CONTAINMENT ZONE ALERTING APPPLICATION
USING CLOUD

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING
IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

Team ID	: **PNT2022TMID32423**
Abinaya S	810019106002
Indira priyaa VR	810019106031
Kanmani R	810019106039
Kaviyasree L	810019106041

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING UNIVERSITY COLLEGE OF ENGINEERING, TIRUCHIRAPALLI-6200024

ABSTRACT

The World Health Organization has declared the outbreak of the novel coronavirus, Covid-19 as pandemic across the world. With its alarming surge of affected cases throughout the world, lockdown, and awareness (social distancing, use of masks etc.) among people are found to be the only means for restricting the community transmission. In a densely populated country like India, it is very difficult to prevent the community transmission even during lockdown without social awareness and precautionary measures taken by the people. Recently, several containment zones had been identified throughout the country and divided into red, orange and green zones, respectively. The red zones indicate the infection hotspots, orange zones denote some infection and green zones indicate an area with no infection. This paper mainly focuses on development of an Android application which can inform people of the Covid-19 containment zones and prevent trespassing into these zones. This Android application updates the locations of the areas in a Google map which are identified to be the containment zones. The application also notifies the users if they have entered a containment zone and uploads the user's IMEI number to the online database. To achieve all these functionalities, many tools, and APIs from Google are used in this application. Therefore, this application can be used as a tool for creating further social awareness about the arising need of precautionary measures to be taken by the people of India.

TABLE OF FIGURES

Containment zone Alerting Application

1.INTRODUCTION

This application helps the people to know about the containment zone in their area and stay away from it. It is a simple application which uses the GPS to find out the location of the user and tells the user about the containment zone in the area. Key benefits of the application are monitoring people's activity and alerting them to their safety movements.

1.1 Project overview

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. The location of the individual must be stored in the Database. Alerts are sent using the notification service.

1.2 Purpose

To alert the people while they are entering the containment zone. In case of any disease, that should not be spreader to other people.

2. LITERATURE SURVEY

Ideation phase

Literature Survey

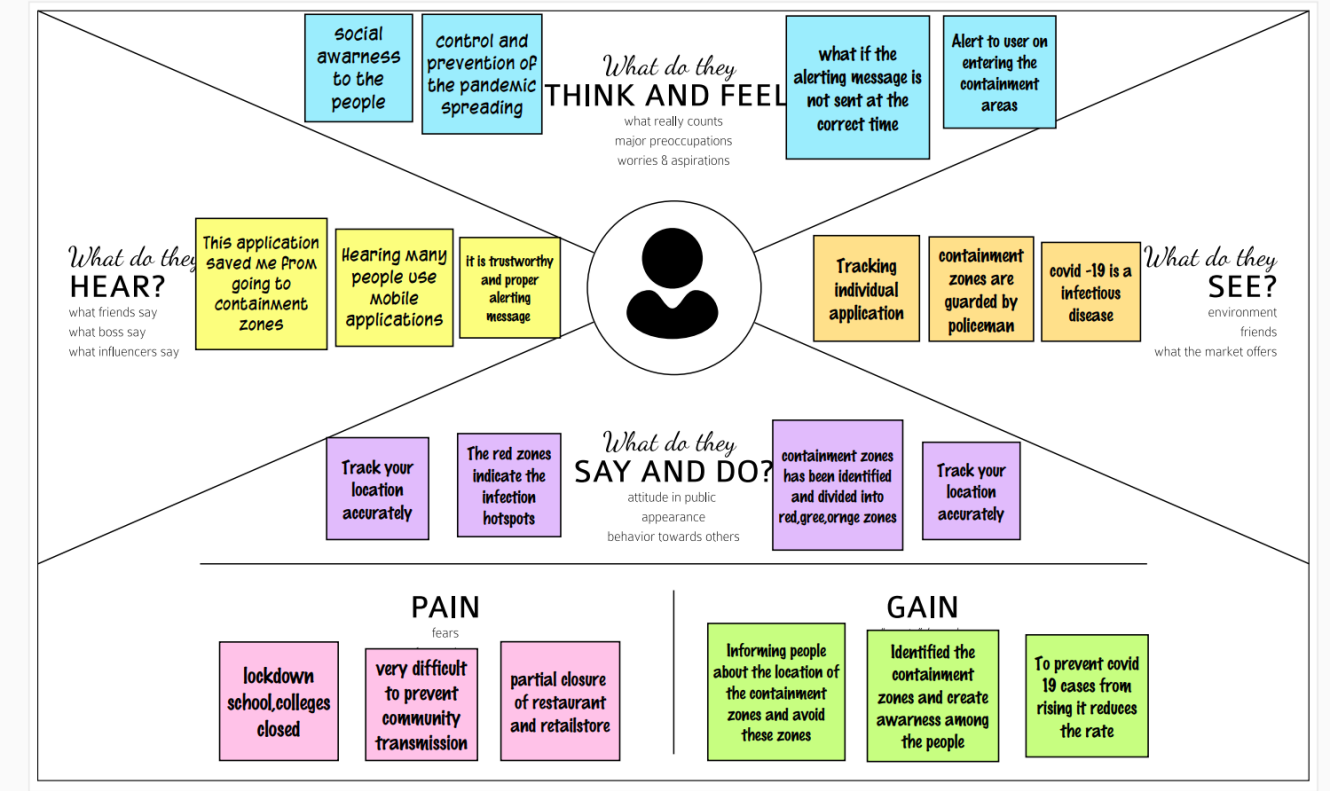
Containment Zone Alerting Application

Literature Survey:

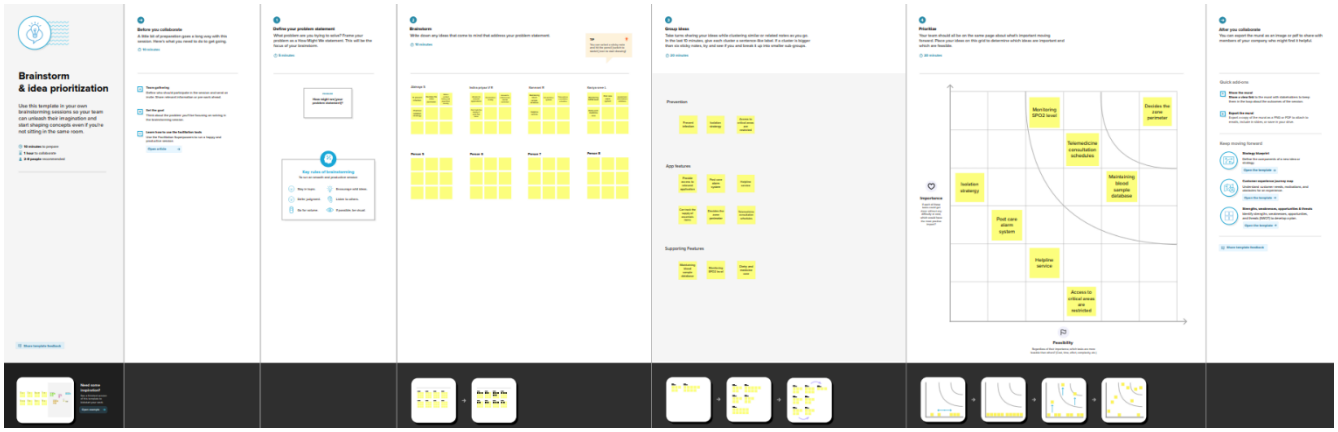
S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	DOMAIN	REMARKS
1	Development of an Android Application for containment Zones and monitoring violators who are trespassing into it using firebase and Geo-fencing	To develop a mobile based Application to provide information regarding the Covid-19 containment	Android SDK firebase Cloud fire-store.	Cloud Application	It provides an efficient way of showing the identified Covid-19 Zone
2	Applications of digital technology in COVID-19 pandemic planning and response(2020)	This Viewpoint provides a framework for the application of digital technologies in pandemic management and response.	Artificial intelligence; digital thermometers; mobile phone applications; thermal cameras; web-based toolkits. Advantages Allows visual depiction of spread; directs border restrictions; guides resource allocation; informs forecasts.	Cloud Application	Could breach privacy; involves high costs; requires management and regulation
3	Development of an Android Application for viewing Covid-19 containment Zones and Monitoring violators who and trespassing into It using firebase and Geo-fencing	This article is mostly about developing an Android app that informs individuals about Covid-19 Containment zones	It is based on Geo-fence and MC technologies	Cloud Application	It can be easily monitored & tracked It is not effective for people not having mobile phone
4	Development of an Android Application for viewing Covid-19 containment zones and monitoring violators who are trespassing into it using firebase and Geo-fencing	In this paper, We focus on developing a mobile based application to provide information regarding the Covid-19 containment zone	Fire base, & Geo-fencing API are used in this Application	Cloud Application	It tracker the user location & check whether it presented in list of containment zone

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Brainstorming



3.4 Problem Solution fit

Project Title: Containment zone alerting **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID32423

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) C

One who uses our product(app)is our customer

6. CUSTOMER CONSTRAINTS

- The user must have google account with email
- Location and mobile internet must be turned on while using the app

5. AVAILABLE SOLUTIONS

- The user is being aware of the nearby containment zones using this app

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS

- To alert the customer about the containment zones and to create awareness about it.
- Designing the application using web framework
- For data storage IBM DB is used

9. PROBLEM ROOT CAUSE RC

The customer may not be aware of the fluctuations in no.of.cases in a particular area.

They may not get an alert while entering affected area due to their carefree actions when they don't use this app

7. BEHAVIOUR BE

If the user faces any issues they can report them to the authority

When the user find this app helpful they can share it to the one who really needs this

Focus on J&P, tap into BE, understand RC

3. TRIGGERS

seeing people us using this app can create curiosity and urge to use this app for their ease of work

TR

10. YOUR SOLUTION

This app helps us to stay away from the particular infected/affected/alerted zone since we get alert immediately in our app

Alert email is also sent if the user visits the containment zone using send grid.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

User can login into the app and can know about current situations in the searched zones

User may also receive the alert for fluctuating cases as daily update if they wish so, with the help of notification.

8.2 OFFLINE

The customer can view the downloaded information and analyse them for their further reference and can also share them to the needed one.

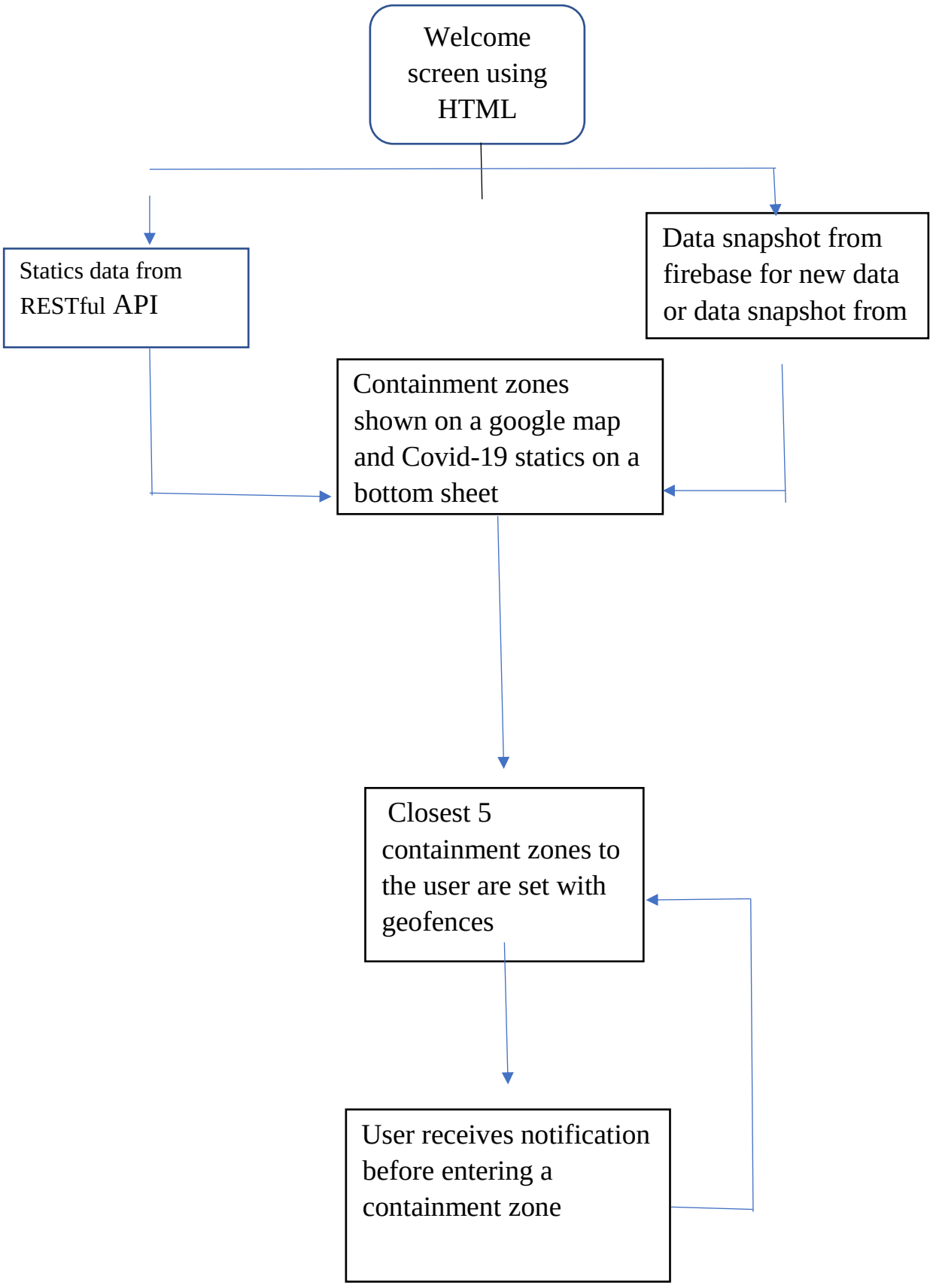
4. EMOTIONS: BEFORE / AFTER EM

Before using this app they are unaware of the risk on their way

After using this app they'll feel like they have a guide accompanying them in their journey.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



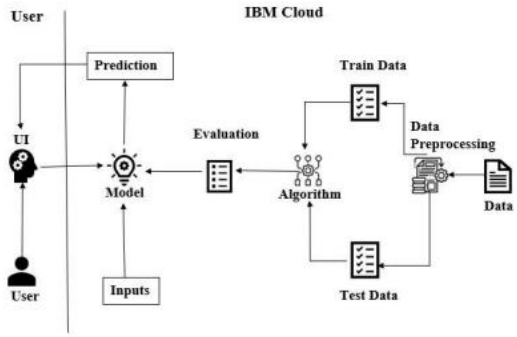
5.1 Solution Architecture

Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	15 October 2022
Team ID	PNT2022TMID32553
Project Name	Project – Smart lender applicant credibility prediction for loan approval
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User interact with our application through web User Interface.	HTML, CSS and Python flask.
2.	Application Logic-1-Login.	When the user click on the login button , he/she is directed to login page, if they are registered already.	HTML ,CSS, Python flask.
3.	Application Logic-Registration	When the user click on the Register button , he/she is directed to Register page for further process.	HTML,CSS, Python flask.
4.	Application Logic-Credibility details	After Logged in , when the user click on the credibility details form button,he/she directed to the form page to enter the details of applicant for prediction.	Front end- HTML ,CSS , MySQL, Pythonflask Back end-Python
5.	Database	Data type - String ,Numeric.	MySQL.
6.	Cloud Database	Database Service on Cloud	IBM.
7.	File Storage	File storage requirements	NIL
8.	External API-1	Purpose of External API used in the application	NIL
9.	External API-2	Purpose of External API used in the application	Aadhar API
10.	Machine Learning Model	Get the data from the user and predict the data with tested and trained dataset models	Data Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	NIL

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	International Business Machines.	Cloud.
2.	Security Implementations	Access permission for login page using CAPTCHA	Encryptions.
3.	Scalable Architecture	The key of Three tier architecture is improving scalability.	Three Tier architecture.
4.	Availability	Load balancer or ADC is the key component that ensures high availability by sending request.	Load balancer.
5.	Performance	The system should be able to handle large number of users at the time	Load balancer.

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	19 september 2022
Team ID	PNT2022TMID32423
Project Name	Project – Containment Zone Alerting Application
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	User: I can register for the application by entering my email, password and verifying password.	3	High	Kanmani .R
		USN-2	User: I will receive a confirmation email once I have registered for the application.	2	High	Abinaya .S
		USN-3	User: I can register for the application through Gmail.	5	Medium	Indirapriyaa .V.R
		USN-4	Management: I need to register my hospitals on the site.	2	High	Kaviyasree .L
		USN-5	User: I can log into the application by entering my email & password	3	High	Kanmani .R
	Login	USN-6	Management: I need to login into my dashboard with my given hospital id and password.	5	Medium	Abinaya .S
	Dashboard	USN-7	User: I need to give permission to access my Contacts, Location, and Storage	5	High	Indirapriyaa .V.R
Sprint-2		USN-8	User: I get access to the dashboard which shows a map with containment zones	5	High	Kaviyasree .L
		USN-9	Management: I need to enter the case information of the patient that visits our hospital.	5	High	Kanmani .R
	Services	USN-10	Admin: I need to provide valid information about the pandemic out there.	5	High	Abinaya .S
Sprint-3	Dashboard	USN-11	Management: I need to store all the patient information on the cloud.	5	High	Indirapriyaa .V.R
	Services	USN-12	Admin: I need to provide medical advice through a chatbot.	5	Medium	Kaviyasree .L
		USN-13	Admin: I need to provide medical recommendations by collaborating with top hospitals.	5	Low	Kanmani .R
		USN-14	Admin: I need to provide preventive measures when they travel through it.	5	High	Abinaya .S
	Registration	USN-15	User: I can register for the application through Facebook.	2	Low	Indirapriyaa .V.R

Sprint-4		USN-16	User: I can register for the application through Twitter.	2	Low	Kaviyasree .L
	Services	USN-17	Admin: I need to alert the user when they enter pandemic zones.	3	Medium	Kanmani .R
		USN-18	Admin: I need to provide special services for premium users by giving services like monitoring health by their smart bands.	3	Low	Abinaya .S
	Data Collection	USN-19	Admin: I need to store all the user information on the cloud	5	Medium	Indirapriyaa V.R
		USN-20	Admin: I need to collect the recent list of diseases in the world.	5	Low	kaviyasree .L

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 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

Velocity:
It will be updated after the first week of work is completed.

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

Burndown Chart:

It will be updated after the first week of work is completed.

---

7. CODING & SOLUTION

Project : CONTAINMENT ZONE ALERTING APPLICATION

Team ID : PNT2022TMID32423

APP.PY

```
from logging import error from flask import *
from jinja2.utils import select_autoescape import bcrypt
from flask_mysqldb import MySQL
import json
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
# initialization
app = Flask(__name__)
# config
app.secret_key =
"\x19Ts\xbe\xe7\x8c_\r\x12Q\x14\x13>q\xb7'WTH0\x9f\xe4\xec\xb1"
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'zone2'
mysql = MySQL(app)
# functions
def send_mail(email):
 print(email)
 message = Mail(from_email='varundutia.h@gmail.com',
to_emails=email,
```

```
subject='caution',
plain_text_content='Please Stay Safe',
html_content='<h2>You are entering into a containment Zone</h2>')
 try:
 sg = SendGridAPIClient(
'SG.7BJDtQDlS8unH0r5_TufVQ.Ykpcz19QcqgcNwYZC3a0mNRPhGksG117YURqOTa
2HL') response = sg.send(message)
print(response.status.code)
print(response.body)
print(response.headers)
except Exception as e:
print(e)
def create_bcrypt_hash(password): # convert the string to bytes
 password_bytes = password.encode()
 # generate a salt
salt = bcrypt.gensalt(14) # calculate a hash as bytes
 password_hash_bytes = bcrypt.hashpw(password_bytes, salt)
 # decode bytes to a string
 password_hash_str = password_hash_bytes.decode() return password_hash_str
def verify_password(password, hash_from_database):
 password_bytes = password.encode() hash_bytes =
hash_from_database.encode()
 # this will automatically retrieve the salt from the hash,
 # then combine it with the password (parameter 1) # and then hash that, and
compare it to the user's hash does_match = bcrypt.checkpw(password_bytes,
hash_bytes)
```

return does_match

Api's

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

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

PNT2022TMID48441

initialize the cursor

signup_cursor = mysql.connection.cursor()

check whether user already exists user_result = signup_cursor.execu

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)

def verify_password(password, hash_from_database):

password_bytes = password.encode()

hash_bytes = hash_from_database.encode()

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

```
return redirect(url_for("home"))
else:
return render_template('login.html', error=1)
else:
return render_template('login.html', error=2)
return render_template('login.html', error=3)
@app.route("/signup", methods=["POST", "GET"])
def create_bcrypt_hash(password):
# convert the string to bytes
password_bytes = password.encode()
# generate a salt
salt = bcrypt.gensalt(14)
# calculate a hash as bytes
password_hash_bytes = bcrypt.hashpw(password_bytes, salt)
# decode bytes to a string
password_hash_str = password_hash_bytes.decode()
return password_hash_st
# then combine it with the password (parameter 1)
# and then hash that, and compare it to the user's hash
does_match = bcrypt.checkpw(password_bytes, hash_bytes)
return does_match
# Api's
@app.route("/", methods=["GET", "POST"])
def login():
if(request.method == "POST"):
# get the data from the form
```

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

```
name = request.form['name']
email = request.form['email']
 password = request.form['password']
 # hash the password
 pw_hash = create_bcrypt_hash(password)
 # initialize the cursor
 signup_cursor = mysql.connection.cursor()
 # check whether user already exists
user_result = signup_cursor.execute(
 "SELECT * FROM USERS WHERE user_email=%s", [email]
 )
 if(user_result > 0):
 signup_cursor.close()
return render_template('signup.html', error=True)
else:
 # execute the query
signup_cursor.execute(
 'INSERT INTO USERS(user_name,user_email,user_password,user_type)
VALUES(%s,%s,%s,%s)', (
 name, email, str(pw_hash), "2"
 )
 )
 mysql.connection.commit()
signup_cursor.close()
 return redirect(url_for('login'))
 return render_template('signup.html', error=False)
```

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

```
name, email, str(pw_hash), "1"
)
)
mysql.connection.commit()
id_result = signup_cursor.execute(
'SELECT user_id FROM USERS WHERE user_email = %s', [email]
)
if(id_result > 0):
id = signup_cursor.fetchone()
return {"id": id[0]}
```

8.Result:

Containment zone Alerting application using cloud is developed and executed at the level of completed progress .

9.ADVANTAGES & DISADVANTAGES

Advantages:-

- By alerting people to areas where there is a risk of infection, they can avoid these areas and help to contain the spread of disease.
- By providing information on the location of the outbreak, authorities can more easily deploy resources to the affected area.
- This can help to contain the outbreak and prevent it from spreading further.
- Better user experience.

Disadvantages:-

- Difficult to keep track of all the different containment zones that have been set up. This can lead to confusion and frustration for users.
- These zones are hard to maintain and enforce.

10. CONCLUSION:

It can be concluded that the containment zone alerting application is feasible and can be developed. The application can be developed using the Android platform and can be deployed on the Google Play Store. The application can be used by the government authorities to alert the citizens about the containment zones in their area. The application can also be used by the citizens to check the containment zones in their area and to plan their travel

Project demo link

https://drive.google.com/file/d/1-CxNUsdxHLmTf8iNC0HxLMaGlGIBF-dO/view?usp=drivesdk