COVID-19 Daily Report – Webpage

By

Revanth J.R.

Hare Ram

Rohit surya k.s.

Rithish kumar

Objective:

Building an UI/UX for Covid19 Open API tracker site to know the daily report.

Users of the System:

1. Government

2. Public

Functional Requirements:

i) An application that Public can access and view the COVID19 status and previous records.

ii) Maintain a user's Profile & medical records

Page Contents:

1. **Contents of daily report:**

   1. State Name

   2. Confirmed

   3. Deaths

   4. Fatality Rate

   5. Latitude

   6. Longitude

   7. Last Update

2. **Page contents:**
   - Option to Filter Low to High affected areas
   - A database for individual information, public information and COVID19 portfolio.
   - Aadhar integration for intimating individual reports to the public.

3. **Extra contents links to be provided in the page:**
   - Apps storing an individual's vaccination or test records
   - Apps that provide medical, treatment, vaccine, testing, or other related information for COVID-19.

- Apps that support COVID-19-related response, containment, research, or education/training efforts.
- Apps that support services used to respond specifically to COVID-19, for example, apps that provide social support (food stamps, payment), healthcare, loans, etc., specifically in response to COVID-19.
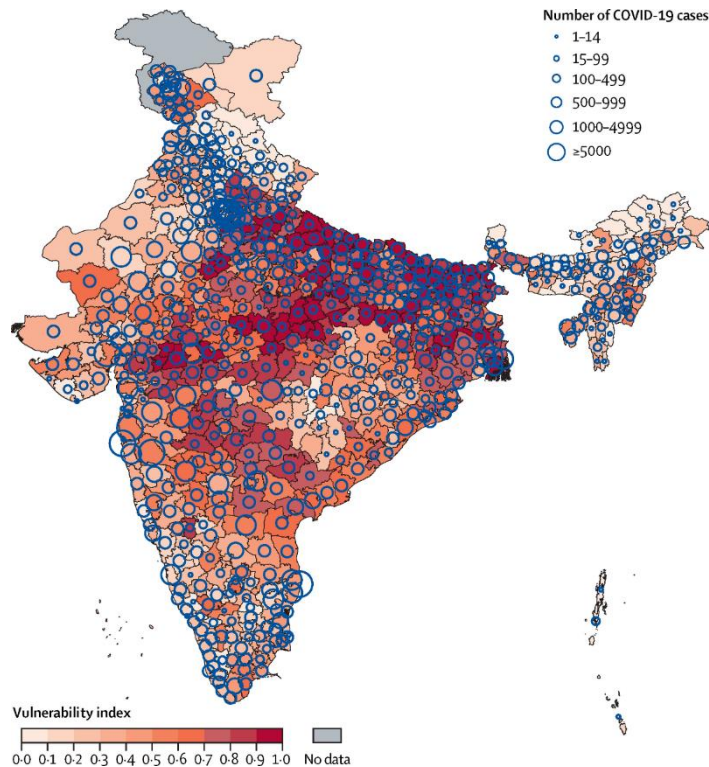
Output/ Post Condition:

- Records Persisted in Success & Failure Collections
- Standalone application / Deployed in an app Container

Non-Functional Requirements:

- Security
- Apps must have a publicly accessible privacy policy that comprehensively discloses the access, collection, use and sharing of personal and sensitive user data.
- Apps created specifically for the COVID-19 response may not access personal and sensitive data that is not required to directly support the public health emergency, and may only use the data collected to support COVID-19-related efforts or epidemiological research
- You must disclose the use of COVID-19 related data in user-facing privacy disclosures (e.g., privacy policy and/or in-app disclosures) for your app(s).
- Apps must handle all personal or sensitive user data securely, including transmitting it using modern cryptography (for example, over HTTPS).

Standard Features:

1. Number of tested
2. Confirmed cases
3. Deaths in the country
4. A heatmap of the largest concentrations of confirmed covid-19 cases

Number of COVID-19 cases
- 1–14
- 15–99
- 100–499
- 500–999
- 1000–4999
- ≥5000

Vulnerability index
0·0 0·1 0·2 0·3 0·4 0·5 0·6 0·7 0·8 0·9 1·0 No data

5. Locations of public testing centers in each state
6. News updates from major health org's
7. Logging:
   - The system should support logging(app/web/DB) at all levels
8. Cloud:
   - The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
9. Browser Compatible:
   - All latest browsers
10. Technology Stack:
    - HTML&CSS
    - JavaScript

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 canActivate 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.

Validations:

- Basic mobile validation should be performed.

- Aadhar Password validations should be performed

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
- submitButton
- signupLink
- loginBox

    b. **API endpoint Url: http://localhost:8000/login**

    c. **Output screenshot**

4. Dashboard / Home: Design a home page component named (Name the component as homepagefor 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
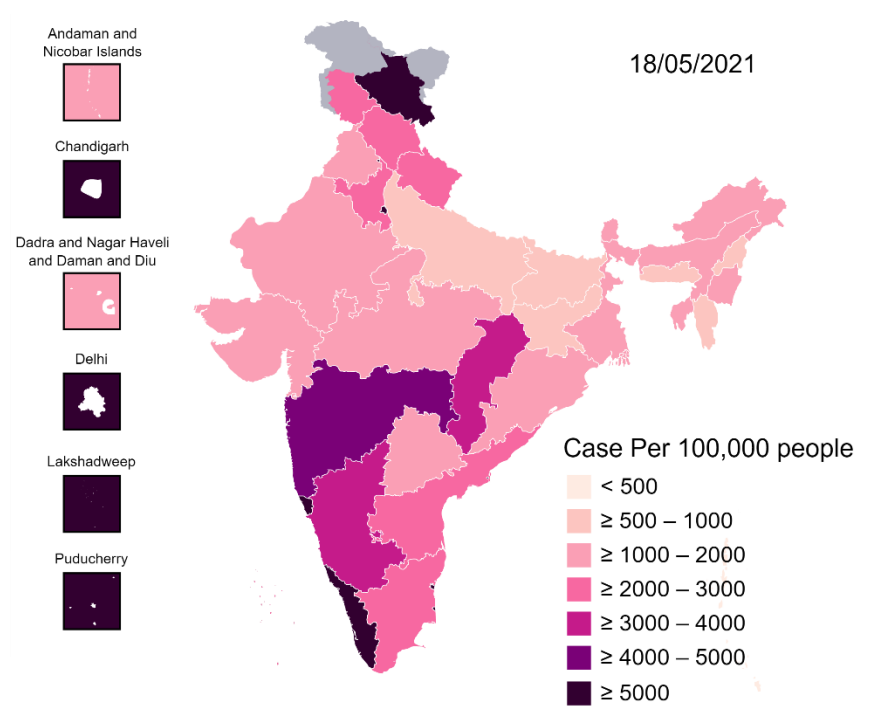
a. Ids:

1. userNavbar

2. HomeButton

3. Personnel data

4. Over all State data

5. logoutButton

b. API endpoint Url:

http://localhost:8000/home

c. 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
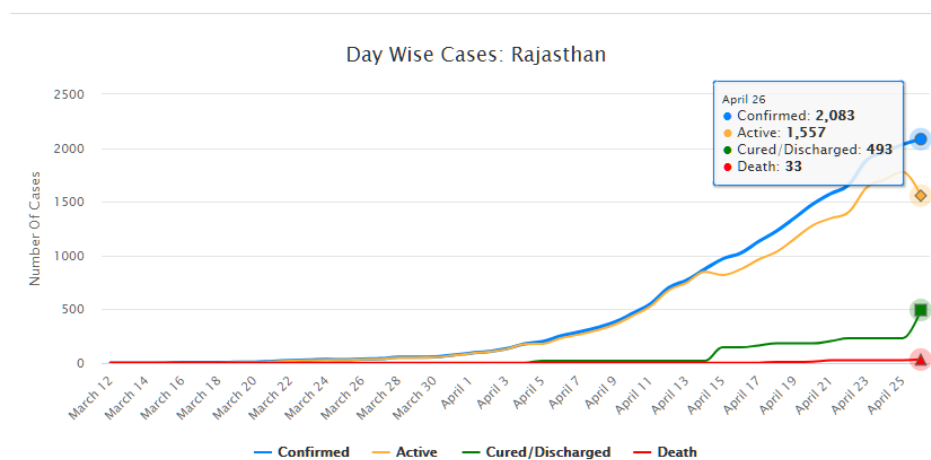
3.District Name

4.Aadhar Number

5.affected

6.Recovered

7.adddataButton

ii.API endpoint Url:

http://localhost:8000/addProduct

c.Screenshot

Day Wise Cases: Rajasthan

April 26
● Confirmed: **2,083**
● Active: **1,557**
● Cured/Discharged: **493**
● Death: **33**

— Confirmed   — Active   — Cured/Discharged   — Death

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.