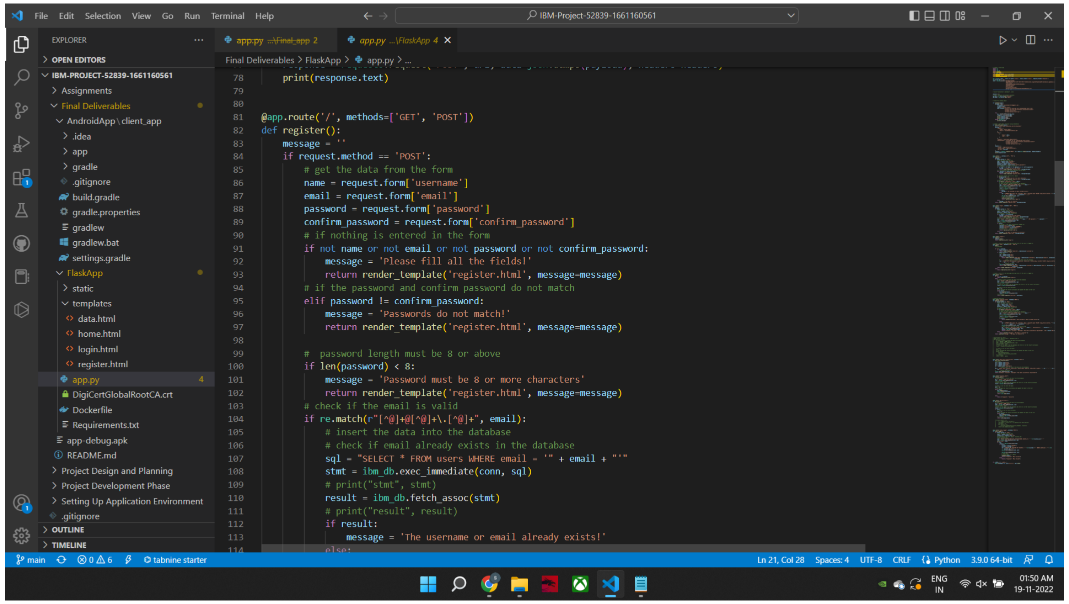
Creating API'S in Flask

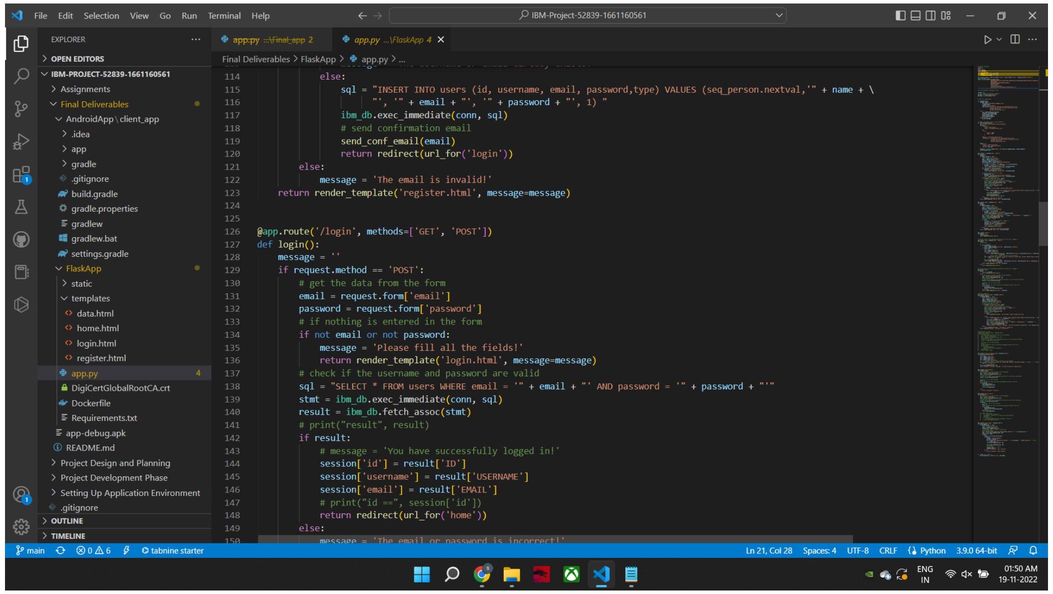
Team ID	PNT2022TMID13325
Project Name	Containment zone alerting application

CREATING API'S IN FLASK

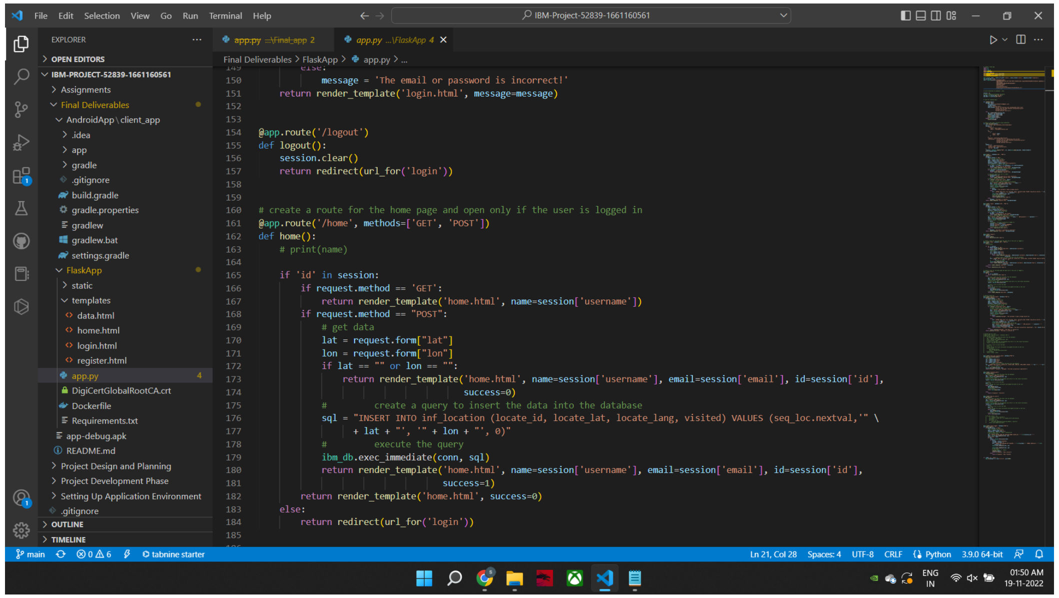
Fig: This following code represent we can create an API's in flask successfully



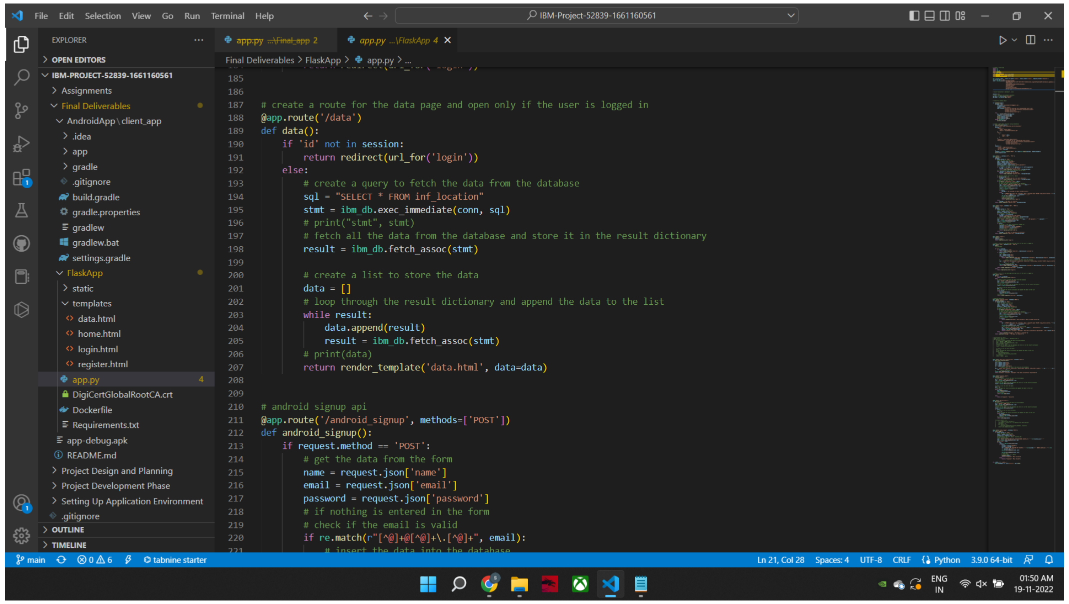
```python
    print(response.text)


@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:
```



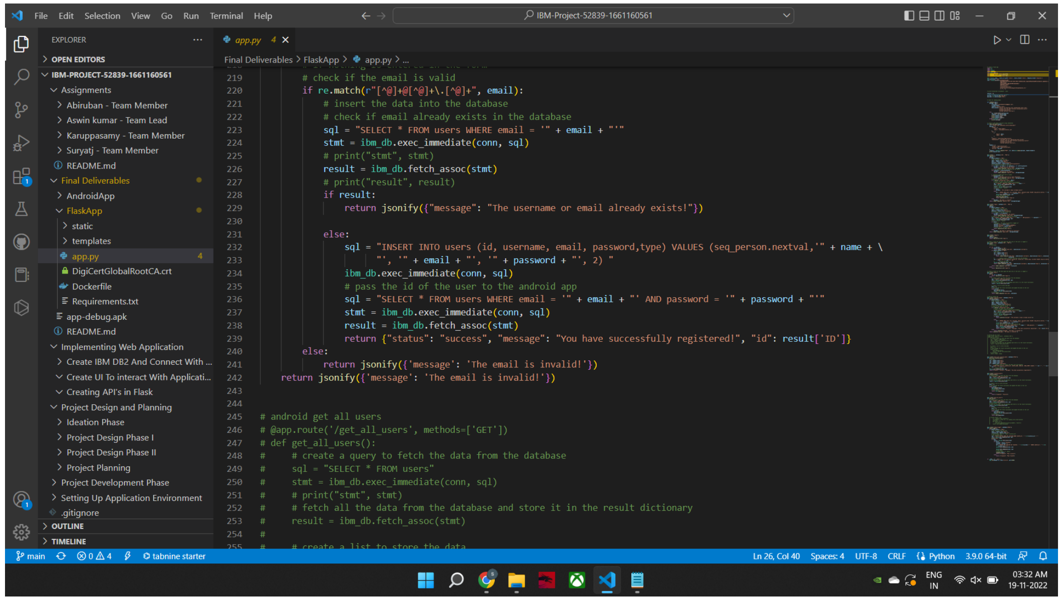
```python
            else:
                sql = "INSERT INTO users (id, username, email, password,type) VALUES (seq_person.nextval,'" + name + \
                      "', '" + email + "', '" + password + "', 1) "
                ibm_db.exec_immediate(conn, sql)
                # send confirmation email
                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!'
```



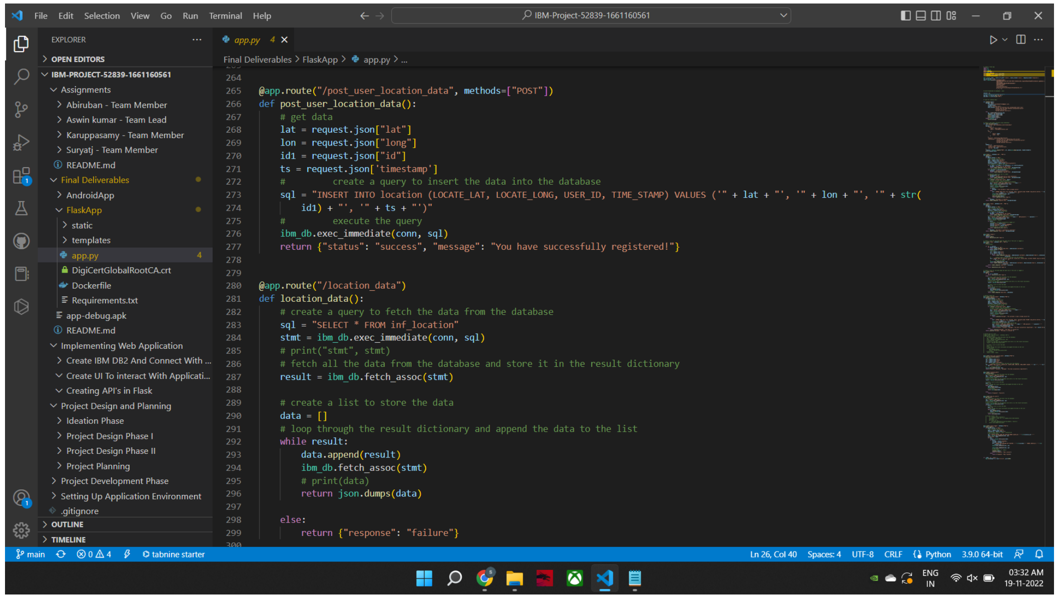
```python
            message = 'The email or password is incorrect!'
    return render_template('login.html', message=message)


@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'))

```



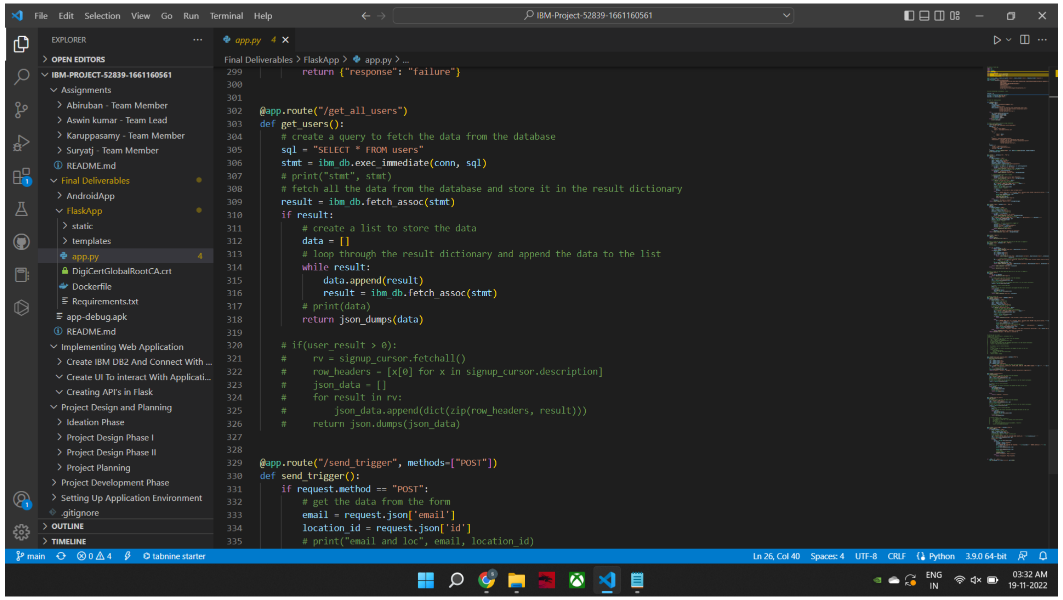
```python


# 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
```


File Edit Selection View Go Run Terminal Help
IBM-Project-52839-1661160561
EXPLORER
OPEN EDITORS
IBM-PROJECT-52839-1661160561
Assignments
Abiruban - Team Member
Aswin kumar - Team Lead
Karuppasamy - Team Member
Suryatj - Team Member
README.md
Final Deliverables
AndroidApp
FlaskApp
static
templates
app.py
DigiCertGlobalRootCA.crt
Dockerfile
Requirements.txt
app-debug.apk
README.md
Implementing Web Application
Create IBM DB2 And Connect With ...
Create UI To interact With Applicati...
Creating API's in Flask
Project Design and Planning
Ideation Phase
Project Design Phase I
Project Design Phase II
Project Planning
Project Development Phase
Setting Up Application Environment
.gitignore
OUTLINE
TIMELINE
app.py 4
Final Deliverables > FlaskApp > app.py > ...
main
tabnine starter
Ln 26, Col 40 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit
ENG IN
03:32 AM
19-11-2022

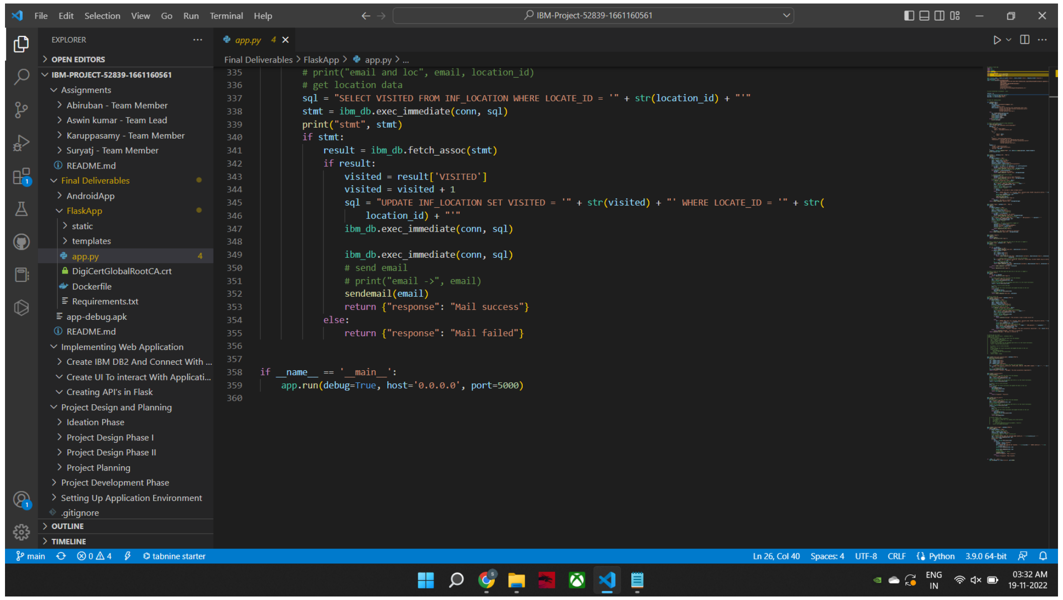
```python
        # check if the email is valid
        if re.match(r"[^@]+@[^@]+\.[^@]+", email):
            # insert the data into the database
            # check if email already exists in the database
            sql = "SELECT * FROM users WHERE email = '" + email + "'"
            stmt = ibm_db.exec_immediate(conn, 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
```



```python
@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"}
```



```python
            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)
```



```python
            # 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)

```