Backend Implementation

Technology Stack: Node.js, Express.js, MongoDB (Mongoose for schema handling)

1. Initialize the Database

Fetch data from the provided third-party API and populate the database.

Steps:

1. Install dependencies:

bash

Copy code

```
npm install express mongoose axios
```

2. Create a Mongoose schema for transactions:

javascript

Copy code

```
const mongoose = require('mongoose');

const transactionSchema = new mongoose.Schema({
  id: String,
  title: String,
  description: String,
  price: Number,
  category: String,
  dateOfSale: Date,
  sold: Boolean
});

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

3. Fetch and seed the database:

javascript

Copy code

```
const axios = require('axios');
const Transaction = require('./models/transaction');
const seedDatabase = async () => {
   const url = 'https://s3.amazonaws.com/roxiler.com/product_transaction.json';
   const response = await axios.get(url);
   await Transaction.insertMany(response.data);
};
```

2. Create APIs

a. Transactions API:

javascript

Copy code

```
app.get('/api/transactions', async (req, res) => {
   const { page = 1, perPage = 10, search = '', month } = req.query;

   const query = {
      ...(month ? { dateOfSale: { $regex: `-${month}-`, $options: 'i' } } : {}),
      ...(search ? { $or: [
         { title: { $regex: search, $options: 'i' } },
         { description: { $regex: search, $options: 'i' } },
         { price: Number(search) }
      ] } : {})
   };

   const transactions = await Transaction.find(query)
      .skip((page - 1) * perPage)
      .limit(Number(perPage));

   res.json(transactions);
});
```

b. Statistics API:

javascript

Copy code

```
app.get('/api/statistics', async (req, res) => {
  const { month } = req.query;
  const monthRegex = new RegExp(`-${month}-`, 'i');

  const totalSaleAmount = await Transaction.aggregate([
    { $match: { dateOfSale: monthRegex } },
    { $group: { _id: null, total: { $sum: '$price' } } }
  ]);

  const totalSold = await Transaction.countDocuments({ dateOfSale: monthRegex, sold: true });
  const totalNotSold = await Transaction.countDocuments({ dateOfSale: monthRegex, sold: false
});

  res.json({ totalSaleAmount, totalSold, totalNotSold });
});
```

c. Bar Chart API:

javascript

Copy code

```
app.get('/api/bar-chart', async (req, res) => {
  const { month } = req.query;
  const monthRegex = new RegExp(`-${month}-`, 'i');

  const priceRanges = [
    { min: 0, max: 100 },
    { min: 101, max: 200 },
    { min: 201, max: 300 },
    { min: 301, max: 400 },
    { min: 401, max: 500 },
    { min: 501, max: 600 },
    { min: 601, max: 700 },
    { min: 701, max: 800 },
    { min: 801, max: 900 },
    { min: 901, max: Infinity },
  ];

  const result = await Promise.all(priceRanges.map(async ({ min, max }) => {
    const count = await Transaction.countDocuments({
      dateOfSale: monthRegex,
      price: { $gte: min, $lt: max }
    });
    return { range: `${min}-${max}`, count };
  }));

  res.json(result);
});
```

d. Pie Chart API:

javascript

Copy code

```
app.get('/api/pie-chart', async (req, res) => {
  const { month } = req.query;
  const monthRegex = new RegExp(`-${month}-`, 'i');

  const categories = await Transaction.aggregate([
    { $match: { dateOfSale: monthRegex } },
    { $group: { _id: '$category', count: { $sum: 1 } } }
  ]);

  res.json(categories);
});
```

e. Combined API:

javascript

Copy code

```
app.get('/api/combined', async (req, res) => {
  const { month } = req.query;

  const [transactions, statistics, barChart, pieChart] = await Promise.all([
    Transaction.find({ dateOfSale: { $regex: `-${month}-`, $options: 'i' } }),
    axios.get('/api/statistics', { params: { month } }),
    axios.get('/api/bar-chart', { params: { month } }),
    axios.get('/api/pie-chart', { params: { month } })
  ]);

  res.json({ transactions, statistics: statistics.data, barChart: barChart.data, pieChart: pieChart.data
});
});
```

Frontend Implementation

Technology Stack: React.js, Chart.js (or any chart library)

1. **Setup React**:

bash

Copy code

```
npx create-react-app roxiler-challenge
cd roxiler-challenge
npm install axios chart.js react-chartjs-2
```

2. **Create Components**:
   - **Dropdown and Table**: Use the transactions API to display paginated data with a search box and a dropdown for months.
   - **Statistics Cards**: Fetch and display total sales, sold items, and unsold items.
   - **Bar Chart and Pie Chart**: Render charts using the BarChart and PieChart APIs.
3. **Sample Code**:

javascript

Copy code

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

```
const Dashboard = () => {
  const [transactions, setTransactions] = useState([]);
  const [statistics, setStatistics] = useState({});
  const [barChartData, setBarChartData] = useState([]);
  const [pieChartData, setPieChartData] = useState([]);
  const [month, setMonth] = useState('March');

  useEffect(() => {
    const fetchData = async () => {
      const transRes = await axios.get('/api/transactions', { params: { month } });
      const statRes = await axios.get('/api/statistics', { params: { month } });
      const barRes = await axios.get('/api/bar-chart', { params: { month } });
      const pieRes = await axios.get('/api/pie-chart', { params: { month } });

      setTransactions(transRes.data);
      setStatistics(statRes.data);
      setBarChartData(barRes.data);
      setPieChartData(pieRes.data);
    };

    fetchData();
  }, [month]);

  return (
    <div>
      {/* Dropdown, Table, Charts */}
    </div>
  );
};
```

```
export default Dashboard;
```