**🏥 4. Hospital Management System**

A basic system to store patient records.

**Schema:**

* **Patient**: name, age, disease, doctorID
* **Doctors:** name, specialization, doctorID

**Tasks:**

1. **Register Patients** (POST /patients)
2. **Register Doctors** (POST /patients)
3. **Assign a Doctor to a Patient** (PATCH /patients/:id)
4. **Get All Patients of a Doctor** (GET /doctors/:id/patients)
5. **Delete Patient Record** (DELETE /patients/:id)
6. **Delete Doctors Record** (DELETE /doctors/:id)

**✅ Hospital Management System - API Endpoints & URLs**

|  |  |  |
| --- | --- | --- |
| Operation | Method | URL Example |
| Register Patients | POST | http://localhost:5000/patients?name=John&age=30&disease=Flu&doctorID=D101 |
| Register Doctors | POST | http://localhost:5000/doctors?name=Dr. Smith&specialization=Cardiology&doctorID=D101 |
| Assign a Doctor to a Patient | PATCH | http://localhost:5000/updatepatient?id=P202&doctorID=D102 |
| Get All Patients of a Doctor | GET | http://localhost:5000/doctors?id=D101&patients=true |
| Delete Patient Record | DELETE | http://localhost:5000/deletepatient?id=P202 |
| Delete Doctor Record | DELETE | http://localhost:5000/deletedoctor?id=D101 |

**CODE**

const express = require("express");

const mongoose = require("mongoose");

const app = express();

app.use(express.json());

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

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

const Patient = mongoose.model("Patient", new mongoose.Schema({

    name: String, age: Number, disease: String, doctorID: String

}));

const Doctor = mongoose.model("Doctor", new mongoose.Schema({

    name: String, specialization: String, doctorID: String

}));

// ✅ Register Patients

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

// ✅ Register Doctors

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

// ✅ Assign a Doctor to a Patient

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

    const { id, doctorID } = req.query;

    if (!id || !doctorID) return res.status(400).send({ message: "Patient ID and Doctor ID are required" });

    res.send(await Patient.findByIdAndUpdate(id, { doctorID }, { new: true }));

});

// ✅ Get All Patients of a Doctor

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

    if (req.query.patients) return res.send(await Patient.find({ doctorID: req.query.id }));

    res.send(await Doctor.findById(req.query.id));

});

// ✅ Delete Patient Record

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

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

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

});

// ✅ Delete Doctor Record

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

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

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

});

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

**1. Retrieve all patients in the hospital**

**db.patients.find();**

**2. Find all patients assigned to a specific doctor**

(Replace *"doctor123"* with an actual *doctorID*)

**db.patients.find({ doctorID: "doctor123" });**

**3. Retrieve all doctors with a specific specialization**

(Replace *"Cardiologist"* with the required specialization)

**db.doctors.find({ specialization: "Cardiologist" });**

**4. Count the number of patients under each doctor**

**db.patients.aggregate([**

**{ $group: { \_id: "$doctorID", totalPatients: { $sum: 1 } } }**

**]);**

**5. Find all patients who are older than 60**

**db.patients.find({ age: { $gt: 60 } });**

**6. Update the disease of a specific patient**

(Replace *{patient\_id}* with an actual patient's ID)

**db.patients.updateOne({ \_id: ObjectId("{patient\_id}") }, { $set: { disease: "Flu" } });**

**7. Delete all patients who have recovered (disease field is empty)**

**db.patients.deleteMany({ disease: "" });**

**8. Retrieve patients whose names start with "A"**

**db.patients.find({ name: /^A/ });**

**9. Sort doctors alphabetically by their names**

**db.doctors.find().sort({ name: 1 });**

**10. Find the most common disease among patients**

**db.patients.aggregate([**

**{ $group: { \_id: "$disease", count: { $sum: 1 } } },**

**{ $sort: { count: -1 } },**

**{ $limit: 1 }**

**]);**

**10 API Queries** (For Testing the Express API)

**11. Register a New Patient**

**POST /patients?name=John Doe&age=45&disease=Diabetes&doctorID=doctor123**

**12. Register a New Doctor**

**POST /doctors?name=Dr. Smith&specialization=Neurologist&doctorID=doc456**

**13. Get Details of a Specific Doctor**

(Replace *{doctor\_id}* with the actual *doctorID*)

**GET /doctors?id={doctor\_id}**

**14. Get All Patients of a Specific Doctor**

(Replace *{doctor\_id}* with the actual *doctorID*)

**GET /doctors?id={doctor\_id}&patients=true**

**15. Update a Patient's Assigned Doctor**

(Replace *{patient\_id}* with an actual patient's ID and *{doctor\_id}* with a doctor’s ID)

**PATCH /updatepatient?id={patient\_id}&doctorID={doctor\_id}**

**16. Retrieve All Doctors**

**GET /doctors**

**17. Retrieve All Patients**

**GET /patients**

**18. Delete a Specific Patient**

(Replace *{patient\_id}* with the actual patient's ID)

**DELETE /deletepatient?id={patient\_id}**

**19. Delete a Specific Doctor**

(Replace *{doctor\_id}* with the actual doctor's ID)

**DELETE /deletedoctor?id={doctor\_id}**

**20. Find all patients suffering from "Asthma"**

**GET /patients?disease=Asthma**