

Name: Krutika Salve

Roll no: TIB68

Practical 3B

App.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import 'bootstrap/dist/css/bootstrap.min.css';
function App() {
  const [formData, setFormData] = useState({ name: "", email: "", age: "" });
  const [users, setUsers] = useState([]);
  const [editId, setEditId] = useState(null);

  const fetchUsers = async () => {
    const res = await axios.get("http://localhost:5000/api/users");
    setUsers(res.data);
  };
  useEffect(() => {
    fetchUsers();
  }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!formData.name || !formData.email || !formData.age) {
      alert("Please fill all fields!");
      return;
    }
    if (editId) {
      await axios.put(`http://localhost:5000/api/users/${editId}`, formData);
      setEditId(null);
    } else {
      await axios.post("http://localhost:5000/api/users", formData);
    }
    setFormData({ name: "", email: "", age: "" });
    fetchUsers();
  };
  const handleEdit = (user) => {
    setFormData({ name: user.name, email: user.email, age: user.age });
    setEditId(user._id);
  };
  const handleDelete = async (id) => {
    await axios.delete(`http://localhost:5000/api/users/${id}`);
    fetchUsers();
  };
};
```

```

return (
  <div className="container mt-5">
    <div className="card shadow-lg">
      <div className="card-header bg-primary text-white">
        <h3 className="text-center mb-0">{editId ? "Edit User" : "Add New User"}</h3>
      </div>
      <div className="card-body">
        <form onSubmit={handleSubmit} className="row g-3">
          <div className="col-md-4">
            <input
              type="text"
              className="form-control"
              placeholder="Enter Name"
              value={formData.name}
              onChange={(e) => setFormData({ ...formData, name: e.target.value })}
            />
          </div>
          <div className="col-md-4">
            <input
              type="email"
              className="form-control"
              placeholder="Enter Email"
              value={formData.email}
              onChange={(e) => setFormData({ ...formData, email: e.target.value })}
            />
          </div>
          <div className="col-md-2">
            <input
              type="number"
              className="form-control"
              placeholder="Age"
              value={formData.age}
              onChange={(e) => setFormData({ ...formData, age: e.target.value })}
            />
          </div>
          <div className="col-md-2 d-grid">
            <button type="submit" className="btn btn-success">
              {editId ? "Update" : "Add"}
            </button>
          </div>
        </form>
      </div>
    </div>
    <div className="mt-4 card shadow-sm">
      <div className="card-header bg-dark text-white">
        <h4 className="mb-0">User List</h4>
      </div>

```

```

<div className="card-body p-0">
  <table className="table table-striped mb-0">
    <thead className="table-dark">
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Age</th>
        <th style={{ width: "150px" }}>Actions</th>
      </tr>
    </thead>
    <tbody>
      {users.length === 0 ? (
        <tr>
          <td colspan="4" className="text-center py-3">No users found</td>
        </tr>
      ) : (
        users.map((user) => (
          <tr key={user._id}>
            <td>{user.name}</td>
            <td>{user.email}</td>
            <td>{user.age}</td>
            <td>
              <button
                className="btn btn-warning btn-sm me-2"
                onClick={() => handleEdit(user)}
              >
                Edit
              </button>
              <button
                className="btn btn-danger btn-sm"
                onClick={() => handleDelete(user._id)}
              >
                Delete
              </button>
            </td>
          </tr>
        ))
      )}
    </tbody>
  </table>
</div>
</div>
</div>
);
}
export default App;

```

server.js

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const userRoutes = require("./routes/userRoutes");

const app = express();
app.use(cors());
app.use(express.json());

mongoose.connect("mongodb://127.0.0.1:27017/assignment2c", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

app.use("/api/users", userRoutes);

app.get("/", (req, res) => {
  res.send("API is running... Use /api/users for CRUD operations.");
});

app.listen(5000, () => {
  console.log("Server started on http://localhost:5000");
});
```

userRoutes.js

```
const express = require("express");
const router = express.Router();
const User = require("../models/User");

router.post("/", async (req, res) => {
  try {
    const user = new User(req.body);
    await user.save();
    res.status(201).json(user);
  } catch (err) {
    res.status(500).json({ message: "Error saving user", error: err });
  }
});

router.get("/", async (req, res) => {
  try {
```

```

    const users = await User.find();
    res.json(users);
  } catch (err) {
    res.status(500).json({ message: "Error fetching users", error: err });
  }
});

router.put("/:id", async (req, res) => {
  try {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.send(user);
  } catch (err) {
    res.status(500).json({ message: "Error updating user", error: err });
  }
});

router.delete("/:id", async (req, res) => {
  try {
    await User.findByIdAndDelete(req.params.id);
    res.send({ message: "User deleted" });
  } catch (err) {
    res.status(500).json({ message: "Error deleting user", error: err });
  }
});
module.exports = router;

```

User.js

```

const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  age: Number,
});
module.exports = mongoose.model("User", userSchema);

```

OUTPUT

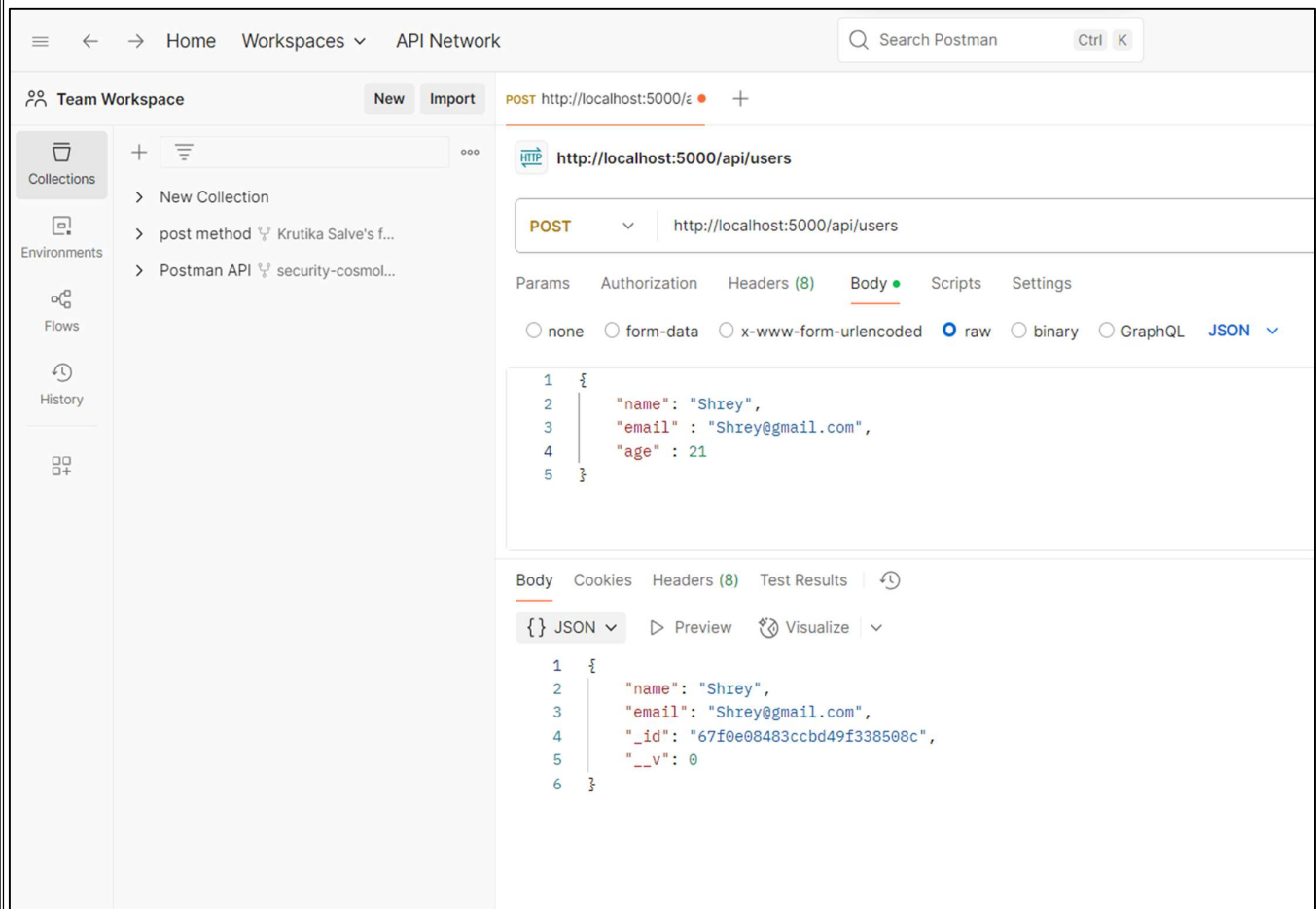
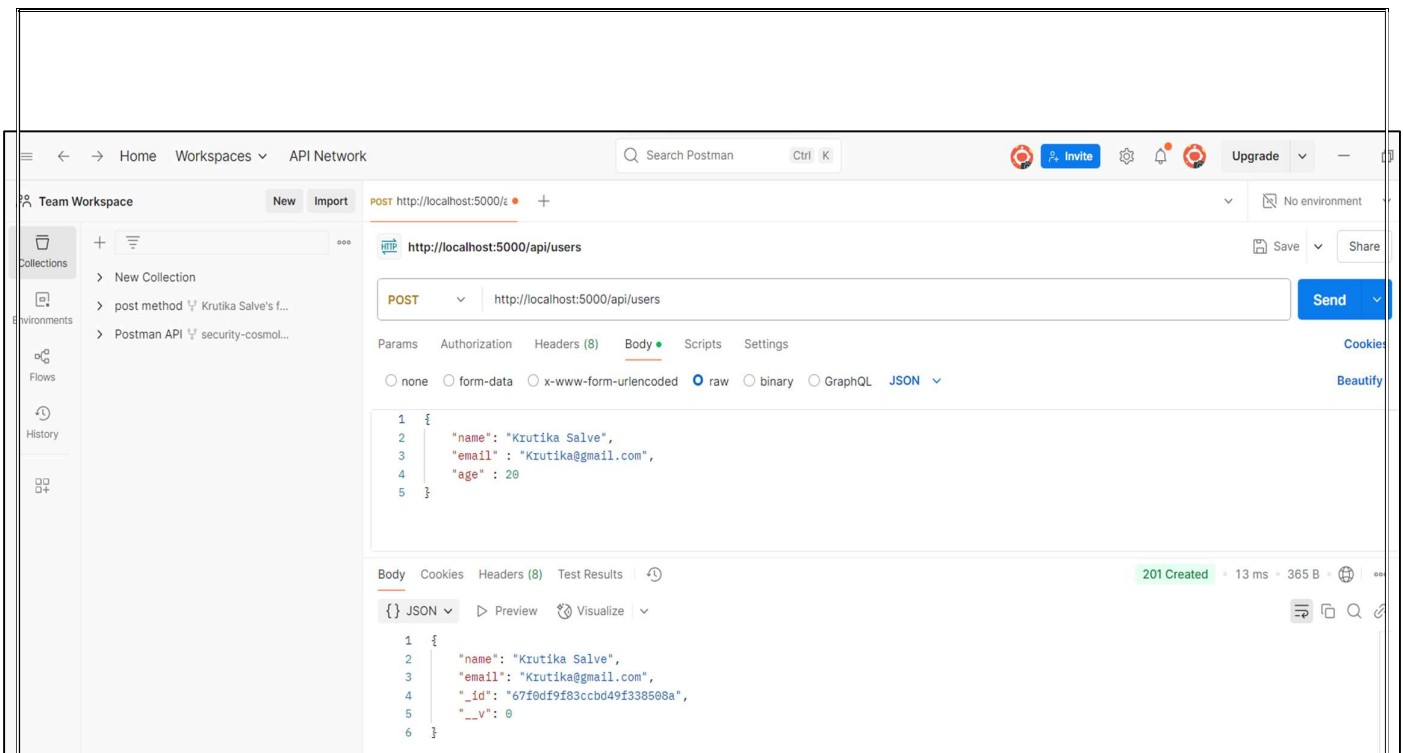
Test on Postman

1. **Post Method** – <http://localhost:5000/api/users>

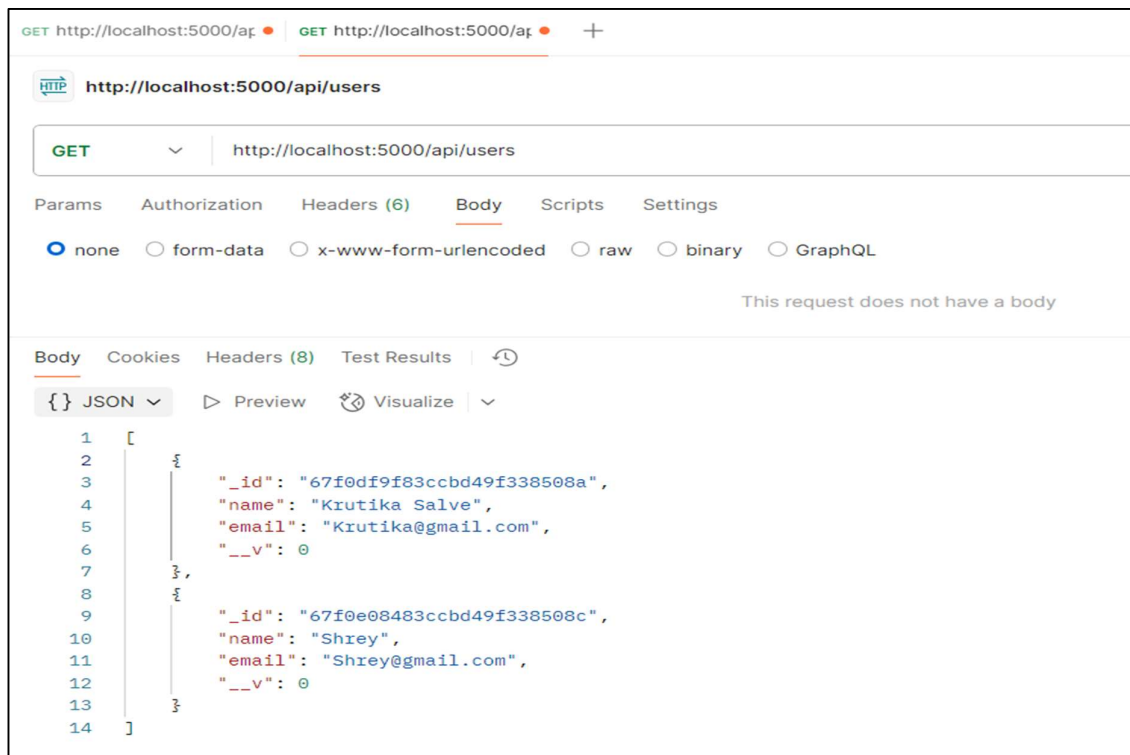
Select – Body

Select - raw

Select - JSON



1. Get Method -- <http://localhost:5000/api/users>



GET http://localhost:5000/api/users

GET http://localhost:5000/api/users

Params Authorization Headers (6) **Body** Scripts Settings

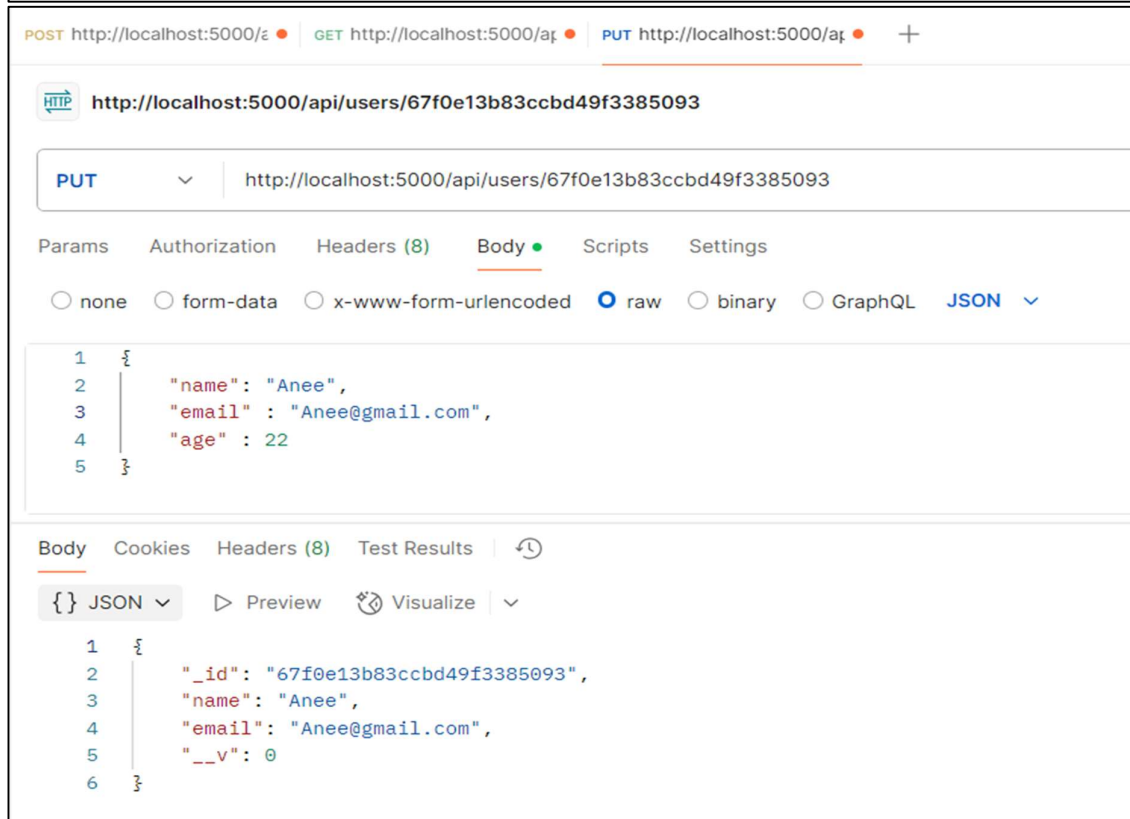
☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1 [
2   {
3     "_id": "67f0df9f83ccbd49f338508a",
4     "name": "Krutika Salve",
5     "email": "Krutika@gmail.com",
6     "__v": 0
7   },
8   {
9     "_id": "67f0e08483ccbd49f338508c",
10    "name": "Shrey",
11    "email": "Shrey@gmail.com",
12    "__v": 0
13  }
14 ]
```



POST http://localhost:5000/ GET http://localhost:5000/api/ PUT http://localhost:5000/api/

PUT http://localhost:5000/api/users/67f0e13b83ccbd49f3385093

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "name": "Anee",
3   "email": "Anee@gmail.com",
4   "age": 22
5 }
```

Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1 {
2   "_id": "67f0e13b83ccbd49f3385093",
3   "name": "Anee",
4   "email": "Anee@gmail.com",
5   "__v": 0
6 }
```

1. Put Method - <http://localhost:5000/api/users/<user-id>>

POST http://localhost:5000/ GET http://localhost:5000/api/users PUT http://localhost:5000/api/users

HTTP http://localhost:5000/api/users

GET http://localhost:5000/api/users

Params Authorization Headers (6) Body Scripts Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

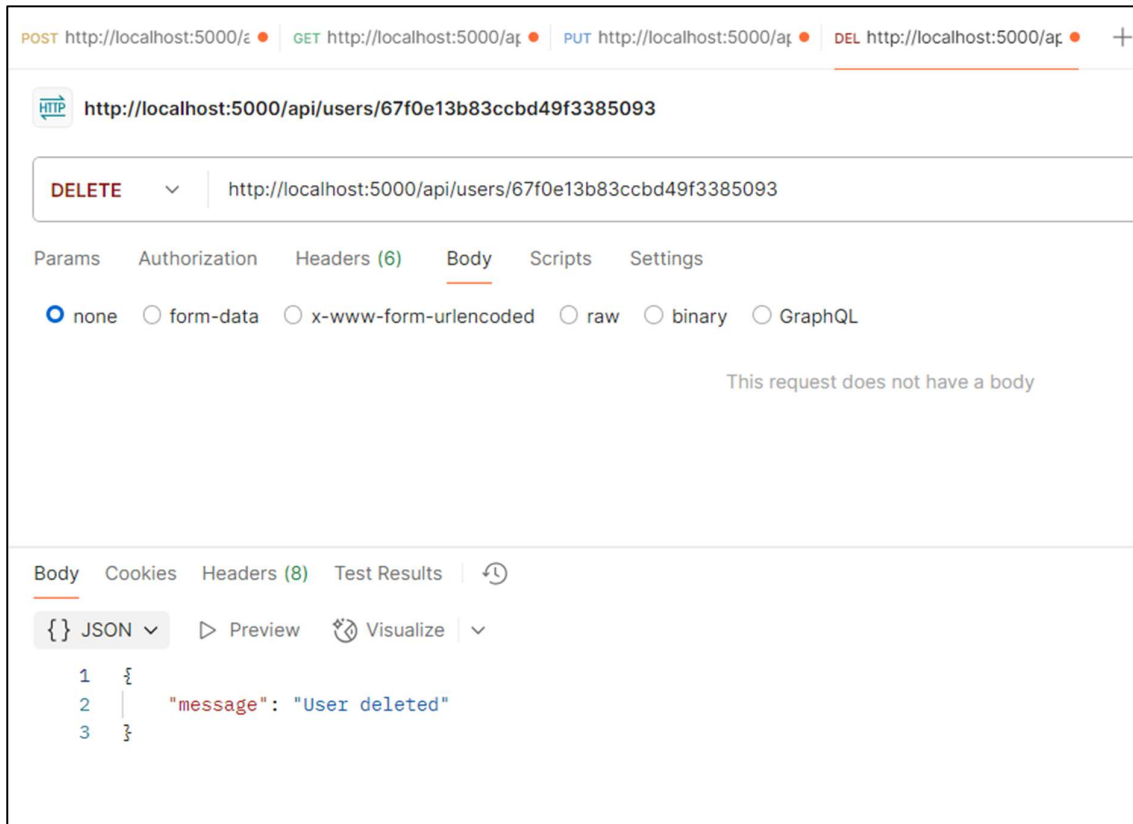
This request does not have a body

Body Cookies Headers (8) Test Results

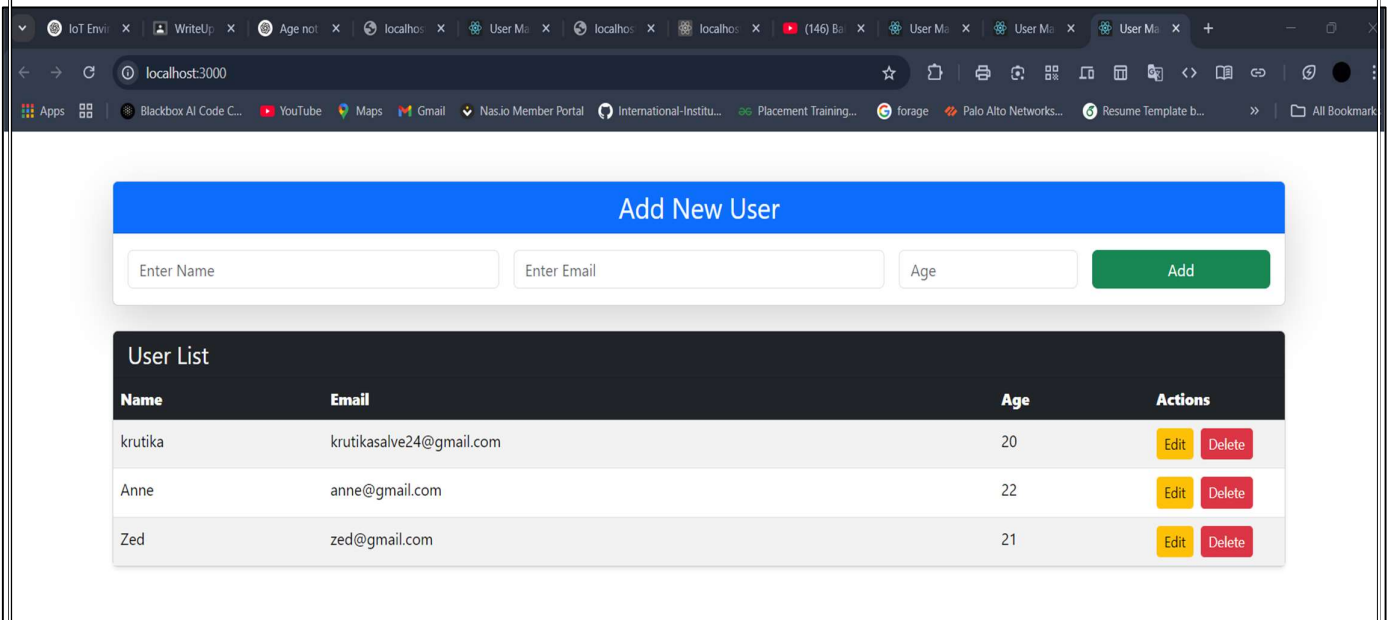
{ } JSON Preview Visualize

```
1  [
2    {
3      "_id": "67f0df9f83ccbd49f338508a",
4      "name": "Krutika Salve",
5      "email": "Krutika@gmail.com",
6      "__v": 0
7    },
8    {
9      "_id": "67f0e08483ccbd49f338508c",
10     "name": "Shrey",
11     "email": "Shrey@gmail.com",
12     "__v": 0
13   },
14   {
15     "_id": "67f0e13b83ccbd49f3385093",
16     "name": "Anee",
17     "email": "Anee@gmail.com",
18     "__v": 0
19   }
20 ]
```

4. Delete Method -- <http://localhost:5000/api/users/<user-id>>



Test on Browser using localhost:port



MongoDB Compass - localhost:27017/assignment2c.users

Connections Edit View Collection Help

Compass

{ } My Queries

CONNECTIONS (10)

Search connections

- localhost:27017
- localhost:27017
- localhost:27017
 - App
 - admin
 - assignment2c
 - users**
 - config
 - local
 - mernDB
 - mern_app
 - users
 - mydatabase
- localhost:27017
- localhost:27017

localhost:27017 > assignment2c > users

Documents 3 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

ADD DATA EXPORT DATA UPDATE DELETE

```
{
  "_id": ObjectId('67f0ebf039424fb3313e7db7'),
  "name": "krutika",
  "email": "krutikasalve24@gmail.com",
  "age": 20,
  "__v": 0
}
```

```
{
  "_id": ObjectId('67f0ec5639424fb3313e7dba'),
  "name": "Anne",
  "email": "anne@gmail.com",
  "age": 22,
  "__v": 0
}
```

```
{
  "_id": ObjectId('67f0ec6839424fb3313e7dbd'),
  "name": "Zed",
  "email": "zed@gmail.com",
  "age": 21,
  "__v": 0
}
```