**1) Develop a website to fetch movie details using API integration. The front end must get the movie and year details from user and perform input validation using Java Script.**

movie-details-app/

│

├── public/ # Static assets (if needed)

│ └── style.css # CSS styles

│

├── src/

│ ├── index.html # Main HTML page

│ └── script.js # JavaScript for validation and fetch

│

├── server/ # (Optional backend)

│ ├── server.js # Node.js/Express server file

│ └── .env # To store API key securely

├── package.json *# Project dependencies*

**src/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Movie Details Finder</title>

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

</head>

<body>

<div class="container">

<h1>Fetch Movie Details</h1>

<form id="movieForm">

<input type="text" id="title" placeholder="Enter Movie Title" required>

<input type="number" id="year" placeholder="Enter Release Year" required>

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

</form>

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

</div>

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

</body>

</html>

**public/style.css (Simple Styling)**

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 50px;

}

.container {

width: 50%;

margin: auto;

}

input, button {

padding: 10px;

margin: 5px;

}

**src/script.js**

document.getElementById('movieForm').addEventListener('submit', async function (e) {

e.preventDefault();

const title = document.getElementById('title').value.trim();

const year = document.getElementById('year').value.trim();

const resultDiv = document.getElementById('result');

// Input Validation

if (title === '' || year === '') {

alert("Please fill out both fields.");

return;

}

if (isNaN(year) || year.length !== 4 || year < 1900 || year > new Date().getFullYear()) {

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

return;

}

// API Call

try {

const response = await fetch(`/api/movie?title=${encodeURIComponent(title)}&year=${year}`);

const data = await response.json();

if (data.Response === "False") {

resultDiv.innerHTML = `<p>${data.Error}</p>`;

} else {

resultDiv.innerHTML = `

<h2>${data.Title} (${data.Year})</h2>

<img src="${data.Poster}" alt="Poster" height="300">

<p><strong>Genre:</strong> ${data.Genre}</p>

<p><strong>Plot:</strong> ${data.Plot}</p>

<p><strong>IMDB Rating:</strong> ${data.imdbRating}</p>

`;

}

} catch (error) {

resultDiv.innerHTML = `<p>Error fetching movie details.</p>`;

}

});

**Dependencies**

npm init -y

npm install express dotenv node-fetch cors

**server/server.js**

require('dotenv').config();

const express = require('express');

const fetch = require('node-fetch');

const cors = require('cors');

const app = express();

const PORT = 3000;

app.use(cors());

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

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

const { title, year } = req.query;

const apiKey = process.env.OMDB\_API\_KEY;

const apiUrl = `http://www.omdbapi.com/?t=${title}&y=${year}&apikey=${apiKey}`;

try {

const response = await fetch(apiUrl);

const data = await response.json();

res.json(data);

} catch (error) {

res.status(500).json({ error: 'Server error' });

}

});

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

**.env file**

OMDB\_API\_KEY=https://www.omdbapi.com/apikey.aspx

**2. Develop a website to integrate API for fetching random jokes and insert them into MongoDB**

**collection using Webserver.**

random-joke-app/

│

├── public/ *# Frontend files*

│ ├── index.html

│ ├── style.css

│ └── script.js

│

├── server/ *# Backend code*

│ ├── server.js

│ └── .env *# Environment variables (Mongo URI)*

│

├── package.json

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Random Joke Fetcher</title>

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

</head>

<body>

<div class="container">

<h1>Random Joke Generator</h1>

<button id="fetchJoke">Get a Joke</button>

<div id="jokeBox"></div>

</div>

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

</body>

</html>

**style.css**

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 100px;

}

button {

padding: 10px 20px;

margin-top: 20px;

}

#jokeBox {

margin-top: 30px;

font-size: 1.2em;

}

**script.js**

document.getElementById('fetchJoke').addEventListener('click', async () => {

const response = await fetch('/get-joke');

const data = await response.json();

const box = document.getElementById('jokeBox');

if (data.setup && data.punchline) {

box.innerHTML = `<p><strong>Setup:</strong> ${data.setup}</p><p><strong>Punchline:</strong> ${data.punchline}</p>`;

} else {

box.innerHTML = `<p>Error fetching joke.</p>`;

}

});

**Dependencies to Install**

npm init -y

npm install express mongodb dotenv node-fetch

**.env**

MONGO\_URI=your\_mongodb\_connection\_string

**server.js**

require('dotenv').config();

const express = require('express');

const { MongoClient } = require('mongodb');

const fetch = require('node-fetch');

const path = require('path');

const app = express();

const PORT = 3000;

const client = new MongoClient(process.env.MONGO\_URI);

const dbName = "jokesDB";

app.use(express.static(path.join(\_\_dirname, '../public')));

// Route to fetch and store a joke

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

try {

const response = await fetch('https://official-joke-api.appspot.com/random\_joke');

const joke = await response.json();

await client.connect();

const db = client.db(dbName);

const jokesCollection = db.collection('jokes');

await jokesCollection.insertOne(joke);

res.json(joke);

} catch (err) {

console.error(err);

res.status(500).json({ error: "Failed to fetch or store joke" });

}

});

app.listen(PORT, () => {

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

});

**3. Develop a student registration form with good input validation using Java script and use HTTP**

**POST method to send those details to the webserver and print the details in console.**

student-registration/

│

├── public/ *# Frontend files*

│ ├── index.html

│ ├── style.css

│ └── script.js

│

├── server/ *# Backend code*

│ ├── server.js

│ └── .env *# (Optional, if needed)*

│

├── package.json

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Student Registration Form</title>

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

</head>

<body>

<div class="form-container">

<h2>Student Registration</h2>

<form id="studentForm">

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

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

<input type="number" id="age" placeholder="Age" required>

<input type="text" id="course" placeholder="Course" required>

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

</form>

<p id="message"></p>

</div>

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

</body>

</html>

**style.css**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

padding-top: 80px;

}

.form-container {

background: white;

padding: 30px;

border-radius: 8px;

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

input, button {

display: block;

margin: 10px auto;

padding: 10px;

width: 80%;

}

button {

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

}

**script.js**

document.getElementById('studentForm').addEventListener('submit', async function (e) {

e.preventDefault();

const name = document.getElementById('name').value.trim();

const email = document.getElementById('email').value.trim();

const age = parseInt(document.getElementById('age').value.trim());

const course = document.getElementById('course').value.trim();

const message = document.getElementById('message');

// Input validation

if (!name || !email || !age || !course) {

message.textContent = "All fields are required.";

return;

}

if (age < 10 || age > 100) {

message.textContent = "Enter a valid age (10 - 100).";

return;

}

// Send to server using POST

try {

const res = await fetch('/register', {

method: 'POST',

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

body: JSON.stringify({ name, email, age, course })

});

const result = await res.json();

message.textContent = result.message;

} catch (err) {

message.textContent = "Submission failed!";

}

});

**Dependencies to Install**

npm init -y

npm install express body-parser cors dotenv

**server.js**

const express = require('express');

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

const cors = require('cors');

const path = require('path');

const app = express();

const PORT = 3000;

app.use(cors());

app.use(bodyParser.json());

app.use(express.static(path.join(\_\_dirname, '../public')));

// POST route to receive registration data

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

const { name, email, age, course } = req.body;

console.log("📌 Student Registered:");

console.log(`Name: ${name}`);

console.log(`Email: ${email}`);

console.log(`Age: ${age}`);

console.log(`Course: ${course}`);

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

});

app.listen(PORT, () => {

console.log(`✅ Server running at http://localhost:${PORT}`);

});

**4. Design a front-end page for our college using React with minimum four routing components to**

**display different departments. Also use hook to count the number of visitors for the web page.**

college-website/

│

├── public/

│ └── index.html

│

├── src/

│ ├── components/

│ │ ├── Home.jsx

│ │ ├── CSE.jsx

│ │ ├── ECE.jsx

│ │ ├── MECH.jsx

│ │ └── CIVIL.jsx

│ │

│ ├── App.jsx

│ ├── VisitorCounter.jsx

│ └── main.jsx

│

├── package.json

└── vite.config.js

Create your React app using Vite:

npm create vite@latest college-website -- --template react

cd college-website

Install dependencies:

npm install react-router-dom

App.jsx

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

import Home from "./components/Home";

import CSE from "./components/CSE";

import ECE from "./components/ECE";

import MECH from "./components/MECH";

import CIVIL from "./components/CIVIL";

import VisitorCounter from "./VisitorCounter";

export default function App() {

return (

<Router>

<div className="p-4 text-center">

<h1 className="text-2xl font-bold">Welcome to Our College</h1>

<VisitorCounter />

<nav className="space-x-4 mt-4">

<Link to="/">Home</Link>

<Link to="/cse">CSE</Link>

<Link to="/ece">ECE</Link>

<Link to="/mech">MECH</Link>

<Link to="/civil">CIVIL</Link>

</nav>

<Routes>

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

<Route path="/cse" element={<CSE />} />

<Route path="/ece" element={<ECE />} />

<Route path="/mech" element={<MECH />} />

<Route path="/civil" element={<CIVIL />} />

</Routes>

</div>

</Router>

);

}

VisitorCounter.jsx

import { useEffect, useState } from "react";

export default function VisitorCounter() {

const [visits, setVisits] = useState(0);

useEffect(() => {

setVisits(prev => prev + 1);

}, []);

return (

<p className="text-sm text-gray-500 mt-2">Visitor Count (this session): {visits}</p>

);

}

**Sample for Department Component (e.g.,**CSE.jsx**)**

export default function CSE() {

return (

<div className="mt-5">

<h2 className="text-xl font-semibold">Department of Computer Science</h2>

<p>Welcome to the CSE department. Learn cutting-edge technologies here!</p>

</div>

);

}

Do similarly for ECE.jsx, MECH.jsx, and CIVIL.jsx.

Home.jsx

export default function Home() {

return (

<div className="mt-5">

<h2 className="text-xl font-semibold">Home Page</h2>

<p>This is our college website. Navigate to different departments to learn more.</p>

</div>

);

}

main.jsx

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

import "./index.css";

ReactDOM.createRoot(document.getElementById("root")).render(

<React.StrictMode>

<App />

</React.StrictMode>

);

**5. Design a web application to get the employee details and web server to perform CRUD operations in the MongoDB.**

employee-crud-app/

│

├── client/ *# Frontend*

│ ├── index.html

│ ├── style.css

│ └── script.js

│

├── server/ *# Backend*

│ ├── server.js

│ ├── db.js

│ └── models/

│ └── Employee.js

│

├── .env *# MongoDB URI*

├── package.json

**Dependencies to Install**

npm init -y

npm install express mongoose cors dotenv body-parser

**.env (in root)**

MONGO\_URI=mongodb://127.0.0.1:27017/employees

PORT=5000

**db.js (inside server/)**

const mongoose = require('mongoose');

require('dotenv').config();

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then(() => console.log("✅ MongoDB connected"))

.catch(err => console.error("❌ DB connection error:", err));

models/Employee.js

const mongoose = require('mongoose');

const employeeSchema = new mongoose.Schema({

name: String,

email: String,

position: String,

salary: Number,

});

module.exports = mongoose.model('Employee', employeeSchema);

**server.js**

const express = require('express');

const cors = require('cors');

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

require('dotenv').config();

require('./db');

const Employee = require('./models/Employee');

const app = express();

app.use(cors());

app.use(bodyParser.json());

// Create

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

const emp = new Employee(req.body);

await emp.save();

res.json({ message: "Employee added" });

});

// Read

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

const employees = await Employee.find();

res.json(employees);

});

// Update

app.put('/employee/:id', async (req, res) => {

await Employee.findByIdAndUpdate(req.params.id, req.body);

res.json({ message: "Employee updated" });

});

// Delete

app.delete('/employee/:id', async (req, res) => {

await Employee.findByIdAndDelete(req.params.id);

res.json({ message: "Employee deleted" });

});

app.listen(process.env.PORT, () => {

console.log(`🚀 Server running on http://localhost:${process.env.PORT}`);

});

**Frontend Setup (**client/**)**

**index.html**

<!DOCTYPE html>

<html>

<head>

<title>Employee CRUD</title>

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

</head>

<body>

<h2>Employee Management</h2>

<form id="employeeForm">

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

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

<input type="text" id="position" placeholder="Position" required>

<input type="number" id="salary" placeholder="Salary" required>

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

</form>

<div id="employeeList"></div>

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

</body>

</html>

**style.css**

body {

font-family: Arial;

padding: 20px;

}

input, button {

margin: 5px;

padding: 8px;

}

#employeeList {

margin-top: 20px;

}

**script.js**

const form = document.getElementById('employeeForm');

const list = document.getElementById('employeeList');

const apiUrl = "http://localhost:5000";

form.addEventListener('submit', async e => {

e.preventDefault();

const employee = {

name: form.name.value,

email: form.email.value,

position: form.position.value,

salary: form.salary.value

};

await fetch(`${apiUrl}/employee`, {

method: 'POST',

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

body: JSON.stringify(employee)

});

form.reset();

loadEmployees();

});

async function loadEmployees() {

const res = await fetch(`${apiUrl}/employees`);

const data = await res.json();

list.innerHTML = "";

data.forEach(emp => {

list.innerHTML += `

<div>

<p><strong>${emp.name}</strong> (${emp.position}) - ₹${emp.salary}</p>

<button onclick="deleteEmployee('${emp.\_id}')">Delete</button>

</div>

`;

});

}

async function deleteEmployee(id) {

await fetch(`${apiUrl}/employee/${id}`, { method: 'DELETE' });

loadEmployees();

}

loadEmployees();

**6. Design a web application to store and retrieve student details from the client web page to the**

**MongoDB using NodeJS.**

student-mongo-app/

│

├── client/

│ ├── index.html

│ ├── style.css

│ └── script.js

│

├── server/

│ ├── server.js

│ ├── db.js

│ └── models/

│ └── Student.js

│

├── .env

├── package.json

**Backend Setup**

**Install Dependencies**

npm init -y

npm install express mongoose cors dotenv body-parser

**.env (in root)**

PORT=5000

MONGO\_URI=mongodb://127.0.0.1:27017/studentDB

**db.js (inside server/)**

const mongoose = require('mongoose');

require('dotenv').config();

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then(() => console.log("✅ MongoDB connected"))

.catch(err => console.error("❌ MongoDB error:", err));

models/Student.js

const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({

name: String,

roll: String,

department: String,

email: String

});

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

**server.js (inside server/)**

const express = require('express');

const cors = require('cors');

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

require('dotenv').config();

require('./db');

const Student = require('./models/Student');

const app = express();

app.use(cors());

app.use(bodyParser.json());

// Add student

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

const newStudent = new Student(req.body);

await newStudent.save();

res.json({ message: "Student added successfully" });

});

// Get all students

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

const students = await Student.find();

res.json(students);

});

app.listen(process.env.PORT, () => {

console.log(`🚀 Server running on http://localhost:${process.env.PORT}`);

});

**Frontend Setup (client/)**

**index.html**

<!DOCTYPE html>

<html>

<head>

<title>Student Form</title>

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

</head>

<body>

<h2>Student Registration</h2>

<form id="studentForm">

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

<input type="text" id="roll" placeholder="Roll No" required>

<input type="text" id="department" placeholder="Department" required>

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

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

</form>

<h3>Student List</h3>

<div id="studentList"></div>

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

</body>

</html>

**style.css**

body {

font-family: Arial, sans-serif;

padding: 20px;

}

input, button {

display: block;

margin: 10px 0;

padding: 8px;

}

#studentList div {

border: 1px solid #ccc;

padding: 8px;

margin-top: 10px;

}

**script.js**

const form = document.getElementById('studentForm');

const studentList = document.getElementById('studentList');

const apiUrl = 'http://localhost:5000';

form.addEventListener('submit', async (e) => {

e.preventDefault();

const student = {

name: form.name.value,

roll: form.roll.value,

department: form.department.value,

email: form.email.value

};

await fetch(`${apiUrl}/student`, {

method: 'POST',

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

body: JSON.stringify(student)

});

form.reset();

fetchStudents();

});

async function fetchStudents() {

const res = await fetch(`${apiUrl}/students`);

const students = await res.json();

studentList.innerHTML = '';

students.forEach(s => {

studentList.innerHTML += `

<div>

<strong>${s.name}</strong> (${s.roll}) - ${s.department} - ${s.email}

</div>

`;

});

}

fetchStudents();

**7. Develop a website to integrate weather API to display the current weather of the location. The client page has to perform input validation for the location using Java Script.**

weather-app/

│

├── index.html

├── style.css

└── script.js

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Weather App</title>

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

</head>

<body>

<h1>Current Weather</h1>

<form id="weatherForm">

<input type="text" id="locationInput" placeholder="Enter city name" required>

<button type="submit">Get Weather</button>

</form>

<div id="weatherResult"></div>

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

</body>

</html>

**style.css**

body {

font-family: Arial, sans-serif;

text-align: center;

padding: 30px;

}

input, button {

padding: 10px;

margin: 10px;

}

#weatherResult {

margin-top: 20px;

font-size: 18px;

}

**script.js**

const form = document.getElementById('weatherForm');

const input = document.getElementById('locationInput');

const resultDiv = document.getElementById('weatherResult');

const apiKey = 'https://openweathermap.org/api'; // Replace with your actual API key

form.addEventListener('submit', async (e) => {

e.preventDefault();

const location = input.value.trim();

// Input validation

if (location === '' || !/^[a-zA-Z\s]+$/.test(location)) {

alert('Please enter a valid city name using only letters.');

return;

}

resultDiv.textContent = "Fetching weather...";

try {

const response = await fetch(

`https://api.openweathermap.org/data/2.5/weather?q=${location}&appid=${apiKey}&units=metric`

);

if (!response.ok) {

throw new Error('City not found');

}

const data = await response.json();

resultDiv.innerHTML = `

<strong>${data.name}</strong><br>

Temperature: ${data.main.temp}°C<br>

Weather: ${data.weather[0].description}<br>

Humidity: ${data.main.humidity}%

`;

} catch (err) {

resultDiv.textContent = err.message;

}

});

**8. Develop a website to integrate Animal facts API to display facts about any animal provided by user. The client page has to perform input validation for the animal’s name and also show error if no such animal name exists using Java Script.**

animal-facts-app/

│

├── index.html

├── style.css

└── script.js

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Animal Facts</title>

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

</head>

<body>

<h1>Animal Facts</h1>

<form id="animalForm">

<input type="text" id="animalInput" placeholder="Enter animal name (e.g., cat)" required>

<button type="submit">Get Facts</button>

</form>

<div id="factsResult"></div>

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

</body>

</html>

**style.css**

body {

font-family: Arial, sans-serif;

text-align: center;

padding: 30px;

}

input, button {

padding: 10px;

margin: 10px;

}

#factsResult {

margin-top: 20px;

font-size: 18px;

color: #333;

}

.error {

color: red;

font-weight: bold;

}

**script.js**

const form = document.getElementById('animalForm');

const input = document.getElementById('animalInput');

const resultDiv = document.getElementById('factsResult');

form.addEventListener('submit', async (e) => {

e.preventDefault();

const animal = input.value.trim().toLowerCase();

// Input validation: only allow letters and spaces

if (animal === '' || !/^[a-zA-Z\s]+$/.test(animal)) {

resultDiv.innerHTML = '<p class="error">Please enter a valid animal name.</p>';

return;

}

resultDiv.textContent = "Fetching facts...";

try {

const response = await fetch(`https://animal-facts-api.herokuapp.com/api/v1/${animal}`);

if (!response.ok) {

throw new Error(`No facts found for "${animal}"`);

}

const data = await response.json();

if (data.length === 0) {

throw new Error(`No facts found for "${animal}"`);

}

// Display the facts

resultDiv.innerHTML = `

<strong>Facts about ${animal.charAt(0).toUpperCase() + animal.slice(1)}:</strong><br>

${data.map(fact => `<p>${fact.text}</p>`).join('')}

`;

} catch (err) {

resultDiv.innerHTML = `<p class="error">${err.message}</p>`;

}

});

**9. Develop a React front-end page that fetches details of books and animals by integrating respective APIs in different components using routing.**

**Install Dependencies:**

npx create-react-app react-api-app

cd react-api-app

npm install react-router-dom

react-api-app/

│

├── public/

│ └── index.html

│

├── src/

│ ├── components/

│ │ ├── BookDetails.jsx

│ │ ├── AnimalDetails.jsx

│ │ └── Navbar.jsx

│ ├── App.jsx

│ ├── index.js

│ └── style.css

│

├── package.json

**public/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Book and Animal API</title>

</head>

<body>

<div id="root"></div>

</body>

</html>

**src/style.css**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

.container {

width: 80%;

margin: auto;

}

button {

padding: 10px 20px;

font-size: 16px;

margin: 10px;

}

ul {

list-style-type: none;

padding: 0;

}

li {

padding: 5px;

background-color: #fff;

margin: 5px 0;

}

h1 {

text-align: center;

padding: 20px;

color: #333;

}

**src/components/Navbar.jsx**

import React from 'react';

import { Link } from 'react-router-dom';

const Navbar = () => {

return (

<nav>

<ul>

<li><Link to="/books">Books</Link></li>

<li><Link to="/animals">Animals</Link></li>

</ul>

</nav>

);

};

export default Navbar;

**src/components/BookDetails.jsx**

import React, { useState } from 'react';

const BookDetails = () => {

const [books, setBooks] = useState([]);

const [error, setError] = useState(null);

const fetchBooks = async () => {

try {

const response = await fetch('https://openlibrary.org/subjects/science.json');

const data = await response.json();

setBooks(data.works);

} catch (err) {

setError('Failed to fetch book data');

}

};

return (

<div className="container">

<h1>Books</h1>

<button onClick={fetchBooks}>Fetch Books</button>

{error && <p>{error}</p>}

<ul>

{books.map((book) => (

<li key={book.key}>

<strong>{book.title}</strong><br />

Author: {book.authors ? book.authors[0].name : 'Unknown'}

</li>

))}

</ul>

</div>

);

};

export default BookDetails;

**src/components/AnimalDetails.jsx**

import React, { useState } from 'react';

const AnimalDetails = () => {

const [animal, setAnimal] = useState('');

const [animalInfo, setAnimalInfo] = useState(null);

const [error, setError] = useState(null);

const fetchAnimal = async () => {

try {

const response = await fetch(`https://animal-facts-api.herokuapp.com/api/v1/${animal}`);

if (!response.ok) throw new Error('Animal not found');

const data = await response.json();

setAnimalInfo(data);

} catch (err) {

setError('Failed to fetch animal data');

}

};

return (

<div className="container">

<h1>Animal Facts</h1>

<input

type="text"

placeholder="Enter animal name"

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

/>

<button onClick={fetchAnimal}>Fetch Animal Facts</button>

{error && <p>{error}</p>}

{animalInfo && (

<div>

<h2>{animalInfo.name}</h2>

<ul>

{animalInfo.map((fact, index) => (

<li key={index}>{fact.text}</li>

))}

</ul>

</div>

)}

</div>

);

};

export default AnimalDetails;

**src/App.jsx**

import React from 'react';

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

import BookDetails from './components/BookDetails';

import AnimalDetails from './components/AnimalDetails';

import Navbar from './components/Navbar';

import './style.css';

const App = () => {

return (

<Router>

<Navbar />

<Routes>

<Route path="/books" element={<BookDetails />} />

<Route path="/animals" element={<AnimalDetails />} />

</Routes>

</Router>

);

};

export default App;

**src/index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));

**10. Develop a web application to perform CRUD operations on MongoDB using NodeJS with details**

**shown or got from the front-end client using HTTP GET and POST.**

mongodb-crud-app/

│

├── client/ # Front-end (HTML + JS)

│ ├── index.html # Form to Add/Display Users

│ └── script.js # JavaScript for GET and POST

│

├── server/ # Backend

│ ├── server.js # Main Node.js server

│ └── db.js # MongoDB connection file

│

├── package.json

└── .env # MongoDB URI

**client/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>CRUD with MongoDB</title>

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

</head>

<body>

<h2>User Registration</h2>

<form id="userForm">

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

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

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

</form>

<h2>All Users</h2>

<ul id="userList"></ul>

</body>

</html>

**client/script.js**

document.getElementById('userForm').addEventListener('submit', async (e) => {

e.preventDefault();

const name = document.getElementById('name').value.trim();

const email = document.getElementById('email').value.trim();

const response = await fetch('http://localhost:3000/api/users', {

method: 'POST',

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

body: JSON.stringify({ name, email })

});

if (response.ok) {

alert('User added!');

loadUsers();

}

});

async function loadUsers() {

const response = await fetch('http://localhost:3000/api/users');

const users = await response.json();

const list = document.getElementById('userList');

list.innerHTML = '';

users.forEach(user => {

const li = document.createElement('li');

li.textContent = `${user.name} - ${user.email}`;

list.appendChild(li);

});

}

window.onload = loadUsers;

**server/server.js**

const express = require('express');

const cors = require('cors');

const mongoose = require('mongoose');

require('dotenv').config();

const app = express();

app.use(cors());

app.use(express.json());

const User = require('./userModel');

require('./db');

// GET all users

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

const users = await User.find();

res.json(users);

});

// POST a new user

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

const { name, email } = req.body;

const user = new User({ name, email });

await user.save();

res.status(201).json({ message: 'User created' });

});

app.listen(3000, () => console.log('Server running on <http://localhost:3000')>);

server/userModel.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

name: String,

email: String

});

module.exports = mongoose.model('User', userSchema);

**server/db.js**

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => console.log('MongoDB connected'))

.catch(err => console.error(err));

**.env File**

MONGO\_URI=mongodb://127.0.0.1:27017/crudDB

**Backend Dependencies**

npm init -y

npm install express mongoose cors dotenv  
  
  
  
**11. Design an online shopping for toys using React components and routing, perform online purchase by adding order details to the MongoDB.**

online-toy-shop/

│

├── client/ *# React Frontend*

│ ├── public/

│ └── src/

│ ├── components/

│ │ ├── Home.jsx

│ │ ├── ToyList.jsx

│ │ ├── ToyDetails.jsx

│ │ ├── Cart.jsx

│ │ └── Checkout.jsx

│ ├── App.jsx

│ └── main.jsx

│

├── server/ *# Node.js Backend*

│ ├── server.js

│ ├── db.js

│ └── orderModel.js

│

├── package.json

├── client/package.json

└── .env *# Mongo URI*

**Backend Setup (Node.js + Express + MongoDB)**

**server/db.js**

const mongoose = require('mongoose');

require('dotenv').config();

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => console.log('MongoDB Connected'));

server/orderModel.js

const mongoose = require('mongoose');

const orderSchema = new mongoose.Schema({

customerName: String,

items: Array,

total: Number,

address: String,

});

module.exports = mongoose.model('Order', orderSchema);

server/server.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const Order = require('./orderModel');

require('./db');

const app = express();

app.use(cors());

app.use(express.json());

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

const newOrder = new Order(req.body);

await newOrder.save();

res.status(201).send({ message: 'Order placed successfully!' });

});

app.listen(5000, () => console.log('Server running at <http://localhost:5000')>);

.env

MONGO\_URI=mongodb://127.0.0.1:27017/toyshop

**Install Dependencies (in**server/**)**

npm init -y

npm install express mongoose cors dotenv

**Frontend Setup (React + React Router + Axios)**

**Inside client/:**

npm create vite@latest .

npm install react-router-dom axios

**ToyList.jsx**

import { Link } from 'react-router-dom';

const toys = [

{ id: 1, name: "Teddy Bear", price: 500 },

{ id: 2, name: "Lego Set", price: 1200 }

];

export default function ToyList() {

return (

<div>

<h2>Available Toys</h2>

<ul>

{toys.map(toy => (

<li key={toy.id}>

<Link to={`/toys/${toy.id}`}>{toy.name} - ₹{toy.price}</Link>

</li>

))}

</ul>

</div>

);

}

**Checkout.jsx**

import axios from 'axios';

export default function Checkout() {

const order = {

customerName: "Sample User",

address: "123 Main St",

total: 1700,

items: [{ name: "Teddy Bear", price: 500 }, { name: "Lego Set", price: 1200 }]

};

const placeOrder = async () => {

await axios.post('http://localhost:5000/api/orders', order);

alert("Order placed successfully!");

};

return (

<div>

<h2>Checkout</h2>

<button onClick={placeOrder}>Place Order</button>

</div>

);

}

**App.jsx**

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

import Home from './components/Home';

import ToyList from './components/ToyList';

import ToyDetails from './components/ToyDetails';

import Cart from './components/Cart';

import Checkout from './components/Checkout';

function App() {

return (

<BrowserRouter>

<Routes>

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

<Route path="/toys" element={<ToyList />} />

<Route path="/toys/:id" element={<ToyDetails />} />

<Route path="/cart" element={<Cart />} />

<Route path="/checkout" element={<Checkout />} />

</Routes>

</BrowserRouter>

);

}

export default App;

**12. Develop a web application that gets student roll number and date of birth to fetch exam result from backend using HTTP GET.**

student-result-app/

│

├── server/ # Backend

│ ├── server.js

│ └── studentData.js

│

├── client/ # Frontend

│ ├── index.html

│ └── script.js

│

└── package.json

**Backend – Node.js + Express**

**server/studentData.js**

module.exports = [

{ roll: "1001", dob: "2005-04-15", name: "Ravi", marks: 87 },

{ roll: "1002", dob: "2005-08-10", name: "Divya", marks: 92 },

{ roll: "1003", dob: "2005-11-20", name: "Karan", marks: 76 },

];

**server/server.js**

const express = require('express');

const cors = require('cors');

const data = require('./studentData');

const app = express();

app.use(cors());

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

const { roll, dob } = req.query;

const student = data.find(s => s.roll === roll && s.dob === dob);

if (student) {

res.json({ status: 'success', result: student });

} else {

res.status(404).json({ status: 'fail', message: 'Student not found or DOB incorrect' });

}

});

app.listen(5000, () => console.log("Server running at http://localhost:5000"));

**Install Dependencies**

npm init -y

npm install express cors

**Frontend – HTML + JavaScript**

**client/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Exam Result</title>

</head>

<body>

<h2>Check Exam Result</h2>

<form id="resultForm">

<label>Roll Number: <input type="text" id="roll" required></label><br>

<label>Date of Birth: <input type="date" id="dob" required></label><br>

<button type="submit">Get Result</button>

</form>

<div id="output"></div>

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

</body>

</html>

**client/script.js**

document.getElementById("resultForm").addEventListener("submit", async function (e) {

e.preventDefault();

const roll = document.getElementById("roll").value.trim();

const dob = document.getElementById("dob").value;

if (!roll || !dob) {

alert("Please enter both fields.");

return;

}

const response = await fetch(`http://localhost:5000/api/result?roll=${roll}&dob=${dob}`);

const output = document.getElementById("output");

if (response.ok) {

const data = await response.json();

output.innerHTML = `<h3>Result for ${data.result.name}</h3><p>Marks: ${data.result.marks}</p>`;

} else {

const err = await response.json();

output.innerHTML = `<p style="color:red;">${err.message}</p>`;

}

});

**13. Develop a web application that gets user account number from the front-end and uses HTTP POST to retrieve the account details from MongoDB.**

account-details-app/

│

├── server/ # Backend folder

│ ├── server.js

│ └── model.js

│

├── client/ # Frontend folder

│ ├── index.html

│ └── script.js

│

└── package.json

**Backend Setup (Node.js + Express + MongoDB)**

**Dependencies**

npm init -y

npm install express mongoose cors body-parser

**server/model.js**

const mongoose = require('mongoose');

const accountSchema = new mongoose.Schema({

accNumber: String,

name: String,

balance: Number

});

module.exports = mongoose.model('Account', accountSchema);

**server/server.js**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

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

const Account = require('./model');

const app = express();

app.use(cors());

app.use(bodyParser.json());

// MongoDB connection

mongoose.connect('mongodb://127.0.0.1:27017/bank', {

useNewUrlParser: true,

useUnifiedTopology: true,

}).then(() => console.log("MongoDB connected"))

.catch(err => console.log("MongoDB Error", err));

// POST route to retrieve account details

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

const { accNumber } = req.body;

try {

const account = await Account.findOne({ accNumber });

if (account) {

res.json({ status: 'success', account });

} else {

res.status(404).json({ status: 'fail', message: 'Account not found' });

}

} catch (err) {

res.status(500).json({ status: 'error', message: err.message });

}

});

app.listen(5000, () => console.log("Server running at [http://localhost:5000")](http://localhost:5000%22)));

**client/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Account Lookup</title>

</head>

<body>

<h2>Enter Account Number</h2>

<form id="accountForm">

<input type="text" id="accNumber" required>

<button type="submit">Get Details</button>

</form>

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

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

</body>

</html>  
**client/script.js**

document.getElementById("accountForm").addEventListener("submit", async function (e) {

e.preventDefault();

const accNumber = document.getElementById("accNumber").value.trim();

if (!accNumber) {

alert("Please enter account number.");

return;

}

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

method: 'POST',

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

body: JSON.stringify({ accNumber })

});

const resultDiv = document.getElementById("result");

if (response.ok) {

const data = await response.json();

resultDiv.innerHTML = `<p><strong>Name:</strong> ${data.account.name}<br><strong>Balance:</strong> ₹${data.account.balance}</p>`;

} else {

const error = await response.json();

resultDiv.innerHTML = `<p style="color:red">${error.message}</p>`;

}

});

**14. Develop a web site for college cultural festival using minimum five react components and perform any conditional rendering on the website based on user choice.**

college-fest-website/

│

├── public/

│ └── index.html

│

├── src/

│ ├── components/

│ │ ├── Navbar.jsx

│ │ ├── Home.jsx

│ │ ├── Events.jsx

│ │ ├── Gallery.jsx

│ │ ├── Contact.jsx

│ │ └── Registration.jsx

│ │

│ ├── App.jsx

│ └── main.jsx

│

├── package.json

└── vite.config.js

**Create Project with Vite**:

npm create vite@latest college-fest-website --template react

cd college-fest-website

npm install

**Install Routing**:

npm install react-router-dom

**main.jsx**

import React from 'react';

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import App from './App';

ReactDOM.createRoot(document.getElementById('root')).render(

<BrowserRouter>

<App />

</BrowserRouter>

);

**App.jsx**

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

import Navbar from './components/Navbar';

import Home from './components/Home';

import Events from './components/Events';

import Gallery from './components/Gallery';

import Registration from './components/Registration';

import Contact from './components/Contact';

function App() {

return (

<>

<Navbar />

<Routes>

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

<Route path="/events" element={<Events />} />

<Route path="/gallery" element={<Gallery />} />

<Route path="/register" element={<Registration />} />

<Route path="/contact" element={<Contact />} />

</Routes>

</>

);

}

export default App;

**Navbar.jsx**

import { Link } from 'react-router-dom';

export default function Navbar() {

return (

<nav>

<Link to="/">Home</Link> |

<Link to="/events">Events</Link> |

<Link to="/gallery">Gallery</Link> |

<Link to="/register">Register</Link> |

<Link to="/contact">Contact</Link>

</nav>

);

}

**Events.jsx**

import { useState } from 'react';

export default function Events() {

const [type, setType] = useState('all');

const technical = ['Coding Contest', 'Tech Quiz', 'Debugging'];

const nonTechnical = ['Dance', 'Singing', 'Drama'];

const filteredEvents =

type === 'technical'

? technical

: type === 'non-technical'

? nonTechnical

: [...technical, ...nonTechnical];

return (

<div>

<h2>Event Categories</h2>

<select onChange={(e) => setType(e.target.value)}>

<option value="all">All</option>

<option value="technical">Technical</option>

<option value="non-technical">Non-Technical</option>

</select>

<ul>

{filteredEvents.map((event, index) => (

<li key={index}>{event}</li>

))}

</ul>

</div>

);

}

**Home.jsx, Gallery.jsx, Contact.jsx, Registration.jsx**

export default function Home() {

return (

<div>

<h1>Welcome to Our College Cultural Festival</h1>

<p>Join us for a day full of fun, talent, and creativity!</p>

</div>

);

}

**Repeat similar structure for other components.**

**15. Design a web application for online books purchase. The web page must have details of books**

**fetched from API and options to add books to cart.**

online-book-store/

│

├── public/

│ └── index.html

│

├── src/

│ ├── components/

│ │ ├── Navbar.jsx

│ │ ├── BookList.jsx

│ │ ├── Cart.jsx

│ │ └── BookCard.jsx

│ │

│ ├── App.jsx

│ ├── main.jsx

│ └── context/

│ └── CartContext.jsx

│

├── package.json

└── vite.config.js

**Dependencies**

**Install with Vite**

npm create vite@latest online-book-store --template react

cd online-book-store

npm install

**Install Routing & State Management**

npm install react-router-dom

**Context Setup (**CartContext.jsx**)**

import { createContext, useState } from 'react';

export const CartContext = createContext();

export function CartProvider({ children }) {

const [cart, setCart] = useState([]);

const addToCart = (book) => {

setCart([...cart, book]);

};

return (

<CartContext.Provider value={{ cart, addToCart }}>

{children}

</CartContext.Provider>

);

}

**main.jsx**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import { BrowserRouter } from 'react-router-dom';

import { CartProvider } from './context/CartContext';

ReactDOM.createRoot(document.getElementById('root')).render(

<BrowserRouter>

<CartProvider>

<App />

</CartProvider>

</BrowserRouter>

);

**App.jsx**

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

import Navbar from './components/Navbar';

import BookList from './components/BookList';

import Cart from './components/Cart';

export default function App() {

return (

<>

<Navbar />

<Routes>

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

<Route path="/cart" element={<Cart />} />

</Routes>

</>

);

}

**Navbar.jsx**

import { Link } from 'react-router-dom';

export default function Navbar() {

return (

<nav>

<Link to="/">Books</Link> | <Link to="/cart">Cart</Link>

</nav>

);

}

**BookList.jsx**

import { useEffect, useState } from 'react';

import BookCard from './BookCard';

export default function BookList() {

const [books, setBooks] = useState([]);

useEffect(() => {

fetch('https://openlibrary.org/subjects/fiction.json?limit=10')

.then((res) => res.json())

.then((data) => {

setBooks(data.works);

});

}, []);

return (

<div>

<h2>Available Books</h2>

{books.map((book) => (

<BookCard key={book.key} book={book} />

))}

</div>

);

}

**BookCard.jsx**

import { useContext } from 'react';

import { CartContext } from '../context/CartContext';

export default function BookCard({ book }) {

const { addToCart } = useContext(CartContext);

return (

<div style={{ border: '1px solid black', margin: '10px', padding: '10px' }}>

<h3>{book.title}</h3>

<p>Author: {book.authors[0]?.name}</p>

<button onClick={() => addToCart(book)}>Add to Cart</button>

</div>

);

}

**Cart.jsx**

import { useContext } from 'react';

import { CartContext } from '../context/CartContext';

export default function Cart() {

const { cart } = useContext(CartContext);

return (

<div>

<h2>Your Cart</h2>

{cart.length === 0 ? (

<p>No items in cart</p>

) : (

cart.map((item, index) => (

<div key={index}>

<h4>{item.title}</h4>

<p>{item.authors[0]?.name}</p>

</div>

))

)}

</div>

);

}

**16. Design a web application to get online survey for the recent movie “Retro” with various input fields (minimum of 8) and perform validation on input using Java Script and send them to server using HTTP POST**

retro-movie-survey/

├── client/

│ ├── index.html

│ ├── style.css

│ └── script.js

│

├── server/

│ ├── server.js

│ └── package.json

**Frontend (client/index.html)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Retro Movie Survey</title>

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

</head>

<body>

<h2>Retro Movie Survey</h2>

<form id="surveyForm">

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

<input type="email" id="email" placeholder="Your Email" required><br>

<input type="number" id="age" placeholder="Your Age" required><br>

<select id="gender" required>

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

<option>Male</option>

<option>Female</option>

<option>Other</option>

</select><br>

<input type="text" id="city" placeholder="City" required><br>

<input type="text" id="favCharacter" placeholder="Favorite Character" required><br>

<label>Rate the movie (1-5):</label>

<input type="number" id="rating" min="1" max="5" required><br>

<textarea id="feedback" placeholder="Your Feedback" required></textarea><br>

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

</form>

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

</body>

</html>

**JavaScript Validation (client/script.js)**

document.getElementById("surveyForm").addEventListener("submit", async function (e) {

e.preventDefault();

const name = document.getElementById("name").value.trim();

const email = document.getElementById("email").value.trim();

const age = parseInt(document.getElementById("age").value);

const gender = document.getElementById("gender").value;

const city = document.getElementById("city").value.trim();

const favCharacter = document.getElementById("favCharacter").value.trim();

const rating = parseInt(document.getElementById("rating").value);

const feedback = document.getElementById("feedback").value.trim();

// Simple validation

if (!name || !email || !age || !gender || !city || !favCharacter || !rating || !feedback) {

alert("Please fill in all fields.");

return;

}

const data = { name, email, age, gender, city, favCharacter, rating, feedback };

const res = await fetch("http://localhost:5000/survey", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify(data)

});

if (res.ok) {

alert("Thank you for submitting the survey!");

document.getElementById("surveyForm").reset();

} else {

alert("Something went wrong. Please try again.");

}

});

**Backend (server/server.js)**

const express = require("express");

const cors = require("cors");

const app = express();

const PORT = 5000;

app.use(cors());

app.use(express.json());

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

console.log("Survey Submitted:", req.body);

res.status(200).json({ message: "Survey received!" });

});

app.listen(PORT, () => {

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

});

**Backend Setup**

**Step 1: Initialize and Install**

cd server

npm init -y

npm install express cors

**17. Develop a web page for online tutorial website using Express routing to handle multiple routes from the client. The server must handle both GET and POST requests &amp; perform routing accordingly.**

**Solution:**

online-tutorial/

│

├── server.js

├── routes/

│ └── tutorials.js

├── public/

│ ├── index.html

│ └── contact.html

├── views/

│ └── success.html

├── package.json

**1. Initialize Project**mkdir online-tutorial

cd online-tutorial

npm init -y

npm install express body-parser

**2. server.js**

const express = require('express');

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

const path = require('path');

const tutorialRoutes = require('./routes/tutorials');

const app = express();

const PORT = 3000;

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

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

app.use('/', tutorialRoutes);

// Server listen

app.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

**3. routes/tutorials.js**

const express = require('express');

const router = express.Router();

const path = require('path');

// GET Home page

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

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

});

// GET Contact page

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

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

});

// POST Contact form submission

router.post('/contact', (req, res) => {

const { name, email, message } = req.body;

console.log(`Received: ${name}, ${email}, ${message}`);

res.sendFile(path.join(\_\_dirname, '../views/success.html'));

});

module.exports = router;

**4. public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Online Tutorials</title>

</head>

<body>

<h1>Welcome to the Online Tutorial Site</h1>

<p>Learn Web Development, Python, AI, and more!</p>

<a href="/contact">Contact Us</a>

</body>

</html>

**5. public/contact.html**

<!DOCTYPE html>

<html>

<head>

<title>Contact Us</title>

</head>

<body>

<h2>Contact Form</h2>

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

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

<input type="email" name="email" placeholder="Your Email" required><br><br>

<textarea name="message" placeholder="Your Message" required></textarea><br><br>

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

</form>

</body>

</html>

**6. views/success.html**

<!DOCTYPE html>

<html>

<head>

<title>Thank You</title>

</head>

<body>

<h2>Thank you for contacting us!</h2>

<p>We'll get back to you soon.</p>

<a href="/">Back to Home</a>

</body>

</html>

**18. Develop a web page for online tutorial website using Express routing to handle multiple routes from the client. The server must handle both GET and POST requests &amp; perform routing accordingly.**

online-tutorial/

├── server.js

├── routes/

│ └── pages.js

├── public/

│ ├── index.html

│ ├── about.html

│ ├── contact.html

│ └── success.html

├── package.json

**Step 1: Initialize Project**

mkdir online-tutorial

cd online-tutorial

npm init -y

npm install express body-parser

**2: server.js**

const express = require('express');

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

const path = require('path');

const pagesRouter = require('./routes/pages');

const app = express();

const PORT = 3000;

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

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', pagesRouter);

app.listen(PORT, () => {

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

});

**3: routes/pages.js**

const express = require('express');

const router = express.Router();

const path = require('path');

// GET: Home

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

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

});

// GET: About

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

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

});

// GET: Contact

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

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

});

// POST: Contact Form

router.post('/contact', (req, res) => {

const { name, email, message } = req.body;

console.log(`Contact Form Received:\nName: ${name}\nEmail: ${email}\nMessage: ${message}`);

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

});

module.exports = router;

**4. public/index.html**

<!DOCTYPE html>

<html>

<head><title>Online Tutorials</title></head>

<body>

<h1>Welcome to Our Tutorial Website</h1>

<a href="/about">About</a> |

<a href="/contact">Contact</a>

</body>

</html>

**public/about.html**

<!DOCTYPE html>

<html>

<head><title>About Us</title></head>

<body>

<h1>About Our Tutorial Platform</h1>

<p>We offer tutorials on web development, AI, and more!</p>

<a href="/">Home</a>

</body>

</html>

**public/contact.html**

<!DOCTYPE html>

<html>

<head><title>Contact Us</title></head>

<body>

<h1>Contact Us</h1>

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

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

<input name="email" type="email" placeholder="Your Email" required><br><br>

<textarea name="message" placeholder="Your Message" required></textarea><br><br>

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

</form>

<a href="/">Home</a>

</body>

</html>

**public/success.html**

<!DOCTYPE html>

<html>

<head><title>Success</title></head>

<body>

<h1>Thank you for contacting us!</h1>

<p>We’ll reach out to you soon.</p>

<a href="/">Back to Home</a>

</body>

</html>

**19. Develop a web page for session feedback on the FDP “AI in Blockchain”. Perform input validation using Java Script and use HTTP Request and Response to get some response based on input using NodeJS.**

fdp-feedback/

├── server.js

├── public/

│ ├── index.html

│ └── script.js

├── package.json

**1: Initialize Project**

mkdir fdp-feedback

cd fdp-feedback

npm init -y

npm install express body-parser

**2: server.js**

const express = require('express');

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

const path = require('path');

const app = express();

const PORT = 3000;

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

app.use(express.static(path.join(\_\_dirname, 'public')));

app.post('/submit-feedback', (req, res) => {

const { name, email, rating, comments } = req.body;

console.log(`Feedback received:\nName: ${name}\nEmail: ${email}\nRating: ${rating}\nComments: ${comments}`);

let message = "Thanks for your feedback!";

if (rating < 3) {

message = "We're sorry you didn't enjoy the session. We'll work to improve!";

}

res.send(`<h2>${message}</h2><a href="/">Back</a>`);

});

app.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

**3: public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>FDP Feedback – AI in Blockchain</title>

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

</head>

<body>

<h1>Session Feedback: AI in Blockchain</h1>

<form id="feedbackForm" action="/submit-feedback" method="POST" onsubmit="return validateForm()">

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

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

<label>Rate the Session (1 to 5):</label><br>

<input type="number" name="rating" id="rating" min="1" max="5" required><br><br>

<textarea name="comments" id="comments" placeholder="Any comments..." required></textarea><br><br>

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

</form>

</body>

</html>

**4: public/script.js**

function validateForm() {

const name = document.getElementById("name").value.trim();

const email = document.getElementById("email").value.trim();

const rating = parseInt(document.getElementById("rating").value);

const comments = document.getElementById("comments").value.trim();

if (!name || !email || !comments) {

alert("Please fill all fields.");

return false;

}

if (rating < 1 || rating > 5) {

alert("Rating must be between 1 and 5.");

return false;

}

return true;

}

**20. Develop a web page to apply for Passport by getting necessary details. Perform input validation using Java Script and use HTTP POST to send the data to web server and print it on console.**

passport-application/

├── server.js

├── public/

│ ├── index.html

│ └── script.js

├── package.json

**1: Initialize Node Project**

mkdir passport-application

cd passport-application

npm init -y

npm install express body-parser

**2: server.js**

const express = require('express');

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

const path = require('path');

const app = express();

const PORT = 3000;

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

app.use(express.static(path.join(\_\_dirname, 'public')));

app.post('/apply-passport', (req, res) => {

console.log("Passport Application Received:");

console.log(req.body);

res.send(`<h2>Application submitted successfully!</h2><a href="/">Back</a>`);

});

app.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

**3: public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Passport Application</title>

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

</head>

<body>

<h1>Passport Application Form</h1>

<form id="passportForm" action="/apply-passport" method="POST" onsubmit="return validateForm()">

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

<input type="date" name="dob" id="dob" placeholder="Date of Birth" required><br><br>

<input type="text" name="nationality" id="nationality" placeholder="Nationality" required><br><br>

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

<input type="text" name="phone" id="phone" placeholder="Phone Number" required><br><br>

<textarea name="address" id="address" placeholder="Address" required></textarea><br><br>

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

</form>

</body>

</html>

**4: public/script.js**

function validateForm() {

const fullname = document.getElementById("fullname").value.trim();

const dob = document.getElementById("dob").value;

const email = document.getElementById("email").value.trim();

const phone = document.getElementById("phone").value.trim();

if (!fullname || !dob || !email || !phone) {

alert("Please fill all required fields.");

return false;

}

const phoneRegex = /^\d{10}$/;

if (!phoneRegex.test(phone)) {

alert("Phone number must be 10 digits.");

return false;

}

return true;

}

**21. Design a web page to get details for applying Aadhar card with details and use HTTP POST to send the data to web server and store it in MongoDB.**

aadhar-application/

├── server.js

├── public/

│ ├── index.html

│ └── script.js

├── .env

├── package.json

**1: Setup Project**

mkdir aadhar-application

cd aadhar-application

npm init -y

npm install express body-parser mongoose dotenv

**2: Create .env**

MONGO\_URI=mongodb://localhost:27017/aadharDB

**3: server.js**

require('dotenv').config();

const express = require('express');

const mongoose = require('mongoose');

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

const path = require('path');

const app = express();

const PORT = 3000;

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

app.use(express.static(path.join(\_\_dirname, 'public')));

// MongoDB Connection

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log("MongoDB Connected"))

.catch(err => console.error("MongoDB Error:", err));

// Schema

const AadharSchema = new mongoose.Schema({

fullname: String,

dob: String,

gender: String,

phone: String,

address: String

});

const Aadhar = mongoose.model("Aadhar", AadharSchema);

// POST Route

app.post('/apply-aadhar', async (req, res) => {

const { fullname, dob, gender, phone, address } = req.body;

try {

const newApplication = new Aadhar({ fullname, dob, gender, phone, address });

await newApplication.save();

res.send(`<h2>Aadhaar Application Submitted!</h2><a href="/">Apply Another</a>`);

} catch (error) {

res.status(500).send("Failed to submit application.");

}

});

app.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

**4: public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Aadhaar Card Application</title>

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

</head>

<body>

<h1>Aadhaar Card Application Form</h1>

<form id="aadharForm" action="/apply-aadhar" method="POST" onsubmit="return validateForm()">

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

<input type="date" name="dob" id="dob" required><br><br>

<select name="gender" id="gender" required>

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

<option value="Male">Male</option>

<option value="Female">Female</option>

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

</select><br><br>

<input type="text" name="phone" id="phone" placeholder="Phone Number" required><br><br>

<textarea name="address" id="address" placeholder="Address" required></textarea><br><br>

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

</form>

</body>

</html>

**5: public/script.js**

function validateForm() {

const fullname = document.getElementById("fullname").value.trim();

const phone = document.getElementById("phone").value.trim();

if (!fullname || fullname.length < 3) {

alert("Full name must be at least 3 characters.");

return false;

}

const phoneRegex = /^\d{10}$/;

if (!phoneRegex.test(phone)) {

alert("Phone number must be exactly 10 digits.");

return false;

}

return true;

}

**22. Design a web page using React components and perform routing to display various movie contents for an online movie streaming platform.**

movie-streaming-app/

├── public/

├── src/

│ ├── components/

│ │ ├── Home.js

│ │ ├── Movies.js

│ │ ├── MovieDetails.js

│ │ ├── About.js

│ ├── App.js

│ └── index.js

├── package.json

**1: Create React App**

npx create-react-app movie-streaming-app

cd movie-streaming-app

npm install react-router-dom

**2: App.js**

import React from 'react';

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

import Home from './components/Home';

import Movies from './components/Movies';

import MovieDetails from './components/MovieDetails';

import About from './components/About';

function App() {

return (

<Router>

<nav>

<Link to="/">Home</Link> |

<Link to="/movies">Movies</Link> |

<Link to="/about">About</Link>

</nav>

<Routes>

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

<Route path="/movies" element={<Movies />} />

<Route path="/movies/:id" element={<MovieDetails />} />

<Route path="/about" element={<About />} />

</Routes>

</Router>

);

}

export default App;

**3: Components**

**Home.js**

import React from 'react';

const Home = () => <h2>Welcome to MovieStream!</h2>;

export default Home;

**Movies.js**

import React from 'react';

import { Link } from 'react-router-dom';

const Movies = () => {

const movieList = [

{ id: 1, title: "Inception" },

{ id: 2, title: "Interstellar" },

{ id: 3, title: "The Matrix" }

];

return (

<div>

<h2>Movies</h2>

<ul>

{movieList.map(movie => (

<li key={movie.id}>

<Link to={`/movies/${movie.id}`}>{movie.title}</Link>

</li>

))}

</ul>

</div>

);

};

export default Movies;

**MovieDetails.js**

import React from 'react';

import { useParams } from 'react-router-dom';

const MovieDetails = () => {

const { id } = useParams();

return <h3>Movie Details for Movie ID: {id}</h3>;

};

export default MovieDetails;

**About.js**

import React from 'react';

const About = () => <h2>About MovieStream: Your Online Movie Platform</h2>;

export default About;

api = [OMDb API](http://www.omdbapi.com/)

**23. Design a web page that collects starting and ending meter readings in the front-end and perform input validation to check the ending reading is always greater than starting reading. Use HTTP GET request and response to calculate the bill amount at the server using NodeJS.**

meter-reading-app/

├── server.js

├── public/

│ ├── index.html

│ └── script.js

├── package.json

**1: Initialize Project**

mkdir meter-reading-app

cd meter-reading-app

npm init -y

npm install express

**2.server.js**

const express = require('express');

const app = express();

const path = require('path');

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

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

const start = parseFloat(req.query.start);

const end = parseFloat(req.query.end);

if (isNaN(start) || isNaN(end) || end <= start) {

return res.send("Invalid meter readings.");

}

const unitRate = 5; // 5 ₹ per unit

const units = end - start;

const amount = units \* unitRate;

res.send(`<h2>Bill Amount: ₹${amount}</h2>`);

});

app.listen(3000, () => {

console.log("Server running at http://localhost:3000");

});

**public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Meter Reading</title>

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

</head>

<body>

<h1>Electricity Bill Calculator</h1>

<form id="billForm">

<label>Starting Reading:</label><br>

<input type="number" id="start" required><br><br>

<label>Ending Reading:</label><br>

<input type="number" id="end" required><br><br>

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

</form>

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

</body>

</html>

**public/script.js**

document.getElementById('billForm').addEventListener('submit', function(e) {

e.preventDefault();

const start = parseFloat(document.getElementById('start').value);

const end = parseFloat(document.getElementById('end').value);

if (isNaN(start) || isNaN(end)) {

alert("Please enter valid numbers.");

return;

}

if (end <= start) {

alert("Ending reading must be greater than starting reading.");

return;

}

fetch(`/calculate-bill?start=${start}&end=${end}`)

.then(res => res.text())

.then(data => {

document.getElementById('result').innerHTML = data;

});

});

**24. Design a web page using HTTP Request and Response to send and retrieve movie tickets details for online ticket booking and perform CRUD operations in the MongoDB using NodeJS.**

movie-ticket-booking/

├── server.js

├── public/

│ ├── index.html

│ └── script.js

├── models/

│ └── Ticket.js

├── package.json

**1: Setup and Install**

npm init -y

npm install express mongoose body-parser cors

**models/Ticket.js**

const mongoose = require('mongoose');

const ticketSchema = new mongoose.Schema({

movie: String,

name: String,

seats: Number,

date: String

});

module.exports = mongoose.model('Ticket', ticketSchema);

**server.js**

const express = require('express');

const mongoose = require('mongoose');

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

const Ticket = require('./models/Ticket');

const path = require('path');

const app = express();

mongoose.connect('mongodb://127.0.0.1:27017/tickets', { useNewUrlParser: true, useUnifiedTopology: true });

app.use(bodyParser.json());

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

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

// Create Ticket

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

const ticket = new Ticket(req.body);

await ticket.save();

res.send('Ticket booked!');

});

// Read Tickets

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

const tickets = await Ticket.find();

res.json(tickets);

});

// Update Ticket

app.put('/api/tickets/:id', async (req, res) => {

await Ticket.findByIdAndUpdate(req.params.id, req.body);

res.send('Ticket updated!');

});

// Delete Ticket

app.delete('/api/tickets/:id', async (req, res) => {

await Ticket.findByIdAndDelete(req.params.id);

res.send('Ticket deleted!');

});

app.listen(3000, () => console.log('Server running at <http://localhost:3000')>);

**public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Movie Ticket Booking</title>

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

</head>

<body>

<h1>Book a Ticket</h1>

<form id="ticketForm">

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

Movie: <input type="text" id="movie" required><br>

Seats: <input type="number" id="seats" required><br>

Date: <input type="date" id="date" required><br>

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

</form>

<h2>Booked Tickets</h2>

<ul id="ticketList"></ul>

</body>

</html>

**public/script.js**

const form = document.getElementById('ticketForm');

const ticketList = document.getElementById('ticketList');

form.addEventListener('submit', async (e) => {

e.preventDefault();

const data = {

name: form.name.value,

movie: form.movie.value,

seats: form.seats.value,

date: form.date.value

};

await fetch('/api/tickets', {

method: 'POST',

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

body: JSON.stringify(data)

});

form.reset();

loadTickets();

});

async function loadTickets() {

const res = await fetch('/api/tickets');

const tickets = await res.json();

ticketList.innerHTML = '';

tickets.forEach(ticket => {

const li = document.createElement('li');

li.innerHTML = `${ticket.name} booked for ${ticket.movie} (${ticket.seats} seats on ${ticket.date})

<button onclick="deleteTicket('${ticket.\_id}')">Delete</button>`;

ticketList.appendChild(li);

});

}

async function deleteTicket(id) {

await fetch(`/api/tickets/${id}`, { method: 'DELETE' });

loadTickets();

}

window.onload = loadTickets;

**25. Develop a website to display the recipe for a food item given by the user. Integrate appropriate API to fetch recipes details and use HTTP POST method to fetch the json data from API. Perform input validation for the recipe name using JavaScript.**

recipe-finder/

├── public/

│ ├── index.html

│ └── script.js

├── server.js

├── package.json

mkdir recipe-finder

cd recipe-finder

npm init -y

npm install express body-parser node-fetch

**server.js**

const express = require('express');

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

const fetch = require('node-fetch');

const app = express();

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

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

app.use(bodyParser.json());

app.post('/get-recipe', async (req, res) => {

const recipeName = req.body.recipe;

try {

const apiURL = `https://www.themealdb.com/api/json/v1/1/search.php?s=${recipeName}`;

const response = await fetch(apiURL);

const data = await response.json();

res.json(data);

} catch (error) {

res.status(500).json({ error: "Failed to fetch recipe" });

}

});

app.listen(3000, () => {

console.log("Server running on http://localhost:3000");

});

**public/index.html**

<!DOCTYPE html>

<html>

<head>

<title>Recipe Finder</title>

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

</head>

<body>

<h1>Find a Recipe</h1>

<form id="recipeForm">

<input type="text" id="recipe" placeholder="Enter food item" required>

<button type="submit">Get Recipe</button>

</form>

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

</body>

</html>

public/script.js

document.getElementById('recipeForm').addEventListener('submit', async (e) => {

e.preventDefault();

const recipe = document.getElementById('recipe').value.trim();

if (recipe === '') {

alert('Please enter a recipe name.');

return;

}

const response = await fetch('/get-recipe', {

method: 'POST',

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

body: JSON.stringify({ recipe })

});

const data = await response.json();

const resultDiv = document.getElementById('result');

resultDiv.innerHTML = '';

if (!data.meals) {

resultDiv.innerHTML = '<p>No recipe found.</p>';

return;

}

const meal = data.meals[0];

resultDiv.innerHTML = `

<h2>${meal.strMeal}</h2>

<img src="${meal.strMealThumb}" width="300"><br><br>

<strong>Category:</strong> ${meal.strCategory}<br>

<strong>Area:</strong> ${meal.strArea}<br>

<p>${meal.strInstructions}</p>

`;

});

**26. Develop an online News website that showcases various categories of news articles using React routing components and fetch news data from any API.**

news-website/

├── public/

├── src/

│ ├── components/

│ │ ├── Navbar.js

│ │ └── NewsList.js

│ ├── App.js

│ └── index.js

├── .env (for API key)

└── package.json

npx create-react-app news-website

cd news-website

npm install react-router-dom axios

**.env file**

REACT\_APP\_NEWS\_API\_KEY=your\_newsapi\_key\_here

**App.js**

import React from 'react';

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

import Navbar from './components/Navbar';

import NewsList from './components/NewsList';

function App() {

return (

<Router>

<Navbar />

<Routes>

<Route path="/" element={<NewsList category="general" />} />

<Route path="/business" element={<NewsList category="business" />} />

<Route path="/sports" element={<NewsList category="sports" />} />

<Route path="/technology" element={<NewsList category="technology" />} />

</Routes>

</Router>

);

}

export default App;

**Navbar.js – Navigation Bar**

import { Link } from 'react-router-dom';

const Navbar = () => (

<nav>

<h2>NewsPortal</h2>

<ul>

<li><Link to="/">General</Link></li>

<li><Link to="/business">Business</Link></li>

<li><Link to="/sports">Sports</Link></li>

<li><Link to="/technology">Tech</Link></li>

</ul>

</nav>

);

export default Navbar;

**NewsList.js**

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

import axios from 'axios';

const NewsList = ({ category }) => {

const [articles, setArticles] = useState([]);

useEffect(() => {

const fetchNews = async () => {

const apiKey = process.env.REACT\_APP\_NEWS\_API\_KEY;

const url = `https://newsapi.org/v2/top-headlines?country=us&category=${category}&apiKey=${apiKey}`;

try {

const response = await axios.get(url);

setArticles(response.data.articles);

} catch (error) {

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

}

};

fetchNews();

}, [category]);

return (

<div>

<h2>{category.toUpperCase()} News</h2>

{articles.map((article, index) => (

<div key={index} style={{ marginBottom: "20px" }}>

<h3>{article.title}</h3>

<p>{article.description}</p>

{article.urlToImage && <img src={article.urlToImage} alt="news" width="300" />}

<br />

<a href={article.url} target="\_blank" rel="noopener noreferrer">Read More</a>

</div>

))}

</div>

);

};

export default NewsList;

**27. Develop a website for patient details management using React as front-end to get &amp; display patient details. Use NodeJS to perform Read and Create operation on MongoDB.**

FRONTEND

client/

├── src/

│ ├── App.js

│ ├── components/

│ │ ├── AddPatient.js

│ │ └── PatientList.js

│ └── index.js

BACKEND

server/

├── models/

│ └── Patient.js

├── routes/

│ └── patientRoutes.js

├── server.js

**Backend (Node.js + Express)**

**1. Initialize backend**

mkdir server

cd server

npm init -y

npm install express mongoose cors body-parser

**2. MongoDB Model (models/Patient.js)**

const mongoose = require('mongoose');

const patientSchema = new mongoose.Schema({

name: String,

age: Number,

gender: String,

disease: String

});

module.exports = mongoose.model('Patient', patientSchema);

**3. Routes (routes/patientRoutes.js)**

const express = require('express');

const router = express.Router();

const Patient = require('../models/Patient');

router.post('/add', async (req, res) => {

const newPatient = new Patient(req.body);

await newPatient.save();

res.json({ message: 'Patient added successfully' });

});

router.get('/all', async (req, res) => {

const patients = await Patient.find();

res.json(patients);

});

module.exports = router;

**4. Server Setup (server.js)**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

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

const patientRoutes = require('./routes/patientRoutes');

const app = express();

app.use(cors());

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/patientDB', {

useNewUrlParser: true,

useUnifiedTopology: true

});

app.use('/patients', patientRoutes);

app.listen(5000, () => console.log('Server running on port 5000'));

**Frontend (React)**

1. **Create React App**

npx create-react-app client

cd client

npm install axios

**2. AddPatient.js**

import React, { useState } from 'react';

import axios from 'axios';

const AddPatient = () => {

const [form, setForm] = useState({ name: '', age: '', gender: '', disease: '' });

const handleChange = e => {

setForm({ ...form, [e.target.name]: e.target.value });

};

const handleSubmit = async e => {

e.preventDefault();

await axios.post('http://localhost:5000/patients/add', form);

alert('Patient Added!');

setForm({ name: '', age: '', gender: '', disease: '' });

};

return (

<form onSubmit={handleSubmit}>

<h2>Add Patient</h2>

<input name="name" placeholder="Name" onChange={handleChange} value={form.name} required />

<input name="age" type="number" placeholder="Age" onChange={handleChange} value={form.age} required />

<input name="gender" placeholder="Gender" onChange={handleChange} value={form.gender} required />

<input name="disease" placeholder="Disease" onChange={handleChange} value={form.disease} required />

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

</form>

);

};

export default AddPatient;

1. **PatientList.js**

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

import axios from 'axios';

const PatientList = () => {

const [patients, setPatients] = useState([]);

useEffect(() => {

axios.get('http://localhost:5000/patients/all')

.then(res => setPatients(res.data));

}, []);

return (

<div>

<h2>All Patients</h2>

<ul>

{patients.map((p, i) => (

<li key={i}>{p.name}, {p.age}, {p.gender}, {p.disease}</li>

))}

</ul>

</div>

);

};

export default PatientList;

1. **App.js**

import React from 'react';

import AddPatient from './components/AddPatient';

import PatientList from './components/PatientList';

function App() {

return (

<div>

<AddPatient />

<PatientList />

</div>

);

}

export default App;

**28. Design a web page for displaying various painting collections and their price details using React components and routing. When user wishes to buy a painting gather user details to send it to server using HTTP request and display the details in console.**

painting-store/

├── client/ # React frontend

└── server/ # Node.js backend

**Backend (Node.js + Express)**

**1. Set up server**

mkdir server

cd server

npm init -y

npm install express cors body-parser

**2. Create server.js**

const express = require('express');

const cors = require('cors');

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

const app = express();

app.use(cors());

app.use(bodyParser.json());

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

console.log('Purchase Details:', req.body);

res.json({ message: 'Purchase received!' });

});

app.listen(5000, () => console.log('Server running on port 5000'));

**Frontend (React + Axios + React Router)**

**1. Set up React**

npx create-react-app client

cd client

npm install axios react-router-dom

**2. Create Components**

**App.js**

import React from 'react';

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

import Paintings from './Paintings';

import Purchase from './Purchase';

function App() {

return (

<Router>

<Routes>

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

<Route path="/buy/:title/:price" element={<Purchase />} />

</Routes>

</Router>

);

}

export default App;

**Paintings.js**

import React from 'react';

import { Link } from 'react-router-dom';

const paintings = [

{ title: "Starry Night", price: 1500, img: "https://via.placeholder.com/150" },

{ title: "Mona Lisa", price: 2000, img: "https://via.placeholder.com/150" },

{ title: "The Scream", price: 1800, img: "https://via.placeholder.com/150" }

];

const Paintings = () => {

return (

<div>

<h1>Painting Collections</h1>

{paintings.map((p, i) => (

<div key={i}>

<img src={p.img} alt={p.title} />

<h3>{p.title}</h3>

<p>Price: ${p.price}</p>

<Link to={`/buy/${p.title}/${p.price}`}>Buy Now</Link>

</div>

))}

</div>

);

};

export default Paintings;

**Purchase.js**

import React, { useState } from 'react';

import { useParams } from 'react-router-dom';

import axios from 'axios';

const Purchase = () => {

const { title, price } = useParams();

const [user, setUser] = useState({ name: '', email: '', address: '' });

const handleChange = e => {

setUser({ ...user, [e.target.name]: e.target.value });

};

const handleSubmit = async e => {

e.preventDefault();

const purchaseData = { ...user, painting: title, price };

await axios.post('http://localhost:5000/buy', purchaseData);

alert('Purchase submitted!');

};

return (

<form onSubmit={handleSubmit}>

<h2>Buy: {title} (${price})</h2>

<input name="name" placeholder="Name" onChange={handleChange} required />

<input name="email" type="email" placeholder="Email" onChange={handleChange} required />

<input name="address" placeholder="Address" onChange={handleChange} required />

<button type="submit">Confirm Purchase</button>

</form>

);

};

export default Purchase;

**29. Develop a web site using HTML for event management and perform input validation using JavaScript. Then store those events registration details in MongoDB using NodeJS.**

event-management/

├── client/ # Frontend (HTML + JavaScript)

└── server/ # Backend (Node.js + MongoDB)

**Backend Setup (Node.js + MongoDB)**

**1. Install necessary packages for the server:**

mkdir server

cd server

npm init -y

npm install express mongoose cors body-parser

**MongoDB model (models/Event.js):**

const mongoose = require('mongoose');

const eventSchema = new mongoose.Schema({

name: String,

email: String,

phone: String,

event: String

});

module.exports = mongoose.model('Event', eventSchema);

**backend (server.js):**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

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

const Event = require('./models/Event');

const app = express();

app.use(cors());

app.use(bodyParser.json());

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/eventDB', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Event registration route (POST)

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

const { name, email, phone, event } = req.body;

const newEvent = new Event({ name, email, phone, event });

await newEvent.save();

res.json({ message: 'Event registration successful' });

});

// Start server

app.listen(5000, () => {

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

});

**Frontend (HTML + JavaScript)**

**1. Set up the frontend folder:**

mkdir client

cd client

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Event Registration</title>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

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

</head>

<body>

<h1>Event Registration</h1>

<form id="eventForm">

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

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

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

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

<label for="phone">Phone:</label>

<input type="tel" id="phone" name="phone" required /><br /><br />

<label for="event">Event:</label>

<select id="event" name="event" required>

<option value="Music Concert">Music Concert</option>

<option value="Tech Conference">Tech Conference</option>

<option value="Art Exhibition">Art Exhibition</option>

</select><br /><br />

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

</form>

<div id="message"></div>

</body>

</html>

**app.js**

// Select form and message div

const eventForm = document.getElementById('eventForm');

const messageDiv = document.getElementById('message');

// Form submission handler

eventForm.addEventListener('submit', async function (e) {

e.preventDefault();

// Get form data

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

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

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

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

// Input validation

if (!name || !email || !phone || !event) {

messageDiv.innerHTML = "<p style='color: red;'>All fields are required!</p>";

return;

}

// Send data to server using Axios

try {

const response = await axios.post('http://localhost:5000/register', {

name, email, phone, event

});

// Display success message

messageDiv.innerHTML = `<p style='color: green;'>${response.data.message}</p>`

// Clear form after submission

eventForm.reset();

} catch (error) {

// Display error message

messageDiv.innerHTML = `<p style='color: red;'>Error: ${error.message}</p>`;

}

});

**30. Develop an online food ordering website using react components and routing. Handle the ordering at the web server using various Express routes for operations like viewing foods, order status, delivery status, payment status.**

/food-ordering-website

/client (React frontend)

/public

/src

/components

Header.js

FoodList.js

FoodItem.js

OrderSummary.js

Payment.js

OrderStatus.js

DeliveryStatus.js

App.js

index.js

/server (Express backend)

/routes

foodRoutes.js

orderRoutes.js

app.js

package.json

**1. Setting Up the Backend (Express)**

npm init -y

npm install express cors body-parser

**Create server/app.js**

const express = require('express');

const cors = require('cors');

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

const app = express();

// Middleware setup

app.use(cors());

app.use(bodyParser.json());

// Import Routes

const foodRoutes = require('./routes/foodRoutes');

const orderRoutes = require('./routes/orderRoutes');

// Routes setup

app.use('/api/foods', foodRoutes);

app.use('/api/orders', orderRoutes);

// Server

const PORT = 5000;

app.listen(PORT, () => {

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

});

**Create server/routes/foodRoutes.js**

const express = require('express');

const router = express.Router();

// Sample data for foods

const foods = [

{ id: 1, name: 'Pizza', price: 10 },

{ id: 2, name: 'Burger', price: 5 },

{ id: 3, name: 'Pasta', price: 7 }

];

// Get all foods

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

res.json(foods);

});

// Get a specific food by id

router.get('/:id', (req, res) => {

const food = foods.find(f => f.id === parseInt(req.params.id));

if (food) {

res.json(food);

} else {

res.status(404).send('Food not found');

}

});

module.exports = router;

Create server/routes/orderRoutes.js

const express = require('express');

const router = express.Router();

// Order status, payment status, and delivery status

let orderStatus = {

status: 'Pending',

paymentStatus: 'Not Paid',

deliveryStatus: 'Not Delivered'

};

// Create a new order

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

const { foodId, quantity, address } = req.body;

// Process the order here (e.g., saving to DB or some logic)

orderStatus.status = 'In Progress';

orderStatus.paymentStatus = 'Not Paid';

orderStatus.deliveryStatus = 'Not Delivered';

res.json({ message: 'Order placed successfully', orderStatus });

});

// Get order status

router.get('/:orderId/status', (req, res) => {

res.json(orderStatus);

});

// Update payment status

router.put('/:orderId/payment', (req, res) => {

orderStatus.paymentStatus = req.body.paymentStatus;

res.json({ message: 'Payment status updated', orderStatus });

});

// Update delivery status

router.put('/:orderId/delivery', (req, res) => {

orderStatus.deliveryStatus = req.body.deliveryStatus;

res.json({ message: 'Delivery status updated', orderStatus });

});

module.exports = router;

**2. Setting Up the Frontend (React)**

npx create-react-app client

cd client

npm install axios react-router-dom

**Create client/src/components/Header.js**

import React from 'react';

import { Link } from 'react-router-dom';

const Header = () => {

return (

<header>

<h1>Food Ordering Website</h1>

<nav>

<Link to="/">Home</Link>

<Link to="/order-status">Order Status</Link>

<Link to="/delivery-status">Delivery Status</Link>

<Link to="/payment-status">Payment Status</Link>

</nav>

</header>

);

};

export default Header;

**Create client/src/components/FoodList.js**

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

import axios from 'axios';

import FoodItem from './FoodItem';

const FoodList = () => {

const [foods, setFoods] = useState([]);

useEffect(() => {

axios.get('http://localhost:5000/api/foods')

.then(response => setFoods(response.data))

.catch(error => console.log(error));

}, []);

return (

<div>

{foods.map(food => (

<FoodItem key={food.id} food={food} />

))}

</div>

);

};

export default FoodList;

**Create client/src/components/FoodItem.js**

import React from 'react';

const FoodItem = ({ food }) => {

return (

<div>

<h3>{food.name}</h3>

<p>Price: ${food.price}</p>

<button>Add to Cart</button>

</div>

);

};

export default FoodItem;

**Create client/src/components/OrderSummary.js**

import React from 'react';

const OrderSummary = ({ orderDetails }) => {

return (

<div>

<h2>Order Summary</h2>

<p>Food: {orderDetails.food}</p>

<p>Quantity: {orderDetails.quantity}</p>

<p>Total: ${orderDetails.total}</p>

</div>

);

};

export default OrderSummary;

**Routing in client/src/App.js**

import React from 'react';

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

import Header from './components/Header';

import FoodList from './components/FoodList';

import OrderStatus from './components/OrderStatus';

import DeliveryStatus from './components/DeliveryStatus';

import PaymentStatus from './components/PaymentStatus';

const App = () => {

return (

<Router>

<Header />

<Switch>

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

<Route path="/order-status" component={OrderStatus} />

<Route path="/delivery-status" component={DeliveryStatus} />

<Route path="/payment-status" component={PaymentStatus} />

</Switch>

</Router>

);

};

export default App;

**31. Design a web page for student’s internal mark management using HTML and perform input validation using JavaScript to check marks are less than 100. Perform CRUD operations on the MongoDB using NodeJS based on student roll number.**

/student-mark-management

/public

index.html

script.js

/server

app.js

routes

studentRoutes.js

package.json

**Backend Setup (Node.js with MongoDB)**

npm init -y

npm install express mongoose body-parser cors

Create server/app.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

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

const studentRoutes = require('./routes/studentRoutes');

const app = express();

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/student\_marks', { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log("MongoDB connected"))

.catch(err => console.log("MongoDB connection error:", err));

// Routes

app.use('/api/students', studentRoutes);

// Server

const PORT = 5000;

app.listen(PORT, () => {

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

});

**Create server/routes/studentRoutes.js**

const express = require('express');

const router = express.Router();

const Student = require('../models/studentModel');

// CREATE Student

router.post('/', async (req, res) => {

const { rollNo, name, marks } = req.body;

// Validation

if (marks > 100 || marks < 0) {

return res.status(400).send('Marks must be between 0 and 100.');

}

const newStudent = new Student({ rollNo, name, marks });

try {

await newStudent.save();

res.status(201).json(newStudent);

} catch (err) {

res.status(400).send(err);

}

});

// READ Student by Roll Number

router.get('/:rollNo', async (req, res) => {

const { rollNo } = req.params;

try {

const student = await Student.findOne({ rollNo });

if (!student) {

return res.status(404).send('Student not found');

}

res.json(student);

} catch (err) {

res.status(500).send(err);

}

});

// UPDATE Student

router.put('/:rollNo', async (req, res) => {

const { rollNo } = req.params;

const { name, marks } = req.body;

// Validation

if (marks > 100 || marks < 0) {

return res.status(400).send('Marks must be between 0 and 100.');

}

try {

const updatedStudent = await Student.findOneAndUpdate(

{ rollNo },

{ name, marks },

{ new: true }

);

if (!updatedStudent) {

return res.status(404).send('Student not found');

}

res.json(updatedStudent);

} catch (err) {

res.status(500).send(err);

}

});

// DELETE Student

router.delete('/:rollNo', async (req, res) => {

const { rollNo } = req.params;

try {

const deletedStudent = await Student.findOneAndDelete({ rollNo });

if (!deletedStudent) {

return res.status(404).send('Student not found');

}

res.json({ message: 'Student deleted successfully' });

} catch (err) {

res.status(500).send(err);

}

});

module.exports = router;

**Create server/models/studentModel.js**

const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({

rollNo: { type: String, required: true, unique: true },

name: { type: String, required: true },

marks: { type: Number, required: true }

});

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

**2. Frontend Setup (HTML & JavaScript)**

**Create public/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Marks Management</title>

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

</head>

<body>

<h1>Student Internal Marks Management</h1>

<!-- Form to Add or Update Student -->

<form id="studentForm">

<label for="rollNo">Roll Number:</label>

<input type="text" id="rollNo" required>

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

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

<label for="marks">Marks:</label>

<input type="number" id="marks" required>

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

</form>

<h2>Search Student by Roll Number</h2>

<input type="text" id="searchRollNo" placeholder="Enter roll number">

<button onclick="searchStudent()">Search</button>

<div id="studentDetails"></div>

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

</body>

</html>

**Create public/script.js**

document.getElementById('studentForm').addEventListener('submit', function(e) {

e.preventDefault();

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

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

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

// Validate marks

if (marks < 0 || marks > 100) {

alert('Marks should be between 0 and 100');

return;

}

const student = { rollNo, name, marks };

// Send the data to the backend (POST or PUT request)

fetch('http://localhost:5000/api/students', {

method: 'POST',

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

body: JSON.stringify(student)

})

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

.then(data => alert('Student added successfully'))

.catch(error => console.error('Error:', error));

});

function searchStudent() {

const rollNo = document.getElementById('searchRollNo').value;

fetch(`http://localhost:5000/api/students/${rollNo}`)

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

.then(data => {

if (data.message) {

document.getElementById('studentDetails').innerHTML = `<p>${data.message}</p>`;

} else {

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

<p>Roll No: ${data.rollNo}</p>

<p>Name: ${data.name}</p>

<p>Marks: ${data.marks}</p>

`;

}

})

.catch(error => console.error('Error:', error));

}

**32. Develop an online technical discussion forum where user can post questions, answer other’s questions, search for answers, rate the answers. Perform CRUD operations on the MongoDB that contains all post.**

technical-forum/

│

├── server/

│ ├── app.js

│ ├── models/

│ │ ├── Question.js

│ │ └── Answer.js

│ └── routes/

│ └── forumRoutes.js

│

├── public/

│ ├── index.html

│ └── script.js

│

└── package.json

**MongoDB Schemas**

**models/Question.js**

const mongoose = require('mongoose');

const questionSchema = new mongoose.Schema({

title: String,

description: String,

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

});

module.exports = mongoose.model('Question', questionSchema);

models/Answer.js

const mongoose = require('mongoose');

const answerSchema = new mongoose.Schema({

questionId: { type: mongoose.Schema.Types.ObjectId, ref: 'Question' },

content: String,

rating: { type: Number, default: 0 },

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

});

module.exports = mongoose.model('Answer', answerSchema);

**Backend with Express**

**server/app.js**

const express = require('express');

const mongoose = require('mongoose');

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

const cors = require('cors');

const forumRoutes = require('./routes/forumRoutes');

const app = express();

app.use(cors());

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/tech\_forum', {

useNewUrlParser: true,

useUnifiedTopology: true,

});

app.use('/api/forum', forumRoutes);

app.listen(5000, () => {

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

});

**routes/forumRoutes.js**

const express = require('express');

const router = express.Router();

const Question = require('../models/Question');

const Answer = require('../models/Answer');

// CREATE a Question

router.post('/questions', async (req, res) => {

const { title, description } = req.body;

const question = new Question({ title, description });

await question.save();

res.json(question);

});

// GET all Questions

router.get('/questions', async (req, res) => {

const questions = await Question.find();

res.json(questions);

});

// GET a Question by ID (with answers)

router.get('/questions/:id', async (req, res) => {

const question = await Question.findById(req.params.id);

const answers = await Answer.find({ questionId: req.params.id });

res.json({ question, answers });

});

// ADD Answer

router.post('/answers', async (req, res) => {

const { questionId, content } = req.body;

const answer = new Answer({ questionId, content });

await answer.save();

res.json(answer);

});

// RATE an Answer

router.put('/answers/:id/rate', async (req, res) => {

const { value } = req.body; // value = +1 or -1

const answer = await Answer.findByIdAndUpdate(

req.params.id,

{ $inc: { rating: value } },

{ new: true }

);

res.json(answer);

});

// DELETE Question

router.delete('/questions/:id', async (req, res) => {

await Question.findByIdAndDelete(req.params.id);

await Answer.deleteMany({ questionId: req.params.id });

res.sendStatus(200);

});

module.exports = router;

**Frontend (public/index.html + script.js) index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Technical Forum</title>

</head>

<body>

<h1>Ask a Question</h1>

<form id="questionForm">

<input type="text" id="title" placeholder="Title" required />

<textarea id="description" placeholder="Describe your question" required></textarea>

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

</form>

<h2>All Questions</h2>

<ul id="questionList"></ul>

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

</body>

</html>

**script.js**

document.getElementById('questionForm').addEventListener('submit', async function (e) {

e.preventDefault();

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

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

await fetch('http://localhost:5000/api/forum/questions', {

method: 'POST',

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

body: JSON.stringify({ title, description })

});

loadQuestions();

});

async function loadQuestions() {

const res = await fetch('http://localhost:5000/api/forum/questions');

const questions = await res.json();

const list = document.getElementById('questionList');

list.innerHTML = '';

questions.forEach(q => {

const li = document.createElement('li');

li.innerHTML = `<strong>${q.title}</strong>: ${q.description}`;

list.appendChild(li);

});

}

loadQuestions();