

```

<!--Week1 : Build a responsive web application for shopping cart with registration, login,
catalog and cart pages using CSS3 features, flex and grid.-->
<!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>
<div class="grid-container" style="color: blue;">
<div class="grid-item">Laptop</div>
<div class="grid-item">Printer</div>
<div class="grid-item">Bag</div>
<div class="grid-item">Laptop</div>
<div class="grid-item">Bag</div>
<div class="grid-item">Laptop</div>
</div>
<footer>
<div class="footer-bottom">2025-CJITS::CSE
</div>
</footer></body> </html>

```

```
<!--CSS Styles-->
```

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
}
```

```
header {  
    background-color: lightblue;  
    color: white;  
    padding: 10px;  
    text-align: center;  
}  
.footer-bottom{  
position: fixed;  
left:0;  
bottom: 0;  
width: 100%;  
background-color: red;  
color: white;  
text-align: center;  
}
```

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

```
}
```

```
main  
{  
padding: 20px;
```

```
}
```

```
.grid-container {  
display: grid;  
column-gap: 6px;  
grid-template-columns: 203px 203px 203px;  
padding: 10px;  
}
```

```
.grid-item {  
padding: 5px;  
font-size: 30px;  
text-align: center;  
}
```

Week 2:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Shopping Cart :: CJITS</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoLi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
      <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container">
        <a class="navbar-brand" href="#">Shopping Cart</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="#">Catalog</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Cart</a>
            </li>
            <li class="nav-item dropdown">
              <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
                Products
              </a>
              <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                <li><a class="dropdown-item" href="#">Mobiles</a></li>
                <li><a class="dropdown-item" href="#">Grocery</a></li>
                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item" href="#">Electronics</a></li>
              </ul>
            </li>
          </ul>
        </div>
        <form class="d-flex" role="search">
          <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        </form>
      </div>
    </nav>
  </body>

```

```

        <button class="btn btn-outline-success" type="submit">Search Product</button>
    </form>
</div>
</div>
</nav>

<div class="container my-5">
<h1>Shopping Cart</h1>
<div class="col-lg-8 px-0">
    <p class="fs-5">Shoping Cart is a place to buy products online... </p>

    <hr class="col-1 my-4">

    <a href="../exp3/#login" class="btn btn-primary">Login</a>
    <a href="../exp3/#signup" class="btn btn-secondary">Sign Up</a>
</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
<script src="main.js"></script>
</body>
</html>

```

<!—CSS Styles-- >

```

body
{
font-family: 'Arial', sans-serif; margin: 0; padding: 0;
}
header
{
background-color: #333; color: #fff; padding: 10px; text-align: center;
}
.footer-bottom{
position: fixed;
left:0;
bottom: 0;
width: 100%;
background-color: red;
color: white;
text-align: center;
}
nav ul { list-style: none; padding: 0; display: flex; justify-content: center; }
nav li { margin: 0 10px; } main { padding: 20px; }

```

Week 3 :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link href="bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <link rel="stylesheet" href="styles.css">
<title>LogIn and SignUp Page</title>
</head>
<body>
    <div class="box">
        
        <div class="page">
            <div class="header">
                <a id="login" class="active" href="#login">login</a>
                <a id="signup" href="#signup">signup</a>
            </div>
            <div id="errorMsg"></div>
            <div class="content">
                <form class="login" name="loginForm" onsubmit="return validateLoginForm()"
method="POST">
                    <input type="text" name="name" id="logName"
placeholder="Username">
                    <input type="password" name="password" id="logPassword"
placeholder="Password">
                    <div id="check">
                        <input type="checkbox" id="remember">
                        <label for="remember">Remember me</label>
                    </div>
                    <br><br>
                    <input type="submit" value="Login">
                    <a href="#">Forgot Password?</a>
                </form>
                <form class="signup" name="signupForm" onsubmit="return validateSignupForm()"
method="POST">
                    <input type="email" name="email" id="signEmail" placeholder="Email">
                    <input type="text" name="name" id="signName" placeholder="Username">
                    <input type="password" name="password" id="signPassword" placeholder="Password"><br>
                    <input type="submit" value="SignUp">
                </form>
            </div>
        </div>
    </div>
<script src="jquery-3.3.1.min.js">
```

```
integrity="sha256-FgpCb/KJQILNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
crossorigin="anonymous">></script>
<script>
    $(window).on("hashchange", function () {
        if (location.hash.slice(1) == "signup") {
            $(".page").addClass("extend");
            $("#login").removeClass("active");
            $("#signup").addClass("active");
        } else {
            $(".page").removeClass("extend");
            $("#login").addClass("active");
            $("#signup").removeClass("active");
        }
    });
    $(window).trigger("hashchange");
function validateLoginForm()
{
    var name = document.getElementById("logName").value;
    var password = document.getElementById("logPassword").value;
    if (name == "" || password == "") {
        document.getElementById("errorMsg").innerHTML = "Please fill the required fields"
        return false;
    }
    else if (password.length < 8)
    {
        document.getElementById("errorMsg").innerHTML = "Your password must include atleast
8 characters"
        return false;
    }
    else if(name=="cse" && password=="cse@2025")
    {
        alert("Successfully logged in");
        return true;
    }
    else
    {
        document.getElementById("errorMsg").innerHTML = "You are not a authorized person to
login!.."
        return false;
    }
}
function validateSignupForm()
{
    var mail = document.getElementById("signEmail").value;
    var name = document.getElementById("signName").value;
```

```

var password = document.getElementById("signPassword").value;
if (mail == "" || name == "" || password == "")
{
    document.getElementById("errorMsg").innerHTML = "Please fill the required fields"
    return false;
}
else if (password.length < 8)
{
    document.getElementById("errorMsg").innerHTML = "Your password must include atleast
8 characters"
    return false;
}
else
{
    alert("Successfully signed up");
    return true;
}
}
</script>
</body>
</html>

```

CSS STYLES

```

body{
margin: 0;
padding: 0;
background-image: url(background.jpg);
background-size: cover;
background-repeat: no-repeat;
font-family : Verdana,Tahoma, sans-serif;
}
.box{
width: 380px;
height: 500px;
background-color:rgba(0, 0, 0, 0.7);
padding: 20px;
margin: 8% auto 0;
text-align: center;
position: relative;
box-sizing: border-box;
flex-direction: column;
}
.box img{
width:100px;
margin-top:-50px ;

```

```
}

.header {
    width: 300px;
    height: 50px;
    text-transform: uppercase;
    display: inline-flex;
    padding-top: 20px;
    padding-bottom: 0;
    justify-content: space-between;
}
.header a {
    font-size: 20px;
    text-decoration: none;
    color: #ffffff;
    display: inline-flex;
    padding-left: 25px;
    padding-right: 25px;
    justify-content: center;
    cursor:pointer;
}
.header .active {
    color: #df80ff;
    font-weight: bold;
    position: relative;
}
.header .active:after {
    position: absolute;
    border: none;
    content: "";
}
#errorMsg{
    color:red;
    text-align: center;
    font-size: 12px;
    padding-bottom: 20px;
}
.content{
    display: inline-flex;
    overflow: hidden;
}
form {
    position: relative;
    display: inline-flex;
    width: 100%;
    height: 100%;
```

```
flex-shrink:  
0;  
flex-direction: column;  
transition: right 0.5s;  
}  
.login a{  
text-decoration: none;  
color: #ffffff;  
font-size: 15px;  
text-align: center;  
cursor: pointer;  
}  
.extend form {  
right: 100%;  
}  
input[type=text], input[type=password], input[type=email]{  
display:block;  
position: relative;  
border: 4px solid #ffffff;  
padding: 12px;  
margin-bottom: 15px;  
width: 100%;  
box-sizing: border-box;  
font-size: 17px;  
outline: none;  
}  
input[type=text]:hover, input[type=password]:hover, input[type=email]:hover{  
border: 4px solid #df80ff;  
}  
input[type=submit]{  
display:block;  
position: relative;  
border: 3px solid #df80ff;  
padding: 12px;  
margin-bottom: 20px;  
width: 100%;  
border-radius: 25px;  
background-color:#600080 ;  
text-align:center;  
font-size: 20px;  
color: #ffffff;  
cursor:pointer;  
outline: none;  
}  
input[type=submit]:hover{  
background-color:#ffffff;
```

```
color: #000000;
font-weight: bold;
}
input[type=checkbox]{
background-color:#df80ff;
}
#check{
margin-top: 10px;
text-align: left;
color: #ffffff;
font-size: 15px;
}
```

Week 4:

```
<!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>Weather Information</title>
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<style>
body {
  font-family: Arial, sans-serif;
  margin: 20px;
}
canvas {
  max-width: 600px;
  margin: 20px auto;
}
</style>
</head>
<body>
  <div class="container">
    <h1>Weather Information</h1>
    <label for="city">Enter City:</label>
    <input type="text" id="city" placeholder="City Name">
    <button id="getWeather">Get Weather</button>
  </div>
  <canvas id="weatherGraph"></canvas>
  <script>
    document.addEventListener('DOMContentLoaded', () => {
      const apiKey = 'f0c4d53e17dfb685a9aa5cebabd90eff'; // Replace with your OpenWeatherMap
      API key
      const getWeatherButton = document.getElementById('getWeather');
      const cityInput = document.getElementById('city');
      const weatherGraph = document.getElementById('weatherGraph').getContext('2d');

      // Function to fetch weather data using async/await
      const fetchWeatherData = async (city, callback) => {
        const apiUrl =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;
        try {
          const response = await axios.get(apiUrl);
          callback(null, response.data); // Use callback to pass data
        } catch (error) {
          callback(error, null); // Handle errors in the callback
        }
      }
    });
  </script>
</body>
```

```

        }
    };
// Function to update the graph
const updateGraph = (data) => {
    const temperature = data.main.temp;
    const feelsLike = data.main.feels_like;
    const humidity = data.main.humidity;
    new Chart(weatherGraph, {
        type: 'bar',
        data: {
            labels: ['Temperature (°C)', 'Feels Like (°C)', 'Humidity (%)'],
            datasets: [
                {
                    label: `Weather Data for ${data.name}`,
                    data: [temperature, feelsLike, humidity],
                    backgroundColor: ['#36A2EB', '#FF6384', '#FFCD56'],
                },
            ],
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true,
                },
            },
        },
    });
};

// Event listener for button click
getWeatherButton.addEventListener('click', () => {
    const city = cityInput.value.trim();
    if (!city) {
        alert('Please enter a city name');
        return;
    }
    // Fetch weather data and update the graph
    fetchWeatherData(city, (error, data) => {
        if (error) {
            console.error('Error fetching weather data:', error.message);
            alert('Unable to fetch weather data. Please try again.');
        } else {
            updateGraph(data);
        }
    });
});
});    </script> </body> </html>

```

Week 5:

```
import java.sql.*;
import java.util.Scanner;

public class MySQLCrudApp {

    static final String URL = "jdbc:mysql://localhost:3306/cjits";
    static final String USER = "root";
    static final String PASSWORD = "";

    static Connection connection = null;

    public static void main(String[] args) {
        try {

            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Connected to the database successfully!");

            Scanner scanner = new Scanner(System.in);

            while (true) {
                System.out.println("\nSelect an operation:");
                System.out.println("1. Create a new user");
                System.out.println("2. Read all users");
                System.out.println("3. Update user");
                System.out.println("4. Delete user");
                System.out.println("5. Exit");

                int choice = scanner.nextInt();
                scanner.nextLine(); // Consume the newline character

                switch (choice) {
                    case 1:
                        createUser(scanner);
                        break;
                    case 2:
                        readUsers();
                        break;
                    case 3:
                        updateUser(scanner);
                        break;
                    case 4:
                        deleteUser(scanner);
                        break;
                    case 5:
                        System.out.println("Exiting...");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void createUser(Scanner scanner) {
        // Implementation for creating a new user
    }

    private void readUsers() {
        // Implementation for reading all users
    }

    private void updateUser(Scanner scanner) {
        // Implementation for updating a user
    }

    private void deleteUser(Scanner scanner) {
        // Implementation for deleting a user
    }
}
```

```

        connection.close();
        System.exit(0);
    default:
        System.out.println("Invalid choice. Please try again.");
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}
}

// Create a new user in the database
private static void createUser(Scanner scanner) {
try {
    System.out.println("Enter name:");
    String name = scanner.nextLine();
    System.out.println("Enter email:");
    String email = scanner.nextLine();
    System.out.println("Enter age:");
    int age = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    String sql = "INSERT INTO users (name, email, age) VALUES (?, ?, ?)";
    PreparedStatement statement = connection.prepareStatement(sql);
    statement.setString(1, name);
    statement.setString(2, email);
    statement.setInt(3, age);

    int rowsInserted = statement.executeUpdate();
    if (rowsInserted > 0) {
        System.out.println("A new user was inserted successfully!");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

// Read all users from the database
private static void readUsers() {
try {
    String sql = "SELECT * FROM users";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sql);

    while (resultSet.next()) {
        int id = resultSet.getInt("id");
        String name = resultSet.getString("name");
        String email = resultSet.getString("email");
        int age = resultSet.getInt("age");
        System.out.println(id + " | " + name + " | " + email + " | " + age);
    }
}
}

```

```

        String name = resultSet.getString("name");
        String email = resultSet.getString("email");
        int age = resultSet.getInt("age");
        System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email + ", Age: " +
age);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

// Update a user in the database
private static void updateUser(Scanner scanner) {
try {
    System.out.println("Enter user ID to update:");
    int userId = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    System.out.println("Enter new name:");
    String name = scanner.nextLine();
    System.out.println("Enter new email:");
    String email = scanner.nextLine();
    System.out.println("Enter new age:");
    int age = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    String sql = "UPDATE users SET name = ?, email = ?, age = ? WHERE id = ?";
    PreparedStatement statement = connection.prepareStatement(sql);
    statement.setString(1, name);
    statement.setString(2, email);
    statement.setInt(3, age);
    statement.setInt(4, userId);

    int rowsUpdated = statement.executeUpdate();
    if (rowsUpdated > 0) {
        System.out.println("An existing user was updated successfully!");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

// Delete a user from the database
private static void deleteUser(Scanner scanner) {
try {
    System.out.println("Enter user ID to delete:");

```

```
int userId = scanner.nextInt();
scanner.nextLine(); // Consume the newline character

String sql = "DELETE FROM users WHERE id = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, userId);

int rowsDeleted = statement.executeUpdate();
if (rowsDeleted > 0) {
    System.out.println("A user was deleted successfully!");
}
} catch (SQLException e) {
    e.printStackTrace();
}
}
```

Week 6:

```
import java.io.File;
import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
import org.w3c.dom.Document;

public class XMLValidator {

    public static void main(String[] args) {
        String xmlFile = "bookstore.xml";
        String dtdFile = "bookstore.dtd";
        String xsdFile = "bookstore.xsd";

        // Validate XML with DTD
        if (validateWithDTD(xmlFile, dtdFile)) {
            System.out.println("XML is valid against DTD.");
        } else {
            System.out.println("XML is NOT valid against DTD.");
        }

        // Validate XML with XSD
        if (validateWithXSD(xmlFile, xsdFile)) {
            System.out.println("XML is valid against XSD.");
        } else {
            System.out.println("XML is NOT valid against XSD.");
        }
    }

    public static boolean validateWithDTD(String xmlFile, String dtdFile) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            factory.setValidating(true);
            factory.setNamespaceAware(true);
            DocumentBuilder builder = factory.newDocumentBuilder();
            builder.setErrorHandler(null);
            Document document = builder.parse(new File(xmlFile));
            return true;
        } catch (Exception e) {
            System.err.println("DTD Validation Error: " + e.getMessage());
            return false;
        }
    }
}
```

```
        }
    }

public static boolean validateWithXSD(String xmlFile, String xsdFile) {
    try {
        SchemaFactory factory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = factory.newSchema(new File(xsdFile));
        Validator validator = schema.newValidator();
        Source source = new StreamSource(new File(xmlFile));
        validator.validate(source);
        return true;
    } catch (Exception e) {
        System.err.println("XSD Validation Error: " + e.getMessage());
        return false;
    }
}
```

Week 7:

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

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/cjitscse", "root", "");
            PreparedStatement ps = con.prepareStatement(
                "SELECT * FROM users WHERE username=? AND password=?");

            ps.setString(1, username);
            ps.setString(2, password);

            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                out.println("<h3>Login successful</h3>");
            } else {
                out.println("<h3>Unauthorized access</h3>");
            }

            rs.close();
            ps.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace(out);
        }
    }
}
```

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

public class RegisterServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String email = request.getParameter("email");
        String name = request.getParameter("name");

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/cjitscse", "root", "");
            PreparedStatement ps = con.prepareStatement(
                "INSERT INTO users (username, password, email, name) VALUES (?, ?, ?, ?)");
            ps.setString(1, username);
            ps.setString(2, password);
            ps.setString(3, email);
            ps.setString(4, name);

            int i = ps.executeUpdate();
            if (i > 0) {
                out.println("<h3>Registration successful</h3>");
            } else {
                out.println("<h3>Registration failed</h3>");
            }

            ps.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace(out);
        }
    }
}
```

Week 8:

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

public class WelcomeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();

        // Check for reset action
        String reset = request.getParameter("reset");
        if ("true".equals(reset)) {
            // Invalidate cookie
            Cookie cookie = new Cookie("username", "");
            cookie.setMaxAge(0); // Delete cookie
            response.addCookie(cookie);

            // Invalidate session data
            session.invalidate();

            out.println("<h2>Cookie and session have been reset.</h2>");
            out.println("<p><a href='index.html'>Go Back</a></p>");
            return;
        }

        // Get username from request or session
        String username = request.getParameter("username");
        if (username == null) {
            username = (String) session.getAttribute("username");
        } else {
            session.setAttribute("username", username);

            // Set cookie
            Cookie nameCookie = new Cookie("username", username);
            nameCookie.setMaxAge(60 * 60); // 1 hour
            response.addCookie(nameCookie);
        }

        // Visit count
        Integer visitCount = (Integer) session.getAttribute("visitCount");
        visitCount = (visitCount == null) ? 1 : visitCount + 1;
        session.setAttribute("visitCount", visitCount);
    }
}
```

```

if (username != null) {
    out.println("<h2>Hello, " + username + "</h2>");
    out.println("<p>This is your visit number: " + visitCount + "</p>");
} else {
    out.println("<p>No username found. Please <a href='index.html'>go back</a> and enter
your name.</p>");
}

out.println("<p><a href='index.html'>Go Back</a></p>");
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doPost(request, response);
}
}

```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<web-app xmlns="http://jakarta.ee/xml/ns/jakartaee" version="5.0">
<servlet>
<servlet-name>WelcomeServlet</servlet-name>
<servlet-class>WelcomeServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>WelcomeServlet</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>index.html</welcome-file>
</welcome-file-list>
</web-app>

```

```
<!DOCTYPE html>
<html>
<head>
    <title>Session and Cookie Example</title>
</head>
<body>
    <h2>Enter Your Name</h2>
    <form action="welcome" method="post">
        Name: <input type="text" name="username" required>
        <input type="submit" value="Submit">
    </form>

    <h3>OR</h3>

    <form action="welcome" method="post">
        <input type="hidden" name="reset" value="true">
        <input type="submit" value="Reset Cookie">
    </form>
</body>
</html>
```

```
javac -classpath "C:\xampp\tomcat\lib\servlet-api.jar" WelcomeServlet.java
```

Week 9:

```
// Import core modules
const http = require('http');
const os = require('os');
const path = require('path');
const events = require('events');

// Create an event emitter
const eventEmitter = new events.EventEmitter();

// Event handler for custom event
eventEmitter.on('userVisited', (url) => {
  console.log(`User visited: ${url}`);
});

// Create HTTP server
const server = http.createServer((req, res) => {
  // Emit custom event
  eventEmitter.emit('userVisited', req.url);

  // Routing
  if (req.url === '/') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end('<h1>Welcome to Custom Node Server</h1>');
  } else if (req.url === '/system') {
    // Use OS module
    const systemInfo = {
      hostname: os.hostname(),
      platform: os.platform(),
      architecture: os.arch(),
      cpus: os.cpus().length,
      memory: `${(os.totalmem() / (1024 * 1024 * 1024)).toFixed(2)} GB`,
      uptime: `${(os.uptime() / 60).toFixed(2)} minutes`
    };

    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify(systemInfo));
  } else if (req.url === '/path') {
    // Use path module
    const filePath = path.join(__dirname, 'public', 'index.html');
    const pathInfo = {
      base: path.basename(filePath),
      dir: path.dirname(filePath),
      ext: path.extname(filePath),
      resolved: path.resolve(filePath)
    };
  }
});
```

```
res.writeHead(200, { 'Content-Type': 'application/json' });
res.end(JSON.stringify(pathInfo));
} else {
  res.writeHead(404, { 'Content-Type': 'text/plain' });
  res.end('404 Not Found');
}
});

// Start server
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});

const http = require("http")

// Creating server
const server = http.createServer((req, res) => {
  // Sending the response
  res.write("This is the response from the server")
  res.end();
})

// Server listening to port 3000
server.listen((3000), () => {
  console.log("Server is Running");
})
```

Week 10:

```
const express = require('express');
const fs = require('fs');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;

app.use(bodyParser.json());

const readData = () => {
  const data = fs.readFileSync('students.json');
  return JSON.parse(data);
};

const writeData = (data) => {
  fs.writeFileSync('students.json', JSON.stringify(data, null, 2));
};

// Create Student (POST)
app.post('/students', (req, res) => {
  const students = readData();
  const newStudent = {
    id: Date.now(),
    name: req.body.name,
    age: req.body.age,
    course: req.body.course
  };
  students.push(newStudent);
  writeData(students);
  res.status(201).json(newStudent);
});

// Get All Students (GET)
app.get('/students', (req, res) => {
  const students = readData();
  res.json(students);
});

// Get Single Student (GET)
app.get('/students/:id', (req, res) => {
  const students = readData();
  const student = students.find(s => s.id === parseInt(req.params.id));
  if (student) {
    res.json(student);
  } else {
    res.status(404).json({ message: 'Student not found' });
  }
});
```

```

    }
});

// Update Student (PUT)
app.put('/students/:id', (req, res) => {
  const students = readData();
  const index = students.findIndex(s => s.id === parseInt(req.params.id));
  if (index !== -1) {
    students[index] = {
      ...students[index],
      name: req.body.name || students[index].name,
      age: req.body.age || students[index].age,
      course: req.body.course || students[index].course
    };
    writeData(students);
    res.json(students[index]);
  } else {
    res.status(404).json({ message: 'Student not found' });
  }
});

// Delete Student (DELETE)
app.delete('/students/:id', (req, res) => {
  let students = readData();
  const newStudents = students.filter(s => s.id !== parseInt(req.params.id));
  if (students.length === newStudents.length) {
    return res.status(404).json({ message: 'Student not found' });
  }
  writeData(newStudents);
  res.json({ message: 'Student deleted successfully' });
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});

[
  {
    "id": 1001,

```

```
        "name": "Ravi",
        "age": 22,
        "course": "Computer Science"
    },
    {
        "id": 1002,
        "name": "Avani",
        "age": 20,
        "course": "Mathematics"
    },
    {
        "id": 1003,
        "name": "Kiran",
        "age": 23,
        "course": "Physics"
    },
    {
        "id": 1004,
        "name": "Sneha",
        "age": 21,
        "course": "Chemistry"
    },
    {
        "id": 1005,
        "name": "Arjun",

```

```
"age": 24,  
"course": "Biology"  
,  
{  
    "id": 1006,  
    "name": "Meena",  
    "age": 22,  
    "course": "Electronics"  
,  
{  
    "id": 1007,  
    "name": "Rahul",  
    "age": 20,  
    "course": "Information Technology"  
,  
{  
    "id": 1008,  
    "name": "Divya",  
    "age": 23,  
    "course": "Data Science"  
,  
{  
    "id": 1009,  
    "name": "Suresh",  
    "age": 25,
```

```
"course": "Mechanical Engineering"  
},  
{  
    "id": 1010,  
    "name": "Lavanya",  
    "age": 21,  
    "course": "Civil Engineering"  
}  
]
```

Exercise-11:

For the above application create authorized end points using JWT (JSON Web Token).

Solution:

Step 1: Install Required Packages

- Install djangorestframework djangorestframework-simplejwt:
pip install djangorestframework djangorestframework-simplejwt

Step 2: Configure Django Settings

- Add 'rest_framework' and 'rest_framework_simplejwt' to your INSTALLED_APPS in settings.py:

```
# settings.py
INSTALLED_APPS = [
    # ...
    'rest_framework',
    'rest_framework_simplejwt',
]
· Configure authentication and include the simplejwt settings:
# settings.py
REST_FRAMEWORK =
{
    'DEFAULT_AUTHENTICATION_CLASSES':
        [
            'rest_framework_simplejwt.authentication.JWTAuthentication',
        ],
}
SIMPLE_JWT =
{
    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=30),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
    'SLIDING_TOKEN_REFRESH_LIFETIME': timedelta(days=14),
    'SLIDING_TOKEN_LIFETIME': timedelta(days=14),
    'ROTATE_REFRESH_TOKENS': False,
    'ALGORITHM': 'HS256',
    'SIGNING_KEY': SECRET_KEY,
    'VERIFYING_KEY': None,
    'AUTH_HEADER_TYPES': ('Bearer',),
```

```

    'USER_ID_FIELD': 'id',
    'USER_ID_CLAIM': 'user_id',
    'AUTH_TOKEN_CLASSES':
        ('rest_framework_simplejwt.tokens.AccessToken',),
    'TOKEN_TYPE CLAIM': 'token_type',
    'JTI CLAIM': 'jti',
    'SLIDING_TOKEN_REFRESH_EXP CLAIM': 'refresh_exp',
    'SLIDING_TOKEN_EXP CLAIM': 'exp',
    'ROTATE_REFRESH_TOKENS': False,
    'BLACKLIST_AFTER_ROTATION': True,
    'UPDATE_LAST_LOGIN': False,
}

```

Step 3: Create Django Models and Views

- Create models and views for your Django application.
- Create a simple model for a student and a protected API endpoint:

```

# models.py
from django.db import models
class Student(models.Model):
    name = models.CharField(max_length=255)
    email = models.EmailField(unique=True)

# views.py
from rest_framework import generics, permissions
from .models import Student
from .serializers import StudentSerializer
class StudentListCreateView(generics.ListCreateAPIView):
    queryset = Student.objects.all()
    serializer_class =
    StudentSerializer
    permission_classes =
    [permissions.IsAuthenticated]

```

Step 4: Create Serializers

- Create serializers for your models:

```

# serializers.py
from rest_framework import serializers
from .models import Student
class StudentSerializer(serializers.ModelSerializer):

```

```
class Meta:  
    model = Student  
    fields = '__all__'
```

Step 5: Configure URLs

Configure your URLs to include the protected endpoint:

```
# urls.py  
from django.urls import path  
from .views import StudentListCreateView  
urlpatterns = [  
    path('students/', StudentListCreateView.as_view(), name='student-list-create'),  
    # Include other URLs as needed  
]
```

Step 6: Generate JWT Tokens

To generate JWT tokens for testing, you can use the following code snippet:

```
from rest_framework_simplejwt.tokens import RefreshToken  
def generate_tokens(user):  
    refresh = RefreshToken.for_user(user)  
    return {  
        'refresh': str(refresh),  
        'access': str(refresh.access_token),  
    }
```

Step 7: Use Tokens to Access Protected Endpoint

Use the generated access token to access the protected endpoint in your API by including it in the Authorization header:

GET /students/ Authorization: Bearer <access_token>

Replace <access_token> with the actual access token generated using the generate_tokens function.

Output:

The screenshot shows the Postman application interface. In the top navigation bar, 'Postman' is selected. The main workspace shows a GET request to 'http://localhost:3000/user/data'. The 'Headers' tab is active, containing an 'Authorization' header with the value 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJ1...' and a 'Description' field. The 'Body' tab displays a JSON response with a status of 200 OK, time 36 ms, and size 513 B. The response body is a JSON object:

```
1 = {
  "message": "Successful log in",
  "authorizedData": [
    {
      "user": {
        "username": "user123",
        "password": "1234",
        "firstName": "Jon",
        "lastName": "Doe",
        "dob": "12/11/1991",
        "email": "user@gmail.com",
        "address": {
          "street": "555 Bayshore Blvd",
          "city": "Tampa",
          "state": "Florida",
          "zip": "33813"
        }
      },
      "iat": 1532916610,
      "exp": 1532920210
    }
  ]
}
```

Exercise-12:

Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.

Solution:

Step 1: Set Up the Django Environment:

Create a new Django project:

```
django-admin startproject student_management  
cd student_management
```

Create a new Django app:

```
python manage.py startapp student
```

Step 2: Set Up the Django Models

Define models for students in student/models.py:

```
# student/models.py  
from django.db import models  
class Student(models.Model):  
    username = models.CharField(max_length=100, unique=True)  
    password = models.CharField(max_length=100)  
    full_name = models.CharField(max_length=255)  
    email = models.EmailField(unique=True)  
    # Add other fields as needed
```

Run migrations:

```
python manage.py makemigrations  
python manage.py migrate
```

Step 3: Set Up Django REST Framework

Install Django REST framework:

```
pip install djangorestframework
```

Configure the app in settings.py:

```
# student_management/settings.py  
INSTALLED_APPS = [  
    # ...  
    'rest_framework',  
    'student',  
]
```

Step 4: Create Django Views and API Endpoints

- Create views in student/views.py for registration, login, contact, and about pages.
- Define API endpoints using Django REST framework.

Step 5: Set Up React App

Create a new React app inside the Django project:

```
npx create-react-app student_management_ui  
cd student_management_ui
```

Install dependencies:

```
npm install axios react-router-dom
```

Step 6: Create React Components

Create components for registration, login, contact, and about pages in the src folder.

Registration Component (Registration.js):

```
// src/components/Registration.js  
  
import React, { useState } from 'react';  
import axios from 'axios';  
  
const Registration = () => {  
  const [formData, setFormData] = useState({  
    username: "",  
    password: "",  
    fullName: "",  
    email: "",  
  });  
  const handleInputChange = (e) => {  
    setFormData({ ...formData, [e.target.name]: e.target.value });  
  };  
  const handleRegistration = async (e) => {  
    e.preventDefault();  
    try {  
      // Make a POST request to your Django API endpoint for registration  
      const response = await axios.post('http://localhost:8000/api/register/', formData);  
      console.log(response.data); // Handle the response as needed  
    } catch (error) {  
      console.error('Registration failed:', error);  
    }  
  };  
}
```

```
return (
  <div>
    <h2>Registration</h2>
    <form onSubmit={handleRegistration}>
      <label>
        Username:
        <input type="text" name="username" onChange={handleInputChange} required />
      </label>
      <br />
      <label>
        Password:
        <input type="password" name="password" onChange={handleInputChange} required />
      </label>
      <br />
      <label>
        Full Name:
        <input type="text" name="fullName" onChange={handleInputChange} required />
      </label>
      <br />
      <label>
        Email:
        <input type="email" name="email" onChange={handleInputChange} required />
      </label>
      <br />
      <button type="submit">Register</button>
    </form>
  </div>
);
};

export default Registration;
```

Login Component (Login.js):

```
// src/components/Login.js

import React, { useState } from 'react';
import axios from 'axios';

const Login = () => {
  const [formData, setFormData] = useState({
    username: '',
    password: '',
  });
  const handleInputChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      // Make a POST request to your Django API endpoint for login
      const response = await axios.post('http://localhost:8000/api/login/', formData);
      console.log(response.data); // Handle the response as needed
    } catch (error) {
      console.error('Login failed:', error);
    }
  };
  return (
    <div>
      <h2>Login</h2>
      <form onSubmit={handleLogin}>
        <label>
          Username:
          <input type="text" name="username" onChange={handleInputChange} required />
        </label>
        <br />
        <label>
          Password:
          <input type="password" name="password" onChange={handleInputChange} required />
        </label>
      </form>
    </div>
  );
}

export default Login;
```

```
</label>
<br />
<button type="submit">Login</button>
</form>
</div>
);
};

export default Login;
```

Contact Component (Contact.js):

```
// src/components/Contact.js
import React from 'react';
const Contact = () => {
  return (
    <div>
      <h2>Contact</h2>
      <p>This is the contact page content.</p>
    </div>
  );
};

export default Contact;
```

About Component (About.js):

```
// src/components/About.js
import React from 'react';
const About = () => {
  return (
    <div>
      <h2>About</h2>
      <p>This is the about page content.</p>
    </div>
  );
};

export default About;
```

Implement React Router to navigate between pages. Update src/App.js:

```
// src/App.js
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Registration from './components/Registration';
import Login from './components/Login';
import Contact from './components/Contact';
import About from './components/About';
function App() {
  return (
    <Router>
      <Switch>
        <Route path="/registration" component={Registration} />
        <Route path="/login" component={Login} />
        <Route path="/contact" component={Contact} />
        <Route path="/about" component={About} />
      </Switch>
    </Router>
  );
}
export default App;
```

Step 8: Connect React with Django API

- Make API requests from React components to Django API endpoints.
- Use the axios library for making HTTP requests.

Step 9: Test Your Application

Run both Django and React development servers:

```
# In the Django project directory
python manage.py runserver
```

```
# In the React project directory
```

```
  npm start
```

- Visit <http://localhost:3000> to see your React app. Navigate to the specified routes (e.g., /registration, /login, /contact, /about) to test the integration with Django.

Exercise-13:

Create a service in react that fetches the weather information from openweathermap.org and the display the current and historical weather information using graphical representation using chart.js

Solution:

Steps to create the application:

Step 1:

- Set up React project using the below command in VSCode IDE.
npx create-react-app weather-app

Step 2:

- Navigate to the newly created project folder by executing the below command.

```
cd weather-app
```

Step 3:

- As we are using various packages for the project, we need to install them.
- Use the below command to install all the packages that are specified in package.json file.

```
npm install axios react-loader-spinner @fortawesome/react-fontawesome  
@fortawesome/free-solid-svg-icons
```

- Example: Insert the below code in the respective files.

App.js:

- All the logic that is been used in the application is programmatically coded in this file.
- From this file, all the buttons, cards, and icons are been rendered in the web browser.

App.css:

- This file consists of all the styling code, whether it is h1 tag styling or background styling.
- All the look and feel of the application are specified in this styling file.

```
//App.js
```

```
import { faFrown } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import axios from 'axios';
import React, { useState } from 'react';
import { Oval } from 'react-loader-spinner';
import './App.css';

function WeatherApp() {
  const [input, setInput] = useState("");
  const [weather, setWeather] = useState({
```

```
loading: false,
data: {},
error: false,
});
const toDateFunction = () => {
const months = [
'January',
'February',
'March',
'April',
'May',
'June',
'July',
'August',
'September',
'October',
'November',
'December',
];
const WeekDays = [
'Sunday',
'Monday',
'Tuesday',
'Wednesday',
'Thursday',
'Friday',
'Saturday',
];
const currentDate = new Date();
const date = `${WeekDays[currentDate.getDay()]} ${currentDate.getDate()} ${months[currentDate.getMonth()]}`;
return date;
};
const search = async (event) => {
if (event.key === 'Enter') {
event.preventDefault();
setInput('');
setWeather({ ...weather, loading: true });
const url = 'https://api.openweathermap.org/data/2.5/weather';
const api_key = 'f00c38e0279b7bc85480c3fe775d518c';
```

```

await axios
  .get(url, {
    params: {
      q: input,
      units: 'metric',
      appid: api_key,
    },
  })
  .then((res) => {
    console.log('res', res);
    setWeather({ data: res.data, loading: false, error: false });
  })
  .catch((error) => {
    setWeather({ ...weather, data: {}, error: true });
    setInput("");
    console.log('error', error);
  });
}
};

return (
  <div className="App">
    <h1 className="app-name">
      Live Weather App
    </h1>
    <div className="search-bar">
      <input type="text"
        className="city-search"
        placeholder="Enter City Name.."
        name="query"
        value={input}
        onChange={(event) => setInput(event.target.value)}
        onKeyPress={search}
      />
    </div>
    {weather.loading && (
      <>
        <br />
        <br />
        <Oval type="Oval" color="black" height={100} width={100} />
      </>
    )}
  </div>
)

```

```

        )}
{weather.error && (
    <>
    <br />
    <br />
    <span className="error-message">
        <FontAwesomeIcon icon={faFrown} />
        <span style={{ fontSize: '20px' }}>City not found</span>
    </span>
    </>
)}
{weather && weather.data && weather.data.main && (
    <div>
        <div className="city-name">
            <h2>
                {weather.data.name}, <span>{weather.data.sys.country}</span>
            </h2>
        </div>
        <div className="date">
            <span>{toDateFunction()}</span>
        </div>
        <div className="icon-temp">
            <img className="" alt="Weather icon" src={`https://openweathermap.org/img/wn/${weather.data.weather[0].icon}@2x.png`}
            alt={weather.data.weather[0].description}
            />
            {Math.round(weather.data.main.temp)}
            <sup className="deg">°C</sup>
        </div>
        <div className="des-wind">
            <p>{weather.data.weather[0].description.toUpperCase()}</p>
            <p>Wind Speed: {weather.data.wind.speed}m/s</p>
        </div>
    </div>
    )};
}
export default WeatherApp;

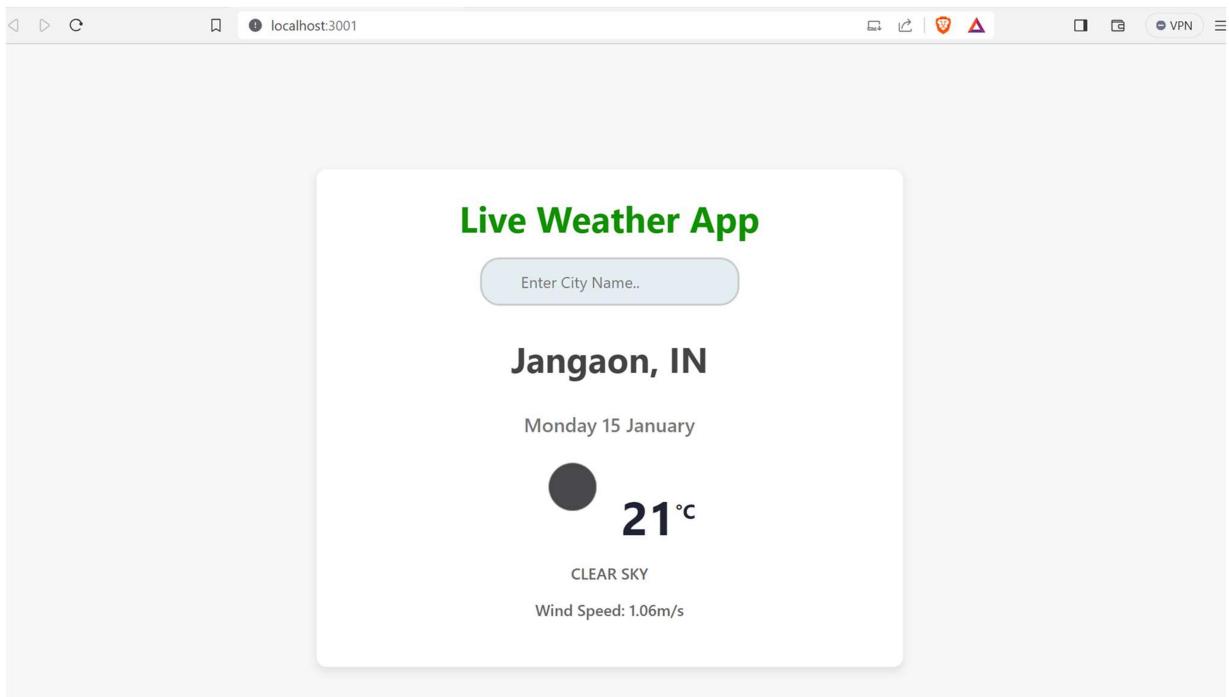
```

```
/* App.css */
* {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
    Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}
html {
    background-color: #f7f7f7;
}
.app-name {
    font-size: 2.3rem;
    color: rgb(17, 144, 0);
    margin-bottom: 16px;
}
.App {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 600px;
    min-height: 440px;
    background-color: rgb(255, 255, 255);
    text-align: center;
    margin: 128px auto;
    border-radius: 10px;
    padding-bottom: 32px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}
.city-search {
    width: 100%;
    max-width: 400px;
    box-sizing: border-box;
    border: 2px solid rgb(204, 204, 204);
    outline: none;
    border-radius: 20px;
    font-size: 16px;
    background-color: #e5eef0;
    background-position: 10px 12px;
    background-repeat: no-repeat;
    padding: 12px 40px 12px 40px;
    -webkit-transition: width 0.4s ease-in-out;
```

```
        transition: width 0.4s ease-in-out;
        color: #333;
    }
    .city-search:focus {
        width: 100%;
    }
    .city-name {
        font-size: 1.5rem;
        color: #444;
        margin-bottom: 8px;
    }
    .date {
        font-size: 1.25em;
        font-weight: 500;
        color: #777;
    }
    .icon-temp {
        font-size: 3rem;
        font-weight: 700;
        color: #1e2432;
        text-align: center;
    }
    .deg {
        font-size: 1.3rem;
        vertical-align: super;
    }
    .des-wind {
        font-weight: 500;
        color: #666;
    }
    .error-message {
        display: block;
        text-align: center;
        color: #d32f2f;
        font-size: 24px;
        margin-top: auto;
    }
    .Loader {
        display: flex;
        justify-content: center;
        align-items: center;
    }
```

```
height: 100%;  
}  
.Loader>div {  
margin: 0 auto;  
}  
.weather-icon {  
display: flex;  
justify-content: center;  
align-items: center;  
}  
.weather-icon img {  
width: 100px;  
height: 100px;  
border-radius: 50%;  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);  
}
```

Output:



Exercise-14:

Create a TODO application in react with necessary components and deploy it into GitHub.

Solution:

Steps to create the Application:

NPX: It is a package runner tool that comes with npm 5.2+, npx is easy to use CLI tools.

- . The npx is used for executing Node packages.

```
npx create-react-app todo-react
```

- . Now, goto the folder

```
cd todo-react
```

- . Install the bootstrap and react-bootstrap module

```
npm install bootstrap
```

```
npm install react-bootstrap
```

// App.js File

```
import "bootstrap/dist/css/bootstrap.css";
import React, { Component } from "react";
import Button from "react-bootstrap/Button";
import Col from "react-bootstrap/Col";
import Container from "react-bootstrap/Container";
import FormControl from "react-bootstrap/FormControl";
import InputGroup from "react-bootstrap/InputGroup";
import ListGroup from "react-bootstrap/ListGroup";
import Row from "react-bootstrap/Row";
```

```
class App extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    // Setting up state
```

```
    this.state = {
```

```
      userInput: "",
```

```
      list: [],
```

```
    };
```

```
}
```

```
  // Set a user input value
```

```
  updateInput(value) {
```

```
    this.setState({
```

```

    userInput: value,
  });
}

// Add item if user input in not empty
addItem() {
  if (this.state.userInput !== "") {
    const userInput = {
      // Add a random id which is used to delete
      id: Math.random(),

      // Add a user value to list
      value: this.state.userInput,
    };

    // Update list
    const list = [...this.state.list];
    list.push(userInput);

    // reset state
    this.setState({
      list,
      userInput: "",
    });
  }
}

// Function to delete item from list use id to delete
deleteItem(key) {
  const list = [...this.state.list];

  // Filter values and leave value which we need to delete
  const updateList = list.filter((item) => item.id !== key);

  // Update list in state
  this.setState({
    list: updateList,
  });
}

editItem = (index) => {

```

```

const todos = [...this.state.list];
const editedTodo = prompt('Edit the todo:');
if (editedTodo !== null && editedTodo.trim() !== '') {
  let updatedTodos = [...todos]
  updatedTodos[index].value= editedTodo
  this.setState({
    list: updatedTodos,
  });
}
}

render() {
  return (
    <Container>
      <Row
        style={{
          display: "flex",
          justifyContent: "center",
          alignItems: "center",
          fontSize: "3rem",
          fontWeight: "bolder",
        }}>
        TODO LIST
      </Row>

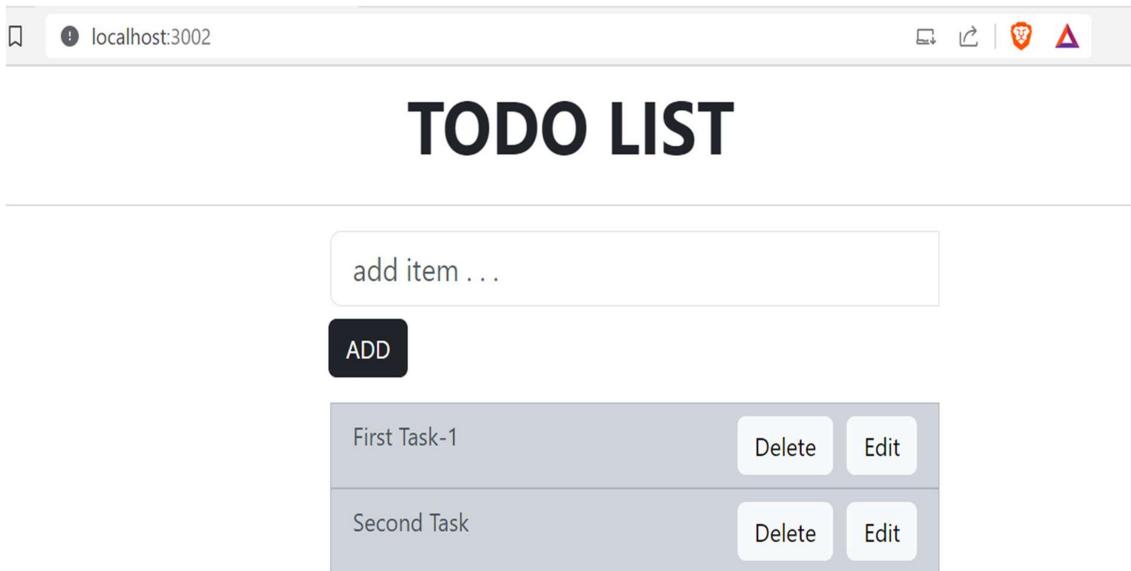
      <hr />
      <Row>
        <Col md={{ span: 5, offset: 4 }}>
          <InputGroup className="mb-3">
            <FormControl
              placeholder="add item . . . "
              size="lg"
              value={this.state.userInput}
              onChange={(item) =>
                this.updateInput(item.target.value)
              }
              aria-label="add something"
              aria-describedby="basic-addon2" />
          <InputGroup>
            <Button
              variant="dark"

```

```
        className="mt-2"
        onClick={() => this.addItem()} >
        ADD
      </Button>
    </InputGroup>
  </InputGroup>
</Col>
</Row>
<Row>
<Col md={{ span: 5, offset: 4 }}>
  <ListGroup>
    {/* map over and print items */}
    {this.state.list.map((item, index) => {
      return (
        <div key = {index} >
          <ListGroup.Item
            variant="dark"
            action
            style={{display:"flex",
              justifyContent:'space-between'
            }}
        >
          {item.value}
          <span>
            <Button style={{marginRight:"10px"}}
              variant = "light"
              onClick={() => this.deleteItem(item.id)}>
              Delete
            </Button>
            <Button variant = "light"
              onClick={() => this.editItem(index)}>
              Edit
            </Button>
          </span>
        </ListGroup.Item>
      </div>
    );
  )})
</ListGroup>
</Col>
</Row>
```

```
</Container>
);
}
}
export default App;
```

Output:



2. Deploying a Django project to GitHub involves several steps.

Below are the general steps to deploy a Django project to a GitHub repository:

Step 1: Create a GitHub Repository

1. Sign in or Sign up:

- If you don't have a GitHub account, sign up for one. If you already have an account, sign in.

2. Create a New Repository:

- Click on the "+" icon in the upper right corner of the GitHub website.
- Select "New Repository."
- Fill in the repository name, description, and other options.
- Optionally, initialize the repository with a README file.
- Click on the "Create repository" button.

Step 2: Initialize a Git Repository Locally

Open a Terminal:

- Open a terminal or command prompt on your local machine.

Navigate to Your Django Project:

- Change your current working directory to your Django project's root folder.

Initialize a Git Repository:

- Run the following commands:

```
git init
```

```
git add .
```

```
git commit -m "Initial commit"
```

Step 3: Connect to the GitHub Repository

1. Copy the Repository URL:

- o On the GitHub repository page, copy the repository URL.
- o It should look like <https://github.com/your-username/your-repository.git>.

2. Add the GitHub Repository as a Remote:

- Run the following command in your terminal, replacing the URL with your repository URL:

```
git remote add origin https://github.com/your-username/your-repository.git
```

3. Push Your Code to GitHub:

- Push your code to the GitHub repository:

```
git push -u origin master
```

Step 4: Configure .gitignore

- Create a `.gitignore` file in your project's root folder to exclude files and directories that shouldn't be tracked by Git.
- For Django projects, common entries include `__pycache__`, `*.pyc`, `*.pyo`, `*.pyd`, `*.sqlite3`, and `venv/` (if you are using a virtual environment).

Step 5: Set Up Environment Variables

- If your Django project uses environment variables for sensitive information (e.g., secret keys, API keys), ensure that you don't include them in your code.
- Use a tool like `python-decouple` or `python-dotenv` to manage environment variables, and add the appropriate entries to your `.env` or `.env.example` file.
- Also, make sure not to push sensitive information to the repository.