

CONTAINMENT ZONE ALERT SYSTEM

PROJECT BASED LEARNING

Submitted by

NAVEEN U (191001053)

LOGESH G (191001041)

NAVEEN N (191001052)

NAVEEN KUMAR G (191001051)

Bachelor of Technology

In

Information Technology

Table of Content

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	
	1.1 PROJECT OVERVIEW	4
	1.2 PURPOSE	5
2	LITERATURE SURVEY	
	2.1 EXISTING SYSTEM	6
	2.2 REFERENCES	8
	2.3 PROBLEM STATEMENT DEFINITION	8
3	IDEATION & PROPOSED SOLUTION	
	3.1 EMPATHY MAP CANVAS	9
	3.2 IDEATION & BRAINSTROMING	9
	3.3 PROPOSED SOLUTION	10
	3.4 PROBLEM SOLUTION FIT	10
4	REQUIREMENT ANALYSIS	
	4.1 FUNCTIONAL REQUIREMENT	11
	4.2 NON – FUNCTIONAL REQUIREMENT	11
5	PROJECT DESIGN	
	5.1 DATA FLOW DIAGRAMS	12
	5.2 ARCHITECTURES	12
	5.2.1 SOLUTION ARCHITECTURE	13
	5.2.2 TECHNICAL ARCHITECTURE	14
	5.3 USER STORIES	14

6	PROJECT PLANNING AND SCHEDULING	
	6.1 SPRINT PLANNING & ESTIMATION	15
	6.2 SPRINT DELIVERY SCHEDULE	15
	6.3 REPORTS FROM JIRA	16
7	ADVANTAGES & DISADVANTAGES	
	7.1 ADVANTAGES	17
	7.2 DISADVANTAGES	17
8	SCREENSHOTS	18
9	CONCLUSION	33
10	FUTURE SCOPE	34
11	APPENDIX	
	11.1 SOURCE CODE	35
	11.2 GITHUB/PROJECT DEMO LINK	79

CHAPTER 1

1. INTRODUCTION

The World Health Organization has classified the Covid-19 coronavirus outbreak as a global pandemic. Lockdown and awareness (social distance, wearing of masks, etc.) among individuals are found to be the only ways to stop the community spread of this disease given the worrisome increase in affected cases around the world. Without widespread public awareness and proactive actions taken by the populace, it is exceedingly challenging to stop the communal transmission even during a lockdown in a highly populated nation like India. Recently, red, orange, and green zones were established for a number of containment zones spread out around the nation. The red zones represent infection hotspots, the orange zones represent some infection, and the green zones represent an infection-free environment. This essay mostly examines

1.1 Project Overview

Coronaviruses are large group of viruses that cause illness in humans and animals. Rarely, animal coronaviruses can evolve and infect people and then spread between people such as has been seen with MERS and SARS. Although most human coronavirus infections are mild, the epidemics of the severe acute respiratory syndrome coronavirus (SARS-CoV) and Middle East respiratory syndrome coronavirus (MERS-CoV), have caused more than 10,000 cumulative cases in the past two decades, with mortality rates of 10% for SARS-CoV and 37% for MERS-CoV. The outbreak of Novel coronavirus disease (COVID-19) was initially noticed in a seafood market in Wuhan city in Hubei Province of China in mid-December, 2019, has now spread to 214 countries/territories/areas worldwide.

Currently there are several research works undergoing in the country to prevent Covid-19 cases from rising. Previously our country was importing medical kits like PPE (Personal Protection Kits), mask from outside, but now it has been successful in developing these kits. Along with taking initiatives to fight this disease, our country has also taken steps to make people aware of the disease. The news and media have a great part in creating this awareness by informing the public about the preventive measures that can keep them away from infection. Awareness among the people to carry out all the preventive measures can immensely help to reduce spread of the virus. The country has created containment zones throughout the cities wherever Covid-19 cases have been reported to prevent further spread of the virus. These containment zones have been kept isolated from the outside public to ensure no contamination occurs outside.

After more than 2 months of the lockdown, the government has relaxed some of the lockdown rules and has permitted reopening of government offices, bus and other road transportation facilities and shopping markets. People can move inside the city for work and other purposes. But the

containment zones are still being kept isolated, and new containment zones are being formed wherever Covid-19 cases have been reported. These zones are highly contagious as droplets with virus coughed out from an unscreened asymptomatic patient can travel up to 8 m (Bahletal. [2020](#)). Though these containment zones are guarded by policemen, still there remains a chance that people might unknowingly step into them. In this situation where people can move in the city, these containment zones pose a risk of infection to these city dwellers. Therefore, informing people about the location of the containment zones can help them bypass and avoid these zones and thereby reduce the chance of community transmission.

1.2 Purpose

This app is designed to help organisations (including the Government of Meghalaya) to maintain accountability and responsibility towards members and society. The app accomplishes this by monitoring the geographical movements of members and ensuring they are following proper work from home protocol and social distancing policies set by the organisation. Data will not be used for any purpose other than the safety of the members. Members have the right to activate/inactivate location as per their discretion. This app sends coordinates to the server if the user activates location. Users can check in at their home location and will be alerted if they leave the region around home location. Administrator/support cell will also get the list of users who are within the circle or outside the circle. Only authorized admin can access the backend services for the purpose of safety of registered users. App provides an option for the user to recheck in at a new location with the approval from administrator/Unit manager via OTP. App provides more information like emergency contact.

CHAPTER 2

Use-case: Containment Zone Alerting Application

Literature Survey:

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
1	Development of an Android Application for containment Zones and monitoring violators who are trespassing into it using firebase and Geo-fencing	To develop a mobile based Application to provide information regarding the Covid-19 containment in West Bengal	Developed on Android SDK and uses firebase Cloud fire-store.	Cloud Application	It provides an efficient way of showing the identified Covid-19 Zone
2	Applications of digital technology in COVID-19 pandemic planning and response(2020)	This Viewpoint provides a framework for the application of digital technologies in pandemic management and response.	Artificial intelligence; digital thermometers; mobile phone applications; thermal cameras; web-based toolkits. Advantages Allows visual depiction of spread; directs border restrictions; guides resource allocation; informs forecasts.	Cloud Application	Could breach privacy; involves high costs; requires management and regulation
3	Development of an Android Application for viewing Covid-19 containment Zones and Monitoring violators who and trespassing into It using firebase and Geo-fencing	This article is mostly about developing an Android app that informs individuals about Covid-19 Containment zones	It is based on Geo-fence and MC technologies	Cloud Application	It can be easily monitored & tracked It is not effective for people not having mobile phone
4	Development of an Android Application for viewing Covid-19 containment zones and monitoring violators who are trespassing into it using firebase and Geo-fencing	In this paper, We focus on developing a mobile based application to provide information regarding the Covid-19 containment zone	Fire base, & Geo-fencing API are used in this Application	Cloud Application	It tracker the user location & check whether it presented in list of containment zone

2.1 Existing System

Aarogya Setu:

Aarogya Setu is a mobile application developed by the Government of India to connect essential health services with the people of India in our combined fight against COVID-19. The app is aimed at augmenting the initiatives of the Government of India, particularly the Department of Health, in proactively reaching out to and informing the users of the app regarding risks, best practices and relevant advisories pertaining to the containment of COVID-19 (AarogyaSetu 2020)

CoBuddy-Covid19 tool:

CoBuddy-Covid 19 Coronavirus Help Tool-to help stop the spread of Covid 19, get info and help from the Government. The app makes sure that the people quarantined are within their location, communicate directly with them, provide information, and receive alerts if the quarantined are in need of any help. Location tracking and user verification with heat-maps, communication management, notifications and alerts, health tracking and feedback, essential operations management (CoBuddy-Covid19 tool 2020).

CORONTINE:

This app is designed to help organizations (including the Government of Meghalaya) to maintain accountability and responsibility towards members and society. The app accomplishes this by monitoring the geographical movements of members and ensuring they are following proper work from home protocol and social distancing policies set by the organization. Data will not be used for any purpose other than the safety of the members. Members have the right to activate/inactivate location as per their discretion. This app sends coordinates to the server if the user activates location. Users can check in at their home location and will be alerted if they leave the region around home location. Administrator/support cell will also get the list of users who are within the circle or outside the circle. Only authorized admin can access the backend services for the purpose of safety of registered users. App provides an option for the user to recheck in at a new location with the approval from administrator/Unit manager via OTP. App provides more information like emergency contact numbers and similar important information for the users to access in a short time at the hour of need (CORONTINE 2020).

2.2 References

1. Gkelios,S., Sophokleous (2022) Application for Covid-19 Real Time Counter.
2. Clemencia Siro (2021) S-Nav: Safety Aware IoT Navigation Tool for Avoiding COVID-19 Hotspots.
3. Oishik Chatterjee (2020) Tracking the Covid zones through geofencing technique.
4. Le Wu, Xiangnan (2020) Learning fashion compatibility across categories with deep multimodalneural networks
5. Oishik Chatterjee (2020) Tracking the Covid zones through geofencing technique.
6. Le Wu, Xiangnan (2020) Learning fashion compatibility across categories with deep multimodalneural networks

2.3 Problem Statement Definition:

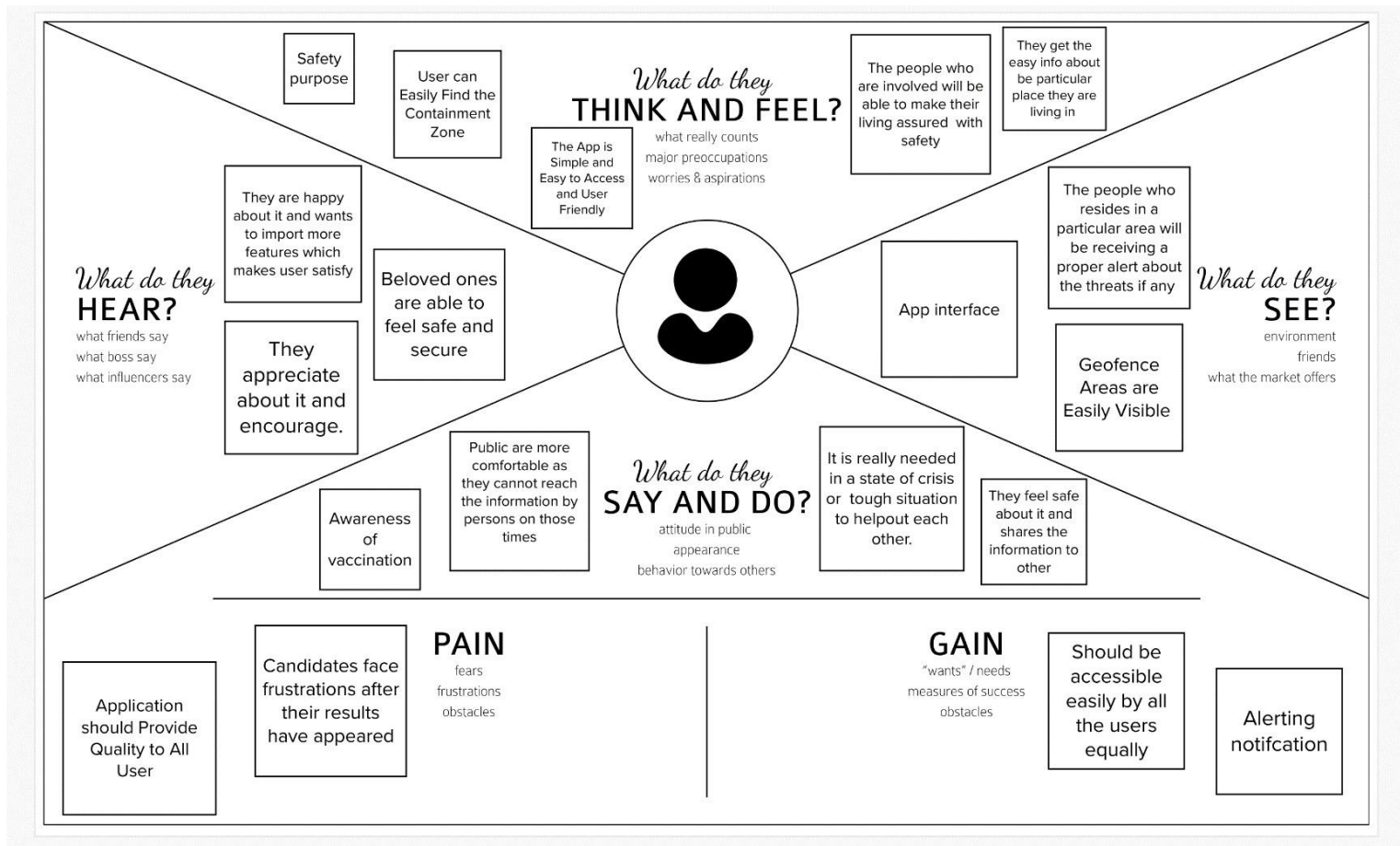


Problem Statement	I am	I'm trying to	But	Because	Which makes me feel
1	Traveler	Evade the containment zones	I accidentally step into the containment zones	I don't have any medium to alert when I enter the containment zones	I'm vulnerable
2	E-commerce delivery person	Not delivering E-commerce products inside containment zones	I don't know where are the containment zones	I don't have any resource to notify me when I enter into a containment zone	I feel fear when delivering because of the feeling of knowing about the containment zones

CHAPTER 3

3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:



3.2 Ideation & Brainstorming:

Ideation:

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.

Brainstorming:

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

 5 minutes

PROBLEM

To Alert the user When the
user Entering near to
containment Zone



Key rules of brainstorming

To run an smooth and productive session



Stay in topic.



r



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

Naveen U

safety

lifesaver

import
medicine

Vaccination
place

Zone alert

easily
accessible

safety app

Awareness

Protector

lockdown

hospital
bed

user friendly

Analysis

Unsafe zone

Alert

vaccination
time

find corona
ward

Covid
updates

Naveen N

Safe zone

lifeguard

vaccination

Sanitizer

e-passa

quarantine

user
information

Treatment

Mask

covid
variants

safety

ambulance

lockdown

Vaccination

financial loss

covazin

pandemic

people health

Logesh G

Naveen Kumar G

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Zone Identification

Finding the containment zone based on the location

Using information from hospital and take data analysis and allocate zone

Contaminant Zone Shown in Google Map

Notify on visited zones removed from containment

Decentralized zone information

To Make the User to Easily access we can provide Covid Statistics on a Bottom Sheet in Google Map

Alert User

Contaminant Zone Shown in Google Map

By using other user information weather he have corona or not and depends on zone

Closest 50 containment zone to the user are set with geofences

Provides notification alert if the user has entered a containment zone

24/7 Monitoring of affected zone and alerting it

By Sending SMS the user can be alerted

User Tracking

Android 's geofencing client is used to create geofences around the containment zones.

By GPS

Cellular triangulation for tracking

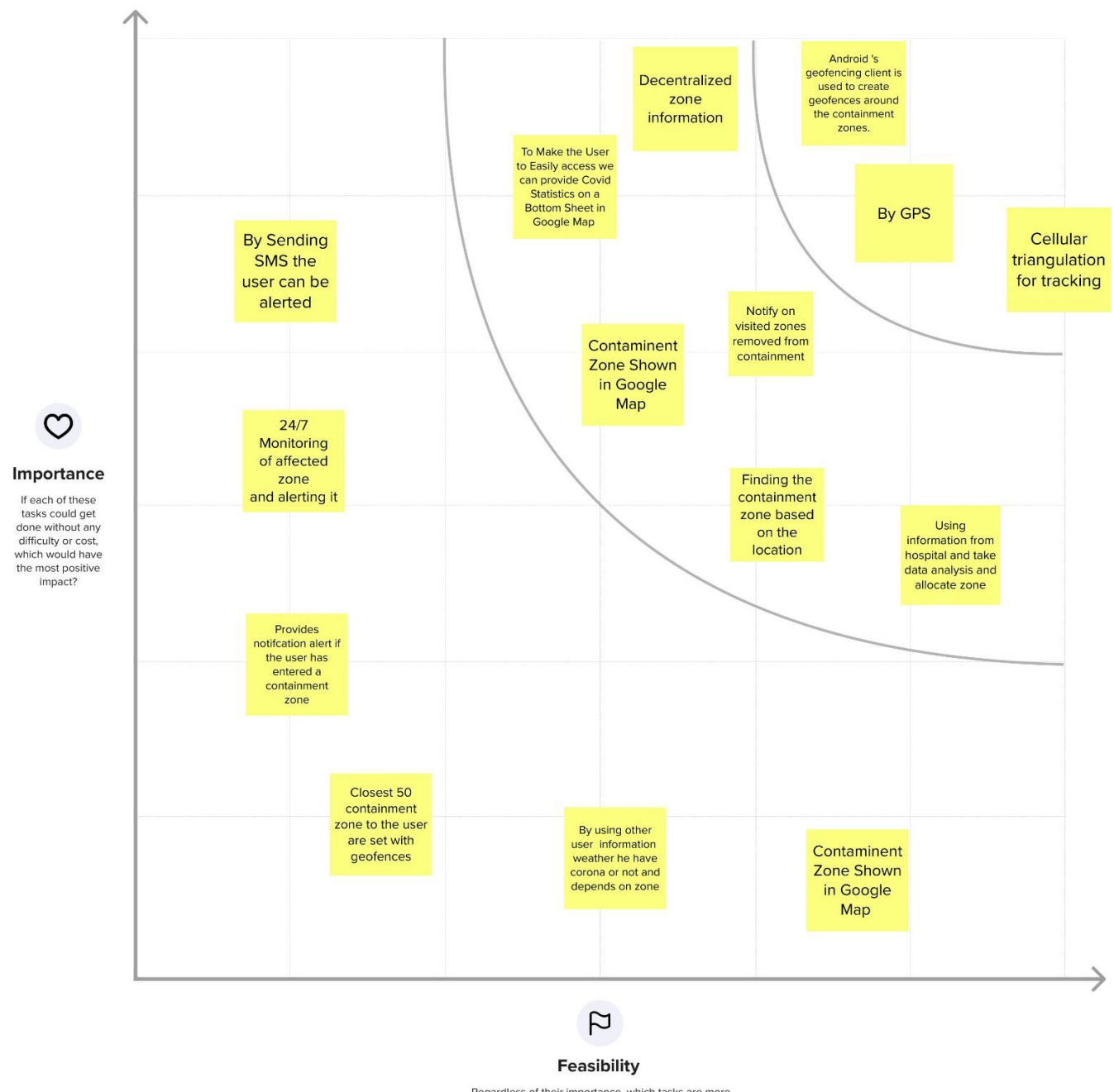
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3 Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Development of an android application for viewing the covid containment zones and also alerting the users not to enter the affected area using cloud and geofencing by sending notification.
2.	Idea / Solution description	To create an easy-to-use android application to alert the user when they enter a Containment Zone. To provide accurate results and alerting at the exact time when they enter the zone. This is done with the help of integration of Google Maps.
3.	Novelty / Uniqueness	<ul style="list-style-type: none">• Development of an Android application is necessary which can inform people of the Covid-19 containment zones and prevent trespassing into these zones.• 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 and upload the details of individual in online database.
4.	Social Impact / Customer Satisfaction	The application saves people's life from restricting them from entering the Containment zone which saves them from catching the disease. Also shows precautionary measures when they entered the zones
5.	Business Model (Revenue Model)	Can tie up with people with normal and premium charges. <ul style="list-style-type: none">• The data that is derived can be used in Government sectors.• Can tie up the Government and get profit through that.
6.	Scalability of the Solution	The application will be useful for all people from saving their life's from catching the disease by alerting them when they accidentally entered the containment zone.

Define CS, fit into CC 1 RC	1. CUSTOMER SEGMENT(S) CS	6. CUSTOMER CONSTRAINTS CC	5. AVAILABLE SOLUTIONS AS	Explore AS, differentiate TR & EM
	3.4 Problem Solution <			

CHAPTER 4

3 REQUIREMENT ANALYSIS

3.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	User can register through Email id or current phone Number.
FR-2	User Confirmation	Confirmation can be done by verification code through Mail or OTP.
FR-3	Track the location	Trace the trespassers by using Google map API.
FR-4	Affected areas are shown	Containment zones were marked and trespassers are Indicated by geofencing.
FR-5	Alert notification	By tracking their location using GPS system, notification or message will be send if the user enters the Containment zone.

3.2 Non-functional Requirements:

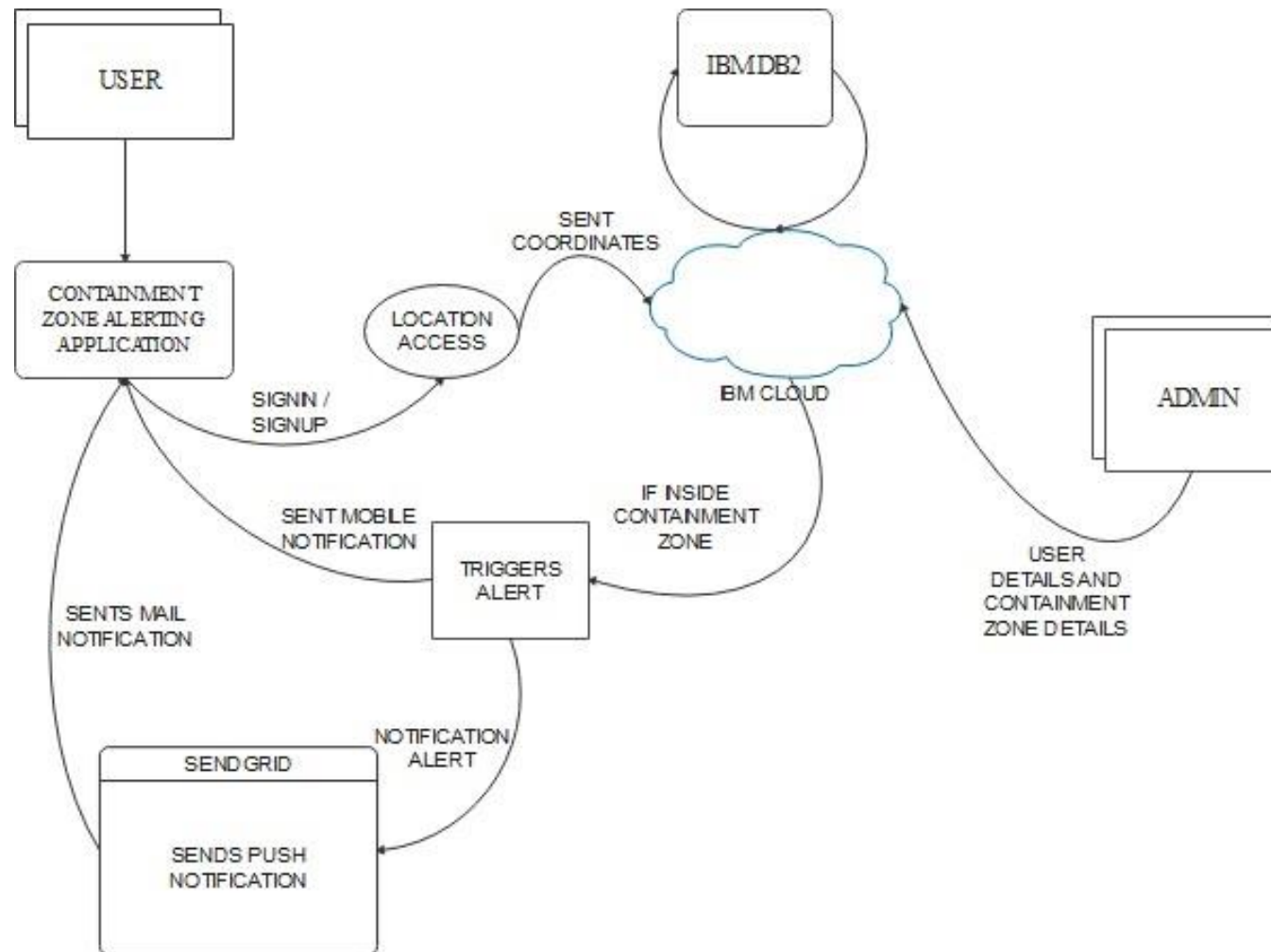
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User interface is very effective to use when compared With other.
NFR-2	Security	Data from the user will be secured properly.
NFR-3	Reliability	User can trust this application and travel safely.
NFR-4	Performance	Most appropriate results can be achieved due to using the Geofencing and GPS.
NFR-5	Availability	The application uses the network to load the google Maps to retrieve containment zones. It is available for good range of network bandwidth.
NFR-6	Scalability	This application can be accessed from anyplace and Information about the zones are up to date.

CHAPTER 5

5 PROJECT DESIGN

5.1 Data Flow Diagrams:



5.2 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail	I can register & access the dashboard with Google	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password	I can see the homepage	High	Sprint-1
	Dashboard	USN – 5	As a user, I can see the options available for User account.	I can see the dashboard	Medium	Sprint – 2
	Background running	USN – 6	As a user, I allow the app to run in background.	I should change the app settings to run app in background	High	Sprint – 2
	GPS	USN – 7	As a user, I allow the app to access my location.	I should accept the permission to access my location	High	Sprint - 2
	Google Maps	USN – 8	As a user, I can see the containment zones using the maps via Google Maps.	I should accept location permission	High	Sprint – 3
	Notification	USN – 9	As a user, I allow notification access for the application.	I should allow notification access	High	Sprint – 3
Administrator	Login	USN - 1	As admin, I log into the administrator portal.	I can access the admin account.	High	Sprint – 1
	Cloud	USN – 2	As admin, I use the cloud services to maintain users and the contaminated zones data.	I work with cloud services	High	Sprint – 2
	Cloud Database	USN – 3	As admin, I store the user details in the cloud database.	I get the details of the user and store in the cloud database.	High	Sprint – 2
	Maps	USN – 4	As admin, I will enter the containment zone's location.	I should enter correct coordinates of containment zones	High	Sprint – 3
	Mail	USN – 5	As admin, I set up a mail system to alert users when they enter a containment zone.	I use online mail system to send mail to users.	High	Sprint – 3
	Updating	USN – 6	As admin, I should frequently update the details and the location of the containment zones.	I fetch data from internet and update the zones and the relevant details.	High	Sprint – 4

3 PROJECT PLANNING & SCHEDULING

3.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	User: I can register for the application by entering my email, password and verifying password.	3	High	LOGESH
		USN-2	User: I will receive a confirmation email once I have registered for the application.	2	High	LOGESH
		USN-3	User: I can register for the application through Gmail.	5	Medium	NAVEEN U
		USN-4	Management: I need to register my hospitals on the site.	2	High	NAVEEN KUMAR G
	Login	USN-5	User: I can log into the application by entering my email & password	3	High	LOGESH
		USN-6	Management: I need to login into my dashboard with my given hospital id and password.	5	Medium	NAVEEN
	Dashboard	USN-7	User: I need to give permission to access my Contacts, Location, and Storage	5	High	LOGESH
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
		USN-8	User: I get access to the dashboard which shows a map with containment zones	5	High	NAVEEN U, NAVEEN KUMAR G

Sprint-2		USN-9	Management: I need to enter the case information of the patient that visits our hospital.	5	High	LOGESH G, NAVEEN N
	Services	USN-10	Admin: I need to provide valid information about the pandemic out there.	5	High	NAVEEN U, NAVEEN KUMAR G
Sprint-3	Dashboard	USN-11	Management: I need to store all the patient information on the cloud.	5	High	LOGESH G, NAVEEN N
	Services	USN-12	Admin: I need to provide medical advice through a chatbot.	5	Medium	LOGESH G, NAVEEN N
		USN-13	Admin: I need to provide medical recommendations by collaborating with top hospitals.	5	Low	LOGESH G, NAVEEN N
		USN-14	Admin: I need to provide preventive measures when they travel through it.	5	High	NAVEEN U, NAVEEN KUMAR G
Sprint-4	Registration	USN-15	User: I can register for the application through Facebook.	2	Low	LOGESH G, NAVEEN N
		USN-16	User: I can register for the application through Twitter.	2	Low	NAVEEN U, NAVEEN KUMAR G
	Services	USN-17	Admin: I need to alert the user when they enter pandemic zones.	3	Medium	SARAVANAN VENKTESH
		USN-18	Admin: I need to provide special services for premium users by giving services like monitoring health by their smart bands.	3	Low	NAVEEN U, NAVEEN KUMAR G
	Data Collection	USN-19	Admin: I need to store all the user information on the cloud	5	Medium	LOGESH G, NAVEEN N
		USN-20	Admin: I need to collect the recent list of diseases in the world	5	Low	NAVEEN U, NAVEEN KUMAR G

3.1 Sprint Delivery Schedule:

- ✓ The product owner typically determines the duration of the sprint and checks with the team to make sure it aligns with its workloads and resources.
- ✓ While there may be multiple project heads collaborating on a sprint, it's ultimately important to have one owner who oversees all aspects of sprint planning. Likewise, there should be one single schedule to avoid confusion and keep projects running according to a set plan.
- ✓ Teams often run into trouble when they create more than one schedule. This can create conflict and derail projects midway through their cycles. To ensure things stay on track, one schedule makes sense. In case you're unfamiliar, a sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.

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

- ✓ Ideally, you should create a sprint schedule early on in the development process—before you get to the planning stage.
- ✓ Of course, sprint schedules should be highly fluid in the beginning. You will almost certainly have to make changes before you wind up with a final plan that everyone agrees on.
- ✓ Even so, it's important to at least have an initial plan in place heading into a sprint planning meeting instead of coming to the table with nothing.

3.2 Reports from JIRA:

Reports in Jira help teams analyze progress on a project, track issues, manage their time, and predict future performance. They offer critical, real-time insights for Scrum, Kanban, and other agile methodologies, so that data-driven decisions can be made (the very best kind).

Content	OCT							NOV						NOV						NOV					
	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
Registration	<div></div>																								
Login	<div></div>																								
Dashboard								<div></div>																	
Service														<div></div>											
Data collection																				<div></div>					

Figure: JIRA Report

CHAPTER 7

3 ADVANTAGES & DISADVANTAGES

3.1 Advantages:

In this situation where people can move in the city, these containment zones pose a risk of infection to these city dwellers. Therefore, informing people about the location of the containment zones can help them bypass and avoid these zones and thereby reduce the chance of community transmission.

3.2 Disadvantages:

The parameters used are similar, but the exact criteria applied varies, and usually depends on local conditions. These have also evolved with time, and are under constant review. In general, containment zones are getting smaller with time as the number of cases are increasing — from entire localities, to colonies or neighbourhood, to streets and lanes, to particular buildings, and now just particular floors.

CHAPTER 8

1. DOCKER CREATION:

Creation of Namespaces:

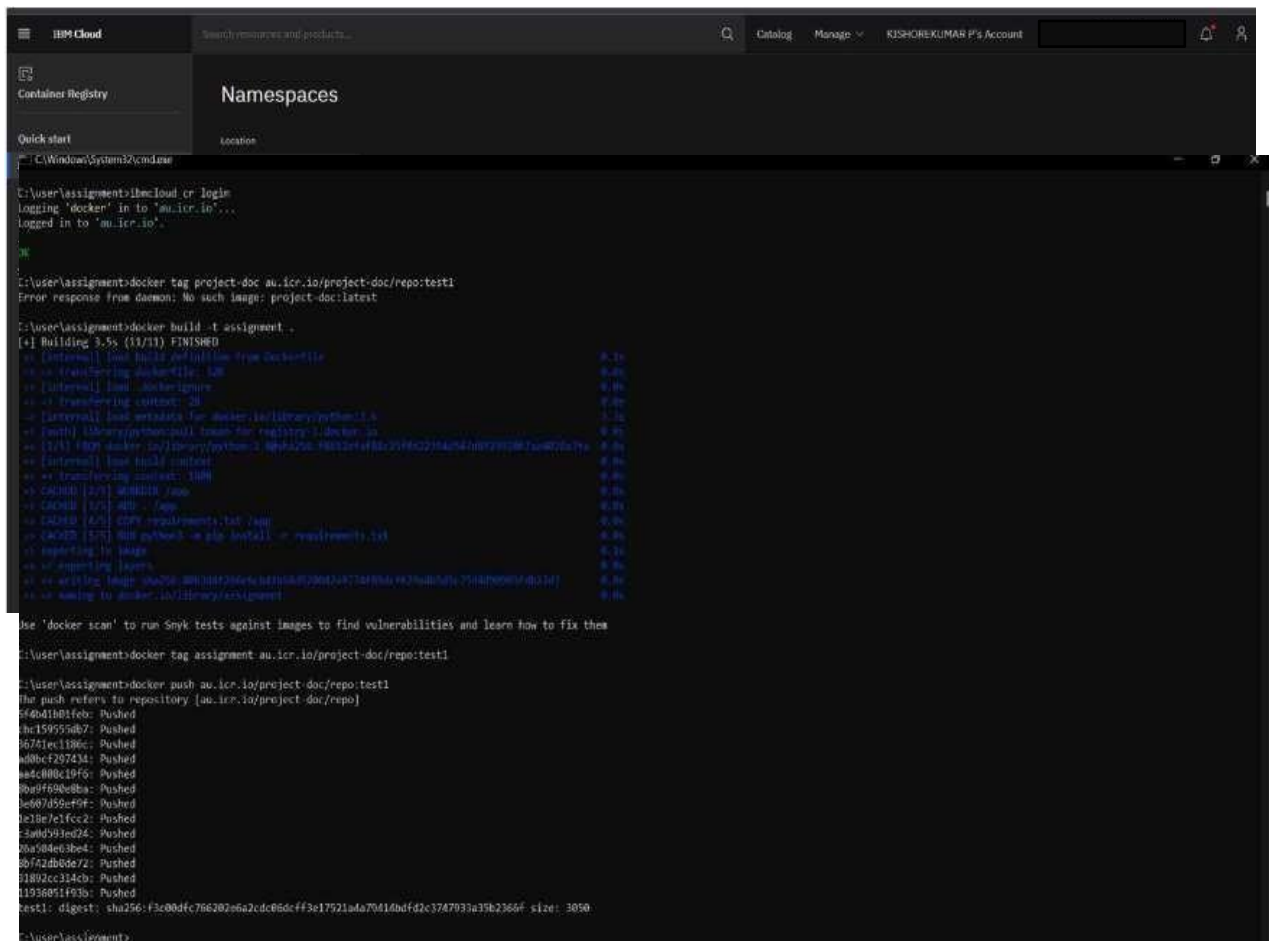


Figure: pushing image from docker

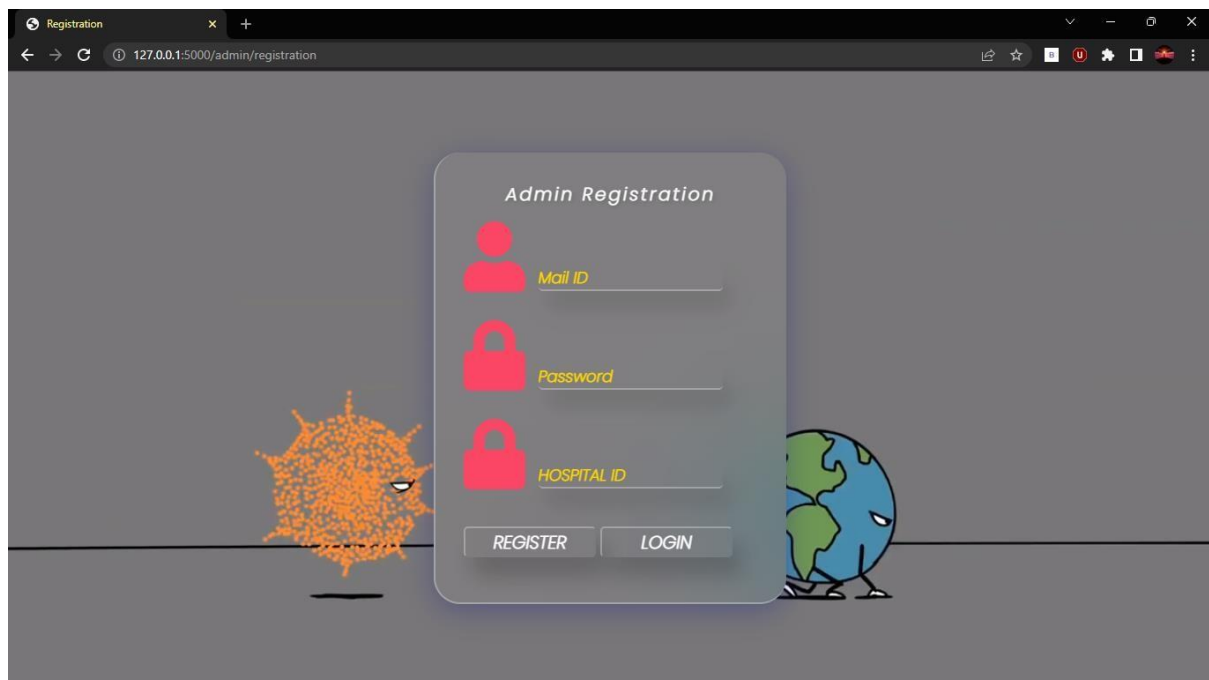
Generation of port

```
C:\Windows\System32\cmd.exe: docker run -p 8080:8080 assignment
C:\user\assignment>docker run -p 8080:8080 assignment
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5888/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 738-711-189
```

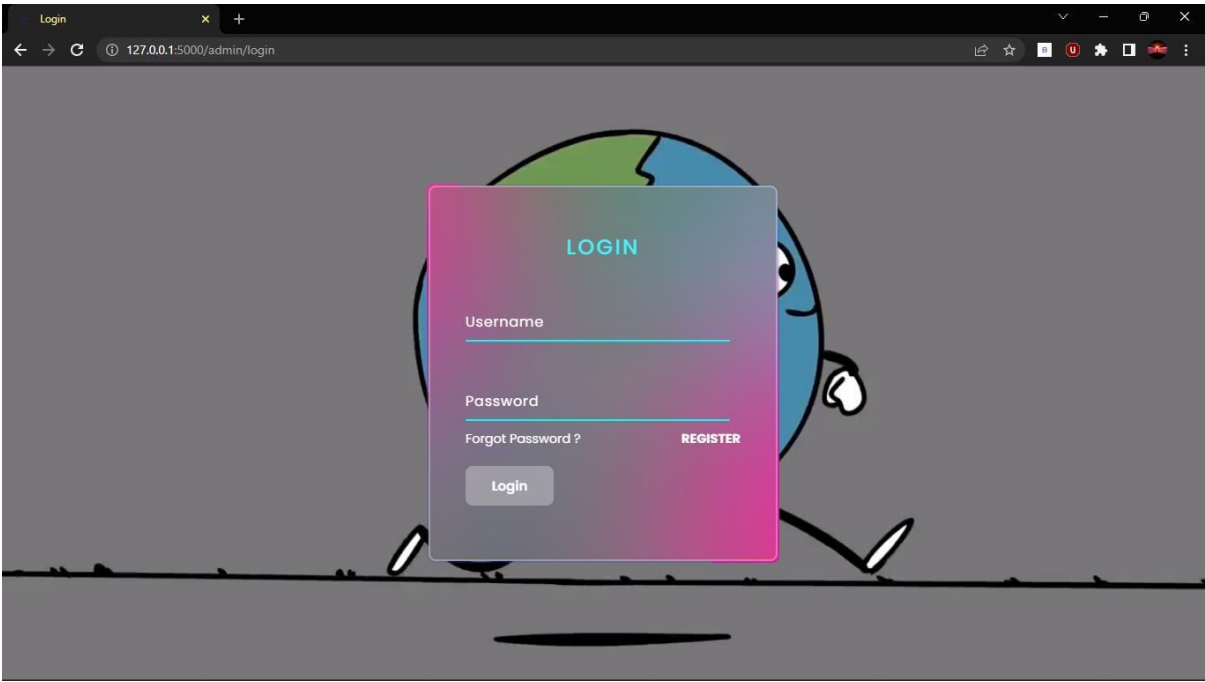
Figure: Generation of port

2. Creating Web Application


Register page:




Login Page:



Adding Containment Details:



Updating Data




Insert Data To Add Containment Zone

S.No	Name	City	Latitude	Longitude	Mail_id	Button
1	<input type="text" value="SALMANKHAN"/>	<input type="text" value="MADURAI"/>	<input type="text" value="11.023652619058344"/>	<input type="text" value="88.91169279498317"/>	<input type="text" value="venktesh2002@gmail.com"/>	<button>Delete</button>

ADD ROW

Submit



Delete Data From Containment Zone

S.No	Name	City	Latitude	Longitude	Mail_id	Button
------	------	------	----------	-----------	---------	--------

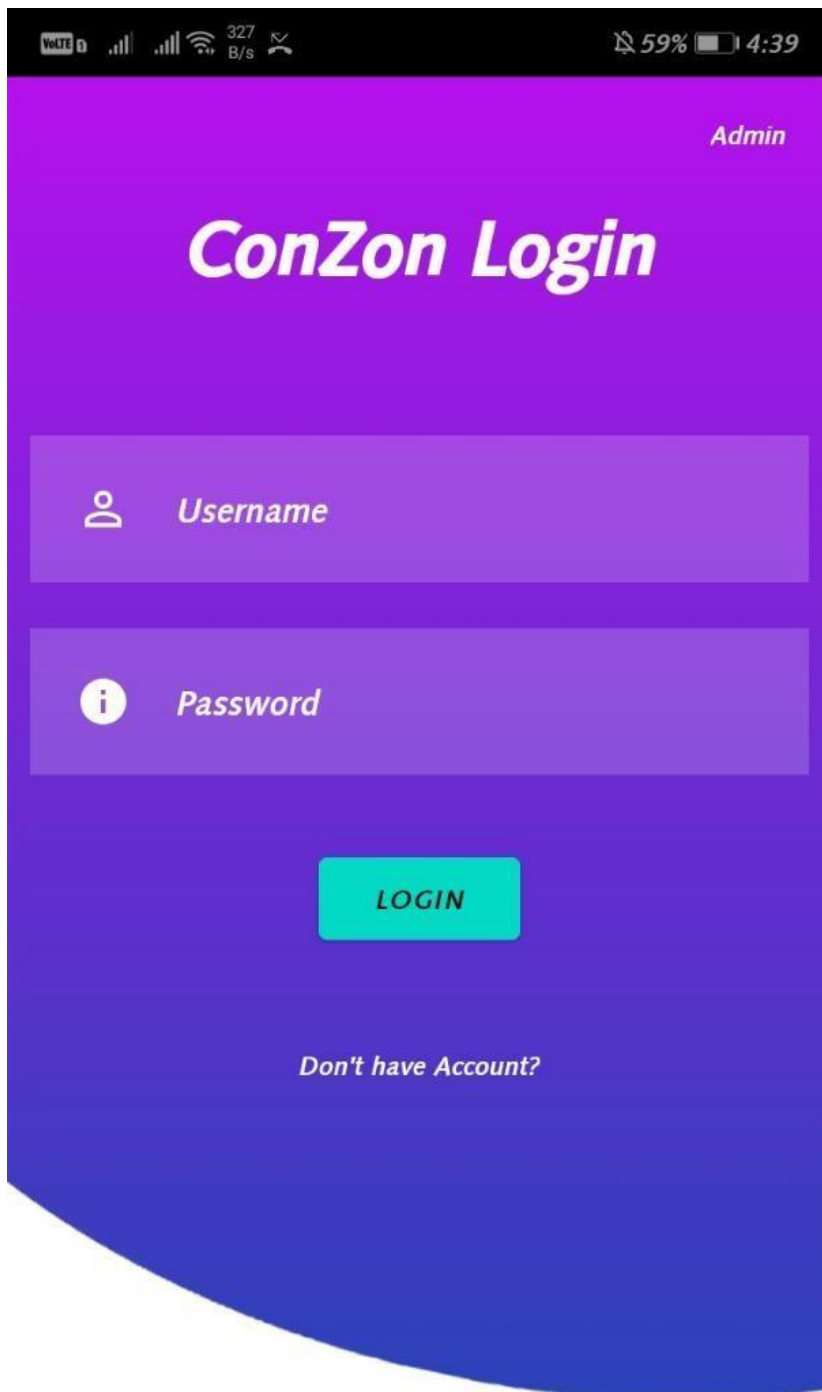
ADD ROW

Submit

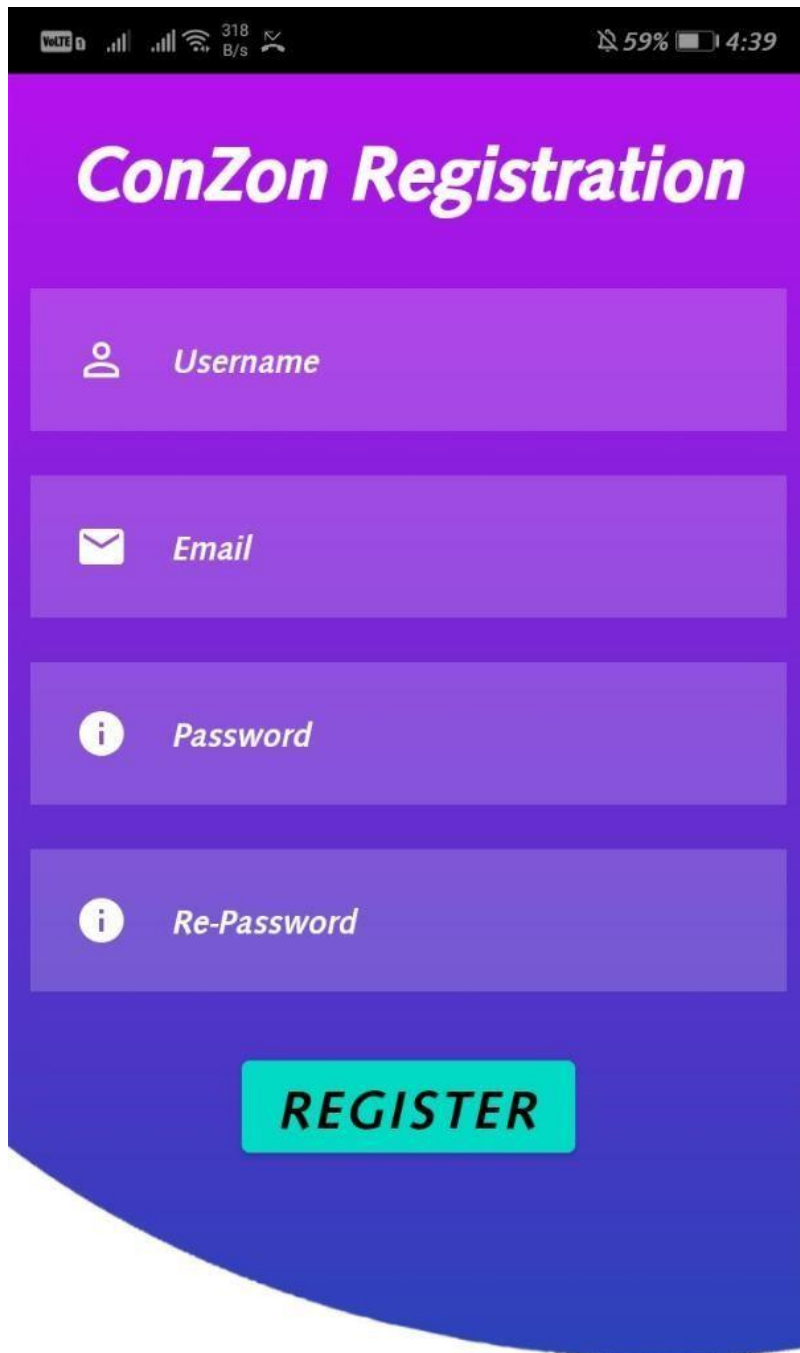
Admin Registration Conformation:

Mobile Application:

Login Page:





Registration Page:





The image shows a mobile application interface for 'ConZon Registration'. At the top is a black status bar with icons for VoLTE, signal strength, 318 B/s data speed, and a battery level of 59% at 4:39. Below the status bar is a purple header with the text 'ConZon Registration' in white. The main form area has a blue-to-purple gradient background. It contains four input fields, each with an icon on the left and a label: a person icon for 'Username', an envelope icon for 'Email', an 'i' icon for 'Password', and another 'i' icon for 'Re-Password'. At the bottom of the form is a red 'REGISTER' button. The bottom of the screen features a white background with a Google logo and a Facebook logo.

ConZon Registration

 *Username*

 *Email*

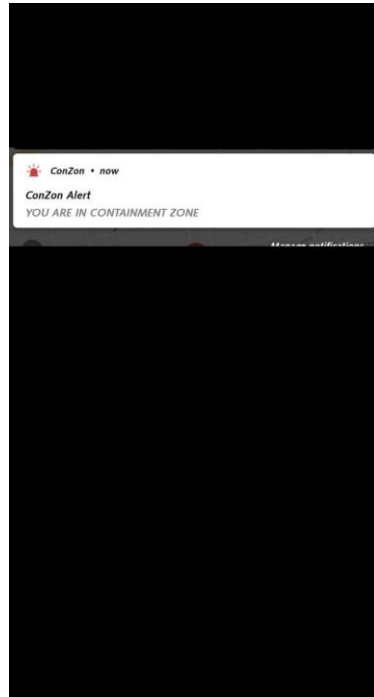
 *Password*

 *Re-Password*

REGISTER



Alert Notification



CHAPTER 9

- 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. This application further tracks the user's location and checks whether it is present in the list of identified containment zones. It sends separate notification alerts to the user on entering. The developed android application further extracts the IMEI Number of the trespasser in the containment zones which can be useful to the local police to track and identify people who are frequently trespassing the containment zones. Thereby this application identifies the containment zones and highlights the need for taking further precautionary measures for combating Covid-19. The application has been tested in various locations and has been found to yield accurate results.
- The application can be further used for many purposes like maritime and forest safety to prevent users from entering restricted areas.

CHAPTER 10

10 FUTURE SCOPE

The developed android application further extracts the IMEI Number of the trespasser in the containment zones which can be useful to the local police to track and identify people who are frequently trespassing the containment zones. Thereby this application identifies the containment zones and highlights the need for taking further precautionary measures for combating Covid-19. The application has been tested in various locations and has been found to yield accurate results

CHAPTER 11

3 APPENDIX

11.1 Source Code:

Android Application:

LoginPage.java

```
package com.example.conzon;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

import

androidx.appcompat.app.AppCompatActivity;import

androidx.work.OneTimeWorkRequest; import

androidx.work.Operation;

import androidx.work.WorkManager;

import androidx.work.WorkRequest;

import com.example.conzon.ApiRequest.ApiClient;

import com.example.conzon.backgroud.Feching_condetails;

import com.example.conzon.models.API_Response;

import com.example.conzon.models.Login_API_Request;


import retrofit2.Call;

import retrofit2.Callback;
```



```

import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    TextView textView;

    Operation work;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        workprocess();

    }

    public void workprocess() {

        WorkRequest uploadWorkRequest = new OneTimeWorkRequest.Builder (
Feching_condetails.class ) . build();

        work = WorkManager.getInstance(this).enqueue(uploadWorkRequest);

    }

    public void buttonWorkManager(View view) {

        textView = findViewById(R.id.loginbtn);

        TextView username = findViewById(R.id.username);

        TextView password = findViewById(R.id.password);

        if (username.getText().toString().isEmpty() ||
password.getText().toString().isEmpty())

            Toast.makeText(this, "Please Fill All the Details",
Toast.LENGTH_LONG).show();

        else {

            Login_API_Request loginRequest = new Login_API_Request();

```

```
loginRequest.setUsername(username.getText().toString());

loginRequest.setPassword(password.getText().toString());

Call<API_Response> loginresponseCall = ApiClient.getService().loginUser(loginRequest);

loginresponseCall.enqueue(new Callback<API_Response>() {

    @Override

    public void onResponse(Call<API_Response> call, Response<API_Response> response) {

        if (response.isSuccessful()) {

            String result = response.body().getResult();

            if (result.equals("Logged Successfully \uD83D\uDE0D")) {

                Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show();

                startActivity(new Intent(MainActivity.this, Loading_Screen.class));

            } else

                Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show();

        } else {

            String messages = "Please Try again Later. .. ";

            Toast.makeText(MainActivity.this, messages, Toast.LENGTH_LONG).show();

        }

    }

    @Override

    public void onFailure(Call<API_Response> call, Throwable t) {

        String messages = "Failed connect with server";

        Toast.makeText(MainActivity.this, messages, Toast.LENGTH_LONG).show();

    }

}
```

```

        }    });}}

    public void Register(View view) {

        textView = findViewById(R.id.Regusername);

        startActivity(new Intent(MainActivity.this, Registration.class));

    }

    public void admin(View view) {

        startActivity(new Intent(MainActivity.this, Admin_usage.class));

    }

    @Override

    public void onBackPressed() {

        android.os.Process.killProcess(android.os.Process.myPid());

        // This above line close correctly

    }}

```

Registration.java

```

package com.example.conzon;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;


import

androidx.appcompat.app.AppCompatActivity;import

androidx.work.OneTimeWorkRequest; import

androidx.work.Operation;

```

```
import androidx.work.WorkManager;

import androidx.work.WorkRequest;

import com.example.conzon.ApiRequest.ApiClient;

import com.example.conzon.backgroud.Feching_condetails;

import com.example.conzon.models.API_Response;

import com.example.conzon.models.Login_API_Request;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    TextView textView;

    Operation work;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        workprocess();

    }

    public void workprocess() {

        WorkRequest uploadWorkRequest = new OneTimeWorkRequest . Builder(
Feching_condetails.class ).build();

        work = WorkManager.getInstance(this).enqueue(uploadWorkRequest);

    }
```

```

public void buttonWorkManager(View view) {

    textView = findViewById(R.id.loginbtn);

    TextView username = findViewById(R.id.username);

    TextView password = findViewById(R.id.password);

    if (username.getText().toString().isEmpty() || password.getText().toString().isEmpty())

        Toast.makeText(this, "Please Fill All the Details", Toast.LENGTH_LONG).show();

    else {

        Login_API_Request loginRequest = new Login_API_Request();

        loginRequest.setUsername(username.getText().toString());

        loginRequest.setPassword(password.getText().toString());

        Call<API_Response> loginresponseCall =
ApiClient.getService().loginUser(loginRequest);

        loginresponseCall.enqueue(new Callback<API_Response>() {

            @Override

            public void onResponse(Call<API_Response> call, Response<API_Response>
response) {

                if (response.isSuccessful()) {

                    String result = response.body().getResult();

                    if (result.equals("Logged Successfully \uD83D\uDE0D")) {

                        Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show();

                        startActivity(new Intent(MainActivity.this, Loading_Screen.class));

                    } else

                        Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show();

```

```

    } else {

String messages = "Please Try again Later. .. ";

Toast.makeText(MainActivity.this, messages, Toast.LENGTH_LONG).show();

    } }

@Override

public void onFailure(Call<API_Response> call, Throwable t) {

String messages = "Failed connect with server";

    Toast.makeText(MainActivity.this, messages, Toast.LENGTH_LONG).show();    }));}}

public void Register(View view) {

textView = findViewById(R.id.Regusername);

startActivity(new Intent(MainActivity.this, Registration.class));}

    public void admin(View view) {

        startActivity(new Intent(MainActivity.this, Admin_usage.class));

    }

@Override

public void onBackPressed() {

    android.os.Process.killProcess(android.os.Process.myPid()); }}

```

ContainmentZone.java

```

package com.example.conzon;

import android.annotation.SuppressLint;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import          android.graphics.Color;

```

```
import android.os.Build;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.NotificationCompat;

import androidx.core.app.NotificationManagerCompat;

import androidx.work.OneTimeWorkRequest;

import androidx.work.WorkManager;

import androidx.work.WorkRequest;

import com.example.conzon.backgroud.alerting;

import com.example.conzon.backgroud.current_location;

import com.example.conzon.models.containment_API_request;

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 java.util.ArrayList;

public class ContainmentZone extends AppCompatActivity implements OnMapReadyCallback {

    private static final String CHANNEL_ID = "ContainmentZone Alert";

    private static final int NOTIFICATION_ID = 01;
```

```
@SuppressWarnings("StaticFieldLeak")

static NotificationCompat.Builder builder;

@SuppressWarnings("StaticFieldLeak")

static NotificationManagerCompat notificationManagerCompat;

static NotificationManager notificationManager;

private GoogleMap mMap;

WorkRequest uploadWorkRequest;

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_containment_zone);

    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);

    assert mapFragment != null;

    mapFragment.getMapAsync(this);

    uploadWorkRequest = new OneTimeWorkRequest.Builder(alerting.class).build();

    builder = new NotificationCompat.Builder(ContainmentZone.this, CHANNEL_ID);

    notificationManagerCompat = NotificationManagerCompat.from(this);

    notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

    WorkManager.getInstance(this).enqueue(uploadWorkRequest);

}

@Override
```



```
public void onMapReady(@NonNull GoogleMap googleMap) {

    mMap = googleMap;

    process();

}

public void process() {

    // creating array list for adding all our locations.

    ArrayList<LatLng> locationArrayList = containment_API_request.getLocationArrayList();

    LatLng current = new LatLng(current_location.getLati(), current_location.getLongi());

    for (int i = 0; i < locationArrayList.size(); i++) {

        // below line is use to add marker to each location of our array list.

        mMap.addCircle(new CircleOptions()

            .center(locationArrayList.get(i))

            .radius(20)

            .strokeColor(Color.RED)

            .fillColor(Color.YELLOW));

        // below lin is use to zoom our camera on map.

        mMap.animateCamera(CameraUpdateFactory.zoomTo(18.8f));

        mMap.setMinZoomPreference(16);

        // below line is use to move our camera to the specific location.

    }

    mMap.addMarker(new MarkerOptions().position(current));

    mMap.moveCamera(CameraUpdateFactory.newLatLng(current));

}
```

```
}
```

```
public static void showAlert(String message) {
```

```
    createNotificationChannel();
```

```
    builder.setSmallIcon(R.drawable.ic_alert);
```

```
    builder.setTitle("ConZon Alert");
```

```
    //description
```

```
    builder.setText(message);
```

```
    builder.setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

```
    notificationManagerCompat.notify(NOTIFICATION_ID, builder.build());
```

```
}
```

```
private static void createNotificationChannel() {
```

```
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
        //swipe notifications
```

```
        CharSequence name = "ConZon Alert";
```

```
        String description = "ALERTING WHEN USER ENTER INTO CONTAINMENT ZONE";
```

```
        //importance of your notification
```

```
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
```

```
        NotificationChannel notificationchannel = new NotificationChannel(CHANNEL_ID, name, importance);
```

```
        notificationchannel.setDescription(description);
```

```

        notificationManager.createNotificationChannel(notificationchannel)    }

    }

    @Override

    public void onBackPressed() {

        android.os.Process.killProcess(android.os.Process.myPid());

        // This above line close correctly

    }

}

```

Activity_main.xml

```

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:background="@drawable/background"

    tools:context=".MainActivity">

    <TextView

        android:id="@+id/signin"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_marginStart="50dp"

        android:layout_marginTop="50dp"

        android:layout_marginEnd="50dp"

```

android:layout_marginBottom="50dp"

android:gravity="center"

android:text="ConZon Login"

android:textColor="@color/white"

android:textSize="35dp"

android:textStyle="bold" />

<EditText

android:id="@+id/username"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@id/signin"

android:layout_margin="10dp"

android:background="#30ffffff"

android:drawableLeft="@drawable/ic_baseline_person_outline_24"

android:drawablePadding="20dp"

android:hint="Username"

android:padding="20dp"

android:textColor="@color/white"

android:textColorHint="@color/white" />

<EditText

android:id="@+id/password"

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/username"
android:layout_marginStart="10dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="10dp"
android:layout_marginBottom="10dp"
android:background="#30ffffff"
android:drawableLeft="@drawable/ic_baseline_info_24"
android:drawablePadding="20dp"
android:hint="Password"
android:inputType="textPassword"
android:padding="20dp"
android:textColor="@color/white"
android:textColorHint="@color/white" />
```

<Button

```
android:id="@+id/loginbtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/password"
android:layout_centerHorizontal="true"
android:layout_marginStart="20dp"
android:layout_marginTop="20dp"
```

```
android:layout_marginEnd="20dp"

android:layout_marginBottom="20dp"

android:backgroundTint="@color/design_default_color_secondary"

android:onClick="buttonWorkManager"

android:text="LOGIN"

tools:ignore="UsingOnClickInXml" />
```

<TextView

```
android:id="@+id/forgotpass"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_below="@id/loginbtn"

    android:layout_centerHorizontal="true"

    android:layout_marginStart="20dp"

    android:layout_marginTop="20dp"

    android:layout_marginEnd="20dp"

    android:layout_marginBottom="20dp"

    android:text="Don't have Account?"

    android:textColor="@color/white"

    android:onClick="Register"

    tools:ignore="UsingOnClickInXml" />
```

<TextView

```
android:id="@+id/Admin"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_below="@id/loginbtn"

android:layout_alignParentEnd="true"

android:layout_marginStart="20dp"

android:layout_marginTop="-386dp"

android:layout_marginEnd="20dp"

android:layout_marginBottom="20dp"

android:text="Admin"

android:onClick="admin"

android:textColor="@color/white"

tools:ignore="UsingOnClickInXml" />
```

<TextView

```
android:id="@+id/others"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_above="@id/socialicons"

android:layout_centerHorizontal="true"

android:text="or sign in with" />
```

<LinearLayout

```
android:id="@+id/socialicons"

android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"

android:layout_alignParentBottom="true"

android:gravity="center">
```

```
<ImageView
```

```
    android:layout_width="48dp"

    android:layout_height="48dp"

    android:layout_margin="20dp"

    android:src="@drawable/google" />
```

```
<ImageView
```

```
    android:layout_width="48dp"

    android:layout_height="48dp"

    android:layout_margin="20dp"

    android:src="@drawable/fb" />
```

```
</LinearLayout>
```

```
</RelativeLayout>
```

Registrarion.xml

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

```
<RelativeLayout 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"
```



```
android:background="@drawable/background"
```

```
tools:context=".MainActivity">
```

```
<TextView
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/signuptitle"
```

```
    android:text="ConZon Registration"
```

```
    android:textSize="35dp"
```

```
    android:textStyle="bold"
```

```
    android:textColor="@color/white"
```

```
    android:gravity="center"
```

```
    android:layout_margin="20dp"/>
```

```
<EditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/Regusername"
```

```
    android:layout_below="@id/signuptitle"
```

```
    android:background="#30ffffff"
```

```
    android:hint="Username"
```

```
    android:textColorHint="@color/white"
```

```
    android:textColor="@color/white"
```

```
    android:layout_margin="10dp"
```

android:padding="20dp"

android:drawableLeft="@drawable/ic_baseline_person_outline_24"

android:drawablePadding="20dp"/>

<EditText

android:id="@+id/email"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@id/Regusername"

android:layout_marginStart="10dp"

android:layout_marginTop="10dp"

android:layout_marginEnd="10dp"

android:layout_marginBottom="10dp"

android:background="#30ffffff"

android:drawableLeft="@drawable/ic_baseline_email_24"

android:drawablePadding="20dp"

android:hint="Email"

android:padding="20dp"

android:textColor="@color/white"

android:textColorHint="@color/white" />

<EditText

android:id="@+id/password"

android:layout_width="match_parent"

```
android:layout_height="wrap_content"
android:layout_below="@id/email"
android:layout_marginStart="10dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="10dp"
android:layout_marginBottom="10dp"
android:background="#30ffffff"
android:drawableLeft="@drawable/ic_baseline_info_24"
android:drawablePadding="20dp"
android:hint="Password"
android:inputType="textPassword"
android:padding="20dp"
android:textColor="@color/white"
android:textColorHint="@color/white" />
```

<EditText

```
android:id="@+id/repassword"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/password"
android:layout_marginStart="10dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="10dp"
```

```
android:layout_marginBottom="10dp"

android:background="#30ffffff"

android:drawableLeft="@drawable/ic_baseline_info_24"

android:drawablePadding="20dp"

android:hint="Re-Password"

android:inputType="textPassword"

android:padding="20dp"

android:textColor="@color/white"

android:textColorHint="@color/white" />
```

```
<com.google.android.material.button.MaterialButton
```

```
    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/signupbtn"

    android:text="REGISTER"

    android:onClick="Register"

    android:textSize="25dp"

    android:layout_below="@id/repassword"

    android:layout_centerHorizontal="true"

    android:backgroundTint="@color/design_default_color_secondary"

    android:layout_margin="15dp"

    tools:ignore="UsingOnClickInXml" />
```

```
<TextView
```

```
android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:id="@+id/info"

android:layout_above="@id/socialicons"

android:text="or sign up with"

android:layout_centerHorizontal="true"/>
```

```
<LinearLayout
```

```
    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:id="@+id/socialicons"

    android:gravity="center"

    android:layout_alignParentBottom="true">
```

```
<ImageView
```

```
    android:layout_width="48dp"

    android:layout_height="48dp"

    android:layout_margin="20dp"

    android:src="@drawable/google"/>
```

```
<ImageView
```

```
    android:layout_width="48dp"

    android:layout_height="48dp"

    android:layout_margin="20dp"

    android:src="@drawable/fb"/>
```

</LinearLayout>

</RelativeLayout>

WebApplication:

ConZonController.py

```
#.....importing Packages.....#

from flask import render_template, request, redirect, jsonify,
Flask
from flask_mail import Mail

import ConZon_Verification

# -----Creation Application and Some configurations ----- #

ConZon = Flask(__name_)

mails = Mail(ConZon)

ConZon.secret_key = '[_CvOeNnZkOtNeCsOhN_%]'

ConZon.config['MAIL_SERVER'] = 'smtp.gmail.com'

ConZon.config['MAIL_PORT'] = 465

ConZon.config['MAIL_USERNAME'] = '20euit511@skcet.ac.in'

ConZon.config['MAIL_PASSWORD'] = ""

ConZon.config['MAIL_USE_TLS'] = False

ConZon.config['MAIL_USE_SSL'] = True

mails = Mail(ConZon)

#.....Our WebApplication.....#

# -----Home Page----- #

@ConZon.route('/')

def home_page():
```

```

if request.cookies.get('ConZon_login') == 'True':

    print(request.cookies.get('userName'))

    return render_template('home_page.html', ConZon_user=request.cookies.get('userName'),

                           count_data=ConZon_Verification.containmentZone(),

count_data_without=ConZon_Verification.containmentZone_withoutCommon())

else:

    return redirect('/admin/login')

#.....Login Route.....-.....#

@ConZon.route('/admin/login',      methods=['POST',

'GET'])def admin_login():

    if request.method == 'GET':

        return render_template("adminLogin.html")

    elif request.method == 'POST':

        username = request.form.get('mail')

        res = ConZon_Verification.admin_login_verification(username,

request.form.get('password'))

        if res is True:

            res = redirect('/')

            print("ok")

            res.set_cookie('ConZon_login', 'True')

            res.set_cookie('userName', request.form.get('mail'))

            return res

```

```

        else:

            return render_template('adminLogin.html',
data=res)# -----Logout Route  #

@ConZon.get('/logout')

def admin_logout():

    res = redirect('/')

    res.delete_cookie('ConZon_login')

    return res

#.....Admin Registration Route.....#

@ConZon.route('/admin/registration', methods=['POST', 'GET'])

def admin_register():

    if request.method == 'GET':

        return render_template('adminRegistration.html')

    elif request.method == 'POST':

        res = ConZon_Verification.admin_register(request.form.get('mail'),
request.form.get('password'),

                                request.form.get('reqid'))

        if res is True:

            return redirect('/')

        else:

            return render_template('adminRegistration.html', data=res)

#.....Admin Dashboard Routes.....#

@ConZon.route('/display_data_add', methods=['POST'])

```



```

def display_add():
    try:
        res = ConZon_Verification.dashboard_data_process(list(request.form.listvalues()))
        if res:
            return redirect('/')
        else:
            return 'Failed'
    except(type):
        return render_template('error.html', data=type)
@ConZon.route('/delete_data', methods=['POST'])
def display_delete():
    try:
        if ConZon_Verification.dashboard_data_delete(list(request.form.listvalues())):
            return redirect('/')
        else:
            return 'Failed'
    except(type):
        return render_template('error.html', data=type)
@ConZon.route('/display_data')
def display_datas():
    return jsonify({"data":
ConZon_Verification.dashboard_data()})#
.....Mobile Application.....#
@ConZon.route('/mobile')

```

```

def sample():

    return "hello"

@ConZon.route('/User_Registration', methods=['POST'])

def UserRegistration():

    ss = request.get_data().decode()

    return '{"result":"" + ConZon_Verification.userRegprocess(ss) + ""}'

@ConZon.route('/User_login', methods=['POST'])

def UserLogin():

    ss = request.get_data().decode()

    return '{"result":"" + ConZon_Verification.userloginprocess(ss) + ""}'

@ConZon.route('/userlocation', methods=['POST'])

def userlocation():

    ss = request.get_data().decode()

    return '{"result":"" + ConZon_Verification.locationprocess(ss) + ""}'

@ConZon.route('/Con_data')

def Containment_data():

    return jsonify(ConZon_Verification.Con_details())

# .....Running Application.....#

if __name__ == '__main__':

    ConZon.run(debug=True, host="0.0.0.0")

```

ConZon_Db_connection.py

```

import ibm_db

```

```

conn = ibm_db.connect(

    "{IBM DB2 ODBC DRIVER};DATABASE=bludb;HOSTNAME=0c77d6f2-5da9-
48a9-81f8-
86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31198;SECURI
TY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=hv
s11698;PWD=xHqh4sBBGY10Ci3V","", "")

def execution(cmd):

    return ibm_db.execute(cmd)

def execution_immediate(cmd):

    return ibm_db.exec_immediate(conn, cmd)

def Prepare_db(cmd):

    return ibm_db.prepare(conn, cmd)

```

ConZon_Controller.py

```

import haversine as hs

import ibm_db

import ConZon_Mail_config

from ConZon_Db_connection import execution, Prepare_db, execution_immediate

def admin_login_verification(name, password):

    query = "Select mail_id,password from admin_profile where mail_id = ?"

    prep_stmt = Prepare_db(query)

    ibm_db.bind_param(prepare_stmt, 1, name)

    execution(prepare_stmt)

    res = ibm_db.fetch_both(prepare_stmt)

```

if res is False:

return 'Account Not Found 😞'

else:

if password == res[1]:

return True

else:

return 'Password is Incorrect 😞'

def admin_register(name, passwords, req_id):

if req_id == '2002':

query = "Select mail_id from admin_profile where mail_id =?;"

prep_stmt = Prepare_db(query)

ibm_db.bind_param(prepare_stmt, 1, name)

execution(prepare_stmt)

res = ibm_db.fetch_both(prepare_stmt)

if res is not False:

return "⚠️Account is already available"

else:

query = "INSERT INTO admin_profile (mail_id,password,Hospital_ID) VALUES
(?,?,?)"

prep_stmt = Prepare_db(query)

ibm_db.bind_param(prepare_stmt, 1, name)

ibm_db.bind_param(prepare_stmt, 2, passwords)

```
ibm_db.bind_param(prepare_stmt, 3, req_id)
```

```
execute(prepare_stmt)
```

```
ConZon_Mail_config.assing_mail('ConZo Account Creation', 'Account was  
successfully Created', name)
```

```
return True
```

```
else:
```

```
return "Hospital ID is Incorrect 😞"def dashboard_data():
```

```
query = 'Select * from containment_details LIMIT 5'
```

```
stmt = execute_immediate(query)
```

```
dictionary = ibm_db.fetch_both(stmt)
```

```
employee = []
```

```
while dictionary != False:
```

```
content = { }
```

```
content = {'ID': dictionary[0], 'Name': dictionary[1], 'City': dictionary[2], 'Latitude':  
dictionary[3],
```

```
'Longitude': dictionary[4], 'Address': dictionary[5]}
```

```
employee.append(content)
```

```
content = { }
```

```
dictionary = ibm_db.fetch_both(stmt)
```

```
return employee
```

```
def dashboard_data_add(name, city, latitude, longitude, address):
```

```
query = "Select address from containment_details where address = ?"
```

```
prep_stmt = Prepare_db(query)
```

```
ibm_db.bind_param(prep_stmt, 1, address)
```

```
execution(prep_stmt)
```

```
res = ibm_db.fetch_both(prep_stmt)
```

```
if res is not False:
```

```
    return False # "Patient Already Inserted"
```

```
else:
```

```
    execute = "INSERT INTO containment_details (name, city, latitude, longitude,  
address) VALUES (?, ?, ?, ?, ?)"
```

```
    prep_stmt = Prepare_db(execute)
```

```
    ibm_db.bind_param(prep_stmt, 1, name)
```

```
    ibm_db.bind_param(prep_stmt, 2, city)
```

```
    ibm_db.bind_param(prep_stmt, 3, latitude)
```

```
    ibm_db.bind_param(prep_stmt, 4, longitude)
```

```
    ibm_db.bind_param(prep_stmt, 5, address)
```

```
    execution(prep_stmt)
```

```
    return False # "Patient Added"
```

```
def dashboard_data_process(data):
```

```
    try:
```

```
        res_data = []
```

```
        row_len = len(data)
```

```
        col_len = len(data[0])
```

```

for colCnt in range(col_len):

    for rowCnt in range(row_len):

        res_data.insert(rowCnt, data[rowCnt][colCnt])

        dashboard_data_add(res_data[0], res_data[1], res_data[2], res_data[3], res_data[4])

    res_data.clear()

return True

except():

    return False

def dashboard_data_delete(data):

    try:

        col_len = len(data[0])

        for colCnt in range(col_len):

            query = "DELETE FROM containment_details WHERE Address = ?"

            prep_stmt = Prepare_db(query)

            ibm_db.bind_param(prepare_stmt, 1, data[4][colCnt])

            print("ok")

            execution(prepare_stmt)

        return True

    except():

        return False

def containmentZone():

    query = 'select COUNT(*) from containment_details'

```

```
result = execution_immediate(query)
```

```
res = ibm_db.fetch_both(result)
```

```
return res[0]
```

```
def containmentZone_withoutCommon():
```

```
    query = 'select COUNT(DISTINCT LATITUDE) from containment_details'
```

```
    result = execution_immediate(query)
```

```
    res = ibm_db.fetch_both(result)
```

```
    return res[0]
```

```
#_____USer_db_____#
```

```
def user_register(username, name, passwords):
```

```
    query = "Select mail_id from user_profile where mail_id =?;"
```

```
    prep_stmt = Prepare_db(query)
```

```
    ibm_db.bind_param(prepare_stmt, 1, name)
```

```
    execution(prepare_stmt)
```

```
    res = ibm_db.fetch_both(prepare_stmt)
```

```
    if res is not False:
```

```
        return "⚠Account is already available"
```

```
    else:
```

```
        query = "INSERT INTO user_profile (username,mail_id,password) VALUES (?,?)"
```

```
        prep_stmt = Prepare_db(query)
```

```
        ibm_db.bind_param(prepare_stmt, 1, username)
```

```
        ibm_db.bind_param(prepare_stmt, 2, name)
```



```
ibm_db.bind_param(prepare_stmt, 3, passwords)
```

```
execution(prepare_stmt)
```

```
ConZon_Mail_config.assing_mail('ConZo Account Creation', 'Account was  
successfully Created', name)
```

```
return "Created successfully 😊"
```

```
def user_login_verification(name, password):
```

```
    query = "Select mail_id,password from user_profile where mail_id = ?"
```

```
    prepare_stmt = Prepare_db(query)
```

```
    ibm_db.bind_param(prepare_stmt, 1, name)
```

```
    execution(prepare_stmt)
```

```
    res = ibm_db.fetch_both(prepare_stmt)
```

```
    if res is False:
```

```
        return 'Account Not Found 😞'
```

```
    else:
```

```
        if password == res[1]:
```

```
            return 'Logged Successfully 😊'
```

```
        else:
```

```
            return 'Password is Incorrect 😞'
```

```
def Con_details():
```

```
    query = 'Select * from containment_details'
```

```
    stmt = execution_immediate(query)
```

```
dictionary = ibm_db.fetch_both(stmt)
```

```
employee = []
```

```
while dictionary != False:
```

```
    content = { }
```

```
    content = {'Latitude': dictionary[3],
```

```
              'Longitude': dictionary[4]}
```

```
    employee.append(content)
```

```
    content = { }
```

```
    dictionary = ibm_db.fetch_both(stmt)
```

```
return employee
```

```
def userloginprocess(res):
```

```
    ss1 = res.replace("\'", "'")
```

```
    ss1 = ss1.replace("{", "")
```

```
    ss1 = ss1.replace("}", "")
```

```
    resa = ss1.split(',')
```

```
    return user_login_verification(resa[0].split(":")[1], resa[1].split(":")[1])
```

```
def userRegprocess(res):
```

```
    print(res)
```

```
    ss1 = res.replace("\'", "'")
```

```
    ss1 = ss1.replace("{", "")
```

```
    ss1 = ss1.replace("}", "")
```

```
    resa = ss1.split(',')
```

```

    return user_register(resa[2].split(":")[1], resa[0].split(":")[1], resa[1].split(":")[1])

def locationprocess(res):

    ss1 = res.replace("\'", "").replace("{", "").replace("}", "").split(',')

    userlati = float(ss1[0].split(':')[1])

    userlongi = float(ss1[1].split(':')[1])

    query = 'Select * from containment_details'

    stmt = execution_immediate(query)

    dictionary = ibm_db.fetch_both(stmt)

    s5 = Con_details()

    for i in range(len(s5)):    dis = hs.haversine((userlati, userlongi),
(float(s5[i]["Latitude"]), float(s5[i]["Longitude"])))

        if (dis < 0.2):

            return "YOU ARE IN CONTAINMENT ZONE"

    return "YOU ARE NOT IN CONTAINMENT ZONE"

locationprocess({'"Latitude":'11.022742869111418','Longitude':"76.9071527570486'})

```

ConZon_Mail_config.py

```

from flask_mail import Message

from ConZon_Controller import mails

message = 'sample'

bmessage = 'sample'

remail = 'sample'

def assing_mail(messages, bmessages, remails):

```

global message, bmessage, remail

message = messages

bmessage = bmessages

remail = remails

index()

def index():

```
    msg = Message(message, sender='20euit511@skcet.ac.in',  
recipients=[remail])
```

```
    msg.body = bmessage
```

```
    mails.send(msg); return 'sent'
```

3.1 GitHub & Project Demo Link:

PROJECT LINK:

<https://github.com/IBM-EPBL/IBM-Project-20226-1659715091>

DEMO LINK:

<https://drive.google.com/file/d/1V52DdUnHgfaNAnYDX2nvjcVGgVkjCpUp/view?usp=sharing>