The MERN Stack Unleashed: A Full-Stack Demo for Modern Developers

HARI BABU MUTCHAKALA

This document provides a comprehensive demonstration of the MERN stack (MongoDB, Express.js, React.js, Node.js) by building a simple task management application. It outlines the setup, implementation, and key considerations for each component of the stack, offering a practical guide for developers looking to build modern, full-stack web applications.

Introduction to the MERN Stack

The MERN stack is a popular JavaScript-based technology stack used for building full-stack web applications. It consists of the following technologies:

- **MongoDB:** A NoSQL database that stores data in flexible, JSON-like documents.
- **Express.js:** A Node.js web application framework that provides a robust set of features for building web and mobile applications.
- **React.js:** A JavaScript library for building user interfaces, known for its component-based architecture and virtual DOM.
- **Node.js:** A JavaScript runtime environment that allows you to run JavaScript on the server-side.

Project Setup

Before diving into the code, let's set up the project structure. We'll create two main directories: backend for the server-side code and frontend for the client-side code.

1. **Create Project Directory:**

```
```bash
```

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

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

```
```
```

2. **Initialize Backend:**

```
```bash
```

```
mkdir backend
```
```

```
cd backend
```

```
npm init -y
```

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

```
npm install --save-dev nodemon
```

```

```

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

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

```
*   `cors`:  For enabling Cross-Origin Resource Sharing.
```

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

```
*   `nodemon`:  For automatically restarting the server during development.
```

```

```

```
Add the following script to `package.json` in the `backend` directory:
```

```

```

```
```json
```

```
"scripts": {
```

```
  "start": "node server.js",
```

```
  "dev": "nodemon server.js"
```

```
}
```

```
```
```

3. **Initialize Frontend:**

```
```bash
```
```

```
cd ..
```

```
mkdir frontend
```

```
cd frontend
```

```
npx create-react-app .
```

```
npm install axios
```

```
```

```
*   `create-react-app`:  Sets up a new React project with a pre-configured build process.
```

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

Backend Implementation (Node.js & Express.js)

1. **Create** server.js **in the** backend **directory:**

```
```javascript
```

```
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());
```

```
const uri = process.env.ATLAS_URI;
```

```
mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }
```

```
);
```

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

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

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

```
})
```

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

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

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

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

```
});
```

```
```

2. **Create** .env **file in the** backend **directory:**

```
```

```
ATLAS_URI=YOUR_MONGODB_URI
```

```
```

```
Replace `YOUR_MONGODB_URI` with your MongoDB connection string.
```

3. **Create** models/task.model.js **in the** backend **directory:**

```
```javascript
```

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

```
const Schema = mongoose.Schema;
```

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

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

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

```
}, {
```

```
  timestamps: true,
```

```
});
```

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

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

```
```
```

4. **Create** routes/tasks.js **in the** backend **directory:**

```
```javascript
```

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

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

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

```
  Task.find()
```

```
    .then(tasks => res.json(tasks))
```

```
    .catch(err => res.status(400).json('Error: ' + err));
```

```
});
```

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

```
  const description = req.body.description;
```

```
  const newTask = new Task({description});
```

```
  newTask.save()
```

```
    .then(() => res.json('Task added!'))
```

```
    .catch(err => res.status(400).json('Error: ' + err));
```

```
});
```

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

```
  Task.findById(req.params.id)
```

```
    .then(task => res.json(task))
```

```
    .catch(err => res.status(400).json('Error: ' + err));
```

```
});
```

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

```
  Task.findByIdAndDelete(req.params.id)
```

```
    .then(() => res.json('Task deleted.'))
```

```
    .catch(err => res.status(400).json('Error: ' + err));
```

```
});
```

```
router.route('/update/:id').post((req, res) => {
```

```
  Task.findById(req.params.id)
```

```
    .then(task => {
```

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

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

```
      task.save()
```

```
        .then(() => res.json('Task updated!'))
```

```
        .catch(err => res.status(400).json('Error: ' + err));
```

```
    })
```

```
    .catch(err => res.status(400).json('Error: ' + err));
```

```
});
```

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

```
```

Frontend Implementation (React.js)

1. **Create** components/TaskList.js **in the** frontend/src **directory:**

```javascript
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function TaskList() {
  const [tasks, setTasks] = useState([]);

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

  const deleteTask = (id) => {
    axios.delete('http://localhost:5000/tasks/'+id)
      .then(response => { console.log(response.data)});

    setTasks(tasks.filter(el => el._id !== id));
```

```
  }
```

```
  const taskList = () => {
```

```
    return tasks.map(currenttask => {
```

```
      return <Task task={currenttask} deleteTask={deleteTask}
key={currenttask._id}/>;
```

```
    })
```

```
  }
```

```
  return (
```

```
    <div>
```

```
      <h3>Task List</h3>
```

```
      <table className="table">
```

```
        <thead className="thead-light">
```

```
          <tr>
```

```
            <th>Description</th>
```

```
            <th>Actions</th>
```

```
          </tr>
```

```
        </thead>
```

```
        <tbody>
```

```
          { taskList() }
```

```
        </tbody>
```

```
        </table>
```

```
      </div>
```

```
    )
```

```
}
```

```
const Task = props => (
```

```
  <tr>
```

```
    <td>{props.task.description}</td>
```

```
    <td>
```

```
      <a href="#" onClick={() => { props.deleteTask(props.task._id)
}}>delete</a>
```

```
    </td>
```

```
  </tr>
```

```
)
```

```
export default TaskList;
```

```
```
```

2. **Create** components/CreateTask.js **in the** frontend/src **directory:**

```
```javascript
```

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

```
import axios from 'axios';
```

```
function CreateTask() {
```

```
  const [description, setDescription] = useState('');
```

```
  const onChangeDescription = (e) => {
```

```
    setDescription(e.target.value);
```

```
  }
```

```
  const onSubmit = (e) => {
```

```
    e.preventDefault();
```

```
    const task = {
```

```
      description: description
```

```
    }
```

```
    axios.post('http://localhost:5000/tasks/add', task)
```

```
      .then(res => console.log(res.data));
```