

CONTAINMENT ZONE ALERTING APPLICATION

PROJECT REPORT

Submitted by

AJAYKUMAR J (710719104007)

JENCY S (710719104048)

KARTHIKA S (710719104052)

AARTHI A P (710719104001)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY: CHENNAI – 600 025

NOV 2022

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

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), 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 (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.

1.2. 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:

S. No	Title	Technology Used	Algorithm /Tools	Details
1	Application for Covid-19 Real Time Counter (2022)	Java	Android Studio	This project shows the real time count of the COVID-19 cases and spread awareness to the people about the outworld pandemic
2	S-Nav: Safety-Aware IoT Navigation Tool for Avoiding COVID-19 Hotspots (2021)	Internet of Things (IoT)	Q-learning model, reinforcement learning (RL), shortest path.	This project navigates people from not entering COVID-19 hotspot region and shows the safe path to make their journey safe
3	Tracking the Covid zones through geo-fencing technique (2020)	Data science	RSA algorithm	This paper developed an algorithm to track people from entering containment zone and alerting before they getting into risk
4	A Compact Wearable-IoT (W-IoT) System for Health Safety and Protection of Outgoers in the Post- Lockdown World (2019)	Internet of Things (IoT)	electronic face mask, automatic sanitizer dispenser	This project created a prototype that make an analysis on our health conditions like monitoring our health conditions and giving analysis
5	Development of An Android Application for Viewing Covid-19 Containment Zones and Monitoring Violators (2020)	Cloud & Flask	Geo-fencing Algorithm	It is an application that developed to show alert message when people enter the containment zone

2.1. EXISTING SYSTEM:

There are several applications that exist to alert people of containment zones. One such application is the Aarogya Setu app, which was developed by the Indian government. The app uses GPS to track the location of the user and sends alerts if the user enters a containment zone. Other apps that provide similar functionality include the Covid-19 India app and the Corona Kavach app.

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

PAPER 2:

TITLE: Geofencing 2.0: Taking Location-based Notifications to the Next Level

AUTHOR NAME: Sandro Rodriguez Garzon Bersant Deva

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

PAPER 4:

TITLE: Aarogya Setu

AUTHOR NAME: National Informatics Centre, Ministry of Electronics & Information Technology, Government of India

2.3. Problem Statement Definition:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A passenger	Go to my friend's house	I don't know if it is safe for me	Covid-19 infections are spreading rapidly	Worried and scared for my safety

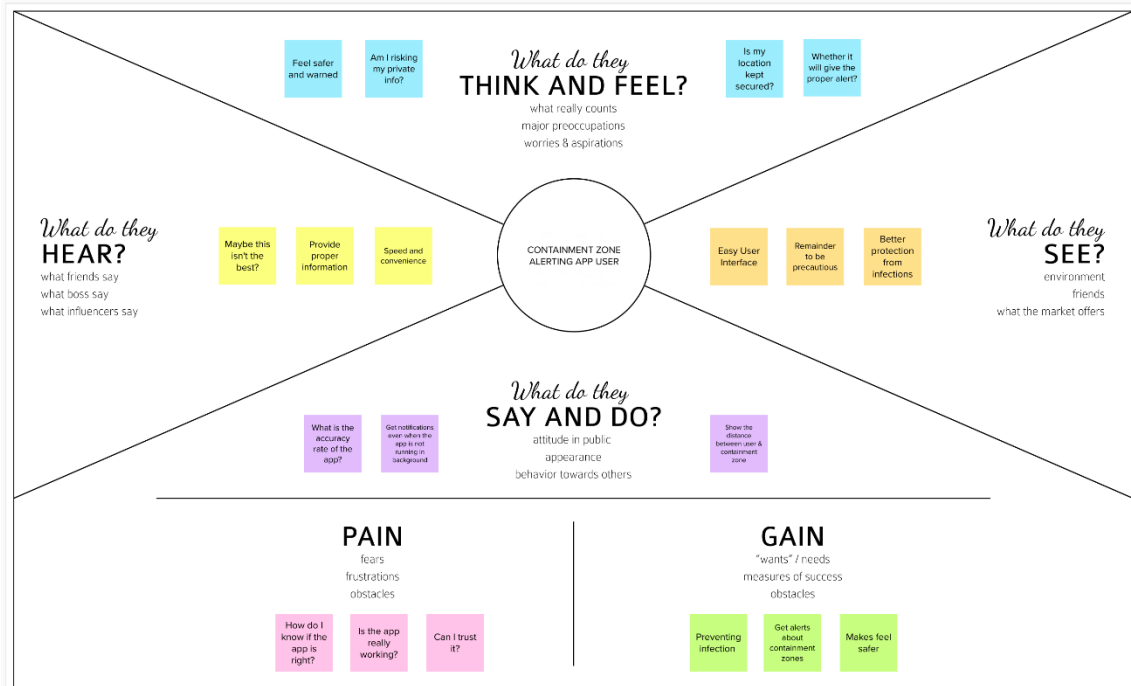
PS-2	A social worker	Help people during the pandemic	I want to know the containment zones	I have to take precautionary measures while travelling	Prepared and Warned on my movements
PS-3	Cab Driver	Get unaffected and shortest routes for travel	I am getting unclear and indirect routes	There are no proper guidance or reference map for containment zones	Annoyed and Avoided
PS-4	Police	To warn people on the containment zones and prevent people from trespassing	Getting incorrect results	Zones are not getting updated regularly	Uncertain of the zones

3. IDEATION & PROPOSED SOLUTION:

3.1. 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 help 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.

Containment Zone Alerting Application




3.2. IDEATION & BRAINSTORMING:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

STEP – 1:

Team Gathering, Collaboration and Select the Problem Statement



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare
🕒 1 hour to collaborate
👤 2-8 people recommended

[Share template feedback](#)

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

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

During this pandemic, All are advised to maintain social distance. But everyone is busy to keep note of the containment zones and so it is easier for the spread of Covid-19 infection.



Key rules of brainstorming

To run an smooth and productive session

🗣️ Stay in topic.	💡 Encourage wild ideas.
⏸️ Defer judgment.	👂 Listen to others.
🗣️ Go for volume.	👁️ If possible, be visual.

STEP-2 Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

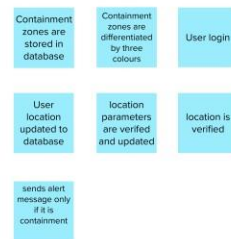
AARTHY A P



JENCY S



KARTHIKA S



AJAYKUMAR J



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, 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

User Login



Updating Zones



Collecting Details



Verifying Location



Differentiating Zones



Sending Alerts



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:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	During the pandemic, people are unable to keep track of the containment zones since they are not regularly updated and informed to the public in advance.
2.	Idea / Solution description	<p>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.</p> <p>Containment zones will be identified throughout the country and decided into red, orange & green zones.</p> <ul style="list-style-type: none">• Red Zone – Infection Hotspots• Orange Zone – Some Infection Zone• Green Zone – No Infection Zone
3.	Novelty / Uniqueness	Key benefits of the application are monitoring people's activity and alerting them of their safety movements. The monitoring process will be based on the containment zones which is added by the Admin. The app captures the user's IMEI.
4.	Social Impact / Customer Satisfaction	Considering the areas affected by Covid-19 across the country, this application helps in providing safety to public and the government officials.
5.	Business Model (Revenue Model)	Social media is the best platform to develop this application. This application will increase the confidence among the people. It is great to use, amazing convenience and have subscription once user hits certain services.
6.	Scalability of the Solution	This app helps the police to keep an eye on the people who are frequently violating the lockdown rules and monitor them.

3.4. PROBLEM SOLUTION FIT:

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS <ul style="list-style-type: none">The solution is intended for all customer segments since it is a general health-related application	6. CUSTOMER CONSTRAINTS CC <ul style="list-style-type: none">The proposed solution requires the customer to allow location access and be connected to the Internet always	5. AVAILABLE SOLUTIONS AS <ul style="list-style-type: none">Newspapers inform the public of Containment ZonesContainment Zones could be searched online by the people	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P <ul style="list-style-type: none">Analysis of COVID-19 StatisticsIdentification of Containment ZonesDetection of user entry into Containment Zones	9. PROBLEM ROOT CAUSE RC <ul style="list-style-type: none">When entering a place, people are not aware of whether the place is a Containment Zone or notAs a result, there is a higher risk of them getting affected by COVID-19	7. BEHAVIOUR BE <ul style="list-style-type: none">The proposed solution tracks user locationOnce an user enters a Containment Zone, instant alert delivery is doneCOVID statistics are shown by the app	
Focus on J&P, tap into BE, understand RC	3. TRIGGERS TR <ul style="list-style-type: none">Whenever an user gets into a Containment Zone, the proposed solution(app) will alert the user immediately	10. OUR SOLUTION SL <ul style="list-style-type: none">An Alerting application that delivers instant alerts whenever a person enters a COVID Containment ZoneReal Time Tracking of user locationDisplay of COVID statistics & precautions	8. CHANNELS OF BEHAVIOUR CH 8.1 ONLINE <ul style="list-style-type: none">Update of Containment Zones and COVID Statistics	Extract online & offline CH of BE
	4. EMOTIONS EM <ul style="list-style-type: none">Before : Users might feel fearful, knowing if a zone is a Containment zone or notAfter : No need for the user to panic since the proposed solution would alert the user of Containment Zones every now and then		8.2 OFFLINE <ul style="list-style-type: none">Pre-loaded / downloaded data can be viewed offline	
Identify strong TR & EM				

4. REQUIREMENT ANALYSIS:

4.1. FUNCTIONAL REQUIREMENTS:

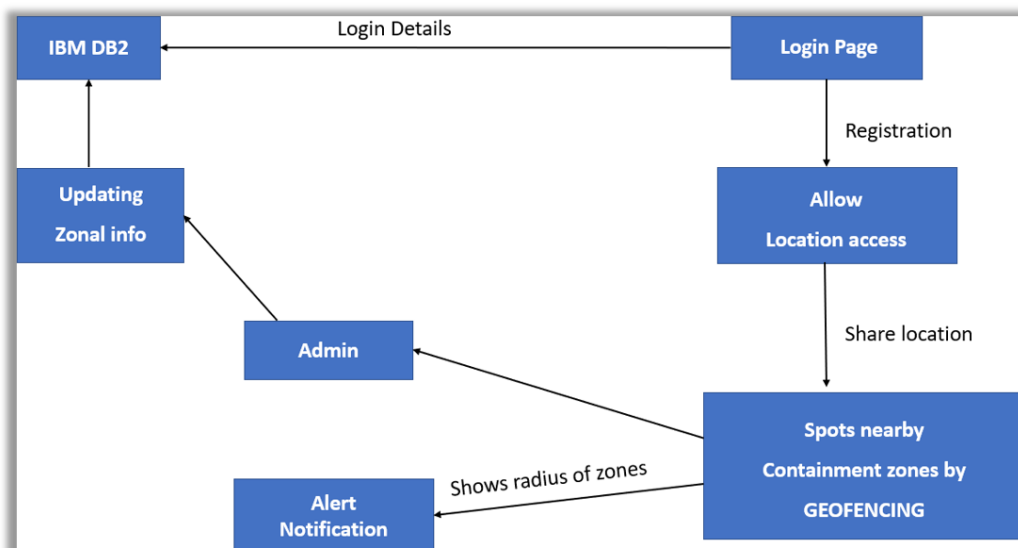
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Mobile app using mobile number.
FR-2	User Confirmation	Confirmation via OTP.
FR-3	User Login	Login using Mobile number and password.
FR-4	Track The Location	Tracking their location using GPS system.
FR-5	Show Containment Zones	Name and details of zones are given. Infected people count and categories of containment zones are shown. Shows the level of containment zone by Geo-fencing.
FR-6	Notification Alert	By sending an SMS to the phone number if user enters the containment zone.
FR-7	Monitoring and Updating of Containment Zones	Trace the trespassers by using Google Map API and update the containment zones from trust-worthy resources.

4.2. NON – FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	<ul style="list-style-type: none">• The application should provide a good UI/UX design with seamless navigation across sections of the app.• It can be used by a wide variety of users as it is very simple to learn and not complex to proceed .• Easy to use, User-friendly and Responsive.
NFR-2	Security	<ul style="list-style-type: none">• The passwords must be hashed and only the hashed values should be stored in the database.• Privacy and security of the user must be maintained.
NFR-3	Reliability	<ul style="list-style-type: none">• Users can trust this application and travel safely.• Data corruption should not exist.
NFR-4	Performance	The accurate result can be achieved due to real-time location sharing, Geo-fencing and GPS.
NFR-5	Availability	App works properly if the network bandwidth of the user is of good range.
NFR-6	Scalability	The system's performance should not be affected by a change in the number of users and the number of containment zones.

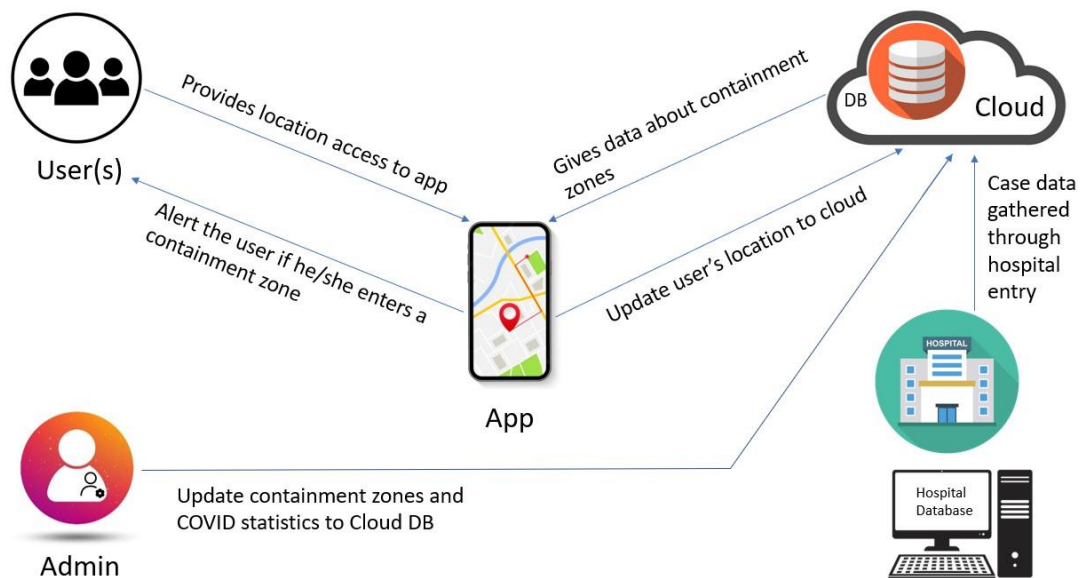
5. PROJECT DESIGN:

5.1. DATAFLOW DIAGRAM:

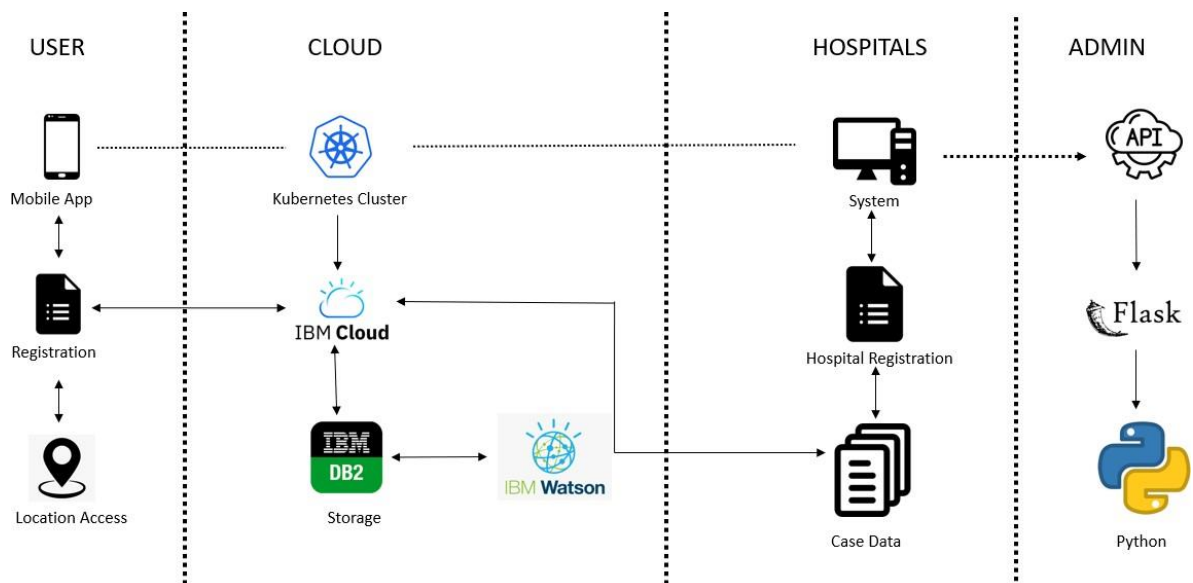


5.2. SOLUTION & TECHNICAL ARCHITECTURE:

Solution Architecture:



Technical Architecture:



Components & Technologies:

S. No	Component	Description	Technology
1	User Interface	User access to the application through the mobile application	HTML, CSS, JavaScript, Flask
2	Application Logic-1	Creating an application interface	Python & Flask
3	Application Logic-2	Creating an AI assistant that gives medical services to the user	IBM Watson Assistance
4	Application Logic-3	Sending notifications to the user when he is in a containment zone.	Send Grid service
5	Cloud Database	DB2 product is extended with the support of Object- Oriented features and non-relational structures with XML	IBM DB2 & Mango DB
6	File Storage	Files are stored in the local storage and stored in the cloud.	IBM Block Storage or Other Storage Service or Local Filesystem
7	External API-1	Use this REST API to manage locations. Get all locations. URI, /admin/resources/locations. Method, GET	IBM Location REST API
8	Deep Learning Model	Creating an algorithm to calculate case information provides by the hospitals	Object Recognition Model
9	Infrastructure (Cloud)	IBM Cloud App Configuration is a centralized feature-management and configuration service on IBM Cloud.	IBM Cloud Foundry & Kubernetes

Application Characteristics:

S. No	Characteristics	Description	Technology
1	Open-Source Frameworks	There are no open-source frameworks in this application.	Python
2	Security Implementations	Blockchain technology is used for Security implementation its private framework protects all data	Block – chain
3	Scalable Architecture	Users are provided with medical services online and giving awareness to people by giving therapeutic medicines and monitoring user movements in pandemic zones and alerts before they are affected.	IBM Cloud
4	Availability	Medicinal Recommendations, Test kits, Doctor suggestions, and Updated Contaminated zones are available in applications.	IBM Watson Assistant
5	Performance	The geo-fencing algorithm is updated daily and shows the day-to-day updates of the contaminated zones.	Geo – fence

5.3. 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 Gmail	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	Medium	Sprint – 2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
				background		
	GPS	USN – 7	As a user, I allow the app to access my location	I should accept the permission to access my location	Medium	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	Medium	Sprint – 2
	Notifications	USN – 9	As a user, I allow notification access for the application.	I should allow notification access	Medium	Sprint – 2
Administrator	Login	USN – 1	As admin, I log into the administrator portal.	I can access the admin account.	High	Sprint – 1
	Services	USN – 2	As admin, I use the cloud services to maintain users and the contaminated zones data.	I work with cloud services	High	Sprint – 3
	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.	Medium	Sprint – 3
	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 System	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 – 4
	Updating Zones	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
	Data Collections	USN - 7	As an admin, I need to store all the user information on the cloud	I can access the information when I needed	Medium	Sprint - 4

6. PROJECT PLANNING & SCHEDULING:

6.1. SPRINT PLANNING & ESTIMATION:

TITLE	DESCRIPTION	DATE
Literature Survey	Literature survey on the selected project by gathering information.	24 OCTOBER 2022
Empathy Map	Prepare Empathy Map Canvas to capture the user	24 OCTOBER 2022
Ideation	List the by organizing the brainstorming session to prioritize the top 3 ideas based on the feasibility & importance.	24 OCTOBER 2022
Proposed Solution	Prepare the proposed solution document.	28 OCTOBER 2022
Problem Solution Fit	Prepare problem - solution fit.	29 OCTOBER 2022
Solution Architecture	Prepare solution architecture.	29 OCTOBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences.	30 OCTOBER 2022
Functional Requirements	Prepare the functional requirements document.	30 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams. submit for review.	30 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram for the project.	30 OCTOBER 2022
Prepare Milestone & activity List	Prepare the milestones & activity list of the project.	4 NOVEMBER 2022
Project Delivery of Sprint-1, 2, 3 & 4	Develop and submit the developed code by testing it.	19 NOVEMBER 2022

6.2. SPRINT DELIVERY SCHEDULE:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint - 1	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	4	High	Jency.S
Sprint - 1		USN - 2	As a user, I will receive confirmation email once I have registered for the application	4	High	Ajaykumar. J
Sprint - 1		USN - 3	As a user, I can register for the application through Gmail	4	Medium	Karthika.S

Sprint - 1	Login	USN - 4	As a user, I can log into the application by entering email & password	4	High	Aarthi.A.P
Sprint – 2	Dashboard	USN - 5	As a user, I can see the options available for User account.	4	Medium	Jency.S
Sprint – 2	Background running	USN – 6	As a user, I allow the app to run in background.	4	Medium	Ajaykumar. J
Sprint – 2	GPS	USN – 7	As a user, I allow the app to access my location	4	Medium	Karthika.S
Sprint – 2	Google Maps	USN – 8	As a user, I can see the containment zones using the maps via Google Maps.	4	Medium	Aarthi.A.P
Sprint – 2	Notifications	USN – 9	As a user, I allow notification access for the application.	4	Medium	Jency.S
Sprint – 1	Login	USN – 1	As admin, I log into the administrator portal.	4	High	Ajaykumar. J
Sprint – 3	Services	USN – 2	As admin, I use the cloud services to maintain users and the contaminated zones data.	7	High	Karthika.S
Sprint – 3	Cloud Database	USN – 3	As admin, I store the user details in the cloud database.	5	Medium	Aarthi.A.P

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint – 3	Maps	USN – 4	As admin, I will enter the containment zone's location.	8	High	Jency.S
Sprint – 4	Mail System	USN – 5	As admin, I set up a mail system to alert users when they enter a containment zone.	8	High	Ajaykumar. J
Sprint - 4	Updating Zones	USN - 6	As admin, I should frequently update the details and the location of the containment zones.	5	High	Aarthi.A.P
Sprint - 4	Data Collections	USN - 7	As an admin, I need to store all the user information on the cloud	7	Medium	Jency.S

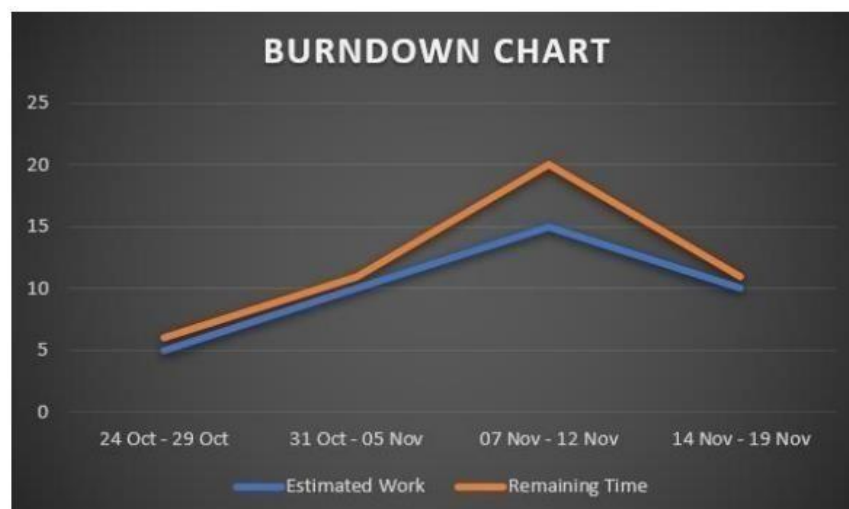
Project Tracker, Velocity & Burndown Chart:

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	03 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	10 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	17 Nov 2022

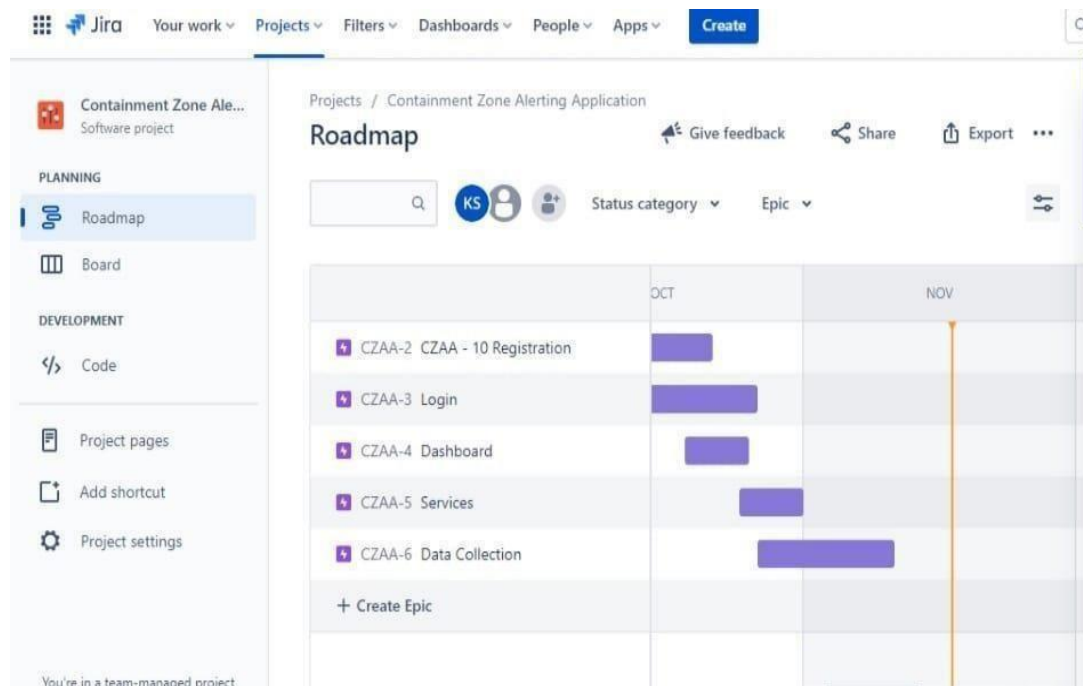
Velocity:

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

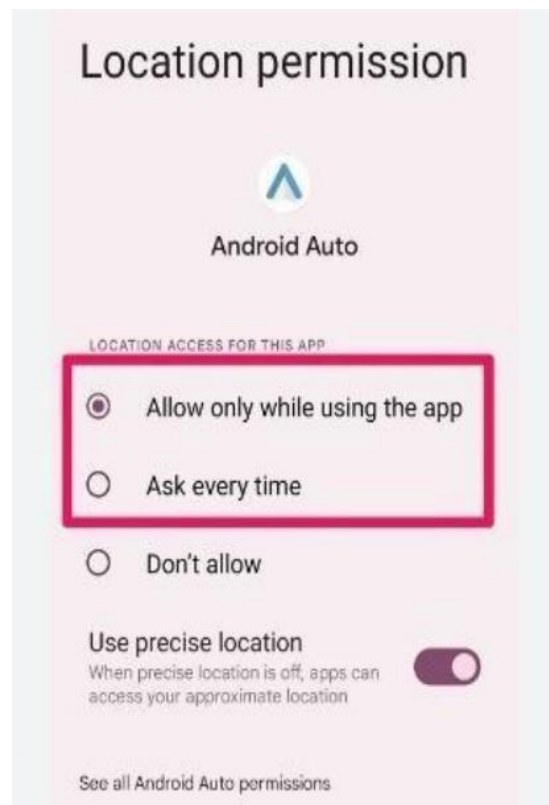
$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

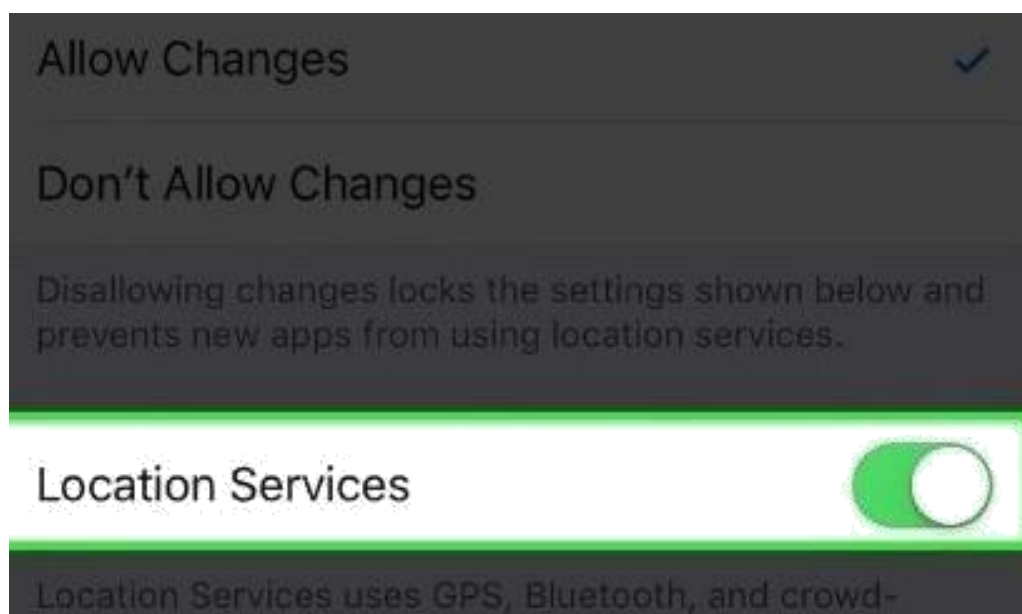
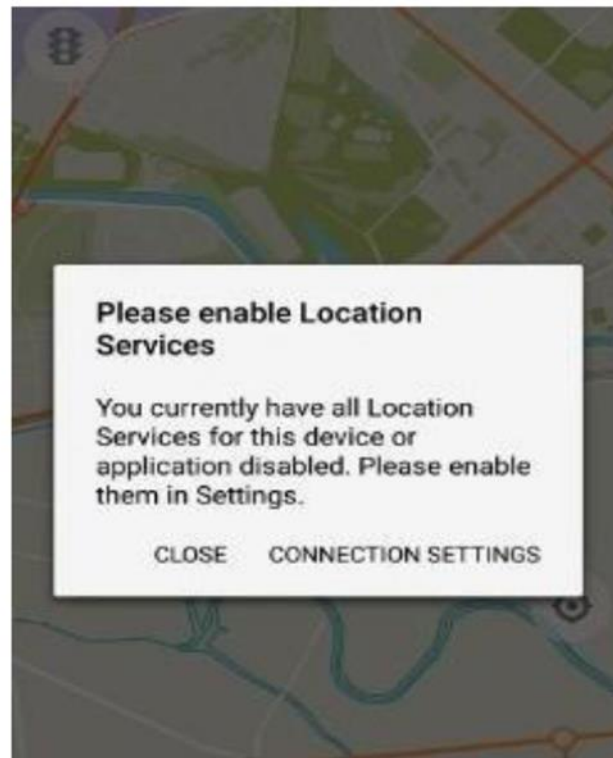


6.3. REPORTS FROM JIRA:

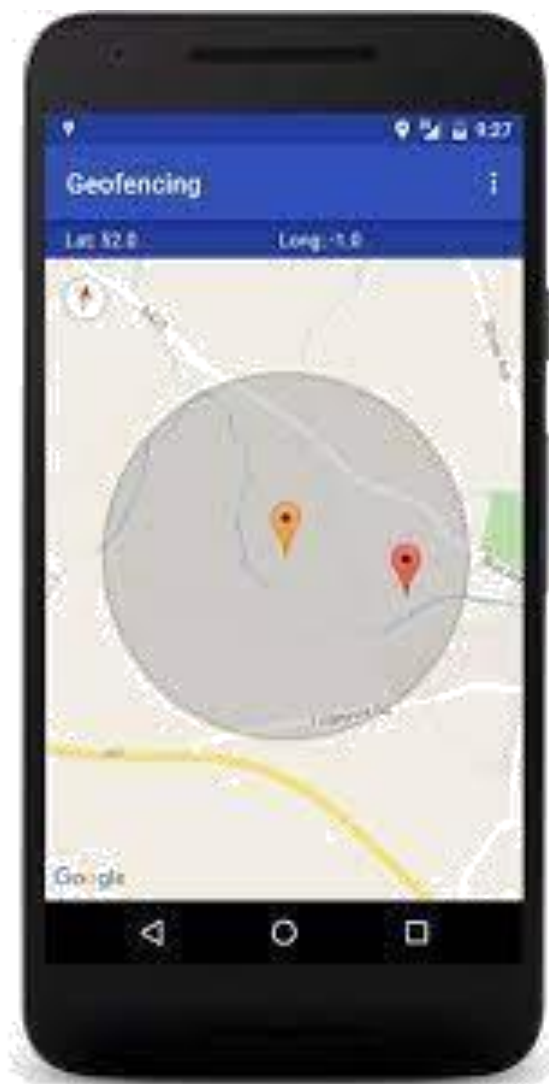


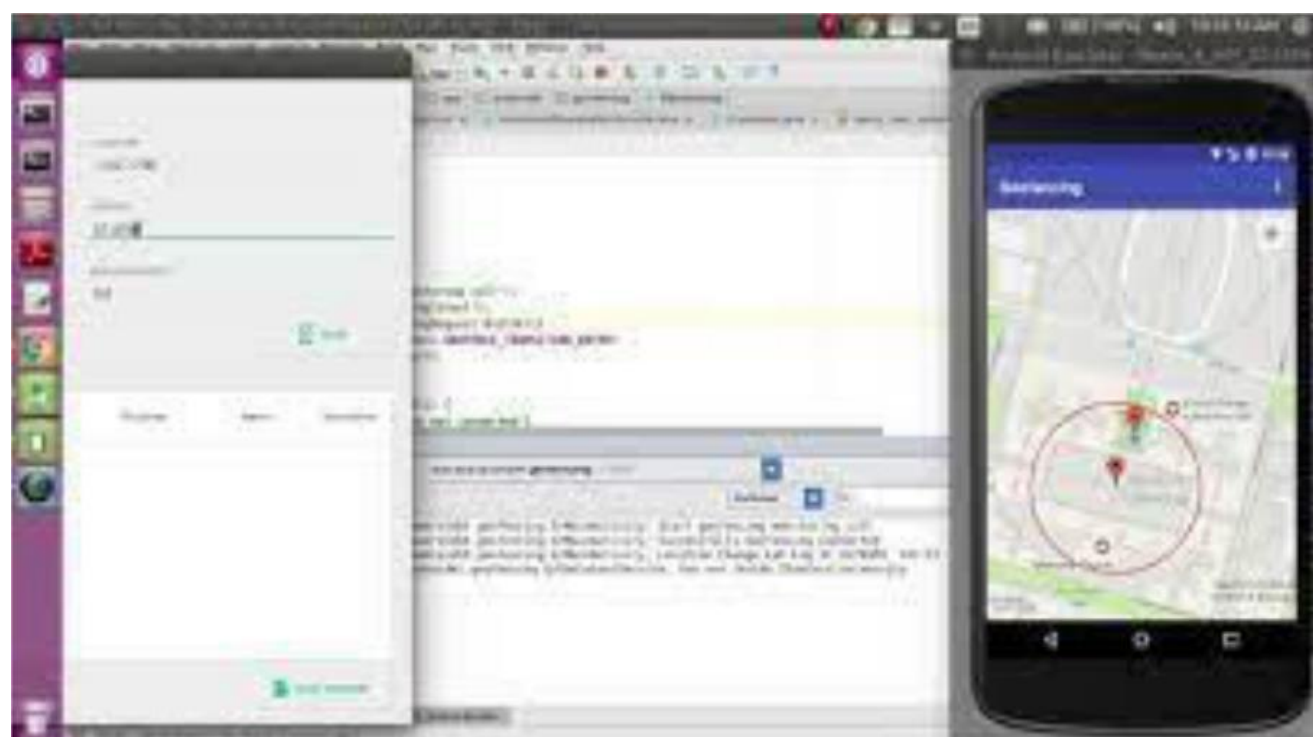
7. CODING & SOLUTIONING:





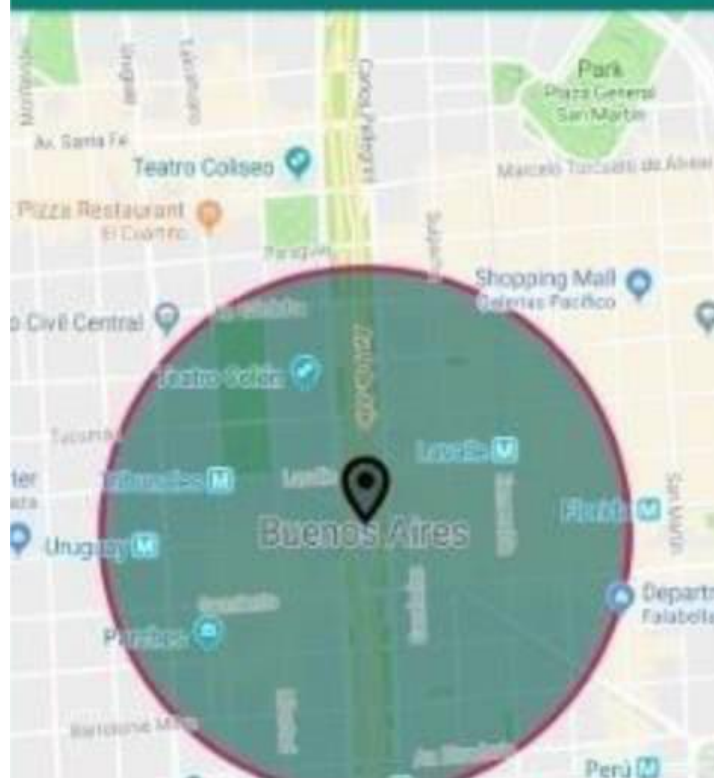
Geofence in Android Apps:







Remind me there



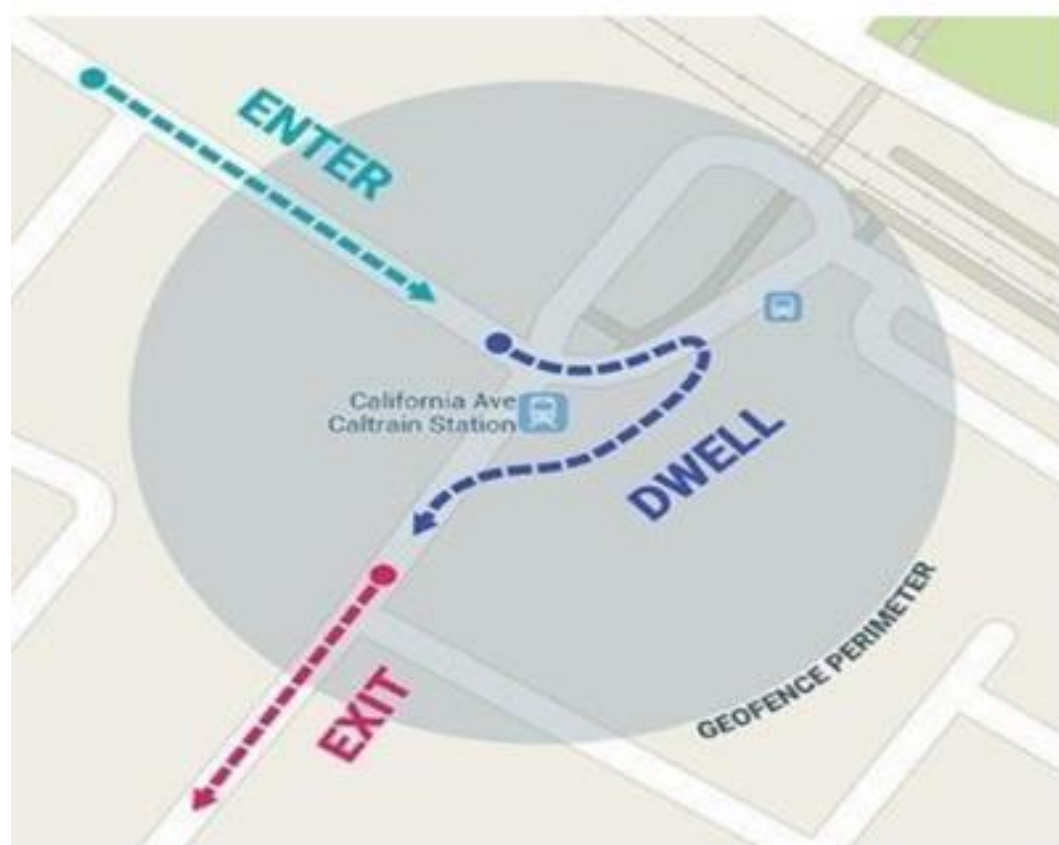
Choose the radius



600 meters

CONTINUE





8. TESTING:

8.1. TESTCASE:

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/Signup popup displayed or not	http://169.51.204.215:30106/	Login Signup popup should display	Working as expected	PASS	Successful			JENCY S KARTHIKA S AJAYKUMAR J
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login Signup popup	1.Enter URL and click go 2.Click on Signup button for User 3. Verify login/Signup popup with below UI elements: a.Id text box b.password text box c.Login button with orange colour 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			AARTHI A P JENCY S
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://shopnizer.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			AJAYKUMAR J

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			KARTHIKA S AJAYKUMAR J
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: Testing12367868 6786876876	Application should show 'Incorrect email or password' validation message.	Working as expected	PASS	Successful			JENCY S
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			KARTHIKA S AARTHI A P

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1 Enter URL:http://169.51.204.21:530106 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: 5434 password: Testing123	Application should show correct email or password validation message	Working as expected	PASS	Successful			AJAYKUMAR J AARTH A P
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1 Enter URL:http://169.51.204.21:530106 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			AJAYKUMAR J JENCY S
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1 Enter URL:http://169.51.204.21:530106 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	http://169.51.204.21:530106/	Customer database is visible	Working as expected	PASS	Successful			JENCY S AJAYKUMAR J
LoginPage_TC_O10	Functional	USER REGISTER	Verify Id sent to customer email address	1 Enter URL:http://169.51.204.21:530106 and click go 1 Register the account by giving credentials 2 Click on button Submit	http://169.51.204.21:530106/	Email sent successfully	Working as expected	PASS	Successful			KARTHIKA S JENCY S
LoginPage_TC_O11	Functional	AGENT REGISTER	Verify AGENT is able to log into application with Valid Credentials	1 Enter URL:http://169.51.204.21:530106 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	ID sent successfully Application should show correct email or password validation message.	PASS	Successful				JENCY S
LoginPage_TC_O12	Functional	Login page for ADMIN	Verify User is able to log into application with Invalid Credentials	1 Enter URL:http://169.51.204.21:530106 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: 1111 password: 5678	Application should show Incorrect ID or password validation message.	Working as expected	PASS	Successful			KARTHIKA S
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://169.51.204.21:530106 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			AJAYKUMAR J
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://169.51.204.21:530106 and click go 2 To the User Login page and submit Your Credentials	http://169.51.204.21:530106/	USER Home Page popup should display	Working as expected	PASS	Successful			JENCY S AARTH A P
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://169.51.204.21:530106 and click go 2 To the Admin Login page and submit Your Credentials	http://169.51.204.21:530106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			JENCY S
LoginPage_TC_O16	Functional	AGENT PAGE	On delete Button the user Credentials will be deleted	1 Enter URL:http://169.51.204.21:530106 and click go 2 To the Admin Page and select on User Credentials	http://169.51.204.21:530106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			JENCY S KARTHIKA S AJAYKUMAR J AARTH A P

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

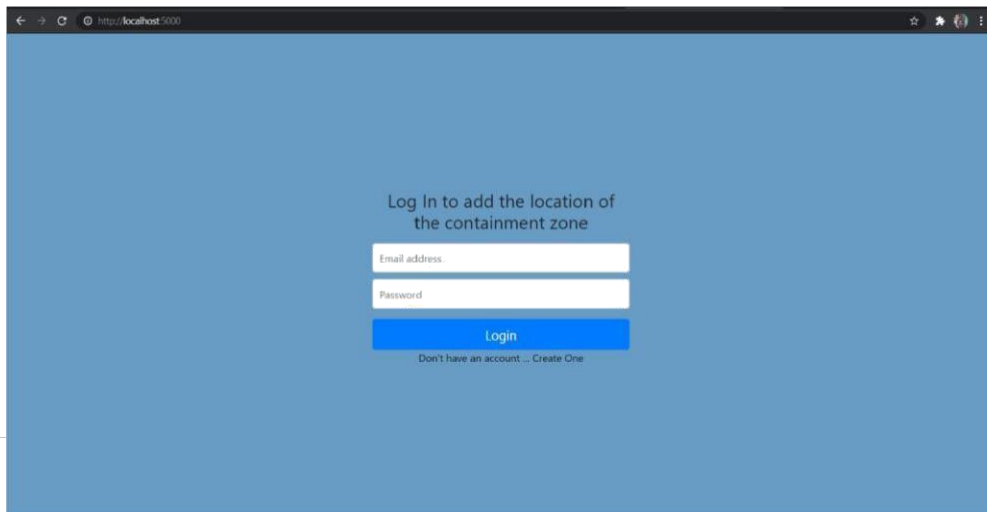
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS:

9.1. PERFORMANCE TESTING:

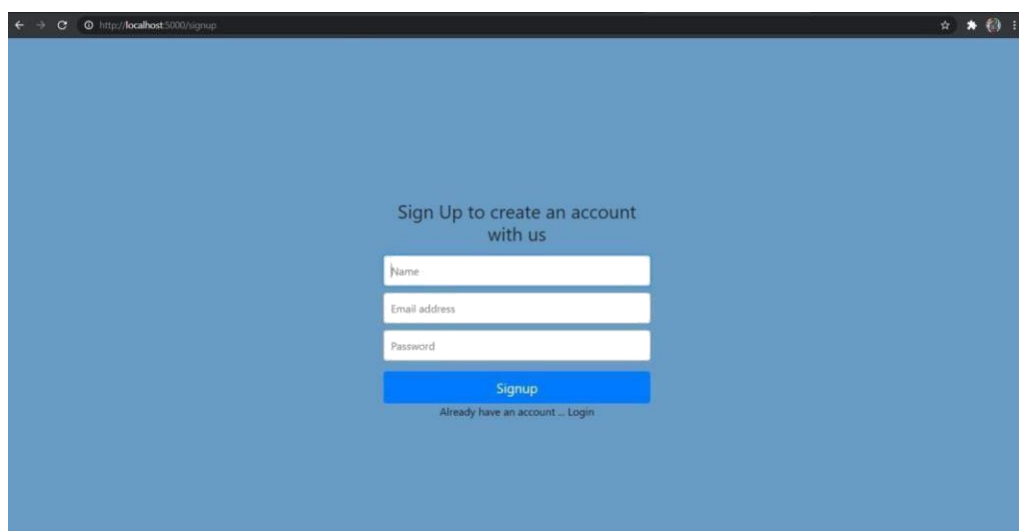
Admin App:

Login Page:



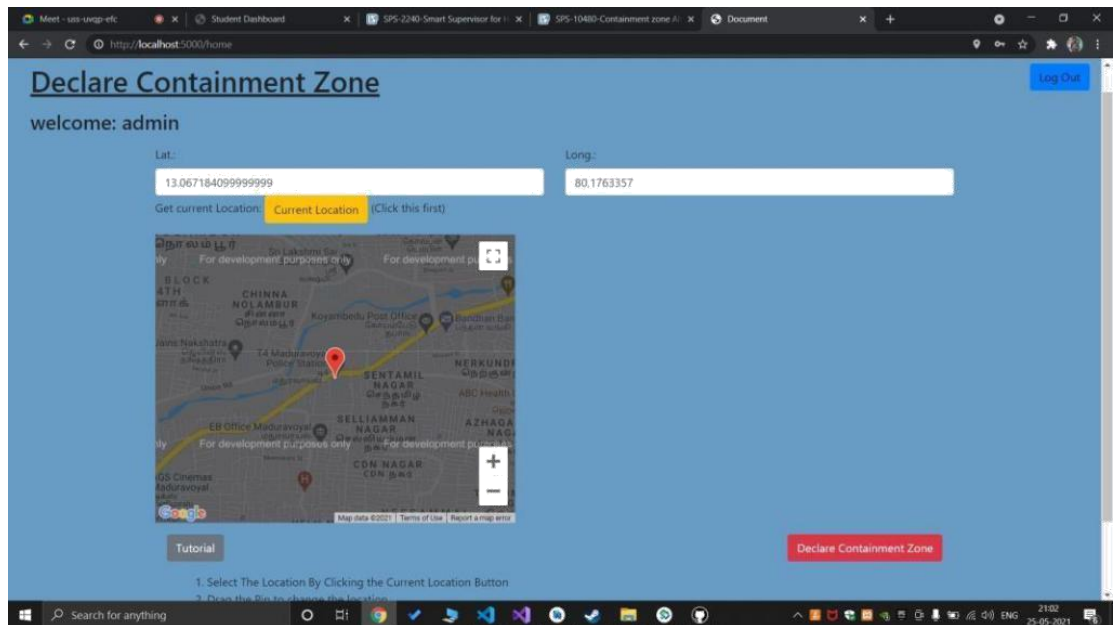
A screenshot of a web browser displaying the login page of an Admin App. The browser's address bar shows 'http://localhost:5000'. The page has a solid blue background. In the center, there is a white login form. The form contains the text 'Log In to add the location of the containment zone' above two input fields: 'Email address' and 'Password'. Below these fields is a blue 'Login' button. At the bottom of the form, there is a link that reads 'Don't have an account ... Create One'.

Register Page:



A screenshot of a web browser displaying the register page of an Admin App. The browser's address bar shows 'http://localhost:5000/signup'. The page has a solid blue background. In the center, there is a white registration form. The form contains the text 'Sign Up to create an account with us' above three input fields: 'Name', 'Email address', and 'Password'. Below these fields is a blue 'Signup' button. At the bottom of the form, there is a link that reads 'Already have an account ... Login'.

Home Page:



Location Data Page:

The screenshot shows a web browser window with the URL <http://localhost:5000/data>. The page has a blue header with the title "Location data and Visited People". Below the header is a table with 4 columns: "S.No", "Latitude", "Longitude", and "No_Visited". The table contains 7 rows of data. At the bottom right, there is a red button labeled "Go to location update Page".

S.No	Latitude	Longitude	No_Visited
1	13.069148883848849	80.17551259999999	0
2	13.068498821079215	80.1704513893798	0
3	12.979174795975714	77.59973092596437	0
4	14.469858338299407	75.91959519903565	0
5	13.062359612480321	77.5638966135254	0
6	15.840542738858232	76.64209647695924	0
7	15.3172775	75.7138884	0

Client Application:

Register screen:

9:06

Client_Containment

Sign Up

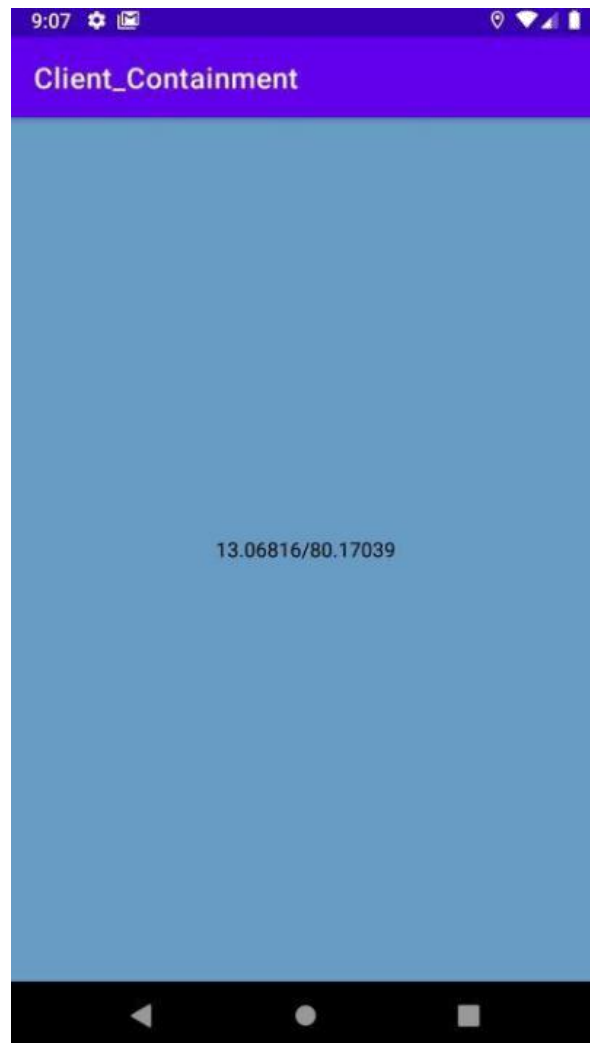
Name

Email

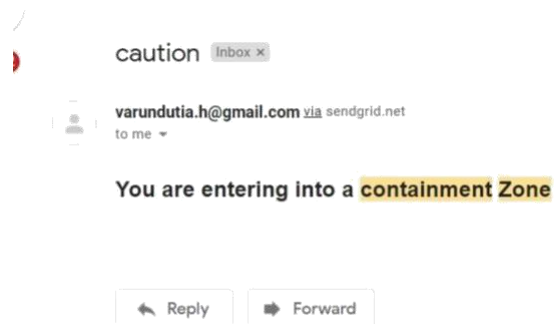
Password

SIGN UP

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 individual's location. Key benefits of the application are monitoring people's 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 quarter.

13.APPENDIX:

Source Code:

APP.PY

```
from logging import error from flask import *

from jinja2.utils import select_autoescape import bcrypt

from flask_mysqlldb 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.7BJDtQDIS8unH0r5_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]
```

```
return redirect(url_for("home"))
```

```
else:
```

```
session['email'] == return render_template('login.html', error=1)
```

```
else:
```

```
return render_template('login.html', render_template('login.html', error=3))
```

```
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

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"
    )
    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 = []
        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+Hhvx8T/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-
+YQ4JLhgyBLPDQt//l+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&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-38660-1660384152>

YOUTUBE VIDEO LINK:

<https://youtu.be/S0ZS55VNmPg>