Objective:

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

Users of the System:

1. Government (admin)

2. Public (Users)

Functional Requirements:

- 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.

  Output/Post Condition:
- Records Persisted to COVID19 related information
- Standalone application / Deployed in an app Container

  Non-Functional Requirements:
  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. Performance

    Peak load Performance (during the times when the covid cases rises)

    > ➤ Surfing by users -<3sec
    > ➤ Admin application (Government) <2 sec

    Non-Peak Load Performance

    > ➤ Surfing by users < 2 Sec
    > ➤ Admin Application < 2 Sec

3. Availability

    > ➤ 99.98% Availability

4. Standard Features

    > ➤ Number of tested
    > ➤ Confirmed cases
    > ➤ Deaths in the country
    > ➤ A heatmap of the largest concentrations of confirmed covid-19 cases
    > ➤ Locations of public testing centres in each state
    > ➤ Awareness from major health organization
    > ➤ Direct updates from the government about the covid cases active in the country

5. Logging & Auditing

    > ➤ The system should support logging(app/web/DB) & auditing at all levels

6. Cloud

    > ➤ The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure

7. Browser Compatible

    > ➤ All latest browsers

8. Technology Stack

    > ➤ HTML&CSS
    > ➤ JavaScript
    > ➤ Database Used – MySQL or ORACLE

Key points to remember:
> ➤ The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
> ➤ Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
> ➤ Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types. Adhere strictly to the endpoints given below.

Application assumptions:
1. The login page should be the first page rendered when the application loads.
2. Manual routing should be restricted by using AuthGuard by implementing the can Activate interface. For example, if the user enters as http://localhost:8000/signup or http://localhost:8000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:
- Basic mobile validation through an OTP should be performed.
- Aadhar and Password validations should be performed by sending an OTP to the registered mobile number

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
  - For apps that collect information in the foreground or use foreground service
  - For apps that collect information when running as a background service
- API requirements
- Editorial and quality requirements
- App review and visibility

Covid19 Heat Map
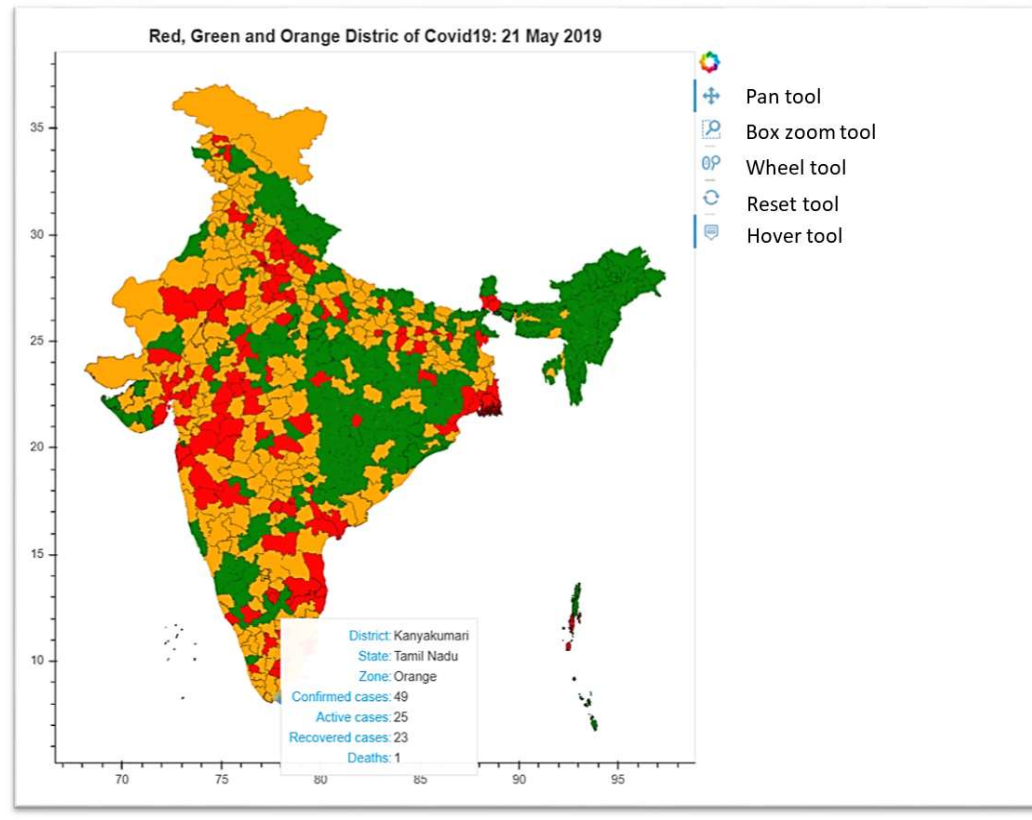
Frontend:

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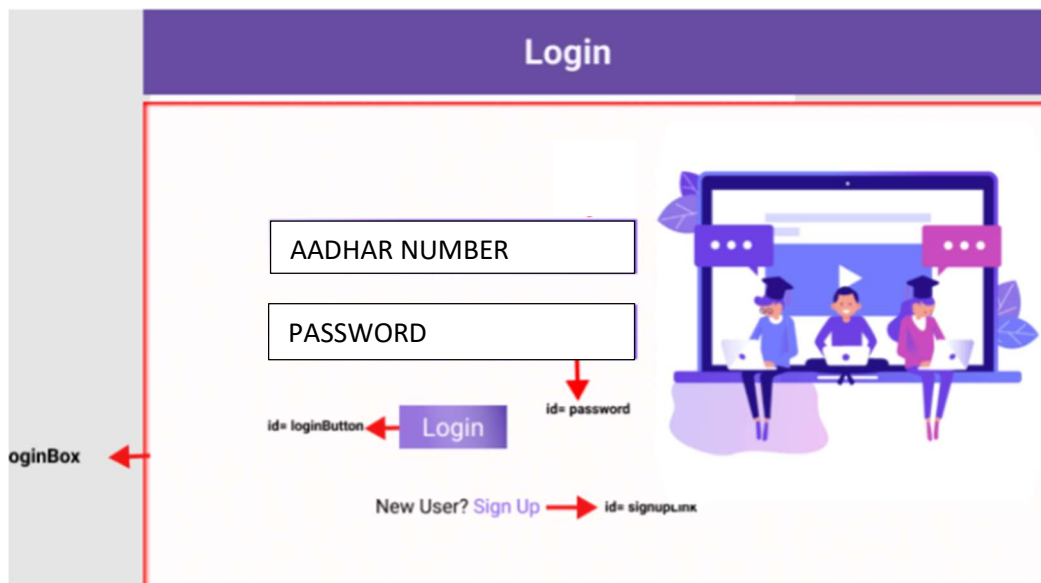
4. Dashboard / Home: Design a home page component named (Name the component as homepage for angular app whereas HomePage for react app. Once the component is created in react app, name the jsx file as same as component name i.e HomePage.jsx file) that has the navigation bar

- 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

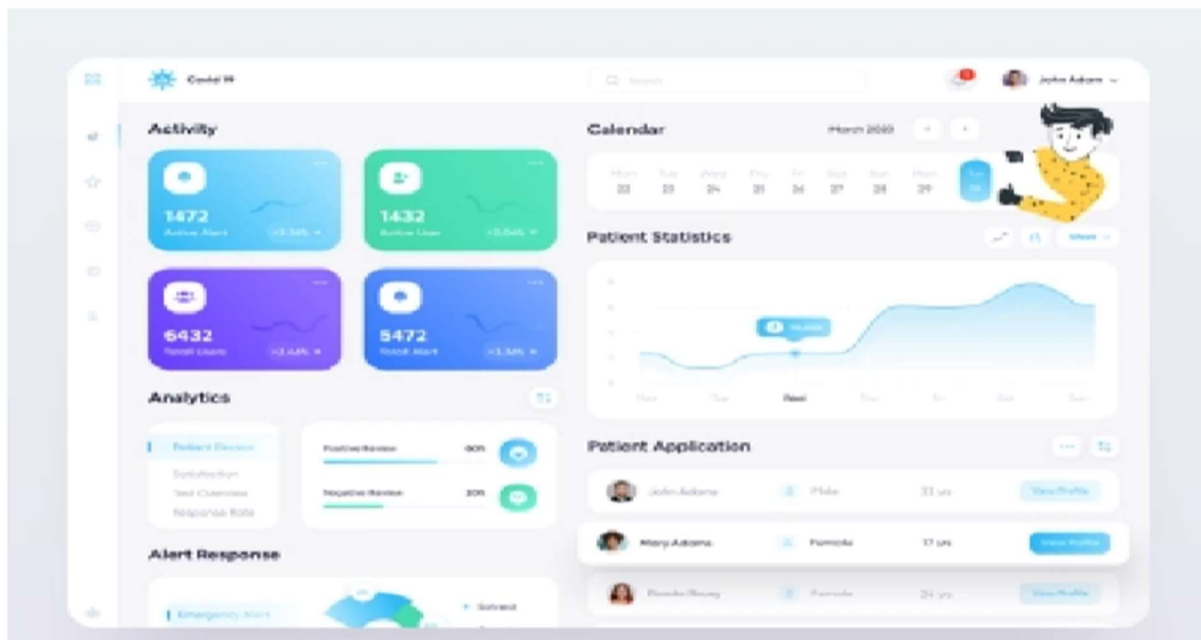2.StateName

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<state> getstate(): This method helps the admin to fetch all datas from the database.

    ii. List<District> getDistric (): This method helps to retrieve all the datas from the database.

    iii. Patient Details EditData(String id): This method helps to retrieve a affected details from the database based on the Aadhar id.

    iv. Patient Details Edit Save(Patient Details data): This method helps to edit a Patient Details and save it to the database.

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

    vi. Patient Details Delete (String id): This method helps to delete a Patient Details from the database.