CONTAINMNET ZONE ALTERING APPLICATION

Team id: PNT2022MID00605

TEAM MEMBERS

Dharshini.B	211419104063
Dharshini.P.K	211419104064
Karthika.M	211419104124
Nithiyashree.S	211419104184

INTRODUCTION

1.1 PROJECT OVERVIEW

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

1.2 PURPOSE

The containment zone tracker is a quick and easy way to find out if a particular locality, home or office, lies in a containment zone. The tracker constitutes a dashboard which shows areas on the map that are currently containment zones

2. LITERATURE SURVEY

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
1	S-Nav: Safety-Aware IoT Navigation Tool for Avoiding COVID19 Hotspots (2021)	In this project they make a navigation system that navigates people from not entering into COVID19 hot spot region and shows the safe path to make their journey safe	Q-learning model, reinforcement learning (RL), shortest path.	Internet of Things (IoT)	It is a navigation based projects that's works on showing the safe route to travel and it for only the COVID virus analysis
2	Development of An Android Application for Viewing Covid-19 Containment Zones and Monitoring Violators Who are Trespassing into It Using Firebase and Geo-fencing	It is an application that developed to show alert message when people enter into the containment zone	Geo-fencing Algorithm	Cloud & Flask	In this app people only knows the containment zone but we does not know how fast it spread
3	A Compact Wearable-IoT (W-IoT) System for Health Safety and Protection of Out-goers in the PostLockdown World (2019)	In this project they create an prototype that make an analysis on our health	electronic face mask, automatic sanitizer dispenser	Internet of Things (IoT)	It is a prototype but we cant wear it anywhere we need to go and it wont be use full after the

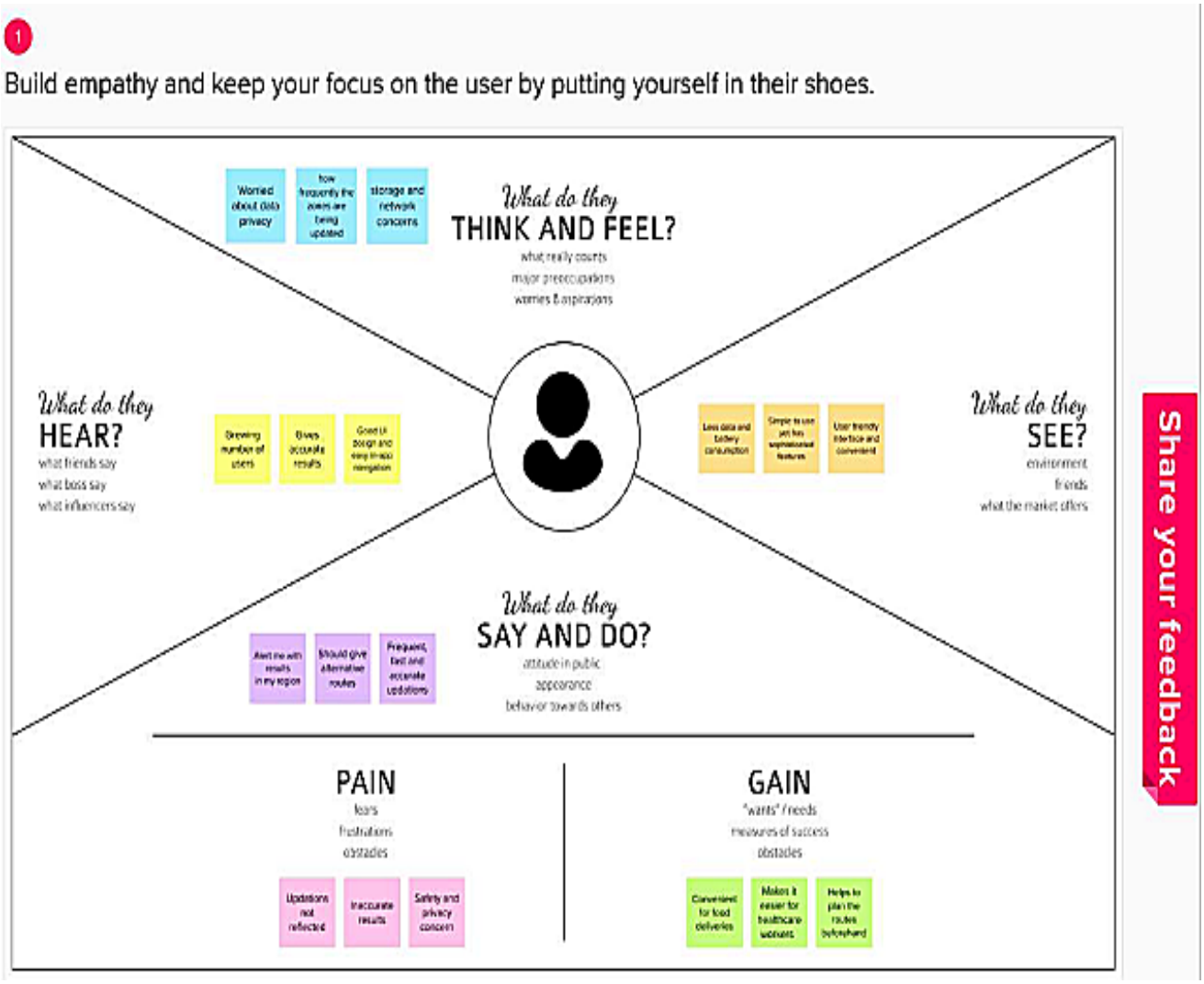
		conditions like monitoring or health conditions and giving analysis			pandemic pass out
4	Tracking the Covid zones through geo-fencing technique (2020)	In this paper they developed an algorithm to track people from entering into containment zone and alerting before they getting into risk	RSA algorithm	Data science	It this paper it if typical to handle the data of covid in and out cases
5	Application for Covid-19 Real Time Counter (2022)	In this project they made an app to show the real time count of the covid 19 cases and spread awareness to the people about the out world pandemic	Java	Android studio	Even though it gives us more awareness about the pandemic outside it makes people more panic on deaths they listed
6	Privacy-aware energy-efficient framework Using the internet of medical things for covid-19 (2020)	developed that can analyze BP, , oxygen level, temperature	CR-IoMT , CR-IoT	Cloud computing, Big data & IoT	Not all peoples can afford to buy it

2.3 PROBLEM STATEMENT DEFENITION

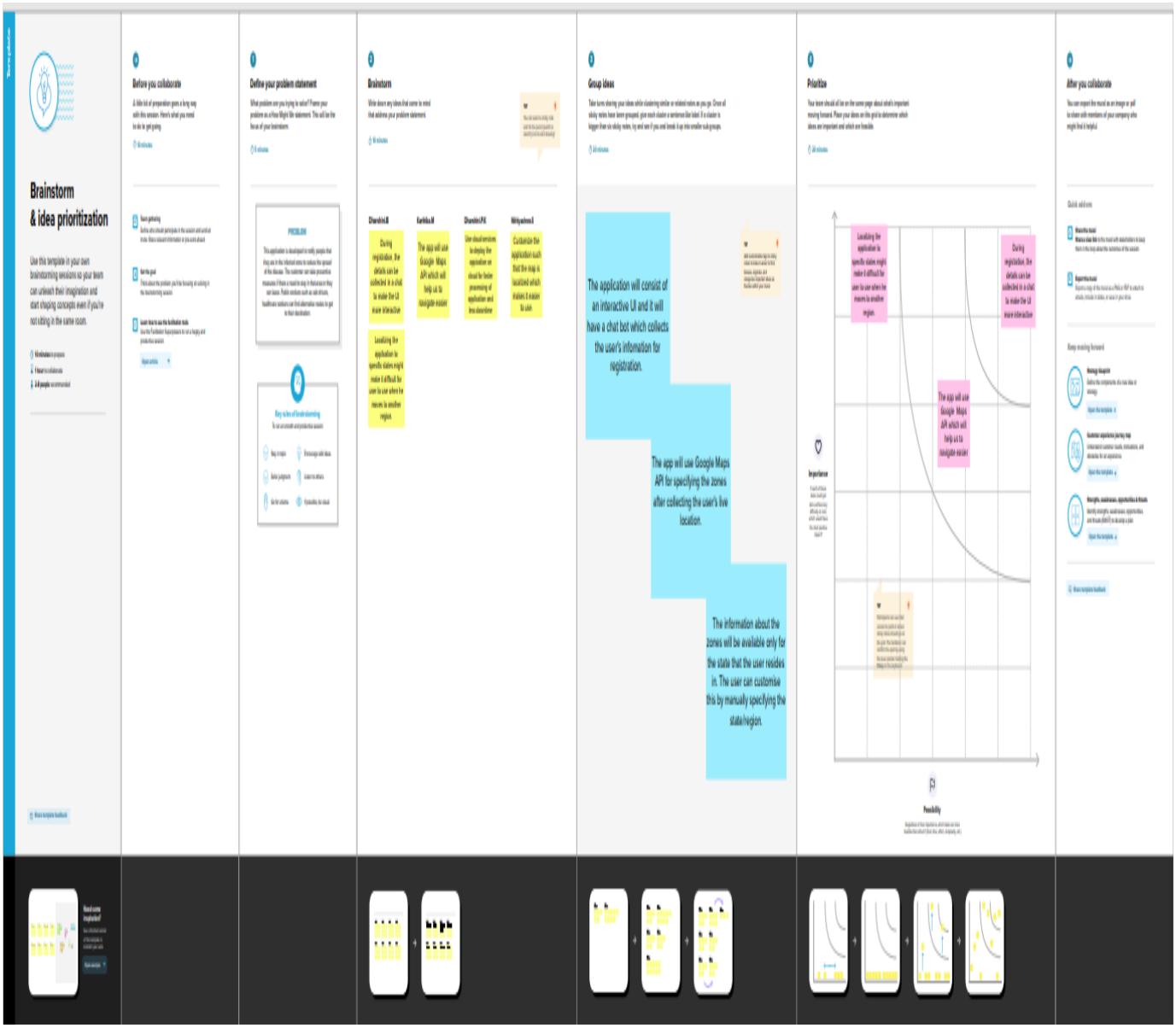
This application is intended to provide information about containment zones in a particular region by alerting people, through continuous monitoring of an individual's location. Key benefits of the application are monitoring people's activity and alerting them of their safety movements.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



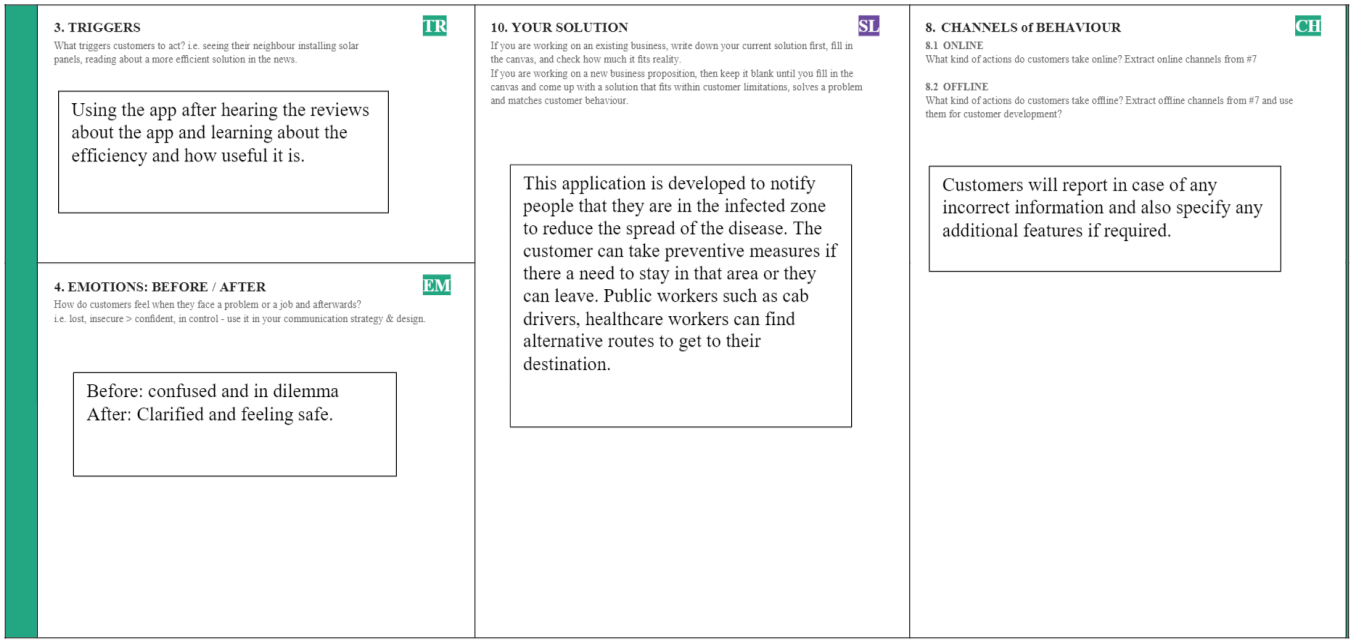
3.2 Ideation & Brainstorming



3.3 Proposed Solution

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

3.4 Problem Solution fit



3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Using the app after hearing the reviews about the app and learning about the efficiency and how useful it is.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before: confused and in dilemma
After: Clarified and feeling safe.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

This application is developed to notify people that they are in the infected zone to reduce the spread of the disease. The customer can take preventive measures if there a need to stay in that area or they can leave. Public workers such as cab drivers, healthcare workers can find alternative routes to get to their destination.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development?

Customers will report in case of any incorrect information and also specify any additional features if required.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

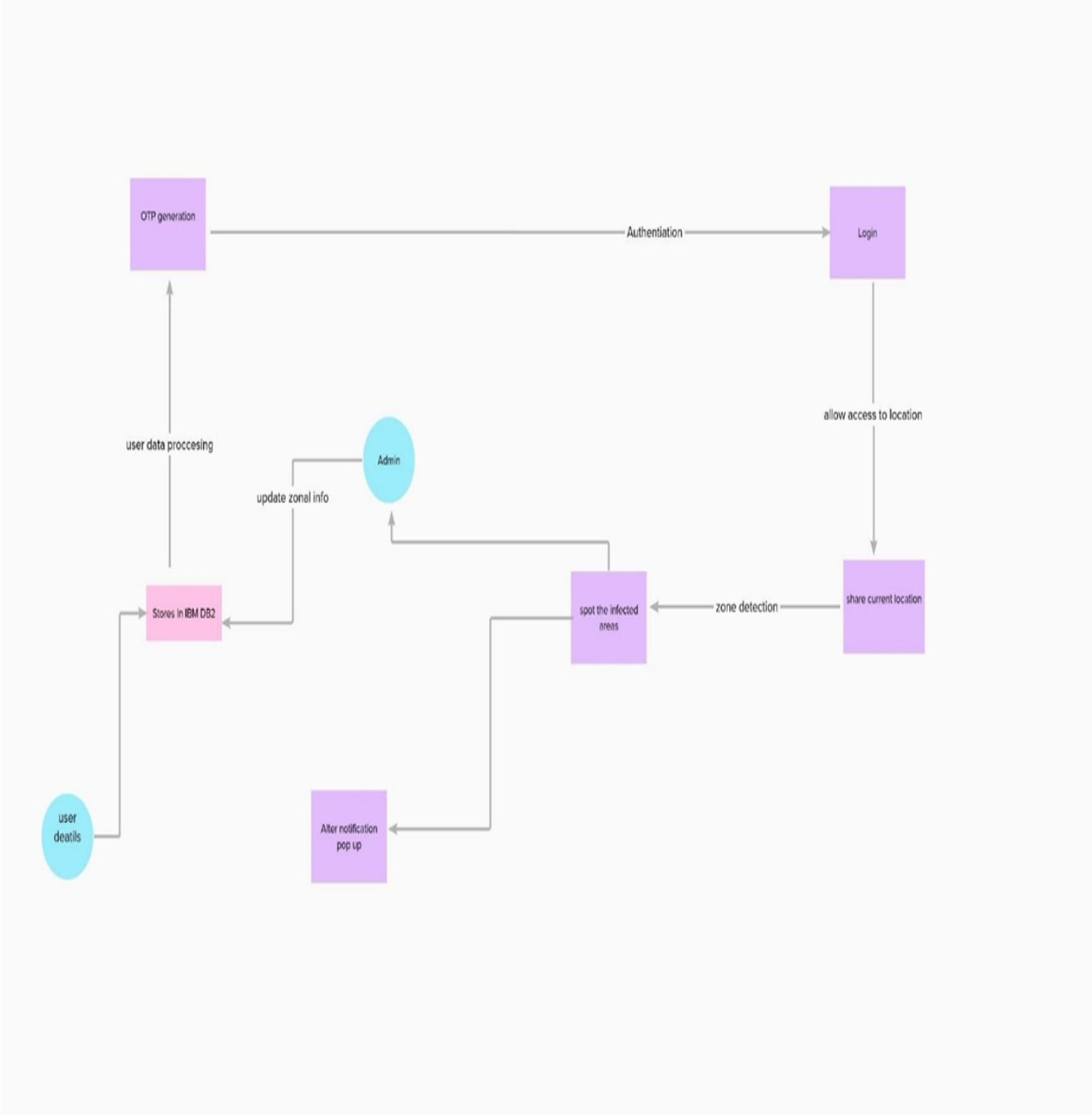
FR No.	Functional Requirement	Sub Requirement
FR 1	User Registration	Registration through Form using Mail or Phone number
FR 2	User Confirmation	Confirmation via Email Confirmation via OTP
FR 3	Alert message via Notification	GPS tracking and notification services
FR 4	Show infected zones	Geo-fence sketching
FR 5	Track alternate routes	Using Google Maps API

4.2 Non-Functional requirements

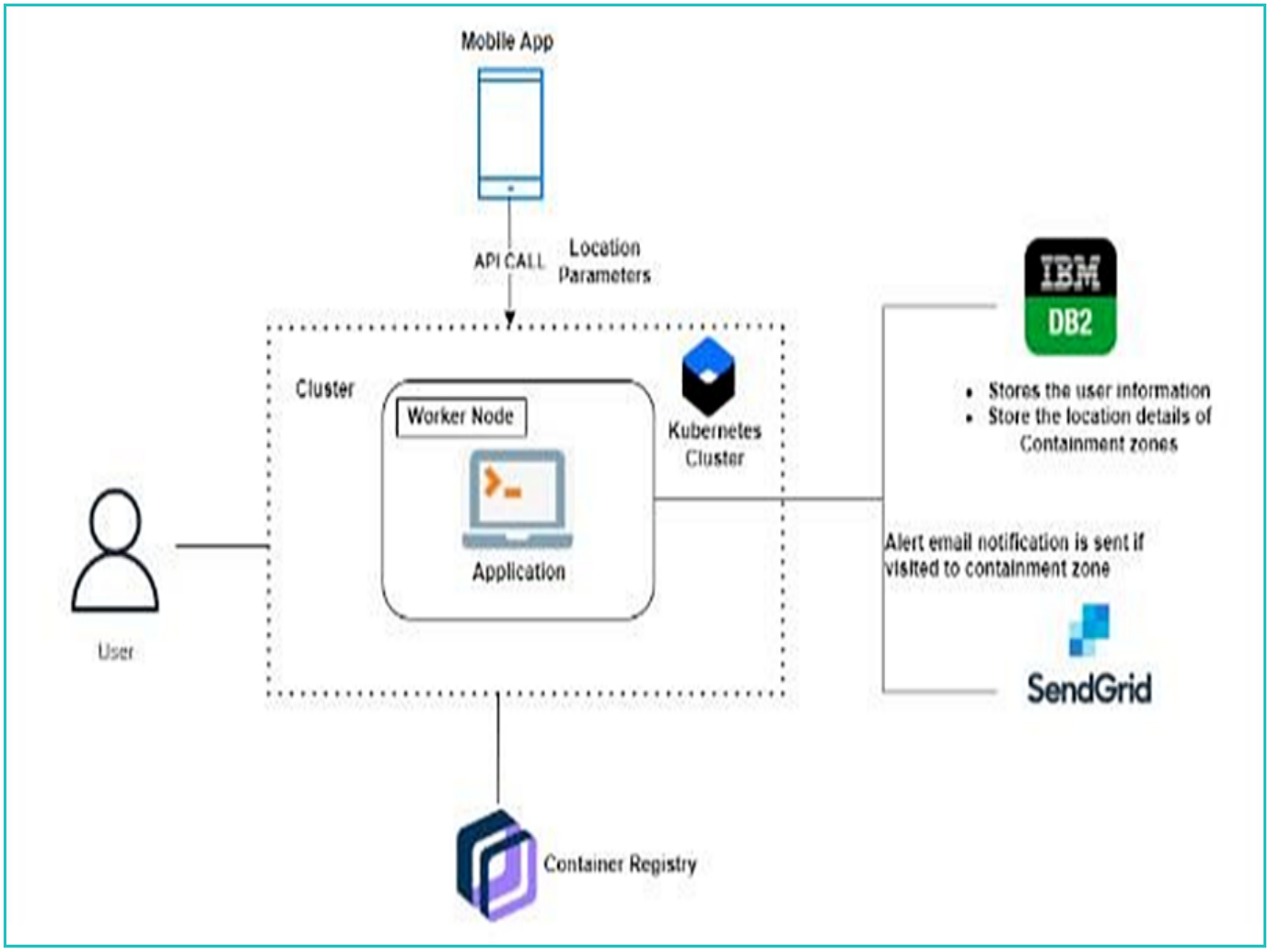
FR No.	**Non-Functional Requirement**	Description
NFR-1	Usability	GUI is easier to interact
NFR-2	Security	The data collected from the user will be stored securely
NFR-3	Reliability	The user can trust the results and navigate safely.
NFR-4	Performance	Accurate results can be achieved due to real-time location sharing
NFR-5	Availability	Available if the network bandwidth of the user is of good range.
NFR-6	Scalability	The application can be used from anywhere and can also be implemented for both mobile and web apps for the user to interact

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution Architecture



5.3 User Stories

User Type	Functional Requirement	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	As a user, I can register for the application through mobile number and the OTP that I have received.	High	Sprint-1
		USN-3	As a user, I can register for the application through mobile number and the OTP that I have received.	I can register & access my zone's information	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
	Zonal Information	USN-6	As a user, I will share my location and get the results which will include the zones that are infected within the area that I live in.	I can get the results and use it to navigate safely.	High	Sprint-3
Customer (Web user)	Alert message via notification	USN-7	As a user, I can travel safely and get out of the infected zone		Medium	Sprint-3
Administrator	Update infected zone information	USN-8	The administrator gets the information regarding the infected zones and updates it.	I can get the results and update it.	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

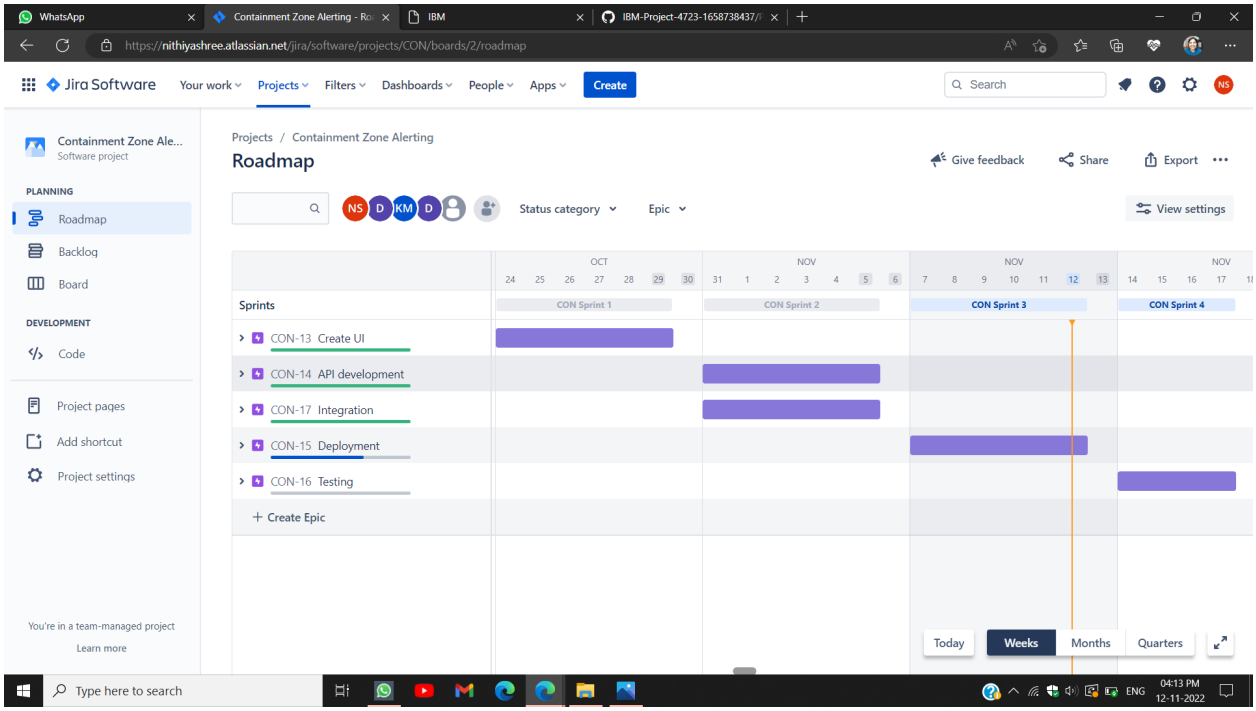
Sprint	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration (Web and Android)	USN-1	**USER** : I can register for the application by entering my email, password, and confirming my password.	2	High
		USN-2	**USER** : I will receive confirmation email once I have registered for the application	1	High
	Login (Web and Android)	USN-3	USN-3	3	High
Sprint-2	Dashboard	USN-4	**USER** : I need to give permission to access my location	5	High
		USN-5	**USER :** I can view the map with the containment zones	5	High
Sprint-3	Service	USN-6	**ADMIN :** I need to updated the containment zones	5	High
		USN-7	**ADMIN** : I need to differentiate the containment zones based on the intensity of infection	3	Medium
Sprint-4	Service	USN-8		5	HIgh

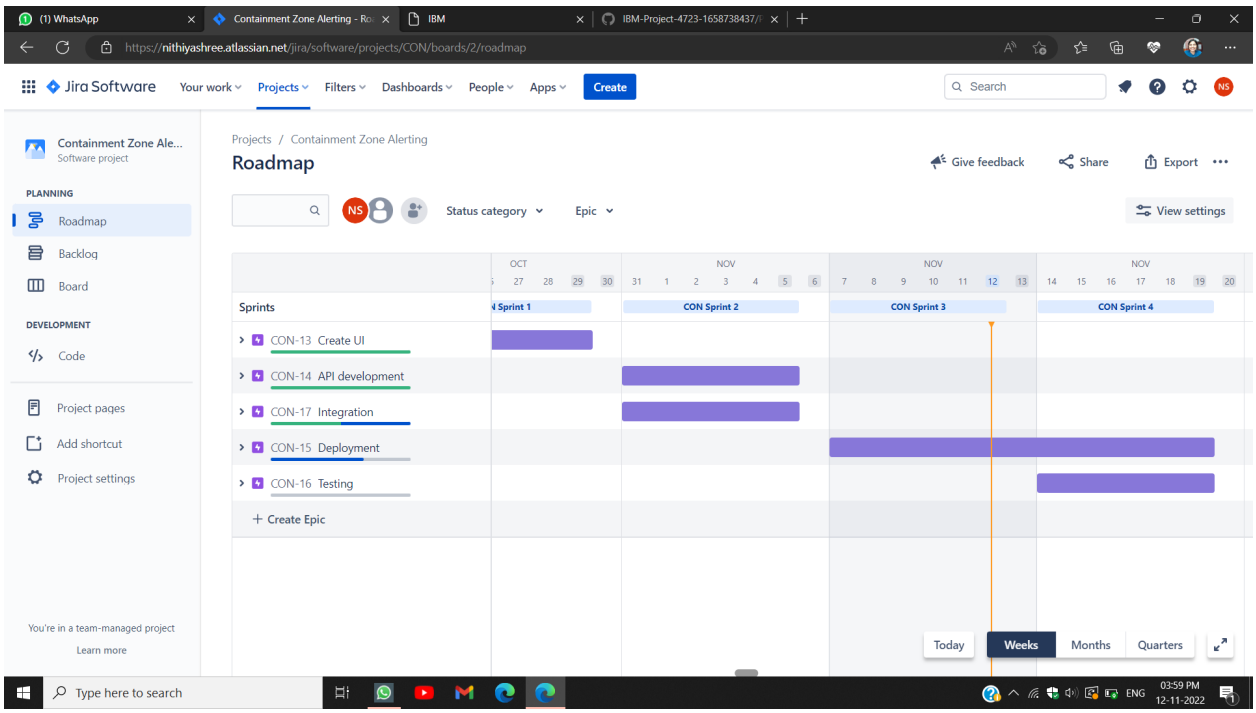
			ADMIN : I need to alert the user when they enter the containment zone through the notification.		
	Data Collection	USN-9	**ADMIN :** I need to store the user details on the cloud	5	High
		USN-10	**ADMIN :** I need to collect details about the containment zones from verified sources	5	High

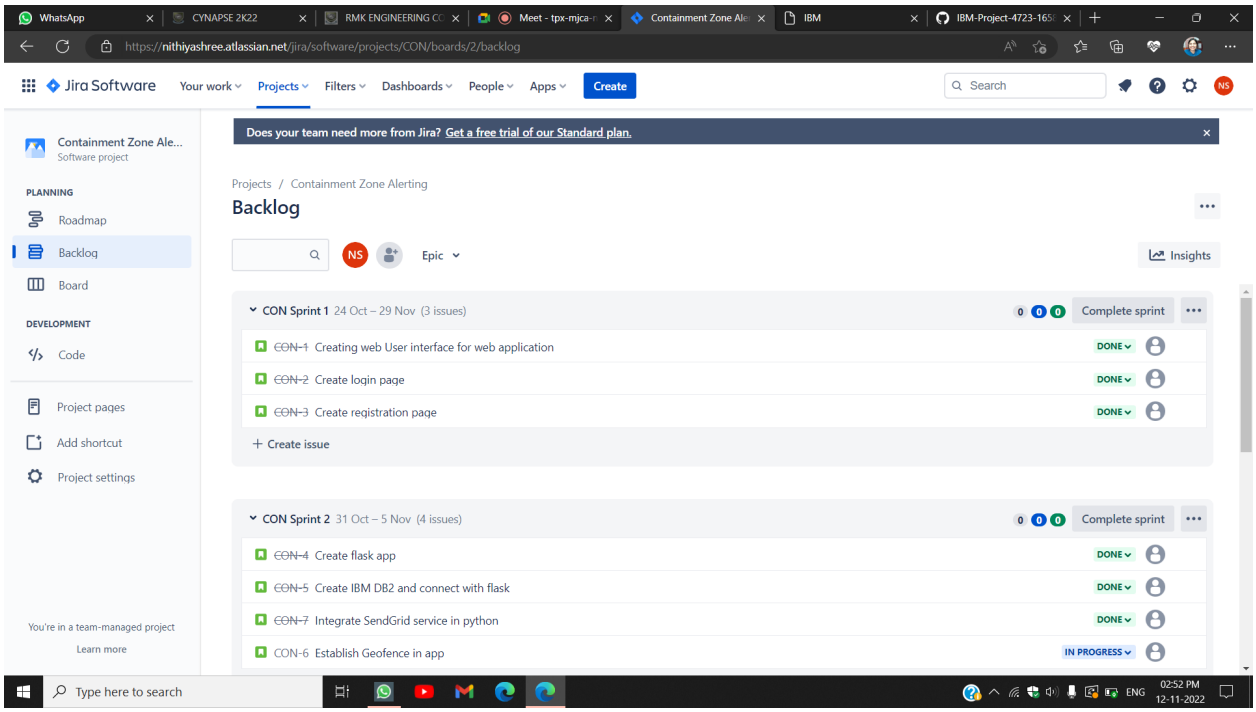
6.2 Sprint Delivery Schedule

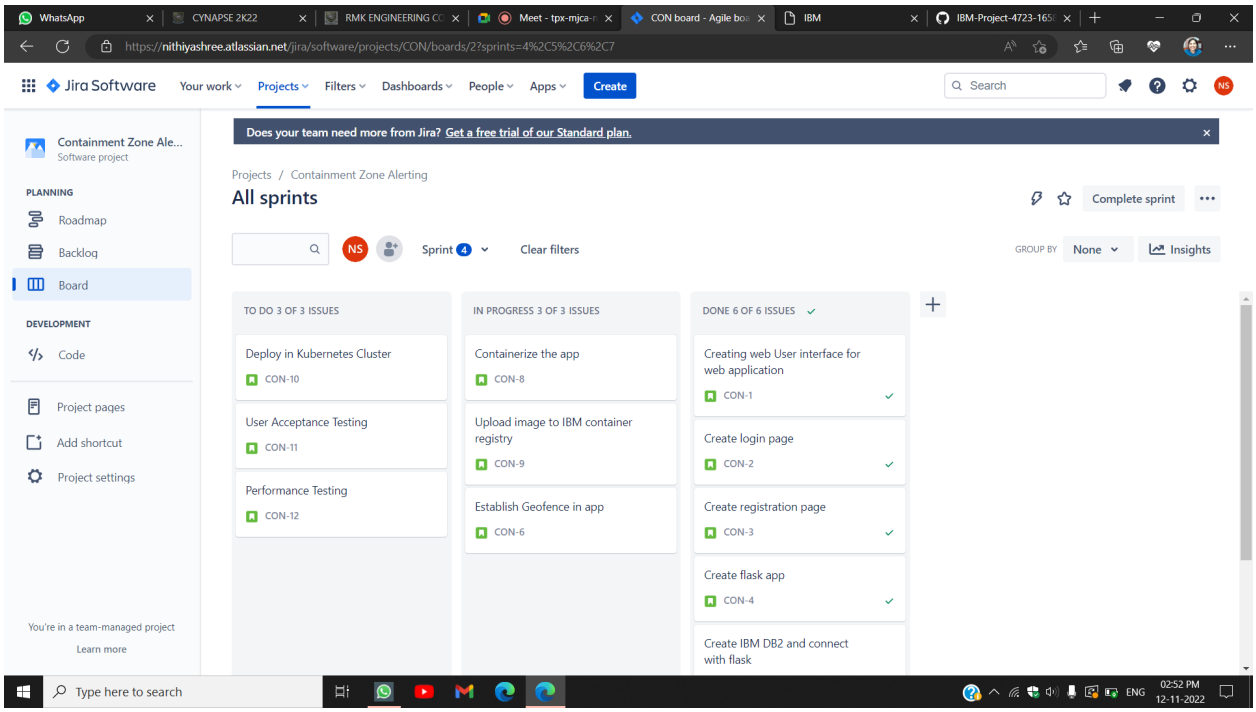
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24OCT 2022	29OCT 2022	20	29OCT 2022
Sprint-2	20	6 Days	31OCT 2022	05NOV 2022	20	05NOV 2022
Sprint-3	20	6 Days	07 NOV 2022	12 NOV 2022	20	12 NOV 2022
Sprint-4	20	6 Days	14 NOV 2022	19 NOV 2022	20	19 NOV 2022

6.3 Planning using JIRA






https://nithiyashree.atlassian.net/jira/software/projects/CON/boards/2/backlog
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Containment Zone Ale...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Containment Zone Alerting
Backlog
Epic
Insights
CON Sprint 1 24 Oct – 29 Nov (3 issues)
Complete sprint
CON-1 Creating web User interface for web application
CON-2 Create login page
CON-3 Create registration page
DONE
Create issue
CON Sprint 2 31 Oct – 5 Nov (4 issues)
CON-4 Create flask app
CON-5 Create IBM DB2 and connect with flask
CON-7 Integrate SendGrid service in python
CON-6 Establish Geofence in app
IN PROGRESS
Type here to search
ENG
02:52 PM
12-11-2022


https://nithiyashree.atlassian.net/jira/software/projects/CON/boards/2?sprints=4%2C5%2C6%2C7
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Containment Zone Ale...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Containment Zone Alerting
All sprints
Complete sprint
Sprint 4
Clear filters
GROUP BY
None
Insights
TO DO 3 OF 3 ISSUES
Deploy in Kubernetes Cluster
CON-10
User Acceptance Testing
CON-11
Performance Testing
CON-12
IN PROGRESS 3 OF 3 ISSUES
Containerize the app
CON-8
Upload image to IBM container registry
CON-9
Establish Geofence in app
CON-6
DONE 6 OF 6 ISSUES
Creating web User interface for web application
CON-1
Create login page
CON-2
Create registration page
CON-3
Create flask app
CON-4
Create IBM DB2 and connect with flask
Type here to search
ENG
02:52 PM
12-11-2022

7. CODING & SOLUTIONING

7.1 Feature 1(Admin Web app)

Admin login

admin can login with their respective login credentials to the account.

- **coding**

```
<html lang="en">
<head>

</head>
<body>
 <style>
 body{
margin:0;
color:#6a6f8c;
background:#c8c8c8;
font:600 16px/18px 'Open Sans',sans-serif;
}
*,:after,:before{box-sizing:border-box}
.clearfix:after,.clearfix:before{content:'';display:table}
.clearfix:after{clear:both;display:block}
a{color:inherit;text-decoration:none}

.login-wrap{
width:100%;
margin:auto;
max-width:525px;
min-height:670px;
position:relative;
background:url(https://cdn.shrm.org/image/upload/c_crop%2ch_617%2cw_1099%2
cx_0%2cy_0/c_fit%2cf_auto%2cq_auto%2cw_767/v1/Legal%20and%20Compliance/
coronavirus4m_utz5jt?databtoa=eyIxNng5Ijp7IngiOjAsInkiOjAsIngyIjoxMDk5LCJ5MiI6
NjE3LCJ3IjoxMDk5LCJoIjo2MTd9fQ%3d%3d) no-repeat center;

box-shadow:0 12px 15px 0 rgba(0,0,0,.24),0 17px 50px 0 rgba(0,0,0,.19);
}
.login-html{
width:100%;
height:100%;
position:absolute;
```

```
padding:90px 70px 50px 70px;
background:rgba(40,57,101,.9);
}
.login-html .sign-in-htm,
.login-html .sign-up-htm{
      top:0;
      left:0;
      right:0;
      bottom:0;
      position:absolute;
      transform:rotateY(180deg);
      back-face-visibility:hidden;
      transition:all .4s linear;
}
.login-html .sign-in,
.login-html .sign-up,
.login-form .group .check{
      display:none;
}
.login-html .tab,
.login-form .group .label,
.login-form .group .button{
      text-transform:uppercase;
}
.login-html .tab{
      font-size:22px;
      margin-right:15px;
      padding-bottom:5px;
      margin:0 15px 10px 0;
      display:inline-block;
      border-bottom:2px solid transparent;
}
.login-html .sign-in:checked + .tab,
.login-html .sign-up:checked + .tab{
      color:#fff;
      border-color:#1161ee;
}
.login-form{
      min-height:345px;
      position:relative;
      perspective:1000px;
      transform-style:preserve-3d;
}
.login-form .group{
      margin-bottom:15px;
}
```

```
.login-form .group .label,
.login-form .group .input,
.login-form .group .button{
      width:100%;
      color:#fff;
      display:block;
}
.login-form .group .input,
.login-form .group .button{
      border:none;
      padding:15px 20px;
      border-radius:25px;
      background:rgba(255,255,255,.1);
}
.login-form .group input[data-type="password"]{
      text-security:circle;
      -webkit-text-security:circle;
}
.login-form .group .label{
      color:#aaa;
```

ISBN-4723-166218126-4

```
      font-size:12px;
}
.login-form .group .button{
      background:#1161ee;
}
.login-form .group label .icon{
      width:15px;
      height:15px;
      border-radius:2px;
      position:relative;
      display:inline-block;
      background:rgba(255,255,255,.1);
}
.login-form .group label .icon:before,
.login-form .group label .icon:after{
      content:'';
      width:10px;
      height:2px;
      background:#fff;
```

```
      position:absolute;
      transition:all .2s ease-in-out 0s;
  }
  .login-form .group label .icon:before{
      left:3px;
      width:5px;
      bottom:6px;
      transform:scale(0) rotate(0);
  }
  .login-form .group label .icon:after{
      top:6px;
      right:0;
      transform:scale(0) rotate(0);
  }
  .login-form .group .check:checked + label{
      color:#fff;
  }
  .login-form .group .check:checked + label .icon{
      background:#1161ee;
  }
  .login-form .group .check:checked + label .icon:before{
      transform:scale(1) rotate(45deg);
  }
  .login-form .group .check:checked + label .icon:after{
      transform:scale(1) rotate(-45deg);
  }
  .login-html .sign-in:checked + .tab + .sign-up + .tab + .login-form .sign-in-htm{
      transform:rotate(0);
  }
  .login-html .sign-up:checked + .tab + .login-form .sign-up-htm{
      transform:rotate(0);
  }

  .hr{
      height:2px;
      margin:60px 0 50px 0;
  background:rgba(255,255,255,.2);
  }
  .foot-lnk{
      text-align:center;
  }
   </style>

  <div class="login-wrap">
  <div class="login-html">
type="radio" name="tab" class="sign-in" checked><label for="tab-2"
```

```
Instructions</label>
  <input id="tab-1" type="radio" name="tab" class="sign-up">
  <label for="tab-1" class="tab">Sign Up</label>
  <div class="login-form">
  <div class="sign-in-htm">
  <p>If you are new to this site please select "sign up" option in the tab <p><br>
  <label for="tab-2"><b>Already Member?Please login<b></label>
e="color:white"> <a href="/reg"> <u>Click Here!!!! </a> </p>
  </div>
  <div class="login-form">
  <form action="/log" method="post">

  <div class="msg" style="color:white">{{ msg }}</div>
  <br>

  <div class="sign-up-htm">
 <br>
  <div class="group">
  <label for="username" class="label">Username</label>
  <input type="text" name="username" class="input"  id="username" required>
  </div>
  <div class="group">
  <label for="email" class="label">Email Address</label>
  <input type="email" name="email"  class="input" id="email"required>
  </div>
  <div class="group">
  <label for="password" class="label">Password</label>
  <input type="password" name="password" class="input"  id="password"required>
  <input type="submit" class="button" value="Sign Up">
  </div>
  <div class="hr"></div>
  <div class="foot-link">
  <label for="tab-1">Already Member?Please login</label>
  <a href="\reg">Click Here!!!!</a>
  </div>
  </div>
  </div>
  </div>
  </div>
  </body>
  </html>
```

7.2 Feature 2 (android app)

```
package com.example.client_containment;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONException;
import org.json.JSONObject;
public class SignUp extends AppCompatActivity {
private EditText name;
private EditText email;
private EditText password;
SharedPreferences shared preferences;


    @Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sign_up);
     name = findViewById(R.id.name);
     email = findViewById(R.id.email);
    password = findViewById(R.id.password);

        shared-preferences = getApplicationContext().getSharedPreferences("user_data", 0);
        SharedPreferences.Editor editor = sharedpreferences.edit();
        editor.clear();
        editor.commit();
        if(sharedpreferences.getAll().size() >= 3){
            Intent intent = new Intent(this,MainActivity.class);
            startActivity(intent);
        }

    }

    public void signUp(View view) {
  if(!name.getText().equals("") || !email.getText().equals("") || !
password.getText().equals("")){
postDataUsingVolley(name.getText().toString(),email.getText().toString()
,password.getText().toString());
        }
    }
    private void postDataUsingVolley(String name, String email,String password) {
        final RequestQueue queue = Volley.newRequestQueue(this);
        String url = "http://192.168.1.11:8080/android_sign_up";
```

```
        JSONObject postpartum = new JSONObject();
        try {
            postparams.put("name", name);
            postparams.put("email", email);
            postparams.put("password", password);
        } catch (JSONException e) {
            e.printStackTrace();
JsonObjectRequest jsonObjReq = new JsonObjectRequest(Request.Method.POST,
 url, postparams)
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        Log.d("response",response.toString());

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


SharedPreferences.Editor editor = sharedpreferences.edit();
editor.putString("name", name);
  editor.putString("email", email);
  editor.putInt("id", userId );
editor.commit();
  Intent intent = new Intent(SignUp.this,MainActivity.class);
   startActivity(intent);

     } catch (JSONException e) {
          e.printStackTrace();
                        }

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

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

FEATURE 3

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

</head>
<body>
    <style>
    body{
      margin:0;
      color:#6a6f8c;    background:#c8c8c8;
      font:600 16px/18px 'Open Sans',sans-serif;
}
*,:after,:before{box-sizing:border-box}
.clearfix:after,.clearfix:before{content:'';display:table}
.clearfix:after{clear:both;display:block}
a{color:inherit;text-decoration:none}

.login-wrap{
      width:100%;
      margin:auto;
      max-width:525px;
      min-height:670px;
      position:relative;

background:url(https://cdn.shrm.org/image/upload/c_crop%2ch_617%2cw_1099%2cx_0%2cy_0/c_fi
jo2MTd9fQ%3d%3d) no-repeat center;
      box-shadow:0 12px 15px 0 rgba(0,0,0,.24),0 17px 50px 0 rgba(0,0,0,.19);
}
.login-html{
      width:100%;
      height:100%;
      position:absolute;
      padding:90px 70px 50px 70px;
      background:rgba(40,57,101,.9);
}
.login-html .sign-in-htm,
.login-html .sign-up-htm{
      top:0;
      left:0;
      right:0;
      bottom:0;
      position:absolute;
      transform:rotateY(180deg);
      backface-visibility:hidden;
      transition:all .4s linear;
```

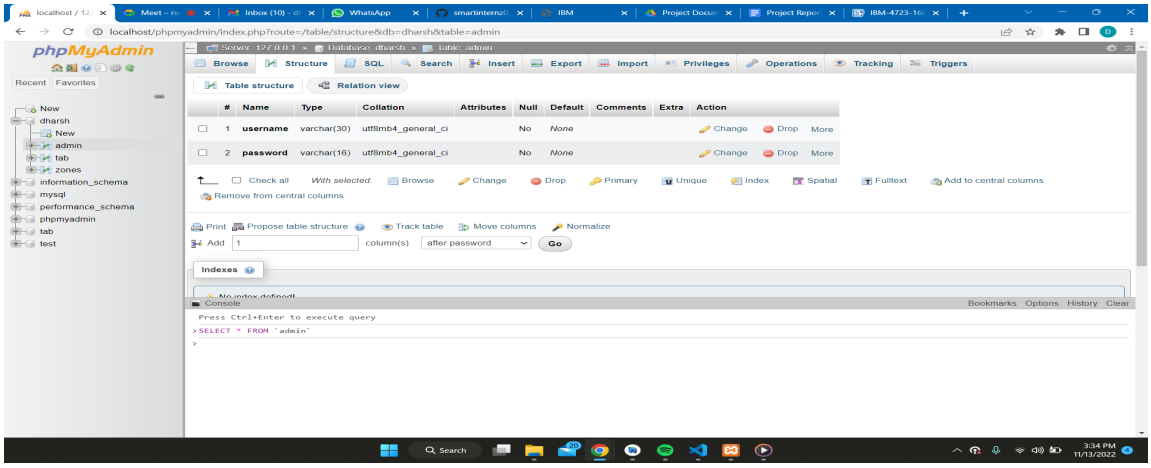
```
}
.login-html .sign-in,
.login-html .sign-up,
.login-form .group .check{
      display:none;
}
.login-html .tab,
.login-form .group .label,
.login-form .group .button{
      text-transform:uppercase;
}
.login-html .tab{
      font-size:22px;
      margin-right:15px;
      padding-bottom:5px;
      margin:0 15px 10px 0;
      display:inline-block;
      border-bottom:2px solid transparent;
}
.login-html .sign-in:checked + .tab,
.login-html .sign-up:checked + .tab{
      color:#fff;
      border-color:#1161ee;
}
.login-form{
      min-height:345px;
      position:relative;
      perspective:1000px;
      transform-style:preserve-3d;
}
.login-form .group{
      margin-bottom:15px;
}
.login-form .group .label,
.login-form .group .input,
.login-form .group .button{
      width:100%;
      color:#fff;
      display:block;
}
.login-form .group .input,
.login-form .group .button{
      border:none;
      padding:15px 20px;
      border-radius:25px;
      background:rgba(255,255,255,.1);
}
.login-form .group input[data-type="password"]{
```

```
      text-security:circle;
      -webkit-text-security:circle;
}
.login-form .group .label{
      color:#aaa;
      font-size:12px;
}
.login-form .group .button{
      background:#1161ee;
}
.login-form .group label .icon{
      width:15px;
      height:15px;
      border-radius:2px;
      position:relative;
      display:inline-block;
      background:rgba(255,255,255,.1);
}
.login-form .group label .icon:before,
.login-form .group label .icon:after{
      content:'';
      width:10px;
      height:2px;
      background:#fff;
      position:absolute;
      transition:all .2s ease-in-out 0s;
}
.login-form .group label .icon:before{
      left:3px;
      width:5px;
      bottom:6px;
      transform:scale(0) rotate(0);
}
.login-form .group label .icon:after{
      top:6px;
      right:0;
      transform:scale(0) rotate(0);
}
.login-form .group .check:checked + label{
      color:#fff;
}
.login-form .group .check:checked + label .icon{
      background:#1161ee;
}
.login-form .group .check:checked + label .icon:before{
      transform:scale(1) rotate(45deg);
}
.login-form .group .check:checked + label .icon:after{
```

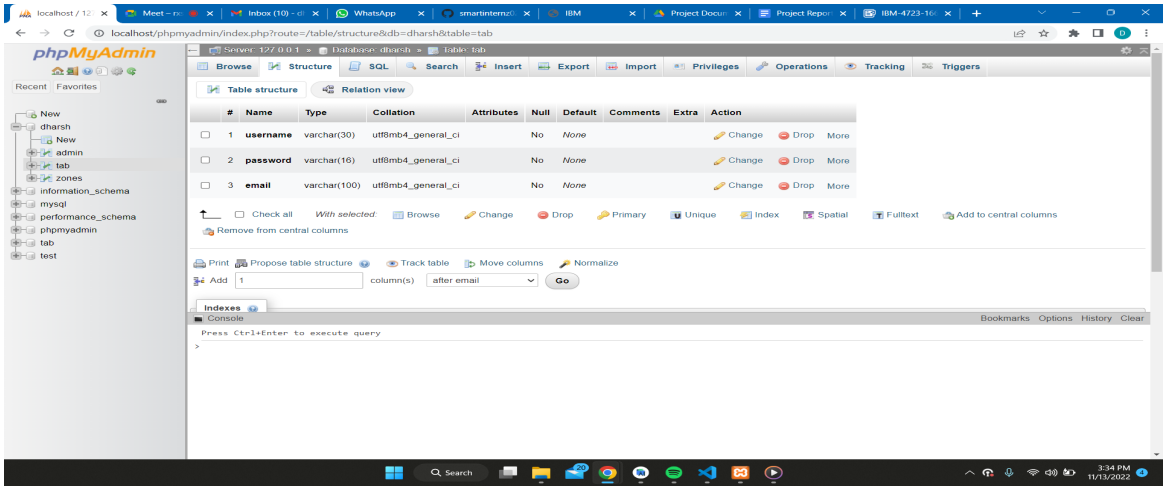
```
      transform:scale(1) rotate(-45deg);
}
.login-html .sign-in:checked + .tab + .sign-up + .tab + .login-form .sign-in-htm{
      transform:rotate(0);
}
.login-html .sign-up:checked + .tab + .login-form .sign-up-htm{
      transform:rotate(0);
}

.hr{
      height:2px;
      margin:60px 0 50px 0;
      background:rgba(255,255,255,.2);
}
.foot-lnk{
      text-align:center;
}
    </style>
<div class="login-wrap">
<div class="login-html">
  <label for="tab-2" class="tab" style="color:white">ADD ZONE</label>
     <div class="login-form">

    <form action="/addzone" method="post">

<div class="group"<label for="latitude" class="label">Latitude</label>
<input type="number" step="0.01" name="latitude" class="input"
 id="latitude" required</div><div class="group">
<label for="longitude" class="label">Longitude</label>
<input type="number" step="0.01" name="longitude"  class="input"
id="longitude"required>
 </div>
<div class="group">
 <label for="city" class="label">City</label>
 <input type="text" name="city" class="input"  id="city" required>
</div>
<div class="group">
 <label for="pincode" class="label">Pincode</label>
<input type="number" name="pincode" class="input"  id="pincode" required>
</div>
<div class="group">
 <input type="submit" class="button" value="Add Zone">
</div> </div>
  </div></body>
</html>
```

7.3 DATABASE SCHEMA

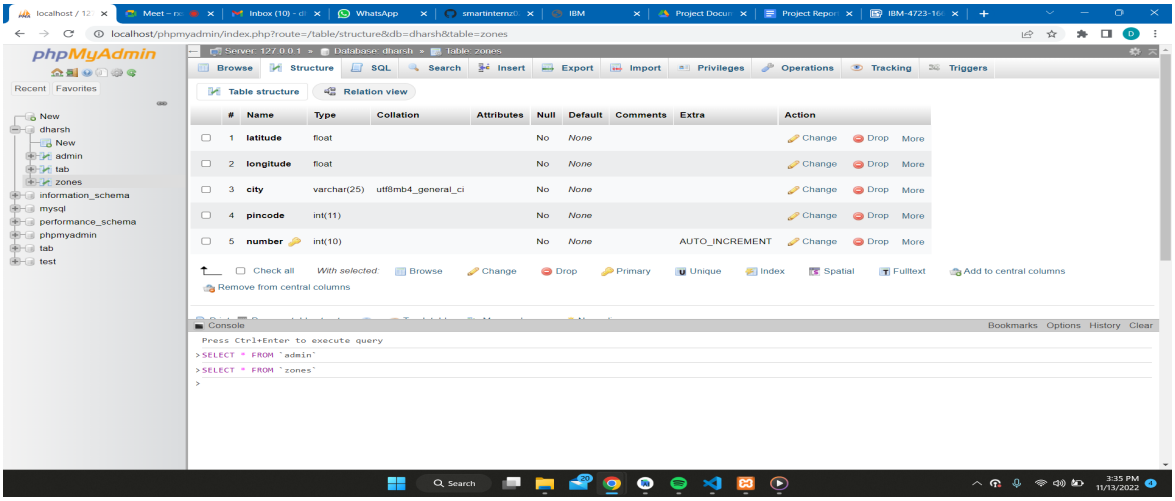
ADMIN TABLE:



USER INFO TABLE:



ZONE INFO TABLE:



8. TESTING

8.1 Test Cases

				Date	03-Nov-22						
				Team ID	PNT2022TMID00605						
				Project Name	Project - Containment Zone Alerting Application						
				Maximum Marks	4 marks						
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)
HomePage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on user+ button		1.Enter URL and click go 2.Click on user + dropdown button 3.Verify login/Singup popup displayed or not		Login/Signup popup should display	Working as expected	Pass		N
HomePage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on user + dropdown button 3.Verify login/Singup popup with below UI elements: a.sign in b.sign up		Application should show below UI elements: Verify login/Singup popup with below UI elements: a.sign in b.sign up	Working as expected	Pass		N
LoginPage_TC_OO3	Functional	Sign in page	Verify user is able to log into application with Valid credentials	user should have registered for an account already	1.Click on user+ dropdown button 2.Enter Valid username/email in Email text box 3.Enter valid password in password text box 4.Click on login button	Username: chalam@gmail.com password: Testing123	Application should display logged sucessfully User should navigate to user account dashboard	Working as expected	Pass		N
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box	Username: chalam@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass		N
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box	Username: chalam@gmail.com password: Testing123678678676876	Application should show password validation message.	Working as expected	Pass		N

TEST CASES FOR ADMIN PANEL

F12 | TEST CASES FOR ADMIN PANEL

				Date	03-Nov-22						
				Team ID	PNT2022TMID00605						
				Project Name	Project - Containment Zone Alerting Application						
				Maximum Marks	4 marks						
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)
					TEST CASES FOR ADMIN PANEL						
ADD ZONE PAGE_TC_OO7	UI	ZONE PAGE	Whether the Admin is able to add the zone in their respective fields	The Admin must know the latitude and longitude of the infected zones	1.Enter Latitude. 2.Enter Longitude. 3.Enter City. 4.Enter Pincode. 5.Click on Add Zone.	Longitude: 87.34 Longitude: 34.23 City: Chennai Pincode:600018 Testing123678678676876	Application should show Added Zone successfully	Working as expected	Pass		N
Display ZONE PAGE_TC_OO8	UI	ZONE PAGE	Whether the Admin is able to display the zone list	The Admin must display the infected zones	1.Enter PinCode 2.Click on Display Zone.	Pincode:600018 Testing123678678676876	Application should Display the Newly added Zone successfully	Working as expected	Pass		N
Remove ZONE PAGE_TC_OO9	UI	ZONE PAGE	Whether the Admin is able to remove the zone list	The Admin must remove the infected zones	1.Enter the Number. 2.Click on Remove Zone.	Number: 22 Testing123678678676876	Application should Remove the Zone successfully	Working as expected	Pass		N
Admin Panel PAGE_TC_OO10	UI	Login PAGE	Whether the Admin is able to remove the zone list	The Admin must be able to login.	1.Enter the Username. 2.Enter Password 3.Click on Login.	Username: John44@gmail.com Password: Testing123678678676876	Application should allow the admin to login successfully	Working as expected	Pass		N

8.2 USER ACCEPTANCE TESTING

DEFECT ANALYSIS

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Totals	22	14	8	26	57

TESTCASE ANALYSIS

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	30	0	0	30
Security	2	0	0	2
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 PERFORMANCE METRICS

NFT - RISK ASSESSMENT

NFT - Risk Assessment									
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volnem Changes	Risk Score	Justification
1	Containment zone alerting application	Existing	Low	No Changes	Moderate	This app is help to protect containment zones.	No Changes	RED	As we have seen the changes
2	Admin Panel page	New	Low	No Changes	Moderate	The page convenient to Login	>5to10%	ORANGE	As we have seen the Admin login.
3	Add Zone page	New	Low	No Changes	Moderate	The page allows to add zone.	>5to10%	ORANGE	As we have seen the added Zones.
4	Zone List	New	Low	No Changes	Moderate	The page displays the Zone List.	>5to10%	ORANGE	As we have seen the zone list displayed.
5	Sign Up Page	New	Low	No Changes	Moderate	The Page allows the user to register.	>5to10%	ORANGE	As we have seen the user has registered.
6	Sign in Page	New	Low	No Changes	Moderate	The Page allows the user to to login.	>5to10%	ORANGE	As we have seen the user has logged in.

END OF TEST PLAN

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	sumptions/Dependencies/Ris	Approvals/SignOff
1	Containment zone alerting application.	Tests are clear.	The App tend to protect the containment zone.	Approved

End Of Test Report								
S.No	Project Overview	FT Test approac	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Containment zone alerting application	Tests are clear.	No-met	Good	Go	The App need more cloud memory to store the	No issues.	Approvals.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The users are able to know that they are in a containment zone and are able to take preventive measures by moving out to avoid the spread of the diseases.

- Public workers like cab drivers and healthcare workers can travel safely to reach their destination by avoiding the containment zone.

DISADVANTAGES:

- User has to keep their live location active every minute to get updates.

- The application can track the users location only if the user has a uninterrupted internet connection.

- The major task is to monitor and update the location which are infected.

11. CONCLUSION

This alerting application about containment zones is developed using flask, python and deployed to cloud using docker.This application is intended to provide information about the containment zone to the users immediately. This helps users to protect themselves from getting infected by COVID by alerting them immediately through emails to their devices.It holds a great social cause in protecting the society and contributes to better health of people who has poor immunity due to old age and other allergies.

12. FUTURE SCOPE

This Application is deployed on cloud using docker compatible for the developers in few years to improve the application with ease. The application allows the user to get informed much faster and the UI for the app is even desirable.
This holds a social cause as yearsahead , fatal diseases might come and this app will become indispensable for the people to be aware of their surroundings that take health and hygiene seriously and has concern on the society to avoid the spread.

13. APPENDIX

SOURCE CODE

```
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re
import requests
from flask_bootstrap import Bootstrap
import smtplib
from email.message import EmailMessage
import sendgrid
import os
from sendgrid.helpers.mail import Mail, Email, To, Content

app = Flask(__name__)
global email

bootstrap = Bootstrap(app)
app.secret_key = 'a'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'dharsh'
app.config['MYSQL_PASSWORD'] = 'y7SaG8yR5o'
app.config['MYSQL_DB'] = 'dharsh'
mysql = MySQL(app)
def send_email(email):
    sg = sendgrid.SendGridAPIClient

    ('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')
    from_email = Email("dharshdummy@gmail.com")  # Change to your verified sender
    to_email = To(email)  # Change to your recipient
    subject = "Alert! You are in a Containment zone"
    content = Content("text/plain", "You are in a Containment zone.

            Move out to a safer zone and take preventive measures.

Get tested to check if you are infected too for the safety of your

family members and friends. Get vaccinated too. Stay safe!")
    mail = Mail(from_email, to_email, subject, content)
    # Get a JSON-ready representation of the Mail object
    print("MAIL JSON")
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
```

```
        #print("ReSPONSE STATUS CODE LINE")
    print(response.status_code)
    #print("Body")
    print(response.body)
    #print("Headers")
    print(response.headers)

def check_if_in_zone(email):
    email=email
    user=get_user_details()
    cur = mysql.connection.cursor()
    cur.execute(f'SELECT * FROM zones WHERE pincode={user[3]}')
    mysql.connection.commit()
    dataa = cur.fetchall()
    list_of_locations=[]
    for x in dataa:
        x=list(x)
        x[-1]=0
        list_of_locations.append(x)
    if user in list_of_locations:
        send_email(email)
def get_user_details():
    res = requests.get('https://ipinfo.io/')
    dataa = res.json()
    city=dataa['city']
    pincode=int(dataa['postal'])
    location = dataa['loc'].split(',')
    latitude = location[0]
    latitude=float(latitude[0:5])
    longitude = location[1]
    longitude=float(longitude[0:5])
    user=[latitude, longitude, city, pincode, 0]
    return user
@app.route('/')
def homer():
    return render_template('base.html')
@app.route('/log', methods =['GET', 'POST'])
def register():
    global email
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM tab WHERE username = % s', (username, ))
        account = cursor.fetchone()
        if account:
```

```
                msg = 'Account already exists !'
            return render_template('log.html', msg=msg)
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
            return render_template('log.html', msg=msg)
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
            return render_template('log.html', msg=msg)
        else:
            cursor.execute('INSERT INTO tab VALUES (% s, % s, % s)',

            (username,password,email))
            mysql.connection.commit()
            msg = 'You have successfully registered !'
            return render_template('signin.html', msg=msg
    return render_template('log.html', msg = msg)
@app.route('/reg',methods =['GET', 'POST'])
def login():
    global email
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM tab WHERE username = % s AND password = % s',

(username, password ),)
        account = cursor.fetchone()
        if account:
            session['loggedin'] = True
            session['id'] = account[0]
            userid=  account[0]
            session['username'] = account[1]
            msg = 'Logged in successfully !'

            msg = 'Logged in successfully !'
            return render_template('main.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('signin.html', msg = msg)


@app.route('/host', methods=['POST', 'GET'])
def admin_login():
    global email
    msg = ''
    if request.method == 'POST' :
        username = request.form['uname']
```

```
        password = request.form['pwd']
                cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM admin WHERE username = % s AND password = % s',

        (username, password))
        account = cursor.fetchone()
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account[0]
            userid=account[0]
            session['uname'] = account[1]
            msg = 'Logged in successfully !'
            return render_template('logged.html', msg=msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('host.html', msg = msg)

@app.route('/loc', methods=['POST'])
def main_html():
    if request.method == 'POST':
        email=request.form.get("mail")
        if email:
            check_if_in_zone(email)
            res = requests.get('https://ipinfo.io/')
            dataa = res.json()
            city=dataa['city']
            return render_template('location.html', city=city)
    return render_template('main.html')

@app.route('/check',methods =['GET', 'POST'])
def display():

    res = requests.get('https://ipinfo.io/')
    dataa = res.json()
    city=dataa['city']
    return render_template('location.html', city=city)

@app.route('/table')
def table():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM zones")
    data = cur.fetchall()
    return render_template('table.html', data=data)

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

```
        latitude = request.form['latitude']
        longitude = request.form['longitude']
                city = request.form['city']
        pincode = request.form['pincode']
        number= 'NULL'
        cur = mysql.connection.cursor()
        cur.execute('INSERT INTO zones VALUES (% s, % s, % s, % s, %s)',

(latitude, longitude, city, pincode,number))
        mysql.connection.commit()
        return render_template('success.html')

    return render_template('addzone.html')

@app.route('/getpincode', methods=['GET','POST'])
def get_pin_code():
    if request.method=='POST':
        pincode=request.form['pincode']
        cur = mysql.connection.cursor()
        cur.execute(f'SELECT * FROM zones WHERE pincode={pincode}')
        mysql.connection.commit()
        data = cur.fetchall()
        return render_template('display_for_pincode.html', data=data)

    return render_template('get_pincode.html')



@app.route('/removezone', methods=['GET', 'POST'])
def remove_zone():
    if request.method== 'POST':
        # latitude = request.form['latitude']
        # longitude = request.form['longitude']
        # city = request.form['city']
        number = request.form['number']
        cur = mysql.connection.cursor()
        cur.execute(f'DELETE FROM zones WHERE number={number}')
        mysql.connection.commit()
        return render_template('success.html')

    return render_template('removezone.html')

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

@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'
        signup_cursor = mysql.connection.cursor()


        user_result = signup_cursor.execute(
            "SELECT * FROM tab WHERE email=%s", [email]
        )
        if(user_result > 0):
            signup_cursor.close()
            return {'status': 'failure'}
        else:
            # execute the query
            signup_cursor.execute(
                'INSERT INTO tab (username,password,email) VALUES(%s,%s,%s)', (
                    name,password,email)
                )
            mysql.connection.commit()
            id_result = signup_cursor.execute(
                'SELECT username FROM tab WHERE email = %s', [email]
            )
            if(id_result > 0):
                id = signup_cursor.fetchone()
                return {"id": id[0]}
            signup_cursor.close()
    return {"status": "failure"}
if __name__ == '__main__':
    app.run(host='0.0.0.0',debug = True,port = 8080)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-4723-1658738437

PROJECT DEMO LINK:

https://github.com/IBM-EPBL/IBM-Project-4723-1658738437/blob/main/Final%20Deliverables/demo__video.mp4