**Experiment 6**

**Student Name:** Rahul Saxena **UID:** 24MCI10204

**Branch:** MCA AI & ML **Section/Group:** 3-B

**Semester:** II  **Date of Performance:** 28/03/2025

**Subject Name:** Advanced Internet Programming Lab **Subject Code:** 24CAP-652

**Aim/Overview of the practical:** Create and consume Restful web services for accessing employee data application securely.

**Task to be done:**

**Set Up Environment**

* Install Node.js and necessary packages (Express.js, Mongoose, Body-parser, CORS).
* Set up MongoDB Atlas or MySQL as the database.

**Create Project Structure**

* Organize files into routes, models, controllers, and config.

**Establish Database Connection**

* Connect Node.js to MongoDB using Mongoose or MySQL using mysql2.

**Implement CRUD Operations**

* **Create**: Add a new record to the database.
* **Read**: Fetch all or specific records from the database.
* **Update**: Modify existing records.
* **Delete**: Remove records from the database.

**Set Up API Routes**

* Use Express.js to handle API endpoints for CRUD operations.

**Code for experiment/practical:**

**Index.html**Top of Form

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>MongoDB User List</title>

    <link rel="stylesheet" href="style.css" />

  </head>

  <body>

    <h1>MongoDB User List</h1>

    <input type="text" id="name" placeholder="Enter Name" />

    <input type="email" id="email" placeholder="Enter Email" />

    <button onclick="addUser()">Add User</button>

    <button onclick="updateUser()" id="updateBtn" style="display: none">

      Update User

    </button>

    <h2>Users:</h2>

    <table>

      <thead>

        <tr>

          <th>Name</th>

          <th>Email</th>

          <th>Actions</th>

        </tr>

      </thead>

      <tbody id="userTable"></tbody>

    </table>

    <script>

      let editingUserId = null;

      async function fetchUsers() {

        const response = await fetch("/get-users");

        const users = await response.json();

        document.getElementById("userTable").innerHTML = users

          .map(

            (user) =>

              `<tr>

                        <td>${user.name}</td>

                        <td>${user.email}</td>

                        <td class="actions">

                            <button class="edit" onclick="editUser('${user.\_id}', '${user.name}', '${user.email}')">Edit</button>

                            <button class="delete" onclick="deleteUser('${user.\_id}')">Delete</button>

                        </td>

                    </tr>`

          )

          .join("");

      }

      async function addUser() {

        const name = document.getElementById("name").value;

        const email = document.getElementById("email").value;

        if (!name || !email) {

          alert("Please enter both name and email.");

          return;

        }

        await fetch("/add-user", {

          method: "POST",

          headers: { "Content-Type": "application/json" },

          body: JSON.stringify({ name, email }),

        });

        document.getElementById("name").value = "";

        document.getElementById("email").value = "";

        fetchUsers();

      }

      async function deleteUser(id) {

        await fetch(`/delete-user/${id}`, { method: "DELETE" });

        fetchUsers();

      }

      function editUser(id, name, email) {

        document.getElementById("name").value = name;

        document.getElementById("email").value = email;

        editingUserId = id;

        document.getElementById("updateBtn").style.display = "inline-block";

      }

      async function updateUser() {

        const name = document.getElementById("name").value;

        const email = document.getElementById("email").value;

        if (!name || !email) {

          alert("Please enter both name and email.");

          return;

        }

        await fetch(`/update-user/${editingUserId}`, {

          method: "PUT",

          headers: { "Content-Type": "application/json" },

          body: JSON.stringify({ name, email }),

        });

        document.getElementById("name").value = "";

        document.getElementById("email").value = "";

        document.getElementById("updateBtn").style.display = "none";

        editingUserId = null;

        fetchUsers();

      }

      window.onload = fetchUsers;

    </script>

  </body>

</html>

**Server.js**

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const app = express();

const PORT = 3000;

// MongoDB Connection

const mongoURI = "mongodb+srv://saxenaa332:iDjFy0RktXGwSwcP@experiment5.sn4mj.mongodb.net";

mongoose.connect(mongoURI, { useNewUrlParser: true, useUnifiedTopology: true })

  .then(() => console.log("MongoDB connected"))

  .catch(err => console.log(err));

app.use(cors());

app.use(express.json());

app.use(express.static("public"));

const UserSchema = new mongoose.Schema({

  name: String,

  email: String

});

const User = mongoose.model("User", UserSchema);

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

  res.sendFile(\_\_dirname + "/public/index.html");

});

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

  const users = await User.find();

  res.json(users);

});

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

  const { name, email } = req.body;

  const newUser = new User({ name, email });

  await newUser.save();

  res.json({ message: "User added successfully!" });

});

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

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

  res.json({ message: "User deleted successfully!" });

});

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

  const { name, email } = req.body;

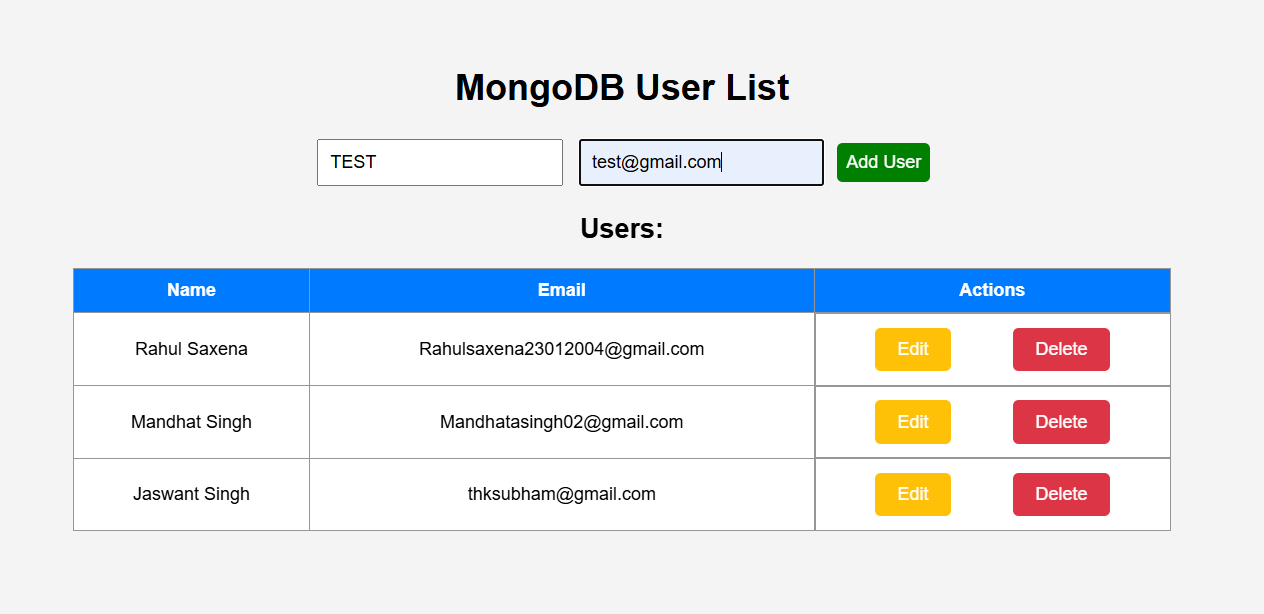
  await User.findByIdAndUpdate(req.params.id, { name, email });

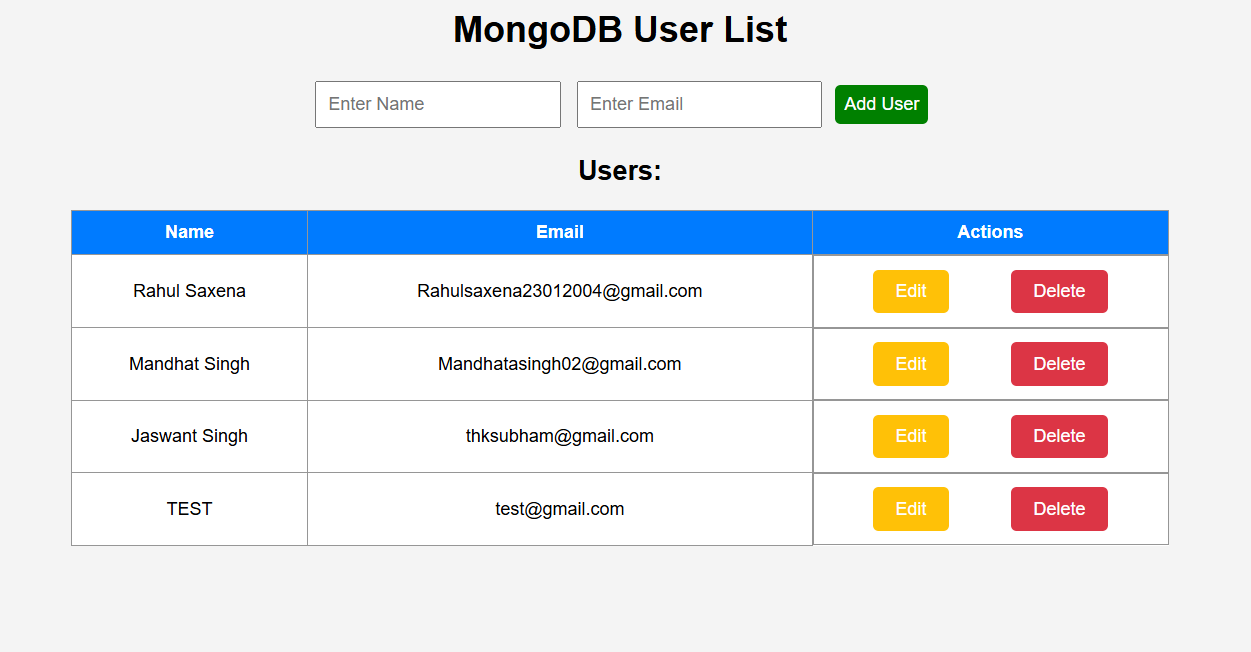
  res.json({ message: "User updated successfully!" });

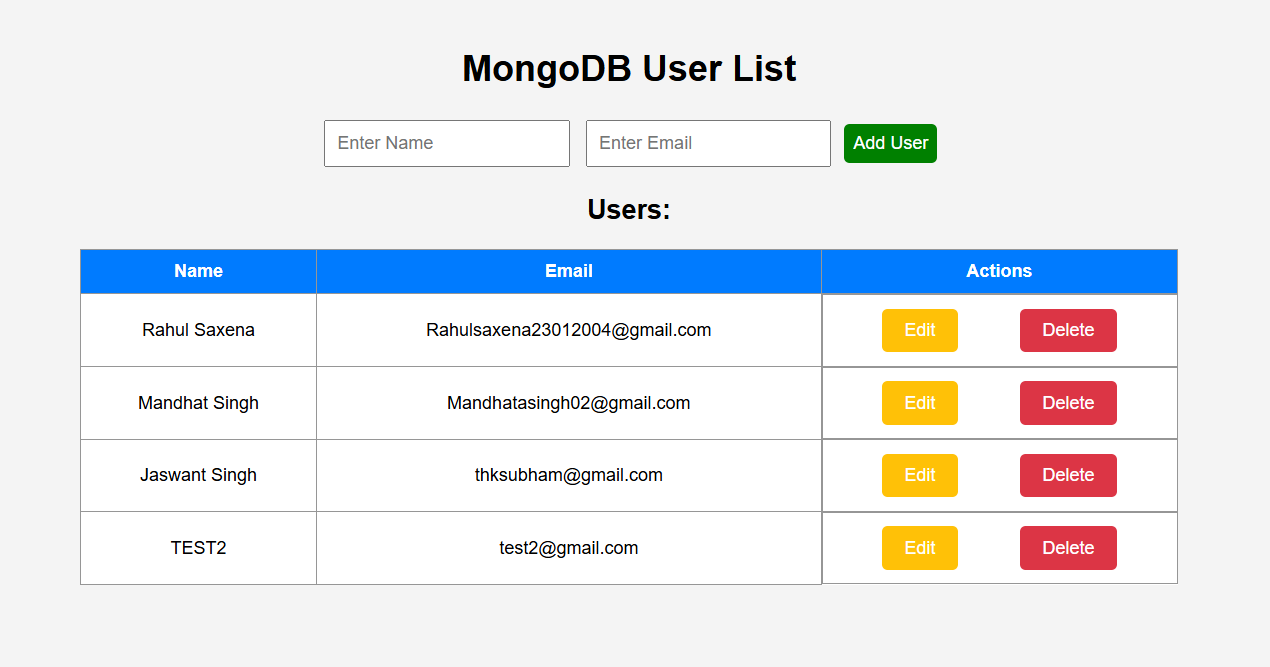
});

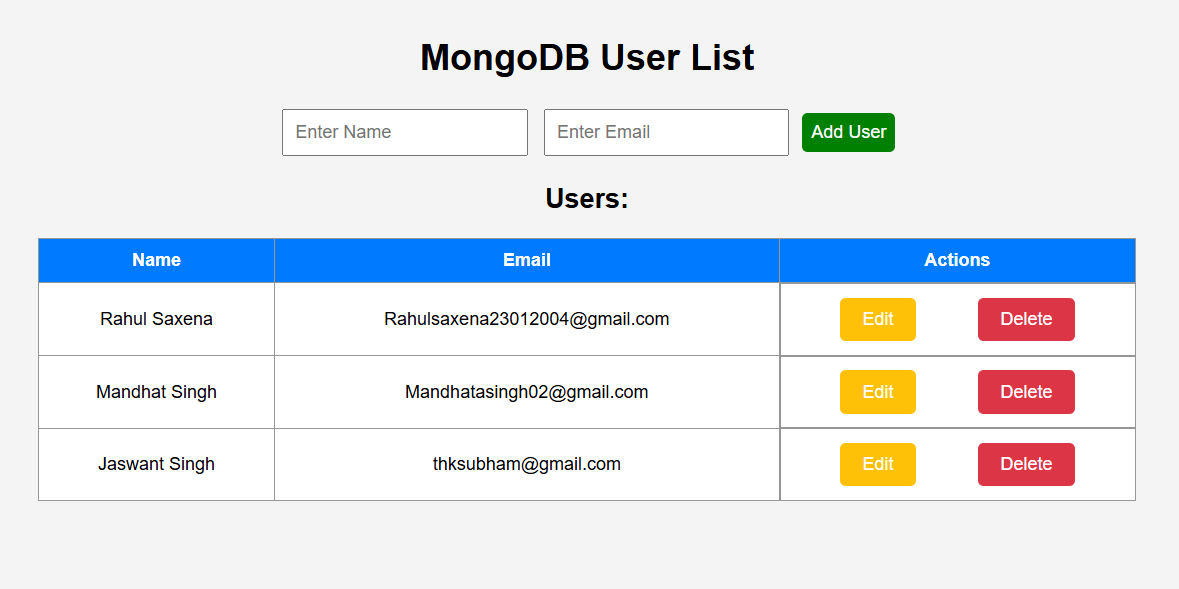
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));

**Output:**

**CREATE**

**READ**

**Update**

**Delete**

**Learning outcomes:**

* **Understand Node.js and Express.js**
  + Learn how to create a backend using Express.js.
* **Database Connectivity**
  + Learn how to connect Node.js with MongoDB (Mongoose) or MySQL.
* **RESTful API Development**
  + Learn to create API routes for handling requests.
* **Perform CRUD Operations**
  + Gain hands-on experience in database operations.
* **Error Handling & Validation**
  + Learn to handle request errors and validate input data.

**Evaluation Grid:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Demonstration and Performance |  | 5 |
| 2. | Worksheet |  | 10 |
| 3. | Post Lab Quiz |  | 5 |