MM804 Assignment 3

Visualization of 2017 Edmonton General Elections

Live: https://mm804-visualization-assignment-d.netlify.app/
Github: https://github.com/dharanUoA/MM804-Visualization-Assignment
Dataset: LINK

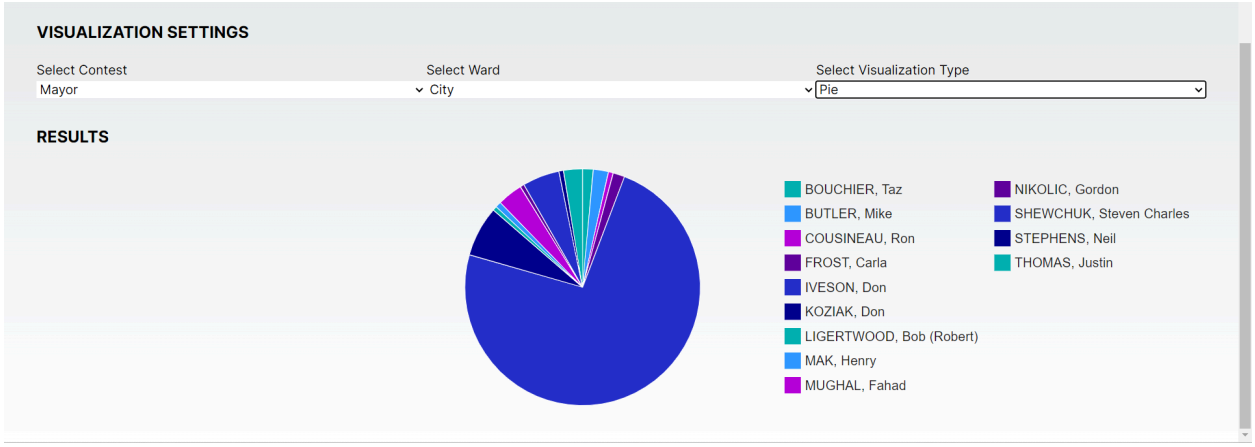
This assignment aims to create various data visualizations using interactive plots. There were no restrictions on choosing a programming language to use. I used Next.js (a javascript framework based on react.js). I used `2017 Edmonton General Election - Official Results` dataset from https://data.edmonton.ca/browse. Along with next mui charts library is used to visualize data.

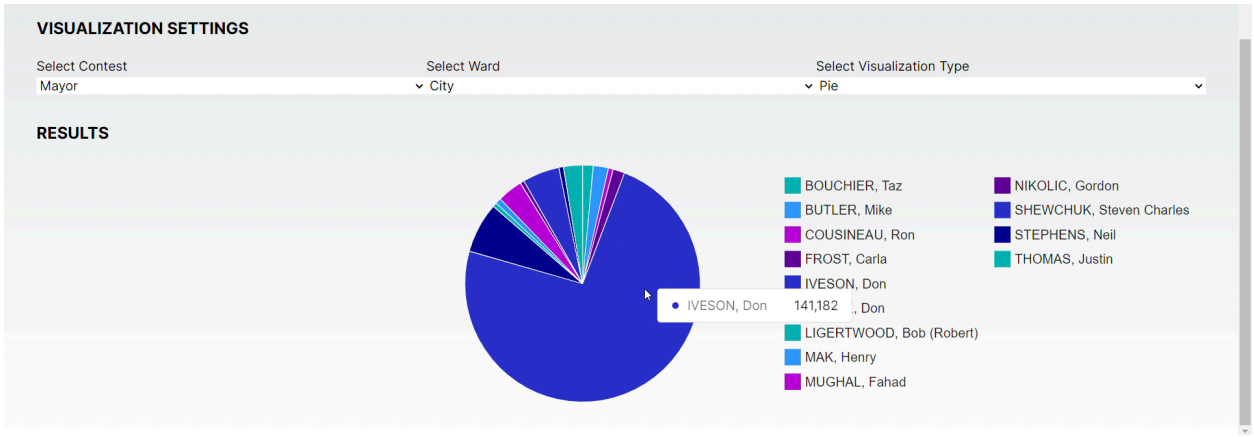
1. Dataset Info

- Dataset name: 2017 Edmonton General Election - Official Results
- File format & size: N/A (No dataset downloaded. API Endpoint was provided. Hence used those. API ENDPOINT: https://data.edmonton.ca/resource/gg6p-rhvt.json)
- Number of data points: 131 rows

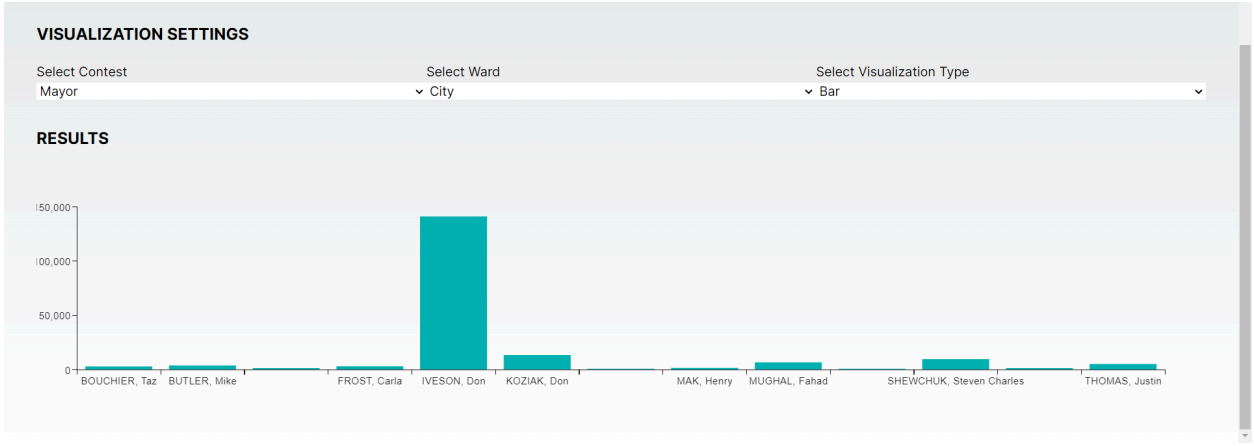
2. First 10 Rows - Screenshot

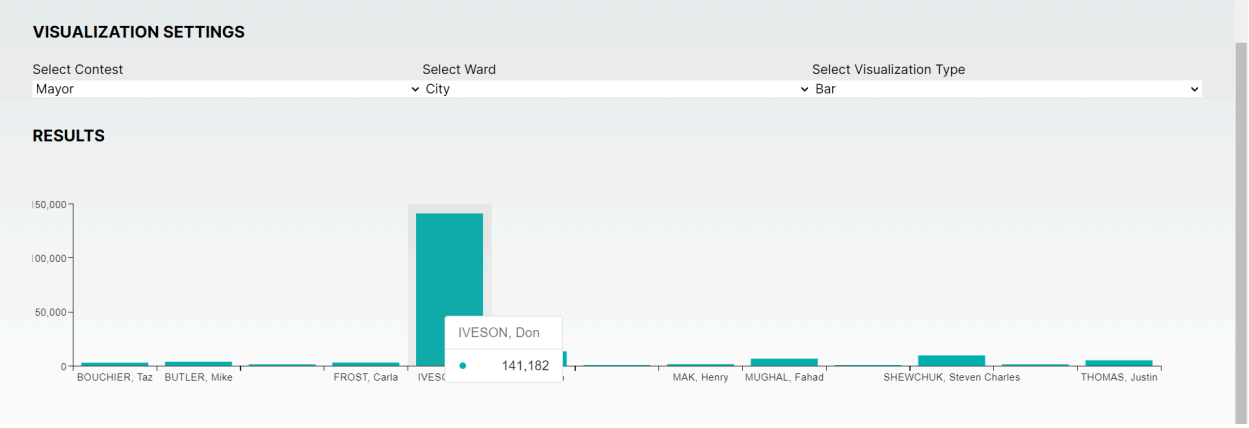
3. First Chart - Screenshot (with and without hover)





4. Second Chart - Screenshot (with and without hover)





5. Source-code Summary

- Function for getting API data

```javascript
export async function getData() {
 try {
  const response = await fetch(process.env.NEXT_PUBLIC_GET_DATA_URL);
  if (response.ok) {
   return response.json();
  }
 } catch (e) {
  console.log(e);
 }
}
```

- .env file where `process.env.NEXT_PUBLIC_GET_DATA_URL` value is stored.

```unset
NEXT_PUBLIC_GET_DATA_URL=https://data.edmonton.ca/resource/gg6p-rhvt.json
```

- React component code for displaying the first 10 rows of the dataset. Here, `data` is values obtained from the API call.

```javascript
import React from "react";
```

```
export async function TableView({ data }) {
 const dataToDisplay = data.slice(0, 10);
 return (
  <>
   <div className="w-full">
    <div className="text-xl uppercase font-bold mb-5">First 10 data points</div>
    <table className="table-auto w-full mb-8">
     <thead>
      <tr>
       <th>#</th>
       <th>Election data</th>
       <th>Contest</th>
       <th>Ward name</th>
       <th>Candidate name</th>
       <th>Votes received</th>
       <th>Precent</th>
      </tr>
     </thead>
     <tbody>
      {dataToDisplay.map((item, index) => (
       <tr key={index}>
        <td className="text-center">{index + 1}</td>
        <td className="text-center">
         {new Date(item.election_date).toDateString()}
        </td>
        <td className="text-center">{item.contest}</td>
        <td className="text-center">{item.ward_name}</td>
        <td className="text-center">{item.candidate_name}</td>
        <td className="text-center">{item.votes_received}</td>
        <td className="text-center">{item.percent} %</td>
       </tr>
      ))}
     </tbody>
    </table>
   </div>
  </>
 );
}
```

- Function to generate visualization data parameters.

JavaScript

```
const [pieChartData, setPieChartData] = useState();
const [barChartData, setBarChartData] = useState();

// prepare visualtion data and parameters
const visualizeData = (contest, ward, type) => {

  // if no value selected return
  if (!contest || !ward || !type) return;

 // filter data based on selected values
  const filteredColumns =
   data.filter(
    (item) => item.contest == contest && item.ward_name == ward
   ) ?? [];

 // separate login based on visualization type
  switch (type) {
   case "pie":
    // prepare data for pie chart
    const pieSeriesData = filteredColumns.map((item, index) => ({
     id: index, // unique index for each data
     value: +item.votes_received, // amount
     label: item.candidate_name, // label value to display
    }));
    if (pieSeriesData) {
     setPieChartData({ seriesData: [{ data: pieSeriesData }] });
    } else {
     setPieChartData(null);
    }
    setBarChartData(null);
    break;
   case "bar":
   // prepare data for bar chart
    const barSeriesData = filteredColumns.map(
     (item, index) => +item.votes_received
    );
   // set candidate names in x-axis name
    const axisData = filteredColumns.map((item) => item.candidate_name);
    if (barSeriesData) {
     setBarChartData({
      seriesData: [{ data: barSeriesData }],
      axisData: [{ scaleType: "band", data: axisData }],
     });
    } else {
```

```
     setBarChartData(null);
    }
    setPieChartData(null);
    break;
  }
 };


// visualize charts
return <>
       {pieChartData && (
             <PieChart series={pieChartData.seriesData} height={300} />
        )}
        {barChartData && (
             <BarChart
                height={300}
                series={barChartData.seriesData}
                xAxis={barChartData.axisData}
              />
)}</>;
```

6. Readme section

- The readme section is available at: https://github.com/dharanUoA/MM804-Visualization-Assignment