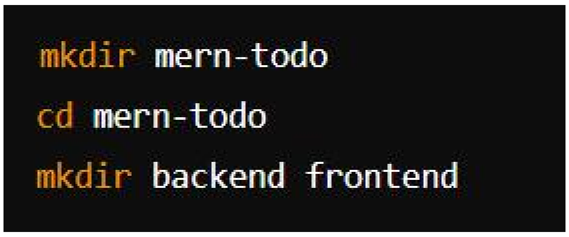
: Develop a Todo application using the MERN stack. Include setup instructions, code snippets, and screenshots showing the application’s functionality.

Developing a Todo application using the MERN stack involves setting up the MongoDB database, Express.js server, React.js frontend, and Node.js runtime.

Step 1: Setup Project Structure

Create a project directory and initialize two folders for the backend and frontend.

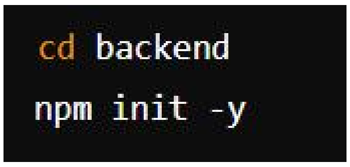


```
mkdir mern-todo
cd mern-todo
mkdir backend frontend
```

Step 2: Backend Setup

Initialize Node.js Project

Navigate to the backend directory and initialize a Node.js project.



```
cd backend
npm init -y
```

Install Dependencies

Install the necessary dependencies for the backend.

```
npm install express mongoose cors dotenv
npm install --save-dev nodemon
```

Setup Environment Variables

Create a `.env` file to store your environment variables.

```
touch .env
```

Add the following content to the .env file:

PORT=5000

MONGO_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/todoapp?retryWrites=true&w=majority

Replace <username> and <password> with your MongoDB Atlas credentials.

Setup Express Server

Create a new file `server.js` in the backend directory and add the following code:

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const dotenv = require('dotenv');

```
dotenv.config();
const app = express();
app.use(cors());
app.use(express.json());
const PORT = process.env.PORT || 5000;
const MONGO_URI = process.env.MONGO_URI;
mongoose.connect(MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log('Connected to MongoDB');
}).catch((error) => {
  console.error('Error connecting to MongoDB', error);
});

const todoSchema = new mongoose.Schema({
  title: String,
  completed: Boolean,
});
const Todo = mongoose.model('Todo', todoSchema);
app.get('/todos', async (req, res) => {
  const todos = await Todo.find();
  res.json(todos);
});
app.post('/todos', async (req, res) => {
  const newTodo = new Todo({
    title: req.body.title,
    completed: false,
  });
  await newTodo.save();
  res.json(newTodo);
});

app.delete('/todos/:id', async (req, res) => {
  const { id } = req.params;
  await Todo.findByIdAndDelete(id);
  res.json({ message: 'Todo deleted' });
});

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

Run the Backend Server

Update the `scripts` section of your `package.json` to include the following:

```
"scripts": {
  "start": "node server.js",
  "dev": "nodemon server.js"
}
```

Start the backend server.

```
npm run dev
```

Step 3: Frontend Setup

Initialize React Project

Navigate to the frontend directory and create a new React project using Create React App.

```
cd ../frontend
npx create-react-app .
```

Install Axios

Install Axios for making HTTP requests

```
npm install axios
```

Setup Todo Component

Replace the content of `src/App.js` with the following code:

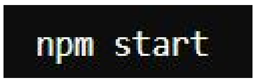
```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function App() {
  const [todos, setTodos] = useState([]);
  const [title, setTitle] = useState('');

  useEffect(() => {
    async function fetchTodos() {
      const response = await axios.get('http://localhost:5000/todos');
      setTodos(response.data);
    }
    fetchTodos();
  }, []);

  const addTodo = async () => {
    const response = await axios.post('http://localhost:5000/todos', { title });
    setTodos([...todos, response.data]);
    setTitle('');
  };
```

```
 const deleteTodo = async (id) => {
   await axios.delete(`http://localhost:5000/todos/${id}`);
   setTodos(todos.filter(todo => todo._id !== id));
 };

 return (
   <div className="App">
     <h1>Todo List</h1>
     <input
       type="text"
       value={title}
       onChange={(e) => setTitle(e.target.value)}
     />
     <button onClick={addTodo}>Add Todo</button>
     <ul>
       {todos.map((todo) => (
         <li key={todo._id}>
           {todo.title}
           <button onClick={() => deleteTodo(todo._id)}>Delete</button>
         </li>
       ))}
     </ul>
   </div>
```

```
  );
}

export default App;
```

Run the Frontend

Start the React development server.

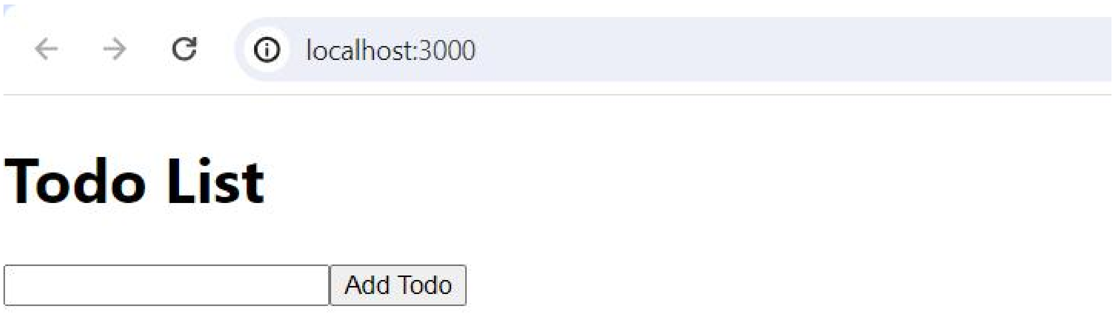


```
npm start
```

Step 4: Running the Application

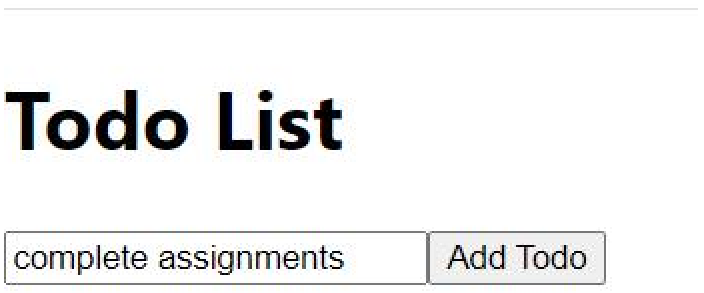
Now, you should have both the backend and frontend servers running. Open your browser and navigate to http://localhost:3000 to see your Todo application in action.

Screenshots:


localhost:3000

Todo List

Add Todo


Todo List

complete assignments

Add Todo