Priyanshu Dhyani

MERN Todo Application Documentation

Introduction

This project is a simple Todo application built using the MERN stack, which includes MongoDB, Express, React, and Node.js. The application allows users to add, view, complete, and delete tasks.

Project Structure

```
mern-todo/
├── client/
│   ├── public/
│   ├── src/
│   │   ├── components/
│   │   │   └── Todo.js
│   │   ├── App.js
│   │   ├── index.css
│   │   └── index.js
├── server/
│   ├── config/
│   ├── controllers/
│   ├── models/
│   │   └── Todo.js
│   ├── routes/
│   │   └── todos.js
│   └── server.js
├── .env
├── index.js
├── package.json
└── README.md
```

Prerequisites

- Node.js and npm installed
- MongoDB installed and running locally

Setup Instructions

Step 1: Clone the Repository

```
git clone https://github.com/your-username/mern-todo-app.git
cd mern-todo-app
```

Step 2: Install Dependencies

Backend

```
cd server
npm install
```

Frontend

```
cd client
npm install
```

Step 3: Environment Variables

Create a .env file in the root of the project and add the following:

```
MONGO_URI=mongodb://localhost:27017/todos
PORT=5000
```

Step 4: Running the Application

Backend

```
cd server
npm run dev
```

Frontend

```
cd client
npm start
```

Project Details

Backend

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');

dotenv.config();

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

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

mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log("MongoDB connected"))
  .catch(err => console.error(err));

const todoRoutes = require('./routes/todos');
app.use('/todos', todoRoutes);

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

models/Todo.js

```
const mongoose = require('mongoose');
const TodoSchema = new mongoose.Schema({
  text: {
    type: String,
    required: true
  },
  completed: {
    type: Boolean,
    default: false
  }
});
module.exports = mongoose.model('Todo', TodoSchema);
```

routes/todos.js

```
const express = require('express');
const Todo = require('../models/Todo');

const router = express.Router();

// Get all todos
router.get('/', async (req, res) => {
  const todos = await Todo.find();
  res.json(todos);
});

// Create a new todo
router.post('/', async (req, res) => {
  const newTodo = new Todo({
    text: req.body.text
  });
  const savedTodo = await newTodo.save();
  res.json(savedTodo);
});

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

// Toggle complete status
router.patch('/:id', async (req, res) => {
  const { id } = req.params;
  const todo = await Todo.findById(id);
  todo.completed = !todo.completed;
  await todo.save();
  res.json(todo);
});
module.exports = router;
```

Frontend

components/Todo.js

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './Todo.css';

const Todo = () => {
  const [todos, setTodos] = useState([]);
  const [text, setText] = useState(");

  useEffect(() => {
    axios.get('http://localhost:5000/todos')
      .then(response => setTodos(response.data))
      .catch(error => console.error(error));
  }, []);

  const addTodo = () => {
    axios.post('http://localhost:5000/todos', { text })
      .then(response => setTodos([...todos, response.data]))
      .catch(error => console.error(error));
  };
  const deleteTodo = (id) => {
    axios.delete(http://localhost:5000/todos/${id})
      .then(() => setTodos(todos.filter(todo => todo._id !== id)))
      .catch(error => console.error(error));
  };
  const toggleComplete = (id) => {
    const todo = todos.find(todo => todo._id === id);
    axios.patch(http://localhost:5000/todos/${id}, { completed: !todo.completed })
      .then(response => {
        const updatedTodos = todos.map(todo =>
          todo._id === id ? response.data : todo
        );
        setTodos(updatedTodos);
      })
      .catch(error => console.error(error));
  };
  return (
    <div className="todo-container">
      <h1>Todo List</h1>
      <div className="todo-input">
        <input
          type="text"
          value={text}
          onChange={e => setText(e.target.value)}
          placeholder="Add a new task"
        />
        <button onClick={addTodo}>Add</button>
      </div>
      <ul className="todo-list">
        {todos.map(todo => (
          <li key={todo._id} className={todo.completed ? 'completed' : ''}>
            {todo.text}
            <div>
              <button className="complete-btn" onClick={() => toggleComplete(todo._id)}>
                {todo.completed ? 'Undo' : 'Complete'}
              </button>
              <button className="delete-btn" onClick={() => deleteTodo(todo._id)}>Delete</button>
            </div>
          </li>
        ))}
      </ul>
    </div>
  );
};
export default Todo;
```

App.js

```
import React from 'react';
import Todo from './components/Todo';
import './App.css';

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

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

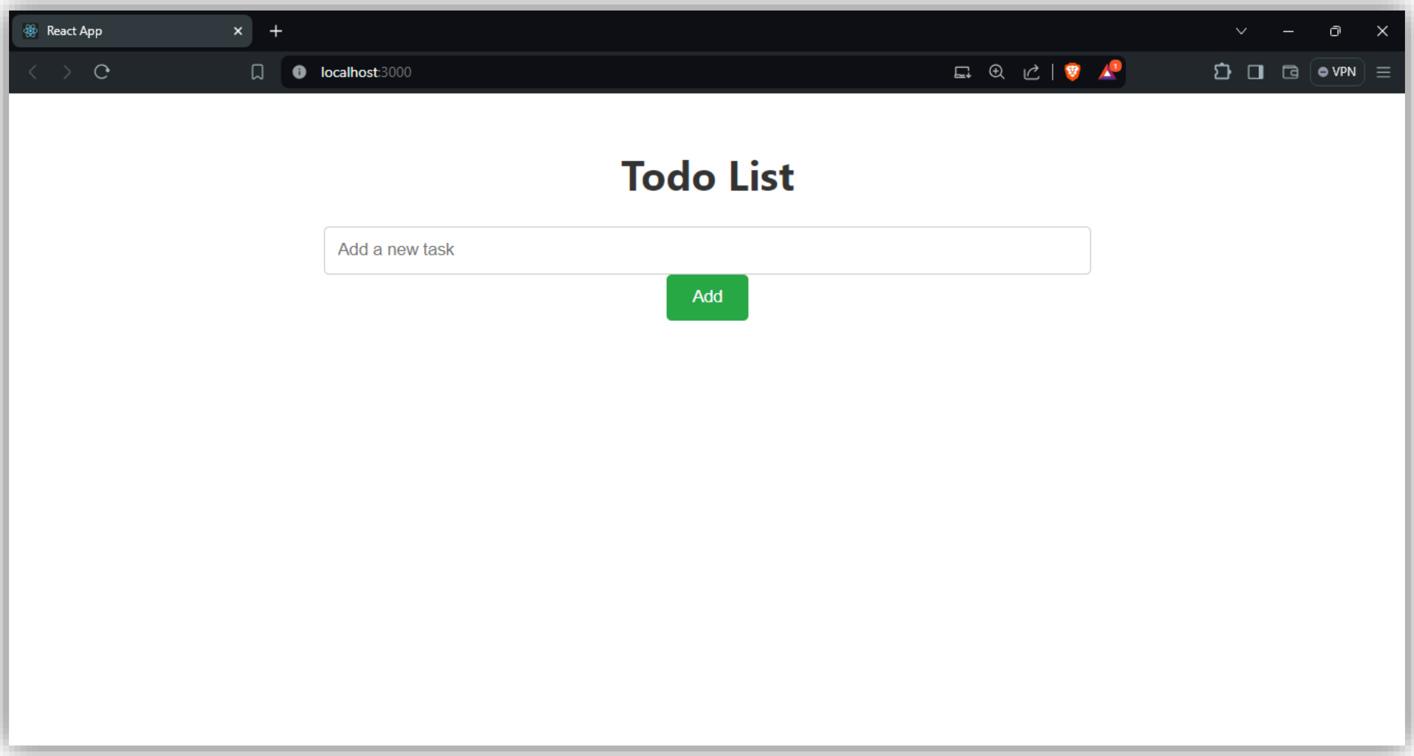
Todo.css

```
.todo-container {
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
  text-align: center;
}

.todo-container h1 {
  margin-bottom: 20px;
  color: 333;
}
```

```
.todo-input {
  margin-bottom: 20px;
}

.todo-input input {
  padding: 10px;
  width: calc(100% - 22px);
  margin-right: 10px;
  border: 1px solid ccc;
  border-radius: 4px;
}

.todo-input button {
  padding: 10px 20px;
  border: none;
  background-color: 28a745;
  color: white;
  border-radius: 4px;
  cursor: pointer;
}

.todo-input button:hover {
  background-color: 218838;
}

.todo-list {
  list-style: none;
  padding: 0;
}

.todo-list li {
  padding: 10px;
  border-bottom: 1px solid ddd;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.todo-list li.completed {
  text-decoration: line-through;
  color: 777;
}

.todo-list button {
  margin-left: 10px;
  padding: 5px 10px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.todo-list button.complete-btn {
  background-color: 007bff;
  color: white;
}

.todo-list button.complete-btn:hover {
  background-color: 0056b3;
}

.todo-list button.delete-btn {
  background-color: dc3545;
  color: white;
}

.todo-list button.delete-btn:hover {
  background-color: c82333;
}
```

Screenshots:

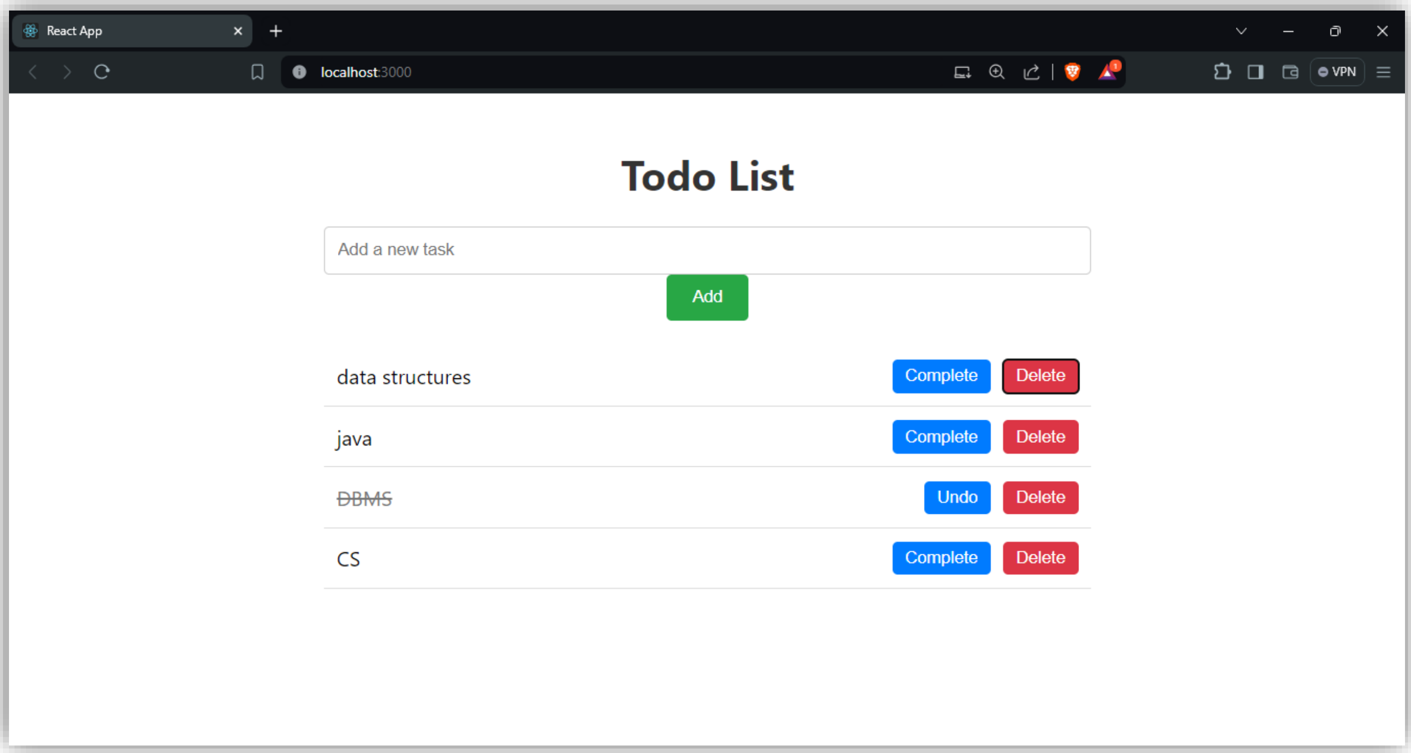
Priyanshu Dhyani



Todo List

Add a new task

Add



Todo List

Add a new task

Add

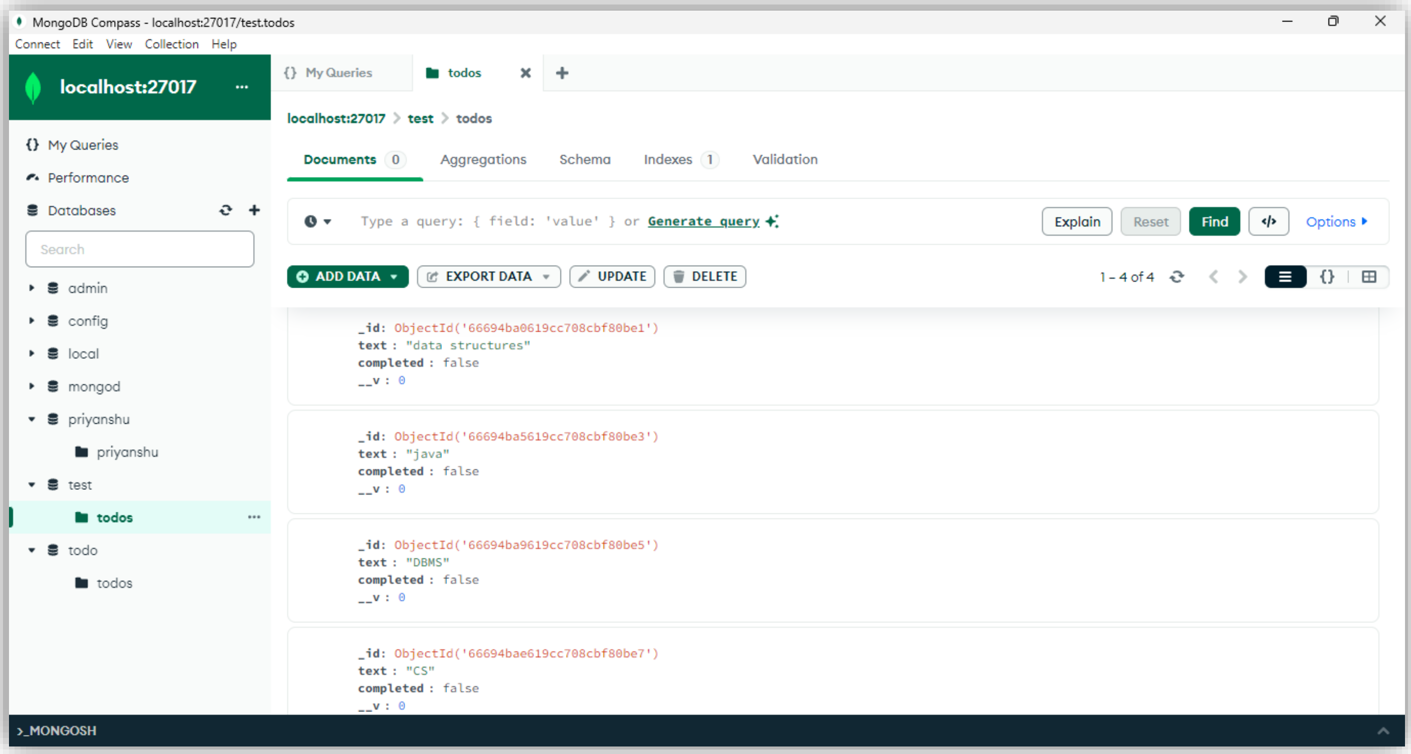
data structures Complete Delete

java Complete Delete

~~DBMS~~ Undo Delete

CS Complete Delete

Priyanshu Dhyani



```
_id: ObjectId('66694ba0619cc708cbf80be1')
text : "data structures"
completed : false
__v : 0

_id: ObjectId('66694ba5619cc708cbf80be3')
text : "java"
completed : false
__v : 0

_id: ObjectId('66694ba9619cc708cbf80be5')
text : "DBMS"
completed : false
__v : 0

_id: ObjectId('66694bae619cc708cbf80be7')
text : "CS"
completed : false
__v : 0
```