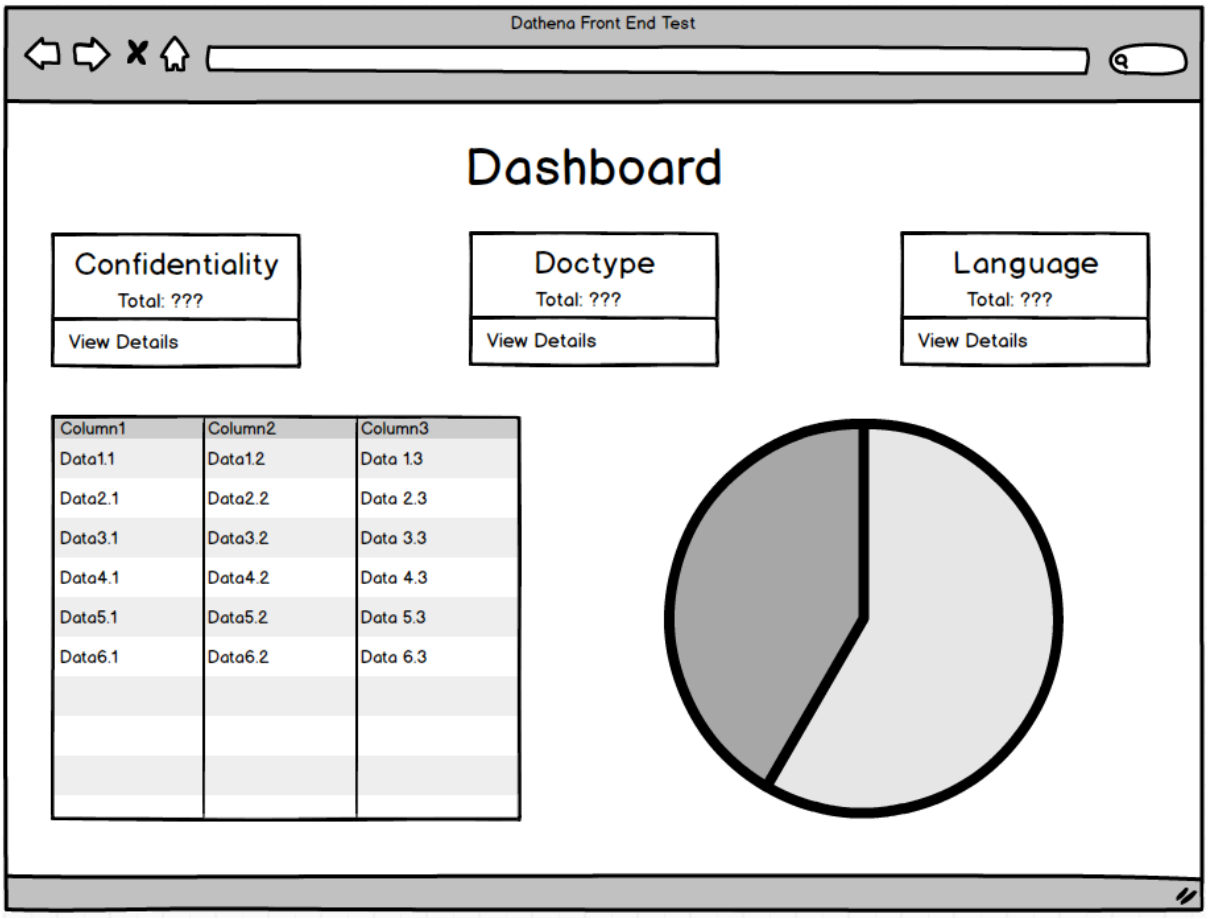
Dathena Front End Test

Applicant: Lam Ngoc Tuan

Wireframe

I use Balsamiq to draw the mockup of this webApp. Since this is just a Simple Page, only one wireframe is enough



The dashboard will contain

3 panel with the total number of documents of each category. If user want to see in detail, they can click in View Details to see it

After clicking at View Details of a category, a table and pie chart will appear to display more detail

Here they can see the table of subtypes of the chosen category and also the visualization of its population

Framework

The framework I am choosing is to create a ReactJS App since this is the core UI of the company.

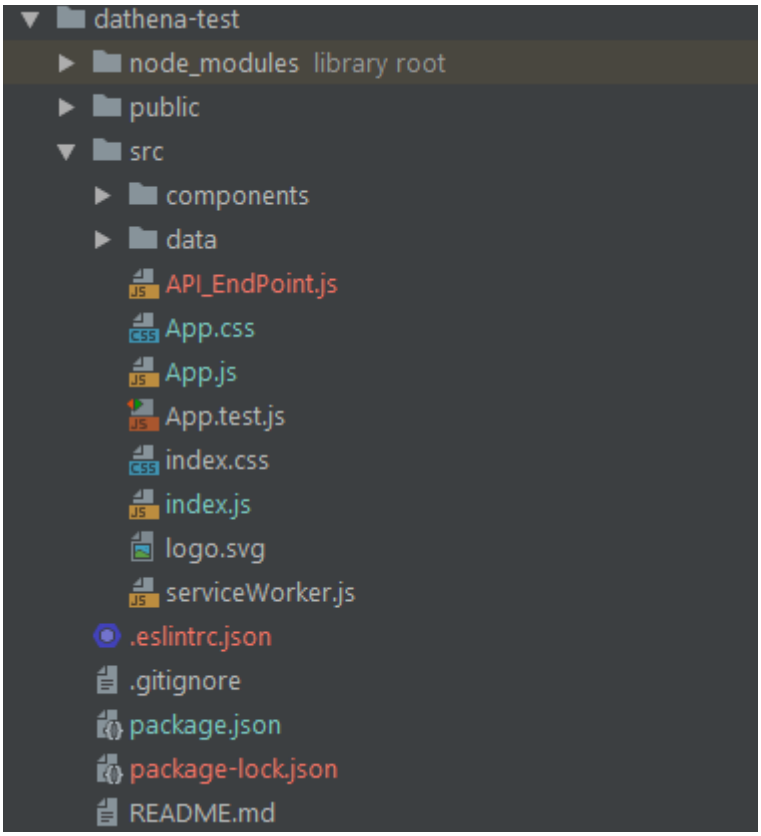
Code quality Tools

I use Eslint AirBnb for this test

Third party library

Bootstrap, FontAwesome, React Chart JS 2, is used in this app.

Directory Structure



- A package.json file which have all the dependencies and command
- An eslint file for code checking, this is only for developing environment
- Src folder contains the WebApp
    - App.js will be the file contains all of the code for the dashboard.
    - 2 components which is the table and the pie chart in an additional directory name “components”. This is just a practice to encapsulate components for future reuse.
    - The data folder contains all the json file

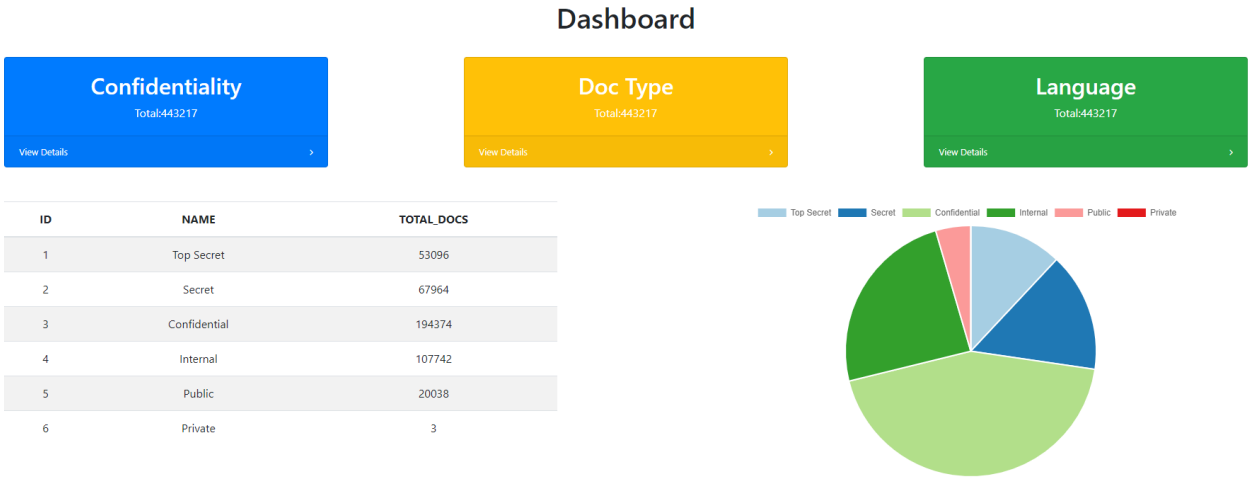
Installation

1/ **Run the command terminal** at the main directory (which contains package.json file)

2/ Please use npm to install the dependencies needed for the app. Doing it by run **npm install**

3/ Run **npm start** to run the app. The app should appear in port 3000 (http://localhost:3000)

Snapshot



Code explanation

App.js

The main JS file is App.js . Here I create a class App and define some state for it.

```
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      confidentiality: confidentialityJson,
      docType: docTypeJson,
      language: languageJson,
      viewConfidentiality: false,
      viewDoctype: false,
      viewLanguage: false,

    };
    this.getTotalConfidentiality = this.getTotalConfidentiality.bind(this);
    this.getTotalDocType = this.getTotalDocType.bind(this);
    this.getTotalLanguage = this.getTotalLanguage.bind(this);
  }
```

Confidentiality, doctype and language will store the data for the corresponding category after retrieving it from the json file.

viewConfidentiality, viewDoctype and viewLanguage is to keep track which one they want to see.

In order to retrieve the data, for this test, I directly imported

```
// I only take the data json file since it already contain the information of the correspondign label json file.
import confidentialityJson from './data/confidentiality_data.json';
import docTypeJson from './data/doctype_data';
import languageJson from './data/language_data';
```

However, in the real app, the data usually took it from another server. In order to do that, I create a js file name API_EndPoint to define the needed url and a sample of code inside componentDidMount to fetch the data from the url. However, in the context of this test, this is no need so I just comment them out

```
// Only for retrieving data from API server
// import { API } from './API_EndPoint';
```

```
componentDidMount() {
    // Since the current code is to retrive the data directly from the local file, I use import statement above to do it
    // However, in a real app, the data usually come from an API server. If that is the case, we need to fetch it like
    // below and wrap it in the componentDidMount.
    // fetch('').then(result => result.json()).then(result => console.log(result));
}
```

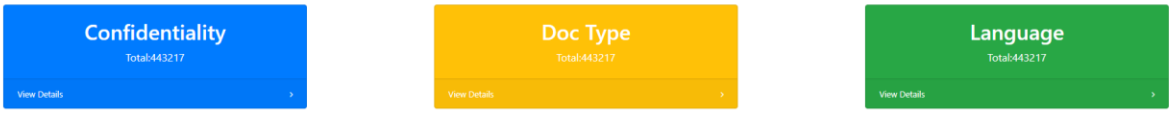
Total number of documents will be calculated first after getting the data by using simple maps and reduce of Javascript

```
// Get the sum of docs for each category, this will be display in the top panels
getTotalConfidentiality() {
    const { confidentiality } = this.state;
    return confidentiality.map(data => data.total_docs).reduce((sum, data) => sum + data);
}

getTotalDocType() {
    const { docType } = this.state;
    return docType.map(data => data.total_docs).reduce((sum, data) => sum + data);
}

getTotalLanguage() {
    const { language } = this.state;
    return language.map(data => data.total_docs).reduce((sum, data) => sum + data);
}
```

By doing this , I reflect it in the app



If user wish to view more details , they click at the corresponding “View Details” button. This will set the corresponding view in state to true and display the correct table and graph.

Lets say user click View Details of Confidentiality

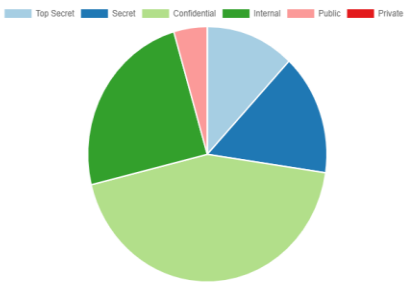
```
onClick={() => this.setState({ viewConfidentiality: true, viewDoctype: false, viewLanguage: false })}
```

viewConfidentiality will be set to true , and the other two become false

```
{(viewConfidentiality) ? (
  <div className="row justify-content-around">
    <div className="col-5">
      <DataTable data={confidentiality} />
    </div>
    <div className="col-5">
      <PieChart data={confidentiality} />
    </div>
  </div>
) : ''}
```

Display Data Table and Pie Chart. This is the two components I write and put it in the components folder. After that , this will appear

ID	NAME	TOTAL_DOCS
1	Top Secret	53096
2	Secret	67964
3	Confidential	194374
4	Internal	107742
5	Public	20038
6	Private	3



DataTable.js

```
function DataTable(props) {
  const { data } = props;
  // Retrieve the column name based on the first element
  const columns = Object.keys(data[0]);
  return (
    <table className="table table-striped">
      <thead>
        <tr>
          {
              columns.map(column => <th>{column.toLocaleUpperCase()}</th>)
          }
        </tr>
      </thead>
      <tbody>
        { data.map(each => (
          <tr>
            {
              columns.map(column => <td>{each[column]}</td>)
            }
          </tr>
        ))

        }
```

Since this is just a Pure component, it does not retrieve data or update data, I just make it simple as a function and return the table (no state)

Here I get the column name based on the first element, then fill in the remaining table

Redux ?

From my experience working with React, its better to integrate with Redux. However, since this is just a single page application with not many components, using Redux not utilize its advantage. That's why I didn't use it.

Thanks for your time reading this. I look forward to hearing from you.