**Containment Zone Alerting Application**

**TEAM ID : PNT2022TMID22562**

***Logesh P.R***

***Jayadev Kavle P***

***Balasubramaniyam V.R***

***Yogesh Thulasiraman***

**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

**5. PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

**6. PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

**7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

7.1 Data.html

7.2 home.html

7.3 login.html

7.4 signup.html

7.5 Dockerfile

7.6 app.py

**8. TESTING**

8.1 Test Cases

8.2 User Acceptance Testing

**9. RESULT**

**1. 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 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**

**Existing System :**

The Existing Containment Zone Alerting system detects the Containment Zone using Bluetooth Technology and it does not alert for a particular street it shows a particular region. And it can't be used without turning ON Bluetooth Feature

**Proposed System :**

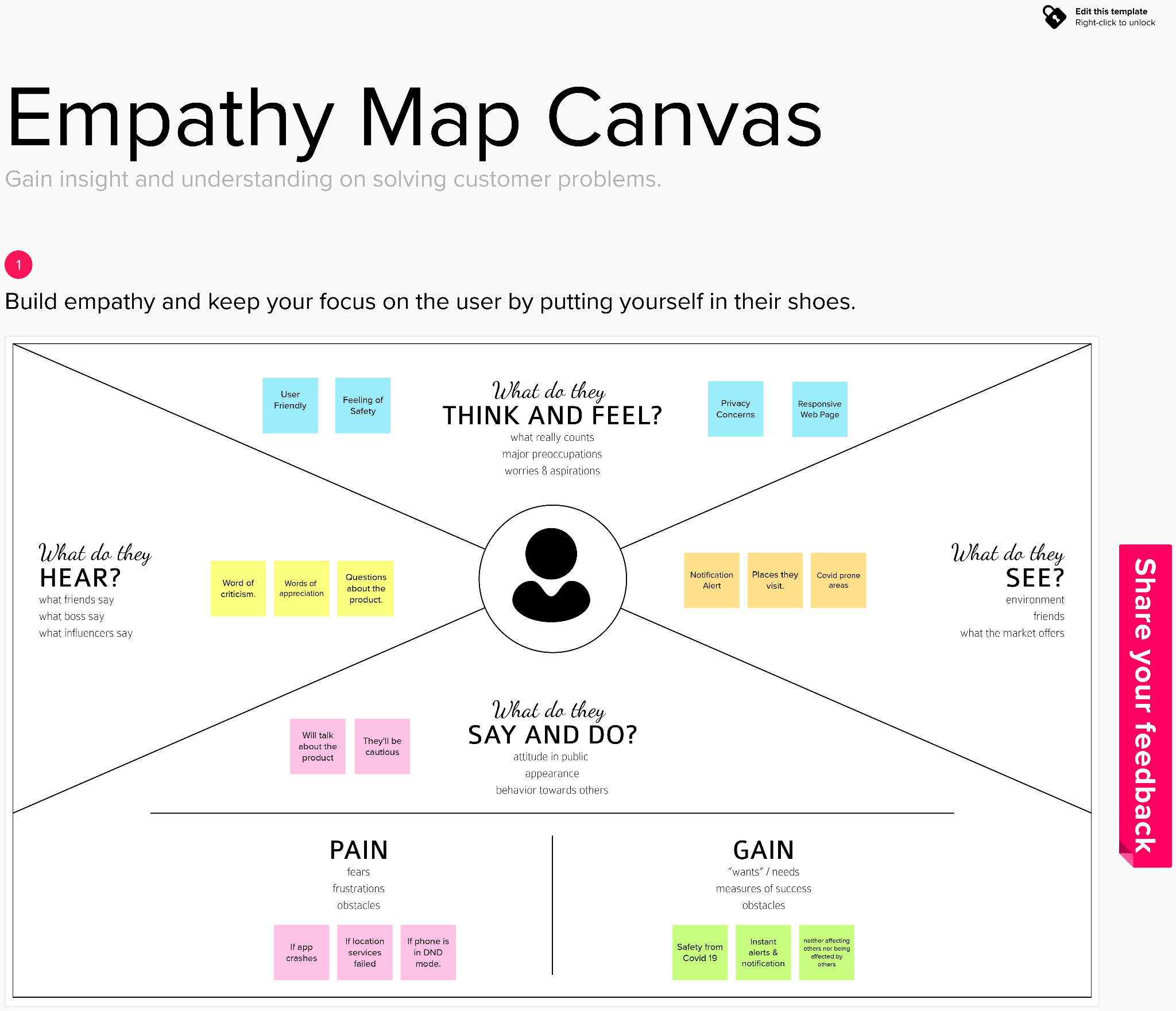
Containment Zone Alerting Application is designed to send the accurate street location of the location of the Containment Zone using location feature and it sends the alert through notiﬁcation of our Application.

**Problem Statement Definition :**

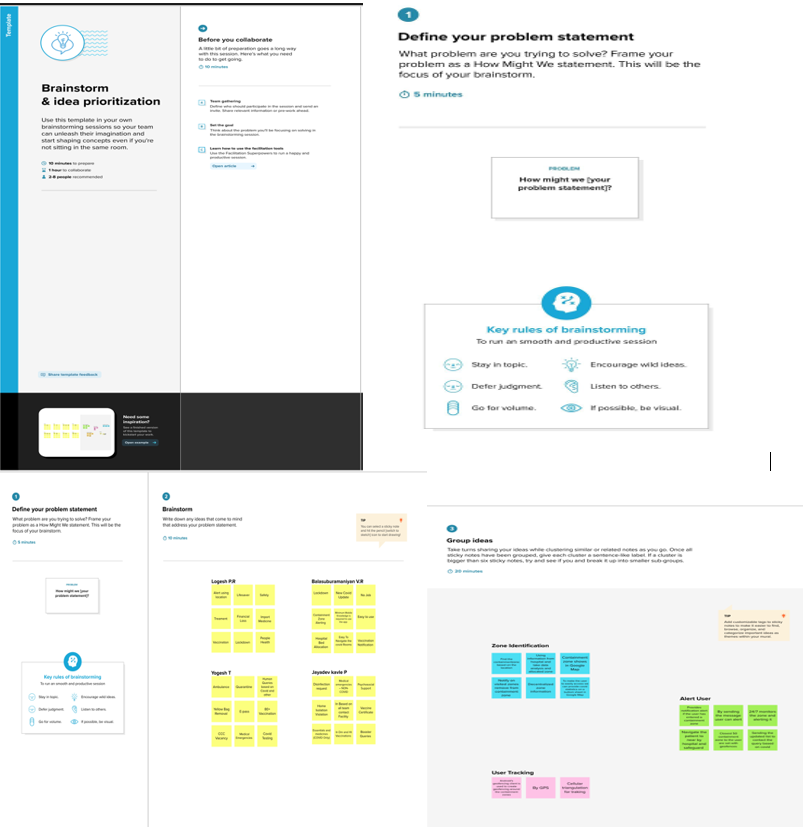
|  |  |
| --- | --- |
| Who does the problem affect? | People who are traveling to places being unaware of COVID Containment Zones. |
| What is the issue? | If people get into a COVID containment zone, there arises a high risk for them getting into contact with an affected person and thereby it results in  the transmission of the disease. |
| When does the issue occur? | When a healthy person gets into contact with an affected person living in a COVID prone area. |
| Where does the issue occur? | Getting affected by COVID might occur anywhere.  Nonetheless, the risk of getting affected by the disease would be very high in Containment Zones. |
| Why is it important that we ﬁx the problem? | It is important that we ﬁx the problem that people become aware of Containment Zones and stay away from them to protect themselves from  COVID-19 |
| What solution to solve this issue? | An Alerting application that delivers instant alerts whenever a person enters a COVID Containment  Zone. |
| What methodology used to solve the issue? | Geofencing of Containment Zones, Constant location monitoring of the users of the application,  and instant alert delivery. |

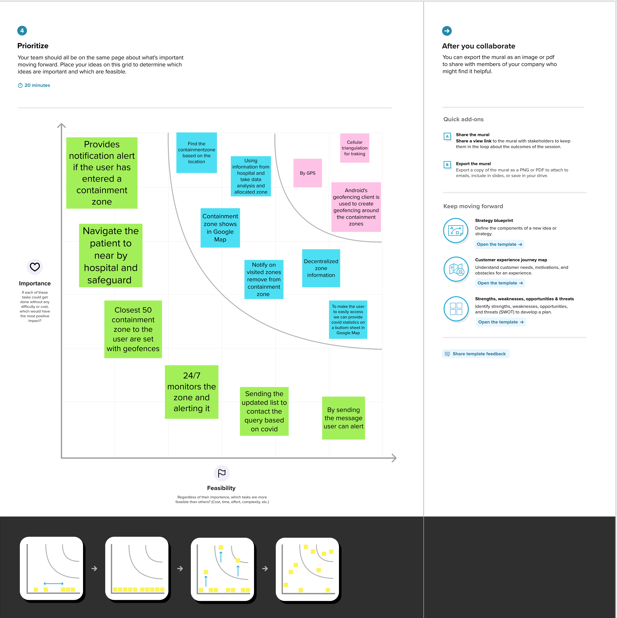
**3. IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**



**3.2 Ideation & Brainstorming:**

****

****

**3.3 Proposed Solution**

**Project team shall ﬁll the following information in proposed solution template.**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| **1.** | Problem Statement (Problem to be solved) | The application uses Firestore which is a ﬂexible and scalable database for mobile, web and server developments from Firebase and Google cloud  platform. |
| **2.** | Idea / Solution description | A collection is created in Cloud  Firestore with containment zones as documents. Each document has four ﬁelds: latitude, longitude, location name and radius |
| **3.** | Novelty / Uniqueness | 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 |
| **4.** | Social Impact / Customer Satisfaction | The application further extracts the IMEI number of the trespasser and uploads it to the online database. |

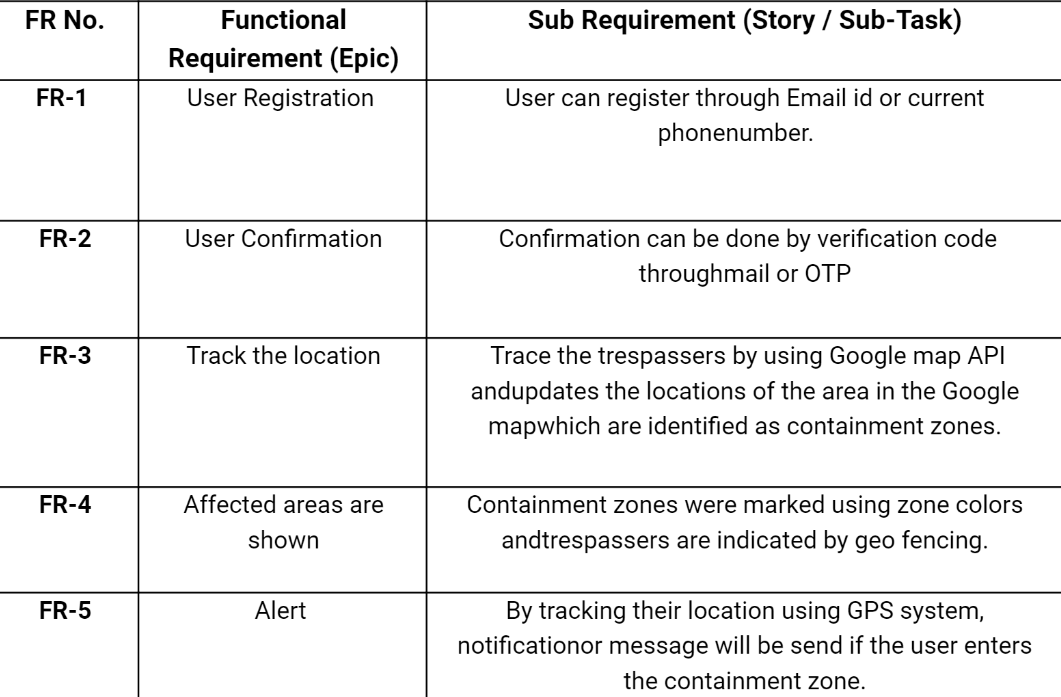
|  |  |  |
| --- | --- | --- |
| **5.** | Scalability of the Solution | Tests have been carried out in various containment zones across West Bengal for the validation of the Android  application. |
| **6.** | Business Model (Revenue Model) | With the help of getters each data from the document is retrieved and are  converted to string. |

**3.4 Problem Solution fit**

****

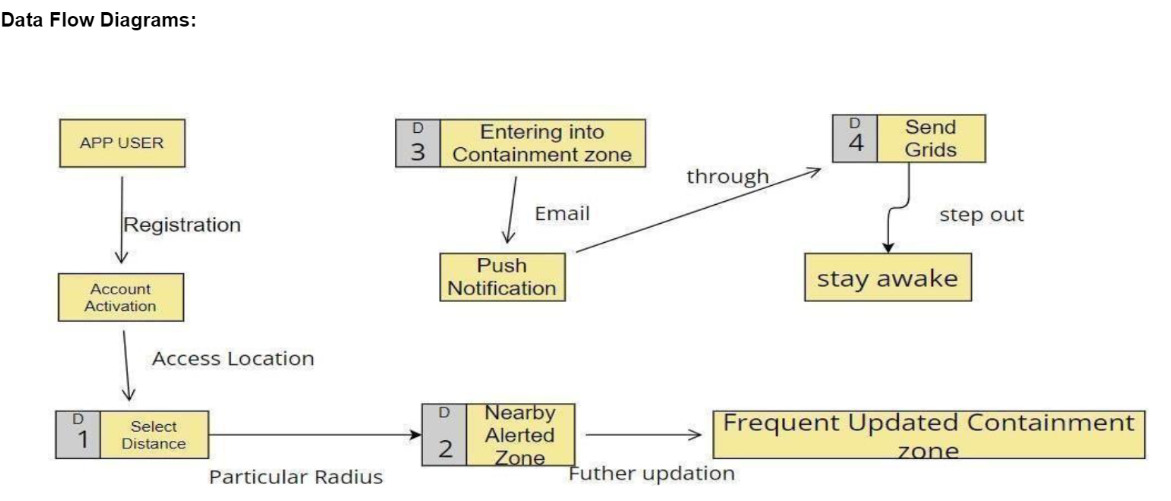
**4. REQUIREMENT ANALYSIS**

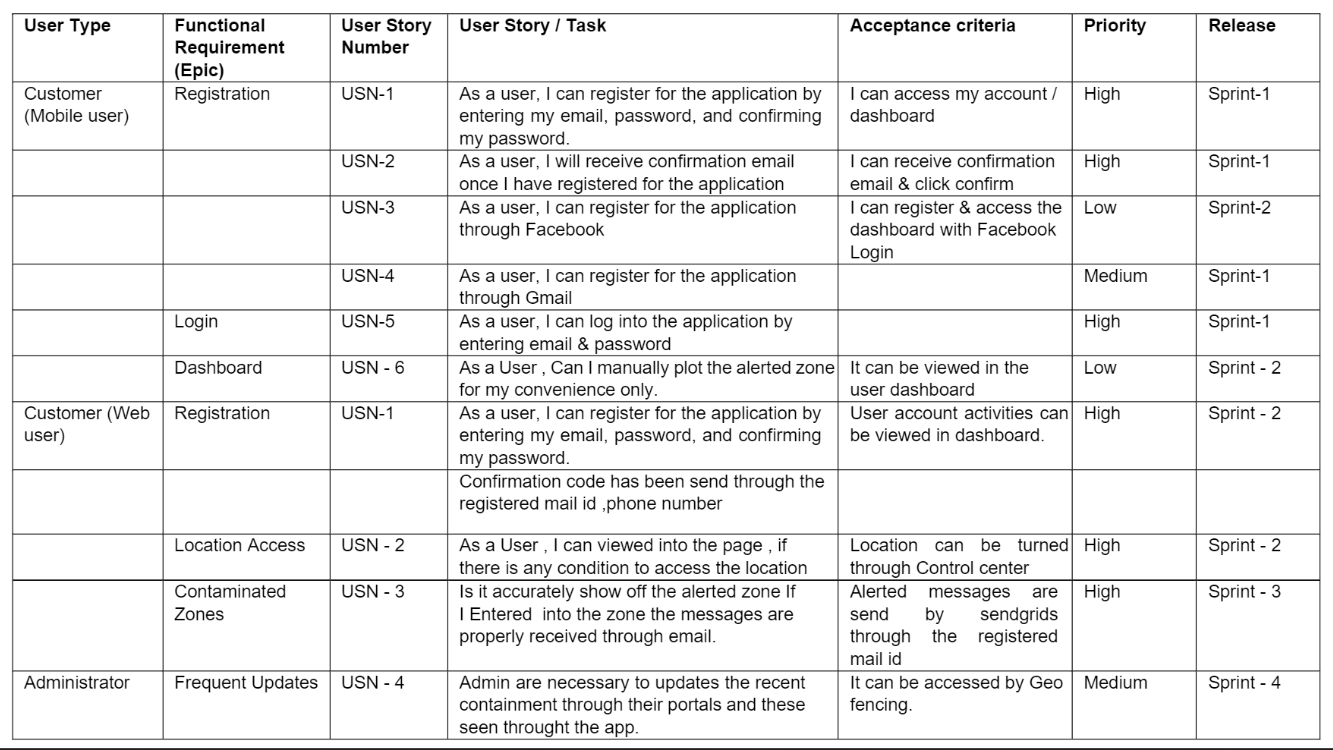
**4.1 Functional requirement:**



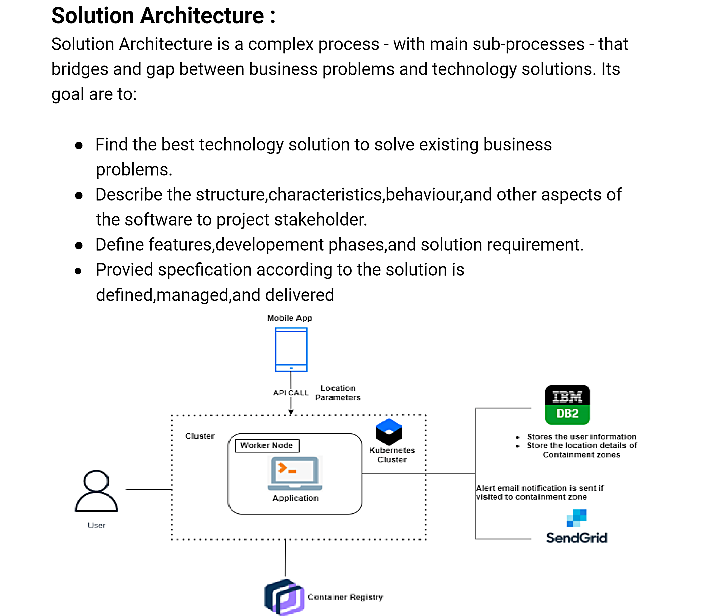
**5. PROJECT DESIGN**

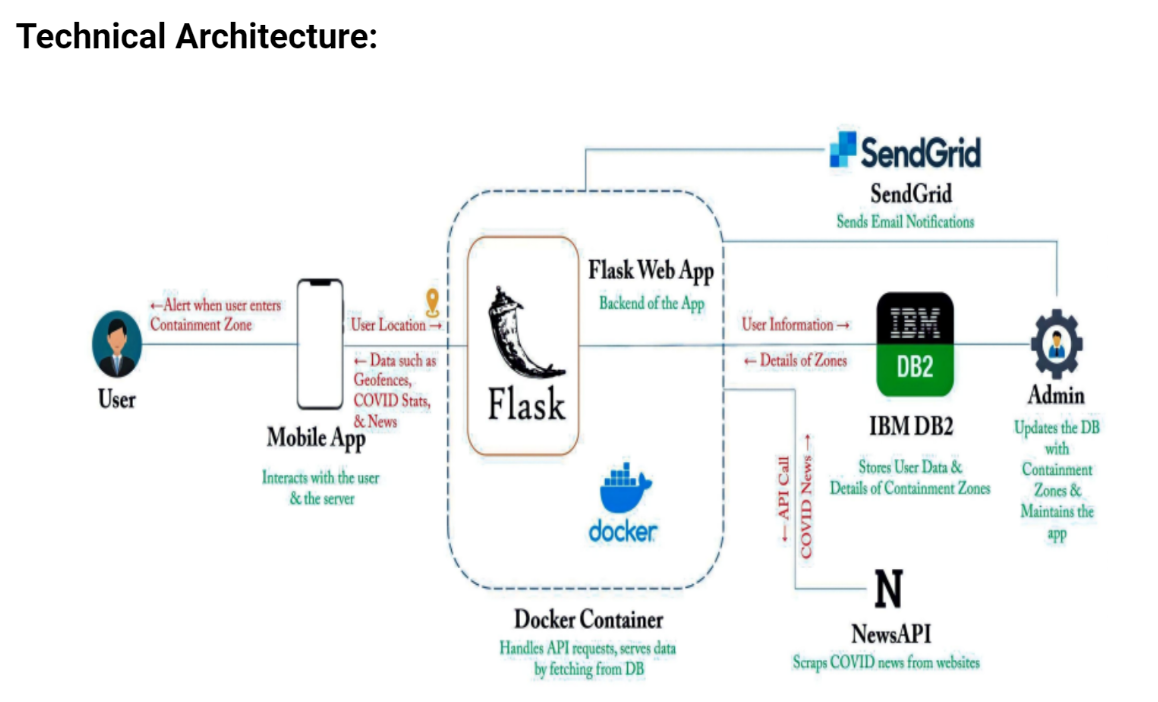
**5.1 Data Flow Diagrams:**

****

****

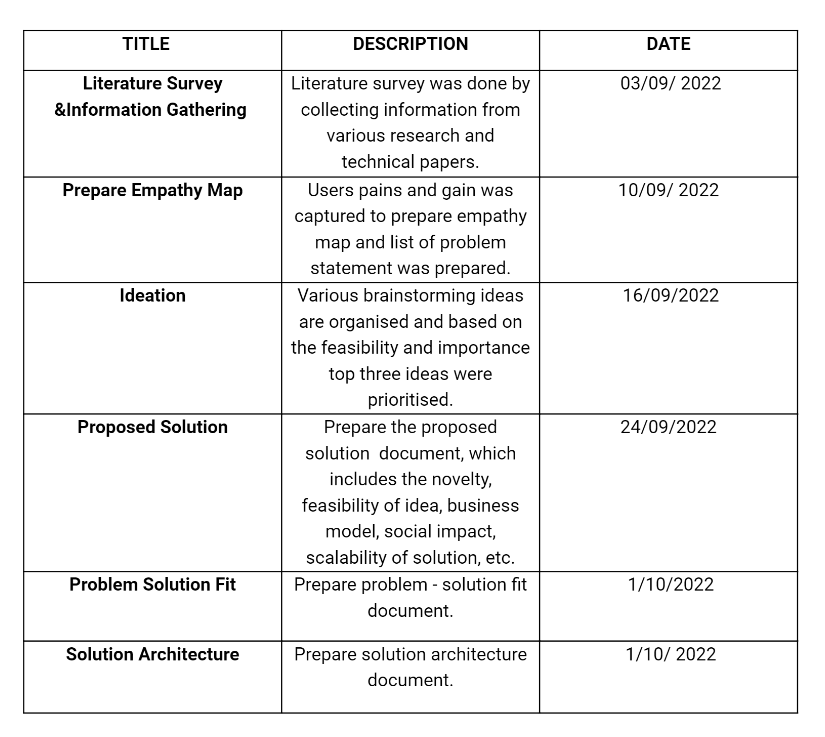
**5.2 Solution & Technical Architecture :**



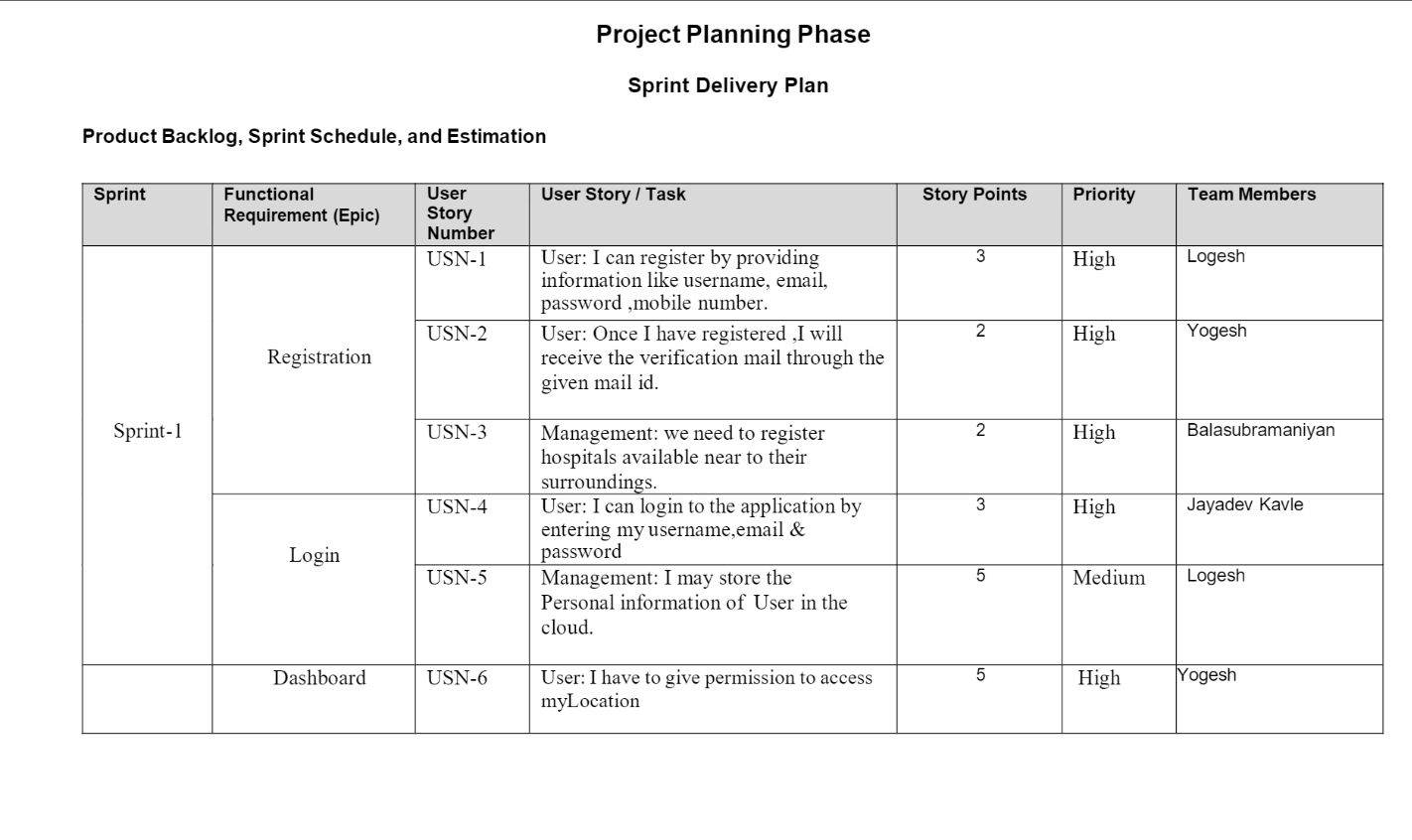


**6. PROJECT PLANNING & SCHEDULING**

**6.1 Sprint Planning & Estimation :**

****

**6.2 Sprint Delivery Schedule :**

****

**7. CODING & SOLUTIONING**

**Data.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Zones</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous" />

<style>

body {

padding-top: 30px;

padding-bottom: 30px;

background-color: #699cc5;

}

a {

color: black;

}

</style>

</head>

<body>

<div class="m-4 container">

<h1><u>Location data and Visited People</u></h1>

</div>

<div class="m-4 container">

<table class="table">

<thead>

<tr>

<th scope="col">S.No</th>

<th scope="col">Latitude</th>

<th scope="col">Longitude</th>

<th scope="col">No\_Visited</th>

</tr>

</thead>

<tbody>

{%- for row in responses %}

<tr>

<th scope="row">{{loop.index}}</th>

<td>{{row[1]}}</td>

<td>{{row[2]}}</td>

<td>{{row[3]}}</td>

</tr>

{%- endfor %}

</tbody>

</table>

</div>

<div class="m-3 float-right">

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

</div>

</body>

</html>

**Home.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous" />

<style>

body {

padding-top: 30px;

padding-bottom: 30px;

background-color: #699cc5;

}

a {

color: black;

}

</style>

</head>

<body>

{% if success == True %}

<script>

alert("Location Uploaded Successfully");

</script>

{% elif success == 0 %}

<script>

alert("Enter Proper Location data");

</script>

{% endif %}

<div class="m-3 float-right">

<button type="button" class="btn btn-primary"><a href={{url\_for("logout")}}>Log Out</a></button>

</div>

<div class="container m-3">

<h1><u>Declare Containment Zone</u></h1>

</div>

<div class="container m-3">

<h3>welcome: {{name}}</h3>

</div>

<form method="POST" action="/home">

<div class="container">

<div class="form-group row">

<div class="col-sm-6">

<label class="control-label">Lat.:</label>

<input type="text" class="form-control" id="lat" name="lat" />

</div>

<div class="col-sm-6">

<label>Long.:</label>

<input type="text" class="form-control" id="lon" name="lon" />

</div>

<div class="col-sm-6">

<label>Get current Location:</label>

<button type="button" class="btn btn-warning" onclick="getLocation()">Current Location</button>

<label>(Click this first)</label>

</div>

</div>

<!-- map -->

<div id="map\_disp" style="height: 400px;width: 500px;"></div>

<div class="m-3 float-right">

<button type="submit" class="btn btn-danger">Declare Containment Zone</button>

</div>

<div class="m-3">

<button onclick="toggleTips()" type="button" class="btn btn-secondary">Tutorial</button>

<div id="tips" class="m-3">

<ol>

<li>Select The Location By Clicking the Current Location Button</li>

<li>Drag the Pin to change the location</li>

<li>Click on Declare Containment Zone to save the location to the database </li>

</ol>

</div>

</div>

<div class="m-3 float-right">

<button type="button" class="btn btn-warning"><a href="{{url\_for('data')}}">Click Here To View The

Containment Zones and Number of

people visited</a></button>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"

integrity="sha384-+YQ4JLhjyBLPDQt//I+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd026JF"

crossorigin="anonymous"></script>

<script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>

<script src="https://maps.google.com/maps/api/js?sensor=false&amp;libraries=places"></script>

<script

src="https://rawgit.com/Logicify/jquery-locationpicker-plugin/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>

**Login.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous" />

<link rel="stylesheet" href="style.css" />

<title>Log In</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body class="text-center">

{% if error == 1 %}

<script>

alert("Incorrect Password");

</script>

{% elif error == 2%}

<script>

alert("Create An Account");

</script>

{% else %}

{% endif %}

<form class="form-login" method="POST" action="/">

<h1 class="h3 mb-3 font-weight-normal">Log In to add the location of the containment zone</h1>

<label for="email" class="sr-only">Email address</label>

<input type="email" name="email" class="form-control" placeholder="Email address" required autofocus />

<label for="password" class="sr-only">Password</label>

<input type="password" class="form-control" placeholder="Password" name="password" required />

<button type="submit" class="btn btn-lg btn-primary btn-block mt-3">

Login

</button>

<a href={{url\_for("signup")}}>Don't have an account ... Create One</a>

</form>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"

integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"

crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"

crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"

integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"

crossorigin="anonymous"></script>

</body>

</html>

**Signup.html**

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous" />

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

<title>Sign Up</title>

</head>

<body class="text-center">

{% if error %}

<script>

alert("Email id already exists in the database");

</script>

{% endif %}

<form class="form-login" method="POST" action="/signup">

<h1 class="h3 mb-3 font-weight-normal">Sign Up to create an account with us</h1>

<label for="name" class="sr-only">Email address</label>

<input type="text" name="name" class="form-control" placeholder="Name" required autofocus />

<label for="email" class="sr-only">Email address</label>

<input type="email" name="email" class="form-control" placeholder="Email address" required />

<label for="password" class="sr-only">Password</label>

<input type="password" class="form-control" placeholder="Password" name="password" required />

<button type="submit" class="btn btn-lg btn-primary btn-block mt-3">

Signup

</button>

<a href={{url\_for("login")}}>Already have an account ... Login</a>

</form>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"

integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"

crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"

crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"

integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"

crossorigin="anonymous"></script>

</body>

</html>

**Dockerfile:**

FROM python:3.6.5-alpine

WORKDIR /Admin

ADD . /Admin

# Install required packages

RUN set -e; \

apk add --no-cache --virtual .build-deps \

gcc \

libc-dev \

linux-headers \

mariadb-dev \

python3-dev \

postgresql-dev \

;

RUN apk --update --upgrade add gcc musl-dev jpeg-dev zlib-dev libffi-dev cairo-dev pango-dev gdk-pixbuf-dev

COPY requirements.txt /Admin

RUN pip install -r requirements.txt

CMD ["python","app.py"]

**app.py**

# import statements

from logging import error

from flask import \*

from jinja2.utils import select\_autoescape

import bcrypt

from flask\_mysqldb import MySQL

import json

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

# initialization

app = Flask(\_\_name\_\_)

# config

app.secret\_key = "\x19Ts\xbe\xe7\x8c\_\r\x12Q\x14\x13>q\xb7'WTH0\x9f\xe4\xec\xb1"

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = ''

app.config['MYSQL\_DB'] = 'zone2'

mysql = MySQL(app)

# functions

def send\_mail(email):

print(email)

message = Mail(from\_email='varundutia.h@gmail.com',

to\_emails=email,

subject='caution',

plain\_text\_content='Please Stay Safe',

html\_content='<h2>You are entering into a containment Zone</h2>')

try:

sg = SendGridAPIClient(

'SG.7BJDtQDlS8unH0r5\_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG117YURqOTa2HL')

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"}

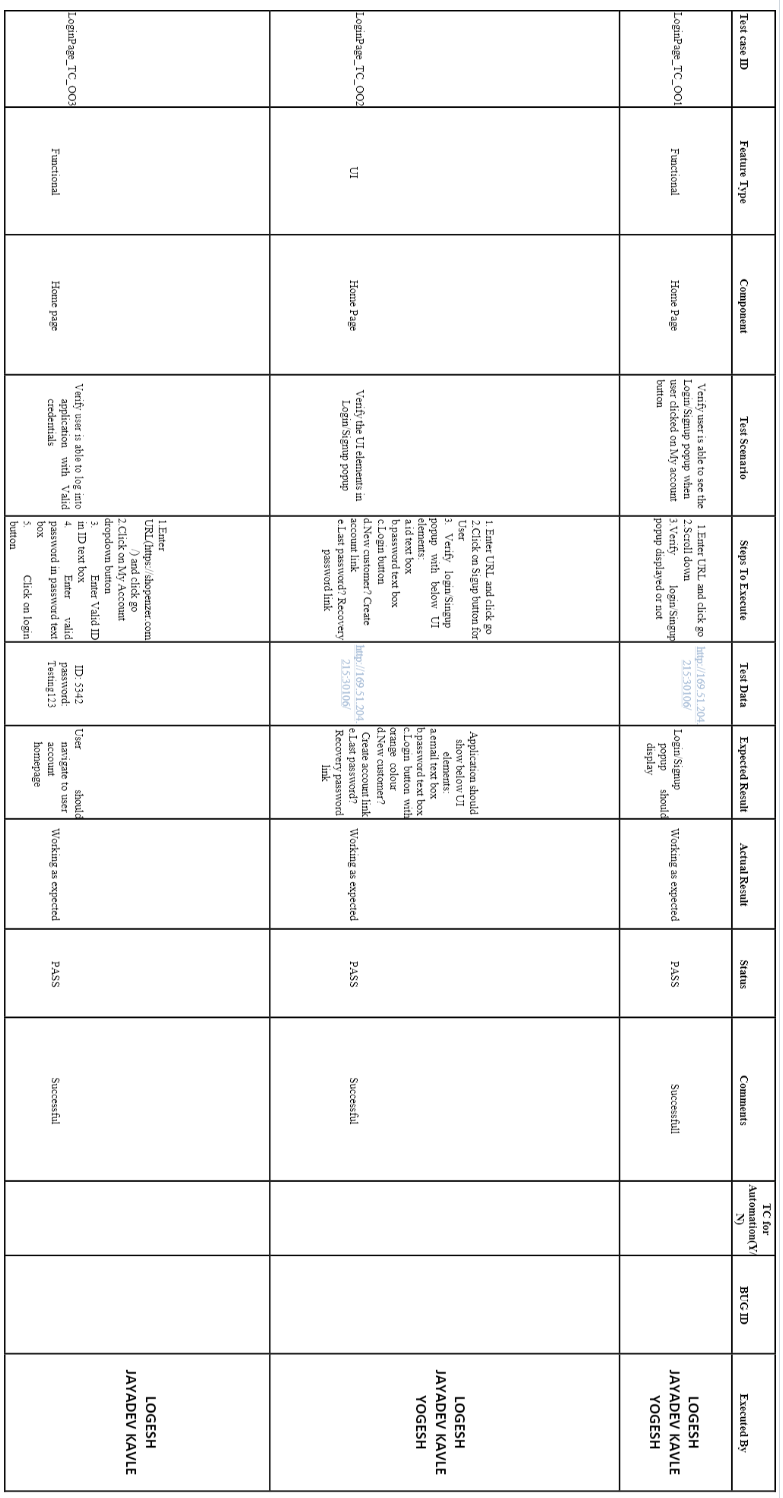
# main

if \_\_name\_\_ == "\_\_main\_\_":

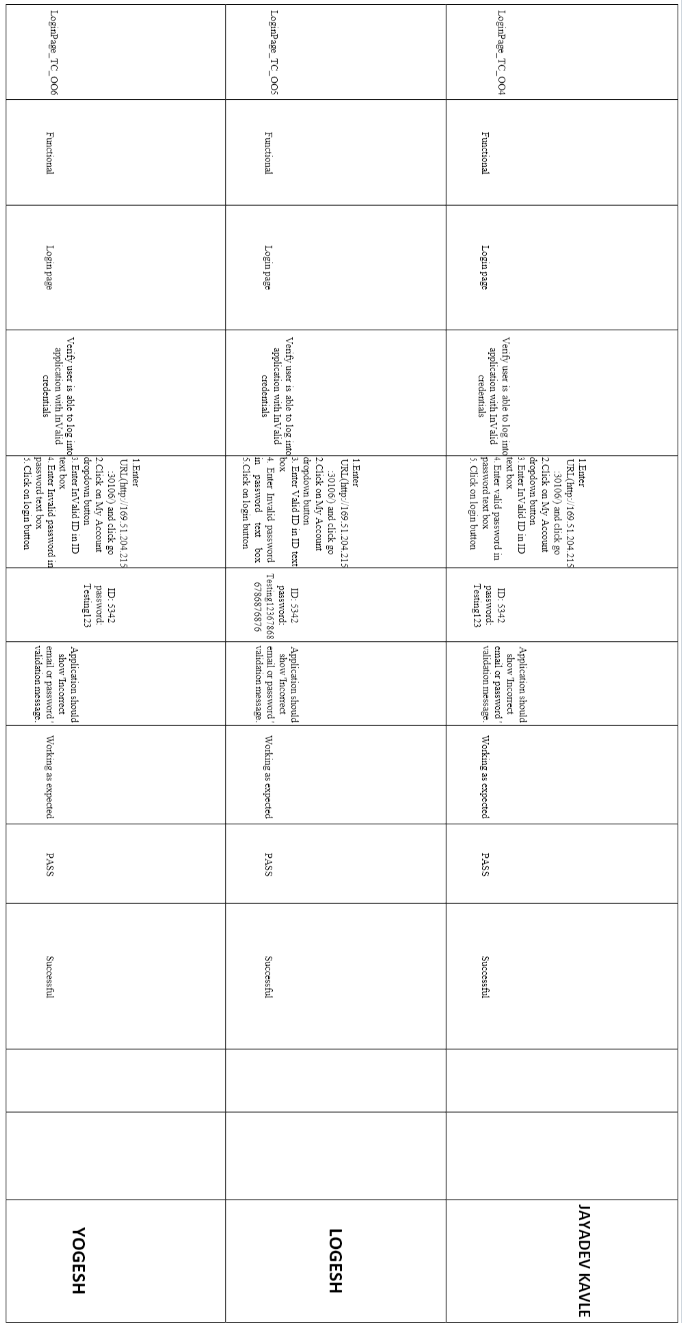
app.run(host='0.0.0.0', port=5000)

**8. TESTING**

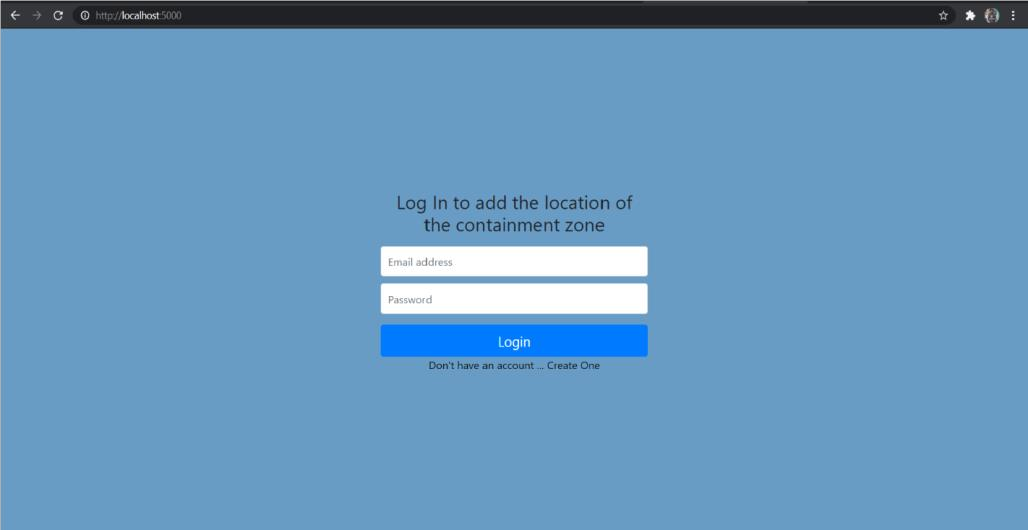
**8.1 Test Cases :**

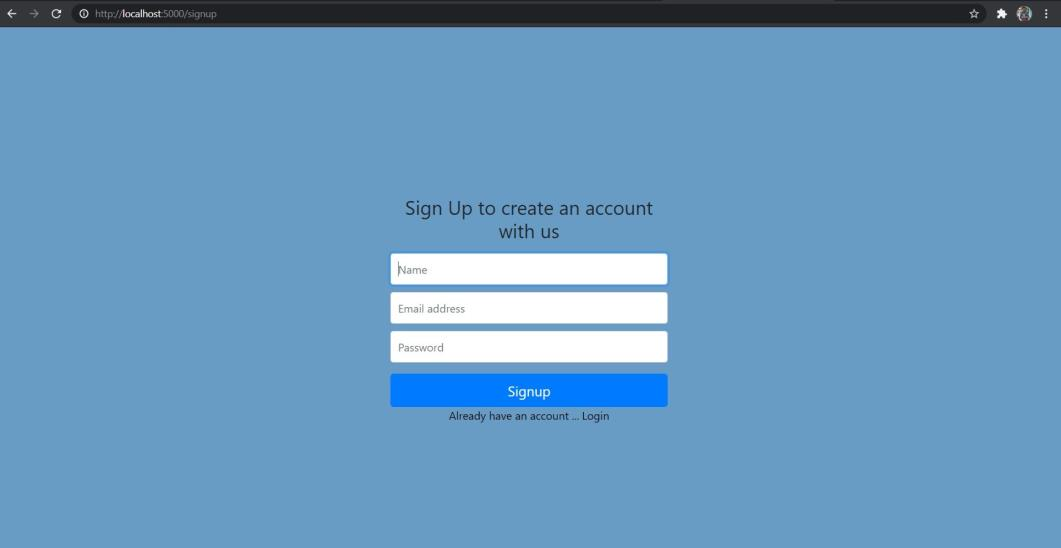
****

**8.2 User Acceptance Testing :**

****

**9.RESULT**

****

****

