**Index.js**

**import express from 'express';**

**import dotenv from 'dotenv';**

**import appRoutes from './src/routes/index.js';**

**dotenv.config();**

**const PORT = parseInt(process.env.PORT); // Parse PORT as an integer**

**const app = express();**

**app.use(express.json());**

**app.use('/', appRoutes);**

**app.listen(PORT, () => console.log(`App listening on port ${PORT}`));**

**In the src folder under controller subfolder**

**Mentor.js**

**import mentorModel from "../modules/mentor.js";**

**import studentModel from "../modules/student.js";**

**let createMentor = async (req,res)=>{**

**try {**

**let mentor = await mentorModel.findOne({mentorName:req.body.mentorName})**

**if(mentor)**

**{**

**res.status(400).send({**

**message:`Mentor ${req.body.mentorName} already exist`**

**})**

**}**

**else{**

**let mentorName = req.body**

**await mentorModel.create(mentorName)**

**res.status(201).send({**

**message:`Mentor ${req.body.mentorName} created successfully**

**})**

**}**

**} catch (error) {**

**res.status(500).send({**

**message:"Internal server Error",**

**error:error.message**

**})**

**}**

**}**

**const assignStudentsToMentors = async (req, res) => {**

**try {**

**const { mentorId, \_id: studentId } = req.body;**

**const student = await studentModel.findOne({ \_id: studentId });**

**if (!student) {**

**return res.status(400).send({**

**message: "Student not found"**

**});**

**} else if (student.currentMentor) {**

**return res.status(400).send({**

**message: "Mentor already assigned to the student"**

**});**

**}**

**const mentor = await mentorModel.findOne({ \_id: mentorId });**

**if (!mentor) {**

**return res.status(400).send({**

**message: "Mentor not found"**

**});**

**}**

**await studentModel.updateOne(**

**{ \_id: studentId },**

**{**

**$set: {**

**currentMentor: mentorId,**

**mentorName: mentor.mentorName // Assigning mentor's name to the student**

**}**

**}**

**);**

**const updatedStudent = await studentModel.findOne({ \_id: studentId });**

**if (!updatedStudent) {**

**return res.status(400).send({**

**message: "Student not found after assignment"**

**});**

**}**

**return res.status(201).send({**

**message: "Assigned student to mentor successfully",**

**updatedStudent**

**});**

**} catch (error) {**

**return res.status(500).send({**

**message: "Internal server Error",**

**error: error.message**

**});**

**}**

**};**

**const getAllStudentsByMentor = async (req, res) => {**

**try {**

**const { mentorid: mentorId } = req.params;**

**if (!mentorId) {**

**return res.status(400).send({**

**message: "Invalid or missing mentorId"**

**});**

**}**

**const students = await studentModel.find({ currentMentor: mentorId });**

**if (students.length > 0) {**

**return res.status(200).send({**

**message: "Students fetched successfully",**

**students**

**});**

**} else {**

**return res.status(400).send({**

**message: "No students assigned to this mentor"**

**});**

**}**

**} catch (error) {**

**return res.status(500).send({**

**message: "Internal server Error",**

**error: error.message**

**});**

**}**

**};**

**export default {**

**createMentor,**

**assignStudentsToMentors,**

**getAllStudentsByMentor**

**};**

**Student.js**

**import studentModel from "../modules/student.js";**

**import mentorModel from "../modules/mentor.js";**

**import mentor from "./mentor.js";**

**const createStudents = async(req,res)=>{**

**try {**

**const studentName = req.body;**

**if(studentName){**

**const student = await studentModel.findOne({studentName:req.body.studentName})**

**if(!student)**

**{**

**const student = await studentModel.create(req.body)**

**res.status(201).send({**

**message:"Student Created Successfully",**

**student**

**})**

**}**

**else{**

**res.status(404).send({**

**message:`student ${req.body.studentName} is Already Exist`**

**})**

**}**

**}**

**else**

**{**

**res.status(404).send({**

**message:"StudentName is Required"**

**})**

**}**

**} catch (error) {**

**res.status(500).send({**

**message:"Internal server Error",**

**error:error.message**

**})**

**}**

**}**

**const changeMentorforStudent = async (req, res) => {**

**try {**

**const { studentId, mentorId } = req.body;**

**const student = await studentModel.findOne({ \_id: studentId });**

**if (!student) {**

**return res.status(400).send({**

**message: "Student not found"**

**});**

**}**

**const mentor = await mentorModel.findOne({ \_id: mentorId });**

**if (!mentor) {**

**return res.status(400).send({**

**message: "Mentor not found"**

**});**

**}**

**await studentModel.updateOne({\_id:studentId},{$set:{previousMentor:student.mentorName}})**

**await studentModel.updateOne(**

**{ \_id: studentId },**

**{**

**$set: {**

**currentMentor: mentorId,**

**mentorName: mentor.mentorName**

**}**

**}**

**);**

**const newmentor = await studentModel.findOne({\_id:studentId})**

**res.status(201).send({**

**message: "Mentor assigned successfully for the student",**

**newmentor**

**});**

**} catch (error) {**

**res.status(500).send({**

**message: "Internal server Error",**

**error: error.message**

**});**

**}**

**};**

**const getPreviousMentor = async (req, res) => {**

**try {**

**const { studentName } = req.body;**

**const student = await studentModel.findOne({ studentName });**

**if (!student) {**

**return res.status(400).send({**

**message: "Student not found"**

**});**

**}**

**const previousMentor = await mentorModel.findOne({ mentorName: student.previousMentor });**

**if (!previousMentor) {**

**return res.status(400).send({**

**message: "Previous mentor not found for this student"**

**});**

**}**

**return res.status(200).send({**

**message: "Previous mentor fetched successfully",**

**previousMentor**

**});**

**} catch (error) {**

**return res.status(500).send({**

**message: "Internal server Error",**

**error: error.message**

**});**

**}**

**};**

**export default {**

**createStudents,**

**changeMentorforStudent,**

**getPreviousMentor**

**};**

**In the src folder under modules sub folder**

**Connect.js**

**import mongoose from "mongoose";**

**import dotenv from 'dotenv'**

**dotenv.config()**

**try {**

**mongoose.connect(`${process.env.dbUrl}/${process.env.dbName}`)**

**console.log("mongoose connect")**

**} catch (error) {**

**console.log(error)**

**}**

**export default mongoose**

**mentor.js**

**import mongoose from './connect.js';**

**const mentorSchema = new mongoose.Schema({**

**mentorName:{type:String,required:["Mentor Name Is required"]}**

**},**

**{**

**versionKey:false**

**})**

**const mentorModel = mongoose.model("mentors",mentorSchema)**

**export default mentorModel**

**Student.js**

**import mongoose from './connect.js';**

**const studentSchema = new mongoose.Schema({**

**studentName: {**

**type: String,**

**required: [true, "Student name is required"]**

**},**

**mentorName:{**

**type:String**

**},**

**currentMentor: {**

**type: mongoose.Schema.Types.ObjectId, // Assuming the mentor is referenced by ObjectId**

**ref: 'mentors' // Assuming mentors are stored in a collection named 'mentors'**

**},**

**previousMentor: {**

**type:String,**

**ref: 'mentors'**

**}**

**}, { versionKey: false });**

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

**export default studentModel;**

**In the src folder under the routes subfolder**

**Index.js**

**import express from 'express'**

**import mentorRoutes from './mentor.js'**

**import studenRoutes from './student.js'**

**const router = express.Router(**

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

**res.status(200).send(`<h1> Welcome to Backend application of Students asssigning to Mentor </h>`)**

**})**

**router.use('/mentor',mentorRoutes)**

**router.use('/student',studenRoutes)**

**export default router;**

**Student.js**

**import express from 'express'**

**import studentCtr from '../controller/student.js'**

**const router = express.Router()**

**router.post('/create',studentCtr.createStudents)**

**router.post('/assignChangeMentorForStudent',studentCtr.changeMentorforStudent)**

**router.get('/previouslyAssignedmentor',studentCtr.getPreviousMentor)**

**export default router**

**Index.js**

**import express from 'express'**

**import mentorRoutes from './mentor.js'**

**import studenRoutes from './student.js'**

**const router = express.Router()**

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

**res.status(200).send(`<h1> Welcome to Backend application of Students asssigning to Mentor </h>`)**

**})**

**router.use('/mentor',mentorRoutes)**

**router.use('/student',studenRoutes)**

**export default router**