Project Development Phase

Sprint - 2

Team ID	PNT2022TMID35229
Project Name	CONTAINMENT ZONE ALERTING

Home.html

```
<!Doctype html>
<html>
<head>
<title>Dashboard</title>
<link rel="stylesheet" href="static/style.css">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Open+Sans&display=swap"
rel="stylesheet">
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.cs
s"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhO
f23Q9Ifjh" crossorigin="anonymous" />
<style>
body {
padding-top: 30px;
padding-bottom: 30px;
background-color: #0C1017;
color : #F0F6FC;
font-family: 'Open Sans', sans-serif;
padding: 30px;
}
.m-3 float-right {
background-color : #0C1017;
}
a {
color: #F0F6FC;
}
<!-- design the whole html page to beautify-->
```

```
</style>
</head>
<body>
{% if success %}
<script>
alert("Location Uploaded Successfully");
</script>
{% elif not success %}
<script>
alert("Enter Proper Location data");
</script>
{% endif %}
<div class="logout">
<button type="button" class="btn btn-primary"><a
href={{url_for("logout")}}>Log
Out</a></button>
</div>
<div class="logout">
<h1>Declare Containment Zone</h1>
</div>
<h2>Welcome: {{name}}</h2>
<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+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd
026JF"
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>
</form>
</body>
</html>
```

App.py

```
import os
import re
import ibm_db
import requests
from flask import *
from glances.globals import json_dumps
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
app = Flask(__name__, static_url_path='/static', static_folder='static',
template_folder='templates')
app.secret_key = 'sus'
conn = ibm_db.pconnect("DATABASE=BLUDB;"
"HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.a
ppdomain.cloud;"
"PORT=32733;"
"UID=dnp80914;PWD=5OtVWHTwkHw2GmU3;"
"PROTOCOL=TCPIP;"
```

```
"Security=SSL;"
"sslConnection=true;"
"SSLServerCertificate=DigiCertGlobalRootCA.crt;"
"", "", "")
# print("Connected to database", conn)
session = {}
# import sendgrid environment variables
api_key = os.environ.get('API_KEY')
api_key2 = os.environ.get('API2')
# route for sending email
def sendemail(mail):
message = Mail(
from_email='caliphshah@gmail.com',
to_emails=mail,
subject='Cautious Alert',
html_content='<h1>You are entering into contaminated zone!!</h1>'
'<p>Stay safe and take necessary precautions</p><br>'
'<p>Thank you</p><br>')
try:
sg = SendGridAPIClient(api_key)
response = sg.send(message)
print(response.status_code)
print(response.body)
print(response.headers)
except Exception as e:
print(e.message)
# create a sent email function using sendinblue
def send_conf_email(email):
url = "https://api.sendinblue.com/v3/smtp/email"
payload = {
"sender": {
"name": "Shafeeq Ur Rahman",
"email": "caliphshah@gmail.com"
},
"to": [
{
"email": email,
"name": 'Hey'
}
],
"subject": "Confirming Registration",
"htmlContent": "<h1>Thank you for registering with us</h1>"
"<p>Stay safe and take necessary precautions</p><br>"
"<p>Thank you</p><br>",
}
headers = {
```

```
'accept': "application/json",
'content-type': "application/json",
'api-key': api_key2
}
response = requests.request("POST", url, data=json.dumps(payload),
headers=headers)
print(response.text)
@app.route('/', methods=['GET', 'POST'])
def register():
message = ''
if request.method == 'POST':
# get the data from the form
name = request.form['username']
email = request.form['email']
password = request.form['password']
confirm_password = request.form['confirm_password']
# if nothing is entered in the form
if not name or not email or not password or not confirm_password:
message = 'Please fill all the fields!'
return render_template('register.html', message=message)
# if the password and confirm password do not match
elif password != confirm_password:
message = 'Passwords do not match!'
return render_template('register.html', message=message)
# password length must be 8 or above
if len(password) < 8:
message = 'Password must be 8 or more characters'
return render_template('register.html', message=message)
# check if the email is valid
if re.match(r"[^@]+@[^@]+\.[^@]+", email):
# insert the data into the database
# check if email already exists in the database
sql = "SELECT * FROM users WHERE email = '" + email + "'"
stmt = ibm_db.exec_immediate(conn, sql)
# print("stmt", stmt)
result = ibm_db.fetch_assoc(stmt)
# print("result", result)
if result:
message = 'The username or email already exists!'
else:
sql = "INSERT INTO users (id, username, email, password,type) VALUES
(seq_person.nextval,'" + name + \
"', '" + email + "', '" + password + "', 1) "
ibm_db.exec_immediate(conn, sql)
# send confirmation email
send_conf_email(email)
```

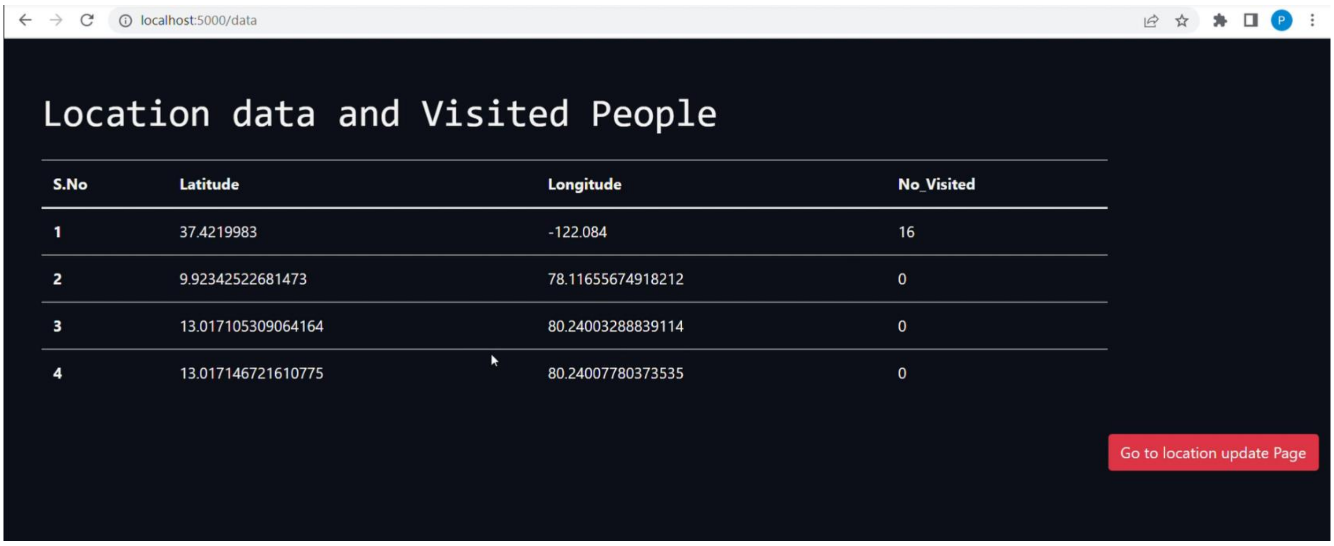
```
return redirect(url_for('login'))
else:
message = 'The email is invalid!'
return render_template('register.html', message=message)
@app.route('/login', methods=['GET', 'POST'])
def login():
message = ''
if request.method == 'POST':
# get the data from the form
email = request.form['email']
password = request.form['password']
# if nothing is entered in the form
if not email or not password:
message = 'Please fill all the fields!'
return render_template('login.html', message=message)
# check if the username and password are valid
sql = "SELECT * FROM users WHERE email = '" + email + "' AND password =
'" +
password + "'"
stmt = ibm_db.exec_immediate(conn, sql)
result = ibm_db.fetch_assoc(stmt)
# print("result", result)
if result:
# message = 'You have successfully logged in!'
session['id'] = result['ID']
session['username'] = result['USERNAME']
session['email'] = result['EMAIL']
# print("id ==", session['id'])
return redirect(url_for('home'))
else:
message = 'The email or password is incorrect!'
return render_template('login.html', message=message)
@app.route('/logout')
def logout():
session.clear()
return redirect(url_for('login'))
# create a route for the home page and open only if the user is logged in
@app.route('/home', methods=['GET', 'POST'])
def home():
# print(name)
if 'id' in session:
if request.method == 'GET':
return render_template('home.html', name=session['username'])
if request.method == "POST":
# get data
lat = request.form["lat"]
```

```
lon = request.form["lon"]
if lat == "" or lon == "":
return render_template('home.html', name=session['username'],
email=session['email'], id=session['id'],
success=0)
# create a query to insert the data into the database
sql = "INSERT INTO inf_location (locate_id, locate_lat, locate_lang, visited)
VALUES
(seq_loc.nextval,'" \
+ lat + "', '" + lon + "', 0)"
# execute the query
ibm_db.exec_immediate(conn, sql)
return render_template('home.html', name=session['username'],
email=session['email'], id=session['id'],
success=1)
return render_template('home.html', success=0)
else:
return redirect(url_for('login'))
# create a route for the data page and open only if the user is logged in
@app.route('/data')
def data():
if 'id' not in session:
return redirect(url_for('login'))
else:
# create a query to fetch the data from the database
sql = "SELECT * FROM inf_location"
stmt = ibm_db.exec_immediate(conn, sql)
# print("stmt", stmt)
# fetch all the data from the database and store it in the result dictionary
result = ibm_db.fetch_assoc(stmt)
# create a list to store the data
data = []
# loop through the result dictionary and append the data to the list
while result:
data.append(result)
result = ibm_db.fetch_assoc(stmt)
# print(data)
return render_template('data.html', data=data)
# android signup api
@app.route('/android_signup', methods=['POST'])
def android_signup():
if request.method == 'POST':
# get the data from the form
name = request.json['name']
email = request.json['email']
password = request.json['password']
```

```
# if nothing is entered in the form
# check if the email is valid
if re.match(r"[^@]+@[^@]+\.[^@]+", email):
# insert the data into the database
# check if email already exists in the database
sql = "SELECT * FROM users WHERE email = '" + email + "'"
stmt = ibm_db.exec_immediate(conn, sql)
# print("stmt", stmt)
result = ibm_db.fetch_assoc(stmt)
# print("result", result)
if result:
return jsonify({"message": "The username or email already exists!"})
else:
sql = "INSERT INTO users (id, username, email, password,type) VALUES
(seq_person.nextval,'" + name + \
"', '" + email + "', '" + password + "', 2) "
ibm_db.exec_immediate(conn, sql)
# pass the id of the user to the android app
sql = "SELECT * FROM users WHERE email = '" + email + "' AND password =
'" +
password + "'"
stmt = ibm_db.exec_immediate(conn, sql)
result = ibm_db.fetch_assoc(stmt)
return {"status": "success", "message": "You have successfully registered!",
"id":
result['ID']}
else:
return jsonify({'message': 'The email is invalid!'})
return jsonify({'message': 'The email is invalid!'})
# android get all users
# @app.route('/get_all_users', methods=['GET'])
# def get_all_users():
# # create a query to fetch the data from the database
# sql = "SELECT * FROM users"
# stmt = ibm_db.exec_immediate(conn, sql)
# # print("stmt", stmt)
# # fetch all the data from the database and store it in the result dictionary
# result = ibm_db.fetch_assoc(stmt)
#
# # create a list to store the data
# data = []
# # loop through the result dictionary and append the data to the list
# while result:
# data.append(result)
# result = ibm_db.fetch_assoc(stmt)
# # print(data)
# return {'data': data}
@app.route("/post_user_location_data", methods=["POST"])
def post_user_location_data():
# get data
lat = request.json["lat"]
lon = request.json["long"]
id1 = request.json["id"]
ts = request.json['timestamp']
# create a query to insert the data into the database
sql = "INSERT INTO location (LOCATE_LAT, LOCATE_LONG, USER_ID,
TIME_STAMP)
VALUES ('" + lat + "', '" + lon + "', '" + str(
id1) + "', '" + ts + "')"
# execute the query
ibm_db.exec_immediate(conn, sql)
return {"status": "success", "message": "You have successfully registered!"}
@app.route("/location_data")
def location_data():
# create a query to fetch the data from the database
sql = "SELECT * FROM inf_location"
stmt = ibm_db.exec_immediate(conn, sql)
# print("stmt", stmt)
# fetch all the data from the database and store it in the result dictionary
result = ibm_db.fetch_assoc(stmt)
# create a list to store the data
data = []
# loop through the result dictionary and append the data to the list
while result:
data.append(result)
ibm_db.fetch_assoc(stmt)
# print(data)
return json.dumps(data)
else:
return {"response": "failure"}
@app.route("/get_all_users")
def get_users():
# create a query to fetch the data from the database
sql = "SELECT * FROM users"
stmt = ibm_db.exec_immediate(conn, sql)
# print("stmt", stmt)
# fetch all the data from the database and store it in the result dictionary
result = ibm_db.fetch_assoc(stmt)
if result:
# create a list to store the data
data = []
# loop through the result dictionary and append the data to the list
```

```
while result:
data.append(result)
result = ibm_db.fetch_assoc(stmt)
# print(data)
return json_dumps(data)
# 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("/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']
# print("email and loc", email, location_id)
# get location data
sql = "SELECT VISITED FROM INF_LOCATION WHERE LOCATE_ID = '" +
str(location_id) + "'"
stmt = ibm_db.exec_immediate(conn, sql)
print("stmt", stmt)
if stmt:
result = ibm_db.fetch_assoc(stmt)
if result:
visited = result['VISITED']
visited = visited + 1
sql = "UPDATE INF_LOCATION SET VISITED = '" + str(visited) + "' WHERE
LOCATE_ID = '" + str(
location_id) + "'"
ibm_db.exec_immediate(conn, sql)
ibm_db.exec_immediate(conn, sql)
# send email
# print("email ->", email)
sendemail(email)
return {"response": "Mail success"}
else:
return {"response": "Mail failed"}
if __name__ == '__main__':
app.run(debug=True, host='0.0.0.0', port=5000)
```

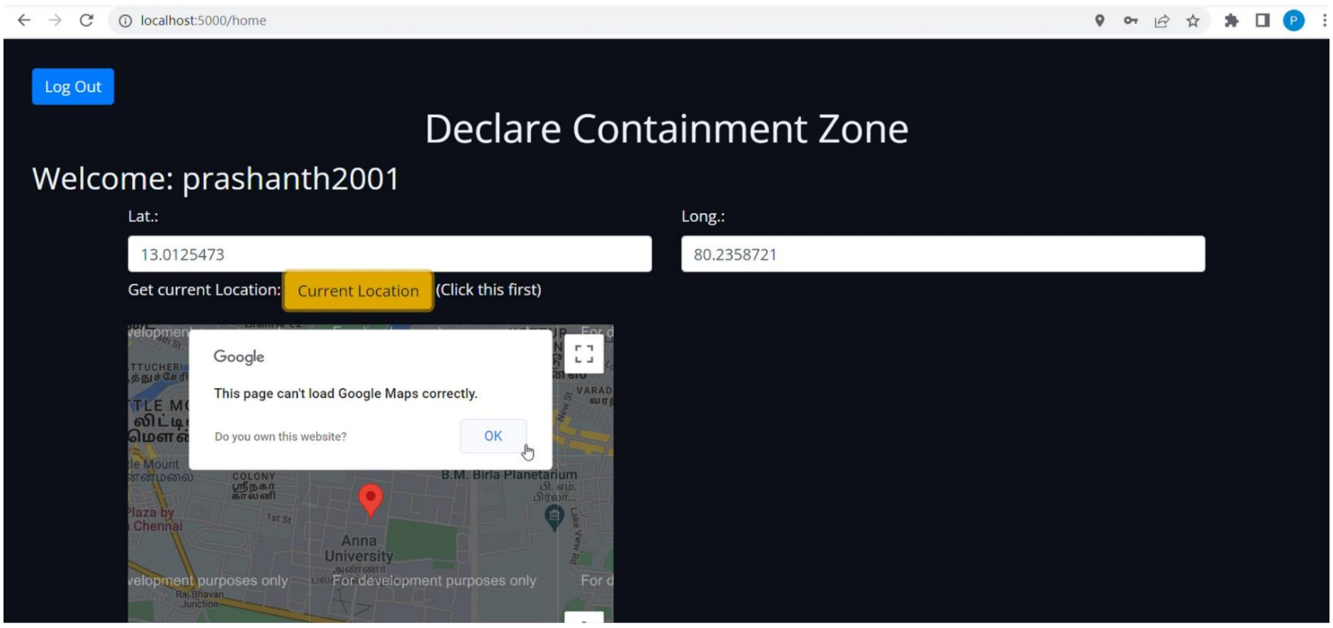
Screenshots



localhost:5000/data

Location data and Visited People

S.No	Latitude	Longitude	No_Visited
1	37.4219983	-122.084	16
2	9.92342522681473	78.11655674918212	0
3	13.017105309064164	80.24003288839114	0
4	13.017146721610775	80.24007780373535	0



localhost:5000/home

Declare Containment Zone

Welcome: prashanth2001

Lat.: 13.0125473

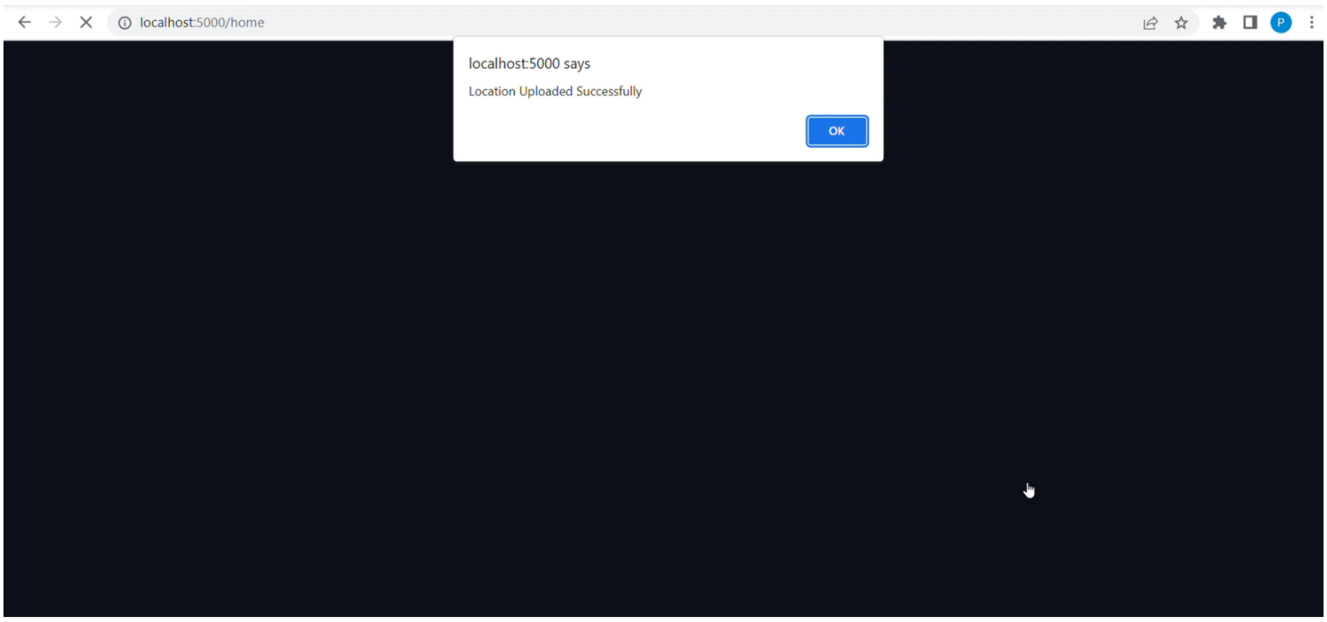
Long.: 80.2358721

Get current Location: Current Location (Click this first)

Google

This page can't load Google Maps correctly.

Do you own this website?



localhost:5000 says

Location Uploaded Successfully