

## BACKEND TASK:

**Create an API to list the all transactions - API should support search and pagination on product transactions - Based on the value of search parameters, it should match search text on product title/description/price and based on matching result it should return the product transactions - If search parameter is empty then based on applied pagination it should return all the records of that page number - Default pagination values will be like page = 1, per page = 10:**

```
const express = require('express');
const axios = require('axios');

const app = express();
const PORT = 3000;

app.get('/api/transactions', async (req, res) => {
  const month = req.query.month; // Expected format: 'January', 'February', etc.
  const page = parseInt(req.query.page) || 1; // Default page is 1
  const perPage = parseInt(req.query.perPage) || 10; // Default per page is 10
  const search = req.query.search || ""; // Search term

  try {
    const response = await
    axios.get('https://s3.amazonaws.com/roxiler.com/product_transaction.json');
    const items = response.data;

    // Filter by month
    const filteredItems = items.filter(item => {
      const itemDate = new Date(item.dateOfSale);
      return itemDate.toLocaleString('default', { month: 'long' }) === month;
    });

    // Search functionality
    const searchResults = filteredItems.filter(item => {
      return (
        item.title.toLowerCase().includes(search.toLowerCase()) ||
        item.description.toLowerCase().includes(search.toLowerCase()) ||
        item.price.toString().includes(search)
      );
    });

    // Pagination
    const startIndex = (page - 1) * perPage;
    const paginatedItems = searchResults.slice(startIndex, startIndex + perPage);

    res.json({
```

```
      totalItems: searchResults.length,  
      currentPage: page,  
      totalPages: Math.ceil(searchResults.length / perPage),  
      items: paginatedItems,  
    });  
  } catch (error) {  
    res.status(500).send('Error fetching data');  
  }  
});  
  
app.listen(PORT);
```

**Create an API for pie chart Find unique categories and number of items from that category for the selected month regardless of the year.:**

```
const express = require('express');
const axios = require('axios');

const app = express();

app.get('/category-statistics', async (req, res) => {
  const month = req.query.month;
  const url = "https://s3.amazonaws.com/roxiler.com/product_transaction.json";

  try {
    const response = await axios.get(url);
    const data = response.data;

    const categoryCounts = { };

    data.forEach(item => {
      const dateOfSale = new Date(item.dateOfSale);
      const itemMonth = String(dateOfSale.getMonth() + 1).padStart(2, '0');
      if (itemMonth === month) {
        const category = item.category;
        if (categoryCounts[category]) {
          categoryCounts[category]++;
        } else {
          categoryCounts[category] = 1;
        }
      }
    });

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

const port = process.env.PORT || 3000;
app.listen(port);
```

Pretty print ☒

```
{  
  "women's clothing": 2,  
  "electronics": 2,  
  "men's clothing": 1,  
  "jewelery": 1  
}
```

**Create an API for bar chart ( the response should contain price range and the number of items in that range for the selected month regardless of the year ) :**

```
const express = require('express');
const axios = require('axios');

const app = express();
const PORT = 3000;

app.get('/api/bar-chart', async (req, res) => {
  const month = parseInt(req.query.month); // Expected format: 'MM'
  const priceRanges = [
    { range: '0-100', count: 0 },
    { range: '101-200', count: 0 },
    { range: '201-300', count: 0 },
    { range: '301-400', count: 0 },
    { range: '401-500', count: 0 },
    { range: '501-600', count: 0 },
    { range: '601-700', count: 0 },
    { range: '701-800', count: 0 },
    { range: '801-900', count: 0 },
    { range: '901-above', count: 0 },
  ];

  try {
    const response = await
    axios.get('https://s3.amazonaws.com/roxiler.com/product_transaction.json');
    const items = response.data;

    items.forEach(item => {
      const itemDate = new Date(item.dateOfSale);
      if (itemDate.getMonth() + 1 === month) {
        const price = item.price;

        if (price <= 100) priceRanges[0].count++;
        else if (price <= 200) priceRanges[1].count++;
        else if (price <= 300) priceRanges[2].count++;
        else if (price <= 400) priceRanges[3].count++;
        else if (price <= 500) priceRanges[4].count++;
        else if (price <= 600) priceRanges[5].count++;
        else if (price <= 700) priceRanges[6].count++;
        else if (price <= 800) priceRanges[7].count++;
        else if (price <= 900) priceRanges[8].count++;
        else priceRanges[9].count++;
      }
    });
  }
});
```

```
        res.json(priceRanges);
    } catch (error) {
        res.status(500).send('Error fetching data');
    }
});

app.listen(PORT);
```

Pretty print ☒

```
[
  {
    "range": "0-100",
    "count": 2
  },
  {
    "range": "101-200",
    "count": 0
  },
  {
    "range": "201-300",
    "count": 0
  },
  {
    "range": "301-400",
    "count": 0
  },
  {
    "range": "401-500",
    "count": 1
  },
  {
    "range": "501-600",
    "count": 0
  },
  {
    "range": "601-700",
    "count": 0
  },
  {
    "range": "701-800",
    "count": 0
  },
  {
    "range": "801-900",
    "count": 0
  },
  {
    "range": "901-above",
    "count": 0
  }
]
```

**Create an API which fetches the data from all the 3 APIs mentioned above, combines the response and sends a final response of the combined JSON:**

```
const express = require('express');
const axios = require('axios');

const app = express();
const port = 3000;

app.get('/data', async (req, res) => {
  try {
    const [response1, response2, response3] = await Promise.all([
      axios.get('http://localhost:3000/statistics?month=2022-01'),
      axios.get('http://localhost:3000/category-statistics?month=01'),
      axios.get('http://localhost:3000/api/transactions?month=January&page=1&perPage=10&search=backpack'),
    ]);

    const combinedData = {
      data1: response1.data,
      data2: response2.data,
      data3: response3.data,
    };

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

app.listen(port);
```



## FRONTEND TASK:

### 1.Ts-analyticsApi-client:

```
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;
  }
};

```

## 2.productApiClient-Api:

```
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;
}
};
```

### **3.Main.tsx:**

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'
import { SearchContextProvider } from './contexts/SearchContext.tsx'
import { MonthProvider } from './contexts/MonthContext.tsx'
```

```
ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <SearchContextProvider>
      <MonthProvider>

        <App />
      </MonthProvider>
    </SearchContextProvider>
```

</React.StrictMode>,

)

OUTPUT:

Transaction List

Search...

September

#	Title	Description	Price	Category	Sold	Image
21	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	\$439.80	Men's Clothing	Yes	
22	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...	\$111.50	Men's Clothing	No	
23	Mens Cotton Jacket	Great Outerwear Jackets For Springautumnwinter Suitable For Many Occasions Such As Working Hiking Camping Mountainrock Climbing Cycling Traveling Or Other...	\$559.90	Men's Clothing	No	
24	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...	\$31.98	Men's Clothing	Yes	
25	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...	\$695.00	Jewelery	No	

27°C Mostly cloudy 20:55 15-08-2024

#	Title	Description	Price	Category	Sold	Image
	Station Chain Bracelet	Wear Facing Inward To Be Bestowed With Love And...				
26	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 Exchan...	\$504.00	Jewelery	No	
27	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...	\$79.92	Jewelery	Yes	
28	Pierced Owl Rose Gold Plated Stainless Steel Double	Rose Gold Plated Double Flared Tunnel Plug Earrings. Made Of 316L Stainless Steel	\$131.88	Jewelery	Yes	
29	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...	\$640.00	Electronics	No	
30	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 Intern...	\$218.00	Electronics	Yes	

Prev 1 2 3 4 5 6 Next

Activate Windows  
Go to Settings to activate Windows.

27°C Mostly cloudy 20:56 15-08-2024

