**CRUD practice Tasks (1)**

**🏫 1. Student-Teacher Management System**

A simple system to manage students and teachers.

**Schema:**

* **Student**: name, age, class, teacherId
* **Teacher**: name, subject, teacherId

**Tasks:**

1. **Create Students** (POST /students)
2. **Create Teachers** (POST /teachers)
3. **Assign a Teacher to a Student** (PATCH /students/:id)
4. **Get All Students of a Teacher** (GET /teachers/:id/students)
5. **Update Student Info** (PATCH /students/:id)
6. **Delete a Student** (DELETE /students/:id)

**✅ API URLs & Methods (Using Query Params)**

|  |  |  |
| --- | --- | --- |
| Operation | Method | URL Example |
| Create a Student | POST | http://localhost:5000/students?name=John&age=12&roll  =21&class=6th&TeacherId=T1999 |
| Create a Teacher | POST | http://localhost:5000/teachers?name=Mr.Smith&email=smith@mail.com& subject=Math&TeacherId=T1999 |
| Assign a Teacher to a Student | PATCH | http://localhost:5000/updatestudent?id=studentID&TeacherId=T1999 |
| Get All Students of a Teacher | GET | http://localhost:5000/studentsbyteacher?TeacherId=T1999 |
| Update Student Info | PATCH | http://localhost:5000/updatestudent?id=studentID&name=NewName&age=15 |
| Delete a Student | DELETE | http://localhost:5000/deletestudent?id=studentID |

**CODE**

const express = require("express");

const mongoose = require("mongoose");

const app = express();

app.use(express.json());

mongoose.connect("mongodb://127.0.0.1:27017/schoolDB", { useNewUrlParser: true, useUnifiedTopology: true })

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

const Student = mongoose.model("Student", new mongoose.Schema({ name: String, age: Number, roll: Number, class: String, TeacherId: String }));

const Teacher = mongoose.model("Teacher", new mongoose.Schema({ name: String, email: String, subject: String, TeacherId: String }));

// ✅ Create Student

app.post("/students", async (req, res) => res.send(await new Student(req.query).save()));

// ✅ Create Teacher

app.post("/teachers", async (req, res) => res.send(await new Teacher(req.query).save()));

// ✅ Assign a Teacher / Update Student

app.patch("/updatestudent", async (req, res) => {

    const { id, ...updateData } = req.query;

    if (!id) return res.status(400).send({ message: "Student ID is required" });

    res.send(await Student.findByIdAndUpdate(id, updateData, { new: true }));

});

// ✅ Get All Students of a Teacher

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

    if (!req.query.TeacherId) return res.status(400).send({ message: "Teacher ID is required" });

    res.send(await Student.find({ TeacherId: req.query.TeacherId }));

});

// ✅ Delete a Student

app.delete("/deletestudent", async (req, res) => {

    if (!req.query.id) return res.status(400).send({ message: "Student ID is required" });

    res.send(await Student.findByIdAndDelete(req.query.id));

});

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

Here are **10 MongoDB queries** and **5 API queries** to practice using your **schoolDB** database with Student and Teacher collections.

### ****1. Find All Students in Class "10A"****

**db.students.find({ class: "10A" });**

### ****2. Find Students Assigned to a Specific Teacher****

(Replace *"T123"* with an actual *TeacherId*)

**db.students.find({ TeacherId: "T123" });**

### ****3. Find Teachers Who Teach "Mathematics"****

**db.teachers.find({ subject: "Mathematics" });**

### ****4. Count the Number of Students in a Particular Class****

**db.students.countDocuments({ class: "10A" });**

### ****5. Find Students Aged Between 14 and 16****

**db.students.find({ age: { $gte: 14, $lte: 16 } });**

### ****6. Update a Student’s Class****

(Replace *{student\_id}* with the actual ID)

**db.students.updateOne({ \_id: ObjectId("{student\_id}") }, { $set: { class: "11B" } });**

### ****7. Increase Age of All Students by 1 Year****

**db.students.updateMany({}, { $inc: { age: 1 } });**

### ****8. Delete All Students from Class "9C"****

**db.students.deleteMany({ class: "9C" });**

### ****9. Find the Oldest Student****

**db.students.find().sort({ age: -1 }).limit(1);**

### ****10. Group Students by Class and Count Them****

**db.students.aggregate([**

**{ $group: { \_id: "$class", totalStudents: { $sum: 1 } } }**

**]);**

### ****1. Add a New Student****

**POST /students?name=Rahul&age=15&roll=101&class=10A&TeacherId=T123**

### ****2. Assign a Teacher to a Student****

(Replace *{student\_id}* with an actual student’s ID)

**PATCH /updatestudent?id={student\_id}&TeacherId=T456**

### ****3. Get All Students Under a Specific Teacher****

(Replace *T123* with an actual *TeacherId*)

**GET /studentsbyteacher?TeacherId=T123**

### ****4. Delete a Specific Student****

(Replace *{student\_id}* with an actual student’s ID)

**DELETE /deletestudent?id={student\_id}**

### ****5. Delete a Teacher by ID****

(Replace *{teacher\_id}* with an actual teacher’s ID)

**DELETE /deleteteacher?id={teacher\_id}**