**//1. Write React Js code to Create Simple form with states management**

**App.js**

import React from 'react';

import {useState} from 'react';

const App = () => {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const handleSubmit = (e) => {

e.preventDefault(); // Prevent page reload

alert(`Form Submitted with: \nName: ${name} \nEmail: ${email}`);

};

return (

<div>

<h1>Simple React Form</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Name:</label>

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

placeholder="Enter your name"

/>

</div>

<div>

<label>Email:</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

placeholder="Enter your email"

/>

</div>

<div>

<button type="submit">Submit</button>

</div>

</form>

</div>

);

};

export default App;

**// 2. Write ReactJs code for Client-side form validation.**

 \S+: One or more non-whitespace characters.

 @: The literal @ symbol.

 \S+: One or more non-whitespace characters after the @.

 \.: A literal dot ..

 \S+: One or more non-whitespace characters after the dot.

import React from 'react';

import FormComponent from './FormComponent';

import react, {useState} from 'react';

const App = () => {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const [errors, setErrors] = useState({});

const validateForm = () => {

const formErrors = {};

if (!name) formErrors.name = 'Name is required';

if (!email) formErrors.email = 'Email is required';

if (!/\S+@\S+\.\S+/.test(email)) formErrors.email = 'Email is invalid';

setErrors(formErrors);

return Object.keys(formErrors).length === 0;

};

const handleSubmit = (e) => {

e.preventDefault(); // Prevent page reload

if (validateForm()) {

alert(`Form Submitted with: \nName: ${name} \nEmail: ${email}`);

}

};

return (

<div>

<h1>Simple React Form</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Name:</label>

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

placeholder="Enter your name"

/>

{errors.name && <p style={{ color: 'red' }}>{errors.name}</p>}

</div>

<div>

<label>Email:</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

placeholder="Enter your email"

/>

{errors.email && <p style={{ color: 'red' }}>{errors.email}</p>}

</div>

<div>

<button type="submit">Submit</button>

</div>

</form>

</div>

);

};

export default App;

**// 3. Write ReactJs code for Applying form components.**

1. Header.js 2. Sider.js 3. MainContent.js 4. Footer.js 5. App.css 6. App.js

1. Header.js

import React from 'react';

const Header = () => {

return (

<header className="header">

<h1>Simple Web Page Layout</h1>

</header>

);

};

export default Header;

1. Sider.js

import React from 'react';

const Sider = () => {

return (

<div className="sider">

<h3>Sidebar</h3>

<ul>

Links here

</ul>

</div>

);

};

export default Sider;

1. MainContent.js

import React from 'react';

const MainContent = () => {

return (

<div className="main-content">

<h2>Main Content</h2>

<p>This is the main content area. You can place any content you like here, such as articles, images, or videos.</p>

</div>

);

};

export default MainContent;

1. Footer.js

import React from 'react';

const Footer = () => {

return (

<footer className="footer">

<h1>Marathe Creation @ Company</h1>

</footer>

);

};

export default Footer;

1. App.css

/\* General layout styling \*/

.container {

display: flex;

flex-direction: column;

height: 100vh;

}

/\* Header Styling \*/

.header {

background-color: #4caf50;

color: white;

text-align: center;

padding: 15px;

font-size: 20px;

height:150px;

}

/\* Content Container Styling \*/

.content-container {

display: flex;

flex: 1;

overflow: hidden;

}

/\* Sider (Sidebar) Styling \*/

.sider {

background-color: #f4f4f4;

width: 250px;

padding: 15px;

border-right: 1px solid #ddd;

box-shadow: inset -1px 0 0 #ddd;

overflow-y: auto;

}

/\* Main Content Styling \*/

.main-content {

flex: 1;

padding: 20px;

background-color: #fff;

overflow-y: auto;

}

/\* Footer Styling \*/

.footer {

background-color: #4caf50;

color: white;

text-align: center;

padding: 10px;

font-size: 16px;

}

1. App.js

import React from 'react';

import './App.css'; // Import the single CSS file

import Header from './Header';

import Sider from './Sider';

import MainContent from './MainContent';

import Footer from './Footer';

const App = () => {

return (

<div className="container">

<Header />

<div className="content-container">

<Sider />

<MainContent />

</div>

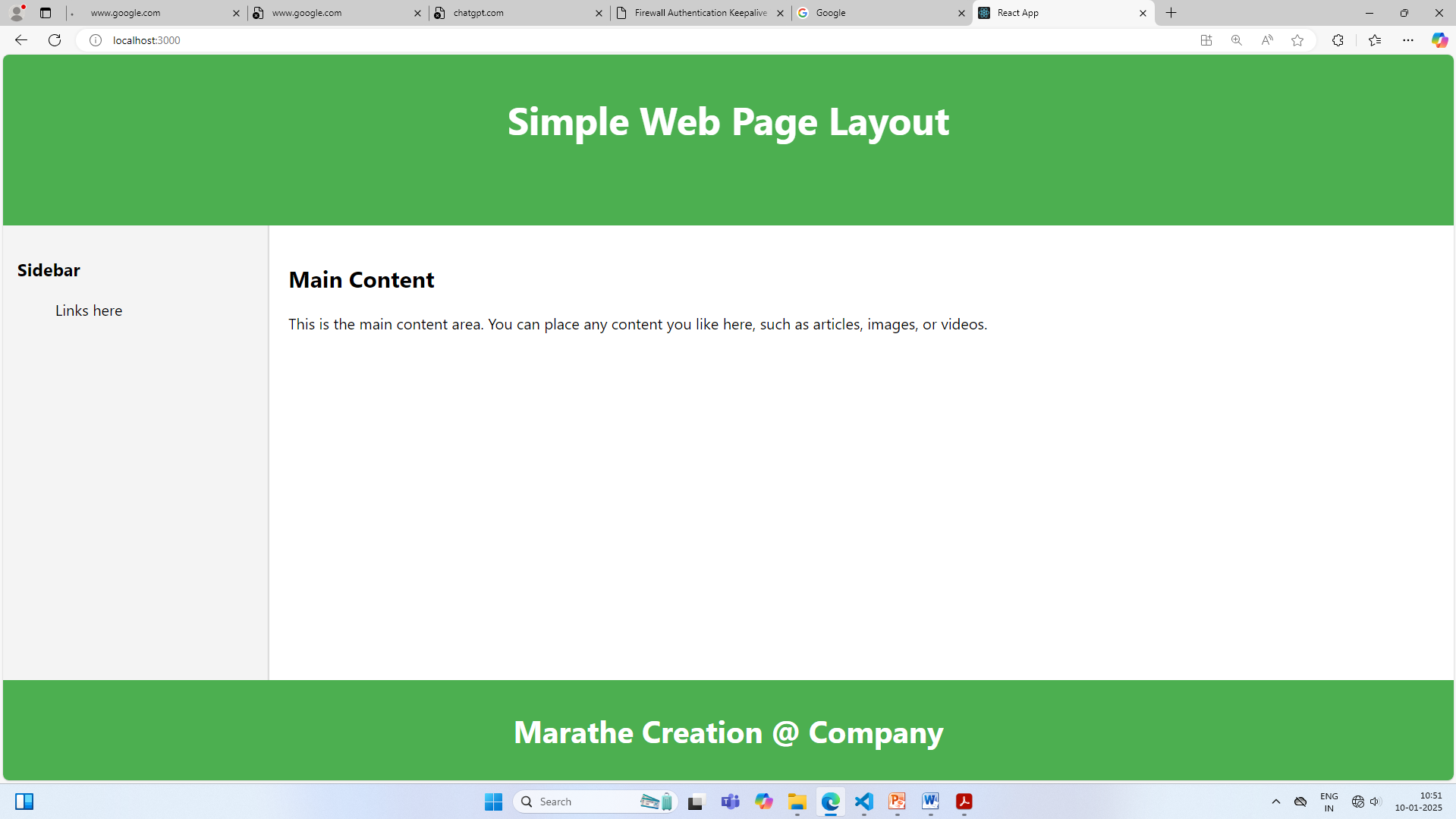
<Footer />

</div>

);

};

export default App;



//// **4. Write ReactJs code to create Simple Registration Form.**

import React from 'react';

import {useState} from 'react';

function App() {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const [m, setmale] = useState('');

const [Contact, setcontact] = useState('');

const handleSubmit = (e) => {

e.preventDefault(); // Prevent page reload

alert(`Form Submitted with: \nName:

${name} \nEmail: ${email} \nGender: ${m} Gender: ${Contact}`);

};

return (

<div className="App">

<h1>React Registration Form</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Name:</label>

<input type="text" value={name}

onChange={(e) => setName(e.target.value)}

placeholder="Enter your name" />

</div>

<div>

<label>Email:</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

placeholder="Enter your email"/>

</div>

<div>

<label>Gender:</label>

<input

type="radio" name="m"

value="Male"

onChange={(e) => setmale(e.target.value)} /> Male

&nbsp;

<input

type="radio" name="m"

value="Female"

onChange={(e) => setmale(e.target.value)} /> Female

</div>

<div>

<label>Contact</label>

<input

type="text"

value={Contact}

onChange={(e) => setcontact(e.target.value)}

placeholder="Enter your Contact No"/>

</div>

<div>

<button type="submit">Submit</button>

</div>

</form>

</div>

);

}

export default App;

//5. **Write ReactJs code to create Simple Login Form.**

import React from 'react';

import {useState} from 'react';

function App() {

const [uname, setName] = useState('');

const [pass, setEmail] = useState('');

const handleSubmit = (e) => {

e.preventDefault(); // Prevent page reload

if (uname === "RCP" && pass === "IMRD") {

alert("Login Successful");

}

else

{

alert("Login not Successful")

}

};

return (

<div>

<center>

<h1>Login Form</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Name:</label>

<input

type="text"

value={uname}

onChange={(e) => setName(e.target.value)}

placeholder="Enter your name"

/>

</div>

<div>

<label>Password:</label>

<input

type="pasword" value={pass} onChange={(e) => setEmail(e.target.value)}

placeholder="Enter your Password"

/>

</div>

<div>

<button type="submit">Submit</button>

</div>

</form>

</center>

</div>

);

};

export default App;

**// 6 & 7 Write ReactJsCreate a Single Page Application / Write ReactJs code to Applying Routing.**

Home.js

**src/Home.js**

import React from 'react';

const Home = () => (

<div>

<h1>Home Page</h1>

<p>Welcome to our React SPA!</p>

</div>

);

export default Home;

**src/pages/About.js**

import React from 'react';

const About = () => (

<div>

<h1>About Page</h1>

<p>This is the about page of our React SPA.</p>

</div>

);

export default About;

**src/Contact.js**

import React from 'react';

const Contact = () => (

<div>

<h1>Contact Page</h1>

<p>Feel free to reach out to us!</p>

</div>

);

export default Contact;

**src/App.js**

import React from 'react';

import { Routes, Route, Link } from 'react-router-dom';

import Home from './Home';

import About from './pages/About';

import Contact from './Contact';

const App = () => {

return (

<div>

<nav style={navStyle}>

<Link to="/" style={linkStyle}>Home</Link>

<Link to="/about" style={linkStyle}>About</Link>

<Link to="/contact" style={linkStyle}>Contact</Link>

</nav>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/about" element={<About />} />

<Route path="/contact" element={<Contact />} />

</Routes>

</div>

);

};

const navStyle = {

display: 'flex',

justifyContent: 'space-around',

padding: '1em',

background: 'Blue',

color: 'orange',

};

const linkStyle = {

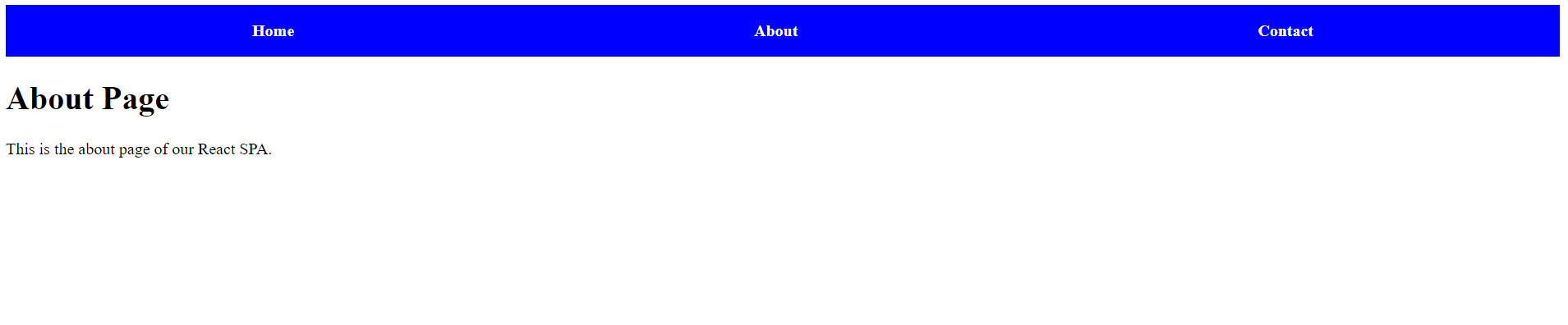
color: 'White',

textDecoration: 'none',

fontWeight: 'bold',

};

export default App;



**Assig 08. Write ReactJs/ NodeJs code to demonstrate the use of POST Method.**

// index.js

const express = require('express');

const app = express();

app.use(express.json()); // Middleware to parse JSON requests

// Sample data (in-memory)

let users = [

{ id: 1, name: 'Marathe' },

{ id: 2, name: 'Patil' }

];

// GET all users

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

res.json(users);

});

// GET user by ID

app.get('/users/:id', (req, res) => {

const user = users.find(u => u.id === parseInt(req.params.id));

if (!user) return res.status(404).send('User not found');

res.json(user);

});

// POST to create a new user

app.post('/users', (req, res) => {

const user = {

id: users.length + 1,

name: req.body.name

};

users.push(user);

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

});

// Start server on port 3000

const PORT = 3000;

app.listen(PORT, () => {

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

});

**Assig 09. Write ReactJs/ NodeJs code to demonstrate the use of GET Method.**

* **npm init –y**
* **npm install express**

// index.js

const express = require('express');

const app = express();

app.use(express.json()); // Middleware to parse JSON requests

// Sample data (in-memory)

let users = [

{ id: 1, name: 'Marathe' },

{ id: 2, name: 'Patil' }

];

// GET all users

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

res.json(users);

});

// GET user by ID

app.get('/users/:id', (req, res) => {

const user = users.find(u => u.id === parseInt(req.params.id));

if (!user) return res.status(404).send('User not found');

res.json(user);

});

// Start server on port 3000

const PORT = 3000;

app.listen(PORT, () => {

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

});

**node index.js**

**//Assig 10 To demonstrate REST API in Node JS**

GET: Retrieve data from the server (e.g., get a list of books or a single book).

POST: Send data to the server to create a new resource (e.g., add a new book).

PUT: Update an existing resource with new data (e.g., update details of an existing book).

DELETE: Remove a resource from the server (e.g., delete a book).

// index.js

const express = require('express');

const app = express();

app.use(express.json()); // Middleware to parse JSON requests

// Sample data (in-memory)

let users = [

{ id: 1, name: 'Marathe' },

{ id: 2, name: 'Patil' }

];

// GET all users

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

res.json(users);

});

// GET user by ID

app.get('/users/:id', (req, res) => {

const user = users.find(u => u.id === parseInt(req.params.id));

if (!user) return res.status(404).send('User not found');

res.json(user);

});

// POST to create a new user

app.post('/users', (req, res) => {

const user = {

id: users.length + 1,

name: req.body.name

};

users.push(user);

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

});

// DELETE user by ID

app.delete('/users/:id', (req, res) => {

const index = users.findIndex(u => u.id === parseInt(req.params.id));

if (index === -1) return res.status(404).send('User not found');

users.splice(index, 1);

res.send('User deleted');

});

// Start server on port 3000

const PORT = 3000;

app.listen(PORT, () => {

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

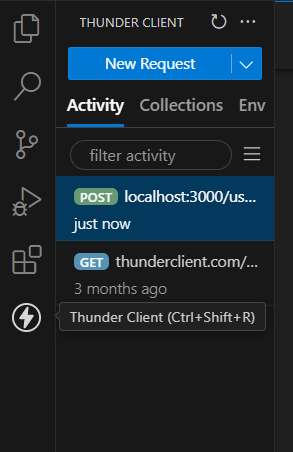
});

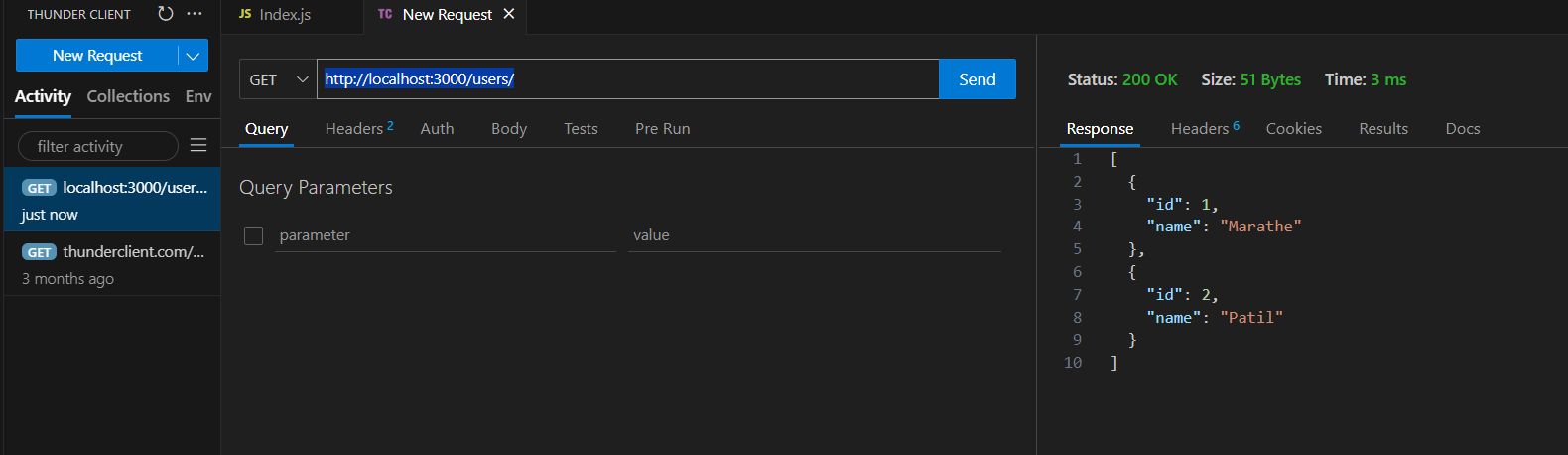
* **npm init –y**
* **npm install express**

**After**

* **node index.js**

**Open Postman->**

****

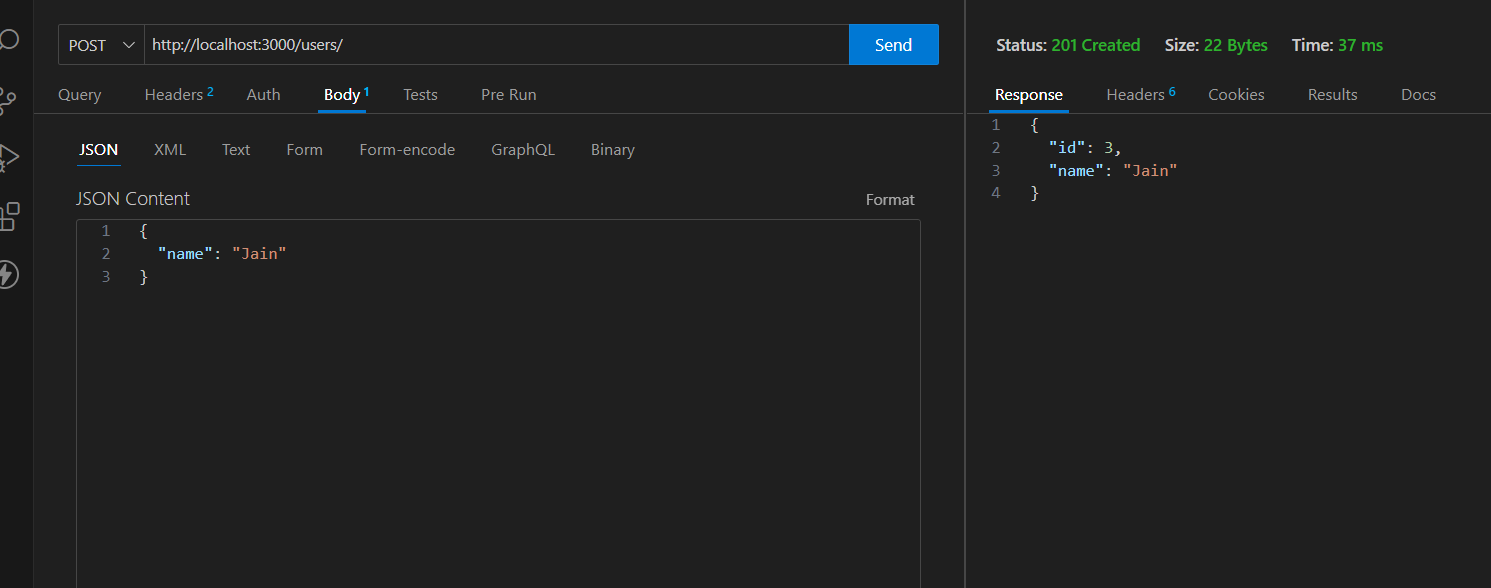
****

**GET** [**http://localhost:3000/users/1**](http://localhost:3000/users/1)

**GET** [**http://localhost:3000/users/**](http://localhost:3000/users/)

**For POST- Select Body**

**POST** [**http://localhost:3000/users/**](http://localhost:3000/users/)

****

**DELETE** [**http://localhost:3000/users/4**](http://localhost:3000/users/4)

**Assignment No 11: Create Node JS Application for to stored student information in database.**

**npm install mongodb**

**Demo.js**

const { MongoClient } =require('mongodb');

// URL of the MongoDB instance (this should be localhost if you're using MongoDB Compass)

consturl='mongodb://localhost:27017';

constdbName='Marathe'; // Database name

constcollectionName='datas'; // Collection name

// Data to insert

constnewUser= {

name:"IMRD",

age:89

};

// Create a MongoClient object

MongoClient.connect(url)

.then(client=> {

console.log('Connected to MongoDB');

const db=client.db(dbName); // Access the database

const collection=db.collection(collectionName); // Access the collection

// Insert the new document

Return collection.insertOne(newUser);

})

.then(result=> {

console.log('Inserted document:', result.insertedId);

})

.catch(error=> {

console.error('Error:', error);

});

**12:Create Node JS Application for login credentials.**

**reg Collection**

**/\*\***

**\* Paste one or more documents here**

**\*/**

**{**

**"\_id": {**

**"$oid": "67a43b4ee0f79e0b4b729300"**

**},"uname":"ram", "pass":"imrd"**

**}**

const { MongoClient } =require('mongodb');

// MongoDB connection URI

const url='mongodb://localhost:27017';

const dbName='Marathe';

const cN='reg';

async function login(un, ps) {

const client=new MongoClient(url);

try {

// Connect to the MongoDB client

await client.connect();

const db=client.db(dbName);

const cl=db.collection(cN);

// Find the user document by username

const user=await cl.findOne({ uname:un });

if (!user) {

console.log('User not found');

return;

}

// Compare the provided password with the stored plaintext password

if (user.pass===ps) {

console.log('Login successful!');

} else {

console.log('Incorrect password');

}

} catch (error) {

console.error('Error while logging in:', error);

}

}

// Example usage (replace with actual credentials)

login('ram', 'imrd');

1. **Create Node JS Application to update, display and delete student information.**

const { MongoClient } =require('mongodb');

// MongoDB connection URI

Const url='mongodb://localhost:27017';

Const dbName='Marathe';

Const cN='datas';

async function main() {

const client=new MongoClient(url);

try {

// Connect to MongoDB

Await client.connect();

const db=client.db(dbName);

const cl=db.collection(cN);

// 1. Insert a new user document

Const newUser= { name:'ram', age:34 };

Const ir=await cl.insertOne(newUser);

console.log('Inserted document with ID:', ir.insertedId);

// 2. Select (find) the newly inserted document by \_id

Const fu=await cl.find({}).toArray();;

console.log('Found document:', fu);

// 3. Update the user's name

Const uR=await cl.updateOne(

{ name:'ram' },

{ $set: { age:56 } });

console.log('Updated document count:', uR.modifiedCount);

// 5. Delete the user document

Const deleteResult=await cl.deleteOne({ name:'IMRD' });

console.log('Deleted document count:', deleteResult.deletedCount);

}

catch (err) {

console.error('Error:', err);

} finally {

// Close the connection

Await client.close();

}

}

main().catch(console.error);

**//15 Write React Code for Uncontrolled form components**

import React, { useRef } from 'react';

const Uf = () => {

const nameRef = useRef();

const handleSubmit = (event) => {

event.preventDefault();

alert('Form submitted: ' + nameRef.current.value);

};

return (

<div className="App">

<form onSubmit={handleSubmit}>

<label>

Name:

<input type="text" ref={nameRef} />

</label>

<button type="submit">Submit</button>

</form>

</div>

);

}

export default Uf;

**//16 React code for event generation.**

import React, { useState } from 'react';

const Eh = () => {

const [count, setCount] = useState(0);

const handleClick = () => {

setCount(count + 1);

};

return (

<div>

<button onClick={handleClick}>Click me!</button>

<p>You clicked {count} times.</p>

</div>

);

};

export default Eh;