**Db.js**

import express from 'express';

import mongoose from 'mongoose';

const app = express();

mongoose.connect('mongodb://localhost:27017/users', {

useNewUrlParser: true,

useUnifiedTopology: true

})

.then(() => console.log('Connected to MongoDB'))

.catch(err => console.error('Failed to connect to MongoDB', err));

const userSchema = new mongoose.Schema({

title: String,

description: String,

price: Number,

category: String,

sold: Number,

image: String

});

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

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

User.find({})

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

.catch(err => {

console.error(err);

res.status(500).json({ error: 'Internal server error' });

});

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

**charts.js**

import Transaction from '../models/Transaction.js';

const getBarChart = async (req, res) => {

const { month } = req.query;

const priceRanges = [100, 200, 300, 400, 500, 600, 700, 800, 900];

const priceRangesLabels = ['0-100', '101-200', '201-300', '301-400', '401-500', '501-600', '601-700', '701-800', '801-900', '901-above'];

const chartData = await Promise.all(priceRangesLabels.map(async (label, index) => {

let lowerBound = priceRanges[index - 1] || 0;

let upperBound = priceRanges[index] || Infinity;

const count = await Transaction.countDocuments({

dateOfSale: { $gte: new Date(2023, month - 1, 1), $lt: new Date(2023, month, 1) },

price: { $gte: lowerBound, $lt: upperBound }

});

return { label, count };

}));

res.json(chartData);

};

const getPieChart = async (req, res) => {

const { month } = req.query;

const chartData = await Transaction.aggregate([

{

$match: {

dateOfSale: { $gte: new Date(2023, month - 1, 1), $lt: new Date(2023, month, 1) }

}

},

{

$group: {

\_id: '$category',

count: { $sum: 1 }

}

}

]);

res.json(chartData);

};

export {getBarChart,getPieChart};

**transaction.controller**

import Transaction from '../models/Transaction.js';

const listTransactions = async (req, res) => {

const { page = 1, perPage = 10, search = '', month } = req.query;

const searchRegex = new RegExp(search, 'i');

if (!month || isNaN(month) || month < 1 || month > 12) {

return res.status(400).json({ error: 'Invalid month parameter' });

}

const query = {

$and: [

{ dateOfSale: { $gte: new Date(2023, month - 1, 1), $lt: new Date(2023, month, 1) } },

{

$or: [

{ title: searchRegex },

{ description: searchRegex },

{ price: searchRegex }

]

}

]

};

const transactions = await Transaction.find(query)

.skip((page - 1) \* perPage)

.limit(perPage);

const sanitizedTransactions = transactions.map(transaction => ({

...transaction.toObject(),

dateOfSale: transaction.dateOfSale.toISOString()

}));

res.json(sanitizedTransactions);

};

export default listTransactions**;**

**init.js**

import express from "express";

import fetch from 'node-fetch';

import Transaction from "../models/Transaction.js";

const router = express.Router();

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

try {

// Fetch data from the third-party API

const response = await fetch('https://s3.amazonaws.com/roxiler.com/product\_transaction.json');

if (!response.ok) {

throw new Error(`Error fetching data: ${response.statusText}`);

}

const data = await response.json();

// Clear the existing data in the database

await Transaction.deleteMany({});

// Insert the new data

await Transaction.insertMany(data);

res.status(200).send('Database initialized with seed data');

} catch (error) {

// Log the error for debugging purposes

console.error('Error initializing database:', error);

res.status(500).send(`Error initializing database: ${error.message}`);

}

});

export default router;

**frontend**

**transtable.js**

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

const TransTable = ({ selectedMonth }) => {

const [transactions, setTransactions] = useState([]);

const [search, setSearch] = useState('');

const [page, setPage] = useState(1);

useEffect(() => {

fetchTransactions();

}, [selectedMonth, search, page]);

const fetchTransactions = async () => {

try {

const url = new URL('/api/transactions', window.location.origin);

url.searchParams.append('month', selectedMonth);

url.searchParams.append('search', search);

url.searchParams.append('page', page);

// Await the fetch call

const response = await fetch(url);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

// Await the JSON conversion

const data = await response.json();

const jsonArr = JSON.parse(data);

setTransactions(jsonArr);

} catch (error) {

console.error(`Error fetching transactions: ${error}`);

}

};

return (

<div>

<input

type="text"

value={search}

onChange={(e) => setSearch(e.target.value)}

placeholder="Search transactions"

/>

<table>

<thead>

<tr>

<th>Title</th>

<th>Description</th>

<th>Price</th>

<th>Date</th>

</tr>

</thead>

<tbody>

{transactions.map((transaction) => (

<tr key={transaction.\_id}>

<td>{transaction.title}</td>

<td>{transaction.description}</td>

<td>{transaction.price}</td>

<td>{new Date(transaction.dateOfSale).toLocaleDateString()}</td>

</tr>

))}

</tbody>

</table>

<button onClick={() => setPage((prev) => Math.max(prev - 1, 1))}>Previous</button>

<button onClick={() => setPage((prev) => prev + 1)}>Next</button>

</div>

);

};

export default TransTable;

**page.js**

import TransTable from './components/TransTable';

import { useState } from 'react';

export default function Home() {

const [selectedMonth, setSelectedMonth] = useState(new Date().getMonth() + 1); // Default to current month

return (

<main>

<h1>Transactions</h1>

<TransTable selectedMonth={selectedMonth} />

</main>

);

