1.Design and develop Spring boot application to add, delete, list student records using JPA and MySQL.

Spring Boot Application for Student Management

1. Setup Your Environment

1. Install Java 17 or later.

2. Install an IDE (IntelliJ IDEA or Eclipse).

3. Install MySQL Server.

4. Install Maven (comes with IntelliJ/Eclipse).

2. Create a New Spring Boot Project

1. Go to Spring Initializr (https://start.spring.io/).

2. Select:

  - Project: Maven

  - Language: Java

  - Dependencies: Spring Web, Spring Data JPA, MySQL Driver, Spring Boot DevTools

3. Click 'Generate' and import the project into your IDE.

3. Configure application.properties

In the src/main/resources/application.properties file, add the following:

```
# MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/studentdb
spring.datasource.username=root
spring.datasource.password=your_password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# Server Configuration
server.port=8080
```

4. Create the Database

Run the following SQL command in your MySQL client:

```
CREATE DATABASE studentdb;
```

5. Define the Student Entity

Create a Student class in src/main/java/com/example/demo/entity:

```
package com.example.demo.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
    private int age;

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

6. Create the Repository

Create a StudentRepository interface in src/main/java/com/example/demo/repository:

```
package com.example.demo.repository;

import com.example.demo.entity.Student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StudentRepository extends JpaRepository<Student, Long> {
}
```

7. Create the Service Layer

Create a StudentService class in src/main/java/com/example/demo/service:

```
package com.example.demo.service;

import com.example.demo.entity.Student;
import com.example.demo.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
```

```
public class StudentService {

    @Autowired
    private StudentRepository studentRepository;

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    public Student getStudentById(Long id) {
        return studentRepository.findById(id).orElse(null);
    }

    public Student saveStudent(Student student) {
        return studentRepository.save(student);
    }

    public void deleteStudent(Long id) {
        studentRepository.deleteById(id);
    }
}
```

8. Create the Controller

Create a StudentController class in src/main/java/com/example/demo/controller:

```
package com.example.demo.controller;

import com.example.demo.entity.Student;
import com.example.demo.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/students")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @GetMapping
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    @GetMapping("/{id}")
    public Student getStudentById(@PathVariable Long id) {
        return studentService.getStudentById(id);
    }

    @PostMapping
    public Student createStudent(@RequestBody Student student) {
        return studentService.saveStudent(student);
    }

    @DeleteMapping("/{id}")
    public void deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
    }
}
```

9. Run the Application

1. Start your Spring Boot application by running DemoApplication in src/main/java/com/example/demo.

2. Access the following endpoints using Postman or cURL:

Endpoints

1. Add a Student (POST): http://localhost:8080/students

Body:

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "age": 22
}
```

2. List All Students (GET): http://localhost:8080/students

3. Get a Student by ID (GET): http://localhost:8080/students/{id}

4. Delete a Student (DELETE): http://localhost:8080/students/{id}

---

2. Design and develop PHP application where employee records could be added and employee list could be listed on web page.

PHP Employee Management System

1. Introduction

This document explains how to create a PHP application for managing employee records. The application allows adding employee records and displaying them in a list.

2. Environment Setup

1. Install a local server such as XAMPP, WAMP, or LAMP.
2. Ensure MySQL is installed and running.
3. Create a database named 'employee_management'.

3. Database Configuration

Run the following SQL commands to create the database and employees table:

```sql
CREATE DATABASE employee_management;

USE employee_management;

CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    position VARCHAR(100),
    salary DECIMAL(10, 2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. Project Structure

Create the following files and directories:

```
employee_management/
├── index.php          // Main page to display employee list
├── add_employee.php   // Page to add employee
├── db.php             // Database connection file
├── styles.css         // Optional styling
```

5. Database Connection (db.php)

```php
<?php
$host = "localhost";
$dbname = "employee_management";
$username = "root";
$password = "";

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

6. Employee List Page (index.php)

```php
<?php
include 'db.php';
```

```
$query = "SELECT * FROM employees ORDER BY created_at DESC";
$employees = $conn->query($query)->fetchAll(PDO::FETCH_ASSOC);
?>
<!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>Employee Management</title>
</head>
<body>
    <h1>Employee List</h1>
    <a href="add_employee.php">Add Employee</a>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Email</th>
                <th>Position</th>
                <th>Salary</th>
                <th>Created At</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach ($employees as $employee): ?>
                <tr>
                    <td><?= $employee['id'] ?></td>
                    <td><?= htmlspecialchars($employee['name']) ?></td>
                    <td><?= htmlspecialchars($employee['email']) ?></td>
                    <td><?= htmlspecialchars($employee['position']) ?></td>
                    <td><?= htmlspecialchars($employee['salary']) ?></td>
                    <td><?= htmlspecialchars($employee['created_at']) ?></td>
                </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
</body>
</html>
```

7. Add Employee Page (add_employee.php)

```
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $position = $_POST['position'];
    $salary = $_POST['salary'];

    $stmt = $conn->prepare("INSERT INTO employees (name, email, position, salary) VALUES (?, ?, ?,
?)");
    $stmt->execute([$name, $email, $position, $salary]);

    header("Location: index.php");
    exit();
}
?>
<!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>Add Employee</title>
</head>
<body>
    <h1>Add Employee</h1>
    <form method="POST" action="">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <label for="position">Position:</label>
        <input type="text" id="position" name="position">
        <label for="salary">Salary:</label>
        <input type="number" step="0.01" id="salary" name="salary">
        <button type="submit">Add Employee</button>
    </form>
</body>
</html>
```

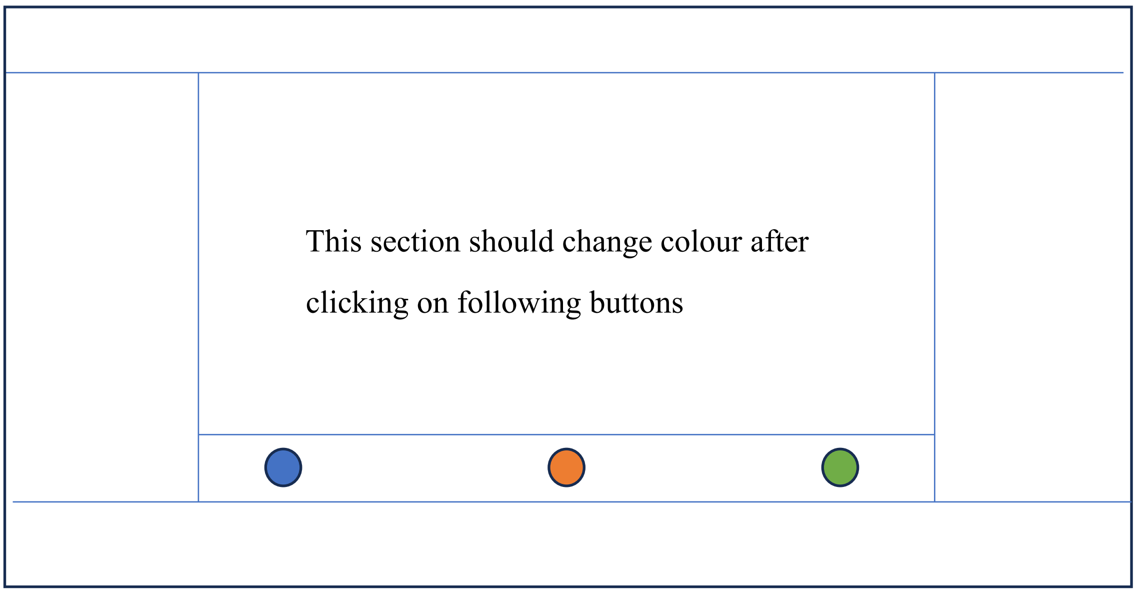
8. Styling (styles.css)

```
body {
    font-family: Arial, sans-serif;
    margin: 20px;
    line-height: 1.6;
}

table {
    width: 100%;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 8px;
    text-align: left;
}

th {
    background-color: #f4f4f4;
}

h1 {
    color: #333;
}

a {
    display: inline-block;
    margin: 10px 0;
    padding: 8px 12px;
    background-color: #007BFF;
    color: white;
    text-decoration: none;
    border-radius: 4px;
}

a:hover {
    background-color: #0056b3;
}

form label {
    display: block;
    margin: 10px 0 5px;
}

form input, form button {
    padding: 8px;
    width: 100%;
    margin-bottom: 10px;
}

form button {
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 4px;
}

form button:hover {
    background-color: #0056b3;
}
```

---

3. Design following responsive layout using html.

Use <header>, <footer>, <div> and appropriate tags


This section should change colour after clicking on following buttons

Steps to Create the Layout

1. **Create an HTML file** named `index.html`.
2. Use the `<header>`, `<footer>`, `<div>`, and appropriate tags to structure the layout.
3. Add **CSS for styling** the layout and making it responsive.
4. Use **JavaScript** to handle button clicks and change the color of the central section.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Responsive Layout</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <header>
      <h1>Responsive Layout Header</h1>
   </header>

   <div class="container">
      <aside class="sidebar"></aside>

      <main class="content" id="content-section">
         <p>This section should change color after clicking the following buttons.</p>
      </main>

      <aside class="sidebar"></aside>
   </div>

   <footer>
      <button class="color-button blue" onclick="changeColor('blue')"></button>
      <button class="color-button orange" onclick="changeColor('orange')"></button>
      <button class="color-button green" onclick="changeColor('green')"></button>
   </footer>

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

css

```
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   display: flex;
   flex-direction: column;
   height: 100vh;
}

header, footer {
   background-color: #f4f4f4;
   text-align: center;
   padding: 10px;
}

.container {
   display: flex;
```

```
    flex: 1;
}

.sidebar {
    flex: 1;
    background-color: #eaeaea;
}

.content {
    flex: 3;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: white;
    border: 1px solid #ccc;
    padding: 20px;
}

footer {
    display: flex;
    justify-content: center;
    gap: 10px;
}

.color-button {
    width: 30px;
    height: 30px;
    border-radius: 50%;
    border: none;
    cursor: pointer;
}

.color-button.blue {
    background-color: blue;
}

.color-button.orange {
    background-color: orange;
}

.color-button.green {
    background-color: green;
}
```

JS

```
function changeColor(color) {
    const contentSection = document.getElementById('content-section');
    contentSection.style.backgroundColor = color;
}
```

4. Develop a responsive web application using PHP/Spring boot and MySQL for restaurant food order management. Make assumption wherever required

Step 1: Set Up the Database

1. Create a MySQL database named `restaurant`.
2. Create the following tables:

SQL Script

```
CREATE DATABASE restaurant;


USE restaurant;


-- Menu Table
CREATE TABLE menu (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL
);


-- Orders Table
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(100) NOT NULL,
    food_id INT,
    quantity INT NOT NULL,
    total_price DECIMAL(10, 2),
    status ENUM('Pending', 'Completed') DEFAULT 'Pending',
    FOREIGN KEY (food_id) REFERENCES menu(id)
```

);

Step 2: Backend Configuration

1. Set up a PHP environment (XAMPP, WAMP, or similar).
2. Create a file `db.php` to handle database connections.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "restaurant";

$conn = new mysqli($servername, $username, $password, $database);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Step 3: Add Food Items (Admin Feature)

Create a file `add_food.php` for adding food items to the menu.

add_food.php

```
php
Copy code
<?php include 'db.php'; ?>

<!DOCTYPE html>
<html>
<head>
    <title>Add Food Item</title>
</head>
<body>
    <h1>Add Food to Menu</h1>
    <form method="POST" action="">
        <label>Food Name:</label>
        <input type="text" name="name" required><br><br>
        <label>Description:</label>
        <textarea name="description" required></textarea><br><br>
        <label>Price:</label>
        <input type="number" step="0.01" name="price" required><br><br>
        <button type="submit" name="submit">Add Food</button>
    </form>

    <?php
    if (isset($_POST['submit'])) {
        $name = $_POST['name'];
        $description = $_POST['description'];
        $price = $_POST['price'];

        $sql = "INSERT INTO menu (name, description, price) VALUES
('$name', '$description', $price)";
        if ($conn->query($sql) === TRUE) {
            echo "Food added successfully!";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
    }
    ?>
</body>
</html>
```

Step 4: Display Menu for Customers

Create `menu.php` to display all food items dynamically.

menu.php

```
php
Copy code
<?php include 'db.php'; ?>

<!DOCTYPE html>
<html>
<head>
    <title>Food Menu</title>
    <style>
        .menu-item {
            border: 1px solid #ccc;
            padding: 10px;
            margin: 10px;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <h1>Food Menu</h1>
    <?php
    $result = $conn->query("SELECT * FROM menu");
    while ($row = $result->fetch_assoc()) {
        echo "<div class='menu-item'>
                <h3>{$row['name']} - {$row['price']} ₹</h3>
                <p>{$row['description']}</p>
                <a href='order.php?food_id={$row['id']}'>Order Now</a>
              </div>";
    }
    ?>
</body>
</html>
```

Step 5: Place an Order

Create `order.php` to handle food ordering.

order.php

```
php
Copy code
<?php include 'db.php'; ?>

<!DOCTYPE html>
<html>
<head>
    <title>Place Order</title>
</head>
<body>
    <?php
    $food_id = $_GET['food_id'];
    $food = $conn->query("SELECT * FROM menu WHERE id=$food_id")-
>fetch_assoc();
    ?>
    <h1>Order <?php echo $food['name']; ?></h1>
    <form method="POST" action="">
        <label>Your Name:</label>
        <input type="text" name="customer_name" required><br><br>
        <label>Quantity:</label>
        <input type="number" name="quantity" required><br><br>
        <button type="submit" name="submit">Place Order</button>
    </form>

    <?php
    if (isset($_POST['submit'])) {
        $customer_name = $_POST['customer_name'];
        $quantity = $_POST['quantity'];
        $total_price = $quantity * $food['price'];

        $sql = "INSERT INTO orders (customer_name, food_id, quantity,
total_price)
                VALUES ('$customer_name', $food_id, $quantity,
$total_price)";
        if ($conn->query($sql) === TRUE) {
            echo "Order placed successfully!";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
    }
    ?>
</body>
</html>
```

Step 6: View Orders (Admin Panel)

Create `orders.php` to list all orders and update their status.

orders.php

```
php
Copy code
<?php include 'db.php'; ?>

<!DOCTYPE html>
<html>
<head>
    <title>Orders</title>
</head>
<body>
    <h1>All Orders</h1>
    <table border="1">
        <tr>
            <th>Order ID</th>
            <th>Customer Name</th>
            <th>Food Item</th>
            <th>Quantity</th>
            <th>Total Price</th>
            <th>Status</th>
            <th>Action</th>
        </tr>
        <?php
        $result = $conn->query("SELECT orders.*, menu.name AS food_name
                                FROM orders JOIN menu ON orders.food_id =
menu.id");
        while ($row = $result->fetch_assoc()) {
            echo "<tr>
                    <td>{$row['id']}</td>
                    <td>{$row['customer_name']}</td>
                    <td>{$row['food_name']}</td>
                    <td>{$row['quantity']}</td>
                    <td>{$row['total_price']} ₹</td>
                    <td>{$row['status']}</td>
                    <td>
                        <form method='POST'>
                            <input type='hidden' name='order_id'
value='{$row['id']}'>
                            <button type='submit' name='complete'>Mark as
Completed</button>
                        </form>
                    </td>
                  </tr>";
        }

        if (isset($_POST['complete'])) {
            $order_id = $_POST['order_id'];
            $conn->query("UPDATE orders SET status='Completed' WHERE
id=$order_id");
            echo "<script>window.location.reload();</script>";
        }
        ?>
    </table>
</body>
</html>
```

---

3. Test the Application

1. Add food items using `add_food.php`.
2. View the menu on `menu.php`.

3. Place orders using `order.php`.
4. View and update orders using `orders.php`.

---

1. 5. Develop a currency converter application using ReactJS that allows users to input an amount dollar and convert it to rupees. In this problem, you can use a hard-coded exchange rate. Take advantage of React state and event handlers to manage the input and conversion calculations.

   Setup React Project
    - Initialize a React project using `create-react-app` or any preferred method.
    - Install dependencies if needed (e.g., React-Bootstrap for styling).
2. **Create Application Components**
    - Build a single `CurrencyConverter` component for handling input and displaying the conversion.
3. **Implement State Management**
    - Use the `useState` hook to manage the input amount and converted value.
4. **Add Event Handlers**
    - Handle input changes and calculate the conversion dynamically based on the hard-coded exchange rate.
5. **Style the Application**
    - Use simple CSS or a library like Bootstrap to style the interface.
6. **Run and Test**
    - Start the application and test for various inputs.

Code

Here’s the complete implementation:

App.js

```
javascript
Copy code
import React, { useState } from "react";
import "./App.css";

function CurrencyConverter() {
  const [dollars, setDollars] = useState(""); // State for input in dollars
  const [rupees, setRupees] = useState("");  // State for converted value
  const exchangeRate = 83.50; // Hard-coded exchange rate (1 USD = 83.50
INR)

  // Handle input changes
  const handleInputChange = (e) => {
    const inputAmount = e.target.value;
    // Check if the input is a valid number or empty
    if (!isNaN(inputAmount)) {
      setDollars(inputAmount); // Update dollar value
      setRupees((inputAmount * exchangeRate).toFixed(2)); // Convert to
rupees
    }
  };

  return (
    <div className="currency-converter">
      <h1>Currency Converter</h1>
      <label htmlFor="dollar-input">Enter Amount in Dollars (USD):</label>
      <input
        type="text"
        id="dollar-input"
        placeholder="e.g., 10"
        value={dollars}
        onChange={handleInputChange}
      />
      <h2>Converted Amount: ₹{rupees || "0.00"}</h2>
    </div>
  );
}

function App() {
  return (
    <div className="App">
      <CurrencyConverter />
    </div>
  );
}

export default App;
```

App.css

css
Copy code

```css
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f9;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.currency-converter {
  background: #ffffff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  text-align: center;
  width: 300px;
}

h1 {
  color: #333;
  font-size: 1.8rem;
}

label {
  display: block;
  margin: 15px 0 5px;
  color: #555;
}

input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 1rem;
  margin-bottom: 20px;
}

h2 {
  color: #007bff;
  font-size: 1.5rem;
}
```

Running the Application

1. **Start the Application**
   - Run `npm start` in the terminal.
2. **Test Features**
   - Enter different amounts in the input field and check the converted value in rupees.
3. **Modify as Needed**
   - Adjust the exchange rate or enhance the UI with more features like conversion for other currencies.

---

6. Develop a currency converter application using PHP that allows users to input an amount dollar and convert it to rupees. This problem, you can use a hard-coded exchange rate.

1. **Set Up the Environment**
   - Install a local server like XAMPP or WAMP to run PHP scripts.
   - Create a directory for your project, e.g., `currency-converter`.
2. **Create an HTML Form**
   - Add an input field for entering the amount in dollars and a submit button.
3. **Process the Form in PHP**

   - Use PHP to retrieve the input amount, calculate the conversion, and display the result.
4. **Style the Page**
   - Use CSS for basic styling.
5. **Test the Application**
   - Run the script and verify the conversion works as expected.

Code

index.php

```
php
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Currency Converter</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .converter {
            background: #ffffff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
            text-align: center;
            width: 300px;
        }
        h1 {
            color: #333;
            font-size: 1.8rem;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            color: #555;
        }
        input[type="text"], input[type="submit"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ddd;
            border-radius: 4px;
            font-size: 1rem;
        }
        input[type="submit"] {
            background-color: #007bff;
            color: white;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #0056b3;
        }
        .result {
            font-size: 1.5rem;
            color: #007bff;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="converter">
        <h1>Currency Converter</h1>
        <form method="POST" action="">
            <label for="dollar">Enter Amount in Dollars (USD):</label>
            <input type="text" id="dollar" name="dollar" placeholder="e.g.,
10" required>
            <input type="submit" value="Convert to INR">
        </form>

        <?php
        if ($_SERVER["REQUEST_METHOD"] === "POST") {
            $dollar = $_POST['dollar'];
            $exchangeRate = 83.50; // Hard-coded exchange rate
            if (is_numeric($dollar)) {
                $rupees = $dollar * $exchangeRate;
                echo "<div class='result'>₹" . number_format($rupees, 2) .
"</div>";
            } else {
                echo "<div class='result'>Please enter a valid
number.</div>";
            }
        }
        ?>
    </div>
</body>
</html>
```

How It Works

1. **HTML Form**
    - The form accepts user input for the dollar amount and submits it using the POST method.
2. **PHP Script**
    - Retrieves the input amount using `$_POST`.
    - Validates that the input is numeric.
    - Calculates the rupee equivalent using a hard-coded exchange rate.
    - Displays the converted value or an error message.
3. **Styling**
    - The page is styled with a clean and minimal design using inline CSS.

Running the Application

1. Save the code as `index.php` in the `htdocs` folder (XAMPP) or your project directory.
2. Start your local server.
3. Open a browser and navigate to `http://localhost/currency-converter/index.php`.
4. Enter a dollar amount and click **Convert to INR**.

---

7. Design and develop a chessboard. The board should be alternating colours and an eight-by-eight grid. Use <header>, <footer>, <body>, <div>, <table> and other tags. Chessboard must be responsive in nature.

1. **Setup the HTML Structure**
   - Use semantic HTML tags like `<header>`, `<footer>`, `<div>`, and `<table>`.
   - Structure the chessboard inside a `<table>` tag.
2. **Style the Chessboard**
   - Use CSS to apply alternating colors to the chessboard cells.
   - Use `nth-child` selectors for alternating colors.
   - Ensure the chessboard is square and scales responsively.
3. **Make it Responsive**
   - Use CSS properties like `max-width`, `width`, and `height` with percentages and `vh`/`vw`.
   - Use `@media` queries to handle different screen sizes.
4. **Test the Design**
   - Check responsiveness across devices and screen sizes.

Code

`index.html`

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Chessboard</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
```

```
        <h1>Responsive Chessboard</h1>
    </header>
    <main>
        <div class="chessboard-container">
            <table class="chessboard">
                <!-- Generate chessboard rows -->
                <?php for ($row = 0; $row < 8; $row++): ?>
                    <tr>
                        <!-- Generate chessboard columns -->
                        <?php for ($col = 0; $col < 8; $col++): ?>
                            <td class="<?= ($row + $col) % 2 === 0 ?
'white' : 'black' ?>"></td>
                        <?php endfor; ?>
                    </tr>
                <?php endfor; ?>
            </table>
        </div>
    </main>
    <footer>
        <p>© 2024 Chessboard Design</p>
    </footer>
</body>
</html>
```

---

styles.css

```
css
Copy code
/* Reset and global styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
}

header {
    text-align: center;
    margin-bottom: 20px;
}

h1 {
    font-size: 2rem;
    color: #333;
}

/* Chessboard container */
.chessboard-container {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 90vw;
    max-width: 600px;
    aspect-ratio: 1 / 1;
}

/* Chessboard styles */
.chessboard {
    width: 100%;
    height: 100%;
    border-collapse: collapse;
    border: 2px solid #333;
}

.chessboard td {
    width: 12.5%;
    height: 12.5%;
    padding: 0;
}

.chessboard td.white {
    background-color: #fff;
}

.chessboard td.black {
    background-color: #000;
}

/* Footer styles */
footer {
    text-align: center;
    margin-top: 20px;
    color: #666;
}

@media (max-width: 768px) {
    h1 {
        font-size: 1.5rem;
    }

    footer p {
        font-size: 0.9rem;
    }
}
```

How It Works

1. **HTML Structure**
   - The chessboard is constructed using a PHP `for` loop to generate rows and cells dynamically.
   - Each cell alternates between white and black based on its position using modulo arithmetic.
2. **CSS Styling**
   - The `chessboard` table uses alternating classes (`white` and `black`) to style the cells.
   - A responsive design is achieved using the `aspect-ratio` property and percentage widths.
3. **Responsiveness**

- The chessboard adjusts its size dynamically, maintaining its square shape using `aspect-ratio: 1 / 1`.
- The width is limited to `90vw` for responsiveness.

Testing the Design

1. Open the HTML file in a browser.
2. Resize the browser window or test on different devices.
3. Verify the chessboard adjusts properly while maintaining its square shape and alternating colors.

---

8. Write React application for registering complaint for students in college. Use React, NodeJS and MySQL/MongoDB for frontend and backend.

a) create login page for student

b) create complaint page

c) create login page for admin

d) list all complaints on admin login

1. Setting Up Backend (Node.js with MongoDB)

Step 1.1: Install Dependencies

First, you need to initialize a new Node.js project and install the required dependencies.

```
bash
Copy code
mkdir complaint-system-backend
cd complaint-system-backend
npm init -y
npm install express mongoose bcryptjs jsonwebtoken cors dotenv
```

Step 1.2: Create MongoDB Atlas Cluster

- Create a MongoDB Atlas account (https://www.mongodb.com/cloud/atlas).
- Create a cluster and get your connection string from the MongoDB Atlas dashboard.
- Make sure to add a `students` collection and a `complaints` collection.

Step 1.3: Create Backend Files

In the `complaint-system-backend` directory, create the following files:

- `server.js`
- `models/student.js`

- `models/complaint.js`
- `routes/studentRoutes.js`
- `routes/complaintRoutes.js`
- `.env` (for storing your MongoDB Atlas URI)

`server.js` (Backend Entry Point)

js
Copy code
```js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

// Middleware
app.use(cors());
app.use(express.json());

// Connect to MongoDB
mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch((err) => console.log(err));

// Routes
const studentRoutes = require('./routes/studentRoutes');
const complaintRoutes = require('./routes/complaintRoutes');
app.use('/api/students', studentRoutes);
app.use('/api/complaints', complaintRoutes);

// Start server
app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

`.env` (MongoDB Connection URI)

env
Copy code
```env
MONGODB_URI=your_mongodb_atlas_connection_string_here
```

`models/student.js` (Student Model)

js
Copy code
```js
const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
});

module.exports = mongoose.model('Student', studentSchema);
```

`models/complaint.js` (Complaint Model)

js
Copy code
```js
const mongoose = require('mongoose');
const complaintSchema = new mongoose.Schema({
  studentId: { type: mongoose.Schema.Types.ObjectId, ref: 'Student' },
  complaint: { type: String, required: true },
  date: { type: Date, default: Date.now },
});

module.exports = mongoose.model('Complaint', complaintSchema);
```

`routes/studentRoutes.js` (Student Routes)

```
js
Copy code
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const Student = require('../models/student');
const router = express.Router();

// Register student
router.post('/register', async (req, res) => {
  const { username, password } = req.body;
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(password, salt);

  const newStudent = new Student({ username, password: hashedPassword });
  await newStudent.save();
  res.status(201).json({ message: 'Student registered successfully' });
});

// Login student
router.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const student = await Student.findOne({ username });

  if (!student) return res.status(400).json({ message: 'Invalid
credentials' });

  const isMatch = await bcrypt.compare(password, student.password);
  if (!isMatch) return res.status(400).json({ message: 'Invalid
credentials' });

  const token = jwt.sign({ studentId: student._id }, 'secretkey');
  res.json({ token });
});

module.exports = router;
```

`routes/complaintRoutes.js` (Complaint Routes)

```
js
Copy code
const express = require('express');
const Complaint = require('../models/complaint');
const router = express.Router();

// Submit a complaint
router.post('/submit', async (req, res) => {
  const { studentId, complaint } = req.body;

  const newComplaint = new Complaint({ studentId, complaint });
  await newComplaint.save();
  res.status(201).json({ message: 'Complaint submitted successfully' });
});
```

```
// List all complaints (for admin)
router.get('/admin', async (req, res) => {
  const complaints = await Complaint.find().populate('studentId',
'username');
  res.json(complaints);
});

module.exports = router;
```

---

2. Setting Up Frontend (React)

Step 2.1: Create React App

Create a new React app in a new directory:

```
bash
Copy code
npx create-react-app complaint-system-frontend
cd complaint-system-frontend
npm install axios react-router-dom
```

Step 2.2: Create Components

- **Login Page for Student (`StudentLogin.js`)**
- **Complaint Submission Page (`ComplaintPage.js`)**
- **Admin Login Page (`AdminLogin.js`)**
- **Admin Dashboard (`AdminDashboard.js`)**

`src/components/StudentLogin.js`

```
jsx
Copy code
import React, { useState } from 'react';
import axios from 'axios';
import { useHistory } from 'react-router-dom';

const StudentLogin = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const history = useHistory();

  const handleLogin = async () => {
    try {
      const response = await
axios.post('http://localhost:5000/api/students/login', { username, password
});
      localStorage.setItem('token', response.data.token);
      history.push('/complaint');
    } catch (err) {
      console.error(err);
      alert('Invalid credentials');
    }
  };

  return (
    <div>
      <h2>Student Login</h2>
      <input type="text" placeholder="Username" onChange={(e) =>
setUsername(e.target.value)} />
      <input type="password" placeholder="Password" onChange={(e) =>
setPassword(e.target.value)} />
      <button onClick={handleLogin}>Login</button>
    </div>
  );
};

export default StudentLogin;
```

src/components/ComplaintPage.js

jsx
Copy code

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

const ComplaintPage = () => {
  const [complaint, setComplaint] = useState('');

  const handleSubmit = async () => {
    const studentId = localStorage.getItem('studentId');
    const token = localStorage.getItem('token');

    try {
      await axios.post('http://localhost:5000/api/complaints/submit', {
studentId, complaint }, {
        headers: { Authorization: `Bearer ${token}` }
      });
      alert('Complaint submitted successfully');
    } catch (err) {
      alert('Failed to submit complaint');
    }
  };

  return (
    <div>
      <h2>Submit a Complaint</h2>
      <textarea onChange={(e) => setComplaint(e.target.value)} />
      <button onClick={handleSubmit}>Submit Complaint</button>
    </div>
  );
};

export default ComplaintPage;
```

src/components/AdminLogin.js

jsx
Copy code

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

const AdminLogin = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    try {
      // Assume admin credentials are hardcoded for this demo
      const response = await
axios.post('http://localhost:5000/api/students/login', { username, password
});
      localStorage.setItem('token', response.data.token);
      // Redirect to Admin Dashboard
    } catch (err) {
      console.error(err);
      alert('Invalid credentials');
    }
  };

  return (
    <div>
      <h2>Admin Login</h2>
      <input type="text" placeholder="Username" onChange={(e) =>
setUsername(e.target.value)} />
      <input type="password" placeholder="Password" onChange={(e) =>
setPassword(e.target.value)} />
      <button onClick={handleLogin}>Login</button>
    </div>
  );
};

export default AdminLogin;
```

src/components/AdminDashboard.js

```
jsx
Copy code
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const AdminDashboard = () => {
  const [complaints, setComplaints] = useState([]);

  useEffect(() => {
    const fetchComplaints = async () => {
      const response = await
axios.get('http://localhost:5000/api/complaints/admin');
      setComplaints(response.data);
    };

    fetchComplaints();
  }, []);

  return (
    <div>
      <h2>All Complaints</h2>
      {complaints.map((complaint) => (
        <div key={complaint._id}>
          <p>Student: {complaint.studentId.username}</p>
          <p>Complaint: {complaint.complaint}</p>
        </div>
      ))}
    </div>
  );
};

export default AdminDashboard;
```

Step 2.3: Routing and App Setup

In your `src/App.js`, set up routing to navigate between different pages:

```
jsx
Copy code
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import StudentLogin from './components/StudentLogin';
import ComplaintPage from './components/ComplaintPage';
import AdminLogin from './components/AdminLogin';
import AdminDashboard from './components/AdminDashboard';

function App() {
  return (
    <Router>
      <Switch>
        <Route path="/" exact component={StudentLogin} />
        <Route path="/complaint" component={ComplaintPage} />
        <Route path="/admin/login" component={AdminLogin} />
        <Route path="/admin/dashboard" component={AdminDashboard} />
      </Switch>
    </Router>
  );
}

export default App;
```

---

9. Create web page for calculator using HTML, JavaScript and CSS. It should have basic functions like +, -, *, / and %. Use appropriate tags like <table>, <div>, <header>, <section>, <footer>

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Calculator</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <header>

    <h1>Calculator</h1>
```

```
</header>

<section class="calculator">
  <table>
    <tr>
      <td colspan="4">
        <input type="text" id="display" disabled>
      </td>
    </tr>
    <tr>
      <td><button onclick="appendToDisplay('1')">1</button></td>
      <td><button onclick="appendToDisplay('2')">2</button></td>
      <td><button onclick="appendToDisplay('3')">3</button></td>
      <td><button onclick="appendToDisplay('+')">+</button></td>
    </tr>
    <tr>
      <td><button onclick="appendToDisplay('4')">4</button></td>
      <td><button onclick="appendToDisplay('5')">5</button></td>
      <td><button onclick="appendToDisplay('6')">6</button></td>
      <td><button onclick="appendToDisplay('-')">-</button></td>
    </tr>
    <tr>
      <td><button onclick="appendToDisplay('7')">7</button></td>
      <td><button onclick="appendToDisplay('8')">8</button></td>
      <td><button onclick="appendToDisplay('9')">9</button></td>
      <td><button onclick="appendToDisplay('*')">*</button></td>
    </tr>
    <tr>
      <td><button onclick="appendToDisplay('0')">0</button></td>
      <td><button onclick="clearDisplay()">C</button></td>
```

```
        <td><button onclick="calculateResult()">=</button></td>
        <td><button onclick="appendToDisplay('/')">/</button></td>
      </tr>
      <tr>
        <td><button onclick="appendToDisplay('%')">%</button></td>
      </tr>
    </table>
  </section>

  <footer>
    <p>&copy; 2024 Calculator App</p>
  </footer>

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

CSS

```
/* style.css */
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  background-color: #f4f4f4;
}

header {
  text-align: center;
```

```css
  margin-bottom: 20px;
}

h1 {
  font-size: 2rem;
  color: #333;
}

.calculator {
  background-color: #fff;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

table {
  width: 100%;
  border-spacing: 10px;
}

button {
  width: 100%;
  height: 50px;
  font-size: 1.2rem;
  border: none;
  background-color: #f1f1f1;
  cursor: pointer;
  transition: background-color 0.3s;
}
```

```
button:hover {
  background-color: #ddd;
}

#display {
  width: 100%;
  height: 50px;
  font-size: 2rem;
  text-align: right;
  padding: 10px;
  margin-bottom: 20px;
  border: 2px solid #ddd;
  border-radius: 5px;
}

footer {
  text-align: center;
  margin-top: 20px;
}

footer p {
  font-size: 1rem;
  color: #555;
}
```

script

```
// script.js

let display = document.getElementById('display');

function appendToDisplay(value) {
  display.value += value;
}

function clearDisplay() {
  display.value = '';
}

function calculateResult() {
  try {
    // Evaluate the expression entered in the display
    display.value = eval(display.value);
  } catch (error) {
    display.value = 'Error';
  }
}
```

---

10. Write a PHP script to: -
a) transform a string all uppercase letters.
b) transform a string all lowercase letters.
c) make a string's first character uppercase.
d) make a string's first character of all the words uppercase.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>String Transformation</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            padding: 20px;
            background-color: #f4f4f4;
```

```
}
.container {
   background-color: white;
   border-radius: 8px;
   padding: 20px;
   box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
   max-width: 500px;
   margin: 0 auto;
}
input[type="text"], button {
   width: 100%;
   padding: 10px;
   margin: 10px 0;
   font-size: 1.1rem;
   border-radius: 5px;
   border: 1px solid #ddd;
}
button {
   background-color: #4CAF50;
   color: white;
   border: none;
   cursor: pointer;
}
button:hover {
   background-color: #45a049;
}
.output {
   background-color: #f8f8f8;
   padding: 10px;
   border: 1px solid #ddd;
```

```
            margin-top: 10px;
            font-size: 1.1rem;
        }
    </style>
</head>
<body>

    <div class="container">
        <h2>String Transformation</h2>
        <form method="POST">
            <label for="stringInput">Enter a string:</label>
            <input type="text" name="string" id="stringInput" required>

            <button type="submit" name="transform">Transform</button>
        </form>

        <?php
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            // Input string from the user
            $string = $_POST['string'];

            // a) Transform all letters to uppercase
            $uppercase = strtoupper($string);

            // b) Transform all letters to lowercase
            $lowercase = strtolower($string);

            // c) Make the first character uppercase
            $capitalizedFirst = ucfirst($string);
            // d) Make the first character of all words uppercase
            $capitalizedWords = ucwords($string);
            ?>

            <div class="output">
                <h3>Results:</h3>
                <p><strong>Uppercase:</strong> <?php echo $uppercase; ?></p>
                <p><strong>Lowercase:</strong> <?php echo $lowercase; ?></p>
                <p><strong>First character uppercase:</strong> <?php echo $capitalizedFirst;
?></p>
                <p><strong>First character of each word uppercase:</strong> <?php echo
$capitalizedWords; ?></p>
            </div>

            <?php
        }
        ?>
    </div>

</body>
</html>
```

---

11. Write web application for registering complaint for students in college. Use PHP and MySQL for frontend and backend.

a) create login page for student

b) create complaint page

c) create login page for admin

d) list all complaints on admin login

1. Create Database:

```
sql
Copy code
CREATE DATABASE college_complaints;

USE college_complaints;

CREATE TABLE students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE complaints (
    complaint_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT NOT NULL,
    complaint_text TEXT NOT NULL,
    complaint_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);
```

2. Create `config.php` for Database Connection:

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'college_complaints';
$username = 'root'; // Use your MySQL username
$password = ''; // Use your MySQL password

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

3. Create Student Login Page (`student_login.php`):

```
php
Copy code
<?php
session_start();
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    include 'config.php';

    $username = $_POST['username'];
    $password = $_POST['password'];

    $stmt = $conn->prepare("SELECT * FROM students WHERE username =
:username");
    $stmt->bindParam(':username', $username);
    $stmt->execute();
    $student = $stmt->fetch(PDO::FETCH_ASSOC);
    if ($student && password_verify($password, $student['password'])) {
        $_SESSION['student_id'] = $student['student_id'];
        header('Location: complaint_page.php');
        exit();
    } else {
        echo "Invalid login credentials.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Login</title>
</head>
<body>
    <form method="POST" action="">
        <label for="username">Username:</label>
        <input type="text" name="username" required><br>
        <label for="password">Password:</label>
        <input type="password" name="password" required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

4. Create Complaint Page for Student (`complaint_page.php`):

```
php
Copy code
<?php
session_start();
if (!isset($_SESSION['student_id'])) {
    header('Location: student_login.php');
    exit();
}

include 'config.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $student_id = $_SESSION['student_id'];
    $complaint_text = $_POST['complaint_text'];

    $stmt = $conn->prepare("INSERT INTO complaints (student_id,
complaint_text) VALUES (:student_id, :complaint_text)");
    $stmt->bindParam(':student_id', $student_id);
    $stmt->bindParam(':complaint_text', $complaint_text);
    $stmt->execute();

    echo "Complaint submitted successfully.";
}
?>

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

```
<body>
    <h1>Register Complaint</h1>
    <form method="POST" action="">
        <textarea name="complaint_text" required></textarea><br>
        <button type="submit">Submit Complaint</button>
    </form>
</body>
</html>
```

5. Create Admin Login Page (`admin_login.php`):

```
php
Copy code
<?php
session_start();
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Use hardcoded admin credentials or a database for authentication
    $admin_username = 'admin';
    $admin_password = 'admin123'; // In a real app, store securely in the
database

    if ($_POST['username'] == $admin_username && $_POST['password'] ==
$admin_password) {
        $_SESSION['admin_logged_in'] = true;
        header('Location: admin_dashboard.php');
        exit();
    } else {
        echo "Invalid login credentials.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Login</title>
</head>
<body>
    <form method="POST" action="">
        <label for="username">Username:</label>
        <input type="text" name="username" required><br>
        <label for="password">Password:</label>
        <input type="password" name="password" required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

6. Create Admin Dashboard Page to List Complaints (`admin_dashboard.php`):

```
php
Copy code
<?php
session_start();
if (!isset($_SESSION['admin_logged_in'])) {
    header('Location: admin_login.php');
    exit();
}

include 'config.php';
```

```
$stmt = $conn->prepare("SELECT complaints.complaint_text,
students.username, complaints.complaint_date
                       FROM complaints
                       JOIN students ON complaints.student_id =
students.student_id");
$stmt->execute();
$complaints = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
</head>
<body>
    <h1>All Complaints</h1>
    <table border="1">
        <tr>
            <th>Username</th>
            <th>Complaint</th>
            <th>Date</th>
        </tr>
        <?php foreach ($complaints as $complaint): ?>
            <tr>
                <td><?php echo htmlspecialchars($complaint['username']);
?></td>
                <td><?php echo
htmlspecialchars($complaint['complaint_text']); ?></td>
                <td><?php echo $complaint['complaint_date']; ?></td>
            </tr>
        <?php endforeach; ?>
    </table>
</body>
</html>
```

12. Design and develop PHP application to add, delete, list student records use CSS for styling and JavaScript for validating form.

1. Create Database and Table:

```
sql
Copy code
CREATE DATABASE student_records;

USE student_records;

CREATE TABLE students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL
);
```

2. Create `config.php` for Database Connection:

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'student_records';
$username = 'root'; // Use your MySQL username
$password = ''; // Use your MySQL password

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

3. Create Student Add Form (`add_student.php`):

```
php
Copy code
<?php
include 'config.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $age = $_POST['age'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];

    $stmt = $conn->prepare("INSERT INTO students (name, age, email, phone)
VALUES (:name, :age, :email, :phone)");
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':age', $age);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':phone', $phone);
    $stmt->execute();

    echo "Student added successfully.";
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add Student</title>
    <link rel="stylesheet" href="styles.css">
    <script src="validate.js"></script>
</head>
<body>
    <h2>Add Student</h2>
    <form method="POST" action="" onsubmit="return validateForm()">
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" required><br>

        <label for="age">Age:</label>
        <input type="number" name="age" id="age" required><br>
        <label for="email">Email:</label>
        <input type="email" name="email" id="email" required><br>

        <label for="phone">Phone:</label>
        <input type="text" name="phone" id="phone" required><br>

        <button type="submit">Add Student</button>
    </form>
</body>
</html>
```

4. Create Student List Page (`list_students.php`):

```
php
Copy code
<?php
include 'config.php';

$stmt = $conn->prepare("SELECT * FROM students");
$stmt->execute();
$students = $stmt->fetchAll(PDO::FETCH_ASSOC);

if ($_SERVER['REQUEST_METHOD'] == 'GET' && isset($_GET['delete'])) {
    $id = $_GET['delete'];
    $deleteStmt = $conn->prepare("DELETE FROM students WHERE student_id =
:id");
    $deleteStmt->bindParam(':id', $id);
    $deleteStmt->execute();
    header('Location: list_students.php');
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>List Students</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h2>Student Records</h2>
    <table border="1">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Email</th>
            <th>Phone</th>
            <th>Action</th>
        </tr>
        <?php foreach ($students as $student): ?>
            <tr>
                <td><?php echo htmlspecialchars($student['name']); ?></td>
                <td><?php echo $student['age']; ?></td>
                <td><?php echo htmlspecialchars($student['email']); ?></td>
                <td><?php echo htmlspecialchars($student['phone']); ?></td>
                <td><a href="?delete=<?php echo $student['student_id'];
?>">Delete</a></td>
            </tr>
        <?php endforeach; ?>
    </table>
    <a href="add_student.php">Add New Student</a>
</body>
</html>
```

5. Create Validation JavaScript (`validate.js`):

javascript
Copy code
```
function validateForm() {
    let name = document.getElementById('name').value;
    let age = document.getElementById('age').value;
    let email = document.getElementById('email').value;
    let phone = document.getElementById('phone').value;

    if (name == "" || age == "" || email == "" || phone == "") {
        alert("All fields must be filled out");
        return false;
    }

    if (isNaN(age)) {
        alert("Age must be a number");
        return false;
    }

    if (!validateEmail(email)) {
        alert("Invalid email format");
        return false;
    }

    return true;
}

function validateEmail(email) {
    let pattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
    return pattern.test(email);
}
```

6. Create CSS Styling (`styles.css`):

css
Copy code
```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
}

h2 {
    color: #4CAF50;
}

form {
    margin: 20px 0;
}

label {
    margin: 5px 0;
    display: block;
}

input {
    padding: 10px;
    margin: 5px 0;
    width: 300px;
}

button {
    padding: 10px 15px;
    background-color: #4CAF50;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

table {
    width: 100%;
    margin: 20px 0;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 10px;
    text-align: left;
}

a {
    color: red;
    text-decoration: none;
}
```

---

13. Demonstrate jQuery for coping contents from one list control to another list. Also demonstrate how to create new element in HTML page using jQuery.

1. Add jQuery to Your HTML File

You can include jQuery from a CDN. Add the following script tag in the `<head>` section of your HTML file:

```
html
Copy code
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

2. HTML Structure for List Controls

```
html
Copy code
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>jQuery List Control</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <style>
        ul {
            list-style-type: none;
            padding: 0;
        }
        li {
            padding: 5px;
        }
    </style>
</head>
<body>
    <h2>Copy Items Between Lists Using jQuery</h2>

    <div>
        <h3>Available Items</h3>
        <ul id="list1">
            <li>Item 1</li>
            <li>Item 2</li>
            <li>Item 3</li>
            <li>Item 4</li>
        </ul>

        <button id="copyBtn">Copy to Selected List</button>
    </div>

    <div>
        <h3>Selected Items</h3>
        <ul id="list2"></ul>
    </div>

    <br>

    <div>
        <h3>Create New Item</h3>
        <input type="text" id="newItem" placeholder="Enter new item">
        <button id="addItemBtn">Add Item</button>
    </div>

    <script>
        // jQuery code to copy items from list1 to list2
        $("#copyBtn").click(function() {
            $("#list1 li:selected").each(function() {
                // Append selected items to list2
                $("#list2").append("<li>" + $(this).text() + "</li>");
            });
        });

        // jQuery code to create a new element in HTML
        $("#addItemBtn").click(function() {
            var newItemText = $("#newItem").val();
            if (newItemText) {
                // Add new item to list1
                $("#list1").append("<li>" + newItemText + "</li>");
                $("#newItem").val(''); // Clear input field
            } else {
                alert("Please enter a valid item.");
            }
        });
    </script>
</body>
</html>
```

Explanation of Key Elements:

1. **HTML Structure:**
   - **List 1 (`#list1`)**: Contains available items.
   - **List 2 (`#list2`)**: Will display the copied items from List 1.
   - **Copy Button**: When clicked, items are copied from List 1 to List 2.
   - **Text Input and Button**: Allow the user to add new items to List 1.
2. **jQuery:**
   - **Copy Items Button** (`#copyBtn`): Copies the selected items from `#list1` to `#list2`. The `click` event handler is attached to the button to trigger the copying action.
   - **Add New Item Button** (`#addItemBtn`): Adds a new item to `#list1` from the text input (`#newItem`) using the `append` method in jQuery.

Steps to Run:

1. Save the code as an `.html` file.
2. Open the file in a web browser.
3. You will see two lists: one with available items and another for selected items.
4. Use the "Copy to Selected List" button to copy items from the first list to the second.
5. Enter a new item in the input field and click the "Add Item" button to add it to the first list.

---

14. Design and develop a responsive website to calculate Electricity bill using Node JS Condition for first 50 units – Rs. 3.50/unit, for next 100 units – Rs. 4.00/unit, for next 100 units – Rs. 5.20/unit and for units above 250 – Rs. 6.50/unit. You can make the use of bootstrap as well as jQuery.

Install Node.js and Express

Make sure you have Node.js installed. Then, initialize a new Node.js project:

```
bash
Copy code
mkdir electricity-bill-calculator
cd electricity-bill-calculator
npm init -y
npm install express
npm install ejs
```

2. Create Project Structure

Create the following directory structure:

```
java
Copy code
electricity-bill-calculator/
├── public/
│   ├── css/
│   │   └── style.css
│   └── js/
│       └── script.js
├── views/
│   └── index.ejs
├── app.js
└── package.json
```

3. Implement `app.js` (Backend Logic in Node.js)

```
javascript
Copy code
// app.js
const express = require('express');
const path = require('path');
const app = express();

// Set EJS as the templating engine
app.set('view engine', 'ejs');

// Serve static files (CSS, JS)
app.use(express.static(path.join(__dirname, 'public')));

// Parse incoming request data
app.use(express.urlencoded({ extended: true }));

// Route for the home page
app.get('/', (req, res) => {
    res.render('index');
});

// Route to calculate electricity bill
app.post('/calculate', (req, res) => {
    const units = parseFloat(req.body.units);
    let bill = 0;

    // Calculate bill based on units
    if (units <= 50) {
        bill = units * 3.50;
    } else if (units <= 150) {
        bill = 50 * 3.50 + (units - 50) * 4.00;
    } else if (units <= 250) {
        bill = 50 * 3.50 + 100 * 4.00 + (units - 150) * 5.20;
    } else {
        bill = 50 * 3.50 + 100 * 4.00 + 100 * 5.20 + (units - 250) * 6.50;
    }

    // Render result page
    res.render('index', { bill: bill.toFixed(2), units: units });
});

// Set the server to listen on port 3000
```

```
app.listen(3000, () => {
    console.log('Server is running on port 3000');
});
```

4. Create `index.ejs` (Frontend - HTML Template)

```
html
Copy code
<!-- views/index.ejs -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Electricity Bill Calculator</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>

<div class="container">
    <h1 class="text-center mt-5">Electricity Bill Calculator</h1>
    <form method="POST" action="/calculate" class="mt-4">
        <div class="mb-3">
            <label for="units" class="form-label">Enter Units Used</label>
            <input type="number" class="form-control" id="units"
name="units" required>
        </div>
        <button type="submit" class="btn btn-primary">Calculate
Bill</button>
    </form>

    <% if (bill) { %>
        <div class="alert alert-success mt-4">
            <strong>Bill for <%= units %> Units:</strong> Rs. <%= bill %>
        </div>
    <% } %>

</div>

<!-- Bootstrap JS & jQuery -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>
</html>
```

5. Create Custom Styles in `style.css` (CSS)

```
css
Copy code
/* public/css/style.css */
body {
    background-color: #f8f9fa;
    font-family: Arial, sans-serif;
}

.container {
    margin-top: 50px;
    max-width: 500px;
}

.alert {
    font-size: 1.2em;
}
```

6. Create Custom JavaScript in `script.js` (Optional for future enhancements)

```
javascript
Copy code
/* public/js/script.js */
// You can add custom JS here for future interactive features if needed.
```

7. Run the Application

1. Make sure you are in the project directory.
2. Start the server:

```
bash
Copy code
node app.js
```

---

15. Design and develop a responsive website to calculate Electricity bill using Spring boot. Condition for first 50 units – Rs. 3.50/unit, for next 100 units – Rs. 4.00/unit, for next 100 units – Rs. 5.20/unit and for units above 250 – Rs. 6.50/unit. You can make the use of bootstrap as well as jQuery.

Setup Spring Boot Project

Create a new Spring Boot project using Spring Initializr or by setting up manually. For this example, we will use Spring Initializr.

Go to Spring Initializr and generate a new project with the following dependencies:

- Spring Web
- Thymeleaf (for rendering HTML templates)
- Spring Boot DevTools (optional, for development convenience)

You can download the zip file, extract it, and open it in your preferred IDE (IntelliJ IDEA, Eclipse, etc.).

2. Create the Project Structure

Your project directory will look like this:

```
css
Copy code
electricity-bill-calculator/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           └── electricity/
│   │   │               ├── ElectricityBillCalculatorApplication.java
│   │   │               ├── controller/
│   │   │               │   └── BillController.java
│   │   ├── resources/
│   │   │   ├── static/
│   │   │   │   ├── css/
│   │   │   │   │   └── style.css
│   │   │   │   └── js/
│   │   │   │       └── script.js
│   │   │   ├── templates/
│   │   │   │   └── index.html
│   │   ├── application.properties
├── pom.xml
```

3. Create the Main Application Class

```
java
Copy code
//
src/main/java/com/example/electricity/ElectricityBillCalculatorApplication.
java

package com.example.electricity;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ElectricityBillCalculatorApplication {

    public static void main(String[] args) {
        SpringApplication.run(ElectricityBillCalculatorApplication.class,
args);
    }
}
```

4. Create the Controller

```
java
Copy code
// src/main/java/com/example/electricity/controller/BillController.java

package com.example.electricity.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class BillController {

    // Show the home page
    @GetMapping("/")
    public String showHomePage() {
        return "index";
    }

    // Handle the bill calculation
    @PostMapping("/calculate")
    public String calculateBill(@RequestParam("units") int units, Model
model) {
        double bill = 0;

        // Calculate bill based on units
        if (units <= 50) {
            bill = units * 3.50;
        } else if (units <= 150) {
            bill = 50 * 3.50 + (units - 50) * 4.00;
        } else if (units <= 250) {
            bill = 50 * 3.50 + 100 * 4.00 + (units - 150) * 5.20;
        } else {
            bill = 50 * 3.50 + 100 * 4.00 + 100 * 5.20 + (units - 250) *
6.50;
        }

        model.addAttribute("bill", String.format("%.2f", bill));
        model.addAttribute("units", units);
        return "index";
    }
}
```

5. Create the `index.html` Template

This template will display the form for users to input their electricity usage and show the calculated bill.

```
html
Copy code
<!-- src/main/resources/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>Electricity Bill Calculator</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="/css/style.css">
</head>
<body>

<div class="container">
    <h1 class="text-center mt-5">Electricity Bill Calculator</h1>
    <form method="POST" action="/calculate" class="mt-4">
        <div class="mb-3">
            <label for="units" class="form-label">Enter Units Used</label>
            <input type="number" class="form-control" id="units"
name="units" required>
        </div>
        <button type="submit" class="btn btn-primary">Calculate
Bill</button>
    </form>

    <!-- Display the calculated bill -->
    <div class="mt-4">
        <h3>Bill for <span class="badge bg-info"><%= units %>
Units</span></h3>
        <div class="alert alert-success">
```

```
            <strong>Amount:</strong> Rs. <span id="bill"><%= bill %></span>
        </div>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

6. Create the Custom Styles

```
css
Copy code
/* src/main/resources/static/css/style.css */
body {
    background-color: #f8f9fa;
    font-family: Arial, sans-serif;
}

.container {
    margin-top: 50px;
    max-width: 500px;
}

.alert {
    font-size: 1.2em;
}
```

7. Create the Custom JavaScript (Optional for future interactive features)

```
javascript
Copy code
/* src/main/resources/static/js/script.js */
// Optional for enhancing interactivity in the future
```

8. Update `application.properties` (Optional)

For configuring the port and other properties in Spring Boot, you can modify `src/main/resources/application.properties` as needed.

Example:

```
properties
Copy code
# Set the port for the application
server.port=8080
```

9. Run the Application

1. Make sure you are in the project directory.
2. Run the Spring Boot application using your IDE or the following command in the terminal:

```
bash
Copy code
mvn spring-boot:run
```

16. Design and develop a responsive web page for your CV using multiple column layouts having video background. You can make the use of bootstrap as well as jQuery.

Setup the Project Directory

Create the following folder structure:

```
arduino
Copy code
cv-website/
├── index.html
├── css/
│   └── style.css
├── js/
│   └── script.js
└── videos/
    └── background.mp4
```

2. HTML (index.html)

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

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="css/style.css">

</head>
<body>

    <!-- Video Background -->
    <div class="video-background">
        <video autoplay muted loop id="bg-video">
            <source src="videos/background.mp4" type="video/mp4">
        </video>
    </div>

    <!-- CV Content -->
    <div class="container">
        <header class="text-center text-white pt-5 pb-3">
            <h1>My CV</h1>
            <p>Your Name - Web Developer</p>
        </header>
        <div class="row">
            <!-- Left Column: Personal Information -->
            <div class="col-lg-4 col-md-6">
                <div class="card">
                    <img src="https://via.placeholder.com/300" class="card-
img-top" alt="profile picture">
                    <div class="card-body">
                        <h5 class="card-title">Personal Information</h5>
                        <ul>
                            <li><strong>Name:</strong> Your Name</li>
                            <li><strong>Email:</strong>
your.email@example.com</li>
                            <li><strong>Phone:</strong> +123 456 7890</li>
                            <li><strong>Location:</strong> City,
Country</li>
                        </ul>
                    </div>
                </div>
            </div>

            <!-- Middle Column: Skills and Experience -->
            <div class="col-lg-4 col-md-6">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Skills & Experience</h5>
                        <p><strong>Skills:</strong> HTML, CSS, JavaScript,
React, Node.js, Python</p>
                        <p><strong>Experience:</strong></p>
                        <ul>
                            <li>Software Developer at XYZ Company</li>
                            <li>Frontend Developer at ABC Ltd.</li>
                        </ul>
                    </div>
                </div>
            </div>

            <!-- Right Column: Education & Hobbies -->
            <div class="col-lg-4 col-md-12">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Education & Hobbies</h5>
                        <p><strong>Education:</strong> Bachelor's Degree in
Computer Science</p>
                        <p><strong>Hobbies:</strong> Coding, Traveling,
Photography, Gaming</p>
                    </div>
                </div>
            </div>
        </div>

        <footer class="text-center text-white mt-5 py-3">
            <p>&copy; 2024 Your Name | All Rights Reserved</p>
        </footer>
    </div>

    <!-- Bootstrap and jQuery JS -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="js/script.js"></script>
</body>
```

```
</html>
```

3. CSS (style.css)

```
css
Copy code
/* General Body Styling */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

/* Video Background */
.video-background {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: -1;
}

#bg-video {
    object-fit: cover;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
}

/* Container */
.container {
    z-index: 10;
    position: relative;
    padding-top: 100px;
}

/* Cards */
.card {
    margin-bottom: 30px;
}

/* Header Styling */
header h1 {
    font-size: 3rem;
    font-weight: 700;
}

header p {
    font-size: 1.25rem;
    color: #fff;
}

/* Footer */
footer {
    background-color: #333;
    color: #fff;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

```
/* Media Queries for Responsiveness */
@media (max-width: 768px) {
    .container {
        padding-top: 50px;
    }

    header h1 {
        font-size: 2.5rem;
    }

    header p {
        font-size: 1rem;
    }

    .card-body ul {
        list-style-type: none;
    }

    .card-body ul li {
        padding-bottom: 10px;
    }
}
```

4. JavaScript (script.js)

```
javascript
Copy code
// For any future interactive features, you can add JS here
$(document).ready(function() {
    // Example: Smooth scroll or animations (if needed)
});
```

5. Video Background

You can use any video file for the background. Save the video in the `videos/` folder and name it `background.mp4`. You can find a free video from online sources, such as Pexels or Coverr, and use it here.

6. How to Run the Application

1. Ensure you have the project structure set up as mentioned above.
2. Open the `index.html` file in a browser to see the web page.

==

17. Design and develop a website using toggleable or dynamic tabs or pills with bootstrap and jQuery to show the relevance of SDP, EDI, DT and Course projects in VIT.

Project Structure

Here is the folder structure for the project:

```
css
Copy code
college-website/
├── index.html
├── css/
│   └── style.css
├── js/
│   └── script.js
└── img/
    └── vit-logo.png
```

2. HTML (index.html)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>VIT College Subjects & Projects</title>

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="css/style.css">

</head>
<body>

    <!-- Header Section -->
    <header class="text-center py-4">
        <img src="img/vit-logo.png" alt="VIT Logo" width="150">
        <h1 class="mt-3">VIT College Subjects & Course Projects</h1>
        <p>Learn about the relevance of various subjects and course
projects in VIT College</p>
    </header>

    <!-- Main Content Section -->
    <div class="container mt-5">

        <!-- Tab Navigation (Pills) -->
        <ul class="nav nav-pills" id="subjectTabs" role="tablist">
            <li class="nav-item" role="presentation">
                <a class="nav-link active" id="sdp-tab" data-bs-
toggle="pill" href="#sdp" role="tab" aria-controls="sdp" aria-
selected="true">SDP (Software Development Practices)</a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="edi-tab" data-bs-toggle="pill"
href="#edi" role="tab" aria-controls="edi" aria-selected="false">EDI
(Entrepreneurship Development & Innovation)</a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="dt-tab" data-bs-toggle="pill"
href="#dt" role="tab" aria-controls="dt" aria-selected="false">DT (Digital
Transformation)</a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="projects-tab" data-bs-toggle="pill"
href="#projects" role="tab" aria-controls="projects" aria-
selected="false">Course Projects</a>
            </li>
        </ul>

        <!-- Tab Content -->
        <div class="tab-content mt-4" id="subjectTabsContent">
            <!-- SDP Tab -->
            <div class="tab-pane fade show active" id="sdp" role="tabpanel"
aria-labelledby="sdp-tab">
                <h3>Software Development Practices (SDP)</h3>
                <p>This subject focuses on the methodologies and practices
involved in software development. It emphasizes teamwork, communication,
and iterative development to produce quality software products.</p>
                <ul>
                    <li>Understanding Agile, Scrum, and Waterfall
models</li>
                    <li>Project planning and task management</li>
                    <li>Version control and team collaboration</li>
                </ul>
            </div>

            <!-- EDI Tab -->
            <div class="tab-pane fade" id="edi" role="tabpanel" aria-
labelledby="edi-tab">
                <h3>Entrepreneurship Development & Innovation (EDI)</h3>
                <p>EDI focuses on nurturing entrepreneurial skills and
fostering innovation. Students learn how to think creatively, develop
business ideas, and bring them to life.</p>
                <ul>
                    <li>Business idea generation and validation</li>
                    <li>Understanding market dynamics and trends</li>
                    <li>Planning and launching a startup</li>
                </ul>
            </div>

            <!-- DT Tab -->
            <div class="tab-pane fade" id="dt" role="tabpanel" aria-
labelledby="dt-tab">
                <h3>Digital Transformation (DT)</h3>
                <p>DT explores how digital technologies are transforming
industries and businesses. The subject covers the latest trends in
digitalization, automation, and the impact on business strategies.</p>
                <ul>
                    <li>Artificial Intelligence and Machine Learning</li>
                    <li>Cloud computing and IoT</li>
                    <li>Digital strategies for business transformation</li>
                </ul>
            </div>

            <!-- Projects Tab -->
            <div class="tab-pane fade" id="projects" role="tabpanel" aria-
labelledby="projects-tab">
                <h3>Course Projects</h3>
                <p>Course projects are an integral part of the curriculum, 
where students apply their knowledge in real-world scenarios, collaborating 
on developing software, business solutions, and digital strategies.</p>
                <ul>
                    <li>Building web and mobile applications</li>
                    <li>Developing business plans and models</li>
                    <li>Digital marketing and SEO strategies</li>
                </ul>
            </div>
        </div>

    </div>

    <!-- Footer Section -->
    <footer class="text-center py-4 mt-5">
        <p>&copy; 2024 VIT College | All Rights Reserved</p>
    </footer>

    <!-- Bootstrap and jQuery JS -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="js/script.js"></script>

</body>
</html>
```

3. CSS (style.css)

```
css
Copy code
/* General Body Styling */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}

/* Header Styling */
header {
    background-color: #343a40;
    color: #fff;
    padding: 30px 0;
}

/* Tabs Navigation */
.nav-pills .nav-link {
    background-color: #343a40;
    color: white;
    border-radius: 0;
    margin-right: 10px;
}

.nav-pills .nav-link.active {
    background-color: #007bff;
    color: white;
}

/* Tab Content */
.tab-content {
    padding: 20px;
    background-color: #ffffff;
    border-radius: 5px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

footer {
    background-color: #343a40;
    color: white;
    padding: 10px 0;
}
```

4. JavaScript (script.js)

```
javascript
Copy code
// Add any additional interactive functionality if required
$(document).ready(function () {
    // Optional: Handle additional tab functionalities or dynamic content
if necessary
});
```

5. How to Run the Application

1. Ensure you have the project structure set up as mentioned above.
2. Save the VIT logo image in the `img/` folder with the filename `vit-logo.png` (you can find a logo online or use a placeholder).
3. Save a sample video or relevant content for the subjects as required.
4. Open the `index.html` file in a browser to view and interact with the website.

---

18. Design and develop a website to demonstrate (a) searching and sorting array for integer elements using JavaScript (b) array for named entities using JavaScript. You can make the use of bootstrap as well as jQuery.

1. **Searching and Sorting Arrays for Integer Elements using JavaScript**
2. **Working with an Array of Named Entities using JavaScript**

This website will provide functionality to search and sort an array of integers and also handle an array of named entities (such as a list of people). We will use **Bootstrap** for styling, **jQuery** for DOM manipulation, and **JavaScript** for the logic of searching and sorting.

1. Project Structure

```
arduino
Copy code
array-demo/
├── index.html
├── css/
│   └── style.css
├── js/
│   └── script.js
└── img/
    └── logo.png (optional)
```

2. HTML (index.html)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Array Demo: Searching and Sorting</title>

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="css/style.css">

</head>
<body>

    <!-- Header Section -->
    <header class="text-center py-5">
        <h1>Array Operations: Searching and Sorting</h1>
        <p>Demonstrating searching and sorting for integer arrays and named
entities using JavaScript.</p>
    </header>

    <!-- Main Content Section -->
    <div class="container">

        <!-- Integer Array Operations -->
        <section class="mt-5">
            <h3>Search and Sort Integer Array</h3>
            <div class="form-group mb-3">
                <label for="integerSearch">Search Integer:</label>
                <input type="number" class="form-control"
id="integerSearch" placeholder="Enter integer to search">
            </div>
            <div class="form-group mb-3">
                <button class="btn btn-primary"
id="searchIntegerBtn">Search Integer</button>
            </div>

            <div id="integerResult"></div>

            <div class="form-group mt-3">
                <button class="btn btn-success" id="sortIntegerBtn">Sort
Array</button>
```

```
            </div>

            <div id="sortedIntegerResult" class="mt-3"></div>
        </section>

        <!-- Named Entity Array Operations -->
        <section class="mt-5">
            <h3>Named Entities</h3>
            <div class="form-group mb-3">
                <label for="entitySearch">Search Name:</label>
                <input type="text" class="form-control" id="entitySearch"
placeholder="Enter name to search">
            </div>
            <div class="form-group mb-3">
                <button class="btn btn-primary" id="searchEntityBtn">Search
Name</button>
            </div>

            <div id="entityResult"></div>

            <div class="form-group mt-3">
                <button class="btn btn-success" id="sortEntityBtn">Sort
Names Alphabetically</button>
            </div>

            <div id="sortedEntityResult" class="mt-3"></div>
        </section>

    </div>

    <!-- Footer Section -->
    <footer class="text-center py-4 mt-5">
        <p>&copy; 2024 Array Demo | All Rights Reserved</p>
    </footer>

    <!-- Bootstrap and jQuery JS -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="js/script.js"></script>

</body>
</html>
```

3. CSS (style.css)

```
css
Copy code
/* Body Styling */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}

/* Header Section */
header {
    background-color: #343a40;
    color: #fff;
    padding: 30px 0;
}
```

```
h1, h3 {
    font-size: 2em;
}

/* Form Section */
.form-group {
    margin-bottom: 15px;
}

/* Footer */
footer {
    background-color: #343a40;
    color: white;
    padding: 10px 0;
}
```

4. JavaScript (script.js)

```
javascript
Copy code
// Integer Array for Searching and Sorting
let integerArray = [12, 45, 67, 34, 23, 89, 90, 11, 56, 78];

// Named Entity Array
let entityArray = ['John Doe', 'Alice Smith', 'Bob Johnson', 'Sara
Williams', 'David Brown', 'Emily Davis'];

// Search Function for Integer Array
$('#searchIntegerBtn').click(function() {
    let searchValue = $('#integerSearch').val();
    let searchResult = integerArray.includes(Number(searchValue)) ?
        `Integer ${searchValue} found in the array.` :
        `Integer ${searchValue} not found in the array.`;
    $('#integerResult').text(searchResult);
});

// Sort Function for Integer Array
$('#sortIntegerBtn').click(function() {
    let sortedArray = [...integerArray].sort((a, b) => a - b);
    $('#sortedIntegerResult').text(`Sorted Integer Array:
${sortedArray.join(', ')}`);
});

// Search Function for Named Entity Array
$('#searchEntityBtn').click(function() {
    let searchValue = $('#entitySearch').val().toLowerCase();
    let result = entityArray.filter(entity =>
entity.toLowerCase().includes(searchValue));
    if (result.length > 0) {
        $('#entityResult').text(`Matching Names: ${result.join(', ')}`);
    } else {
        $('#entityResult').text('No matching names found.');
    }
});

// Sort Function for Named Entity Array
$('#sortEntityBtn').click(function() {
    let sortedEntities = [...entityArray].sort();
    $('#sortedEntityResult').text(`Sorted Names: ${sortedEntities.join(',
')}`);
});
```

5. Explanation of Features

1. **Integer Array Operations:**
   - **Search Integer:** A user can input an integer, and the script will check if it exists in the predefined array (`integerArray`).
   - **Sort Integer Array:** Clicking the button will sort the integer array in ascending order.
2. **Named Entity Array Operations:**
   - **Search Name:** Users can type in a name or part of a name, and the script will display all names in the `entityArray` that contain the input text.
   - **Sort Names Alphabetically:** The names in the `entityArray` are sorted in alphabetical order.

=============================

---

19. Design and develop a responsive website to calculate Electricity bill using Spring boot/React Condition for first 50 units – Rs. 3.50/unit, for next 100 units – Rs. 4.00/unit, for next 100 units – Rs. 5.20/unit and for units above 250 – Rs. 6.50/unit. You can make the use of bootstrap as well as jQuery.

1. Backend - Spring Boot

Steps:

1. Set up a Spring Boot application with basic dependencies.
2. Create an API endpoint to calculate the electricity bill based on the input units.
3. Set up the logic for calculating the bill based on the provided conditions.

Spring Boot Setup

1. **Create a new Spring Boot Project:** You can generate a Spring Boot project using Spring Initializr.
   - Select **Maven** as the project type.
   - Select **Java** as the language.
   - Add dependencies: **Spring Web** and **Spring Boot DevTools**.
2. **Directory Structure:**

```
css
Copy code
electricity-bill-calculator/
```

```
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           └── electricitycalculator/
│   │   │               ├── ElectricityBillCalculatorApplication.java
│   │   │               └── controller/
│   │   │                   └── BillController.java
│   │   ├── resources/
│   │   │   ├── static/
│   │   │   └── application.properties
│   └── pom.xml
│
```

3. **BillController.java** (Controller for Bill Calculation):

```
java
Copy code
package com.example.electricitycalculator.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class BillController {

    @GetMapping("/calculate-bill")
    public double calculateBill(@RequestParam double units) {
        double bill = 0;

        if (units <= 50) {
            bill = units * 3.50;
        } else if (units <= 150) {
            bill = 50 * 3.50 + (units - 50) * 4.00;
        } else if (units <= 250) {
            bill = 50 * 3.50 + 100 * 4.00 + (units - 150) * 5.20;
        } else {
            bill = 50 * 3.50 + 100 * 4.00 + 100 * 5.20 + (units -
250) * 6.50;
        }

        return bill;
    }
}
```

4. **ElectricityBillCalculatorApplication.java** (Main Application Class):

```
java
Copy code
package com.example.electricitycalculator;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ElectricityBillCalculatorApplication {

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

```
SpringApplication.run(ElectricityBillCalculatorApplication.class,
args);
    }
}
```

5. **application.properties** (Spring Boot Configuration):

```
properties
Copy code
server.port=8080
```

6. **Build the Spring Boot Application**:
   - Run the application with `mvn spring-boot:run` or use your IDE to run the `ElectricityBillCalculatorApplication` class.
   - Test the API at `http://localhost:8080/calculate-bill?units=200`.

---

2. Frontend - React

Steps:

1. Set up a React app using **Create React App**.
2. Create a form for inputting the number of units.
3. Display the calculated electricity bill.

React Setup

1. **Create React App:** If you don't already have a React app, you can create one using:

```
bash
Copy code
npx create-react-app electricity-bill-frontend
```

2. **Directory Structure:**

```
java
Copy code
electricity-bill-frontend/
├── src/
│   ├── components/
│   │   └── BillCalculator.js
│   ├── App.js
│   ├── index.js
│   └── styles.css
├── public/
├── package.json
└── node_modules/
```

3. **App.js** (Main Component):

```
javascript
Copy code
```

```
import React, { useState } from 'react';
import './styles.css';
import BillCalculator from './components/BillCalculator';

function App() {
    return (
        <div className="App">
            <header className="App-header">
                <h1>Electricity Bill Calculator</h1>
            </header>
            <BillCalculator />
        </div>
    );
}

export default App;
```

4. **BillCalculator.js** (Component for Calculating the Bill):

```
javascript
Copy code
import React, { useState } from 'react';
import axios from 'axios';

function BillCalculator() {
    const [units, setUnits] = useState('');
    const [bill, setBill] = useState(null);
    const [error, setError] = useState('');

    const handleSubmit = async (e) => {
        e.preventDefault();
        setError('');
        setBill(null);

        try {
            const response = await
axios.get(`http://localhost:8080/calculate-bill?units=${units}`);
            setBill(response.data);
        } catch (err) {
            setError('Failed to calculate bill. Please try again.');
        }
    };

    return (
        <div className="calculator-container">
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label htmlFor="units">Enter Units:</label>
                    <input
                        type="number"
                        id="units"
                        value={units}
                        onChange={(e) => setUnits(e.target.value)}
                        className="form-control"
                        placeholder="Enter number of units"
                        required
                    />
                </div>
                <button type="submit" className="btn btn-
primary">Calculate Bill</button>
```

```
            </form>

            {bill && (
                <div className="result">
                    <h3>Total Bill: Rs. {bill}</h3>
                </div>
            )}

            {error && (
                <div className="error">
                    <p>{error}</p>
                </div>
            )}
        </div>
    );
}

export default BillCalculator;
```

5. **styles.css** (Styling for the React App):

```
css
Copy code
body {
    font-family: Arial, sans-serif;
    background-color: #f8f9fa;
    margin: 0;
    padding: 0;
}

.App {
    text-align: center;
    padding: 20px;
}

.calculator-container {
    width: 50%;
    margin: 0 auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}

.form-group {
    margin-bottom: 20px;
}

.form-control {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    margin: 5px 0;
}

.btn {
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
    border: none;
```

```
            cursor: pointer;
        }

        .btn:hover {
            background-color: #0056b3;
        }

        .result {
            margin-top: 20px;
            font-size: 18px;
            color: green;
        }

        .error {
            margin-top: 20px;
            font-size: 18px;
            color: red;
        }
```

6. **Run the React App:**

```
bash
Copy code
npm start
```

- The React app will be accessible at `http://localhost:3000`.
- Ensure the Spring Boot backend is running at `http://localhost:8080`.

---

20. Design and develop a responsive website to calculate Electricity bill using PHP. Condition for first 50 units – Rs. 3.50/unit, for next 100 units – Rs. 4.00/unit, for next 100 units – Rs. 5.20/unit and for units above 250 – Rs. 6.50/unit. You can make the use of bootstrap as well as jQuery.

Create HTML Form (index.php)

This form allows the user to input the number of units, which will then be passed to the PHP script to calculate the bill.

```
php
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Electricity Bill Calculator</title>
    <!-- Bootstrap CSS -->
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css
" rel="stylesheet">
    <!-- jQuery -->
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <style>
        body {
            background-color: #f4f4f4;
            font-family: Arial, sans-serif;
        }
        .container {
            margin-top: 50px;
        }
        .result {
            margin-top: 30px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-6 offset-md-3">
                <h2 class="text-center">Electricity Bill Calculator</h2>
                <form action="index.php" method="POST" id="billForm">
                    <div class="form-group">
                        <label for="units">Enter Units:</label>
                        <input type="number" class="form-control"
id="units" name="units" placeholder="Enter number of units" required>
                    </div>
                    <button type="submit" class="btn btn-primary btn-
block">Calculate Bill</button>
                </form>

                <?php
                if ($_SERVER["REQUEST_METHOD"] == "POST") {
                    $units = $_POST['units'];
                    $bill = 0;

                    // Electricity bill calculation logic
                    if ($units <= 50) {
                        $bill = $units * 3.50;
                    } elseif ($units <= 150) {
                        $bill = 50 * 3.50 + ($units - 50) * 4.00;
                    } elseif ($units <= 250) {
                        $bill = 50 * 3.50 + 100 * 4.00 + ($units - 150) *
5.20;
                    } else {
                        $bill = 50 * 3.50 + 100 * 4.00 + 100 * 5.20 +
($units - 250) * 6.50;
                    }

                    echo "<div class='result text-center'><h3>Total Bill:
Rs. " . number_format($bill, 2) . "</h3></div>";
                }
                ?>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS and dependencies -->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"><
/script>
</body>
</html>
```

Explanation of Code:

1. **HTML Form**:
   - The form allows users to input the number of electricity units (`<input type="number" id="units">`).
   - It uses the **POST** method to submit the units to the same page (`action="index.php"`).
2. **PHP Logic**:
   - When the form is submitted, PHP processes the input to calculate the bill based on the provided conditions:
     - First 50 units at Rs. 3.50/unit
     - Next 100 units at Rs. 4.00/unit
     - Next 100 units at Rs. 5.20/unit
     - Above 250 units at Rs. 6.50/unit
3. **Responsive Layout**:
   - The page uses **Bootstrap 4** for responsiveness and styling.
   - The form is centered within the page with a clean and professional layout.
4. **jQuery (Optional)**:
   - If you want to enhance the page with additional jQuery functionalities, such as showing a loading spinner while the bill is being calculated, you can use the following snippet:

```
javascript
Copy code
$(document).ready(function(){
    $("#billForm").submit(function(event){
        event.preventDefault();  // Prevents form submission
        var units = $("#units").val();

        if (units) {
            $(".result").html("<h3>Calculating...</h3>");
            $.post("index.php", { units: units }, function(data){
                $(".result").html(data);
            });
        }
    });
});
```

---

21. Design and develop a responsive website to prepare one semester result of VIT students using REACT Spring boot and MySQL. Take any four subjects with MSE Marks (30%) ESE Marks (70%).

Spring Boot Backend

1.1. Set up Spring Boot Project

1. Create a Spring Boot application (use Spring Initializr to generate the project).
   - Dependencies: `Spring Web`, `Spring Data JPA`, `MySQL Driver`.
2. Configure the `application.properties` file to connect to MySQL.

```
properties
Copy code
spring.datasource.url=jdbc:mysql://localhost:3306/student_results
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBD
ialect
```

1.2. Database Schema (MySQL)

```
sql
Copy code
CREATE DATABASE student_results;

USE student_results;

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    roll_number VARCHAR(50) NOT NULL,
    subject1_mse INT,
    subject1_ese INT,
    subject2_mse INT,
    subject2_ese INT,
    subject3_mse INT,
    subject3_ese INT,
    subject4_mse INT,
    subject4_ese INT
);
```

1.3. Entity Class (Student.java)

```
java
Copy code
package com.example.resultapp.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String rollNumber;
    private int subject1Mse;
    private int subject1Ese;
    private int subject2Mse;
    private int subject2Ese;
    private int subject3Mse;
    private int subject3Ese;
    private int subject4Mse;
    private int subject4Ese;

    // Getters and Setters
}
```

1.4. Repository (StudentRepository.java)

```
java
Copy code
package com.example.resultapp.repository;

import com.example.resultapp.model.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {
}
```

1.5. Service Layer (ResultService.java)

```
java
Copy code
package com.example.resultapp.service;

import com.example.resultapp.model.Student;
import com.example.resultapp.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class ResultService {

    @Autowired
    private StudentRepository studentRepository;

    public Student calculateResult(Student student) {
        // Calculate the total marks for each subject
        double subject1 = (student.getSubject1Mse() * 0.3) +
(student.getSubject1Ese() * 0.7);
        double subject2 = (student.getSubject2Mse() * 0.3) +
(student.getSubject2Ese() * 0.7);
        double subject3 = (student.getSubject3Mse() * 0.3) +
(student.getSubject3Ese() * 0.7);
        double subject4 = (student.getSubject4Mse() * 0.3) +
(student.getSubject4Ese() * 0.7);

        double total = subject1 + subject2 + subject3 + subject4;
        student.setTotalMarks(total);

        return student;
    }

    public Student saveStudent(Student student) {
        return studentRepository.save(student);
    }

    public Student getStudentById(Long id) {
        return studentRepository.findById(id).orElse(null);
    }
}
```

1.6. Controller (ResultController.java)

```
java
Copy code
package com.example.resultapp.controller;

import com.example.resultapp.model.Student;
import com.example.resultapp.service.ResultService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/results")
public class ResultController {

    @Autowired
    private ResultService resultService;

    @PostMapping("/submit")
    public Student submitStudent(@RequestBody Student student) {
        return resultService.calculateResult(student);
    }

    @GetMapping("/{id}")
    public Student getResult(@PathVariable Long id) {
        return resultService.getStudentById(id);
    }
}
```

---

2. React Frontend

2.1. Create React App

Create a new React app using Create React App:

```
bash
Copy code
npx create-react-app result-calculator
cd result-calculator
```

2.2. Install Axios

Install Axios to interact with the backend:

```
bash
Copy code
npm install axios
```

2.3. App.js (Frontend)

```
jsx
Copy code
import React, { useState } from 'react';
import axios from 'axios';
import 'bootstrap/dist/css/bootstrap.min.css';
```

```
function App() {
  const [name, setName] = useState('');
  const [rollNumber, setRollNumber] = useState('');
  const [subject1Mse, setSubject1Mse] = useState('');
  const [subject1Ese, setSubject1Ese] = useState('');
  const [subject2Mse, setSubject2Mse] = useState('');
  const [subject2Ese, setSubject2Ese] = useState('');
  const [subject3Mse, setSubject3Mse] = useState('');
  const [subject3Ese, setSubject3Ese] = useState('');
  const [subject4Mse, setSubject4Mse] = useState('');
  const [subject4Ese, setSubject4Ese] = useState('');
  const [result, setResult] = useState(null);

  const handleSubmit = async (event) => {
    event.preventDefault();

    const student = {
      name,
      rollNumber,
      subject1Mse: parseInt(subject1Mse),
      subject1Ese: parseInt(subject1Ese),
      subject2Mse: parseInt(subject2Mse),
      subject2Ese: parseInt(subject2Ese),
      subject3Mse: parseInt(subject3Mse),
      subject3Ese: parseInt(subject3Ese),
      subject4Mse: parseInt(subject4Mse),
      subject4Ese: parseInt(subject4Ese),
    };

    try {
      const response = await
axios.post('http://localhost:8080/api/results/submit', student);
      setResult(response.data);
    } catch (error) {
      console.error('There was an error!', error);
    }
  };

  return (
    <div className="container">
      <h2 className="text-center mt-5">VIT Semester Result Calculator</h2>
      <form onSubmit={handleSubmit} className="mt-4">
        <div className="form-group">
          <label>Name:</label>
          <input
            type="text"
            className="form-control"
            value={name}
            onChange={(e) => setName(e.target.value)}
            required
          />
        </div>

        <div className="form-group">
          <label>Roll Number:</label>
          <input
            type="text"
            className="form-control"
            value={rollNumber}
            onChange={(e) => setRollNumber(e.target.value)}
            required
          />
        </div>

        <div className="form-group">
          <label>Subject 1 MSE Marks:</label>
          <input
            type="number"
            className="form-control"
            value={subject1Mse}
            onChange={(e) => setSubject1Mse(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label>Subject 1 ESE Marks:</label>
          <input
            type="number"
            className="form-control"
            value={subject1Ese}
            onChange={(e) => setSubject1Ese(e.target.value)}
            required
          />
        </div>

        {/* Repeat similar input fields for other subjects */}

        <button type="submit" className="btn btn-primary">Submit</button>
      </form>

      {result && (
        <div className="mt-5">
          <h3>Result for {result.name} ({result.rollNumber}):</h3>
          <ul>
            <li>Subject 1: {result.subject1Mse} MSE, {result.subject1Ese}
ESE</li>
            <li>Subject 2: {result.subject2Mse} MSE, {result.subject2Ese}
ESE</li>
            <li>Subject 3: {result.subject3Mse} MSE, {result.subject3Ese}
ESE</li>
            <li>Subject 4: {result.subject4Mse} MSE, {result.subject4Ese}
ESE</li>
            <li>Total Marks: {result.totalMarks}</li>
          </ul>
        </div>
      )}
    </div>
  );
}

export default App;
```

3. Result Calculation

- The result is calculated based on the formula:
  - MSE Marks (30%) + ESE Marks (70%)
  - The total is shown after the student submits their marks.

4. Run the Application

1. Start the Spring Boot application:

```
bash
Copy code
./mvnw spring-boot:run
```

2. Start the React app:

```
bash
Copy code
npm start
```

---

22. Design and develop a responsive website to prepare one semester result of VIT students using PHP and MySQL. Take any four subjects with MSE Marks (30%) ESE Marks (70%).

MySQL Database Setup

1.1. Database and Table Creation

```
sql
Copy code
CREATE DATABASE student_results;

USE student_results;

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    roll_number VARCHAR(50) NOT NULL,
    subject1_mse INT,
    subject1_ese INT,
    subject2_mse INT,
    subject2_ese INT,
    subject3_mse INT,
    subject3_ese INT,
    subject4_mse INT,
    subject4_ese INT,
    total_marks DECIMAL(5,2)
);
```

This will create a database `student_results` and a table `students` to store the student name, roll number, MSE, ESE marks for four subjects, and the calculated total marks.

2. PHP Backend

2.1. Database Connection (db.php)

Create a file `db.php` to handle database connections:

```
php
Copy code
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "student_results";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

2.2. Result Calculation and Insertion (submit_result.php)

Create a file `submit_result.php` to calculate the total marks and insert the data into the database:

```
php
Copy code
<?php
include('db.php');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get student data from the form
    $name = $_POST['name'];
    $roll_number = $_POST['roll_number'];
    $subject1_mse = $_POST['subject1_mse'];
    $subject1_ese = $_POST['subject1_ese'];
    $subject2_mse = $_POST['subject2_mse'];
    $subject2_ese = $_POST['subject2_ese'];
    $subject3_mse = $_POST['subject3_mse'];
    $subject3_ese = $_POST['subject3_ese'];
    $subject4_mse = $_POST['subject4_mse'];
    $subject4_ese = $_POST['subject4_ese'];

    // Calculate total marks for each subject
    $subject1_total = ($subject1_mse * 0.3) + ($subject1_ese * 0.7);
    $subject2_total = ($subject2_mse * 0.3) + ($subject2_ese * 0.7);
    $subject3_total = ($subject3_mse * 0.3) + ($subject3_ese * 0.7);
    $subject4_total = ($subject4_mse * 0.3) + ($subject4_ese * 0.7);

    // Calculate overall total marks
    $total_marks = $subject1_total + $subject2_total + $subject3_total +
$subject4_total;

    // Insert data into database
    $sql = "INSERT INTO students (name, roll_number, subject1_mse,
subject1_ese, subject2_mse, subject2_ese, subject3_mse, subject3_ese,
subject4_mse, subject4_ese, total_marks)
```

```
            VALUES ('$name', '$roll_number', '$subject1_mse', 
'$subject1_ese', '$subject2_mse', '$subject2_ese', '$subject3_mse', 
'$subject3_ese', '$subject4_mse', '$subject4_ese', '$total_marks')";

    if ($conn->query($sql) === TRUE) {
        echo "Result submitted successfully!<br>";
        echo "Total Marks: " . $total_marks;
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}

$conn->close();
?>
```

2.3. Display Results (view_results.php)

Create a file `view_results.php` to fetch and display the results for a specific student:

```
php
Copy code
<?php
include('db.php');

if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $sql = "SELECT * FROM students WHERE id = $id";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            echo "<h3>Result for: " . $row['name'] . "</h3>";
            echo "Roll Number: " . $row['roll_number'] . "<br>";
            echo "Subject 1: " . $row['subject1_mse'] . " MSE, " . 
$row['subject1_ese'] . " ESE<br>";
            echo "Subject 2: " . $row['subject2_mse'] . " MSE, " . 
$row['subject2_ese'] . " ESE<br>";
            echo "Subject 3: " . $row['subject3_mse'] . " MSE, " . 
$row['subject3_ese'] . " ESE<br>";
            echo "Subject 4: " . $row['subject4_mse'] . " MSE, " . 
$row['subject4_ese'] . " ESE<br>";
            echo "Total Marks: " . $row['total_marks'] . "<br>";
        }
    } else {
        echo "No result found.";
    }
}

$conn->close();
?>
```

3. Frontend

3.1. HTML Form for Result Submission (index.php)

Create a file `index.php` to allow students to enter their marks and submit them:

```
php
```

Copy code
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>VIT Semester Result Calculator</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/css/bootstrap.min.c
ss" rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h2 class="text-center">VIT Semester Result Calculator</h2>
        <form action="submit_result.php" method="POST">
            <div class="form-group">
                <label for="name">Student Name</label>
                <input type="text" class="form-control" id="name"
name="name" required>
            </div>
            <div class="form-group">
                <label for="roll_number">Roll Number</label>
                <input type="text" class="form-control" id="roll_number"
name="roll_number" required>
            </div>

            <!-- Subject 1 -->
            <div class="form-group">
                <label for="subject1_mse">Subject 1 MSE Marks</label>
                <input type="number" class="form-control" id="subject1_mse"
name="subject1_mse" required>
            </div>
            <div class="form-group">
                <label for="subject1_ese">Subject 1 ESE Marks</label>
                <input type="number" class="form-control" id="subject1_ese"
name="subject1_ese" required>
            </div>

            <!-- Subject 2 -->
            <div class="form-group">
                <label for="subject2_mse">Subject 2 MSE Marks</label>
                <input type="number" class="form-control" id="subject2_mse"
name="subject2_mse" required>
            </div>
            <div class="form-group">
                <label for="subject2_ese">Subject 2 ESE Marks</label>
                <input type="number" class="form-control" id="subject2_ese"
name="subject2_ese" required>
            </div>

            <!-- Subject 3 -->
            <div class="form-group">
                <label for="subject3_mse">Subject 3 MSE Marks</label>
                <input type="number" class="form-control" id="subject3_mse"
name="subject3_mse" required>
            </div>
            <div class="form-group">
                <label for="subject3_ese">Subject 3 ESE Marks</label>
                <input type="number" class="form-control" id="subject3_ese"
name="subject3_ese" required>
            </div>
            <!-- Subject 4 -->
            <div class="form-group">
                <label for="subject4_mse">Subject 4 MSE Marks</label>
                <input type="number" class="form-control" id="subject4_mse"
name="subject4_mse" required>
            </div>
            <div class="form-group">
                <label for="subject4_ese">Subject 4 ESE Marks</label>
                <input type="number" class="form-control" id="subject4_ese"
name="subject4_ese" required>
            </div>

            <button type="submit" class="btn btn-primary btn-block">Submit
Marks</button>
        </form>
    </div>

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.bundle.
min.js"></script>
</body>
</html>
```

---

23. Design and develop a responsive website to prepare one semester result of VIT students using JavaScript, React and Node JS and MySQL. Take any four subjects with MSE Marks (30%) ESE Marks (70%).

Step 1: MySQL Database Setup

Create the `student_results` database and a `students` table.

```
sql
Copy code
CREATE DATABASE student_results;

USE student_results;

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    roll_number VARCHAR(50) NOT NULL,
    subject1_mse INT,
    subject1_ese INT,
    subject2_mse INT,
    subject2_ese INT,
    subject3_mse INT,
    subject3_ese INT,
    subject4_mse INT,
    subject4_ese INT,
    total_marks DECIMAL(5,2)
);
```

Step 2: Node.js Backend Setup

2.1. Install Necessary Packages

Make sure you have Node.js installed. Then initialize your project and install necessary dependencies.

```
bash
Copy code
npm init -y
npm install express mysql2 cors body-parser
```

2.2. Database Connection (db.js)

Create a file `db.js` to handle the MySQL connection.

```
javascript
Copy code
const mysql = require('mysql2');

const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root', // Change this to your MySQL username
    password: '', // Change this to your MySQL password
    database: 'student_results'
});

connection.connect((err) => {
    if (err) {
        console.error('Error connecting to the database:', err.stack);
        return;
    }
    console.log('Connected to the database');
});

module.exports = connection;
```

2.3. Backend Routes (server.js)

Create a file `server.js` to handle the API requests for submitting and retrieving student results.

```
javascript
Copy code
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const connection = require('./db');

const app = express();
const port = 5000;
```

```
app.use(cors());
app.use(bodyParser.json());

// Route to submit the result
app.post('/submit-result', (req, res) => {
    const { name, roll_number, subject1_mse, subject1_ese, subject2_mse,
subject2_ese, subject3_mse, subject3_ese, subject4_mse, subject4_ese } =
req.body;

    const subject1_total = (subject1_mse * 0.3) + (subject1_ese * 0.7);
    const subject2_total = (subject2_mse * 0.3) + (subject2_ese * 0.7);
    const subject3_total = (subject3_mse * 0.3) + (subject3_ese * 0.7);
    const subject4_total = (subject4_mse * 0.3) + (subject4_ese * 0.7);

    const total_marks = subject1_total + subject2_total + subject3_total +
subject4_total;

    const query = 'INSERT INTO students (name, roll_number, subject1_mse,
subject1_ese, subject2_mse, subject2_ese, subject3_mse, subject3_ese,
subject4_mse, subject4_ese, total_marks) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?)';

    connection.query(query, [name, roll_number, subject1_mse, subject1_ese,
subject2_mse, subject2_ese, subject3_mse, subject3_ese, subject4_mse,
subject4_ese, total_marks], (err, results) => {
        if (err) {
            console.log(err);
            res.status(500).json({ message: 'Error submitting result' });
        } else {
            res.status(200).json({ message: 'Result submitted
successfully!' });
        }
    });
});

// Route to get student result by id
app.get('/get-result/:id', (req, res) => {
    const { id } = req.params;

    const query = 'SELECT * FROM students WHERE id = ?';

    connection.query(query, [id], (err, results) => {
        if (err) {
            console.log(err);
            res.status(500).json({ message: 'Error retrieving result' });
        } else {
            res.status(200).json(results[0]);
        }
    });
});

app.listen(port, () => {
    console.log(`Server is running on port ${port}`);
});
```

Step 3: React Frontend

3.1. Set up React App

Create a new React app using Create React App.

```
bash
Copy code
npx create-react-app vit-semester-result
cd vit-semester-result
npm install axios bootstrap
```

3.2. React Components

App.js (Main Component)

In `App.js`, create the form for entering marks and submitting them.

```
javascript
Copy code
import React, { useState } from 'react';
import axios from 'axios';
import 'bootstrap/dist/css/bootstrap.min.css';

function App() {
  const [name, setName] = useState('');
  const [rollNumber, setRollNumber] = useState('');
  const [subject1MSE, setSubject1MSE] = useState('');
  const [subject1ESE, setSubject1ESE] = useState('');
  const [subject2MSE, setSubject2MSE] = useState('');
  const [subject2ESE, setSubject2ESE] = useState('');
  const [subject3MSE, setSubject3MSE] = useState('');
  const [subject3ESE, setSubject3ESE] = useState('');
  const [subject4MSE, setSubject4MSE] = useState('');
  const [subject4ESE, setSubject4ESE] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();

    const result = {
      name,
      roll_number: rollNumber,
      subject1_mse: subject1MSE,
      subject1_ese: subject1ESE,
      subject2_mse: subject2MSE,
      subject2_ese: subject2ESE,
      subject3_mse: subject3MSE,
      subject3_ese: subject3ESE,
      subject4_mse: subject4MSE,
      subject4_ese: subject4ESE
    };

    try {
      const response = await axios.post('http://localhost:5000/submit-
result', result);
      alert(response.data.message);
    } catch (error) {
      alert('Error submitting result');
    }
  };
  return (
    <div className="container mt-5">
      <h2 className="text-center">VIT Semester Result Calculator</h2>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label>Student Name</label>
          <input type="text" className="form-control" value={name}
onChange={(e) => setName(e.target.value)} required />
        </div>
        <div className="form-group">
          <label>Roll Number</label>
          <input type="text" className="form-control" value={rollNumber}
onChange={(e) => setRollNumber(e.target.value)} required />
        </div>

        {/* Subject 1 */}
        <div className="form-group">
          <label>Subject 1 MSE Marks</label>
          <input type="number" className="form-control" value={subject1MSE}
onChange={(e) => setSubject1MSE(e.target.value)} required />
        </div>
        <div className="form-group">
          <label>Subject 1 ESE Marks</label>
          <input type="number" className="form-control" value={subject1ESE}
onChange={(e) => setSubject1ESE(e.target.value)} required />
        </div>

        {/* Subject 2 */}
        <div className="form-group">
          <label>Subject 2 MSE Marks</label>
          <input type="number" className="form-control" value={subject2MSE}
onChange={(e) => setSubject2MSE(e.target.value)} required />
        </div>
        <div className="form-group">
          <label>Subject 2 ESE Marks</label>
          <input type="number" className="form-control" value={subject2ESE}
onChange={(e) => setSubject2ESE(e.target.value)} required />
        </div>

        {/* Subject 3 */}
        <div className="form-group">
          <label>Subject 3 MSE Marks</label>
          <input type="number" className="form-control" value={subject3MSE}
onChange={(e) => setSubject3MSE(e.target.value)} required />
        </div>
        <div className="form-group">
          <label>Subject 3 ESE Marks</label>
          <input type="number" className="form-control" value={subject3ESE}
onChange={(e) => setSubject3ESE(e.target.value)} required />
        </div>

        {/* Subject 4 */}
        <div className="form-group">
          <label>Subject 4 MSE Marks</label>
          <input type="number" className="form-control" value={subject4MSE}
onChange={(e) => setSubject4MSE(e.target.value)} required />
        </div>
        <div className="form-group">
          <label>Subject 4 ESE Marks</label>
          <input type="number" className="form-control" value={subject4ESE}
onChange={(e) => setSubject4ESE(e.target.value)} required />
```

```
        </div>

        <button type="submit" className="btn btn-primary btn-
block">Submit</button>
      </form>
    </div>
  );
}

export default App;
```

3.3. Run the Application

1. Start the backend server:

   ```
   bash
   Copy code
   node server.js
   ```

2. Start the React frontend:

   ```
   bash
   Copy code
   npm start
   ```

---

24. Design and develop a responsive website for an online book store using Spring boot and MySQL having 1) Home Page2) Login Page 3) Catalogue Page: 4) Registration Page: (database)

Step 1: MySQL Database Setup

Create a MySQL database and tables for users and books.

Database schema:

```
sql
Copy code
CREATE DATABASE online_bookstore;

USE online_bookstore;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    description TEXT,
    stock_quantity INT NOT NULL
);
```

Step 2: Create Spring Boot Project

You can create a Spring Boot application using Spring Initializr (https://start.spring.io/). Select the following dependencies:

- **Spring Web**
- **Spring Data JPA**
- **Spring Boot DevTools**
- **Spring Security** (for login)
- **MySQL Driver**

After generating the project, unzip it and open it in your IDE.

Add the MySQL Database Configuration in `application.properties`:

```
properties
Copy code
spring.datasource.url=jdbc:mysql://localhost:3306/online_bookstore
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

Step 3: Create Models and Repositories

User Entity (User.java):

```
java
Copy code
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private String password;

    // Getters and setters
}
```

Book Entity (Book.java):

```
java
Copy code
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;
    private BigDecimal price;
    private String description;
    private int stockQuantity;

    // Getters and setters
}
```

User Repository (UserRepository.java):

```
java
Copy code
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByEmail(String email);
}
```

Book Repository (BookRepository.java):

```
java
Copy code
public interface BookRepository extends JpaRepository<Book, Long> {
    List<Book> findAll();
    List<Book> findByTitleContainingOrAuthorContaining(String title, String
author);
}
```

Step 4: Create Controllers and Views

Home Controller (HomeController.java):

```
java
Copy code
@Controller
public class HomeController {

    @Autowired
    private BookRepository bookRepository;

    @GetMapping("/")
    public String homePage(Model model) {
        List<Book> books = bookRepository.findAll();
        model.addAttribute("books", books);
        return "home";
    }

    @GetMapping("/catalogue")
    public String cataloguePage(Model model, @RequestParam(defaultValue =
"") String search) {
        List<Book> books;
        if (search.isEmpty()) {
            books = bookRepository.findAll();
        } else {
            books =
bookRepository.findByTitleContainingOrAuthorContaining(search, search);
        }
        model.addAttribute("books", books);
        model.addAttribute("search", search);
        return "catalogue";
    }
}
```

User Controller (UserController.java):

```
java
Copy code
@Controller
public class UserController {

    @Autowired
    private UserRepository userRepository;

    @PostMapping("/register")
    public String registerUser(@RequestParam String name, @RequestParam
String email, @RequestParam String password) {
        User user = new User();
        user.setName(name);
        user.setEmail(email);
        user.setPassword(new BCryptPasswordEncoder().encode(password));
        userRepository.save(user);
        return "redirect:/login";
    }

    @GetMapping("/login")
    public String loginPage() {
        return "login";
    }

    @PostMapping("/login")
    public String login(@RequestParam String email, @RequestParam String
password) {
        Optional<User> user = userRepository.findByEmail(email);
        if (user.isPresent() && new
BCryptPasswordEncoder().matches(password, user.get().getPassword())) {
            return "redirect:/";
        }
        return "login";
    }
}
```

Step 5: Authentication (Login & Registration)

For login and registration, you can use **Spring Security**. Add the following to your `SecurityConfig.java`:

Security Configuration (SecurityConfig.java):

```
java
Copy code
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserRepository userRepository;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests()
                .antMatchers("/login", "/register", "/").permitAll()
                .anyRequest().authenticated()
            .and()
            .formLogin()
```

```
                .loginPage("/login")
                .defaultSuccessUrl("/", true)
            .and()
            .logout().logoutUrl("/logout").logoutSuccessUrl("/");
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.userDetailsService(email -> {
            User user = userRepository.findByEmail(email).orElseThrow(() ->
new UsernameNotFoundException("User not found"));
            return new UserDetailsImpl(user);
        });
    }
}
```

Step 6: Frontend Design with HTML/CSS/Bootstrap

Home Page (home.html)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Bookstore</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h1 class="text-center mt-5">Welcome to the Online Bookstore</h1>
        <div class="row">
            <div class="col-12">
                <a href="/catalogue" class="btn btn-primary">Go to
Catalogue</a>
            </div>
        </div>
    </div>
</body>
</html>
```

Catalogue Page (catalogue.html)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Catalogue</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
```

```
        <h1 class="text-center mt-5">Catalogue</h1>
        <form method="get" action="/catalogue">
            <input type="text" name="search" class="form-control"
placeholder="Search by title or author" value="${search}" />
            <button type="submit" class="btn btn-secondary mt-
2">Search</button>
        </form>
        <div class="row mt-3">
            <th:block th:each="book : ${books}">
                <div class="col-md-3">
                    <div class="card">
                        <img src="#" class="card-img-top" alt="book image">
                        <div class="card-body">
                            <h5 class="card-title"
th:text="${book.title}">Book Title</h5>
                            <p class="card-text"
th:text="${book.author}">Author Name</p>
                            <p class="card-text">Price: <span
th:text="${book.price}">Price</span></p>
                            <a href="#" class="btn btn-primary">Add to
Cart</a>
                        </div>
                    </div>
                </div>
            </th:block>
        </div>
    </div>
</body>
</html>
```

Login Page (login.html)

html
Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h1 class="text-center mt-5">Login</h1>
        <form action="/login" method="POST">
            <div class="form-group">
                <label for="email">Email</label>
                <input type="email" class="form-control" id="email"
name="email" required />
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password"
name="password" required />
            </div>
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
        <a href="/register">Create an account</a>
    </div>
```

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

Registration Page (register.html)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h1 class="text-center mt-5">Register</h1>
        <form action="/register" method="POST">
            <div class="form-group">
                <label for="name">Name</label>
                <input type="text" class="form-control" id="name"
name="name" required />
            </div>
            <div class="form-group">
                <label for="email">Email</label>
                <input type="email" class="form-control" id="email"
name="email" required />
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password"
name="password" required />
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
    </div>
</body>
</html>

===============================================================
```

25.Design and develop a responsive website for an online book store using REACT, Node JS and MySQL/ MongoDB having 1) Home Page2) Login Page 3) Catalogue Page: 4) Registration Page: (database)

MySQL Schema:

1. **Create Database:**

```
sql
Copy code
CREATE DATABASE online_bookstore;

USE online_bookstore;
```

2. **Create Users Table:**

```
sql
Copy code
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

3. **Create Books Table:**

```
sql
Copy code
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    description TEXT,
    stock_quantity INT NOT NULL
);
```

Step 2: Backend Setup with Node.js, Express, and MySQL

1. **Create a New Node.js Project:**

```
bash
Copy code
mkdir online-bookstore
cd online-bookstore
npm init -y
npm install express mysql2 bcryptjs jsonwebtoken cors dotenv
```

2. **Create `server.js` File:**

```
js
Copy code
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
require('dotenv').config();

const app = express();
const PORT = 5000;

app.use(express.json());
app.use(cors());

// Create MySQL connection
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'yourpassword',
    database: 'online_bookstore'
});

// Connect to MySQL
db.connect((err) => {
    if (err) throw err;
    console.log('Connected to MySQL Database');
});

// Register User API
app.post('/register', (req, res) => {
    const { name, email, password } = req.body;

    bcrypt.hash(password, 10, (err, hashedPassword) => {
        if (err) return res.status(500).json({ error: 'Error hashing
password' });

        db.query(
            'INSERT INTO users (name, email, password) VALUES (?, ?, ?)',
            [name, email, hashedPassword],
            (err, result) => {
                if (err) return res.status(500).json({ error: err.message
});
                res.status(201).json({ message: 'User registered
successfully' });
            }
        );
    });
});

// Login User API
app.post('/login', (req, res) => {
    const { email, password } = req.body;

    db.query('SELECT * FROM users WHERE email = ?', [email], (err, result)
=> {
        if (err) return res.status(500).json({ error: err.message });
        if (result.length === 0) return res.status(400).json({ error: 'User
not found' });

        const user = result[0];
        bcrypt.compare(password, user.password, (err, isMatch) => {
            if (err) return res.status(500).json({ error: err.message });
            if (!isMatch) return res.status(400).json({ error: 'Incorrect
password' });

            const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET,
{
                expiresIn: '1h'
            });
            res.json({ token });
        });
    });
});

// Get All Books API
app.get('/books', (req, res) => {
    db.query('SELECT * FROM books', (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json(result);
    });
```

```
});

// Get Book by ID API
app.get('/books/:id', (req, res) => {
    const { id } = req.params;
    db.query('SELECT * FROM books WHERE id = ?', [id], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        if (result.length === 0) return res.status(404).json({ error: 'Book
not found' });
        res.json(result[0]);
    });
});

// Start the server
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

3. **Environment File `.env`**:

```
plaintext
Copy code
JWT_SECRET=your-secret-key
```

Step 3: Frontend Setup with React

1. **Create a New React App:**

```
bash
Copy code
npx create-react-app online-bookstore-frontend
cd online-bookstore-frontend
npm install axios react-router-dom
```

2. **Frontend Directory Structure**:

```
css
Copy code
src/
├── components/
│   ├── HomePage.js
│   ├── LoginPage.js
│   ├── CataloguePage.js
│   ├── RegisterPage.js
│   └── Navbar.js
├── App.js
└── index.js
```

3. **App Component (App.js)**:

```
js
Copy code
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import HomePage from './components/HomePage';
import LoginPage from './components/LoginPage';
import CataloguePage from './components/CataloguePage';
import RegisterPage from './components/RegisterPage';
import Navbar from './components/Navbar';

function App() {
  return (
    <Router>
      <Navbar />
      <Switch>
        <Route path="/" exact component={HomePage} />
        <Route path="/login" component={LoginPage} />
        <Route path="/catalogue" component={CataloguePage} />
        <Route path="/register" component={RegisterPage} />
      </Switch>
    </Router>
  );
}

export default App;
```

4. **Navbar Component (Navbar.js)**:

```
js
Copy code
import React from 'react';
import { Link } from 'react-router-dom';

function Navbar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/login">Login</Link></li>
                <li><Link to="/catalogue">Catalogue</Link></li>
                <li><Link to="/register">Register</Link></li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

5. **Home Page (HomePage.js)**:

```
js
Copy code
import React from 'react';

function HomePage() {
    return (
        <div>
            <h1>Welcome to the Online Bookstore</h1>
            <p>Your one-stop shop for books</p>
        </div>
    );
}

export default HomePage;
```

6. **Login Page (LoginPage.js)**:

```
js
Copy code
import React, { useState } from 'react';
import axios from 'axios';
import { useHistory } from 'react-router-dom';

function LoginPage() {
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const history = useHistory();

    const handleSubmit = (e) => {
        e.preventDefault();
        axios
            .post('http://localhost:5000/login', { email, password })
            .then((response) => {
                localStorage.setItem('token', response.data.token);
                history.push('/');
            })
            .catch((error) => {
                console.error('There was an error logging in:', error);
            });
    };

    return (
        <div>
            <h2>Login</h2>
            <form onSubmit={handleSubmit}>
                <input
                    type="email"
                    placeholder="Email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                />
                <input
                    type="password"
                    placeholder="Password"
                    value={password}
                    onChange={(e) => setPassword(e.target.value)}
                />
                <button type="submit">Login</button>
            </form>
        </div>
    );
}

export default LoginPage;
```

7. **Catalogue Page (CataloguePage.js)**:

```
js
Copy code
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function CataloguePage() {
    const [books, setBooks] = useState([]);

    useEffect(() => {
        axios.get('http://localhost:5000/books')
```

```
            .then((response) => {
                setBooks(response.data);
            })
            .catch((error) => {
                console.error('There was an error fetching the books:',
error);
            });
    }, []);

    return (
        <div>
            <h2>Book Catalogue</h2>
            <ul>
                {books.map((book) => (
                    <li key={book.id}>{book.title} by {book.author}</li>
                ))}
            </ul>
        </div>
    );
}

export default CataloguePage;
```

8. **Register Page (RegisterPage.js)**:

```
js
Copy code
import React, { useState } from 'react';
import axios from 'axios';

function RegisterPage() {
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        axios
            .post('http://localhost:5000/register', { name, email, password
})
            .then((response) => {
                alert('Registration successful');
            })
            .catch((error) => {
                console.error('There was an error registering:', error);
            });
    };

    return (
        <div>
            <h2>Register</h2>
            <form onSubmit={handleSubmit}>
                <input
                    type="text"
                    placeholder="Name"
                    value={name}
                    onChange={(e) => setName(e.target.value)}
                />
                <input
                    type="email"
                    placeholder="Email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                />
                <input
                    type="password"
                    placeholder="Password"
                    value={password}
                    onChange={(e) => setPassword(e.target.value)}
                />
                <button type="submit">Register</button>
            </form>
        </div>
    );
}

export default RegisterPage;
```

Step 4: Run the Application

1. **Backend:**
   - Run the backend server:

```
bash
Copy code
node server.js
```

2. **Frontend:**
   - Run the React app:

```
bash
Copy code
npm start
```

==

26. Design PHP login module with user registration form, login form. System should use cookies to track user. Use session handling and database MySQL for login.

Step 1: Database Setup

Create a MySQL Database and Tables

1. **Create Database:**

```
sql
Copy code
CREATE DATABASE login_system;

USE login_system;
```

2. **Create Users Table:**

```
sql
Copy code
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Step 2: Backend Setup

Configure Database Connection

1. **Database Configuration (`db.php`):**

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'login_system';
$username = 'root';
$password = '';

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

Step 3: User Registration

2. **Registration Page (`register.php`):**

```
php
Copy code
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

    $stmt = $conn->prepare("INSERT INTO users (name, email, password)
VALUES (:name, :email, :password)");
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':password', $password);

    if ($stmt->execute()) {
        echo "Registration successful. <a href='login.php'>Login here</a>";
    } else {
        echo "Error: " . $stmt->errorInfo()[2];
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
</head>
<body>
    <h2>Register</h2>
    <form method="POST">
        <input type="text" name="name" placeholder="Name" required><br>
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Step 4: User Login

3. **Login Page (`login.php`):**

```
php
Copy code
<?php
include 'db.php';
session_start();

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    $stmt = $conn->prepare("SELECT * FROM users WHERE email = :email");
    $stmt->bindParam(':email', $email);
    $stmt->execute();
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];

        // Set a cookie to track the user
        setcookie('user_email', $email, time() + (86400 * 30), "/");
        header("Location: dashboard.php");
    } else {
        echo "Invalid email or password.";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form method="POST">
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Step 5: Dashboard and Logout

4. **Dashboard Page (`dashboard.php`):**

```
php
Copy code
<?php
session_start();

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

echo "Welcome to the Dashboard!<br>";
echo "Logged in as: " . $_COOKIE['user_email'] . "<br>";

echo "<a href='logout.php'>Logout</a>";
?>
```

5. **Logout Page (`logout.php`):**

```
php
Copy code
<?php
session_start();

// Unset session variables
session_unset();
session_destroy();

// Remove cookie
setcookie('user_email', '', time() - 3600, "/");
```

```
header("Location: login.php");
exit();
?>
```

Step 6: CSS for Styling (Optional)

Add basic styling to the HTML pages using a `styles.css` file:

```
css
Copy code
body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 20px;
}
form {
    display: inline-block;
    text-align: left;
}
input {
    margin: 5px 0;
    padding: 10px;
    width: 200px;
}
button {
    padding: 10px 20px;
    background-color: #007BFF;
    color: white;
    border: none;
    cursor: pointer;
}
button:hover {
    background-color: #0056b3;
}
```

---

27. Design and develop attendance system using PHP and MySQL.

a. student must be able to register himself

b. Teacher should be able to take attendance online using check boxes, roll no and name

Step 1: Database Setup

Create Database and Tables

```
sql
Copy code
CREATE DATABASE attendance_system;
```

```
USE attendance_system;

-- Table for student registration
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    roll_no VARCHAR(20) NOT NULL UNIQUE,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

-- Table for attendance
CREATE TABLE attendance (
    id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT NOT NULL,
    date DATE NOT NULL,
    status ENUM('Present', 'Absent') NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(id)
);
```

Step 2: Backend Setup

Database Connection (`db.php`)

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'attendance_system';
$username = 'root';
$password = '';

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

Step 3: Student Registration

Student Registration Form (`register.php`)

```
php
Copy code
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $roll_no = $_POST['roll_no'];
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

    $stmt = $conn->prepare("INSERT INTO students (roll_no, name, email,
password) VALUES (:roll_no, :name, :email, :password)");
```

```
    $stmt->bindParam(':roll_no', $roll_no);
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':password', $password);

    if ($stmt->execute()) {
        echo "Registration successful.";
    } else {
        echo "Error: " . $stmt->errorInfo()[2];
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Student Registration</title>
</head>
<body>
    <h2>Register</h2>
    <form method="POST">
        <input type="text" name="roll_no" placeholder="Roll No" 
required><br>
        <input type="text" name="name" placeholder="Name" required><br>
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password" 
required><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Step 4: Teacher Attendance System

Teacher Login Form (`teacher_login.php`)

php
Copy code

```
<?php
// For simplicity, hardcoding teacher credentials
$teacher_email = 'teacher@example.com';
$teacher_password = 'teacher123';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    if ($email === $teacher_email && $password === $teacher_password) {
        header("Location: take_attendance.php");
    } else {
        echo "Invalid credentials.";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Teacher Login</title>
</head>
```

```
<body>
    <h2>Teacher Login</h2>
    <form method="POST">
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Take Attendance (`take_attendance.php`)

php
Copy code

```
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $date = date('Y-m-d');

    foreach ($_POST['attendance'] as $student_id => $status) {
        $stmt = $conn->prepare("INSERT INTO attendance (student_id, date,
status) VALUES (:student_id, :date, :status)");
        $stmt->bindParam(':student_id', $student_id);
        $stmt->bindParam(':date', $date);
        $stmt->bindParam(':status', $status);
        $stmt->execute();
    }
    echo "Attendance saved successfully.";
}

$stmt = $conn->prepare("SELECT * FROM students");
$stmt->execute();
$students = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html>
<head>
    <title>Take Attendance</title>
</head>
<body>
    <h2>Take Attendance</h2>
    <form method="POST">
        <table border="1">
            <tr>
                <th>Roll No</th>
                <th>Name</th>
                <th>Status</th>
            </tr>
            <?php foreach ($students as $student): ?>
            <tr>
                <td><?php echo $student['roll_no']; ?></td>
                <td><?php echo $student['name']; ?></td>
                <td>
                    <input type="radio" name="attendance[<?php echo
$student['id']; ?>]" value="Present" required> Present
                    <input type="radio" name="attendance[<?php echo
$student['id']; ?>]" value="Absent" required> Absent
                </td>
            </tr>
            <?php endforeach; ?>
        </table>
        <button type="submit">Save Attendance</button>
    </form>
</body>
</html>
```

Step 5: View Attendance (Optional)

View Attendance (`view_attendance.php`)

php
Copy code

```
<?php
include 'db.php';

$stmt = $conn->prepare("SELECT students.roll_no, students.name,
attendance.date, attendance.status
                        FROM attendance
                        INNER JOIN students ON attendance.student_id =
students.id");
$stmt->execute();
$attendance_records = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html>
<head>
    <title>View Attendance</title>
</head>
<body>
    <h2>Attendance Records</h2>
    <table border="1">
        <tr>
            <th>Roll No</th>
            <th>Name</th>
            <th>Date</th>
            <th>Status</th>
        </tr>
        <?php foreach ($attendance_records as $record): ?>
        <tr>
            <td><?php echo $record['roll_no']; ?></td>
            <td><?php echo $record['name']; ?></td>
            <td><?php echo $record['date']; ?></td>
            <td><?php echo $record['status']; ?></td>
        </tr>
        <?php endforeach; ?>
    </table>
</body>
</html>
```

28. Design and develop online shopping system where farmers can sell their agriculture products online using PHP and MySQL make assumptions about how system should be.

Step 1: Database Setup

Create Database and Tables

```
sql
Copy code
CREATE DATABASE farming_shop;

USE farming_shop;

-- Table for farmer registration
CREATE TABLE farmers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

-- Table for product listing
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    farmer_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    quantity INT NOT NULL,
    FOREIGN KEY (farmer_id) REFERENCES farmers(id)
);

-- Table for customer registration
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

-- Table for orders
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Step 2: Backend Setup

Database Connection (`db.php`)

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'farming_shop';
$username = 'root';
$password = '';

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

Step 3: Farmer Registration and Login

Farmer Registration (`register_farmer.php`)

```
php
Copy code
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

    $stmt = $conn->prepare("INSERT INTO farmers (name, email, password)
VALUES (:name, :email, :password)");
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':password', $password);

    if ($stmt->execute()) {
        echo "Farmer registered successfully.";
    } else {
        echo "Error: " . $stmt->errorInfo()[2];
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Farmer Registration</title>
</head>
<body>
    <h2>Register as Farmer</h2>
    <form method="POST">
        <input type="text" name="name" placeholder="Name" required><br>
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Farmer Login (`login_farmer.php`)

```
php
Copy code
<?php
include 'db.php';
session_start();

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    $stmt = $conn->prepare("SELECT * FROM farmers WHERE email = :email");
    $stmt->bindParam(':email', $email);
    $stmt->execute();
    $farmer = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($farmer && password_verify($password, $farmer['password'])) {
        $_SESSION['farmer_id'] = $farmer['id'];
        header("Location: farmer_dashboard.php");
    } else {
        echo "Invalid email or password.";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Farmer Login</title>
</head>
<body>
    <h2>Login as Farmer</h2>
    <form method="POST">
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password"
required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Step 4: Product Listing by Farmers

Farmer Dashboard (`farmer_dashboard.php`)

```
php
Copy code
<?php
include 'db.php';
session_start();

if (!isset($_SESSION['farmer_id'])) {
    header("Location: login_farmer.php");
}

$farmer_id = $_SESSION['farmer_id'];
```

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $description = $_POST['description'];
    $price = $_POST['price'];
    $quantity = $_POST['quantity'];

    $stmt = $conn->prepare("INSERT INTO products (farmer_id, name,
description, price, quantity) VALUES (:farmer_id, :name, :description,
:price, :quantity)");
    $stmt->bindParam(':farmer_id', $farmer_id);
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':description', $description);
    $stmt->bindParam(':price', $price);
    $stmt->bindParam(':quantity', $quantity);

    if ($stmt->execute()) {
        echo "Product listed successfully.";
    } else {
        echo "Error: " . $stmt->errorInfo()[2];
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Farmer Dashboard</title>
</head>
<body>
    <h2>List a Product</h2>
    <form method="POST">
        <input type="text" name="name" placeholder="Product Name"
required><br>
        <textarea name="description" placeholder="Description"
required></textarea><br>
        <input type="number" name="price" placeholder="Price" required><br>
        <input type="number" name="quantity" placeholder="Quantity"
required><br>
        <button type="submit">Add Product</button>
    </form>
</body>
</html>
```

Step 5: Customer Registration and Login

Repeat Steps 3 for Customers with Changes in Table Names

Step 6: Product Catalog and Order Placement

Customer Dashboard (`catalog.php`)

```
php
Copy code
<?php
include 'db.php';
```

```
$stmt = $conn->prepare("SELECT * FROM products");
$stmt->execute();
$products = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html>
<head>
    <title>Product Catalog</title>
</head>
<body>
    <h2>Product Catalog</h2>
    <table border="1">
        <tr>
            <th>Name</th>
            <th>Description</th>
            <th>Price</th>
            <th>Quantity</th>
            <th>Action</th>
        </tr>
        <?php foreach ($products as $product): ?>
        <tr>
            <td><?php echo $product['name']; ?></td>
            <td><?php echo $product['description']; ?></td>
            <td><?php echo $product['price']; ?></td>
            <td><?php echo $product['quantity']; ?></td>
            <td>
                <form method="POST" action="place_order.php">
                    <input type="hidden" name="product_id" value="<?php
echo $product['id']; ?>">
                    <input type="number" name="quantity" required>
                    <button type="submit">Order</button>
                </form>
            </td>
        </tr>
        <?php endforeach; ?>
    </table>
</body>
</html>
```

---

29. Design and develop a PHP script to limit the maximum number of concurrent sessions for a user to 3. Set session expiration time out to 5 minutes.

Steps to Implement the System

1. **Database Setup**:
   - Create a database table to track user sessions.
2. **Session Management**:
   - On login, check active sessions for the user and enforce the limit.

    - Expire old sessions based on the timeout.
3. **Session Timeout**:
    - Check if a session is older than 5 minutes and expire it.
4. **Logout Handling**:
    - Remove the session from the database on user logout.

Database Setup

Create a `user_sessions` table to track sessions.

```
sql
Copy code
CREATE DATABASE user_sessions_db;

USE user_sessions_db;

CREATE TABLE user_sessions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    session_id VARCHAR(255) NOT NULL,
    last_activity DATETIME NOT NULL,
    UNIQUE(session_id)
);
```

PHP Code

Database Connection (`db.php`)

```
php
Copy code
<?php
$host = 'localhost';
$dbname = 'user_sessions_db';
$username = 'root';
$password = '';

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

Login Script (`login.php`)

```
php
Copy code
<?php
session_start();
include 'db.php';

$user_id = 1; // Replace with actual user ID after authentication
$current_session = session_id();
$timeout_minutes = 5;
```

```
$max_sessions = 3;

// Expire old sessions
$expiration_time = date('Y-m-d H:i:s', strtotime("-$timeout_minutes
minutes"));
$stmt = $conn->prepare("DELETE FROM user_sessions WHERE last_activity <
:expiration_time");
$stmt->bindParam(':expiration_time', $expiration_time);
$stmt->execute();

// Check active sessions
$stmt = $conn->prepare("SELECT COUNT(*) AS session_count FROM user_sessions
WHERE user_id = :user_id");
$stmt->bindParam(':user_id', $user_id);
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_ASSOC);

if ($result['session_count'] >= $max_sessions) {
    die("You have reached the maximum number of concurrent sessions.");
}

// Add current session to the database
$stmt = $conn->prepare("INSERT INTO user_sessions (user_id, session_id,
last_activity) VALUES (:user_id, :session_id, NOW())
    ON DUPLICATE KEY UPDATE last_activity = NOW()");
$stmt->bindParam(':user_id', $user_id);
$stmt->bindParam(':session_id', $current_session);
$stmt->execute();

echo "Login successful. Session started.";
?>
```

Session Validation Script (`validate_session.php`)

```
php
Copy code
<?php
session_start();
include 'db.php';

$current_session = session_id();

// Check if the session exists and is valid
$stmt = $conn->prepare("SELECT * FROM user_sessions WHERE session_id =
:session_id");
$stmt->bindParam(':session_id', $current_session);
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$result) {
    die("Your session has expired. Please log in again.");
}

// Update the last activity time
$stmt = $conn->prepare("UPDATE user_sessions SET last_activity = NOW()
WHERE session_id = :session_id");
$stmt->bindParam(':session_id', $current_session);
$stmt->execute();

echo "Session validated.";
?>
```

Logout Script (`logout.php`)

```
php
Copy code
<?php
session_start();
include 'db.php';

$current_session = session_id();

// Remove session from the database
$stmt = $conn->prepare("DELETE FROM user_sessions WHERE session_id =
:session_id");
$stmt->bindParam(':session_id', $current_session);
$stmt->execute();

session_destroy();
echo "You have been logged out.";
?>
```

---

30. Design and develop Spring boot application where employee records could be added or employee list could be listed as JSON format. Use postman as a client.

1. **Set up the Spring Boot project**:
   - Use Spring Initializer to create a Spring Boot application with dependencies:
     - Spring Web
     - Spring Data JPA
     - H2 Database
   - Add other required dependencies to the `pom.xml`.
2. **Define the Employee entity**:
   - Create a Java class to represent employee records.
3. **Create the Repository**:
   - Use `JpaRepository` to manage employee data.
4. **Implement the Service Layer**:
   - Add business logic to manage employee operations.
5. **Create the Controller**:
   - Define endpoints for adding and listing employees.
6. **Test using Postman**:
   - Test the endpoints for adding and fetching employee records.

Code Implementation

1. Project Dependencies (pom.xml)

```
xml
Copy code
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

---

2. Employee Entity

```
java
Copy code
package com.example.employeemanagement.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String department;
    private double salary;

    // Constructors
    public Employee() {}

    public Employee(String name, String department, double salary) {
        this.name = name;
        this.department = department;
        this.salary = salary;
    }

    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

---

3. Employee Repository

```
java
Copy code
package com.example.employeemanagement.repository;

import com.example.employeemanagement.entity.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {
}
```

---

4. Employee Service

```
java
Copy code
package com.example.employeemanagement.service;

import com.example.employeemanagement.entity.Employee;
import com.example.employeemanagement.repository.EmployeeRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    private final EmployeeRepository employeeRepository;

    public EmployeeService(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    public Employee addEmployee(Employee employee) {
        return employeeRepository.save(employee);
    }
    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }
}
```

---

5. Employee Controller

```
java
Copy code
package com.example.employeemanagement.controller;

import com.example.employeemanagement.entity.Employee;
import com.example.employeemanagement.service.EmployeeService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    private final EmployeeService employeeService;

    public EmployeeController(EmployeeService employeeService) {
        this.employeeService = employeeService;
    }

    @PostMapping("/add")
    public ResponseEntity<Employee> addEmployee(@RequestBody Employee
employee) {
        Employee savedEmployee = employeeService.addEmployee(employee);
        return ResponseEntity.ok(savedEmployee);
    }

    @GetMapping("/list")
    public ResponseEntity<List<Employee>> getAllEmployees() {
        List<Employee> employees = employeeService.getAllEmployees();
        return ResponseEntity.ok(employees);
    }
}
```

---

6. H2 Database Configuration (application.properties)

```
properties
Copy code
spring.datasource.url=jdbc:h2:mem:employee_db
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

Testing the Application with Postman

1. **Start the Application**:
   - Run the Spring Boot application.

2. **Add Employee Endpoint**:
    - **URL**: `http://localhost:8080/api/employees/add`
    - **Method**: POST
    - **Body (JSON)**:

```
json
Copy code
{
    "name": "John Doe",
    "department": "IT",
    "salary": 75000
}
```

3. **List Employees Endpoint**:
    - **URL**: `http://localhost:8080/api/employees/list`
    - **Method**: GET
    - **Response (Example)**:

```
json
Copy code
[
    {
        "id": 1,
        "name": "John Doe",
        "department": "IT",
        "salary": 75000
    }
]
```