Project Report

Containment Zone Alerting Application

Team Id: PNT2022MID24049

TABLE OF CONTENTS

ABSTRACT

To stop the number of Covid-19 instances from escalating, numerous research projects were conducted throughout the nation. Our nation had taken action not only to combat the disease but also to raise public awareness of it. The public was made more aware of the issue by the news and media, who informed them of the precautions they can take to avoid getting sick. It could greatly aid in reducing viral spread if individuals are aware of the need to take all protective precautions. Wherever Covid-19 instances had been reported, the nation had established containment zones to stop the virus's spread. In order to prevent contamination outside, these containment zones had been kept private from the general public. The government had 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 could travel around the city for several reasons, including work. However, the containment zones were still maintained in isolation, and new containment zones were being established in all areas where Covid-19 cases have been reported. Since virus-carrying droplets spit out by an unscreened, asymptomatic patient can travel up to 8 meters, these areas are extremely infectious. Despite the fact that police officers were stationed in these containment zones, it was still possible for anyone to wander inside without realizing it. These containment zones provide a risk of contamination in this scenario where people freely travel about the town. 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 od the disease. In this project, our main goal is to create a mobile application that will inform users about the Covid-19 containment zones. The solution would keep track of the user's whereabouts and provide an alarm if the user had entered a containment zone. Additionally, the system would notify users on daily Covid-19 case statistics. The location data will be stored in a cloud datastore and used by the Android application. To build geofences around the containment zones, Android's geofencing client is to be used, and notification manager is to be utilized to provide notifications. The solution would also provide a web-based application for the administrator to manage the containment zones.

LIST OF FIGURES

1. INTRODUCTION

1.1 Project Overview

Since the outbreak of Covid-19 pandemic, lots of preventive measures have been automated by the technical Industry. One of the important measures is to isolate Covid prone zones from normal zones. For this, we seek technical support to create an alert system which makes the user aware of the containment zones by facilitation the integration of the geolocation of the containment areas with a mobile app through cloud so that real time alerting could be made possible. For the updation and storage purposes for the respective zones we use cloud support to make the application complete.

1.2 Purpose

The purpose of the application is to monitor the locations of user continuously and provide alert while trespassing a containment zone. Furthermore, the application is intended to provide information about Covid-containment zones in a particular area. The prime objective of the project is to build an application that provides information about the containment zones of a particular region in order to make the people aware of containment zones in real time. Besides the basic functionalities, the application is designated to provide a few Covid related additional information to its users.

2. LITERATURE SURVEY

2.1 Existing problem

Without widespread public awareness and proactive measures taken by the populace, it is very challenging to stop the community transmission even during a lockdown in a densely populated nation like India. There were several containment zones spread out around the nation, and they were classified into red, orange, and green zones, accordingly. The red zones represent infection hotspots, the orange zones represent some infection, and the green zones represent an infection-free area. It is not a really easy task to inform every citizen of the country of every Covid Containment Zone. Even if it is done, the statistics and the zone details would change every now and then. Also, people can not remember all the containment zones announced in the news or radio. A solution has to be brought not only to keep the people informed of Containment Zone, but also to assist them in staying away from the zones. When a person travels to a place, it is highly unlikely that he or she knows if it is a containment zone or not. There arises a confusion if the person should enter the zone or not.

2.2 References

Aarogya Setu

Designed & Maintained By	National Informatics Centre, Ministry of Electronics & Information Technology, Government of India.
Published on	Google Play Store, Apple Store

Survey

The most popular containment zone alert application among the options currently in use in India is called Aarogya Setu. The Indian government created a mobile application to link the public with crucial health services. Its primary features include geo-location-based COVID-19 data, user risk status, automatic contact tracing using Bluetooth, and much more. The movement of an infected individual is tracked using Bluetooth and GPS technology, and the system notifies the public of the locations the infected person has visited while designating those locations as vulnerable ones. It employs cellular triangulation to determine a person's location in the absence of GPS technology. While Aarogya Setu can track down contacts and notify those who have come into touch with someone who has COVID-19, it also actively keeps track of quarantine or containment zones and alerts users who enter them.

The Terms of Use and Privacy Policy must be accepted at the time of registration when installing the application on any Android or iOS mobile device, and ongoing use of the application denotes continued acceptance. Name, age, sex, occupation, phone number, overseas travel within the previous 28–45 days, and whether the user is a smoker are all pieces of information that the app gathers.

This data is kept on a server that is under the jurisdiction of the Indian government. It is hashed and sent to the user's mobile application along with a special digital ID (DID). The user is recognised using the DID. In order for the user's mobile phone to exchange information with another device that has the app when it gets within range, the Bluetooth and GPS services must be turned on.

Their individual IDs, along with the time and GPS location, are kept on the two phones when two users come into close proximity. The format in which this data is kept is encrypted. Only after a person tests positive is it posted to the government-controlled servers of the app.

Citation / Reference	https://www.aarogyasetu.gov.in/

Development of An Android Application for Viewing Covid-19Containment Zones and Monitoring Violators Who are Trespassing into It Using Firebase and Geofencing

Researcher(s)	Ranajoy Mallik Amlan Protim Hazarika Sudarshana Ghosh Dastidar Dilip Sing Rajib Bandyopadhyays
Published on	National Center For Biotechnology Information (https://www.ncbi.nlm.nih.gov/)

Survey

In this study, the authors concentrated on creating a mobile application to deliver details about the Covid-19 containment zones in West Bengal. The programme also keeps track of the user's whereabouts and sends an alarm if the user enters a containment zone. To keep users informed, the application also offers daily Covid-19 case statistics. The application is made with the Android SDK, and the location information is kept in the Firebase Cloud Firestore. The containment zones are surrounded by geofences made using the Android geofencing client, and notifications are sent using the notification manager. To display the Covid-19 cases in West Bengal, the application also makes use of RESTful web services.

They tested their app with a variety of users in various West Bengal areas, and they discovered that it operated effectively and helped them reach their goal.

Citation / Reference	Mallik R, Hazarika AP, Ghosh Dastidar S, Sing D, Bandyopadhyay R. Development of An Android Application for Viewing Covid-19 Containment Zones and Monitoring Violators Who are Trespassing into It Using Firebase and Geofencing. Trans Indian Natl. Acad. Eng. 2020;5(2):163–79. doi: 10.1007/s41403-020-00137-3. Epub 2020 Jul 1. PMCID: PMC7328652.

Tracking the Covid zones through geo-fencing technique

Researcher(s)	Anto Arockia Rosaline R Lalitha R Hariharan G Lokesh N
Published on	International Journal of Pervasive Computing and Communications (

Survey

Following the tracking of a suspicious person, the geo-fenced layer is mapped out in the vicinity, and the virtual perimeter is then employed for the subsequent trapping procedure. As soon as the Covid monitoring team updates this geo-fenced layer, the public can view it. The idea of creating a virtual perimeter region is known as geo-fencing. Effective containment zone monitoring is made possible by this virtual perimeter monitoring technology. By utilising an automated system based on wireless infrastructure, it lowers operational costs. Additionally, it promptly alerts the law enforcement to find the offenders. As a result, it facilitates the inspection of containment areas and the monitoring of those who disobey governmental regulations.

Users can receive updates from the Covid team on the alert zone. The Covid team has a number of modules for suspect tracking, hotspot fencing, etc. The Covid team must seek a service from the service network provider in the case of suspect tracking, and following authorization, they will offer the coordinates. According to our telecommunication legislation, it is illegal to share data; nonetheless, exchanging personal information without the individual's knowledge via any means is occasionally allowed with governmental approval for investigative purposes.

Citation / Reference	R., A.A.R., R., L., G., H. and N., L. (2020), "Tracking the Covid zones through geo-fencing technique", International Journal of Pervasive Computing and Communications, Vol. 16 No. 5, pp. 409-417. https://doi.org/10.1108/IJPCC-06-2020-0057

Gofencing 2.0: Taking Location-based Notificationsto the Next Level

Researcher(s)	Sandro Rodriguez Garzon Bersant Deva
Published on	UbiComp '14: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing

Survey

The basic Android application that served as the prototype Geofencing client was used. This client is primarily responsible for carrying out the geofencing server's ongoing location update strategy. This must be accomplished with little energy consumption because the Geofencing client is located on a mobile device. We made the decision to employ the low-energy Geofencing features of the Android operating system to keep an eye on the safety zone. As a result, a safety zone is considered as a single circular geofence with a required exit on the mobile device. However, they discovered that there was occasionally a significant lag time between leaving the safety zone and receiving a notification from the system about the leave.

In order to address this issue, a specific amount of the safety zone's radius is decreased. While the safety zone and how it is implemented have a significant impact on overall energy consumption, it is also important to make the right choice when it comes to a placement mechanism. In order to reduce power consumption without compromising the necessary position precision, they used a device-based smart combination of various positioning mechanisms introduced by. By temporarily deactivating placement when a device is not in motion, the Geofencing client also makes use of cutting-edge mobile sensing capabilities integrated into the Android operating system's activity recognition unit. Mobile users who live close to a geo-border fence's will find this to be of particular utility. If the Geofencing server notifies the Geofencing client about a geo-notice, the notification will appear right away.

Citation / Reference	Sandro Rodriguez Garzon and Bersant Deva. 2014. Geofencing 2.0: taking location-based notifications to the next level. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14). Association for Computing Machinery, New York, NY, USA, 921–932. https://doi.org/10.1145/2632048.2636093

Covid Watch

Designed & Maintained By	The State Governement, Karnataka State, India.
Published on	Google Play Store
Survey This app displays the locations of patients with coronary artery disease and a 14-day movement history. This can be used by the general public to track their movements in particular locations. They are urged to dial help line numbers 104, 080-46848600, or 080 66692000 if they are discovered in such places. The app also makes it easier for residents to locate nearby coronavirus treatment facilities, such as sample collecting facilities and testing labs.	
Citation / Reference	https://covid19.karnataka.gov.in/english

2.3 Problem Statement Definition

People travel to different places unaware of the fact that it is a COVID-19 containment zone and hence don't take necessary precautions. As a result, the people getting inside a containment zone are at a higher risk of getting affected by the disease.

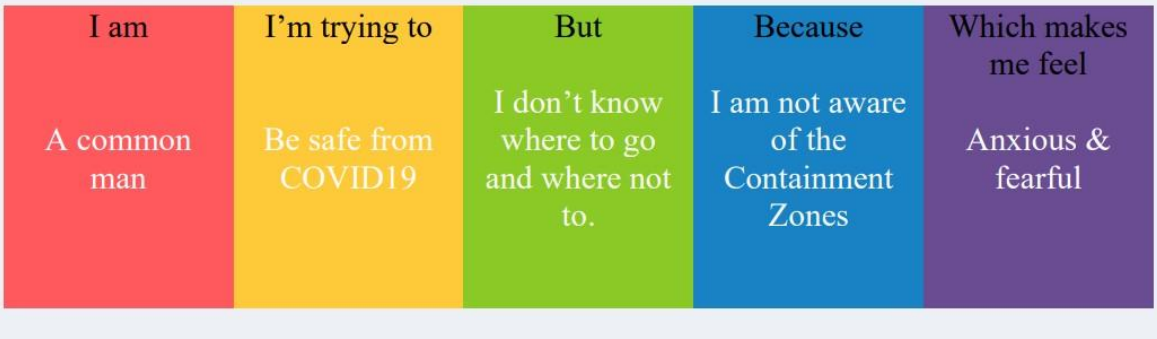

Fig. 2.1. Problem Statement

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Teams can use an empathy map as a collaborative tool to learn more about their clients. An empathy map can represent a group of users, such as a customer segment, in a manner similar to user personas. Qualitative research data, personas, and the team have been gathered to prepare the empathy map.

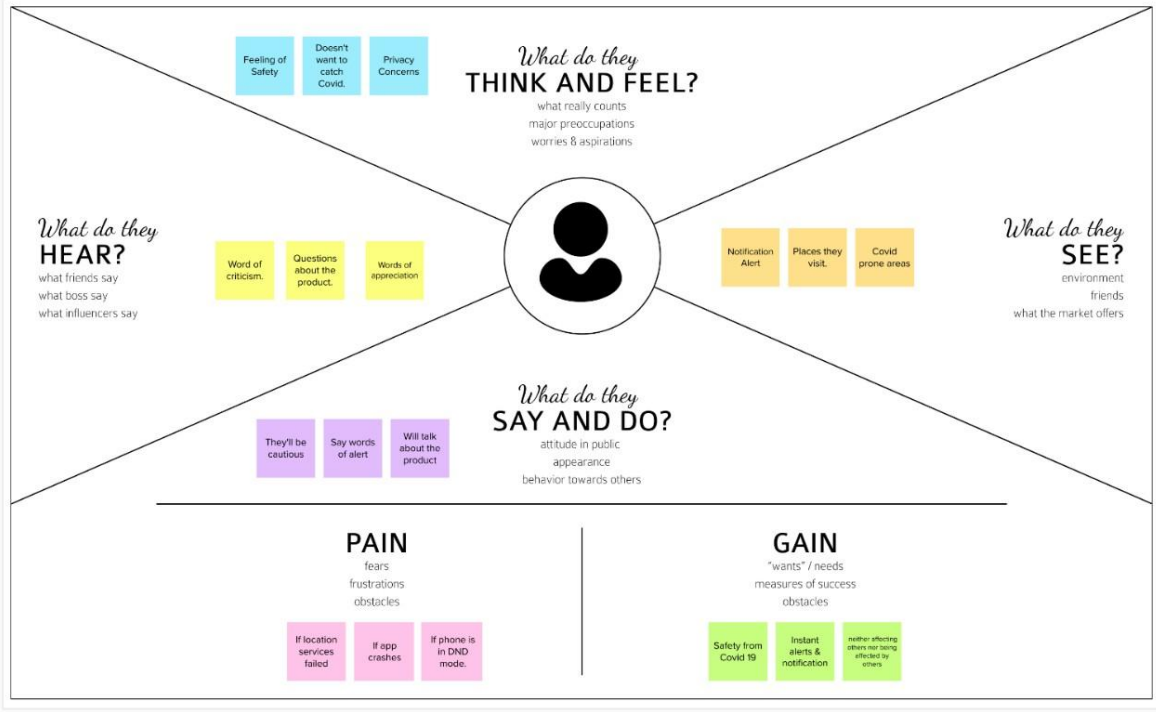

Fig. 3.1. Empathy Map Canvas

3.2 Ideation and Brainstorming

The most common method of ideation is brainstorming. Brainstorming is a process that can help you come up with more creative ideas. It's one of many techniques for ideation, the act of generating fresh concepts. Ideas about the containment zone alert application were the prima facie of discussion during the brainstorming sessions.

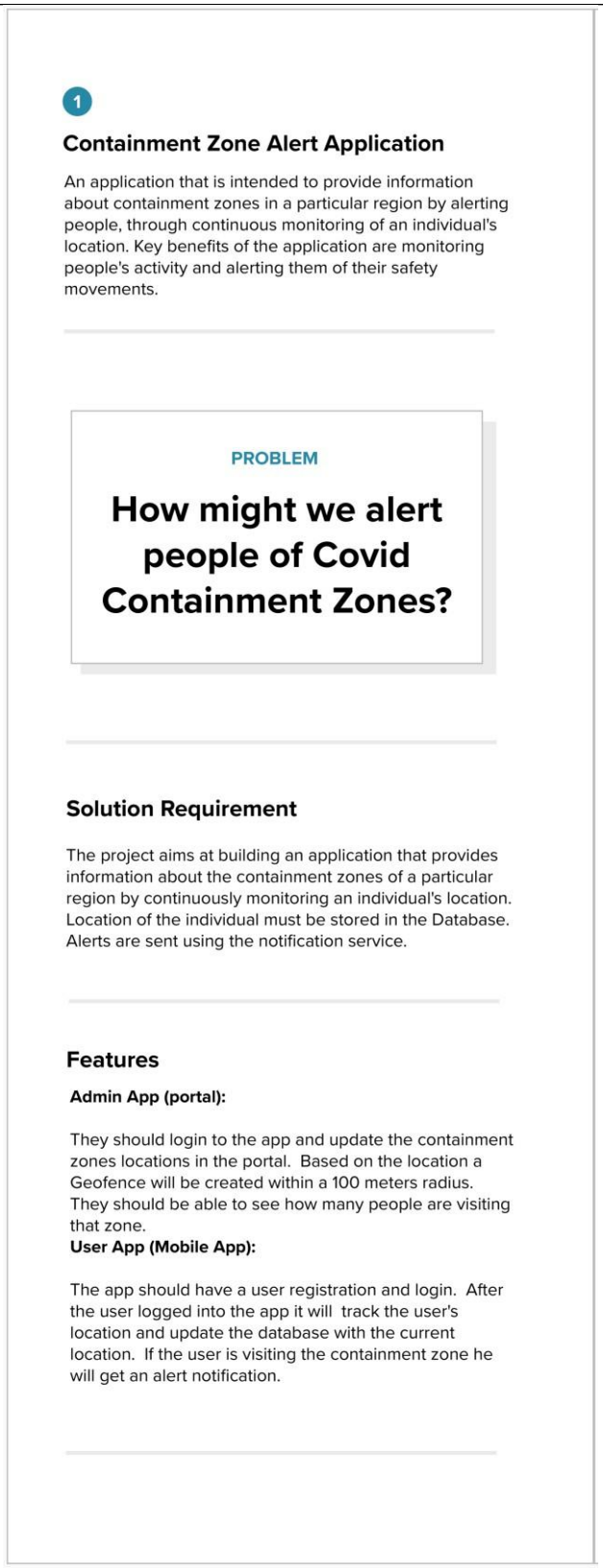

Fig. 3.2. Team Gathering, Collaboration and Selecting the Problem Statement

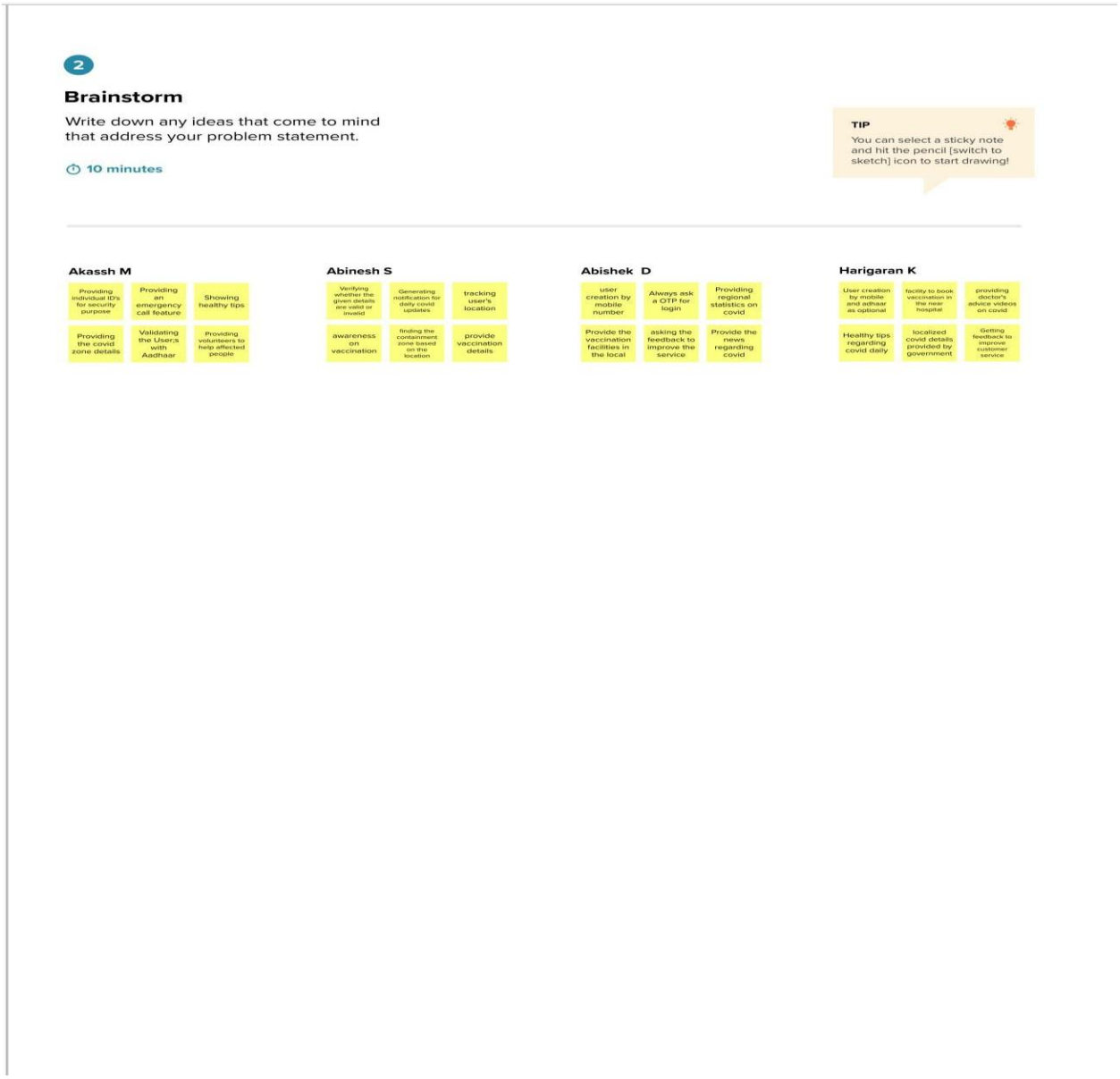

Fig. 3.3. Brainstorming and Idea Listing

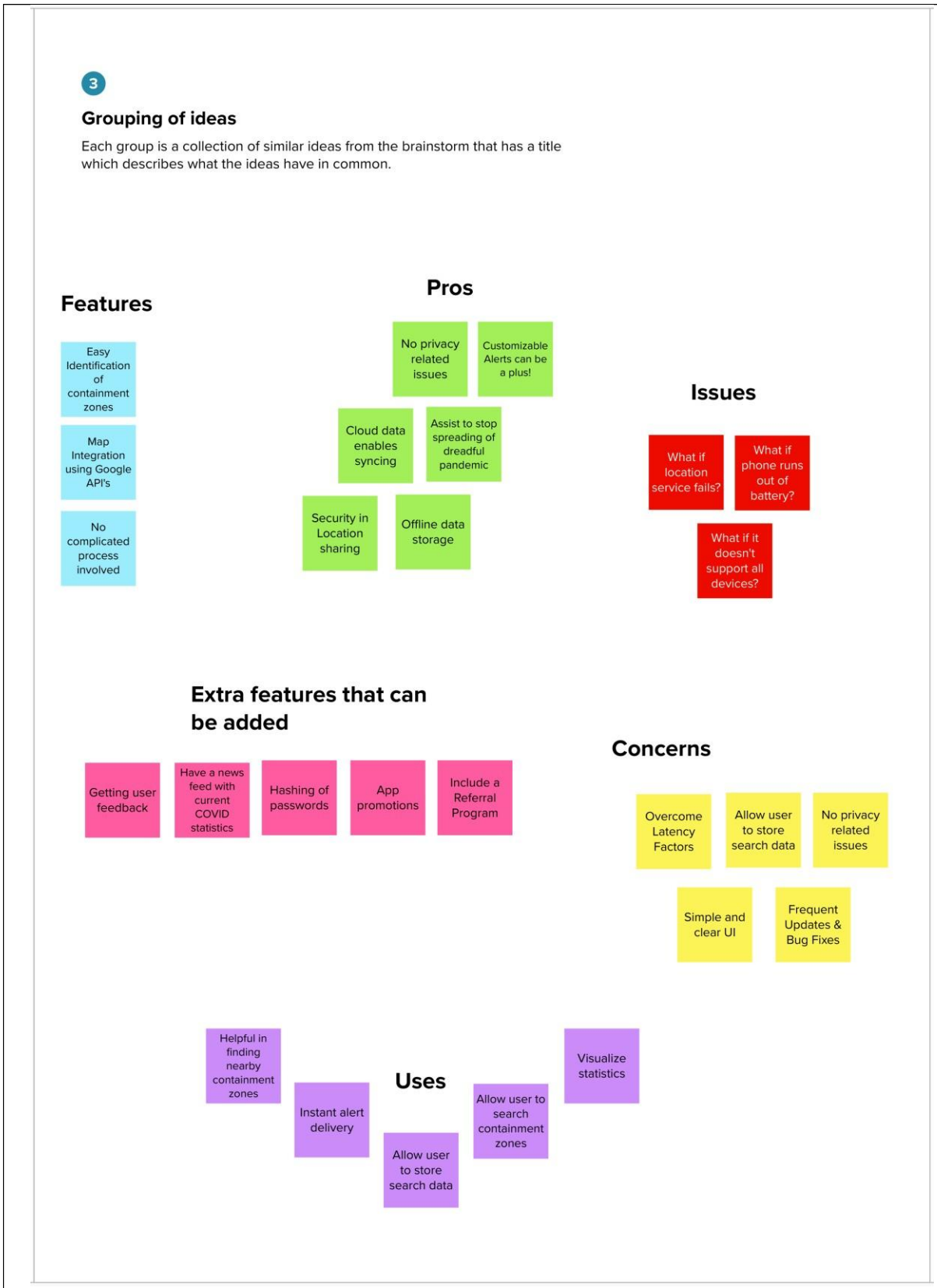

Fig. 3.4. Idea Grouping

Fig. 3.5. Idea Prioritization

3.3 Proposed Solution

An Alerting application that delivers instant alerts whenever a person enters a COVID Containment Zone. This application is intended to provide information about containment zones in a particular region by alerting people, through continuous monitoring of an individual's location. The proposed solution makes use of a cloud store to update and store geolocations of containment zones in real time. For containment zones, geofences are created with the help of the Location Based services of Android.

Fig. 3.6. Proposed Solution

3.4 Problem Solution fit

The phrase "problem-solution fit" refers to the fact that the fundamental issue that gives rise to a business idea is genuinely present, and the suggested solution is effective in resolving it. The suggested solution is thoroughly analyzed and the problem-solution fit is documented.

Fig. 3.7. Problem-Solution Fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirements

Admin App (Web Portal):

To update the containment zone locations in the portal, the admin should log into the web app. Within a 100-meter radius, a Geofence will be constructed based on the location. They ought to be able to observe the volume of visitors to that area.

User App (Mobile App):

The app needs to allow users to sign up and log in. The program will track the user's location after they log in and update the database with their current position. The user will receive an alarm signal if he enters the containment zone.

#	Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
1	**Admin Login**	♦ Separate Portal for Administrator to login and make use of.

2	**Admin Dashboard**	♦ A responsive dashboard that shows all statistics of the app and gives the admin full control over the application. ♦ Controls to perform CRUD operations on the application's database.
3	**User Registration**	♦ A simple and clear interface for user interface that requires the necessary details to be input by the user. ♦ All fields should be validated according to the required detail.
4	**User Login**	♦ One time login, required whence installing using the app for the first time.
5	**Primary Specifics**	♦ Highlight of containment zones in a map interface is to be shown to the user. ♦ Listing of details of Containment zones. ♦ Instant notification delivery on user-entry into a containment zone.
6	**Additional Features**	♦ COVID statistics to be shown to the user. ♦ Covid news from NEWSAPI

4.2 Non-Functional requirements

#	Non-Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
1	**Usability**	♦ The application should provide a good UI/UX design with seamless navigation across sections of the app.

2	**Security**	♦ Passwords should not be stored by the application. ♦ The passwords must be hashed and only the hashed values should be stored in the database. ♦ Privacy and security of the user must be maintained.
3	**Reliability**	♦ The data provided by the application should be reliable. ♦ At any point, data corruption should not exist.
4	**Performance**	♦ The system's performance should not be affected by a change in the number of users.
5	**Availability**	♦ The application and its resources must be available at all times.
6	**Scalability**	♦ The application must be extendable to a larger scale of users.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

A data flow diagram (DFD) is a graphical or visual representation that describes how data is moved through an organization's operations using a standardized set of symbols and notations. A data-flow diagram presents a representation of data movement through a system or a process. The DFD also provides details about each entity's inputs and outputs as well as the process itself. They are used to demonstrate how application flows affect and change information as it travels through them. They comprise of both automated and manual processes.

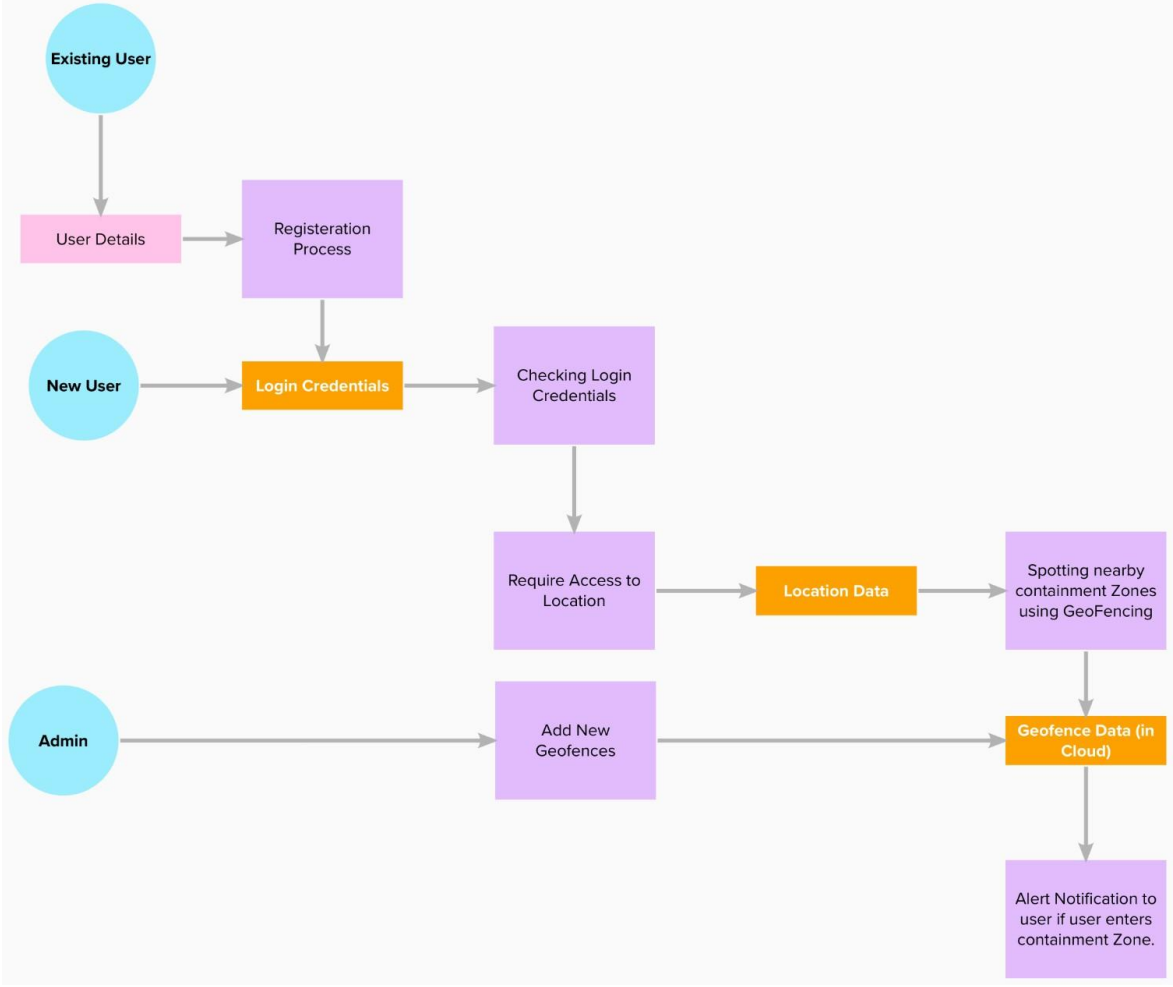

Fig. 5.1. Data Flow Diagram

5.2 Solution and Technical Architecture

Maps

- Uses CloudStore to update and store geolocations of containment zones in real time.
- Achieved through Location Based services of Android.

Statistics

- Provides the Covid-19 key statistics of a Geographical span to the user which he/she may find useful.
- Uses RESTful Web Services for this.

User Support

- Instruction Regarding how to operate the application to help the user in the flow of navigation.

Alerts Module

- Track's the user Location in real time through his/her mobile GPS and provides alert if he/she enters a containment zone.
- Achieved through Geofences.

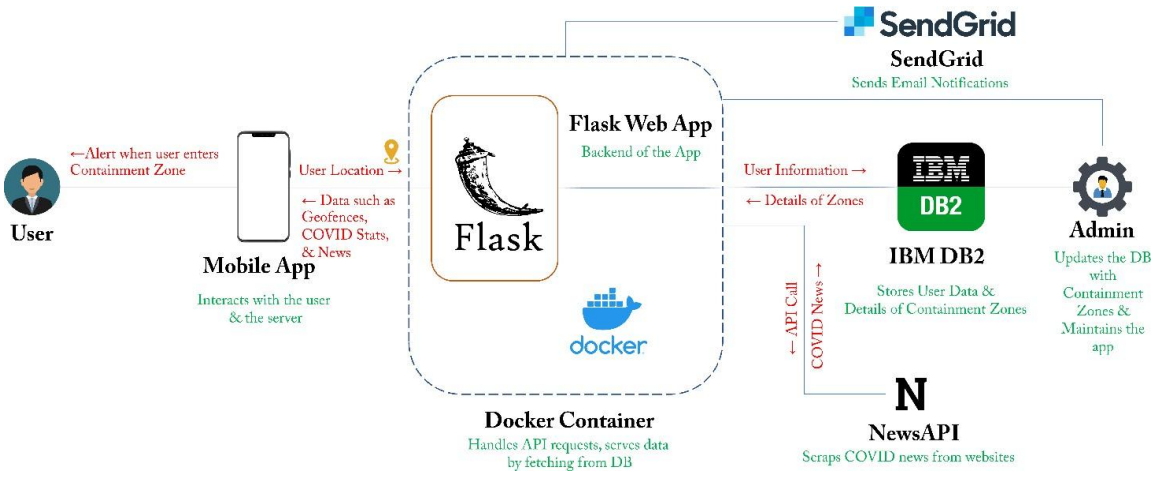

Fig. 5.2. Technical Architecture

5.3 User Stories

User Type	Functional Requirement	User Story	User Story/Task	Acceptance Criteria
Admin	Login	USN-1	As an admin, I want to log in to my dashboard.	I can access my dashboard.
	Dashboard	USN-2	As an Admin, I want to access a dashboard with specialized controls.	I can use my dashboard to update or add information to the database.
User	Registration	USN-3	I would like to see a highlighted version of containment zones in a map interface.	I am displayed a map with the containment zones highlighted in red.
	Login	USN-4	I need to view a listing of details of Containment zones.	I can view a listing of the containment zones.
	Primary Specifics	USN-5	I need instant notification delivery on my entry into a containment zone.	I am notified instantly whenever I step into a containment zone.
		USN-6	COVID statistics	I am provided with COVID statistics.
		USN-7	I would appreciate COVID news.	I get access to COVID news from the web.
	Ease of Use	USN-8	If I forget my password, I need help to reset it.	I can reset my password easily.

6. PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration	USN-1	User: I can register by providing information like username, email, password ,mobile number.	3	High
		USN-2	User: Once I have registered ,I will receive the verification mail through the given mail id.	2	High
		USN-3	Management: we need to register hospitals available near to their surroundings.	2	High
	Login	USN-4	User: I can login to the application by entering my username,email & password	3	High
		USN-5	Management: I may store the Personal information of User in the cloud.	5	Medium
	Dashboard	USN-6	User: I have to give permission to accessmyLocation	5	High
Sprint-2		USN-7	User: After Login Page the dashboard which shows a map with containment zones will appear.	5	High
		USN-8	Management: I have to update the daily cases based on the information Provide by government.	5	High
	Services	USN-9	Admin: I need to provide valid information about the pandemic situation.	5	High
Sprint-3	Dashboard	USN-10	Management: It is necessary to store information of user in cloud for safety measures.	5	High

	Services	USN-11	Admin: I can provide medical Advice and suggestions through a chatbot.	5	Medium
		USN-12	Admin: I need to provide medical advices to the patients based on the consultation provided by doctors.	5	Low
		USN-13	Admin: I need to provide precautionary measures for the users.	5	High
	Services	USN-14	Admin: I have to alter the trespasserby sending notification about 500meter before the containment zones.	3	Medium
	Data Collection	USN-15	Admin: I need to store all the user information on the cloud	5	Medium
		USN-16	Admin: I need to collect the recent List ofsymptoms, death total and daily affected cases.	5	Low

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

JIRA Road Map

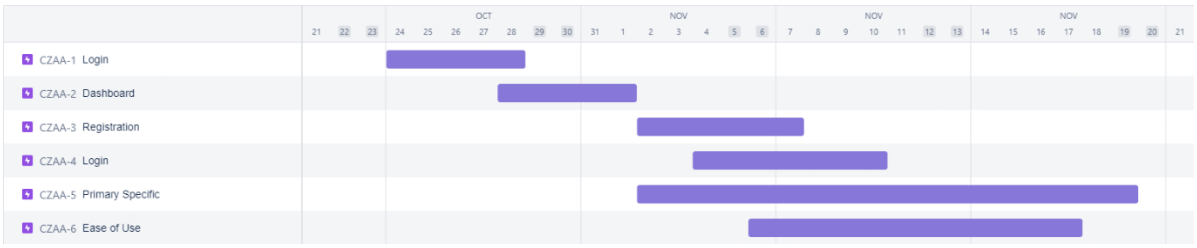

Fig. 6.1. JIRA Roadmap

JIRA Board

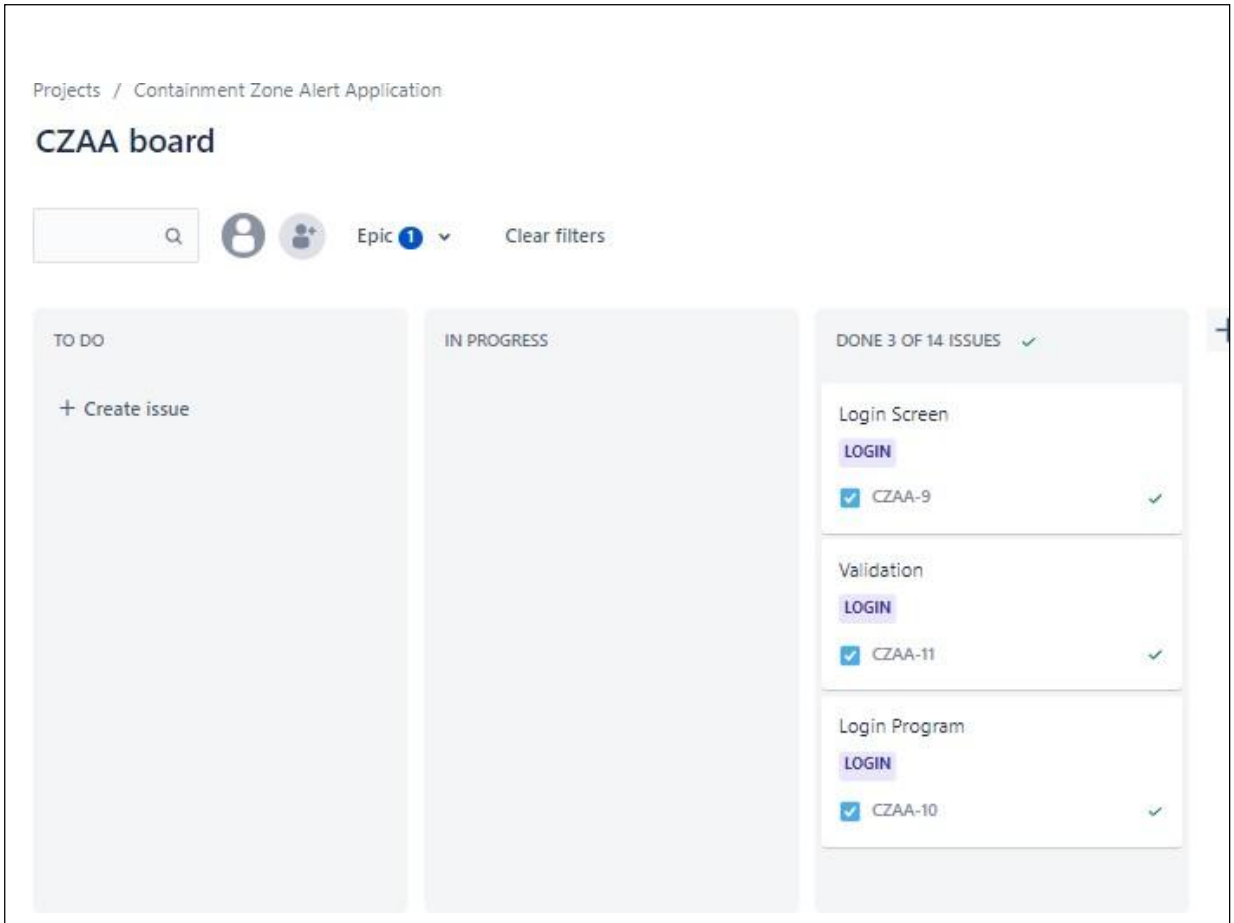

Fig. 6.2. Snip from JIRA Board

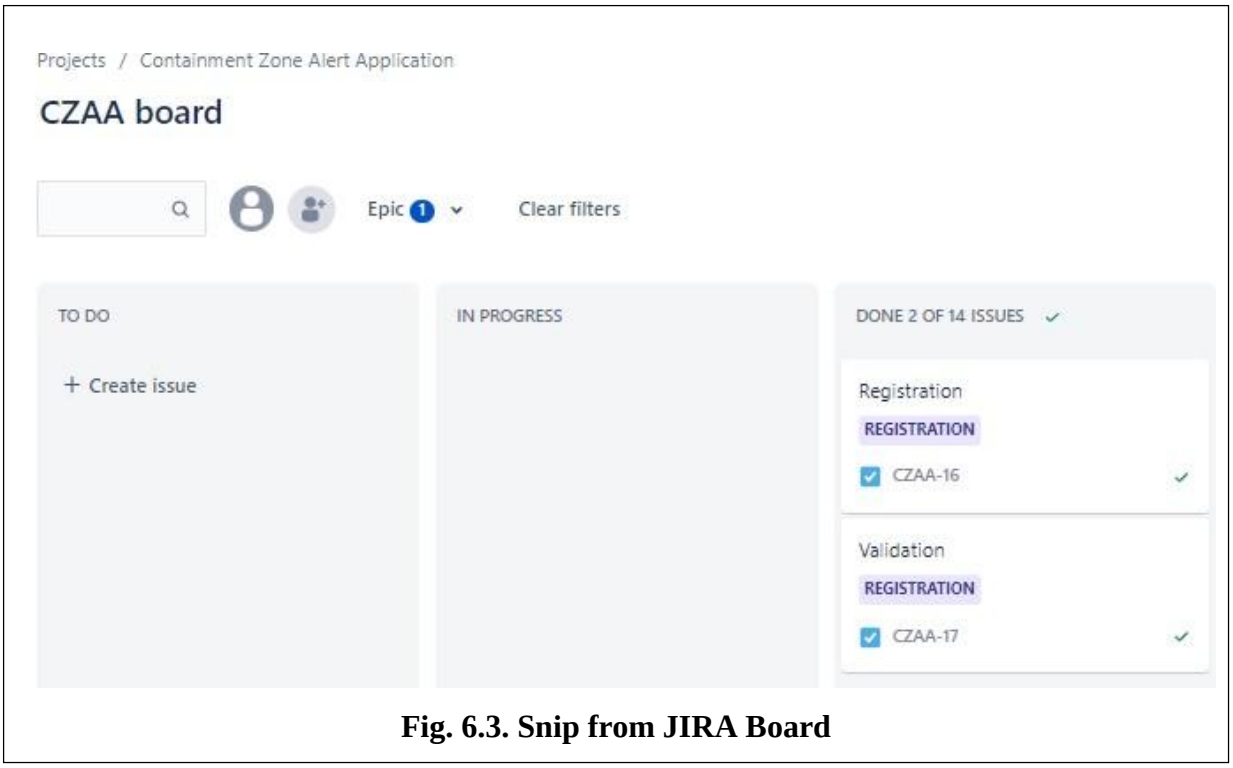

Fig. 6.3. Snip from JIRA Board

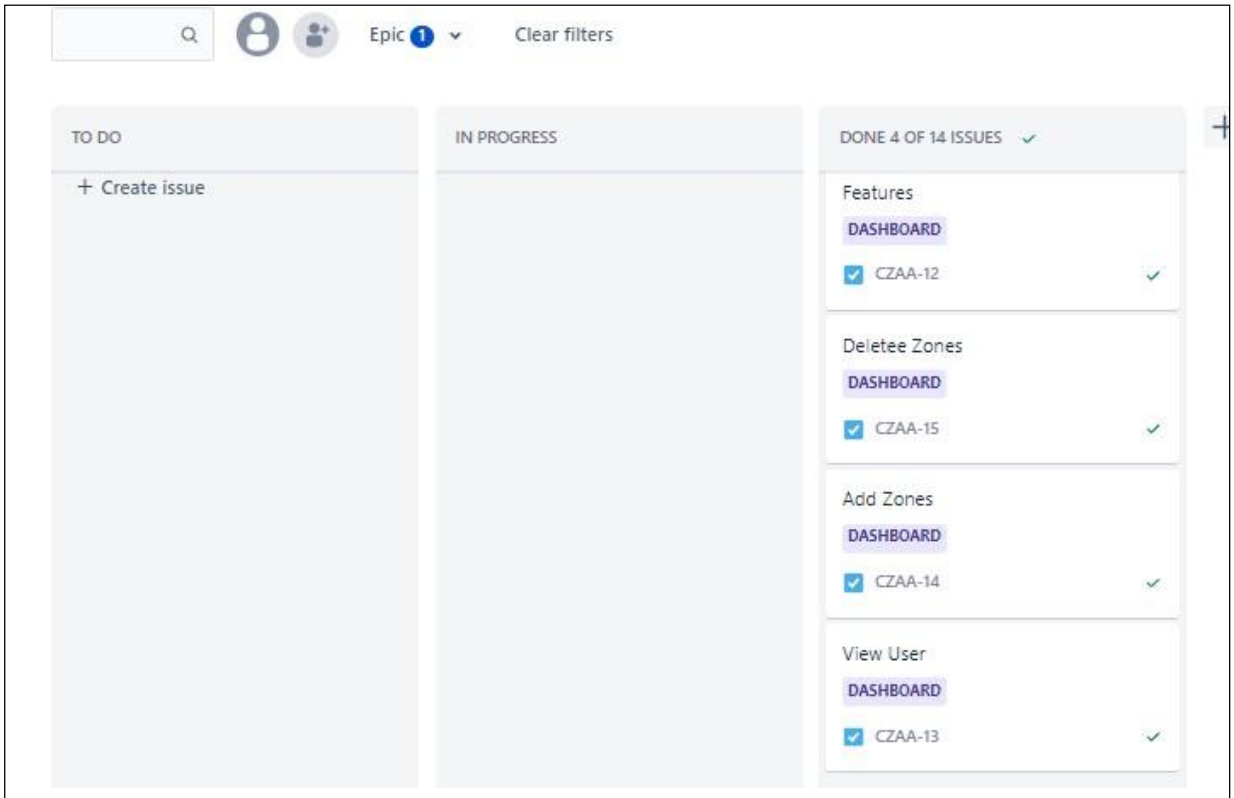

Fig. 6.4. Snip from JIRA Board

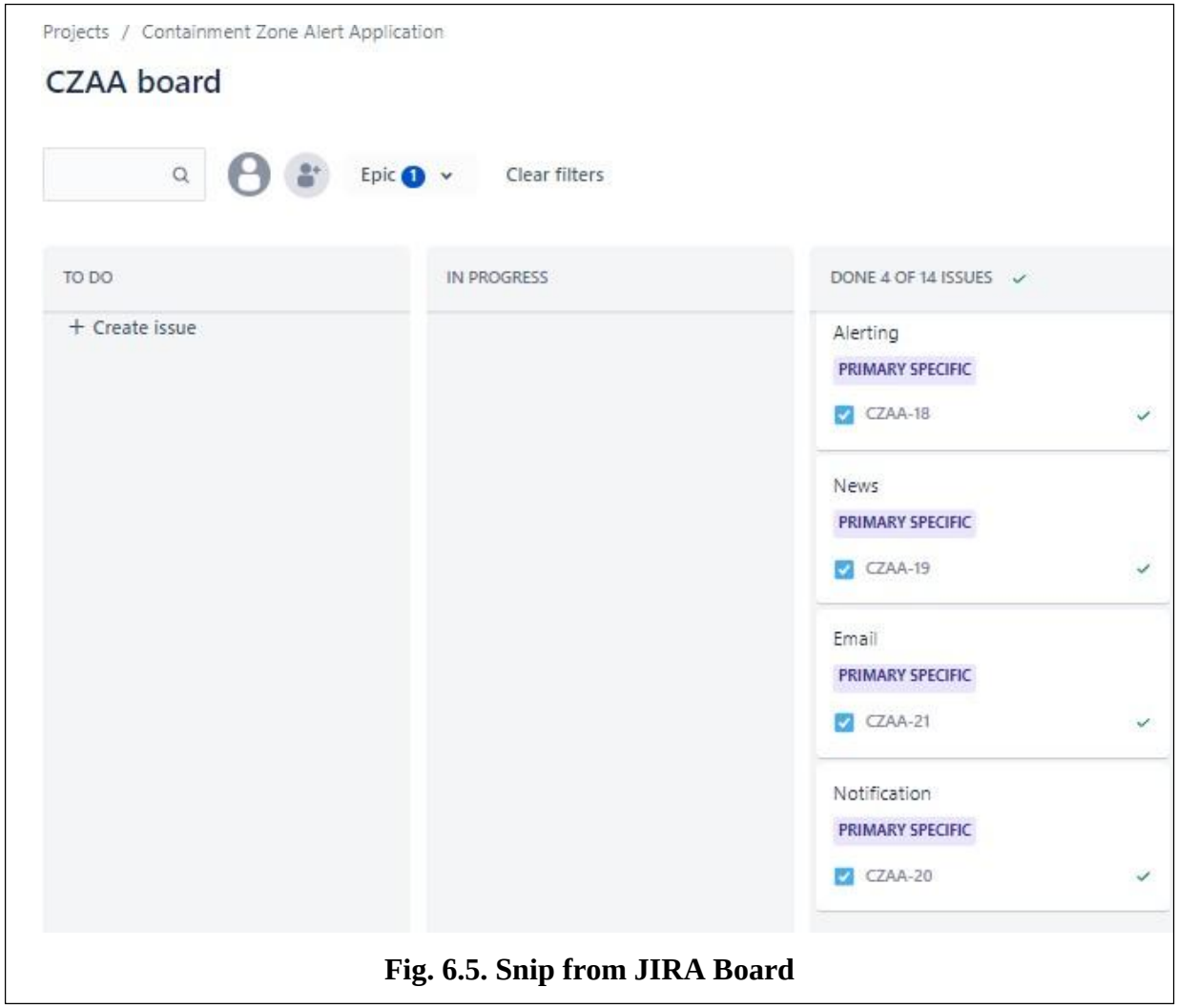

Fig. 6.5. Snip from JIRA Board

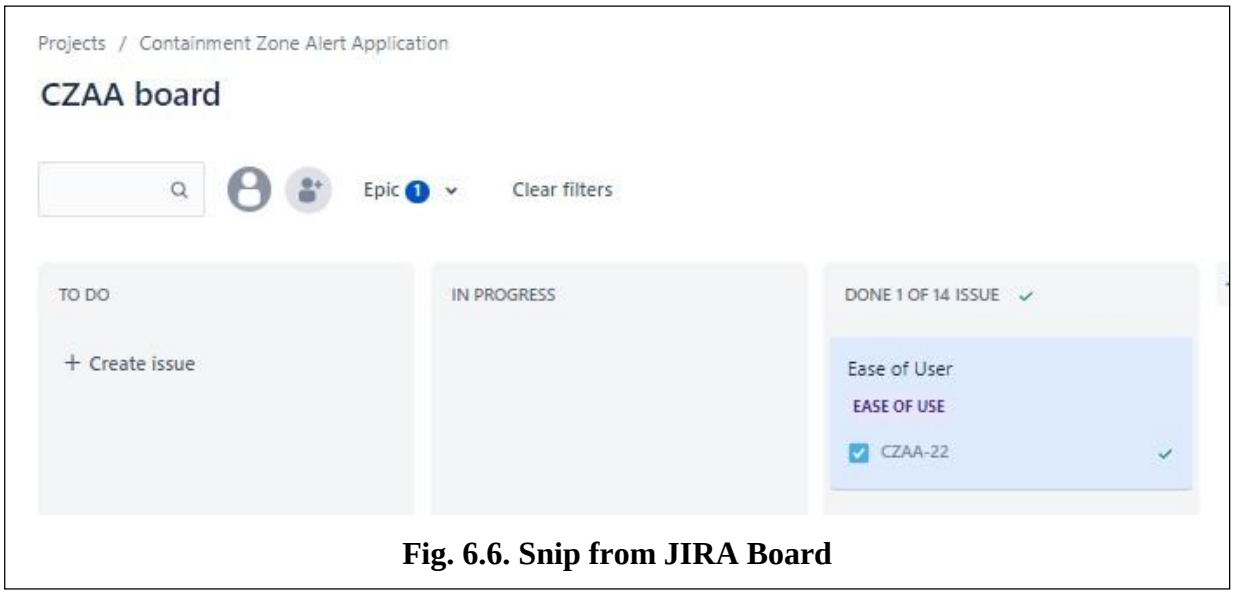

Fig. 6.6. Snip from JIRA Board

7. CODING & SOLUTIONING

7.1 Feature 1 (Android App)

Geofencing can be used to erect virtual walls or fences around specific areas. By giving latitudes, longitudes, and a radius to show how close a location is, developers can install geofences at various locations. The use of geofencing technology involves detecting the user's current position and determining whether it falls within the predefined geofences. Each geofence has the ability to request information from Google's location services concerning entering, residing, and exit events that are triggered by app users as they enter or exit the geofence after it has been created. The geofences can be configured to alert the user when they enter or leave a specific area after receiving a trigger event. These geofences can help to stop users from trespassing inside the designated areas.

MainActivity.java

```
package com.example.finalgeofence;


import androidx.appcompat.app.AppCompatActivity;


import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {


    boolean valid;
    Button loginButton;
    EditText emailIDET;
    EditText passwordET;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        emailIDET = findViewById(R.id.emailID);
        passwordET = findViewById(R.id.userPassword);
        loginButton = findViewById(R.id.loginButton);
        loginButton.setOnClickListener(view -> {
            checkFields(emailIDET);
            checkFields(passwordET);
            if(emailIDET.getText().toString().equals("jebaswinston55@gmail.com") &&
passwordET.getText().toString().equals("Jebas123")) {
                Toast.makeText(this, "LoggedIn Successfully!",
Toast.LENGTH_LONG).show();
                startActivity(new Intent(this, Home.class));
            } else {
                Toast.makeText(this, "Try again, your credentials are not valid!",
Toast.LENGTH_LONG).show();
            }
        });
    }

    public boolean checkFields(EditText textField) {
        if(textField.getText().toString().isEmpty())
{
            valid = false;
            textField.setError("Enter a value!");
        } else {
            valid = true;
        }
        return valid;
    }
}
```

Home.java

```
package com.example.finalgeofence;
```

```
package com.example.finalgeofence;
import androidx.appcompat.app.AppCompatActivity;
import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;
```

```
import android.widget.Toast;


public class Home extends AppCompatActivity {


    Button checkButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        checkButton = findViewById(R.id.checkButton);
        checkButton.setOnClickListener(view -> {
            startActivity(new Intent(this, MapsActivity.class));
        });
    }
}
```

MapsActivity.java

```
import android.Manifest;
import android.app.PendingIntent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingClient;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.CircleOptions;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
GoogleMap.OnMapLongClickListener {

    private static final String TAG = "MapsActivity";

    private GoogleMap mMap;
    private GeofencingClient geofencingClient;
```

```
    private GeofenceHelper geofenceHelper;

    private float GEOFENCE_RADIUS = 200;
    private String GEOFENCE_ID = "SOME_GEOFENCE_ID";

    private int FINE_LOCATION_ACCESS_REQUEST_CODE = 10001;
    private int BACKGROUND_LOCATION_ACCESS_REQUEST_CODE = 10002;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        geofencingClient = LocationServices.getGeofencingClient(this);
        geofenceHelper = new GeofenceHelper(this);
    }


    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng eiffel = new LatLng(48.8589, 2.29365);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(eiffel, 16));
```

```
        enableUserLocation();

        mMap.setOnMapLongClickListener(this);
    }

    private void enableUserLocation() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            mMap.setMyLocationEnabled(true);
        } else {
            //Ask for permission
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.ACCESS_FINE_LOCATION)) {
                //We need to show user a dialog for displaying why the permission is needed and
then ask for the permission...
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
FINE_LOCATION_ACCESS_REQUEST_CODE);
            } else {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
FINE_LOCATION_ACCESS_REQUEST_CODE);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
```

```
    if (requestCode == FINE_LOCATION_ACCESS_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                // TODO: Consider calling
                //    ActivityCompat#requestPermissions
                // here to request the missing permissions, and then overriding
                //        public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                //                                int[] grantResults)
                // to handle the case where the user grants the permission. See the
documentation
                // for ActivityCompat#requestPermissions for more details.
                return;
            }
            mMap.setMyLocationEnabled(true);
        } else {



        }
    }

    if (requestCode == BACKGROUND_LOCATION_ACCESS_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
            //We have the permission
            Toast.makeText(this, "You can add geofences...",
Toast.LENGTH_SHORT).show();
        } else {
            //We do not have the permission..
            Toast.makeText(this, "Background location access is neccessary for geofences to
trigger...", Toast.LENGTH_SHORT).show();
        }
      }
   }

   @Override
   public void onMapLongClick(LatLng latLng) {
      if (Build.VERSION.SDK_INT >= 29) {
         //We need background permission
         if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_BACKGROUND_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            handleMapLongClick(latLng);
         } else {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.ACCESS_BACKGROUND_LOCATION)) {
               //We show a dialog and ask for permission
               ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_BACKGROUND_LOCATION},
BACKGROUND_LOCATION_ACCESS_REQUEST_CODE);
            } else {
               ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_BACKGROUND_LOCATION},
BACKGROUND_LOCATION_ACCESS_REQUEST_CODE);
            }
```

```
        }

    } else {
        handleMapLongClick(latLng);
    }

}

private void handleMapLongClick(LatLng latLng) {
    mMap.clear();
    addMarker(latLng);
    addCircle(latLng, GEOFENCE_RADIUS);
    addGeofence(latLng, GEOFENCE_RADIUS);
}

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();

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    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);
                    e.printStackTrace();
                }
            });
    }



    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);
```

```
    }
}
```

GeofenceHelper.java

```
package com.example.finalgeofence;

import android.app.PendingIntent;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;

import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofenceStatusCodes;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.maps.model.LatLng;

public class GeofenceHelper extends ContextWrapper {

    private static final String TAG = "GeofenceHelper";
    PendingIntent pendingIntent;

    public GeofenceHelper(Context base) {
        super(base);
    }

    public GeofencingRequest getGeofencingRequest(Geofence geofence) {
        return new GeofencingRequest.Builder()
                .addGeofence(geofence)
                .setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
                .build();
```

```
    }

    public Geofence getGeofence(String ID, LatLng latLng, float radius, int transitionTypes)
{
        return new Geofence.Builder()
                .setCircularRegion(latLng.latitude, latLng.longitude, radius)
                .setRequestId(ID)
                .setTransitionTypes(transitionTypes)
                .setLoiteringDelay(5000)
                .setExpirationDuration(Geofence.NEVER_EXPIRE)
                .build();
    }

    public PendingIntent getPendingIntent() {
        if (pendingIntent != null) {
            return pendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        pendingIntent = PendingIntent.getBroadcast(this, 2607, intent,
PendingIntent.FLAG_UPDATE_CURRENT);

        return pendingIntent;
    }

    public String getErrorString(Exception e) {
        if (e instanceof ApiException) {
            ApiException apiException = (ApiException) e;
            switch (apiException.getStatusCode()) {
                case GeofenceStatusCodes
                        .GEOFENCE_NOT_AVAILABLE:
                    return "GEOFENCE_NOT_AVAILABLE";
```

```
            case GeofenceStatusCodes
                    .GEOFENCE_TOO_MANY_GEOFENCES:
                return "GEOFENCE_TOO_MANY_GEOFENCES";
            case GeofenceStatusCodes
                    .GEOFENCE_TOO_MANY_PENDING_INTENTS:
                return "GEOFENCE_TOO_MANY_PENDING_INTENTS";
          }
       }
       return e.getLocalizedMessage();
    }
}
```

GeofenceBroadcastReceiver.java

```
package com.example.finalgeofence;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.util.Log;
import android.widget.Toast;

import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingEvent;

import java.util.List;

public class GeofenceBroadcastReceiver extends BroadcastReceiver {

    private static final String TAG = "GeofenceBroadcastReceiv";
```

```
    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
//        Toast.makeText(context, "Geofence triggered...", Toast.LENGTH_SHORT).show();

        NotificationHelper notificationHelper = new NotificationHelper(context);

        GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);

        if (geofencingEvent.hasError()) {
            Log.d(TAG, "onReceive: Error receiving geofence event...");
            return;
        }

        List<Geofence> geofenceList = geofencingEvent.getTriggeringGeofences();
        for (Geofence geofence: geofenceList) {
            Log.d(TAG, "onReceive: " + geofence.getRequestId());
        }
//        Location location = geofencingEvent.getTriggeringLocation();
        int transitionType = geofencingEvent.getGeofenceTransition();

        switch (transitionType) {
            case Geofence.GEOFENCE_TRANSITION_ENTER:
                Toast.makeText(context,                    "GEOFENCE_TRANSITION_ENTER",
Toast.LENGTH_SHORT).show();

notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION_ENTER",
"", MapsActivity.class);
                break;
            case Geofence.GEOFENCE_TRANSITION_DWELL:
```

```
            Toast.makeText(context, "GEOFENCE_TRANSITION_DWELL",
Toast.LENGTH_SHORT).show();

notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION_DWELL",
"", MapsActivity.class);
            break;
          case Geofence.GEOFENCE_TRANSITION_EXIT:
            Toast.makeText(context, "GEOFENCE_TRANSITION_EXIT",
Toast.LENGTH_SHORT).show();

notificationHelper.sendHighPriorityNotification("GEOFENCE_TRANSITION_EXIT", "",
MapsActivity.class);
            break;
       }

   }
}
```

NotificationHelper.java

```
package com.example.finalgeofence;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import  android.app.PendingIntent;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
```

```
import androidx.annotation.RequiresApi;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import java.util.Random;

public class NotificationHelper extends ContextWrapper {

    private static final String TAG = "NotificationHelper";

    public NotificationHelper(Context base) {
        super(base);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            createChannels();
        }
    }

    private String CHANNEL_NAME = "High priority channel";
    private String CHANNEL_ID = "com.example.notifications" + CHANNEL_NAME;

    @RequiresApi(api = Build.VERSION_CODES.O)
    private void createChannels() {
        NotificationChannel notificationChannel = new NotificationChannel(CHANNEL_ID,
CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
        notificationChannel.enableLights(true);
        notificationChannel.enableVibration(true);
        notificationChannel.setDescription("this is the description of the channel.");
        notificationChannel.setLightColor(Color.RED);
        notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
        NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
```

```
        manager.createNotificationChannel(notificationChannel);
    }

    public void sendHighPriorityNotification(String title, String body, Class activityName) {

        Intent intent = new Intent(this, activityName);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 267, intent,
PendingIntent.FLAG_UPDATE_CURRENT);

        Notification notification = new NotificationCompat.Builder(this, CHANNEL_ID)
//                .setContentTitle(title)
//                .setContentText(body)
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setStyle(new
NotificationCompat.BigTextStyle().setSummaryText("summary").setBigContentTitle(title)
.bigText(body))
                .setContentIntent(pendingIntent)
                .setAutoCancel(true)
                .build();

        NotificationManagerCompat.from(this).notify(new Random().nextInt(), notification);
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.FinalGeofence"
        tools:targetApi="31">

        <receiver
            android:name=".GeofenceBroadcastReceiver"
            android:enabled="true"
            android:exported="true" />

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBJJAA5fyTCN4JrpmIj2uKat6ZxwIjUXkc" />

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>

    <activity
        android:name=".Home"
        android:exported="false">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>

    <activity
        android:name=".MapsActivity"
        android:exported="false">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
  </application>

</manifest>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="WeCare Zone Alerts"
        android:textColor="@color/black"
        android:textSize="40sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.134" />

    <ImageView
        android:layout_width="325dp"
        android:layout_height="236dp"
        android:contentDescription="Login Image"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.341"
        app:srcCompat="@drawable/login" />

    <TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter Login Credentials"
    android:textColor="@color/black"
    android:textSize="75px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.584" />

<EditText
    android:id="@+id/emailID"
    android:layout_width="238dp"
    android:layout_height="49dp"
    android:ems="10"
    android:hint="Email ID"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.542"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.699" />

<EditText
    android:id="@+id/userPassword"
    android:layout_width="237dp"
    android:layout_height="54dp"
    android:ems="10"
```

```
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.54"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.799" />

    <Button
        android:id="@+id/loginButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="LogIn"
        android:backgroundTint="@color/green_200"
        android:textColor="@color/black"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.897" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".Home">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Containment Zones"
    android:textColor="@color/black"
    android:textSize="40sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.097" />

<androidx.cardview.widget.CardView
    android:layout_width="409dp"
    android:layout_height="209dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.281">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Zone ID: 1"
        android:textColor="@color/black"
        android:textSize="75px"
```

```
    android:layout_marginLeft="350px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.584" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="150px"
    android:layout_marginLeft="50px"
    android:text="Latitude: 48"
    android:textColor="@color/black"
    android:textSize="75px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.584" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="275px"
    android:layout_marginLeft="50px"
    android:text="Longitude: 2"
    android:textColor="@color/black"
    android:textSize="75px"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.584" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="400px"
    android:layout_marginLeft="50px"
    android:text="Location: Paris"
    android:textColor="@color/black"
    android:textSize="75px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.584" />

<ImageView
    android:layout_width="350px"
    android:layout_height="350px"
    android:contentDescription="Login Image"
    android:layout_marginTop="150px"
    android:layout_marginLeft="650px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.341"
        app:srcCompat="@drawable/bluemarker" />
</androidx.cardview.widget.CardView>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Check Your Current Location"
    android:textColor="@color/black"
    android:textSize="75px"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.489"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.659" />

<Button
    android:id="@+id/checkButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/green_200"
    android:text="Check"
    android:textColor="@color/black"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.759" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```


```
<?xml version="1.0" encoding="utf-8"?>
```


```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" >
    <item android:id="@+id/geofence" android:title="Create Geofence"
app:showAsAction="never"/>
    <item android:id="@+id/clear" android:title="Clear Geofence"
app:showAsAction="never" />
</menu>
```

main_menu.xml

build.gradle

```
plugins {
    id 'com.android.application'
}

android {
    namespace 'com.example.finalgeofence'
    compileSdk 32
```

```
    defaultConfig {
        applicationId "com.example.finalgeofence"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.android.gms:play-services-maps:18.1.0'
    implementation 'com.google.android.gms:play-services-location:21.0.1'
```

```
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
}
```

7.2 Feature 2 (Web App)

The web application developed using Python Flask is designated for the management of users and the geofences for the app. The operations include addition, updation, editing, and deletion of the geofences as well as users. SendGrid service is integrated with the Python Flask application to provide email alert notifications. The admin is provided with an interactive dashboard to handle all the management features. IBM DB2 is the database used by the app to store the data remotely over cloud.

app.py

```
#* Import Statements
from flask import Flask, render_template, request, session, jsonify
from decouple import config
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *
from flask_mail import Mail, Message
import os
import ibm_db
import re

#* Object Creation for flask
app = Flask(_name_)
app.secret_key = 'a'

#! Important: Connecting to IBM DB2 Database
try:
```

```
    url = "DATABASE=" + config('DATABASE') + "; HOSTNAME=" + config('HOSTNAME') + "; PORT=" + config('PORT') + "; SECURITY=" + config('SECURITY') + "; SSLServerCertificate=" + config('SSLServerCertificate') + "; UID=" + config('UID') + "; PWD=" + config('PWD')
    conn = ibm_db.connect(url, "", "")
    print(" * Connected to IBM DB")
except:
    print(" * Unable to connect to IBM DB")

#? Main Routes
#* Route to login
@app.route('/')
@app.route('/login')
def login():
    return render_template('index.html')

#* Route to home
@app.route('/home', methods=['GET', 'POST'])
def home():
    global userid
    msg = ''

    if request.method == 'POST':
        username = request.form['UserName']
        password = request.form['Password']
        sql = "SELECT * FROM ADMIN WHERE Name = ? AND Password = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
```

```
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['NAME']
            userid = account['NAME']
            session['UserName'] = account['NAME']
            msg = 'Logged in Successfully!'
            return render_template('home.html', user = username)
        else:
            msg = 'Incorrect UserName or Password!'
            return render_template('index.html', msg = msg)

#* Route to logout
@app.route('/logout')
def logout():
    session.pop('Loggedin', None)
    session.pop('id', None)
    session.pop('UserName', None)
    return render_template('index.htmL')

#? User Routes
#* Route to user details manipulation
@app.route('/user')
def user():
    sql = "SELECT * FROM USER"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    userList = []
    while ibm_db.fetch_row(stmt) != False:
        users = {}
        users["UserName"] = ibm_db.result(stmt, 0)
```

```
        users["EmailID"] = ibm_db.result(stmt, 1)
        users["PhoneNumber"] = ibm_db.result(stmt, 2)
        userList.append(users)
    return render_template('user.html', users=userList);

#* Route to add new User
@app.route('/new')
def new():
    return render_template('addUser.html')

@app.route('/user/new')
def newUser():
    if request.method == 'POST':
        username = request.form['UserName']
        email = request.form['EmailAddress']
        phonenumber = request.form['PhoneNumber']
        password = request.form['Password']
        sql = "SELECT * FROM USER WHERE Name = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists!'
        elif not re.match(r'[^]', email):
            msg = 'Invalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Name must contain characters and numbers'
        else:
            insert_sql = "INSERT into USER values (?, ?, ?, ?)"
```

```
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phonenumber)
            ibm_db.bind_param(prep_stmt, 4, password)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered'
            return render_template('addUser.html', msg=msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form'
        return render_template('addUser.html', msg = msg)

#? Zone Routes
#* Route to Containment Zones
@app.route('/zones')
def zones():
    return render_template('zones.html')

#* Route to add new zones
@app.route('/zones/add')
def zoneAddPage():
    return render_template('addZone.html')

#* Adding new zones to DB2
@app.route('/zones/new', methods=['POST'])
def zoneAdd():
    if request.method == 'POST':
        zid = request.form['ZoneID']
        latitude = request.form['Latitude']
        longitude = request.form['Longitude']
        zoneName = request.form['ZoneName']
```

```
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        print(zone)
        if zone:
            msg = 'Zone already exists!'
        else:
            insert_sql = "INSERT INTO ZONES VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, zid)
            ibm_db.bind_param(prep_stmt, 2, latitude)
            ibm_db.bind_param(prep_stmt, 3, longitude)
            ibm_db.bind_param(prep_stmt, 4, zoneName)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully added'
        return render_template('addZone.html', msg=msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form'
        return render_template('addZone.html', msg = msg)

#* Route to update old zones
@app.route('/zones/update')
def zoneUpdatePage():
    return render_template('updateZone.html')

@app.route('/zones/alter', methods=['POST'])
def zoneAlter():
    if request.method == 'POST':
        zid = request.form['ZoneID']
```

```
        latitude = request.form['Latitude']
        longitude = request.form['Longitude']
        zoneName = request.form['ZoneName']
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        print(zone)
        if zone:
            update_sql = "UPDATE ZONES SET ZID = ?, Latitude = ?, Longitude = ?, Name
= ? WHERE ZID = ?"
            prep_stmt = ibm_db.prepare(conn, update_sql)
            ibm_db.bind_param(prep_stmt, 1, zid)
            ibm_db.bind_param(prep_stmt, 2, latitude)
            ibm_db.bind_param(prep_stmt, 3, longitude)
            ibm_db.bind_param(prep_stmt, 4, zoneName)
            ibm_db.bind_param(prep_stmt, 5, zid)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully added'
        else:
            msg = 'Zone not exists!'
        return render_template('updateZone.html', msg=msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form'
        return render_template('updateZone.html', msg = msg)


#* Route to display all zones
@app.route('/zones/display')
def zoneDisplay():
    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)
    return render_template('displayZone.html', zones=zoneList)

#* Route to delete old zones
@app.route('/zones/delete')
def zoneDeletePage():
    return render_template('deleteZone.html')

@app.route('/zones/remove', methods=['POST'])
def removeZone():
    msg = ''
    if request.method == 'POST':
        zid = request.form['ZoneID']
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        if(zone):
            delete_query = "DELETE FROM ZONES WHERE ZID = ?"
            prep_stmt = ibm_db.prepare(conn, delete_query)
            ibm_db.bind_param(prep_stmt, 1, zid)
```

```
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully deleted.'
        else:
            msg = 'Sorry! Deletion Failed, Zone not exists.'
        return render_template('deleteZone.html', msg = msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form'
        return render_template('deleteZone.html', msg = msg)

#? APIs for User App
#* All zone locations
@app.route('/location')
def location():
    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)
    return jsonify(value = zoneList)

htmlTemplate = ''

#* SendGrid Integration
@app.route('/alert/<name>')
def alert(name):
```

```
    sql = "SELECT EmailID FROM USER WHERE Name = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, name)
    ibm_db.execute(stmt)
    email = ibm_db.fetch_assoc(stmt)
    print(email)
    message = Mail(
        from_email=config('WECARE'),
        to_emails=email,
        subject='Alert!!!!!!!!!!!!!!!!!!!!!!!',
        html_content='SendGridMail.html')
    try:
        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
    return "Success"

#* Run the flask server
if__name__== "_main_":
    app.run(host='0.0.0.0', debug=True)
```

styles.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;300&display=swap');

*{
    margin: 0px;
```

```
    padding: 0px;
    box-sizing: border-box;
    font-family: "Poppins", sans-serif;
}

body {
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #082032;
}

.rounded-corners {
    border-radius: 10px;
}

.wrapper {
    display: flex;
    align-items: center;
    justify-content: center;
}

.wrapper .static-txt {
    color: #fff;
    font-size: 60px;
    font-weight: 400;
}

.dynamic-txt {
    margin-left: 15px;
```

```
    line-height: 90px;
    height: 90px;
}

.dynamic-txt p {
    color: #ff4c29;
    font-size: 60px;
    font-weight: 500;
    position: relative;
    animation: slide 12s steps(4) infinite;
}

.dynamic-txt p {
    position: relative;
}

.dynamic-txt p::after {
    content: " ";
    position: absolute;
    left: 0;
    height: 100%;
    width: 100%;
    background-color: #082032;
    border-left: 2px solid #ff4c29;
    animation: typing 3s steps(10) infinite;
}

@keyframes typing {
    40%,
    60% {
        left: calc(100% + 10px);
```

```
    }
    100% {
        left: 0;
    }
}

.nav-bar {
    height: 100%;
    width: 100%;
    position: absolute;
}

nav {
    display: flex;
    justify-content: center;
    align-items: center;
    padding-top: 40px;
    padding-left: 10%;
    padding-right: 10%;
    border-radius: 50px;
}

nav ul li {
    list-style-type:  none;
    display: inline-block;
    padding: 10px 20px;
}

nav ul li a {
    color: white;
    text-decoration: none;
```

```
    font-weight: bold;
}

nav ul li a:hover {
    color: #ff4c29;
    transition: .3s;
}
```

main.js

```
// Registration Page
function sendData() {
    var userName = document.getElementById('UserName').value;
    var email = document.getElementById('email').value;
    var phoneNumber = document.getElementById('phoneNumber').value;

    sessionStorage.setItem('USER', userName);
    sessionStorage.setItem('EMAIL', email);
    sessionStorage.setItem('PHONE', phoneNumber);

    return;
}
// Home Page
window.addEventListener('load', ()=>
{
    var userName = sessionStorage.getItem('USER');
    var email = sessionStorage.getItem('EMAIL');
    var phoneNumber = sessionStorage.getItem('PHONE');

    document.getElementById('USER').innerHTML = userName;
    // document.getElementById('email').innerHTML = email;
})
```

requirements.txt

```
flask
ibm_db
sendgrid
docopt
```

Dockerfile

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

# install the dependencies and packages in the requirements file
RUN pip freeze > requirements.txt
RUN pip install -r requirements.txt
COPY ./requirements.txt /app/requirements.txt
RUN pip install flask
RUN pip install ibm_db
# copy every content from the local file to the image
COPY . /myapp
# configure the container to run in an executed manner
ENTRYPOINT [ "python" ]
CMD ["app.py" ]
```

sendgrid.env

```
"export SENDGRID_API_KEY='SG.fibT2-EhSfqNn53UPZsfA.Q0tCMM7ZS98PYoCmg
3iPnaBIsBVLUPCEnUYLS51B1_4'"
```

8. TESTING

8.1 User Acceptance Testing

The below table is to briefly explain the test coverage and open issues of the Containment Zone Alert Application project at the time of the release to User Acceptance Testing (UAT). It 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	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score
1	Containment Zone Alerting Application	New	Low	No Changes	Moderate	NA	>10 to 30%	GREEN

NFT - Detailed Test Plan				
#	**Project Overview**	**NFT Test approach**	**Assumptions/ Dependencies/ Risks**	**Approvals/ SignOff**
1	Login Page	1) Open the Containment Zone Alerting Application 2) Login with user Credentials	No Risks	N/A
2	Signup Page	1) Open the Containment Zone Alerting Application 2) Enter the Details and Create a new User	No Risks	N/A
3	Dashboard	1) Log in to Containment Zone Alerting Application 2) User location is got automatically	No Risks	N/A
4	User Data	1) Log in to Containment Zone Alerting Application 2) Location data and visited people will be shown	No Risks	N/A
5	Zone Addition	1) Log in to Containment Zone Alerting Application 2) Admin declares the containment zone.	No Risks	N/A
6	Email Acknowledgement	1) Mails are Sent to the Registered user if the user enters into the containment zone	No Risks	N/A

			End Of Test Report	
S.No	**Project Overview**	**NFT Test approach**	**NFR - Met**	**Test Outcome**
1	Containment Zone Alerting Application	1) Log in to Containment Zone Alerting Application 2) Test for all Testcases 3) Log out to Containment Zone Alerting Application	YES	Test Passed

9. RESULTS

9.1 Screenshots (Android App)

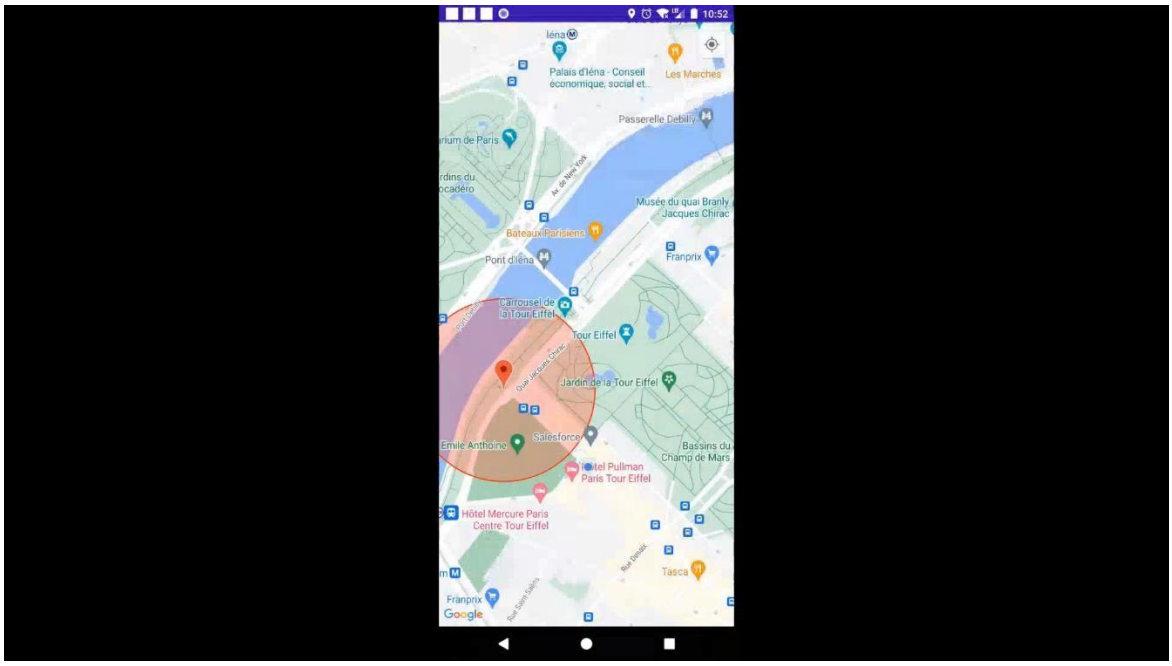

Fig. 9.1. A Containment Zone

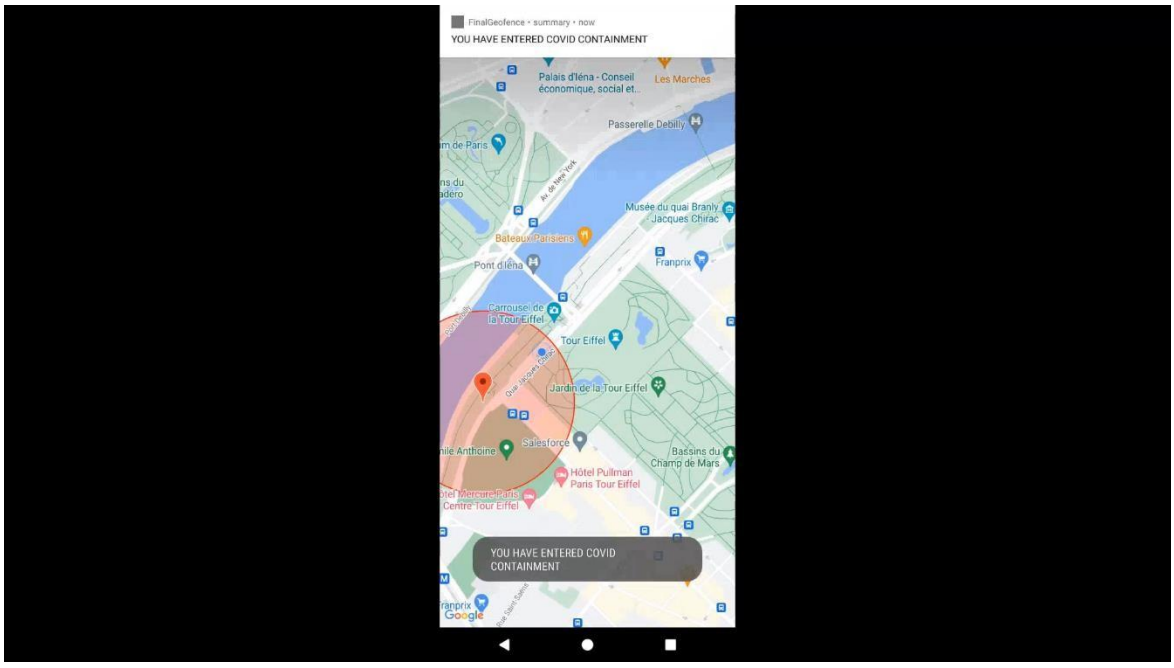

Fig. 9.2. User enters a Containment Zone

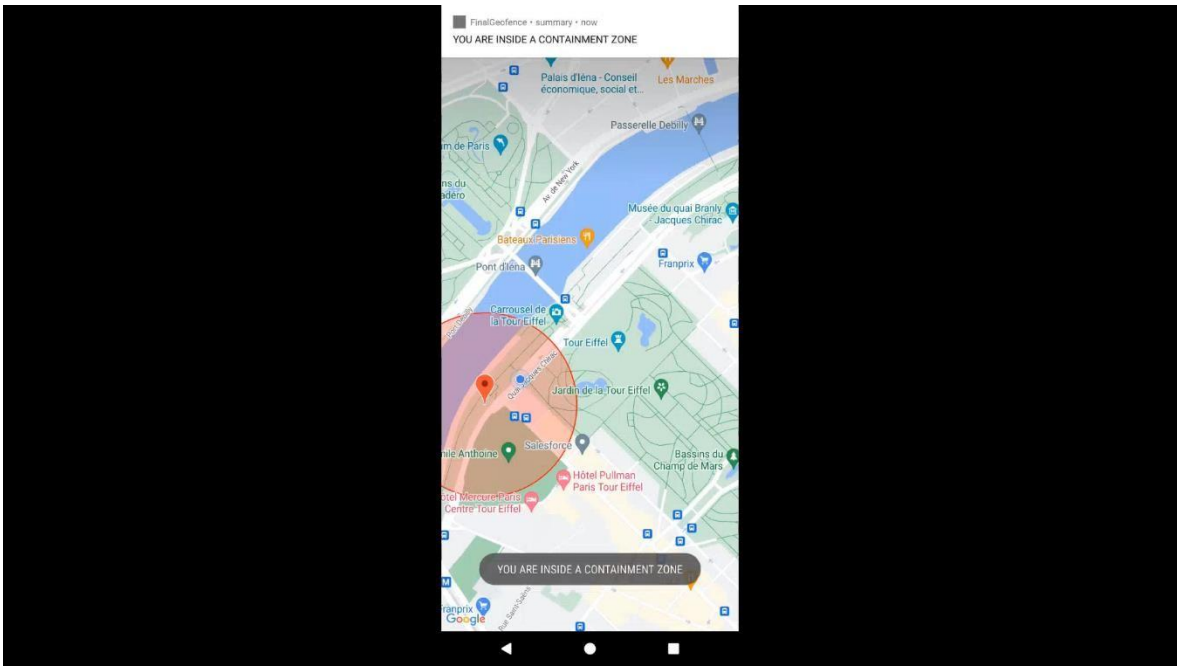

Fig. 9.3. User is inside a Containment Zone

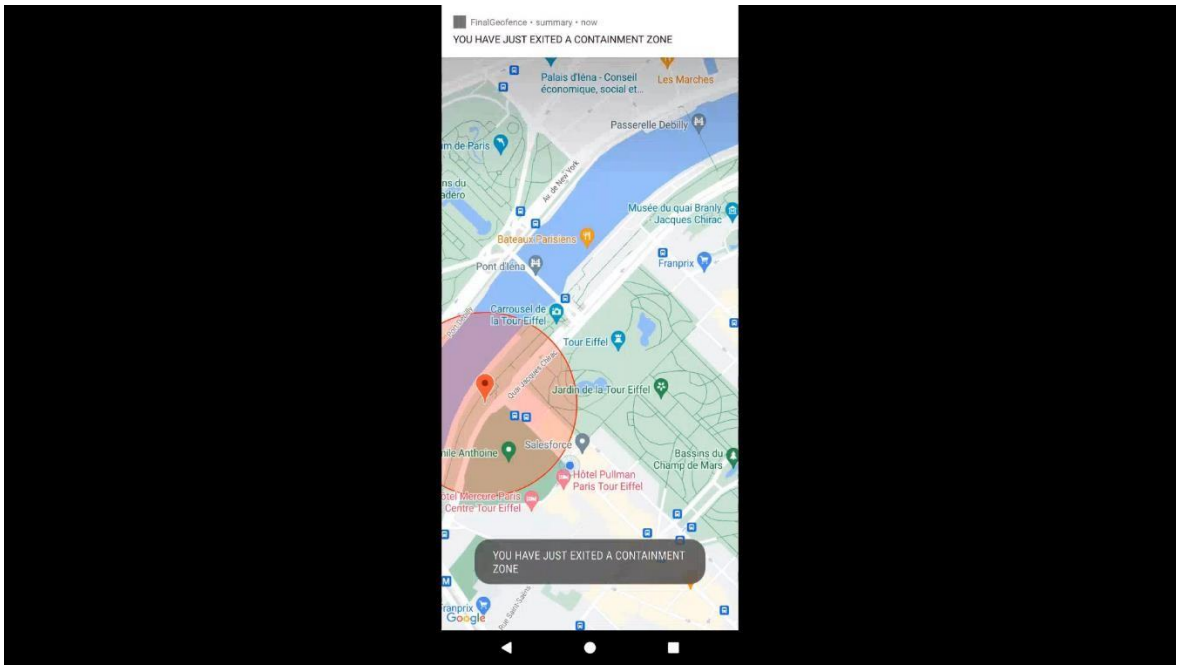

Fig. 9.4. User exits a Containment Zone

9.2 Screenshots (Web App)

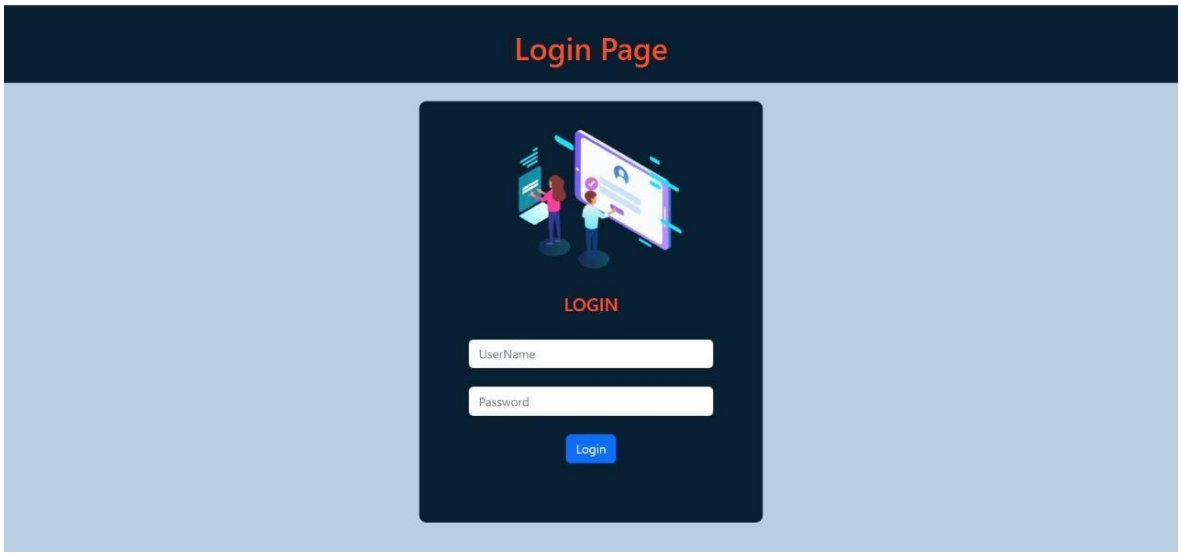

Fig. 9.5. Login Page

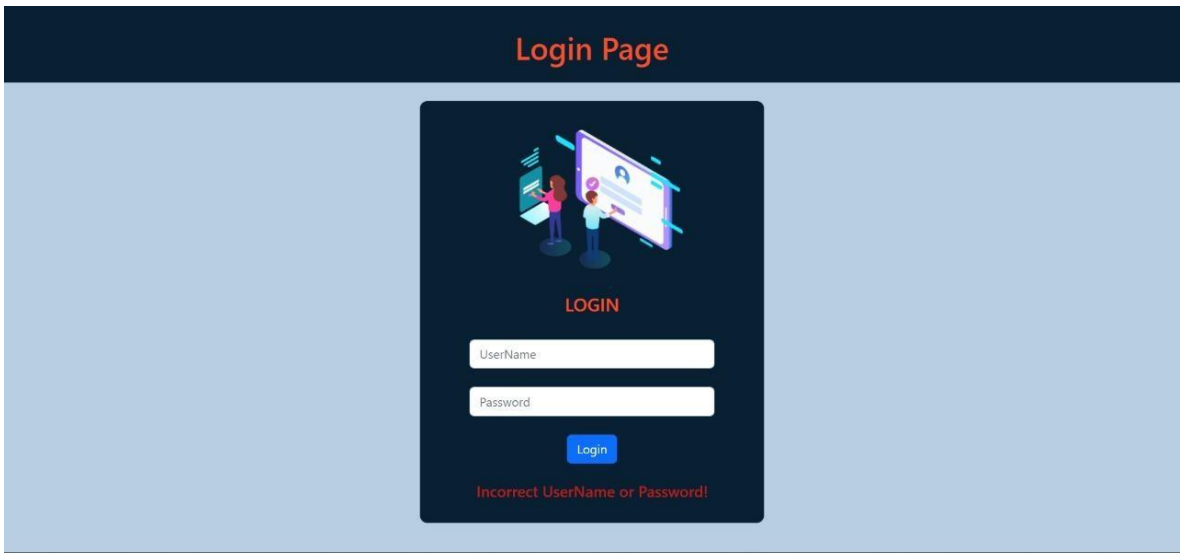

Fig. 9.6. Login Validation

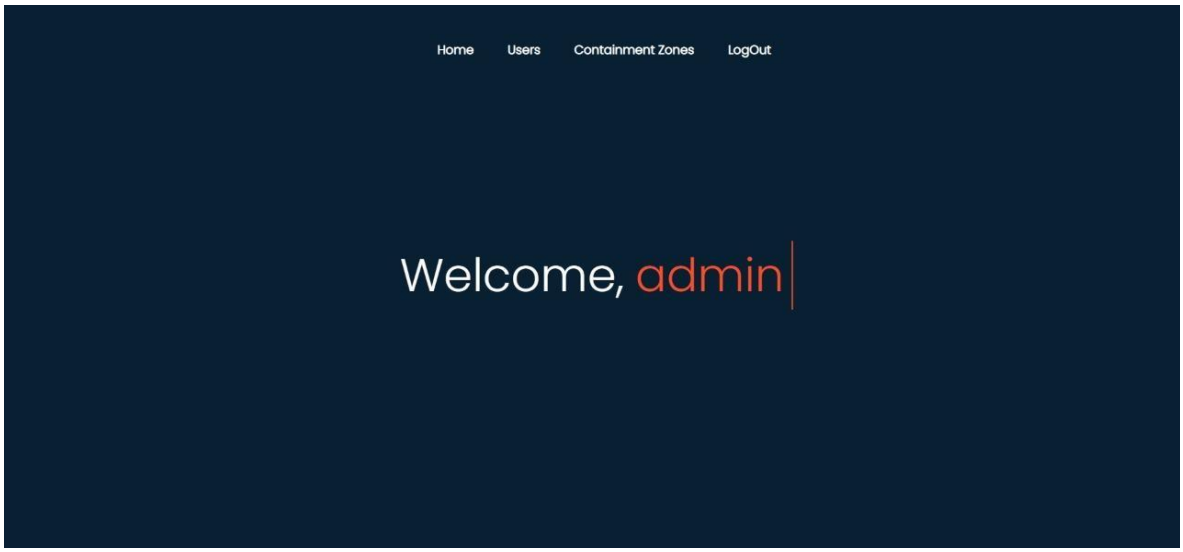

Fig. 9.7. Admin Welcome Screen

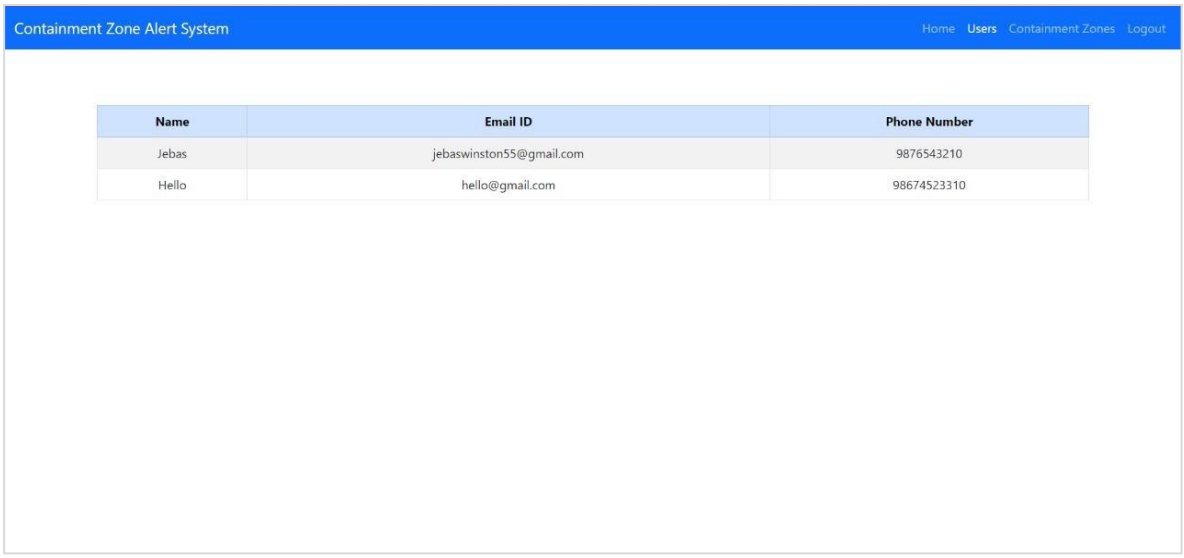

Fig. 9.8. User Details View

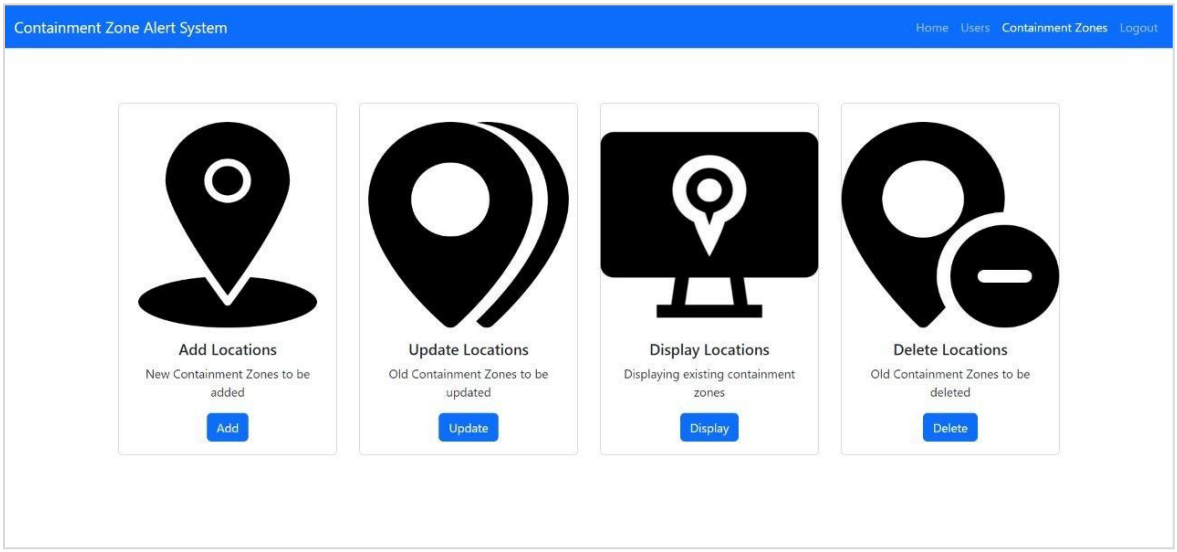

Fig. 9.9. Admin Dashboard

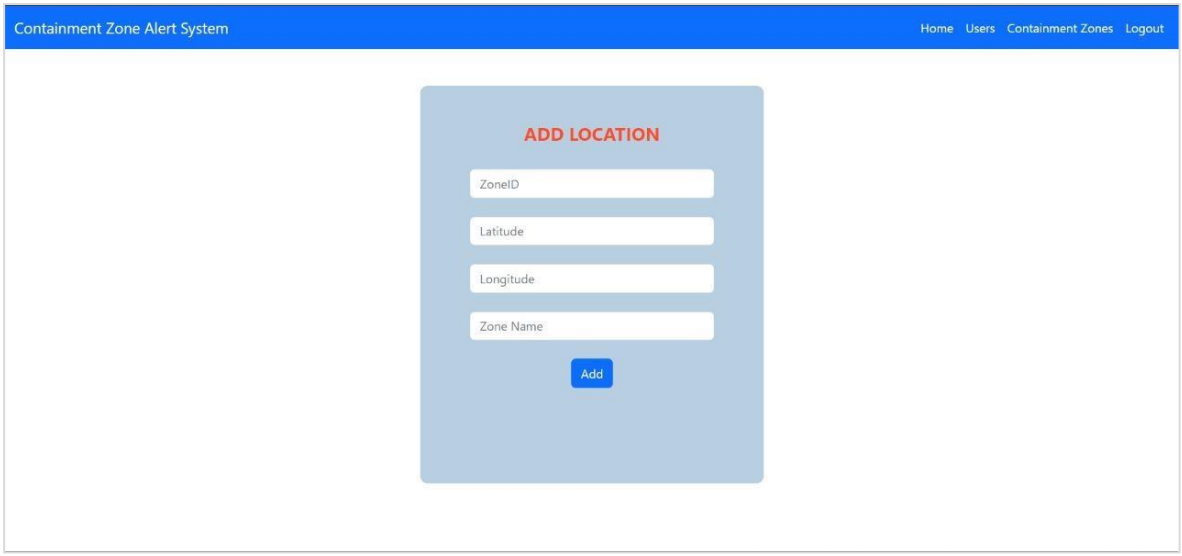

Fig. 9.10. Adding a Containment Zone

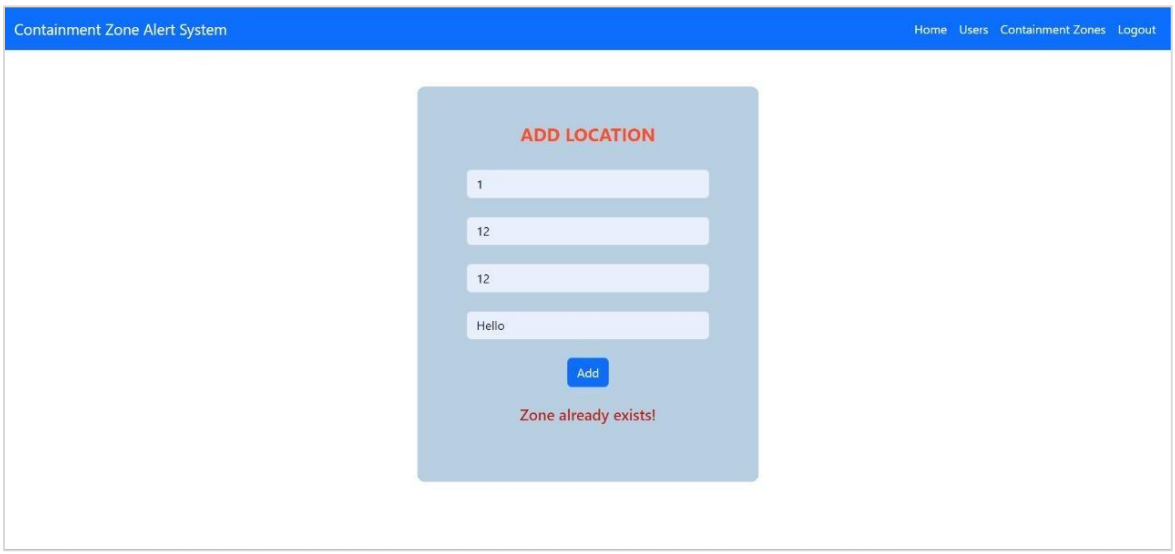

Fig. 9.11. Validation of Zone Addition

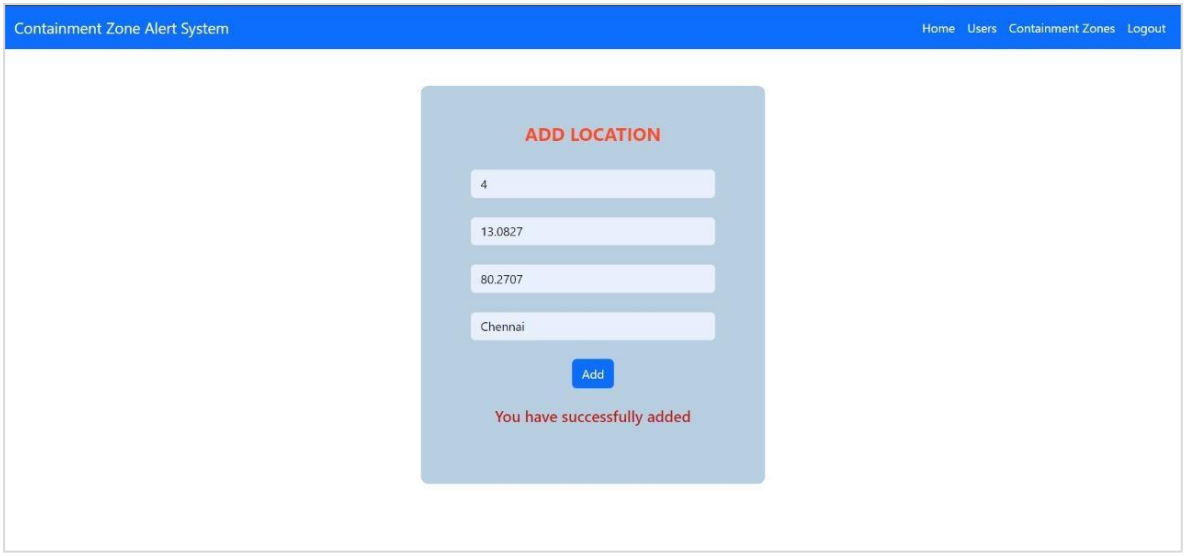

Fig. 9.12. Zone Addition Successful

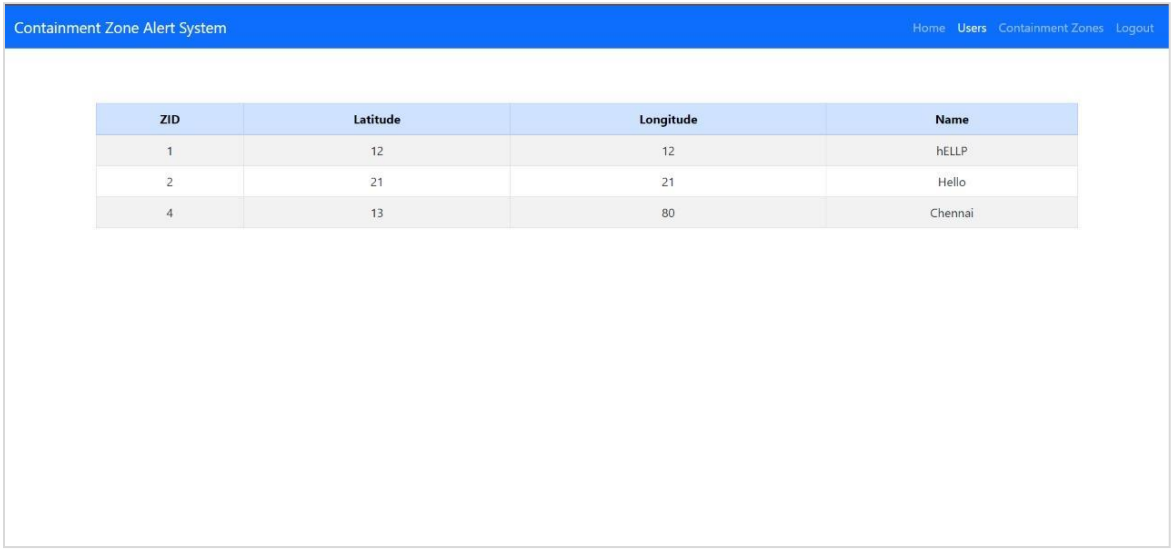

Fig. 9.13. Viewing Zone Details

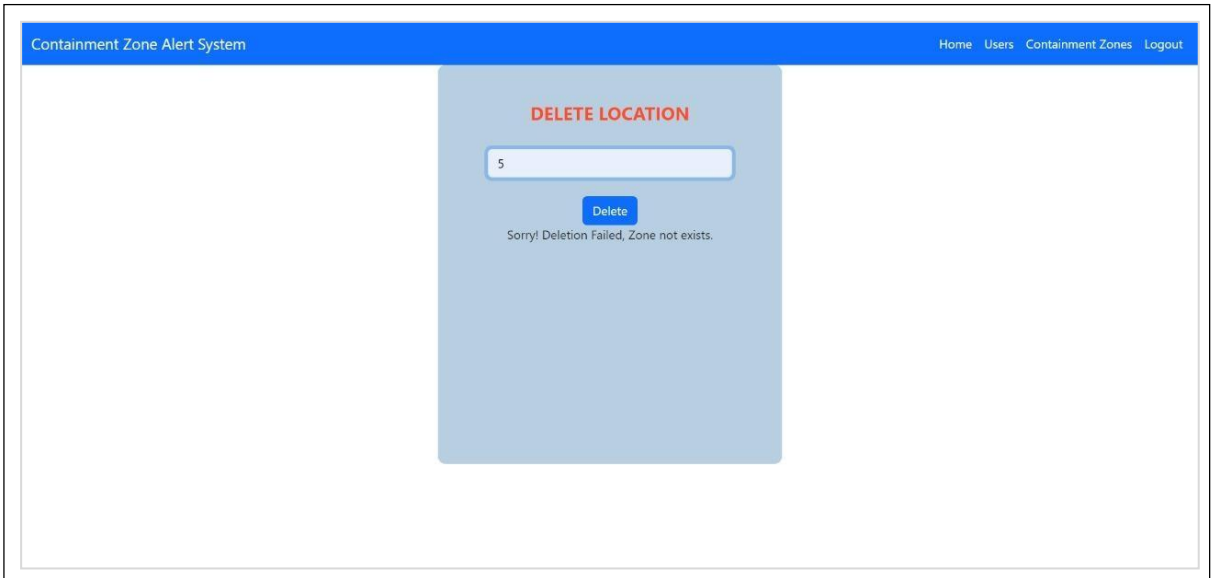

Fig. 9.14. Validation of Zone Deletion

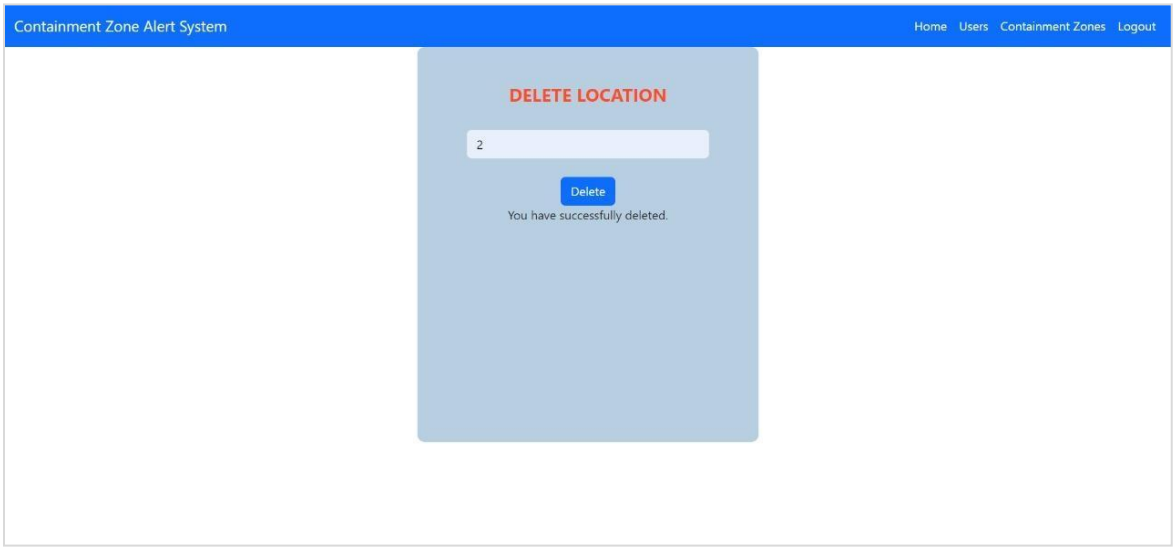

Fig. 9.15. Zone Deletion Successful

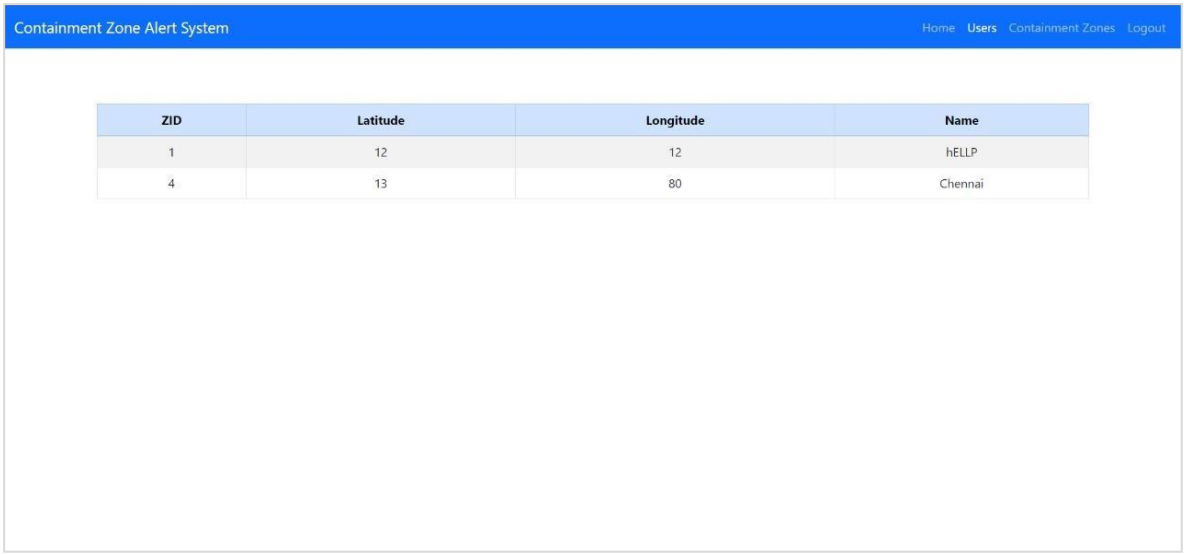

Fig. 9.16. Zone removed from table

10. ADVANTAGES & DISADVANTAGES

10.1. Advantages

The advantages of the project are given out as follows

- The application aims at warning the users before entering a Covid containment zone, thereby providing a chance for precautionary measures.
- The application solves user confusions on information about containment zones.
- The user need not worry if to enter a zone or not due to the lack of information on Covid Containment Zones.
- The app monitors the user's location at real time.
- Even if the application is closed, the application is programmed to run in the background.
- User privacy is maintained. Very little information on the user is collected.
- Not only the application delivers instant alerts, it also emails the alert notification.
- Easy administration of the users and zones through a separate portal.

10.2. Disadvantages

Every solution comes with its own disadvantages. Rather than terming them as disadvantages, they could be known as "limitations". Some of the limitations of the application are given below.

- The application is useful for the user only if the user carries his/her mobile phone while traveling.
- The application requires the user to have access to Internet.
- The location services in the user's smartphone should be turned on for the application to monitor the real time location of the user and alert, in case if the user enters a containment zone.

11. CONCLUSION

Users of the application can quickly view the designated Covid-19 containment zones on a Google map. With the worrisome rise in Covid-19-affected cases around the globe, this developed application might be used as a tool to raise more public awareness. This application keeps track of the user's location and determines if it appears on the list of designated containment zones. If a user enters or is within a Covid Containment Zone, an immediate alert

notification is given to the user. Additionally, emails are provided to the user as distinct notification alerts. To sum up, this application delineates the containment areas and emphasizes the necessity for additional precautions to be taken in the fight against COVID-19. The application has been tested in various locations and has been found to yield accurate results.

12. FUTURE SCOPE

The application has a broad scope of improvements in the future. The location details of the places visited could be analyzed with the help of Machine Learning to discover interesting patterns. If the application is teamed up with a government authority, the authenticity of the application will spike. Furthermore, future improvements to the application could include measures to send out an emergency signal in case if GPS signal or connectivity to the Internet is lost and to improve more precise tracking of user location.

13. APPENDIX

Source Code

The source code of the application could be found in chapter, and the same has been made available in the GitHub repository IBM-EBPL/ IBM-Project-49579-1660827337.

Project Demo Link

A demonstration video illustrating the working of the project has been made available nested inside the “Final Deliverables” folder in the GitHub repository IBM-EBPL/ IBM-Project- 49579-1660827337.