Week-1

→ Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex & grid.

Aim :-

Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex & grid.

Description :

HTML, CSS and Javascript are essential components for creating a functional responsive web application. A condensed example of a shopping cart with registration, login, catalogue and cart pages is provided.

Project structures :

1. index.html - Main HTML file containing the structure of the web application.
2. styles.css - CSS file for styling the web pages
3. script.js - Javascript file for handling interactions and logic
4. Images/ - Folder for storing images.

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="styles.css">
   <title> Shopping Cart </title>
 </head>
 <body>
   <header>
   <h1>shopping Cart</h1>
   <nav>
     <ul>
       <li><a href="#catalog">catalog</a></li>
       <li><a href="#cart">cart</a></li>
       <li><a href="#login">Login</a></li>
       <li><a href="#register">Register</a></li>
    </ul>
   </nav>
  </header>
  <main id="content">
   <!-- content will be loaded dynamically using
        JavaScript -->
</main>
  <script src="script.JS"></script>
</body>
</html>
```

styles.css:

```
body
{
front-family :'Arial'; sans-serif; margin:0;
        padding:0;

}
header
{
background-color: #333; color: #fff; padding;
        10px; text-align : center ;

}
nav ul {list-style: none ; padding: 0; display:
        flex ; justify-content : center ; }
nav li {margin: 0 10px ; } main {padding: 20px;}
```

Index html :-

Output :-

Shopping cart

- Catalog
- Cart
- Login
- Register

Week-2

1. Make the above web applications responsive web application using Bootstrap framework.

Aim :- Make the above web application responsive web application using Bootstrap framework.

Description : Bootstrap is a popular CSS framework that makes it easy to create responsive web applications. The previous example can be modified using Bootstrap by following these steps:

Project structure :

1. Index.html - Main HTML file containing the structure of the web application with Bootstrap.
2. script.js - Javascript file for handling interactions & logic (no changes from the previous example)
3. styles.css - You can include additional custom styles if needed.
4. images/ - Folder for storing images.

Index.html :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=
        device-width, initial-scale=1.0">
    <!--Bootstrap CSS-->
    <link href="https://cdn.jsdelivr.net/
      npm/bootstrap@5.3.0/dist/css/bootstrap.
      min.css" rel="stylesheet">
```

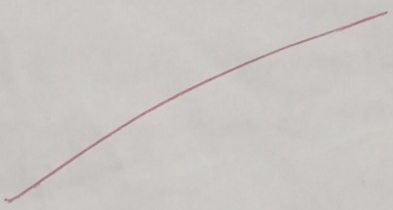
```
    <!-- Custom CSS -->
    <link rel="stylesheet" href="styles.css">
    <title>Shopping Cart</title>
</head>
<body>
    <header class="bg-dark text-white text-
                center py-3">
        <h1>Shopping Cart</h1>
        <nav>
            <ul class="nav justify-content-center">
                <li class="nav-item"><a class=
                    "nav-link" href="#catalog">Catalog
                    </a></li>
                <li class="nav-item"><a class="nav-
                    link" href="#cart">Cart</a></li>
                <li class="nav-item"><a class="nav-
                    link" href="#login">Login</a></li>
                <li class="nav-item"><a class="nav-
                    link" href="#register">Register</a></li>
            </ul>
        </nav>
    </header>
    <main class="container mt-3" id="content">
        <!-- Content will be loaded dynamically
                using JavaScript -->
    </main>
    <!-- Bootstrap JS (optional, for certain
                features) --> <script
    src="https://cdn.jsdelivr.net/npm/bootstrap
    @5.3.0/dist/js/bootstrap.bundle.min.js">
    </script><script src="script.js"></script>
```

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

Output:-

Shopping Cart

Catalog Cart login Register.



Week 3 :-

```
<html>
<head>
  <title > Welcome to NNRG e-Book's
           website </title>
  <script language = "javascript">
  function validate () {
   //username validation
  var uname = f1. username . value ;
  if (uname . length <=0)
  {
   alert ("Please Enter UserName");
   f1. username . focus ();
   return false ;
  }
  if (uname . length<8)
  {
   alert ("Please enter UserName not less
           than 8");
   f1. username . focus ();
   return false ;
  }
  // password validation
  var pwd = f1. password . value ;
  if (pwd . length <=0)
  {
   alert ("Please Enter password");
   f1. password . focus ();
   return false ;
  }
```

```
if(pwd.length<6)
{
  alert("Please enter Password not less
        than 6");
  f1.password.focus();
  return false;
}
// email validation
var email = f1.email.value;
if(email.length<=0)
{
  alert("Please Enter email");
  f1.email.focus();
  return false;
}
else {
  let eflag=false;
  for(i=0;i<email.length;i++) {
    if (email.charAt(i)=="@")
      {
        eflag=true;
      }
  }
    if(!(eflag))
    {
      alert("Please enter a valid Email ID");
      f1.email.focus();
      return false;
    }
}
```

```
//phone number validation
var phno=f1.phno.value;
if(phno.length<=0)
{
  alert("Please Enter Phone Number");
  f1.phno.focus();
  return false;
}
if(isNaN(phno))
{
  alert("Please Enter Valid Phone Number");
  f1.phnno.focus();
  return false;
}
if(phno.length!=10)
{
  alert("Please Enter Valid Phone Number");
  f1.phno.focus();
  return false;
}
//gender validation
let flag=false;
for(i=0;i<f1.gen.length;i++)
  if(f1.gen[i].checked)
    flag=true;
  if(!(flag))
  {
```

```
alert("Please choose a Gender");
return false;
}
// Language validation
flag=false;
for(i=0;i<fl.lang.length;i++)
    if(fl.lang[i].checked)
        flag=true;
if(!(flag))
{
alert("Please select at least one of the
        language options.");
return false;
}
// address validation
var addr=fl.address.value;
if(addr.length<=0)
{
  alert("Please enter address");
  fl.address.focus();
  return false;
}
//to display success message
alert("Registration Successful")
}
</script>
</head>
```

```
<body>
<center><br>
  <h3>Registration Form </h3>
  <br/>
  <form name ="f1">
  <table cellpadding="1" align ="center">
  <tr><td> User Name :* </td>
  <td><input type ="text" name ="username">
                              </td></tr>

  <tr><td> Password :* </td>
  <td><input type ="text" "password"
       name ="password"></td></tr>

  <tr><td> Email ID : *</td>
  <td >input type ="text" name ="email">
                              </td></tr>

  <tr><td> Phone Number :* </td>
  <td><input type ="text"name ="phno">
                              </td></tr>

  <tr><td valign ="top"> Gender :* </td>
  <td><input type ="radio" name ="gen"
       value ="Male"> Male   
  <input type ="radio" name ="gen" value =
       "Female">Female </td></tr>

  <tr><td valign ="top"> Language known :* </td>
  <td><input type ="checkbox" name ="lang"
       value ="English"> English <br/>
  <input type ="checkbox" name ="lang" value =
       "Telugu"> Telugu <br>
```

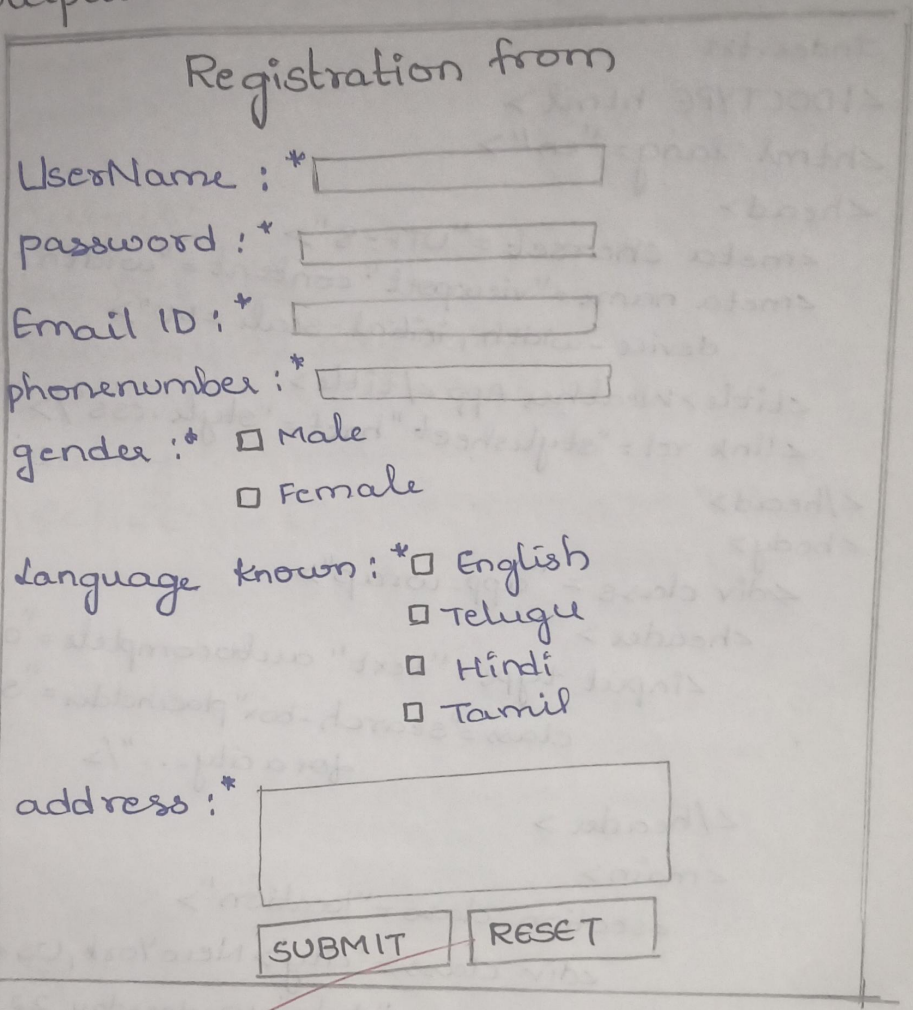
```
<input type ="checkbox" name ="lang" value =
        "Hindi" > Hindi <br>
<input type ="checkbox" name ="lang" value =
        "Tamil " > Tamil

</td></tr>
<tr><td valign ="top" >Address: *</td>
<td> <textarea name ="address "></textarea>
                                    </td>

<tr><td></td><td ><input type ="button"
        value ="SUBMIT" hspace ="10" onclick =
                "validate()">

<input type ="reset" value ="RESET"></td></tr
<tr><td colspan = 2 >* <font color ="
        #FF0000"> fields are mandatory</font
</td>
</tr>
</table>
</form>
</center>
</body>
</html>
```

Output:



29/9/24

Week 4

Index.txt

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=
        device-width, initial-scale=1.0">
    <title>Weather App</title>
    <link rel="stylesheet" href="style.css"/>
</head>
<body>
    <div class="app-wrap">
      <header>
        <input type="text" autocomplete="off"
            class="search-box" placeholder="search
                for a city..."/>
      </header>
      <main>
        <section class="location">
          <div class="city">New York, US</div>
          <div class="date">Wednesday 22 July
              2020</div>

        </section>
        <div class="current">
          <div class="temp">15<span>°C
              </span></div>
          <div class="weather">Sunny</div>
          <div class="hi-low">13°C / 16°C</div>
      </div>
    </main>
</div>
```

```
<script src="script.js"></script>
</body>
</html>
```

Output:

New York, US
Wednesday 22 July 2020
15°C
Sunny
13°C/16°C

style.txt

```
* {
  margin : 10 px ;
  padding : 10 px ;
  box-sizing : border-box;
}
body {
  font-family : 'Arial Black', sans-serif ;
  background-color : lightblue ;
  background-image : url ("bg.jpg");
  background-size : cover ;
  background-position : top center ;
}
.app-wrap {
  display : flex ;
  flex-direction : column;
  min-height : 100vh ;
  background-image : linear-gradient (to bottom,
                     rgba (0,0,0,0.3),
                     rgba (0,0,0,0.3));
}
header {
  display : flex ;
  justify-content : center;
  align-items : center;
  padding : 50px 15px 15px;
}
header input {
  width : 100 % ;
  max-width : 280px;
  padding : 10px 15px :
  border : none ;
```

```
outline : none;
background-color : lightyellow;
border-radius : 0px 16px 0px 16px;
border-bottom : 3px solid gray;

color : #313131;
font-size : 20px;
font-weight : 300;
transition : 0.2s ease-out;

}
header input :focus {
    background-color : rgba(255, 255, 255, 0.6);
}
main {
 flex : 1 1 100%;
 padding : 25px 25px 50px;
 display : flex;
 flex-direction : column;
 align-items : center;
 text-align : center;

}
.location, city {
    color : #fff.
 font-size : 32px;
 font-weight : 500;
  margin-bottom : 5px;
 }
.location.date {
    color : #fff
  font-size : 16px;
}
```

```
.current .temp {
  color : #fff;
  font-size : 102px;
  font-weight : 900;
  margin : 30px 0px;
  text-shadow : 2px 10px rgba(0,0,0,0.6);
}
.current .temp span {
  font-weight : 500;
}
.current .weather {
  color : #fff;
  font-size : 32px;
  font-weight : 700;
  font-style : italic;
  margin-bottom : 15px;
  text-shadow : 0px 3px rgba(0,0,0,0.4);
}
.current .hi-low {
  color : #fff ;
  font-size : 24px ;
  font-weight : 500;
  text-shadow : 0px 4px rgba(0,0,0,0.4);
}
```

```
const api = {
    key: "fcc
    base = "https://api.openweathermap.org/data/2.5/"
}
const searchbox = document.querySelector('.search-box');
searchbox.addEventListener('keypress', setQuery);
function setQuery(evt) {
    if(evt.keyCode == 13) {
        getResults(searchbox.value);
    }
}
function getResults(query) {
    fetch(`${api.base}weather?q=${query}&units=metric&APPID=${api.key}`).
    .then(weather => {
        return weather.json();
    }).then(displayResults);
}
function displayResults(weather) {
    let city = document.querySelector('.location.city');
    city.innerText = `${weather.name}, ${weather.sys.country}`;
    let now = new Date();
    let data = document.querySelector('.location.date');
    date.innerText = dateBuilder(now);
    let temp = document.querySelector('.current.temp');
    temp.innerHTML = `${Math.round(weather.main.temp)}<span>°c /</span>`;
    let hilow = document.querySelector('.hi-low');
    hilow.innerText = `${Math.round(weather.main.temp_min)}°c / ${Math.round(weather.main.temp_max)}°c`;
```

```
}
function dateBuilder (d) {
  let month = ["January", "February", "March",
      "April", "May", "June", "July", "August", "September",
      "October", "November", "December"];
  let days = ["Sunday", "Monday", "Tuesday",
      "Wednesday", "Thursday", "Friday", "Saturday"];
  let day = days[d.getDay()];
  let date = d.getDate();
  let month = months[d.getMonth()];
  let year = d.getFullYear();

  return `${day} ${date} ${month} ${year}`;
}
```

5. Develop a jara standalone application that connects with the database & perform the CRUD operation on the database tables

Aim :- Develop a jara standalone application that connects with the database (oracle/mysql) & perform the CRUD operation on the database tables

Description : Let's create a simple jara standdlone application that connects to a MySQL database & perform CRUD operations on a table

Prerequisites :

Make sure you have mySQL installed and known name, pwd, download the mySQL JDBC driver.

Example Java Applications

Let's assume we have a table named employees with columns id, name and salary

```
import jara.sql.*;
public class CRUDE Example {
private static final string JDBC-URL = "jdbc:mysql://
private static final string USERNAME = "your_username";
private static final string PASSWORD = "your-password";
public static void main (String[] args) {
try {
// step 1: Establishing a connection
connection = Driver Manager.getConnection
// step 2: Creating a statement.
statement = connection.create statement
// step 3: Performing CRUD operations
createRecords (statement, "John", 50000);
```

```
readRecords(statement);
updateRecord(statement, 1, "John updated", 55000);
readRecords(statement);
deleteRecords(statement, 1);
readRecords(statement);

// Step 4: closing stat resources
statement.close();
connection.close();
}
catch(SQLException e) {
e.printStackTrace();
}}

// Create a new record in the database
private static void createRecord(statemet, string
name, int salary) throws SQLException
{
String insertQuery = "INSERT INTO employees(
name, salary) VALUES('" + name + "', '" + salary + "')";
statement.executeUpdate(insertQuery);
System.out.println("Record created successfully");
}

// Read all records from the database
private static void readRecords(statement) throws
SQL exception {
string selectQuery = 'SELECT" From employees";
ResultSet
resultset = statement.executeQuery(selectQuery);
system.out.println("ID\tName\tSalary");
while(resultSet.next()){
int id = resultSet.getInt("id");
String name = resultSet.getString("name");
```

```
int salary = resultSet.getInt("salary");
system.out.println(id + "\t" + name + "\t" + salary);
}
system.out.println();
}
// update a record in the database
private static void updateRecord(statement, int
    id, string newName, int newsalary)
throws SQLException
{
string updateQuery = "UPDATE employees SET
    name = " + newName + "', salary = " +
    newsalary + "where id = " + id;
statement.executeUpdate(updateQuery);
system.out.println("Record updated successfully");
}
// Delete a record from the database
private static void deleteRecord(statement, int id)
                throws SQLException {
string deleteQuery = "DELETE FROM employee
    WHERE id = " + id; statement.execute
    Update(deleteQuery);
System.out.println("Record deleted successfully");
}
}
```

O/P.

Record created successfully

ID Name Salary

1 John Doe 50000

Record updated successfully

1 John Updated 55000

Record deleted successfully

This demonstrates a simple java standalone application for CRUD operation on a MySQL database using JDBC.

6. Create an xml for the bookstore. validate the same using both DTD & XSD

Aim: Create an xml for the bookstore. validate the same using both DTD & XSD

Description: let's create an XML file for a simple bookstore & validate it using both Document Type Definition & XML schema Definition XSD.

bookstore.xml:

```
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, author, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

XSD File

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/
                    XMLSchema">
<xs.element name="bookstore" type="bookstoreType"/>
<xs.complexType name="bookstoreType">
<xs:sequence>
<xs:element name="book" maxOccurs="unbounded"/>
</xs:sequence>
</xs.complexType>
<xs:complexType name="bookType">
<xs:sequence>
<xs:element name="title" ="xs.string"/>
<xs:element name="author" type="xs.string"/>
<xs:element name="price" type="xs.decimal"/>
</xs.sequence>
</xs:complexType>
</xs:schema>
```

Validation: You can validate the xml file using a tool or programming language that supports DTD and xsd validation.

Java program for validation

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import javax.xml.XMLConstants;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
import org.xml.sax.SAXException;
public class XMLValidator {
    public static void main(String[] args){
        validateWithDTD("bookstore.xml", "bookstore.dtd");
        validateWithXSD("bookstore.xml", "bookstore.xsd");
    }
    private static void validateWithDTD(String xmlFile, String dtdFile) {
    try {
    InputStream xmlStream = new FileInputStream(
                new File(xmlFile));
    InputStream dtdStream = new FileInputStream(
                new File(dtdFile));
        SchemaFactory schemaFactory =
        SchemaFactory.newInstance(XMLConstants.XML
            _DTD_NS_URI);
    Schema schema = schemaFactory.newSchema
                (new StreamSource(dtdStream));
    Validator validator = schema.newValidator();
```

```
validator.validate(new StreamSource(xmlstream));
System.out.println("Validation with DTD successfull.");
}
catch(SAXException | IOException e) {
  system.out.println("Validation with DTD failed.
                Reason:"+e.getMessage());
}
}
private static void validateWithXSD(String xmlFile,
        String xsdFile)
{
try {
InputStream xmlstream = new FileInputStream(
                new File(xmlFile));
InputStream xsdStream = new FileInputStream(new
                File(xsdFile));
SchemaFactory schemaFactory =
SchemaFactory.newInstance(XMLConstants.W3C_
        XML_SCHEMA_NS_URI);
Schema schema = schemaFactory.newSchema(
                new StreamSource(xsdStream));
validator validator.validate(new StreamSource(XML
                Stream));
}
System.out.println("Validation with XSD failed.
        Reason:"+e.getMessage());
}
}
}
```

Output :-

If the XML file adheres to the specified DTD & XSD, you will see output message like :

Validation with DTS Successfull
Validation with XSD successfull

If there are issues with the XML file, the program will print error messages explaining the validation failure.

Make sure to replace "bookstore.xml", "bookstore.dtb" & "bookstore-xsd" with the actual file paths in your projects.

7. Design a controller with servlet that provides the interaction with application dereloped in exp 1 & the database created in ex 5.

Aim :- Design a controller with servlet that provides the interaction with application dereloped in exp1 : Build a responsive web app for shopping cart with reg, login, catalog and cart pages using CSS3 features, flex & grid and the database created in ex. 5 : Derelop a java standalone application that connects with the database & perform the CRUD operation on the database table.

Description :

To design a servlet controller that interacts with the shopping cart application. Ex1 : & the database, Ex 5 : you would typically handle HTTP requests from from the web app, process, the data & communicate with the database to perform CRUD operation.

Servlet Controller (shoppingCartController.java),

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
Import java.servlet.http.HttpServletResponse;
@WebServlet("/shoppingCartController") public
    class shoppingCartController extends
HttpServlet {
    protected void doPost(HttpServletRequest
        request, HttpServletResponse throws
        IOException {
String action = request.getParameter("action");
if (action != null) {
    switch (action) {
```

```
case "register":
    handleRegistration(request, response);
break;
case "login":
    handleLogin(request, response);
    break;
case "addToCart";
    handleAddToCart(request, response);
    break;
// Add more cases for other actions as needed
default:
response.sendError(HttpServletResponse.SC_BAD_REQUEST,
        "Invalid action");
  }
}
else {
response.sendError(HttpServletResponse.SC_BAD_REQUEST,
        "Action parameter missing");
  }
}
private void handleRegistration(HttpServletRequest
        request, HttpServletResponse response)
throws IOException {
string username = request.getParameter("username");
string password = request.getParameter("password");
// Perform reg logic
// send response to the client
PrintWriter out = response.getWriter();
out.println("Registration successful"); }
private void handleLogin(Http servlet Request Respon)
throws IOException
{
```

```
// Extract cart data from request
string productId = request.getParameter("productID");
// Additional parameters as needed
// Perform logic to add the product to the user's
   cart (eg. update database)
// Send response to the client
PrintWriter out = response.getWriter();
out.println("product added to cart");
}
// Add more methods for other actions as needed
}
```

Output :-

8. Maintaining the transactional history of any user is very important. Explore the various session tracking mechanism

Aim: Maintaining the transactional history of any user is very important explore the various session tracking mechanism

Description

Session tracking mechanisms are crucial for maintaining the state of a user's interactions with a web application. Two methods for session tracking are cookies & HTTP sessions

1. Cookies :- cookies are small data pieces stored on a user's devices by a web browser, used to maintain user-specific information b/w the client and user

Suppose you want to track the user's language perference.

Server-side [in a web server script eg: python with flask);

```
from flask import Flask, request, render_template, make_response
app = Flask(__name__)
@app.route('/')
def index();
#check if the language cookie is set
user_language = request.cookies.get(user_language)
return
render_template('index.html', user_language=user_language)
@app.route("/set_language/<language>")
def set_language(language);
#set the language perference in a cookie
```

```
response = make_response(render_template('set_
        language.html'))
response.set_cookie('user_language', language)
return response
if __name__ == '__main__':
app.run(debug=True)
```

HTML Templates

```
<!--index.html-->
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-
        width, initial-scale=1.0">
 <title>cookie Example</title>
 </head>
 <body>
 <h1>welcome to the website</h1>
 {% if user_language %}
 <p>Your prefered language is : {{user_language}}
 </p>
 {% else %}
 <p>Your language preference is not set.</p>
 {% endif %}
 <p><a href="/set_language/en">Set language
                to English</a></p>
 <p><a href="/set_language/es">Set language
                to spanish</a></p>

 </body>
 </html>
 <!--set_language.html-->
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
                initial-scale=1.0">
  <title>set language</title>
</head>
<body>
  <h2>language set successfully!</h2>
  <p><a href="/">Go back to the home page
                </a></p>
</body>
</html>
```

output :

when a user visits the site for the first time, the lang preferences is not set. when the user clicks on "set language to English", the prefeerce is stored in a cookie

2 HTTP session : An HTTP session is a way to store infomation on the server side b/w requests from the client. Each client gets a unique sessionID, which is used to retriere session data (same)

- Example : suppose you want to track the no of visits for each user. Server-side.

```
from flask import Flask, request, render_template,
                                        session
app = Flask(__name__)
app.secret_key = 'super_secret_key'
#Set a secret key for session management
@app.route('/')
def index():
#Increment the visit count in the session
```

```
    session['visit_count'] = session.get('visit_count', 0) + 1
    return render_template('index_session.html', visit_count = session['visit_count'])

if __name__ == '__main__':
    app.run(debug=True)
```

HTML Template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Session Example</title>
</head>
<body>
    <h1>Welcome to the website</h1>
    <p>This is your visit number: {{ visit_count }}</p>
</body>
</html>
```

Output

- Each time a user visits the site, the server increments the visit count stored in the session
- The visit count is unique to each user & persists across multiple requests until the session expires.

9. Create a custom server using http module & explore the other modules of NodeJS like os, path, event.

Aim: Create a custom server using http module & explore the other modules of NodeJS like os, path, event.

Description: Let's create a simple custom server using the http module in Node.js and then explore the os, path & events modules with examples

1. Creating a Custom Server with http Module:

Server-side (server.js);

```
const http = require("http");
const server = http.createServer((req, res) => {
res.writeHead(200, {'content-Type':'text/plain'
});
res.end("Hello, this is a custom server!");});
const PORT = 30000, server.listen(PORT, () => {
console.log(Server is Listening on port ${PORT;});});
```

To run the server: node server.js.

Output:

Visit http://localhost:3000 in your browser or use a tool like curl or postman to make a request & you should see the response. "Hello" this is a custom server!

2. Exploring Node.js Module.

Ex:-

```
const os = require('os');
console.log('os Platform:', os.platform());
console.log('OS Architecture:', os.arch());
console.log("Total Memory (in bytes):", os.totalmem());
console.log("Free Memory (in bytes):", os.freemem());
```

Output:

This will output info about the os platform, architecture, total memory, & free memory