# CRUD

## Node.js + Express CRUD + MongoDB & EJS

**Folder structure**

Make **Folder** backend

* mkdir backend
* cd backend
* npm init -y
* npm install express mongoose dotenv ejs
* npm i -D nodemon
* nodemon server.js / npm run dev

backend/

│── node\_modules/

│── config/

│   └── db.js                # MongoDB connection

│

│── controllers/

│   └── contactController.js # CRUD logic

│

│── models/

│   └── contactModel.js      # Contact schema

│

│── routes/

│   └── contactRoutes.js     # API routes

│

│── views/

│   ├── index.ejs            # Home page (list + form)

│   └── edit.ejs             # Edit contact page

│

│── .env                     # Environment variables (PORT, DB URI)

│── package.json             # Project metadata & scripts

│── server.js                # Main entry file

│── package-lock.json

**Backend**

package.json

"scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node server.js",

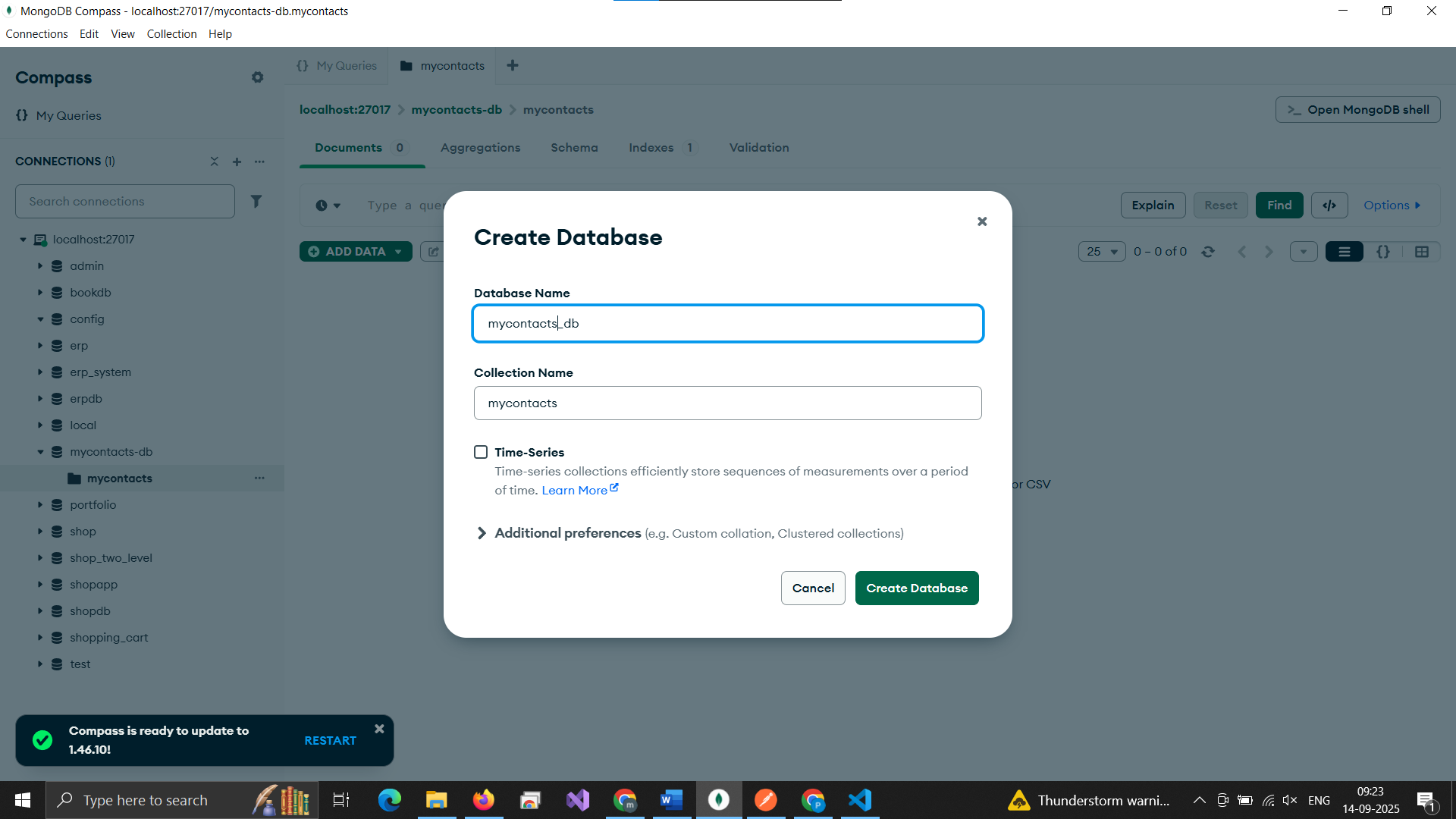
    "dev": "nodemon server.js"

  },

In mongoose

Click on connections (+) -> connect

Then click on the connection (+) e.g. **localhost:27017 (+)**



.env

PORT=3000

CONNECTION\_STRING=mongodb://localhost:27017/mycontacts-db

Make config folder

config -> db.js

const mongoose = require("mongoose");

const connectDb = async () => {

  try {

    const connect = await mongoose.connect(process.env.CONNECTION\_STRING);

    console.log(

      "Database connected:",

      connect.connection.host,

      connect.connection.name

    );

  } catch (error) {

    console.log(error);

    process.exit(1);

  }

};

module.exports = connectDb;

**Check connection**

server.js

const express = require('express');

const dotenv = require('dotenv');

const connectDb = require('./config/db');

dotenv.config();

connectDb();

const app = express();

const PORT = process.env.PORT || 5001;

app.listen(PORT, () => {

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

});

Make models folders

models -> contactModel.js

const mongoose = require("mongoose");

const contactShema = mongoose.Schema(

  {

    name: {

      type: String,

      required: [true, "Please add a name"],

    },

    email: {

      type: String,

      required: [true, "Please add an email"],

    },

    phone: {

      type: String,

      required: [true, "Please add a phone number"],

    },

  },

  {

    timestamps: true,

  }

);

module.exports = mongoose.model("Contact", contactShema);

Make controllers folder

controllers -> contactController.js

const Contact = require("../models/contactModel");

// @desc Get all contacts

const getContacts = async (req, res) => {

  try {

    const contacts = await Contact.find();

    res.status(200).json(contacts);

  } catch (error) {

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

  }

};

// @desc Get single contact

const getContact = async (req, res) => {

  try {

    const contact = await Contact.findById(req.params.id);

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

    res.status(200).json(contact);

  } catch (error) {

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

  }

};

// @desc Create contact

const createContact = async (req, res) => {

  try {

    const { name, email, phone } = req.body;

    if (!name || !email || !phone) {

      return res.status(400).json({ message: "All fields are required" });

    }

    const contact = await Contact.create({ name, email, phone });

    res.status(201).json(contact);

  } catch (error) {

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

  }

};

// @desc Update contact

const updateContact = async (req, res) => {

  try {

    const contact = await Contact.findById(req.params.id);

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

    const updatedContact = await Contact.findByIdAndUpdate(

      req.params.id,

      req.body,

      { new: true }

    );

    res.status(200).json(updatedContact);

  } catch (error) {

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

  }

};

// @desc Delete contact

const deleteContact = async (req, res) => {

  try {

    const contact = await Contact.findById(req.params.id);

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

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

    res.status(200).json({ message: "Contact deleted" });

  } catch (error) {

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

  }

};

module.exports = {

  getContacts,

  getContact,

  createContact,

  updateContact,

  deleteContact,

};

Make routes folder

routes -> contactRoutes.js

const express = require("express");

const router = express.Router();

const {

  getContacts,

  getContact,

  createContact,

  updateContact,

  deleteContact,

} = require("../controllers/contactController");

// CRUD Routes

router.get("/", getContacts);        // Read all

router.get("/:id", getContact);      // Read one

router.post("/", createContact);     // Create

router.put("/:id", updateContact);   // Update

router.delete("/:id", deleteContact); // Delete

module.exports = router;

in server.js

const express = require("express");

const dotenv = require("dotenv");

const connectDb = require("./config/db");

const contactRoutes = require("./routes/contactRoutes");

dotenv.config();

connectDb();

const app = express();

// Middleware to parse JSON

app.use(express.json());

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

// Routes

app.use("/api/contacts", contactRoutes);

const PORT = process.env.PORT || 5001;

app.listen(PORT, () => {

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

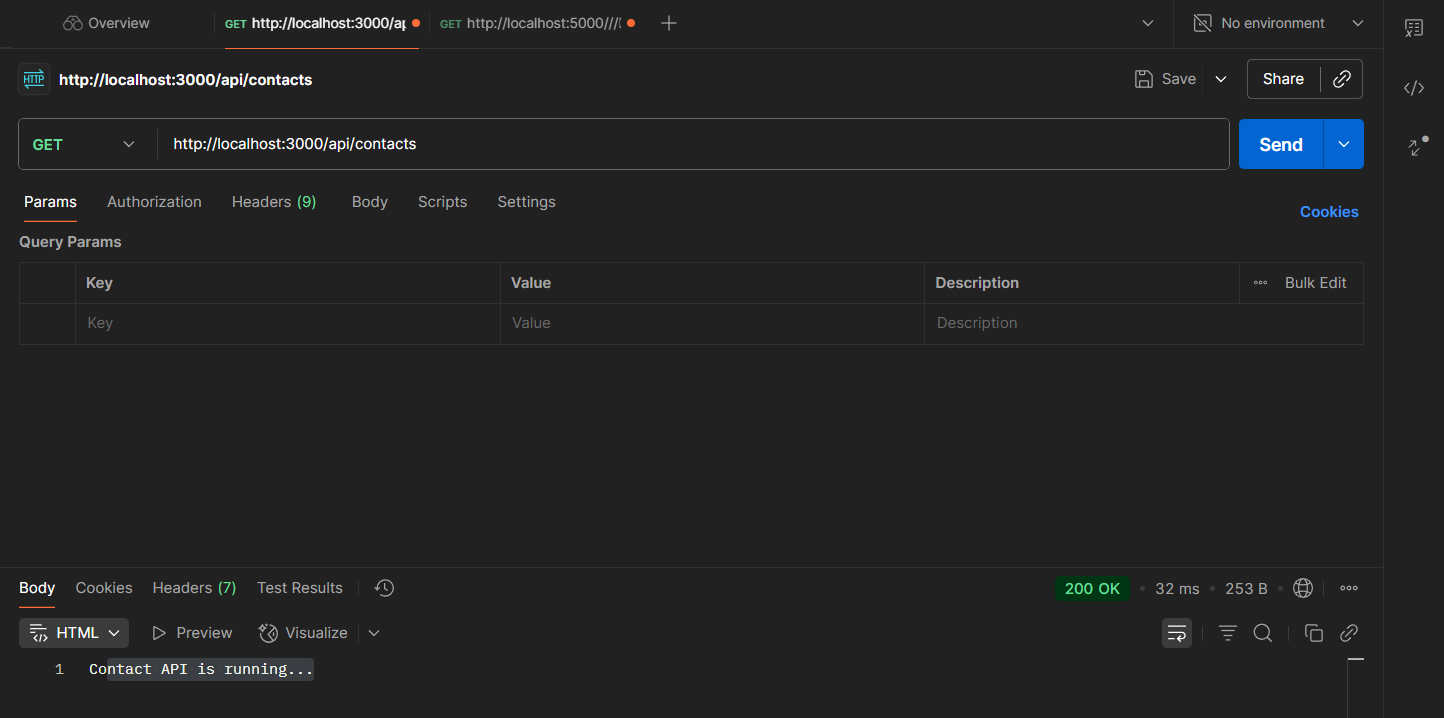
});

In postman check the URLs

**API Endpoints (Postman Testing)**

Base URL → <http://localhost:3000/api/contacts>

|  |  |
| --- | --- |
| Method | URLs |
| POST | http://localhost:3000/api/contacts |
| GET | http://localhost:3000/api/contacts |
| GET (single contact) | http://localhost:3000/api/contacts/:id |
| PUT | http://localhost:3000/api/contacts/:id |
| DELETE | http://localhost:3000/api/contacts/:id |



views -> index.ejs

<!DOCTYPE html>

<html>

  <head>

    <title>Contacts</title>

    <link

      rel="stylesheet"

      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"

    />

  </head>

  <body class="container mt-5">

    <h1 class="mb-4">Contacts</h1>

    <form action="/add" method="POST" class="mb-4">

      <div class="row">

        <div class="col">

          <input

            type="text"

            name="name"

            placeholder="Name"

            class="form-control"

            required

          />

        </div>

        <div class="col">

          <input

            type="email"

            name="email"

            placeholder="Email"

            class="form-control"

            required

          />

        </div>

        <div class="col">

          <input

            type="text"

            name="phone"

            placeholder="Phone"

            class="form-control"

            required

          />

        </div>

        <div class="col">

          <button type="submit" class="btn btn-primary">Add Contact</button>

        </div>

      </div>

    </form>

    <table class="table table-bordered">

      <thead>

        <tr>

          <th>Name</th>

          <th>Email</th>

          <th>Phone</th>

          <th>Actions</th>

        </tr>

      </thead>

      <tbody>

        <% contacts.forEach(contact => { %>

        <tr>

          <td><%= contact.name %></td>

          <td><%= contact.email %></td>

          <td><%= contact.phone %></td>

          <td>

            <a href="/edit/<%= contact.\_id %>" class="btn btn-warning btn-sm"

              >Edit</a

            >

            <a href="/delete/<%= contact.\_id %>" class="btn btn-danger btn-sm"

              >Delete</a

            >

          </td>

        </tr>

        <% }) %>

      </tbody>

    </table>

  </body>

</html>

edit.ejs

<!DOCTYPE html>

<html>

<head>

  <title>Edit Contact</title>

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="container mt-5">

  <h1 class="mb-4">Edit Contact</h1>

  <form action="/update/<%= contact.\_id %>" method="POST">

    <div class="mb-3">

      <label>Name</label>

      <input type="text" name="name" value="<%= contact.name %>" class="form-control" required>

    </div>

    <div class="mb-3">

      <label>Email</label>

      <input type="email" name="email" value="<%= contact.email %>" class="form-control" required>

    </div>

    <div class="mb-3">

      <label>Phone</label>

      <input type="text" name="phone" value="<%= contact.phone %>" class="form-control" required>

    </div>

    <button type="submit" class="btn btn-success">Update</button>

    <a href="/" class="btn btn-secondary">Cancel</a>

  </form>

</body>

</html>

server.js

// Set view engine

app.set("view engine", "ejs");

// Home page (list all contacts)

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

  const contacts = await Contact.find();

  res.render("index", { contacts });

});

// Add contact from form

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

  const { name, email, phone } = req.body;

  await Contact.create({ name, email, phone });

  res.redirect("/");

});

// Show edit form

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

  const contact = await Contact.findById(req.params.id);

  res.render("edit", { contact });

});

// Handle update

app.post("/update/:id", async (req, res) => {

  const { name, email, phone } = req.body;

  await Contact.findByIdAndUpdate(req.params.id, { name, email, phone });

  res.redirect("/");

});

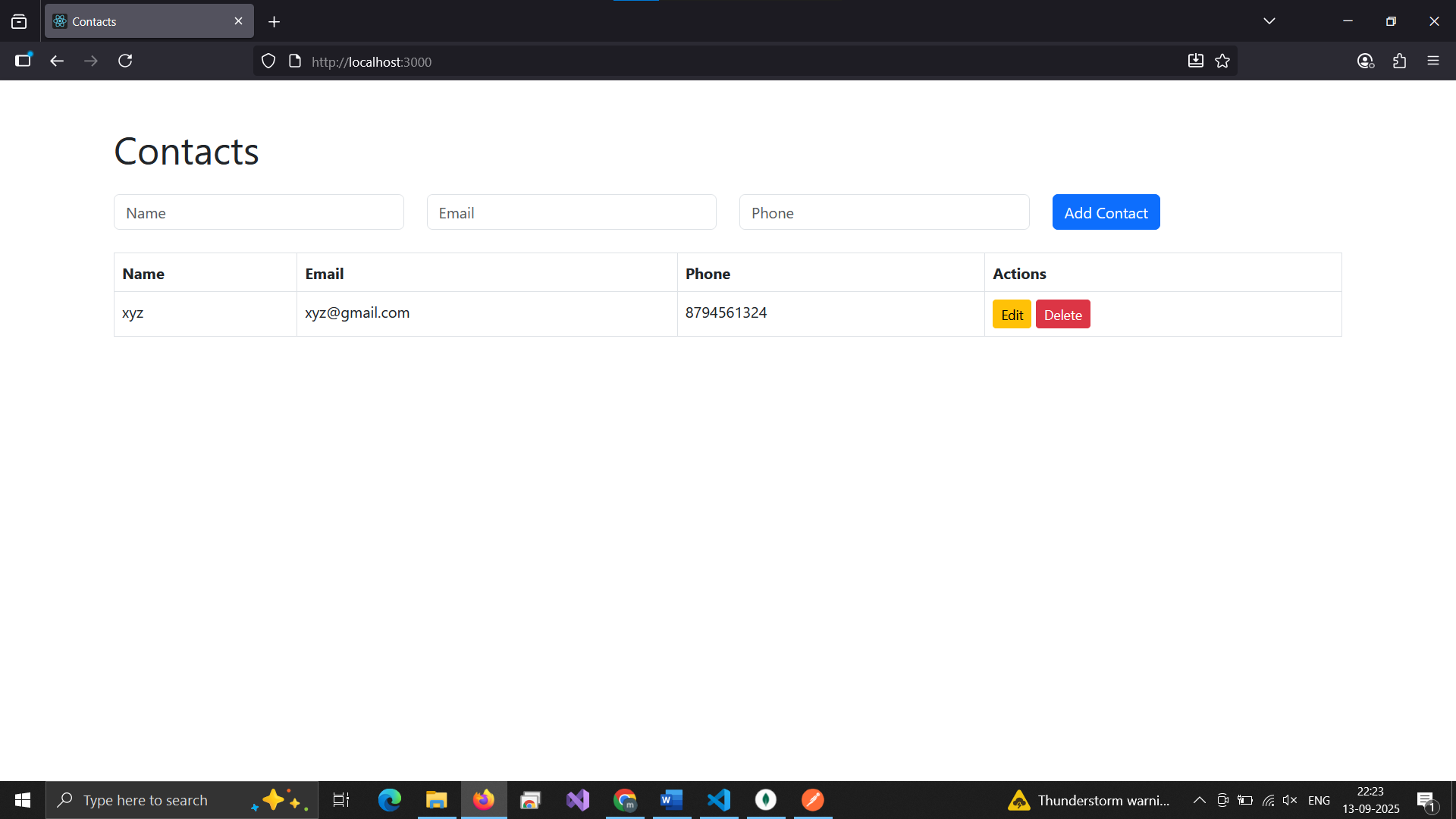
// Delete contact

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

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

  res.redirect("/");

});



## Static Cookie

**Step 1: Install cookie-parser**

npm install cookie-parser

**Step 2: Setup in cookieRoutes.js**

const express = require("express");

const router = express.Router();

// Set a cookie

router.get("/set-cookie", (req, res) => {

  res.cookie("username", "JohnDoe", { maxAge: 60000, httpOnly: true });

  res.send("Cookie has been set!");

});

// Get cookies

router.get("/get-cookie", (req, res) => {

  res.send(req.cookies);   // will show { username: "JohnDoe" }

});

// Clear cookie

router.get("/clear-cookie", (req, res) => {

  res.clearCookie("username");

  res.send("Cookie cleared!");

});

module.exports = router;

**step 3: in server.js**

const cookieParser = require("cookie-parser");

const cookieRoutes = require("./routes/cookieRoutes"); // 👈 add this

// Cookie Routes

app.use("/cookies", cookieRoutes); // 👈 mount cookies under /cookies

const PORT = process.env.PORT || 5001;

app.listen(PORT, () => {

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

});

**Step 4: Test in Browser/Postman**

Now you can hit these URLs:

* **Set cookie:**  
  http://localhost:3000/cookies/set-cookie
* **Get cookie:**  
  http://localhost:3000/cookies/get-cookie
* **Clear cookie:**  
  http://localhost:3000/cookies/clear-cookie

## Dynamic Cookie from URL Parameter

cookieRoutes.js

const express = require("express");

const router = express.Router();

// ✅ Set a cookie dynamically using URL param

router.get("/set-cookie/:username", (req, res) => {  //changes here :username

  const { username } = req.params;              //changes here Set cookie named 'username' with value from URL param

  res.cookie("username", username, { maxAge: 60000, httpOnly: true });

  res.send(`Cookie has been set! for ${username}`);

});

// Get cookies

router.get("/get-cookie", (req, res) => {

  res.send(req.cookies); //shows { username: "something" }

});

// Clear cookie

router.get("/clear-cookie", (req, res) => {

  res.clearCookie("username");

  res.send("Cookie cleared!");

});

module.exports = router;

server.js

const cookieParser = require("cookie-parser");

const cookieRoutes = require("./routes/cookieRoutes"); // 👈 add this

app.use(cookieParser()); // ✅ enable cookies

// Cookie Routes

app.use(cookieParser());             // ✅ must come before using cookies

app.use("/cookies", cookieRoutes); // 👈 mount cookies under /cookies

## Dynamic Cookie from Contact Form Submission

server.js

// Add contact from form and set cookie dynamically

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

  const { name, email, phone } = req.body;

  // Save to DB

  const contact = await Contact.create({ name, email, phone });

  // Set a dynamic cookie with contact name

  res.cookie("lastAddedContact", contact.name, {

    maxAge: 5 \* 60 \* 1000, // 5 minutes

    httpOnly: true,

  });

  res.redirect("/");

});

// Show edit form

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

  const contact = await Contact.findById(req.params.id);

  const lastAddedContact = req.cookies.lastAddedContact || null; // ✅ pass cookie

  res.render("edit", { contact, lastAddedContact });

});

// Handle update

app.post("/update/:id", async (req, res) => {

  const { name, email, phone } = req.body;

  await Contact.findByIdAndUpdate(req.params.id, { name, email, phone });

  // Update cookie if you want

  res.cookie("lastAddedContact", name, { maxAge: 5 \* 60 \* 1000, httpOnly: true });

  res.redirect("/");

});

Update Home Route to Read Cookie

// Home page (list all contacts)

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

  const contacts = await Contact.find();

  // ✅ Read cookie

  const lastAddedContact = req.cookies.lastAddedContact || null;

  res.render("index", { contacts, lastAddedContact });

});

Update views/index.ejs — Display Last Added Contact

     <h1 class="mb-4">Contacts</h1>

    <% if (typeof lastAddedContact !== 'undefined' && lastAddedContact) { %>

    <div class="alert alert-success">

      Last added contact: <strong><%= lastAddedContact %></strong>

    </div>

    <% } %>

