**FRONTEND TASK**

**src/components/TransactionsTable.js:**

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

import axios from 'axios';

const TransactionsTable = ({ month, onSearch }) => {

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

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

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

const [totalCount, setTotalCount] = useState(0);

useEffect(() => {

const fetchTransactions = async () => {

const response = await axios.get('http://localhost:5000/api/transactions', {

params: { page, perPage: 10, search, month }

});

setTransactions(response.data.transactions);

setTotalCount(response.data.totalCount);

};

fetchTransactions();

}, [page, search, month]);

const handleSearchChange = (e) => {

setSearch(e.target.value);

setPage(1); // Reset to first page on search

};

const handleNextPage = () => setPage((prevPage) => prevPage + 1);

const handlePrevPage = () => setPage((prevPage) => prevPage - 1);

return (

<div>

<input

type="text"

value={search}

onChange={handleSearchChange}

placeholder="Search by title, description or price"

/>

<table>

<thead>

<tr>

<th>Title</th>

<th>Description</th>

<th>Price</th>

<th>Date of Sale</th>

<th>Category</th>

<th>Sold</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>

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

<td>{transaction.sold ? 'Yes' : 'No'}</td>

</tr>

))}

</tbody>

</table>

<div>

<button onClick={handlePrevPage} disabled={page === 1}>Previous</button>

<button onClick={handleNextPage} disabled={page \* 10 >= totalCount}>Next</button>

</div>

</div>

);

};

export default TransactionsTable;

**src/components/Statistics.js:**

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

import axios from 'axios';

const Statistics = ({ month }) => {

const [stats, setStats] = useState({ totalSalesAmount: 0, totalSoldItems: 0, totalNotSoldItems: 0 });

useEffect(() => {

const fetchStatistics = async () => {

const response = await axios.get('http://localhost:5000/api/stats/statistics', {

params: { month }

});

setStats(response.data);

};

fetchStatistics();

}, [month]);

return (

<div>

<h2>Statistics</h2>

<p>Total Sales Amount: ${stats.totalSalesAmount}</p>

<p>Total Sold Items: {stats.totalSoldItems}</p>

<p>Total Not Sold Items: {stats.totalNotSoldItems}</p>

</div>

);

};

export default Statistics;

**src/components/BarChart.js:**

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

import axios from 'axios';

import { BarChart, Bar, XAxis, YAxis, Tooltip, CartesianGrid, Legend } from 'recharts';

const BarChartComponent = ({ month }) => {

const [data, setData] = useState([]);

useEffect(() => {

const fetchBarChartData = async () => {

const response = await axios.get('http://localhost:5000/api/stats/bar-chart', {

params: { month }

});

setData(response.data);

};

fetchBarChartData();

}, [month]);

return (

<BarChart width={600} height={300} data={data}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="range" />

<YAxis />

<Tooltip />

<Legend />

<Bar dataKey="count" fill="#8884d8" />

</BarChart>

);

};

export default BarChartComponent;

**src/components/PieChart.js:**

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

import axios from 'axios';

import { PieChart, Pie, Tooltip, Cell } from 'recharts';

const PieChartComponent = ({ month }) => {

const [data, setData] = useState([]);

const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042', '#FF69B4'];

useEffect(() => {

const fetchPieChartData = async () => {

const response = await axios.get('http://localhost:5000/api/stats/pie-chart', {

params: { month }

});

setData(response.data);

};

fetchPieChartData();

}, [month]);

return (

<PieChart width={400} height={400}>

<Tooltip />

<Pie data={data} dataKey="count" nameKey="\_id" outerRadius={150} fill="#8884d8" label>

{data.map((entry, index) => (

<Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />

))}

</Pie>

</PieChart>

);

};

export default PieChartComponent;

**src/App.js:**

import React, { useState } from 'react';

import TransactionsTable from './components/TransactionsTable';

import Statistics from './components/Statistics';

import BarChart from './components/BarChart';

import PieChart from './components/PieChart';

const App = () => {

const [month, setMonth] = useState('03'); // Default to March

const handleMonthChange = (e) => {

setMonth(e.target.value);

};

return (

<div>

<h1>Sales Analyzer</h1>

<select value={month} onChange={handleMonthChange}>

{['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12'].map((m) => (

<option key={m} value={m}>

{new Date(`2024-${m}-01`).toLocaleString('default', { month: 'long' })}

</option>

))}

</select>

<TransactionsTable month={month} />

<Statistics month={month} />

<BarChart month={month} />

<PieChart month={month} />

</div>

);

};

export default App;

**src/index.js:**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import './index.css';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**vite.config.js:**

import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

export default defineConfig({

plugins: [react()],

});

**BACKEND TASK**

**models/transaction.js:**

const mongoose = require('mongoose');

const transactionSchema = new mongoose.Schema({

title: String,

description: String,

price: Number,

dateOfSale: Date,

category: String,

sold: Boolean,

});

module.exports = mongoose.model('Transaction', transactionSchema);

**database.js:**

const mongoose = require('mongoose');

const connectDB = async () => {

try {

await mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log('Database connected');

} catch (err) {

console.error(err.message);

process.exit(1);

}

};

module.exports = connectDB;

**.env:**

const express = require('express');

const cors = require('cors');

const axios = require('axios');

const connectDB = require('./database');

const Transaction = require('./models/transaction');

const transactionsRoutes = require('./routes/transactions');

const statsRoutes = require('./routes/stats');

require('dotenv').config();

connectDB();

const app = express();

app.use(cors());

app.use(express.json());

app.use('/api/transactions', transactionsRoutes);

app.use('/api/stats', statsRoutes);

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

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

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

try {

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

const data = response.data;

await Transaction.deleteMany({});

await Transaction.insertMany(data);

res.status(200).json({ message: 'Database initialized successfully' });

} catch (error) {

res.status(500).json({ message: 'Error initializing database', error });

}

});

**server.js:**

const express = require('express');

const cors = require('cors');

const axios = require('axios');

const connectDB = require('./database');

const Transaction = require('./models/transaction');

const transactionsRoutes = require('./routes/transactions');

const statsRoutes = require('./routes/stats');

require('dotenv').config();

connectDB();

const app = express();

app.use(cors());

app.use(express.json());

app.use('/api/transactions', transactionsRoutes);

app.use('/api/stats', statsRoutes);

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

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

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

try {

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

const data = response.data;

await Transaction.deleteMany({});

await Transaction.insertMany(data);

res.status(200).json({ message: 'Database initialized successfully' });

} catch (error) {

res.status(500).json({ message: 'Error initializing database', error });

}

});

**routes/transactions.js:**

const express = require('express');

const router = express.Router();

const Transaction = require('../models/transaction');

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

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

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

const startDate = new Date(`2024-${month}-01`);

const endDate = new Date(`2024-${parseInt(month) + 1}-01`);

try {

const transactions = await Transaction.find({

dateOfSale: { $gte: startDate, $lt: endDate },

$or: [

{ title: regex },

{ description: regex },

{ price: regex }

]

}).skip((page - 1) \* perPage).limit(Number(perPage));

const totalCount = await Transaction.countDocuments({

dateOfSale: { $gte: startDate, $lt: endDate },

$or: [

{ title: regex },

{ description: regex },

{ price: regex }

]

});

res.json({

transactions,

totalCount

});

} catch (error) {

res.status(500).json({ message: 'Error fetching transactions', error });

}

});

module.exports = router;

**routes/stats.js:**

const express = require('express');

const router = express.Router();

const Transaction = require('../models/transaction');

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

const { month } = req.query;

const startDate = new Date(`2024-${month}-01`);

const endDate = new Date(`2024-${parseInt(month) + 1}-01`);

try {

const totalSales = await Transaction.aggregate([

{ $match: { dateOfSale: { $gte: startDate, $lt: endDate }, sold: true } },

{ $group: { \_id: null, totalAmount: { $sum: '$price' }, totalSoldItems: { $sum: 1 } } }

]);

const totalNotSoldItems = await Transaction.countDocuments({

dateOfSale: { $gte: startDate, $lt: endDate },

sold: false

});

res.json({

totalSalesAmount: totalSales[0]?.totalAmount || 0,

totalSoldItems: totalSales[0]?.totalSoldItems || 0,

totalNotSoldItems

});

} catch (error) {

res.status(500).json({ message: 'Error fetching statistics', error });

}

});

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

const { month } = req.query;

const startDate = new Date(`2024-${month}-01`);

const endDate = new Date(`2024-${parseInt(month) + 1}-01`);

try {

const priceRanges = [

{ $match: { dateOfSale: { $gte: startDate, $lt: endDate }, price: { $gte: 0, $lte: 100 } } },

{ $group: { \_id: '0-100', count: { $sum: 1 } } },

];

const barChartData = await Transaction.aggregate(priceRanges);

res.json(barChartData);

} catch (error) {

res.status(500).json({ message: 'Error fetching bar chart data', error });

}

});

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

const { month } = req.query;

const startDate = new Date(`2024-${month}-01`);

const endDate = new Date(`2024-${parseInt(month) + 1}-01`);

try {

const categories = await Transaction.aggregate([

{ $match: { dateOfSale: { $gte: startDate, $lt: endDate } } },

{ $group: { \_id: '$category', count: { $sum: 1 } } }

]);

res.json(categories);

} catch (error) {

res.status(500).json({ message: 'Error fetching pie chart data', error });

}

});

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

const { month } = req.query;

try {

const [statistics, barChart, pieChart] = await Promise.all([

axios.get(`http://localhost:5000/api/stats/statistics?month=${month}`),

axios.get(`http://localhost:5000/api/stats/bar-chart?month=${month}`),

axios.get(`http://localhost:5000/api/stats/pie-chart?month=${month}`)

]);

res.json({

statistics: statistics.data,

barChart: barChart.data,

pieChart: pieChart.data

});

} catch (error) {

res.status(500).json({ message: 'Error fetching combined data', error });

}

});

module.exports = router;

**OUTPUT :**

**Start the server**

node server.js

**Start the development server:**

npm run dev







