**MERN PROJECT**

LINKS

[**LinkedIn**](https://www.linkedin.com/posts/adebayo-awolola-a8b6b8185_mern-aws-webdevelopment-activity-7250012940987228160-H9be?utm_source=share&utm_medium=member_desktop)

[**Github**](https://github.com/Adeoluwatosin/MERN.git)

Here's a reorganized and formatted version of the MERN project documentation, with the required code for the files:

## **MERN Stack Project Overview**

**Technologies Used:**

* **MongoDB**: NoSQL database used to store application data in a scalable, flexible way.
* **Express.js**: Backend framework for Node.js that handles routing, HTTP requests, and middleware.
* **React.js**: Frontend JavaScript library used to build interactive UIs.
* **Node.js**: Runtime environment that executes JavaScript code server-side.

### **Step-by-Step Implementation**

### **1. Set Up EC2 Instance**

1. Log in and update your Ubuntu instance.

Run the following commands:  
bash  
Copy code  
sudo apt update

sudo apt upgrade

Install Node.js:  
bash  
Copy code  
curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt-get install -y nodejs

Verify the installation:  
bash  
Copy code  
node -v

### **2. Application Code Setup**

**Initialize the Project**: Create a new directory and initialize the project.  
bash  
Copy code  
mkdir Todo

cd Todo

npm init

**Install Express**: Install Express.js to simplify routing and middleware management.  
bash  
Copy code  
npm install express

### **3. Building Routes**

**Create the Routes Folder**:  
bash  
Copy code  
mkdir routes

cd routes

touch api.js

**Define Routes in api.js**:  
javascript  
Copy code  
const express = require('express');

const router = express.Router();

router.get('/todos', (req, res, next) => {

// Logic to get all todos

});

router.post('/todos', (req, res, next) => {

// Logic to create a new todo

});

router.delete('/todos/:id', (req, res, next) => {

// Logic to delete a todo by id

});

module.exports = router;

### **4. Database Models with MongoDB and Mongoose**

**Install Mongoose**:  
bash  
Copy code  
npm install mongoose

**Create the Models Folder**:  
bash  
Copy code  
mkdir models

cd models

touch todo.js

**Define Todo Model in todo.js**:  
javascript  
Copy code  
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const TodoSchema = new Schema({

action: {

type: String,

required: [true, 'The todo text field is required']

}

});

const Todo = mongoose.model('todo', TodoSchema);

module.exports = Todo;

### **5. Connecting to MongoDB**

**Create a .env file** in the root directory with your MongoDB connection string:  
bash  
Copy code  
DB = 'mongodb+srv://<username>:<password>@<network-address>/<dbname>?retryWrites=true&w=majority'

**Update index.js to Connect to MongoDB**:  
javascript  
Copy code  
const express = require('express');

const mongoose = require('mongoose');

const routes = require('./routes/api');

require('dotenv').config();

const app = express();

const port = process.env.PORT || 5000;

mongoose.connect(process.env.DB, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log(`Database connected successfully`))

.catch(err => console.log(err));

app.use(express.json());

app.use('/api', routes);

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

### **6. Frontend Creation with React**

**Create React App**:  
bash  
Copy code  
npx create-react-app client

1. **Install Dependencies**:

Concurrently: Run multiple commands:  
bash  
Copy code  
npm install concurrently --save-dev

Nodemon: Monitor server changes:  
bash  
Copy code  
npm install nodemon --save-dev

**Update package.json** to streamline development:  
json  
Copy code  
"scripts": {

"start": "node index.js",

"start-watch": "nodemon index.js",

"dev": "concurrently \"npm run start-watch\" \"cd client && npm start\""

}

**Set Up Proxy for Frontend**: In the client folder’s package.json, add:  
json  
Copy code  
"proxy": "http://localhost:5000"

**Run the Application**:  
bash  
Copy code  
npm run dev

### **7. Creating React Components**

1. **Create Components**: Inside src/components, create three files:

Input.js:  
javascript  
Copy code  
import React, { Component } from 'react';

import axios from 'axios';

class Input extends Component {

state = { action: "" };

addTodo = () => {

const task = { action: this.state.action };

if(task.action.length > 0) {

axios.post('/api/todos', task)

.then(res => {

this.props.getTodos();

this.setState({ action: "" });

})

.catch(err => console.log(err));

}

};

handleChange = (e) => this.setState({ action: e.target.value });

render() {

return (

<div>

<input type="text" onChange={this.handleChange} value={this.state.action} />

<button onClick={this.addTodo}>Add Todo</button>

</div>

);

}

}

export default Input;

ListTodo.js:  
javascript  
Copy code  
import React from 'react';

const ListTodo = ({ todos, deleteTodo }) => (

<ul>

{todos.length > 0 ? todos.map(todo => (

<li key={todo.\_id} onClick={() => deleteTodo(todo.\_id)}>{todo.action}</li>

)) : <li>No todos left</li>}

</ul>

);

export default ListTodo;

Todo.js:  
javascript  
Copy code  
import React, { Component } from 'react';

import axios from 'axios';

import Input from './Input';

import ListTodo from './ListTodo';

class Todo extends Component {

state = { todos: [] };

componentDidMount() {

this.getTodos();

}

getTodos = () => {

axios.get('/api/todos')

.then(res => this.setState({ todos: res.data }))

.catch(err => console.log(err));

};

deleteTodo = (id) => {

axios.delete(`/api/todos/${id}`)

.then(() => this.getTodos())

.catch(err => console.log(err));

};

render() {

return (

<div>

<h1>My Todos</h1>

<Input getTodos={this.getTodos} />

<ListTodo todos={this.state.todos} deleteTodo={this.deleteTodo} />

</div>

);

}

}

export default Todo;

### **8. Styling the Application**

**Update App.js**:  
javascript  
Copy code  
import React from 'react';

import Todo from './components/Todo';

import './App.css';

const App = () => <div className="App"><Todo /></div>;

export default App;

**Style the App in App.css**:  
css  
Copy code  
.App {

text-align: center;

width: 60%;

margin: 0 auto;

}

input {

height: 40px;

width: 50%;

border-bottom: 2px solid #101113;

font-size: 1.5rem;

}

button {

height: 45px;

width: 25%;

background: #101113;

color: #787a80;

font-size: 25px;

}

ul {

list-style: none;

padding: 15px;

background: #171a1f;

}

li {

padding: 15px;

background: #282c34;

}