**MERN To-Do List Project Documentation**

**BACKEND SETUP**

**1. Install Node.js**

1. Download and install Node.js (LTS version) from [nodejs.org](https://nodejs.org/).
2. Verify the installation:
3. node -v
4. npm -v

This should display the installed versions of Node.js and npm.

**2. Initialize Backend**

1. Open a terminal and navigate to the project directory:
2. cd Desktop
3. mkdir to\_do\_backend
4. cd to\_do\_backend
5. Initialize a Node.js project:
6. npm init
7. Follow the prompts:
   * **Package name**: to\_do\_backend *(default, press Enter)*
   * **Version**: 1.0.0 *(default, press Enter)*
   * **Description**: for learn MERN
   * **Entry point**: index.js *(default, press Enter)*
   * **Test command**: *(leave blank, press Enter)*
   * **Git repository**: https://github.com/navatharshini/to\_do\_MERN.git
   * **Keywords**: *(leave blank, press Enter)*
   * **Author**: Navatharshini
   * **License**: *(default: ISC, press Enter)*
   * Confirm with yes.
8. This generates a package.json file with project metad

## ****Procedure****

### **1. Install Express**

First, ensure that Express is installed in your project directory. Run the following command in your terminal:

bash

CopyEdit

npm install express

### **2. Create the Server File**

Create a new file, for example, server.js, and add the following code to set up the Express server:

// Import Express

const express = require("express");

// Create an instance of Express

const app = express();

// Define a Route

app.get('/', (req, res) => {

res.send("Hello World!");

});

// Start the Server

const port = 3000;

app.listen(port, () => {

console.log("Server is listening to port " + port);

});

### **3. Run the Server**

Once you have added the code, start the server by running the following command in the terminal:

node server.js

### **4. Access the Server**

Open your web browser and visit the following URL:

http://localhost:3000/

You should see the message **"Hello World!"** displayed.

\*Backend setup Completed\*

## 7. Create Item API

### **nstall Additional Dependencies**

To handle JSON data, install the body-parser middleware (built into Express 4.16+):

npm install cors

Modify server.js to include JSON parsing:

//create a to\_do item

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

    const {title,description}=req.body;

    const newTodo={

        id:todos.length +1,

        title,

        description

    };

    todos.push(newTodo);

    console.log(todos);

    res.status(201).json(newTodo);

})

### **Test the API**

1. Start the server:

node server.js

1. Use Postman or cURL to send a POST request:
   * **URL:** http://localhost:3000/todos
   * **Method:** POST
   * **Body (JSON):**
   * {
   * "title": "Complete MERN Project",
   * "description": "Finish the To-Do List backend"

}

### **Expected Response:**

{

"id": 1,

"title": "Complete MERN Project",

"description": "Finish the To-Do List backend"

}

✅ **Item creation API completed!**

---------------------------------------------------------------------------------

### **Route: Get All Items**

#### **Endpoint:** /todos

* **Method:** GET
* **Description:** This route retrieves and returns a list of all the to-do items stored in the system. The response will be in JSON format.

#### **Code Implementation:**

app.get("/todos", (req, res) => {

res.json(todos);

});

**Integrating MongoDB**

#### **Step 1: Install MongoDB Locally**

1. **Download MongoDB**:
   * Visit the official MongoDB website to download the MongoDB Community Edition installer:
     + [MongoDB Download Center](https://www.mongodb.com/try/download/community)
   * Select the appropriate version for your operating system and download the MSI installer.
2. **Install MongoDB**:
   * Run the downloaded MSI installer and follow the on-screen instructions.
   * During installation, ensure that you select the option to **Install MongoDB as a Service**. This ensures MongoDB will run automatically when your computer starts.
   * Also, select the option to install **MongoDB Compass**, MongoDB's graphical interface for managing your databases.

#### **Step 2: Start MongoDB Service**

* MongoDB should start automatically as a service after installation. You can check if MongoDB is running by opening a terminal (Command Prompt or PowerShell) and typing the following command:

mongod

This command starts the MongoDB server and will display logs in the terminal.

#### **Step 3: Connect to MongoDB Using MongoDB Compass**

1. **Launch MongoDB Compass**:
   * Open **MongoDB Compass** from the Start menu (or search for it).
2. **Connect to Local MongoDB Instance**:
   * In MongoDB Compass, the **Connection** window will appear.
   * By default, it should attempt to connect to mongodb://localhost:27017. This is the default connection string for a local MongoDB server.
   * Click **Connect** to establish a connection to your local MongoDB instance.

## ****Integrating MongoDB with Mongoose****

### **Step 1: Install Mongoose**

To interact with MongoDB, install Mongoose using npm:

bash

CopyEdit

npm install mongoose

### **Step 2: Connect to MongoDB**

Add the following code in your server.js file to connect your Node.js app to MongoDB:

* //Connecting moongoose
* mongoose.connect("mongodb://localhost:27017/mern-app")
* .then (()=>{
* console.log("DB connected!")
* })
* .catch((err)=>{
* console.log(err)
* })
* **localhost** → Refers to your local machine.
* **27017** → Default MongoDB port.
* **mern-app** → Name of the database (can be changed).

### **Step 3: Define a Mongoose Schema**

Create a schema for your to-do list:

//creating schema

const todoSchema =new mongoose.Schema({

title:String,

description:String

})

This schema defines the structure for storing to-do items in MongoDB.

Create a model for your to-do list

const todoModel =mongoose.model("todo",todoSchema)

**### Create a To-Do Item API (POST /todos)\_\_\_\_\_\_\_\_(work with MongoDB)**

#### **Description:**

This API allows users to create a new To-Do item and save it to the MongoDB database.

#### **Endpoint:**

POST /todos

#### **Request Body (JSON Format):**

{

"title": "Buy groceries",

"description": "Milk, Bread, Eggs"

}

app.post('/todos', async (req, res) => {

    const { title, description } = req.body;

    try {

        // Create a new To-Do item

        const newTodo = new todoModel({ title, description });

        // Save to MongoDB

        await newTodo.save();

        // Respond with the created item

        res.status(201).json(newTodo);

    } catch (error) {

        console.log(error);

        res.status(500).json({ message: "Internal Server Error" });

    }

});

Handling Errors

Make this Change

//creating schema

const todoSchema =new mongoose.Schema({

title:{

    required:true,

    type:String

},

description:String

})

//create a to\_do item

//save on mongodb

app.post('/todos', async (req, res) => {

    const { title, description } = req.body;

    try {

        const newTodo = new todoModel({ title, description })

        await newTodo.save();

        res.status(201).json(newTodo);

    }

    catch (error) {

        console.log(error);

        res.status(500).json({ message: error.message })

    }

})

Get All Item (API) from MongoDB

//get all items

app.get("/todos", async (req, res) => {

    try {

        const todos = await todoModel.find();

        res.json(todos)

    } catch (error) {

        console.log(error);

        res.status(500).json({ message: error.message })

    }

})

// 1. Start the server using Nodemon:

// - Open a terminal and navigate to your project folder

// - Run the following command to start the server with automatic restarts:

// npx nodemon server.js

Update Items API

// // Update to\_do item

app.put("/todos/:id",async(req,res)=>{

    try{

        const { title, description } = req.body;

        const id=req.params.id;

        const updatedTodo= await todoModel.findByIdAndUpdate(

    id,

    {title,description},

    {new: true}

        )

    if (!updatedTodo){

        return res.status(404).json({message:"Todo not found"})

    }

     // Log the updated document for debugging

     console.log("Updated Todo: ", updatedTodo);

       // Only return the necessary properties

       res.status(200).json(updatedTodo);

    }catch(error){

        console.log(error);

        res.status(500).json({ message: error.message })

    }

})

Delete Item API

//delete to\_do item

app.delete('/todos/:id',async (req,res)=>{

    try{

        const id =req.params.id;

    await todoModel.findByIdAndDelete(id);

    res.status(205).end();

    }catch(error){

        console.log(error);

        res.status(500).json({ message: error.message })

    }

})

**FRONTEND SETUP**

 main.jsx (for React) instead of index.js

 vite.config.js instead of webpack.config.js (if applicable)

React+vite

**npm create vite@latest to-do --template react**