**Components:**

**DisplayStudent.js:**

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

import { Link, useParams } from "react-router-dom";

const DisplayStudent = () => {

const { id } = useParams();

const [empdata, setempdatachange] = useState([]);

const HandlerEmpDetails = () => {

fetch(`https://vercel-mongodb-server-alpha.vercel.app/getStudent/${id}`)

.then((res) => {

if (!res.ok) {

throw new Error("Failed to fetch student");

}

return res.json();

})

.then((resp) => {

setempdatachange(resp); // Ensure this state update is correct

console.log(resp); // Log to confirm the correct data

})

.catch((err) => {

alert("Failed to get: " + err.message);

});

};

console.log("Student Details:", empdata);

useEffect(() => {

HandlerEmpDetails();

}, []);

return (

<div>

<h4 className="text-secondary" style={{ textAlign: "center" }}>

Details Student Content Here .......

</h4>

<div

className="card"

style={{

width: 500,

marginLeft: 500,

marginTop: 100,

backgroundColor: "#9FE2BF",

}}

>

<div className="card-header">

<h4 className="text-primary" style={{ textAlign: "center" }}>

Student Details

</h4>

</div>

<div className="card-body">

<div class="card text-white bg-info mb-3">

<div

class="card-header text-danger"

style={{ fontSize: 18, fontWeight: "bold" }}

>

{empdata.htno}

</div>

<div class="card-body">

<h5 class="card-title">{empdata.fullname}</h5>

<p class="card-text">

<i>{empdata.email}</i>

</p>

<p class="card-text" style={{ fontWeight: "bold" }}>

{empdata.mobile}

</p>

</div>

</div>

</div>

<div className="card-footer">

<Link to="/" className="text-danger">

Back

</Link>

</div>

</div>

</div>

);

};

export default DisplayStudent;

**EditStudent.js:**

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

import { Link, useParams, useNavigate } from "react-router-dom";

const EditStudentComponent = () => {

const { id } = useParams();

const navigate = useNavigate();

const [htno, setHtno] = useState("");

const [fullname, setFullname] = useState("");

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

const [group, setGroup] = useState("");

const [email, setEmail] = useState("");

const [mobile, setMobile] = useState("");

const [gender, setGender] = useState("male");

useEffect(() => {

fetch(`https://vercel-mongodb-server-alpha.vercel.app/getStudent/${id}`)

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

.then((data) => {

setHtno(data.htno);

setFullname(data.fullname);

setPassword(data.password);

setGroup(data.group);

setEmail(data.email);

setMobile(data.mobile);

setGender(data.gender);

})

.catch((error) => {

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

});

}, [id]);

const handleEditStudent = async (event) => {

event.preventDefault();

try {

const response = await fetch(

`https://vercel-mongodb-server-alpha.vercel.app/updateStudent/${id}`,

{

method: "PUT",

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

body: JSON.stringify({

htno,

fullname,

password,

group,

email,

mobile,

gender,

}),

}

);

if (!response.ok) {

// Check if response status is not ok

throw new Error("Failed to update student");

} else {

alert("Student Updated Successfully");

navigate("/");

}

} catch (err) {

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

alert("Failed: " + err.message);

}

};

return (

<div className="offset-lg-6 col-lg-6" style={{ marginLeft: 350 }}>

<form onSubmit={handleEditStudent} className="container">

<div className="card">

<div className="card-title">

<h4 style={{ textAlign: "center" }}>Edit Student</h4>

</div>

<div className="card-body">

<div className="row">

<div className="col-lg-6">

<div className="form-group">

<label>Hallticket No</label>

<input

type="text"

className="form-control"

value={htno}

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

placeholder="Enter Hallticket No."

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Name</label>

<input

type="text"

className="form-control"

value={fullname}

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

placeholder="Enter Name"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Password</label>

<input

type="password"

className="form-control"

value={password}

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

placeholder="Enter Password"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Group</label>

<select

className="form-control"

value={group}

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

required

>

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

<option value="Group-7">Group-7</option>

<option value="Group-8">Group-8</option>

</select>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Email</label>

<input

type="email"

className="form-control"

value={email}

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

placeholder="Enter Email"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Mobile</label>

<input

type="tel"

className="form-control"

value={mobile}

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

placeholder="Enter Mobile No."

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>Gender</label>

<div>

<label className="radio-inline">

<input

type="radio"

name="gender"

value="male"

checked={gender === "male"}

onChange={() => setGender("male")}

/>

Male

</label>

<label className="radio-inline">

<input

type="radio"

name="gender"

value="female"

checked={gender === "female"}

onChange={() => setGender("female")}

/>

Female

</label>

</div>

</div>

</div>

</div>

</div>

<div className="card-footer">

<button className="btn btn-sm btn-success" type="submit">

Update

</button>

<Link

className="btn btn-sm btn-danger"

to="/"

style={{ marginLeft: 10 }}

>

Back

</Link>

</div>

</div>

</form>

</div>

);

};

export default EditStudentComponent;

**NewStudentPage.js:**

import React, { useState } from "react";

import { Link, useNavigate } from "react-router-dom";

const AddStudentComponent = () => {

const [htno, setHtno] = useState("");

const [fullname, setFullname] = useState("");

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

const [group, setGroup] = useState("");

const [email, setEmail] = useState("");

const [mobile, setMobile] = useState("");

const [gender, setGender] = useState("male");

const navigate = useNavigate();

const handleAddStudent = async (event) => {

event.preventDefault();

try {

const response = await fetch(

"https://vercel-mongodb-server-alpha.vercel.app/addStudent",

{

method: "POST",

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

body: JSON.stringify({

htno,

fullname,

password,

group,

email,

mobile,

gender,

}),

}

);

if (!response.ok) {

// Check if response status is not ok

if (response.status === 422) {

// Handle specific status code

alert("Invalid Registration, Username Already Exists");

console.error(htno, fullname, password, group, email, mobile, gender);

console.error(await response.text());

} else {

throw new Error("Failed to add student");

}

} else {

// Handle success

alert("Student Registered Successfully");

navigate("/");

}

} catch (err) {

console.error(htno, fullname, password, group, email, mobile, gender);

alert("Failed: " + err.message);

}

};

return (

<div className="offset-lg-6 col-lg-6" style={{ marginLeft: 350 }}>

<form onSubmit={handleAddStudent} className="container">

<div className="card">

<div className="card-title">

<h4 style={{ textAlign: "center" }}>New Student</h4>

</div>

<div className="card-body">

<div className="row">

<div className="col-lg-6">

<div className="form-group">

<label>

Hallticket No <span className="errmsg">\*</span>

</label>

<input

type="text"

className="form-control"

value={htno}

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

placeholder="Enter Hallticket No."

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Name <span className="errmsg">\*</span>

</label>

<input

type="text"

className="form-control"

value={fullname}

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

placeholder="Enter Name"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Password <span className="errmsg">\*</span>

</label>

<input

type="password"

className="form-control"

value={password}

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

placeholder="Enter Password"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Group <span className="errmsg">\*</span>

</label>

<select

className="form-control"

value={group}

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

required

>

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

<option value="Group-7">Group-7</option>

<option value="Group-8">Group-8</option>

</select>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Email <span className="errmsg">\*</span>

</label>

<input

type="email"

className="form-control"

value={email}

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

placeholder="Enter Email"

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Mobile <span className="errmsg">\*</span>

</label>

<input

type="tel"

className="form-control"

value={mobile}

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

placeholder="Enter Mobile No."

required

/>

</div>

</div>

<div className="col-lg-6">

<div className="form-group">

<label>

Gender <span className="errmsg">\*</span>

</label>

<div>

<label className="radio-inline">

<input

type="radio"

name="gender"

value="male"

checked={gender === "male"}

onChange={() => setGender("male")}

/>

Male

</label>

<label className="radio-inline">

<input

type="radio"

name="gender"

value="female"

checked={gender === "female"}

onChange={() => setGender("female")}

/>

Female

</label>

</div>

</div>

</div>

</div>

</div>

<div className="card-footer">

<button className="btn btn-sm btn-success" type="submit">

Register

</button>

<Link

className="btn btn-sm btn-danger"

to="/StudentPage"

style={{ marginLeft: 10 }}

>

Back

</Link>

</div>

</div>

</form>

</div>

);

};

export default AddStudentComponent;

**StudentPage.js:**import React, { useEffect, useState } from "react";

import { Link, useNavigate } from "react-router-dom";

const StudentPage = () => {

const [studentData, setStudentData] = useState([]);

const navigate = useNavigate();

const handleFetchStudent = async () => {

try {

const response = await fetch(

"https://vercel-mongodb-server-alpha.vercel.app/getStudents"

);

if (!response.ok) {

throw new Error("Failed to fetch student data");

}

const studentsData = await response.json();

setStudentData(studentsData);

} catch (error) {

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

}

};

useEffect(() => {

handleFetchStudent();

}, []);

const handleLoadStdDetails = async (id) => {

navigate(`/StudentPage/display/${id}`);

};

const handleEditStd = (id) => {

navigate(`/StudentPage/edit/${id}`);

};

const handleDeleteStd = async (id) => {

const confirmDelete = window.confirm(

"Are you sure you want to delete this student?"

);

if (confirmDelete) {

try {

const response = await fetch(

`https://vercel-mongodb-server-alpha.vercel.app/deleteStudent/${id}`,

{

method: "DELETE",

}

);

if (!response.ok) {

throw new Error("Failed to delete student");

}

handleFetchStudent(); // Fetch updated student data after deletion

} catch (error) {

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

alert("Failed to delete student. Please try again.");

}

}

};

return (

<div>

<h4 className="text-danger" style={{ textAlign: "center" }}>

Display StudentPage Content Here .......

</h4>

<div className="card" style={{ width: "80%", margin: "auto" }}>

<div className="card-header">

<div className="card-title">

<h4 style={{ textAlign: "center" }}>Students Data</h4>

</div>

</div>

<div className="card-body">

<table

className="table table-striped table-bordered table-hover"

style={{ textAlign: "center" }}

>

<thead className="thead-dark">

<tr>

<th scope="col">S.No</th>

<th scope="col">Ht No.</th>

<th scope="col">Name</th>

<th scope="col">Group</th>

<th scope="col">EmailId</th>

<th scope="col">Mobile</th>

<th scope="col">Gender</th>

<th scope="col">Actions</th>

</tr>

</thead>

<tbody>

{studentData.map((student, index) => (

<tr key={student.std\_htno}>

<th>{index + 1}</th>

<td>{student.htno}</td>

<td>{student.fullname}</td>

<td>{student.group}</td>

<td>{student.email}</td>

<td>{student.mobile}</td>

<td>{student.gender}</td>

<td style={{ textAlign: "center" }}>

<a

className="text-primary"

onClick={() => handleLoadStdDetails(student.\_id)}

style={{ marginLeft: 10 }}

>

View

</a>

<a

className="text-success"

onClick={() => handleEditStd(student.\_id)}

style={{ marginLeft: 10 }}

>

Edit

</a>

<a

className="text-danger"

onClick={() => handleDeleteStd(student.\_id)}

style={{ marginLeft: 10 }}

>

Delete

</a>

</td>

</tr>

))}

</tbody>

</table>

</div>

<div className="card-footer">

<Link className="btn btn-sm btn-success" to="/NewStudentPage">

Add Student

</Link>

</div>

</div>

</div>

);

};

export default StudentPage;

**connectMongoDB.js:**

const mongoose = require("mongoose");

const connectDB = async () => {

try {

await mongoose.connect(process.env.MONGODB\_CONNECT\_URL);

console.log("Connect to MongoDB successfully");

} catch (error) {

console.log("Connection failed");

}

};

module.exports = connectDB;

**usermodel.js:**

var mongoose = require("mongoose");

// Define Schemas

const userSchema = mongoose.Schema({

username: String,

password: String,

fullname: String,

email: String,

phone: Number,

country: String,

address: String,

});

const employeeSchema = mongoose.Schema({

empid: { type: String, required: true, unique: true }, // Define empid field

empname: String,

empemail: String,

empphone: String,

emprole: String,

empactive: Boolean,

});

// Define Student Schema

const studentSchema = new mongoose.Schema({

htno: String,

fullname: String,

password: String,

group: String,

email: String,

mobile: String,

gender: {

type: String,

enum: ["male", "female"], // To restrict values to 'male' or 'female'

required: true,

},

});

// Create Student Model

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

// create models based on schemas

const users = mongoose.model("users", userSchema);

const employees = mongoose.model("employees", employeeSchema);

module.exports = { users, employees, students };

**Index.js:**

var express = require("express");

const cors = require("cors");

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

require("dotenv").config();

const { users, employees, students } = require("./usermodel");

//use the express

var app = express();

const PORT = process.env.PORT;

const connectDB = require("./connectMongoDB");

connectDB();

//bodyparser to be used for sending and receiving data

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

app.use(bodyParser.json());

app.use(

cors({

// Allow requests from multiple origins, including your GitHub Pages site

origin:"\*",

optionsSuccessStatus: 200,

methods: ["GET", "POST", "PUT", "DELETE"], // Methods you want to allow

allowedHeaders: ["Content-Type", "Authorization"], // Headers to allow

credentials: true, // If you want to allow cookies/credentials

})

);

//Database connection Establishment

//writing registration page route to push the data

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

const {

username,

password,

fullname,

email,

phone,

country,

address,

gender,

} = req.body;

if (

(!username ||

!password ||

!fullname ||

!email ||

!phone ||

!country ||

!address,

!gender)

) {

return res

.status(422)

.json({ error: "Please fill the fields properly!!!" });

}

try {

const userExist = await users.findOne({ username });

console.log("Creating New user with username: ", username);

if (userExist) {

return res.status(422).json({ error: "Username Already Exists!!!" });

} else {

const newuser = new users({

username,

password,

fullname,

email,

phone,

country,

address,

});

await newuser.save();

console.log("New User Registered Successfully...");

res.status(201).json({ message: "User Registered Successfully..." });

}

} catch (error) {

console.log(error.message);

}

});

//writing registration page route to get the data

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

const { username, password } = req.body;

try {

const user = await users.findOne({ username: username });

console.log("Trying to Login with username:", user);

if (user) {

if (user.password === password) {

res.status(200).json({ message: "Success", fullname: user.fullname });

console.log("User Logined Successfully....");

} else {

res.json("Password is incorrect...");

}

} else {

res.json("No record exists for this username...");

}

} catch (error) {

console.error("Error:", error.message);

res.status(500).json("An error occurred while processing your request...");

}

});

//writing homepage route to display data

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

try {

const user = await users.find();

return res.json(user);

} catch (error) {

console.log(error.message);

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

}

});

//implementing post route to push new employee data

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

const { empid, empname, empemail, empphone, emprole, empactive } = req.body;

if (

!empid ||

!empname ||

!empemail ||

!empphone ||

!emprole ||

empactive === undefined

) {

return res

.status(422)

.json({ error: "Please fill all the fields properly!!!" });

}

try {

// Check if employee with the given empid already exists

const existingEmployee = await employees.find({ empid: empid });

console.log("Checking for Employee with Empid:", empid);

if (existingEmployee.length > 0) {

return res

.status(422)

.json({ error: "Employee with the same ID already exists!!!" });

} else {

// Create a new employee document

const newEmployee = new employees({

empid,

empname,

empemail,

empphone,

emprole,

empactive,

});

// Save the new employee document to the database

await newEmployee.save();

console.log("New Employee Registered Successfully...");

// Send success response

return res

.status(201)

.json({ message: "Employee registered successfully." });

}

} catch (error) {

console.log(error.message);

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

}

});

//addStudent

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

const { htno, fullname, password, group, email, mobile, gender } = req.body;

if (

!htno ||

!fullname ||

!password ||

!group ||

!email ||

!mobile ||

!gender

) {

return res

.status(422)

.json({ error: "Please fill all the fields properly!!!" });

}

try {

// Check if student with the given htno already exists

const existingStudent = await students.findOne({ htno });

console.log("Checking for Student with HTNO:", htno);

if (existingStudent) {

return res.status(422).json({

error: "Student with the same HTNO already exists!!!",

});

} else {

// Create a new student document

const newStudent = new students({

htno,

fullname,

password,

group,

email,

mobile,

gender,

});

// Save the new student document to the database

await newStudent.save();

console.log("New Student Registered Successfully...");

// Send success response

return res

.status(201)

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

}

} catch (error) {

console.log(error.message);

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

}

});

//get students

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

try {

const Students = await students.find();

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

} catch (error) {

console.log(error.message);

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

}

});

app.get("/getStudent/:id", async (req, res) => {

try {

const student = await students.findById(req.params.id);

if (!student) {

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

}

res.json(student);

} catch (error) {

console.error(error);

res

.status(500)

.json({ error: "An error occurred while retrieving doctor details" });

}

});

//delete students

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

try {

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

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

} catch (error) {

console.log(error.message);

}

});

//update student by id

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

const { htno, fullname, password, group, email, mobile, gender } = req.body;

try {

// Find the doctor by ID and update the fields

const updatedStudent = await students.findByIdAndUpdate(

req.params.id,

{

htno,

fullname,

password,

group,

email,

mobile,

gender,

},

{ new: true } // Set to true to return the updated document

);

if (!updatedStudent) {

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

}

// Return the updated Student

return res.json(updatedStudent);

} catch (err) {

console.error(err.message);

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

}

});

//implementing get route to get employees data

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

try {

const allemployees = await employees.find();

return res.json(allemployees);

} catch (error) {

console.log(error.message);

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

}

});

//specific id based search to get specific employee data

app.get("/getemployee/:empid", async (req, res) => {

const employeeEmpid = req.params.empid;

console.log("Get employee with empid:", employeeEmpid);

try {

const employeedata = await employees.findOne({ empid: employeeEmpid });

return res.json(employeedata);

} catch (error) {

console.log(error.message);

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

}

});

// Update Employee Details based on specific ID

app.put("/updateemployee/:empid", async (req, res) => {

const employeeEmpid = req.params.empid;

try {

// Find the employee by empid

let employee = await employees.findOne({ empid: employeeEmpid });

if (!employee) {

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

}

// Update the employee's data with the data from the request body

Object.assign(employee, req.body);

// Save the updated employee data

await employee.save();

return res.json({ message: "Employee updated successfully", employee });

} catch (error) {

console.error("Error updating employee:", error.message);

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

}

});

// DELETE endpoint to delete a single document by ID

// If empid is your unique identifier, query by it

app.delete("/deleteemployee/:empid", async (req, res) => {

const employeeEmpid = req.params.empid;

console.log("Deleting employee with empid:", employeeEmpid);

try {

const deletedEmployee = await employees.deleteOne({ empid: employeeEmpid });

if (deletedEmployee.deletedCount === 0) {

console.log("Employee not found with empid:", employeeEmpid);

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

}

console.log("Employee deleted successfully");

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

} catch (error) {

console.error("Error deleting employee:", error.message);

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

}

});

//Server Connection Establishment

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

if (error) {

console.log("Failed to connect server");

} else {

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

}

});

**App.js:**

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

import "./App.css";

import StudentPage from "./components/Student/StudentPage";

import AddStudentComponent from "./components/Student/NewStudentPage";

import RegistrationPage from "./components/Pages/RegistrationPage";

import EmployeeDetails from "./components/Employee/EmployeeDetails";

import DetailsEmployee from "./components/Employee/DetailsEmployeePage";

import CreateEmployee from "./components/Employee/CreateEmployeePage";

import EditEmployee from "./components/Employee/EditEmployeePage";

import Login from "./components/Pages/LoginPage";

import HomeComponent from "./components/HomePage";

import EditStudentComponent from "./components/Student/EditStudent";

import DisplayStudent from "./components/Student/DisplayStudent";

function App() {

return (

<div className="App">

<h3 className="text-secondary" style={{ textAlign: "center" }}>

Welcome to Sample MERN Project

</h3>

<div className="container-fluid">

<BrowserRouter basename="2211cs010547\_mern\_stack/">

<Routes>

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

<Route path="/NewStudentPage" element={<AddStudentComponent />} />

<Route

path="/StudentPage/edit/:id"

element={<EditStudentComponent />}

/>

<Route

path="/StudentPage/display/:id"

element={<DisplayStudent />}

/>

</Routes>

</BrowserRouter>

</div>

</div>

);

}

export default App;