

Backend (Express + MongoDB)

MERN stack application where React handles user input, Express exposes a REST API, and MongoDB stores the data.

backend/index.js

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

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

// MongoDB connection
mongoose.connect("mongodb://127.0.0.1:27017/textdb");

// Schema & Model
const TextSchema = new mongoose.Schema({
  message: String
});

const TextModel = mongoose.model("Text", TextSchema);

// API to save text
app.post("/save", async (req, res) => {
  await TextModel.create({ message: req.body.message });
  res.send("Saved");
});

app.listen(5000, () => {
  console.log("Server running on port 5000");
});
```

Install backend dependencies

```
npm init -y
npm install express mongoose cors
node index.js
```

Frontend (Vite + React)

Create Vite app

```
npm create vite@latest frontend
cd frontend
npm install
npm run dev
```

frontend/src/App.jsx

```
import { useState } from "react";

function App() {
  const [text, setText] = useState("");

  const saveText = async () => {
    await fetch("http://localhost:5000/save", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ message: text })
    });
    setText("");
    alert("Saved to MongoDB");
  };

  return (
    <div style={{ padding: "40px" }}>
      <input
        type="text"
        value={text}
        onChange={(e) => setText(e.target.value)}
        placeholder="Type something"
      />
      <button onClick={saveText}>Save</button>
    </div>
  );
}

export default App;
```

Result

1. Page opens
 2. Text box appears
 3. You type text
 4. Click **Save**
 5. Text is stored in **MongoDB collection**
-

MongoDB Data (example)

```
{
  "_id": "65bxxxxx",
  "message": "Hello MongoDB"
}
```

STEP 0: Start MongoDB + Compass

1. Make sure **MongoDB service is running**
 - On Windows: MongoDB runs automatically
 - On Mac/Linux:
 - mongod
2. Open **MongoDB Compass**
3. Connect using this URI:

mongodb://127.0.0.1:27017

Click **Connect**

STEP 1: Run Backend (Express + MongoDB)

Go to backend folder

```
cd backend
```

Start backend server

```
node index.js
```

You should see:

Server running on port 5000

Keep this terminal open

STEP 2: Run Frontend (Vite + React)

Open a new terminal

```
cd frontend
```

Start Vite dev server

```
npm run dev
```

You'll get:

Local: http://localhost:5173/

Open this URL in your browser.

STEP 3: Use the App

1. Page opens
 2. Text box appears
 3. Type any text
 4. Click **Save**
 5. Alert: **Saved to MongoDB**
-

STEP 4: Verify Data in MongoDB Compass

1. In **MongoDB Compass**, look at the left sidebar
2. You'll see database:
3. textdb
4. Click **textdb**
5. Click collection:
6. texts

(Mongoose auto-creates plural lowercase name)

7. Click **Documents**

You'll see something like:

```
{
  "_id": ObjectId("65f..."),
  "message": "Hello MongoDB",
  "__v": 0
}
```

COMMON COMPASS DOUBTS

I don't see the database/collection

MongoDB creates DB **only after first insert**
Click **Save** once from the app

Why collection name is texts not Text

Mongoose auto-pluralizes model names

```
mongoose.model("Text", TextSchema);
// becomes → texts
```

React Topics Involved

You are learning these **core React concepts**:

Functional Components

```
function App() { ... }
```

Modern React standard

useState Hook

```
const [text, setText] = useState("");
```

Managing input state

Controlled Components

```
<input value={text} onChange={...} />
```

React controls form input

Event Handling

```
onClick={saveText}
```

Handling user actions

Fetch API (HTTP request)

```
fetch("http://localhost:5000/save", {...})
```

Connecting frontend to backend

JSON Data Handling

Basic CRUD (Create) Application

```
JSON.stringify({ message: text })
```

Sending structured data

EXPRESS / NODE Topics Involved

You are learning **backend fundamentals**:

Express Server Setup

```
const app = express();  
app.listen(5000);
```

Creating a server

Middleware

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

Handling request data & cross-origin access

REST API (POST method)

```
app.post("/save", ...)
```

Creating an API endpoint

Request–Response Cycle

```
req.body.message  
res.send("Saved")
```

Backend receives & responds

MongoDB + Mongoose Topics

You're learning **NoSQL database basics**:

Database Connection

```
mongoose.connect(...)
```

Schema Design

```
const TextSchema = new mongoose.Schema({  
  message: String  
});
```

Model

```
mongoose.model("Text", TextSchema);
```

Mapping JS → MongoDB

Create Operation (CRUD)

```
TextModel.create(...)
```