CRUD PROJECT

Objective

To develop a web application that allows users to perform Create, Read, Update, and Delete operations on user data.

Technologies Used

- Frontend: HTML, CSS, JavaScript (Optionally React)

- Backend: Node.js with Express.js

- Database: MongoDB (Mongoose)

- Tools: VS Code, Postman

Functionalities

- Add new users (Create)

- List all users (Read)

- Edit user information (Update)

- Remove users (Delete)

Implementation Steps

1. Database Schema (User Model)

javascript

```
// models/User.js

const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({

  name: String,
  email: String,});
module.exports = mongoose.model('User', UserSchema);
```

2. Backend: Express Server & Routes

javascript

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const User = require('./models/User');
const cors = require('cors');
const app = express();
app.use(cors());
app.use(express.json());
mongoose.connect('mongodb://localhost:27017/crudapp');
// Create
app.post('/users', async (req, res) => {
   const user = new User(req.body);
   await user.save();
   res.send(user);});
// Read
app.get('/users', async (req, res) => {
   const users = await User.find();
   res.send(users);});
```

// Update

```
app.put('/users/:id', async (req, res) => {
  const user = await User.findByIdAndUpdate(req.params.id,
req.body, { new: true });
  res.send(user);});
```

// Delete

```
app.delete('/users/:id', async (req, res) => {
  await User.findByIdAndDelete(req.params.id);
  res.send({ message: 'User deleted' });});
app.listen(5000, () => console.log('Server running on port 5000'));
```

3. Frontend Example (HTML/JS Fetch Example)

```
html
<!– index.html -->
<form id="userForm">
  <input type="text" id="name" placeholder="Name" required>
  <input type="email" id="email" placeholder="Email" required>
  <button type="submit">Add User</button>
</form>
<ul id="userList"></ul>
<script>
  async function fetchUsers() {
    const res = await fetch('http://localhost:5000/users');
```

```
        const users = await res.json();
        const userList = document.getElementById('userList');
        userList.innerHTML = '';
        users.forEach(user => {
            const li = document.createElement('li');
            li.textContent = user.name + ' - ' + user.email;
            // Add edit and delete buttons as needed
            userList.appendChild(li);
        });
    }
    document.getElementById('userForm').addEventListener('submit',
async (e) => {
        e.preventDefault();
        const name = document.getElementById('name').value;
        const email = document.getElementById('email').value;
        await fetch('http://localhost:5000/users', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({ name, email })
        });
        fetchUsers();
    });
```

```
  fetchUsers();
</script>
```

How It Works

- Create: Fill out the form and submit to add a user.

- Read: The user list is fetched from the backend and displayed.

- Update: Implement edit button in each list item to update info.

- Delete: Add delete button in each list item to remove user.

Conclusion

This structure gives the complete code and workflow to implement a basic CRUD application managing user records.

THANK YOU

BY:

S. SEMBARUTHI