COVID RADAR

SRS DOCUMENT

Objective:

To examine the impact of COVID-19.

Users of the System:

1. Admin
2. Customer

Functional Requirements:

* Additional information of Covid-19 is given in read mode.
* Each side navigation icon has its own specific contents.
* Vaccinated and unvaccinated are also mention with the percentage.
* Vaccine 1st and 2nd dose are vaccinated by 54.9% of the people in India currently.
* Affected people of Covid-19 are differentiated by confirmed, active, deceased, recovered.
* Total no. of people vaccinated doses are also administered.
* The tabular column referred here is mentioned the no. of people affected by Covid-19 are differentiated state-wise and differentiated by confirmed, active, deceased, recovered.
* India map mentioned here indicates the states with coloured effects, mentions the Covid affected and delivers contents of

state.

While the above ones are the basic functional features expected, the below ones can

be nice to Have add-on features:

* Helpline number
* Household things
* Nearby Govt. hospital
* Oxygen supply
* municipality
* medicinal supply
* reporting side effects
* Home remedies for Covid-19
* District-wise vaccination camp
* languages(english,tamil,hindi)
* vaccination certificate linkage
* chatbox and faq
* statistics
* online doctor appointment

Non-Functional Requirements:

Security

* App Platform –User Name/Password-Based Credentials
* Sensitive data has to be categorized and stored in a secure manner
* Secure connection for transmission of any data

Performance

* Peak Load Performance (during Festival days, National holidays etc)
* eCommerce
* Admin application &lt;
* Non-Peak Load Performance
* eCommerce &lt;
* Admin Application &lt;

Availability

99.99 % Availability

Standard Features

* Scalability
* Maintainability
* Usability
* Availability
* Failover

Logging &amp; Auditing

* The system should support logging(app/web/DB) &amp; auditing at all levels

Monitoring

* Should be able to monitor via as-is enterprise monitoring tools

Cloud

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

Browser Compatible

All latest browsers

Output/ Post Condition:

* Records Persisted in Success
* Standalone website / Deployed in an website Container

Technology Stack

Front End : css, html, java script

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be

modified at any cost. Failing to do so may fail test cases.

2. Remember to check the screenshots provided with the SRS. Strictly adhere to id

mapping and attribute mapping. Failing to do may fail test cases.

3. 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 website loads.

2. . Unless logged into the system, the user cannot navigate to any other pages.

3. Logging out must again redirect to the login page.

4. To navigate to the admin side, you can store a user type as admin in the

database with a username and password as admin.

5. Use admin/admin as the username and password to navigate to the admin

dashboard.

Validations:

1. Basic email validation should be performed.

Password validations should be performed

Project Tasks:

API Endpoints:

USER

Action URL Method Response

Login /login POST true/false

Signup /signup POST true/false

Get All features – Home /home GET Array of features

ADMIN

Action URL Method Response

Get All features /admin GET Array of features

Add features /admin/add features POST features added

Delete features /admin/delete/{id} GET features deleted

features Edit /admin/featureEdit/{id} GET Get All details of Particular id

feature Edit /admin/featureEdit/{id} POST Save the Changes

Frontend:

Customer:

1. Auth: Design an auth component (Name the component as auth for angular website

whereas Auth for react website. Once the component is created in react website, 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 sign up for

angular website whereas Signup for react website. Once the component is created in react

website, 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:

* email
* username
* mobile number
* password
* confirm password
* 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 website whereas Login for react app. Once the component is created in react

website, 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:

* email
* 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 homepage for angular website whereas HomePage for react website. Once the

component is created in react website, name the jsx file as same as component name

grid elements with appropriate filter options.

a. Ids:

1. userNavbar

2. helpline numbers button

3. Medicinal supply

4. reporting side effects button

5. languages button

6. chatbox and faq

7.online doctor appointment button

8.logout button

a. API endpoint Url: http://localhost:8000/home

b. Screenshot

Admin:

6. Admin Dashboard: Design a dashboard page named (Name the

component

as dashboard for angular website whereas Dashboard for react website. Once the

component is created in react website, name the jsx file as same as component name i.e

Dashboard.jsx file) where the list of features is displayed on the admin side.

a. Admin Navigation: Design a navigation component (Name the component

as admin homepage for angular app whereas AdminHomePage for react website. Once

the component is created in react website, name the jsx file as same as component

name i.e AdminHomePage.jsx file) that can navigate to features of a website.

i.Ids:

1. adminNavbar

2. adminfeatureButton

3. adminconfirmButton

4. logoutButton

b. Add feature: Design an add feature component (Name the component

as add feature for angular website whereas Add Feature for react website. Once the

component is created in react website, name the jsx file as same as component name i.e

Addfeature.jsx file) in which the admin can add new products to the inventory.

i.Ids:

1.featureName

2.featureDescription

3.featureImageURL

4.addfeatureButton

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

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. email: String

ii. password: String

iii. username: String

iv. mobileNumber: String

v. active: Boolean

vi. role: String

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

a. Attributes:

i. email: String

ii. password: String

3. featureModel: This class stores the details of the product.

a. Attributes:

i. featureId: String

ii. imageUrl: String

iii. featureName: String

iv. price: String

v. description: String

vi. quantity: String

4.Controller Layer:

a.SignupController: This class control the user signup

b. Methods:

i. saveUser(UserModel user): This method helps to store users in the

database and return true or false based on the database transaction.

5. LoginController: This class controls the user login.

a. Methods:

i. checkUser(LoginModel data): This method helps the user to sign up for

the application must return true or false.

8. featureController: This class controls the add/edit/update/view feature

a. Methods:

i. List<featureModel> getfeature(): This method helps the admin to fetch

all features from the database.

ii. List<featureModel> getHomefeature(): This method helps to retrieve

all the features from the database.

iii. featureModel productEditData(String id): This method helps to retrieve

a feature from the database based on the productid.

iv. featureEditSave(ProductModel data): This method helps to edit a

feature and save it to the database.

v. featureSave(featureModel data): This method helps to add a new

feature to the database.

vi. featureDelete (String id): This method helps to delete a feature from

the database.