The MERN Stack in Action: A Practical Full-Stack Demo

HARI BABU MUTCHAKALA

This document provides a practical demonstration of building a full-stack web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js). It outlines the key components, setup, and implementation steps involved in creating a basic application, showcasing the power and versatility of this popular technology stack. This demo will guide you through setting up a simple task management application, covering everything from the backend API to the frontend user interface.

Introduction to the MERN Stack

The MERN stack is a collection of JavaScript-based technologies used to develop full-stack web applications. Each component plays a crucial role:

- **MongoDB:** A NoSQL database that stores data in JSON-like documents. It's flexible and scalable, making it suitable for modern web applications.
- **Express.js:** A Node.js web application framework that provides a robust set of features for building web and mobile applications. It simplifies routing, middleware integration, and API development.
- **React.js:** A JavaScript library for building user interfaces. It uses a component-based architecture and a virtual DOM to efficiently update the UI.
- **Node.js:** A JavaScript runtime environment that allows you to run JavaScript on the server-side. It's event-driven and non-blocking, making it ideal for building scalable network applications.

Setting Up the Development Environment

Before diving into the application, ensure you have the following installed:

- **Node.js and npm (Node Package Manager):** Download and install the latest version of Node.js from the official website [https://nodejs.org/]. npm comes bundled with Node.js.
- **MongoDB:** Download and install MongoDB Community Edition from the official website [https://www.mongodb.com/]. Ensure the MongoDB server is running.
- **Text Editor/IDE:** Choose a code editor like VS Code, Sublime Text, or Atom.

Backend (Node.js and Express.js)

Project Setup

1. **Create a project directory:**

```bash
```

```
mkdir mern-task-manager
```

```
cd mern-task-manager
```

```
```

2. **Initialize a Node.js project:**

```bash
npm init -y
```

```
```

3. **Install Express.js and Mongoose:**

```bash
npm install express mongoose cors dotenv
```

```
```

```
*   `express`: For creating the web server and handling routes.
```

```
*   `mongoose`: For interacting with MongoDB.
```

```
*   `cors`: For handling Cross-Origin Resource Sharing (CORS) issues.
```

```
*   `dotenv`: For managing environment variables.
```

4. **Create** index.js**:**

```javascript
// index.js
```

```
const express = require('express');
```

```
const mongoose = require('mongoose');
```

```
const cors = require('cors');
```

```
require('dotenv').config();
```

```
const app = express();
```

```
const port = process.env.PORT || 5000;
```

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

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

```
// MongoDB Connection
```

```
mongoose.connect(process.env.MONGODB_URI ||
'mongodb://localhost:27017/task-manager', {
```

```
    useNewUrlParser: true,
```

```
    useUnifiedTopology: true,
```

```
});
```

```
const connection = mongoose.connection;
```

```
connection.once('open', () => {
```

```
    console.log('MongoDB database connection established successfully');
```

```
});
```

```
// Routes
```

```
const tasksRouter = require('./routes/tasks');
```

```
app.use('/tasks', tasksRouter);
```

```
app.listen(port, () => {
```

```
    console.log(`Server is running on port: ${port}`);
```

```
});
```

```
```

5. **Create** .env **file:**

```
```

```
MONGODB_URI=mongodb+srv://<username>:<password>@<cluster>.mongodb.net/task-manag
er?retryWrites=true&w=majority
```

```
```

```
Replace `<username>`, `<password>`, and `<cluster>` with your MongoDB Atlas
credentials or leave it as `mongodb://localhost:27017/task-manager` for local
MongoDB.
```

Defining the Task Model

1. **Create** models/task.model.js**:**

```
```javascript
```

```
// models/task.model.js
```

```
const mongoose = require('mongoose');
```

```
const taskSchema = new mongoose.Schema({
```

```
    description: { type: String, required: true },
```

```
    completed: { type: Boolean, default: false },
```
```

```
}, {
```

```
    timestamps: true,
```

```
});
```

```
const Task = mongoose.model('Task', taskSchema);
```

```
module.exports = Task;
```

```
```

Creating the API Routes

1. **Create** routes/tasks.js**:**

```
```javascript
```

```
// routes/tasks.js
```

```
const express = require('express');
```

```
const router = express.Router();
```

```
const Task = require('../models/task.model');
```

```
// Get all tasks
```

```
router.get('/', async (req, res) => {
```

```
    try {
```

```
        const tasks = await Task.find();
```

```
        res.json(tasks);
```

```
    } catch (err) {
```
```

```
        res.status(500).json({ message: err.message });
```

```
    }
```

```
});
```

```
// Get a specific task
```

```
router.get('/:id', async (req, res) => {
```

```
    try {
```

```
        const task = await Task.findById(req.params.id);
```

```
        if (!task) {
```

```
            return res.status(404).json({ message: 'Task not found' });
```

```
        }
```

```
        res.json(task);
```

```
    } catch (err) {
```

```
        return res.status(500).json({ message: err.message });
```

```
    }
```

```
});
```

```
// Create a new task
```

```
router.post('/', async (req, res) => {
```

```
    const task = new Task({
```

```
        description: req.body.description,
```

```
        completed: req.body.completed,
```

```
    });
```

```
    try {
```

```
        const newTask = await task.save();
```

```
        res.status(201).json(newTask);
```

```
    } catch (err) {
```

```
        res.status(400).json({ message: err.message });
```

```
    }
```

```
});
```

```
// Update a task
```

```
router.patch('/:id', async (req, res) => {
```

```
    try {
```

```
        const task = await Task.findById(req.params.id);
```

```
        if (!task) {
```

```
            return res.status(404).json({ message: 'Task not found' });
```

```
        }
```

```
        if (req.body.description != null) {
```

```
            task.description = req.body.description;
```

```
        }
```

```
        if (req.body.completed != null) {
```

```
            task.completed = req.body.completed;
```

```
        }
```

```
        const updatedTask = await task.save();
```

```
        res.json(updatedTask);
```

```
    } catch (err) {
```

```
        res.status(400).json({ message: err.message });
```

```
    }
```

```
});
```

```
// Delete a task
```

```
router.delete('/:id', async (req, res) => {
```

```
    try {
```

```
        const task = await Task.findByIdAndDelete(req.params.id);
```

```
        if (!task) {
```

```
            return res.status(404).json({ message: 'Task not found' });
```

```
        }
```

```
        res.json({ message: 'Task deleted' });
```

```
    } catch (err) {
```

```
        res.status(500).json({ message: err.message });
```

```
    }
```

```
});
```

```
module.exports = router;
```

```
```

Frontend (React.js)

Project Setup

1. **Create a new React app:**

```
```bash
```

```
npx create-react-app client
```

```
cd client
```

```
```
```

2. **Install Axios:**

```
```bash
```

```
npm install axios
```

```
```
```

```
*   `axios`: For making HTTP requests to the backend API.
```

Creating Components

1. **Create** src/components/TaskList.js**:**

```
```javascript
```
```

```
// src/components/TaskList.js

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

import axios from 'axios';

function TaskList() {

    const [tasks, setTasks] = useState([]);

    const [newTask, setNewTask] = useState('');

    useEffect(() => {

        fetchTasks();

    }, []);

    const fetchTasks = async () => {

        try {

            const response = await axios.get('http://localhost:5000/tasks');

            setTasks(response.data);

        } catch (error) {

            console.error('Error fetching tasks:', error);

        }

    };

    const addTask = async () => {
```

```
        try {
```

```
            await axios.post
```