**Ex No. : 1** **Image Mapping**
**Date** :

## Aim

The aim of this exercise is to understand and implement image mapping in HTML. Image mapping enables the creation of clickable regions on an image, linking each area to different destinations, thereby enhancing user interaction and navigation on web pages.

## Procedure

1. Create an HTML File: Initiate an HTML file and insert an image utilizing the tag.
2. Implement Image Map: Utilize the usemap attribute within the tag to associate the image with a map.
3. Define Map: Use the tag with a unique name, specifying the clickable areas using the tag.
4. Utilize the shape attribute to define the shape of the area (e.g., rectangle, circle, polygon).
5. Specify the coords attribute to determine the coordinates of the area.
6. Employ the href attribute within the tag to indicate the link destination.
7. Style and Enhance: Add CSS to customize the appearance of the map and associated elements for better visual representation.
8. Test and Review: Test the functionality of the created image map to ensure the accuracy of clickable regions and their linked destinations.

## Program

index.html:

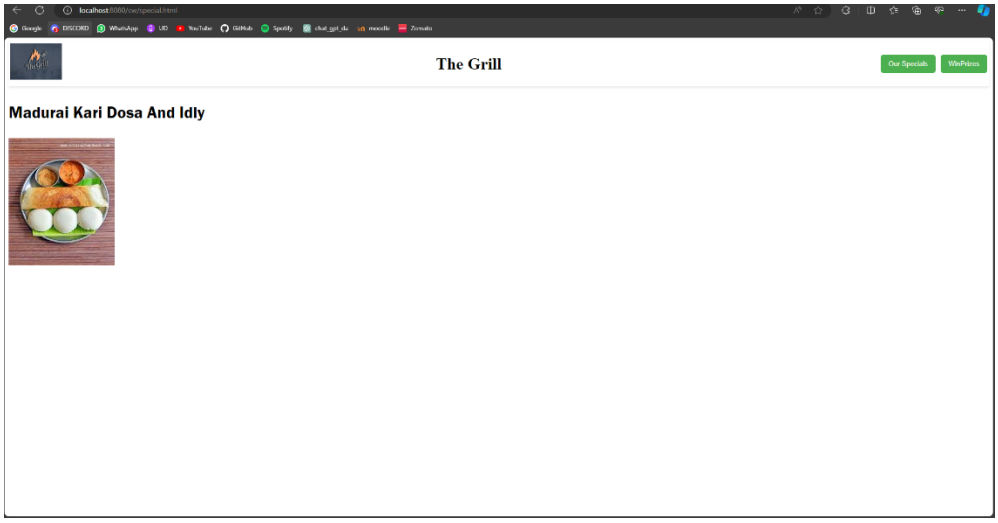
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>minjur bhavan</title>
    <style>
        .header {
    display: flex;
    justify-content: space-between;
```

```
    align-items: center;
    padding: 3px;
    background-color: #fff;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }
  .logo {
    display: flex;
    align-items: center;
    height: 10px;
  }
  .logo img {
    height: 70px;
    width: 100px;
    margin-top: -10px;/* Adjust as needed */
  }
  .menu-right {
    display: flex;
    align-items: center;
  }
  .rating {
    margin-right: 20px;
  }
  .header-button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 15px;
    border: none;
    cursor: pointer;
    border-radius: 5px;
    margin-left: 10px;
  }

    </style>
</head>
<body>
    <div class="header">
        <div class="logo">
          <img src="download.jpg" alt="Logo"> <!-- Replace with your logo URL
-->
```

```
        </div>
        <h1>The Grill</h1>
        <div class="menu-right">
          <a href="special.html">
            <button class="header-button">Our Specials</button>
          </a>
          <a href="rating.html">
            <button class="header-button" >WinPrizes</button>
          </a>

        </div>
      </div>
    <p class="heading" style="font-weight: bolder;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    font-size: xx-large;">Madurai Kari Dosa And Idly</p>
    <img src="dosa_idly.jpg" usemap="#food" alt="picture"/>
    <map name="food">
        <area shape="circle" coords="60 147 30"
href="https://www.google.com/search?q=dosa+idly+%5Bpicture&tbm=isch&v
ed=2ahUKEwiltN-4z8mAAxWs5DgGHUjOAKYQ2-
cCegQIABAA&oq=dosa+idly+%5Bpicture&gs_lcp=CgNpbWcQAzoKCAAQig
UQsQMQQzoHCAAQigUQQzoFCAAQgAQ6CAgAEIAEELEDOgsIABCABB
CxAxCDAToICAAQsQMQgwE6BAgAEAM6BggAEAgQHjoECAAQHlDGB1jE
NmCbOGgAcAB4AYABYIgBzhCSAQIzMJgBAKABAaoBC2d3cy13aXotaW1
nsAEAwAEB&sclient=img&ei=9mbQZKXhIqzJ4-
EPyJyDsAo&bih=775&biw=1440" alt="idly" target="_blank">
        <area shape="circle" coords="110 157 30"
href="https://www.google.com/search?q=dosa+idly+%5Bpicture&tbm=isch&v
ed=2ahUKEwiltN-4z8mAAxWs5DgGHUjOAKYQ2-
cCegQIABAA&oq=dosa+idly+%5Bpicture&gs_lcp=CgNpbWcQAzoKCAAQig
UQsQMQQzoHCAAQigUQQzoFCAAQgAQ6CAgAEIAEELEDOgsIABCABB
CxAxCDAToICAAQsQMQgwE6BAgAEAM6BggAEAgQHjoECAAQHlDGB1jE
NmCbOGgAcAB4AYABYIgBzhCSAQIzMJgBAKABAaoBC2d3cy13aXotaW1
nsAEAwAEB&sclient=img&ei=9mbQZKXhIqzJ4-
EPyJyDsAo&bih=775&biw=1440" alt="idly" target="_blank">
        <area shape="circle" coords="158 154 30"
href="https://www.google.com/search?q=dosa+idly+%5Bpicture&tbm=isch&v
ed=2ahUKEwiltN-4z8mAAxWs5DgGHUjOAKYQ2-
cCegQIABAA&oq=dosa+idly+%5Bpicture&gs_lcp=CgNpbWcQAzoKCAAQig
UQsQMQQzoHCAAQigUQQzoFCAAQgAQ6CAgAEIAEELEDOgsIABCABB
CxAxCDAToICAAQsQMQgwE6BAgAEAM6BggAEAgQHjoECAAQHlDGB1jE
```

```
NmCbOGgAcAB4AYABYIgBzhCSAQIzMJgBAKABAaoBC2d3cy13aXotaW1
nsAEAwAEB&sclient=img&ei=9mbQZKXhIqzJ4-
EPyJyDsAo&bih=775&biw=1440" alt="idly" target="_blank">
        <area shape="poly" coords="37 92 177 88 196 130 23 134"
href="https://www.google.com/search?sca_esv=554340491&q=dosa&tbm=is
ch&source=lnms&sa=X&ved=2ahUKEwjQq4X30smAAxVHbmwGHfEfCh0Q
0pQJegQICRAB&biw=1440&bih=775&dpr=1" alt="idly" target="_blank">

    </map>


</body>
</html>
```

## Output




## Result

The implementation of image mapping was successful. The image map was created and the clickable regions were linked to their respective destinations.

## Ex No. : 2 CSS (Internal, External, Inline)
## Date :

## Aim

The objective of this exercise is to explore and implement various approaches to apply Cascading Style Sheets (CSS) within an HTML document. The aim is to comprehend the three primary methods of incorporating CSS—internal, external, and inline—offering flexibility and control over the presentation and styling of HTML content.

## Procedure

1. Create HTML Structure: Develop an HTML file and incorporate content representing the subject matter.
2. Implement Internal CSS: Insert a <style> tag within the HTML's <head> section to apply internal CSS styles directly to the document.
3. External CSS Linkage: Create an external CSS file, and use the tag within the HTML to connect the external CSS file to the document, enabling centralized and reusable styling.
4. Utilize Inline CSS: Apply inline CSS directly to specific HTML elements for targeted styling.
5. Styling the Content: Define and customize CSS rules to enhance the appearance, layout, and interactivity of the HTML content. Employ selectors and properties to style elements and create visual effects.
6. Testing and Verification: Ensure the proper application of each CSS method and verify that the styling is correctly reflected in the HTML content.

## Program

### index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Namma Sangeetha</title>
<link rel="stylesheet" href="home.css">
<style>
```

```
  .logo {
  display: flex;
  align-items: center;
  height: 10px;
}
.logo img {
  height: 70px;
  width: 100px;
  margin-top: -10px;/* Adjust as needed */
}
</style>
</head>
<body>

<div class="header" style=" display: flex;
justify-content: space-between;
align-items: center;
padding: 3px;
background-color: #fff;
box-shadow: 0 2px 4px rgba(0,0,0,0.1);">
  <div class="logo">
    <img src="download.jpg" alt="Logo"> <!-- Replace with your logo URL -->
  </div>
  <h1>The Grill</h1>
  <div class="menu-right">
    <a href="special.html">
      <button class="header-button">Our Specials</button>
    </a>
    <a href="quiz.html">
      <button class="header-button" >WinPrizes</button>
    </a>
    <a href="validate.html">
      <button class="header-button" >register</button>
    </a>
    <a href="favourite.html">
      <button class="header-button" >favourites</button>
    </a>

  </div>
</div>

<div class="card-container">
  <!-- Replace with your actual content -->
```

```
  <div class="card">
    <h2>Farm to Table Freshness</h2>
    <p>Indulge in our seasonal menu, crafted from the freshest local produce, picked from farm to your plate for unparalleled taste and quality.</p>
  </div>
  <div class="card">
    <h2>Signature Cocktails</h2>
    <p>Sip on our unique blend of handcrafted cocktails, each with a twist of local flavors and the finest spirits, tailored to complement your meal</p>
  </div>
  <div class="card">
    <h2>Private Dining Experience</h2>
    <p>Celebrate your special moments in our exclusive private dining area, offering a bespoke service and an intimate atmosphere for you and your guests.</p>
  </div>
  <div class="card">
    <h2>Culinary Masterclass</h2>
    <p>Join our chefs in a journey of culinary discovery with our interactive cooking classes, where you can learn, taste, and enjoy the art of fine cooking.</p>
  </div>



  <!-- Repeat for as many cards as needed -->
</div>

<a href="cookie.html">
  <button class="header-button" >cookies</button>
</a>

<a href="hidden.html">
  <button class="header-button" >hidden</button>
</a>
<a href="p9_url.html">
  <button class="header-button" >url</button>
</a>

<a href="p9_url.html">
  <button class="header-button" >session</button>
</a>
```

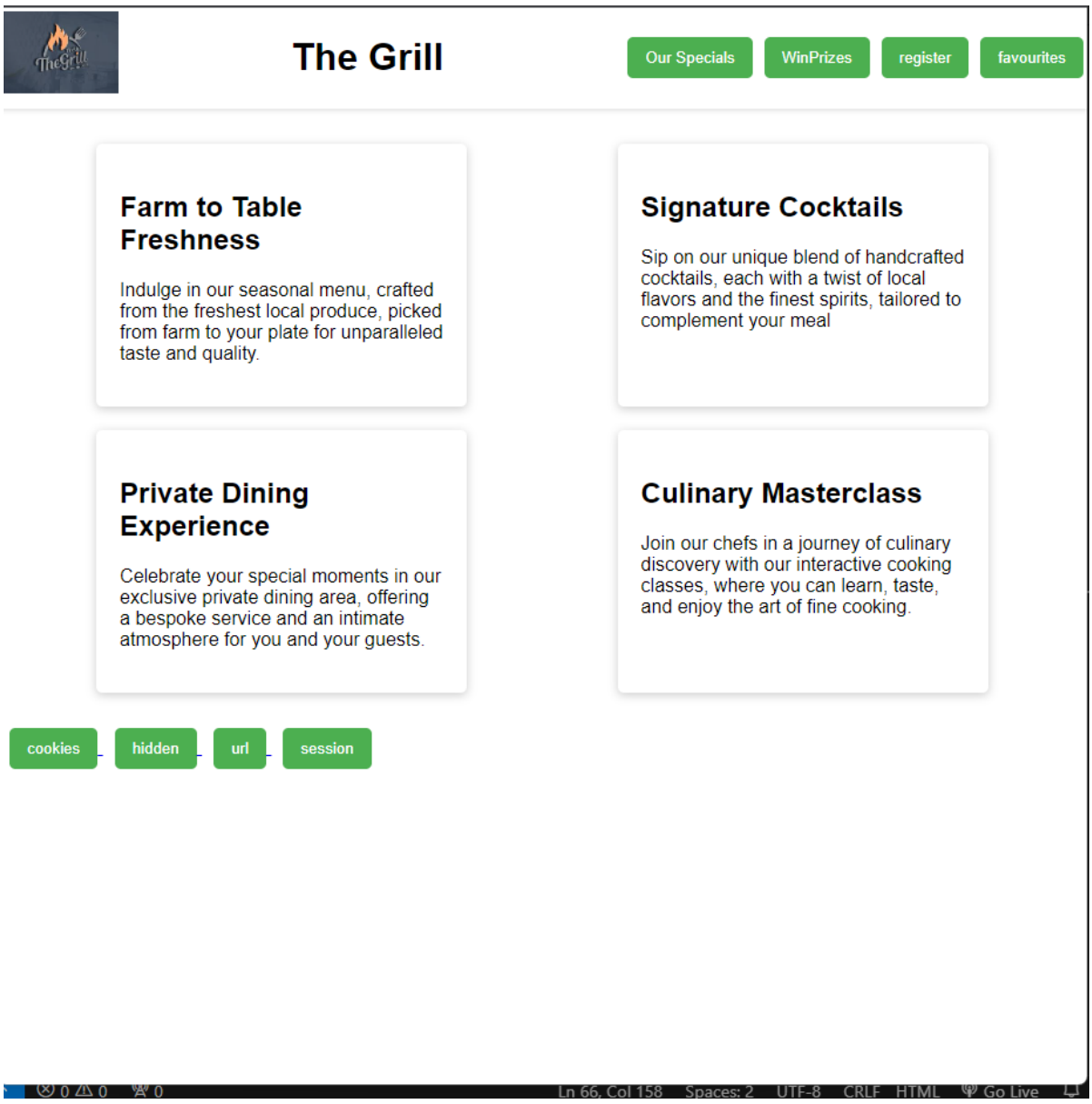
```
 </body>
</html>
```

style.css:

```
body {
  font-family: 'Arial', sans-serif;
  margin: 0;
  padding: 0;
}
.menu-right {
  display: flex;
  align-items: center;
}
.rating {
  margin-right: 20px;
}
.header-button {
  background-color: #4CAF50;
  color: white;
  padding: 10px 15px;
  border: none;
  cursor: pointer;
  border-radius: 5px;
  margin-left: 10px;
}
.card-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
  padding: 20px;
}
```

```
.card {
  box-shadow: 0 2px 8px rgba(0,0,0,0.2);
  margin: 10px;
  width: calc(33.333% - 20px); /* Three cards per row */
  padding: 20px;
  background-color: #fff;
  border-radius: 5px;
}
.hi{
  background-color: aqua;
  border: 10px;
}
@media (max-width: 768px) {
  .card {
    width: calc(50% - 20px); /* Two cards per row on smaller screens */
  }
}
@media (max-width: 480px) {
  .card {
    width: 100%; /* One card per row on mobile screens */
  }
```

## Output




## Result

The implementation of CSS was successful. The HTML content was styled using the three methods of CSS—internal, external, and inline.

**Ex No. : 3** **Javascript Form Validation**
**Date :**

## Aim

The primary goal of this exercise is to understand and implement JavaScript-based form validation. This involves leveraging JavaScript to ensure the accuracy and integrity of user inputs within a web form, in this case, for selecting BMW-related data. The objective is to provide real-time validation and fetch specific BMW data based on the user's selection.

## Procedure

1. Declare the HTML5 doctype.
2. Open the HTML tag with the appropriate language attribute.
3. Create the head and body sections.
4. Inside the head section, add meta tags for the character set and viewport settings.
5. Write a JavaScript function (`validateForm`) for form validation:
   - Check if the email is in a valid format.
   - Ensure the password is not empty.
   - Display appropriate alerts for validation results.
   - Return `true` if the form is valid (for submission).
6. Add a style section within the head to define the CSS.
   - Style the body font, form width, input fields, button, and container.
7. Inside the body, create a container `<div>` to center the form.
8. Inside the container, create a form with a `name` attribute, `onsubmit` event, and `method` set to "post".
9. Create input fields for email and password.
10. Use the `<label>` element for better accessibility.
11. Add a submit button.
12. Test the form in various browsers to ensure compatibility.
13. Ensure the form is responsive on different devices.
14. Refine the code based on testing results.

## Program

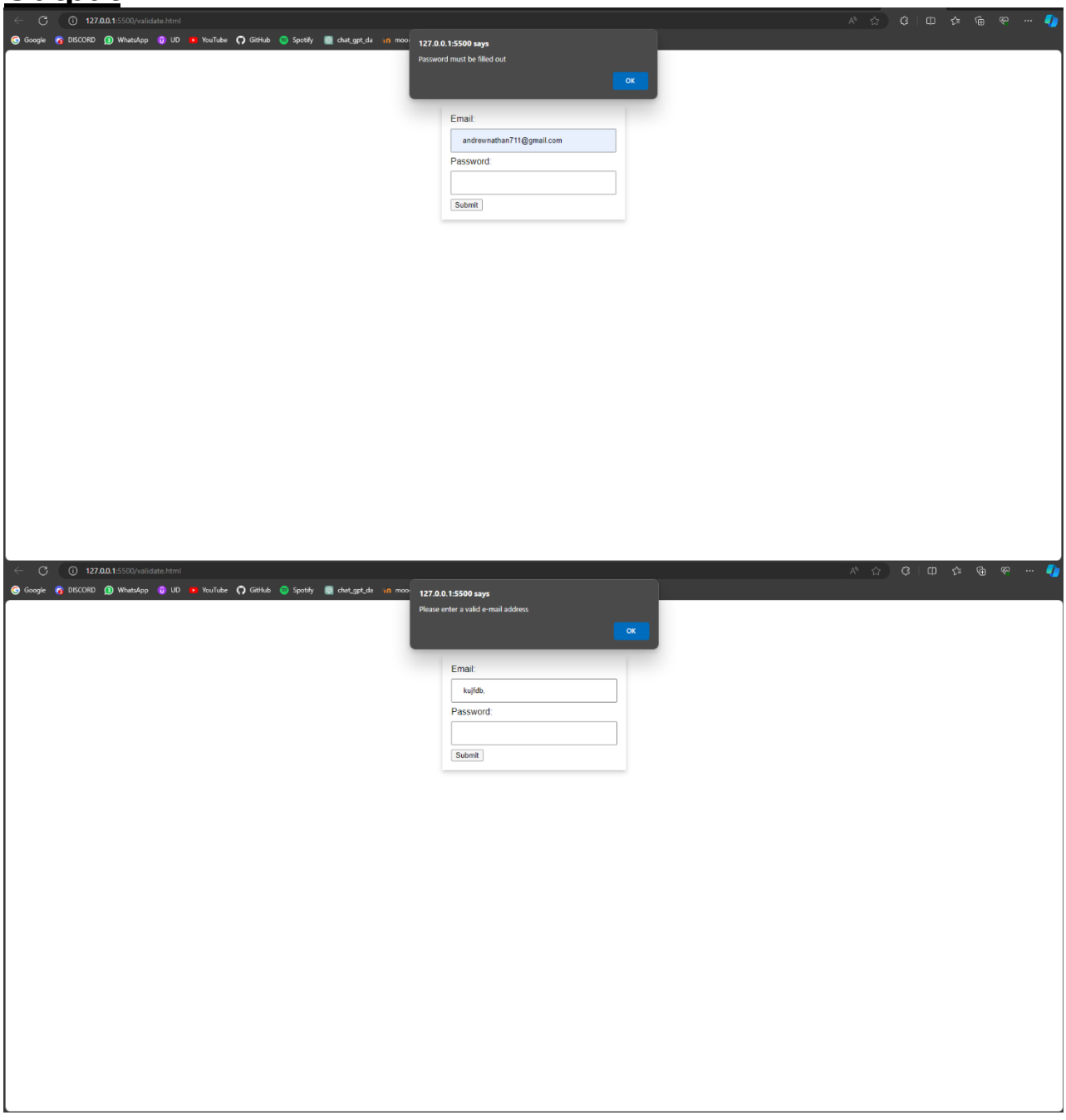
main.html

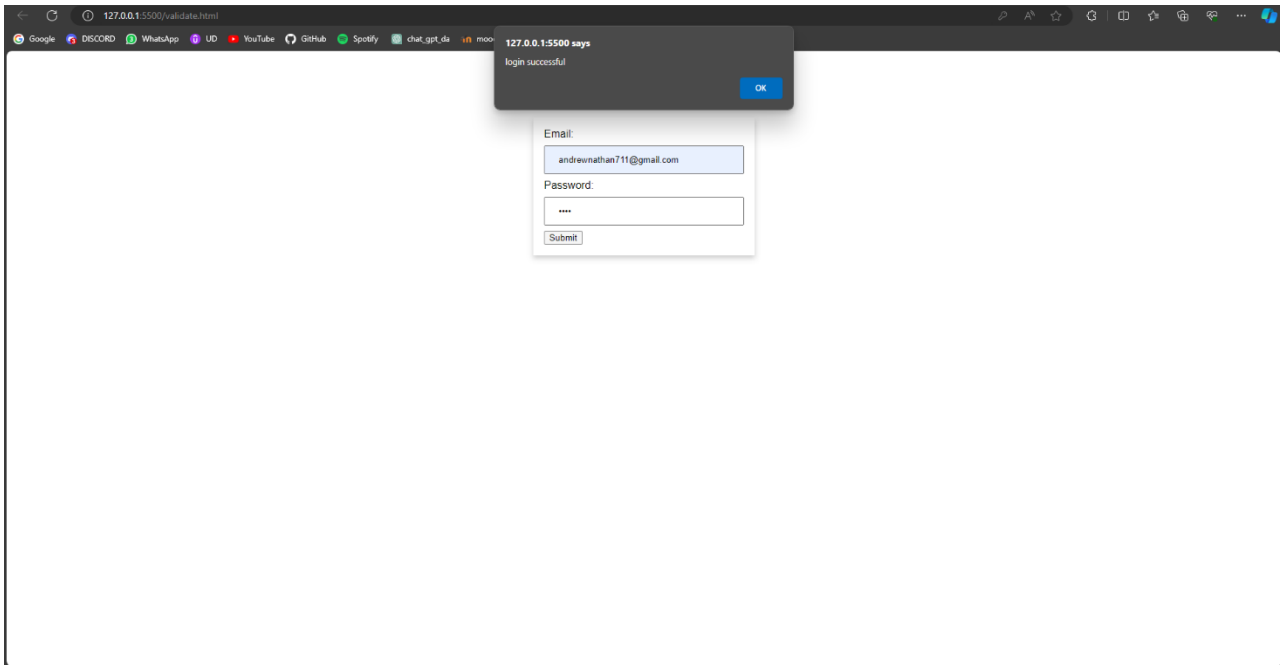
```
<!DOCTYPE html>
<html>
<head>
<script>
    function validateForm() {
```

```
        var x = document.forms["myForm"]["email"].value;
        var y = document.forms["myForm"]["password"].value;
        var atposition=x.indexOf("@");
        var dotposition=x.lastIndexOf(".");
        if (atposition==-1)
        {
            alert("Please enter a valid e-mail address");
            return false;
        }
        if (y == "") {
            alert("Password must be filled out");
            return false;
        }
        alert("login successful");
    }
</script>
<style>
  body {
    font-family: Arial, sans-serif;
}

form {
    width: 300px;
    margin: 0 auto;
}

input[type="text"], input[type="password"] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
}
```

```
button:hover {
    opacity: 0.8;
}

.container {
    padding: 16px;
    background-color: white;
    max-width: 300px;
    margin: 0 auto;
    margin-top: 100px;
    box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
}
</style>
</head>
<body>
  <div class="container">
      <form name="myForm"  onsubmit="return validateForm()"
method="post">
          Email: <input type="text" name="email">
          <br>
          Password: <input type="password" name="password">
          <br>
          <input type="submit" value="Submit">
      </form>
  </div>
</body>
</html>
```

# Output


127.0.0.1:5500/validate.html
Google
DISCORD
WhatsApp
UD
YouTube
GitHub
Spotify
chat_gpt_da
127.0.0.1:5500 says
Password must be filled out
OK
Email:
andrewnathan711@gmail.com
Password:
Submit
127.0.0.1:5500/validate.html
Google
DISCORD
WhatsApp
UD
YouTube
GitHub
Spotify
chat_gpt_da
127.0.0.1:5500 says
Please enter a valid e-mail address
OK
Email:
kujfdb,
Password:
Submit




## Result

The implementation of JavaScript-based form validation was successful. The form was validated in real-time, and the BMW data was fetched based on the user's selection.

**Ex No. : 4** **Installation of Tomcat**
**Date :**

## Aim

The aim of this exercise is to learn about the installation of Tomcat.

## Procedure

Create a folder to keep all your work.

1. Download and install Apache Tomcat separately OR install XAMPP package which comes with both Apache server and Tomcat.
2. Check if Java is installed. If not, install it.
3. Go to Windows environment variables and make sure that both JDK and JRE binaries are added to Path. 5. Configure the Tomcat server.
6. Start Tomcat server. - Note: The Tomcat's executable programs and scripts are kept in the "bin" sub-directory of the Tomcat installed directory. - For Windows, suppose Tomcat is installed in location "C:" (which is the default location in case of Xampp installation). Launch a CMD shell command issue:

```C: [ Change
drive ] cd
startup [ Run startup.bat to start tomcat server ]
```

7. Open a web browser of your choice (Chrome, Firefox, Bing) which acts as an HTTP client. Type the URL "localhost:8080" to view the Tomcat server's welcome page.

- The hostname "localhost" which refers to the IP address 127.0.0.1 is meant for local loop- back testing within the same machine.

8. 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:8080".

9. Try opening “http://localhost:8080/examples” to view the servlet and JSP examples. Try running some of the servlet examples.

10. Shutdown the Server

- For Windows, you can shut down the tomcat server by either:
- Press ‘Ctrl’ + ‘C’ on the Tomcat console.

#### Run "<TOMCAT_HOME>\bin\shutdown.bat" script.

- Open a new cmd terminal and issue: C: [
Change the current drive ] cd
shutdown [ Run shutdown.bat to shutdown the server ]

## Result

Thus, Apache Tomcat web server has been installed successfully.

## Ex No. : 5A Simple Servlet Calling from HTML File
## Date :

## Aim

The goal of this exercise is to comprehend servlets and understand their invocation from an HTML file. The focus is on linking an HTML file to a Java Servlet to display BMW-related data through the servlet upon a user-triggered action.

## Procedure

1. HTML (login.html):
   - Create "login.html" in the text editor.
   - Declare HTML5 doctype, set language to "en".
   - Add head section with charset and viewport settings.
   - Title the page "Login - Namma Sangeetha".
   - Create a container div with class "login-container" to center the form.
   - Inside the container, create a form with action="login" and method="post".
   - Add input fields for username and password, each with a corresponding label.
   - Style the form using CSS for a clean layout.

2. Java Servlet (login.java):
   - Create "login.java" in the text editor.
   - Define a class "login" extending HttpServlet.
   - Implement doPost to handle HTTP POST requests.
   - Obtain HttpSession from the request.
   - Retrieve username and password parameters from the request.
   - Implement a basic check for credentials (e.g., "a" and "b" for demo).
   - If valid, redirect to "home.html" and store username in the session.
   - Ensure proper servlet mapping in web.xml.

3. web.xml Configuration:
   - Create "web.xml" in the WEB-INF directory.
   - Configure servlet with name, class, and mapping.
  - Map servlet to a URL pattern, like "/login".

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login - Namma Sangeetha</title>
<style>
  .login-container {
    width: 300px;
    margin: 0 auto;
    padding: 20px;
  }
  input[type="text"], input[type="password"] {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ddd;
    box-sizing: border-box;
  }
  input[type="submit"] {
    width: 100%;
    padding: 10px;
    border: none;
    background-color: #4CAF50;
    color: white;
    cursor: pointer;
  }
</style>
</head>
<body>

<div class="login-container">
  <form action="login" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
```

```
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>

    <input type="submit" value="Login">
  </form>
 </div>

 </body>
 </html>
```

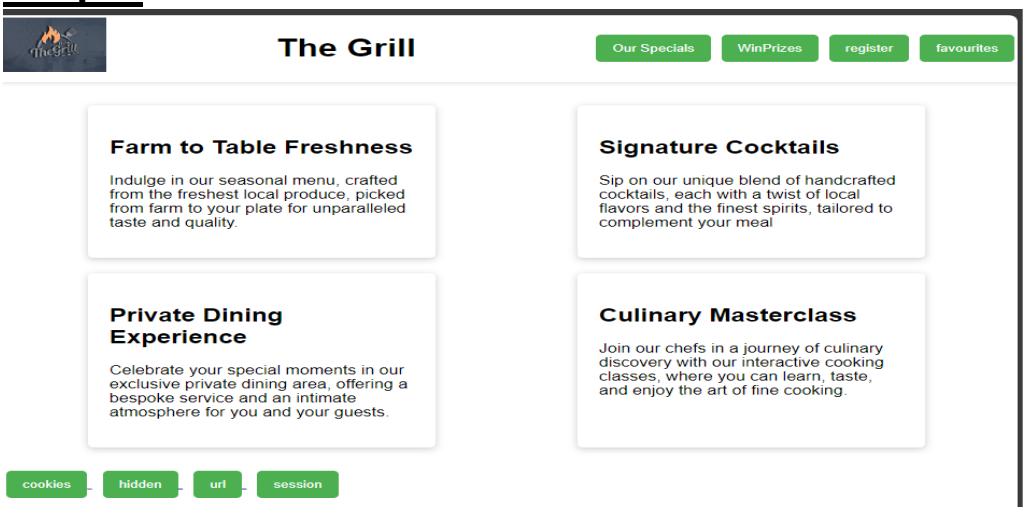
## Servlet.java

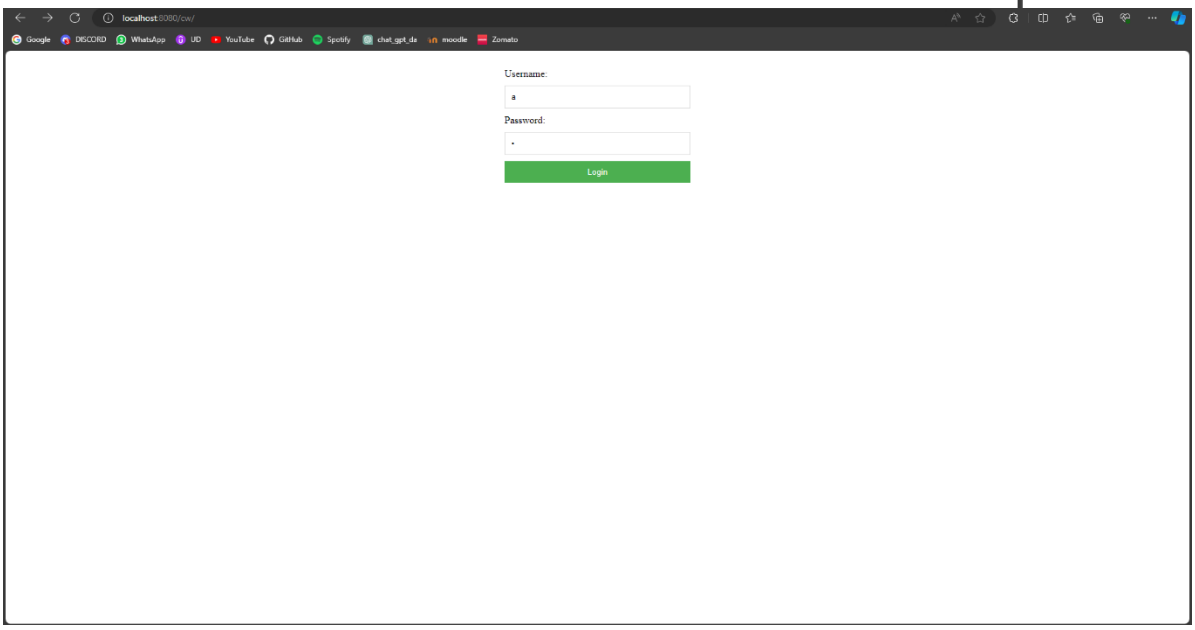
```
 import java.io.*;
 import javax.servlet.ServletException;
 import javax.servlet.http.*;

 public class login extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse
 response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        if ("a".equals(username) && "b".equals(password)) {
            response.sendRedirect("home.html");

            // Store the username as an attribute in the session
            session.setAttribute("username", username);
        }
    }
}
```

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>LoginServlet</servlet-name>
        <servlet-class>login</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

</web-app>
```

# Output




## Result

Thus the simple servlets are executed using Tomcat and Xampp.

**Ex No. : 5B Servlet Basics Form printing using Servlets**
**Date :**

## Aim

The aim here is to gain a comprehensive understanding of servlets and how they interact with HTML forms. The objective is to use a Java Servlet to retrieve and print user- submitted data from an HTML form onto a web page.

## Procedure

1. **HTML (index.html):**
   - Create an HTML file (e.g., "index.html") in your text editor.
   - Define the document type and open the HTML tag.
   - Include a head section with the title "Food Preference."
   - Create a form with the name "myForm" that submits to "food" using the POST method.
   - Add three questions about favorite cuisine, food, and drink with corresponding text input fields.
   - Include a submit button.

2. **Java Servlet (favour.java):**
   - Create a Java file (e.g., "favour.java") in your text editor.
   - Define a class "favour" extending HttpServlet.
   - Implement the `doPost` method to handle HTTP POST requests.
   - Set the response content type to HTML.
   - Retrieve parameters for favorite cuisine, food, and drink from the request.
   - Use PrintWriter to write an HTML response displaying the user's food preferences.

3. **web.xml Configuration:**
   - Open or create the "web.xml" file in the WEB-INF directory.
   - Configure the servlet with a name ("Hellos") and class ("favour").
   - Map the servlet to the URL pattern "/food".

## Program

### index.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Food Preference</title>
</head>
<body>
   <form name="myForm" action="food" method="post">
      <p>Question 1: What is your favorite cuisine?</p>
      <input type="text" name="cuisine" required>
      <br><br>
      <p>Question 2: What is your favorite food?</p>
      <input type="text" name="food" required>
      <br><br>
      <p>Question 3: What is your favorite drink?</p>
      <input type="text" name="drink" required>
      <br><br>
      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

### servlet.java

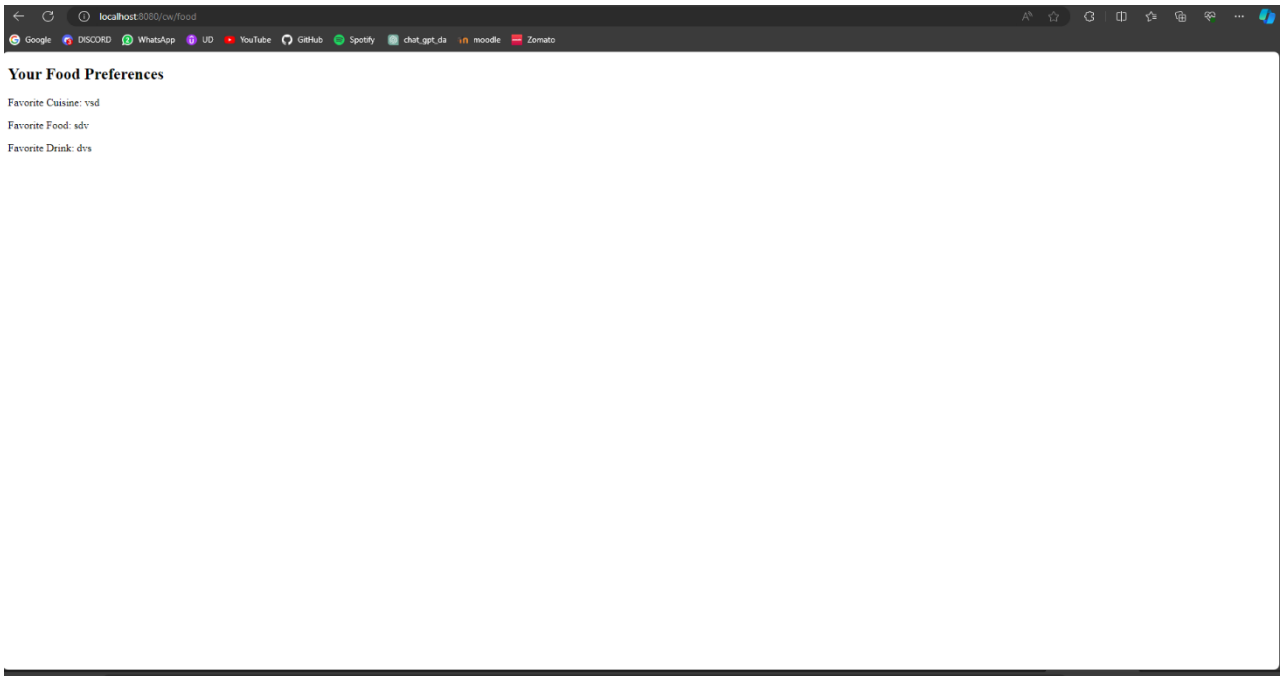
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class favour extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // Set the content type to HTML as the output will be rendered in a
browser.
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Get the parameters from the request.
        String favoriteCuisine = request.getParameter("cuisine");
        String favoriteFood = request.getParameter("food");
        String favoriteDrink = request.getParameter("drink");
```

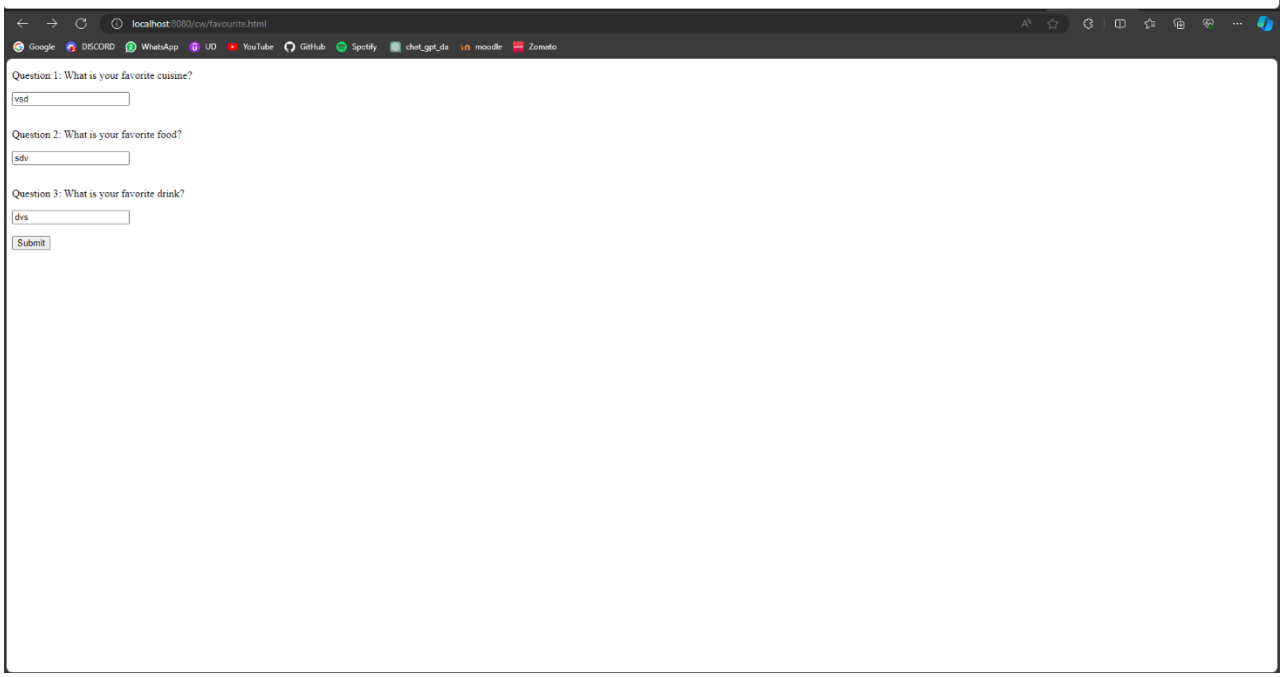
```
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Food Preference</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h2>Your Food Preferences</h2>");
        out.println("<p>Favorite Cuisine: " + favoriteCuisine + "</p>");
        out.println("<p>Favorite Food: " + favoriteFood + "</p>");
        out.println("<p>Favorite Drink: " + favoriteDrink + "</p>");
        out.println("</body>");
        out.println("</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>FoodServlet</servlet-name>
        <servlet-class>favour</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FoodServlet</servlet-name>
        <url-pattern>/food</url-pattern>
    </servlet-mapping>

</web-app>
```

# Output


localhost:8080/cw/food
Google DISCORD WhatsApp UD YouTube GitHub Spotify chat_gpt_da moodle Zomato
Your Food Preferences
Favorite Cuisine: vsd
Favorite Food: sdv
Favorite Drink: dvs


localhost:8080/cw/favourite.html
Google DISCORD WhatsApp UD YouTube GitHub Spotify chat_gpt_da moodle Zomato
Question 1: What is your favorite cuisine?
vsd
Question 2: What is your favorite food?
sdv
Question 3: What is your favorite drink?
dvs
Submit

## Result

The implementation of servlets was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. : 6A** **Session Management - Cookies**
**Date :**

## Aim

To learn how to utilize cookies for session management in a web application.

## Procedure

1. HTML Page (index.html):
   - Create an HTML file named "index.html" in the text editor.
   - Develop a form that includes a text input for the user's favorite cookie.
   - Set the form action attribute to "cookieC1" and the method attribute to "POST".
   - Add a submit button to send the form data to the server.

2. Java Servlet (cook.java):
   - Create a Java file named "cook.java" in the text editor.
   - Define a class "cook" that extends HttpServlet.
   - Implement the `doPost` method to handle HTTP POST requests.
   - Retrieve the user's favorite cookie from the request parameters.
   - Create a new Cookie named "Ucookie" and set its value to the user's input.
   - Add the cookie to the response.
   - Generate an HTML response displaying the user's cookie preference.

3. Web.xml Configuration (web.xml):
   - Open or create the "web.xml" file in the WEB-INF directory using the text editor.
   - Configure the servlet with a servlet-name ("Hel") and servlet-class ("cook").
   - Map the servlet to the URL pattern "/cookieC1".

## Program

### index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="cookieC1" method="POST">
        <p>Your favourite cookie: <input type="text" name="uname" /></p>
        <input type="submit" value="Login" />
    </form>
</body>
</html>
```

### Cookie.java

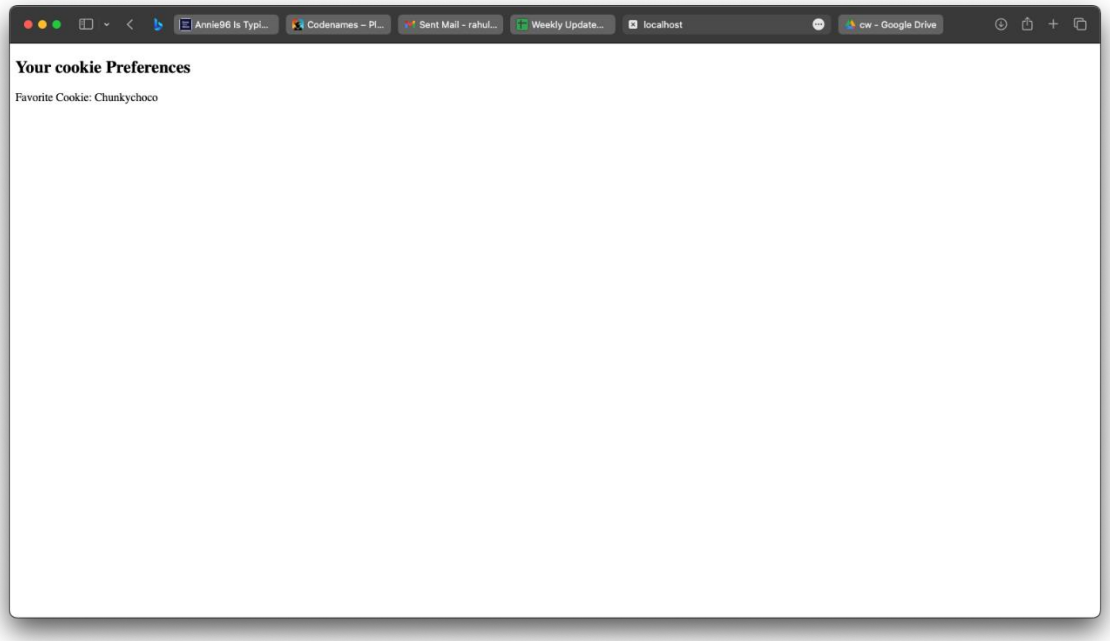
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class cook extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String in_data = request.getParameter("uname");
            Cookie c = new Cookie("Ucookie", in_data);
            response.addCookie(c);
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Cookie Preference</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h2>Your Cookie Preferences</h2>");
```

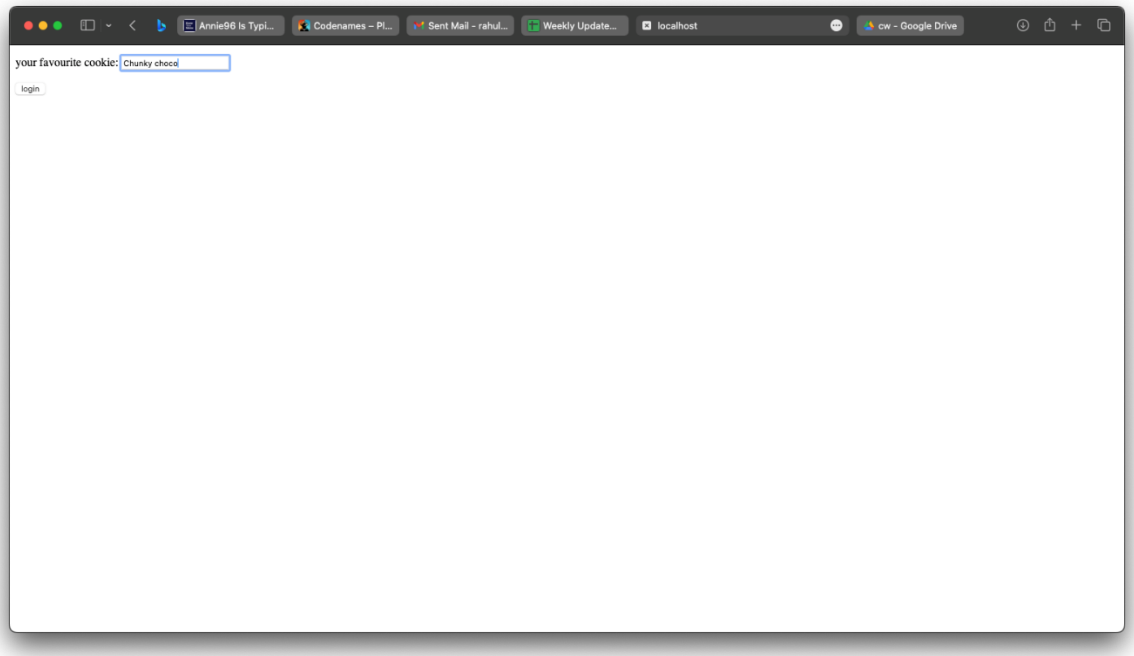
```
            out.println("<p>Favorite Cookie: " + in_data + "</p>");
            out.println("</body>");
            out.println("</html>");
            out.close();
        } catch (Exception exp) {
            System.out.println(exp);
        }
    }
}
```

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>Hel</servlet-name>
        <servlet-class>cook</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Hel</servlet-name>
        <url-pattern>/cookieC1</url-pattern>
    </servlet-mapping>

</web-app>
```

## Output




## Result

The implementation of cookies was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. :** 6B **Session Management - Hidden Form Fields**
**Date** :

## Aim

To understand and implement hidden form fields for session management in web applications.

## Procedure

1. Java Project Setup: Create a Java project.
2. Add a Servlet: Incorporate a servlet to manage sessions using hidden form fields.
3. Web.xml Configuration: Configure servlet mappings in the web.xml file.
4. HTML Form for Data Entry: Create an HTML form with hidden fields to store session data.
5. Processing Hidden Fields in Servlet: Develop the servlet to process hidden form fields.
6. Display Session Information: Implement a servlet to display session information obtained from hidden fields.
7. Apply Styles with CSS: Utilize CSS to style the application interface.

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hidden Form Fields</title>
</head>
<body>
    <form method="GET" action="hidden1">
        Is your hidden talent cooking: <input type="text"
```

```
 name="username"/><br/>
        <input type="submit" value="Go"/>
    </form>
 </body>
</html>
```

## Hidden1.java

```
 import java.io.*;
 import javax.servlet.*;
 import javax.servlet.http.*;

 public class hidden1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
 response) throws ServletException, IOException {
        try {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            String n = request.getParameter("username");
            out.print("Welcome to the First Page " + n);

            // Creating a form with an invisible textfield
            out.print("<form action='hidden2' method='POST'>");
            out.print("<input type='hidden' name='uname' value='" + n + "'>");
            out.print("<input type='submit' value='Go to the second page'>");
            out.print("</form>");

            out.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
 }
```

## RestoreHidden.java

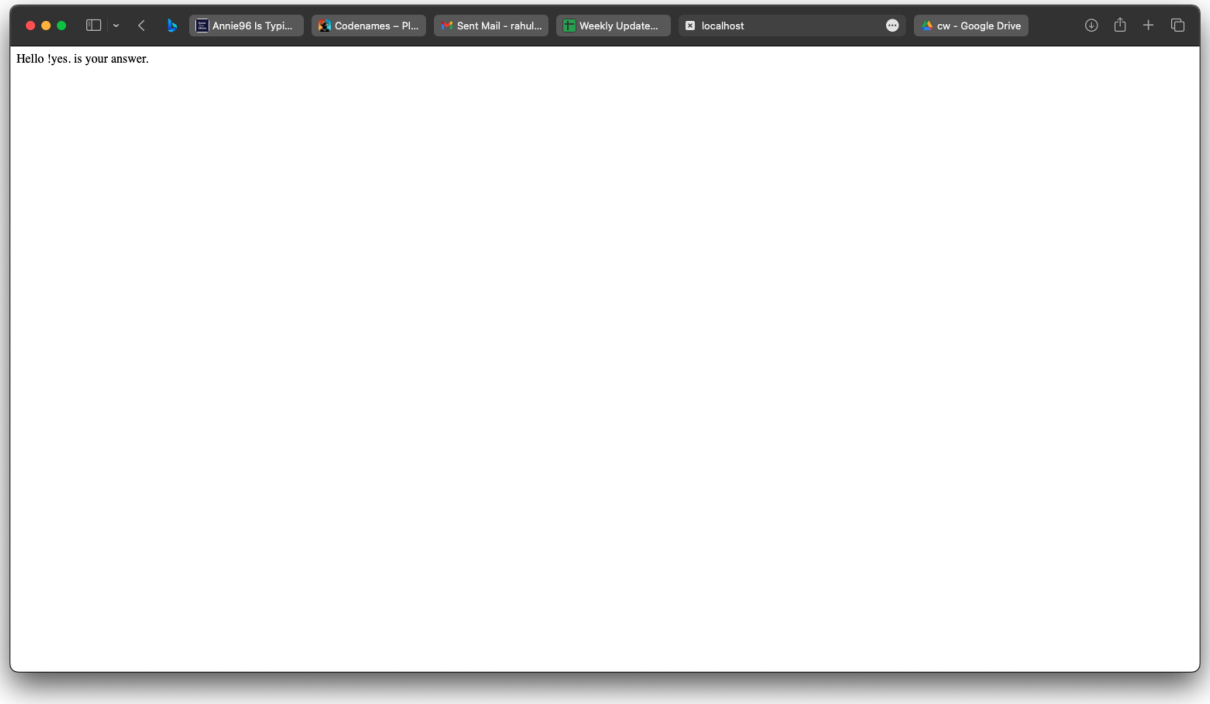
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class hidden2 extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            // Getting the value from the hidden field
            String n = request.getParameter("uname");
            out.print("Hello, " + n + ". Is your answer.");

            out.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

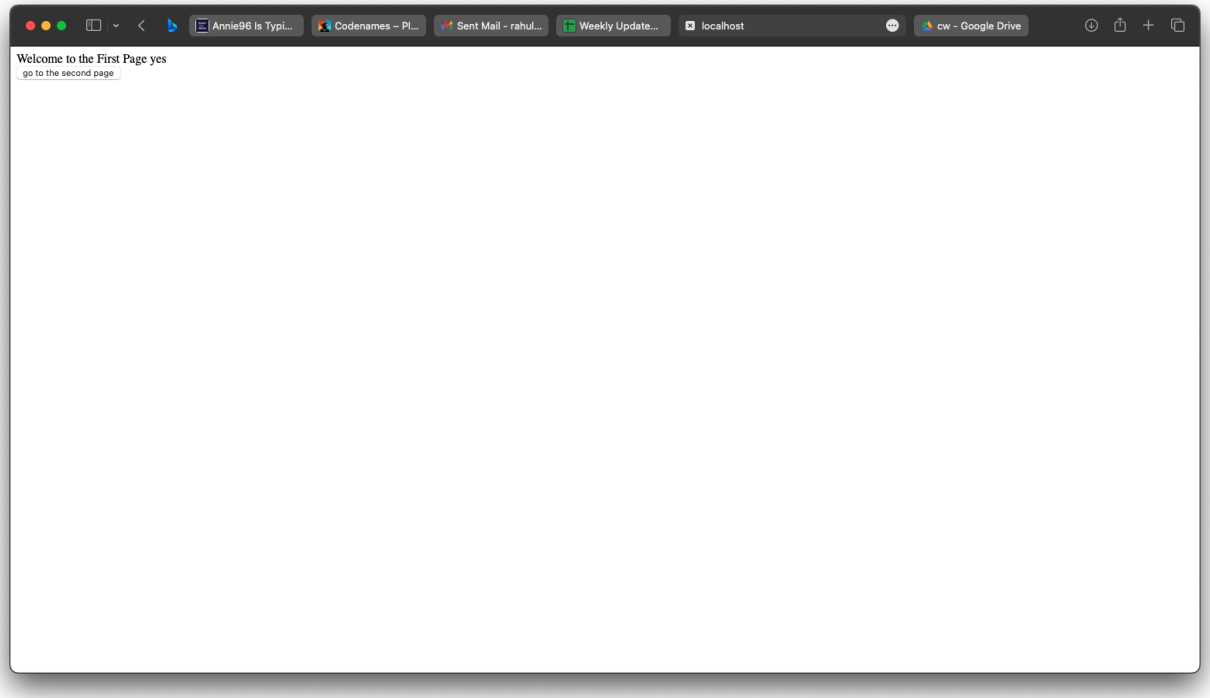
## web.xml

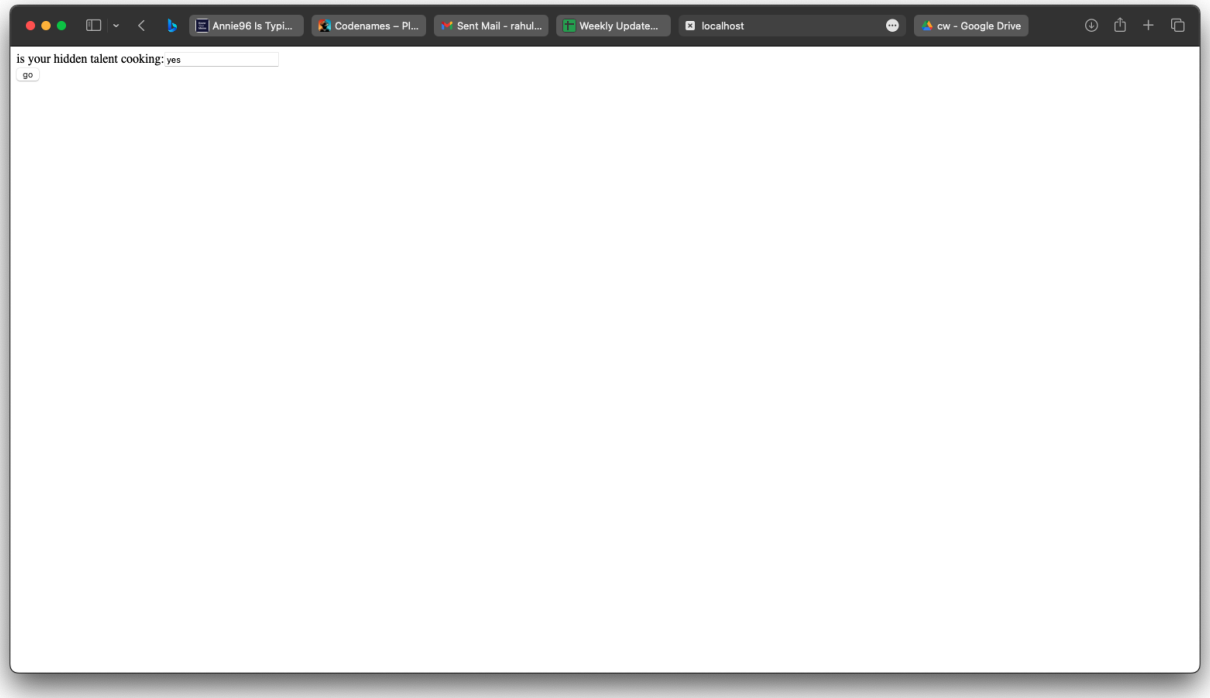
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
        version="4.0">

    <servlet>
        <servlet-name>H1</servlet-name>
        <servlet-class>hidden1</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>H1</servlet-name>
        <url-pattern>/hidden1</url-pattern>
    </servlet-mapping>

    <servlet>
```

```
        <servlet-name>H2</servlet-name>
        <servlet-class>hidden2</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>H2</servlet-name>
        <url-pattern>/hidden2</url-pattern>
    </servlet-mapping>

</web-app>
```

## Output






## Result

The implementation of hidden form fields was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. :** 6C **Session Management - URL Rewriting**
**Date :**

## Aim

To explore URL rewriting as a means of managing sessions in a web application.

## Procedure

1. Java Project Initialization: Begin by creating a Java project.
2. Servlet Integration: Develop a servlet to manage sessions through URL rewriting.
3. Web.xml Configuration: Configure servlet mappings in the web.xml file.
4. HTML Form Design: Create an HTML form to input session data.
5. URL Generation in Servlet: Write servlet logic to generate URLs using session data.
6. Display URL Information: Develop a servlet to display session information extracted from URLs.
7. Enhance UI with CSS: Use CSS to style the application interface for better user experience.

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form method="GET" action="url1">
        Name:<input type="text" name="userName"/><br/>
        <input type="submit" value="go"/>
    </form>
```

```
 </body>
</html>
```

## RestoreURL.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class p9_u2 extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    try{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        //getting value from the query string
        String n=request.getParameter("uname");
        out.print("Hello! Welcome back "+n);
        out.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

## url.java

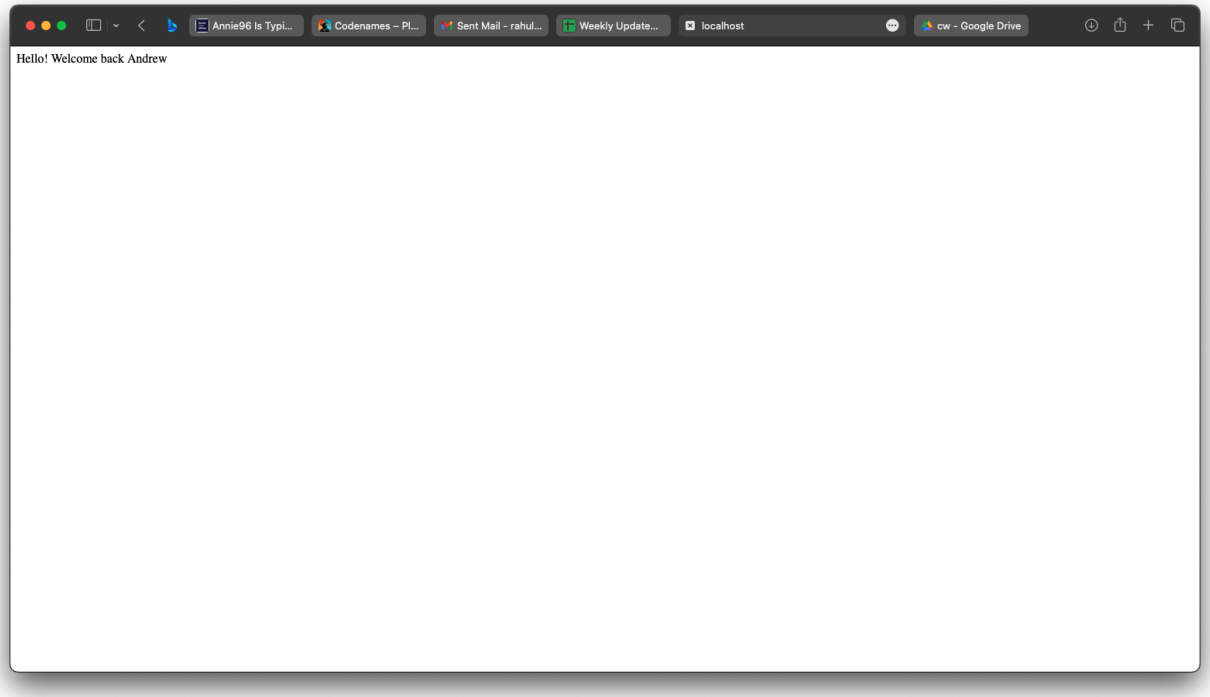
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class p9_u1 extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException{
        try{
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
```

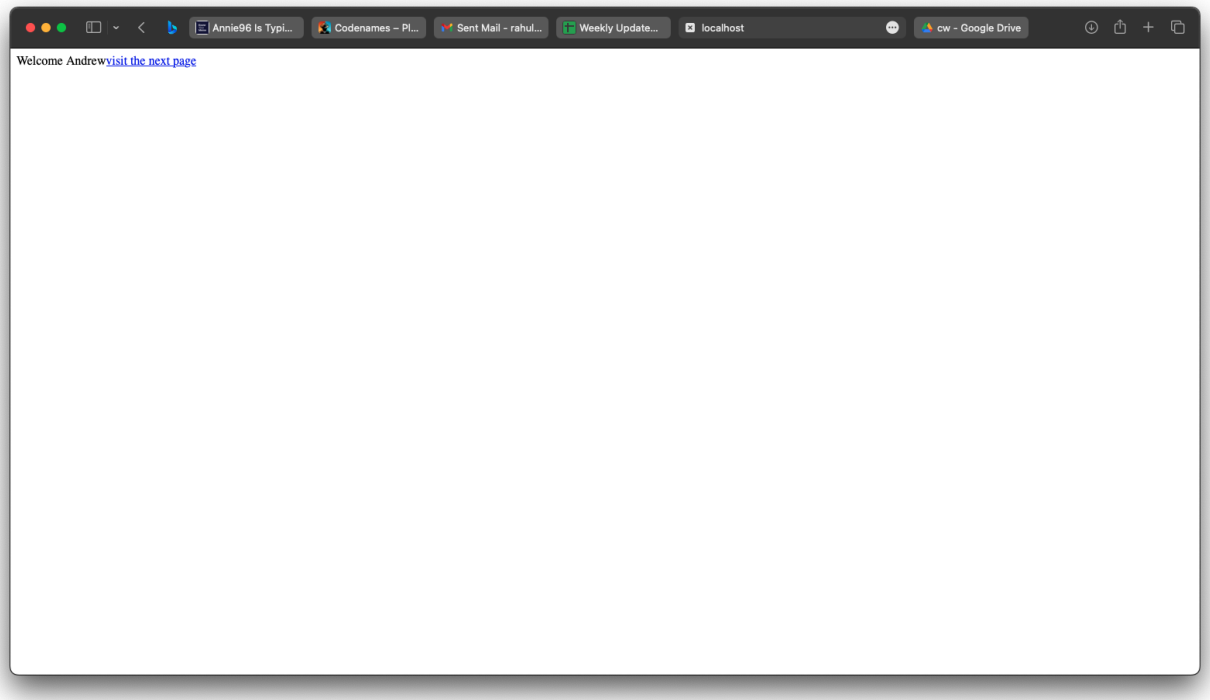
```
            String n=request.getParameter("userName");
            out.print("Welcome "+n);
            //appending the username in the query string
            out.print("<a href='url2?uname="+n+"'>visit the next page</a>");
            out.close();
        }
        catch(Exception e){System.out.println(e);}
    }
}
```

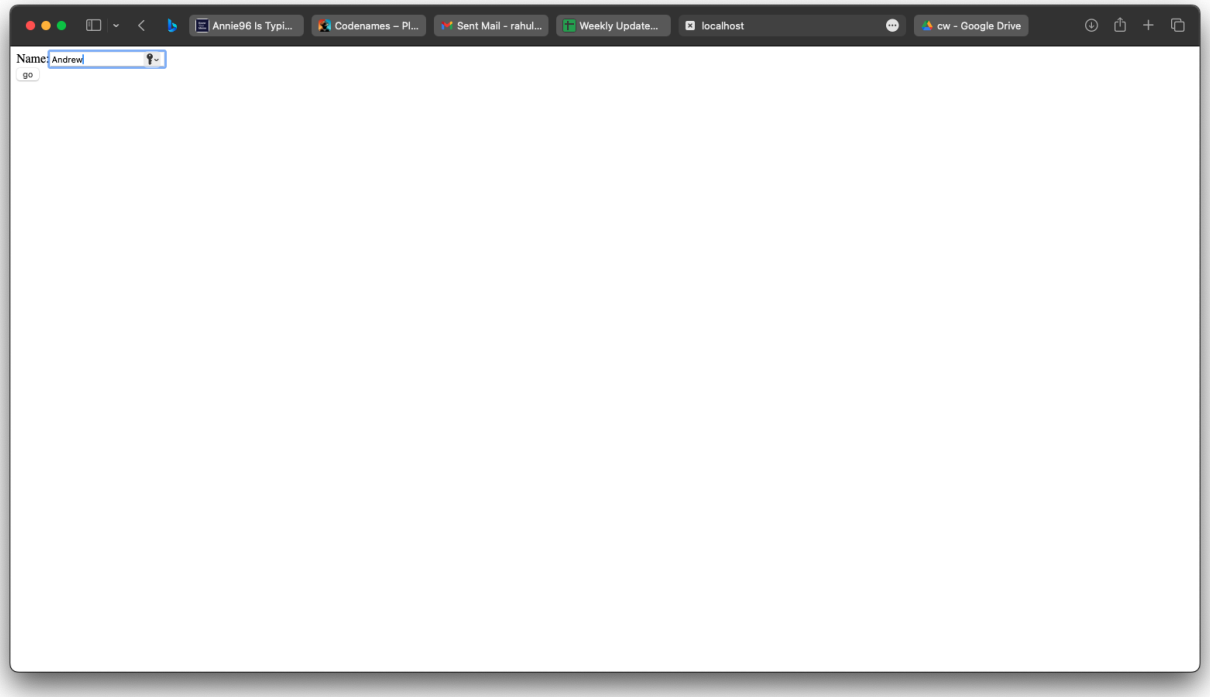
web.xml

```
<servlet>
        <servlet-name>p9_u1</servlet-name>
        <servlet-class>p9_u1</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>p9_u1</servlet-name>
        <url-pattern>/url1</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>p9_u2</servlet-name>
        <servlet-class>p9_u2</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>p9_u2</servlet-name>
        <url-pattern>/url2</url-pattern>
    </servlet-mapping>
```

# Output


Annie96 Is Typi...
Codenames – Pl...
Sent Mail - rahul...
Weekly Update...
localhost
cw - Google Drive
Hello! Welcome back Andrew


Annie96 Is Typi...
Codenames – Pl...
Sent Mail - rahul...
Weekly Update...
localhost
cw - Google Drive
Welcome Andrewvisit the next page




## Result

The implementation of URL rewriting was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. :** 6D **Session Management - HTTP Session Objects**
**Date :**

## Aim

To comprehend and utilize HTTP session objects for session management within a web application.

## Procedure

1. **HTML Page (index.html):**
   - Create an HTML file with a form asking for the user's name.
   - Set the form action to "sessid1" for data submission.

2. **Java Servlet (p10_sessid1.java):**
   - Write a Java file to handle the first servlet.
   - Capture the user's name from the form.
   - Display a welcome message.
   - Store the name in a session attribute ("login_name").
   - Provide a link to "sessid2" for the user to visit the second page.

3. **Java Servlet (p10_sessid2.java):**
   - Create another Java file for the second servlet.
   - Retrieve the user's name from the session.
   - Display a welcome-back message to the user.

4. **Web.xml Configuration (web.xml):**
   - Configure a servlet ("p10_s1") for "p10_sessid1" and map it to "/sessid1".

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="sessid1">
```

```
        Name: <input type="text" name="userName"/><br/>
        <input type="submit" value="Go"/>
    </form>
 </body>
</html>
```

RestoreSession.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class p10_sessid2 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
 response) throws ServletException, IOException {
        try {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            HttpSession session = request.getSession(false);
            String n = (String)session.getAttribute("login_name");

            out.print("Hello!! Welcome Back " + n);
            out.close();
        } catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Session1.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class p10_sessid1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
```

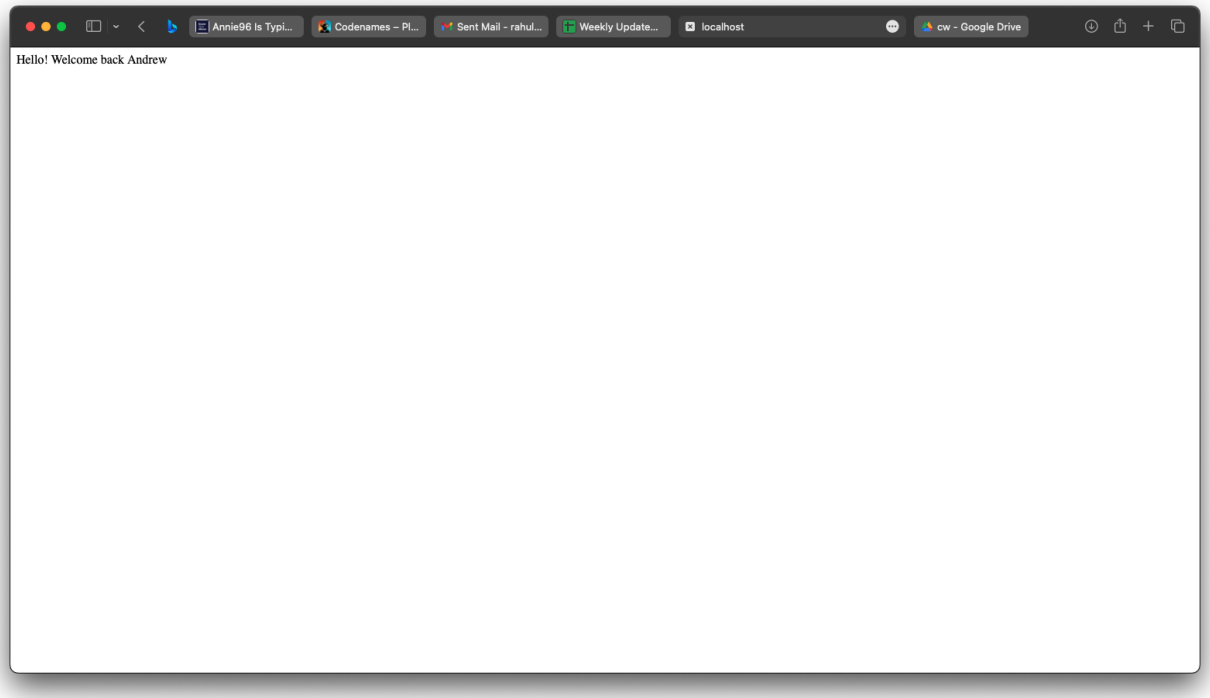
```
      response) throws ServletException, IOException {
       try {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          String n = request.getParameter("userName");
          out.print("Welcome " + n);

          HttpSession session = request.getSession();
          session.setAttribute("login_name", n);

          out.print("<a href='sessid2'>Visit</a>");
          out.close();
       } catch(Exception e) {
          System.out.println(e);
       }
   }
}
```

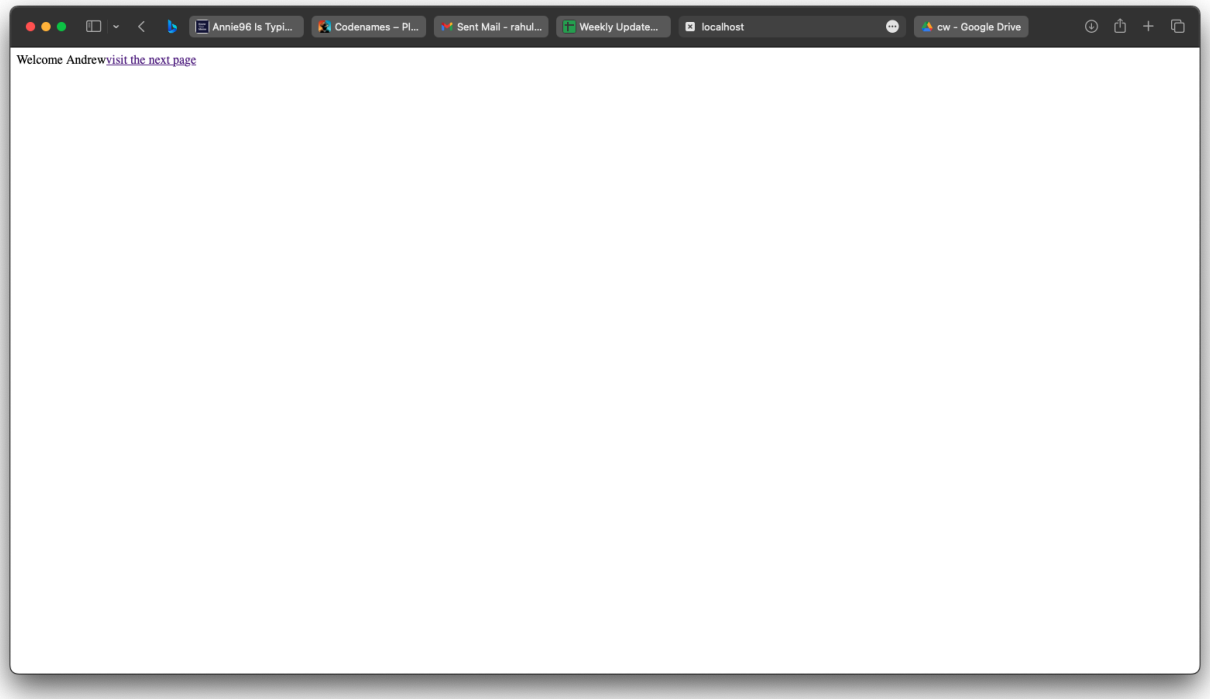
web.xml

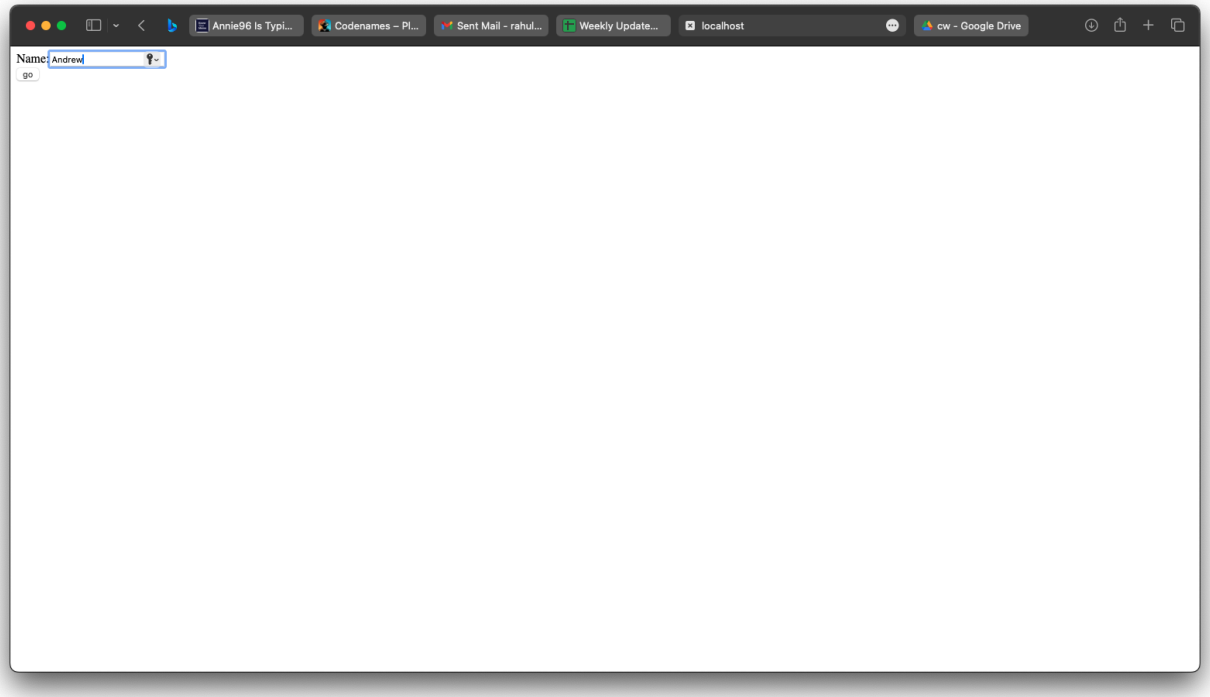
```
<servlet>
   <servlet-name>p10_s1</servlet-name>
   <servlet-class>p10_sessid2</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>p10_s1</servlet-name>
   <url-pattern>/sessid2</url-pattern>
</servlet-mapping>
<servlet>
   <servlet-name>p10_s2</servlet-name>
   <servlet-class>p10_sessid2</servlet-class>
</servlet>
<servlet-mapping>
```

```
    <servlet-name>p10_s2</servlet-name>
    <url-pattern>/sessid2</url-pattern>
</servlet-mapping>
```

## Output






## Result

The implementation of HTTP session objects was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. :** 7A **Database Management using Servlets**
**Date :**

## Aim

The aim of this exercise is to comprehend database management through servlets.

## Procedure

1. Project Creation and Servlet Addition:
- Begin by creating a Java project and incorporate a servlet.
2. web.xml Configuration:
- Include the servlet in the web.xml file to map the servlet.
3. HTML Form Creation:
- Develop an HTML form that gathers data.
4. Link Servlet to Form:
- Assign the servlet's URL to the form's action attribute.
5. JavaScript for Form Validation:
- Include JavaScript within the tag to validate the form.
6. Validate and Store Form Data:
- In the servlet, retrieve and store form data in a database.
7. Display Database Information:
- Create a response to display the database information to the user.

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css" />
    <title>food Database</title>
</head>
<body>
    <div class="flex-col">
        <h1>Name - food Database</h1>
```

```
        <form class="flex-col" action="storespace" method="post">
            <label for="name">Name</label>
            <input type="text" name="name" id="name" placeholder="Enter
Name" required />
            <label for="food">food</label>
            <input type="text" name="food" id="food" placeholder="Enter Food"
required />
            <input type="submit" />
        </form>
    </div>
</body>
</html>
```

## Access.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class AccessSpace extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            PrintWriter out = response.getWriter();
            out.println("<html><head>");
            out.println("<link rel='stylesheet' href='style.css' /> <title>Access
food</title></head>");
            out.println("<body class='flex-col'><h1>food Database</h1>");
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://172.18.0.2:3306/space","root","
10022004");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from space");
            out.println("<table><tr><th>Name</th><th>food</th></tr>");
            while(rs.next()) {
                out.println("<tr><td>" + rs.getString(1) + "</td><td>" +
rs.getString(2) + "</td></tr>");
            }
            out.println("</table>");
            out.println("</body></html>");
```

```
        } catch (Exception e) {
            System.err.println(e);
        }
    }
}
```

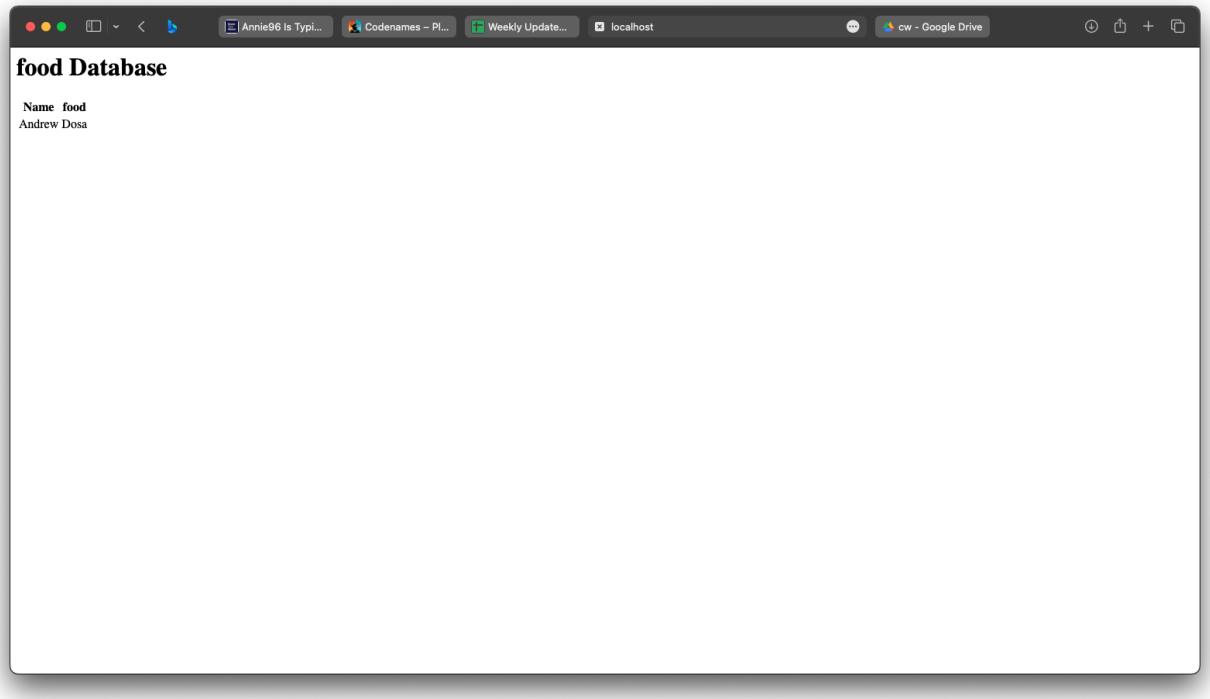
## Store.java

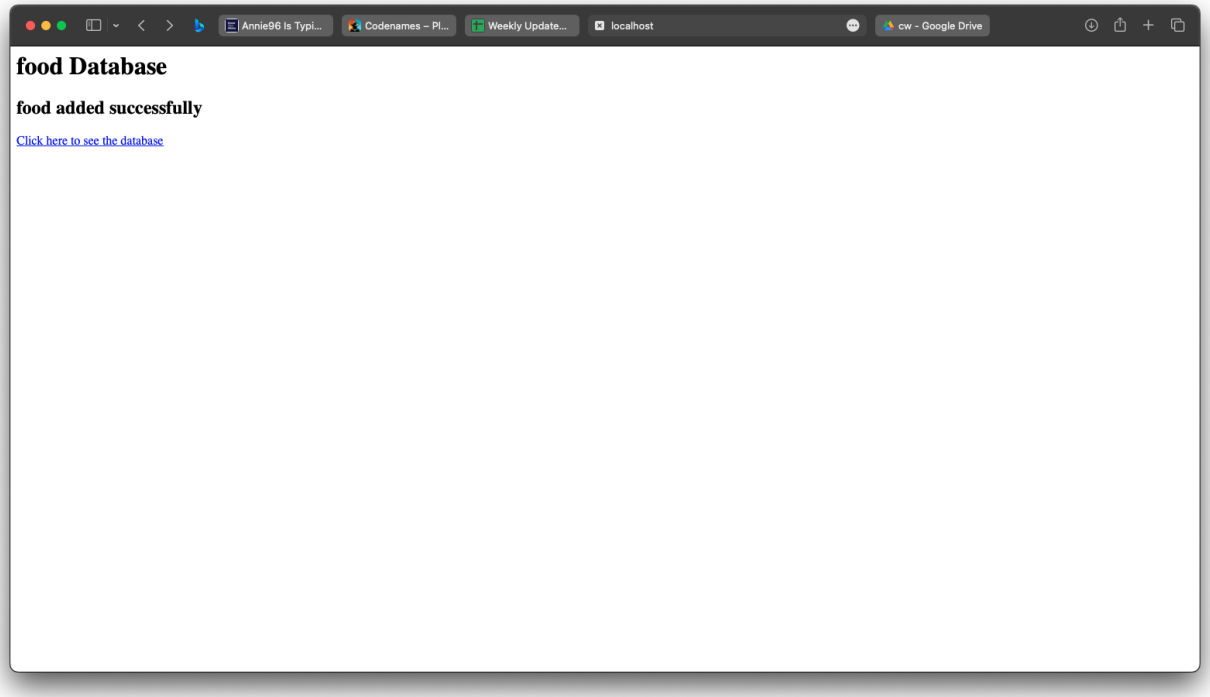
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class StoreSpace extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            String name = request.getParameter("name");
            String food = request.getParameter("food");
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://172.18.0.2:3306/space","root","
10022004");
            Statement stmt = con.createStatement();
            stmt.executeUpdate("insert into space values('" + name + "','" + food
+ "')");
            stmt.close();
            con.close();
            PrintWriter out = response.getWriter();
            out.println("<html><head>");
            out.println("<link rel='stylesheet' href='style.css' /> <title>Store
food</title></head>");
            out.println("<body class='flex-col'><h1>food Database</h1>");
            out.println("<h2>food added successfully</h2>");
            out.println("<a href='accessspace'>Click here to see the
database</a>");
            out.println("</body></html>");
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

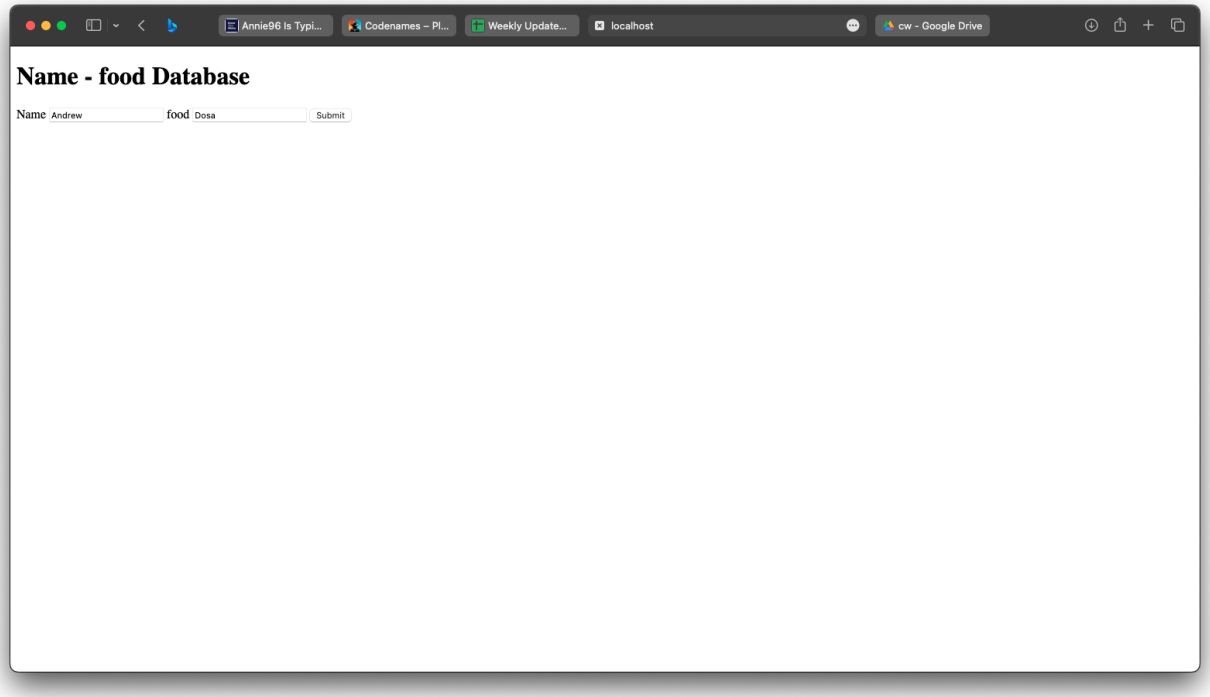
web.xml

```
<web-app>
<servlet>
  <servlet-name>StoreSpace</servlet-name>
  <servlet-class>StoreSpace</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>StoreSpace</servlet-name>
    <url-pattern>/storespace</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>AccessSpace</servlet-name>
  <servlet-class>AccessSpace</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>AccessSpace</servlet-name>
    <url-pattern>/accessspace</url-pattern>
</servlet-mapping>
</web-app>
```

# Output


Annie96 Is Typi...
Codenames – Pl...
Weekly Update...
localhost
cw - Google Drive
food Database
Name food
Andrew Dosa


Annie96 Is Typi...
Codenames – Pl...
Weekly Update...
localhost
cw - Google Drive
food Database
food added successfully
Click here to see the database




## Result

The implementation of database management was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.

**Ex No. :** 7B **Online Exam Evaluation using Servlets**
**Date :**

## Aim

The aim of this exercise is to learn about online exam evaluation utilizing servlets.

## Procedure

1. Project Initialization and Servlet Addition:
   - Initialize a Java project and add a servlet for login verification.
2. web.xml Configuration:
   - Configure the servlet mapping within the web.xml file.
3. HTML Form Design for Exam Login:
   - Create an HTML form for user login before accessing the exam.
4. Link Servlet to Login Form:
   - Link the servlet with the login form using the action attribute.
5. JavaScript for Form Validation:
   - Include JavaScript in the tag to validate the login form.
6. Evaluate Exam and Display Result:
   - Develop a servlet to evaluate the exam based on user responses and display the score.
7. Session Handling:
   - Utilize session management to authenticate and validate users attempting the exam.

## Program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Questions Form</title>
```

```
<style>
  .header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 3px;
    background-color: #fff;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }
  .logo {
    display: flex;
    align-items: center;
    height: 10px;
  }
  .logo img {
    height: 70px;
    width: 100px;
    margin-top: -10px;/* Adjust as needed */
  }
  .menu-right {
    display: flex;
    align-items: center;
  }
  .rating {
    margin-right: 20px;
  }
  .header-button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 15px;
    border: none;
    cursor: pointer;
    border-radius: 5px;
    margin-left: 10px;
  }
  body {
    font-family: Arial, sans-serif;
    padding: 20px;
    background-color: #f4f4f4;
```

```
  }
  .card {
    background-color: #ffffff;
    margin-bottom: 15px;
    margin-top: 15px;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }
  .container {
    max-width: 600px;
    margin-top: 100px;
  }
  input[type="text"] {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 15px;
    border-radius: 4px;
    border: 1px solid #ddd;
    box-sizing: border-box; /* So padding doesn't affect width */
  }
  button {
    background-color: #5cb85c;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }
  button:hover {
    background-color: #4cae4c;
  }
</style>
</head>
<body>
  <div class="header">
    <div class="logo">
      <img src="download.jpg" alt="Logo"> <!-- Replace with your logo URL -->
```

```
    </div>
    <h1>The Grill</h1>
    <div class="menu-right">
      <a href="special.html">
        <button class="header-button">Our Specials</button>
      </a>
      <a href="rating.html">
        <button class="header-button" >WinPrizes</button>
      </a>

    </div>
  </div>

<div class="container">
  <form action="submitAnswers" method="post">
    <!-- Card 1 -->
    <div class="card">
      <p>Question 1: What is the capital of France?</p>
      <input type="text" name="answer1" required>
    </div>
    <!-- Card 2 -->
    <div class="card">
      <p>Question 2: Who wrote 'Romeo and Juliet'?</p>
      <input type="text" name="answer2" required>
    </div>
    <!-- Card 3 -->
    <div class="card">
      <p>Question 3: What is the boiling point of water?</p>
      <input type="text" name="answer3" required>
    </div>
    <!-- Card 4 -->
    <div class="card">
      <p>Question 4: How many continents are there on Earth?</p>
      <input type="text" name="answer4" required>
    </div>
    <!-- Card 5 -->
    <div class="card">
      <p>Question 5: What is the largest ocean on Earth?</p>
      <input type="text" name="answer5" required>
```

```
    </div>
    <!-- Submit button for the form -->
    <button type="submit">Submit Answers</button>
  </form>
</div>

</body>
</html>
```

## LoginExam.java

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

public class quiz extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
    try{
        PrintWriter out = response.getWriter();
        Connection conn=null;
        Statement stmt=null;
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
DriverManager.getConnection("jdbc:mysql://172.18.0.3:3306/college","root",
"10022004");
        stmt = conn.createStatement();
        PreparedStatement pstmt = conn.prepareStatement("insert into user
values(?,?,?,?,?)");
        // Retrieve the answers from the request
        String[] answers = new String[5];
        answers[0] = request.getParameter("answer1");
        answers[1] = request.getParameter("answer2");
        answers[2] = request.getParameter("answer3");
        answers[3] = request.getParameter("answer4");
        answers[4] = request.getParameter("answer5");
```

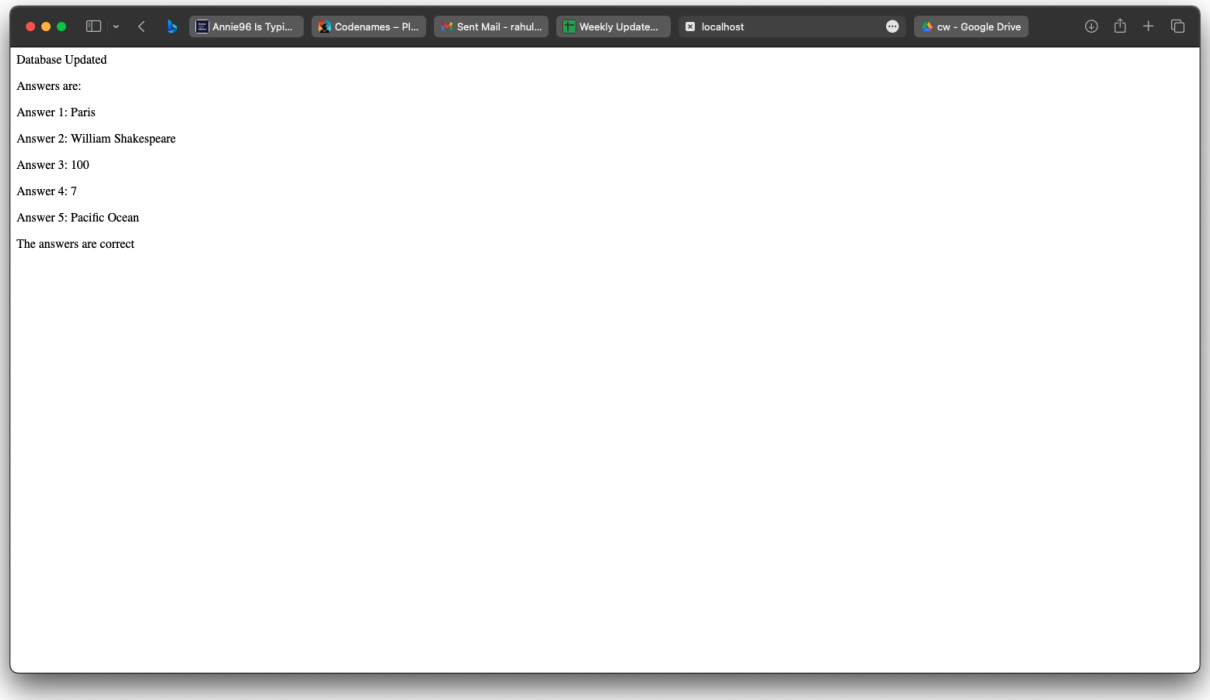
```
        pstmt.setString(1, answers[0]);
        pstmt.setString(2, answers[1]);
        pstmt.setString(3, answers[2]);
        pstmt.setString(4, answers[3]);
        pstmt.setString(5, answers[4]);
        pstmt.executeUpdate();
        out.println("<html><body><p> Database Updated</p>");
        response.setContentType("text/html");

        String sql = "SELECT * FROM user";
        pstmt = conn.prepareStatement(sql);
        //pstmt.setString(1, department);
        ResultSet rs = pstmt.executeQuery();
        // Extract data from result set
        while(rs.next())
        {
        //Retrieve by column name
        String did = rs.getString("answer");


        //Display values
        out.println("<p> answer: " + did + "<br>");

        }
        out.println("</body></html>");
        // Clean-up environment
        rs.close();
        stmt.close();
        conn.close();
    }catch(Exception e){
        System.out.println(e);
    }

    }
}
```

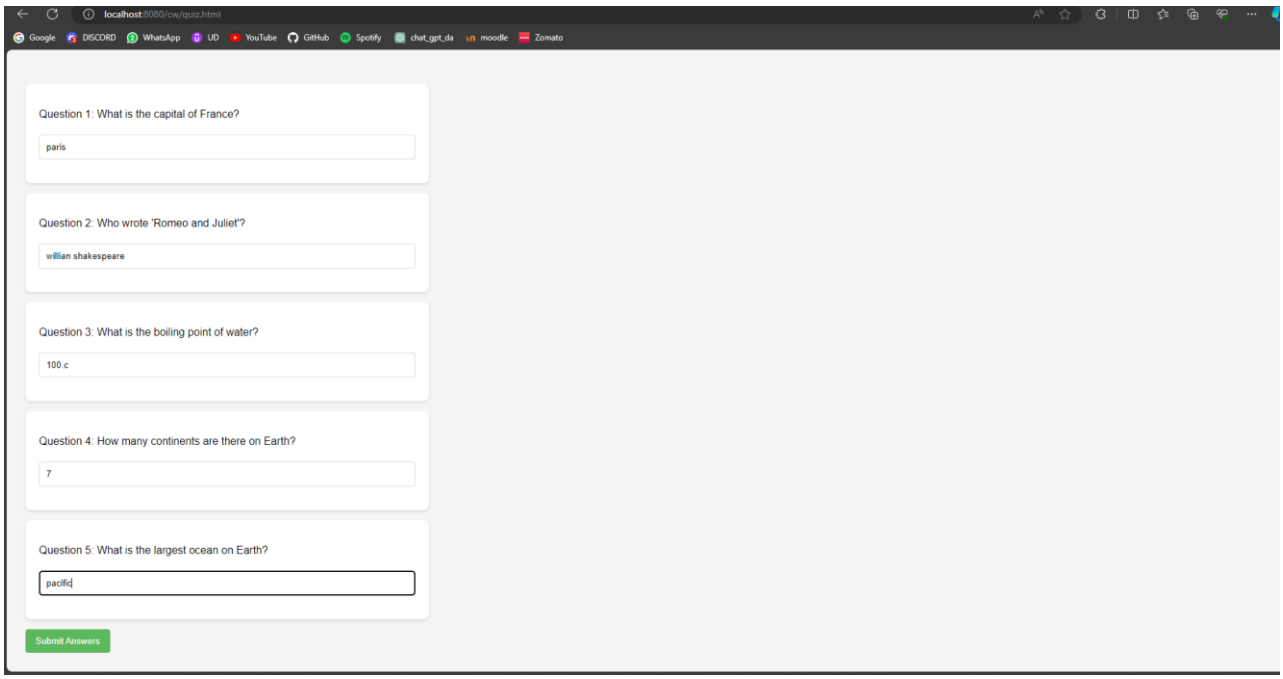
## **Web.xml**

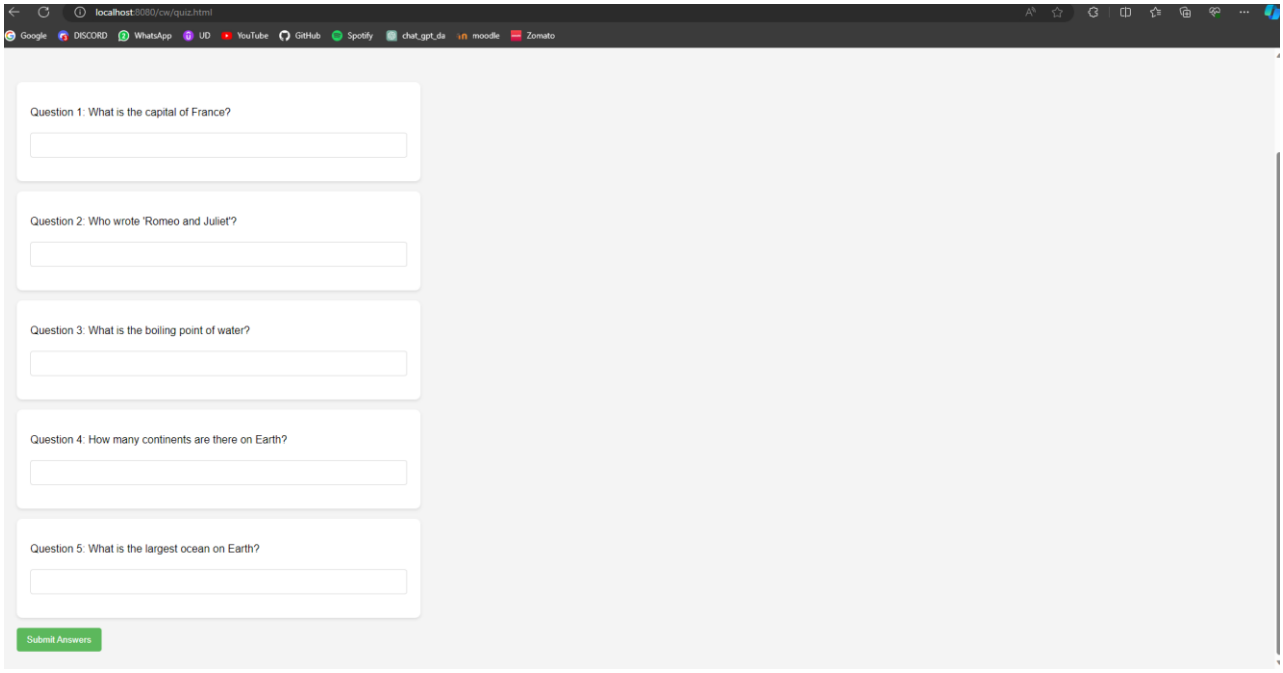
```
 <servlet>
        <servlet-name>p10_s2</servlet-name>
        <servlet-class>quiz</servlet-class>
    </servlet>
```

```
<servlet-mapping>
    <servlet-name>p10_s2</servlet-name>
    <url-pattern >submitAnswers </url-pattern>
</servlet-mapping>
```

## Output






## Result

The implementation of online exam evaluation was successful. The servlet was able to retrieve and print the user-submitted data onto the web page.