrieves the details from the database table with respect to the form input.
   c. Server displays the mark list to the client as the response.

CODE

marklist.jsp:

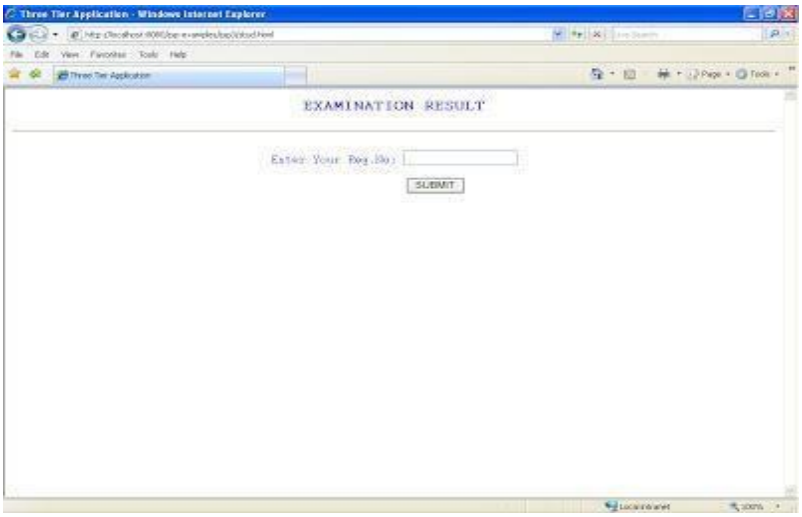
```
<%@ page contentType="text/html" language="java" import="java.sql.*"%>
<html>
<head>
<title>Three Tier Application</title>
<style type="text/css"> body{color:blue;font-family:courier;text-align:center}
</style>
</head>
<body>
<h2>EXAMINATION RESULT</h2><hr/>
<%
String str=request.getParameter("regno");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:markDS");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT*FROM markTab WHERE rno="+str);
while(rs.next())
{
%>
Register No:<%=rs.getObject(1)%><br/>
```

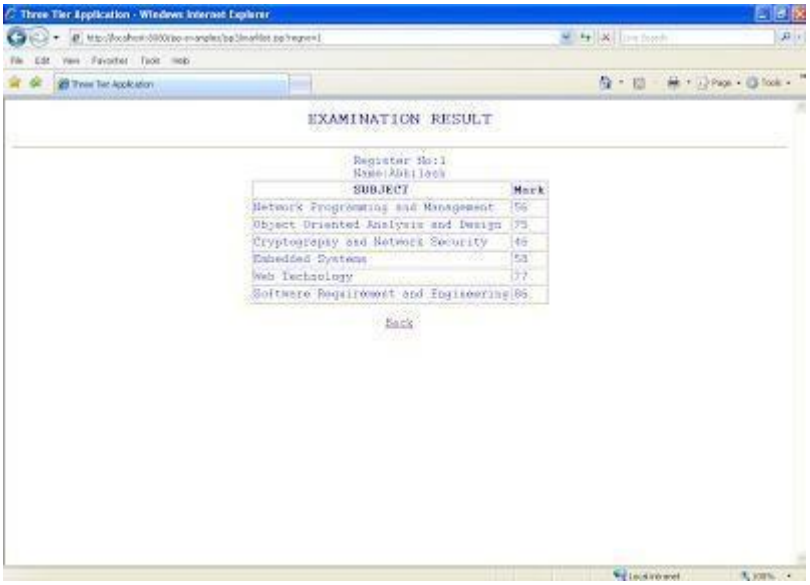
```
Name:<%=rs.getObject(2)%><br/>
<table border="1">
<th>SUBJECT</th><th>Mark</th> <tr><td>Network
Programming and
Management</td><td><%=rs.getObject(3)%></td></tr>
<tr><td>Object Oriented Analysis and Design</td><td><%=rs.getObject(4)%></td></tr>
<tr><td>Cryptography and Network Security</td><td><%=rs.getObject(5)%></td></tr>
<tr><td>Embedded Systems</td><td><%=rs.getObject(6)%></td></tr>
<tr><td>Web Technology</td><td><%=rs.getObject(7)%></td></tr>
<tr><td>Software Requirement and Engineering</td><td><%=rs.getObject(8)%></td></tr>
</table>
<%
}
%>
<br/>
<a href="stud.html">Back</a>
</body>
</html>
```

stud.html

```
<html>
<head>
<title>Three Tier Application</title>
<style type="text/css"> 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>
```

OUTPUT





RESULT

Thus a simple web application using JSP and Database had been created displaying student mark list.

Ex No: 6

PROGRAMS USING XML – SCHEMA – XSLT/XSL AIM

To write a XML scheme to generate CD Collection details.

SOFTWARE REQUIRED

Dreamweaver or Notepad and Browser.

DESCRIPTION

In HTML documents, tags are predefined but in XML documents, tags are not predefined. World Wide Web Consortium (W3C) developed XSL to understand and style an XML document, which can act as XML based Stylesheet Language.

An XSL document specifies how a browser should render an XML document. **Main parts of XSL Document**

- XSLT: It is a language for transforming XML documents into various other types of documents.
- XPath: It is a language for navigating in XML documents.
- XQuery: It is a language for querying XML documents.
- XSL-FO: It is a language for formatting XML documents.

How XSLT Works?

The XSLT stylesheet is written in XML format. It is used to define the transformation rules to be applied on the target XML document. The XSLT processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. At the end it is used by XSLT formatter to generate the actual output and displayed on the end-user.

ALGORITHM

Step 1: Start the program

Step 2: Use Xml Style Sheet code to define link

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

3: Use the catalog tag to define CD collection details.

Step 4: Use the necessary heading for appropriate XML tag.

Step 5: Provide necessary information for CD collection details Step

6: Stop the program

CODE CDCatalog.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<?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>
```

RESULT

Thus the program to write a XML scheme to generate CD Collection details.

Ex No: 7

PROGRAMS USING ANGULAR JS

Aim:
To develop a web-based Student Management System using AngularJS that allows users to add, display, and delete student records dynamically on a single page.

Software Required:

1. **Operating System:** Windows / Linux / macOS
2. **Code Editor:** Visual Studio Code / Sublime Text / Notepad++
3. **Browser:** Google Chrome / Firefox
4. **Software / Libraries:**
    - AngularJS (v1.8.x)
    - HTML5 o CSS3
    - JavaScript

Algorithm:

1. Start
2. Initialize the AngularJS application module (myApp).
3. Create a controller (StudentController) to manage student data.
4. Define an array to store student objects.
5. Implement functions:
    - addStudent() → to add new student data to the array.
    - deleteStudent(index) → to remove a student from the array.
6. Bind form inputs to model variables using ng-model.
7. Display the list of students using ng-repeat.
8. Allow real-time updates and deletion without refreshing the page.
9. Stop

Steps:

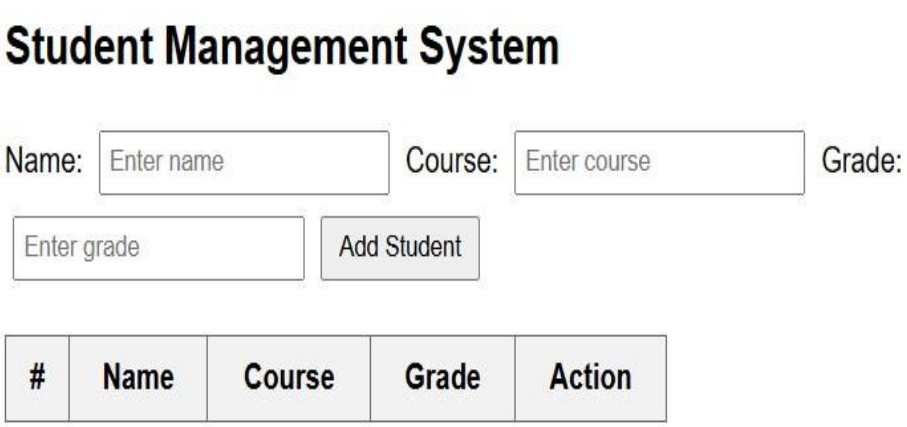
1. Create an HTML file — studentApp.html
2. Include AngularJS Library from CDN.
3. Define the AngularJS application using ng-app.
4. Create a controller to manage data and functions.
5. Bind HTML elements with ng-model, ng-repeat, and ng-click.
6. Test the app by adding and deleting students dynamically.

Program:

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Management System - AngularJS</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    body { font-family: Arial; margin: 40px; }     table { border-
collapse: collapse; width: 60%; margin-top: 20px; }     th, td {
border: 1px solid #666; padding: 10px; text-align: center; }     th {
background-color: #f2f2f2; }     input { padding: 5px; margin: 5px; }
button { padding: 5px 10px; }
  </style>
</head>

<body ng-app="myApp" ng-controller="StudentController">
  <h2>Student Management System</h2>

  <div>
    <label>Name:</label>
    <input type="text" ng-model="studentName" placeholder="Enter name">

    <label>Course:</label>
    <input type="text" ng-model="studentCourse" placeholder="Enter course">

    <label>Grade:</label>
    <input type="text" ng-model="studentGrade" placeholder="Enter grade">

    <button ng-click="addStudent()">Add Student</button>
  </div>

  <table>
    <tr>
      <th>#</th>
      <th>Name</th>
      <th>Course</th>
      <th>Grade</th>
      <th>Action</th>
    </tr>
    <tr ng-repeat="student in students">
      <td>{{$index + 1}}</td>
      <td>{{student.name}}</td>
      <td>{{student.course}}</td>
      <td>{{student.grade}}</td>
      <td><button ng-click="deleteStudent($index)">Delete</button></td>
    </tr>
  </table>

  <script>
    var app = angular.module("myApp", []);
    app.controller("StudentController", function($scope) {
      $scope.students = [];
```

```
    $scope.addStudent = function() {
      if ($scope.studentName && $scope.studentCourse && $scope.studentGrade) {
        $scope.students.push({
name: $scope.studentName,
course: $scope.studentCourse,
          grade: $scope.studentGrade
        });
        $scope.studentName = "";
        $scope.studentCourse = "";
        $scope.studentGrade = "";
      }
    };

    $scope.deleteStudent = function(index) {
      $scope.students.splice(index, 1);
    };
  });
 </script>
</body>
</html>
```

Output:



Student Management System

Name: Alice Course: Angular JS Grade:

A Add Student

#	Name	Course	Grade	Action

Student Management System

Name: Enter name Course: Enter course Grade:

Enter grade Add Student

#	Name	Course	Grade	Action
1	Alice	Angular JS	A	Delete

Result:

Thus the program for Angular JS had been created.