**Backend**

**model.js:**

const mongoose = require("mongoose");

const studentSchema = new mongoose.Schema({

std\_name: { type: String, required: true },

std\_rno: { type: String, required: true },

group: { type: String, required: true },

});

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

module.exports = Student;

**server.js:**

const mongoose = require("mongoose");

const express = require("express");

const cors = require("cors");

const app = express();

app.use(cors());

app.use(express.json());

const URL =

"mongodb+srv://subhapreetpatro2004:Patro202172112@cluster0.38gjjjz.mongodb.net/";

const PORT = 3547;

const Student = require("./models/model");

mongoose

.connect(URL)

.then(() => {

console.log("Database connected successfully");

})

.catch((error) => {

console.error("Failed to connect to the database", error);

});

// Create student

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

const { std\_name, std\_rno, group } = req.body;

if (!std\_name || !std\_rno || !group) {

return res.status(400).json({ error: "Please fill all the fields" });

}

try {

const newStudent = new Student({ std\_name, std\_rno, group });

await newStudent.save();

return res.status(201).json({ message: "Successfully added" });

} catch (error) {

console.error(error);

return res.status(500).json({ error: "Failed to add student" });

}

});

// Read all students

app.get("/getStudentList", async (req, res) => {

try {

const students = await Student.find();

return res.status(200).json(students);

} catch (error) {

console.error(error);

return res.status(500).json({ error: "Internal server error" });

}

});

// Update student

app.put("/updateStudent/:id", async (req, res) => {

const { id } = req.params;

const { std\_name, std\_rno, group } = req.body;

try {

const updatedStudent = await Student.findByIdAndUpdate(

id,

{ std\_name, std\_rno, group },

{ new: true }

);

if (!updatedStudent) {

return res.status(404).json({ error: "Student not found" });

}

return res

.status(200)

.json({ message: "Successfully updated", student: updatedStudent });

} catch (error) {

console.error(error);

return res.status(500).json({ error: "Failed to update student" });

}

});

// Delete student

app.delete("/deleteStudent/:id", async (req, res) => {

const { id } = req.params;

try {

const deletedStudent = await Student.findByIdAndDelete(id);

if (!deletedStudent) {

return res.status(404).json({ error: "Student not found" });

}

return res.status(200).json({ message: "Successfully deleted" });

} catch (error) {

console.error(error);

return res.status(500).json({ error: "Failed to delete student" });

}

});

app.listen(PORT, (error) => {

if (!error) {

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

} else {

console.error("Error starting the server:", error);

}

});

**Client:**

**App.js:**

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

import "./App.css";

import profile from "./profile-placeholder.png";

function App() {

const [students, setStudents] = useState([]);

const [newStudent, setNewStudent] = useState({

std\_name: "",

std\_rno: "",

group: "",

});

const [editStudent, setEditStudent] = useState(null);

useEffect(() => {

fetch("http://localhost:3547/getStudentList")

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

.then((data) => setStudents(data))

.catch((error) => console.error("Error fetching students:", error));

}, []);

const handleAddStudent = async () => {

try {

const response = await fetch("http://localhost:3547/addStudent", {

method: "POST",

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

body: JSON.stringify(newStudent),

});

const data = await response.json();

if (response.ok) {

setStudents([...students, data]);

setNewStudent({ std\_name: "", std\_rno: "", group: "" });

window.location.reload("false");

} else {

console.error("Error adding student:", data.error);

}

} catch (error) {

console.error("Error adding student:", error);

}

};

const handleUpdateStudent = async (id) => {

try {

const response = await fetch(

`http://localhost:3547/updateStudent/${id}`,

{

method: "PUT",

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

body: JSON.stringify(editStudent),

}

);

const data = await response.json();

if (response.ok) {

setStudents(

students.map((student) =>

student.\_id === id ? data.student : student

)

);

setEditStudent(null);

} else {

console.error("Error updating student:", data.error);

}

} catch (error) {

console.error("Error updating student:", error);

}

};

const handleDeleteStudent = async (id) => {

try {

const response = await fetch(

`http://localhost:3547/deleteStudent/${id}`,

{

method: "DELETE",

}

);

const data = await response.json();

if (response.ok) {

setStudents(students.filter((student) => student.\_id !== id));

} else {

console.error("Error deleting student:", data.error);

}

} catch (error) {

console.error("Error deleting student:", error);

}

};

return (

<div className="container">

<h1 className="title">Student Management</h1>

<form>

<div>

<div>

<input

type="text"

placeholder="Enter Name"

value={newStudent.std\_name}

required

onChange={(e) =>

setNewStudent({ ...newStudent, std\_name: e.target.value })

}

/>

</div>

<div>

<input

type="text"

placeholder="Enter Roll Number"

value={newStudent.std\_rno}

required

onChange={(e) =>

setNewStudent({ ...newStudent, std\_rno: e.target.value })

}

/>

</div>

<div>

<input

type="text"

placeholder="Enter Group"

value={newStudent.group}

required

onChange={(e) =>

setNewStudent({ ...newStudent, group: e.target.value })

}

/>

</div>

<button

onClick={handleAddStudent}

style={{

backgroundColor: "green",

}}

>

Add Student

</button>

</div>

</form>

<div className="cards">

{students.map((student) => (

<div className="profile-card" key={student.\_id}>

<div className="background-block"></div>

<div className="profile">

<img src={profile} alt="profile" className="profile-img" />

</div>

<div className="card-content">

{editStudent && editStudent.\_id === student.\_id ? (

<>

<div>

<label>Name:</label>

<input

type="text"

value={editStudent.std\_name}

onChange={(e) =>

setEditStudent({

...editStudent,

std\_name: e.target.value,

})

}

/>

</div>

<div>

<label>Roll No.:</label>

<input

type="text"

value={editStudent.std\_rno}

onChange={(e) =>

setEditStudent({

...editStudent,

std\_rno: e.target.value,

})

}

/>

</div>

<div>

<label>Group:</label>

<input

type="text"

value={editStudent.group}

onChange={(e) =>

setEditStudent({

...editStudent,

group: e.target.value,

})

}

/>

</div>

<button onClick={() => handleUpdateStudent(student.\_id)}>

Save

</button>

<button onClick={() => setEditStudent(null)}>Cancel</button>

</>

) : (

<>

<h2>{student.std\_name}</h2>

<p>Roll Number: {student.std\_rno}</p>

<p>Group: {student.group}</p>

<button onClick={() => setEditStudent(student)}>Edit</button>

<button onClick={() => handleDeleteStudent(student.\_id)}>

Delete

</button>

</>

)}

</div>

</div>

))}

</div>

</div>

);

}

export default App;

**App.css:**

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

.title {

text-align: center;

margin-bottom: 20px;

font-size: 2.5em;

color: #333;

}

form {

display: flex;

justify-content: center;

margin-bottom: 20px;

}

form input {

padding: 10px;

margin-right: 10px;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 1em;

}

form button {

padding: 10px 20px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1em;

}

form button[type="button"] {

background-color: #dc3545;

}

form button:hover {

opacity: 0.8;

}

.cards {

display: flex;

flex-wrap: wrap;

gap: 20px;

justify-content: center;

}

.profile-card {

background-color: rgb(251, 158, 44);

border-radius: 10px;

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

overflow: hidden;

width: 300px;

text-align: center;

transition: transform 0.2s;

}

.profile-card:hover {

transform: scale(1.05);

}

.background-block {

height: 150px;

background: #3382d5;

}

.profile {

position: relative;

top: -50px;

}

.profile-img {

width: 150px;

height: 150px;

border-radius: 50%;

border: 5px solid white;

background-color: white;

}

.card-content {

padding: 20px;

}

.card-content h2 {

margin: 10px 0;

font-size: 1.5em;

color: #333;

}

.card-content p {

margin: 5px 0;

font-size: 1em;

color: #050505;

}

.card-content button {

margin-top: 10px;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 0.9em;

}

.card-content button:hover {

opacity: 0.8;

}

.card-content button:first-child {

background-color: #ffc107;

margin-right: 10px;

}

.card-content button:last-child {

background-color: #dc3545;

}