PROJECT REPORT

IBM-Project--5385-1658761336

TITLE:

Containment Zone Alerting Application

TEAM LEADER:

RAKESH A

TEAM MEMBER:

- GHULAM DASTHAGEER G
- VISHAL RAJ R
- SATHAIAH BALAJI S

TEAM ID:

PNT2022TMID23042

1 INTRODUCTION

1.1 Project Overview

Our whole nation is fighting against COVID-19. Millions of people had lost their lives ; lost their loved ones; many dreams and goals has been given up. But on the other side people roaming in the midst of pandemic without any prior precautions, social distancing, masks….so and so.

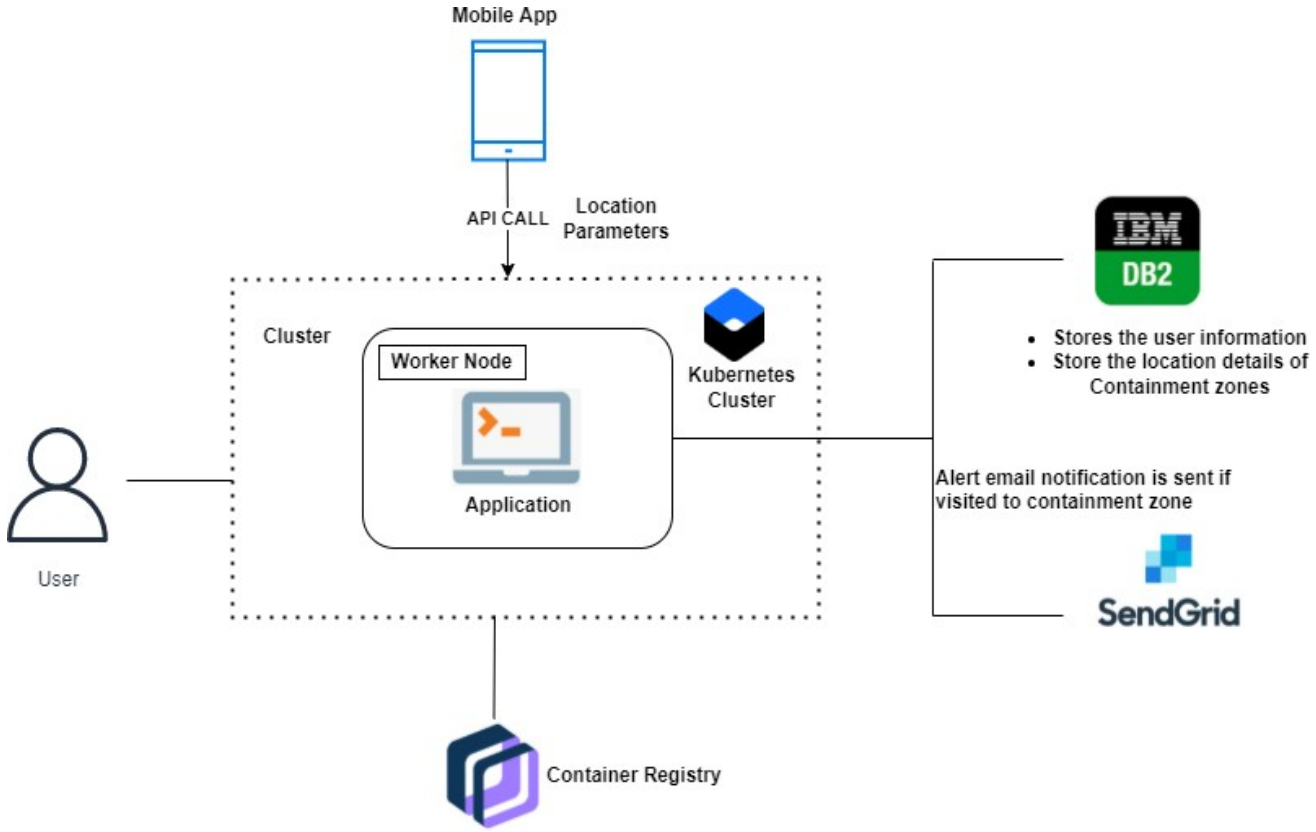
We must understand the struggle of doctors, nurses and other employees towards Covid free nation. Atleast we must have a common awareness towards this deadly virus. Start it from your area. Are you aware of containment zones? Atleast Do you have an idea whether your area is under containment zone or not?...

But how to keep yourself updated about the containment zones?

we have developed a Web app in which you can verify your area with containment zones. It also provides you with the our nation’s covid counts.

1.2 Purpose

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.



2 LITERATURE SURVEY

2.1 Existing problem

Title : Development of an Android application for viewing Covid-19 containment zones and monitoring violators.

Author Name : Amlan Protim Hazarika

Publication website : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7328652/

Published Date : May 11th, 2020

Objective :

This 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. All these functionalities are achieved by the help of Firebase and Geofencing tools from Google

Technology used:

Firebase Cloud Firestore - A real-time database is created in Google Cloud Firestore which contains all the data related to the containment zones like latitudes, longitude, radius and zone names

Geofencing – A feature in a software program that uses the Global Positioning

System(GPS) or radio frequency identification (RFID) to define geographical area

Title : aarogya setu

Developed by : National Informatics Centre under the Ministry of Electronics and Information Technology (MeitY)

Publication website : https://play.google.com/store/apps/details?id=nic.goi.aarogyasetu &hl=en

Published Date : 2020

Objective :

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

Functionality:

It will give the daily database of covid patients getting admitted,discharged ,updation of positive and negative cases and deaths occuring in a day and also it gives the total database of the covid cases.

It also provide information of each and every person getting vaccinated.

It will provide database in provitional manner.

Title :Corona watch

Developed by : Rhys Fenwick

Publication website :

https://play.google.com/store/apps/details?id=com.ksrsac.drawshapefile&hl=n

Published Date : February 19, 2020

Objective : This app is for showing the locations of Corona Affected Patients and their movement history of 14 days. General Public can use this to identify their movements in those areas. If found to be in such locations, they are requested to call help line numbers The app also facilitates citizens to identify the nearest hospitals which can treat for coronavirus including the sample collection centers and testing labs.

Functionalies:

It regularly sends two messages(12hrs each) for a day to the patients for checking about their status of their health condition.

Staffs are remotely monitored based on shift basis(12hrs each).

English and Spanish are the languages available in corona watch.

References

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7328652/

https://play.google.com/store/apps/details?id=nic.goi.aarogyasetu&hl=en

https://play.google.com/store/apps/details?id=com.ksrsac.drawshapefile&hl=n

2.2 Problem Statement Definition

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.

In this project you will be working on two modules :

1. Admin and
2. User

Admin:

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

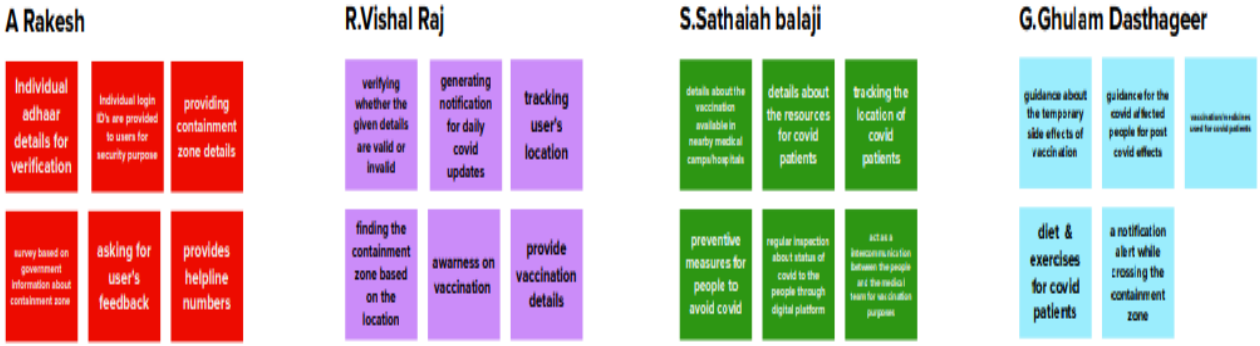
User :

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

3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming



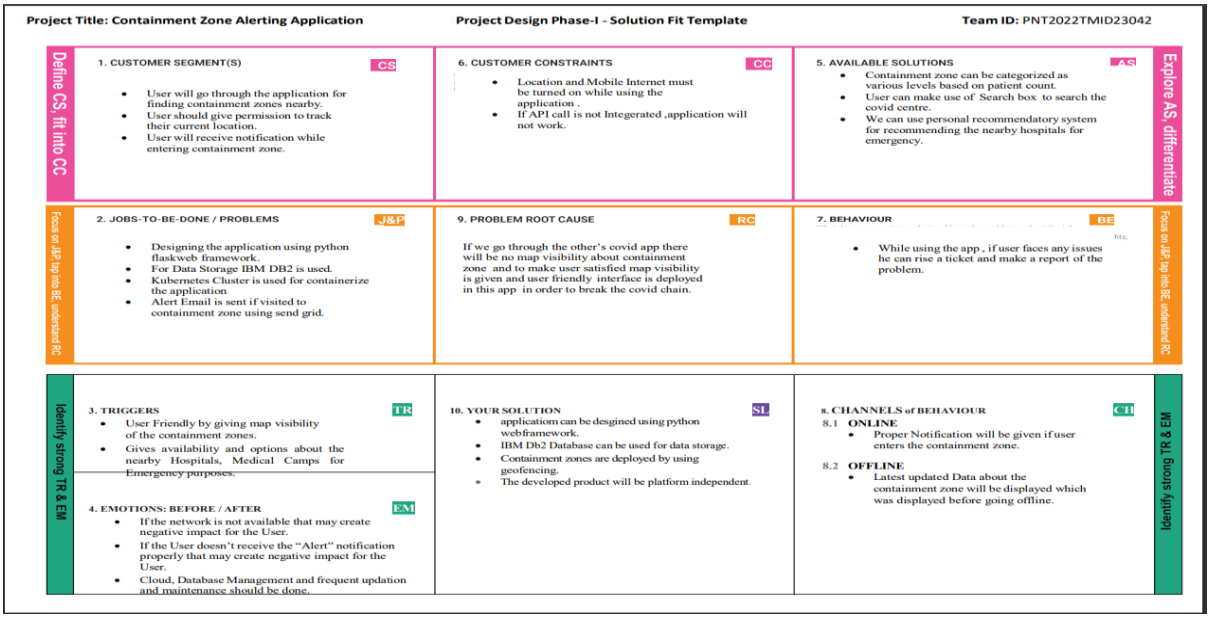
3.3 Proposed Solution

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 using cloud firebase database. Alerts are sent using the notification service. Features of the Application ***Admin App (portal):*** Admin can login to the app and update the containment zones locations in the portal. Based on the containment zones locations a Geofence will be created within a 100 meters radius. Admin should be able to see how many people are visiting that particular containment zone. ***User App (Mobile App):*** The app should have a user registration and login. After the user logged into the app it will track the user's location and update the database with the current location. If the user

		is visiting the containment zone he will get an alert notification.
3.	Novelty / Uniqueness	User will be able to see the containment zone in a map format and not in the table which only has the containment zone names. This would be helpful for the people who are new to the city.
4.	Social Impact / Customer Satisfaction	This application would make a big impact in reducing the covid cases as well as give people an idea about the disease and the cautionary measures against it.
5.	Business Model (Revenue Model)	There are many applications currently available in this regard. Our solution, once developed well, has enough possibility to become a good product to save people against the deadly diseases.
6.	Scalability of the Solution	Our proposed solution is very scalable i.e., in future, there are a lot of rooms for evolving our present model by adding new features to enhance our application in the future

3.4 Problem Solution fit



4 REQUIREMENT ANALYSIS

4.1 Functional requirement

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 and OTP.

FR-3	**Authentication**	It verifies the confirmation of the password.
FR-4	**Business rule**	For subscriber's we give first 3 day's free trail. For unsubscribes the user needs to watch some advertisement for knowing the zone alert for first 3 days.

4.2 Non-Functional requirements

NFR 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 users.
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.

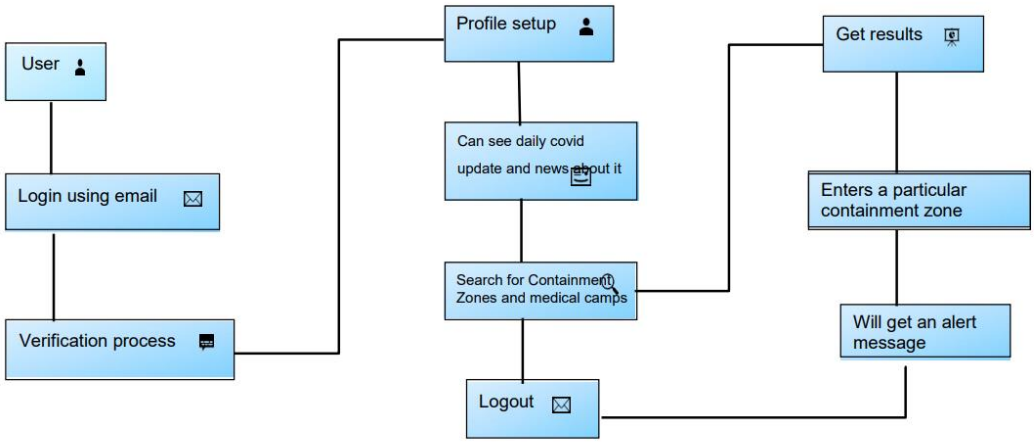
NFR-5	**Availability**	Stands for the system's reliability and accessibility to the user.
NFR-6	**Scalability**	The website is enough to support almost 1,00,000 users at a time.

5 PROJECT DESIGN

5.1 Data Flow Diagrams

Data Flow Diagrams:

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



5.2 Solution & Technical Architecture

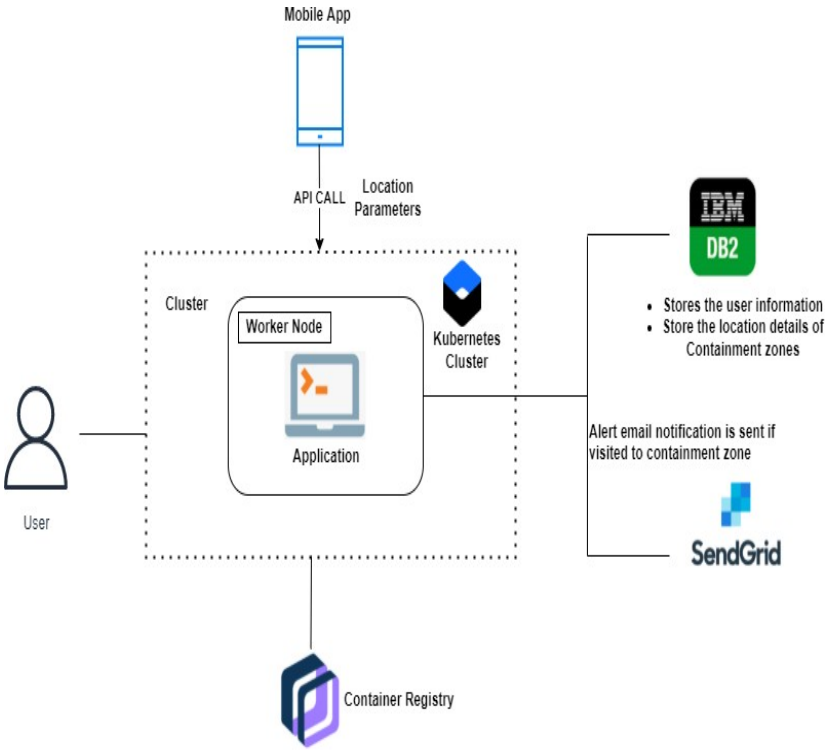
Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Developing Interface	Developing Application for the task	Java / Python
3.	Geofencing	Location-based service in which an app or other software program uses	Tile38,Geo(JavaScript) by Amazon Location Service (ALS)

		radio frequency identification	
4.	Alert Notifications	Broadcast of messages to one or many	sendgrid
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration : 	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application number of requests per sec, use of Cache, use of CDN's)	Technology used



5.3 User Stories

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
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 Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-5	As a user, I can access my dashboard after signing in.	I can access my account / dashboard	High	Sprint-1
Customer Care Executive	Geofencing	USN-6	As a user, I can see containment zones from the maps by tracking my current location.		High	Sprint-1
Administrator	DBMS	USN-7	As a administrator, I can keep the applications updated with containment zone details and regular covid related news.	I can perform various modifications in the applications according to user feedback.	High	Sprint-1

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration (web and android)	USN-1	USER: I can register for the application by entering my email and password	6	High	Rakesh Vishal Raj Sathaiah balaji Ghulam
		USN-2	USER: I will receive a confirmation email once I have registered for the application	6	High	Rakesh Vishal Raj Sathaiah balaji Ghulam
	Login (web and android)	USN-3	USER: I can log into the application by entering my	8	High	Rakesh Vishal Raj Sathaiah balaji Ghulam

4

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Dashboard	USN-4	USER: need to give permission to access my location	10	High	Rakesh Vishal Raj Sathaiah balaji Ghulam
		USN-5	As a user, I can log into the application by entering email & password	10	High	Rakesh Vishal Raj Sathaiah balaji Ghulam

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 3	Service	USN 6	ADMIN: I need to update the containment zones.	10	High	Rakesh Vishal Raj Sathaiah balaji Ghulam
		USN 7	ADMIN: I need to differentiate the containment zones based on the intensity of infection.	10	Medium	Rakesh Vishal Raj Sathaiah balaji Ghulam

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 4	Service	USN 8	ADMIN: I need to alert the user when they enter the containment zone through the notification	6	Medium	Rakesh Vishal Raj Sathaiah balaji Ghulam
	Data collection	USN 9	ADMIN: I need to store user details on the cloud	6	Medium	Rakesh Vishal Raj Sathaiah balaji Ghulam

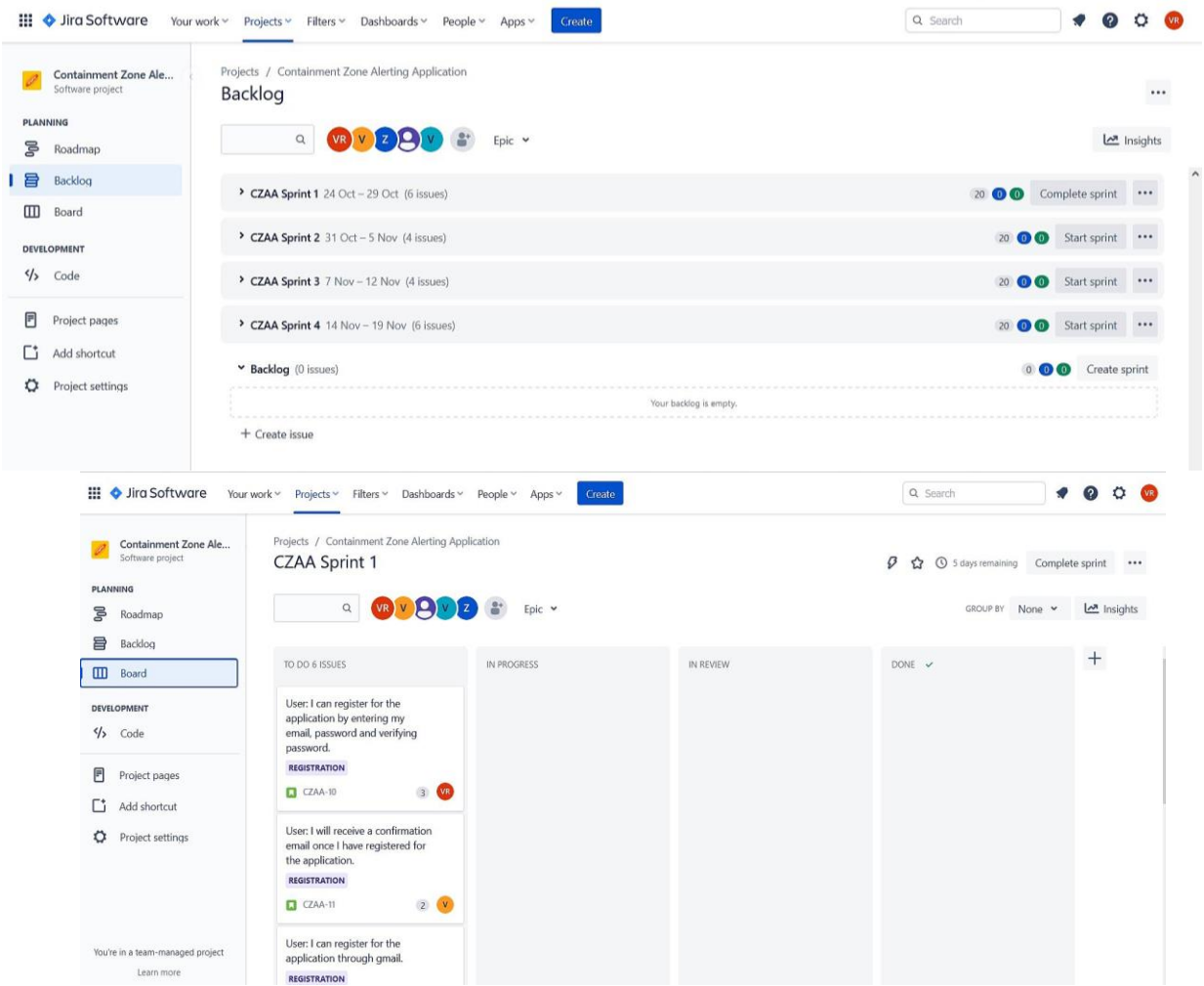
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
		USN 10	ADMIN: I need to collect details about covid -19 cases from verified sources	8	Medium	Rakesh Vishal Raj Sathaiah balaji Ghulam

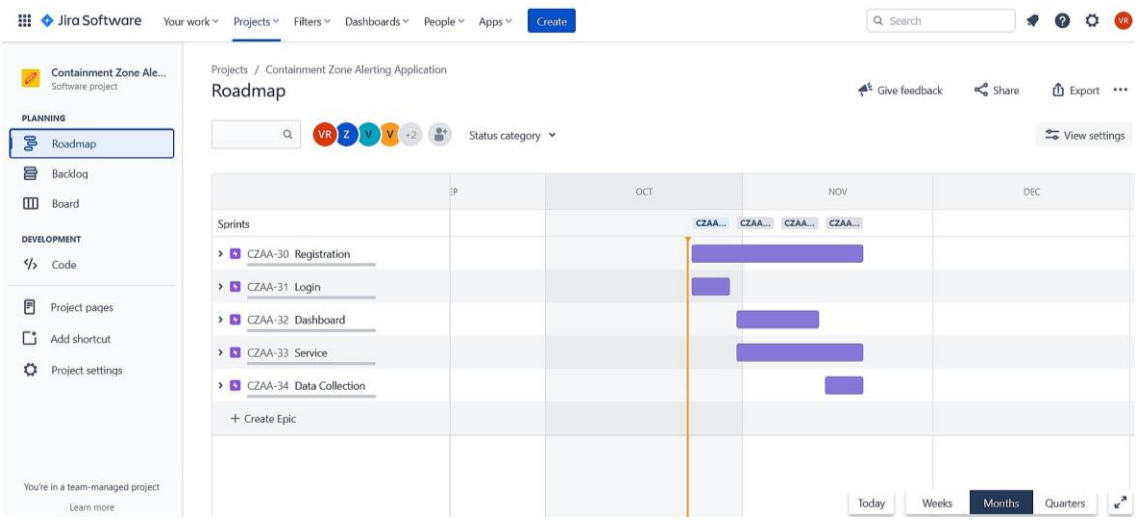
6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	7 Days	01 nov 2022	04 nov 2022	20	8 Nov 2022
Sprint-2	20	6 Days	04Nov 2022	08 Nov 2022	20	
Sprint-3	20	5 Days	09 Nov 2022	12 Nov 2022	20	
Sprint-4	20	6 Days	12 Nov 2022	17 Nov 2022	20	

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

6.3 Reports from JIRA





7 CODING & SOLUTIONING

7.1 Feature 1`

Admin App (portal):

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

7.2 Feature 2

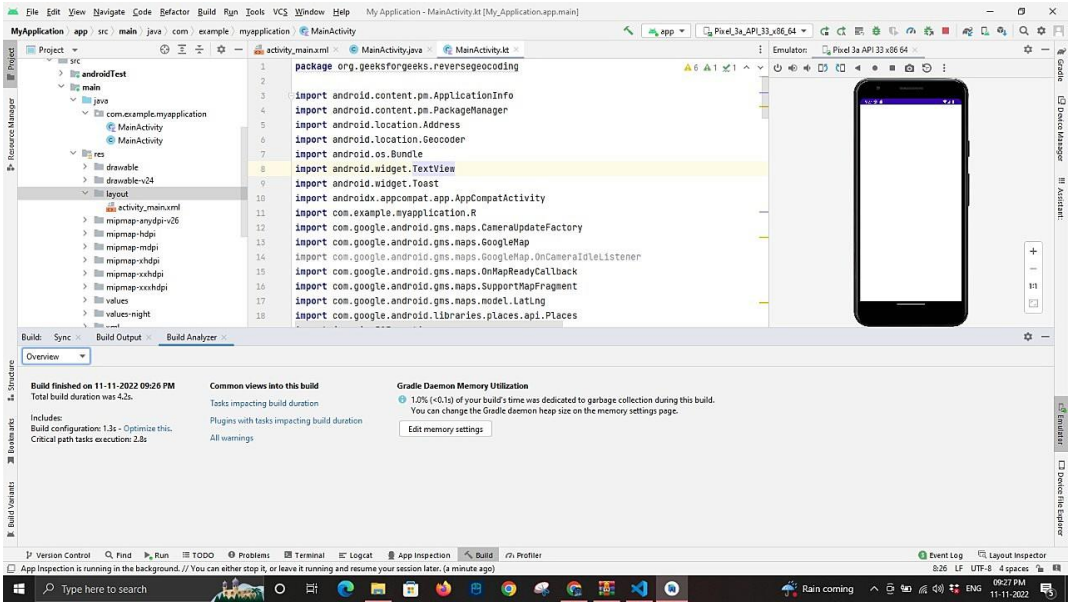
User App (Mobile App):

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

8 TESTING

8.1 Test Cases

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.



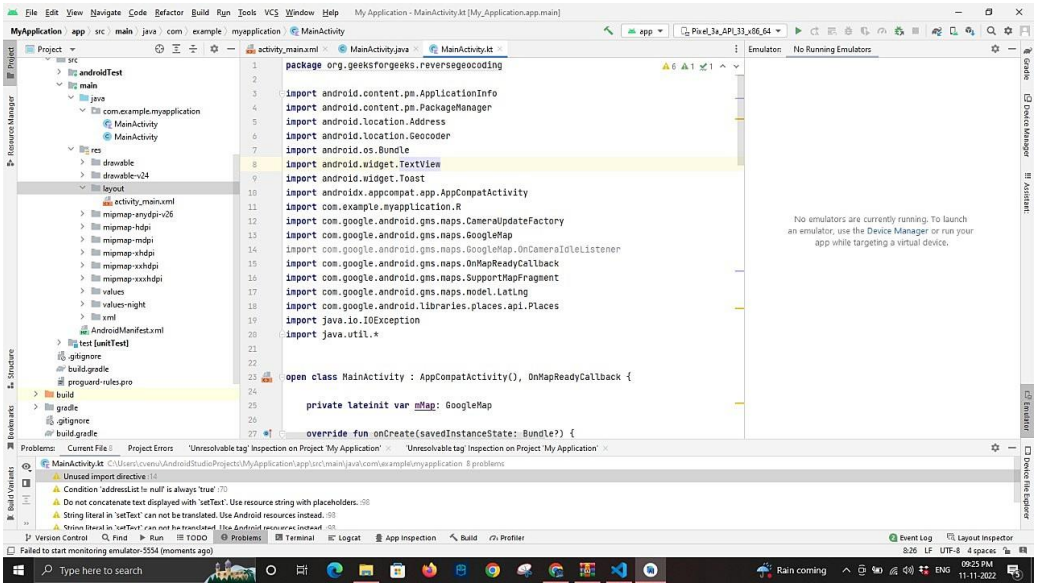
This Software is tested and evaluated successfully.

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Inventory Management System project at the time of the release to User Acceptance Testing (UAT)

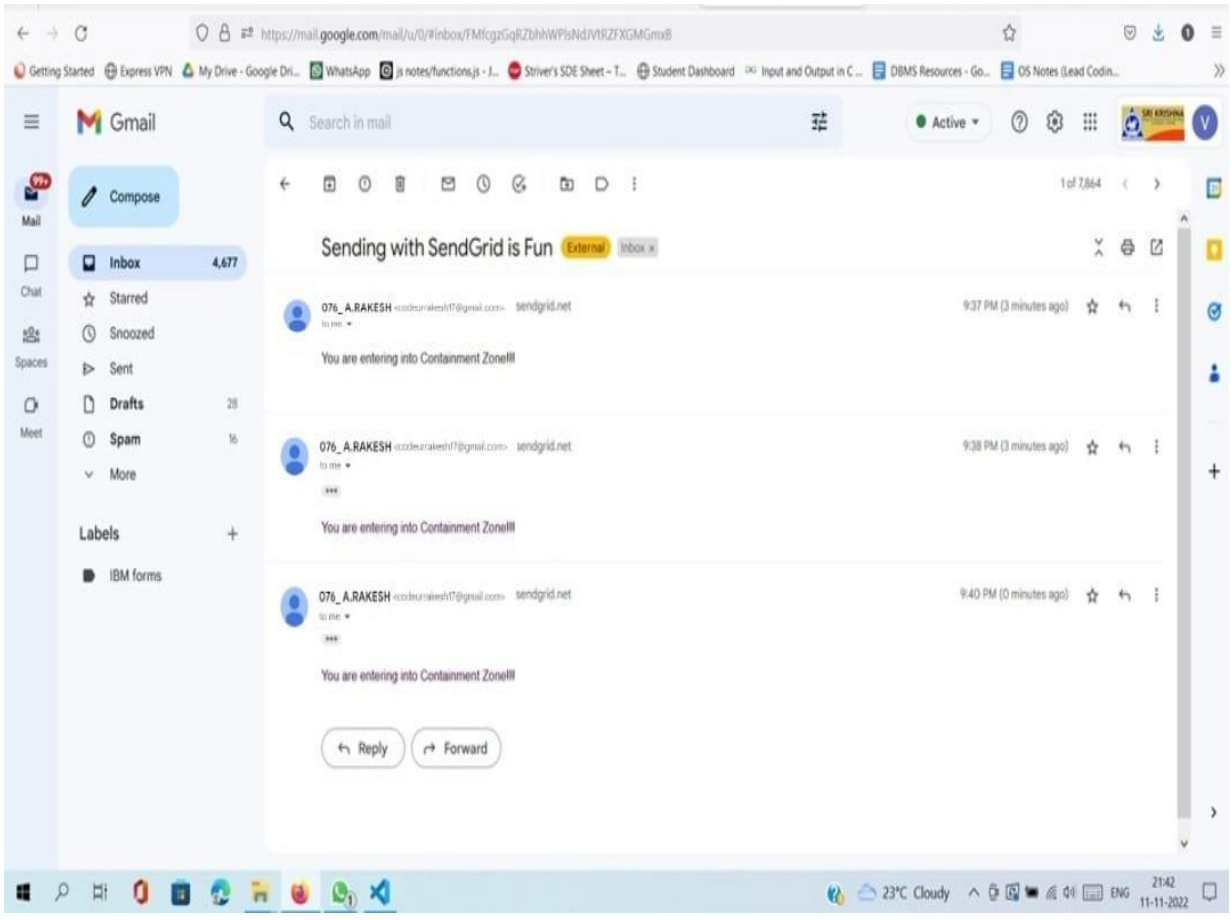
User Acceptance Testing is carried out in a separate testing environment. A change, an update, or a new feature is requested and developed. Unit and integration tests are run. All seems to be in order. But then, after it is released to the public, serious problems appear. Rework and retesting are not the most expensive consequences when that happens. Loss of reputation is.



9 RESULTS

Tests have been carried out in various containment zones across Tamil Nadu for the validation of the Android application. The identified containment zones chosen for the testing of the application were visited one by one. It shows various containment zones identified for conducting the test, the date, time of entry, time of receiving the notification alerts upon entering. It is highlighted that the application sends notification alerts within 5–8 seconds on entering.

9.1 Performance Metrics



10 ADVANTAGES & DISADVANTAGES

User Privacy Protection

Location tracking is enabled by the user and is informed to the user via a fixed notification. Before user tests positive for COVID-19 and uploads all his/her locations, the locations are stored in the device's local storage, none but the user has access to it. Once user tests positive for COVID-19 and uploads his/her locations, the identity of the user is preserved and not accessible to any other user. However, administrative access is enabled for tracking down false claims (not implemented yet) for taking legal actions

Efficient Access to potential Huge Server Data Storage

Tracked location data of COVID-19 positive patients will evidently get very large, as the number of affected people is rising each day. Moreover, in many areas people are still reluctant or don't have the luxury to maintain social distancing. To somewhat make the query process of a possible huge data storage a hashing algorithm is implemented. A particular tracked location is converted into its

corresponding square block/s of area 20 meters x 20 meters along with and hourly time frame.

The block generation is similar to hashing function by providing a key that is the particular index for a query, with the additional benefit that the block also defines a radius of presence for any particular location. A block is defined by its bottom left and top right diagonal coordinates.

Anonymous Relief Posts:

Through the app's global news feed, relief requests can be posted without directly sharing personal or family information of a user. A contact button is attached to relief posts through which any other user can call and contact the relief request post's author and reach out for help. This feature especially targets the middle-class families that are suffering greatly in silence and cannot seek help publicly. A user is allowed to make only one relief post every seven days, this is a measure taken to stop misuse of the feature.

11 CONCLUSION

The application provides an efficient way of showing the identified Covid-19 containment zones to the users in a Google map. With the alarming increase of Covid-19 affected cases throughout the world, this developed application can be employed as a tool for creating further social awareness among the people. This application further tracks the user's location and checks whether it is present in the list of identified containment zones. It sends separate notification alerts to the user on entering. The developed android application further extracts the IMEI Number of the trespasser in the containment zones which can be useful to the local police to track and identify people who are frequently trespassing the containment zones. Thereby this application identifies the containment zones and highlights the need for taking further precautionary measures for fighting Covid-19. The application has been tested in various locations and has been found to yield accurate results

.

12 FUTURE SCOPE

The application can be further used for many purposes like maritime and forest safety to prevent users from entering restricted areas.

13 APPENDIX

Source Code

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>
</head>
<body>
    <h1>HOME</h1>
</body>
</html>
```

Login.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">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js">
```

```
    <link rel="stylesheet"
     href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://contzone-bucket.s3.jp-tok.cloud-object-
     storage.appdomain.cloud/AuthenticateStyle.css">
    <title>Login</title>

</head>
<body >
    <div class="container-fluid px-1 px-md-5 px-lg-1 px-xl-5 py-5 mx-auto">
      <div class="card card0 border-0">
        <div class="row d-flex">
          <div class="col-lg-6">
            <div class="card1 pb-5">
              <div class="row px-3 justify-content-center mt-4 mb-5 border-line">
                <img src="https://contzone-bucket.s3.jp-tok.cloud-object-
    storage.appdomain.cloud/login.png" class="image">
              </div>
            </div>
          </div>
          <div class="col-lg-6">
            <form class="card2 card border-0 px-4 py-5" method="post" action="/login">
              <div class="row mb-4 px-3">
                <h6 class="mb-0 mr-4 mt-2">Sign in with</h6>
                <div class="facebook text-center mr-3"><div class="fa fa-
    facebook"></div></div>
                <div class="twitter text-center mr-3"><div class="fa fa-
    twitter"></div></div>
                <div class="linkedin text-center mr-3"><div class="fa fa-
    linkedin"></div></div>
              </div>
              <div class="row px-3 mb-4">
                <div class="line"></div>
                <small class="or text-center">Or</small>
                <div class="line"></div>
```

```
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Email Address</h6></label>
              <input class="mb-4" type="text" name="email" placeholder="Enter a valid email address">
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Password</h6></label>
              <input type="password" name="password" placeholder="Enter password">
            </div>
            <div class="row px-3 mb-4">
              <div class="custom-control custom-checkbox custom-control-inline">
                <input id="chk1" type="checkbox" name="chk" class="custom-control-input">
                <label for="chk1" class="custom-control-label text-sm">Remember me</label>
              </div>
              <a href="#" class="ml-auto mb-0 text-sm">Forgot Password?</a>
            </div>
            <div class="row mb-3 px-3">
              <button type="submit" class="btn btn-blue text-center">Login</button>
            </div>
            <div class="row mb-4 px-3">
              <small class="font-weight-bold">Don't have an account? <a class="text-danger " href="register">Register</a></small>
            </div>
          </form>
        </div>
      </div>
      <div class="bg-blue py-4">
        <div class="row px-3">
```

```
                <small class="ml-4 ml-sm-5 mb-2">Containment Zone Detection</small>
                <div class="err"> {{error}}</div>
                <div class="success"> {{success}}</div>
                <div class="social-contact ml-4 ml-sm-auto">
                    <span class="fa fa-facebook mr-4 text-sm"></span>
                    <span class="fa fa-google-plus mr-4 text-sm"></span>
                    <span class="fa fa-linkedin mr-4 text-sm"></span>
                    <span class="fa fa-twitter mr-4 mr-sm-5 text-sm"></span>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Register.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">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://contzone-bucket.s3.jp-tok.cloud-object-storage.appdomain.cloud/AuthenticateStyle.css">
    <title>Register</title>
```

```
</head>
<body>
  <div class="container-fluid px-1 px-md-5 px-lg-1 px-xl-5 py-5 mx-auto">
    <div class="card card0 border-0">
      <div class="row d-flex">
        <div class="col-lg-6">
          <div class="card1 pb-5">
            <div class="row px-3 justify-content-center mt-4 mb-5 border-line">
              <img src="https://contzone-bucket.s3.jp-tok.cloud-object-storage.appdomain.cloud/reg.png" class="regimage">
            </div>
          </div>
        </div>
        <div class="col-lg-6">
          <form class="card2 card border-0 px-4 py-5" method="post" action="/register">
            <div class="regist">
              <h1>Register Here</h1>
            </div>
            <div class="row px-3 mb-4">
              <div class="line"></div>
              <div class="line"></div>
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Username</h6></label>
              <input class="mb-4" type="text" name="name" placeholder="Enter a username">
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Email Address</h6></label>
              <input class="mb-4" type="text" name="email" placeholder="Enter a valid email address">
```

```
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Password</h6></label>
              <input type="password" name="password" placeholder="Enter password">
            </div>
            <div class="row px-3">
              <label class="mb-1"><h6 class="mb-0 text-sm">Confirm Password</h6></label>
              <input type="password" name="cpassword" placeholder="Enter password">
            </div>
            <div class="row px-3 mb-4">
              <div class="custom-control custom-checkbox custom-control-inline">
                <input id="chk1" type="checkbox" name="chk" class="custom-control-input">
                <label for="chk1" class="custom-control-label text-sm">Remember me</label>
              </div>
              <a href="#" class="ml-auto mb-0 text-sm">Forgot Password?</a>
            </div>
            <div class="row mb-3 px-3">
              <button type="submit" class="btn btn-blue text-center">Register</button>
            </div>
            <div class="row mb-4 px-3">
              <small class="font-weight-bold">I already have an account? <a class="text-danger " href="login">Login</a></small>
            </div>
          </form>
        </div>
      </div>
      <div class="bg-blue py-4">
```

```
            <div class="row px-3">
               <small class="ml-4 ml-sm-5 mb-2">Containment Zone Detection</small>
               <div class="err"> {{error}}</div>
               <div class="success"> {{success}}</div>
               <div class="social-contact ml-4 ml-sm-auto">
                  <span class="fa fa-facebook mr-4 text-sm"></span>
                  <span class="fa fa-google-plus mr-4 text-sm"></span>
                  <span class="fa fa-linkedin mr-4 text-sm"></span>
                  <span class="fa fa-twitter mr-4 mr-sm-5 text-sm"></span>
               </div>
            </div>
         </div>

   </div>
</body>
</html>
```

App.py

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from flask import Flask, render_template, request, redirect, url_for,session
...
... import ibm_db
... import bcrypt
... conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-
    ba32-
    21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SEC
    URITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP
    ;UID=nkq41110;PWD=2jpo8fPJraZ3KYPc","")
...
... app = Flask(__name__)
... app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
...
...
```

```
... @app.route("/",methods=['GET'])
... def home():
...     if 'email' not in session:
...       return redirect(url_for('login'))
...     return render_template('home.html',name='Home')
...
... @app.route("/register",methods=['GET','POST'])
... def register():
...   if request.method == 'POST':
...     name = request.form['name']
...     email = request.form['email']
...     password = request.form['password']
...     cpassword = request.form['cpassword']
...
...     if not email or not name or not password or not cpassword:
...       return render_template('register.html',error='Please fill all fields')
...     if password != cpassword:
...         return render_template('register.html',error='The password is not same')
...     else:
...         hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
...
...     query = "SELECT * FROM LOIGNAUTHENTICATION WHERE useremail=?"
...     stmt = ibm_db.prepare(conn, query)
...     ibm_db.bind_param(stmt,1,email)
...     ibm_db.execute(stmt)
    isUser = ibm_db.fetch_assoc(stmt)

    if not isUser:
      insert_sql = "INSERT INTO LOIGNAUTHENTICATION(USERNAME,
      USEREMAIL, PASSWORD) VALUES (?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, email)
      ibm_db.bind_param(prep_stmt, 3, hash)
```

```
      ibm_db.execute(prep_stmt)
      return render_template('register.html',success="You can login")
    else:
      return render_template('register.html',error='Invalid Credentials')

  return render_template('register.html')

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

      if not email or not password:
        return render_template('login.html',error='Please fill all fields')
      query = "SELECT * FROM LOIGNAUTHENTICATION WHERE useremail=?"
      stmt = ibm_db.prepare(conn, query)
      ibm_db.bind_param(stmt,1,email)
      ibm_db.execute(stmt)
      isUser = ibm_db.fetch_assoc(stmt)
      print(isUser,password)

      if not isUser:
        return render_template('login.html',error='Invalid Credentials')
      #return render_template('login.html',error=isUser['PASSWORD'])
      isPasswordMatch = bcrypt.checkpw(password.encode('utf-
      8'),isUser['PASSWORD'].encode('utf-8'))

      if not isPasswordMatch:
        return render_template('login.html',error='Invalid Credentials')

      session['email'] = isUser['USEREMAIL']
      return redirect(url_for('home'))
```

```
    return render_template('login.html',name='Home')


@app.route('/logout')
def logout():
    session.pop('email', None)
    return redirect(url_for('login'))
if __name__ == "__main__":
```

sendgrid.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 = x19Tsxbexe7x8c_rx12Qx14x13qxb7'WTH0x9fxe4xecxb1
app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'F5shCxBMxe'
app.config['MYSQL_PASSWORD'] = 'g1rMHVIhIq'
app.config['MYSQL_DB'] = 'F5shCxBMxe'

mysql = MySQL(app)

# functions
```

```
def send_mail(email)
    print(email)
    message = Mail(from_email='vishalraj@gmail.com',
                   to_emails=email,
                   subject='caution',
                   plain_text_content='Please Stay Safe',
                   html_content='h2You are entering into a containment Zoneh2')

    try
        sg = SendGridAPIClient(

     'SG.7BJDtQDlS8unH0r5_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG117YURq
     OTa2HL')
        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 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'))

    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)
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-5385-1658761336

Project Demo Link:

https://www.youtube.com/watch?v=VSutS_K45I4&t=57s