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-Api.ts

```
import axios from 'axios';

import { ProductType } from '../types/types';

const API_URI = 'https://s3.amazonaws.com/roxiler.com/product_transaction.json';

const getProductData = async (): Promise<ProductType[]> => {

  try {

    const response = await axios.get(API_URI);

    return response.data;

  } catch (error) {

    console.error('Error fetching data from the API:', error);

    throw error;

  }

};

export { getProductData };
```

3-Analytics_Controller.ts

```
import { ApiError } from "../utils/ApiError";

import { ApiResponse } from "../utils/ApiResponse";

import { asyncHandler } from "../utils/asyncHandler";

import Product from "../models/product.model";

const statisticsOfTheProductRoutes = asyncHandler(async (req, res) => {

  //#swagger.tags = ['statistics']

  const { month } = req.query as any;

  if (!month) return res.status(400).json({ message: "Provide month" });

  const query = {

    $expr: {

      $eq: [{ $month: { $toDate: "$dateOfSale" } }, parseInt(month)],

    },

  };

  const data = await Product.find(query);

  const totalSaleAmount = data.reduce(

    (acc, product) => acc + product.price,

    0

  );

  const soldItem = data.filter((product) => product.sold === true).length;

  const notSoldItem = data.filter((product) => product.sold !== true).length;

  const response = {

    totalSaleAmount,
```

```
    soldItem,
    notSoldItem,
  }
  return res.status(200).json({ statusCode: 200, response, message: "", success: true });
});
const barChartOfTheProductRoutes = asyncHandler(async (req, res) => {
  //#swagger.tags = ['statistics']
  const { month } = req.query;
  const startDate = new Date(`${month}-01T00:00:00Z`);
  const endDate = new Date(`${month}-31T23:59:59Z`);
  const priceRanges = [
    { range: '0-100', min: 0, max: 100 },
    { range: '101-200', min: 101, max: 200 },
    { range: '201-300', min: 201, max: 300 },
    { range: '301-400', min: 301, max: 400 },
    { range: '401-500', min: 401, max: 500 },
    { range: '501-600', min: 501, max: 600 },
    { range: '601-700', min: 601, max: 700 },
    { range: '701-800', min: 701, max: 800 },
    { range: '801-900', min: 801, max: 900 },
    { range: '901-above', min: 901, max: Infinity }
  ];
  const priceRangeCounts = await Promise.all(
    priceRanges.map(
      async ({ range, min, max }) => {
        const count = await Product.countDocuments({
          dateOfSale: { $gte: startDate, $lte: endDate },
          price: { $gte: min, ...(max !== Infinity ? { $lte: max } : {}) }
        });
        return { range, count };
      }
    )
  );
  const response = { priceRangeCounts }
  return res.status(200).json(new ApiResponse(200, response, ""));
});

const pieChartOfTheProductRoutes = asyncHandler(async (req, res) => {
  //#swagger.tags = ['statistics']
```

```
  const { month } = req.query as any;

  if (!month || isNaN(parseInt(month))) {
    throw new ApiError(400, "Provide a valid month");
  }
  const query = {
    $expr: {
      $eq: [{ $month: { $toDate: "$dateOfSale" } }, parseInt(month)],
    },
  };

  const categoryCounts = await Product.aggregate([
    { $match: query },
    {
      $group: {
        _id: "$category",
        count: { $sum: 1 },
      },
    },
  ]);

  const response: { [key: string]: number } = categoryCounts.reduce(
    (acc, categoryCount) => {
      acc[categoryCount._id] = categoryCount.count;
      return acc;
    }, {}
  );
  return res.status(200).json(new ApiResponse(200, response, "Successfully retrieved pie chart data."));
});
const combinedDataAPI = asyncHandler(async (req, res) => {
  //#swagger.tags = ['statistics']
  const { month } = req.query as any;
  if (!month || isNaN(parseInt(month))) {
    throw new ApiError(400, "Provide a valid month");
  }
  const URL = `http://localhost:8000/api/v1/analytics/`;
  const [statistics, barChart, pieChart] = await Promise.all([
    fetch(`${URL}statistics?month=${month}`).then(response => response.json()),
    fetch(`${URL}bar-chart?month=${month}`).then(response => response.json()),
```

```
    fetch(`${URL}pie-chart?month=${month}`).then(response => response.json()),
  ]);
  const response = {
    statistics,
    barChart,
    pieChart,
  };
  return res.status(200).json(new ApiResponse(200, response, "Combined data retrieved successfully"));
});
export {
  statisticsOfTheProductRoutes,
  barChartOfTheProductRoutes,
  pieChartOfTheProductRoutes,
  combinedDataAPI
};
```

4-Product_controller.ts

```
import { ApiError } from "../utils/ApiError";
import { ApiResponse } from "../utils/ApiResponse";
import { asyncHandler } from "../utils/asyncHandler";
import Product from "../models/product.model";
import { getProductData } from "../api/api";
import { SearchParams } from "../types/types";

const initDataHandler = asyncHandler(async (req, res) => {
  //#swagger.tags = ['Products ']
  const existingProducts = await Product.find();
  if (existingProducts.length > 0) {
    return res.status(200).json(new ApiResponse(200, existingProducts, "Database already initialized with seed data"));
  }
  const response = await getProductData();
  // console.log(response);
  const product = await Product.insertMany(response)
  return res.status(200).json(new ApiResponse(200, product, "Database initialized with seed data"));
});
const getAllProductsData = asyncHandler(async (req, res) => {
  //#swagger.tags = ['Products ']
  const products = await Product.find().sort({ id: 'asc' });
```

```
  if (products.length === 0) {
    throw new ApiError(404, "No products found");
  }
  return res.status(200).json(new ApiResponse(200, products, "Products retrieved successfully"));
});
const searchProduct = asyncHandler(async (req, res) => {
  //#swagger.tags = ['Products ']
  // console.log(req.query)
  const query = constructorSearchQuery(req.query);
  const pageSize = 10;
  const pageNumber = parseInt(
    req.query.page ? req.query.page.toString() : "1"
  );
  if (isNaN(pageNumber) || pageNumber < 1) {
    throw new ApiError(400, "Invalid page number");
  }
  const skip = (pageNumber - 1) * pageSize;
  const product = await Product
    .find(query)
    .skip(skip)
    .limit(pageSize);

  const total = await Product.countDocuments();
  const response: SearchParams = {
    data: product,
    pagination: {
      total,
      page: pageNumber,
      pages: Math.ceil(total / pageSize)
    }
  }
  return res.status(200).json(new ApiResponse(200, response, "Products retrieved successfully"));
});
const constructorSearchQuery = (queryParams: any) => {
  const { searchText, selectedMonth } = queryParams;
  let constructedQuery: any = {};
  if (searchText) {
    constructedQuery.$or = [
      { title: new RegExp(searchText, "i") },
```

```
    { description: new RegExp(searchText, "i") },
    { price: !isNaN(parseFloat(searchText)) ? parseFloat(searchText) : null },
   ];
  }
  if (selectedMonth) {
   const startDate = new Date(`${selectedMonth}-01T00:00:00Z`);
   const endDate = new Date(`${selectedMonth}-31T23:59:59Z`);
   constructedQuery.dateOfSale = { $gte: startDate, $lte: endDate };
  }
  return constructedQuery;
};
export {
  initDataHandler,
  getAllProductsData,
  searchProduct
};
```

5-Product_models.ts

```
import { Schema, models, model } from "mongoose";
import { ProductType } from "../types/types";
const productSchema = new Schema<ProductType>(
  {
   id: { type: "number", required: true, },
   title: { type: "string", required: true, lowercase: true, trim: true, index: true, },
   description: { type: "string", required: true, lowercase: true, trim: true, index: true, },
   price: { type: "number", required: true },
   category: { type: "string", required: true },
   image: { type: "string", required: true },
   sold: { type: "boolean", required: true },
   dateOfSale: { type: "string", required: true },
  },
  { timestamps: true }
);

const Product = models.Product || model<ProductType>("Product", productSchema);
export default Product;
```

6-Types.ts

```
import { Document } from "mongoose";


export interface ProductType extends Document {
  _id: string;
  id: number;
  title: string;
  price: number;
  description: string;
  category: string;
  image: string;
  sold: boolean;
  dateOfSale: string;
  timestamp: Date;
}
export type SearchParams = {
  data: ProductType[];
  pagination: {
    total: number;
    page: number;
    pages: number;
  }
}
```

7-Analytics_route.ts

```
import { Router } from "express";
import {
  statisticsOfTheProductRoutes,
  barChartOfTheProductRoutes,
  pieChartOfTheProductRoutes,
  combinedDataAPI
} from "../controllers/analytics.controller";
const router = Router();
router.get("/statistics", statisticsOfTheProductRoutes)
router.get("/bar-chart", barChartOfTheProductRoutes)
router.get("/pie-chart", pieChartOfTheProductRoutes)
router.get("/combined-chart", combinedDataAPI)
export default router;
```

8-App.ts

```
import express, { Request, Response } from 'express';
import cors from 'cors';
import "dotenv/config";
import cookieParser from "cookie-parser";
import path from 'path';

import swaggerUi from "swagger-ui-express";
import swaggerOutput from "./json/swagger_output.json"

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(cors({
  origin: process.env.CORS_ORIGIN,
  credentials: true,
}));

app.use(express.static(path.join(__dirname, "../../frontend/dist")));

app.use(cookieParser());

import productRoutes from "./routes/product.routes"
import analyticsRoutes from "./routes/analytics.routes"

app.use("/api/v1/product", productRoutes)
app.use("/api/v1/analytics", analyticsRoutes)
// app.get("*", (req: Request, res: Response) => {
//   res.sendFile(
//     path.join(__dirname, "../../frontend/dist/index.html")
//   );
// })
const options = { explorer: true, }
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerOutput, options));

export { app };
```

9-Async_Handler.ts

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


interface AsyncRequestHandler {
  (req: Request, res: Response, next: NextFunction): Promise<any>;
}


const asyncHandler = (requestHandler: AsyncRequestHandler) => {
  return (req: Request, res: Response, next: NextFunction) => {
    Promise
      .resolve(requestHandler(req, res, next))
      .catch((err) => next(err));
  };
};


export { asyncHandler, AsyncRequestHandler };
```

10-Swagger.ts

```
import swaggerAutogen from 'swagger-autogen';


const config = {
  info: {
    version: "1.0.0",
    title: " MERN CODING CHALLENGE",
    description: " MERN CODING CHALLENGE",
  },
  servers: [
    { url: 'https://MERN CODING CHALLENGE.onrender.com', description: 'Production server' },
    { url: 'http://localhost:8000', description: 'Local server' },
  ],
  schemes: ['http', 'https'],
  tags: [],
}


const outputfile = './src/json/swagger_output.json';


const routes = [
  './src/app.ts',
```

```
]

const options = {
  openapi: '3.0.0',
  language: 'en-US',
  autoHeaders: true,
  autoBody: true,
  autoQuery: true,
  autoResponses: true,
};

swaggerAutogen(options)(outputfile, routes, config)
```

11-Analytics_Api_Cilent.ts

```
import axios from "axios";
import {
  STATISTICS_URL,
  BAR_CHART_URL,
  PIE_CHART_URL,
  COMBINED_CHART_URL,
} from "../config/config";
import { PieChartType } from "../types/types";

export const getStatisticsData = async (month: string) => {
  try {

    const queryParams = new URLSearchParams();
    queryParams.append('month', month || '');

    const response = await axios.get(`${STATISTICS_URL}?${queryParams}`);
    const { data } = response;

    // console.log("data", data.response)
    return data.response
  } catch (error) {
    console.error(error);
    throw error;
  }
```

```
};
export const getBarChartData = async (month: string) => {
  try {
    const queryParams = new URLSearchParams();
    queryParams.append('month', month || '');

    const response = await axios.get(`${BAR_CHART_URL}?${queryParams}`);
    const { data } = response;

    console.log("data", data.data)
    return data
  } catch (error) {
    console.error(error);
    throw error;
  }
};
export const getPieChartData = async (month: string) => {
  try {
    const queryParams = new URLSearchParams();
    queryParams.append('month', month || '');
    const response = await axios.get(`${PIE_CHART_URL}?${queryParams}`);
    const { data } = response;
    // console.log("data", data.data);
    const pieChartData: PieChartType = {
      data: data.data,
    };
    return pieChartData;
  } catch (error) {
    console.error(error);
    throw error;
  }};
export const combinedDataAPI = async (month: string) => {
  try {
    const queryParams = new URLSearchParams();
    queryParams.append('month', month || '');
    const response = await axios.get(`${COMBINED_CHART_URL}?${queryParams}`);
    // const response = await axios.get(`${COMBINED_CHART_URL}?month=03`);
    const { data } = response;
```

```
    // console.log("data", data.data)
    return data.data
  } catch (error) {
    console.error(error);
    throw error;
  }
};
```

12-Product_Api_Cilent.ts

```
import axios from "axios";
import { GET_ALL_PRODUCT_URL, SEARCH_PRODUCT_URL } from "../config/config";
import { ProductType } from "../types/types";

export const getAllProducts = async (): Promise<ProductType[]> => {
  try {
    const response = await axios.get(GET_ALL_PRODUCT_URL);
    const { data } = response;
    return data.data as ProductType[];
  } catch (error) {
    console.error(error);
    throw error;
  }
};

export const searchProducts = async (
  searchText: string,
  selectedMonth: string,

): Promise<ProductType[]> => {
  try {
    const queryParams = new URLSearchParams();
    queryParams.append('searchText', searchText || '');
    queryParams.append('selectedMonth', selectedMonth || '');


    const response = await axios.get(`${SEARCH_PRODUCT_URL}?${queryParams}`);

    // console.log(response)
    const { data } = response;
```

```
    // console.log(data.data.data)
    return data.data.data;
  } catch (error) {
    console.error(error);
    throw error;
  }
};
```

13-BarChartData.tsx

```
import { BarChartType } from '../../types/types';
import Chart from 'react-apexcharts';
import { ApexOptions } from 'apexcharts';
import { useMonth } from '../../contexts/MonthContext';

const BarChartData = ({ data }: BarChartType) => {
  // console.log(data);

  const { selectedMonthLabel } = useMonth();

  if (!data) {
    return <p className='flex items-center justify-center h-screen text-2xl text-purple-500'>Loading...</p>;
  }

  const series = [{
    name: "series-1",
    data: data.priceRangeCounts.map(item => item.count)
  }];

  const options: ApexOptions = {
    chart: {
      id: "basic-bar",
      width: '100%',
    },
    xaxis: {
      categories: data.priceRangeCounts.map(item => item.range),
    },
```

```
    responsive: [
      {
        breakpoint: 768,
        options: {
          chart: {
            width: '80%',
          },
        },
      },
      {
        breakpoint: 480,
        options: {
          chart: {
            width: '100%',
          },
        },
      },
    ],
  };

  return (
    <div className="w-full h-full bg-[#EDF6F6]">
      <h1 className="m-4 text-2xl">Bar Chart - {selectedMonthLabel}</h1>
      <Chart options={options} series={series} type="bar" width="100%" height="100%" />
    </div>
  );
}

export default BarChartData;
```

14-PieChartData.tsx

```
import Chart from 'react-apexcharts';
import { PieChartType } from '../../types/types';
import { ApexOptions } from 'apexcharts';
import { useMonth } from '../../contexts/MonthContext';

const PieChartData = ({ data }: PieChartType) => {

  const { selectedMonthLabel } = useMonth();
```

```
const seriesData = Object.values(data).map(value => value || 0);

const options: ApexOptions = {
  chart: {
    width: '100%',
    height: '100%',
    type: 'pie',
  },
  labels: ["electronics", "men's clothing", "women's clothing"],
  responsive: [
    {
      breakpoint: 768,
      options: {
        chart: { width: '80%', },
        legend: { position: 'bottom', },
      },
    },
    {
      breakpoint: 480,
      options: {
        chart: { width: '100%', },
        legend: { position: 'bottom', },
      },
    },
  ],
};

if (!data) {
  return (
    <p className='flex items-center justify-center h-screen text-2xl text-purple-500'>
      Loading...
    </p>
  );
}

return (
  <div className="h-full w-full bg-[#EDF6F6]">
    <h1 className="m-4 text-2xl">Pie Chart - {selectedMonthLabel}</h1>
```

```
      <Chart options={options} series={seriesData} type="pie" width="100%" height="100%" />
    </div>
  );
};

export default PieChartData;
```

15-index.html

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <link rel="icon" type="image/svg+xml" href="/vite.svg" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Products</title>
</head>

<body>
  <div id="root"></div>
  <script type="module" src="/src/main.tsx"></script>
</body>

</html>
```

OUTPUT SCREENSHOT:-

1-Transactions Table

Transaction List

Search...

March

#	Title	Description	Price	Category	Sold	Image
1	Fjallraven Foldsack No 1 Backpack Fits 15 Laptops	Your Perfect Pack For Everyday Use And Walks In The Forest. Stash Your Laptop Up To 15 Inches In The Padded Sleeve Your Everyday	$329.85	Men's Clothing	No	
2	Mens Casual Premium Slim Fit Tshirts	Slimfitting Style Contrast Raglan Long Sleeve Threebutton Henley Placket Light Weight Soft Fabric For Breathable And Comfortable Wearing. And Solid Stitched Shirts With Round Neck Made For Durability And A Great...	$44.60	Men's Clothing	No	
3	Mens Cotton Jacket	Great Outerwear Jackets For Springautumnwinter Suitable For Many Occasions Such As Working Hiking Camping Mountainrock Climbing Cycling Traveling Or Other Outdoors. Good Gift Choice For You Or Your...	$615.89	Men's Clothing	Yes	
4	Mens Casual Slim Fit	The Color Could Be Slightly Different Between On The Screen And In Practice. Please Note That Body Builds Vary By Person Therefore Detailed Size Information Should Be Reviewed Below On The Product Description.	$31.98	Men's Clothing	No	
5	John Hardy Womens Legends Naga Gold Silver Dragon Station Chain Bracelet	From Our Legends Collection The Naga Was Inspired By The Mythical Water Dragon That Protects The Oceans Pearl. Wear Facing Inward To Be Bestowed With Love And Abundance Or Outward For Protection.	$6950.00	Jewelery	No	
6	Solid Gold Petite Micropave	Satisfaction Guaranteed. Return Or Exchange Any Order Within 30 Days.Designed And Sold By Hafeez Center In The United States. Satisfaction Guaranteed. Return Or Exchange Any Order Within 30 Days.	$168.00	Jewelery	Yes	
7	White Gold Plated Princess	Classic Created Wedding Engagement Solitaire Diamond Promise Ring For Her. Gifts To Spoil Your Love More For Engagement Wedding Anniversary Valentines Day...	$99.90	Jewelery	Yes	
8	Pierced Owl Rose Gold Plated Stainless Steel Double	Rose Gold Plated Double Flared Tunnel Plug Earrings. Made Of 316l Stainless Steel	$32.97	Jewelery	No	
9	Wd 2tb Elements Portable External Hard Drive Usb 30	Usb 3.0 And Usb 2.0 Compatibility Fast Data Transfers Improve Pc Performance High Capacity Compatibility Formatted Ntfs For Windows 10 Windows 8.1 Windows 7 Reformatting May Be Required For Other...	$704.00	Electronics	Yes	
10	Sandisk Ssd Plus 1tb Internal Ssd Sata Iii 6 Gbs	Easy Upgrade For Faster Boot Up Shutdown Application Load And Response As Compared To 5400 Rpm Sata 2.5 Hard Drive Based On Published Specifications And Internal Benchmarking Tests Using Pcmark...	$763.00	Electronics	No	

1 2 3 4 5 6 Next

2-Transctions Statistics

Statistics - March

Total Sale	1840.93
Total Sold Item	1
Total not Sold item	2

3-Transactions Bar Chart