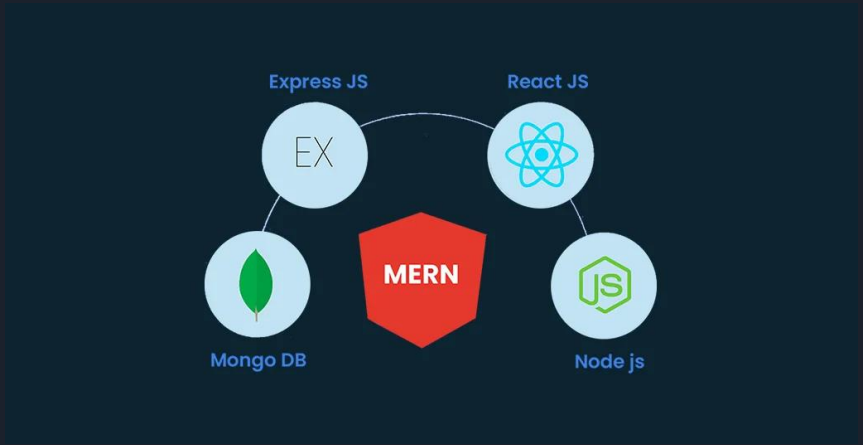
MERN Stack

RC Balaji

B.Tech -IT

Overview

The MERN stack is an open-source full-stack JavaScript solution for rapidly developing and deploying full-stack applications. This, therefore, states that fact that MERN is the most effective platform for creating websites and applications. As a result, each component of MERN has a distinct purpose.



Overview of MongoDB, Express.js, React, and React, and Node.js

MongoDB

A powerful NoSQL solution known for its scalability and flexibility.

Express.js

A fast and minimalist web application framework for Node.js, simplifying server-side side development.

React

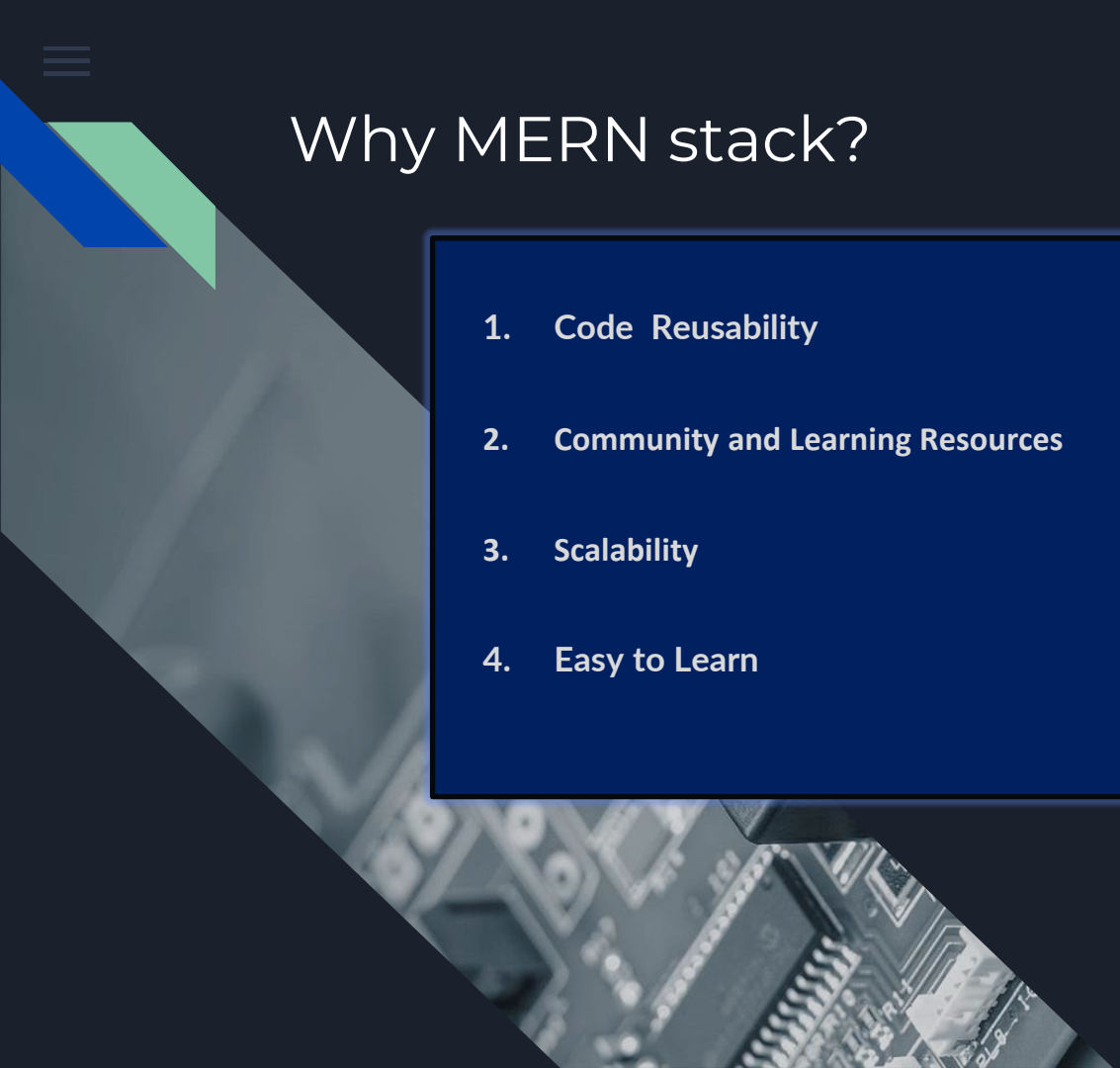
A popular JavaScript library for building dynamic user interfaces.

Node.js

A runtime environment allows JavaScript code to on the server-side.

Why MERN stack?

1. Code Reusability
2. Community and Learning Resources
3. Scalability
4. Easy to Learn



CURD

CRUD

CRUD Operations:-

Create a element

Read a element

Update a element

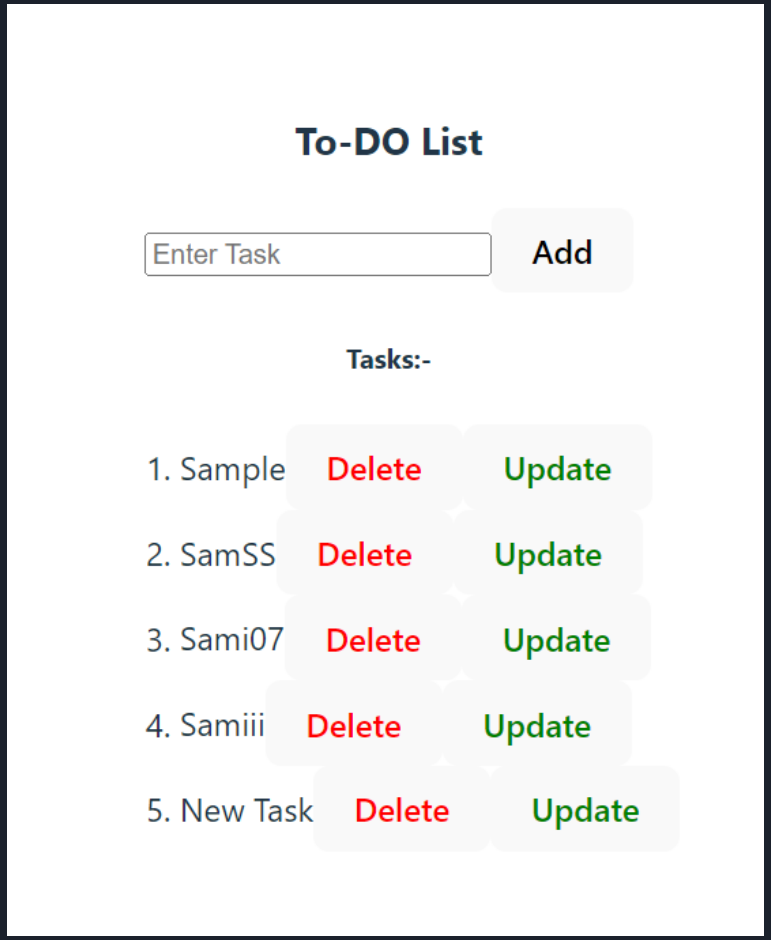
Delete a element

For that:-

We have to develop a Simple Application

Any Guess?

A To-Do List


To-DO List
Enter Task
Add
Tasks:-
1. Sample
Delete
Update
2. SamSS
Delete
Update
3. Sami07
Delete
Update
4. Samiii
Delete
Update
5. New Task
Delete
Update

Shall We START?

404 - Page Locked!!!
You are not Allowed to Next!!!

Why??

You are asked to unlock some pre-request to allow

What are the Pre-request

React :-

1. useState()
2. useEffect()
3. map()
4. try…catch
5. then…catch

MongoDB:-

1. Model Schema
2. Method use for add a element
3. Method use for read a element
4. Method use for update a element
5. Method use for delete a element

What are the Pre-request

Express :-

1. Routing
2. Http Methods
3. http://localhost:5000
4. http://localhost:5000/api/tasks
5. http://localhost:5000/api/:id
6. req and res object

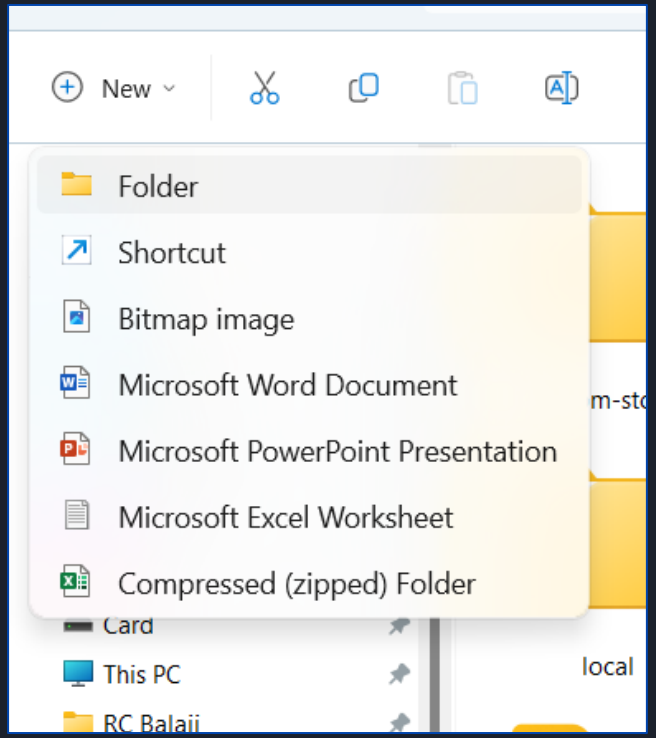
Node :-

1. How to start a Server
2. How to run a JS file
3. Import and export
4. async and await

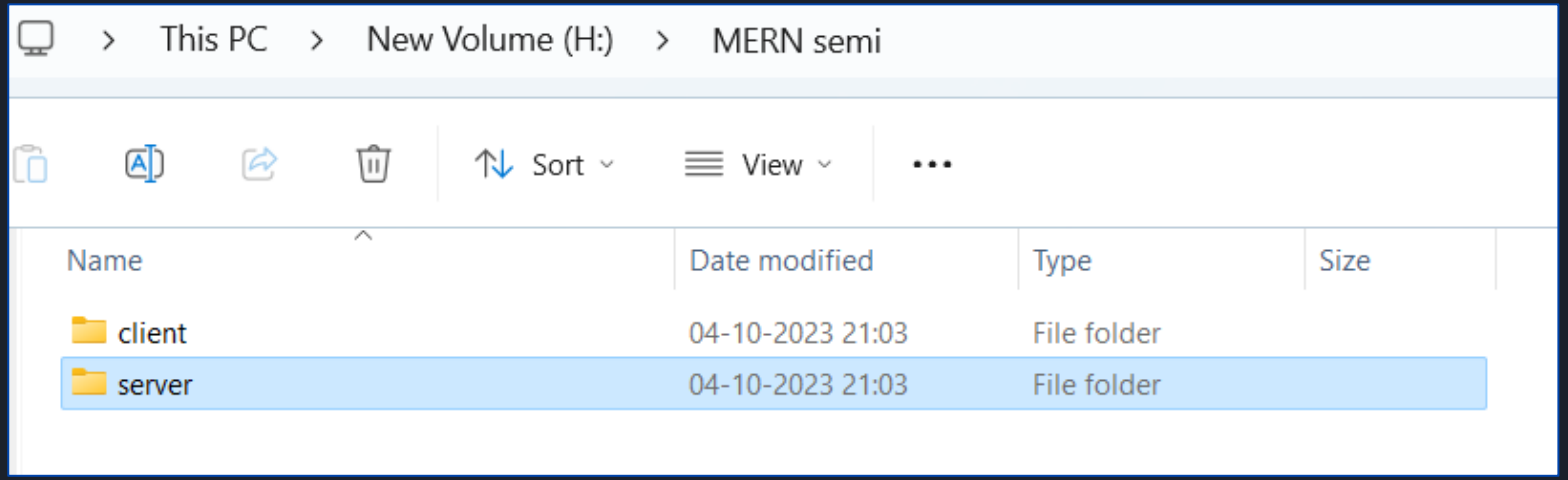
OMG!!! Really you Unlock this??

Oki,That’s fine
Let’s Start →

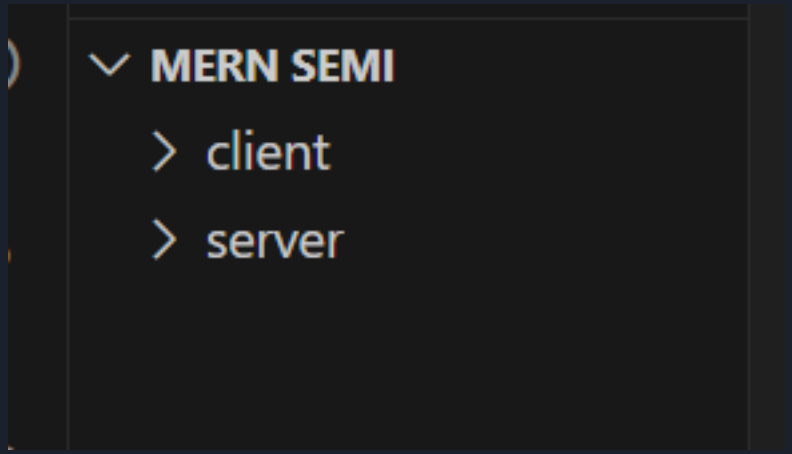
Create a New Folder



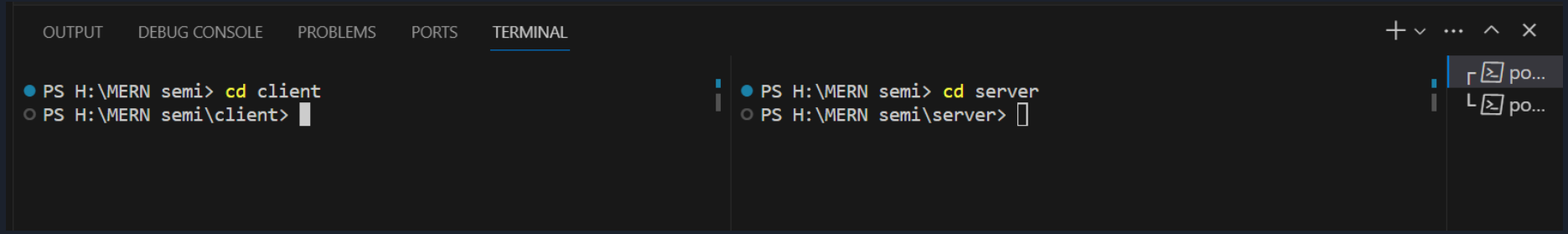
Create 2 folders



OPEN this Folder in VS code



Open and Split the VS Terminal (ctrl + ` or ~)



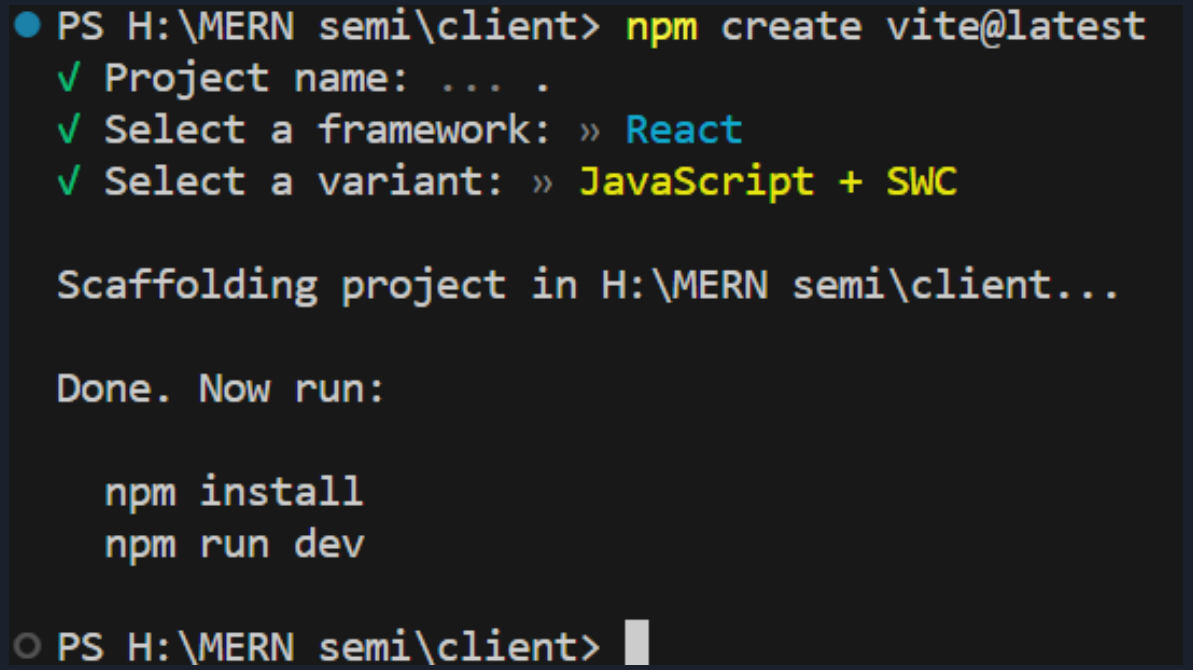
React

Create a React App inside Client folder

```
PS H:\MERN semi> cd client
PS H:\MERN semi\client> npm create vite@latest
```

React

Create a React App inside Client folder



React

Create a React App inside Client folder

```
PS H:\MERN semi\client> npm install
[..................] / idealTree:client: sill idealTree build
```

Install necessary packages

```
PS H:\MERN semi\client> npm i axios
```

React

Start the app

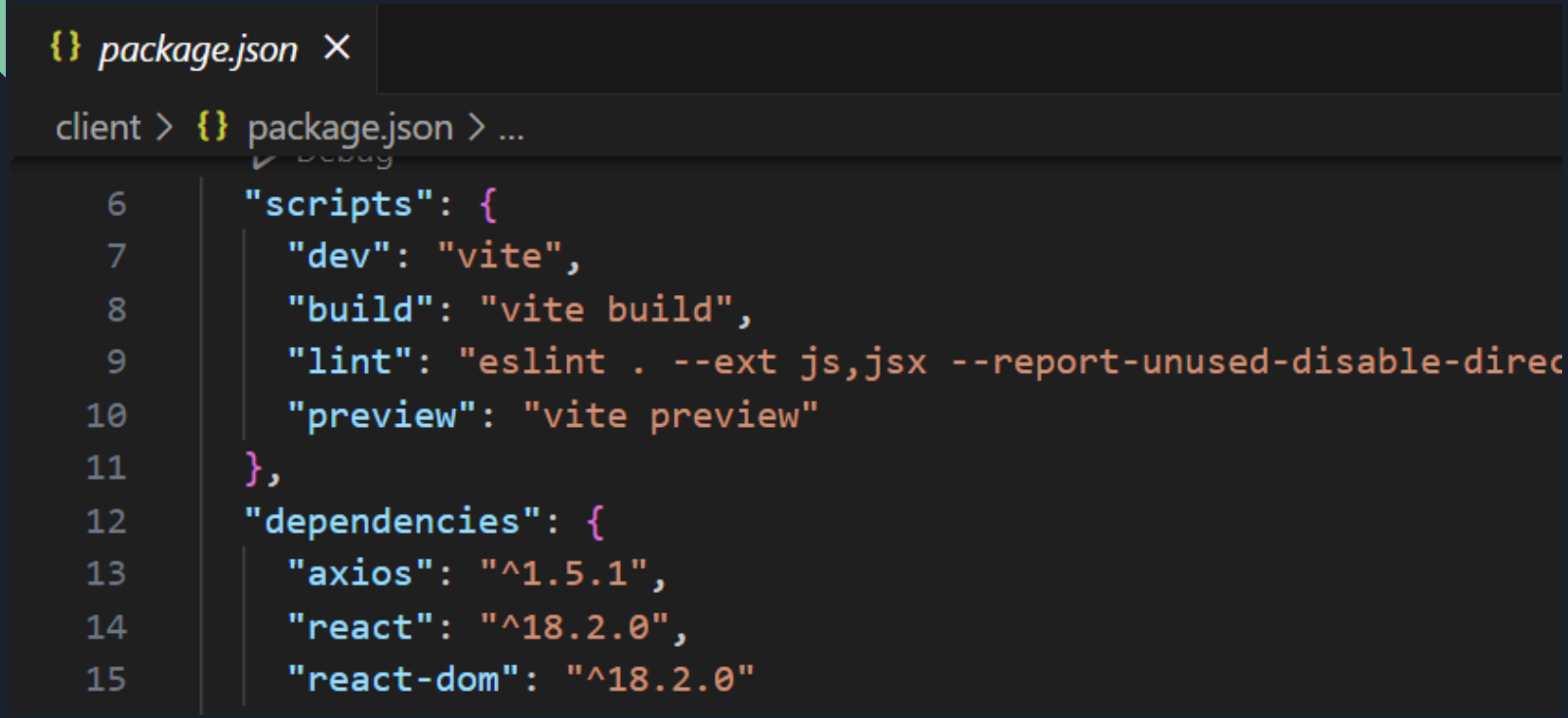
```
PS H:\MERN semi\client> npm run dev
```

```
VITE v4.4.10  ready in 4057 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h to show help
```

Why npm run dev not npm start?

React


{} package.json
client > {} package.json > ...
"scripts": {
"dev": "vite",
"build": "vite build",
"lint": "eslint . --ext js,jsx --report-unused-disable-direc
"preview": "vite preview"
},
"dependencies": {
"axios": "^1.5.1",
"react": "^18.2.0",
"react-dom": "^18.2.0"

React

```
"scripts": {
  "start":"vite",
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint . --ext js,js
  "preview": "vite preview"
},
```

React

Now , start the app using npm start

```
PS H:\MERN semi\client> npm start

> client@0.0.0 start
> vite
```

```
VITE v4.4.10  ready in 4057 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h to show help
]
```

Node

Init the server dir

```
PS H:\MERN semi> cd server
PS H:\MERN semi\server> npm init -y
```

Install necessary packages

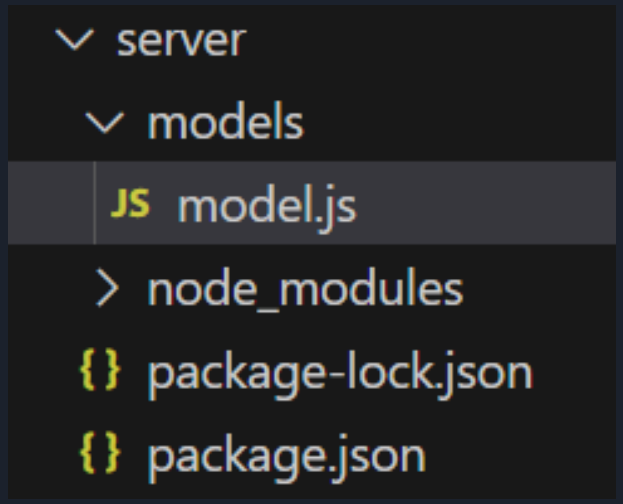
```
PS H:\MERN semi\server> npm i express mongoose cors nodemon
```

Wait, What will be the first??

Frontend or Backend

MongoDB

Create model schema for MongoDB



Node

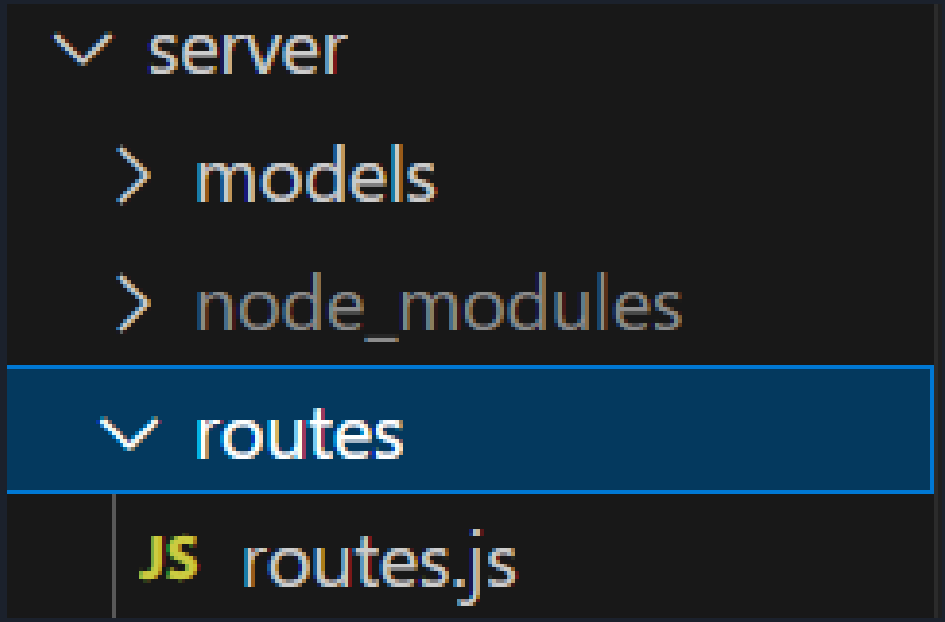
Inside model.js

server > models > JS model.js > taskSchema

```
1   const mongoose = require("mongoose");
2
3   const taskSchema = new mongoose.Schema({
4     text: {
5       type: String,
6       required: true,
7     },
8   });
9
10  module.exports = mongoose.model("Task", taskSchema);
11
```

Express

Create routes for the express



Express

Import necessary packages inside router.js

routes > JS routes.js > router.get("/tasks") callback

```
const express = require("express");
const Task = require("../models/model");

const router = express.Router();
```

Express

Create an element

```
router.post("/tasks", async (req, res) => {
  try {
    const newItem = new Task(req.body);
    await newItem.save();
    res.json(newItem);
  } catch (err) {
    res.json({ message: err.message });
  }
});
```

Express

Read an element

```
router.get("/tasks", async (req, res) => {
  try {
    const tasks = await Task.find();
    res.json(tasks);
  } catch (err) {
    res.json({ message: err.message });
  }
});
```

Express

Update an element

```
router.put("/tasks/:id", async (req, res) => {
  try {
    const updatedItem = await Task.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
    });
    res.json(updatedItem);
  } catch (err) {
    res.json({ message: err.message });
  }
});
```

Express

Delete an element

```
router.delete("/tasks/:id", async (req, res) => {
  try {
    await Task.findByIdAndRemove(req.params.id);
    res.json({ msg: "Item removed" });
  } catch (err) {
    console.error(err);
    res.json({ message: "Something went wrong" });
  }
});
```

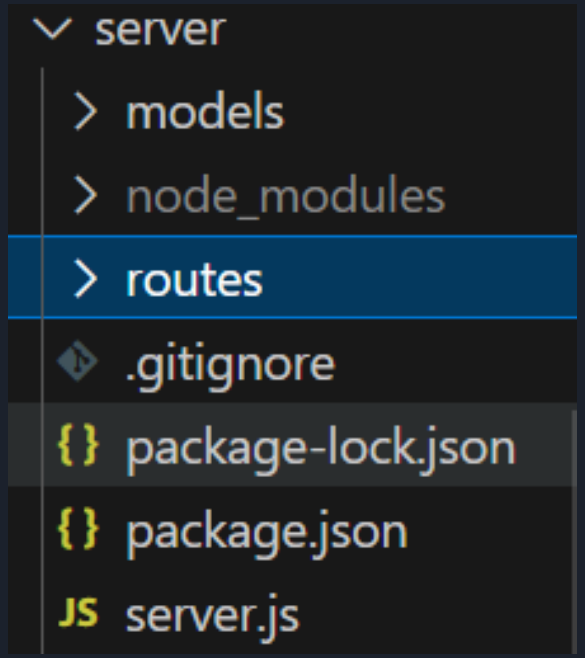
Express

Export the router

```
module.exports = router;
```

Node

Create server.js



Node

Import necessary packages

```
JS server.js > ...
const express = require("express");
const db = require("mongoose");
const cors = require("cors");
const api = require("./routes/routes");
```

Node

Create a object for express & Set the PORT number

```
const app = express();
const PORT = process.env.PORT || 5000;
```

Includes cors ,json,routes

```
app.use(cors());
app.use(express.json());
app.use("/api", api);
```

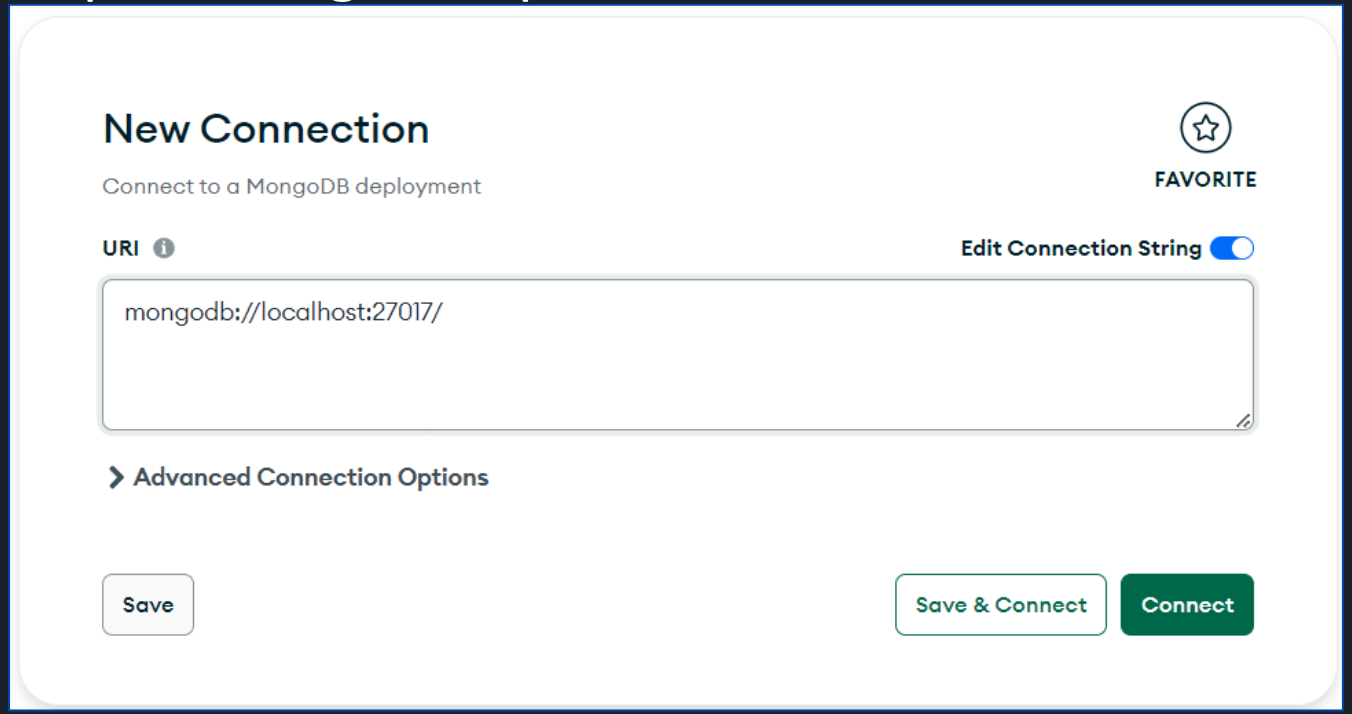
Node

Connect to the MongoDB

```
db.connect("mongodb://127.0.0.1:27017/todoapp")
  .then(() => {
    console.log("Mongo Connected");
  })
  .catch((err) => console.error(err));
```

Node

Open MongoCompuss



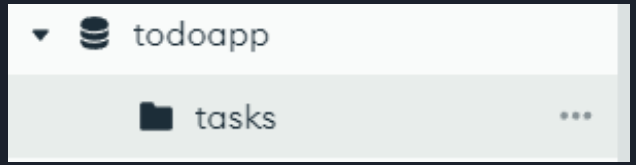
Node

Finally, Listen to the PORT:-

```
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Node

Then start the server,



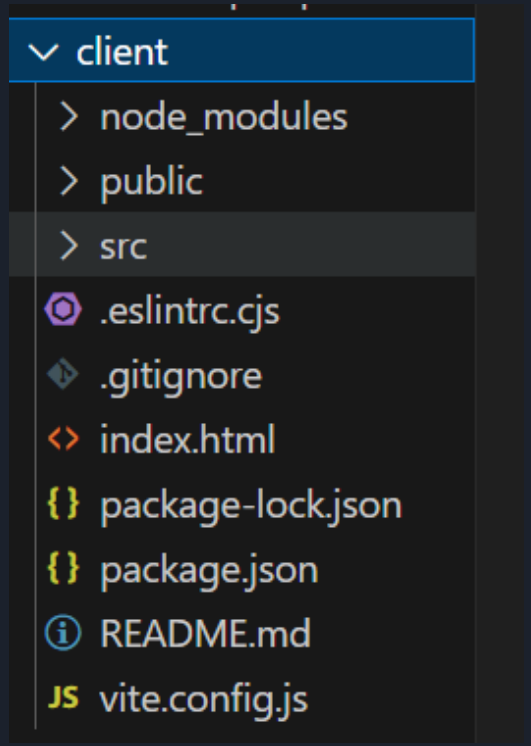

```
PS H:\MERN-semi\server> nodemon server.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 5000
Mongo Connected
```

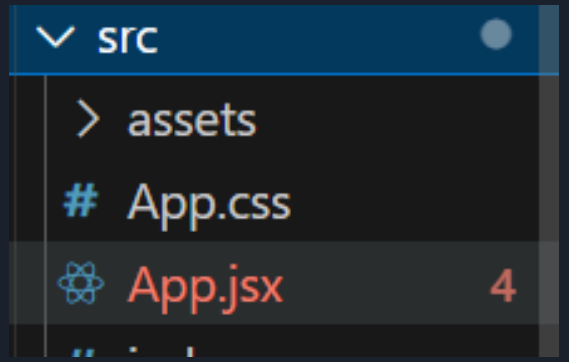
Congratulations!!!

Okie,What Next?

React

Design a frontend





React

Import Necessary Packages

rver.js | App.jsx 4, M | routes.js M

t > src > App.jsx > App > useEffect() callback

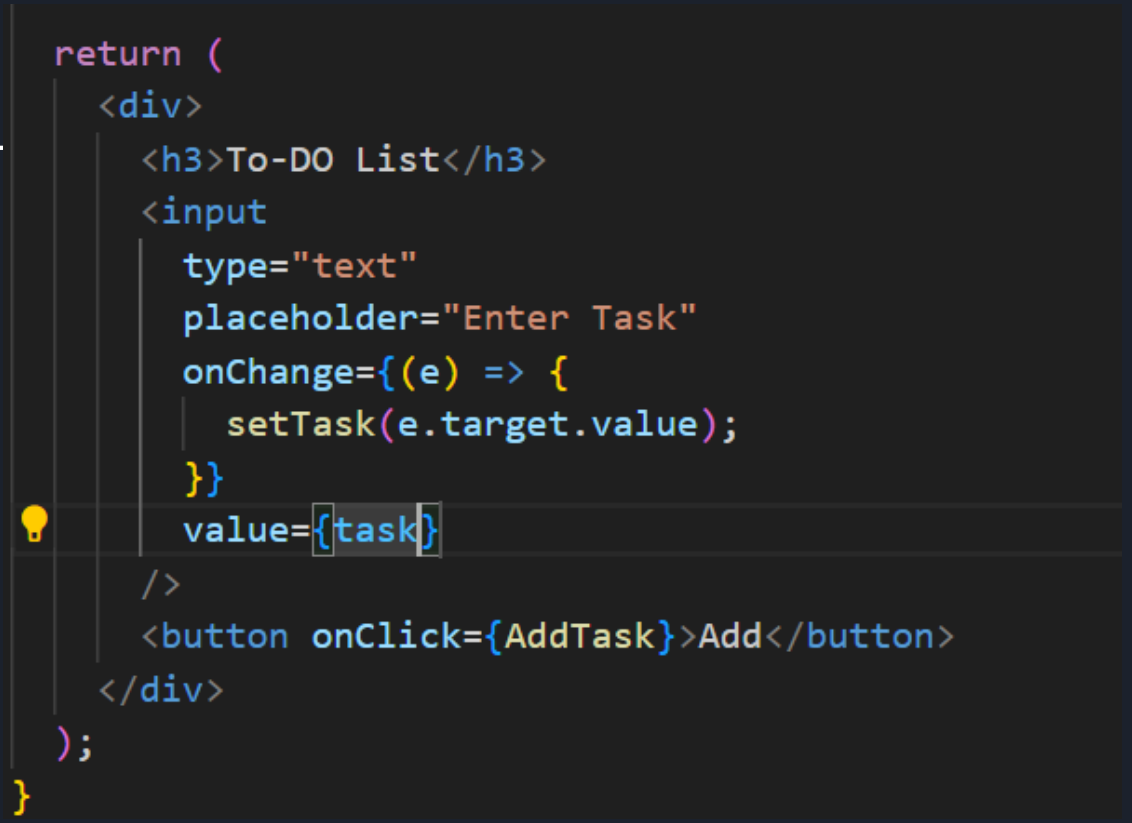
```
import { useState, useEffect } from "react";
import axios from "axios";
import "./App.css";
```

Create a variable :-

```
function App() {
  const [task, setTask] = useState("");
```

React

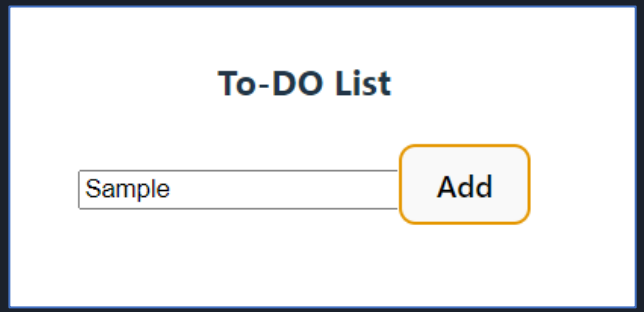
Create a element :-



```
return (
  <div>
    <h3>To-DO List</h3>
    <input
      type="text"
      placeholder="Enter Task"
      onChange={(e) => {
        setTask(e.target.value);
      }}
      value={task}
    />
    <button onClick={AddTask}>Add</button>
  </div>
);
}
```

React

Create a element :-



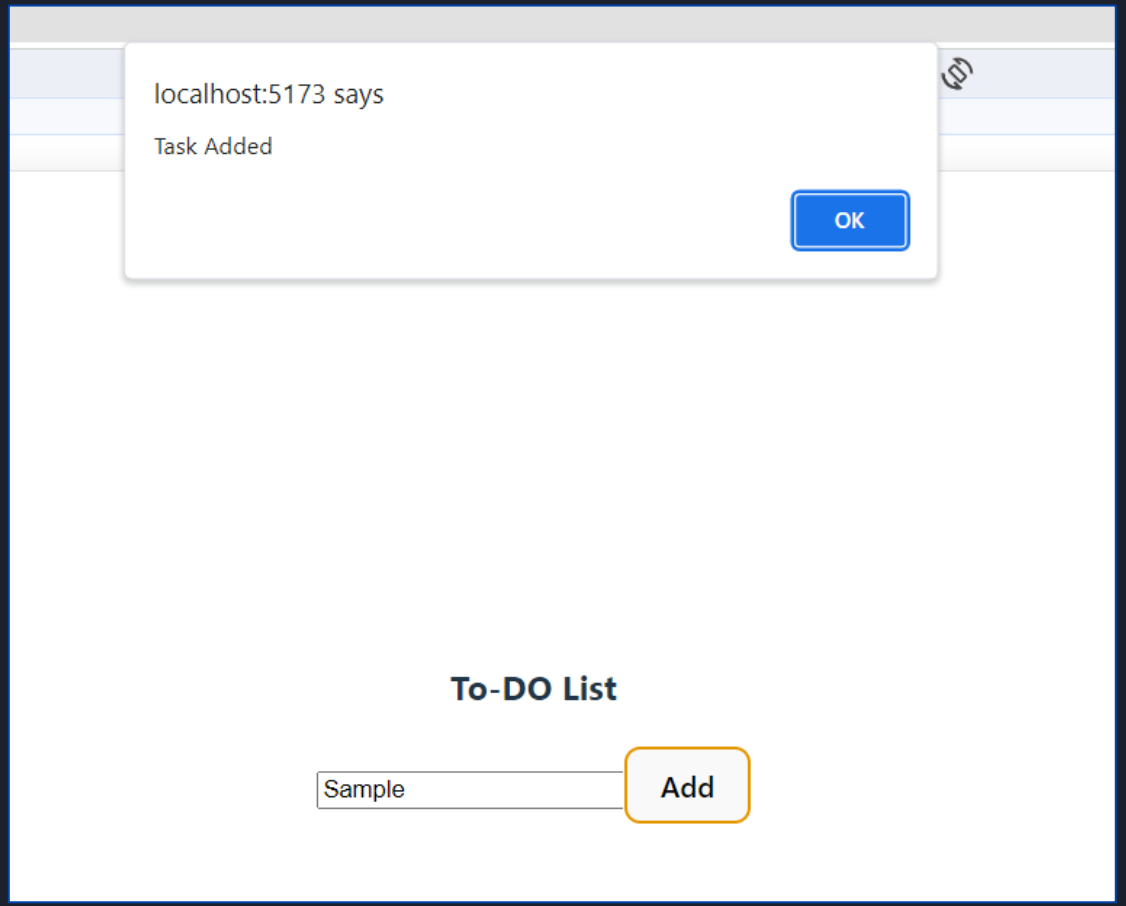

```
return (
  <div>
    <h3>To-DO List</h3>
    <input
      type="text"
      placeholder="Enter Task"
      onChange={(e) => {
        setTask(e.target.value);
      }}
      value={task}
    />
    <button onClick={AddTask}>Add</button>
  </div>
);
}
```

React

Create a fetch request for Create a element

```
const AddTask = async () => {
  await axios
    .post("http://localhost:5000/api/tasks", { text: task })
    .then((req, res) => {
      alert("Task Added");
      setTask("");
      console.log(res.data);
    })
    .catch((err) => console.error(err));
};
```

React



todoapp.tasks

1 DOCUMENTS 1 INDEXES

Documents | Aggregations | Schema | Indexes | Validation

Filter | Type a query: { field: 'value' } or Generate query | Explain | Reset | Find | Options

ADD DATA | EXPORT DATA | 1-1 of 1

```
_id: ObjectId('651d9dcf98c510afec191e64')
text: "Sample"
__v: 0
```

React

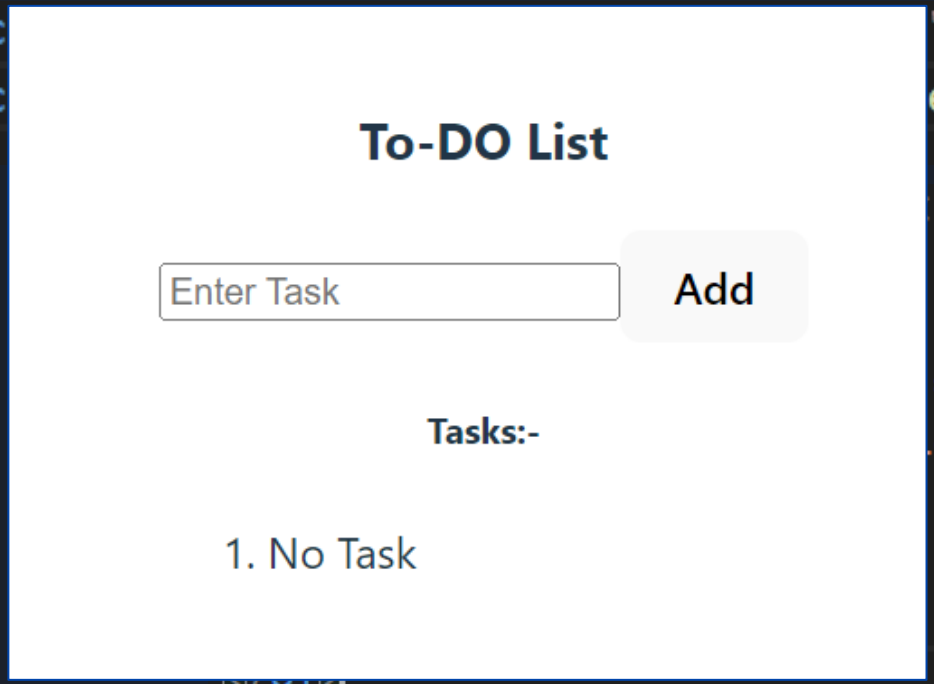
Read a element

```
const [task, setTask] = useState("");
const [taskList, setTaskList] = useState([{ text: "No Task" }]);
```

```
<button onClick={AddTask}>Add</button>
<h5>Tasks:-</h5>
<ol>
  {taskList.map((data, index) => (
    <li style={{ textAlign: "left" }} key={index}>
      {data.text}
    </li>
  ))}
</ol>
```

React

Read a element



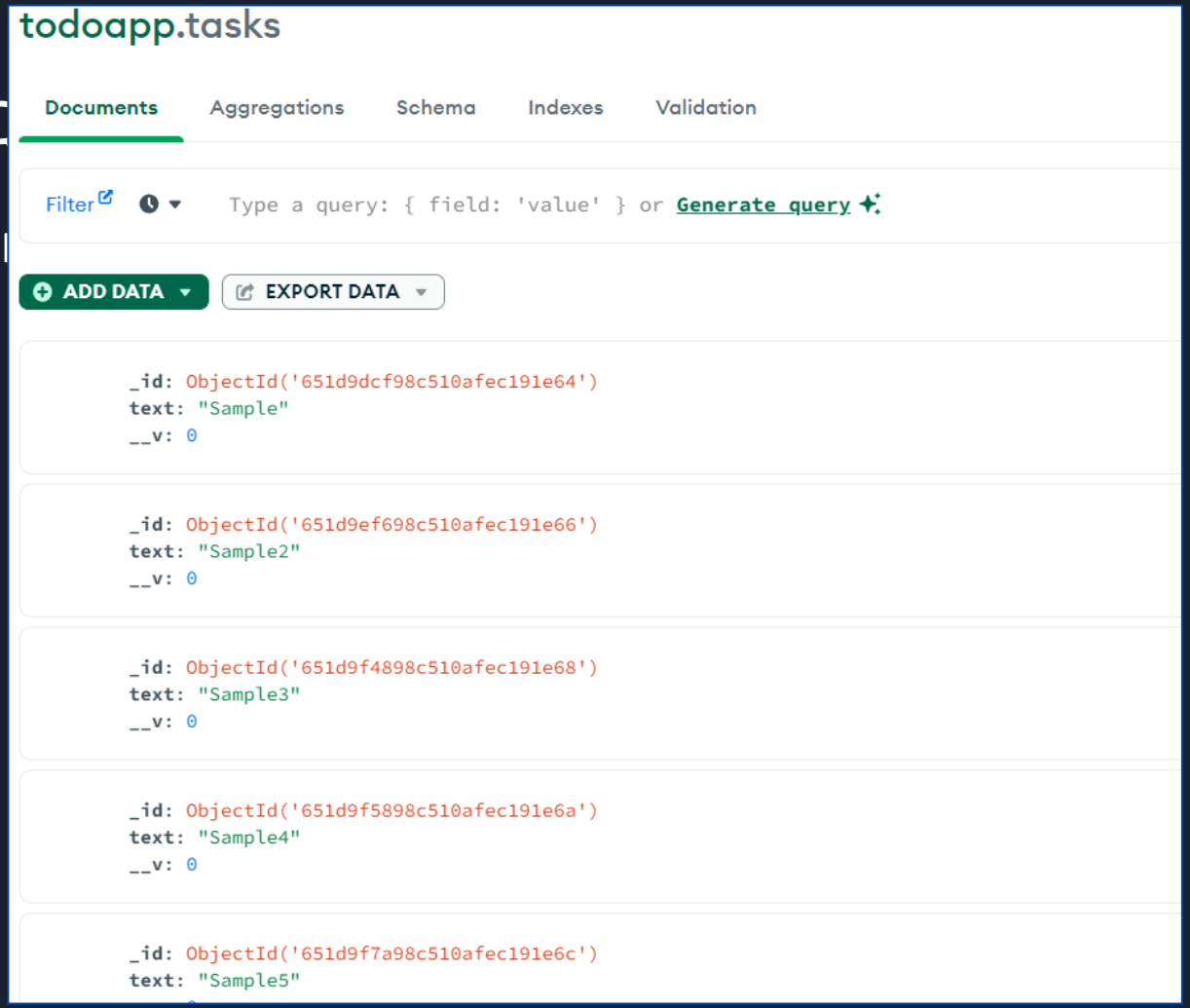

```
");
eState([{ text: "No Task" }]);
/button>
=> (
eft" }} key={index}>
```

React

Create a fetch request for Read a element

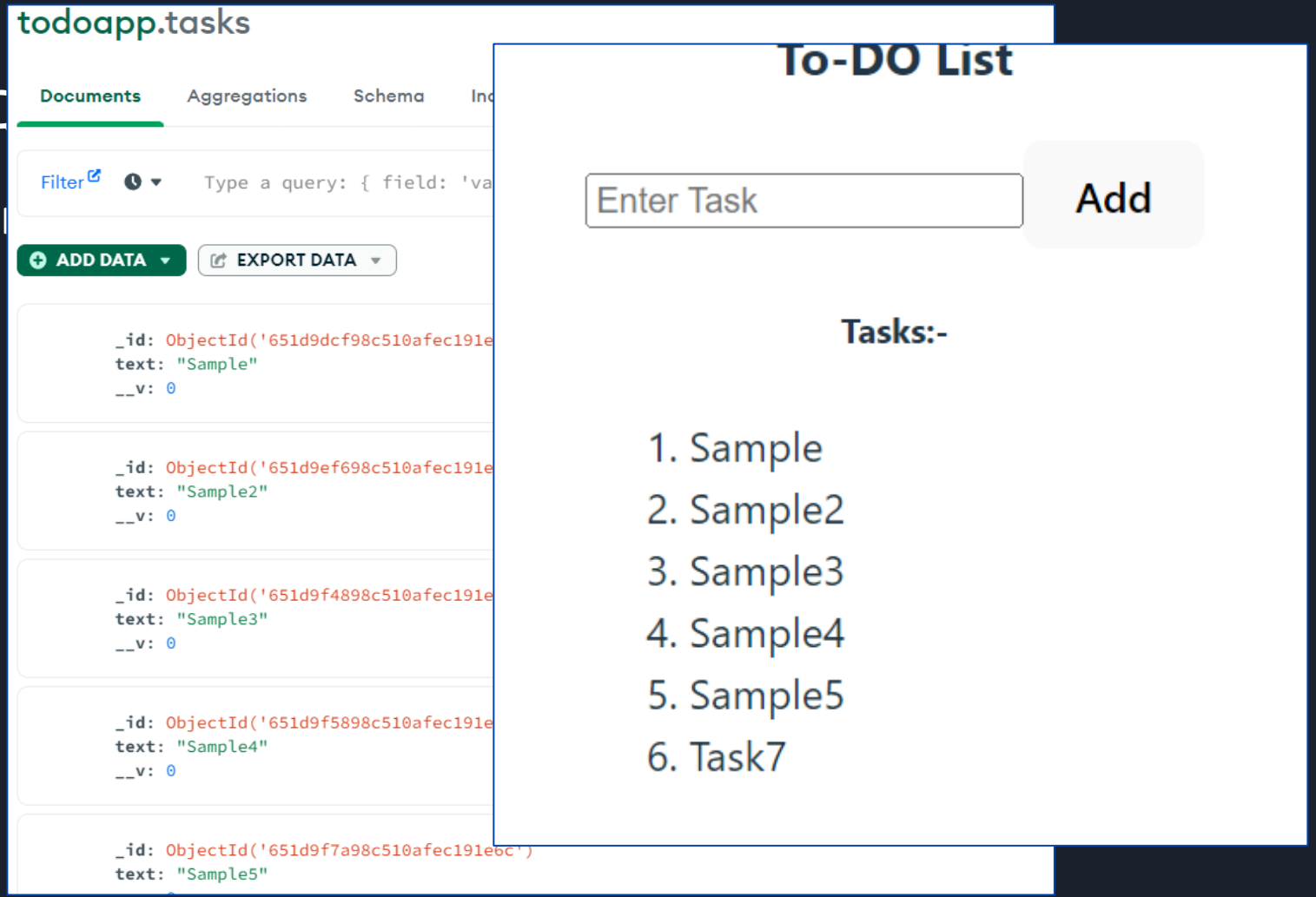
```
useEffect(() => {
  async function fetchTasks() {
    try {
      const response = await axios.get("http://localhost:5000/api/tasks");
      setTaskList(response.data);
    } catch (err) {
      console.error(err);
    }
  }
  fetchTasks();
}, []);
```

F
C


todoapp.tasks
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
_id: ObjectId('651d9dcf98c510afec191e64')
text: "Sample"
__v: 0
_id: ObjectId('651d9ef698c510afec191e66')
text: "Sample2"
__v: 0
_id: ObjectId('651d9f4898c510afec191e68')
text: "Sample3"
__v: 0
_id: ObjectId('651d9f5898c510afec191e6a')
text: "Sample4"
__v: 0
_id: ObjectId('651d9f7a98c510afec191e6c')
text: "Sample5"

F

C


todoapp.tasks
Documents
Aggregations
Schema
Filter
Type a query: { field: 'va
ADD DATA
EXPORT DATA
_id: ObjectId('651d9dcf98c510afec191e
text: "Sample"
__v: 0
_id: ObjectId('651d9ef698c510afec191e
text: "Sample2"
__v: 0
_id: ObjectId('651d9f4898c510afec191e
text: "Sample3"
__v: 0
_id: ObjectId('651d9f5898c510afec191e
text: "Sample4"
__v: 0
_id: ObjectId('651d9f7a98c510afec191e6c')
text: "Sample5"
To-DO List
Enter Task
Add
Tasks:-
1. Sample
2. Sample2
3. Sample3
4. Sample4
5. Sample5
6. Task7

React

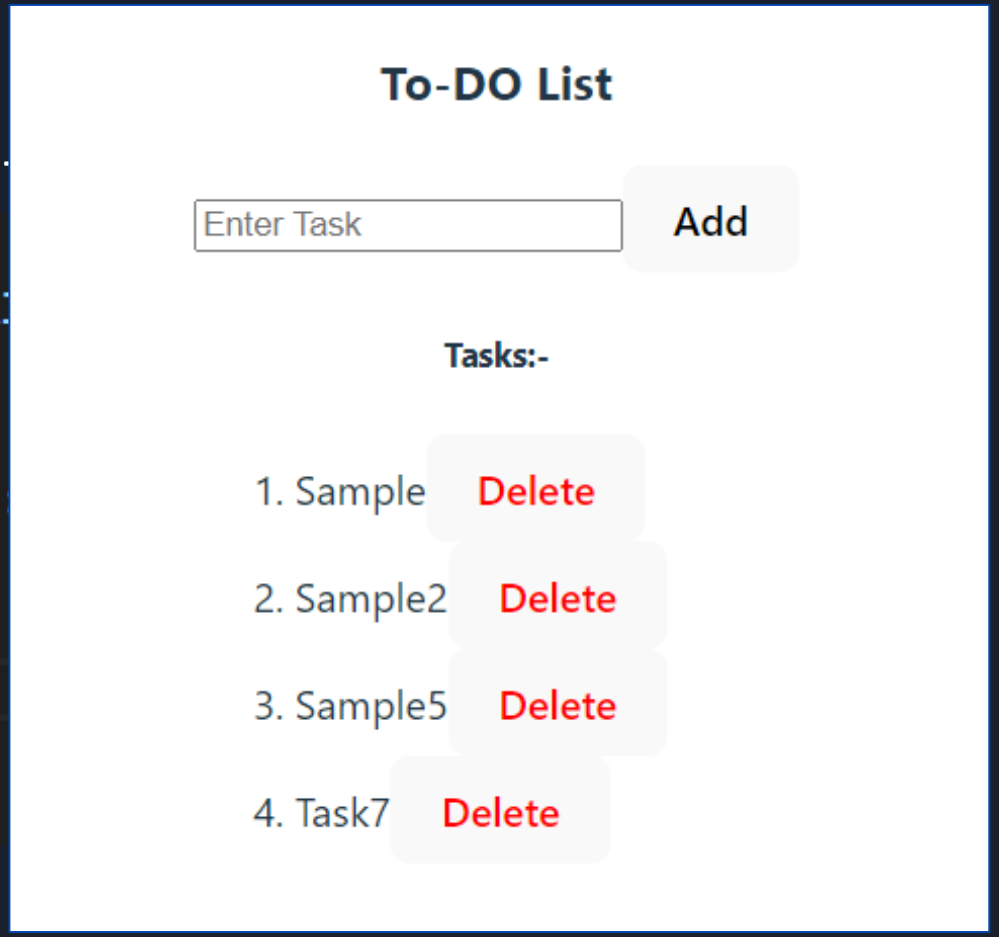
Add button for Delete a element

```
{taskList.map((data, index) => (
  <li style={{ textAlign: "left" }} key={index}>
    {data.text}
    <button
      style={{ color: "red" }}
      onClick={() => DeleteTask(data._id)}
    >
      Delete
    </button>
  </li>
```

React

Add button for Dele

```
{taskList.map((data,
  <li style={{ textA
    {data.text}
    <button
      style={{ color
      onClick={() =>
    >
      Delete
    </button>
  </li>
```



React

Create a method for delete

```
const DeleteTask = async (id) => {
  await axios
    .delete(`http://localhost:5000/api/tasks/${id}`)
    .then((req, res) => {
      console.log("Deleted Succcessfully");
    });
};
```

React

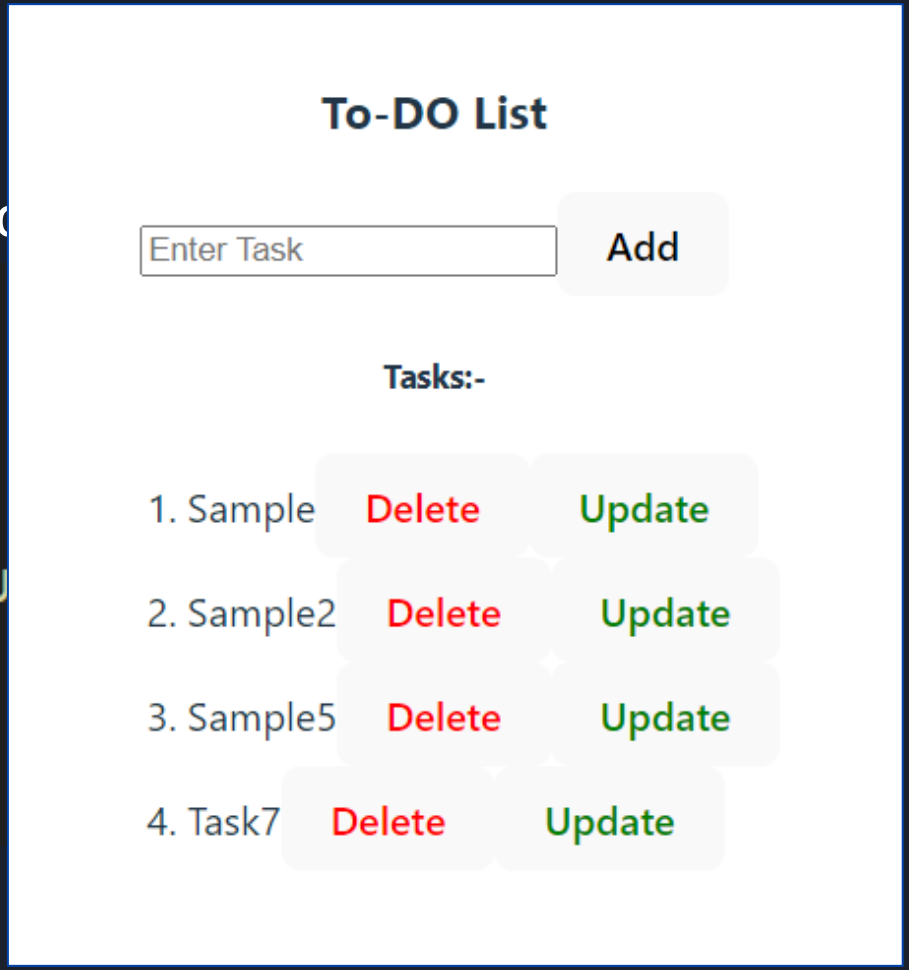
Create a button for update

```
    Delete
  </button>
  <button
    style={{ color: "Green" }}
    onClick={() => UpdateTask(data._id)}
  >
    Update
  </button>
```

React

Create a button for upd

```
    Delete
  </button>
  <button
    style={{ color:
    onClick={() => U
  >
    Update
  </button>
```



React

Create a method for update

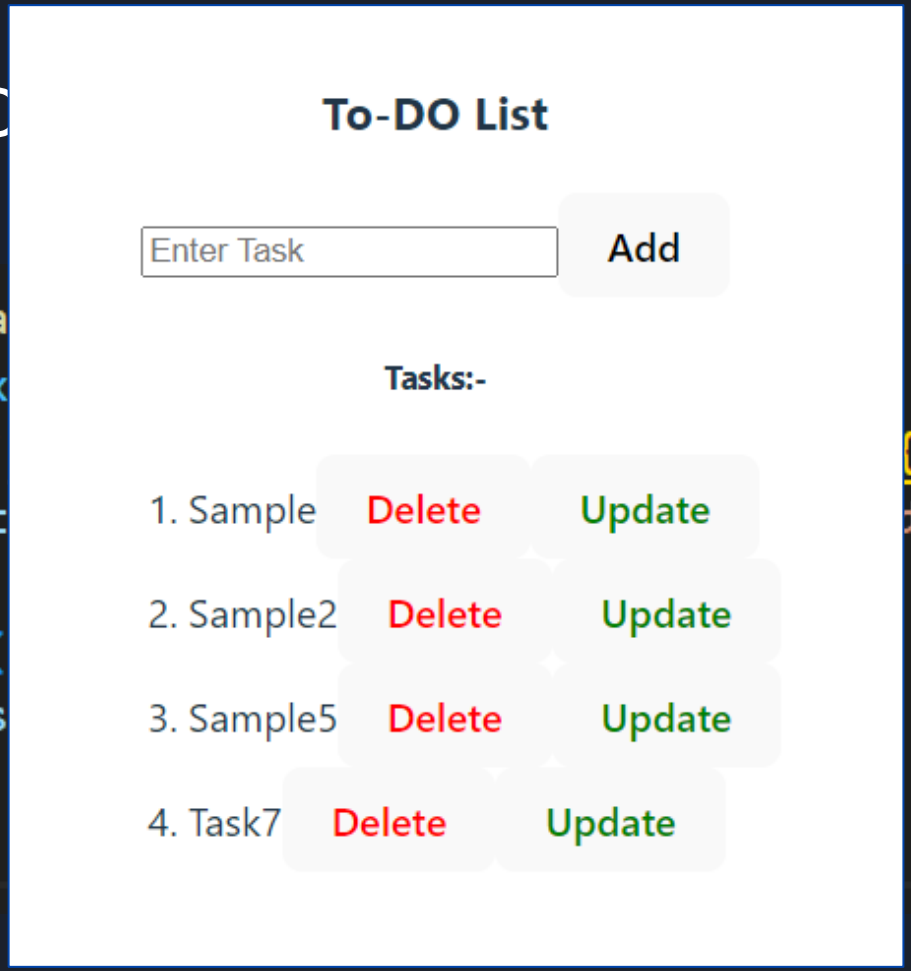
```
const UpdateTask = async (id) => {
  await axios
    .put(`http://localhost:5000/api/tasks/${id}`, {
      text: prompt("Enter the text to be Updated"),
    })
    .then((req, res) => {
      console.log("Updated Succcessfully");
    });
};
```

Reac

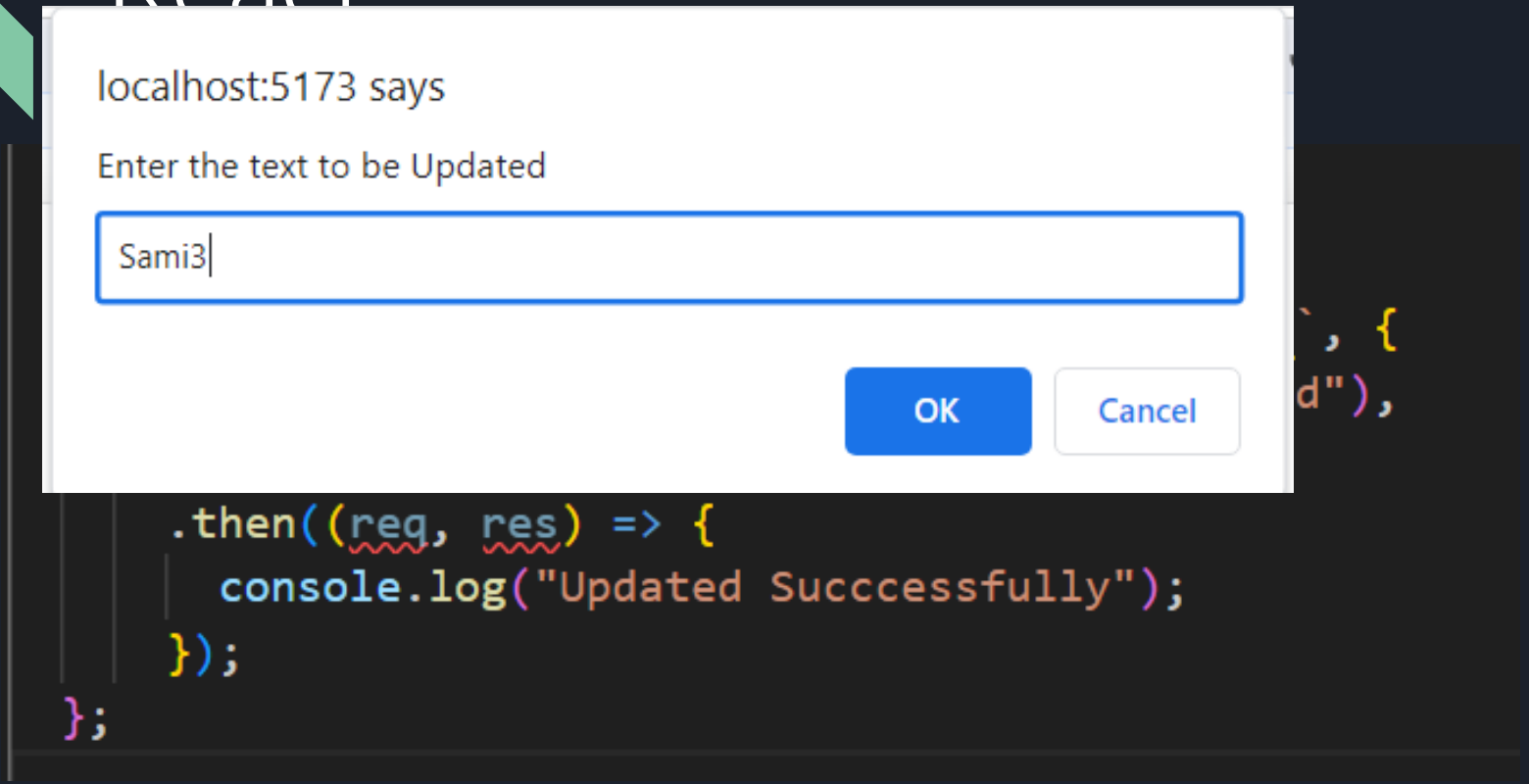
Create a

```
const Upda
  await ax
    .put(`
      text
    })
    .then(
      cons
    });
};
```

```
{id}`, {
dated"),
```



React

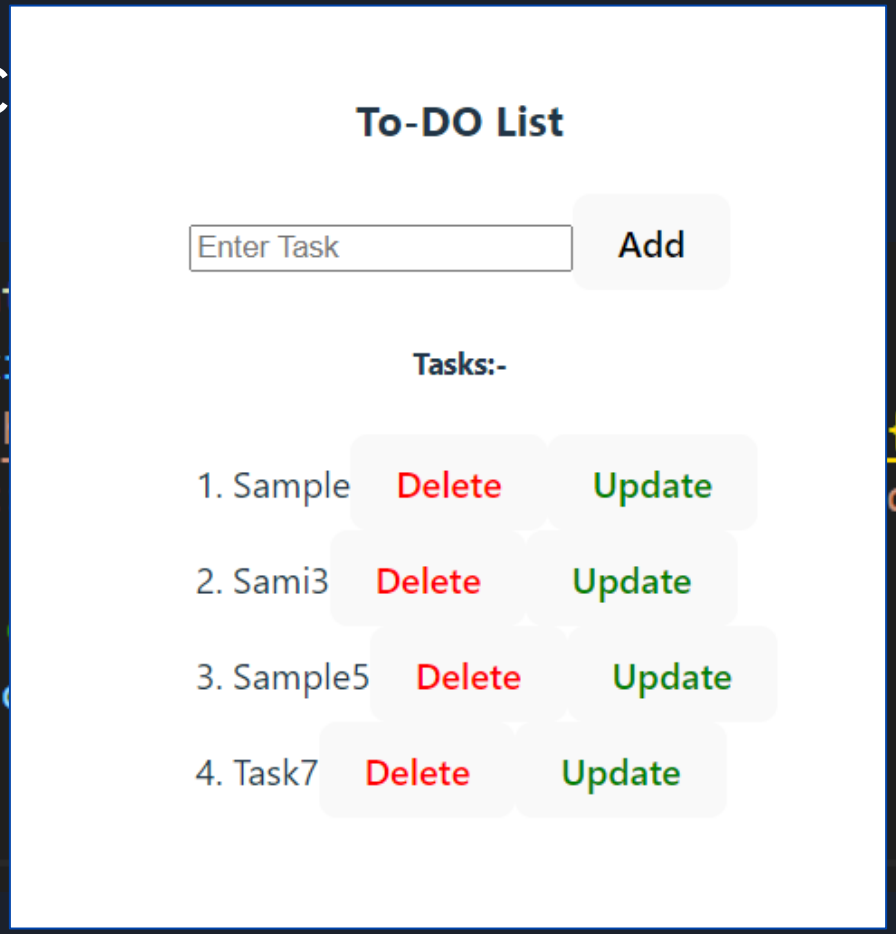

localhost:5173 says
Enter the text to be Updated
Sami3
OK
Cancel
.then((req, res) => {
console.log("Updated Succcessfully");
});
};

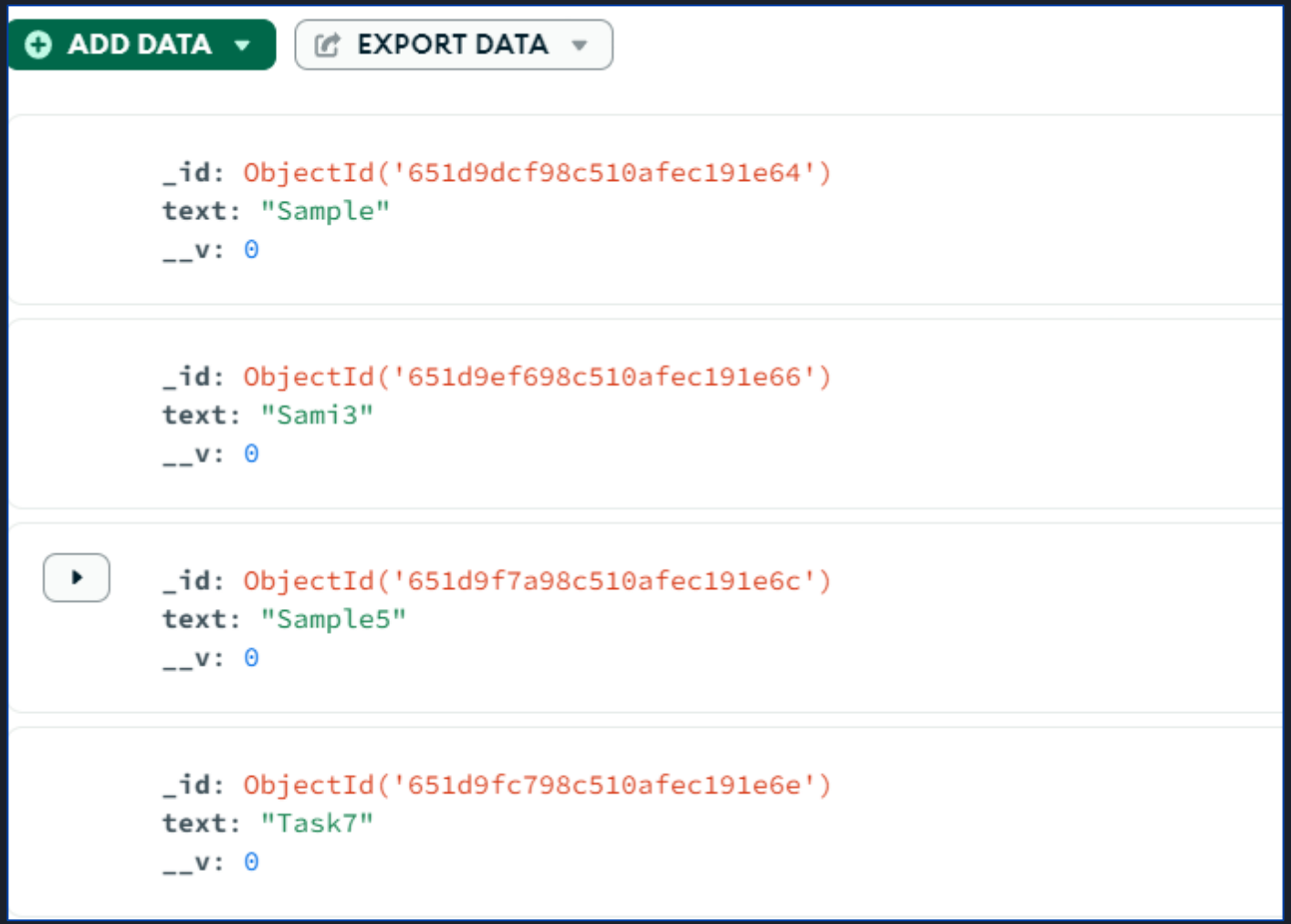
Reac

Create a

```
const Upda
  await ax
    .put(`
      text
    })
    .then(
      cons
    });
};
```

```
{id}`, {
dated"),
```




ADD DATA
EXPORT DATA
_id: ObjectId('651d9dcf98c510afec191e64')
text: "Sample"
__v: 0
_id: ObjectId('651d9ef698c510afec191e66')
text: "Sami3"
__v: 0
_id: ObjectId('651d9f7a98c510afec191e6c')
text: "Sample5"
__v: 0
_id: ObjectId('651d9fc798c510afec191e6e')
text: "Task7"
__v: 0

Congratulations!!!

OUR Final OUTPUT

Demo→→

Wait a Sec…Y?

Why>?

```
useEffect(() => {
  async function fetchTasks() {
    try {
      const response = await axios.get("http://localhost:5000/api/tasks");
      setTaskList(response.data);
    } catch (err) {
      console.error(err);
    }
  }
  fetchTasks();
}, []);
```

Which>?

```
const [task, setTask] = useState("");
const [taskList, setTaskList] = useState([{ text: "No Task" }]);
```

```
useEffect(() => {
  async function fetchTasks() {
    try {
      const response = await axios.get("http://localhos
      setTaskList(response.data);
    } catch (err) {
      console.error(err);
    }
  }
  fetchTasks();
}, [taskList]);
```

Now,Our Final Output

Demo→→

Let’s Celeberate our Victory/.. 🤝<3

Any Queries???

Thank You

For reference use:
https://github.com/rc-balaji/MERN-semi.git