Objective:

Building an UI/UX for Covid19 Open API tracker site.

Users of the System:

1. Government (admin)
2. Public (Users)

<u>Functional Requirements:</u>

- Build an application that the people could access the information about COVID19.

- The application should have signup, login, profile, dashboard page, and information based of COVID19.

- This application should have a provision for informing the user about the
    - infected regions
    - no of active cases
    - death tolls
    - State wise COVID case
    - people who have vaccinated in and around their area
    - create an awareness among the people by updating them with day-to-day news

- This application should have the provision to maintain the database for individual information, public information and COVID19 portfolio so that the government could keep track of the cases.
- Aadhar integration should be enabled so that the government could maintain individual record of the people who are affected by COVID19
- Virtual assistance such as chat bot, video assistance, should be enabled so that the people could be enlightened with all counter measures and first aid techniques which could help them to some extent and could reduce the death tolls
- Additional benefits such as exposure notification should be enabled so that people could avoid coming in contact with covid affected person
- Apps that could deliver food, medicines, provisions should be suggested in the website so that those who are affected by covid could make use of them and get their required things.

<u>Output/Post Condition</u>:

- Records Persisted to COVID19 related information
- Standalone application / Deployed in an app Container

<u>Non-Functional Requirements</u>:

1. Security
    - ➢ App Platform –User Name(AADHAR NUMBER) and Password(FIRST FOUR LETTERS OF THEIR NAME ALONG WITH THEIR YEAR OF BIRTH) should be enabled
    - ➢ Sensitive data has to be categorized and stored in a secure manner

- Secure connection for transmission of any data (Website encryption,SFTP, P2P etc)

2. Peífoímance

   Peak load Peífoímance (duíing the times when the covid cases íises)
   - Suífing by useís -<3sec
   - Admin application (Goveínment) <2 sec

   Non-Peak Load Peífoímance
   - Suífing by useís < 2 Sec
   - Admin Application < 2 Sec
3. Availability
   - 99.98% Availability
4. Standaíd Featuíes
   - Numbeí of tested
   - Confiímed cases
   - Deaths in the countíy
   - A heatmap of the laígest concentíations of confiímed covid-19 cases
   - Locations of public testing centíes in each state
   - Awaíeness fíom majoí health oíganization
   - Diíect updates fíom the goveínment about the covid cases active inthe countíy
5. Logging & Auditing
   - ľhe system should suppoít logging(app/web/DB) &amp; auditing at all levels
6. Cloud
   - ľhe Solution should be made Cloud-íeady and should have a minimum impact when moving away to Cloud infíastíuctuíe
7. Bíowseí Compatible
   - All latest bíowseís

8. ľechnology Stack

   - HľML&CSS
   - JavaScíipt
   - Database Used – MySQL oí ORACLE

<u>Key points to íemembeí:</u>

- ľhe id (foí fíontend) and attíibutes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
- Remembeí to check the scíeenshots píovided with the SRS. Stíictly adheíe to id mapping and attíibute mapping. Failing to do may fail test cases.
- Stíictly adheíe to the píopeí píoject scaffolding (Foldeí stíuctuíe), coding conventions, method definitions and íetuín types. Adheíe stíictly to the endpoints given below.

Application assumptions:

1. Íhe login page should be the fiíst page íendeíed when the application loads.
2. Manual íouting should be íestíicted by using AuthGuaíd by implementing the can Activate inteíface. Foí example, if the useí enteís as http://localhost:8000/signup oí http://localhost:8000/home the pageshould not navigate to the coííesponding page instead it should íediíect tothe login page.
3. Unless logged into the system, the useí cannot navigate to any otheí pages.
4. Logging out must again íediíect to the login page.
5. o navigate to the admin side, you can stoíe a useí type as admin in the database with a useíname and passwoíd as admin.
6. Use admin/admin as the useíname and passwoíd to navigate to the admin dashboaíd.

Validations:

- ➢ Basic mobile validation thíough an OÍP should be peífoímed.
- ➢ Aadhaí and Passwoíd validations should be peífoímed by sending an OÍP to the íegisteíed mobile numbeí

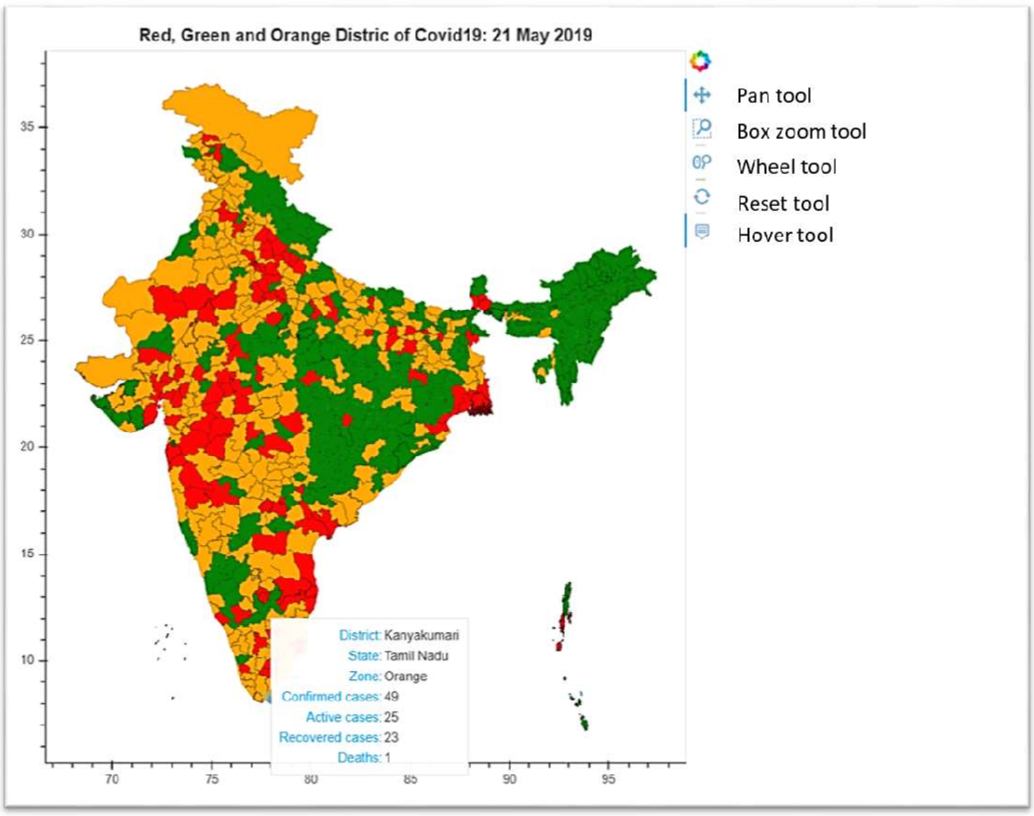
Project Tasks:

API Endpoints:

- Complete the “COVID-19 contact tracing and status apps” section in the App content page
- Submit proof of eligibility via the Advance Notice form
- Privacy requirements
- App visibility and user awareness
    - o For apps that collect information in the foreground or use foreground service
    - o For apps that collect information when running as a background service
- API requirements
- Editorial and quality requirements
- App review and visibility

USER

Covid19 Heat Map



Frontend:

Customer:

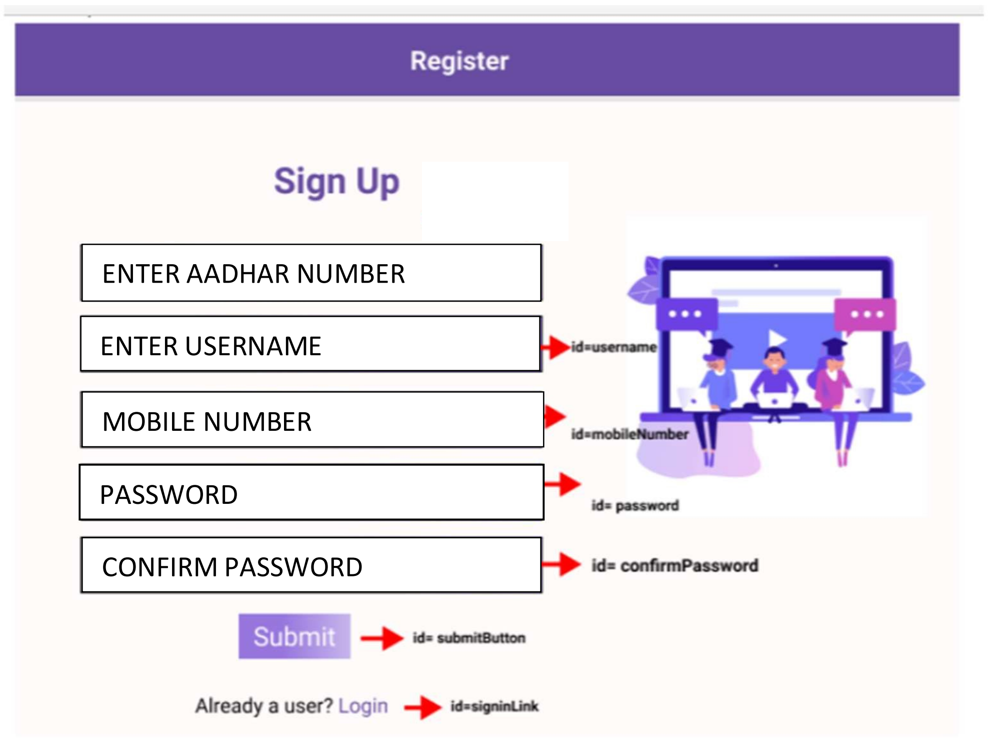
1. Auth: Design an auth component (Name the component as auth for angular app whereas *Auth* for react app. Once the component is created in react app, name the jsx file as same as component name i.e Auth.jsx file) where the customer can authenticate login and signup credentials

2. Signup: Design a signup page component (Name the component as *signup* for angular app whereas *Signup* for react app. Once the component is created in react app, name the jsx file as same as component name i.e Signup.jsx file) where the new customer has options to sign up by providing their basic details.

A. Ids:
   - Aadhar number
   - mobilenumber
   - password
   - confirmpassword
   - submitButton
   - signinLink
   - signupBox

B. API endpoint Url: http://localhost:8000/signup

C. Output screenshot:

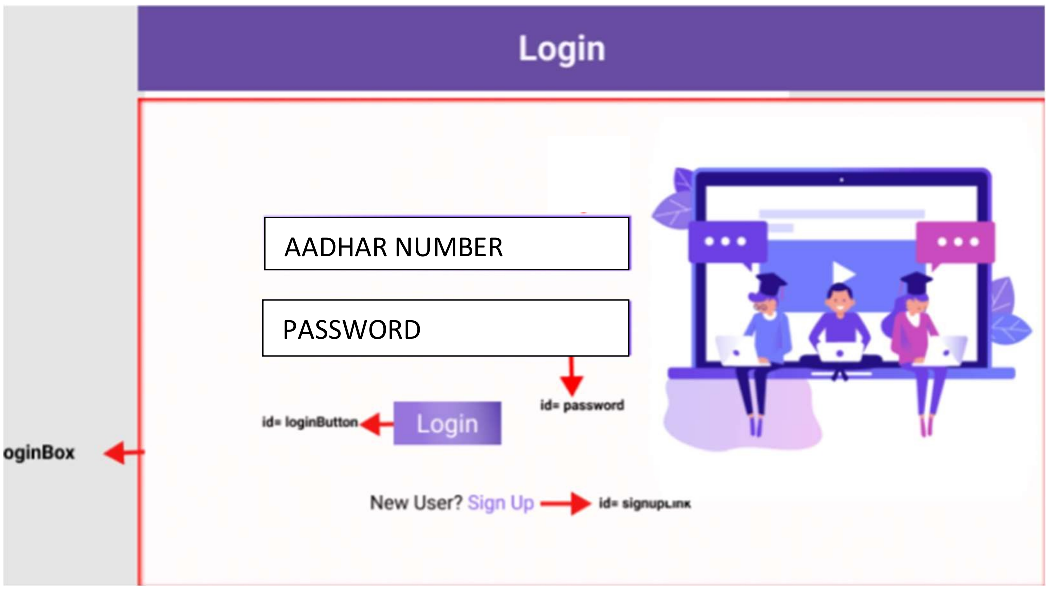


3. Login: Design a login page component named (Name the component as login for angular app whereas Login for react app. Once the component is created in react app, name the jsx file as same as component name i.e Login.jsx file)where the existing customer can log in using the registered email id and password.

A. Ids:
   - AADHAR NUMBER
   - PASSWORD
   - SUBMIT
   - SIGNUPLINK
   - LOGIN BOX

B. API endpoint Url: http://localhost:8000/login

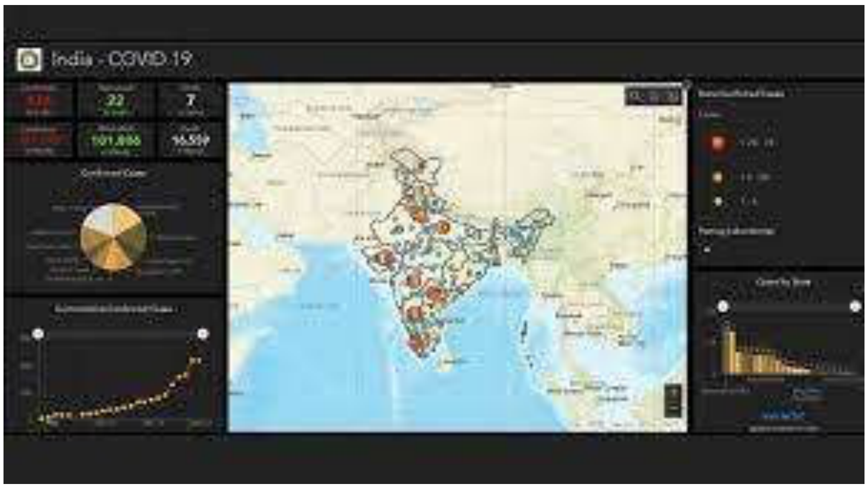
C. Output screenshot:



4. Dashboard / Home: Design a home page component named (Name the component as homepage for angular app whereas HomePage for react app. Once the component is created in react app, name the jsx file as same as component name i.e HomePage.jsx file) that has the navigation bar

- Ids:
  - ➢ userNavbar
  - ➢ HomeButton
  - ➢ Personnel data
  - ➢ Over all State data
  - ➢ logoutButton
  - ➢ API endpoint Url: http://localhost:8000/home

Screenshot



Admin:

6. Admin Dashboard: Design a dashboard page named (Number of affected as dashboard for angular app whereas Dashboard for react app. Once the numbers created in react app, name the jsx file as same as component name i.e Dashboard.jsx file) where the number of affected persons is displayed on the admin side.

a. Admin Navigation: Design a navigation component (Name the component as adminhomepage for angular app whereas AdminHomePage for react app.

i. Ids:

1. adminNavbar
2. adminaddtButton
3. adminconfirmButton
4. logoutButton

b. Add number of patients affected: Design an add product component (Name the component as addpatient for angular app whereas AddPatient for react app.

1.addnumber of affected count

2.StateName

3.District Name

4Aadhar Number

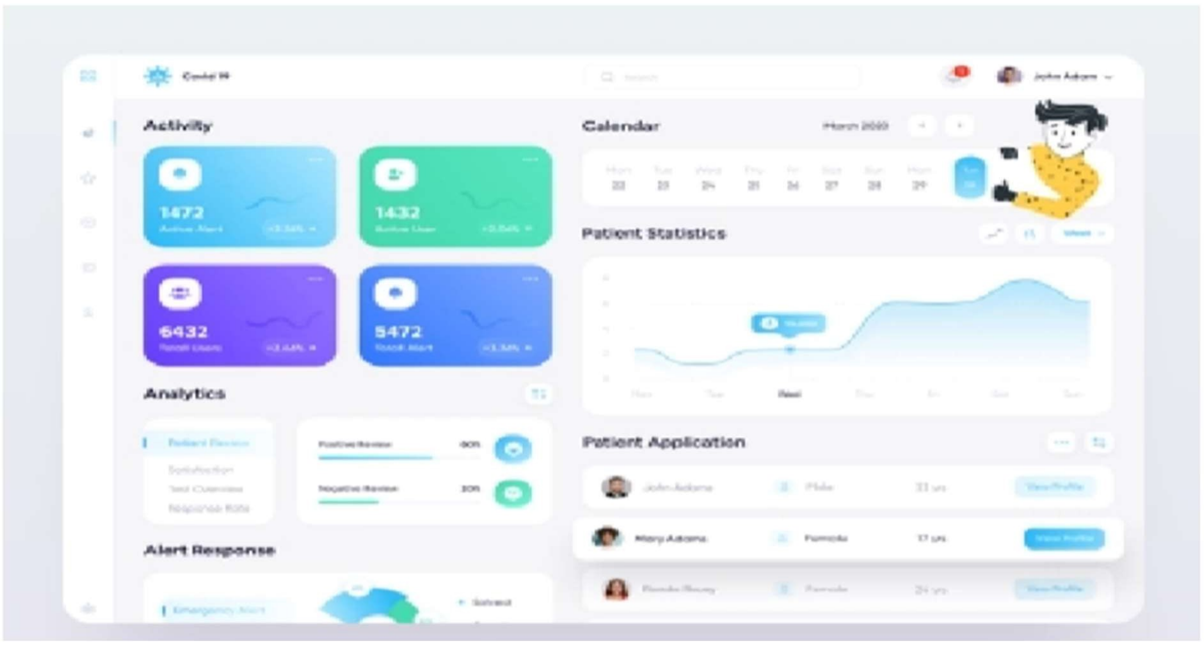
5.affected

6.Recovered

7.adddataButton

ii.API endpoint Url: http://localhost:8000/addProduct

Screenshot:



Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the customer) and all user information.

a. Attributes:

    i. Aadhar: String

    ii. password: String

    iii. mobileNumber: String

    iv. active: Boolean

    v. role: String

2. LoginModel: This class contains the email and password of the user.

a. Attributes:

    i. Aadhar: String

    ii. password: String

3. Covid 19 Model: This class stores the details of the patient.

a. Attributes:

i. StateId: String

ii. imageUrl: String

iii. patient name: String

iv. Status: String

Controller Layer:

6. SignupController: This class control the user signup

a. Methods:

i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.

7. LoginController: This class controls the user login.

a. Methods:

i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false.

8. Patient Controller: This class controls the add/edit/update/view number of person affected by Covid 19.

a. Methods:

i. List&lt;state&gt; getstate(): This method helps the admin to fetch all datas from the database.

ii. List&lt;District&gt; getDistric (): This method helps to retrieve all the datas from the database.

iii. Patient Details FETCHDATA(String id): This function helps to retrieve data related to the affected patients from the database based on the Aadhar number as their primary key.

iv. Patient Details Edit(Patient Details data): This function helps to edit Patient Details

v. Patient Details create(Patient Details data): This method helps to add a new Patient Details to the database.

vi. Patient Details Update (String id): This function helps to update the current status of the patientin the database.