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

// Student.java

package com.example.student;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private 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 String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

// StudentRepository.java

package com.example.student;

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

public interface StudentRepository extends JpaRepository<Student, Long> {

}

// StudentController.java

package com.example.student;

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

import org.springframework.http.ResponseEntity;

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 a new student

@PostMapping

public Student addStudent(@RequestBody Student student) {

return studentRepository.save(student);

}

// Delete a student

@DeleteMapping("/{id}")

public ResponseEntity<?> deleteStudent(@PathVariable Long id) {

if (studentRepository.existsById(id)) {

studentRepository.deleteById(id);

return ResponseEntity.ok().build();

}

return ResponseEntity.notFound().build();

}

}

// application.properties

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

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

// 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>3.2.0</version>

</parent>

<groupId>com.example</groupId>

<artifactId>student-management</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>student-management</name>

<description>Student Management System</description>

<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.mysql</groupId>

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

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

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

<!-- index.php -->

<!DOCTYPE html>

<html>

<head>

<title>Employee Records</title>

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

</head>

<body>

<div class="container">

<h2>Add Employee</h2>

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

<div class="form-group">

<label>Name:</label>

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

</div>

<div class="form-group">

<label>Email:</label>

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

</div>

<div class="form-group">

<label>Position:</label>

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

</div>

<button type="submit">Add Employee</button>

</form>

<h2>Employee List</h2>

<?php

include 'config.php';

$sql = "SELECT \* FROM employees";

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

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

echo "<table>

<tr>

<th>Name</th>

<th>Email</th>

<th>Position</th>

</tr>";

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

echo "<tr>

<td>".$row["name"]."</td>

<td>".$row["email"]."</td>

<td>".$row["position"]."</td>

</tr>";

}

echo "</table>";

} else {

echo "No employees found";

}

?>

</div>

</body>

</html>

<!-- config.php -->

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "employee\_db";

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

if ($conn->connect\_error) {

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

}

?>

<!-- add\_employee.php -->

<?php

include 'config.php';

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

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

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

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

$sql = "INSERT INTO employees (name, email, position)

VALUES ('$name', '$email', '$position')";

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

header("Location: index.php");

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

$conn->close();

?>

<!-- style.css -->

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 15px;

border: none;

cursor: pointer;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

th {

background-color: #4CAF50;

color: white;

}

/\* SQL to create table \*/

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

position VARCHAR(100) NOT NULL

);

3. Design following responsive layout using html.

This section should change colour after

clicking on following buttons

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Responsive Layout</title>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

}

header {

background-color: #333;

color: white;

text-align: center;

padding: 1rem;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

.content-section {

padding: 20px;

margin: 20px 0;

border: 1px solid #ddd;

transition: background-color 0.3s ease;

}

.button-container {

display: flex;

gap: 10px;

margin-bottom: 20px;

flex-wrap: wrap;

}

button {

padding: 10px 20px;

cursor: pointer;

border: none;

border-radius: 5px;

font-size: 16px;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 1rem;

position: fixed;

bottom: 0;

width: 100%;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

.container {

padding: 10px;

}

.content-section {

margin: 10px 0;

}

.button-container {

flex-direction: column;

}

button {

width: 100%;

}

}

</style>

</head>

<body>

<header>

<h1>Responsive Layout Example</h1>

</header>

<div class="container">

<div class="button-container">

<button onclick="changeColor('red')" style="background-color: #ff6b6b">Red Theme</button>

<button onclick="changeColor('blue')" style="background-color: #4834d4">Blue Theme</button>

<button onclick="changeColor('green')" style="background-color: #6ab04c">Green Theme</button>

</div>

<div id="section1" class="content-section">

<h2>Section 1</h2>

<p>This is the content for section 1. Click the buttons above to change the color theme.</p>

</div>

<div id="section2" class="content-section">

<h2>Section 2</h2>

<p>This is the content for section 2. The layout is fully responsive and will adapt to different screen sizes.</p>

</div>

</div>

<footer>

<p>&copy; 2024 Responsive Layout Example</p>

</footer>

<script>

function changeColor(color) {

const sections = document.querySelectorAll('.content-section');

const colors = {

'red': '#ffe0e0',

'blue': '#e3e0ff',

'green': '#e0ffe3'

};

sections.forEach(section => {

section.style.backgroundColor = colors[color];

});

}

</script>

</body>

</html>

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

// Entity Classes

// MenuItem.java

package com.restaurant.model;

import jakarta.persistence.\*;

@Entity

public class MenuItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

private double price;

private String category;

// Getters and Setters

}

// Order.java

package com.restaurant.model;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

import java.util.List;

@Entity

@Table(name = "food\_orders")

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToMany

private List<MenuItem> items;

private LocalDateTime orderTime;

private String status;

private double totalAmount;

private String customerName;

private String tableNumber;

// Getters and Setters

}

// Repository Interfaces

// MenuItemRepository.java

package com.restaurant.repository;

import com.restaurant.model.MenuItem;

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

public interface MenuItemRepository extends JpaRepository<MenuItem, Long> {

List<MenuItem> findByCategory(String category);

}

// OrderRepository.java

package com.restaurant.repository;

import com.restaurant.model.Order;

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

public interface OrderRepository extends JpaRepository<Order, Long> {

List<Order> findByStatus(String status);

}

// Controllers

// MenuController.java

package com.restaurant.controller;

import com.restaurant.model.MenuItem;

import com.restaurant.service.MenuService;

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

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

@RestController

@RequestMapping("/api/menu")

public class MenuController {

@Autowired

private MenuService menuService;

@GetMapping

public List<MenuItem> getAllItems() {

return menuService.getAllItems();

}

@GetMapping("/category/{category}")

public List<MenuItem> getItemsByCategory(@PathVariable String category) {

return menuService.getItemsByCategory(category);

}

@PostMapping

public MenuItem addMenuItem(@RequestBody MenuItem item) {

return menuService.addMenuItem(item);

}

}

// OrderController.java

package com.restaurant.controller;

import com.restaurant.model.Order;

import com.restaurant.service.OrderService;

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

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

@RestController

@RequestMapping("/api/orders")

public class OrderController {

@Autowired

private OrderService orderService;

@PostMapping

public Order createOrder(@RequestBody Order order) {

return orderService.createOrder(order);

}

@GetMapping

public List<Order> getAllOrders() {

return orderService.getAllOrders();

}

@PutMapping("/{id}/status")

public Order updateOrderStatus(@PathVariable Long id, @RequestParam String status) {

return orderService.updateOrderStatus(id, status);

}

}

// Service Classes

// MenuService.java

package com.restaurant.service;

import com.restaurant.model.MenuItem;

import com.restaurant.repository.MenuItemRepository;

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

import org.springframework.stereotype.Service;

@Service

public class MenuService {

@Autowired

private MenuItemRepository menuItemRepository;

public List<MenuItem> getAllItems() {

return menuItemRepository.findAll();

}

public List<MenuItem> getItemsByCategory(String category) {

return menuItemRepository.findByCategory(category);

}

public MenuItem addMenuItem(MenuItem item) {

return menuItemRepository.save(item);

}

}

// OrderService.java

package com.restaurant.service;

import com.restaurant.model.Order;

import com.restaurant.repository.OrderRepository;

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

import org.springframework.stereotype.Service;

@Service

public class OrderService {

@Autowired

private OrderRepository orderRepository;

public Order createOrder(Order order) {

order.setOrderTime(LocalDateTime.now());

order.setStatus("PENDING");

return orderRepository.save(order);

}

public List<Order> getAllOrders() {

return orderRepository.findAll();

}

public Order updateOrderStatus(Long id, String status) {

Order order = orderRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Order not found"));

order.setStatus(status);

return orderRepository.save(order);

}

}

// application.properties

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

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

// Frontend HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Restaurant Order Management</title>

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

</head>

<body>

<div class="container mt-4">

<h1>Restaurant Order Management</h1>

<!-- Menu Section -->

<div class="row mt-4">

<div class="col-md-6">

<h2>Menu</h2>

<div id="menuItems" class="list-group">

<!-- Menu items will be populated here -->

</div>

</div>

<!-- Order Section -->

<div class="col-md-6">

<h2>Current Order</h2>

<div class="mb-3">

<input type="text" id="customerName" class="form-control" placeholder="Customer Name">

<input type="text" id="tableNumber" class="form-control mt-2" placeholder="Table Number">

</div>

<div id="currentOrder" class="list-group">

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

</div>

<div class="mt-3">

<h4>Total: $<span id="totalAmount">0.00</span></h4>

<button class="btn btn-primary" onclick="placeOrder()">Place Order</button>

</div>

</div>

</div>

<!-- Orders List -->

<div class="row mt-4">

<div class="col-12">

<h2>Active Orders</h2>

<div id="activeOrders">

<!-- Active orders will be displayed here -->

</div>

</div>

</div>

</div>

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

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

</body>

</html>

// app.js

let currentOrder = [];

function loadMenu() {

fetch('/api/menu')

.then(response => response.json())

.then(items => {

const menuContainer = document.getElementById('menuItems');

menuContainer.innerHTML = '';

items.forEach(item => {

const itemElement = document.createElement('a');

itemElement.className = 'list-group-item';

itemElement.innerHTML = `

${item.name} - $${item.price}

<button class="btn btn-sm btn-primary float-end"

onclick="addToOrder(${JSON.stringify(item)})">Add</button>

`;

menuContainer.appendChild(itemElement);

});

});

}

function addToOrder(item) {

currentOrder.push(item);

updateOrderDisplay();

}

function updateOrderDisplay() {

const orderContainer = document.getElementById('currentOrder');

orderContainer.innerHTML = '';

let total = 0;

currentOrder.forEach((item, index) => {

const itemElement = document.createElement('div');

itemElement.className = 'list-group-item';

itemElement.innerHTML = `

${item.name} - $${item.price}

<button class="btn btn-sm btn-danger float-end"

onclick="removeFromOrder(${index})">Remove</button>

`;

orderContainer.appendChild(itemElement);

total += item.price;

});

document.getElementById('totalAmount').textContent = total.toFixed(2);

}

function removeFromOrder(index) {

currentOrder.splice(index, 1);

updateOrderDisplay();

}

function placeOrder() {

const customerName = document.getElementById('customerName').value;

const tableNumber = document.getElementById('tableNumber').value;

if (!customerName || !tableNumber) {

alert('Please enter customer name and table number');

return;

}

const order = {

items: currentOrder,

customerName: customerName,

tableNumber: tableNumber,

totalAmount: currentOrder.reduce((sum, item) => sum + item.price, 0)

};

fetch('/api/orders', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(order)

})

.then(response => response.json())

.then(() => {

currentOrder = [];

updateOrderDisplay();

loadActiveOrders();

document.getElementById('customerName').value = '';

document.getElementById('tableNumber').value = '';

});

}

function loadActiveOrders() {

fetch('/api/orders')

.then(response => response.json())

.then(orders => {

const ordersContainer = document.getElementById('activeOrders');

ordersContainer.innerHTML = '';

orders.forEach(order => {

const orderElement = document.createElement('div');

orderElement.className = 'card mb-3';

orderElement.innerHTML = `

<div class="card-body">

<h5 class="card-title">Order #${order.id}</h5>

<p>Customer: ${order.customerName}</p>

<p>Table: ${order.tableNumber}</p>

<p>Status: ${order.status}</p>

<p>Total: $${order.totalAmount}</p>

<select class="form-select mb-2" onchange="updateStatus(${order.id}, this.value)">

<option value="PENDING" ${order.status === 'PENDING' ? 'selected' : ''}>Pending</option>

<option value="PREPARING" ${order.status === 'PREPARING' ? 'selected' : ''}>Preparing</option>

<option value="READY" ${order.status === 'READY' ? 'selected' : ''}>Ready</option>

<option value="DELIVERED" ${order.status === 'DELIVERED' ? 'selected' : ''}>Delivered</option>

</select>

</div>

`;

ordersContainer.appendChild(orderElement);

});

});

}

function updateStatus(orderId, status) {

fetch(`/api/orders/${orderId}/status?status=${status}`, {

method: 'PUT'

})

.then(() => loadActiveOrders());

}

// Initialize the application

document.addEventListener('DOMContentLoaded', () => {

loadMenu();

loadActiveOrders();

});

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.

import { useState } from 'react';

import { Card } from '@/components/ui/card';

const CurrencyConverter = () => {

const [amount, setAmount] = useState('');

const [result, setResult] = useState(null);

// Hard-coded exchange rate (1 USD = 83.15 INR as of a sample date)

const exchangeRate = 83.15;

const handleConvert = (e) => {

e.preventDefault();

const convertedAmount = parseFloat(amount) \* exchangeRate;

setResult(convertedAmount);

};

return (

<div className="flex items-center justify-center min-h-screen bg-gray-100">

<Card className="w-full max-w-md p-6">

<h1 className="text-2xl font-bold text-center mb-6">USD to INR Converter</h1>

<form onSubmit={handleConvert} className="space-y-4">

<div>

<label className="block text-sm font-medium mb-2">

Enter amount in USD:

</label>

<input

type="number"

min="0"

step="0.01"

value={amount}

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

className="w-full p-2 border rounded focus:ring-2 focus:ring-blue-500"

placeholder="Enter amount"

required

/>

</div>

<button

type="submit"

className="w-full bg-blue-500 text-white py-2 px-4 rounded hover:bg-blue-600 transition-colors"

>

Convert

</button>

</form>

{result !== null && (

<div className="mt-6 p-4 bg-green-50 rounded">

<h2 className="text-lg font-semibold mb-2">Conversion Result:</h2>

<p className="text-xl">

{amount} USD = <span className="font-bold">{result.toFixed(2)} INR</span>

</p>

<p className="text-sm text-gray-600 mt-2">

Exchange Rate: 1 USD = {exchangeRate} INR

</p>

</div>

)}

</Card>

</div>

);

};

export default CurrencyConverter;

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.

<!-- index.php -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>USD to INR Converter</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f0f2f5;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

}

.container {

background-color: white;

padding: 2rem;

border-radius: 8px;

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

width: 100%;

max-width: 400px;

}

h1 {

text-align: center;

color: #1a1a1a;

margin-bottom: 1.5rem;

}

.form-group {

margin-bottom: 1rem;

}

label {

display: block;

margin-bottom: 0.5rem;

color: #4a4a4a;

}

input {

width: 100%;

padding: 0.5rem;

border: 1px solid #ddd;

border-radius: 4px;

box-sizing: border-box;

}

button {

width: 100%;

padding: 0.75rem;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

}

button:hover {

background-color: #0056b3;

}

.result {

margin-top: 1.5rem;

padding: 1rem;

background-color: #e8f5e9;

border-radius: 4px;

}

.result h2 {

margin-top: 0;

color: #2e7d32;

font-size: 1.1rem;

}

.exchange-rate {

margin-top: 0.5rem;

color: #666;

font-size: 0.9rem;

}

</style>

</head>

<body>

<div class="container">

<h1>USD to INR Converter</h1>

<form method="POST">

<div class="form-group">

<label for="amount">Enter amount in USD:</label>

<input type="number" id="amount" name="amount" min="0" step="0.01"

value="<?php echo isset($\_POST['amount']) ? htmlspecialchars($\_POST['amount']) : ''; ?>"

required>

</div>

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

</form>

<?php

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

// Hard-coded exchange rate

$exchangeRate = 83.15;

// Get and validate input

$amount = filter\_input(INPUT\_POST, 'amount', FILTER\_VALIDATE\_FLOAT);

if ($amount !== false && $amount >= 0) {

// Calculate conversion

$result = $amount \* $exchangeRate;

echo '<div class="result">

<h2>Conversion Result:</h2>

<p>' . number\_format($amount, 2) . ' USD = <strong>' . number\_format($result, 2) . ' INR</strong></p>

<p class="exchange-rate">Exchange Rate: 1 USD = ' . $exchangeRate . ' INR</p>

</div>';

} else {

echo '<div class="result" style="background-color: #ffebee;">

<h2 style="color: #c62828;">Error:</h2>

<p>Please enter a valid amount.</p>

</div>';

}

}

?>

</div>

</body>

</html>

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.

<!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>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

min-height: 100vh;

display: flex;

flex-direction: column;

}

header {

background-color: #2c3e50;

color: white;

text-align: center;

padding: 1rem;

}

main {

flex: 1;

display: flex;

justify-content: center;

align-items: center;

padding: 20px;

}

.chessboard-container {

width: 100%;

max-width: 600px;

margin: auto;

}

.chessboard {

width: 100%;

border-collapse: collapse;

border: 4px solid #2c3e50;

}

/\* Create responsive square cells \*/

.chessboard tr {

width: 100%;

}

.chessboard td {

position: relative;

width: 12.5%; /\* 100% ÷ 8 \*/

padding-bottom: 12.5%; /\* Makes cells square \*/

}

/\* Chess cell colors \*/

.chessboard tr:nth-child(odd) td:nth-child(even),

.chessboard tr:nth-child(even) td:nth-child(odd) {

background-color: #b58863;

}

.chessboard tr:nth-child(odd) td:nth-child(odd),

.chessboard tr:nth-child(even) td:nth-child(even) {

background-color: #f0d9b5;

}

/\* Chess coordinates \*/

.coordinate {

position: absolute;

font-size: 0.8em;

color: #666;

padding: 2px;

}

.file {

bottom: 0;

right: 2px;

}

.rank {

top: 2px;

left: 2px;

}

/\* Only show coordinates on first row and first column \*/

tr:last-child td .file {

display: block;

}

tr td:first-child .rank {

display: block;

}

footer {

background-color: #2c3e50;

color: white;

text-align: center;

padding: 1rem;

margin-top: auto;

}

/\* Responsive design \*/

@media screen and (max-width: 480px) {

.coordinate {

font-size: 0.6em;

}

.chessboard {

border-width: 2px;

}

header h1 {

font-size: 1.5rem;

}

}

@media screen and (min-width: 481px) and (max-width: 768px) {

.coordinate {

font-size: 0.7em;

}

.chessboard {

border-width: 3px;

}

}

</style>

</head>

<body>

<header>

<h1>Responsive Chessboard</h1>

</header>

<main>

<div class="chessboard-container">

<table class="chessboard">

<tbody>

<?php

$ranks = range(8, 1);

$files = range('a', 'h');

foreach ($ranks as $rank) {

echo "<tr>";

foreach ($files as $file) {

echo "<td>";

echo "<span class='coordinate rank'>$rank</span>";

echo "<span class='coordinate file'>$file</span>";

echo "</td>";

}

echo "</tr>";

}

?>

</tbody>

</table>

</div>

</main>

<footer>

<p>&copy; 2024 Responsive Chessboard</p>

</footer>

<!-- Alternative JavaScript version if PHP is not available -->

<script>

document.addEventListener('DOMContentLoaded', function() {

const tbody = document.querySelector('.chessboard tbody');

if (!tbody.hasChildNodes()) { // Only create board if PHP didn't

const ranks = Array.from({length: 8}, (\_, i) => 8 - i);

const files = Array.from({length: 8}, (\_, i) => String.fromCharCode(97 + i));

tbody.innerHTML = ranks.map(rank => `

<tr>

${files.map(file => `

<td>

<span class="coordinate rank">${rank}</span>

<span class="coordinate file">${file}</span>

</td>

`).join('')}

</tr>

`).join('');

}

});

</script>

</body>

</html>

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

// Frontend (React) - App.js

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import StudentLogin from './components/StudentLogin';

import ComplaintPage from './components/ComplaintPage';

import AdminLogin from './components/AdminLogin';

import AdminDashboard from './components/AdminDashboard';

function App() {

return (

<Router>

<div className="App">

<Switch>

<Route exact path="/" component={StudentLogin} />

<Route path="/complaint" component={ComplaintPage} />

<Route path="/admin" component={AdminLogin} />

<Route path="/admin-dashboard" component={AdminDashboard} />

</Switch>

</div>

</Router>

);

}

// components/StudentLogin.js

function StudentLogin() {

const [formData, setFormData] = useState({

email: '',

password: ''

});

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await fetch('http://localhost:5000/api/student/login', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(formData)

});

// Handle response

} catch (error) {

console.error(error);

}

};

return (

<div className="login-container">

<h2>Student Login</h2>

<form onSubmit={handleSubmit}>

<input

type="email"

placeholder="Email"

value={formData.email}

onChange={(e) => setFormData({...formData, email: e.target.value})}

/>

<input

type="password"

placeholder="Password"

value={formData.password}

onChange={(e) => setFormData({...formData, password: e.target.value})}

/>

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

</form>

</div>

);

}

// components/ComplaintPage.js

function ComplaintPage() {

const [complaint, setComplaint] = useState({

subject: '',

description: '',

category: ''

});

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await fetch('http://localhost:5000/api/complaints', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${localStorage.getItem('token')}`

},

body: JSON.stringify(complaint)

});

// Handle response

} catch (error) {

console.error(error);

}

};

return (

<div className="complaint-container">

<h2>Submit Complaint</h2>

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Subject"

value={complaint.subject}

onChange={(e) => setComplaint({...complaint, subject: e.target.value})}

/>

<textarea

placeholder="Description"

value={complaint.description}

onChange={(e) => setComplaint({...complaint, description: e.target.value})}

/>

<select

value={complaint.category}

onChange={(e) => setComplaint({...complaint, category: e.target.value})}

>

<option value="">Select Category</option>

<option value="academic">Academic</option>

<option value="infrastructure">Infrastructure</option>

<option value="other">Other</option>

</select>

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

</form>

</div>

);

}

// Backend (Node.js with Express)

const express = require('express');

const mongoose = require('mongoose');

const jwt = require('jsonwebtoken');

const app = express();

// MongoDB Connection

mongoose.connect('mongodb://localhost/college\_complaints', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Student Model

const studentSchema = new mongoose.Schema({

email: String,

password: String,

name: String

});

const Student = mongoose.model('Student', studentSchema);

// Complaint Model

const complaintSchema = new mongoose.Schema({

subject: String,

description: String,

category: String,

studentId: mongoose.Schema.Types.ObjectId,

status: { type: String, default: 'pending' },

createdAt: { type: Date, default: Date.now }

});

const Complaint = mongoose.model('Complaint', complaintSchema);

// Routes

app.post('/api/student/login', async (req, res) => {

const { email, password } = req.body;

const student = await Student.findOne({ email, password });

if (student) {

const token = jwt.sign({ id: student.\_id }, 'secret\_key');

res.json({ token });

} else {

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

}

});

app.post('/api/complaints', authenticateToken, async (req, res) => {

const complaint = new Complaint({

...req.body,

studentId: req.user.id

});

await complaint.save();

res.json(complaint);

});

app.get('/api/complaints', authenticateAdmin, async (req, res) => {

const complaints = await Complaint.find().populate('studentId');

res.json(complaints);

});

// Middleware

function authenticateToken(req, res, next) {

const token = req.headers.authorization?.split(' ')[1];

if (!token) return res.sendStatus(401);

jwt.verify(token, 'secret\_key', (err, user) => {

if (err) return res.sendStatus(403);

req.user = user;

next();

});

}

app.listen(5000, () => {

console.log('Server running on port 5000');

});

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

<!DOCTYPE html>

<html>

<head>

<title>Calculator</title>

<style>

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background-color: #f0f0f0;

}

.calculator {

background-color: #333;

border-radius: 10px;

padding: 20px;

box-shadow: 0 0 10px rgba(0,0,0,0.2);

}

header {

color: white;

text-align: center;

padding: 10px;

}

section {

margin: 10px 0;

}

#display {

width: 100%;

height: 60px;

background-color: #fff;

border: none;

text-align: right;

padding: 10px;

font-size: 24px;

margin-bottom: 10px;

}

table {

width: 100%;

border-spacing: 5px;

}

button {

width: 100%;

height: 50px;

font-size: 20px;

border: none;

border-radius: 5px;

cursor: pointer;

background-color: #666;

color: white;

}

button:hover {

background-color: #999;

}

.operator {

background-color: #ff9500;

}

.operator:hover {

background-color: #ffaa33;

}

footer {

color: white;

text-align: center;

padding: 10px;

font-size: 12px;

}

</style>

</head>

<body>

<div class="calculator">

<header>

<h2>Calculator</h2>

</header>

<section>

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

<table>

<tr>

<td><button onclick="clearDisplay()">C</button></td>

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

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

<td><button class="operator" 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 class="operator" 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 class="operator" onclick="appendToDisplay('-')">-</button></td>

</tr>

<tr>

<td><button onclick="appendToDisplay('1')">1</button></td>

<td><button onclick="appendToDisplay('2')">2</button></td>

<td><button onclick="appendToDisplay('3')">3</button></td>

<td><button class="operator" onclick="appendToDisplay('+')">+</button></td>

</tr>

<tr>

<td><button onclick="appendToDisplay('0')">0</button></td>

<td><button onclick="appendToDisplay('.')">.</button></td>

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

<td><button class="operator" onclick="calculate()">=</button></td>

</tr>

</table>

</section>

<footer>

<p>Simple Calculator</p>

</footer>

</div>

<script>

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

function appendToDisplay(value) {

display.value += value;

}

function clearDisplay() {

display.value = '';

}

function calculate() {

try {

let expression = display.value;

// Handle percentage calculations

if (expression.includes('%')) {

expression = expression.replace(/(\d+)%/g, '($1/100)');

}

display.value = eval(expression);

} catch (error) {

display.value = 'Error';

}

}

</script>

</body>

</html>

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

// a) Transform string to uppercase

function convertToUppercase($string) {

return strtoupper($string);

}

// b) Transform string to lowercase

function convertToLowercase($string) {

return strtolower($string);

}

// c) Make first character uppercase

function capitalizeFirst($string) {

return ucfirst($string);

}

// d) Make first character of all words uppercase

function capitalizeWords($string) {

return ucwords($string);

}

// Example usage

$testString = "Hello World! This is a Test String.";

echo "Original string: " . $testString . "\n";

echo "Uppercase: " . convertToUppercase($testString) . "\n";

echo "Lowercase: " . convertToLowercase($testString) . "\n";

echo "First character uppercase: " . capitalizeFirst($testString) . "\n";

echo "All words capitalized: " . capitalizeWords($testString) . "\n";

?>

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

<?php

// config.php

$host = 'localhost';

$dbname = 'college\_complaints';

$username = 'root';

$password = '';

try {

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

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

} catch(PDOException $e) {

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

}

// student\_login.php

session\_start();

?>

<!DOCTYPE html>

<html>

<head>

<title>Student Login</title>

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

</head>

<body>

<div class="login-container">

<h2>Student Login</h2>

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

<input type="text" name="student\_id" placeholder="Student ID" required>

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

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

</form>

</div>

</body>

</html>

<?php

// process\_login.php

session\_start();

require\_once 'config.php';

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

$student\_id = $\_POST['student\_id'];

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

$stmt = $pdo->prepare("SELECT \* FROM students WHERE student\_id = ? AND password = ?");

$stmt->execute([$student\_id, md5($password)]);

if ($stmt->rowCount() > 0) {

$student = $stmt->fetch();

$\_SESSION['student\_id'] = $student['student\_id'];

header('Location: complaint.php');

} else {

header('Location: student\_login.php?error=1');

}

}

// complaint.php

session\_start();

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

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

exit();

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Submit Complaint</title>

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

</head>

<body>

<div class="complaint-container">

<h2>Submit New Complaint</h2>

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

<input type="text" name="subject" placeholder="Subject" required>

<textarea name="description" placeholder="Description" required></textarea>

<select name="category" required>

<option value="">Select Category</option>

<option value="academic">Academic</option>

<option value="infrastructure">Infrastructure</option>

<option value="other">Other</option>

</select>

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

</form>

</div>

</body>

</html>

<?php

// process\_complaint.php

session\_start();

require\_once 'config.php';

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

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

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

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

$student\_id = $\_SESSION['student\_id'];

$stmt = $pdo->prepare("INSERT INTO complaints (student\_id, subject, description, category, status, created\_at)

VALUES (?, ?, ?, ?, 'pending', NOW())");

$stmt->execute([$student\_id, $subject, $description, $category]);

header('Location: complaint.php?success=1');

}

// admin\_login.php

?>

<!DOCTYPE html>

<html>

<head>

<title>Admin Login</title>

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

</head>

<body>

<div class="login-container">

<h2>Admin Login</h2>

<form action="process\_admin\_login.php" method="POST">

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

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

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

</form>

</div>

</body>

</html>

<?php

// admin\_dashboard.php

session\_start();

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

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

exit();

}

require\_once 'config.php';

$stmt = $pdo->query("SELECT c.\*, s.name as student\_name

FROM complaints c

JOIN students s ON c.student\_id = s.student\_id

ORDER BY c.created\_at DESC");

$complaints = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html>

<head>

<title>Admin Dashboard</title>

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

</head>

<body>

<div class="dashboard-container">

<h2>All Complaints</h2>

<table>

<thead>

<tr>

<th>ID</th>

<th>Student</th>

<th>Subject</th>

<th>Category</th>

<th>Status</th>

<th>Date</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<?php foreach ($complaints as $complaint): ?>

<tr>

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

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

<td><?php echo $complaint['subject']; ?></td>

<td><?php echo $complaint['category']; ?></td>

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

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

<td>

<a href="view\_complaint.php?id=<?php echo $complaint['id']; ?>">View</a>

</td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

</div>

</body>

</html>

/\* style.css \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f0f0f0;

}

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

max-width: 800px;

margin: 0 auto;

padding: 20px;

background-color: white;

border-radius: 5px;

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

}

input, textarea, select {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

padding: 10px;

border: 1px solid #ddd;

text-align: left;

}

th {

background-color: #4CAF50;

color: white;

}

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

<?php

// config.php

$host = 'localhost';

$dbname = 'student\_records';

$username = 'root';

$password = '';

try {

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

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

} catch(PDOException $e) {

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

}

// index.php

require\_once 'config.php';

?>

<!DOCTYPE html>

<html>

<head>

<title>Student Records Management</title>

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

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

</head>

<body>

<div class="container">

<h2>Student Records Management</h2>

<!-- Add Student Form -->

<div class="form-container">

<h3>Add New Student</h3>

<form id="studentForm" onsubmit="return validateForm()">

<input type="text" id="name" name="name" placeholder="Full Name" required>

<input type="email" id="email" name="email" placeholder="Email" required>

<input type="tel" id="phone" name="phone" placeholder="Phone Number">

<input type="text" id="roll\_no" name="roll\_no" placeholder="Roll Number" required>

<button type="submit">Add Student</button>

</form>

</div>

<!-- Student List -->

<div class="list-container">

<h3>Student List</h3>

<table id="studentTable">

<thead>

<tr>

<th>Roll No</th>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<?php

$stmt = $pdo->query("SELECT \* FROM students ORDER BY roll\_no");

while($row = $stmt->fetch()) {

echo "<tr>";

echo "<td>".$row['roll\_no']."</td>";

echo "<td>".$row['name']."</td>";

echo "<td>".$row['email']."</td>";

echo "<td>".$row['phone']."</td>";

echo "<td>

<button onclick='editStudent(".$row['id'].")'>Edit</button>

<button onclick='deleteStudent(".$row['id'].")'>Delete</button>

</td>";

echo "</tr>";

}

?>

</tbody>

</table>

</div>

</div>

<script>

function validateForm() {

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

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

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

var roll\_no = document.getElementById('roll\_no').value;

// Name validation

if(!/^[a-zA-Z\s]{2,50}$/.test(name)) {

alert('Please enter a valid name');

return false;

}

// Email validation

if(!/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/.test(email)) {

alert('Please enter a valid email');

return false;

}

// Phone validation

if(phone && !/^\d{10}$/.test(phone)) {

alert('Please enter a valid 10-digit phone number');

return false;

}

// Roll number validation

if(!/^[A-Z0-9]{5,10}$/.test(roll\_no)) {

alert('Please enter a valid roll number');

return false;

}

return true;

}

function editStudent(id) {

$.get('get\_student.php', {id: id}, function(data) {

var student = JSON.parse(data);

$('#name').val(student.name);

$('#email').val(student.email);

$('#phone').val(student.phone);

$('#roll\_no').val(student.roll\_no);

// Change form action for update

$('#studentForm').attr('action', 'update\_student.php?id=' + id);

});

}

function deleteStudent(id) {

if(confirm('Are you sure you want to delete this student?')) {

$.post('delete\_student.php', {id: id}, function(response) {

if(response == 'success') {

location.reload();

} else {

alert('Error deleting student');

}

});

}

}

</script>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f0f0f0;

}

.container {

max-width: 1000px;

margin: 0 auto;

padding: 20px;

background-color: white;

border-radius: 5px;

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

}

.form-container {

margin-bottom: 30px;

}

input {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

table {

width: 100%;

border-collapse: collapse;

}

th, td {

padding: 10px;

border: 1px solid #ddd;

text-align: left;

}

th {

background-color: #4CAF50;

color: white;

}

</style>

</body>

</html>

<?php

// add\_student.php

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

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

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

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

$roll\_no = $\_POST['roll\_no'];

$stmt = $pdo->prepare("INSERT INTO students (name, email, phone, roll\_no) VALUES (?, ?, ?, ?)");

if($stmt->execute([$name, $email, $phone, $roll\_no])) {

echo 'success';

} else {

echo 'error';

}

}

// update\_student.php

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

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

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

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

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

$roll\_no = $\_POST['roll\_no'];

$stmt = $pdo->prepare("UPDATE students SET name = ?, email = ?, phone = ?, roll\_no = ? WHERE id = ?");

if($stmt->execute([$name, $email, $phone, $roll\_no, $id])) {

echo 'success';

} else {

echo 'error';

}

}

// delete\_student.php

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

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

$stmt = $pdo->prepare("DELETE FROM students WHERE id = ?");

if($stmt->execute([$id])) {

echo 'success';

} else {

echo 'error';

}

}

?>

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.

<!DOCTYPE html>

<html>

<head>

<title>jQuery List Control Demo</title>

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

<style>

body {

font-family: Arial, sans-serif;

max-width: 800px;

margin: 20px auto;

padding: 20px;

}

.container {

display: flex;

justify-content: space-between;

margin-bottom: 30px;

}

.list-box {

width: 45%;

padding: 15px;

border: 1px solid #ccc;

border-radius: 5px;

}

.list-box h3 {

margin-top: 0;

color: #333;

}

select {

width: 100%;

height: 200px;

margin-bottom: 10px;

padding: 5px;

}

button {

padding: 8px 15px;

margin: 5px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

.create-element-section {

margin-top: 30px;

padding: 20px;

background-color: #f5f5f5;

border-radius: 5px;

}

input[type="text"] {

padding: 8px;

margin-right: 10px;

width: 200px;

}

#elementContainer {

margin-top: 20px;

padding: 10px;

border: 1px dashed #ccc;

min-height: 50px;

}

.new-element {

margin: 5px;

padding: 10px;

background-color: #e9e9e9;

border-radius: 3px;

display: inline-block;

}

</style>

</head>

<body>

<h2>jQuery List Control Demo</h2>

<!-- List Copy Section -->

<div class="container">

<div class="list-box">

<h3>Source List</h3>

<select id="sourceList" multiple>

<option value="1">Item 1</option>

<option value="2">Item 2</option>

<option value="3">Item 3</option>

<option value="4">Item 4</option>

<option value="5">Item 5</option>

</select>

<div>

<button id="copyRight">Copy to Right →</button>

<button id="moveRight">Move to Right →</button>

</div>

</div>

<div class="list-box">

<h3>Destination List</h3>

<select id="destinationList" multiple>

</select>

<div>

<button id="copyLeft">← Copy to Left</button>

<button id="moveLeft">← Move to Left</button>

</div>

</div>

</div>

<!-- Create Element Section -->

<div class="create-element-section">

<h3>Create New Elements</h3>

<div>

<input type="text" id="newElementText" placeholder="Enter element text">

<button id="createButton">Create Element</button>

<button id="createWithAnimation">Create with Animation</button>

</div>

<div id="elementContainer">

<!-- New elements will be added here -->

</div>

</div>

<script>

$(document).ready(function() {

// Copy items from source to destination

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

$('#sourceList option:selected').each(function() {

var option = $(this).clone();

$('#destinationList').append(option);

});

});

// Move items from source to destination

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

$('#sourceList option:selected').each(function() {

$(this).appendTo('#destinationList');

});

});

// Copy items from destination to source

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

$('#destinationList option:selected').each(function() {

var option = $(this).clone();

$('#sourceList').append(option);

});

});

// Move items from destination to source

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

$('#destinationList option:selected').each(function() {

$(this).appendTo('#sourceList');

});

});

// Create new element (simple)

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

var text = $('#newElementText').val();

if (text) {

$('<div>', {

class: 'new-element',

text: text

}).appendTo('#elementContainer');

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

}

});

// Create new element with animation

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

var text = $('#newElementText').val();

if (text) {

var newElement = $('<div>', {

class: 'new-element',

text: text

}).css({

'opacity': '0',

'transform': 'scale(0.5)'

});

newElement.appendTo('#elementContainer').animate({

'opacity': '1'

}, 500).css({

'transform': 'scale(1)',

'transition': 'transform 0.5s'

});

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

}

});

// Double click to remove new elements

$(document).on('dblclick', '.new-element', function() {

$(this).fadeOut(300, function() {

$(this).remove();

});

});

// Allow drag and drop between lists

$('select').on('dragstart', 'option', function(e) {

e.originalEvent.dataTransfer.setData('text/plain', $(this).index());

});

$('select').on('dragover', function(e) {

e.preventDefault();

});

$('select').on('drop', function(e) {

e.preventDefault();

var sourceIndex = e.originalEvent.dataTransfer.getData('text/plain');

var sourceOption = $('option').eq(sourceIndex);

$(this).append(sourceOption.clone());

});

});

</script>

</body>

</html>

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.

// app.js

const express = require('express');

const app = express();

const path = require('path');

app.use(express.json());

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

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

res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

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

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

let 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);

}

res.json({

units: units,

bill: bill.toFixed(2),

breakdown: {

firstSlab: units > 50 ? 50 : units,

secondSlab: units > 150 ? 100 : (units > 50 ? units - 50 : 0),

thirdSlab: units > 250 ? 100 : (units > 150 ? units - 150 : 0),

fourthSlab: units > 250 ? units - 250 : 0

}

});

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

// public/index.html

<!DOCTYPE html>

<html>

<head>

<title>Electricity Bill Calculator</title>

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

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

<style>

.rate-info {

background-color: #f8f9fa;

padding: 15px;

border-radius: 5px;

margin-bottom: 20px;

}

.result {

display: none;

margin-top: 20px;

}

.breakdown {

margin-top: 20px;

}

.slab {

padding: 10px;

margin: 5px 0;

border-radius: 5px;

}

.slab-1 { background-color: #d1ecf1; }

.slab-2 { background-color: #d4edda; }

.slab-3 { background-color: #fff3cd; }

.slab-4 { background-color: #f8d7da; }

</style>

</head>

<body>

<div class="container mt-5">

<h2 class="mb-4">Electricity Bill Calculator</h2>

<div class="rate-info">

<h4>Rate Card</h4>

<ul>

<li>First 50 units: ₹3.50/unit</li>

<li>Next 100 units: ₹4.00/unit</li>

<li>Next 100 units: ₹5.20/unit</li>

<li>Above 250 units: ₹6.50/unit</li>

</ul>

</div>

<div class="card">

<div class="card-body">

<form id="billForm">

<div class="mb-3">

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

<input type="number" class="form-control" id="units" required min="0">

</div>

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

</form>

<div id="result" class="result">

<h4>Bill Details</h4>

<div class="row">

<div class="col-md-6">

<p>Units Consumed: <span id="totalUnits"></span></p>

<h3>Total Bill: ₹<span id="totalBill"></span></h3>

</div>

</div>

<div class="breakdown">

<h5>Bill Breakdown</h5>

<div class="slab slab-1" id="slab1"></div>

<div class="slab slab-2" id="slab2"></div>

<div class="slab slab-3" id="slab3"></div>

<div class="slab slab-4" id="slab4"></div>

</div>

</div>

</div>

</div>

</div>

<script>

$(document).ready(function() {

$('#billForm').on('submit', function(e) {

e.preventDefault();

const units = $('#units').val();

$.ajax({

url: '/calculate',

method: 'POST',

contentType: 'application/json',

data: JSON.stringify({ units: units }),

success: function(response) {

$('#totalUnits').text(response.units);

$('#totalBill').text(response.bill);

// Update breakdown

$('#slab1').text(`First 50 units: ${response.breakdown.firstSlab} units @ ₹3.50 = ₹${(response.breakdown.firstSlab \* 3.50).toFixed(2)}`);

$('#slab2').text(`Next 100 units: ${response.breakdown.secondSlab} units @ ₹4.00 = ₹${(response.breakdown.secondSlab \* 4.00).toFixed(2)}`);

$('#slab3').text(`Next 100 units: ${response.breakdown.thirdSlab} units @ ₹5.20 = ₹${(response.breakdown.thirdSlab \* 5.20).toFixed(2)}`);

$('#slab4').text(`Above 250 units: ${response.breakdown.fourthSlab} units @ ₹6.50 = ₹${(response.breakdown.fourthSlab \* 6.50).toFixed(2)}`);

$('#result').slideDown();

},

error: function(err) {

alert('Error calculating bill. Please try again.');

}

});

});

});

</script>

</body>

</html>

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.

// src/main/java/com/example/electricitybill/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);

}

}

// src/main/java/com/example/electricitybill/controller/BillController.java

package com.example.electricitybill.controller;

import com.example.electricitybill.model.BillRequest;

import com.example.electricitybill.model.BillResponse;

import com.example.electricitybill.service.BillService;

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

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

@RestController

@RequestMapping("/api")

public class BillController {

@Autowired

private BillService billService;

@PostMapping("/calculate")

public BillResponse calculateBill(@RequestBody BillRequest request) {

return billService.calculateBill(request);

}

}

// src/main/java/com/example/electricitybill/model/BillRequest.java

package com.example.electricitybill.model;

public class BillRequest {

private double units;

public double getUnits() {

return units;

}

public void setUnits(double units) {

this.units = units;

}

}

// src/main/java/com/example/electricitybill/model/BillResponse.java

package com.example.electricitybill.model;

public class BillResponse {

private double totalBill;

private String breakdown;

public double getTotalBill() {

return totalBill;

}

public void setTotalBill(double totalBill) {

this.totalBill = totalBill;

}

public String getBreakdown() {

return breakdown;

}

public void setBreakdown(String breakdown) {

this.breakdown = breakdown;

}

}

// src/main/java/com/example/electricitybill/service/BillService.java

package com.example.electricitybill.service;

import com.example.electricitybill.model.BillRequest;

import com.example.electricitybill.model.BillResponse;

import org.springframework.stereotype.Service;

@Service

public class BillService {

private static final double RATE\_FIRST\_50 = 3.50;

private static final double RATE\_51\_150 = 4.00;

private static final double RATE\_151\_250 = 5.20;

private static final double RATE\_ABOVE\_250 = 6.50;

public BillResponse calculateBill(BillRequest request) {

double units = request.getUnits();

double totalBill = 0;

StringBuilder breakdown = new StringBuilder();

if (units <= 50) {

totalBill = units \* RATE\_FIRST\_50;

breakdown.append(String.format("%.2f units × ₹%.2f = ₹%.2f", units, RATE\_FIRST\_50, totalBill));

} else if (units <= 150) {

totalBill = (50 \* RATE\_FIRST\_50) + ((units - 50) \* RATE\_51\_150);

breakdown.append(String.format("First 50 units × ₹%.2f = ₹%.2f\n", RATE\_FIRST\_50, 50 \* RATE\_FIRST\_50));

breakdown.append(String.format("Next %.2f units × ₹%.2f = ₹%.2f", units - 50, RATE\_51\_150, (units - 50) \* RATE\_51\_150));

} else if (units <= 250) {

totalBill = (50 \* RATE\_FIRST\_50) + (100 \* RATE\_51\_150) + ((units - 150) \* RATE\_151\_250);

breakdown.append(String.format("First 50 units × ₹%.2f = ₹%.2f\n", RATE\_FIRST\_50, 50 \* RATE\_FIRST\_50));

breakdown.append(String.format("Next 100 units × ₹%.2f = ₹%.2f\n", RATE\_51\_150, 100 \* RATE\_51\_150));

breakdown.append(String.format("Next %.2f units × ₹%.2f = ₹%.2f", units - 150, RATE\_151\_250, (units - 150) \* RATE\_151\_250));

} else {

totalBill = (50 \* RATE\_FIRST\_50) + (100 \* RATE\_51\_150) + (100 \* RATE\_151\_250) + ((units - 250) \* RATE\_ABOVE\_250);

breakdown.append(String.format("First 50 units × ₹%.2f = ₹%.2f\n", RATE\_FIRST\_50, 50 \* RATE\_FIRST\_50));

breakdown.append(String.format("Next 100 units × ₹%.2f = ₹%.2f\n", RATE\_51\_150, 100 \* RATE\_51\_150));

breakdown.append(String.format("Next 100 units × ₹%.2f = ₹%.2f\n", RATE\_151\_250, 100 \* RATE\_151\_250));

breakdown.append(String.format("Remaining %.2f units × ₹%.2f = ₹%.2f", units - 250, RATE\_ABOVE\_250, (units - 250) \* RATE\_ABOVE\_250));

}

BillResponse response = new BillResponse();

response.setTotalBill(totalBill);

response.setBreakdown(breakdown.toString());

return response;

}

}

<!-- src/main/resources/static/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">

<style>

.result-box {

display: none;

margin-top: 20px;

padding: 20px;

border-radius: 5px;

background-color: #f8f9fa;

}

.breakdown {

white-space: pre-line;

}

</style>

</head>

<body>

<div class="container mt-5">

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card">

<div class="card-header bg-primary text-white">

<h4 class="mb-0">Electricity Bill Calculator</h4>

</div>

<div class="card-body">

<form id="billForm">

<div class="mb-3">

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

<input type="number" class="form-control" id="units" required min="0" step="0.01">

</div>

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

</form>

<div id="resultBox" class="result-box">

<h5>Bill Details</h5>

<div class="breakdown" id="breakdown"></div>

<hr>

<h4>Total Bill: ₹<span id="totalBill">0.00</span></h4>

</div>

</div>

</div>

</div>

</div>

</div>

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

<script>

$(document).ready(function() {

$('#billForm').on('submit', function(e) {

e.preventDefault();

const units = parseFloat($('#units').val());

$.ajax({

url: '/api/calculate',

method: 'POST',

contentType: 'application/json',

data: JSON.stringify({ units: units }),

success: function(response) {

$('#breakdown').text(response.breakdown);

$('#totalBill').text(response.totalBill.toFixed(2));

$('#resultBox').slideDown();

},

error: function(xhr, status, error) {

alert('Error calculating bill. Please try again.');

}

});

});

});

</script>

</body>

</html>

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.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>My CV</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/css/bootstrap.min.css" rel="stylesheet">

<style>

.video-background {

position: fixed;

right: 0;

bottom: 0;

min-width: 100%;

min-height: 100%;

z-index: -1;

opacity: 0.3;

}

.content {

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

padding: 30px;

margin: 20px;

border-radius: 10px;

}

.section {

margin-bottom: 30px;

}

@media (max-width: 768px) {

.col-md-6 {

margin-bottom: 20px;

}

}

</style>

</head>

<body>

<video autoplay muted loop class="video-background">

<source src="/api/placeholder/400/320" type="video/mp4">

</video>

<div class="container content">

<header class="text-center mb-5">

<h1>John Doe</h1>

<p>Software Developer</p>

</header>

<div class="row">

<div class="col-md-6">

<div class="section">

<h2>Education</h2>

<ul class="list-unstyled">

<li>

<h5>Bachelor of Technology in Computer Science</h5>

<p>VIT University, 2020-2024</p>

</li>

</ul>

</div>

<div class="section">

<h2>Skills</h2>

<ul>

<li>JavaScript, React, Node.js</li>

<li>Python, Java, Spring Boot</li>

<li>HTML5, CSS3, Bootstrap</li>

<li>MySQL, MongoDB</li>

</ul>

</div>

</div>

<div class="col-md-6">

<div class="section">

<h2>Work Experience</h2>

<div class="mb-4">

<h5>Software Developer Intern</h5>

<p>Tech Solutions Inc., Summer 2023</p>

<ul>

<li>Developed responsive web applications</li>

<li>Collaborated with senior developers</li>

<li>Implemented new features using React</li>

</ul>

</div>

</div>

<div class="section">

<h2>Projects</h2>

<ul>

<li>

<h5>E-Commerce Platform</h5>

<p>Built using MERN stack with payment integration</p>

</li>

<li>

<h5>Student Management System</h5>

<p>Developed using Spring Boot and MySQL</p>

</li>

</ul>

</div>

</div>

</div>

<footer class="text-center mt-5">

<p>Contact: john.doe@email.com | Phone: (123) 456-7890</p>

</footer>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

</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.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>VIT Courses</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/css/bootstrap.min.css" rel="stylesheet">

<style>

.tab-content {

padding: 20px;

border: 1px solid #dee2e6;

border-top: 0;

}

.nav-link.active {

font-weight: bold;

}

</style>

</head>

<body>

<div class="container mt-5">

<h2 class="text-center mb-4">VIT Course Information</h2>

<ul class="nav nav-tabs" id="courseTabs" role="tablist">

<li class="nav-item" role="presentation">

<button class="nav-link active" id="sdp-tab" data-bs-toggle="tab" data-bs-target="#sdp" type="button" role="tab">SDP</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="edi-tab" data-bs-toggle="tab" data-bs-target="#edi" type="button" role="tab">EDI</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="dt-tab" data-bs-toggle="tab" data-bs-target="#dt" type="button" role="tab">DT</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="projects-tab" data-bs-toggle="tab" data-bs-target="#projects" type="button" role="tab">Course Projects</button>

</li>

</ul>

<div class="tab-content" id="courseTabContent">

<div class="tab-pane fade show active" id="sdp" role="tabpanel">

<h3>Software Development Practices (SDP)</h3>

<p>SDP focuses on:</p>

<ul>

<li>Agile methodologies and practices</li>

<li>Version control systems</li>

<li>Code quality and testing</li>

<li>Team collaboration and project management</li>

</ul>

<p>This course helps students develop professional software development skills essential in the industry.</p>

</div>

<div class="tab-pane fade" id="edi" role="tabpanel">

<h3>Entrepreneurship Development and Innovation (EDI)</h3>

<p>EDI covers:</p>

<ul>

<li>Business model development</li>

<li>Market analysis and research</li>

<li>Startup fundamentals</li>

<li>Innovation strategies</li>

</ul>

<p>Students learn to transform innovative ideas into viable business ventures.</p>

</div>

<div class="tab-pane fade" id="dt" role="tabpanel">

<h3>Digital Technologies (DT)</h3>

<p>DT explores:</p>

<ul>

<li>Emerging digital technologies</li>

<li>Cloud computing</li>

<li>IoT and connected systems</li>

<li>Digital transformation strategies</li>

</ul>

<p>This course prepares students for the digital future of technology and business.</p>

</div>

<div class="tab-pane fade" id="projects" role="tabpanel">

<h3>Course Projects</h3>

<p>Project-based learning includes:</p>

<ul>

<li>Team-based development projects</li>

<li>Industry-sponsored challenges</li>

<li>Innovation competitions</li>

<li>Research-oriented projects</li>

</ul>

<p>Projects provide hands-on experience in applying learned concepts.</p>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>

<script>

$(document).ready(function() {

$('#courseTabs button').on('click', function (e) {

e.preventDefault();

$(this).tab('show');

});

});

</script>

</body>

</html>

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.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Array Operations</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/css/bootstrap.min.css" rel="stylesheet">

<style>

.result {

margin-top: 20px;

padding: 15px;

background-color: #f8f9fa;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="container mt-5">

<h2 class="mb-4">Array Operations Demo</h2>

<div class="row">

<div class="col-md-6">

<h3>Integer Array Operations</h3>

<div class="mb-3">

<label for="integerInput" class="form-label">Enter integers (comma-separated)</label>

<input type="text" class="form-control" id="integerInput" placeholder="1, 5, 3, 2, 4">

</div>

<button class="btn btn-primary me-2" onclick="sortIntegers()">Sort</button>

<div class="mb-3 mt-3">

<label for="searchInt" class="form-label">Search for number:</label>

<input type="number" class="form-control" id="searchInt">

</div>

<button class="btn btn-primary" onclick="searchInteger()">Search</button>

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

</div>

<div class="col-md-6">

<h3>Named Entities Array</h3>

<div class="mb-3">

<label for="nameInput" class="form-label">Enter names (comma-separated)</label>

<input type="text" class="form-control" id="nameInput" placeholder="John, Alice, Bob">

</div>

<button class="btn btn-primary me-2" onclick="sortNames()">Sort</button>

<div class="mb-3 mt-3">

<label for="searchName" class="form-label">Search for name:</label>

<input type="text" class="form-control" id="searchName">

</div>

<button class="btn btn-primary" onclick="searchName()">Search</button>

<div id="nameResult" class="result"></div>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>

<script>

// Integer Array Operations

function sortIntegers() {

const input = document.getElementById('integerInput').value;

const arr = input.split(',').map(num => parseInt(num.trim()));

const sorted = [...arr].sort((a, b) => a - b);

document.getElementById('integerResult').innerHTML = `

<p><strong>Original array:</strong> ${arr.join(', ')}</p>

<p><strong>Sorted array:</strong> ${sorted.join(', ')}</p>

`;

}

function searchInteger() {

const input = document.getElementById('integerInput').value;

const arr = input.split(',').map(num => parseInt(num.trim()));

const searchValue = parseInt(document.getElementById('searchInt').value);

const index = arr.indexOf(searchValue);

const result = index !== -1

? `Found ${searchValue} at index ${index}`

: `${searchValue} not found in array`;

document.getElementById('integerResult').innerHTML += `

<p><strong>Search result:</strong> ${result}</p>

`;

}

// Named Entities Array Operations

function sortNames() {

const input = document.getElementById('nameInput').value;

const arr = input.split(',').map(name => name.trim());

const sorted = [...arr].sort();

document.getElementById('nameResult').innerHTML = `

<p><strong>Original array:</strong> ${arr.join(', ')}</p>

<p><strong>Sorted array:</strong> ${sorted.join(', ')}</p>

`;

}

function searchName() {

const input = document.getElementById('nameInput').value;

const arr = input.split(',').map(name => name.trim());

const searchValue = document.getElementById('searchName').value.trim();

const index = arr.findIndex(name =>

name.toLowerCase() === searchValue.toLowerCase()

);

const result = index !== -1

? `Found "${searchValue}" at index ${index}`

: `"${searchValue}" not found in array`;

document.getElementById('nameResult').innerHTML += `

<p><strong>Search result:</strong> ${result}</p>

`;

}

</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.

import React, { useState } from 'react';

import { Card, CardHeader, CardContent } from '@/components/ui/card';

const ElectricityBillCalculator = () => {

const [units, setUnits] = useState('');

const [bill, setBill] = useState(null);

const calculateBill = () => {

const unitsNum = parseFloat(units);

let total = 0;

if (unitsNum <= 50) {

total = unitsNum \* 3.50;

} else if (unitsNum <= 150) {

total = (50 \* 3.50) + ((unitsNum - 50) \* 4.00);

} else if (unitsNum <= 250) {

total = (50 \* 3.50) + (100 \* 4.00) + ((unitsNum - 150) \* 5.20);

} else {

total = (50 \* 3.50) + (100 \* 4.00) + (100 \* 5.20) + ((unitsNum - 250) \* 6.50);

}

setBill(total.toFixed(2));

};

return (

<Card className="w-full max-w-md mx-auto">

<CardHeader>

<h2 className="text-2xl font-bold text-center">Electricity Bill Calculator</h2>

</CardHeader>

<CardContent>

<div className="space-y-4">

<div className="bg-gray-100 p-4 rounded-lg">

<h3 className="font-semibold mb-2">Rate Card:</h3>

<ul className="list-disc pl-5 space-y-1">

<li>First 50 units: ₹3.50/unit</li>

<li>51-150 units: ₹4.00/unit</li>

<li>151-250 units: ₹5.20/unit</li>

<li>Above 250 units: ₹6.50/unit</li>

</ul>

</div>

<div className="space-y-2">

<label htmlFor="units" className="block font-medium">

Enter Units Consumed:

</label>

<input

id="units"

type="number"

value={units}

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

className="w-full p-2 border rounded"

placeholder="Enter units"

/>

</div>

<button

onClick={calculateBill}

className="w-full bg-blue-500 text-white p-2 rounded hover:bg-blue-600 transition-colors"

>

Calculate Bill

</button>

{bill && (

<div className="mt-4 p-4 bg-green-100 rounded-lg">

<h3 className="font-semibold">Bill Amount:</h3>

<p className="text-2xl font-bold text-green-700">₹{bill}</p>

</div>

)}

</div>

</CardContent>

</Card>

);

};

export default ElectricityBillCalculator;

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.

<?php

// bill\_calculator.php

function calculateBill($units) {

$bill = 0;

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);

}

return number\_format($bill, 2);

}

?>

<!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://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

</head>

<body>

<div class="container mt-5">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="card">

<div class="card-header">

<h2 class="text-center">Electricity Bill Calculator</h2>

</div>

<div class="card-body">

<form method="POST" id="billForm">

<div class="mb-4">

<h4>Rate Card:</h4>

<ul class="list-group">

<li class="list-group-item">First 50 units: ₹3.50/unit</li>

<li class="list-group-item">51-150 units: ₹4.00/unit</li>

<li class="list-group-item">151-250 units: ₹5.20/unit</li>

<li class="list-group-item">Above 250 units: ₹6.50/unit</li>

</ul>

</div>

<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 w-100">Calculate Bill</button>

</form>

<?php

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

$units = floatval($\_POST['units']);

$bill = calculateBill($units);

echo "<div class='alert alert-success mt-3'>

<h4>Bill Amount:</h4>

<h3 class='text-center'>₹{$bill}</h3>

</div>";

}

?>

</div>

</div>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>

<script>

$(document).ready(function() {

$('#billForm').on('submit', function(e) {

const units = $('#units').val();

if (!units || units < 0) {

e.preventDefault();

alert('Please enter a valid number of units');

}

});

});

</script>

</body>

</html>

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%).

import React, { useState } from 'react';

import { Card, CardHeader, CardContent } from '@/components/ui/card';

const ResultCalculator = () => {

const [subjects, setSubjects] = useState([

{ name: 'Web Programming', mse: '', ese: '', total: null },

{ name: 'Data Structures', mse: '', ese: '', total: null },

{ name: 'Database Systems', mse: '', ese: '', total: null },

{ name: 'Operating Systems', mse: '', ese: '', total: null }

]);

const [studentInfo, setStudentInfo] = useState({

name: '',

rollNo: '',

semester: '5'

});

const calculateTotal = (mse, ese) => {

const mseMark = parseFloat(mse);

const eseMark = parseFloat(ese);

if (!isNaN(mseMark) && !isNaN(eseMark)) {

const msePercentage = (mseMark / 50) \* 30; // MSE is out of 50, converted to 30%

const esePercentage = (eseMark / 100) \* 70; // ESE is out of 100, converted to 70%

return msePercentage + esePercentage;

}

return null;

};

const handleSubjectChange = (index, field, value) => {

const newSubjects = [...subjects];

newSubjects[index][field] = value;

if (field === 'mse' || field === 'ese') {

newSubjects[index].total = calculateTotal(

field === 'mse' ? value : newSubjects[index].mse,

field === 'ese' ? value : newSubjects[index].ese

);

}

setSubjects(newSubjects);

};

const calculateGrade = (total) => {

if (total >= 90) return 'O';

if (total >= 80) return 'A+';

if (total >= 70) return 'A';

if (total >= 60) return 'B+';

if (total >= 50) return 'B';

return 'F';

};

const calculateSGPA = () => {

const validSubjects = subjects.filter(sub => sub.total !== null);

if (validSubjects.length === 0) return 0;

const totalGradePoints = validSubjects.reduce((acc, sub) => {

const grade = calculateGrade(sub.total);

const gradePoint = grade === 'O' ? 10 :

grade === 'A+' ? 9 :

grade === 'A' ? 8 :

grade === 'B+' ? 7 :

grade === 'B' ? 6 : 0;

return acc + gradePoint;

}, 0);

return (totalGradePoints / validSubjects.length).toFixed(2);

};

return (

<Card className="w-full max-w-4xl mx-auto">

<CardHeader>

<h2 className="text-2xl font-bold text-center">VIT Semester Result Calculator</h2>

</CardHeader>

<CardContent>

<div className="grid grid-cols-1 md:grid-cols-3 gap-4 mb-6">

<div>

<label className="block mb-1">Student Name:</label>

<input

type="text"

className="w-full p-2 border rounded"

value={studentInfo.name}

onChange={(e) => setStudentInfo({...studentInfo, name: e.target.value})}

/>

</div>

<div>

<label className="block mb-1">Roll Number:</label>

<input

type="text"

className="w-full p-2 border rounded"

value={studentInfo.rollNo}

onChange={(e) => setStudentInfo({...studentInfo, rollNo: e.target.value})}

/>

</div>

<div>

<label className="block mb-1">Semester:</label>

<input

type="text"

className="w-full p-2 border rounded"

value={studentInfo.semester}

readOnly

/>

</div>

</div>

<div className="overflow-x-auto">

<table className="w-full border-collapse border">

<thead>

<tr className="bg-gray-100">

<th className="border p-2">Subject</th>

<th className="border p-2">MSE (50)</th>

<th className="border p-2">ESE (100)</th>

<th className="border p-2">Total (100)</th>

<th className="border p-2">Grade</th>

</tr>

</thead>

<tbody>

{subjects.map((subject, index) => (

<tr key={index}>

<td className="border p-2">{subject.name}</td>

<td className="border p-2">

<input

type="number"

className="w-full p-1 border rounded"

value={subject.mse}

onChange={(e) => handleSubjectChange(index, 'mse', e.target.value)}

min="0"

max="50"

/>

</td>

<td className="border p-2">

<input

type="number"

className="w-full p-1 border rounded"

value={subject.ese}

onChange={(e) => handleSubjectChange(index, 'ese', e.target.value)}

min="0"

max="100"

/>

</td>

<td className="border p-2 text-center">

{subject.total ? subject.total.toFixed(2) : '-'}

</td>

<td className="border p-2 text-center">

{subject.total ? calculateGrade(subject.total) : '-'}

</td>

</tr>

))}

</tbody>

</table>

</div>

<div className="mt-6 p-4 bg-gray-100 rounded-lg">

<h3 className="text-xl font-bold">Semester GPA: {calculateSGPA()}</h3>

</div>

</CardContent>

</Card>

);

};

export default ResultCalculator;

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%).

<?php

// config.php

$host = 'localhost';

$dbname = 'vit\_results';

$username = 'root';

$password = '';

try {

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

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

} catch(PDOException $e) {

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

}

// Function to calculate grade

function calculateGrade($total) {

if ($total >= 90) return 'O';

if ($total >= 80) return 'A+';

if ($total >= 70) return 'A';

if ($total >= 60) return 'B+';

if ($total >= 50) return 'B';

return 'F';

}

// Function to calculate total marks

function calculateTotal($mse, $ese) {

$msePercentage = ($mse / 50) \* 30; // MSE is out of 50, converted to 30%

$esePercentage = ($ese / 100) \* 70; // ESE is out of 100, converted to 70%

return $msePercentage + $esePercentage;

}

// Save results

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

try {

$stmt = $pdo->prepare("INSERT INTO results (student\_name, roll\_no, semester, subject\_name, mse\_marks, ese\_marks, total\_marks, grade)

VALUES (:name, :roll, :sem, :subject, :mse, :ese, :total, :grade)");

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

$roll\_no = $\_POST['roll\_no'];

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

foreach ($\_POST['subjects'] as $subject) {

$total = calculateTotal($subject['mse'], $subject['ese']);

$grade = calculateGrade($total);

$stmt->execute([

':name' => $student\_name,

':roll' => $roll\_no,

':sem' => $semester,

':subject' => $subject['name'],

':mse' => $subject['mse'],

':ese' => $subject['ese'],

':total' => $total,

':grade' => $grade

]);

}

$success = true;

} catch(PDOException $e) {

$error = "Error: " . $e->getMessage();

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>VIT Result Calculator</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.2/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container mt-5">

<h2 class="text-center mb-4">VIT Semester Result Calculator</h2>

<form method="POST" id="resultForm">

<div class="row mb-4">

<div class="col-md-4">

<div class="form-group">

<label>Student Name:</label>

<input type="text" name="student\_name" class="form-control" required>

</div>

</div>

<div class="col-md-4">

<div class="form-group">

<label>Roll Number:</label>

<input type="text" name="roll\_no" class="form-control" required>

</div>

</div>

<div class="col-md-4">

<div class="form-group">

<label>Semester:</label>

<input type="number" name="semester" class="form-control" value="5" readonly>

</div>

</div>

</div>

<table class="table table-bordered">

<thead>

<tr>

<th>Subject</th>

<th>MSE Marks (50)</th>

<th>ESE Marks (100)</th>

<th>Total</th>

<th>Grade</th>

</tr>

</thead>

<tbody>

<?php

$subjects = ['Web Programming', 'Data Structures', 'Database Systems', 'Operating Systems'];

foreach ($subjects as $index => $subject) {

echo "

<tr>

<td>

<input type='hidden' name='subjects[$index][name]' value='$subject'>

$subject

</td>

<td>

<input type='number' name='subjects[$index][mse]' class='form-control mse'

min='0' max='50' required>

</td>

<td>

<input type='number' name='subjects[$index][ese]' class='form-control ese'

min='0' max='100' required>

</td>

<td class='total'>-</td>

<td class='grade'>-</td>

</tr>";

}

?>

</tbody>

</table>

<div class="text-center mt-4">

<button type="submit" class="btn btn-primary">Save Results</button>

</div>

</form>

<?php if (isset($success)): ?>

<div class="alert alert-success mt-4">

Results saved successfully!

</div>

<?php endif; ?>

<?php if (isset($error)): ?>

<div class="alert alert-danger mt-4">

<?php echo $error; ?>

</div>

<?php endif; ?>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<script>

$(document).ready(function() {

function updateResults(row) {

const mse = parseFloat($(row).find('.mse').val()) || 0;

const ese = parseFloat($(row).find('.ese').val()) || 0;

if (mse <= 50 && ese <= 100) {

const total = (mse / 50 \* 30) + (ese / 100 \* 70);

$(row).find('.total').text(total.toFixed(2));

let grade = 'F';

if (total >= 90) grade = 'O';

else if (total >= 80) grade = 'A+';

else if (total >= 70) grade = 'A';

else if (total >= 60) grade = 'B+';

else if (total >= 50) grade = 'B';

$(row).find('.grade').text(grade);

}

}

$('.mse, .ese').on('input', function() {

updateResults($(this).closest('tr'));

});

$('#resultForm').on('submit', function(e) {

const isValid = Array.from(document.querySelectorAll('.mse, .ese')).every(input => {

const val = parseFloat(input.value);

return !isNaN(val) && val >= 0 && val <= (input.classList.contains('mse') ? 50 : 100);

});

if (!isValid) {

e.preventDefault();

alert('Please enter valid marks (0-50 for MSE, 0-100 for ESE)');

}

});

});

</script>

</body>

</html>

**Result schema :**

CREATE DATABASE vit\_results;

USE vit\_results;

CREATE TABLE results (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100) NOT NULL,

roll\_no VARCHAR(20) NOT NULL,

semester INT NOT NULL,

subject\_name VARCHAR(100) NOT NULL,

mse\_marks FLOAT NOT NULL,

ese\_marks FLOAT NOT NULL,

total\_marks FLOAT NOT NULL,

grade CHAR(2) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

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%).

// App.js

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import StudentResults from './components/StudentResults';

import ResultEntry from './components/ResultEntry';

import Navbar from './components/Navbar';

function App() {

return (

<Router>

<div className="min-h-screen bg-gray-100">

<Navbar />

<div className="container mx-auto px-4 py-8">

<Routes>

<Route path="/" element={<StudentResults />} />

<Route path="/entry" element={<ResultEntry />} />

</Routes>

</div>

</div>

</Router>

);

}

export default App;

// components/StudentResults.js

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

import axios from 'axios';

function StudentResults() {

const [results, setResults] = useState([]);

useEffect(() => {

fetchResults();

}, []);

const fetchResults = async () => {

try {

const response = await axios.get('http://localhost:5000/api/results');

setResults(response.data);

} catch (error) {

console.error('Error fetching results:', error);

}

};

const calculateTotal = (mse, ese) => {

return (mse \* 0.3) + (ese \* 0.7);

};

return (

<div className="bg-white shadow-md rounded-lg p-6">

<h2 className="text-2xl font-bold mb-4">Semester Results</h2>

<div className="overflow-x-auto">

<table className="min-w-full table-auto">

<thead>

<tr className="bg-gray-200">

<th className="px-4 py-2">Roll No</th>

<th className="px-4 py-2">Student Name</th>

{['Subject 1', 'Subject 2', 'Subject 3', 'Subject 4'].map((subject) => (

<React.Fragment key={subject}>

<th className="px-4 py-2">{subject} MSE</th>

<th className="px-4 py-2">{subject} ESE</th>

<th className="px-4 py-2">{subject} Total</th>

</React.Fragment>

))}

<th className="px-4 py-2">Overall Percentage</th>

</tr>

</thead>

<tbody>

{results.map((result) => (

<tr key={result.id}>

<td className="border px-4 py-2">{result.rollNo}</td>

<td className="border px-4 py-2">{result.studentName}</td>

{['subject1', 'subject2', 'subject3', 'subject4'].map((subject) => (

<React.Fragment key={subject}>

<td className="border px-4 py-2">{result[`${subject}Mse`]}</td>

<td className="border px-4 py-2">{result[`${subject}Ese`]}</td>

<td className="border px-4 py-2">

{calculateTotal(result[`${subject}Mse`], result[`${subject}Ese`]).toFixed(2)}

</td>

</React.Fragment>

))}

<td className="border px-4 py-2">

{(

Object.keys(result)

.filter(key => key.endsWith('Mse') || key.endsWith('Ese'))

.reduce((acc, key) => {

const value = result[key];

const weight = key.endsWith('Mse') ? 0.3 : 0.7;

return acc + (value \* weight);

}, 0) / 4

).toFixed(2)}%

</td>

</tr>

))}

</tbody>

</table>

</div>

</div>

);

}

export default StudentResults;

**Node.js Backend:**

// server.js

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(express.json());

const db = mysql.createConnection({

host: 'localhost',

user: 'root',

password: 'your\_password',

database: 'vit\_results'

});

db.connect((err) => {

if (err) {

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

return;

}

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

});

// Get all results

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

const query = 'SELECT \* FROM student\_results';

db.query(query, (err, results) => {

if (err) {

console.error('Error fetching results:', err);

res.status(500).json({ error: 'Error fetching results' });

return;

}

res.json(results);

});

});

// Add new result

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

const {

rollNo,

studentName,

subject1Mse,

subject1Ese,

subject2Mse,

subject2Ese,

subject3Mse,

subject3Ese,

subject4Mse,

subject4Ese

} = req.body;

const query = `

INSERT INTO student\_results

(rollNo, studentName, subject1Mse, subject1Ese, subject2Mse, subject2Ese,

subject3Mse, subject3Ese, subject4Mse, subject4Ese)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

`;

db.query(

query,

[rollNo, studentName, subject1Mse, subject1Ese, subject2Mse, subject2Ese,

subject3Mse, subject3Ese, subject4Mse, subject4Ese],

(err, result) => {

if (err) {

console.error('Error adding result:', err);

res.status(500).json({ error: 'Error adding result' });

return;

}

res.json({ message: 'Result added successfully', id: result.insertId });

}

);

});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

-- Create database

CREATE DATABASE IF NOT EXISTS vit\_results;

USE vit\_results;

-- Create student\_results table

CREATE TABLE IF NOT EXISTS student\_results (

id INT AUTO\_INCREMENT PRIMARY KEY,

rollNo VARCHAR(20) NOT NULL,

studentName VARCHAR(100) NOT NULL,

subject1Mse DECIMAL(5,2) NOT NULL,

subject1Ese DECIMAL(5,2) NOT NULL,

subject2Mse DECIMAL(5,2) NOT NULL,

subject2Ese DECIMAL(5,2) NOT NULL,

subject3Mse DECIMAL(5,2) NOT NULL,

subject3Ese DECIMAL(5,2) NOT NULL,

subject4Mse DECIMAL(5,2) NOT NULL,

subject4Ese DECIMAL(5,2) NOT NULL,

createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Add index for faster lookups

CREATE INDEX idx\_rollno ON student\_results(rollNo);

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)

// Book.java

package com.bookstore.model;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "books")

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private String isbn;

private Double price;

private String description;

private String imageUrl;

private Integer stockQuantity;

}

// User.java

package com.bookstore.model;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(unique = true)

private String email;

private String password;

private String fullName;

private String address;

private String role = "USER"; // USER or ADMIN

}

// BookController.java

package com.bookstore.controller;

import com.bookstore.model.Book;

import com.bookstore.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.\*;

@Controller

@RequestMapping("/books")

public class BookController {

@Autowired

private BookService bookService;

@GetMapping

public String listBooks(Model model) {

model.addAttribute("books", bookService.getAllBooks());

return "catalogue";

}

@GetMapping("/{id}")

public String viewBook(@PathVariable Long id, Model model) {

model.addAttribute("book", bookService.getBookById(id));

return "book-detail";

}

}

// UserController.java

package com.bookstore.controller;

import com.bookstore.model.User;

import com.bookstore.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("/login")

public String loginPage() {

return "login";

}

@GetMapping("/register")

public String registrationPage(Model model) {

model.addAttribute("user", new User());

return "register";

}

@PostMapping("/register")

public String registerUser(@ModelAttribute User user) {

userService.registerUser(user);

return "redirect:/login";

}

}

<!-- templates/home.html -->

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Online Book Store</title>

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

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container">

<a class="navbar-brand" href="/">Book Store</a>

<div class="collapse navbar-collapse">

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

<li class="nav-item">

<a class="nav-link" href="/books">Catalogue</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/login">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/register">Register</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container mt-5">

<div class="jumbotron">

<h1>Welcome to Online Book Store</h1>

<p>Discover your next favorite book today!</p>

<a href="/books" class="btn btn-primary">Browse Books</a>

</div>

</div>

</body>

</html>

<!-- templates/login.html -->

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Login - Book Store</title>

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

</head>

<body>

<div class="container mt-5">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="card">

<div class="card-header">Login</div>

<div class="card-body">

<form th:action="@{/login}" method="post">

<div class="mb-3">

<label for="email" class="form-label">Email</label>

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

</div>

<div class="mb-3">

<label for="password" class="form-label">Password</label>

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

</div>

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

</form>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

<!-- templates/catalogue.html -->

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Catalogue - Book Store</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Book Catalogue</h2>

<div class="row">

<div class="col-md-4 mb-4" th:each="book : ${books}">

<div class="card">

<img th:src="${book.imageUrl}" class="card-img-top" alt="Book Cover">

<div class="card-body">

<h5 class="card-title" th:text="${book.title}">Book Title</h5>

<p class="card-text" th:text="${book.author}">Author</p>

<p class="card-text" th:text="${'$' + book.price}">Price</p>

<a th:href="@{/books/{id}(id=${book.id})}" class="btn btn-primary">View Details</a>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

-- Create database

CREATE DATABASE bookstore;

USE bookstore;

-- Create books table

CREATE TABLE books (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

isbn VARCHAR(13) NOT NULL UNIQUE,

price DECIMAL(10,2) NOT NULL,

description TEXT,

image\_url VARCHAR(255),

stock\_quantity INT NOT NULL DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Create users table

CREATE TABLE users (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

full\_name VARCHAR(255) NOT NULL,

address TEXT,

role VARCHAR(20) NOT NULL DEFAULT 'USER',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Add indexes

CREATE INDEX idx\_email ON users(email);

CREATE INDEX idx\_isbn ON books(isbn);

# application.properties

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

spring.datasource.username=root

spring.datasource.password=your\_password

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.thymeleaf.cache=false

server.port=8080

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)

// App.js

import React from 'react';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Navbar from './components/Navbar';

import Home from './pages/Home';

import Login from './pages/Login';

import Register from './pages/Register';

import Catalogue from './pages/Catalogue';

import { AuthProvider } from './context/AuthContext';

function App() {

return (

<AuthProvider>

<Router>

<div className="min-h-screen bg-gray-50">

<Navbar />

<Routes>

<Route path="/" element={<Home />} />

<Route path="/login" element={<Login />} />

<Route path="/register" element={<Register />} />

<Route path="/catalogue" element={<Catalogue />} />

</Routes>

</div>

</Router>

</AuthProvider>

);

}

export default App;

// components/BookCard.js

import React from 'react';

const BookCard = ({ book }) => {

return (

<div className="bg-white rounded-lg shadow-md overflow-hidden">

<img

src={book.imageUrl}

alt={book.title}

className="w-full h-48 object-cover"

/>

<div className="p-4">

<h3 className="text-lg font-semibold">{book.title}</h3>

<p className="text-gray-600">{book.author}</p>

<p className="text-green-600 font-bold mt-2">${book.price}</p>

<button className="w-full mt-3 bg-blue-600 text-white py-2 rounded hover:bg-blue-700">

View Details

</button>

</div>

</div>

);

};

// pages/Login.js

import React, { useState } from 'react';

import { useNavigate } from 'react-router-dom';

import { useAuth } from '../context/AuthContext';

const Login = () => {

const [formData, setFormData] = useState({

email: '',

password: ''

});

const navigate = useNavigate();

const { login } = useAuth();

const handleSubmit = async (e) => {

e.preventDefault();

try {

await login(formData);

navigate('/');

} catch (error) {

console.error('Login error:', error);

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-gray-50 py-12 px-4 sm:px-6 lg:px-8">

<div className="max-w-md w-full space-y-8">

<div>

<h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">

Sign in to your account

</h2>

</div>

<form className="mt-8 space-y-6" onSubmit={handleSubmit}>

<div className="rounded-md shadow-sm -space-y-px">

<div>

<input

type="email"

required

className="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-t-md focus:outline-none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

placeholder="Email address"

value={formData.email}

onChange={(e) => setFormData({ ...formData, email: e.target.value })}

/>

</div>

<div>

<input

type="password"

required

className="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-b-md focus:outline-none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

placeholder="Password"

value={formData.password}

onChange={(e) => setFormData({ ...formData, password: e.target.value })}

/>

</div>

</div>

<div>

<button

type="submit"

className="group relative w-full flex justify-center py-2 px-4 border border-transparent text-sm font-medium rounded-md text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500"

>

Sign in

</button>

</div>

</form>

</div>

</div>

);

};

// server.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

require('dotenv').config();

const app = express();

app.use(cors());

app.use(express.json());

// MongoDB connection

mongoose.connect(process.env.MONGODB\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Models

const userSchema = new mongoose.Schema({

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

fullName: { type: String, required: true },

address: String,

role: { type: String, default: 'USER' }

});

const bookSchema = new mongoose.Schema({

title: { type: String, required: true },

author: { type: String, required: true },

isbn: { type: String, required: true, unique: true },

price: { type: Number, required: true },

description: String,

imageUrl: String,

stockQuantity: { type: Number, default: 0 }

});

const User = mongoose.model('User', userSchema);

const Book = mongoose.model('Book', bookSchema);

// Middleware

const auth = async (req, res, next) => {

try {

const token = req.header('Authorization').replace('Bearer ', '');

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

const user = await User.findById(decoded.userId);

if (!user) throw new Error();

req.user = user;

next();

} catch (error) {

res.status(401).send({ error: 'Please authenticate' });

}

};

// Routes

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

try {

const { email, password, fullName, address } = req.body;

const hashedPassword = await bcrypt.hash(password, 8);

const user = new User({

email,

password: hashedPassword,

fullName,

address

});

await user.save();

const token = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET);

res.status(201).send({ user, token });

} catch (error) {

res.status(400).send(error);

}

});

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

try {

const { email, password } = req.body;

const user = await User.findOne({ email });

if (!user) throw new Error('Invalid login credentials');

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) throw new Error('Invalid login credentials');

const token = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET);

res.send({ user, token });

} catch (error) {

res.status(400).send({ error: error.message });

}

});

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

try {

const books = await Book.find();

res.send(books);

} catch (error) {

res.status(500).send(error);

}

});

app.post('/api/books', auth, async (req, res) => {

try {

if (req.user.role !== 'ADMIN') {

return res.status(403).send({ error: 'Not authorized' });

}

const book = new Book(req.body);

await book.save();

res.status(201).send(book);

} catch (error) {

res.status(400).send(error);

}

});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

// context/AuthContext.js

import React, { createContext, useContext, useState } from 'react';

import axios from 'axios';

const AuthContext = createContext();

export function useAuth() {

return useContext(AuthContext);

}

export function AuthProvider({ children }) {

const [user, setUser] = useState(null);

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

const login = async (credentials) => {

try {

const response = await axios.post('http://localhost:5000/api/login', credentials);

setUser(response.data.user);

setToken(response.data.token);

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

} catch (error) {

throw error;

}

};

const register = async (userData) => {

try {

const response = await axios.post('http://localhost:5000/api/register', userData);

setUser(response.data.user);

setToken(response.data.token);

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

} catch (error) {

throw error;

}

};

const logout = () => {

setUser(null);

setToken(null);

localStorage.removeItem('token');

};

const value = {

user,

token,

login,

register,

logout

};

return (

<AuthContext.Provider value={value}>

{children}

</AuthContext.Provider>

);

}

// api/books.js

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api';

export const getBooks = async () => {

try {

const response = await axios.get(`${API\_URL}/books`);

return response.data;

} catch (error) {

throw error;

}

};

export const addBook = async (bookData, token) => {

try {

const response = await axios.post(`${API\_URL}/books`, bookData, {

headers: {

Authorization: `Bearer ${token}`

}

});

return response.data;

} catch (error) {

throw error;

}

};

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.

<?php

// config/database.php

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', 'your\_password');

define('DB\_NAME', 'login\_system');

$conn = mysqli\_connect(DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME);

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

// functions/auth.php

function registerUser($email, $password, $name) {

global $conn;

// Hash password

$hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

// Check if email already exists

$check\_query = "SELECT id FROM users WHERE email = ?";

$stmt = mysqli\_prepare($conn, $check\_query);

mysqli\_stmt\_bind\_param($stmt, "s", $email);

mysqli\_stmt\_execute($stmt);

$result = mysqli\_stmt\_get\_result($stmt);

if (mysqli\_num\_rows($result) > 0) {

return ['success' => false, 'message' => 'Email already exists'];

}

// Insert new user

$insert\_query = "INSERT INTO users (email, password, name) VALUES (?, ?, ?)";

$stmt = mysqli\_prepare($conn, $insert\_query);

mysqli\_stmt\_bind\_param($stmt, "sss", $email, $hashed\_password, $name);

if (mysqli\_stmt\_execute($stmt)) {

return ['success' => true, 'message' => 'Registration successful'];

} else {

return ['success' => false, 'message' => 'Registration failed'];

}

}

function loginUser($email, $password, $remember = false) {

global $conn;

$query = "SELECT \* FROM users WHERE email = ?";

$stmt = mysqli\_prepare($conn, $query);

mysqli\_stmt\_bind\_param($stmt, "s", $email);

mysqli\_stmt\_execute($stmt);

$result = mysqli\_stmt\_get\_result($stmt);

if ($user = mysqli\_fetch\_assoc($result)) {

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

// Start session

session\_start();

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['user\_email'] = $user['email'];

$\_SESSION['user\_name'] = $user['name'];

// Set remember me cookie if requested

if ($remember) {

$token = bin2hex(random\_bytes(32));

$expires = time() + (30 \* 24 \* 60 \* 60); // 30 days

// Store token in database

$update\_query = "UPDATE users SET remember\_token = ? WHERE id = ?";

$stmt = mysqli\_prepare($conn, $update\_query);

mysqli\_stmt\_bind\_param($stmt, "si", $token, $user['id']);

mysqli\_stmt\_execute($stmt);

// Set cookie

setcookie('remember\_token', $token, $expires, '/', '', true, true);

}

return ['success' => true, 'message' => 'Login successful'];

}

}

return ['success' => false, 'message' => 'Invalid email or password'];

}

function checkRememberMe() {

if (isset($\_COOKIE['remember\_token'])) {

global $conn;

$token = $\_COOKIE['remember\_token'];

$query = "SELECT \* FROM users WHERE remember\_token = ?";

$stmt = mysqli\_prepare($conn, $query);

mysqli\_stmt\_bind\_param($stmt, "s", $token);

mysqli\_stmt\_execute($stmt);

$result = mysqli\_stmt\_get\_result($stmt);

if ($user = mysqli\_fetch\_assoc($result)) {

session\_start();

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['user\_email'] = $user['email'];

$\_SESSION['user\_name'] = $user['name'];

return true;

}

}

return false;

}

function logout() {

session\_start();

session\_destroy();

setcookie('remember\_token', '', time() - 3600, '/');

header('Location: login.php');

exit();

}

// register.php

session\_start();

require\_once 'config/database.php';

require\_once 'functions/auth.php';

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

$result = registerUser($\_POST['email'], $\_POST['password'], $\_POST['name']);

if ($result['success']) {

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>Register</title>

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

</head>

<body>

<div class="container mt-5">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="card">

<div class="card-header">

<h3>Register</h3>

</div>

<div class="card-body">

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

<div class="mb-3">

<label for="name" class="form-label">Name</label>

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

</div>

<div class="mb-3">

<label for="email" class="form-label">Email</label>

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

</div>

<div class="mb-3">

<label for="password" class="form-label">Password</label>

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

</div>

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

</form>

<p class="mt-3">Already have an account? <a href="login.php">Login here</a></p>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

// login.php

<?php

session\_start();

require\_once 'config/database.php';

require\_once 'functions/auth.php';

// Check remember me cookie

if (!isset($\_SESSION['user\_id']) && checkRememberMe()) {

header('Location: dashboard.php');

exit();

}

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

$remember = isset($\_POST['remember']) ? true : false;

$result = loginUser($\_POST['email'], $\_POST['password'], $remember);

if ($result['success']) {

header('Location: 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>Login</title>

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

</head>

<body>

<div class="container mt-5">

<div class="row justify-content-center">

<div class="col-md-6">

<div class="card">

<div class="card-header">

<h3>Login</h3>

</div>

<div class="card-body">

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

<div class="mb-3">

<label for="email" class="form-label">Email</label>

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

</div>

<div class="mb-3">

<label for="password" class="form-label">Password</label>

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

</div>

<div class="mb-3 form-check">

<input type="checkbox" class="form-check-input" id="remember" name="remember">

<label class="form-check-label" for="remember">Remember me</label>

</div>

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

</form>

<p class="mt-3">Don't have an account? <a href="register.php">Register here</a></p>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

-- Create database

CREATE DATABASE login\_system;

USE login\_system;

-- Create users table

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

name VARCHAR(255) NOT NULL,

remember\_token VARCHAR(64),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Add indexes

CREATE INDEX idx\_email ON users(email);

CREATE INDEX idx\_remember\_token ON users(remember\_token);

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.sql

CREATE DATABASE attendance\_system;

USE attendance\_system;

CREATE TABLE students (

id INT PRIMARY KEY AUTO\_INCREMENT,

roll\_no VARCHAR(20) UNIQUE NOT NULL,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE attendance (

id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

date DATE NOT NULL,

status ENUM('present', 'absent') DEFAULT 'absent',

marked\_by VARCHAR(100),

FOREIGN KEY (student\_id) REFERENCES students(id)

);

CREATE TABLE teachers (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- config.php

<?php

$host = 'localhost';

$dbname = 'attendance\_system';

$username = 'root';

$password = '';

try {

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

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

} catch(PDOException $e) {

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

}

?>

-- student\_register.php

<?php

require\_once 'config.php';

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

$roll\_no = $\_POST['roll\_no'];

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

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

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

try {

$stmt = $pdo->prepare("INSERT INTO students (roll\_no, name, email, password) VALUES (?, ?, ?, ?)");

$stmt->execute([$roll\_no, $name, $email, $password]);

echo "Registration successful!";

} catch(PDOException $e) {

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

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Student Registration</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Student Registration</h2>

<form method="POST">

<div class="mb-3">

<label>Roll Number:</label>

<input type="text" name="roll\_no" class="form-control" required>

</div>

<div class="mb-3">

<label>Name:</label>

<input type="text" name="name" class="form-control" required>

</div>

<div class="mb-3">

<label>Email:</label>

<input type="email" name="email" class="form-control" required>

</div>

<div class="mb-3">

<label>Password:</label>

<input type="password" name="password" class="form-control" required>

</div>

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

</form>

</div>

</body>

</html>

-- mark\_attendance.php

<?php

require\_once 'config.php';

session\_start();

// Verify teacher is logged in

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

header('Location: teacher\_login.php');

exit();

}

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

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

$marked\_by = $\_SESSION['teacher\_name'];

foreach ($\_POST['attendance'] as $student\_id => $status) {

$stmt = $pdo->prepare("INSERT INTO attendance (student\_id, date, status, marked\_by)

VALUES (?, ?, ?, ?)

ON DUPLICATE KEY UPDATE status = ?");

$stmt->execute([$student\_id, $date, $status, $marked\_by, $status]);

}

echo "Attendance marked successfully!";

}

// Fetch all students

$stmt = $pdo->query("SELECT id, roll\_no, name FROM students ORDER BY roll\_no");

$students = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!DOCTYPE html>

<html>

<head>

<title>Mark Attendance</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Mark Attendance</h2>

<form method="POST">

<div class="mb-3">

<label>Date:</label>

<input type="date" name="date" class="form-control" required>

</div>

<table class="table">

<thead>

<tr>

<th>Roll No</th>

<th>Name</th>

<th>Attendance</th>

</tr>

</thead>

<tbody>

<?php foreach ($students as $student): ?>

<tr>

<td><?php echo htmlspecialchars($student['roll\_no']); ?></td>

<td><?php echo htmlspecialchars($student['name']); ?></td>

<td>

<select name="attendance[<?php echo $student['id']; ?>]" class="form-control">

<option value="present">Present</option>

<option value="absent">Absent</option>

</select>

</td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

<button type="submit" class="btn btn-primary">Submit Attendance</button>

</form>

</div>

</body>

</html>

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

-- database.sql

CREATE DATABASE farmer\_shop;

USE farmer\_shop;

CREATE TABLE farmers (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE products (

id INT PRIMARY KEY AUTO\_INCREMENT,

farmer\_id INT,

name VARCHAR(100) NOT NULL,

description TEXT,

price DECIMAL(10,2) NOT NULL,

quantity INT NOT NULL,

unit VARCHAR(20) NOT NULL,

image\_url VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (farmer\_id) REFERENCES farmers(id)

);

CREATE TABLE customers (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE orders (

id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

product\_id INT,

quantity INT NOT NULL,

total\_price DECIMAL(10,2) NOT NULL,

status ENUM('pending', 'confirmed', 'shipped', 'delivered') DEFAULT 'pending',

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (customer\_id) REFERENCES customers(id),

FOREIGN KEY (product\_id) REFERENCES products(id)

);

-- config.php

<?php

$host = 'localhost';

$dbname = 'farmer\_shop';

$username = 'root';

$password = '';

try {

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

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

} catch(PDOException $e) {

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

}

?>

-- add\_product.php

<?php

session\_start();

require\_once 'config.php';

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

header('Location: farmer\_login.php');

exit();

}

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

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

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

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

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

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

// Handle file upload

$image\_url = '';

if (isset($\_FILES['image'])) {

$target\_dir = "uploads/";

$image\_url = $target\_dir . basename($\_FILES["image"]["name"]);

move\_uploaded\_file($\_FILES["image"]["tmp\_name"], $image\_url);

}

try {

$stmt = $pdo->prepare("INSERT INTO products (farmer\_id, name, description, price, quantity, unit, image\_url)

VALUES (?, ?, ?, ?, ?, ?, ?)");

$stmt->execute([$\_SESSION['farmer\_id'], $name, $description, $price, $quantity, $unit, $image\_url]);

echo "Product added successfully!";

} catch(PDOException $e) {

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

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Add Product</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Add New Product</h2>

<form method="POST" enctype="multipart/form-data">

<div class="mb-3">

<label>Product Name:</label>

<input type="text" name="name" class="form-control" required>

</div>

<div class="mb-3">

<label>Description:</label>

<textarea name="description" class="form-control"></textarea>

</div>

<div class="mb-3">

<label>Price:</label>

<input type="number" name="price" step="0.01" class="form-control" required>

</div>

<div class="mb-3">

<label>Quantity:</label>

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

</div>

<div class="mb-3">

<label>Unit:</label>

<select name="unit" class="form-control">

<option value="kg">Kilogram</option>

<option value="piece">Piece</option>

<option value="dozen">Dozen</option>

</select>

</div>

<div class="mb-3">

<label>Image:</label>

<input type="file" name="image" class="form-control">

</div>

<button type="submit" class="btn btn-primary">Add Product</button>

</form>

</div>

</body>

</html>

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

<?php

// session\_manager.php

session\_start();

class SessionManager {

private $pdo;

private $maxSessions = 3;

private $sessionTimeout = 300; // 5 minutes in seconds

public function \_\_construct($pdo) {

$this->pdo = $pdo;

}

public function initSession($userId) {

// Clean expired sessions first

$this->cleanExpiredSessions($userId);

// Count active sessions

$stmt = $this->pdo->prepare("SELECT COUNT(\*) FROM user\_sessions WHERE user\_id = ? AND last\_activity > ?");

$stmt->execute([$userId, time() - $this->sessionTimeout]);

$activeSessions = $stmt->fetchColumn();

if ($activeSessions >= $this->maxSessions) {

return false; // Maximum sessions reached

}

// Create new session

$sessionId = session\_id();

$stmt = $this->pdo->prepare("INSERT INTO user\_sessions (user\_id, session\_id, last\_activity) VALUES (?, ?, ?)");

$stmt->execute([$userId, $sessionId, time()]);

$\_SESSION['user\_id'] = $userId;

$\_SESSION['last\_activity'] = time();

return true;

}

public function updateSession() {

if (isset($\_SESSION['user\_id'])) {

$sessionId = session\_id();

$stmt = $this->pdo->prepare("UPDATE user\_sessions SET last\_activity = ? WHERE session\_id = ?");

$stmt->execute([time(), $sessionId]);

$\_SESSION['last\_activity'] = time();

}

}

private function cleanExpiredSessions($userId) {

$stmt = $this->pdo->prepare("DELETE FROM user\_sessions WHERE user\_id = ? AND last\_activity <= ?");

$stmt->execute([$userId, time() - $this->sessionTimeout]);

}

public function isSessionValid() {

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

return false;

}

if (time() - $\_SESSION['last\_activity'] > $this->sessionTimeout) {

$this->destroySession();

return false;

}

$this->updateSession();

return true;

}

public function destroySession() {

$sessionId = session\_id();

$stmt = $this->pdo->prepare("DELETE FROM user\_sessions WHERE session\_id = ?");

$stmt->execute([$sessionId]);

session\_destroy();

}

}

// Database table creation

CREATE TABLE user\_sessions (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

session\_id VARCHAR(255) NOT NULL,

last\_activity INT NOT NULL,

INDEX (user\_id),

INDEX (session\_id)

);

// Usage example

require\_once 'config.php';

$sessionManager = new SessionManager($pdo);

// On login

if ($sessionManager->initSession($userId)) {

// Session created successfully

header('Location: dashboard.php');

} else {

echo "Maximum sessions reached. Please logout from other devices.";

}

// On each page load

if (!$sessionManager->isSessionValid()) {

header('Location: login.php');

exit();

}

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.

// Employee.java

package com.example.demo.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private String department;

private Double salary;

// Getters and Setters

public Long getId() { return id; }

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

public String getName() { return name; }

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

public String getEmail() { return email; }

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

public String getDepartment() { return department; }

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

public Double getSalary() { return salary; }

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

}

// EmployeeRepository.java

package com.example.demo.repository;

import com.example.demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

// EmployeeController.java

package com.example.demo.controller;

import com.example.demo.model.Employee;

import com.example.demo.repository.EmployeeRepository;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Employee not found"));

return ResponseEntity.ok(employee);

}

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Employee not found"));

employee.setName(employeeDetails.getName());

employee.setEmail(employeeDetails.getEmail());

employee.setDepartment(employeeDetails.getDepartment());

employee.setSalary(employeeDetails.getSalary());

Employee updatedEmployee = employeeRepository.save(employee);

return ResponseEntity.ok(updatedEmployee);

}

@DeleteMapping("/{id}")

public ResponseEntity<?> deleteEmployee(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Employee not found"));

employeeRepository.delete(employee);

return ResponseEntity.ok().build();

}

}

// application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/employee\_db

spring.datasource.username=root

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

// pom.xml dependencies

<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>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>