CLOUD-CONTAINMENT ZONE ALERTING APPLICATION

TEAM ID: PNT2022TMID18386

Submitted by

Lakshman Akash N (1919102074)

Jayapraba S (1919102060)

Jothimani M (1919102064)

Kishorekumar M (1919102073)

TABLE OF CONTENT

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1
    b. Feature 2
    c. Database Schema (if Applicable)
8. **TESTING**
    a. Test Cases
    b. User Acceptance Testing
9. **RESULTS**
    a. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code
GitHub & Project Demo Link

1.INTRODUCTION

Every confirmed Covid-19 case has to be considered as an epicentre and micro-plan activities will need to be done. Containment zones are created to map the local transmission of the disease and prevent the contagion from spreading.

a. PROJECT OVERVIEW

The World Health Organization has declared the outbreak of the novel coronavirus, COVID-19 as pandemic across the world. With its alarming surge of affected cases throughout the world, lockdown and awareness (social distancing, use of masks etc) among people are found to be the only means for restricting the community transmission. In a densely populated country like India, it is very difficult to prevent the community transmission even during lockdown without social awareness and precautionary measures taken by the people. Recently, several containment zones had been identified throughout the country and divided into red, orange and green zones, respectively. The red zones indicate the infection hotspots, orange zones denote some infection and green zones indicate an area with no infection. This paper mainly focuses on development of an Android application which can inform people of the COVID-19 containment zones and prevent trespassing into these zones.

b. PURPOSE

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. Key benefits of the application are monitoring people's activity and alerting them of their safety movements.

2. LITERATURE SURVEY

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 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.

a. EXISTING SOLUTIONS

The project aims at building an application that provides information about the containment zones of a particular region by continuously monitoring an individual's location. The location of the individual must be stored in the Database. Alerts are sent using the notification service.

Features of the Application:

Admin App (portal):

They should have a login to app and update the containment zone locations in the portal. Based on the location a Geofence will be created within a 100 meters radius. They should be able to see how many people are visiting that zone.

User App (Mobile App):

The app should have user registration and login. After the user logged into the app it will track the user location and update the database with the current location. If the user is visiting the containment zone he will get an alert notification.

b. REFERENCES

1. Wawrzyniak and T. Hyla, "Application of Geofencing Technology for the Purpose of Spatial Analyses in Inland Mobile Navigation," 2016 Baltic Geodetic Congress (BGC Geomatics), Gdansk, 2016, pp. 34- 39. https://doi.org/10.1109/BGC.Geomatics.2016.15

2. Cloud Firestore Data model https://firebase.google.com/docs/firestore/data-model

3. Namiot, “Geofence services”, International Journal of Open Information Technologies 9/2013.

4. Kupper, U. Bareth, B. Freese, “Geofencing and background tracking–the next features in LBSs”, Proceedings of 41th annual conference on Geselltschaft fur Informatics, 2011

5. Mouna Berquedich, Amine Berquedich and Oulaid Kamach (March,2020)

In the case of Morocco, the Ministry of Health declared 1113 confirmed cases in Morocco having COVID-19. Given the increase in infection, a new approach has been proposed, which consists of reducing congestion at the level of emergency services by offering remote monitoring via a mobile application connected with the hospital. In this article, we discuss our approach, present the architecture of our mobile application, and illustrate the connection of our application to the electronic health record (EHR) of the patient.

6.Dipali Koshti, Supriya Kamoji, Kevin Cheruthuruthy and Surya Pratap Shahi(May 2021)

This app is a three fold app. The first fold is a Detection System for a user to undergo a Symptomatic Quiz based on a Risk Assessment ML Model to detect the presence of Covid in the user's body. The second fold is an efficient Tracking system that uses Geofencing technology to keep track of all the people who come into contact with the user. The third fold is an Alerting system that sends the alert message to all those people who came into contact with the user.

c. PROBLEM STATEMENT DEFINITION:

People travel to different places unaware of the fact that it is a COVID-19 containmentzone 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.

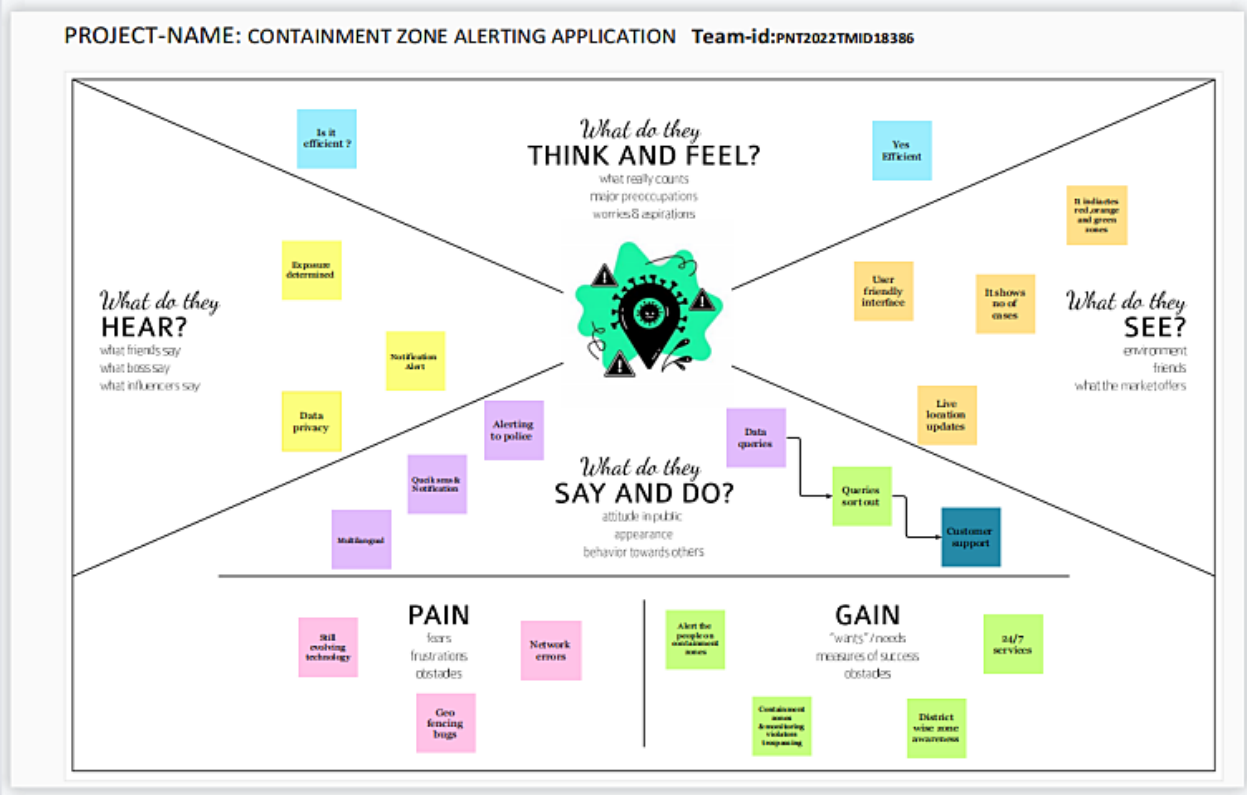
Who does the problemaffect?	People who are traveling to places being unawareof COVID Containment Zones.

issue?	If people get into a COVID containment zone, there arises a high risk for them getting into contact with an affected person and thereby it results in the transmission of the disease.
When does the issue occur?	When a healthy person gets into contact with anaffected person living in a COVID prone area.
Where does the issue occur?	Getting affected by COVID might occur anywhere. Nonetheless, the risk of getting affected by the disease would be very high in Containment Zones.

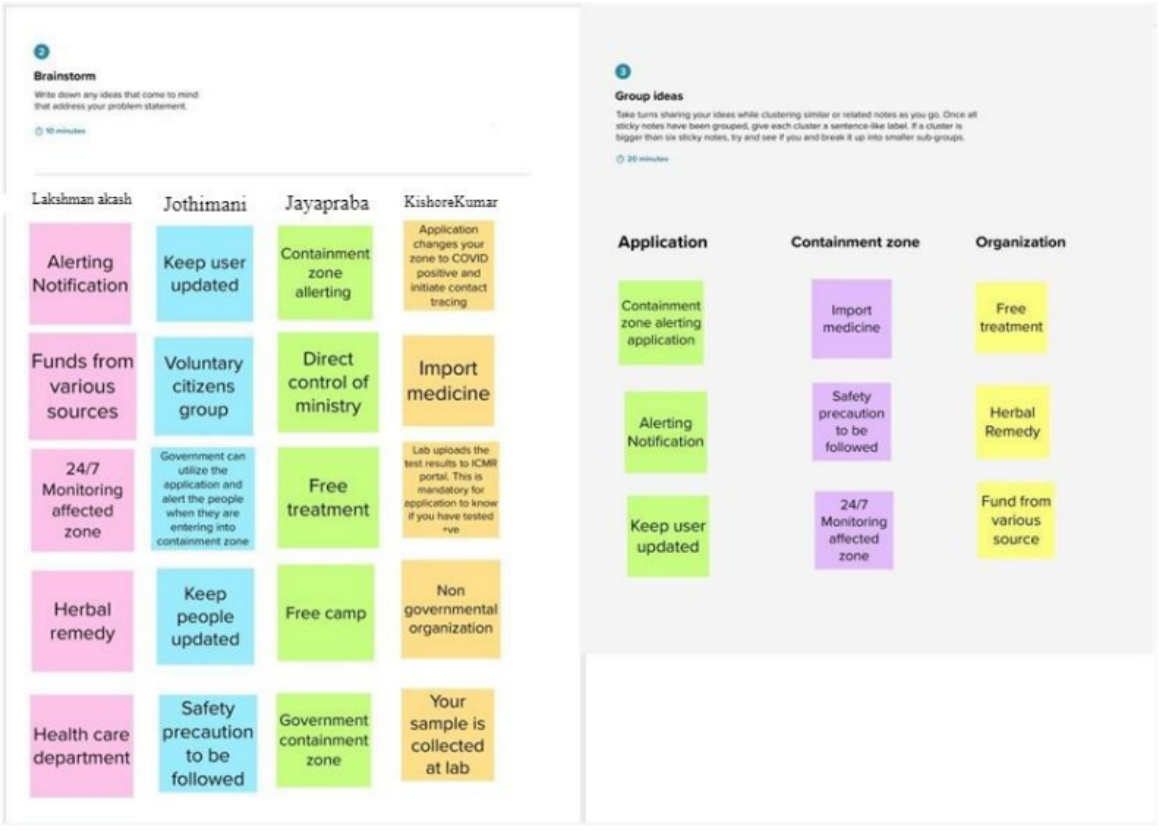
Why is it important that wefix the problem?	It is important that we fix the p What is the roblem that people become aware of Containment Zones and stay awayfrom them to protect themselves from COVID-19

3. IDEATION & PROPOSED SOLUTION:

a. EMPATHY MAP:



b. BRAINSTROMING & GROUP IDEAS:



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

c. PROPOSED SOLUTION

S No.	Parameter	Description
1.	Problem Statement (Problemto be solved)	Application also notifies the users if they have entereda containment Zone.
2.	Idea / Solution description	Isolation is the best solution for Containment zone alerting location. A tool to help your determine if youneed to isolated or take other steps to prevent spreading
3.	Novelty / Uniqueness	Open source application uniqueness. Helpful for future generations. A alert message was issued in time forpeople to take protective action.
4.	Social Impact / CustomerSatisfaction	Much less attention has however been devoted to thesort of services that should be provided to minimize the social impact. Customer satisfaction has been one of the top tools for successful application.
5.	Application Model (Revenue model)	Cost of this application is minimum and efficient to use.
6.	Scalability of the Solution	In this modern world eventhough the covid pandemic threat is about to end there are high chance of pandemic or endemic .so this application is very useful in that situation and we can use this application in seasonal diseases.

d. PROBLEM SOLUTION FIT

1. CUSTOMER SEGMENT(S) Product user is our customer	**4. EMOTION BEFORE /AFTER** Product user feel when they face a problem or error? Data lost Insecure Network error Connectivity issues	**7.BEHAVIOUR** Find the right containment zone app installer.
2. JOBS TO BE DONE/ PROBLEMS Protect the people from the containment zone.	**5. AVAILABLE SOLUTION** Install and register the application. Always "ON" your location in your mobile phone. Everyone noted the message do not refuse alert message.	**8.CHANNELS OF BEHAVIOUR** **OFFLINE:** Establish a emergency alert and maintain our equipment and facility. **ONLINE:** In online mode, incase enter the containment zone it is quickly inform and alert to user.
3. TRIGGER In automated system, presence of people were entered into the containment zone. Then the app identified location and will notification then will in turn triggers the containment zone.	**6. CUSTOMER CONSTRAINTS** What constraints prevent your customers from taking action or limit their choice? Solution :Available devices Budget Network connection Power usability	**9. PROBLEM ROOT CAUSE** What is the real reason that their problem exists? A failure situation on the network usually generates multiple alerts, because a failure condition on one devices may render other devices in accessible.
10. YOUR SOLUTION Identified the containment zone and alerts the people through the message in your mobile phone from anywhere and anyplace.		

4. REQUIREMENT ANALYSIS:

a. FUNCTIONAL REQUIREMENTS:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	It can be registered by valid email id or phone number.
FR-2	**User Confirmation**	Verification code can received by registering email id or phone number
FR-3	**Alert message via notification**	By user access of location will entered in containment area the notification are send by GPS tracking system and push the grids through mail id
FR-4	**Show Infected zones**	Marketed by GEO fencing.
FR-5	**Track alternative routes**	By Google map API or Google dependencies.

b.NON-FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Using Android or IOS or windows application.
NFR-2	Security	The user data is stored securely in IBM Cloud.
NFR-3	Reliability	The Quality Of the Service are trusted.
NFR-4	Performance	It provide smooth user experience.
NFR-5	Availability	The Service are available for 24 /7.

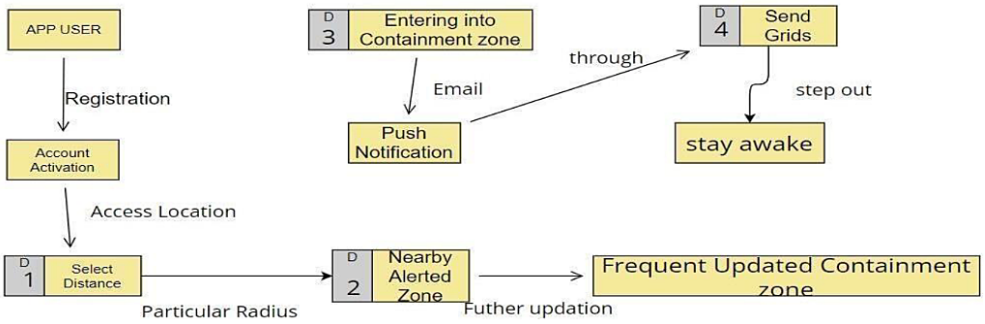
NFR-6	Scalability	It is easy to scalable size for users.

5. PROJECT DESIGN

a. DATA FLOW DIAGRAM:

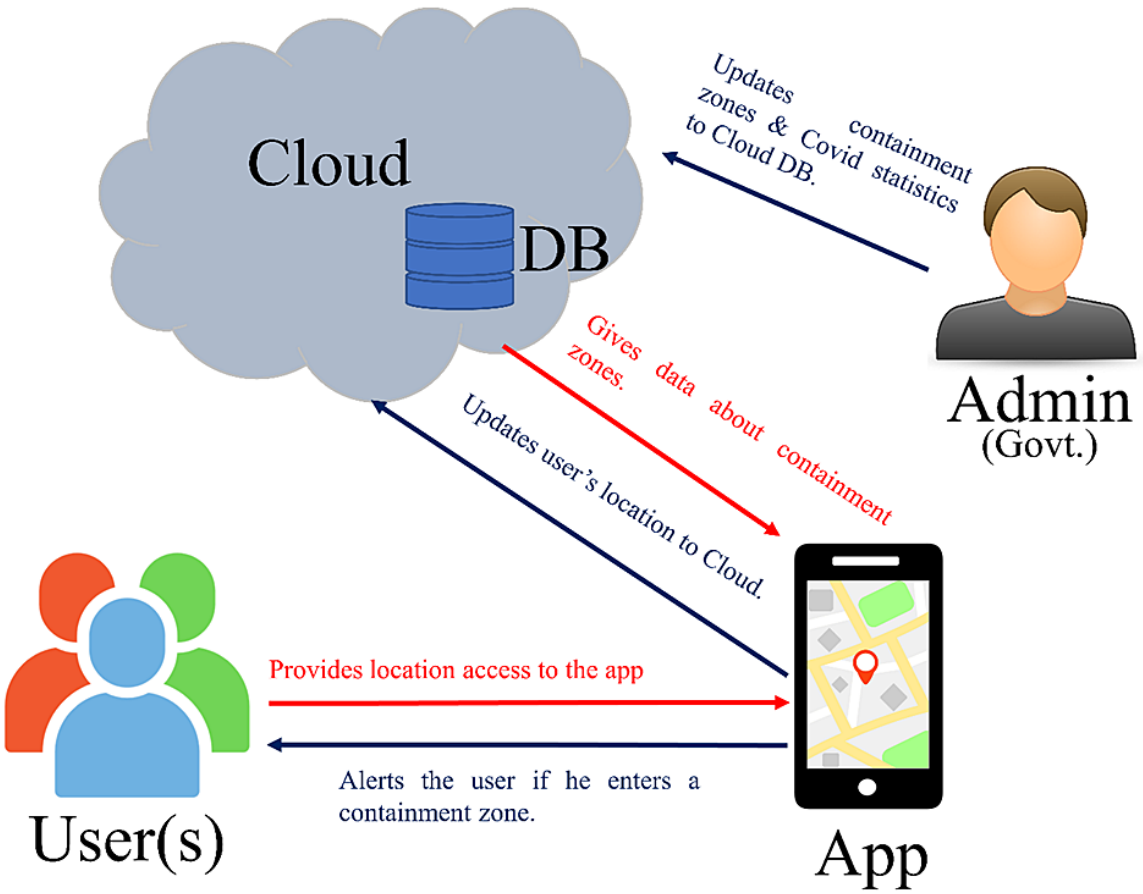
Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

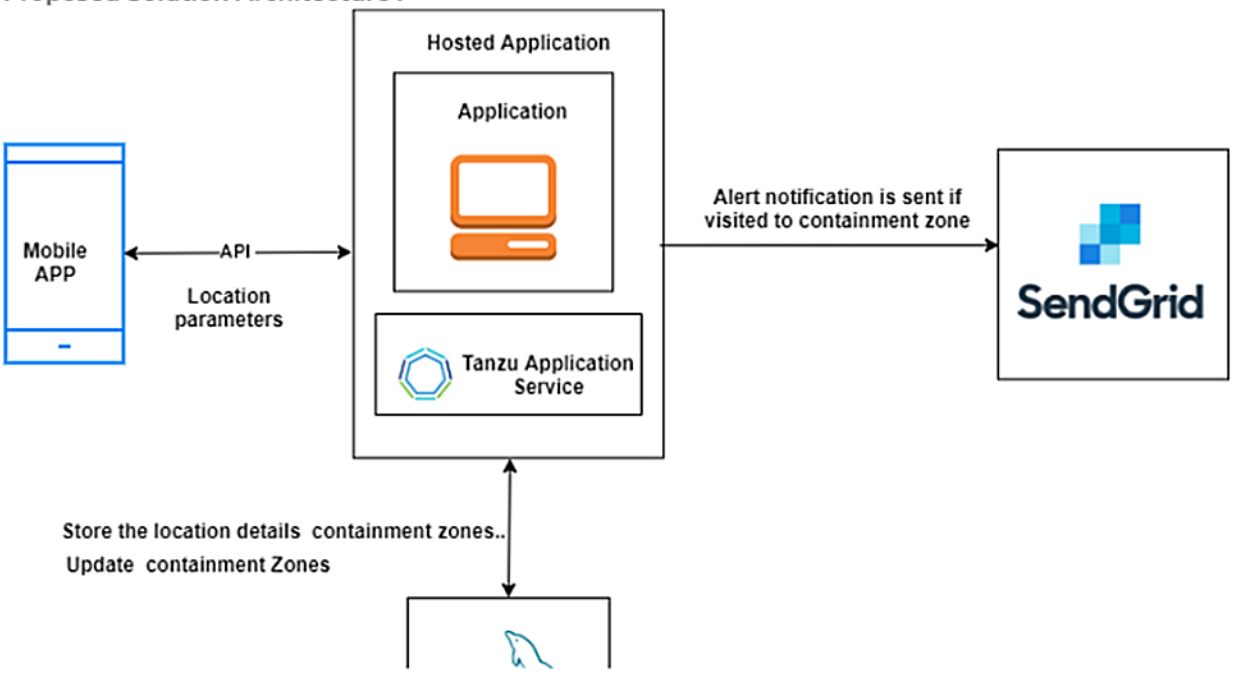


b. SOLUTION ARCHITECTURE:

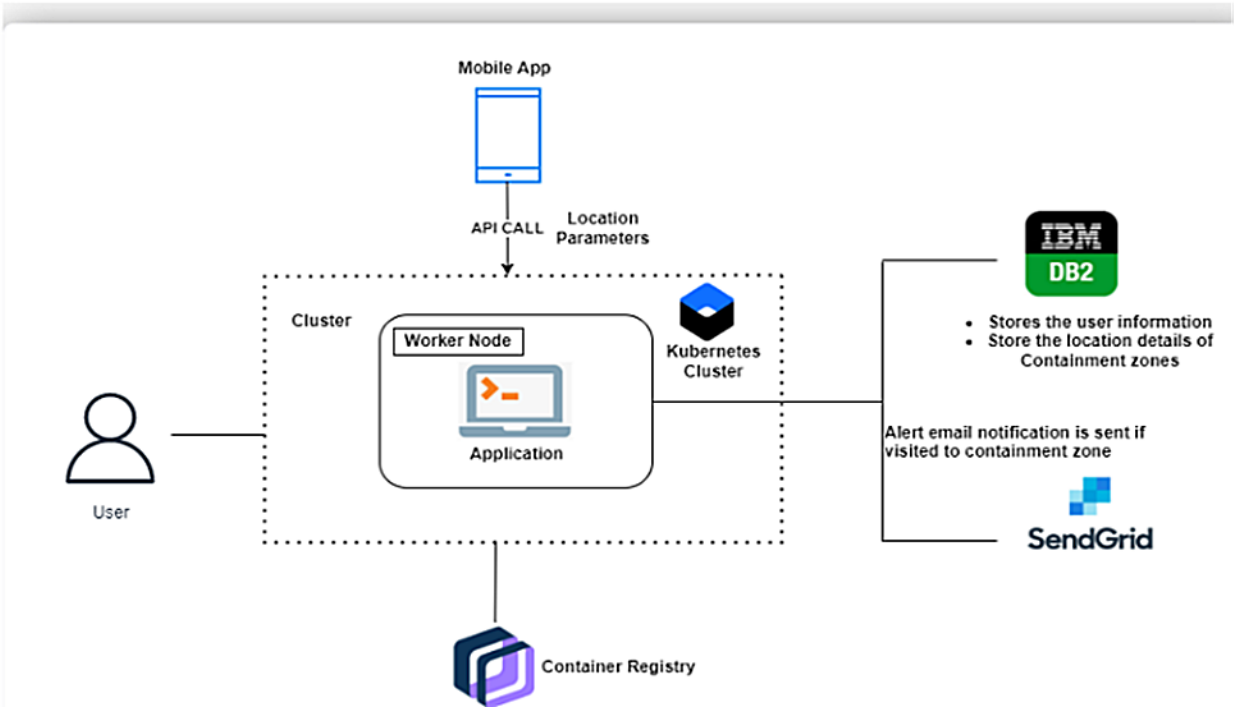
BLOCK DIAGRAM



Proposed Solution Architecture :



Technical Architecture :



5.3. USER STORIES

User Type	Functional Requirement	User Story No.	User Story/Task	Acceptance Criteria	Priority	Release
Admin	Login	USN-1	As an admin, I want to log in to my dashboard.	I can access my dashboard.	High	Sprint 1
	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.	High	Sprint 1

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.	Medium	Sprint 2
	Login	USN-4	I need to view a listing of details of Containment zones.	I can view a listing of the containment zones.	High	Sprint 2

	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.	High	Sprint 3
		USN-6	COVID statistics	I am provided with COVID statistics.	Medium	Sprint 3
		USN-7	I would appreciate COVID news.	I get access to COVID news from the web.	Low	Sprint 4
	Ease of Use	USN-8	If I forget my password, I need help to reset it.	I can reset my password easily.	Medium	Sprint 4

6. PROJECT PLANNING & SCHEDULING

a. SPRINT PLANNING

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 enteringmy email and password	3	High	Lakshman Akash Jayapraba Jothimani Kishorekumar
		USN-2	**USER:** I will receive a confirmation email once I haveregistered for the application	2	High	Lakshman Akash Jayapraba Jothimani Kishorekumar
	Login	USN-3	**USER:** I can log into the application by entering my email & password	3	High	Lakshman Akash Jayapraba Jothimani Kishorekumar

Sprint	Functional Requirement (Epic)	User Story Numb	User Story / Task	Story Points	Priority	Team Members

		er				
SPRINT-2	**Dashboard**	USN-4	**USER:** I need to give permission to access my location	5	High	Lakshman Akash Jayapraba Jothimani Kishorekumar
		USN-5	**USER:** I can view the map with the containmentzones	5	High	Lakshman Akash Jayapraba Jothimani Kishorekumar
	Service	USN-6	**ADMIN:** I need to update the containment zones.	5	High	Lakshman Akash Jayapraba Jothimani Kishorekumar

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
SPRINT-3	**Service**	USN-7	**ADMIN:** I need to differentiate the containment zones based on the intensity of infection.	3	Medium	Lakshman Akash Jayapraba Jothimani Kishorekumar
		USN-8	**ADMIN:** I need to provide precautionary measureswhen they travel.	3	Medium	Lakshman Akash Jayapraba Jothimani Kishorekumar

		USN-9	**ADMIN:** I need to provide information about the nearbyhospitals	3	Low	Lakshman Akash Jayapraba Jothimani Kishorekumar

		USN-10	**ADMIN:** Admin: I need to provide preventive measures when they travel through it.	5	High	Lakshman Akash Jayapraba Jothimani Kishorekumar

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
SPRINT-4	**Service**	USN-11	**ADMIN:** I need to alert the user when they enter the containment zone through email or SMS	5	High	Lakshman Akash Jayapraba Jothimani Kishorekumar
		USN-12	**ADMIN:** I need to provide medical recommendations by collaborating with hospitals.	3	Low	Lakshman Akash Jayapraba Jothimani Kishorekumar
	Data collection	USN-13	**ADMIN:** I need to store user details on the	5	High	Lakshman Akash Jayapraba Jothimani Kishorekum

			cloud			ar
		USN-14	**ADMIN:** I need to collect details about covid-19 casesfrom verified sources	5	High	Lakshman Akash Jayapraba Jothimani Kishorekum ar

SPRINT ESTIMATION:

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 13 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

SV=Sprint duration/Velocity=14/10=1.4

b. 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	25 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	06 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	11 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	17 Nov 2022	20	17 Nov 2022

c.REPORTS FROM JIRA

7.CODING & SOLUTIONING

Signup.html

```
package com.example.containmentzonealerting;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONException;
import org.json.JSONObject;
```

```
public class SignupActivity extends AppCompatActivity {
   private EditText name;
   private EditText email;
   private EditText password;
   SharedPreferences sharedpreferences;
   private Button button1;
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_signup);
     name = findViewById(R.id.name);
      email = findViewById(R.id.email);
      password = findViewById(R.id.password);
      button1 = findViewById(R.id.login_button);
      sharedpreferences = getApplicationContext().getSharedPreferences("user_data", 0);
      SharedPreferences.Editor editor = sharedpreferences.edit();
      editor.clear();
      editor.apply();

      if(sharedpreferences.getAll().size() >= 3){
         Intent intent = new Intent(SignupActivity.this,MainActivity.class);
         startActivity(intent);
      }
      button1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent intent = new Intent(SignupActivity.this,LoginActivity.class);
            startActivity(intent);
         }
      });
}
 public void signUp(View view) {
      if(!name.getText().toString().equals("") || !email.getText().toString().equals("") ||
!password.getText().toString().equals("")){
```

```
postDataUsingVolley(name.getText().toString(),email.getText().toString(),password.getText().toStrin
g());
      }else{
         Toast.makeText(SignupActivity.this, "Some Fields are empty", Toast.LENGTH_LONG).show();
      }
   }
private void postDataUsingVolley(String name, String email,String password) {
      final RequestQueue queue = Volley.newRequestQueue(this);
      String url = "http://192.168.193.227:5000/android_sign_up";

      JSONObject postparams = new JSONObject();
      try {
         postparams.put("name", name);
         postparams.put("email", email);
         postparams.put("password", password);
      } catch (JSONException e) {
         e.printStackTrace();
      }

      JsonObjectRequest jsonObjReq = new JsonObjectRequest(Request.Method.POST, url,
postparams,
            new Response.Listener<JSONObject>() {
               @Override
               public void onResponse(JSONObject response) {
                  Log.d("response",response.toString());

                  try {
                     int userId = response.getInt("id");

                     SharedPreferences.Editor editor = sharedpreferences.edit();
                     editor.putString("name", name);
                     editor.putString("email", email);
                     editor.putInt("id", userId );
                     editor.apply();
```

```
                    Intent intent = new Intent(SignupActivity.this,MainActivity.class);
                    startActivity(intent);

                } catch (JSONException e) {
                    e.printStackTrace();
                                        }
        }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.d("error",error.toString());
                }
            });

        queue.add(jsonObjReq);
    }
}
```

App.py

```
from cProfile import run
from turtle import st
from flask import Flask, render_template, request, redirect, url_for, session,jsonify
from markupsafe import escape
import json
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail


import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30699;SECURITY=SSL;SSL
ServerCertificate=C:/Users/SONA COLLEGE/Desktop/IBM-PROJECT-DEMO/Containment Zone
alerting app/templates/DigiCertGlobalRootCA.crt;UID=qxc23007;PWD=4rIT5VpkoO6D0FQM","")

app = Flask(__name__)
```

```
app.secret_key = "unique secret key"

@app.route('/')
def home():
 return render_template('home.html')

@app.route('/login1')
def login1():
 return render_template('login.html')

@app.route('/adduser')
def new_student():
 return render_template('signup.html')

@app.route('/dashboard')
def dashboard():
 return render_template('dashboard.html')

@app.route('/logout')
def logout():
 return render_template('home.html')

def send_mail(email):
   print(email)
   message = Mail(from_email='lakshmanakash@gmail.com',
                  to_emails=email,
                  subject='caution',
                  plain_text_content='Please Stay Safe',
                  html_content='<h2>You are entering into a containment Zone</h2>')

   try:
      sg = SendGridAPIClient(
         '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')
      response = sg.send(message)
      print(response.status.code)
```

```
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)

@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
  if request.method == 'POST':

    name = request.form['name']
    email = request.form['email']
    password = request.form['pwd']

    sql = "SELECT * FROM user WHERE name =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:
      return render_template('list.html', msg="You are already a member, please login using your
details")
    else:
      insert_sql = "INSERT INTO user VALUES (?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, email)
      ibm_db.bind_param(prep_stmt, 3, password)

      ibm_db.execute(prep_stmt)

    return render_template('home.html', msg="Student Data saved successfuly..")

@app.route('/list')
def list():
```

```
 students = []
 sql = "SELECT * FROM user"
 stmt = ibm_db.exec_immediate(conn, sql)
 dictionary = ibm_db.fetch_both(stmt)
 while dictionary != False:
   # print ("The Name is : ",  dictionary)
   students.append(dictionary)
   dictionary = ibm_db.fetch_both(stmt)

 if students:
   return render_template("list.html", students = students)

# @app.route('/delete/<name>')
# def delete(name):
#   sql = f"SELECT * FROM user WHERE name='{escape(name)}'"
#   print(sql)
#   stmt = ibm_db.exec_immediate(conn, sql)
#   student = ibm_db.fetch_row(stmt)
#   print ("The Name is : ",  student)
#   if student:
#     sql = f"DELETE FROM Students WHERE name='{escape(name)}'"
#     print(sql)
#     stmt = ibm_db.exec_immediate(conn, sql)
#     students = []
#     sql = "SELECT * FROM Students"
#     stmt = ibm_db.exec_immediate(conn, sql)
#     dictionary = ibm_db.fetch_both(stmt)
#     while dictionary != False:
#       students.append(dictionary)
#       dictionary = ibm_db.fetch_both(stmt)
#     if students:
#       return render_template("list.html", students = students, msg="Delete successfully")
# # while student != False:
 ]# #   print ("The Name is : ",  student)
@app.route('/login',methods = ['POST', 'GET'])
```

```
def login():
  if request.method == 'POST':
    email = request.form['email']
    password = request.form['pwd']

   sql = "SELECT * FROM user WHERE email =? AND pwd=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.bind_param(stmt,2,password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:
      session['email']=email
      return render_template('dashboard.html',mail=session['email'])
    else:
      return render_template('login.html',msg="please try again, password email or password!!")

@app.route('/home',methods = ['POST', 'GET'])
def submit():
  if(request.method == "POST"):
        # get data
        lat = request.form["lat"]
        lon = request.form["lon"]
        vis = 0

        sql = "SELECT * FROM location WHERE lat =? AND lon=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,lat)
        ibm_db.bind_param(stmt,2,lon)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
          return render_template('dashboard.html', msg="Already Zone is marked as Containment zone")
```

```
        else:
            insert_sql = "INSERT INTO location(lat,lon,vis) VALUES (?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, lat)
            ibm_db.bind_param(prep_stmt, 2, lon)
            ibm_db.bind_param(prep_stmt, 3, vis)

            ibm_db.execute(prep_stmt)
            return render_template('dashboard.html', msg=" Data saved
successfuly..",mail=session['email'])

@app.route('/data')
def data():
  zones = []
  sql = "SELECT * FROM location"
    stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
    zones.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

  if zones:
    return render_template("data.html", zones = zones)

@app.route("/android_sign_up", methods=["POST"])
def upload():
    if(request.method == "POST"):

        # get the data from the form
        name = request.json['name']
        email = request.json['email']
        password = request.json['password']

        sql = "SELECT * FROM user WHERE email =?"
```

```
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
          return {'status': 'failure'}
        else:
          insert_sql = "INSERT INTO user VALUES (?,?,?)"
          prep_stmt = ibm_db.prepare(conn, insert_sql)
          ibm_db.bind_param(prep_stmt, 1, name)
          ibm_db.bind_param(prep_stmt, 2, email)
          ibm_db.bind_param(prep_stmt, 3, password)
          ibm_db.execute(prep_stmt)
          return {"id": 1}
          # data = []
          # sql = "SELECT * FROM user WHERE email = ?' "
          # stmt = ibm_db.exec_immediate(conn, sql)
          # ibm_db.bind_param(stmt, 1, email)
          # ibm_db.execute(stmt)

          # dictionary = ibm_db.fetch_both(stmt)
          # while dictionary != False:
          #   # print ("The Name is : ",  dictionary)
          #   data.append(dictionary)
          #   dictionary = ibm_db.fetch_both(stmt)

    # return {'status': 'failure'}
    @app.route("/location_data")
    def location_data():
      locationdata = []
      sql = "select * from location"
      stmt = ibm_db.exec_immediate(conn, sql)
      dictionary = ibm_db.fetch_both(stmt)
      while dictionary != False:
```

```
    #  print ("The Name is : ",  dictionary)
     locationdata.append(dictionary)
     dictionary = ibm_db.fetch_both(stmt)
  # # row_headers = [x[0] for x in conn.cursor().description]
  # json_data = []
  # for result in locationdata:
  #   json_data.append(dict(zip(result)))
  return json.dumps(locationdata)

@app.route("/post_user_location_data", methods=["POST"])
def post_user_location():
    if(request.method == "POST"):
      lat = request.json['lat']
      lon = request.json['long']
      id = request.json['id']
      ts = request.json['timestamp']

      insert_sql = "INSERT INTO USER_LOCATION VALUES (?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, lat)
      ibm_db.bind_param(prep_stmt, 2, lon)
      ibm_db.bind_param(prep_stmt, 3, id)
      ibm_db.bind_param(prep_stmt, 4, ts)

      ibm_db.execute(prep_stmt)
      return {"response": "success"}

@app.route("/send_trigger",methods=["POST"])
def send_trigger():
    if(request.method == "POST"):
        # get the data from the form
        email = request.json['email']
        location_id = request.json['id']

        sql = "SELECT vis FROM Location WHERE SNO =?"
```

```
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,location_id)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
          visited=account[0]
          visited=visited+1
          sql = "UPDATE LOCATION SET vis = ? WHERE SNO=?"
          stmt = ibm_db.prepare(conn, sql)
          ibm_db.bind_param(stmt,1,visited)
          ibm_db.bind_param(stmt,1,location_id)
          ibm_db.execute(stmt)

          send_mail(email)
          return {"response": "success"}
        else:
          return {"response": "failure"}
if __name__ == '__main__':
   app.run(debug=True,host='0.0.0.0', port=5000)
```

8.TESTING

a.Test case:

The following section contains the report of the testing phase of the software.

Project Name	Containment zone alerting application
Project Type	Cloud app development
Developer	Lakshman Akash N, Jayapraba S, Jothimani M, Kishore kumar M
Language	Html,Javascript,python,DB2
Total Number of Testcases	45
Number of Testcases executed	45
Total Number of Testcases passed	44
Total Number of Testcases failed	1-System not useful for Blind People

b.User acceptance testing

Purpose of Document:

The purpose of this document is to briefly explain the test coverage and open issue of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

1.Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	4	4	2	3	13
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	5	2	4	5	16
Not Reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	0	0	0	0
Totals	12	9	10	10	41

9.RESULTS

a)performance metric

Containment activities are undertaken in a confined area to prevent
infection from getting established in the community and prevent its spread outside the said area. It intends
to break the cycle of transmission.

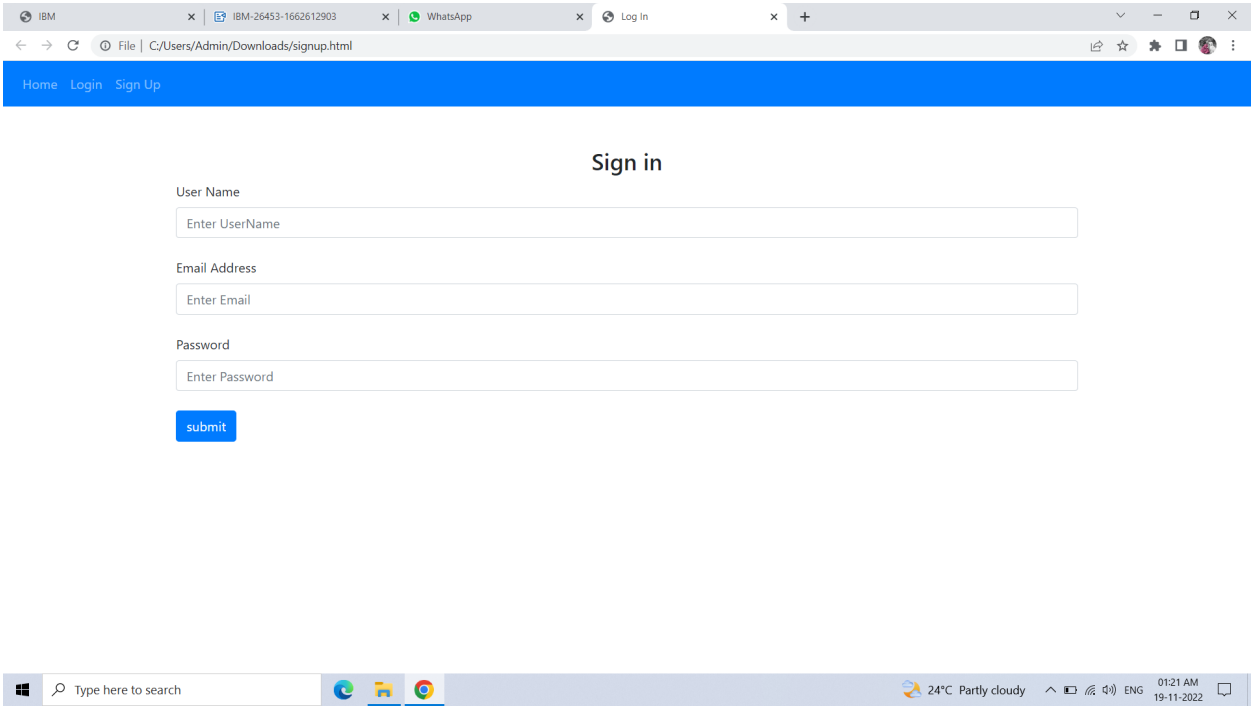
Key Interventions for COVID Containment

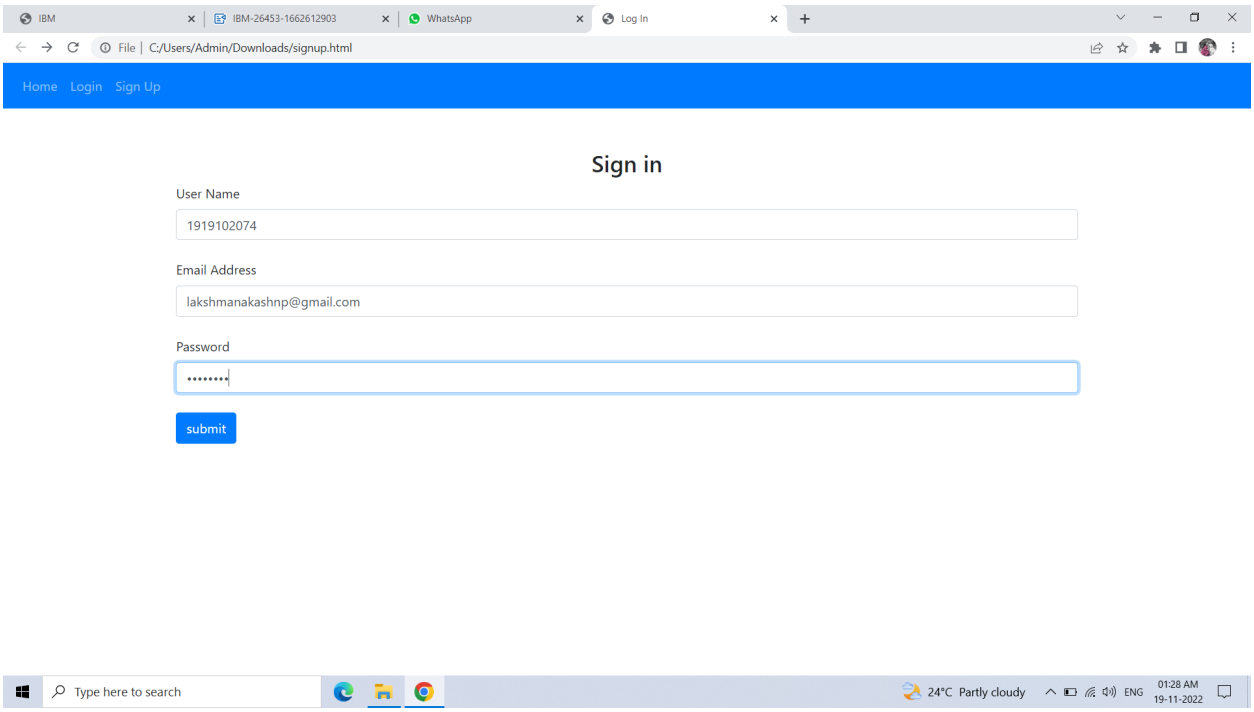
• Community Surveillance (House to House search for suspect cases) in
containment zone

• List and identify contacts

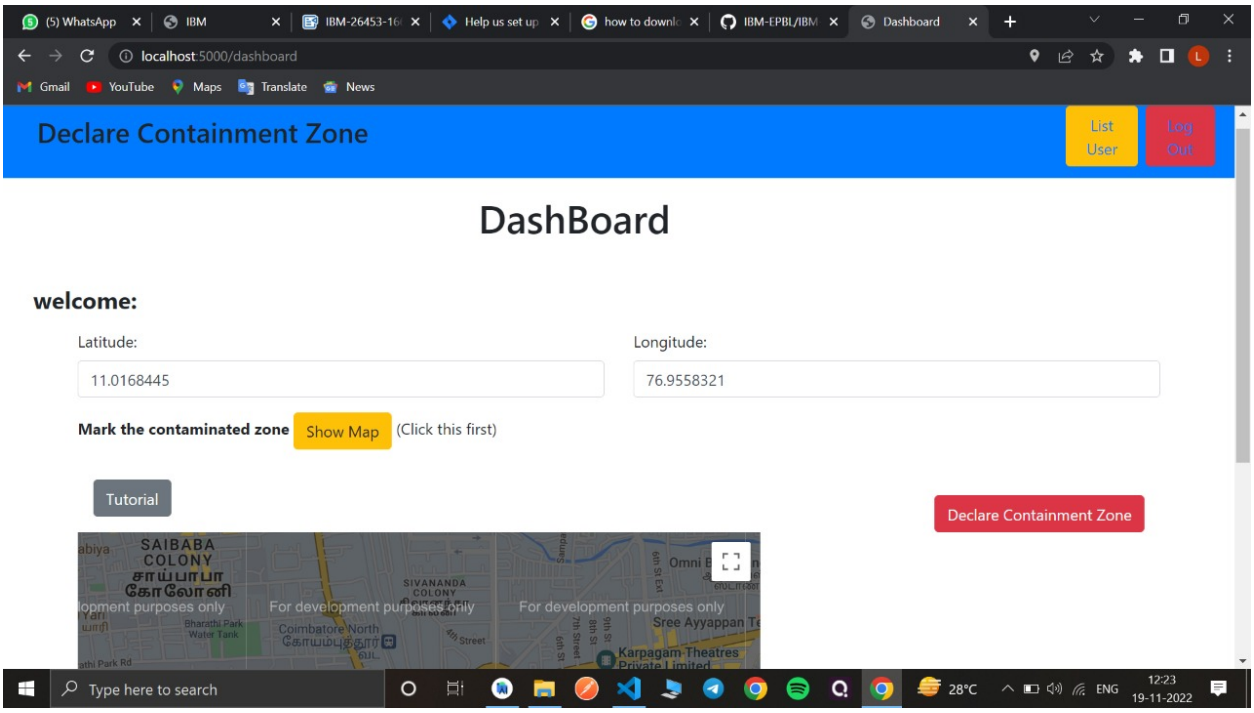
• Quarantine and follow-up of contacts

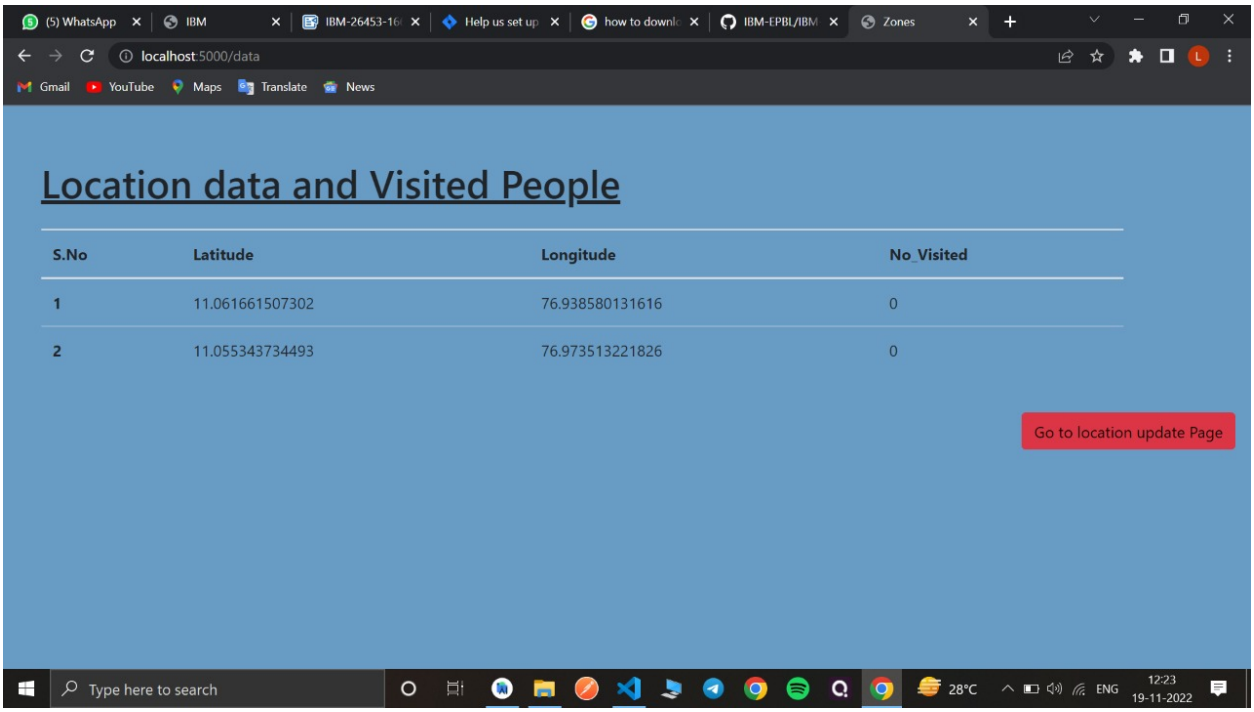
b)output




IBM
IBM-26453-1662612903
WhatsApp
Log In
File | C:/Users/Admin/Downloads/signup.html
Home Login Sign Up
Sign in
User Name
1919102074
Email Address
lakshmanakashnp@gmail.com
Password
submit

Type here to search
24°C Partly cloudy
ENG
01:28 AM
19-11-2022


(5) WhatsApp
IBM
Help us set up
how to downl
IBM-EPBL/IBM
Dashboard
localhost:5000/dashboard
Gmail
YouTube
Maps
Translate
News
Declare Containment Zone
List User
Log Out
DashBoard
welcome:
Latitude:
11.0168445
Longitude:
76.9558321
Mark the contaminated zone
Show Map
(Click this first)
Tutorial
Declare Containment Zone
SAIBABA COLONY
சாய்பாபா கோலோனி
SIVANANDA COLONY
Coimbatore North
Karpagam Theatres Private Limited
For development purposes only
Type here to search
28°C
ENG
12:23
19-11-2022



10. ADVANTAGES & DISADVANTAGES

Advantages:

1.Alert people about the containment zones

2.Works Under Low Data Connection.

3. User Friendly Application.

4. Data Privacy.

5. Easy to Understand.

Disadvantages:

1.It cannot be used without internet connection

11. CONCLUSION

The application provides an efficient way of showing the identified COVID-19 containment zones to the users in a Google map. With the alarming increase of COVID-19 affected cases throughout the world, this developed application can be employed as a tool for creating further social awareness among the people. 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 and exiting the containment areas.

12. FUTURE SCOPE

The scope of this paper is restricted to provide the concept to apply the K-means technique from DataScience on the collected patient's geographical data like locations to define and visually plot the contentment zones (clusters) on the actual map.

13. APPENDIX

GitHub link:

https://github.com/IBM-EPBL/IBM-Project-26453-1660026966

Demo_Link:

my.sharepoint.com/:v:/g/personal/lakshmanakash_19cse_sonatech_ac_in/EfnCnUe4ExZAqCbqHqs-q5UB_Wjl_07eEqosZwHdhc-dIA?e=bALhKY