task\_management/

│ ├── public/

│ ├── src/

│ │ ├── App.jsx

│ │ └── main.jsx

│ ├── index.html

│ ├── package.json

│ └── vite.config.js

├── server/

│ ├── index.js

│ ├── models.js

│ ├── middleware.js

│ └── package.json

└── README.md

models.js

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({

name: String,

email: { type: String, unique: true },

password: String,

createdAt: { type: Date, default: Date.now }

});

userSchema.pre('save', function (next) {

const user = this;

if (!user.isModified('password')) return next();

bcrypt.hash(user.password, 10)

.then(hash => {

user.password = hash;

next();

});

});

const taskSchema = new mongoose.Schema({

title: String,

description: String,

status: { type: String, enum: ['pending', 'in-progress', 'completed'], default: 'pending' },

dueDate: Date,

assignedTo: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }

});

const User = mongoose.model('User', userSchema);

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

module.exports = { User, Task };

middleware.js

const jwt = require('jsonwebtoken');

function authMiddleware(req, res, next) {

const token = req.header('Authorization');

if (!token) return res.status(401).json({ message: 'No token' });

try {

const decoded = jwt.verify(token, 'secretkey');

req.user = decoded;

next();

} catch (err) {

res.status(400).json({ message: 'Invalid token' });

}

}

module.exports = authMiddleware;

index.js

const express = require('express');

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const cors = require('cors');

const { User, Task } = require('./models');

const authMiddleware = require('./middleware');

const app = express();

app.use(cors());

app.use(express.json());

mongoose.connect('mongodb://localhost:27017/task\_management')

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Register

app.post('/api/auth/register', (req, res) => {

const { name, email, password } = req.body;

User.findOne({ email })

.then(existing => {

if (existing) return res.status(400).json({ message: 'User exists' });

const user = new User({ name, email, password });

user.save()

.then(() => res.json({ message: 'User registered' }))

.catch(err => res.status(500).json({ error: err }));

});

});

// Login

app.post('/api/auth/login', (req, res) => {

const { email, password } = req.body;

User.findOne({ email })

.then(user => {

if (!user) return res.status(400).json({ message: 'Invalid credentials' });

bcrypt.compare(password, user.password)

.then(match => {

if (!match) return res.status(400).json({ message: 'Invalid credentials' });

const token = jwt.sign({ id: user.\_id }, 'secretkey');

res.json({ token });

});

});

});

// Create Task

app.post('/api/tasks', authMiddleware, (req, res) => {

const task = new Task({ ...req.body, assignedTo: req.user.id });

task.save()

.then(saved => res.json(saved))

.catch(err => res.status(500).json({ error: err }));

});

// Get All Tasks

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

Task.find({ assignedTo: req.user.id })

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

});

// Get One Task

app.get('/api/tasks/:id', authMiddleware, (req, res) => {

Task.findOne({ \_id: req.params.id, assignedTo: req.user.id })

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

});

// Update Task

app.put('/api/tasks/:id', authMiddleware, (req, res) => {

Task.findOneAndUpdate(

{ \_id: req.params.id, assignedTo: req.user.id },

req.body,

{ new: true }

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

});

// Delete Task

app.delete('/api/tasks/:id', authMiddleware, (req, res) => {

Task.findOneAndDelete({ \_id: req.params.id, assignedTo: req.user.id })

.then(() => res.json({ message: 'Deleted' }));

});

app.listen(5000, () => console.log('Server running on http://localhost:5000'));

App.jsx

import { useEffect, useState } from "react";

//useState-allows us to track state in a function component

//useEffect-allows you to perform side effects in your components(fetching data, directly updating DOM and timers)

function App() {

const [token, setToken] = useState(localStorage.getItem("token") || "");

const [loggedInUser, setLoggedInUser] = useState(localStorage.getItem("user") || "");

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

const [form, setForm] = useState({

name: "",

email: "",

password: "",

title: "",

description: "",

status: "pending",

dueDate: "",

});

const [editTaskId, setEditTaskId] = useState(null);

const [filter, setFilter] = useState("all");

const handleChange = (e) => {

setForm({ ...form, [e.target.name]: e.target.value });

};

const register = () => {

fetch("http://localhost:5000/api/auth/register", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

name: form.name,

email: form.email,

password: form.password,

}),

})

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

.then((data) => alert(data.message));

};

const login = () => {

fetch("http://localhost:5000/api/auth/login", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

email: form.email,

password: form.password,

}),

})

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

.then((data) => {

if (data.token) {

setToken(data.token);

localStorage.setItem("token", data.token);

setLoggedInUser(form.email);

localStorage.setItem("user", form.email);

fetchTasks(data.token);

} else {

alert("Login failed");

}

});

};

const logout = () => {

setToken("");

localStorage.removeItem("token");

setLoggedInUser("");

localStorage.removeItem("user")

setTasks([]);

};

const fetchTasks = (jwt) => {

fetch("http://localhost:5000/api/tasks", {

headers: { Authorization: jwt },

})

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

.then((data) => {

setTasks(Array.isArray(data) ? data : []);

});

};

const createTask = () => {

fetch("http://localhost:5000/api/tasks", {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({

title: form.title,

description: form.description,

status: form.status,

dueDate: form.dueDate,

}),

})

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

.then(() => fetchTasks(token));

};

const updateTask = () => {

fetch(http://localhost:5000/api/tasks/${editTaskId}, {

method: "PUT",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({

title: form.title,

description: form.description,

status: form.status,

dueDate: form.dueDate,

}),

})

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

.then(() => {

setEditTaskId(null);

fetchTasks(token);

});

};

const deleteTask = (id) => {

fetch(http://localhost:5000/api/tasks/${id}, {

method: "DELETE",

headers: { Authorization: token },

})

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

.then(() => fetchTasks(token));

};

const startEdit = (task) => {

setEditTaskId(task.\_id);

setForm({

...form,

title: task.title,

description: task.description,

status: task.status,

dueDate: task.dueDate ? task.dueDate.split("T")[0] : "",

});

};

useEffect(() => {

if (token) {

fetchTasks(token);

}

}, [token]);

return (

<div>

<h2>Register or Login</h2>

<input placeholder="Name" name="name" onChange={handleChange} />

<input placeholder="Email" name="email" onChange={handleChange} />

<input placeholder="Password" name="password" type="password" onChange={handleChange}/>

<button onClick={register}>Register</button>

<button onClick={login}>Login</button>

{token && <button onClick={logout}>Logout</button>}

{token && <p>Logged in as: {loggedInUser}</p>}

<h2>{editTaskId ? "Edit Task" : "Create Task"}</h2>

<input placeholder="Title" name="title" value={form.title} onChange={handleChange} />

<input placeholder="Description" name="description" value={form.description} onChange={handleChange}/>

<select name="status" value={form.status} onChange={handleChange}>

<option value="pending">Pending</option>

<option value="in-progress">In-Progress</option>

<option value="completed">Completed</option>

</select>

<input type="date" name="dueDate" value={form.dueDate} onChange={handleChange} />

{editTaskId ? (<button onClick={updateTask}>Update Task</button>) :

(<button onClick={createTask}>Create Task</button>)}

<h2>Your Tasks</h2>

<div>

<label>Filter:</label>

<select value={filter} onChange={(e) => setFilter(e.target.value)}>

<option value="all">All</option>

<option value="pending">Pending</option>

<option value="in-progress">In-Progress</option>

<option value="completed">Completed</option>

</select>

</div>

{tasks.length === 0 ? (<p>No tasks available</p>) : (

tasks

.filter((task) => filter === "all" || task.status === filter)

.map((task) => (

<div key={task.\_id}>

<p>Title: {task.title}</p>

<p>Status: {task.status}</p>

<p>Due: {task.dueDate ? task.dueDate.slice(0, 10) : "N/A"}</p>

<button onClick={() => startEdit(task)}>Edit</button>

<button onClick={() => deleteTask(task.\_id)}>Delete</button>

<hr />

</div>

))

)}

</div>

);

}

export default App;

Main.jsx

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App.jsx';

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>

);

this is the task management whole proj

ill give another project can you give me all files

use same techiniques in terms of code and files so i wont be confused