**Objective :** Create a RESTful service with Node, Express and MongoDB

**CODE:**

**Model**

const mongoose = require('mongoose');

const employeeSchema = new mongoose.Schema({

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

position: String,

salary: Number,

email: { type: String, unique: true }

});

module.exports = mongoose.model('Employee', employeeSchema);

**Routes**

const express = require('express');

const router = express.Router();

const Employee = require('../models/employee');

// Create a new employee

router.post('/', async (req, res) => {

try {

const employee = new Employee(req.body);

await employee.save();

res.status(201).json(employee);

} catch (err) {

res.status(400).json({ error: err.message });

}

});

// Get all employees

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

const employees = await Employee.find();

res.json(employees);

});

// Get a specific employee

router.get('/:id', async (req, res) => {

try {

const employee = await Employee.findById(req.params.id);

if (!employee) return res.status(404).json({ error: 'Not found' });

res.json(employee);

} catch (err) {

res.status(400).json({ error: err.message });

}

});

// Update an employee

router.put('/:id', async (req, res) => {

try {

const employee = await Employee.findByIdAndUpdate(req.params.id, req.body, { new: true });

res.json(employee);

} catch (err) {

res.status(400).json({ error: err.message });

}

});

// Delete an employee

router.delete('/:id', async (req, res) => {

try {

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

res.json({ message: 'Employee deleted' });

} catch (err) {

res.status(400).json({ error: err.message });

}

});

module.exports = router;

**Server**

const express = require("express");

const mongoose = require("mongoose");

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

const app = express();

const PORT = 3001;

// Middleware

app.use(bodyParser.json());

// MongoDB connection

mongoose.connect("mongodb://127.0.0.1:27017/employeedb")

.then(() => console.log("✅ Connected to MongoDB"))

.catch((err) => console.error("❌ MongoDB connection error:", err));

// Schema

const employeeSchema = new mongoose.Schema({

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

position: String,

salary: Number,

email: String

});

// Model

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

// Routes

// POST: Add new employee(s)

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

try {

const data = req.body;

const result = Array.isArray(data)

? await Employee.insertMany(data)

: await new Employee(data).save();

res.status(201).json(result);

} catch (error) {

res.status(400).json({ error: error.message });

}

});

// GET: All employees

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

try {

const employees = await Employee.find();

res.json(employees);

} catch (error) {

res.status(500).json({ error: error.message });

}

});

// GET: Single employee

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

try {

const employee = await Employee.findById(req.params.id);

if (!employee) return res.status(404).json({ message: "Employee not found" });

res.json(employee);

} catch (error) {

res.status(500).json({ error: error.message });

}

});

// PUT: Update employee

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

try {

const updated = await Employee.findByIdAndUpdate(req.params.id, req.body, { new: true });

if (!updated) return res.status(404).json({ message: "Employee not found" });

res.json(updated);

} catch (error) {

res.status(500).json({ error: error.message });

}

});

// DELETE: Remove employee

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

try {

const deleted = await Employee.findByIdAndDelete(req.params.id);

if (!deleted) return res.status(404).json({ message: "Employee not found" });

res.json({ message: "Employee deleted" });

} catch (error) {

res.status(500).json({ error: error.message });

}

});

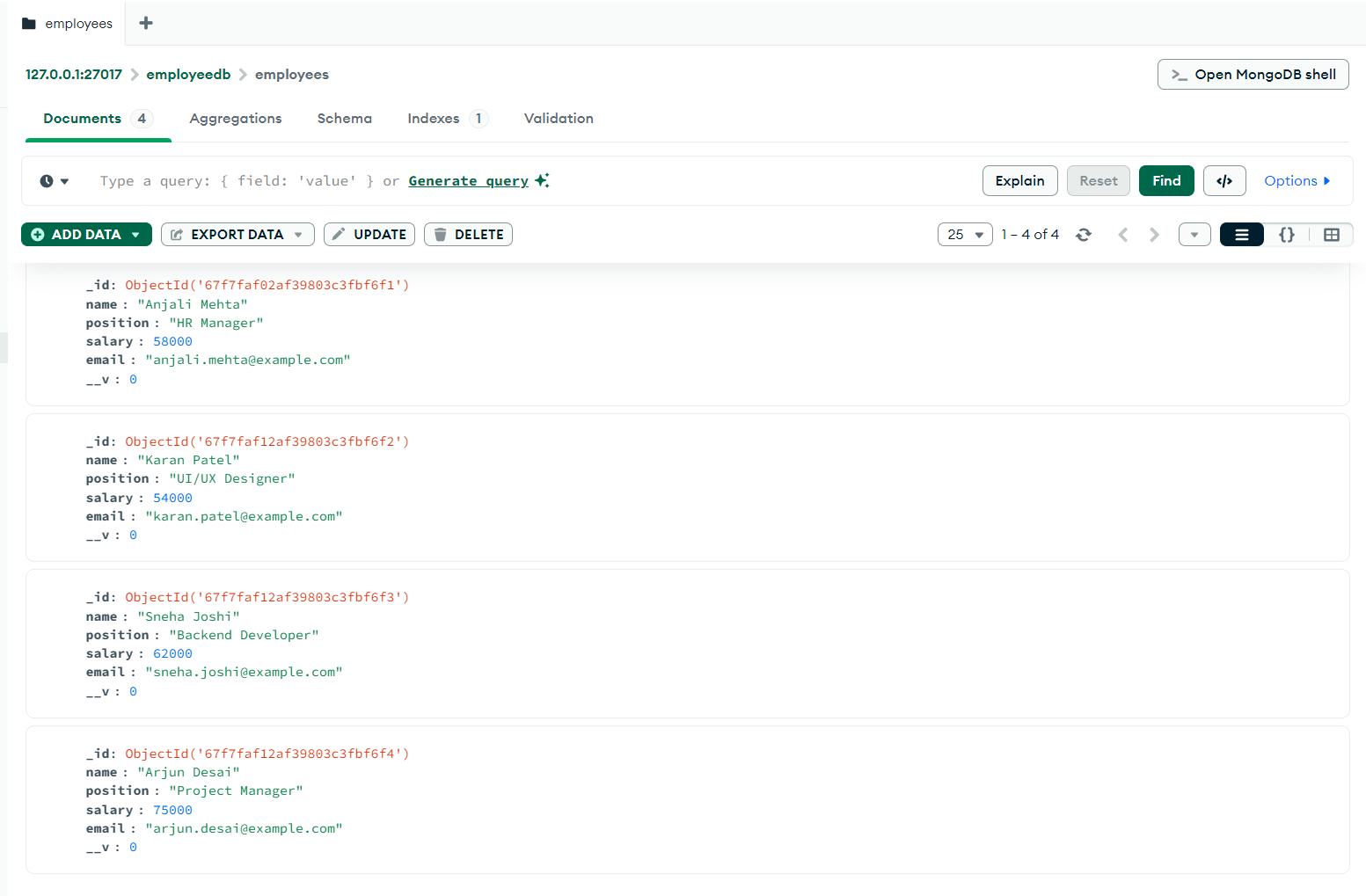
// Start server

app.listen(PORT, () => {

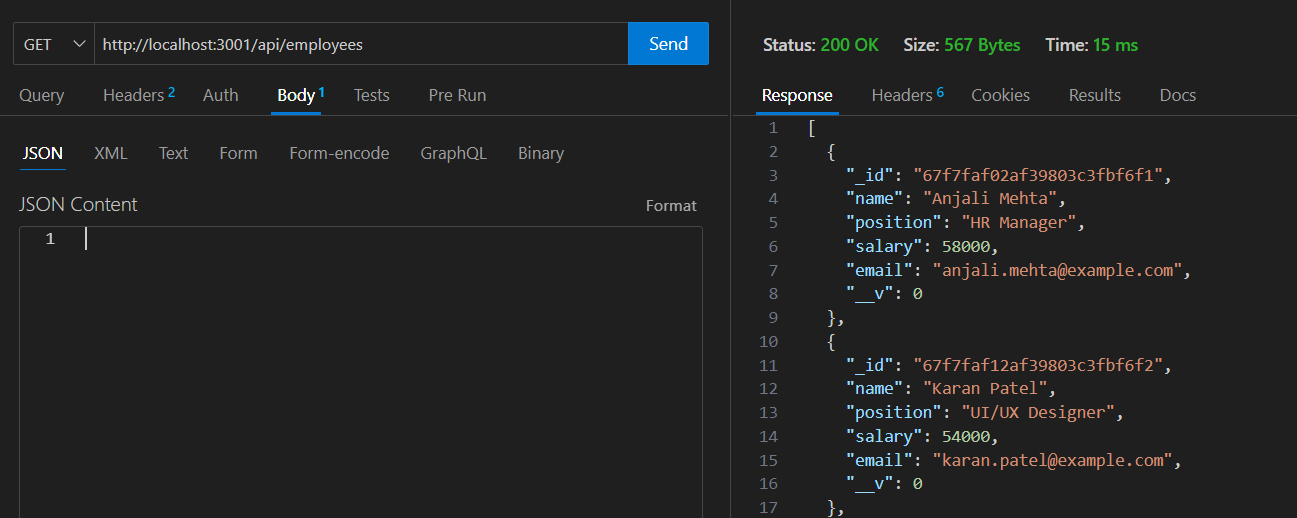
console.log(`🚀 Server running at http://localhost:${PORT}`);

});

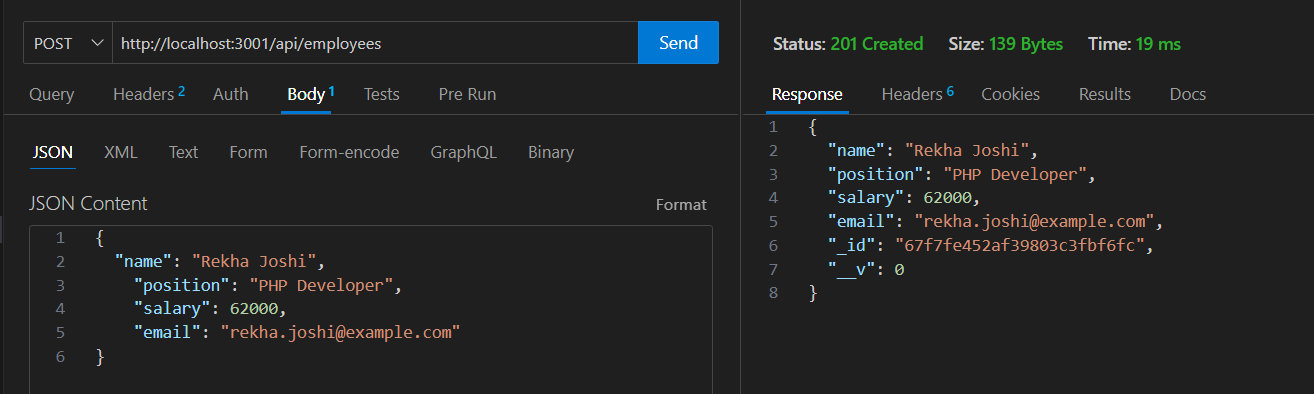
**OUTPUT :**



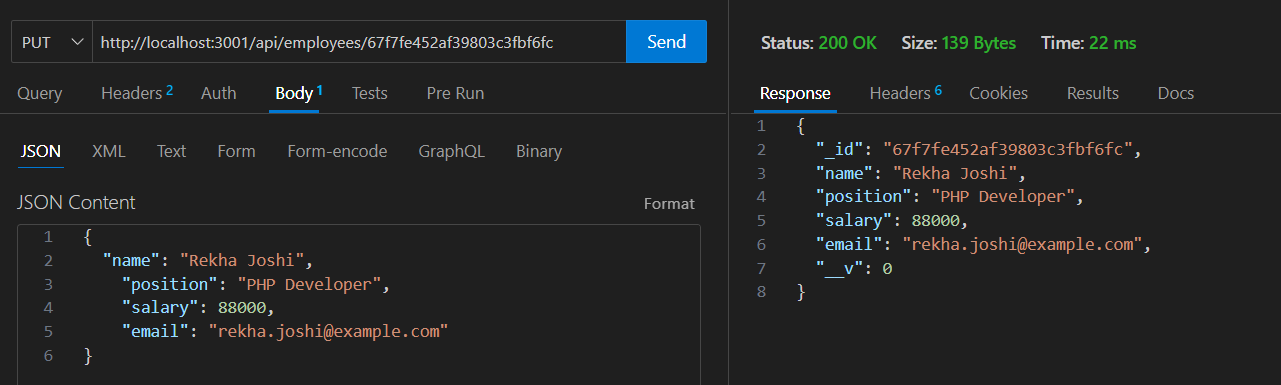
**GET**



**POST**



**PUT**



**DELETE**

