

MERN Stack (MongoDB, Express.js, React, Node.js)

1. **Node.js:** Write a basic Node.js server that listens on port 3000 and returns a "Hello, World!" message when the root URL is accessed.

Initialized Node Project: npm init -y

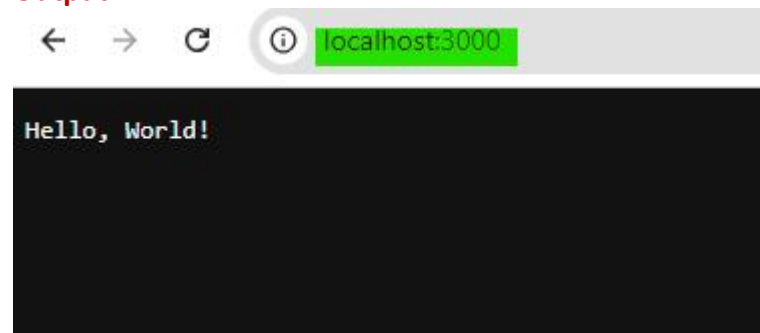
CODE:

```
//MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\1.Node.js\server.js
const http = require('http');

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  if (req.url === '/') {
    res.end("Hello, World!\n");
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end("Not Found\n");
  }
});

server.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

Output:



2. Express.js: Create a simple REST API using Express.js with a single route /users that returns a JSON list of users.

Initialized Node Project: npm init -y

Installed Packages: express

CODE:

```
// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\2.Express.js\server.js
const express = require('express');
const app = express();
const users = require("./listOfUsers");

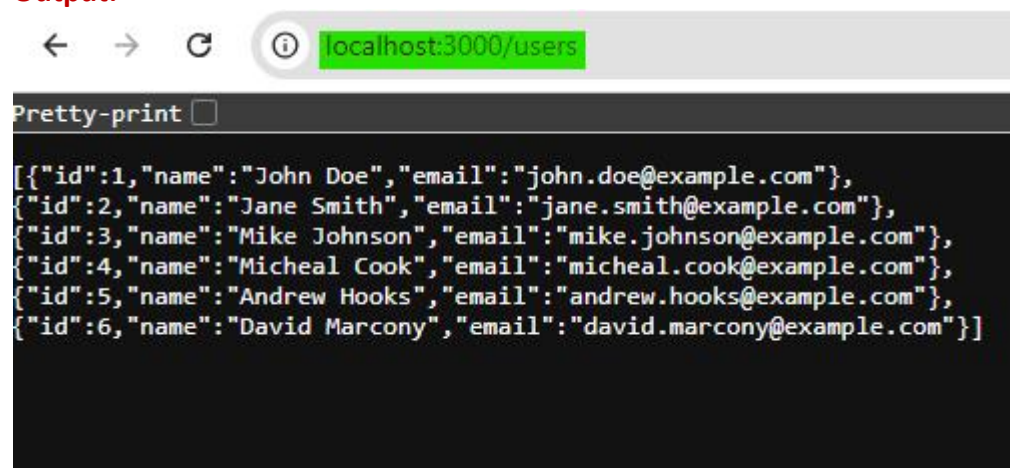
app.get('/users', (req, res) => {
  res.status(200).json(users);
});

app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

```
// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\2.Express.js\listOfUsers.js
const users = [
  { id: 1, name: 'John Doe', email: 'john.doe@example.com' },
  { id: 2, name: 'Jane Smith', email: 'jane.smith@example.com' },
  { id: 3, name: 'Mike Johnson', email: 'mike.johnson@example.com' },
  { id: 4, name: 'Micheal Cook', email: 'micheal.cook@example.com' },
  { id: 5, name: 'Andrew Hooks', email: 'andrew.hooks@example.com' },
  { id: 6, name: 'David Marcony', email: 'david.marcony@example.com' }
];

module.exports = users;
```

Output:



3. React: Build a basic React component that fetches the list of users from the /users API route (from question 2) and displays them in a table.

Created React App: npm create vite@latest

Installed Packages: cors

CODE:

```
// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\3.React\client\src\App.jsx
import UserData from "../components/UserData";

function App() {
  return (
    <>
      <UserData />
    </>
  )
}
export default App
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\3.React\client\src\components\UserData.jsx
import React, { useEffect, useState } from 'react';

function UserData() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  useEffect(() => {
    fetch('http://localhost:3000/users')
      .then((response) => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.json();
      })
      .then((data) => {
        setUsers(data);
        setLoading(false);
      })
      .catch((error) => {
        setError(error);
        setLoading(false);
      });
  }, []);
  if (loading) {
    return <p>Loading...</p>;
  }
  if (error) {
```

```

        return <p>Error: {error.message}</p>;
    }
    return (
        <div>
            <h1>User List</h1>
            <table border="1" cellPadding="10">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Email</th>
                    </tr>
                </thead>
                <tbody>
                    {users.map((user) => (
                        <tr key={user.id}>
                            <td>{user.id}</td>
                            <td>{user.name}</td>
                            <td>{user.email}</td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    );
}
export default UserData;

```

```

// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\2.Express.js\server.js
const cors = require("cors");

const corsOptions = {
    origin: "http://localhost:5173",
    methods: "GET, POST, PUT, DELETE, PATCH, HEAD",
    credentials: true,
};

app.use(cors(corsOptions));

```

Output:

<div><div><div>←</div><div>→</div><div>↺</div><div><div>🔒</div><div>localhost:5173</div></div></div></div>		
<h1>User List</h1>		
ID	Name	Email
1	John Doe	john.doe@example.com
2	Jane Smith	jane.smith@example.com
3	Mike Johnson	mike.johnson@example.com
4	Micheal Cook	micheal.cook@example.com
5	Andrew Hooks	andrew.hooks@example.com
6	David Marcony	david.marcony@example.com

4. MongoDB: Create a MongoDB schema for storing user data (name, email, age), and write a script to insert a new user into the collection.

Initialized Node Project: npm init -y

Installed Packages: mongoose

CODE:

```
// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\4.MongoDB\insertUser.js
const mongoose = require('mongoose');
const User = require('./userModel');

mongoose.connect('mongodb://127.0.0.1:27017/usersDB')
.then(() => {
  console.log('Connected to MongoDB');
}).catch((error) => {
  console.error('Error connecting to MongoDB:', error);
});

const insertUser = async () => {
  try {
    const newUser = new User({
      name: 'Vivek Singh',
      email: 'vivek.singh@example.com',
      age: 25
    });

    const savedUser = await newUser.save();
    console.log('User inserted:', savedUser);
  } catch (error) {
    console.error('Error inserting user:', error);
  } finally {
    mongoose.connection.close();
  }
};

insertUser();
```

```
// MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\4.MongoDB\userModel.js
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  }
});
```

```

    },
    age: {
      type: Number,
      required: true
    }
  });

const User = mongoose.model('User', userSchema);

module.exports = User;

```

Output:

```

PS C:\Users\KIIT\Desktop\LEAMORE\MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\4.MongoDB> node insertUser.js
● Connected to MongoDB
User inserted: {
  name: 'Vivek Singh',
  email: 'vivek.singh@example.com',
  age: 25,
  _id: new ObjectId('66f83621c606bb2b2a862996'),
  __v: 0
}
PS C:\Users\KIIT\Desktop\LEAMORE\MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\4.MongoDB>

```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test> show dbs
admin          40.00 KiB
amazon         72.00 KiB
college        40.00 KiB
config         72.00 KiB
local          76.00 KiB
mern_admin      8.00 KiB
relationDemo   288.00 KiB
test           80.00 KiB
usersDB        28.00 KiB
wanderlust     272.00 KiB
whatsapp       72.00 KiB
test> use usersDB
switched to db usersDB
usersDB> show collections
users
usersDB> db.users.find()
[
  {
    _id: ObjectId('66f83621c606bb2b2a862996'),
    name: 'Vivek Singh',
    email: 'vivek.singh@example.com',
    age: 25,
    __v: 0
  }
]
usersDB>

```

5. Express.js + MongoDB: Create an Express.js route to fetch a user by their email from the MongoDB database.

Initialized Node Project: npm init -y

Installed Packages: express, mongoose

CODE:

```
//  
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\5.Express.jsAndMongoDB\server.js  
const express = require("express");  
const User = require('./models/user');  
  
require("./database");  
const app = express();  
  
app.get('/user/:email', async (req, res) => {  
  const email = req.params.email;  
  try {  
    const user = await User.findOne({ email: email });  
    if (!user) {  
      return res.status(404).json({ message: 'User not found' });  
    }  
    res.json(user);  
  } catch (error) {  
    console.error('Error fetching user:', error);  
    res.status(500).json({ message: 'Internal server error' });  
  }  
});  
  
app.listen(3000, () => {  
  console.log('Server is running on http://localhost:3000');  
});
```

```
//  
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\5.Express.jsAndMongoDB\models\user.js  
const mongoose = require('mongoose');  
  
const userSchema = new mongoose.Schema({  
  name: {  
    type: String,  
    required: true  
  },  
  email: {  
    type: String,  
    required: true,  
    unique: true  
  },  
  age: {  
    type: Number,
```



```

        required: true
      }
    });

const User = mongoose.model('User', userSchema);
module.exports = User;

```

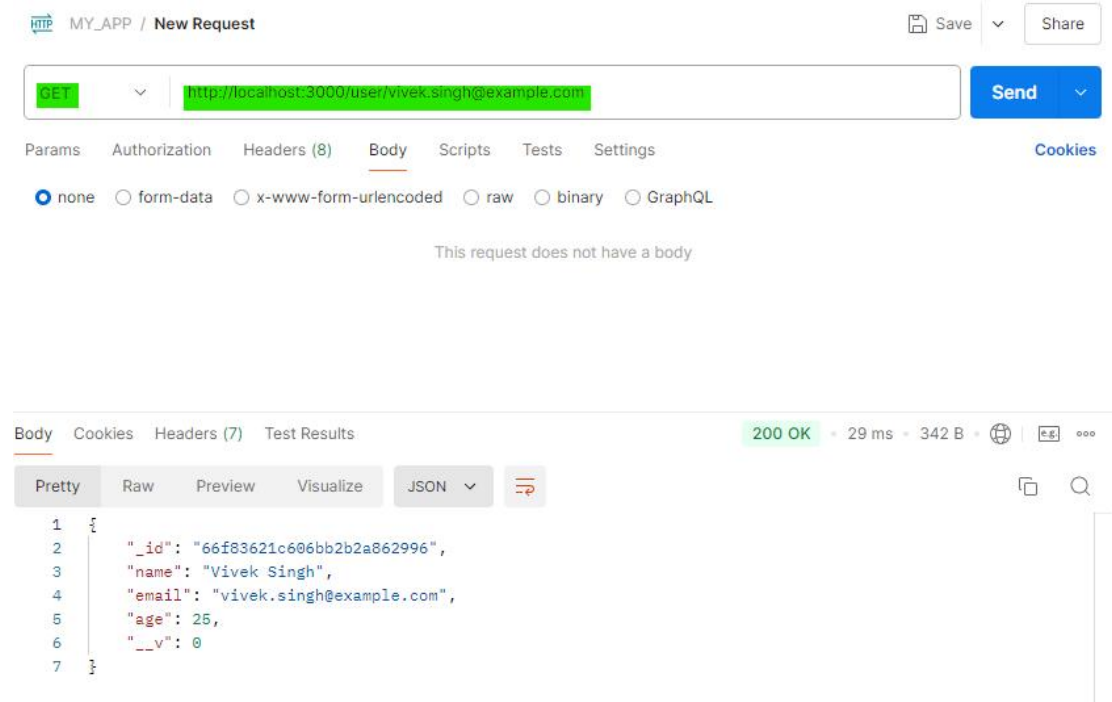
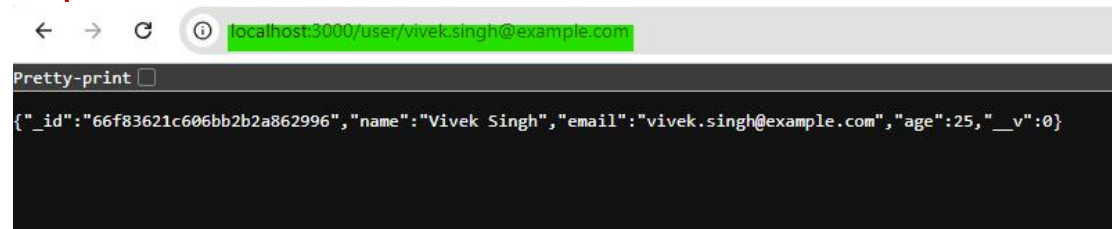
```

//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\5.Express.jsAndMongoDB\database\
index.js
const mongoose = require('mongoose');

mongoose.connect('mongodb://127.0.0.1:27017/usersDB')
.then(() => {
  console.log('Connected to MongoDB');
}).catch((error) => {
  console.error('Error connecting to MongoDB:', error);
});

```

Output:



6. **React + State Management: Build a form component in React that allows users to submit their name, email, and age. On submission, send the data to the backend API and update the state to display the new user.**

Initialized Node Project: npm init -y

Installed Packages: express, mongoose, cors

Created React App: npm create vite@latest

CODE:

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\6.ReactAndState_Management\server\server.js
const cors = require("cors");

const corsOptions = {
  origin: "http://localhost:5173",
  methods: "GET, POST, PUT, DELETE, PATCH, HEAD",
  credentials: true,
};

app.use(cors(corsOptions));
app.use(express.json());

app.post('/users', async (req, res) => {
  const { name, email, age } = req.body;
  try {
    const newUser = new User({ name, email, age });
    const savedUser = await newUser.save();
    res.status(201).json(savedUser);
  } catch (error) {
    console.error('Error creating user:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
});
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\6.ReactAndState_Management\client\src\App.jsx
import Form from "../components/Form";

function App() {
  return (
    <>
      <Form />
    </>
  )
}

export default App
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\6.ReactAndState_Management\client\src\components\Form.jsx
import React, { useState } from 'react';
import User from "../User";

function Form() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    age: ''
  });
  const [newUser, setNewUser] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [name]: value
    }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    try {
      const response = await fetch('http://localhost:3000/users', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(formData)
      });
      if (!response.ok) {
        throw new Error('Failed to add user');
      }
      const result = await response.json();
      setNewUser(result);
      setFormData({ name: '', email: '', age: '' });
    } catch (error) {
      setError(error.message);
    } finally {
      setLoading(false);
    }
  };
}
```

```

return (
  <div>
    <h1>Add New User</h1>
    {error && <p style={{ color: 'red' }}>Error: {error}</p>}
    <form onSubmit={handleSubmit}>
      <div>
        <label>Name: </label>
        <input
          type="text"
          name="name"
          value={formData.name}
          onChange={handleChange}
          required
        />
      </div>
      <div>
        <label>Email: </label>
        <input
          type="email"
          name="email"
          value={formData.email}
          onChange={handleChange}
          required
        />
      </div>
      <div>
        <label>Age: </label>
        <input
          type="number"
          name="age"
          value={formData.age}
          onChange={handleChange}
          required
        />
      </div>
      <button type="submit" disabled={loading}>
        {loading ? 'Submitting...' : 'Submit'}
      </button>
    </form>

    {newUser && (
      <User name={newUser.name} email={newUser.email}
age={newUser.age}/>
    )}
  </div>
);
}

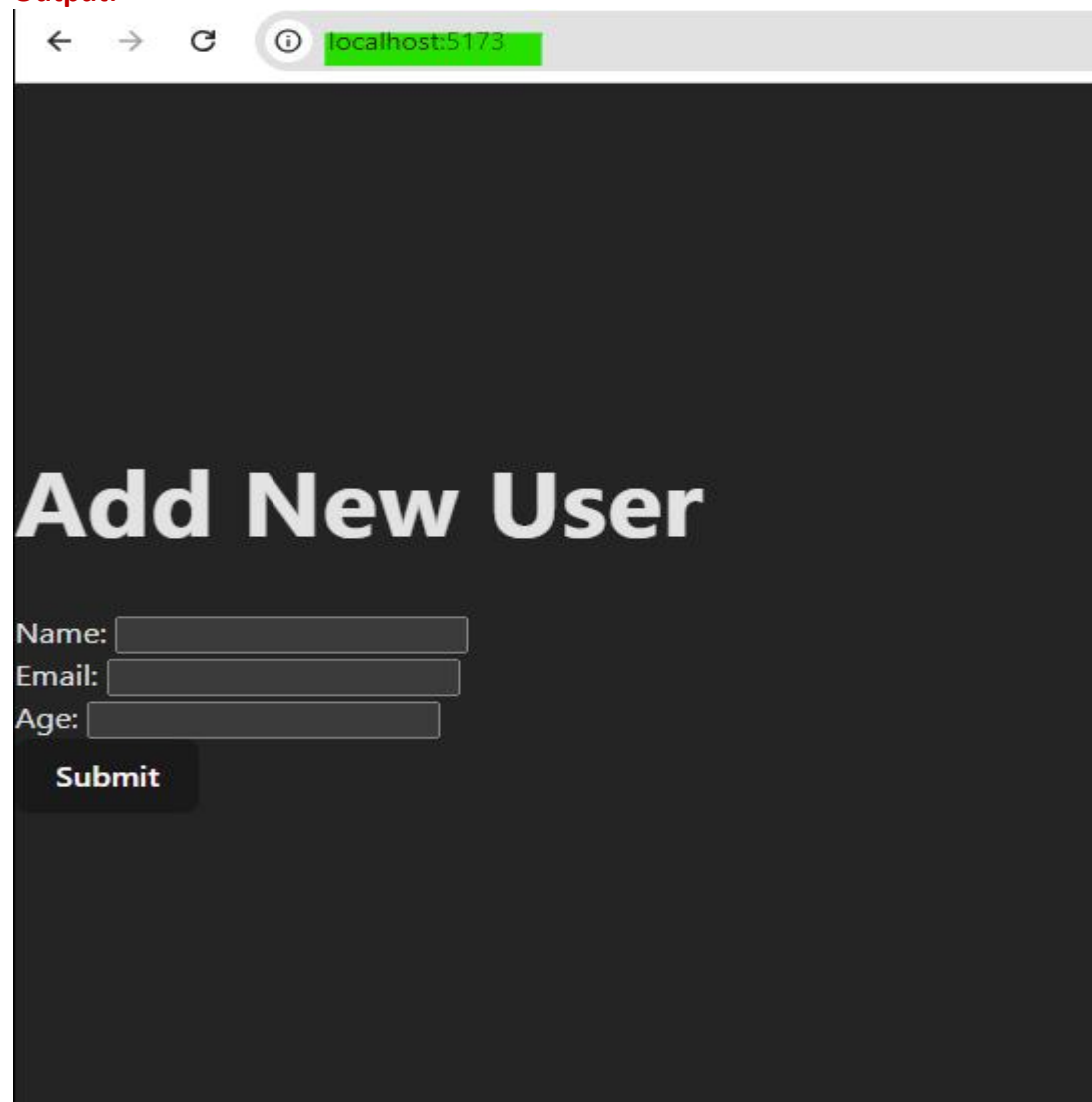
```

```
export default Form;

//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\6.ReactAndState_Management\client\src\components\User.jsx
function User({name, email, age}){
  return(
    <div>
      <h2>New User Added:</h2>
      <p>Name: {name}</p>
      <p>Email: {email}</p>
      <p>Age: {age}</p>
    </div>
  );
}

export default User;
```

Output:



localhost:5173

Add New User

Name:

Email:

Age:

Submit



localhost:5173

Add New User

Name:

Email:

Age:

localhost:5173

Add New User

Name:

Email:

Age:

Submit

New User Added:

Name:

Devang Prasad

Email:

devang.prasad@example.com

Age:

22

7. React Routing: Set up React Router in an application to navigate between a Home page and a Users page.

Initialized Node Project: npm init -y

Installed Packages: express, mongoose, cors, react-router-dom

Created React App: npm create vite@latest

CODE:

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\7.React_Routing\client\src\main.
jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import { BrowserRouter as Router } from 'react-router-dom';

createRoot(document.getElementById('root')).render(
  <Router>
    <StrictMode>
      <App />
    </StrictMode>
  </Router>
)
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\7.React_Routing\client\src\App.j
sx
import { Route, Routes, Link } from 'react-router-dom';
import Home from './components/Home';
import Users from './components/Users';

function App() {
  return (
    <>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/users">Users</Link>
            </li>
          </ul>
        </nav>
        <Routes>
          <Route path="/" element={ <Home /> } />
          <Route path="/users" element={ <Users /> } />
        </Routes>
      </div>
    </>
  )
}
```



```

    </>
  )
}
export default App

```

```

//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\7.React_Routing\client\src\components\Home.jsx
function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      <p>Welcome to the Home page!</p>
    </div>
  );
}

export default Home;

```

```

//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\7.React_Routing\client\src\components\Users.jsx
import React, { useEffect, useState } from 'react';

function Users() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    fetch('http://localhost:3000/getallusers')
      .then((response) => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.json();
      })
      .then((data) => {
        setUsers(data);
        setLoading(false);
      })
      .catch((error) => {
        setError(error);
        setLoading(false);
      });
  }, []);

  if (loading) {
    return <p>Loading...</p>;
  }
  if (error) {
    return <p>Error: {error}</p>;
  }
  return <p>Users: {users}</p>;
}

export default Users;

```

```

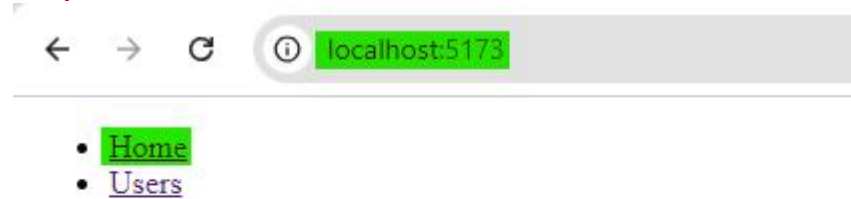
    }

    if (error) {
      return <p>Error: {error.message}</p>;
    }

    return (
      <div>
        <h1>Users Page</h1>
        <table border="1" cellpadding="10">
          <thead>
            <tr>
              <th>ID</th>
              <th>Name</th>
              <th>Email</th>
            </tr>
          </thead>
          <tbody>
            {users.map((user) => (
              <tr key={user.id}>
                <td>{user.id}</td>
                <td>{user.name}</td>
                <td>{user.email}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    );
  }
}
export default Users;

```

Output:



- [Home](#)
- [Users](#)

Users Page

ID	Name	Email
	Vivek Singh	vivek.singh@example.com
	Rohit Sharma	rohit.sharma@example.com
	Vikram Rathore	vikram.rathore@example.com
	Manisha Yadav	manisha.yadav@example.com
	Vaibhav Sharma	vaibhav.sharma@example.com
	Swati Saini	swati.saini@example.com
	Sumit Yadav	sumit.yadav@example.com
	Manoj Desai	manoj.desai@example.com
	Sheetal Rai	sheetal.rao@example.com
	Ragini Rai	ragini.rao@example.com

8. RESTful API Design: Design and implement a REST API in Express.js for a simple blog application with routes for creating, reading, updating, and deleting blog posts.

Initialized Node Project: npm init -y

Installed Packages: express, mongoose, cors

Routes for CRUD Operation:

- **Create a Post** (POST /api/blog/posts): Adds a new blog post.
- **Get All Posts** (GET /api/blog/posts): Retrieves all blog posts.
- **Get a Single Post** (GET /api/blog/posts/:id): Fetches a specific post by its ID.
- **Update a Post** (PUT /api/blog/posts/:id): Updates a post by its ID.
- **Delete a Post** (DELETE /api/blog/posts/:id): Deletes a post by its ID.

CODE:

```
//  
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\8.RESTful_API_Design\server.js  
const express = require("express");  
const cors = require("cors");  
const blogRoute = require("./router/blog-router");  
  
require("./database");  
const app = express();  
  
const corsOptions = {  
  origin: "http://localhost:5173",  
  methods: "GET, POST, PUT, DELETE, PATCH, HEAD",  
  credentials: true,  
};  
  
app.use(cors(corsOptions));  
app.use(express.json());  
  
app.use("/api/blog", blogRoute);  
  
app.listen(3000, () => {  
  console.log('Server is running on http://localhost:3000');  
});
```

```
//  
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\8.RESTful_API_Design\database\index.js  
const mongoose = require('mongoose');  
  
mongoose.connect('mongodb://127.0.0.1:27017/blogDB')  
  .then(() => {  
    console.log('Connected to MongoDB');  
  }).catch((error) => {  
    console.error('Error connecting to MongoDB:', error);  
  });
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\8.RESTful_API_Design\router\blog
-router.js
const express = require('express');
const router = express.Router();
const blogController = require("../controllers/blog-controller");

router.route("/posts").post(blogController.createNewPost);
router.route("/posts").get(blogController.getAllPosts);
router.route("/posts/:id").get(blogController.getSinglePostById);
router.route("/posts/:id").put(blogController.updatePostById);
router.route("/posts/:id").delete(blogController.deletePostById);

module.exports = router;
```

```
//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\8.RESTful_API_Design\controllers
\blog-controller.js
const Post = require("../models/Post");

const createNewPost = async (req, res) => {
  const { title, content, author } = req.body;
  try {
    const newPost = new Post({ title, content, author });
    const savedPost = await newPost.save();
    res.status(201).json(savedPost);
  } catch (error) {
    res.status(500).json({ message: 'Error creating post', error });
  }
};

const getAllPosts = async (req, res) => {
  try {
    const posts = await Post.find();
    res.status(200).json(posts);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching posts', error });
  }
};

const getSinglePostById = async (req, res) => {
  try {
    const post = await Post.findById(req.params.id);
    if (!post) return res.status(404).json({ message: 'Post not
found' });
    res.status(200).json(post);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching post', error });
  }
};
```

```

    }
  };

const updatePostById = async (req, res) => {
  const { title, content, author } = req.body;
  try {
    const updatedPost = await Post.findByIdAndUpdate(
      req.params.id,
      { title, content, author },
      { new: true, runValidators: true }
    );
    if (!updatedPost) return res.status(404).json({ message: 'Post not found' });
    res.status(200).json(updatedPost);
  } catch (error) {
    res.status(500).json({ message: 'Error updating post', error });
  }
};

const deletePostById = async (req, res) => {
  try {
    const deletedPost = await Post.findByIdAndDelete(req.params.id);
    if (!deletedPost) return res.status(404).json({ message: 'Post not found' });
    res.status(200).json({ message: 'Post deleted successfully' });
  } catch (error) {
    res.status(500).json({ message: 'Error deleting post', error });
  }
};

module.exports = { createNewPost, getAllPosts, getSinglePostById, updatePostById, deletePostById };

```

```

//
MERN_DEVELOPER_ASSIGNMENT\1.MERN_STACK\8.RESTful_API_Design\models\Post.js
const mongoose = require('mongoose');

const postSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true
  },
  content: {
    type: String,
    required: true
  },
  author: {

```

```

    type: String,
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

const Post = mongoose.model('Post', postSchema);

module.exports = Post;

```

Output:

MY_APP / New Request Save Share

POST ▼ http://localhost:3000/api/blog/posts Send ▼

Params Authorization Headers (10) **Body** ● Scripts Tests Settings Cookies

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

```

1  {
2    "title": "Neural Network in ML",
3    "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
        human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
        are used for tasks such as classification, regression, image recognition, and more.",
4    "author": "Vivek Singh"
5  }

```

Body Cookies Headers (10) Test Results 201 Created · 15 ms · 800 B · 🌐 📄 Save Response ⋮

Pretty Raw Preview Visualize JSON ▼ 🔧

```

1  {
2    "title": "Neural Network in ML",
3    "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
        human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
        are used for tasks such as classification, regression, image recognition, and more.",
4    "author": "Vivek Singh",
5    "_id": "66f86d7af988fa97756905c8",
6    "createdAt": "2024-09-28T20:56:26.929Z",
7    "__v": 0
8  }

```

HTTP

http://localhost:3000/api/blog/posts

Save

Share

GET

http://localhost:3000/api/blog/posts

Send

ParamsAuthorizationHeaders (7)BodyScriptsTestsSettingsCookie

BodyCookiesHeaders (10)Test Results200 OK · 10 ms · 1.79 KB · Save Response

PrettyRawPreviewVisualizeJSON

```
16      "_v": 0
17    },
18  ],
19  "_id": "66f868bffa97756905be",
20  "title": "Machine Learning",
21  "content": "Machine learning is a subset of artificial intelligence that enables a system to autonomously learn and
              improve using neural networks and deep learning, without being explicitly programmed, by feeding it large
              amounts of data.",
22  "author": "Vimal Shrivastava",
23  "createdAt": "2024-09-28T20:36:15.191Z",
24  "_v": 0
25  },
26  ],
27  "_id": "66f86d7af988fa97756905c8",
28  "title": "Neural Network in ML",
29  "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of
              the human brain. It consists of interconnected layers of nodes (also called neurons or units), which process
              data and are used for tasks such as classification, regression, image recognition, and more.",
30  "author": "Vivek Singh",
31  "createdAt": "2024-09-28T20:56:26.929Z",
32  "_v": 0
33  }
34  ]
```

HTTP

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Save

Share

GET

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Send

ParamsAuthorizationHeaders (7)BodyScriptsTestsSettingsCookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

BodyCookiesHeaders (10)Test Results200 OK · 12 ms · 795 B · Save Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    "_id": "66f86d7af988fa97756905c8",
3    "title": "Neural Network in ML",
4    "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
                human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
                are used for tasks such as classification, regression, image recognition, and more.",
5    "author": "Vivek Singh",
6    "createdAt": "2024-09-28T20:56:26.929Z",
7    "_v": 0
8  }
```


PUT

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Send

Params

Authorization

Headers (10)

Body

Scripts

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "title": "Neural Network in ML",
3   "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
              human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
              are used for tasks such as classification, regression, image recognition, and more.",
4   "author": "Unknown Person"
5 }
```

Body

Cookies

Headers (10)

Test Results

200 OK

13 ms

798 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "_id": "66f86d7af988fa97756905c8",
3   "title": "Neural Network in ML",
4   "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
              human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
              are used for tasks such as classification, regression, image recognition, and more.",
5   "author": "Unknown Person",
6   "createdAt": "2024-09-28T20:56:26.929Z",
7   "__v": 0
8 }
```

GET

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Send

Params

Authorization

Headers (7)

Body

Scripts

Tests

Settings

Cookies

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body

Cookies

Headers (10)

Test Results

200 OK

9 ms

798 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "_id": "66f86d7af988fa97756905c8",
3   "title": "Neural Network in ML",
4   "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the
              human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and
              are used for tasks such as classification, regression, image recognition, and more.",
5   "author": "Unknown Person",
6   "createdAt": "2024-09-28T20:56:26.929Z",
7   "__v": 0
8 }
```

MY_APP / New Request

Save

Share

DELETE

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Send

Params

Authorization

Headers (10)

Body

Scripts

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "title": "Neural Network in ML",
3   "content": "A Neural Network (NN) is a type of machine learning model inspired by the structure and functioning of the human brain. It consists of interconnected layers of nodes (also called neurons or units), which process data and are used for tasks such as classification, regression, image recognition, and more.",
4   "author": "Unknown Person"
5 }
```

Body

Cookies

Headers (10)

Test Results

200 OK

11 ms

380 B

Save Response

...

Pretty

Raw

Preview

Visualize

JSON

...

```
1 {
2   "message": "Post deleted successfully"
3 }
```

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Save

Share

GET

http://localhost:3000/api/blog/posts/66f86d7af988fa97756905c8

Send

Params

Authorization

Headers (7)

Body

Scripts

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (10)

Test Results

404 Not Found

8 ms

376 B

Save Response

...

Pretty

Raw

Preview

Visualize

JSON

...

```
1 {
2   "message": "Post not found"
3 }
```