Project Report

Team ID	PNT2022TMID00349
Project Name	CONTAINMENT ZONE ALERTING APPLICATION

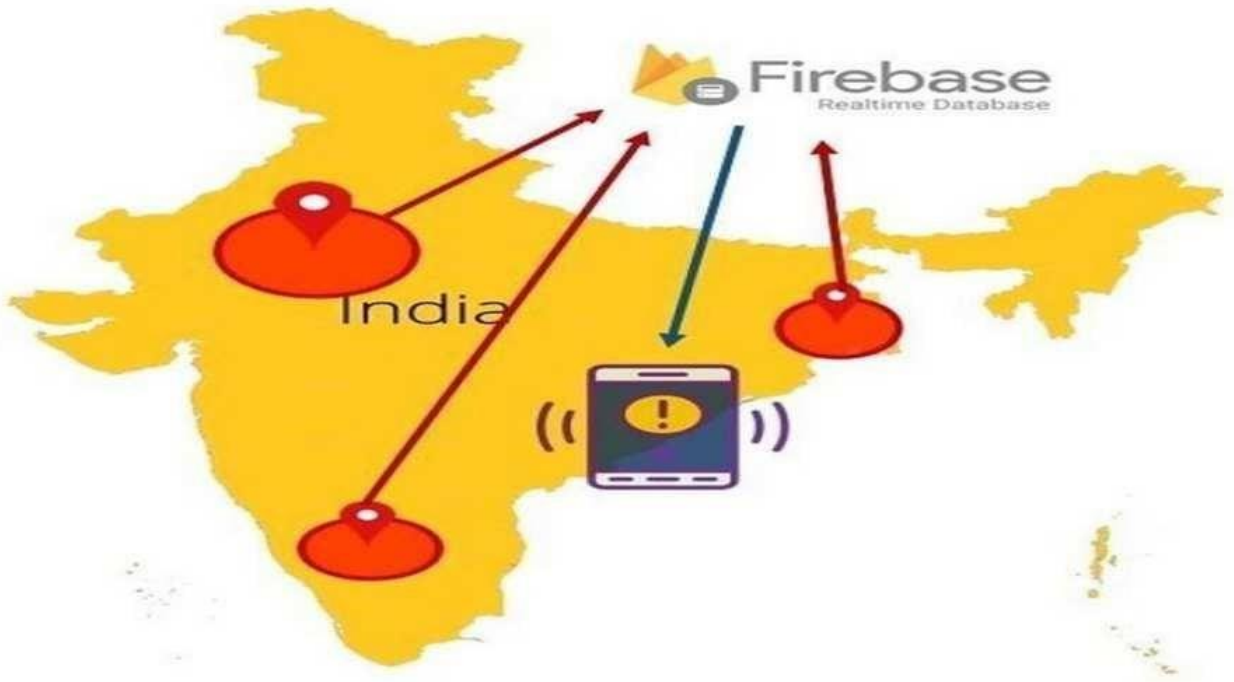
1. INTRODUCTION

1.1 Project Overview:

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), masks 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 (Bahl et al. 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. In this paper, we focus on developing a mobile based application to provide information regarding the Covid-19 containment zones in West Bengal. The application further tracks the user's location and provides notification alert if the user has entered a containment zone. The application also provides daily Covid-19 case statistics to the users to keep them updated. The application is developed on Android SDK and uses Firebase Cloud Firestore to store the location data. Android's geofencing client is used to create geofences around the containment zones and notification manager is used to provide notifications. The application also uses RESTful web services to show the Covid-19 cases in West Bengal. We have tested our application with different users in different locations across West Bengal and it works efficiently and is able to attain our target.

Purpose:

The Android application shows the location of the containment zones to the users. It also notifies the user when he or she trespasses the boundary of a containment zone or stays in the containment zones



2. LITERATURE SURVEY: 2.1 Existing problem:

People doesn't have proper knowledge about containment zones since they do change daily and hard to keep updated and if they are not updated properly, they will lead to wide spread of disease.

2.2 References:

PAPER 1:
TITLE: Tracking the Covid zones through geo-fencing technique
AUTHOR NAME: Anto Arockia Rosaline R ,Lalitha R ,Hariharan G ,Lokesh PUBLICATION **YEAR**: 2017 **DESCRIPTION:**

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

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

PAPER 2:

AUTHOR NAME: Geofencing 2.0: Taking Location-based Notifications to the Next Level **PUBLICATION YEAR:** 2016 **DESCRIPTION:**

Sandro Rodriguez Garzon Bersant Deva The basic Android application that served as the prototype Geofencing client was used. This client is primarily responsible for carrying out the geofencing server's ongoing location update strategy. This must be accomplished with little energy consumption because the Geofencing client is located on a mobile device. We made the decision to employ the low energy Geofencing features of the Android operating system to keep an eye on the safety zone. As a result, a safety zone is considered as a single circular geofence with a required exit on the mobile device. However, they discovered that there was occasionally a significant lag time between leaving the safety zone and receiving a notification from the system about the leave. In order to address this issue, a specific amount of the safety zone's radius is decreased. While the safety zone and how it is implemented have a significant impact on overall energy consumption, it is also important to make the right choice when it comes to a placement mechanism. In order to reduce power consumption without compromising the necessary position precision, they used a device-based smart combination of various positioning mechanisms introduced by. By temporarily deactivating placement when a device is not in motion, the Geofencing client also makes use of cutting-edge mobile sensing capabilities integrated into the Android operating system's activity recognition unit. Mobile users who live close to a geo-border fence's will find this to be of particular utility. If the Geofencing server notifies the Geofencing client about a geo- notice, the notification will appear right away.

PAPER 3

TITLE: Development of An Android Application for Viewing Covid19 Containment Zones Alerting.

AUTHOR NAME: India Ranajoy Mallik, Amlan Protim Hazarika, Sudarshana Ghosh Dastidar, Dilip Sing & Rajib Bandyopadhyay **PUBLICATION YEAR**: 2019 **DESCRIPTION**:

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. This Android application updates the locations of the areas in a Google map which are identified to be the containment

zones. The application also notifies the users if they have entered a containment zone and uploads the user's IMEI number to the online database. To achieve all these functionalities, many tools, and APIs from Google like Firebase and Geofencing API are used in this application. Therefore, this application can be used as a tool for creating further social awareness about the arising need of precautionary measures to be taken by the people of India.

TITLE: Aarogya Setu
AUTHOR NAME: National Informatics Centre, Ministry of Electronics & Information Technology, Government of India **PUBLICATION YEAR:** 2014 **DESCRIPTION:**

The most popular containment zone alert application among the options currently in use in India is called Aarogya Setu. The Indian government created a mobile application to link the public with crucial health services. Its primary features include geo-location-based COVID19 data, user risk status, automatic contact tracing using Bluetooth, and much more. The movement of an infected individual is tracked using Bluetooth and GPS technology, and the system notifies the public of the locations the infected person has visited while designating those locations as vulnerable ones. It employs cellular triangulation to determine a person's location in the absence of GPS technology. While Aarogya Setu can track down contacts and notify those who have come into touch with someone who has COVID-19, it also actively keeps track of quarantine or containment zones and alerts users who enter them. The Terms of Use and Privacy Policy must be accepted at the time of registration when installing the application on any Android or iOS mobile device, and ongoing use of the application denotes continued acceptance. Name, age, sex, occupation, phone number, overseas travel within the previous 28–45 days, and whether the user is a smoker are all pieces of information that the app gathers. This data is kept on a server that is under the jurisdiction of the Indian government. It is hashed and sent to the user's mobile application along with a special digital ID (DID). The user is recognised using the DID. In order for the user's mobile phone to exchange information with another device that has the app when it gets within range, the Bluetooth and GPS services must be turned on. Their individual IDs, along with the time and GPS location, are kept on the two phones when two users come into close proximity. The format in which this data is kept is encrypted. Only after a person tests positive is it posted to the government-controlled servers of the app.

2. 3.Problem Statement Definition:

Customer Problem Statement Template:

1	I AM	Contributor / Developer
2	I'AM TRYING TO	Detect the Containment Zone in the world in a short duration and send a noticification to the people using our application.
3	BUT	People may consider this type of noticifications as fake and might not bother about this type of noticifications.
4	BECAUSE	There are people who spread the rumors in the society within seconds. Where everyone starts sharing the fake alert message to others in the society.
5	WHICH MAKES ME FEEL	Now a days Containment Zone has increased, so the people must have awareness about this in the Society. My Project will help to get rid of all these problems.

3. IDEATION & PROPOSED SOLUTION

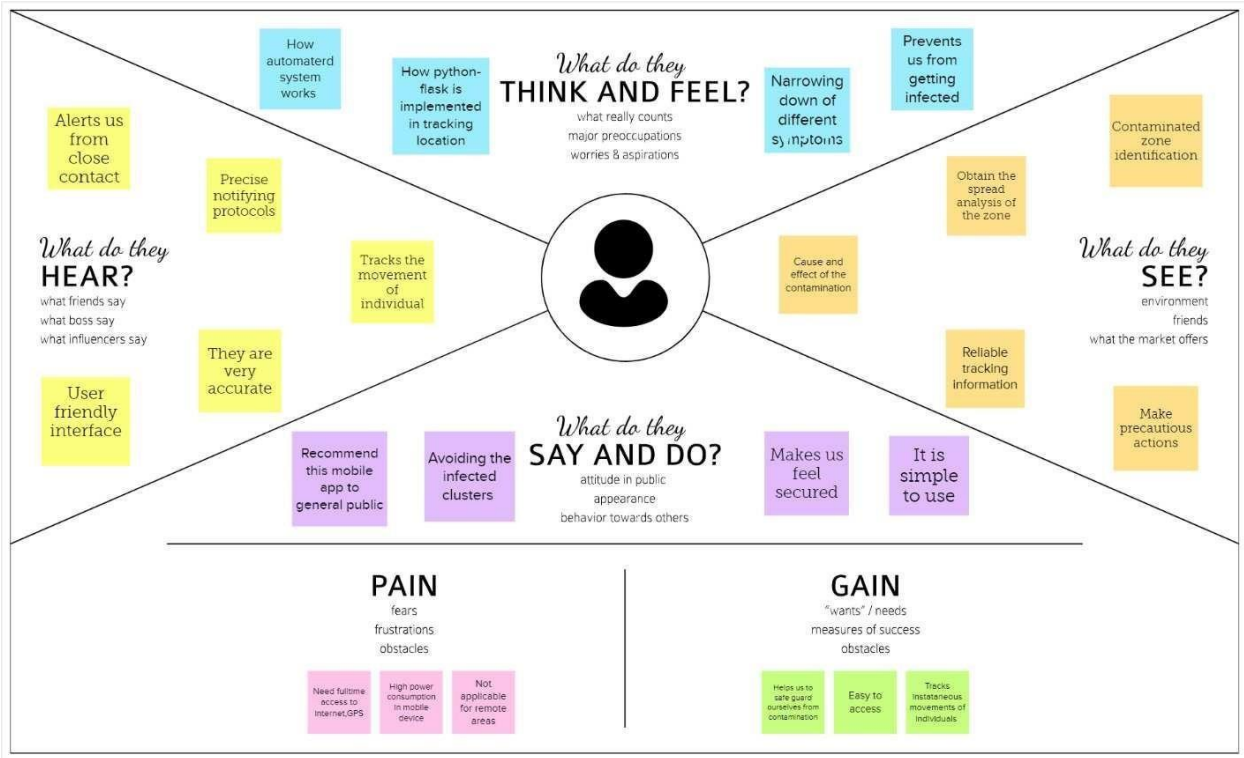
3.1 Proposed Solution Fit

S.NO	PARAMETER	DESCRIPTION
1	Problem Statement (Problem to be solved)	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	Idea / Solution description	The project aims at building an application that provides information about the containment zones of a particular region by continuously monitoring an individual's location. Location of the individual must be stored in the Database. Alerts are sent using the notification service.

3	Novelty / Uniqueness	The uniqueness of containment zone alerting app is it shows the particular area of the district before the 100 meter, and the user's location history is stored in database and this app provides the precautions measurements, list of immunity boosters, location of the vaccination providing places it also gives the list of the affected and admitted patients and discharged patients , percentage of affecting by covid19
4	Social Impact / Customer Satisfaction	Social Stigma is discrimination against a particular group of people, a place, or a nation in the form of a negative attitude. Public health emergencies (such as COVID- 19 pandemic) are stressful situations for people and communities. Fear and anxiety with a lack of knowledge about the disease can leads to social stigma. The containment zone alerting app users are 100% satisfied because of its immediate notification of a particular area, it provides the precautions and awareness about COVID-19.
5	Business Model (Revenue Model)	When User enters some other region which is not the user's home region, user has to subscribe in order to view the containment zones in the new region, in addition, subscribing to personal health tracker allows the user to manage his health efficiently.
6	Scalability of the Solution	In this modern world even though the covid pandemic threat is about to end there are high chance of pandemic or endemic. So, this application is very useful in that situation and we can use this application in seasonal diseases

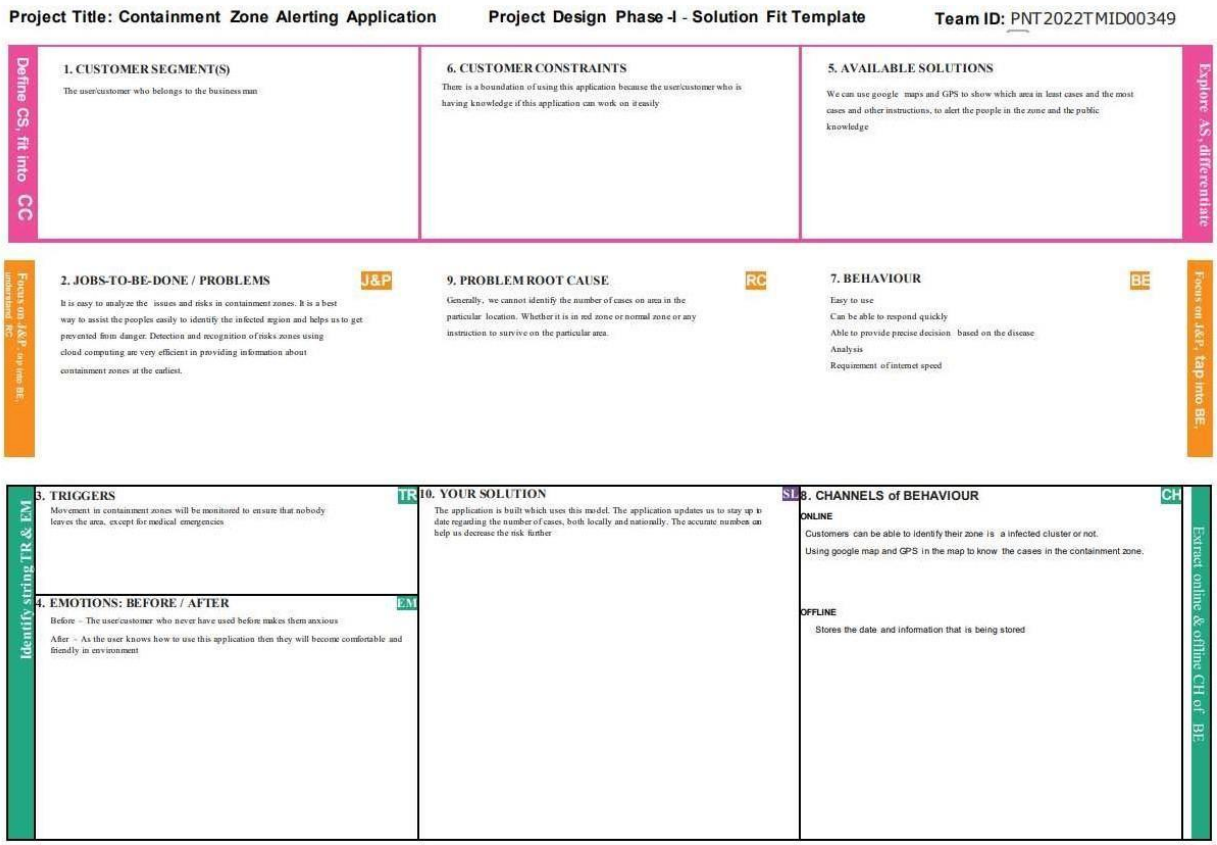
3.2 Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges



3.3 Ideation & Brainstorm

3.4 Problem Solution fit



4. Functional requirement

4.1 Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub- Task)
FR-1	User Registration	Registration through Gmail. Registration through mobile number.
FR-2	User Confirmation	Confirmation via Email. Confirmation via OTP.
FR-3	Authentication	It checking the confirmation of the password.
FR-4	Business rule	For subscriber's we give first 3 day's free trail. For unsubscriber's the user needs to watch some advertisement for knowing the zone alert for first 3 day's. **FR No. FR No.**

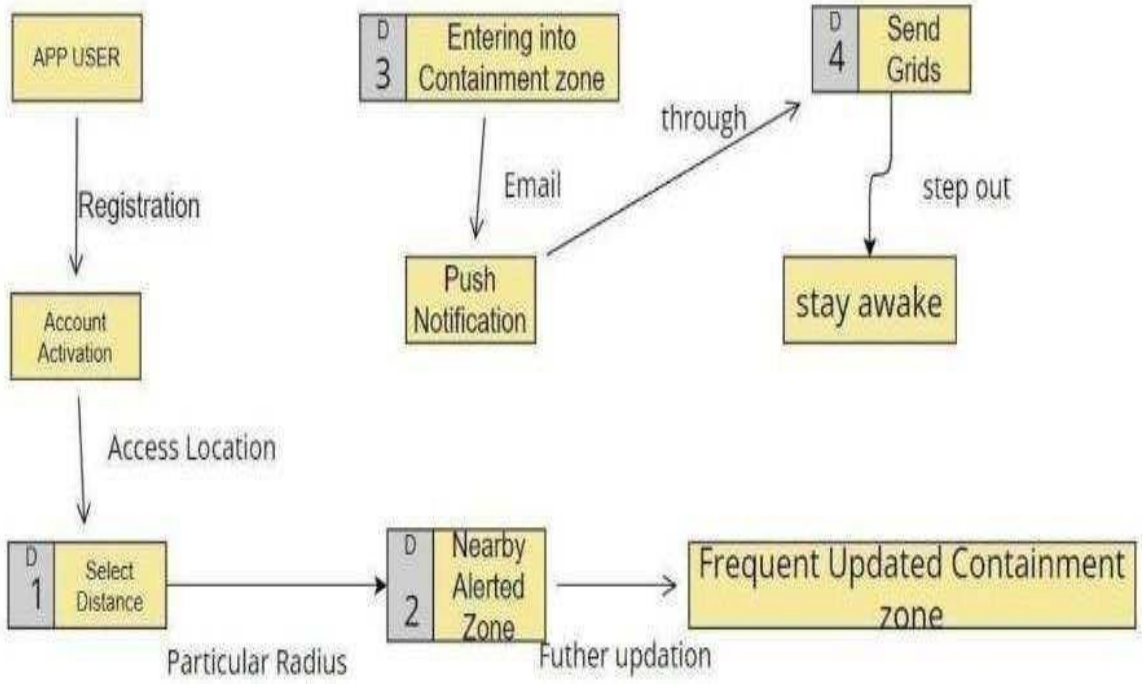
4.2 Non-Functional requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Providing recommendation link by using customer preference .
NFR-2	**Security**	The software team will issue some strong security code for the user's.
NFR-3	**Reliability**	The database update process must rollback all related updates when any update fails.
NFR-4	**Performance**	The loading speed of the server is quick and fast.

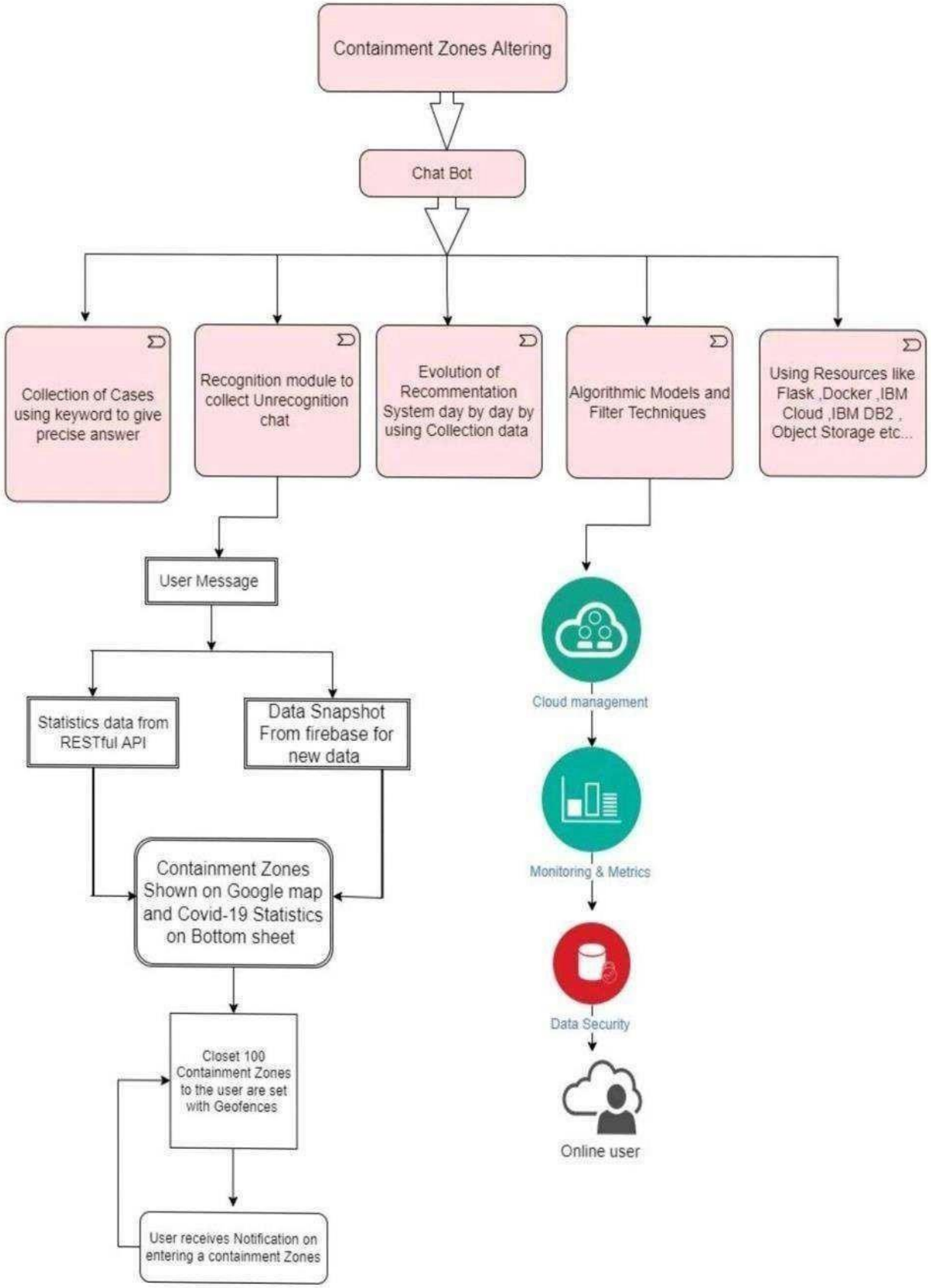
5. PROJECT DESIGN.

5.1 Data flow diagram:



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

5. 2.SOLUTION ARCHITECURE:



TECHNICAL ARCHITECTURE:

S.no	Component	Description	Technology

1.	User Interface	Mobile Application	HTML, CSS, JavaScript.
2.	Application Logic	Logic for a process in the application	Javascript
3.	Database	Data Type, Configurations etc.	Firebase, ibm cloud
4.	Cloud Database	Database Service on Cloud	IBM Cloud
5.	File Storage	File storage requirements	Local Filesystem and IBM cloud
6.	Infrastructure (Server / Cloud)	Application Deployment on Cloud Local Server Configuration	Local and Cloud Foundry

Application Characteristics:

S.no	Characteristics	Description	Technology
1.	Open-Source Frameworks	GitHub	Internet hosting service
2.	Security Implementations	Application security: Veracode.	Network automation
3.	Scalable Architecture	It provides the room for expansion more database of smart bins added additionally can be updated.	Cloud storage
4.	Availability	As the system control is connected to web server it is available 24*7 and can be accessed whenever needed.	Server, Appleixe, reple
5.	Performance	Performance is high it uses 5mb caches	Wireless Sensor Network

5.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Login	Registration (web and android)	USN-1	I can register for the application by	I can control my online account and dashboard.	Medium	Sprint-1

			entering my email and password			
Sign Up	Registration (web and android)	USN-2	I will receive a confirmation email once I have registered for the application	I can handle the waste collection.	High	Sprint-1
Services	Dashboard	USN-3	need to give permission to access my location	I can take the shortest path to reach the waste filled route specified.	Medium	Sprint-2
Services	Service	USN-4	I need to differentiate the containment zones	I can collect the trach, pull it to the truck, and send it out.	Medium	Sprint-3
Data collection	Service	USN-5	. I need to alert the user when they enter the containment zone through the notification	All of these processes are under my control.	High	Sprint-4

6 PROJECT PLANNING & SCHEDULING

6.1 Project Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	24 SEPTEMBER 2022

Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	25 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	27 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	28 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	30 SEPTEMBER 2022
Solution Architecture	Prepare solution architecture document.	28 SEPTEMBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	20 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	16 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	18 OCTOBER 2022

Technology Architecture	Prepare the technology architecture diagram.	18 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	26 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	26 OCTOBER 2022

Velocity:

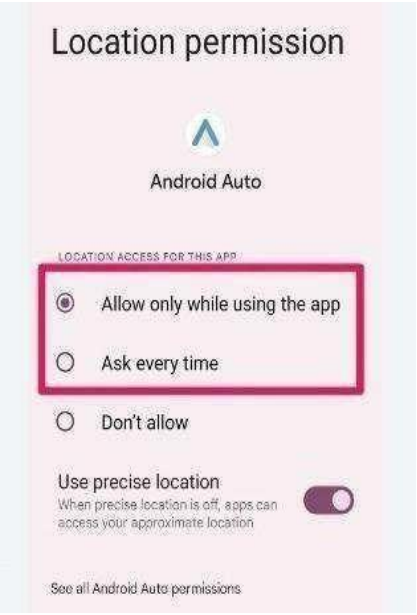
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

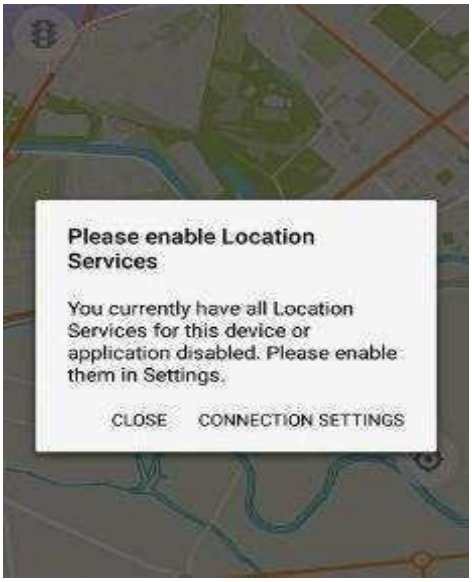
6.2 Sprint Delivery Schedule

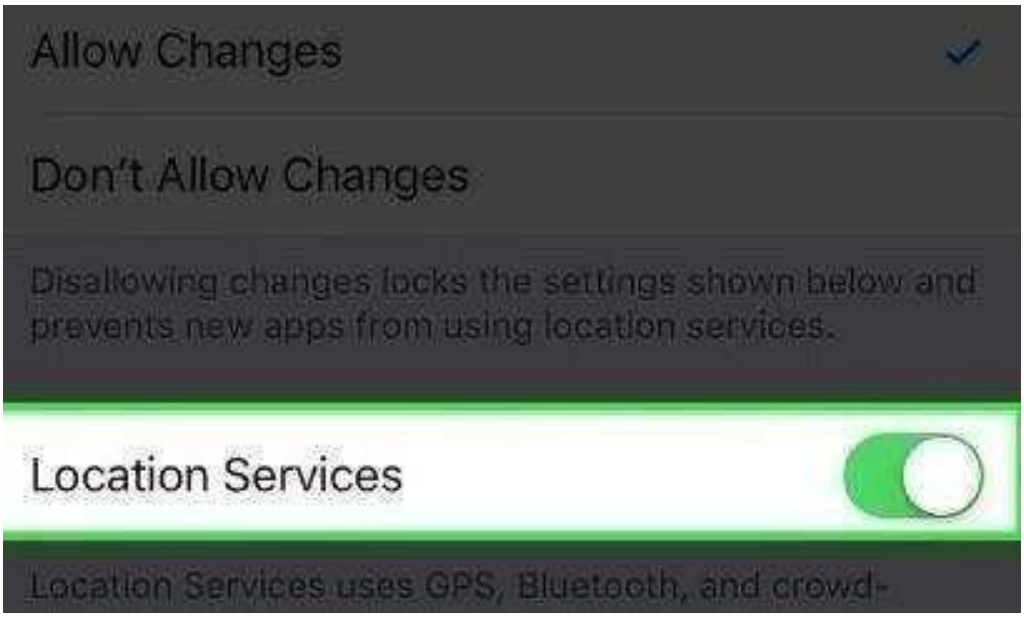
Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

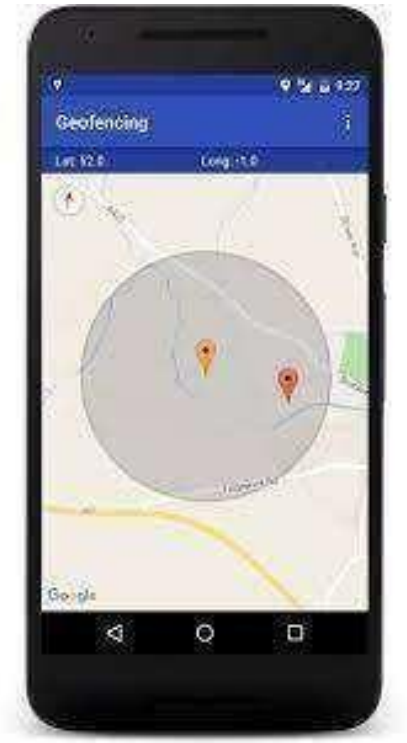
7. CODING & SOLUTIONING





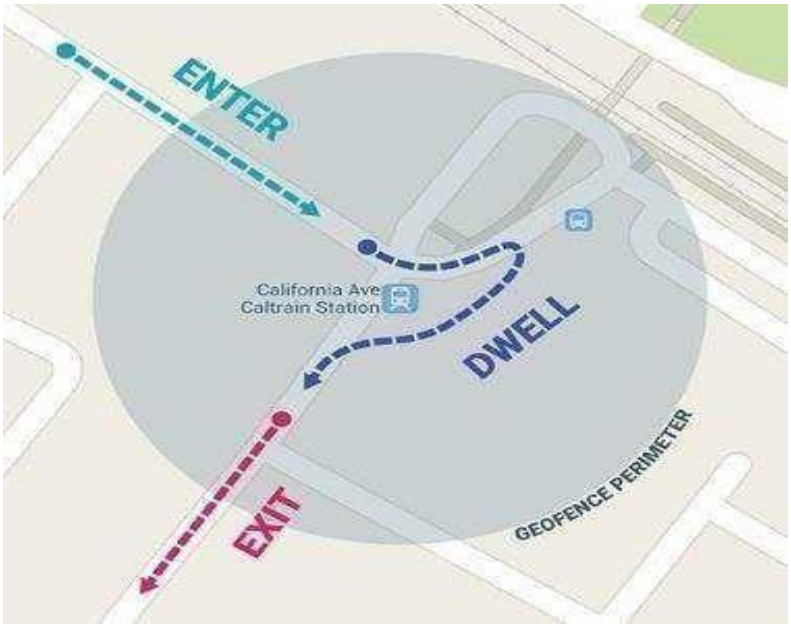


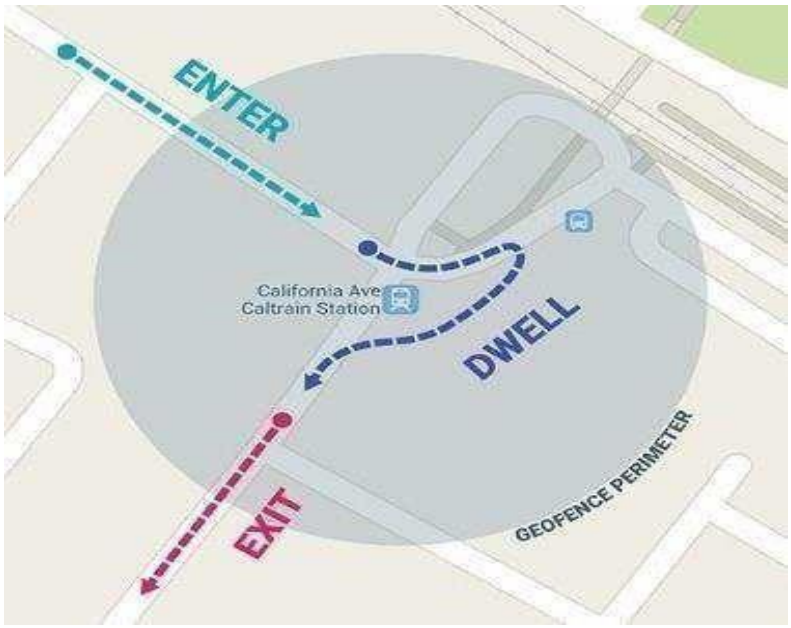
GEOFENCE IN ANDROID APP :




Geofencing
Lat 52.0
Long: -1.0


Remind me there
Park
Teatro Colseo
Shopping Mall
Buenos Aires
Choose the radius
600 meters
CONTINUE


ENTER
California Ave
Caltrain Station
DWELL
EXIT
GEOFENCE PERIMETER


ENTER
California Ave
Caltrain Station
DWELL
EXIT
GEOFENCE PERIMETER

8 TESTING

8.1 Test Cases

LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Enter URL (http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter valid password in password text box 5. Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password' validation message	Working as expected	PASS	Successful			RITHISH
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Enter URL (http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 5342 password: Testing123678678768 76876876	Application should show 'Incorrect email or password' validation message	Working as expected	PASS	Successful			NIHAL
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Enter URL (http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password' validation message	Working as expected	PASS	Successful			RAGHURAM

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1. Enter URL and click go 2. Scroll down 3. Verify login/Singup popup displayed or not	http://169.51.204.215:30106/	Login/Signup popup should display	Working as expected	PASS	Successfull			RITHISH, RAGHURAM
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1. Enter URL and click go 2.Click on Sigup button for User 3. Verify login/Singup popup with below UI elements: a.id text box b.password text box c. Login button d. New customer? Create account link e. Last password? Recovery password link	http://169.51.204.215:30106/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	PASS	Successful			VISHAL.NIHAL. RAGHURAM
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1. Enter URL(https://shopenzer.com/) and click go 2. Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5. Click on login button	ID: 5342 password: Testing123	User should navigate to user account homepage	Working as expected	PASS	Successful			RITHISH, RAGHURAM

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter In Valid ID in ID text box 4. Enter In valid password in password text box 5. Click on login button	ID: 5434 password: Testing123	Application should show 'correct email or password' validation message.	Working as expected	PASS	Successful			RITHISH, RAGHURAM
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5. Click on login button	ID: 1111 password: 5678	Application should show 'correct email or password' validation message.	Working as expected	PASS	Successful			VISHAL,NIHAL
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter In Valid ID in ID text box 4. Enter In valid password in password text box 5. Click on login button	http://169.51.204.215:30106/	Customer database is visible	Working as expected	PASS	Successful			VISHAL

LoginPage_TC_O10	Functional	USER REGISTER	Verify Id sent to customer email address	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Register the account by giving credentials 3. Click on button Submit	http://169.51.204.215:30106/	Email sent successfully	Working as expected	PASS	Successful			RITHISH
LoginPage_TC_O11	Functional	AGENT REGISTER	Verify AGENT is able to log into application with Valid Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter In Valid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 5342 password: Testing123	ID sent successfully	Application should show 'correct email or password' validation message.	PASS	Successful			RAGHURAM
LoginPage_TC_O12	Functional	Login page for ADMIN	Verify User is able to log into application with In Valid Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. Click on My Account dropdown button 3. Enter In Valid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 1111 password: 5678	Application should show 'Incorrect ID or password' validation message.	Working as expected	PASS	Successful			VISHAL
LoginPage_TC_O13	UI	Home page for Agent	Verify user is able to see the agent home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Agent Login page and submit Your Credentials	ID: 1111 password: 5678	AGENT Home Page popup should display	Working as expected	PASS	Successful			NIHAL

LoginPage_TC_O14	UI	Home page for USER	Verify user is able to see the User home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	USER Home Page popup should display	Working as expected	PASS	Successful			NIHAL
LoginPage_TC_O15	UI	Home page for ADMIN	Verify user is able to see the ADMIN home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			RITHISH
LoginPage_TC_O16	Functional	AGENT PAGE	On delete Button the user Credentials will be delected	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Admin Page and delect on User Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			RITHISH, RAGHURAM

8.2 User Acceptance Testing

1. Purpose of Document

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

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	3	1	2	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	40
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	13	12	25	78

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

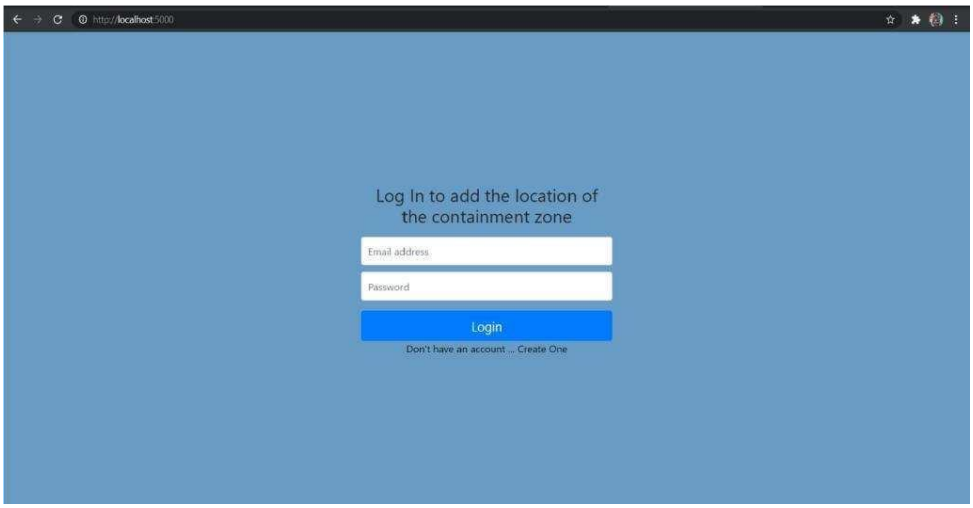
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	2	0	0	2

9. RESULTS:

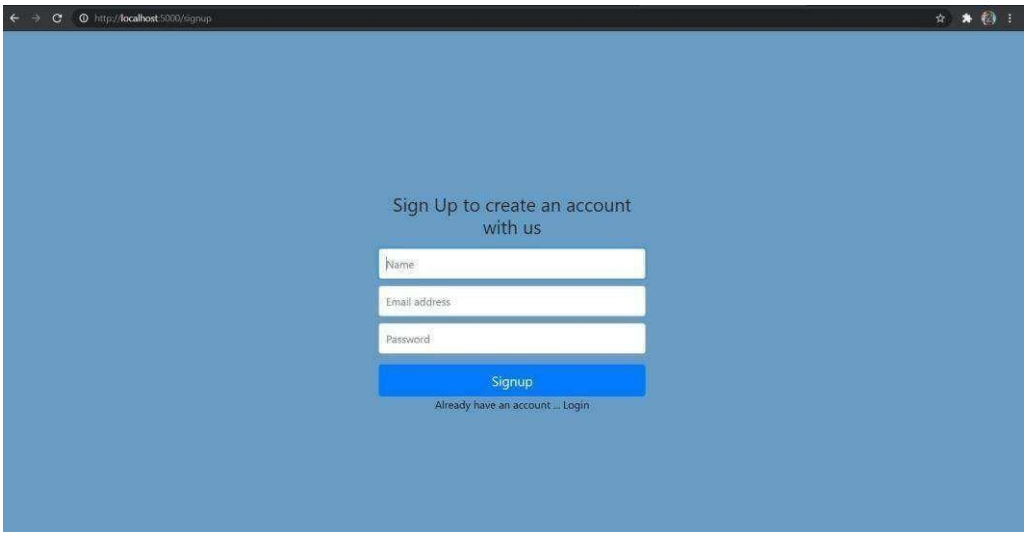
9.1 Performance Testing:

Admin App:

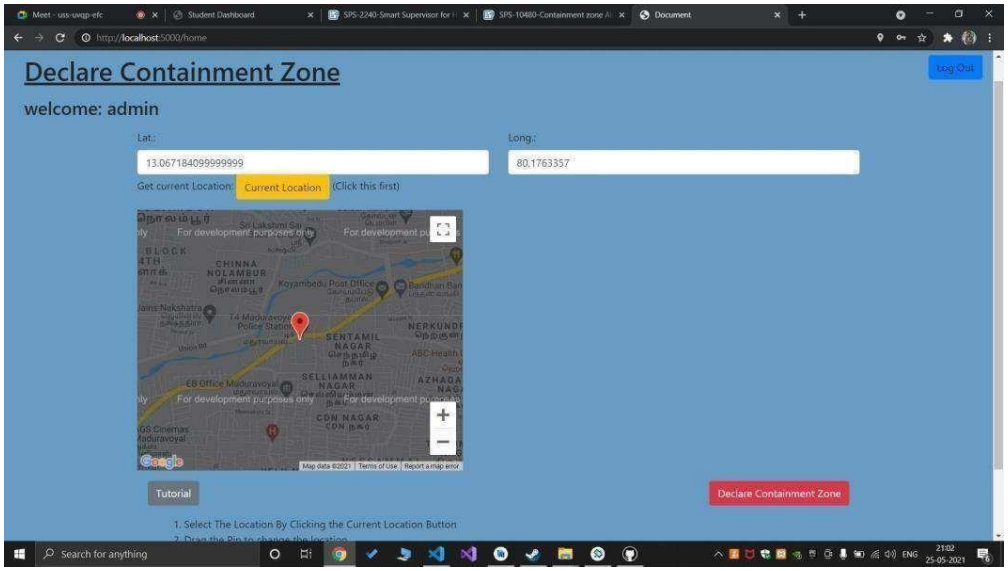
Login Page:


http://localhost:5000
Log In to add the location of
the containment zone
Email address
Password
Login
Don't have an account ... Create One

Register page:

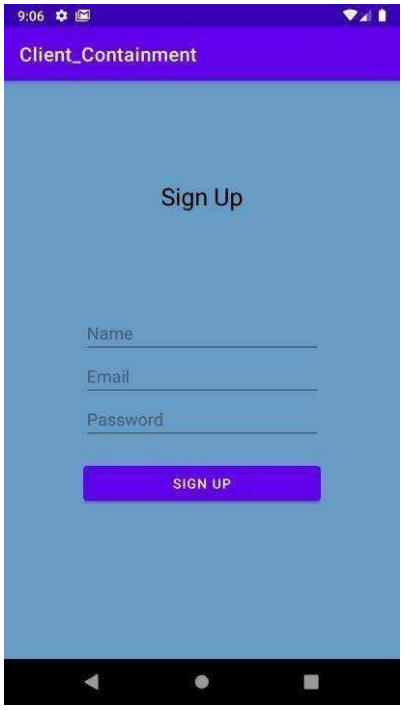


Home page:

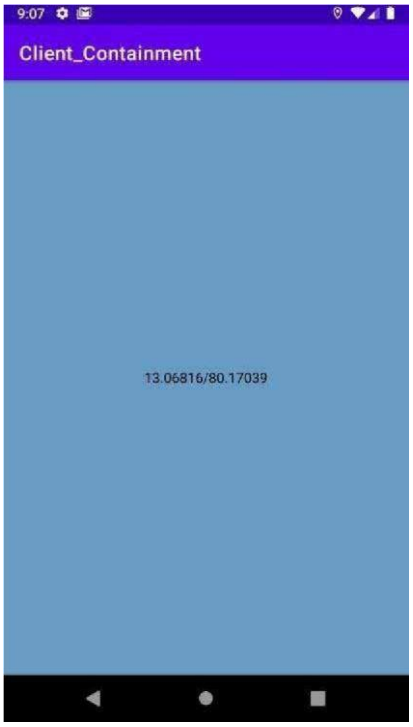


Location data page:

Client Application:
Register screen:



Current Location:



An Email will be sent to the registered mail id if the location is within 100 meters of the locations present in the admin app.

10. ADVANTAGES & DISADVANTAGES ADVANTAGES:

- People can be alerted before entering containment zone.
- Further spread of virus can be reduced considerably.

DISADVANTAGES:

- Accuracy of application depends on the number of data given to the application.
- Application's accuracy is directly proportional to the number of data given to the application
- about the infected patients.

11. CONCLUSION

This application is intended to provide information about containment zones in a particular region by alerting people, through continuous monitoring of an individuals location. Key benefits of the application are monitoring peoples activity and alerting them to their safety movements.

12. FUTURE SCOPE

Although we tried to cover almost all of the aspects during our developmental phase, however we were forced to leave some aspects because of lack of time as well as monetary and other reasons. Just like in the field of software development where there are always some shortcomings and room for improvement our application can be enhanced further:-

1) The application can include various government organization to help act faster.

2) The dataset obtained from the application can be used for predictive analysis to determine prone areas and include special method for tackling the problem in those areas.

3) Emergency signal in case of network failure and internet connection loss.

4) Tackling victim's movements.

5) Improved Google positioning system's precision.

6) The client part of application can be integrated in a single intelligent device.

For analysis purpose, we could use machine learning (ML) algorithms as well as data mining applications. There is a sub branch of machine learning known as time series analysis (TSA), which could be used to predict and analyze the data obtained through this application. Time series analysis is used to predict crop production as well as sales in different quarters.

13. APPENDIX

Source Code

```
# Project : CONTAINMENT ZONE ALERTING
APPLICATION # Team ID : PNT2022TMID00349

APP.PY
from logging import error from flask import *
from jinja2.utils import select_autoescape import
bcrypt from flask_mysqldb import MySQL
```

```
import json
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail #
initialization
app = Flask( name ) #
 config
app.secret_key                                                                    =
"\x19Ts\xbe\xe7\x8c_\r\x12Q\x14\x13>q\xb7'WTH0\x9f\xe4\xec\xb1
" app.config['MYSQL_HOST'] = 'localhost' app.config['MYSQL_USER']
=        'root'        app.config['MYSQL_PASSWORD']        =        "
app.config['MYSQL_DB'] = 'zone2' mysql
=    MySQL(app)    #
 functions            def
 send_mail(email):
   print(email)
   message = Mail(from_email='varundutia.h@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(
'SG.7BJDtQDlS8unH0r5_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG117YURqOTa
2HL') response = sg.send(message)
print(response.status.code)
print(response.body)
print(response.headers)          except
Exception as e:
print(e)
def create_bcrypt_hash(password): # convert the string to bytes password_bytes =
   password.encode()
   # generate a salt
   salt = bcrypt.gensalt(14) # calculate a hash as bytes
   password_hash_bytes = bcrypt.hashpw(password_bytes, salt) #
   decode bytes to a string
   password_hash_str = password_hash_bytes.decode() return password_hash_str def
verify_password(password, hash_from_database):
```

```
    password_bytes = password.encode() hash_bytes =
hash_from_database.encode()

    # this will automatically retrieve the salt from the hash,
    # then combine it with the password (parameter 1) # and then hash that, and compare it
to the user's hash does_match = bcrypt.checkpw(password_bytes, hash_bytes)

    return does_match #Api's

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

        # get the data from the form password = request.form['password'] email
= request.form['email'
    # initialize the cursor
      signup_cursor = mysql.connection.cursor()

       # check whether user already exists user_result = signup_cursor.execute(
        "SELECT * FROM USERS WHERE user_email=%s", [email]
       )

if(user_result > 0):
        data = signup_cursor.fetchone() data_password = data[3] if(verify_password(password,
data_password)):

    signup_cursor.close() session['id'] = data[0] session['name']
    = data[1] session['email'] = data[2] return
redirect(url_for("home")) else:
        return render_template('login.html', error=1) else:
        return render_template('login.html', error=2) return
render_template('login.html', error=3)

@app.route("/signup", methods=["POST", "GET"])
```

```
def verify_password(password, hash_from_database):
password_bytes = password.encode() hash_bytes
= hash_from_database.encode()
# this will automatically retrieve the salt from the hash, # then
combine it with the password (parameter 1) # and then hash
that, and compare it to the user's hash does_match =
bcrypt.checkpw(password_bytes, hash_bytes) return
does_match
# Api's
@app.route("/", methods=["GET",
"POST"]) def login(): if(request.method ==
"POST"):
# get the data from the form
password = request.form['password']
email = request.form['email'] #
initialize the cursor
signup_cursor = mysql.connection.cursor() # check
whether user already exists user_result
= signup_cursor.execute(
"SELECT * FROM USERS WHERE user_email=%s", [email]
) if(user_result >
0):
data = signup_cursor.fetchone() data_password =
data[3] if(verify_password(password,
data_password)): signup_cursor.close()
session['id'] = data[0] session['name'] =
data[1] session['email'] = data[2] return
redirect(url_for("home")) else: return
render_template('login.html', error=1) else:
return render_template('login.html', error=2)
return render_template('login.html', error=3)
@app.route("/signup", methods=["POST", "GET"])

def create_bcrypt_hash(password):
# convert the string to bytes password_bytes =
password.encode() #
generate a salt salt =
bcrypt.gensalt(14) #
calculate a hash as bytes
password_hash_bytes = bcrypt.hashpw(password_bytes, salt) # decode
bytes to a string
password_hash_str = password_hash_bytes.decode()
return password_hash_str PNT2022TMID48441
def verify_password(password, hash_from_database):
password_bytes = password.encode() hash_bytes =
hash_from_database.encode()
# this will automatically retrieve the salt from the hash,
```

```
# then combine it with the password (parameter 1)
#  and then hash that, and compare it to the user's hash
does_match = bcrypt.checkpw(password_bytes, hash_bytes)
return does_match
# Api's
@app.route("/", methods=["GET",
"POST"]) def login(): if(request.method ==
"POST"):
# get the data from the form password =
request.form['password'] email =
request.form['email'] # initialize the cursor
signup_cursor = mysql.connection.cursor() #
check whether user already exists user_result
= signup_cursor.execute(
"SELECT * FROM USERS WHERE user_email=%s", [email]
) if(user_result >
0):
data = signup_cursor.fetchone() data_password =
data[3] if(verify_password(password,
data_password)): signup_cursor.close()
session['id'] = data[0] session['name'] =
data[1] session['email'] = data[2] return
redirect(url_for("home")) else: return
render_template('login.html', error=1) else:
return render_template('login.html', error=2)
return render_template('login.html', error=3)
@app.route("/signup", methods=["POST", "GET"])
def signup():
if(request.method == "POST"):

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

    #  hash  the  password  pw_hash  =
    create_bcrypt_hash(password)

    # initialize the cursor
    signup_cursor = mysql.connection.cursor()

    # check whether user already exists user_result
= signup_cursor.execute(
      "SELECT * FROM USERS WHERE user_email=%s", [email]
    )
    if(user_result > 0):
      signup_cursor.close() return
```

```
        render_template('signup.html',
        error=True)
else:
                                                    # execute the query
signup_cursor.execute(
          'INSERT        INTO        USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)',  (
              name,  email,  str(pw_hash), "2"
          )
        )


        mysql.connection.commit()
signup_cursor.close()
        return redirect(url_for('login'))
   return render_template('signup.html',  error=False)
@app.route("/home",    methods=["POST",    "GET"])
def home():
if(session['id'] == None):
      return      redirect(url_for('login'))
def upload(): if(request.method
== "POST"):
# get the data from the form name   =
       request.json['name']
email  = request.json['email']  password
= request.json['password'] # hash the
password
pw_hash = create_bcrypt_hash(password) # initialize
the cursor
signup_cursor  =  mysql.connection.cursor()  #  check
whether user already exists user_result
= signup_cursor.execute(
"SELECT * FROM  USERS WHERE user_email=%s",  [email]
)   if(user_result   >   0):
signup_cursor.close()
return {'status': 'failure'}
else:
# execute the query signup_cursor.execute(
'INSERT INTO USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)',  ( name,
email,  str(pw_hash), "1"
)
)
mysql.connection.commit()  id_result
= signup_cursor.execute(
'SELECT user_id FROM USERS WHERE user_email  = %s', [email]
)   if(id_result   >
0):
```

```
id = signup_cursor.fetchone() return
{"id":   id[0]}   signup_cursor.close()
return {"status": "failure"}
@app.route("/get_all_users")
def getusers():
signup_cursor = mysql.connection.cursor() # check
whether user already exists user_result
= signup_cursor.execute( "SELECT * FROM USERS"

   if(request.method == "POST"): # get
      data
      lat = request.form["lat"]  lon
      =    request.form["lon"]
vis = 0
  if(lat == "" or lon == ""):
return render_template('home.html',  name=session['name'],  email=session['email'],
id=session['id'], success=0) # create a location cursor
location_cursor = mysql.connection.cursor() #
Execute the query location_cursor.execute(
'INSERT INTO LOCATION(location_lat,location_long,location_visited)  VALUES(%s,%s,%s)',  ( lat,
lon, vis
)
)
mysql.connection.commit()  location_cursor.close()
return render_template('home.html',  name=session['name'],  email=session['email'],  id=session['id'],
success=True)
return       render_template('home.html',       name=session['name'],       email=session['email'],
id=session['id'])

@app.route("/logout")
def logout():
# remove the username from the session if it is there
session['id']  =  None  session['name']  =  None
session['email'] = None return redirect(url_for('login'))
@app.route("/data")  def  data():  if(session['id']  ==
None):
return  redirect(url_for('login'))  location_cursor
= mysql.connection.cursor()  #  check  whether
user already exists user_result =
location_cursor.execute(  "SELECT * FROM
LOCATION"
)
if(user_result == 0):
return render_template("data.html",  responses=0) else:
res = location_cursor.fetchall()
print(res) return      render_template("data.html",
      responses=res) @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'] # hash the
password
pw_hash = create_bcrypt_hash(password) # initialize
the cursor
signup_cursor  =  mysql.connection.cursor()  #  check
whether user already exists user_result
= signup_cursor.execute(
"SELECT * FROM  USERS WHERE user_email=%s",  [email]
)   if(user_result   >   0):
signup_cursor.close()
return {'status': 'failure'}
else:
# execute the query signup_cursor.execute(
'INSERT INTO USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)',  ( name,
email,  str(pw_hash), "1"
)
)
mysql.connection.commit()  id_result
= signup_cursor.execute(
'SELECT user_id FROM USERS WHERE user_email  = %s', [email]
)   if(id_result   >
0):
id = signup_cursor.fetchone()
return         {"id":         id[0]}
signup_cursor.close()   return
{"status":                 "failure"}
@app.route("/get_all_users")
def getusers():
signup_cursor  =  mysql.connection.cursor()  #  check
whether user already exists user_result
= signup_cursor.execute( "SELECT * FROM
USERS" PNT2022TMID48441
)   if(user_result   >
0):
rv = signup_cursor.fetchall()
row_headers         =         [x[0]         for         x         in
signup_cursor.description] json_data = [] for result in
rv:
json_data.append(dict(zip(row_headers,  result))) return
json.dumps(json_data)
@app.route("/post_user_location_data",
methods=["POST"])  def post_user_location():
```

```
if(request.method == "POST"):
# get the data from the form lat
= request.json['lat'] lon =
request.json['long'] id
= request.json['id'] ts =
request.json['timestamp'] #
initialize the cursor
user_location_cursor = mysql.connection.cursor() # execute
the query user_location_cursor.execute(
'INSERT INTO USER_LOCATION(location_lat,location_long,user_id,timestamp)
VALUES(%s,%s,%s,%s)', (
lat, lon, id, ts
)
)
mysql.connection.commit()
return {"response": "success"}
@app.route("/location_data") def
location_data():
location_cursor = mysql.connection.cursor() # check
whether user already exists user_result =
location_cursor.execute( "SELECT * FROM
LOCATION"
) if(user_result !=
0):
res = location_cursor.fetchall() print(res)
row_headers = [x[0] for x in location_cursor.description]
json_data = []
PNT2022TMID48441 for
result in res:
json_data.append(dict(zip(row_headers, result)))
return json.dumps(json_data) else:
return {"response": "failure"}
@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'] location_cursor =
mysql.connection.cursor() # check whether user
already exists user_result =
location_cursor.execute(
"SELECT location_visited FROM LOCATION WHERE location_id=%s", [
location_id]
)
if(user_result == 0):
return {"response": "failure"}
else:
```

```
res = location_cursor.fetchone()
print(res[0]) visited =
res[0] visited
= visited+1
location_cursor.execute(
"UPDATE LOCATION SET location_visited  = %s WHERE location_id=%s",
(visited, location_id)
)
mysql.connection.commit()  send_mail(email)
return {"response": "success"} #
main if_name == "_main_":
app.run(host='0.0.0.0', port=5000)
```

DATA.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Zones</title>
<link rel="stylesheet"
href="https:                    /stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous"  />
<style>  body  {  padding-top:
30px; padding-bottom:  30px;
background-color: #699cc5;
}  a  {  color:
black;
}
</style>
</head>
<body>
<div class="m-4 container">
<h1><u>Location data and Visited People</u></h1>
</div>
<div class="m-4 container">
<table class="table">
<thead>
<tr>
<th scope="col">S.No</th>
<th scope="col">Latitude</th>
<th scope="col">Longitude</th>
<th scope="col">No_Visited</th>
```

```
</tr>
</thead>
<tbody>
{%- for row in responses %}
<tr>
<th scope="row">{{loop.index}}</th>
<td>{{row[1]}}</td>
<td>{{row[2]}}</td>
<td>{{row[3]}}</td>
</tr>
{%- endfor %}
</tbody>
</table>
</div>
<div class="m-3 float-right">


<button type="button" class="btn btn-danger"><a href={{url_for("home")}}>Go to location update
Page</a></button>
  </div>

</body>

</html>
```

HOME.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
<link rel="stylesheet"
href="https: /stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
                integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous" />
  <style> body {
      padding-top: 30px;
padding-bottom: 30px; background-color:
      #699cc5;
    }
a { color: black;
    }
  </style> </head>
```

```
<body>
  {% if success == True %}
  <script>  alert("Location  Uploaded
    Successfully");
  </script>
  {% elif success == 0 %}
  <script> alert("Enter Proper Location
    data");
  </script>
  {% endif %}
  <div class="m-3 float-right">
    <button  type="button"  class="btn  btn-primary"><a  href={{url_for("logout")}}>Log
Out</a></button>
  </div>
  <div class="container m-3">
    <h1><u>Declare Containment Zone</u></h1>
  </div>
  <div class="container m-3">
    <h3>welcome: {{name}}</h3>
  </div>
  <form method="POST" action="/home">
    <div class="container">
      <div class="form-group row">
        <div class="col-sm-6">
          <label class="control-label">Lat.:</label>
          <input type="text" class="form-control" id="lat" name="lat" />
        </div>
        <div class="col-sm-6">
          <label>Long.:</label>
          <input type="text" class="form-control" id="lon" name="lon" />
        </div>
        <div class="col-sm-6">
          <label>Get current Location:</label>
          <button type="button" class="btn btn-warning" onclick="getLocation()">Current
Location</button>
          <label>(Click this first)</label>
        </div>
      </div>

      <!-- map -->
      <div id="map_disp" style="height: 400px;width: 500px;"></div>
      <div class="m-3 float-right">
        <button type="submit" class="btn btn-danger">Declare Containment Zone</button>
      </div>
      <div class="m-3">
      <button  onclick="toggleTips()"  type="button"  class="btn
```

```
btnsecondary">Tutorial</button>
          <div id="tips" class="m-3">
            <ol>
              <li>Select The Location By Clicking the Current Location Button</li>
              <li>Drag the Pin to change the location</li>
              <li>Click on Declare Containment Zone to save the location to the database </li>
            </ol>
            </div>
        </div>
        <div class="m-3 float-right">
         <button type="button" class="btn btn-warning"><a href="{{url_for('data')}}">Click Here
To View The
              Containment Zones and Number of



              people visited</a> </button>
        </div>
      </div>


      <script        src="https:        /cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"
                                                              integrity="sha384-
+YQ4JLhjyBLPDQt  /I+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd026JF" cros
sorigin="anonymous">
</script>
      <script src="https: /code.jquery.com/jquery-2.2.4.min.js">
</script>                                  <script
src="https:  /maps.google.com/maps/api/js?sensor=false&amp;libraries=places"></script>  <script
                                                    src="https: /rawgit.com/Logicify/jquery-
locationpickerplugin/master/dist/locationpicker.jquery.js"></script>

      <script>          function
        getLocation()
{
if (navigator.geolocation)
{ navigator.geolocation.getCurrentPosition(showPosition);
          }   else   {   alert("No
            location");
          }


        }
        function  showPosition(position)  {
$('#map_disp').locationpicker({   location:
{
              latitude: position.coords.latitude,
longitude: position.coords.longitude
            },
```

```
                radius: 0,
inputBinding:
{
                    latitudeInput: $('#lat'), longitudeInput:
                    $('#lon'), },
                enableAutocomplete:  true,
                onchanged: function (currentLocation, radius, isMarkerDropped)
{
 / Uncomment line below to show alert on each Location  Changed event
                     / alert("Location  changed. New location  (" + currentLocation.latitude  + ", " +
currentLocation.longitude  + ")");
                }
            });
        }
        function     toggleTips()     {     var     x     =
            document.getElementById("tips");  if
(x.style.display === "none") {
   x.style.display = "block";
              } else {
                 x.style.display = "none";
} }
</script>
</body>
</html>
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-12528-1659452832

Demo Link:

https://www.youtube.com/watch?v=3MqKJV4L_Sk