Expense Tracker

// server.js

const express = require('express');

const mongoose = require('mongoose');

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

const cors = require('cors');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

const app = express();

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

const secretKey = 'your\_secret\_key';

// Middleware

app.use(bodyParser.json());

app.use(cors());

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/expense-tracker', { useNewUrlParser: true, useUnifiedTopology: true });

// Define Schemas and Models

const UserSchema = new mongoose.Schema({

username: String,

password: String,

});

const ExpenseSchema = new mongoose.Schema({

userId: mongoose.Schema.Types.ObjectId,

category: String,

amount: Number,

comments: String,

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

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

});

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

const Expense = mongoose.model('Expense', ExpenseSchema);

// User Routes

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

const { username, password } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

const newUser = new User({ username, password: hashedPassword });

await newUser.save();

res.send('User created');

});

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

const { username, password } = req.body;

const user = await User.findOne({ username });

if (user && await bcrypt.compare(password, user.password)) {

const token = jwt.sign({ userId: user.\_id }, secretKey);

res.json({ token });

} else {

res.status(401).send('Invalid credentials');

}

});

// Expense Routes

app.post('/add-expense', async (req, res) => {

const { token } = req.headers;

const { userId } = jwt.verify(token, secretKey);

const { category, amount, comments } = req.body;

const newExpense = new Expense({ userId, category, amount, comments });

await newExpense.save();

res.send('Expense added');

});

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

const { token } = req.headers;

const { userId } = jwt.verify(token, secretKey);

const expenses = await Expense.find({ userId }).sort({ createdAt: -1 });

res.json(expenses);

});

app.put('/edit-expense/:id', async (req, res) => {

const { id } = req.params;

const { token } = req.headers;

const { userId } = jwt.verify(token, secretKey);

const { category, amount, comments } = req.body;

await Expense.findOneAndUpdate({ \_id: id, userId }, { category, amount, comments, updatedAt: Date.now() });

res.send('Expense updated');

});

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

const { id } = req.params;

const { token } = req.headers;

const { userId } = jwt.verify(token, secretKey);

await Expense.findOneAndDelete({ \_id: id, userId });

res.send('Expense deleted');

});

app.listen(PORT, () => {

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

});

// SignUp.js

import React, { useState } from 'react';

import axios from 'axios';

const SignUp = () => {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

await axios.post('http://localhost:5000/signup', { username, password });

alert('User created');

};

return (

<form onSubmit={handleSubmit}>

<input type="text" value={username} onChange={(e) => setUsername(e.target.value)} placeholder="Username" />

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} placeholder="Password" />

<button type="submit">Sign Up</button>

</form>

);

};

export default SignUp;

// Login.js

import React, { useState } from 'react';

import axios from 'axios';

const Login = () => {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const res = await axios.post('http://localhost:5000/login', { username, password });

localStorage.setItem('token', res.data.token);

alert('Login successful');

};

return (

<form onSubmit={handleSubmit}>

<input type="text" value={username} onChange={(e) => setUsername(e.target.value)} placeholder="Username" />

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} placeholder="Password" />

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

</form>

);

};

export default Login;

// AddExpense.js

import React, { useState } from 'react';

import axios from 'axios';

const AddExpense = () => {

const [category, setCategory] = useState('');

const [amount, setAmount] = useState('');

const [comments, setComments] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const token = localStorage.getItem('token');

await axios.post('http://localhost:5000/add-expense', { category, amount, comments }, {

headers: { token }

});

alert('Expense added');

};

return (

<form onSubmit={handleSubmit}>

<input type="text" value={category} onChange={(e) => setCategory(e.target.value)} placeholder="Category" />

<input type="number" value={amount} onChange={(e) => setAmount(e.target.value)} placeholder="Amount" />

<input type="text" value={comments} onChange={(e) => setComments(e.target.value)} placeholder="Comments" />

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

</form>

);

};

export default AddExpense;

// ViewExpenses.js

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

import axios from 'axios';

const ViewExpenses = () => {

const [expenses, setExpenses] = useState([]);

useEffect(() => {

const fetchExpenses = async () => {

const token = localStorage.getItem('token');

const res = await axios.get('http://localhost:5000/expenses', {

headers: { token }

});

setExpenses(res.data);

};

fetchExpenses();

}, []);

return (

<table>

<thead>

<tr>

<th>Category</th>

<th>Amount</th>

<th>Created At</th>

<th>Updated At</th>

<th>Comments</th>

</tr>

</thead>

<tbody>

{expenses.map((expense) => (

<tr key={expense.\_id}>

<td>{expense.category}</td>

<td>{expense.amount}</td>

<td>{new Date(expense.createdAt).toLocaleString()}</td>

<td>{new Date(expense.updatedAt).toLocaleString()}</td>

<td>{expense.comments}</td>

</tr>

))}

</tbody>

</table>

);

};

export default ViewExpenses;