create a simple to do list app using node.js and mongodb

**ChatGPT said:**

Got it! Let’s build a **simple To-Do List app** using **Node.js (Express.js)** and **MongoDB (Mongoose)**. I’ll give you a minimal working example you can run locally.

**1. Setup**

Make sure you have:

* Node.js installed
* [MongoDB](https://www.mongodb.com/try/download/community) running locally (or use [MongoDB Atlas](https://www.mongodb.com/atlas))

Create a project folder:

mkdir todo-app

cd todo-app

npm init -y

npm install express mongoose ejs body-parser

**2. Project Structure**

todo-app/

│── server.js

│── models/

│ └── Todo.js

│── views/

│ └── index.ejs

**3. MongoDB Model (models/Todo.js)**

const mongoose = require("mongoose");

const TodoSchema = new mongoose.Schema({

task: { type: String, required: true },

completed: { type: Boolean, default: false }

});

module.exports = mongoose.model("Todo", TodoSchema);

**4. Express Server (server.js)**

const express = require("express");

const mongoose = require("mongoose");

const bodyParser = require("body-parser");

const Todo = require("./models/Todo");

const app = express();

// Middleware

app.use(bodyParser.urlencoded({ extended: true }));

app.set("view engine", "ejs");

// Connect MongoDB

mongoose.connect("mongodb://127.0.0.1:27017/todo\_app", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

// Routes

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

const todos = await Todo.find();

res.render("index", { todos });

});

app.post("/add", async (req, res) => {

const newTodo = new Todo({ task: req.body.task });

await newTodo.save();

res.redirect("/");

});

app.post("/delete/:id", async (req, res) => {

await Todo.findByIdAndDelete(req.params.id);

res.redirect("/");

});

app.post("/toggle/:id", async (req, res) => {

const todo = await Todo.findById(req.params.id);

todo.completed = !todo.completed;

await todo.save();

res.redirect("/");

});

// Start server

app.listen(3000, () => console.log("Server running on http://localhost:3000"));

**5. View Template (views/index.ejs)**

<!DOCTYPE html>

<html>

<head>

<title>To-Do List</title>

<style>

body { font-family: Arial; max-width: 600px; margin: auto; padding: 20px; }

.completed { text-decoration: line-through; color: gray; }

</style>

</head>

<body>

<h1>My To-Do List</h1>

<form action="/add" method="POST">

<input type="text" name="task" placeholder="New Task" required />

<button type="submit">Add</button>

</form>

<ul>

<% todos.forEach(todo => { %>

<li>

<form action="/toggle/<%= todo.\_id %>" method="POST" style="display:inline;">

<button type="submit"><%= todo.completed ? "Undo" : "Done" %></button>

</form>

<span class="<%= todo.completed ? 'completed' : '' %>"><%= todo.task %></span>

<form action="/delete/<%= todo.\_id %>" method="POST" style="display:inline;">

<button type="submit">Delete</button>

</form>

</li>

<% }) %>

</ul>

</body>

</html>

**6. Run the App**

Start MongoDB (if not already running), then:

node server.js

Visit 👉 [**http://localhost:3000**](http://localhost:3000) in your browser.

✅ Features:

* Add new tasks
* Mark tasks as completed / undo
* Delete tasks

Do you want me to also add **edit/update functionality** for tasks, or keep it strictly minimal?

**You said:**

how do I check database

**ChatGPT said:**

Good question 👍  
Since you’re using **MongoDB**, there are a few ways you can check what’s inside your todo\_app database.

**1. Use the MongoDB Shell (mongosh)**

If MongoDB is installed locally, open a terminal and run:

mongosh

Then switch to your database:

use todo\_app

Check collections:

show collections

View documents inside the todos collection:

db.todos.find().pretty()

This will show all your tasks with \_id, task, and completed fields.

**2. Use MongoDB Compass (GUI tool)**

* Download [MongoDB Compass](https://www.mongodb.com/products/compass)
* Connect to your local MongoDB (mongodb://127.0.0.1:27017)
* Open the todo\_app database → todos collection → browse documents in a nice GUI.

**3. If You’re Using MongoDB Atlas (Cloud)**

* Go to the Atlas dashboard
* Open your cluster → **Collections**
* Choose todo\_app → todos
* You’ll see all tasks directly in the web UI.

**4. Quick Check Inside Your App**

You can also temporarily add a debugging route to your server.js:

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

const todos = await Todo.find();

res.json(todos);

});

Then open <http://localhost:3000/debug> → you’ll see raw JSON output of all tasks.