PROJECT REPORT - CONTAINMENT ZONE ALERTING APPLICATION

TEAM ID - **PNT2022TMID35281**

DATE - 26/11/2022

TEAM MEMBERS

NO.	**ROLL NO.**	**NAME**
1.	2019103545	**NANDA KUMAR R**
2.	2019103536	**KAUSHIK NARAYAN R**
3.	2019103068	**SUDHARSAN K**
4.	2019103029	**KIRAN GEORGE GAURDIAN**

ABSTRACT

The World Health Organization has declared the outbreak of the novel coronavirus, Covid-19 as pandemic across the world. With its alarming surge of affected cases throughout the world, lockdown, and awareness (social distancing, use of masks etc.) among people are found to be the only means for restricting the community transmission.

In a densely populated country like India, it is very difficult to prevent the community transmission even during lockdown without social awareness and precautionary measures taken by the people.

This project mainly focuses on development of an Android application which can inform people of the Covid-19 containment zones and prevent trespassing into these zones. This Android application updates the locations of the areas in a Google map which are identified to be the containment zones. The application also notifies the users if they have entered a containment zone.

To implement this application, multiple APIs such as the Sendgrid API for email notifications, Google Maps API for location tracking, and also Google's geofencing client for creating geofences on the map for containment zones are used.

Table of Contents

1. INTRODUCTION

1.1. Project Overview:

To stop the country from seeing an increase in Covid-19 instances, numerous research projects are now being conducted. In the past, our nation had to import medical supplies and masks, but it has recently been successful in producing these supplies. Our nation has taken steps to raise awareness of the disease as well as initiatives to combat it. By educating the public about the preventive steps that can shield them against infection, the news and media play an important role in fostering this knowledge. Increasing public awareness of the need to take all necessary precautions can significantly aid in limiting virus spread. To stop the virus from spreading

further, the nation has established containment zones throughout the cities where Covid-19 cases have been reported. To prevent contamination outside, these containment areas have been maintained closed off from the general public. The government has loosened several lockdown regulations after more than two months and allowed the reopening of government buildings, bus and other road transit facilities, and shopping centers.

People can travel around the city for several reasons, including work. However, the containment zones are still maintained in isolation, and new containment zones are being established in all areas where Covid-19 cases have been reported. These areas are extremely contagious because virus droplets spat out by an unscreened, asymptomatic patient can travel great distances. Despite the fact that police officers are stationed in these containment zones, it is still possible for anyone to wander inside without realizing it. These containment zones present an infectious danger to city residents in this scenario where people freely roam about the metropolis. Because of this, letting people know where the containment zones are can help them get around and avoid them, lowering the likelihood of community transmission.

In this project, we focus on developing a mobile based application to provide information regarding the Covid-19 containment zones. The application further tracks the user’s location and provides notification alert if the user has entered a containment zone. The application also provides daily Covid-19 case statistics to the users to keep them updated. The application is developed using React Native and Flask, and connects to a DB2 database provided by IBM Cloud. Google's geofencing client is used to create geofences around the containment zones and notification manager is used to provide notifications. We have tested our application with different users in different locations and it works efficiently and is able to attain our target.

1.2. Project Purpose:

The purpose of this project is to monitor the locations of the users through the app by using

Google Maps API and notify them of surrounding containment zones, and also to warn them if they are approaching or are inside said containment zonesrom the admin side, the admin can create new geofences(containment zones) and/or update said containment zones on the web application.

2. LITERATURE SURVEY

COVID-19 is an infectious disease caused by the SARS-CoV-2 virus. The novel virus was first identified from an outbreak in Wuhan, China, in December 2019. Since its emergence, it has led to the deaths of more than 6 million people and more than 600 million reported cases worldwide. It is mainly transmitted as airborne droplets or from the transfer of the virus through contact with an infected person or object.

The World Health Organisation (WHO) has declared the outbreak of Covid-19 as a pandemic and recommends maintaining physical contact of at least 1 metre from others, avoiding crowded areas and close contact with infected people. Hence, it would be beneficial to develop an application that can alert people when they enter a quarantined zone or have been in close proximity with someone that tested positive for COVID-19 or any other infectious disease.

When it comes to existing solutions, Aarogya Setu is the most widely used application in India. It is a mobile application developed by the Government of India to connect essential health services with the people. Its key features include automatic contact tracing using Bluetooth, risk status of users, geo-location based COVID-19 statistics and much more. It makes use of Bluetooth and GPS technology to keep track of the movement of an infected person and alerts citizens about areas that person has been to and marks them as vulnerable spots. In the absence of GPS technology, it uses cellular triangulation to estimate the location of a person.

While Aarogya Setu can perform contact tracing and notify people that have been exposed to someone infected with COVID-19, it does actively maintain a list of quarantine or containment zones and warn users that enter these zones. Many applications have been developed in the last two years to tackle COVID-19. Most of these apps deal with broadcasting COVID statistics, precautionary measures against COVID and ensuring that people get the healthcare they need. But there aren't many applications that are capable of identifying containment zones and alerting people in real-time. The difficulty of acquiring the data that is both accurate and real-time is one of the main challenges in creating such an application. A solution to this might be to obtain or make use of containment zone data from the state governments.

Ranajoy Mallik et al. 2020 have developed an android application that performs exactly this. It uses Google APIs like Firebase and Geofencing to achieve this. It is capable of updating locations of containment zones in Google maps and notifies people that are trespassing into it. They have created a real-time database in Cloud Firestore which contains all the data related to the containment zones like coordinates, radius and zone names. The android application can retrieve information from the database.
As the app receives the location data, geofences are created and displayed using Google maps in the application. The user's location is also shown and updated constantly in the map. The main disadvantage of the method proposed by Ranajoy Mallik et al. 2020 is the origin of the

containment zone data. The data that they've used is dependent on the State government regularly updating containment zone data.

If the government decides to stop doing so or does not release it for public use, it risks the app displaying inaccurate information or even no information. One solution to this problem might be to make use of data from multiple sources. Allowing people to update the map with newly infected persons might be one stream of data.

MoveInSync, a software development company based out of Bangalore has also developed a COVID containment zone tracker. It is mainly a dashboard that can tell users if a particular locality lies in a containment zone. It can display containment zone information for 15 cities. The containment zone boundary data was maintained with the help of geoIQ, a geolocation data specialist. The data is crowdsourced and updated every hour.
For the tracker, it utilized Amazon Aurora PostgreSQL, Cloudwatch, Micrometer + Prometheus and Raygun Crash Monitoring to help scaling up.

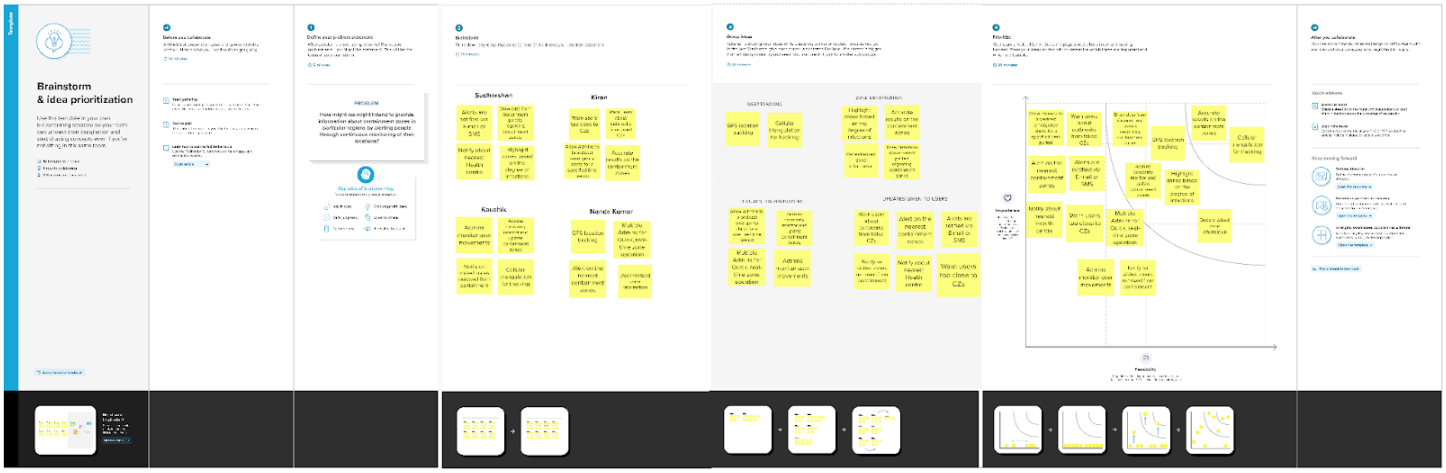
However, one disadvantage is that geoIQ is proprietary, and comes with its own set of limitations. Akira Suyama et al. 2016 proposed a disaster information system using geofencing technology to detect movement of users and inform them of any risks that they may face. This method can be repurposed for the containment zone application since the underlying use-case is the same. This method makes use of a client-server architecture.

The server collects risk information from various sources and the client monitors the user to notify them of necessary information. Geofencing is a mechanism that makes a virtual fence in a specific area. The application sets a geofence at a dangerous area and gives risk information to the user. In this proposed method, geofencing was implemented using the Core Location framework of iOS which provides a detection of the entries and exits of the user within a

specific geographic region. However, the Core Location framework is available only for iOS devices.

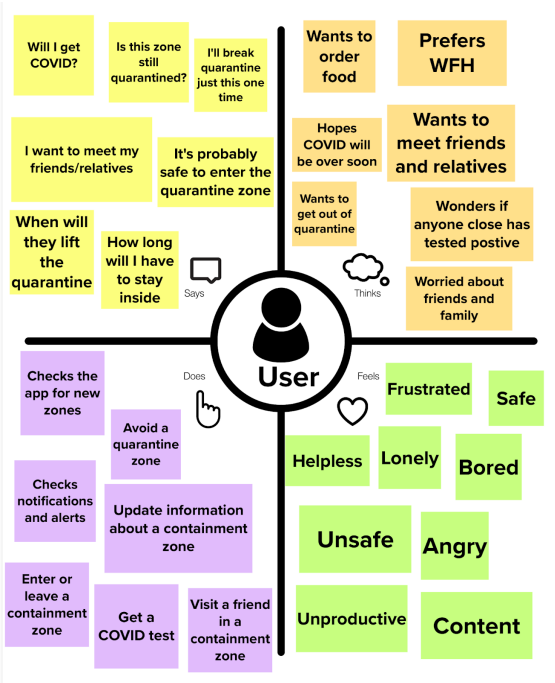
3. IDEATION AND BRAINSTORMING

3.1. Brainstorming process:



3.2. Empathy Map

An empathy map is used to gain insight into a user's perspective.



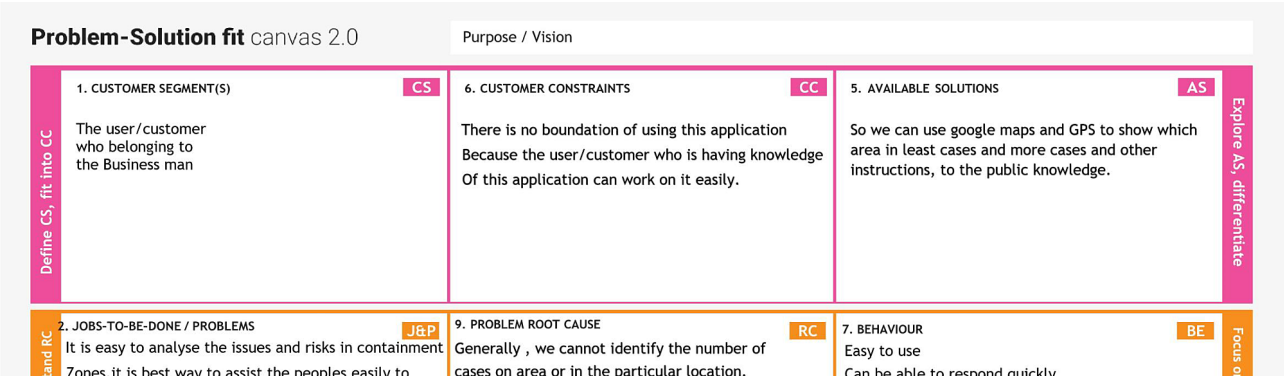
3.3. Proposed Solution:

S.NO.	Paramater	Description
1.	Problem Statement (Problem to be solved)	To Provide information about the containment zones in a particular region by alerting the people trespassing the region through continuous monitoring of their location.
2.	Idea / Solution description	The project aims at building an application that provides information about the containment zones of a particular region by continuously monitoring an individual's location. Location of the individual must be stored in the Database. Alerts are sent using the notification service.
3.	Novelty / Uniqueness	The uniqueness of containment zone alerting app is it shows the particular area of the district before the 100

		meter, and the user's location history is stored in database and this app provides the precautions measurements, list of immunity boosters, location of the vaccination providing places it also gives the list of the affected and admitted patients and discharged patients , percentage of affecting by covid19
4.	Social Impact / Customer Satisfaction	Social Stigma is discrimination against a particular group of people, a place, or a nation in the form of a negative attitude. Public health emergencies are stressful situations for people and communities. Fear and anxiety with a lack of knowledge about the disease can leads to social stigma. The containment zone alerting app users are 100% satisfied because of its immediate notification of a particular area, it provides the

		precautions and awareness about COVID-19.
5.	Business Model (Revenue Model)	When User enters some other region which is not the user's home region, user has to subscribe in order to view the containment zones in the new region, in addition, subscribing to personal health tracker allows the user to manage his health efficiently.
6.	Scalability of the Solution	In this modern world even though the covid pandemic threat is about to end there are high chance of pandemic or endemic. So, this application is very useful in that situation and we can use this application in seasonal diseases

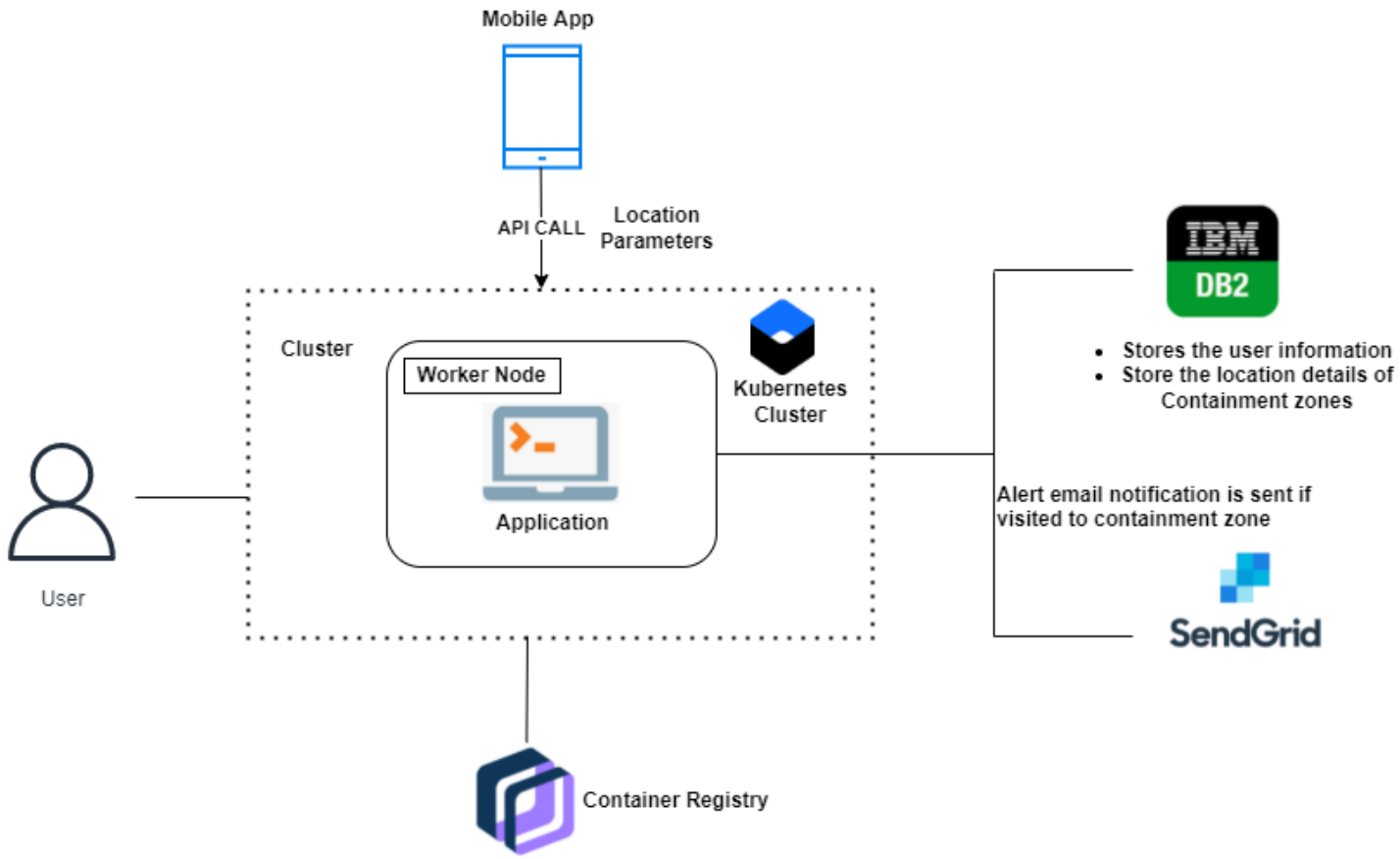
3.4. Problem Solution Fit:



3.5. Solution Architecture:

Solution architecture is the process of developing solutions based on predefined processes, guidelines and best practices with the objective that the developed solution fits within the enterprise architecture in terms of information architecture, system portfolios, integration requirements and many more.

It can then be viewed as a combination of roles, processes and documentation that are intended to address specific business needs, requirements or problems through the design and development of applications and information systems



4. REQUIREMENTS ANALYSIS

4.1. Functional Requirements:

Requirement ID	Requirement Statement	Sub Requirement
FR001	User Registration and Login	Users can register using their email or phone number
FR002	App permissions	Enabling location access
FR003	User Location Tracking	Track user location using

		Google map API and update their location
FR004	Admin App for Updating Containment Zones	Admin can update the containment zone information
FR005	Containment Zones Display	Containment zones are marked using zone colours
FR006	Alert Notification	An alert notification will be sent to the user if they enter a containment zone

4.2. Non Functional Requirements:

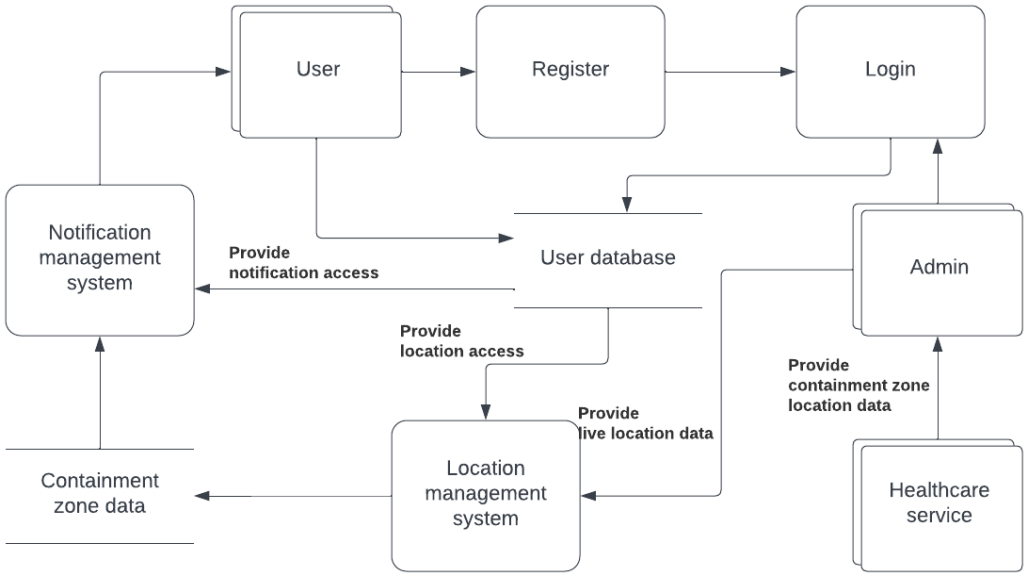
Requirement ID	Requirement Statement	Sub Requirement
NFR001	Usability	User interface must be intuitive and effectively convey information about containment zones
NFR002	Reliability	Users can trust the containment zone information and expect it to be accurate
NFR003	Performance	The app is highly responsive and functions as intended
NFR004	Availability	Anyone from anywhere can

		access it on the Internet
NFR005	Scalability	The application can handle rapid growth of the user base and track increasing number of containment zones without affecting app performance

5. PROJECT DESIGN

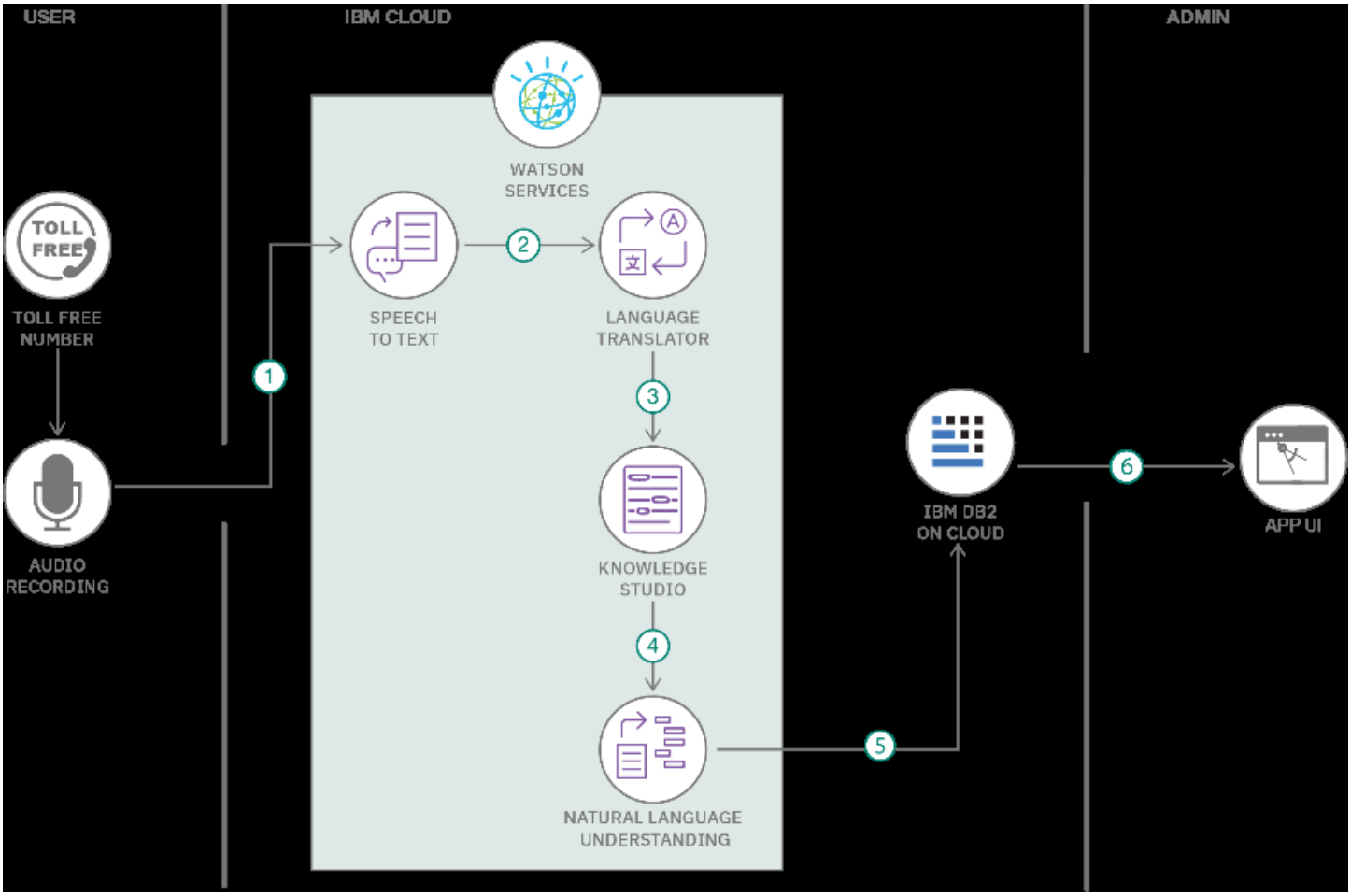
5.1. Data Flow Diagram(DFD):

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.



5.2. Technology Architecture:

Technology architecture associates application components from application architecture with technology components representing software and hardware components.



5.2.1. Table-1: Components and Technologies:

S.NO.	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, etc.	HTML, CSS, JavaScript etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in	IBM Watson STT

		the application	service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Infrastructure(Server/ Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

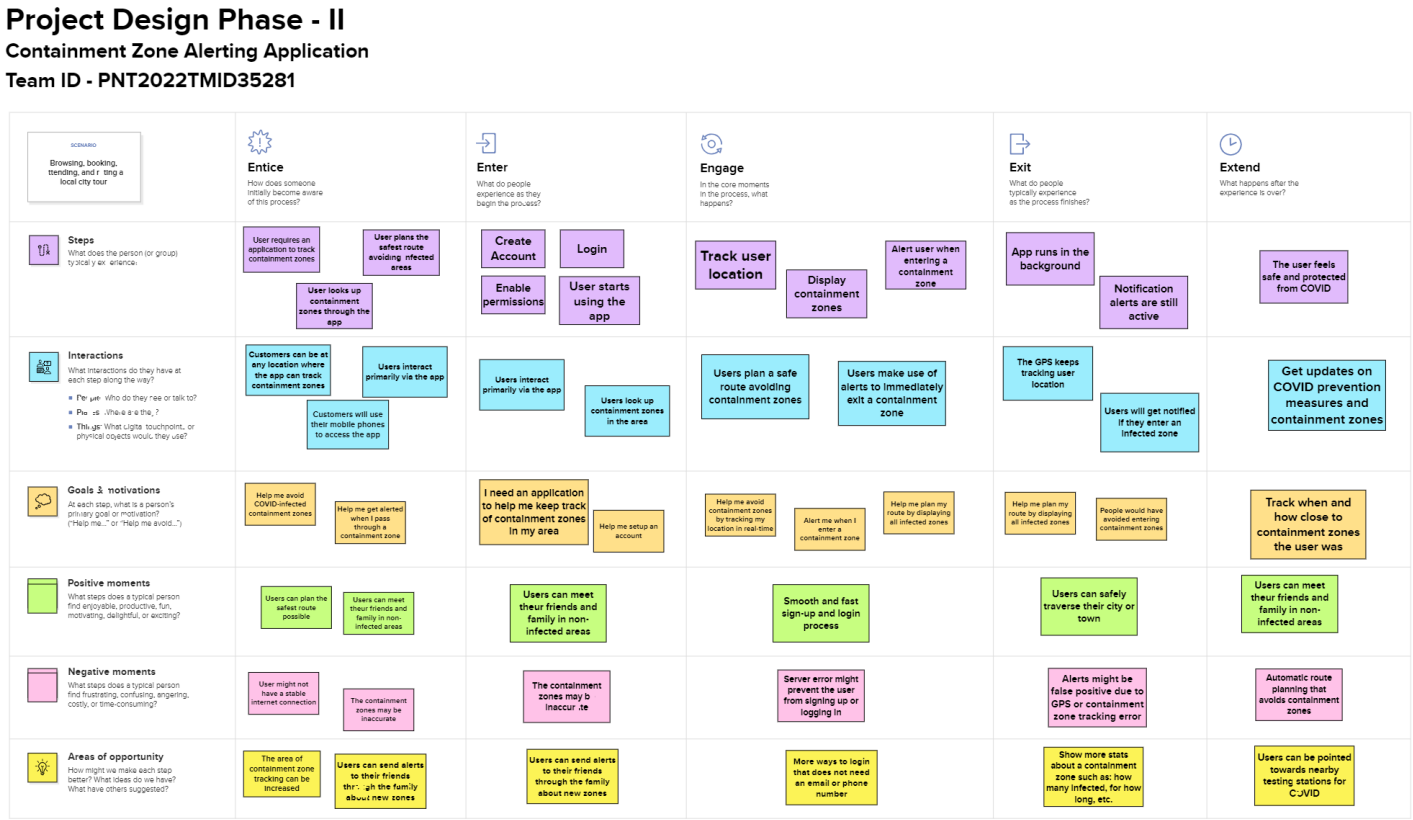
5.2.2. Table-2: Application Characteristics:

S.NO.	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Fire Wall
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Encryptions, IAM Controls, etc.
3.	Scalable Architecture	Justify the scalability of architecture	IBM DB2
4.	Availablity	Justify the scalability of application	Google Map Services
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	IBM Cloud

5.3. Customer Journey Map

Customer Journey Map creation is the process of creating a customer journey map, a visual

story of the customers' interactions with your brand. This exercise helps businesses step into a customer's shoes and see one's business from the customer's perspective. It allows one to gain insights into common customer pain points and how to improve those.



6. PROJECT PLANNING

6.1. Milestones and Activity List

A milestone/activity list is a project management document that identifies all project milestones. A milestone is a significant event or a point in a project. It represents nothing more than a moment in time; hence, when scheduling, milestones should be assigned zero duration.

TITLE	DESCRIPTION	DATE
Literature Survey and Information Gathering	Comprehensive summary of the previous works and research on the topic is done by referring to different research and technical publications.	10 October 2022
Empathy Map Preparation	Prepare an Empathy Map which is used to gain a deeper insight into customers or users by trying to understand how customers think and feel, followed by measurement of user pains and gains.	10 October 2022
Brainstorm and Ideation	Hold a group brainstorming where each member puts forth their ideas, by leveraging the collective thinking of the group, followed by three core ideas to be implemented.	12 October 2022

Proposed Solution	Propose a solution that evaluates multiple parameters of the project such as novelty, business model, scalability, etc.	14 October 2022
Problem Solution Fit	Prepare Problem-Solution Fit document.	16 October 2022
Solution Architecture	Prepare Solution Architecture document.	16 October 2022
Customer Journey	Prepare Customer Journey to understand use cases with the application.	5 November 2022
Functional Requirement	Prepare Functional Requirements Document which may contain functional and non-functional requirements.	6 November 2022
Data Flow Diagrams	Prepare the Data Flow Diagram document which entails the flow of data between different modules of the application.	8 November 2022
Technology Architecture	Prepare Technology Architecture Document.	8 November 2022
Prepare Milestone and Activity List	**Prepare Milestone and Activity List of the project.**	**15 November 2022**
Project Development and	**Complete development of**	**25 November 2022**

Sprint Delivery	**the project and submit according to Sprint Deliveries.**	

6.2. Sprint Delivery Plan

6.2.1. Product Backlog, Sprint Schedule, and Delivery Plan:

Sprint	Functional Requirement	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Registration (Android and Web)	USN-1	User can register in the application by entering email and password	4	High	Kaushik, Sudharshan
	IBM DB2	USN-2	User details have to be stored in the database	4	High	Kaushik, Sudharshan
	SendGrid	USN-3	User will receive confirmation email after registration	4	High	Kiran, NandaKumar
	Login (Android and Web)	USN-4	User can login to the application by entering email and password	4	High	Kaushik, Sudharshan

	Dashboard	USN-5	User can view the dashboard which contains all the information	4	High	Kiran, NandaKumar
Sprint-2	Access	USN-6	Permission needs to be granted in order to access the user location	3	High	Kiran, NandaKumar
	IBM DB2	USN-7	Admin has to collect information regarding COVID-19 cases from verified sources and store in Database	3	High	Kiran, NandaKumar
	IBM DB2	USN-8	Admin needs to update the Containment Zones	4	High	Kaushik, Sudharshan
	IBM DB2	USN-9	Admin needs to differentiate the containment zones on the basis of degree	4	Medium	Kiran, NandaKumar

			of infections			
	SendGrid	USN-10	User has to be alerted through a notification when they enter a containment zone	3	High	Kaushik, Sudharshan
	IBM DB2	USN-11	User travel history details has to be stored in the Database	3	Medium	Kiran, NandaKumar
Sprint-4	Integration	USN-12	Frontend and Backend Integration	5	High	Kaushik, Sudharshan
	Docker	USN-13	Creation of Docker Image	5	High	Kiran, NandaKumar
	Cloud Registry	USN-14	Uploading Docker image to IBM Cloud Registry	5	High	Kaushik, Sudharshan
	Kubernetes	USN-15	Creating docker container and hosting	5	High	Kiran, NandaKumar

6.2.2. Project Tracker, Velocity and Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed	Sprint Release Date (Actual)

					(as on planned end date)	
Sprint-1	20	3 Days	15 Oct 2022	17 Oct 2022	20	15 Oct 2022
Sprint-2	20	3 Days	17 Oct 2022	19 Nov 2022	20	17 Oct 2022
Sprint-3	20	3 Days	20 Nov 2022	22 Nov 2022	20	19 Nov 2022
Sprint-4	20	3 Days	23 Nov 2022	25 Nov 2022	20	25 Nov 2022

Velocity: Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

AV = Sprint Duration / Velocity = 20 / 3 = 6.66

7. CODING AND SOLUTIONING

This section includes source code snippets and explanation of the basic functionalities of the code on both the admin(web app) and user(mobile app) side.

7.1. Feature 1 (Android App)

Functionality for obtaining containment zone location information from the database:

```
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://169.51.204.133:32440/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        ZoneAPI zoneAPI = retrofit.create(ZoneAPI.class);

        Call<ArrayList<ZoneModel>> locations = zoneAPI.getAllZones();
        Log.d("ghhe", "testtestest");
        locations.enqueue(new Callback<ArrayList<ZoneModel>>() {
            @Override
            public void onResponse(Call<ArrayList<ZoneModel>> call, Response<ArrayList<ZoneModel>> response) {
                if (response.isSuccessful()) {
                    for(ZoneModel zone: response.body()) {
                        Log.d("success", zone.getName() + zone.getLatitude() + zone.getLongitude());
                        locations2.add(new LatLng(zone.getLatitude(), zone.getLongitude()));
                        handleAddGeofence(new LatLng(zone.getLatitude(), zone.getLongitude()));
                    }
                }
                else {
                    Log.d("success", "failure");
                }
            }

            @Override
            public void onFailure(Call<ArrayList<ZoneModel>> call, Throwable t) {

            }
        });
    }
```

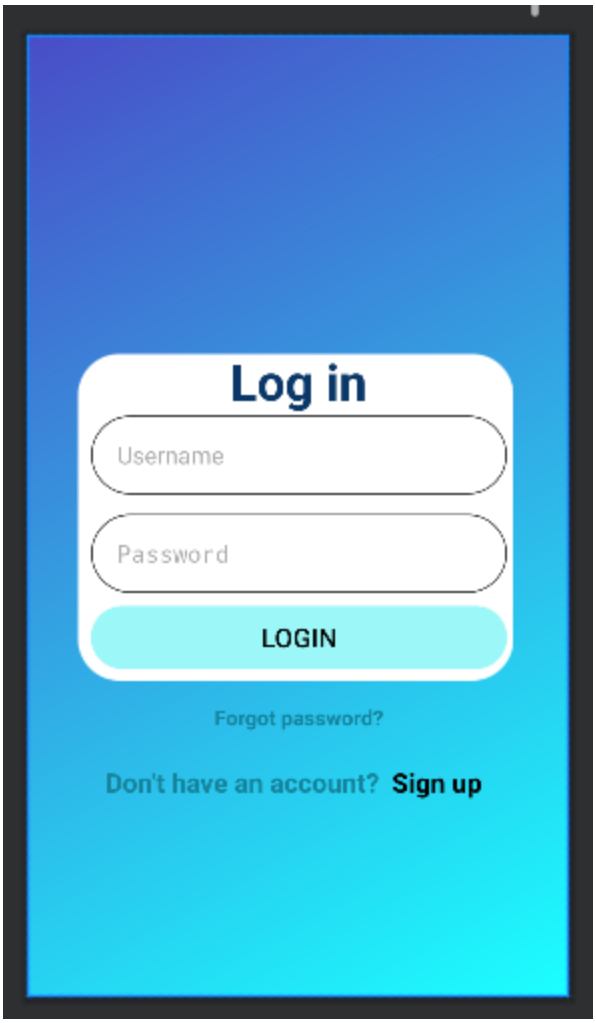
Functionality for adding Geofences:

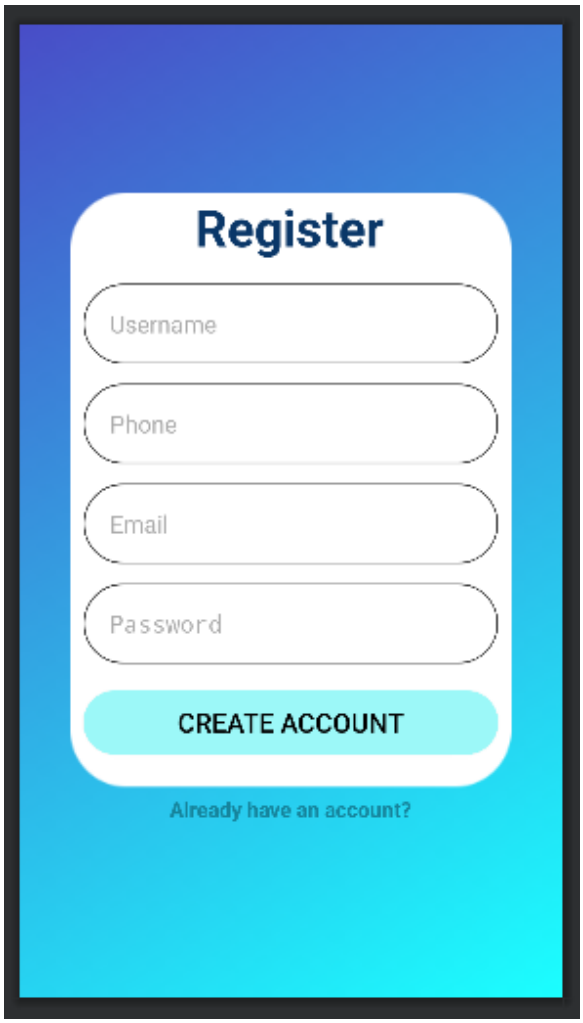
```
private void addGeofence(LatLng latLng, float radius) {

    Geofence geofence = geofenceHelper.getGeofence(GEOFENCE_ID, latLng, radius, Geofence.GEOFENCE_TRANSITION_ENTER |
        Geofence.GEOFENCE_TRANSITION_DWELL | Geofence.GEOFENCE_TRANSITION_EXIT);
    GeofencingRequest geofencingRequest = geofenceHelper.getGeofencingRequest(geofence);
    PendingIntent pendingIntent = geofenceHelper.getPendingIntent();

    geofencingClient.addGeofences(geofencingRequest, pendingIntent)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Log.d(TAG, "onSuccess: Geofence Added...");
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    String errorMessage = geofenceHelper.getErrorString(e);
                    Log.d(TAG, "onFailure: " + errorMessage);
                }
            });
}
```

Functionality for adding markers and circles on containment zones:

```
private void addMarker(LatLng latLng) {
    MarkerOptions markerOptions = new MarkerOptions().position(latLng);
    mMap.addMarker(markerOptions);
}

private void addCircle(LatLng latLng, float radius) {
    CircleOptions circleOptions = new CircleOptions();
    circleOptions.center(latLng);
    circleOptions.radius(radius);
    circleOptions.strokeColor(Color.argb(255, 255, 0,0));
    circleOptions.fillColor(Color.argb(64, 255, 0,0));
    circleOptions.strokeWidth(4);
    mMap.addCircle(circleOptions);
}
```

Login and Registration layouts:





Functionality for Toast notifications

```
switch (transitionType) {
    case Geofence.GEOFENCE_TRANSITION_ENTER:
        Toast.makeText(context, text: "GEOFENCE_TRANSITION_ENTER", Toast.LENGTH_SHORT).show();
        notificationHelper.sendHighPriorityNotification( title: "GEOFENCE_TRANSITION_ENTER", body: "", MapsActivity.class);
        break;
    case Geofence.GEOFENCE_TRANSITION_DWELL:
        Toast.makeText(context, text: "GEOFENCE_TRANSITION_DWELL", Toast.LENGTH_SHORT).show();
        notificationHelper.sendHighPriorityNotification( title: "GEOFENCE_TRANSITION_DWELL", body: "", MapsActivity.class);
        break;
    case Geofence.GEOFENCE_TRANSITION_EXIT:
        Toast.makeText(context, text: "GEOFENCE_TRANSITION_EXIT", Toast.LENGTH_SHORT).show();
        notificationHelper.sendHighPriorityNotification( title: "GEOFENCE_TRANSITION_EXIT", body: "", MapsActivity.class);
        break;
```

API for getting containment zone information:

```
public interface ZoneAPI {
    //http://127.0.0.1:5000/

    @GET("getZonesApp")
    Call<ArrayList<ZoneModel>> getAllZones();
}
```

7.2. Feature 2 (Web App)

Flask application database connection:

```python
hostname = "54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
uid = "zsv28701"
pwd = "tf0qA5FpWEJHSoLF"
database = "bludb"
driver = "{IBM DB2 ODBC DRIVER}"
port = "32733"
protocol = "TCPIP"
security = "SSL"
certificate = "DigiCertGlobalRootCA.crt"

try:
    url = (
        "HOSTNAME = {0};"
        "UID = {1};"
        "PWD = {2};"
        "DATABASE = {3};"
        "PORT = {4};"
        "PROTOCOL = {5};"
        "SECURITY = {6};"
        "SSLServerCertificate = {7};"
        "DRIVER = {8};"
    ).format(
        hostname, uid, pwd, database, port, protocol, security, certificate, driver
    )

    conn = ibm_db.connect(url, "", "")
    print(" * Connected to IBM DB")
except:
    print(" * Unable to connect to IBM DB")
```

Route for sending containment zone information to the app:

```python
@app.route("/getZonesApp")
def returnZonesApp():
    sql = "SELECT * FROM ZONES"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    zoneList = []
    while ibm_db.fetch_row(stmt) != False:
        zones = {}
        zones["ZID"] = ibm_db.result(stmt, 0)
        zones["Latitude"] = ibm_db.result(stmt, 1)
        zones["Longitude"] = ibm_db.result(stmt, 2)
        zones["Name"] = ibm_db.result(stmt, 3)
        zoneList.append(zones)

    print(zoneList)
    return jsonify(zoneList)
```

Dockerfile for web application:

```
FROM python:3.6
# switch working directory
WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt
RUN pip install flask
RUN pip install ibm_db
RUN pip install sendgrid

EXPOSE 5000

ENTRYPOINT [ "python" ]
CMD ["app.py" ]
```

covidtracker.yaml for kubernetes deployment:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: covidtracker2
spec:
  selector:
    app: covidtracker2
  ports:
  - port: 5000
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: covidtracker2
  labels:
    app: covidtracker2
spec:
  selector:
    matchLabels:
      app: covidtracker2
  replicas: 1
  template:
    metadata:
      labels:
        app: covidtracker2
    spec:
      containers:
      - name: covidtracker2
        image: de.icr.io/covidtracker_ns2/new_repo2:latest
        ports:
        - containerPort: 5000
        env:
        - name: DISABLE_WEB_APP
          value: "false"
```

SendGrid notification:

```
# Sendgrid
@app.route("/notify", methods=["POST", "GET"])
def sendMailNotif():
    if request.method == "POST":
        recipient = request.form["recipient"]
        msg = Message("Containment zone alert", recipients=[recipient])
        msg.body = "Containment zone alert!"
        msg.html = (
            "<h1>COVID-19 CONTAINMENT ZONE ALERT</h1>"
            "<p>Warning! You have entered a containment zone. Please be wary of your surroundings.</p>"
        )
        mail.send(msg)
        return "Mail notification sent"
    else:
        return render_template('index.html', msg="404")
```

8. TESTING

8.1 User Acceptance Testing

8.1.1. Purpose:

The purpose of this document is to briefly explain the test coverage and open issues of the Containment Zone Alerting Application project at the time of the release to User Acceptance Testing (UAT).

8.1.2. Defect Analysis:

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub-Total
By Design	5	2	1	2	10
Duplicate	0	0	2	2	4
External	2	3	0	0	5

Fixed	7	5	3	4	19
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	1	1
Won't Fix	0	0	2	1	3
Total	14	10	8	10	42

8.2. Performance Testing

NFT – Risk Assessment

S. No.	Project Name	Scope/ feature	FunctionalChanges	Hardware Changes	Software Changes	Impact	Load/ Volume changes	Risk Score	Justification
1	Containment Zone Alerting Application	New	Low	No Changes	Moderate	NA	>10 to 30%	GREEN	

NFT – Detailed Test Plan

S. No	Project Overview	NFTTest approach	Assumptions/ Dependencies/Risks	Approvals/ SignOff
1	LoginPage	1. Open the Containment ZoneAlerting Application 2. Login with userCredentials	No Risks	N/A
2	Signup Page	1. Open the Containment Zone Alerting Application 2. Enter the Details and Create a newUser	No Risks	N/A
3	Dashboard	1. Log in to Containment Zone Alerting Application 2. User location is obtained automatically	No Risks	N/A
4	User Data	1. Log in to Containment ZoneAlerting Application 2. Location data and visited peoplewill be shown	No Risks	N/A

5	ZoneAddition	1. Log in to Admin portal 2. Admin declares the containmentzone.	NoRisks	N/A
6	Email Notification	1) Mails areSent to the Registered user if the user enters the containment zone	NoRisks	N/A

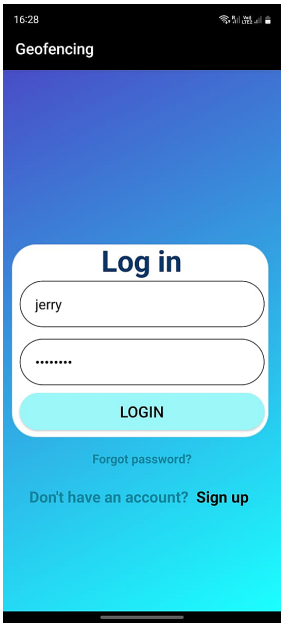
End of Test Report

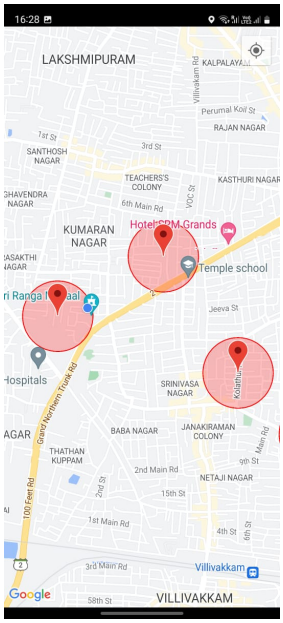
S.No.	Project overview	NFT test approach	NFR – Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified defects (Detected/ Closed/ Open)	Approvals/ Sign off
1	Containment zone alerting	1) Log in to Containment Zone Alerting Application 2) Test for all Testcases 3) Log out of Containment Zone Alerting Application	YES	Test passed	GO	N/A	None	N/A

9. RESULTS

9.1. Screenshots (Android App)

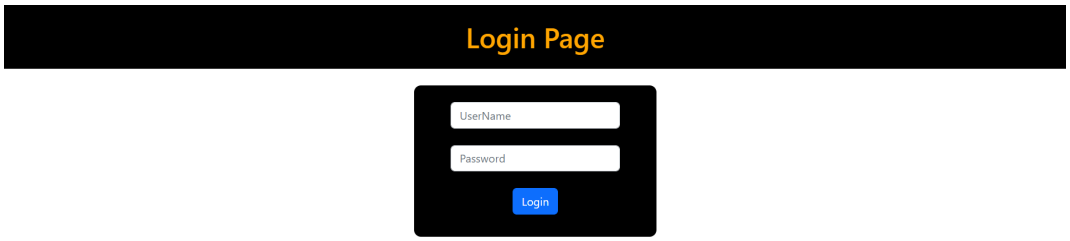
Login and containment zone page



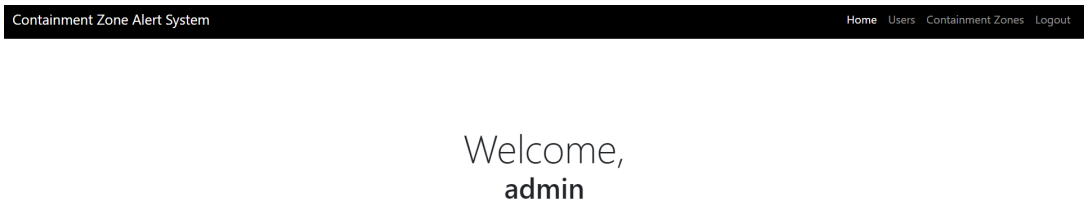


9.2. Screenshots (Web App)

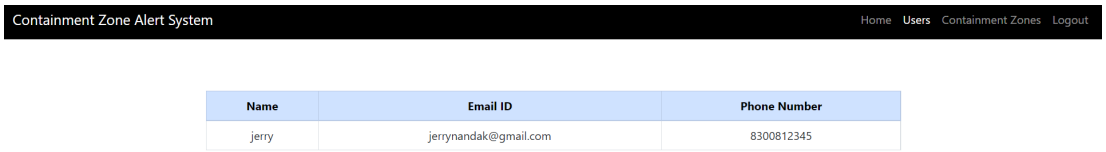
Login page:



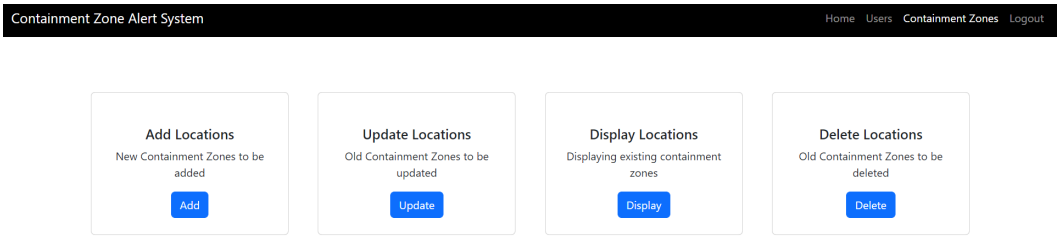
Home page:



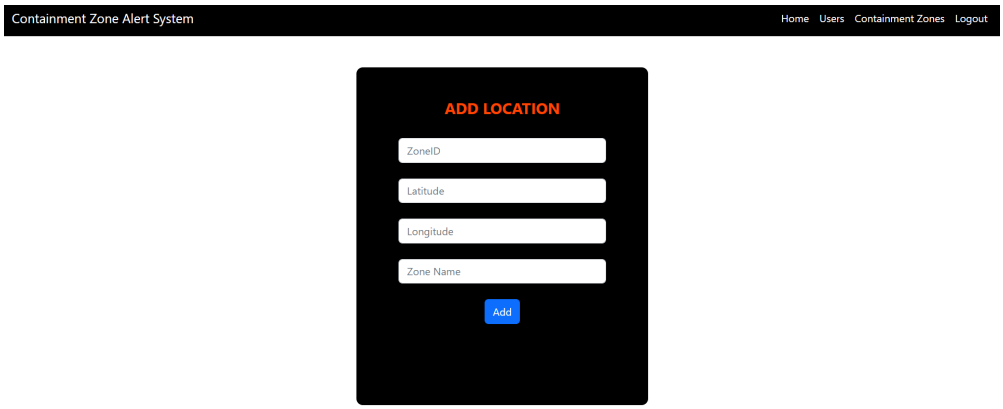
User information page:



Containment zone pages:



Add containment zone page:



10. ADVANTAGES AND DISADVANTAGES

10.1. Advantages

- App is deployable on a majority of android devices as well as operating systems, which means the general user base will be large.
- The application receives updates on the containment zones which ensures that the user gets the latest information on the location of containment zones, eliminating problems related to reliability of information.
- The web application provides a simple, accessible user interface which results in easier web page navigation for admins.

- Both the android app and web app use the same backend, thus eliminating the need for multi-level communication from the android app to the database and also reducing latency for information retrieval.
- User privacy is maintained as very little user information is stored.
- The android application, while displaying the user's current real-time location, also sends notifications/alerts to the user in case they are approaching a containment zone, which improves convenience for users.

10.2. Disadvantages

- The mobile app is limited to android users only.
- Containment zones might become outdated and may require manual change from the admin's end. Live updates from a health data service application would provide better info.
- Too many active containment zones might slow down the application.
- Constant location monitoring may be a privacy concern.

11. CONCLUSION

The application provides an efficient way of showing the identified Covid-19 containment zones to the users in a Google map.

With the alarming increase of Covid-19 affected cases throughout the world, this developed application can be employed as a tool for creating further social awareness among the people.

The application further tracks the user's location and checks whether it is present in the list of identified containment zones, and sends separate notification alerts to the user on entering.

12. FUTURE SCOPE

- The application can be further used for many purposes like maritime and forest safety to prevent users from entering restricted areas.
- The application would also benefit if there were a number of distributed servers that would each store regional information about containment zones, thus distributing workload across various servers which would result in a smoother application experience.
- It would also be better to expand the user base by deploying the application on IoS devices as well.
- The application can also be extended to a variety of other use cases, like maritime and forest safety in order to prevent users from entering restricted forest areas.

13. APPENDIX

Source Code:

The source code can be found in the following GitHub Repository, under Final Deliverables:
https://github.com/IBM-EPBL/IBM-Project-13586-1659522696

Demo Video:

The project demo can be found in the same repository, under Final Deliverables:
https://github.com/IBM-EPBL/IBM-Project-13586-1659522696/blob/main/Final%20Deliverables/IBM-Project-Demo-PNT2022TMID35281.mp4