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

**Spring Initializer**

Create the Spring Boot Project:

* Go to the [Spring Initializr](https://start.spring.io/).
* Select:
  + Project: Maven
  + Language: Java
  + Spring Boot version: Latest stable version
  + Add Dependencies:
    - Spring Web
    - Spring Data JPA
    - MySQL Driver
  + Group: com.example
  + Artifact: student-management

student-management/

├── pom.xml

├── src/

│ ├── main/

│ │ ├── java/com/example/studentmanagement/

│ │ │ ├── StudentManagementApplication.java

│ │ │ ├── entity/

│ │ │ │ └── Student.java

│ │ │ ├── repository/

│ │ │ │ └── StudentRepository.java

│ │ │ └── controller/

│ │ │ └── StudentController.java

│ │ └── resources/

│ │ ├── application.properties

│ │ └── static/

│ └── test/

* **Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.4.0</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>student-management</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>student-management</name>

<description>Student Management System</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter for Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter for JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- MySQL Connector for Database -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter for Validation -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<!-- Lombok for Simplified Code -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<scope>provided</scope>

</dependency>

<!-- Spring Boot Starter for Testing -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

* **Controller(StudentController.java)**

**package** com.example.student\_management.controller;

**import** com.example.student\_management.entity.Student;

**import** com.example.student\_management.repository.StudentRepository;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.\*;

**import** java.util.List;

**import** java.util.Optional;

@RestController

@RequestMapping("/students")

**public** **class** StudentController {

@Autowired

**private** StudentRepository studentRepository;

// POST: Add a new student

@PostMapping("/add")

**public** Student addStudent(@RequestBody Student student) {

**return** studentRepository.save(student); // Save student

}

// DELETE: Delete a student by ID

@DeleteMapping("/delete/{id}")

**public** String deleteStudent(@PathVariable Long id) {

Optional<Student> student = studentRepository.findById(id); // Check if student exists

**if** (student.isPresent()) {

studentRepository.deleteById(id);

**return** "Student with ID " + id + " deleted!";

} **else** {

**return** "Student with ID " + id + " not found!";

}

}

// GET: List all students

@GetMapping("/list")

**public** List<Student> listStudents() {

**return** studentRepository.findAll(); // Return all students

}

}

* **Entity(Studen.java)**

**package** com.example.student\_management.entity;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

@Entity

**public** **class** Student {

@Id

**private** Long id;

**private** String name;

**private** **int** age;

**private** String course;

// 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** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getCourse() {

**return** course;

}

**public** **void** setCourse(String course) {

**this**.course = course;

}

}

* **Repository(StudentRepository.java)**

**package** com.example.student\_management.repository;

**import** com.example.student\_management.entity.Student;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** StudentRepository **extends** JpaRepository<Student, Long> {

// JpaRepository provides all basic CRUD operations including save(), findById(), findAll(), and deleteById()

}

* **Application.properties**

# MySQL Database connection configuration

spring.datasource.url=jdbc:mysql://localhost:3307/studentdb?useSSL=false&serverTimezone=UTC

spring.datasource.username=root # Replace with your MySQL username

spring.datasource.password=Bharati@12 # Replace with your MySQL password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA and Hibernate configuration

hibernate.hbm2ddl.auto=update

# Automatically create or update tables based on entity

spring.jpa.show-sql=true

# Show SQL queries in the logs for debugging

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Optional: HikariCP connection pool configuration (if you're using the default pool)

spring.datasource.hikari.maximum-pool-size=10

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

**Step 1: Set up MySQL Database**

1. **Create the database:**

CREATE DATABASE employee\_db;

1. **Create the employee table:**

USE employee\_db;

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

position VARCHAR(100) NOT NULL,

department VARCHAR(100) NOT NULL,

hire\_date DATE NOT NULL

);

**Step 2: Create the PHP application structure**

/employee\_app

/index.php

/add\_employee.php

/view\_employees.php

/config.php

**Step 3: Create config.php for database connection**

<?php

$host = "localhost"; // Database server

$dbname = "employee\_db"; // Database name

$username = "root"; // Database username

$password = ""; // Database password

try {

// Create connection

$conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

// Set PDO error mode to exception

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch(PDOException $e) {

echo "Connection failed: " . $e->getMessage();

}

?>

**Step 4: Create index.php to display a welcome message and link to add and view employees**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Employee Management</title>

</head>

<body>

<h1>Welcome to Employee Management System</h1>

<a href="add\_employee.php">Add New Employee</a><br>

<a href="view\_employees.php">View Employee List</a>

</body>

</html>

**Step 5: Create add\_employee.php to add a new employee record**

<?php

include('config.php');

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$name = $\_POST['name'];

$position = $\_POST['position'];

$department = $\_POST['department'];

$hire\_date = $\_POST['hire\_date'];

$stmt = $conn->prepare("INSERT INTO employees (name, position, department, hire\_date) VALUES (:name, :position, :department, :hire\_date)");

$stmt->bindParam(':name', $name);

$stmt->bindParam(':position', $position);

$stmt->bindParam(':department', $department);

$stmt->bindParam(':hire\_date', $hire\_date);

if ($stmt->execute()) {

echo "Employee added successfully!";

} else {

echo "Error adding employee.";

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Add Employee</title>

</head>

<body>

<h1>Add New Employee</h1>

<form method="POST" action="add\_employee.php">

<label for="name">Name:</label><br>

<input type="text" id="name" name="name" required><br><br>

<label for="position">Position:</label><br>

<input type="text" id="position" name="position" required><br><br>

<label for="department">Department:</label><br>

<input type="text" id="department" name="department" required><br><br>

<label for="hire\_date">Hire Date:</label><br>

<input type="date" id="hire\_date" name="hire\_date" required><br><br>

<input type="submit" value="Add Employee">

</form>

<br><br>

<a href="index.php">Go Back to Home</a>

</body>

</html>

**Step 6: Create view\_employees.php to display all employees**

<?php

include('config.php');

$stmt = $conn->query("SELECT \* FROM employees");

$employees = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Employee List</title>

</head>

<body>

<h1>Employee List</h1>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Position</th>

<th>Department</th>

<th>Hire Date</th>

</tr>

</thead>

<tbody>

<?php foreach ($employees as $employee): ?>

<tr>

<td><?php echo $employee['id']; ?></td>

<td><?php echo $employee['name']; ?></td>

<td><?php echo $employee['position']; ?></td>

<td><?php echo $employee['department']; ?></td>

<td><?php echo $employee['hire\_date']; ?></td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

<br><br>

<a href="index.php">Go Back to Home</a>

</body>

</html>

**Step 7: Testing the application**

1. **Start a local server** using XAMPP, WAMP, or MAMP (or any PHP server you prefer).
2. **Place all files** in the htdocs directory (or your server’s equivalent directory).
3. **Open index.php** in your browser, and test the system by adding employees and viewing the list of employees.

**3. Design following responsive layout using html.**

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



This section should change colour after

clicking on following buttons

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

**1.1 Database Structure (MySQL)**

CREATE DATABASE restaurant;

USE restaurant;

CREATE TABLE menu (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL

);

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(255) NOT NULL,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(50) DEFAULT 'Pending'

);

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

menu\_item\_id INT,

quantity INT DEFAULT 1,

FOREIGN KEY (order\_id) REFERENCES orders(id),

FOREIGN KEY (menu\_item\_id) REFERENCES menu(id)

);

**1.2 PHP Frontend and Backend**

* **config.php** - Connect to the MySQL database.

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "restaurant";

// Create connection

$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

?>

* **index.php** - Display the menu and allow users to place an order.

<?php

include('config.php');

// Fetch menu items from the database

$stmt = $conn->query("SELECT \* FROM menu");

$menu\_items = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Restaurant - Menu</title>

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

</head>

<body>

<header>

<h1>Welcome to Our Restaurant</h1>

<p>Order your favorite food online!</p>

</header>

<section class="menu">

<h2>Our Menu</h2>

<form action="place\_order.php" method="POST">

<?php foreach ($menu\_items as $item): ?>

<div class="menu-item">

<input type="checkbox" name="order\_items[]" value="<?php echo $item['id']; ?>">

<strong><?php echo $item['name']; ?></strong><br>

<span><?php echo $item['description']; ?></span><br>

<span>Price: $<?php echo $item['price']; ?></span>

</div>

<?php endforeach; ?>

<input type="text" name="customer\_name" placeholder="Your Name" required><br>

<input type="submit" value="Place Order">

</form>

</section>

<footer>

<p>&copy; 2024 Restaurant Order System</p>

</footer>

</body>

</html>

* **place\_order.php** - Process the order and save to database.

<?php

include('config.php');

// Get customer name and selected menu items

$customer\_name = $\_POST['customer\_name'];

$order\_items = $\_POST['order\_items'];

// Insert order into the orders table

$stmt = $conn->prepare("INSERT INTO orders (customer\_name) VALUES (:customer\_name)");

$stmt->bindParam(':customer\_name', $customer\_name);

$stmt->execute();

$order\_id = $conn->lastInsertId(); // Get the last inserted order ID

// Insert selected items into order\_items table

foreach ($order\_items as $menu\_item\_id) {

$stmt = $conn->prepare("INSERT INTO order\_items (order\_id, menu\_item\_id, quantity) VALUES (:order\_id, :menu\_item\_id, 1)");

$stmt->bindParam(':order\_id', $order\_id);

$stmt->bindParam(':menu\_item\_id', $menu\_item\_id);

$stmt->execute();

}

echo "<h2>Thank you, $customer\_name! Your order has been placed successfully.</h2>";

echo "<a href='index.php'>Return to Menu</a>";

?>

* **view\_orders.php** - Admin panel to view all orders.

<?php

include('config.php');

// Fetch all orders

$stmt = $conn->query("SELECT \* FROM orders");

$orders = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>View Orders</title>

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

</head>

<body>

<header>

<h1>View Orders</h1>

</header>

<section class="orders">

<h2>All Orders</h2>

<table>

<tr>

<th>Order ID</th>

<th>Customer Name</th>

<th>Order Date</th>

<th>Status</th>

<th>Action</th>

</tr>

<?php foreach ($orders as $order): ?>

<tr>

<td><?php echo $order['id']; ?></td>

<td><?php echo $order['customer\_name']; ?></td>

<td><?php echo $order['order\_date']; ?></td>

<td><?php echo $order['status']; ?></td>

<td>

<a href="update\_order\_status.php?order\_id=<?php echo $order['id']; ?>">Mark as Completed</a>

</td>

</tr>

<?php endforeach; ?>

</table>

</section>

<footer>

<p>&copy; 2024 Restaurant Order System</p>

</footer>

</body>

</html>

* **update\_order\_status.php** - Update order status (e.g., Completed).

<?php

include('config.php');

if (isset($\_GET['order\_id'])) {

$order\_id = $\_GET['order\_id'];

$stmt = $conn->prepare("UPDATE orders SET status = 'Completed' WHERE id = :order\_id");

$stmt->bindParam(':order\_id', $order\_id);

$stmt->execute();

echo "<h2>Order has been marked as completed!</h2>";

echo "<a href='view\_orders.php'>Go back to Orders</a>";

}

?>

* **style.css** - Basic CSS for layout and design.

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

header {

background-color: #333;

color: white;

padding: 20px;

text-align: center;

}

header h1 {

margin: 0;

}

.menu {

padding: 20px;

text-align: center;

}

.menu-item {

background-color: #fff;

border: 1px solid #ddd;

margin: 10px 0;

padding: 15px;

border-radius: 8px;

text-align: left;

}

.menu-item strong {

font-size: 18px;

}

input[type="text"] {

padding: 8px;

margin: 10px;

width: 80%;

max-width: 400px;

border-radius: 4px;

border: 1px solid #ccc;

}

input[type="submit"] {

padding: 10px 20px;

background-color: #333;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #555;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 15px;

position: fixed;

width: 100%;

bottom: 0;

}

**2. Full System Using Spring Boot**

**2.1 Database Structure (MySQL)**

CREATE DATABASE restaurant;

USE restaurant;

CREATE TABLE menu (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL

);

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(255) NOT NULL,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(50) DEFAULT 'Pending'

);

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

menu\_item\_id INT,

quantity INT DEFAULT 1,

FOREIGN KEY (order\_id) REFERENCES orders(id),

FOREIGN KEY (menu\_item\_id) REFERENCES menu(id)

);

**2.2 Spring Boot Backend**

* **Menu.java (Model)**

@Entity

public class Menu {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

private BigDecimal price;

}

* **MenuRepository.java (Repository)**

public interface MenuRepository extends JpaRepository<Menu, Long> {

}

* **OrderController.java (Controller)**

@Controller

public class OrderController {

@Autowired

private MenuRepository menuRepository;

@GetMapping("/")

public String viewMenu(Model model) {

List<Menu> menuItems = menuRepository.findAll();

model.addAttribute("menuItems", menuItems);

return "menu";

}

@PostMapping("/place\_order")

public String placeOrder(@RequestParam List<Long> menuItems, @RequestParam String customerName) {

// Save the order

Order order = new Order(customerName);

orderRepository.save(order);

// Add the order items

for (Long menuItemId : menuItems) {

OrderItem orderItem = new OrderItem(order, menuItemId, 1);

orderItemRepository.save(orderItem);

}

return "redirect:/order/confirmation";

}

@GetMapping("/order/confirmation")

public String orderConfirmation() {

return "confirmation";

}

}

* **application.properties (Configuration)**

spring.datasource.url=jdbc:mysql://localhost:3306/restaurant

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**2.3 Frontend (HTML)**

* **menu.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Restaurant Menu</title>

</head>

<body>

<h1>Restaurant Menu</h1>

<form action="/place\_order" method="POST">

<input type="text" name="customerName" placeholder="Your Name" required><br>

<h2>Choose your items:</h2>

<ul>

<th:block th:each="menuItem : ${menuItems}">

<li>

<input type="checkbox" name="menuItems" th:value="${menuItem.id}">

<span th:text="${menuItem.name}"></span>

<span th:text="${menuItem.price}"></span>

</li>

</th:block>

</ul>

<input type="submit" value="Place Order">

</form>

</body>

</html>

* **confirmation.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Order Confirmation</title>

</head>

<body>

<h1>Your order has been placed successfully!</h1>

<a href="/">Go back to Menu</a>

</body>

</html>

**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.**

**Step 1: Create React App**

npx create-react-app currency-converter

cd currency-converter

**Step 2: Code Implementation (App.js)**

import React, { useState } from 'react';

import './App.css';

const App = () => {

// State to store input and output

const [dollars, setDollars] = useState('');

const [rupees, setRupees] = useState(null);

// Hardcoded exchange rate (USD to INR)

const exchangeRate = 80; // 1 USD = 80 INR

// Handle input change

const handleInputChange = (e) => {

const value = e.target.value;

setDollars(value);

};

// Handle currency conversion

const convertCurrency = () => {

if (dollars) {

const result = dollars \* exchangeRate;

setRupees(result);

}

};

return (

<div className="App">

<div className="container">

<h1>Currency Converter</h1>

<div className="input-container">

<label htmlFor="dollars">Amount in USD: </label>

<input

type="number"

id="dollars"

value={dollars}

onChange={handleInputChange}

placeholder="Enter amount in USD"

/>

</div>

<button onClick={convertCurrency}>Convert</button>

{rupees !== null && (

<div className="result">

<h3>Converted Amount:</h3>

<p>{dollars} USD = {rupees} INR</p>

</div>

)}

</div>

</div>

);

};

export default App;

**Step 3: Add Styling (App.css)**

body {

font-family: Arial, sans-serif;

background-color: #f7f7f7;

margin: 0;

padding: 0;

}

.App {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #e5e5e5;

}

.container {

background-color: #ffffff;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

text-align: center;

width: 300px;

}

h1 {

color: #333;

font-size: 24px;

}

.input-container {

margin-bottom: 20px;

}

input {

padding: 10px;

width: 100%;

margin-top: 5px;

border-radius: 5px;

border: 1px solid #ccc;

}

button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

width: 100%;

font-size: 16px;

}

button:hover {

background-color: #45a049;

}

.result {

margin-top: 20px;

font-size: 18px;

font-weight: bold;

color: #333;

}

**Step 4: Run the Application**

npm start

**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. index.php**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Currency Converter - USD to INR</title>

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

</head>

<body>

<div class="container">

<h1>Currency Converter</h1>

<form action="index.php" method="post">

<div class="input-container">

<label for="usd">Amount in USD:</label>

<input type="number" id="usd" name="usd" placeholder="Enter amount in USD" required>

</div>

<button type="submit" name="convert">Convert</button>

</form>

<?php

if (isset($\_POST['convert'])) {

// Hardcoded exchange rate: 1 USD = 80 INR

$usdAmount = $\_POST['usd'];

$exchangeRate = 80; // 1 USD = 80 INR

$inrAmount = $usdAmount \* $exchangeRate;

echo "<div class='result'>";

echo "<h3>Converted Amount:</h3>";

echo "<p>{$usdAmount} USD = {$inrAmount} INR</p>";

echo "</div>";

}

?>

</div>

</body>

</html>

**2. style.css**

body {

font-family: Arial, sans-serif;

background-color: #f7f7f7;

margin: 0;

padding: 0;

}

.container {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #e5e5e5;

}

h1 {

color: #333;

font-size: 30px;

margin-bottom: 20px;

}

form {

background-color: #ffffff;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: 300px;

text-align: center;

}

.input-container {

margin-bottom: 20px;

}

input {

padding: 10px;

width: 100%;

margin-top: 5px;

border-radius: 5px;

border: 1px solid #ccc;

}

button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

width: 100%;

font-size: 16px;

}

button:hover {

background-color: #45a049;

}

.result {

margin-top: 20px;

font-size: 18px;

font-weight: bold;

color: #333;

}

**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.**

**Step 1: HTML Structure (index.html)**

<!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="style.css">

</head>

<body>

<header>

<h1>Responsive Chessboard</h1>

</header>

<main>

<div class="chessboard-container">

<table class="chessboard">

<tbody>

<!-- Loop to create 8x8 grid -->

<!-- Row 1 -->

<tr>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

</tr>

<!-- Row 2 -->

<tr>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

</tr>

<!-- Row 3 -->

<tr>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

</tr>

<!-- Row 4 -->

<tr>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

</tr>

<!-- Row 5 -->

<tr>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

</tr>

<!-- Row 6 -->

<tr>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

</tr>

<!-- Row 7 -->

<tr>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

</tr>

<!-- Row 8 -->

<tr>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

<td class="white"></td>

<td class="black"></td>

</tr>

</tbody>

</table>

</div>

</main>

<footer>

<p>Designed with ❤️ by You</p>

</footer>

</body>

</html>

**Step 2: CSS Styling (style.css)**

/\* General styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f5f5f5;

}

header {

background-color: #333;

color: white;

text-align: center;

padding: 10px;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 10px;

position: fixed;

bottom: 0;

width: 100%;

}

/\* Chessboard styles \*/

.chessboard-container {

display: flex;

justify-content: center;

margin-top: 20px;

padding: 20px;

}

table.chessboard {

border-collapse: collapse;

width: 100%;

max-width: 400px;

aspect-ratio: 1;

table-layout: fixed;

}

table.chessboard td {

width: 12.5%;

height: 12.5%;

background-color: #f0f0f0;

border: 1px solid #999;

}

table.chessboard td.white {

background-color: #f8f8f8;

}

table.chessboard td.black {

background-color: #333;

}

/\* Responsive Design \*/

@media (max-width: 600px) {

table.chessboard {

max-width: 100%;

}

table.chessboard td {

width: 12.5vw;

height: 12.5vw;

}

}

@media (max-width: 400px) {

table.chessboard td {

width: 15vw;

height: 15vw;

}

}

**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**

npm init -y

npm install express mysql2 bcryptjs jsonwebtoken cors body-parser

**1.2 Create the backend (server.js)**

const express = require("express");

const mysql = require("mysql2");

const bcrypt = require("bcryptjs");

const jwt = require("jsonwebtoken");

const cors = require("cors");

const bodyParser = require("body-parser");

const app = express();

app.use(cors());

app.use(bodyParser.json());

// Create a MySQL connection

const db = mysql.createConnection({

host: "localhost",

user: "root", // your MySQL username

password: "", // your MySQL password

database: "college\_complaints", // database name

});

// Connect to the database

db.connect((err) => {

if (err) {

console.error("Error connecting to database:", err);

} else {

console.log("Connected to MySQL database.");

}

});

// Register student (signup)

app.post("/api/register", (req, res) => {

const { username, password } = req.body;

const hashedPassword = bcrypt.hashSync(password, 10);

db.query(

"INSERT INTO students (username, password) VALUES (?, ?)",

[username, hashedPassword],

(err, result) => {

if (err) {

return res.status(500).json({ message: "Error registering user." });

}

res.status(200).json({ message: "Registration successful" });

}

);

});

// Student login

app.post("/api/login", (req, res) => {

const { username, password } = req.body;

db.query(

"SELECT \* FROM students WHERE username = ?",

[username],

(err, result) => {

if (err) return res.status(500).json({ message: "Database error." });

if (result.length === 0) return res.status(404).json({ message: "User not found." });

const student = result[0];

if (bcrypt.compareSync(password, student.password)) {

const token = jwt.sign({ id: student.id }, "secret\_key", { expiresIn: "1h" });

res.status(200).json({ token });

} else {

res.status(401).json({ message: "Invalid password." });

}

}

);

});

// Admin login

app.post("/api/admin/login", (req, res) => {

const { username, password } = req.body;

if (username === "admin" && password === "admin123") {

const token = jwt.sign({ role: "admin" }, "secret\_key", { expiresIn: "1h" });

res.status(200).json({ token });

} else {

res.status(401).json({ message: "Invalid credentials." });

}

});

// Submit a complaint

app.post("/api/complaints", (req, res) => {

const { student\_id, complaint } = req.body;

db.query(

"INSERT INTO complaints (student\_id, complaint) VALUES (?, ?)",

[student\_id, complaint],

(err, result) => {

if (err) return res.status(500).json({ message: "Error submitting complaint." });

res.status(200).json({ message: "Complaint submitted successfully." });

}

);

});

// Get all complaints for admin

app.get("/api/complaints", (req, res) => {

db.query("SELECT \* FROM complaints", (err, results) => {

if (err) return res.status(500).json({ message: "Error fetching complaints." });

res.status(200).json(results);

});

});

// Start server

app.listen(5000, () => {

console.log("Server running on http://localhost:5000");

});

**Frontend (React + CSS)**

1. **Install the necessary packages for React** (Axios for HTTP requests):

npm install axios

1. **Project Structure:** Here’s a basic structure of your project:

/src

/components

LoginPage.js

ComplaintPage.js

AdminDashboard.js

App.js

/styles

App.css

LoginPage.css

ComplaintPage.css

AdminDashboard.css

**1. LoginPage.js**

import React, { useState } from "react";

import axios from "axios";

import './LoginPage.css';

function LoginPage({ setToken, isAdmin, setIsAdmin }) {

const [username, setUsername] = useState("");

const [password, setPassword] = useState("");

const [message, setMessage] = useState("");

const handleLogin = async () => {

try {

let url = isAdmin ? "http://localhost:5000/api/admin/login" : "http://localhost:5000/api/login";

const response = await axios.post(url, { username, password });

setToken(response.data.token);

localStorage.setItem("token", response.data.token);

setMessage("Login successful");

} catch (err) {

setMessage("Login failed: " + err.response.data.message);

}

};

return (

<div className="login-container">

<h2>{isAdmin ? "Admin Login" : "Student Login"}</h2>

<input

type="text"

className="input-field"

placeholder="Username"

value={username}

onChange={(e) => setUsername(e.target.value)}

/>

<input

type="password"

className="input-field"

placeholder="Password"

value={password}

onChange={(e) => setPassword(e.target.value)}

/>

<button className="login-button" onClick={handleLogin}>Login</button>

<p className="message">{message}</p>

{!isAdmin && <button className="switch-button" onClick={() => setIsAdmin(true)}>Switch to Admin Login</button>}

</div>

);

}

export default LoginPage;

**1. LoginPage.css**

/\* LoginPage.css \*/

.login-container {

width: 300px;

margin: 0 auto;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

background-color: #f9f9f9;

}

h2 {

text-align: center;

margin-bottom: 20px;

}

.input-field {

width: 100%;

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ccc;

border-radius: 4px;

}

.login-button {

width: 100%;

padding: 10px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.login-button:hover {

background-color: #45a049;

}

.switch-button {

width: 100%;

padding: 10px;

background-color: #2196F3;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.switch-button:hover {

background-color: #0b7dda;

}

.message {

color: red;

text-align: center;

}

**2. ComplaintPage.js**

import React, { useState } from "react";

import axios from "axios";

import './ComplaintPage.css';

function ComplaintPage({ token }) {

const [complaint, setComplaint] = useState("");

const [message, setMessage] = useState("");

const submitComplaint = async () => {

try {

const student\_id = 1; // Hardcoded for demo, get from auth token in real app

await axios.post(

"http://localhost:5000/api/complaints",

{ student\_id, complaint },

{ headers: { Authorization: `Bearer ${token}` } }

);

setMessage("Complaint submitted successfully");

} catch (err) {

setMessage("Error submitting complaint");

}

};

return (

<div className="complaint-container">

<h2>Submit Complaint</h2>

<textarea

className="complaint-input"

value={complaint}

onChange={(e) => setComplaint(e.target.value)}

placeholder="Enter your complaint"

/>

<button className="submit-button" onClick={submitComplaint}>Submit</button>

<p className="message">{message}</p>

</div>

);

}

export default ComplaintPage;

**2. ComplaintPage.css**

/\* ComplaintPage.css \*/

.complaint-container {

width: 400px;

margin: 0 auto;

padding: 20px;

border-radius: 8px;

background-color: #f9f9f9;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

margin-bottom: 20px;

}

.complaint-input {

width: 100%;

height: 100px;

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ccc;

border-radius: 4px;

}

.submit-button {

width: 100%;

padding: 10px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.submit-button:hover {

background-color: #45a049;

}

.message {

color: red;

text-align: center;

}

**3. AdminDashboard.js**

import React, { useState, useEffect } from "react";

import axios from "axios";

import './AdminDashboard.css';

function AdminDashboard({ token }) {

const [complaints, setComplaints] = useState([]);

useEffect(() => {

const fetchComplaints = async () => {

try {

const response = await axios.get("http://localhost:5000/api/complaints", {

headers: { Authorization: `Bearer ${token}` },

});

setComplaints(response.data);

} catch (err) {

console.log("Error fetching complaints:", err);

}

};

fetchComplaints();

}, [token]);

return (

<div className="admin-dashboard">

<h2>All Complaints</h2>

<ul>

{complaints.map((complaint) => (

<li key={complaint.id} className="complaint-item">

{complaint.complaint}

</li>

))}

</ul>

</div>

);

}

export default AdminDashboard;

**3. AdminDashboard.css**

/\* AdminDashboard.css \*/

.admin-dashboard {

width: 600px;

margin: 0 auto;

padding: 20px;

border-radius: 8px;

background-color: #f9f9f9;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

margin-bottom: 20px;

}

.complaint-item {

padding: 10px;

border-bottom: 1px solid #ccc;

margin-bottom: 10px;

}

**4. App.js**

import React, { useState } from "react";

import LoginPage from "./components/LoginPage";

import ComplaintPage from "./components/ComplaintPage";

import AdminDashboard from "./components/AdminDashboard";

import './styles/App.css';

function App() {

const [token, setToken] = useState(localStorage.getItem("token"));

const [isAdmin, setIsAdmin] = useState(false);

return (

<div className="App">

{!token ? (

<LoginPage setToken={setToken} isAdmin={isAdmin} setIsAdmin={setIsAdmin} />

) : isAdmin ? (

<AdminDashboard token={token} />

) : (

<ComplaintPage token={token} />

)}

</div>

);

}

export default App;

**5. App.css**

/\* App.css \*/

.App {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

min-height: 100vh;

display: flex;

justify-content: center;

align-items: center;

padding: 20px;

}

**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>**

**HTML: calculator.html**

<!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="calculator.css">

</head>

<body>

<header>

<h1>Simple Calculator</h1>

</header>

<section class="calculator-container">

<div class="calculator">

<input type="text" id="display" disabled>

<table>

<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 colspan="4"><button onclick="appendToDisplay('%')">%</button></td>

</tr>

</table>

</div>

</section>

<footer>

<p>© 2024 Simple Calculator</p>

</footer>

<script src="calculator.js"></script>

</body>

</html>

**CSS: calculator.css**

/\* calculator.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

header {

text-align: center;

margin-bottom: 20px;

}

.calculator-container {

display: flex;

justify-content: center;

}

.calculator {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

#display {

width: 100%;

padding: 15px;

font-size: 24px;

text-align: right;

margin-bottom: 20px;

border: 1px solid #ccc;

border-radius: 5px;

}

table {

width: 100%;

border-spacing: 10px;

}

button {

width: 100%;

padding: 15px;

font-size: 18px;

border: 1px solid #ccc;

background-color: #f1f1f1;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #e1e1e1;

}

footer {

text-align: center;

margin-top: 20px;

}

**JavaScript: calculator.js**

// calculator.js

function appendToDisplay(value) {

document.getElementById("display").value += value;

}

function clearDisplay() {

document.getElementById("display").value = "";

}

function calculateResult() {

let expression = document.getElementById("display").value;

try {

document.getElementById("display").value = eval(expression);

} catch (error) {

document.getElementById("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.**

<?php

// Sample string

$string = "hello world! welcome to PHP";

// a) Transform the string to all uppercase letters

$uppercase = strtoupper($string);

echo "Uppercase: " . $uppercase . "<br>";

// b) Transform the string to all lowercase letters

$lowercase = strtolower($string);

echo "Lowercase: " . $lowercase . "<br>";

// c) Make the string's first character uppercase

$firstUpper = ucfirst($string);

echo "First character uppercase: " . $firstUpper . "<br>";

// d) Make the first character of all words uppercase

$titleCase = ucwords($string);

echo "First character of each word uppercase: " . $titleCase . "<br>";

?>

**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**

/college\_complaints

/assets

/css

style.css

/includes

config.php

/pages

student\_login.php

complaint.php

submit\_complaint.php

admin\_login.php

admin\_dashboard.php

index.php

**1. Database Setup (MySQL)**

CREATE DATABASE college\_complaints;

USE college\_complaints;

CREATE TABLE complaints (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(255) NOT NULL,

complaint\_text TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

role ENUM('student', 'admin') NOT NULL

);

-- Add an admin user (you can change the username and password)

INSERT INTO users (username, password, role) VALUES ('admin', MD5('adminpassword'), 'admin');

**2. Configuration File (config.php)**

<?php

$servername = "localhost";

$username = "root"; // change this to your MySQL username

$password = ""; // change this to your MySQL password

$dbname = "college\_complaints";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**3. Student Login Page (student\_login.php)**

<?php

session\_start();

if (isset($\_SESSION['username'])) {

header("Location: complaint.php");

exit();

}

?>

<!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>

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

</head>

<body>

<div class="login-container">

<h2>Student Login</h2>

<form action="student\_login\_process.php" method="POST">

<label for="username">Username:</label>

<input type="text" name="username" required>

<label for="password">Password:</label>

<input type="password" name="password" required>

<button type="submit">Login</button>

</form>

</div>

</body>

</html>

**4. Student Login Process (student\_login\_process.php)**

<?php

session\_start();

include('config.php');

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

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

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

$sql = "SELECT \* FROM users WHERE username='$username' AND role='student'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

if (password\_verify($password, $row['password'])) {

$\_SESSION['username'] = $username;

header("Location: complaint.php");

} else {

echo "Invalid credentials!";

}

} else {

echo "No user found!";

}

}

?>

**5. Complaint Registration Page (complaint.php)**

<?php

session\_start();

if (!isset($\_SESSION['username'])) {

header("Location: student\_login.php");

exit();

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Register Complaint</title>

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

</head>

<body>

<div class="complaint-container">

<h2>Register Complaint</h2>

<form action="submit\_complaint.php" method="POST">

<label for="complaint">Complaint:</label>

<textarea name="complaint" rows="5" required></textarea>

<button type="submit">Submit Complaint</button>

</form>

</div>

</body>

</html>

**6. Submit Complaint Process (submit\_complaint.php)**

<?php

session\_start();

include('config.php');

if (!isset($\_SESSION['username'])) {

header("Location: student\_login.php");

exit();

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$complaint = $\_POST['complaint'];

$student\_name = $\_SESSION['username'];

$sql = "INSERT INTO complaints (student\_name, complaint\_text) VALUES ('$student\_name', '$complaint')";

if ($conn->query($sql) === TRUE) {

echo "Complaint submitted successfully!";

} else {

echo "Error: " . $conn->error;

}

}

?>

**7. Admin Login Page (admin\_login.php)**

<?php

session\_start();

if (isset($\_SESSION['username'])) {

header("Location: admin\_dashboard.php");

exit();

}

?>

<!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>

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

</head>

<body>

<div class="login-container">

<h2>Admin Login</h2>

<form action="admin\_login\_process.php" method="POST">

<label for="username">Username:</label>

<input type="text" name="username" required>

<label for="password">Password:</label>

<input type="password" name="password" required>

<button type="submit">Login</button>

</form>

</div>

</body>

</html>

**8. Admin Login Process (admin\_login\_process.php)**

<?php

session\_start();

include('config.php');

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

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

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

$sql = "SELECT \* FROM users WHERE username='$username' AND role='admin'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

if (password\_verify($password, $row['password'])) {

$\_SESSION['username'] = $username;

header("Location: admin\_dashboard.php");

} else {

echo "Invalid credentials!";

}

} else {

echo "No user found!";

}

}

?>

**9. Admin Dashboard (admin\_dashboard.php)**

<?php

session\_start();

if (!isset($\_SESSION['username'])) {

header("Location: admin\_login.php");

exit();

}

include('config.php');

$sql = "SELECT \* FROM complaints";

$result = $conn->query($sql);

?>

<!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>

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

</head>

<body>

<div class="dashboard-container">

<h2>Admin Dashboard</h2>

<table>

<thead>

<tr>

<th>Student Name</th>

<th>Complaint</th>

<th>Date</th>

</tr>

</thead>

<tbody>

<?php while($row = $result->fetch\_assoc()) { ?>

<tr>

<td><?php echo $row['student\_name']; ?></td>

<td><?php echo $row['complaint\_text']; ?></td>

<td><?php echo $row['created\_at']; ?></td>

</tr>

<?php } ?>

</tbody>

</table>

</div>

</body>

</html>

**10. CSS (style.css)**

/\* style.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

margin: 0;

padding: 0;

}

h2 {

color: #333;

}

.login-container, .complaint-container, .dashboard-container {

max-width: 500px;

margin: 50px auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

input[type="text"], input[type="password"], textarea {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

padding: 10px 20px;

background-color: #007bff;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

table, th, td {

border: 1px solid #ddd;

}

th, td {

padding: 10px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

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

school-records/

│

├── config/

│ └── config.php # Database connection configuration

│

├── public/

│ ├── index.php # Main page for adding, listing, and deleting students

│ └── validation.js # JavaScript for form validation

│

├── assets/

│ └── css/

│ └── style.css # CSS styling for the application

│

└── README.md # Project description and setup instructions (optional)

**1. Database Setup (MySQL)**

CREATE DATABASE school;

USE school;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

age INT NOT NULL,

email VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**2. Configuration File (config.php)**

<?php

$servername = "localhost";

$username = "root"; // change this to your MySQL username

$password = ""; // change this to your MySQL password

$dbname = "school";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**3. Main Page (index.php)**

<?php

include('config.php');

// Handle adding a student

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['add\_student'])) {

$name = $\_POST['name'];

$age = $\_POST['age'];

$email = $\_POST['email'];

$sql = "INSERT INTO students (name, age, email) VALUES ('$name', '$age', '$email')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

} else {

echo "Error: " . $conn->error;

}

}

// Handle deleting a student

if (isset($\_GET['delete'])) {

$id = $\_GET['delete'];

$sql = "DELETE FROM students WHERE id=$id";

if ($conn->query($sql) === TRUE) {

echo "Record deleted successfully";

} else {

echo "Error: " . $conn->error;

}

}

// Fetch students list

$sql = "SELECT \* FROM students";

$result = $conn->query($sql);

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Records</title>

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

<script src="validation.js"></script>

</head>

<body>

<div class="container">

<h2>Student Records</h2>

<form action="index.php" method="POST" onsubmit="return validateForm()">

<h3>Add New Student</h3>

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br>

<label for="age">Age:</label>

<input type="number" id="age" name="age" required><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br>

<button type="submit" name="add\_student">Add Student</button>

</form>

<h3>List of Students</h3>

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Age</th>

<th>Email</th>

<th>Actions</th>

</tr>

<?php while($row = $result->fetch\_assoc()) { ?>

<tr>

<td><?php echo $row['id']; ?></td>

<td><?php echo $row['name']; ?></td>

<td><?php echo $row['age']; ?></td>

<td><?php echo $row['email']; ?></td>

<td><a href="index.php?delete=<?php echo $row['id']; ?>" onclick="return confirm('Are you sure?')">Delete</a></td>

</tr>

<?php } ?>

</table>

</div>

</body>

</html>

**4. Form Validation (validation.js)**

function validateForm() {

var name = document.getElementById('name').value;

var age = document.getElementById('age').value;

var email = document.getElementById('email').value;

if (name == "" || age == "" || email == "") {

alert("All fields must be filled out");

return false;

}

// Check for valid age

if (isNaN(age) || age <= 0) {

alert("Please enter a valid age");

return false;

}

// Check for valid email

var emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

if (!emailPattern.test(email)) {

alert("Please enter a valid email address");

return false;

}

return true;

}

**5. CSS for Styling (style.css)**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

margin: 0;

padding: 0;

}

.container {

max-width: 800px;

margin: 50px auto;

padding: 20px;

background-color: white;

border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

h2, h3 {

text-align: center;

color: #333;

}

form {

margin: 20px 0;

}

label {

font-weight: bold;

display: block;

margin: 10px 0 5px;

}

input[type="text"], input[type="number"], input[type="email"] {

width: 100%;

padding: 10px;

margin: 5px 0 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

padding: 10px 20px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

table, th, td {

border: 1px solid #ddd;

}

th, td {

padding: 10px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

a {

color: red;

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

**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.**

jquery-list-copy/

│

├── assets/

│ └── css/

│ └── style.css # Optional CSS for styling

│

├── js/

│ └── script.js # jQuery script for list copy and creating new items

│

├── index.html # Main HTML file with lists and buttons

└── README.md # Optional README for project details

**1. HTML Structure**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>jQuery List Copy and Dynamic Element Creation</title>

<link rel="stylesheet" href="style.css"> <!-- Optional CSS file -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script> <!-- jQuery -->

</head>

<body>

<div class="container">

<h2>jQuery List Copy and Create Element</h2>

<!-- First List -->

<h3>List 1</h3>

<ul id="list1">

<li>Apple</li>

<li>Banana</li>

<li>Cherry</li>

<li>Date</li>

</ul>

<!-- Button to copy items -->

<button id="copyButton">Copy to List 2</button>

<!-- Second List -->

<h3>List 2</h3>

<ul id="list2">

<!-- Copied items will appear here -->

</ul>

<!-- Button to create new item -->

<button id="createElementButton">Create New Item</button>

</div>

<script src="script.js"></script> <!-- jQuery Script -->

</body>

</html>

**2. CSS (Optional - style.css)**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

padding: 20px;

}

.container {

width: 50%;

margin: 0 auto;

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

color: #333;

}

h3 {

color: #555;

}

ul {

list-style-type: none;

padding: 0;

}

ul li {

background-color: #f9f9f9;

margin: 5px 0;

padding: 10px;

border-radius: 4px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

button {

display: block;

margin: 20px auto;

padding: 10px 20px;

background-color: #007bff;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

}

button:hover {

background-color: #0056b3;

}

**3. jQuery Script (script.js)**

$(document).ready(function() {

// Copy items from List 1 to List 2

$('#copyButton').click(function() {

// Get all items from list1

$('#list1 li').each(function() {

// Append each item from list1 to list2

$('#list2').append('<li>' + $(this).text() + '</li>');

});

});

// Create a new item dynamically in List 1

$('#createElementButton').click(function() {

var newItem = prompt('Enter a new item:'); // Prompt user for new item

if (newItem) {

// Add the new item to list1

$('#list1').append('<li>' + newItem + '</li>');

}

});

});

**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.**

**1. Project Structure:**

electricity-bill-calculator/

│

├── public/

│ ├── index.html

│ └── style.css

├── views/

│ └── index.ejs

├── app.js

├── package.json

└── node\_modules/

**2. Dependencies (package.json)**

{

"name": "electricity-bill-calculator",

"version": "1.0.0",

"description": "A simple Node.js app to calculate electricity bills.",

"main": "app.js",

"scripts": {

"start": "node app.js"

},

"dependencies": {

"express": "^4.17.1",

"ejs": "^3.1.6"

}

}

**3. Backend (app.js)**

javascript

Copy code

const express = require('express');

const app = express();

// Set view engine

app.set('view engine', 'ejs');

// Serve static files (CSS, images, etc.)

app.use(express.static('public'));

// Body parser middleware for form data

app.use(express.urlencoded({ extended: true }));

// Route for home page

app.get('/', (req, res) => {

res.render('index');

});

// Route to calculate the bill

app.post('/calculate', (req, res) => {

const units = parseInt(req.body.units);

let billAmount = 0;

// Calculating bill based on the given rates

if (units <= 50) {

billAmount = units \* 3.5;

} else if (units <= 150) {

billAmount = (50 \* 3.5) + ((units - 50) \* 4.0);

} else if (units <= 250) {

billAmount = (50 \* 3.5) + (100 \* 4.0) + ((units - 150) \* 5.2);

} else {

billAmount = (50 \* 3.5) + (100 \* 4.0) + (100 \* 5.2) + ((units - 250) \* 6.5);

}

res.render('index', { billAmount, units });

});

// Start server

const port = 3000;

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});

**4. Frontend (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>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body>

<div class="container">

<h1 class="text-center my-4">Electricity Bill Calculator</h1>

<form action="/calculate" method="POST">

<div class="mb-3">

<label for="units" class="form-label">Enter Units Consumed:</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 (typeof billAmount !== 'undefined') { %>

<div class="mt-4">

<h4>Your Bill: Rs. <%= billAmount %></h4>

<p>For <%= units %> units consumed</p>

</div>

<% } %>

</div>

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

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**5. CSS (public/style.css)**

body {

font-family: Arial, sans-serif;

background-color: #f4f7f6;

margin-top: 30px;

}

.container {

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

h1 {

color: #007bff;

}

form {

margin-bottom: 20px;

}

h4 {

color: green;

}

**6. Running the Application**

1. Install the required dependencies:

npm install

1. Start the application:

npm start

1. Open your browser and go to http://localhost:3000/ to see the electricity bill calculator.

**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.**

**1. Project Structure**

electricity-bill-springboot/

│

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/

│ │ │ └── example/

│ │ │ └── ElectricityBillCalculator/

│ │ │ ├── ElectricityBillController.java

│ │ │ └── ElectricityBillApplication.java

│ │ ├── resources/

│ │ │ ├── static/

│ │ │ │ └── style.css

│ │ │ ├── templates/

│ │ │ │ └── index.html

└── pom.xml

**2. pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

</dependency>

</dependencies>

**3. Controller (ElectricityBillController.java)**

package com.example.ElectricityBillCalculator;

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 ElectricityBillController {

@GetMapping("/")

public String showForm() {

return "index";

}

@PostMapping("/calculate")

public String calculateBill(@RequestParam int units, Model model) {

double billAmount = 0;

if (units <= 50) {

billAmount = units \* 3.5;

} else if (units <= 150) {

billAmount = (50 \* 3.5) + ((units - 50) \* 4.0);

} else if (units <= 250) {

billAmount = (50 \* 3.5) + (100 \* 4.0) + ((units - 150) \* 5.2);

} else {

billAmount = (50 \* 3.5) + (100 \* 4.0) + (100 \* 5.2) + ((units - 250) \* 6.5);

}

model.addAttribute("billAmount", billAmount);

model.addAttribute("units", units);

return "index";

}

}

**4. Frontend (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@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body>

<div class="container">

<h1 class="text-center my-4">Electricity Bill Calculator</h1>

<form action="/calculate" method="POST">

<div class="mb-3">

<label for="units" class="form-label">Enter Units Consumed:</label>

<input type="number" class="form-control" id="units" name="units" required>

</div>

<button type="submit" class="btn btn-primary">Calculate Bill</button>

</form>

<div class="mt-4">

<h4>Your Bill: Rs. <span th:text="${billAmount}"></span></h4>

<p>For <span th:text="${units}"></span> units consumed</p>

</div>

</div>

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

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**5. CSS (src/main/resources/static/style.css)**

body {

font-family: Arial, sans-serif;

background-color: #f4f7f6;

margin-top: 30px;

}

.container {

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

h1 {

color: #007bff;

}

form {

margin-bottom: 20px;

}

h4 {

color: green;

}

**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.**

project-folder/

│

├── index.html # Main HTML file for the resume webpage

├── style.css # CSS file for styling the webpage

├── background.mp4 # Video background for the webpage

└── images/

└── profile-pic.jpg # Profile picture (replace with actual image)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Resume Template</title>

    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">

    <style>

        body {

            font-family: 'Montserrat', sans-serif;

            margin: 0;

            padding: 0;

            color: #333;

        }

        #bg-video {

            position: fixed;

            top: 50%;

            left: 50%;

            min-width: 100%;

            min-height: 100%;

            width: auto;

            height: auto;

            z-index: -100;

            transform: translateX(-50%) translateY(-50%);

            background-size: cover;

        }

        .overlay {

            position: absolute;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background-color: rgba(0, 0, 0, 0.5);

            z-index: -99;

        }

        .container {

            display: flex;

            flex-direction: column;

            max-width: 800px;

            margin: 50px auto;

            background-color: rgba(255, 255, 255, 0.9);

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

            overflow: hidden;

            border-radius: 10px;

        }

        .header, .main-content, .sidebar {

            padding: 20px;

        }

        .header {

            display: flex;

            align-items: center;

            background-color: rgba(255, 255, 255, 0.9);

            border-bottom: 1px solid #ddd;

        }

        .profile-pic {

            border-radius: 50%;

            width: 120px;

            height: 120px;

            object-fit: cover;

            margin-right: 20px;

        }

        .header .name-title {

            display: flex;

            flex-direction: column;

            justify-content: center;

        }

        .header .name-title h1 {

            margin: 0;

            font-size: 24px;

            color: #333;

        }

        .header .name-title p {

            margin: 0;

            font-size: 18px;

            color: #555;

        }

        .main-content {

            display: flex;

        }

        .sidebar {

            width: 35%;

            border-right: 1px solid #ddd;

        }

        .sidebar h2 {

            font-size: 20px;

            color: #333;

            margin-bottom: 10px;

            border-bottom: 1px solid #ddd;

            padding-bottom: 5px;

        }

        .sidebar ul {

            list-style-type: none;

            padding: 0;

        }

        .sidebar ul li {

            margin-bottom: 10px;

            font-size: 14px;

            color: #555;

        }

        .sidebar ul li::before {

            content: '•';

            color: #6C63FF;

            font-weight: bold;

            display: inline-block;

            width: 1em;

            margin-left: -1em;

        }

        .content {

            width: 65%;

            padding-left: 20px;

        }

        .content h2 {

            font-size: 20px;

            color: #333;

            margin-bottom: 10px;

            border-bottom: 1px solid #ddd;

            padding-bottom: 5px;

        }

        .content p, .content ul {

            font-size: 14px;

            color: #555;

        }

        .content ul {

            list-style-type: none;

            padding: 0;

        }

        .content ul li {

            margin-bottom: 20px;

        }

        .content ul li p {

            margin: 5px 0;

        }

    </style>

</head>

<body>

    <video autoplay loop muted id="bg-video">

        <source src="background.mp4" type="video/mp4">

    </video>

    <div class="overlay"></div>

    <div class="container">

        <div class="header">

            <img src="images.png" alt="Profile Picture" class="profile-pic">

            <div class="name-title">

                <h1>Bharati Borhade</h1>

                <p>Software Developer</p>

            </div>

        </div>

        <div class="main-content">

            <div class="sidebar">

                <h2>Contact</h2>

                <ul>

                    <li><a href="tel:1234567890">123-456-7890</a></li>

                    <li><a href="mailto:hello@reallygreatsite.com">bharatiborhade23@gmail.com</a></li>

                    <li>Kondhwa Bk, Pune, Maharashtra</li>

                </ul>

                <h2>Skills</h2>

                <ul>

                    <li>HTML</li>

                    <li>CSS</li>

                    <li>Javascript</li>

                    <li>React.js</li>

                    <li>Node.js</li>

                    <li>Java</li>

                    <li>C</li>

                    <li>MYSQL</li>

                </ul>

                <h2>Education</h2>

                <ul>

                    <li>

                        <strong>Secondary School</strong>

                        <p>SBP Vidyalaya, Junnar</p>

                    </li>

                    <li>

                        <strong>High School</strong>

                        <p>SSC College, Junnar</p>

                        <p>2020 - 2022</p>

                    </li>

                    <li>

                        <strong>Bachelor of Technology</strong>

                        <strong>(Computer Engineering)</strong>

                        <p>VIT, Pune</p>

                        <p>2022 - 2026</p>

                    </li>

                </ul>

            </div>

            <div class="content">

                <section>

                    <h2>About Me</h2>

                    <p>As a software engineer, I have a strong passion for developing innovative and efficient solutions to complex problems. With a solid foundation in computer science and a keen eye for detail, I excel at translating business requirements into functional and scalable software applications.</p>

                </section>

                <section>

                    <h2>Projects</h2>

                    <ul>

                        <li>

                            <h3>1) P2P Carpooling using Blockchain</h3>

                            <ul>

                                <li>Technology: Blockchain, Node.js, MongoDB, React.js</li>

                                <li>It is a decentralized approach that enables individuals to share their cars with others while traveling, eliminating the need for a middleman.</li>

                            </ul>

                        </li>

                        <li>

                            <h3>2) Crop Yield Prediction</h3>

                            <ul>

                                <li>Technology: Python, ML</li>

                                <li>The project aims to help farmers make precise decisions regarding crops to improve yield based on various factors.</li>

                            </ul>

                        </li>

                        <li>

                            <h3>3) Doctor Appointment Management System</h3>

                            <ul>

                                <li>Technology: HTML, CSS, JS, PHP, MYSQL</li>

                                <li>Technology driven solution designed to streamline and simplify the process of scheduling medical appointments.</li>

                            </ul>

                        </li>

                        <li>

                            <h3>4) Brightmind:An AI Avatar Interview Taker</h3>

                            <ul>

                                <li>Technology: Python, HTML, CSS, JS, Flask</li>

                                <li>Project generate questions from the text input and allows user to take MCQ test and virtual interview with an ai Avatar.</li>

                            </ul>

                        </li>

                    </ul>

                </section>

            </div>

        </div>

    </div>

</body>

</html>

**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.**

1. **Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Courses</title>

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

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

</head>

<body>

    <!-- Navigation Menu -->

    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Projects</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="sdp.html">SDP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="edi.html">EDI</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="cp.html">CP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="dt.html">DT</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-5 pt-5">

        <div class="row">

            <div class="col-md-4">

                <section id="Software-Development-Project" class="card mb-4">

                    <div class="card-body">

                        <h2><a href="sdp.html" class="custom-link">Software Development Project (SDP)</a></h2>

                    </div>

                </section>

                <section id="Engineering-Design-And-Innovation" class="card mb-4">

                    <div class="card-body">

                        <h2><a href="sdp.html" class="custom-link">Engineering Design and Innovation (EDI)</a></h2>

                    </div>

                </section>

                <section id="Course-Projects" class="card mb-4">

                    <div class="card-body">

                        <h2><a href="sdp.html" class="custom-link">Course Projects (CP)</a></h2>

                    </div>

                </section>

                <section id="Design-Thinking" class="card mb-4">

                    <div class="card-body">

                        <h2><a href="sdp.html" class="custom-link">Design Thinking (DT)</a></h2>

                    </div>

                </section>

            </div>

        </div>

    </div>

    <img src="course.jpg" alt="Descriptive Alt Text" class="fixed-image">

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="scripts.js"></script>

</body>

</html>

1. **Sdp.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Software Development Project (SDP)</title>

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

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

</head>

<body>

    <!-- Navigation Menu -->

    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Projects</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="index.html">Home</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="edi.html">EDI</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="cp.html">CP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="dt.html">DT</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-5 pt-5">

        <div class="row">

            <div class="col-md-8">

                <section class="card mb-4">

                    <div class="card-body">

                        <h2>Software Development Project (SDP)</h2>

                        <p>

                            The Software Development Project (SDP) is a comprehensive project designed to give students hands-on experience in developing software applications from scratch. This project encompasses various stages of the software development lifecycle, including:

                        </p>

                        <ul>

                            <li><strong>Requirement Analysis:</strong> Identifying the needs and requirements of the project.</li>

                            <li><strong>Design:</strong> Creating the architecture and design of the software system.</li>

                            <li><strong>Implementation:</strong> Writing and coding the software based on the design.</li>

                            <li><strong>Testing:</strong> Verifying and validating the software to ensure it meets the requirements.</li>

                            <li><strong>Deployment:</strong> Releasing the software to the end-users.</li>

                            <li><strong>Maintenance:</strong> Providing ongoing support and updates to the software.</li>

                        </ul>

                        <p>

                            This project allows students to work in teams, apply their technical skills, and understand the real-world challenges of software development. It is a crucial part of the curriculum, designed to prepare students for careers in software engineering and development.

                        </p>

                        <p>

                            Key skills gained from this project include:

                        </p>

                        <ul>

                            <li>Project management and teamwork.</li>

                            <li>Technical skills in various programming languages and tools.</li>

                            <li>Problem-solving and critical thinking.</li>

                            <li>Communication skills through documentation and presentations.</li>

                        </ul>

                    </div>

                </section>

            </div>

        </div>

    </div>

    <img src="course.jpg" alt="Descriptive Alt Text" class="fixed-image">

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="scripts.js"></script>

</body>

</html>

1. **Dt.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Design Thinking (DT)</title>

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

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

</head>

<body>

    <!-- Navigation Menu -->

    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Projects</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="index.html">Home</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="sdp.html">SDP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="edi.html">EDI</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="cp.html">CP</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-5 pt-5">

        <div class="row">

            <div class="col-md-8">

                <section class="card mb-4">

                    <div class="card-body">

                        <h2>Design Thinking (DT)</h2>

                        <p>

                            Design Thinking (DT) is a human-centered approach to innovation that emphasizes understanding users, challenging assumptions, and redefining problems to identify alternative strategies and solutions. The process is iterative and involves several stages to foster creativity and problem-solving. The main aspects of Design Thinking include:

                        </p>

                        <ul>

                            <li><strong>Empathize:</strong> Understanding the needs, experiences, and challenges of users through observation and interaction.</li>

                            <li><strong>Define:</strong> Clearly articulating the problem based on insights gathered during the empathize phase.</li>

                            <li><strong>Ideate:</strong> Generating a wide range of ideas and potential solutions to address the defined problem.</li>

                            <li><strong>Prototype:</strong> Creating tangible representations of ideas to explore how they work in practice.</li>

                            <li><strong>Test:</strong> Evaluating prototypes with users to gather feedback and refine the solutions.</li>

                        </ul>

                        <p>

                            Design Thinking encourages an iterative process where solutions are continually tested and improved based on user feedback. It is a powerful methodology for solving complex problems and creating innovative solutions that are aligned with user needs and expectations.

                        </p>

                        <p>

                            Key skills gained from Design Thinking include:

                        </p>

                        <ul>

                            <li>Empathy and user-centered design.</li>

                            <li>Creative problem-solving and ideation.</li>

                            <li>Prototyping and testing techniques.</li>

                            <li>Effective communication and collaboration skills.</li>

                        </ul>

                    </div>

                </section>

            </div>

        </div>

    </div>

    <img src="course.jpg" alt="Descriptive Alt Text" class="fixed-image">

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="scripts.js"></script>

</body>

</html>

1. **Edi.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Engineering Design and Innovation (EDI)</title>

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

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

</head>

<body>

    <!-- Navigation Menu -->

    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Projects</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="index.html">Home</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="sdp.html">SDP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="cp.html">CP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="dt.html">DT</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-5 pt-5">

        <div class="row">

            <div class="col-md-8">

                <section class="card mb-4">

                    <div class="card-body">

                        <h2>Engineering Design and Innovation (EDI)</h2>

                        <p>

                            Engineering Design and Innovation (EDI) is a key component of the engineering curriculum aimed at fostering creativity, problem-solving skills, and innovative thinking. The course emphasizes the application of engineering principles to design and develop innovative solutions to real-world problems. The major aspects of EDI include:

                        </p>

                        <ul>

                            <li><strong>Problem Identification:</strong> Recognizing and defining problems that need innovative solutions.</li>

                            <li><strong>Concept Generation:</strong> Brainstorming and developing creative ideas and concepts for solving the identified problems.</li>

                            <li><strong>Design Development:</strong> Creating detailed designs and prototypes based on selected concepts.</li>

                            <li><strong>Testing and Evaluation:</strong> Assessing prototypes to ensure they meet design specifications and requirements.</li>

                            <li><strong>Implementation:</strong> Finalizing the design and preparing for practical application or production.</li>

                        </ul>

                        <p>

                            The EDI course promotes hands-on learning through projects and collaboration, allowing students to explore engineering challenges and develop their technical and creative skills. It prepares students to tackle complex problems and innovate within their field of study.

                        </p>

                        <p>

                            Key skills gained from EDI include:

                        </p>

                        <ul>

                            <li>Creative and critical thinking.</li>

                            <li>Technical skills in design and prototyping.</li>

                            <li>Project management and teamwork.</li>

                            <li>Communication skills for presenting and documenting designs.</li>

                        </ul>

                    </div>

                </section>

            </div>

        </div>

    </div>

    <img src="course.jpg" alt="Descriptive Alt Text" class="fixed-image">

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="scripts.js"></script>

</body>

</html>

1. **Cp.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Course Projects (CP)</title>

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

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

</head>

<body>

    <!-- Navigation Menu -->

    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Projects</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="index.html">Home</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="sdp.html">SDP</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="edi.html">EDI</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="dt.html">DT</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-5 pt-5">

        <div class="row">

            <div class="col-md-8">

                <section class="card mb-4">

                    <div class="card-body">

                        <h2>Course Projects (CP)</h2>

                        <p>

                            The Course Projects (CP) are an essential component of the engineering curriculum designed to provide students with practical experience in applying theoretical knowledge to real-world problems. These projects span a range of topics and allow students to work on hands-on assignments that reinforce their learning and develop their skills. Key aspects of CP include:

                        </p>

                        <ul>

                            <li><strong>Project Selection:</strong> Choosing a project topic based on interests and curriculum requirements.</li>

                            <li><strong>Research and Analysis:</strong> Conducting research and analysis to understand the problem and devise potential solutions.</li>

                            <li><strong>Design and Development:</strong> Designing and developing prototypes or solutions as per project requirements.</li>

                            <li><strong>Implementation:</strong> Implementing the project, which may include coding, building models, or other practical tasks.</li>

                            <li><strong>Presentation:</strong> Presenting the project findings, methodologies, and outcomes to peers and instructors.</li>

                        </ul>

                        <p>

                            Course Projects offer a platform for students to apply their technical skills, work in teams, and gain experience in managing and executing projects. They are an important part of preparing students for their professional careers.

                        </p>

                        <p>

                            Key skills gained from Course Projects include:

                        </p>

                        <ul>

                            <li>Practical application of theoretical concepts.</li>

                            <li>Project management and teamwork.</li>

                            <li>Technical skills in various areas such as programming, engineering design, and analysis.</li>

                            <li>Effective communication through presentations and documentation.</li>

                        </ul>

                    </div>

                </section>

            </div>

        </div>

    </div>

    <img src="course.jpg" alt="Descriptive Alt Text" class="fixed-image">

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script src="scripts.js"></script>

</body>

</html>

1. **Style.css**

/\* Include Google Fonts \*/

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&family=Roboto:wght@400;500&display=swap');

body {

    background-color: #fce4ec;

    font-family: 'Roboto', sans-serif;

    display: flex;

    flex-direction: column;

    min-height: 100vh;

    margin: 0;

    color: #333;

}

#wrapper {

    display: flex;

    flex: 1;

}

#sidebar {

    width: 250px;

    height: 100vh;

    background-color: #ffcc80;

    padding-top: 20px;

    position: fixed;

    transition: all 0.3s;

}

.navbar {

    background-color: #ff80ab;

}

.navbar-custom .navbar-brand,

.navbar-custom .nav-link {

    color: #fff;

    font-family: 'Poppins', sans-serif;

}

.navbar-custom .nav-link:hover {

    color: #ffd54f;

}

.sidebar-heading {

    padding: 15px;

    font-size: 1.25rem;

    font-weight: 600;

    font-family: 'Poppins', sans-serif;

    color: #fff;

    text-align: center;

}

.list-group-item {

    border: none;

    padding: 10px 15px;

    color: #fff;

    background-color: #ffcc80;

    transition: background-color 0.3s, color 0.3s;

    font-family: 'Roboto', sans-serif;

}

.list-group-item:hover {

    background-color: #ffb74d;

    color: #fff;

}

.list-group-item.active {

    background-color: #ffb74d;

    color: #fff;

}

#page-content-wrapper {

    margin-left: 250px;

    padding: 20px;

    width: 100%;

    transition: margin-left 0.3s;

}

.nav-pills .nav-link {

    border-radius: 0.25rem;

    font-family: 'Roboto', sans-serif;

}

.nav-pills .nav-link.active {

    background-color: #ff80ab;

    color: #fff;

}

.tab-content {

    margin-top: 20px;

}

.tab-pane {

    background-color: #fff;

    border: 1px solid #dee2e6;

    border-radius: 0.25rem;

    padding: 20px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    animation: fadeIn 0.5s;

}

.card {

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    border-radius: 0.5rem;

    border: none;

}

.card-body {

    padding: 20px;

    background-color: #ffe0b2;

    box-shadow: 4px 4px 8px rgba(0, 0, 0, 0.2);

    font-family: 'Roboto', sans-serif;

}

.card-title {

    font-size: 1.25rem;

    font-family: 'Poppins', sans-serif;

}

.fixed-image {

    position: fixed;

    top: 0;

    right: 0;

    width: 40vw;

    height: 100vh;

    object-fit: cover;

}

.custom-link {

    color: black;

    text-decoration: none;

}

footer {

    background-color: #ff80ab;

    color: #fff;

    text-align: center;

    padding: 10px 0;

    position: fixed;

    bottom: 0;

    left: 0;

    width: 100%;

    font-family: 'Roboto', sans-serif;

}

**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.**

number-list-sorting/

│

├── index.html

│

├── assets/

│ ├── images/

│ └── fonts/

│

├── css/

│ └── style.css

│

├── js/

│ └── script.js

│

└── README.md

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Number List and Sorting</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

    <style>

        body {

            background: linear-gradient(60deg, #2a5298, #1e3c72);

            height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            color: #fff;

            font-family: 'Arial', sans-serif;

        }

        .container {

            background: rgba(0, 0, 0, 0.8);

            padding: 20px;

            border-radius: 15px;

            box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);

            max-width: 1200px;

            text-align: center;

        }

        h1, h2 {

            margin-bottom: 20px;

            color: #f9ca24;

        }

        #numberInput {

            width: 30%;

            padding: 10px;

            border-radius: 5px;

            border: none;

            margin-bottom: 10px;

        }

        select {

            width: 50%;

            padding: 10px;

            border-radius: 5px;

            border: none;

            height: 150px;

            background: #1e3c72;

            color: #fff;

            margin-bottom: 10px;

        }

        .btn-custom {

            background-color: #f0932b;

            border: none;

            color: #fff;

            padding: 10px 10px;

            margin-top: 10px;

            border-radius: 5px;

            cursor: pointer;

            width: 50%;

        }

        .btn-custom:hover {

            background-color: #eb4d4b;

        }

        .flex-container {

            display: flex;

            justify-content: space-between;

            flex-wrap: wrap;

        }

        .flex-item {

            margin: 10px;

            flex: 1;

        }

        .sort-buttons {

            display: flex;

            flex-direction: column;

            justify-content: center;

            align-items: center;

            gap: 10px;

        }

        .search-container {

            margin-top: 20px;

        }

        .result-message {

            color: #f6e58d;

            font-size: 16px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Sorting And Searching</h1>

        <div class="flex-container">

            <div class="flex-item">

                <input type="text" id="numberInput" placeholder="Enter a number">

                <button id="addNumber" class="btn-custom">Add Number</button>

            </div>

        </div>

        <div class="flex-container">

            <div class="flex-item">

                <h2>Number List</h2>

                <select multiple id="numberList"></select>

                <button id="clearList" class="btn-custom">Clear List</button>

            </div>

            <div class="flex-item sort-buttons">

                <button id="bubbleSort" class="btn-custom">Bubble Sort</button>

                <button id="insertionSort" class="btn-custom">Insertion Sort</button>

                <button id="mergeSort" class="btn-custom">Merge Sort</button>

            </div>

            <div class="flex-item">

                <h2>Sorted Number List</h2>

                <select multiple id="sortedNumberList"></select>

                <button id="clearSortedList" class="btn-custom">Clear Sorted List</button>

            </div>

        </div>

        <div class="flex-container search-container">

            <div class="flex-item">

                <input type="text" id="searchNumberInput" placeholder="Enter number to search">

                <button id="searchNumber" class="btn-custom">Search Number</button>

                <p id="searchResult" class="result-message"></p>

            </div>

        </div>

    </div>

    <script>

        $(document).ready(function () {

            $('#addNumber').click(function () {

                let numberInput = $('#numberInput').val();

                if ($.isNumeric(numberInput)) {

                    $('#numberList').append(new Option(numberInput, numberInput));

                    $('#numberInput').val('');

                } else {

                    alert("Please enter a valid number.");

                }

            });

            $('#clearList').click(function () {

                $('#numberList').empty();

            });

            $('#clearSortedList').click(function () {

                $('#sortedNumberList').empty();

            });

            function sortNumberArray(sortFunction) {

                let numberArray = [];

                $('#numberList option').each(function () {

                    numberArray.push(parseInt($(this).val()));

                });

                numberArray = sortFunction(numberArray);

                $('#sortedNumberList').empty();

                numberArray.forEach(number => {

                    $('#sortedNumberList').append(new Option(number, number));

                });

            }

            $('#bubbleSort').click(function () {

                sortNumberArray(function (numberArray) {

                    for (let i = 0; i < numberArray.length - 1; i++) {

                        for (let j = 0; j < numberArray.length - i - 1; j++) {

                            if (numberArray[j] > numberArray[j + 1]) {

                                [numberArray[j], numberArray[j + 1]] = [numberArray[j + 1], numberArray[j]];

                            }

                        }

                    }

                    return numberArray;

                });

            });

            $('#insertionSort').click(function () {

                sortNumberArray(function (numberArray) {

                    for (let i = 1; i < numberArray.length; i++) {

                        let key = numberArray[i];

                        let j = i - 1;

                        while (j >= 0 && numberArray[j] > key) {

                            numberArray[j + 1] = numberArray[j];

                            j = j - 1;

                        }

                        numberArray[j + 1] = key;

                    }

                    return numberArray;

                });

            });

            $('#mergeSort').click(function () {

                sortNumberArray(function mergeSort(array) {

                    if (array.length <= 1) return array;

                    const mid = Math.floor(array.length / 2);

                    const left = array.slice(0, mid);

                    const right = array.slice(mid);

                    return merge(mergeSort(left), mergeSort(right));

                });

                function merge(left, right) {

                    let resultArray = [], leftIndex = 0, rightIndex = 0;

                    while (leftIndex < left.length && rightIndex < right.length) {

                        if (left[leftIndex] < right[rightIndex]) {

                            resultArray.push(left[leftIndex]);

                            leftIndex++;

                        } else {

                            resultArray.push(right[rightIndex]);

                            rightIndex++;

                        }

                    }

                    return resultArray.concat(left.slice(leftIndex)).concat(right.slice(rightIndex));

                }

            });

            $('#searchNumber').click(function () {

                let searchNumber = $('#searchNumberInput').val();

                if ($.isNumeric(searchNumber)) {

                    let found = false;

                    $('#numberList option').each(function (index) {

                        if ($(this).val() === searchNumber) {

                            $('#searchResult').text(`Number found at position: ${index + 1}`);

                            found = true;

                            return false;

                        }

                    });

                    if (!found) {

                        $('#searchResult').text("Number not found in the list.");

                    }

                } else {

                    alert("Please enter a valid number.");

                }

            });

        });

    </script>

</body>

</html>

**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.**

**Backend: Spring Boot**

electricity-bill-backend/

├── src/

│ ├── main/

│ ├── java/com/example/electricitybill/

│ ├── ElectricityBillApplication.java

│ ├── controller/

│ └── BillController.java

│ ├── model/

│ └── BillRequest.java

│ ├── service/

│ └── BillService.java

│ ├── resources/

│ ├── application.properties

**Frontend: React**

electricity-bill-frontend/

├── public/

│ ├── index.html

├── src/

│ ├── components/

│ └── BillCalculator.js

│ ├── App.js

│ ├── index.js

│ ├── styles.css

**Step 1: Spring Boot Backend**

**1.1 ElectricityBillApplication.java**

package com.example.electricitybill;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ElectricityBillApplication {

public static void main(String[] args) {

SpringApplication.run(ElectricityBillApplication.class, args);

}

}

**1.2 BillRequest.java (Model)**

package com.example.electricitybill.model;

public class BillRequest {

private int units;

// Getters and Setters

public int getUnits() {

return units;

}

public void setUnits(int units) {

this.units = units;

}

}

**1.3 BillService.java (Service Layer)**

package com.example.electricitybill.service;

import org.springframework.stereotype.Service;

@Service

public class BillService {

public double calculateBill(int 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;

}

}

**1.4 BillController.java**

package com.example.electricitybill.controller;

import com.example.electricitybill.model.BillRequest;

import com.example.electricitybill.service.BillService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/bill")

@CrossOrigin(origins = "http://localhost:3000")

public class BillController {

@Autowired

private BillService billService;

@PostMapping("/calculate")

public double calculateBill(@RequestBody BillRequest billRequest) {

return billService.calculateBill(billRequest.getUnits());

}

}

**1.5 application.properties**

server.port=8080

spring.datasource.url=jdbc:mysql://localhost:3306/electricitydb

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

**Step 2: React Frontend**

**2.1 Install Required Dependencies**

npx create-react-app electricity-bill-frontend

cd electricity-bill-frontend

npm install axios bootstrap

**2.2 BillCalculator.js**

import React, { useState } from "react";

import axios from "axios";

import "bootstrap/dist/css/bootstrap.min.css";

const BillCalculator = () => {

const [units, setUnits] = useState("");

const [bill, setBill] = useState(null);

const calculateBill = async () => {

try {

const response = await axios.post("http://localhost:8080/api/bill/calculate", { units });

setBill(response.data);

} catch (error) {

console.error("Error calculating bill:", error);

}

};

return (

<div className="container mt-5">

<h1 className="text-center mb-4">Electricity Bill Calculator</h1>

<div className="card shadow">

<div className="card-body">

<div className="mb-3">

<label htmlFor="units" className="form-label">

Enter Units Consumed:

</label>

<input

type="number"

className="form-control"

id="units"

value={units}

onChange={(e) => setUnits(e.target.value)}

placeholder="Enter number of units"

/>

</div>

<button className="btn btn-primary" onClick={calculateBill}>

Calculate Bill

</button>

{bill !== null && (

<div className="mt-3">

<h4>Bill Amount: ₹ {bill.toFixed(2)}</h4>

</div>

)}

</div>

</div>

</div>

);

};

export default BillCalculator;

**2.3 App.js**

from "react";

import BillCalculator from "./components/BillCalculator";

import "./styles.css";

function App() {

return (

<div>

<BillCalculator />

</div>

);

}

export default App;

**2.4 styles.css**

body {

background-color: #f8f9fa;

font-family: Arial, sans-serif;

}

.card {

margin: 0 auto;

max-width: 500px;

}

h1 {

color: #007bff;

}

**2.5 Backend Setup**

mvn spring-boot:run

**2.6 Frontend Setup**

Run the React application:

bash

Copy code

npm start

**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.**

electricity-bill-php/

├── index.php

├── style.css

├── js/

│ └── script.js

├── images/

└── bootstrap/

└── bootstrap.min.css

* index.php: The main PHP file for handling the UI and logic.
* style.css: Custom styles (optional, since we use Bootstrap).
* script.js: jQuery file to handle dynamic functionality.
* bootstrap.min.css: You can download Bootstrap from the official site or use the CDN link.

**2. Download Bootstrap (optional)**

<!-- Add this to the <head> section of your HTML -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

**3. Create the HTML Form (index.php)**

<!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://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body>

<div class="container mt-5">

<h2 class="text-center">Electricity Bill Calculator</h2>

<div class="row justify-content-center mt-4">

<div class="col-md-6">

<!-- Form to get units from user -->

<form id="bill-form" method="post">

<div class="form-group">

<label for="units">Enter Units Consumed:</label>

<input type="number" class="form-control" id="units" name="units" required>

</div>

<button type="submit" class="btn btn-primary btn-block">Calculate Bill</button>

</form>

<!-- Result section -->

<div id="bill-result" class="mt-3">

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$units = $\_POST['units'];

$bill = 0;

// Calculate the bill based on the given conditions

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='alert alert-success'>Total Bill: ₹" . number\_format($bill, 2) . "</div>";

}

?>

</div>

</div>

</div>

</div>

<!-- jQuery and Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script src="js/script.js"></script>

</body>

</html>

**4. Add Custom JavaScript (script.js)**

$(document).ready(function() {

// Handle form submission

$('#bill-form').submit(function(e) {

e.preventDefault(); // Prevent form submission to allow the custom script to work

var units = $('#units').val();

// Check if input is valid

if (units <= 0) {

alert("Please enter a valid number of units.");

} else {

// Submit the form via POST method

$(this).unbind('submit').submit();

}

});

});

**5. Add Some Custom CSS (style.css)**

body {

background-color: #f4f4f9;

}

h2 {

font-size: 32px;

}

#bill-result {

font-size: 20px;

font-weight: bold;

}

**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%).**

vit-results-project

│

├── vit-results-backend (Spring Boot Project in Eclipse)

│ ├── src

│ │ ├── main

│ │ │ ├── java

│ │ │ │ └── com

│ │ │ │ └── example

│ │ │ │ └── vitresults

│ │ │ │ ├── controller

│ │ │ │ │ └── StudentController.java

│ │ │ │ ├── model

│ │ │ │ │ └── Student.java

│ │ │ │ ├── repository

│ │ │ │ │ └── StudentRepository.java

│ │ │ │ └── VitResultsApplication.java

│ │ ├── resources

│ │ │ └── application.properties

│ ├── pom.xml

│

├── vit-results-frontend (React Project)

│ ├── public

│ │ └── index.html

│ ├── src

│ │ ├── components

│ │ │ └── App.js

│ │ ├── App.css

│ │ ├── index.js

│ ├── package.json

**Frontend Project Setup**: npx create-react-app vit-results-frontend

cd vit-results-frontend

npm install axios

**Create Database and Table**

CREATE DATABASE vit\_results;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

subject1\_mse INT,

subject1\_ese INT,

subject2\_mse INT,

subject2\_ese INT,

subject3\_mse INT,

subject3\_ese INT,

subject4\_mse INT,

subject4\_ese INT

);

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/vit\_results

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**2.3 Create Model Class**

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; // Auto-generated ID

private String studentName;

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

}

**2.4 Create Repository**

package com.example.vitresults.repository;

import com.example.vitresults.model.Student;

import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {

}

**2.5 Create Controller**

package com.example.vitresults.controller;

import com.example.vitresults.model.Student;

import com.example.vitresults.repository.StudentRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/students")

public class StudentController {

@Autowired

private StudentRepository studentRepository;

// Get all students

@GetMapping

public List<Student> getAllStudents() {

return studentRepository.findAll();

}

// Add new student

@PostMapping

public Student addStudent(@RequestBody Student student) {

Student savedStudent = studentRepository.save(student);

double totalMarks = calculateResult(savedStudent.getId());

System.out.println("Calculated Result: " + totalMarks);

return savedStudent; // Return saved student

}

@GetMapping("/result/{id}")

public double calculateResult(@PathVariable Long id) {

Student student = studentRepository.findById(id).orElse(null);

if (student != null) {

double totalMarks = 0;

totalMarks += student.getSubject1MSE() \* 0.30 + student.getSubject1ESE() \* 0.70;

totalMarks += student.getSubject2MSE() \* 0.30 + student.getSubject2ESE() \* 0.70;

totalMarks += student.getSubject3MSE() \* 0.30 + student.getSubject3ESE() \* 0.70;

totalMarks += student.getSubject4MSE() \* 0.30 + student.getSubject4ESE() \* 0.70;

return totalMarks;

}

return 0;

}

}

**3. Set Up React Frontend**

**3.1 Create React Project**

Use **Create React App** to set up your frontend:

npx create-react-app vit-results-frontend

cd vit-results-frontend

Install Axios to make API calls to the Spring Boot backend:

npm install axios

**3.2 Create Components**

Create a React component to submit student marks and display the result.

**3.2.1 App.js**

import React, { useState } from 'react';

import axios from 'axios';

function App() {

// State variables for form inputs

const [studentName, setStudentName] = 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);

// Handle form submission

const handleSubmit = async (event) => {

event.preventDefault();

// Create student object

const student = {

studentName,

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 {

// Send student data to the backend

const res = await axios.post('http://localhost:8080/api/students', student);

const studentId = res.data.id; // Get student ID

// Fetch the result for the student based on the ID

const resultRes = await axios.get(`http://localhost:8080/api/students/result/${studentId}`);

setResult(resultRes.data); // Display the result

} catch (error) {

console.error('Error while submitting student data', error);

}

};

return (

<div className="container mt-5">

<h2>Student Marks Input</h2>

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Student Name</label>

<input

type="text"

className="form-control"

value={studentName}

onChange={(e) => setStudentName(e.target.value)}

required

/>

</div>

<div className="form-group">

<label>Subject 1 MSE</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</label>

<input

type="number"

className="form-control"

value={subject1ESE}

onChange={(e) => setSubject1ESE(e.target.value)}

required

/>

</div>

<div className="form-group">

<label>Subject 2 MSE</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</label>

<input

type="number"

className="form-control"

value={subject2ESE}

onChange={(e) => setSubject2ESE(e.target.value)}

required

/>

</div>

<div className="form-group">

<label>Subject 3 MSE</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</label>

<input

type="number"

className="form-control"

value={subject3ESE}

onChange={(e) => setSubject3ESE(e.target.value)}

required

/>

</div>

<div className="form-group">

<label>Subject 4 MSE</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</label>

<input

type="number"

className="form-control"

value={subject4ESE}

onChange={(e) => setSubject4ESE(e.target.value)}

required

/>

</div>

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

</form>

{result && (

<div className="mt-4">

<h3>Result: {result}</h3>

</div>

)}

</div>

);

}

export default App;

**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%).**

vit\_student\_result/

│

├── index.html

├── result.html

├── css/

│ └── styles.css

├── js/

│ └── script.js

├── images/

├── assets/

├── submit.php

├── calculate\_result.php

└── README.md

CREATE DATABASE vit\_results;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

subject1\_mse INT,

subject1\_ese INT,

subject2\_mse INT,

subject2\_ese INT,

subject3\_mse INT,

subject3\_ese INT,

subject4\_mse INT,

subject4\_ese INT

);

2.1 PHP Database Connection(db.php)

<?php

$servername = "localhost";

$username = "root"; // Default MySQL username

$password = ""; // Default MySQL password (leave blank if no password)

$dbname = "vit\_results"; // Name of the database

// Create connection

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

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**2.2 Student Data Insertion(submit.php)**

<?php

include 'db.php';

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$student\_name = $\_POST['student\_name'];

$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'];

$sql = "INSERT INTO students (student\_name, subject1\_mse, subject1\_ese, subject2\_mse, subject2\_ese, subject3\_mse, subject3\_ese, subject4\_mse, subject4\_ese)

VALUES ('$student\_name', '$subject1\_mse', '$subject1\_ese', '$subject2\_mse', '$subject2\_ese', '$subject3\_mse', '$subject3\_ese', '$subject4\_mse', '$subject4\_ese')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

}

?>

2.3 Result Calculation(calculate\_result.php)

<?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) {

$row = $result->fetch\_assoc();

$subject1\_result = $row['subject1\_mse'] \* 0.30 + $row['subject1\_ese'] \* 0.70;

$subject2\_result = $row['subject2\_mse'] \* 0.30 + $row['subject2\_ese'] \* 0.70;

$subject3\_result = $row['subject3\_mse'] \* 0.30 + $row['subject3\_ese'] \* 0.70;

$subject4\_result = $row['subject4\_mse'] \* 0.30 + $row['subject4\_ese'] \* 0.70;

$total\_result = $subject1\_result + $subject2\_result + $subject3\_result + $subject4\_result;

echo "<h3>Result for " . $row['student\_name'] . "</h3>";

echo "Subject 1 Result: " . $subject1\_result . "<br>";

echo "Subject 2 Result: " . $subject2\_result . "<br>";

echo "Subject 3 Result: " . $subject3\_result . "<br>";

echo "Subject 4 Result: " . $subject4\_result . "<br>";

echo "Total Result: " . $total\_result . "<br>";

} else {

echo "No student found with the given ID.";

}

$conn->close();

}

?>

**3. Set Up Frontend**

**3.1 HTML Form for Input**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>VIT Student Result</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

font-family: 'Arial', sans-serif;

background-color: #f7f9fc;

color: #333;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

margin-top: 50px;

}

h2, h3 {

color: #0056b3;

text-align: center;

margin-bottom: 20px;

}

.form-label {

font-weight: bold;

}

.btn {

background-color: #0056b3;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 4px;

transition: background-color 0.3s ease;

}

.btn:hover {

background-color: #004085;

}

.mb-3 {

margin-bottom: 15px;

}

input[type="number"] {

font-size: 16px;

padding: 10px;

width: 100%;

border-radius: 4px;

border: 1px solid #ccc;

margin-top: 5px;

}

.btn-info {

background-color: #17a2b8;

}

.btn-info:hover {

background-color: #138496;

}

.form-container {

max-width: 700px;

margin: 0 auto;

}

.result-container {

margin-top: 30px;

padding: 20px;

background-color: #f1f1f1;

border-radius: 8px;

}

.result-container h4 {

color: #007bff;

}

.result-container p {

font-size: 16px;

margin-bottom: 10px;

}

.result-container .total-result {

font-weight: bold;

font-size: 18px;

}

</style>

</head>

<body>

<div class="container form-container">

<h2>Enter VIT Student Marks</h2>

<form action="submit.php" method="POST">

<div class="mb-3">

<label for="student\_name" class="form-label">Student Name</label>

<input type="text" class="form-control" id="student\_name" name="student\_name" required>

</div>

<div class="mb-3">

<label for="subject1\_mse" class="form-label">Subject 1 MSE Marks (30%)</label>

<input type="number" class="form-control" id="subject1\_mse" name="subject1\_mse" required>

</div>

<div class="mb-3">

<label for="subject1\_ese" class="form-label">Subject 1 ESE Marks (70%)</label>

<input type="number" class="form-control" id="subject1\_ese" name="subject1\_ese" required>

</div>

<div class="mb-3">

<label for="subject2\_mse" class="form-label">Subject 2 MSE Marks (30%)</label>

<input type="number" class="form-control" id="subject2\_mse" name="subject2\_mse" required>

</div>

<div class="mb-3">

<label for="subject2\_ese" class="form-label">Subject 2 ESE Marks (70%)</label>

<input type="number" class="form-control" id="subject2\_ese" name="subject2\_ese" required>

</div>

<div class="mb-3">

<label for="subject3\_mse" class="form-label">Subject 3 MSE Marks (30%)</label>

<input type="number" class="form-control" id="subject3\_mse" name="subject3\_mse" required>

</div>

<div class="mb-3">

<label for="subject3\_ese" class="form-label">Subject 3 ESE Marks (70%)</label>

<input type="number" class="form-control" id="subject3\_ese" name="subject3\_ese" required>

</div>

<div class="mb-3">

<label for="subject4\_mse" class="form-label">Subject 4 MSE Marks (30%)</label>

<input type="number" class="form-control" id="subject4\_mse" name="subject4\_mse" required>

</div>

<div class="mb-3">

<label for="subject4\_ese" class="form-label">Subject 4 ESE Marks (70%)</label>

<input type="number" class="form-control" id="subject4\_ese" name="subject4\_ese" required>

</div>

<button type="submit" class="btn">Submit</button>

</form>

</div>

<div class="container result-container">

<h3>View Result</h3>

<form action="calculate\_result.php" method="GET">

<div class="mb-3">

<label for="student\_id" class="form-label">Enter Student ID to View Result</label>

<input type="number" class="form-control" id="student\_id" name="id" required>

</div>

<button type="submit" class="btn btn-info">View Result</button>

</form>

</div>

<!-- Example of displaying result (for dynamic use) -->

<div class="container result-container">

<h4>Result for Student Name</h4>

<p>Subject 1 Result: 70%</p>

<p>Subject 2 Result: 80%</p>

<p>Subject 3 Result: 85%</p>

<p>Subject 4 Result: 90%</p>

<p class="total-result">Total Result: 325%</p>

</div>

</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%).**

vit-result-system/

├── backend/ # Node.js + Express + MySQL API

│ ├── server.js # Main backend logic

│ └── package.json # Backend dependencies

├── vit-result-system/

│ ├── public/

│ │ ├── index.html

│ │ └── favicon.ico

│ ├── src/

│ │ ├── components/

│ │ │ └── ResultForm.js

│ │ ├── App.js

│ │ ├── index.js

│ │ └── styles.css

│ └── package.json

├── README.md

**Initialize the Project:**

**npx create-react-app vit-result-system**

**cd vit-result-system**

**mkdir backend**

**cd backend**

**npm init -y**

**Install Dependencies**

npm install express mysql2 cors body-parser

**Set Up MySQL Database**

CREATE DATABASE vit\_results;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

subject1\_mse INT,

subject1\_ese INT,

subject2\_mse INT,

subject2\_ese INT,

subject3\_mse INT,

subject3\_ese INT,

subject4\_mse INT,

subject4\_ese INT

);

**Backend: Set Up Express Server**

// server.js

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const bodyParser = require('body-parser');

const app = express();

const port = 5000;

// Middleware

app.use(cors());

app.use(bodyParser.json());

// MySQL database connection

const db = mysql.createConnection({

host: 'localhost',

user: 'root', // MySQL username

password: '', // MySQL password

database: 'vit\_results',

});

db.connect(err => {

if (err) {

console.error('Database connection failed:', err.stack);

return;

}

console.log('Connected to the database');

});

// API to insert student data

app.post('/addStudent', (req, res) => {

const { student\_name, subject1\_mse, subject1\_ese, subject2\_mse, subject2\_ese, subject3\_mse, subject3\_ese, subject4\_mse, subject4\_ese } = req.body;

if (

[subject1\_mse, subject1\_ese, subject2\_mse, subject2\_ese, subject3\_mse, subject3\_ese, subject4\_mse, subject4\_ese].some(

(mark) => mark < 0 || mark > 100

)

) {

return res.status(400).send('Marks should be between 0 and 100');

}

const query = `INSERT INTO students

(student\_name, subject1\_mse, subject1\_ese, subject2\_mse, subject2\_ese, subject3\_mse, subject3\_ese, subject4\_mse, subject4\_ese)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`;

db.query(query, [student\_name, subject1\_mse, subject1\_ese, subject2\_mse, subject2\_ese, subject3\_mse, subject3\_ese, subject4\_mse, subject4\_ese], (err) => {

if (err) {

console.error(err);

res.status(500).send('Error inserting data');

} else {

res.status(200).send('Student data added successfully');

}

});

});

// API to get student result

app.get('/getResult/:id', (req, res) => {

const studentId = req.params.id;

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

db.query(query, [studentId], (err, result) => {

if (err) {

console.error(err);

res.status(500).send('Error retrieving data');

} else if (result.length === 0) {

res.status(404).send('Student not found');

} else {

const student = result[0];

const subject1\_result = student.subject1\_mse \* 0.30 + student.subject1\_ese \* 0.70;

const subject2\_result = student.subject2\_mse \* 0.30 + student.subject2\_ese \* 0.70;

const subject3\_result = student.subject3\_mse \* 0.30 + student.subject3\_ese \* 0.70;

const subject4\_result = student.subject4\_mse \* 0.30 + student.subject4\_ese \* 0.70;

const total\_result = subject1\_result + subject2\_result + subject3\_result + subject4\_result;

res.status(200).json({

student\_name: student.student\_name,

subject1\_result,

subject2\_result,

subject3\_result,

subject4\_result,

total\_result,

});

}

});

});

// Start the server

app.listen(port, () => {

console.log(`Server running on port ${port}`);

});

**Set Up the Frontend with React**

npm install axios

**2.3 React Components**

**App.js**

import React, { useState } from 'react';

import axios from 'axios';

import 'bootstrap/dist/css/bootstrap.min.css';

function App() {

const [studentName, setStudentName] = useState('');

const [marks, setMarks] = useState({

subject1\_mse: 0,

subject1\_ese: 0,

subject2\_mse: 0,

subject2\_ese: 0,

subject3\_mse: 0,

subject3\_ese: 0,

subject4\_mse: 0,

subject4\_ese: 0,

});

const [studentId, setStudentId] = useState('');

const [result, setResult] = useState(null);

const handleSubmit = async (e) => {

e.preventDefault();

try {

await axios.post('http://localhost:5000/addStudent', {

student\_name: studentName,

...marks,

});

alert('Student data added successfully!');

} catch (error) {

console.error('There was an error!', error);

}

};

const handleResult = async () => {

if (!studentId) {

alert('Please enter a valid student ID');

return;

}

try {

const response = await axios.get(`http://localhost:5000/getResult/${studentId}`);

setResult(response.data);

} catch (error) {

console.error('Error fetching result', error);

}

};

return (

<div className="container mt-5">

<h2>VIT Student Result System</h2>

{/\* Student Data Form \*/}

<form onSubmit={handleSubmit} className="mb-4">

<div className="mb-3">

<label className="form-label">Student Name</label>

<input

type="text"

className="form-control"

value={studentName}

onChange={(e) => setStudentName(e.target.value)}

required

/>

</div>

{['subject1', 'subject2', 'subject3', 'subject4'].map((subject, index) => (

<div key={index} className="mb-3">

<label className="form-label">{subject} MSE Marks (30%)</label>

<input

type="number"

className="form-control"

value={marks[`${subject}\_mse`]}

onChange={(e) => setMarks({ ...marks, [`${subject}\_mse`]: e.target.value })}

required

/>

<label className="form-label">{subject} ESE Marks (70%)</label>

<input

type="number"

className="form-control"

value={marks[`${subject}\_ese`]}

onChange={(e) => setMarks({ ...marks, [`${subject}\_ese`]: e.target.value })}

required

/>

</div>

))}

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

</form>

{/\* View Result \*/}

<h4>View Result</h4>

<input

type="number"

className="form-control mb-3"

placeholder="Enter Student ID"

value={studentId}

onChange={(e) => setStudentId(e.target.value)}

/>

<button onClick={handleResult} className="btn btn-success">View Result</button>

{result && (

<div className="mt-4">

<h5>{result.student\_name}</h5>

<ul>

<li>Subject 1 Result: {result.subject1\_result}</li>

<li>Subject 2 Result: {result.subject2\_result}</li>

<li>Subject 3 Result: {result.subject3\_result}</li>

<li>Subject 4 Result: {result.subject4\_result}</li>

<li>Total Result: {result.total\_result}</li>

</ul>

</div>

)}

</div>

);

}

export default App;

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)**

online-book-store/

├── backend/ # Spring Boot backend

│ ├── src/

│ │ ├── main/

│ │ │ ├── java/com/onlinebookstore/

│ │ │ │ ├── controller/ # Controllers

│ │ │ │ │ ├── HomeController.java

│ │ │ │ │ ├── UserController.java

│ │ │ │ │ └── BookController.java

│ │ │ │ ├── model/ # Models

│ │ │ │ │ ├── User.java

│ │ │ │ │ └── Book.java

│ │ │ │ ├── repository/ # Repositories

│ │ │ │ │ ├── UserRepository.java

│ │ │ │ │ └── BookRepository.java

│ │ │ │ ├── service/ # Services

│ │ │ │ │ ├── UserService.java

│ │ │ │ │ └── BookService.java

│ │ │ ├── resources/

│ │ │ │ ├── application.properties # Spring Boot configuration

│ │ │ │ ├── static/ # Static files (e.g., CSS, JS, images)

│ │ │ │ │ └── css/

│ │ │ │ │ └── styles.css

│ │ │ │ ├── templates/ # Thymeleaf templates (HTML pages)

│ │ │ │ │ ├── home.html

│ │ │ │ │ ├── register.html

│ │ │ │ │ ├── login.html

│ │ │ │ │ └── catalogue.html

│ ├── pom.xml # Maven dependencies

├── README.md # Project documentation

└── .gitignore # Git ignore file

**Set Up MySQL Database**

1. Open **phpMyAdmin** or any MySQL client.
2. Create a database named **online\_book\_store**:

CREATE DATABASE online\_book\_store;

**4. Code Implementation**

**a) application.properties**

sspring.datasource.url=jdbc:mysql://localhost:3306/online\_book\_store

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**b) Models**

**User.java (For Registration and Login)**

package com.onlinebookstore.model;

import jakarta.persistence.\*;

@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.java (For Catalogue)**

package com.onlinebookstore.model;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private double price;

private String description;

// Getters and Setters

}

**c) Repositories**

**UserRepository.java**

package com.onlinebookstore.repository;

import com.onlinebookstore.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long>

{ User findByEmail(String email);

}

**BookRepository.java**

package com.onlinebookstore.repository;

import com.onlinebookstore.model.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**d) Services**

**UserService.java**

package com.onlinebookstore.service;

import com.onlinebookstore.model.User;

import com.onlinebookstore.repository.UserRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public void register(User user) {

userRepository.save(user);

}

public User login(String email, String password) {

User user = userRepository.findByEmail(email);

return (user != null && user.getPassword().equals(password)) ? user : null;

}

}

**BookService.java**

package com.onlinebookstore.service;

import com.onlinebookstore.model.Book;

import com.onlinebookstore.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

}

**e) Controllers**

**HomeController.java**

package com.onlinebookstore.controller;

import org.springframework.stereotype.Controller;

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

@Controller

public class HomeController {

@GetMapping("/")

public String home() {

return "home";

}

}

**UserController.java**

package com.onlinebookstore.controller;

import com.onlinebookstore.model.User;

import com.onlinebookstore.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

@Controller

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/register")

public String registerPage() {

return "register";

}

@PostMapping("/register")

public String registerUser(@ModelAttribute User user, Model model) {

userService.register(user);

model.addAttribute("message", "Registration successful!");

return "login";

}

@GetMapping("/login")

public String loginPage() {

return "login";

}

@PostMapping("/login")

public String loginUser(@RequestParam String email, @RequestParam String password, Model model) {

User user = userService.login(email, password);

if (user != null) {

return "redirect:/catalogue";

} else {

model.addAttribute("error", "Invalid credentials!");

return "login";

}

}

}

**BookController.java**

package com.onlinebookstore.controller;

import com.onlinebookstore.service.BookService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

@Controller

public class BookController {

@Autowired

private BookService bookService;

@GetMapping("/catalogue")

public String cataloguePage(Model model) {

model.addAttribute("books", bookService.getAllBooks());

return "catalogue";

}

}

**f) Frontend Files (Thymeleaf Templates)**

**home.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Online Book Store</title>

<link rel="stylesheet" href="/css/styles.css">

</head>

<body>

<h1>Welcome to Online Book Store</h1>

<a href="/login">Login</a> | <a href="/register">Register</a>

</body>

</html>

**register.html**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<title>Register</title>

</head>

<body>

<h1>Register</h1>

<form action="/register" 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>

**login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form action="/login" 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>

**catalogue.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Catalogue</title>

</head>

<body>

<h1>Book Catalogue</h1>

<ul>

<li th:each="book : ${books}">

<p th:text="${book.title}"></p>

<p th:text="${book.author}"></p>

<p th:text="${book.price}"></p>

</li>

</ul>

</body>

</html>

**styles.css**

/\* src/main/resources/static/css/styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

margin: 0;

padding: 0;

}

h1 {

color: #333;

text-align: center;

margin-top: 20px;

}

a {

text-decoration: none;

color: #007bff;

font-weight: bold;

}

a:hover {

text-decoration: underline;

color: #0056b3;

}

form {

max-width: 400px;

margin: 20px auto;

padding: 10px;

background: #fff;

border: 1px solid #ddd;

border-radius: 8px;

}

input[type="text"],

input[type="email"],

input[type="password"],

button {

width: 100%;

padding: 10px;

margin: 5px 0;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

background-color: #007bff;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

**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)**

**Create the project folder:**bash  
Copy code  
mkdir online-book-store

cd online-book-store

**Initialize the Node.js project:**bash  
Copy code  
npm init -y

**Install necessary packages:**bash  
Copy code  
npm install express mysql2 dotenv body-parser cors

#### 1.2 Database Setup (MySQL)

**Install MySQL** if you haven't already, and create the database.  
sql  
Copy code  
CREATE DATABASE online\_book\_store;

**Create tables for Users and Books:**sql  
Copy code  
CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

description TEXT

);

### 2. Frontend (React)

#### 2.1 Setup the React Project

**Install React and create a new React app:**bash  
Copy code  
npx create-react-app online-book-store-frontend

cd online-book-store-frontend

1. **Install Axios to make HTTP requests:**bash  
   Copy code  
   npm install axios

online-book-store/

│

├── backend/ # Backend folder (Node.js application)

│ ├── src/

│ │ ├── controllers/ # Controllers for handling requests

│ │ │ ├── userController.js

│ │ │ └── bookController.js

│ │ ├── models/ # Database models (for SQL/MongoDB)

│ │ │ ├── User.js

│ │ │ └── Book.js

│ │ ├── routes/ # Routes to handle API endpoints

│ │ │ ├── userRoutes.js

│ │ │ └── bookRoutes.js

│ │ ├── services/ # Services for handling business logic

│ │ │ ├── userService.js

│ │ │ └── bookService.js

│ │ ├── app.js # Express app initialization

│ │ ├── database.js # Database connection file

│ │ └── package.json # Node.js dependencies

│ └── .env # Environment variables (DB credentials, etc.)

│

├── frontend/ # Frontend folder (React app)

│ ├── public/

│ │ ├── index.html

│ │ └── favicon.ico

│ ├── src/

│ │ ├── components/

│ │ │ ├── Home.js

│ │ │ ├── Login.js

│ │ │ ├── Register.js

│ │ │ ├── Catalogue.js

│ │ │ └── BookCard.js

│ │ ├── App.js

│ │ ├── App.css

│ │ ├── index.js

│ │ ├── services/

│ │ │ └── api.js # API calls for User and Book services

│ │ └── package.json # Node.js dependencies

│

└── docker-compose.yml # To run both the backend and frontend using Docker

backend/package.json (Node.js Dependencies):

{

"name": "backend",

"version": "1.0.0",

"description": "Node.js backend for the online book store",

"main": "app.js",

"scripts": {

"start": "node app.js"

},

"dependencies": {

"express": "^4.17.1",

"mysql2": "^2.3.3",

"dotenv": "^16.0.0",

"bcryptjs": "^2.4.3",

"jsonwebtoken": "^8.5.1"

}

}

npm install

backend/.env (Environment Variables):

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=password

DB\_NAME=bookstore

JWT\_SECRET=your-jwt-secret-key

backend/database.js:

const mysql = require('mysql2');

const dotenv = require('dotenv');

// Load environment variables

dotenv.config();

const db = mysql.createConnection({

host: process.env.DB\_HOST,

user: process.env.DB\_USER,

password: process.env.DB\_PASSWORD,

database: process.env.DB\_NAME

});

db.connect((err) => {

if (err) {

console.error('Database connection failed: ' + err.stack);

return;

}

console.log('Connected to the MySQL database.');

});

module.exports = db;

**backend/models/User.js**

**const db = require('../database');**

**const bcrypt = require('bcryptjs');**

**// Register new user**

**const createUser = async (username, password) => {**

**const hashedPassword = await bcrypt.hash(password, 10);**

**const query = 'INSERT INTO users (username, password) VALUES (?, ?)';**

**db.query(query, [username, hashedPassword], (err, result) => {**

**if (err) throw err;**

**console.log('User created:', result);**

**});**

**};**

**// Find user by username**

**const findUserByUsername = (username, callback) => {**

**const query = 'SELECT \* FROM users WHERE username = ?';**

**db.query(query, [username], callback);**

**};**

**module.exports = { createUser, findUserByUsername };**

**backend/models/Book.js:**

**const db = require('../database');**

**// Add a new book**

**const addBook = (title, author, genre) => {**

**const query = 'INSERT INTO books (title, author, genre) VALUES (?, ?, ?)';**

**db.query(query, [title, author, genre], (err, result) => {**

**if (err) throw err;**

**console.log('Book added:', result);**

**});**

**};**

**// Get all books**

**const getAllBooks = (callback) => {**

**const query = 'SELECT \* FROM books';**

**db.query(query, callback);**

**};**

**module.exports = { addBook, getAllBooks };**

#### backend/controllers/userController.js (User Controller):

const User = require('../models/User');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

// Register new user

exports.registerUser = async (req, res) => {

const { username, password } = req.body;

try {

User.createUser(username, password);

res.status(201).send('User registered successfully');

} catch (err) {

res.status(500).send('Error registering user');

}

};

// Login user

exports.loginUser = (req, res) => {

const { username, password } = req.body;

User.findUserByUsername(username, (err, results) => {

if (err || results.length === 0) {

return res.status(400).send('User not found');

}

const user = results[0];

bcrypt.compare(password, user.password, (err, match) => {

if (err || !match) {

return res.status(400).send('Invalid credentials');

}

const token = jwt.sign({ id: user.id }, process.env.JWT\_SECRET, { expiresIn: '1h' });

res.status(200).json({ token });

});

});

};

backend/controllers/bookController.js (Book Controller):

const Book = require('../models/Book');

// Get all books

exports.getBooks = (req, res) => {

Book.getAllBooks((err, results) => {

if (err) {

return res.status(500).send('Error fetching books');

}

res.status(200).json(results);

});

};

// Add a new book

exports.addBook = (req, res) => {

const { title, author, genre } = req.body;

try {

Book.addBook(title, author, genre);

res.status(201).send('Book added successfully');

} catch (err) {

res.status(500).send('Error adding book');

}

};

backend/routes/userRoutes.js (User Routes):

const express = require('express');

const { registerUser, loginUser } = require('../controllers/userController');

const router = express.Router();

router.post('/register', registerUser);

router.post('/login', loginUser);

module.exports = router;

backend/routes/bookRoutes.js (Book Routes):

const express = require('express');

const { getBooks, addBook } = require('../controllers/bookController');

const router = express.Router();

router.get('/catalogue', getBooks);

router.post('/add-book', addBook);

module.exports = router;

backend/app.js (Main App):

const express = require('express');

const dotenv = require('dotenv');

const bodyParser = require('body-parser');

const userRoutes = require('./routes/userRoutes');

const bookRoutes = require('./routes/bookRoutes');

// Load environment variables

dotenv.config();

const app = express();

// Middleware

app.use(bodyParser.json());

// Routes

app.use('/api/users', userRoutes);

app.use('/api/books', bookRoutes);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

frontend/package.json (React Dependencies:

{

"name": "frontend",

"version": "1.0.0",

"main": "index.js",

"dependencies": {

"axios": "^0.21.1",

"react": "^17.0.2",

"react-dom": "^17.0.2",

"react-scripts": "4.0.3"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

}

}

frontend/src/services/api.js (API Service):

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api';

export const registerUser = async (userData) => {

try {

const response = await axios.post(`${API\_URL}/users/register`, userData);

return response.data;

} catch (error) {

console.error('Error registering user:', error);

throw error;

}

};

export const loginUser = async (credentials) => {

try {

const response = await axios.post(`${API\_URL}/users/login`, credentials);

return response.data;

} catch (error) {

console.error('Error logging in:', error);

throw error;

}

};

export const getBooks = async () => {

try {

const response = await axios.get(`${API\_URL}/books/catalogue`);

return response.data;

} catch (error) {

console.error('Error fetching books:', error);

throw error;

}

};

frontend/src/App.js (Main App Component):

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import HomePage from './HomePage';

import LoginPage from './LoginPage';

import RegisterPage from './RegisterPage';

import CataloguePage from './CataloguePage';

const App = () => {

return (

<Router>

<Switch>

<Route path="/" exact component={HomePage} />

<Route path="/login" component={LoginPage} />

<Route path="/register" component={RegisterPage} />

<Route path="/catalogue" component={CataloguePage} />

</Switch>

</Router>

);

};

export default App;

**HomePage Component (HomePage.js):**jsx  
Copy code  
import React from 'react';

import { Link } from 'react-router-dom';

const HomePage = () => {

return (

<div>

<h1>Welcome to Online Book Store</h1>

<Link to="/login">Login</Link> | <Link to="/register">Register</Link>

</div>

);

};

export default HomePage;

**LoginPage Component (LoginPage.js):**jsx  
Copy code  
import React, { useState } from 'react';

import axios from 'axios';

import { useHistory } from 'react-router-dom';

const LoginPage = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const history = useHistory();

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('http://localhost:5000/login', { email, password });

alert(response.data.message);

history.push('/catalogue');

} catch (error) {

alert('Invalid credentials');

}

};

return (

<div>

<h1>Login</h1>

<form onSubmit={handleSubmit}>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

placeholder="Email"

required

/>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

placeholder="Password"

required

/>

<button type="submit">Login</button>

</form>

</div>

);

};

export default LoginPage;

**RegisterPage Component (RegisterPage.js):**jsx  
Copy code  
import React, { useState } from 'react';

import axios from 'axios';

import { useHistory } from 'react-router-dom';

const RegisterPage = () => {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const history = useHistory();

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await axios.post('http://localhost:5000/register', { name, email, password });

alert(response.data.message);

history.push('/login');

} catch (error) {

alert('Error registering user');

}

};

return (

<div>

<h1>Register</h1>

<form onSubmit={handleSubmit}>

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

placeholder="Name"

required

/>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

placeholder="Email"

required

/>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

placeholder="Password"

required

/>

<button type="submit">Register</button>

</form>

</div>

);

};

export default RegisterPage;

**CataloguePage Component (CataloguePage.js):**jsx  
Copy code  
import React, { useEffect, useState } from 'react';

import axios from 'axios';

const CataloguePage = () => {

const [books, setBooks] = useState([]);

useEffect(() => {

axios.get('http://localhost:5000/books')

.then(response => setBooks(response.data))

.catch(error => console.error('Error fetching books:', error));

}, []);

return (

<div>

<h1>Catalogue</h1>

<ul>

{books.map(book => (

<li key={book.id}>

<p>{book.title}</p>

<p>{book.author}</p>

<p>${book.price}</p>

<p>{book.description}</p>

</li>

))}

</ul>

</div>

);

};

export default CataloguePage;

### Docker Setup (for MySQL)

#### 3.1 docker-compose.yml (Compose File)

version: "3"

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

backend:

build: ./backend

ports:

- "5000:5000"

environment:

- DB\_HOST=db

- DB\_USER=root

- DB\_PASSWORD=password

- DB\_NAME=bookstore

depends\_on:

- db

db:

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: password

MYSQL\_DATABASE: bookstore

ports:

- "3306:3306"

**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.**

user\_system/

│

├── assets/

│ └── styles.css # Stylesheet for the pages

│

├── db.php # Database connection file

│

├── index.php # Main page (optional, can be used as a landing page)

│

├── login.php # Login form and logic

│

├── logout.php # Logout and session management

│

├── register.php # Registration form and logic

│

├── welcome.php # User dashboard (welcome page)

│

└── .htaccess # Optional (for better URL handling and security)

**Step 1: Database Setup (MySQL)**

First, create a MySQL database to store user details.

-- Create database

CREATE DATABASE user\_system;

-- Create users table

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

This table stores user information like username, email, and password (hashed for security).

**Step 2: PHP Database Connection (db.php)**

<?php

$host = 'localhost';

$username = 'root';

$password = '';

$dbname = 'user\_system';

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 (register.php)**

<?php

require\_once 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

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

$email = $\_POST['email'];

$password = password\_hash($\_POST['password'], PASSWORD\_BCRYPT); // Encrypt the password

// Insert user details into users table

$sql = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";

$stmt = $conn->prepare($sql);

$stmt->execute([$username, $email, $password]);

echo "Registration successful!";

}

?>

<form method="POST">

<input type="text" name="username" placeholder="Username" 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>

**Step 4: User Login (login.php)**

<?php

session\_start();

require\_once 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

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

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

// Check if user exists

$sql = "SELECT \* FROM users WHERE username = ? OR email = ?";

$stmt = $conn->prepare($sql);

$stmt->execute([$username, $username]);

$user = $stmt->fetch();

// Verify password

if ($user && password\_verify($password, $user['password'])) {

// Create session variables

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['username'] = $user['username'];

// Set cookies to remember the user (optional)

setcookie("user\_id", $user['id'], time() + (86400 \* 30), "/"); // Cookie for 30 days

setcookie("username", $user['username'], time() + (86400 \* 30), "/");

echo "Login successful!";

header('Location: welcome.php');

exit();

} else {

echo "Invalid username or password!";

}

}

?>

<!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 rel="stylesheet" href="assets/styles.css">

</head>

<body>

<h1>Login</h1>

<form method="POST">

<input type="text" name="username" placeholder="Username or Email" required><br>

<input type="password" name="password" placeholder="Password" required><br>

<button type="submit">Login</button>

</form>

</body>

</html>

**Step 5: User Dashboard or Welcome Page (welcome.php)**

<?php

session\_start();

// If user is not logged in, check cookies and start session

if (!isset($\_SESSION['user\_id']) && isset($\_COOKIE['user\_id'])) {

$\_SESSION['user\_id'] = $\_COOKIE['user\_id'];

$\_SESSION['username'] = $\_COOKIE['username'];

}

if (!isset($\_SESSION['user\_id'])) {

header('Location: login.php');

exit();

}

echo "Welcome, " . $\_SESSION['username'] . "!<br>";

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

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Welcome</title>

<link rel="stylesheet" href="assets/styles.css">

</head>

<body>

<h1>Welcome to Your Dashboard</h1>

<p>You're successfully logged in!</p>

</body>

</html>

**Step 6: Session Management & Logout (logout.php)**

<?php

session\_start();

// Destroy session and clear cookies

session\_unset();

session\_destroy();

setcookie("user\_id", "", time() - 3600, "/");

setcookie("username", "", time() - 3600, "/");

header('Location: login.php');

exit();

?>

**Step 7: CSS Styling (styles.css)**

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

margin: 0;

padding: 0;

}

form {

max-width: 400px;

margin: 50px auto;

background-color: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

width: 100%;

padding: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

h1 {

text-align: center;

color: #007bff;

}

**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**

attendance-system/

│

├── css/

│ └── styles.css

├── db.php

├── index.php

├── register.php

├── teacher\_login.php

├── take\_attendance.php

├── view\_attendance.php

├── logout.php

└── README.md

**Step 1: Database Design (MySQL)**

-- Create database

CREATE DATABASE attendance\_system;

-- Students Table

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

roll\_no VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL

);

-- Teachers Table

CREATE TABLE teachers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

-- Attendance Table

CREATE TABLE attendance (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

date DATE NOT NULL,

status ENUM('Present', 'Absent') NOT NULL,

FOREIGN KEY (student\_id) REFERENCES students(id)

);

**Step 2: PHP Backend Logic**

**2.1 Database Connection (db.php)**

<?php

$host = 'localhost';

$username = 'root';

$password = '';

$dbname = 'attendance\_system';

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();

}

?>

**2.2 Student Registration (register.php)**

<?php

require\_once 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$name = $\_POST['name'];

$roll\_no = $\_POST['roll\_no'];

$email = $\_POST['email'];

// Check if roll number already exists

$sql\_check = "SELECT \* FROM students WHERE roll\_no = ?";

$stmt\_check = $conn->prepare($sql\_check);

$stmt\_check->execute([$roll\_no]);

if ($stmt\_check->rowCount() > 0) {

echo "Roll number already registered!";

exit();

}

// Insert student details into students table

$sql = "INSERT INTO students (name, roll\_no, email) VALUES (?, ?, ?)";

$stmt = $conn->prepare($sql);

$stmt->execute([$name, $roll\_no, $email]);

echo "Registration successful!";

}

?>

<form method="POST">

<input type="text" name="name" placeholder="Student Name" required>

<input type="text" name="roll\_no" placeholder="Roll Number" required>

<input type="email" name="email" placeholder="Email" required>

<button type="submit">Register</button>

</form>

**2.3 Teacher Login (teacher\_login.php)**

<?php

require\_once 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$email = $\_POST['email'];

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

// Check if teacher exists

$sql = "SELECT \* FROM teachers WHERE email = ?";

$stmt = $conn->prepare($sql);

$stmt->execute([$email]);

$teacher = $stmt->fetch();

if ($teacher && password\_verify($password, $teacher['password'])) {

// Start session for the teacher

session\_start();

$\_SESSION['teacher\_id'] = $teacher['id'];

echo "Login successful!";

header('Location: take\_attendance.php');

} else {

echo "Invalid email or password!";

}

}

?>

<form method="POST">

<input type="email" name="email" placeholder="Teacher Email" required>

<input type="password" name="password" placeholder="Password" required>

<button type="submit">Login</button>

</form>

**2.4 Take Attendance (take\_attendance.php)**

<?php

session\_start();

require\_once 'db.php';

// Check if teacher is logged in

if (!isset($\_SESSION['teacher\_id'])) {

header('Location: teacher\_login.php');

exit();

}

// Fetch all students

$sql = "SELECT \* FROM students";

$stmt = $conn->prepare($sql);

$stmt->execute();

$students = $stmt->fetchAll();

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$date = $\_POST['date'];

// Loop through all students and update attendance status

foreach ($students as $student) {

$status = isset($\_POST['attendance\_' . $student['id']]) ? 'Present' : 'Absent';

// Check if attendance is already recorded for the day

$sql\_check = "SELECT \* FROM attendance WHERE student\_id = ? AND date = ?";

$stmt\_check = $conn->prepare($sql\_check);

$stmt\_check->execute([$student['id'], $date]);

if ($stmt\_check->rowCount() == 0) {

// Insert attendance record

$sql\_insert = "INSERT INTO attendance (student\_id, date, status) VALUES (?, ?, ?)";

$stmt\_insert = $conn->prepare($sql\_insert);

$stmt\_insert->execute([$student['id'], $date, $status]);

}

}

echo "Attendance marked successfully!";

}

?>

<form method="POST">

<input type="date" name="date" required>

<div>

<?php foreach ($students as $student): ?>

<div>

<input type="checkbox" name="attendance\_<?php echo $student['id']; ?>" id="attendance\_<?php echo $student['id']; ?>">

<label for="attendance\_<?php echo $student['id']; ?>"><?php echo $student['name']; ?> (<?php echo $student['roll\_no']; ?>)</label>

</div>

<?php endforeach; ?>

</div>

<button type="submit">Mark Attendance</button>

</form>

**2.5 View Student Attendance (view\_attendance.php)**

<?php

require\_once 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$roll\_no = $\_POST['roll\_no'];

// Fetch student's attendance records

$sql = "SELECT a.date, a.status FROM attendance a

JOIN students s ON a.student\_id = s.id

WHERE s.roll\_no = ?";

$stmt = $conn->prepare($sql);

$stmt->execute([$roll\_no]);

$attendance\_records = $stmt->fetchAll();

echo "<h2>Attendance Records for Roll No: $roll\_no</h2>";

foreach ($attendance\_records as $record) {

echo "Date: " . $record['date'] . " - Status: " . $record['status'] . "<br>";

}

}

?>

<form method="POST">

<input type="text" name="roll\_no" placeholder="Enter Roll Number" required>

<button type="submit">View Attendance</button>

</form>

**2.6 Logout (logout.php)**:

<?php

session\_start();

session\_unset();

session\_destroy();

header('Location: teacher\_login.php');

exit();

**styles.css**

body {

font-family: Arial, sans-serif;

background-color: #f8f9fa;

margin: 0;

padding: 0;

}

h1 {

color: #007bff;

text-align: center;

}

form {

max-width: 500px;

margin: 20px auto;

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

input, textarea, button {

width: 100%;

padding: 10px;

margin: 10px 0;

}

button {

background-color: #007bff;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

label {

font-size: 14px;

}

**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.**

/online\_farmers\_market/

│

├── /css/

│ └── styles.css

│

├── /includes/

│ ├── db.php

│ └── header.php

│

├── /farmer/

│ ├── dashboard.php

│ ├── add\_product.php

│ └── view\_products.php

│

├── /customer/

│ ├── shop.php

│ ├── cart.php

│

├── index.php

├── login.php

├── register.php

└── logout.php

**Database Design**

CREATE DATABASE online\_farmers\_market;

USE online\_farmers\_market;

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,

role ENUM('farmer', 'customer') NOT NULL

);

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

farmer\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

category VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

quantity INT NOT NULL,

description TEXT,

FOREIGN KEY (farmer\_id) REFERENCES users(id)

);

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT NOT NULL,

product\_id INT NOT NULL,

quantity INT NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (customer\_id) REFERENCES users(id),

FOREIGN KEY (product\_id) REFERENCES products(id)

);

**3. Code Files**

**a. Database Connection (includes/db.php)**

<?php

$host = "localhost";

$user = "root";

$password = "";

$dbname = "online\_farmers\_market";

$conn = new mysqli($host, $user, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**b. Header File (includes/header.php)**

<?php

session\_start();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Farmers Market</title>

<link rel="stylesheet" href="../css/styles.css">

</head>

<body>

<header>

<h1>Farmers Market</h1>

<nav>

<a href="../index.php">Home</a>

<?php if (!isset($\_SESSION['user\_id'])): ?>

<a href="../login.php">Login</a>

<a href="../register.php">Register</a>

<?php else: ?>

<?php if ($\_SESSION['role'] === 'farmer'): ?>

<a href="../farmer/dashboard.php">Dashboard</a>

<?php else: ?>

<a href="../customer/shop.php">Shop</a>

<a href="../customer/cart.php">Cart</a>

<?php endif; ?>

<a href="../logout.php">Logout</a>

<?php endif; ?>

</nav>

</header>

**c. Homepage (index.php)**

<?php include('./includes/header.php'); ?>

<div class="main-content">

<h2>Welcome to the Farmers Market!</h2>

<p>Where farmers sell their fresh produce directly to customers.</p>

</div>

</body>

</html>

**d. Registration Page (register.php)**

<?php

include('./includes/db.php');

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$name = $\_POST['name'];

$email = $\_POST['email'];

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

$role = $\_POST['role'];

$sql = "INSERT INTO users (name, email, password, role) VALUES ('$name', '$email', '$password', '$role')";

if ($conn->query($sql) === TRUE) {

header("Location: login.php");

exit();

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

?>

<form method="POST">

<h2>Register</h2>

<input type="text" name="name" placeholder="Full Name" required>

<input type="email" name="email" placeholder="Email Address" required>

<input type="password" name="password" placeholder="Password" required>

<select name="role">

<option value="farmer">Farmer</option>

<option value="customer">Customer</option>

</select>

<button type="submit">Register</button>

</form>

**e. Login Page (login.php)**

<?php

include('./includes/db.php');

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$email = $\_POST['email'];

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

$sql = "SELECT \* FROM users WHERE email = '$email'";

$result = $conn->query($sql);

if ($result->num\_rows === 1) {

$user = $result->fetch\_assoc();

if (password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['role'] = $user['role'];

if ($user['role'] === 'farmer') {

header("Location: farmer/dashboard.php");

} else {

header("Location: customer/shop.php");

}

exit();

}

}

echo "Invalid login credentials.";

}

?>

<form method="POST">

<h2>Login</h2>

<input type="email" name="email" placeholder="Email Address" required>

<input type="password" name="password" placeholder="Password" required>

<button type="submit">Login</button>

</form>

**f. Farmer Dashboard (farmer/dashboard.php)**

<?php include('../includes/header.php'); ?>

<h2>Farmer Dashboard</h2>

<p>Manage your products here.</p>

<a href="add\_product.php">Add Product</a>

<a href="view\_products.php">View Products</a>

</body>

</html>

**g. Add Product (farmer/add\_product.php)**

<?php

include('../includes/db.php');

include('../includes/header.php');

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$name = $\_POST['name'];

$category = $\_POST['category'];

$price = $\_POST['price'];

$quantity = $\_POST['quantity'];

$description = $\_POST['description'];

$farmer\_id = $\_SESSION['user\_id'];

$sql = "INSERT INTO products (farmer\_id, name, category, price, quantity, description)

VALUES ('$farmer\_id', '$name', '$category', '$price', '$quantity', '$description')";

if ($conn->query($sql) === TRUE) {

echo "Product added successfully.";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

?>

<form method="POST">

<h2>Add Product</h2>

<input type="text" name="name" placeholder="Product Name" required>

<input type="text" name="category" placeholder="Category" required>

<input type="number" name="price" placeholder="Price" required>

<input type="number" name="quantity" placeholder="Quantity" required>

<textarea name="description" placeholder="Description"></textarea>

<button type="submit">Add Product</button>

</form>

**4. CSS Styles (css/styles.css)**

body {

font-family: Arial, sans-serif;

background-color: #f8f9fa;

margin: 0;

padding: 0;

}

header {

background-color: #28a745;

color: #fff;

padding: 10px 20px;

text-align: center;

}

nav a {

color: white;

margin: 0 10px;

text-decoration: none;

}

nav a:hover {

text-decoration: underline;

}

form {

width: 300px;

margin: 20px auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

}

input, select, button {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ddd;

}

button {

background-color: #28a745;

color: white;

border: none;

}

button:hover {

background-color: #218838;

}

.product {

background-color: #fff;

border: 1px solid #ddd;

margin: 10px;

padding: 15px;

border-radius: 5px;

}

.product h3 {

margin-top: 0;

}

**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.**

/project-root

/css

style.css

/includes

db.php

session\_manager.php

session\_expiry.php

/public

login.php

dashboard.php

logout.php

/sql

user\_sessions.sql

**SQL Script:**

CREATE TABLE user\_sessions (

session\_id VARCHAR(255) PRIMARY KEY,

user\_id INT NOT NULL,

last\_activity TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**db.php**:

<?php

<?php

$host = getenv('DB\_HOST') ?: 'localhost';

$db = getenv('DB\_NAME') ?: 'your\_database';

$user = getenv('DB\_USER') ?: 'root';

$pass = getenv('DB\_PASS') ?: '';

try {

$pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

die("Connection failed: " . $e->getMessage());

}

?>

**session\_manager.php**:

<?php

include 'db.php';

session\_start();

$userId = 1; // Example user ID. Replace with your actual user authentication logic.

$sessionId = session\_id();

$timeout = 5 \* 60; // 5 minutes

// Delete expired sessions

$stmt = $pdo->prepare("DELETE FROM user\_sessions WHERE UNIX\_TIMESTAMP(NOW()) - UNIX\_TIMESTAMP(last\_activity) > ?");

$stmt->execute([$timeout]);

// Check active sessions for the user

$stmt = $pdo->prepare("SELECT COUNT(\*) AS session\_count FROM user\_sessions WHERE user\_id = ?");

$stmt->execute([$userId]);

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($row['session\_count'] >= 3) {

die("Maximum concurrent sessions exceeded. Please log out from another session to continue.");

}

// Update or insert the current session

$stmt = $pdo->prepare("REPLACE INTO user\_sessions (session\_id, user\_id, last\_activity) VALUES (?, ?, NOW())");

$stmt->execute([$sessionId, $userId]);

// Refresh session activity

$\_SESSION['user\_id'] = $userId;

?>

**session\_expiry.php**:

<?php

function checkSessionTimeout() {

$timeout = 5 \* 60; // 5 minutes

if (isset($\_SESSION['last\_activity']) && (time() - $\_SESSION['last\_activity'] > $timeout)) {

session\_unset();

session\_destroy();

header("Location: login.php"); // Redirect to login page

exit;

}

$\_SESSION['last\_activity'] = time();

}

### 3. dashboard.php

<?php

session\_start();

include 'session\_expiry.php';

checkSessionTimeout();

// Redirect to login if not logged in

if (!isset($\_SESSION['user\_id'])) {

header("Location: login.php");

exit;

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dashboard</title>

</head>

<body>

<h1>Welcome, <?php echo htmlspecialchars($\_SESSION['username']); ?>!</h1>

<p>This is the dashboard. Your session is being monitored for inactivity.</p>

<a href="logout.php">Logout</a>

</body>

</html>

**login.php**

<?php

session\_start();

include 'db.php';

// Simulate user authentication (replace with actual authentication logic)

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

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

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

// Example query to fetch user data (replace with actual database query)

$stmt = $pdo->prepare("SELECT id, username, password FROM users WHERE username = ?");

$stmt->execute([$username]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['password'])) { // Verify hashed password

$\_SESSION['user\_id'] = $user['id']; // Store user ID in session

$\_SESSION['username'] = $user['username']; // Store username in session

// Redirect to the dashboard

header("Location: dashboard.php");

exit;

} else {

$error = "Invalid username or password.";

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<?php if (isset($error)) { echo "<p style='color:red;'>$error</p>"; } ?>

<form method="POST" action="">

<label>Username:</label><br>

<input type="text" name="username" required><br><br>

<label>Password:</label><br>

<input type="password" name="password" required><br><br>

<button type="submit">Login</button>

</form>

</body>

</html>

**logout.php**:

<?php

include 'db.php';

session\_start();

$sessionId = session\_id();

// Delete the current session from the database

$stmt = $pdo->prepare("DELETE FROM user\_sessions WHERE session\_id = ?");

$stmt->execute([$sessionId]);

session\_unset();

session\_destroy();

header("Location: login.php"); // Redirect to login page

exit;

?>

**style.css**:

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.error {

color: red;

font-weight: bold;

}

**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. Project Structure**

employee-records/

│

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/

│ │ │ └── example/

│ │ │ └── employeerecords/

│ │ │ ├── Employee.java

│ │ │ ├── EmployeeController.java

│ │ │ ├── EmployeeService.java

│ │ │ └── EmployeeApplication.java

│ │ ├── resources/

│ │ │ ├── application.properties

├── pom.xml

└── target/

**2. Dependencies (pom.xml)**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-json</artifactId>

</dependency>

</dependencies>

**3. Entity Class (Employee.java)**

package com.example.employeerecords;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String position;

private double salary;

// Constructors, getters, and setters

public Employee() {}

public Employee(String name, String position, double salary) {

this.name = name;

this.position = position;

this.salary = salary;

}

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 getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

**4. Repository Class (EmployeeRepository.java)**

package com.example.employeerecords;

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**5. Service Class (EmployeeService.java)**

package com.example.employeerecords;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Employee addEmployee(Employee employee) {

return employeeRepository.save(employee);

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

}

**6. Controller Class (EmployeeController.java)**

package com.example.employeerecords;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// Endpoint to add a new employee

@PostMapping

public Employee addEmployee(@RequestBody Employee employee) {

return employeeService.addEmployee(employee);

}

// Endpoint to get all employees in JSON format

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

**7. Main Application Class (EmployeeApplication.java)**

package com.example.employeerecords;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeApplication.class, args);

}

}

**8. Application Properties (application.properties)**

# H2 Database Configuration

spring.datasource.url=jdbc:h2:mem:testdb

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

spring.jpa.hibernate.ddl-auto=update

**9. Testing with Postman**

1. **Run the application**:

mvn spring-boot:run

1. **Testing POST request to add an employee**:

http://localhost:8080/employees

Set the request body to JSON and enter the following data:

{

"name": "John Doe",

"position": "Software Engineer",

"salary": 75000.00

}

Click on **Send**, and the employee record will be added.

1. **Testing GET request to list all employees**:

In Postman, use a GET request to retrieve all employees. Set the URL to:

http://localhost:8080/employees

Click on **Send**, and the list of all employees will be returned in JSON format.

Example response:

json

Copy code

[

{

"id": 1,

"name": "John Doe",

"position": "Software Engineer",

"salary": 75000.0

}

]