Integrating SendGrid Service

Building APK file for the Project

Team ID	PNT2022TMID35281
Project	Containment Zone Alerting Application

```
import os
import re
import traceback
import ibm_db
from flask import Flask, jsonify, render_template, request, session
from flask_mail import Mail, Message
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *
from asyncio.windows_events import NULL

app = Flask(__name__)

hostname =
"b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.ap
pdomain.cloud"
uid = "zhw98184"
pwd = "n5oD3WsOV1p53cf1"
database = "bludb"
driver = "{IBM DB2 ODBC DRIVER}"
port = "32716"
protocol = "TCPIP"
security = "SSL"
certificate = "DigiCertGlobalRootCA.crt"

def dbconnect():
    creds = (
            "HOSTNAME = {0};"
            "UID = {1};"
            "PWD = {2};"
            "DATABASE = {3};"
            "PORT = {4};"
            "PROTOCOL = {5};"
            "SECURITY = {6};"
            "SSLServerCertificate = {7};"
```

```
            "DRIVER = {8};"

).format(hostname,uid,pwd,database,port,protocol,security,certificate,d
river)

    try:
        conn = ibm_db.connect(creds,"","")
        print("Connected to DB2")
        return conn

    except:
        traceback.print_exc()
        print("Unable to connect to DB2")
        return NULL


@app.route("/")
@app.route("/login")
def login():
    return render_template("index.html")


@app.route("/home", methods=["GET", "POST"])
def home():
    global userid
    msg = ""

    if request.method == "POST":
        username = request.form["UserName"]
        password = request.form["Password"]
        sql = "SELECT * FROM ADMIN WHERE Name = ? AND Password = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session["loggedin"] = True
            session["id"] = account["NAME"]
            userid = account["NAME"]
            session["UserName"] = account["NAME"]
            msg = "Logged in Successfully!"
```

```
            return render_template("home.html", user=username)
        else:
            msg = "Incorrect UserName or Password!"
            return render_template("index.html", msg=msg)


@app.route("/logout")
def logout():
    session.pop("Loggedin", None)
    session.pop("id", None)
    session.pop("UserName", None)
    return render_template("index.htmL")


# User Routes
@app.route("/user")
def user():
    sql = "SELECT * FROM USER"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    userList = []
    while ibm_db.fetch_row(stmt) != False:
        users = {}
        users["UserName"] = ibm_db.result(stmt, 0)
        users["EmailID"] = ibm_db.result(stmt, 1)
        users["PhoneNumber"] = ibm_db.result(stmt, 2)
        userList.append(users)
    return render_template("user.html", users=userList)


@app.route("/new")
def new():
    return render_template("addUser.html")


@app.route("/user/new")
def newUser():
    if request.method == "POST":
        username = request.form["UserName"]
        email = request.form["EmailAddress"]
        phonenumber = request.form["PhoneNumber"]
        password = request.form["Password"]
        sql = "SELECT * FROM USER WHERE Name = ?"
```

```
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = "Account already exists!"
        elif not re.match(r"[^]", email):
            msg = "Invalid email address"
        elif not re.match(r"[A-Za-z0-9]+", username):
            msg = "Name must contain characters and numbers"
        else:
            insert_sql = "INSERT into USER values (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phonenumber)
            ibm_db.bind_param(prep_stmt, 4, password)
            ibm_db.execute(prep_stmt)
            msg = "You have successfully registered"
            return render_template("addUser.html", msg=msg)
    elif request.method == "POST":
        msg = "Please fill out the form"
        return render_template("addUser.html", msg=msg)


# Zone Routes

@app.route("/zones")
def zones():
    return render_template("zones.html")


@app.route("/zones/add")
def zoneAddPage():
    return render_template("addZone.html")


@app.route("/zones/new", methods=["POST"])
def zoneAdd():
    if request.method == "POST":
        zid = request.form["ZoneID"]
        latitude = request.form["Latitude"]
```

```
        longitude = request.form["Longitude"]
        zoneName = request.form["ZoneName"]
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        print(zone)
        if zone:
            msg = "Zone already exists!"
        else:
            insert_sql = "INSERT INTO ZONES VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, zid)
            ibm_db.bind_param(prep_stmt, 2, latitude)
            ibm_db.bind_param(prep_stmt, 3, longitude)
            ibm_db.bind_param(prep_stmt, 4, zoneName)
            ibm_db.execute(prep_stmt)
            msg = "You have successfully added"
        return render_template("addZone.html", msg=msg)
    elif request.method == "POST":
        msg = "Please fill out the form"
        return render_template("addZone.html", msg=msg)


@app.route("/zones/update")
def zoneUpdatePage():
    return render_template("updateZone.html")


@app.route("/zones/alter", methods=["POST"])
def zoneAlter():
    if request.method == "POST":
        zid = request.form["ZoneID"]
        latitude = request.form["Latitude"]
        longitude = request.form["Longitude"]
        zoneName = request.form["ZoneName"]
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        print(zone)
```

```
        if zone:
            update_sql = "UPDATE ZONES SET ZID = ?, Latitude = ?,
Longitude = ?, Name = ? WHERE ZID = ?"
            prep_stmt = ibm_db.prepare(conn, update_sql)
            ibm_db.bind_param(prep_stmt, 1, zid)
            ibm_db.bind_param(prep_stmt, 2, latitude)
            ibm_db.bind_param(prep_stmt, 3, longitude)
            ibm_db.bind_param(prep_stmt, 4, zoneName)
            ibm_db.bind_param(prep_stmt, 5, zid)
            ibm_db.execute(prep_stmt)
            msg = "You have successfully added"
        else:
            msg = "Zone not exists!"
        return render_template("updateZone.html", msg=msg)
    elif request.method == "POST":
        msg = "Please fill out the form"
        return render_template("updateZone.html", msg=msg)


@app.route("/zones/display")
def zoneDisplay():
    sql = "SELECT * FROM ZONES"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    zoneList = []
    while ibm_db.fetch_row(stmt) != False:
        zones = {}
        zones["ZID"] = ibm_db.result(stmt, 0)
        zones["Latitude"] = ibm_db.result(stmt, 1)
        zones["Longitude"] = ibm_db.result(stmt, 2)
        zones["Name"] = ibm_db.result(stmt, 3)
        zoneList.append(zones)
    return render_template("displayZone.html", zones=zoneList)


@app.route("/zones/delete")
def zoneDeletePage():
    return render_template("deleteZone.html")


@app.route("/zones/remove", methods=["POST"])
def removeZone():
    msg = ""
```

```
    if request.method == "POST":
        zid = request.form["ZoneID"]
        sql = "SELECT * FROM ZONES WHERE ZID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, zid)
        ibm_db.execute(stmt)
        zone = ibm_db.fetch_assoc(stmt)
        if zone:
            delete_query = "DELETE FROM ZONES WHERE ZID = ?"
            prep_stmt = ibm_db.prepare(conn, delete_query)
            ibm_db.bind_param(prep_stmt, 1, zid)
            ibm_db.execute(prep_stmt)
            msg = "You have successfully deleted."
        else:
            msg = "Sorry! Deletion Failed, Zone not exists."
        return render_template("deleteZone.html", msg=msg)
    elif request.method == "POST":
        msg = "Please fill out the form"
        return render_template("deleteZone.html", msg=msg)


# APIs for User App

@app.route("/location")
def location():
    sql = "SELECT * FROM ZONES"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    zoneList = []
    while ibm_db.fetch_row(stmt) != False:
        zones = {}
        zones["ZID"] = ibm_db.result(stmt, 0)
        zones["Latitude"] = ibm_db.result(stmt, 1)
        zones["Longitude"] = ibm_db.result(stmt, 2)
        zones["Name"] = ibm_db.result(stmt, 3)
        zoneList.append(zones)
    return jsonify(value=zoneList)


# SendGrid Integration

@app.route("/alert/<name>")
def alert(name):
```

```
    sql = "SELECT EmailID FROM USER WHERE Name = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, name)
    ibm_db.execute(stmt)
    email = ibm_db.fetch_assoc(stmt)
    print(email)
    message = Mail(
        # How to configure this mail?
        from_email="???",
        to_emails=email,
        subject="Alert!!!!!!!!!!!!!!!!!!!!!!",
        html_content="SendGridMail.html",
    )
    try:
        sg = SendGridAPIClient(os.environ.get("SENDGRID_API_KEY"))
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
    return "Success"


if __name__ == '__main__':
    conn = dbconnect()
    app.run(debug=True)
```