# 1-swagger\_output.json

{

"openapi": "3.0.0",

"info": {

"version": "1.0.0",

"title": " MERN CODING CHALLENGE",

"description": " MERN CODING CHALLENGE"

},

"servers": [

{

"url": "MERN CODING CHALLENGE.onrender.com",

"description": "Production server"

},

{

"url": "http://localhost:8000",

"description": "Local server"

}

],

"paths": {

"/api/v1/product/initialize-seed-data": {

"get": {

"tags": [

"Products "

],

"description": "",

"responses": {

"200": {

"description": "OK"

}

}

}

},

"/api/v1/product/": {

"get": {

"tags": [

"Products "

],

"description": "",

"responses": {

"200": {

"description": "OK"

}

} }

},

"/api/v1/product/search": {

"get": {

"tags": [

"Products "

],

"description": "",

"parameters": [

{

"name": "page",

"in": "query",

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "OK"

}

} }

},

"/api/v1/analytics/statistics": {

"get": {

"tags": [

"statistics"

],

"description": "",

"parameters": [

{

"name": "month",

"in": "query",

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "OK"

},

"400": {

"description": "Bad Request"

}

} }

},

"/api/v1/analytics/bar-chart": {

"get": {

"tags": [

"statistics"

],

"description": "",

"parameters": [

{

"name": "month",

"in": "query",

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "OK"

}

} }

},

"/api/v1/analytics/pie-chart": {

"get": {

"tags": [

"statistics"

],

"description": "",

"parameters": [

{

"name": "month",

"in": "query",

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "OK"

}

} }

},

"/api/v1/analytics/combined-chart": {

"get": {

"tags": [

"statistics"

],

"description": "",

"parameters": [

{

"name": "month",

"in": "query",

"schema": {

"type": "string"

}

}

],

"responses": {

"200": {

"description": "OK"

}

## 2-db.js

## import mongoose from "mongoose";

## export const connectDB = async () => {

## // console.log("Attempting to connect to MongoDB...");

## try {

## const conn = await mongoose.connect(process.env.MONGO\_URL || "mongodb://localhost:27017/mydb",{

## useNewUrlParser: true,

## useUnifiedTopology: true,

## });

## console.log(`Mongo Db connected: ${conn.connection.host}`);

## }

## catch (e) {

## console.log("cannot connect to database");

## process.exit(1);  // One means failure

## }

## };

## 3-Product.combine.js

## import {getproduct, monthlyStatistics, getBarChart, getPieChart } from '../controllers/product.controller';

## export const combineddata = async (req,res) => {

## const {month} = req.query;

## if(!month){

## return res.status(400).json({message:"month require"});

## }

## try{

## const [stat, barchart, piechart] = await Promise.all([

## monthlyStatistics(req, res, true),

## getBarChart(req,res, true),

## getPieChart(req,res, true)

## ]);

## const combinedData = {

## stat : stat.data,

## barchart: barchart.data,

## piechart: piechart.data,

## };

## res.json(combinedData)

## }catch(e){

## res.status(500).json({message:"error fetching data",e})

## }

## };

## 4-Product.controller.js

# import Products from "../models/product.model.js"

# import axios from "axios"

# // month mapping

# const monthMap = {

# January: 1,

# February: 2,

# March: 3,

# April: 4,

# May: 5,

# June: 6,

# July: 7,

# August: 8,

# September: 9,

# October: 10,

# November: 11,

# December: 12

# };

# // transation

# export const getproduct = async (req, res) => {

# try {

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

# await Products.insertMany(responce.data)

# res.status(200).json({ message: "success" })

# } catch (e) {

# res.status(500).json({ message: "error", e })

# }

# const { page = 1, perpage = 6, search = '' } = req.query;

# const query = search

# ? {

# $or: [

# { title: new RegExp(search, 'i') },

# { description: new RegExp(search, 'i') },

# { price: search },

# ]

# }

# : {};

# try {

# const total = await Products.countDocuments(query)

# const product = await Products.find(query)

# .skip((page - 1) \* per\_page)

# .limit(Number(perpage));

# res.json({

# total: totalDocuments,

# page: Number(page),

# per\_page: Number(per\_page),

# data: transactions,

# });

# } catch (e) {

# res.status(500).json({ message: 'error', e })

# }

# };

# // statstics

# export const monthlyStatistics = async (req, res) => {

# const { month } = req.query

# if (!month) {

# return res.status(400).json({ message: "missing month" });

# }

# const monthInt = monthMap[month]

# if (!monthInt) {

# return res.status(400).json({ message: 'invalid month' })

# }

# try {

# const totalSales = await Prodcts.aggregate([

# {

# $match: {

# sold: true,

# $expr: { $eq: [{ $month: "$dateOfSale" }, monthInt] }

# },

# },

# {

# $group: {

# \_id: null,

# totalAmount: { $sum: "$price" },

# },

# },

# ]);

# const totalItemSold = await Products.countDocuments({

# sold: true,

# $expr: { $eq: [{ $month: "$dateOfSale" }, monthInt] },

# });

# const totalNotSoldItems = await Products.countDocuments({

# sold: false,

# $expr: { $eq: [{ $month: "$dateOfSale" }, monthInt] },

# });

# res.json({

# totalsaleamm: totalSales[0]?.totalAmount || 0,

# totalItemSold,

# totalNotSoldItems

# });

# } catch (e) {

# res.status(500).json({ message: "error in reciving monthly stats", error })

# }

# };

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

# const {month} = req.query

# if (!month){

# return res.status(400).json({message: "missing month"})

# }

# const monthInt = monthMap[month]

# if (!monthInt) {

# return res.status(400).json({ message: 'Invalid month' });

# }

# try{

# const priceRange = await Products.aggregate([

# {

# $match:{

# $expr:{$eq:[{$month:"$dateOfSale"}, monthInt]}

# },

# },

# {

# $bucket:{

# groupby:"$price",

# boundaries:[0,101,201,301,401,501,601,701,801,901,Infinity],

# default: "901-above",

# output:{

# count:{$sum:1}

# }

# }

# }

# ]);

# const responce = priceRange.map(bucket =>({

# range: bucket.\_id === "901-above" ? "901-above":'${bucket.\_id - 100}-${bucket.id-1}',

# count: bucket.count

# }));

# res.json(responce);

# }catch(e){

# req.status(500).json({message:"error retriving barchart",e})

# }

# };

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

# const {month} = req.query;

# if (!month) {

# return res.status(400).json({ message: 'Month is required.' });

# }

# const monthInt = monthMap[month];

# if (!monthInt) {

# return res.status(400).json({ message: 'Invalid month provided.' });

# }

# try{

# const catagory = await Products.aggregate([

# {

# $match:{

# $expr:{$eq: [{$month:"$dateOfSale"},monthInt]}

# },

# },

# {

# $group:{

# \_id:"$category",

# count: { $sum:1}

# }

# }

# ]);

# const responce = catagory.map(catagory =>({

# catagory:catagory.\_id,

# count: catagory.count

# }));

# res.json(responce);

# } catch(e){

# res.status(500).json({message:'error retiving piechart'})

# }

# };

# 5-Model.js

## import mongoose from "mongoose";

## const productSchema = new mongoose.Schema({

## title: {

## type: String,

## required: true,

## },

## description: {

## type: String,

## required: true,

## },

## price: {

## type: Number,

## required: true,

## },

## category: {

## type: String,

## required: true,

## },

## dateOfSale: {

## type: Date,

## required: true,

## },

## sold: {

## type: Boolean,

## required: true,

## },

## });

## const Product = mongoose.model('Product',productSchema);

## export default Product;

## 6-routes.js

import { Request, Response, NextFunction } from 'express';

## import express from 'express';

## import { getproduct, monthlyStatistics, getBarChart, getPieChart } from '../controllers/product.controller.js';

## const app = express.Router();

## app.get('/', getproduct);

## app.get('/statistics', monthlyStatistics);

## app.get('/barchart',getBarChart);

## app.get('/piechart', getPieChart);

## export default app;

## 7-Server.js

# // import express from 'express';

# import dotenv from 'dotenv';

# import cors from 'cors';

# import { connectDB } from './config/db.js';

# import productRoutes from './routes/product.routes.js'

# dotenv.config()

# const app = express();

# app.use(cors());

# app.use(express.json());

# app.use('/api/products', productRoutes)

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

# connectDB().then(() => {

# app.listen(PORT, () => {

# console.log(`Server started at http://localhost:${PORT}`);

# });

# }).catch(error => {

# console.error('Failed to connect to database:', error);

# process.exit(1); // Exit process with failure

# });

# 8-BarChart.js

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

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

import axios from 'axios';

import './index.css'

const BarChartComponent = (props) => {

const [barChartData, setBarChartData] = useState([]);

const { selectedMonth } = props;

useEffect(() => {

const getBarChartData = async () => {

try {

const response = await axios.get(`https://roxiler-backend-vijaykumars-projects.vercel.app/bar-chart?month=${selectedMonth}`);

setBarChartData(response.data);

} catch (error) {

console.error('Error fetching bar chart data:', error);

}

};

getBarChartData();

}, [selectedMonth]);

const DataFormatter = (number) => {

if (number > 1000) {

return `${(number / 1000).toString()}k`;

}

return number.toString();

};

return (

<div className='barchart-container'>

<h2>Bar Chart Status - {selectedMonth}</h2>

<ResponsiveContainer width="100%" height={300}>

<BarChart data={barChartData} margin={{ top: 5 }}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis

dataKey="priceRange"

tick={{ stroke: 'gray', strokeWidth: 1 }}

/>

<YAxis

tickFormatter={DataFormatter}

tick={{ stroke: 'gray', strokeWidth: 1 }}

/>

<Legend wrapperStyle={{ padding: 30 }} />

<Bar dataKey="itemCount" name="Number of Items" fill="#37807e" barSize="20%" />

</BarChart>

</ResponsiveContainer>

</div>

);

};

export default BarChartComponent;

# 9-BarChart.css

# #barChartContainer {

# display: flex;

# justify-content: center;

# align-items: center;

# height: 100vh;

# }

# 

# #barChartCanvas {

# width: 80%;

# max-width: 800px;

# margin: 20px;

# border: 1px solid #ddd;

# border-radius: 8px;

# box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

# background-color: #fff;

# }

# .barchart-container{

# width: 50%;

# border: 1px solid black;

# border-radius: 5px;

# padding: 10px;

# }

# 9- transactions.css

table {

border-collapse: collapse;

width: 100%;

border-radius: 20px;

cursor: pointer;

height: 60%;

overflow-y: auto;

}

th, td {

border: 0.5px solid black;

padding: 8px;

text-align: left;

/\* border-bottom: 2px solid black; \*/

/\* border-radius: 25px; \*/

}

th {

background-color: #f0b762;

}

.main-container{

background-color: #d8eaf0;

/\* height: 100vh; \*/

padding: 20px;

display: flex;

flex-direction: column;

justify-content: flex-start;

align-items: center;

}

.first-container{

height: 15%;

width: 15%;

border-radius: 60%;

background-color: white;

display: flex;

flex-direction: row;

justify-content: center;

align-items: center;

padding: 10px;

margin-bottom: 10px;

}

.second-container{

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

width: 100%;

margin-bottom: 20px;

}

.input-element{

outline: none;

background-color: #f0b762;

color: black;

border-radius: 15px;

height: 30px;

width: 45%;

padding-left: 10px;

padding-right: 10px;

font-size: 14px;

}

.dropdown-list{

outline: none;

background-color: #f0b762;

color: black;

border-radius: 15px;

height: 30px;

width: 30%;

cursor: pointer;

}

.selector-element{

background-color: rgb(147, 165, 42);

color: white;

}

.last-container{

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

width: 100%;

font-weight: bold;

}

.next-button{

cursor: pointer;

}

.previous-button{

cursor: pointer;

}

.product-image{

background-color: transparent;

}

.line{

color: black;

}

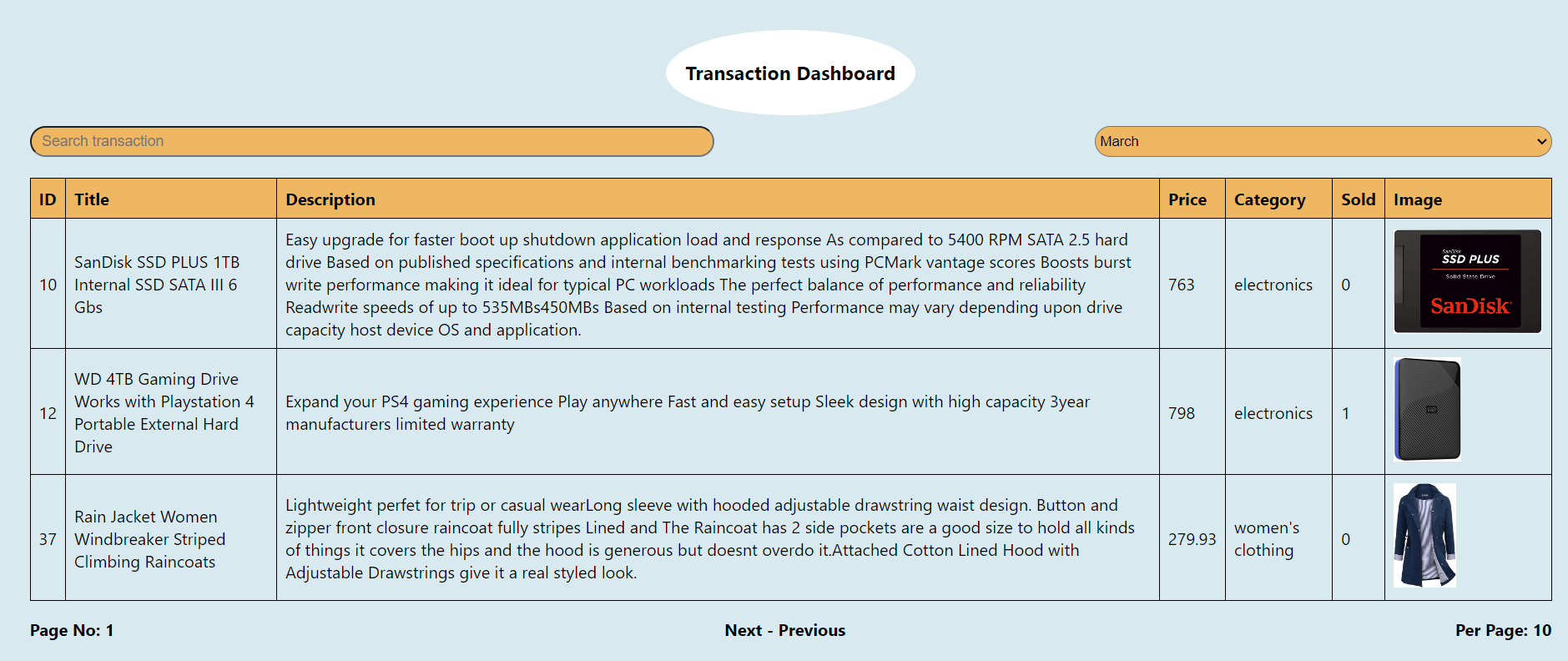
# OUTPUT SCREENSHOT:-

## 1-Server

A screenshot of a computer code

Description automatically generated

## 2-Transactions Table



## 3-Barchat

