**DOCUMENTATION**

This documentation is a go through on the case study given to me in the BCG Recruitment process.

This documentation covers the following aspects: -

* Covers on what is achieved in the assignment that is made.
* Covers on the tech stack used to achieve the requirement and why it was used.
* Covers the UI interface implemented using ANT-D module for both components and charts (according to requirement).
* Covers on key implementation that is done to enhance the user satisfaction.
* Covers on future scope with this application as enhancements and features.

REQUIREMENTS ACHIEVED

1. Implemented a Dashboard where a user can see all the application with their statuses.
2. Implemented an action modal where a user can edit their actions and change the status of the application.
3. There are 4 statuses of Application.
   1. Pending
   2. In Progress
   3. Approved
   4. Connection released
4. Implemented a visualization for the month on month application footfall mechanism.

TECHNOLOGIES USED

1. Frontend – React.JS, react-router-dom, ant-d charts library, ant-d component library, ApolloClient for fetching graphql
2. Backend – Django for Python, graphene as a middleware for processing graphql on dajngo, GraphQL for developing APIs

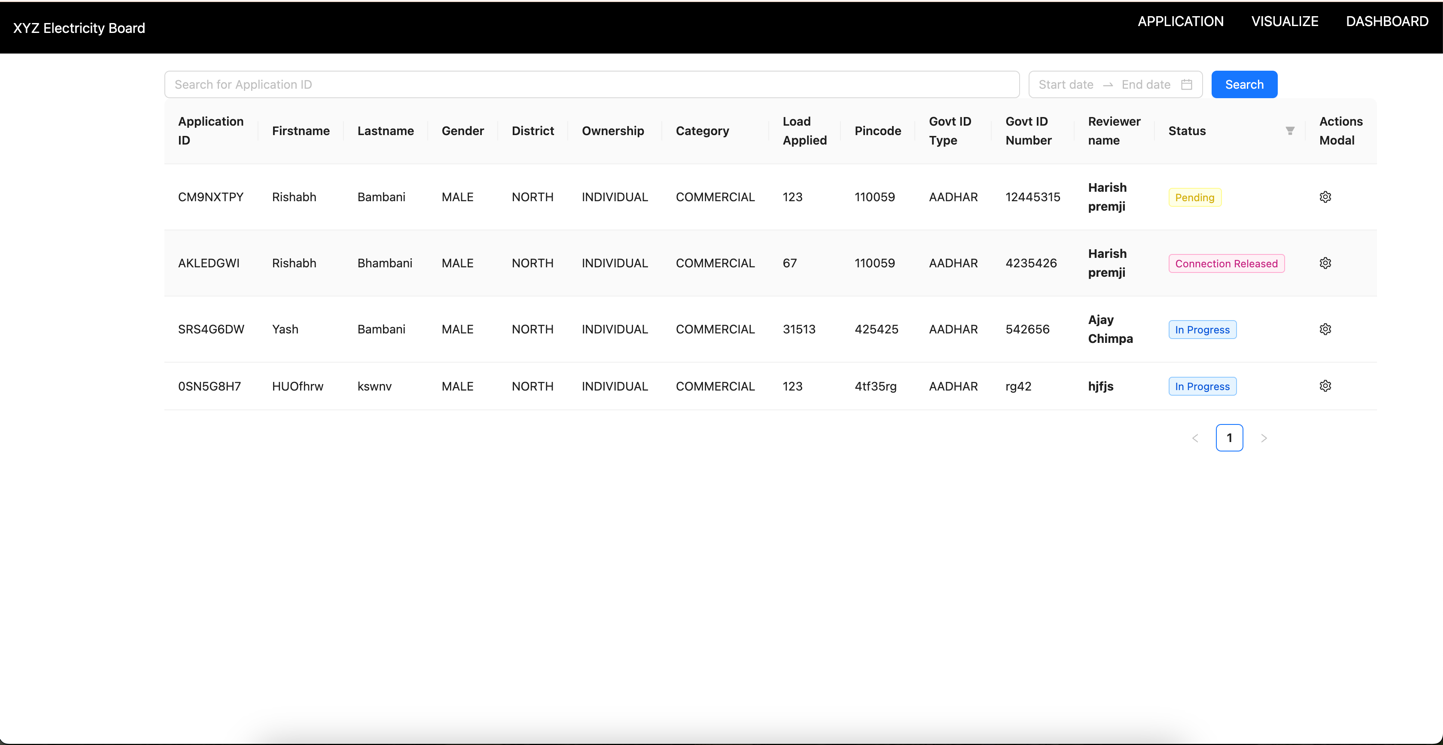
Why Django ?

Django is widely popular for its multiple use cases such as snappy behaviour and MVT architecture which is highly scalable describes why companies like pinterest, instagram use this technology,

Why GraphQL ?

According to AWS , “GraphQL’s focus has been on API performance and flexibility.” We can tweak with APIs and develop them based on certain business use cases rather than fetching them from classic REST API Architecture.

UI OVERVIEW



**Dashboard component**

Table implemented using AntDesign Component library which is flexible with implementing a lot of features such as filtering, sorting, adding icons making user experience more satisfying.

A screenshot of a computer

Description automatically generated

**Actions Modal**

This modal is for reviewers to review their assigned applications and take necessary actions over it, there itself they can change the statuses as well and a mutation is called to save there inputs. This is implemented using Modal component.



**The Filter functionality**

The filter functionality takes **exact** application ID for filtering over the applications and filter it out, this is quite handy if a applicant comes up with their application ID to know their status this feature will come out to be handy, there is a API built for that which is specific to searching.

Next feature that comes out to be handy is filtering the applications based on from and to dates. This is part of the same API and if user wants to know that what applications were requested within a certain time frame we can filter the applications out.

A screenshot of a computer

Description automatically generated**Visualization**

Here a user can visualize that what are the amount of applications that are coming month on month as usual it is a specific requirement which has its own API written and graphql comes out to be handy in this case.

SOME KEY IMPLEMENTATIONS

* Whenever a new application is there a reviewer is assigned right then and there and the default status goes to in progress. (Please read the future scope for this)
* How the reviewers are being assigned is implemented using the fact that whomsoever user out of the reviewers table will be having less number of applications we will assign the application to them this will increase the user satisfaction as it will maintain a balance in the application distribution.
* The actions modal is a quick lookup for seeing a particular application apart from table view for a specific applicant, here a reviewer change the load, add his/her comments and change the status of the application as well.

FUTURE SCOPE

1. We can implement cron jobs and AWS events onto whenever a user is assigned with a new application they can get a new email for the same.
2. We can use Django Q advance queries for querying in database models for implementing debouncing based searching mechanism for now the searching is for exact application ID considering the time constraint of this assignment.
3. When the status is changed or an application is approved we can notify the applicant regarding the statuses either by a text message service or via email.