Content

Abstract:

This project demonstrates how to track the location of a phone number using Python, providing both the location information and a visual representation on a World map.

Introduction:

The project aims to track the location of a phone number using Python code. By leveraging the **`phonenumbers`** library, the code parses the provided phone number and extracts relevant information such as the location and service provider. The **`geocoder`** module further enhances the functionality by providing a descriptive location for the parsed phone number.

To begin, the code utilizes the **`phonenumbers`** library to parse the phone number input. Through the **`geocoder`** module, the code retrieves the location associated with the phone number and prints it as output. Additionally, the project determines the service provider linked to the phone number and displays the name.

Moreover, the code leverages the **`OpenCageGeocode`** API to obtain detailed **geolocation** information based on the parsed location. This includes latitude and longitude coordinates, which are printed as output.

The **`folium`** library plays a crucial role in visualizing the obtained location. It generates an interactive map with a marker placed at the identified coordinates.
The resulting map is saved as an HTML file, allowing for easy access and sharing.

Overall, this project showcases the power of Python in tracking locations based on phone numbers. By combining various libraries and APIs, it offers a comprehensive solution for extracting**, geocoding**, and visualizing location information, providing a valuable tool for location tracking applications.

Methodology Design :-

1. Importing Required Libraries:

- Import the necessary libraries, including `**phonenumbers**`, `folium`, and `**OpenCageGeocode**`. These libraries are used for phone number manipulation, map visualization, and **geocoding**.

2. Collect Phone Number:

- Obtain the phone number that you want to track. This can be done by assigning the phone number to a variable.

3. Parsing Phone Number:

- Use the **`phonenumbers.parse()`** function to parse the phone number and extract its components. This step ensures that the phone number is in a valid format.

4. Retrieving Location:

- Utilize the **`geocoder.description_for_number()`** function from the **`phonenumbers`** library to retrieve the location associated with the phone number. Specify the language for the location information, such as "en" for English.

5. Retrieving Service Provider:

- Use **the `carrier.name_for_number()`** function from the **`phonenumbers`** library to determine the service provider associated with the phone number. Specify the language for the service provider information.

6. Geocoding the Location:

- Initialize the **`OpenCageGeocode`** geocoder with the API key obtained from the **OpenCage Geocoding** service. This service will be used to geocode the location information.

7. Querying Geocoder:

- Convert the obtained location information to a string and use it as a query to the geocoder using the **`geocoder.geocode()`** function. This retrieves the latitude and longitude coordinates of the location.

8. Extracting Coordinates:

- Extract the latitude and longitude coordinates from the geocoder result. These coordinates will be used to pinpoint the location on the map.

9. Creating Map:

- Use **`folium.Map()`** to create a map object, specifying the center location using the latitude and longitude coordinates. Adjust the zoom level according to the desired map display.

10. Adding Marker:

- Use **`folium.Marker()`** to add a marker on the map at the specified latitude and longitude coordinates. Attach the location information as a popup to the marker using the `popup` parameter.

11. Saving the Map:

Save the map as an HTML file using the **`myMap.save()`** method. Provide a filename for the HTML file, such as "myLocation.html".

12. Execution and Result:

- Run the Python script to execute the code. The output will display the location information and service provider in the console. Additionally, a map will be generated and saved as an HTML file, which can be opened in a web browser to visualize the location.

By following this methodology, you can track the location of a phone number using Python, retrieve the associated information, and visualize the location on a map.

Proposed System:

Objective: Develop a program to retrieve the location and service provider information of a phone number and display it on a map.

1. Import necessary libraries:

- phonenumbers: For parsing and processing phone numbers.
- folium: For creating interactive maps.
- opencage.geocoder: For geocoding the location.

2. Define the phone number:

- Assign the phone number you want to track to the variable "num".

3. Retrieve the location information:

- Parse the phone number using the phonenumbers library.
- Use the geocoder module to get the location description for the parsed phone number.
- Print the location information.

4. Retrieve the service provider information:

- Parse the phone number using the phonenumbers library.
- Use the carrier module to get the service provider name for the parsed phone number.
- Print the service provider information.

5. Geocode the location:

- Initialize the OpenCageGeocode object with your API key.
- Convert the location description to a string and store it in the "query" variable.
- Use the geocoder object to geocode the query and store the result.

6. Extract latitude and longitude:

- Retrieve the latitude and longitude values from the geocoder result.
- Print the latitude and longitude values.

7. Create a map:

- Initialize a folium map object with the retrieved latitude and longitude.
- Set an appropriate zoom level for the map.
- Add a marker to the map with the location description as a popup.

8. Save the map:

- Save the map as an HTML file named "myLocation.html".

By following these steps, you will be able to develop a program that retrieves the location and service provider information for a given phone number and displays it on an interactive map. The resulting HTML file can be opened in a web browser to view the map.

Results and Discussion:-

Upon executing the provided code, the following results can be observed:

1. Location:

- The code extracts the location associated with the phone number using the **`geocoder.description_for_number()`** function. The location information is printed to the console using **`print('Location:', yourLocation)`**. This provides the location of the phone number in a human-readable format.

2. Service Provider:

- The code determines the service provider associated with the phone number using the **`carrier.name_for_number()`** function. The service provider information is printed to the console using `print('Service provider:', carrier.name_for_number(service_provider, "en"))`. This displays the name of the service provider.

3. Geocoding and Map Visualization:

- The code utilizes the OpenCage Geocoding service to geocode the obtained location. The geocoder retrieves the latitude and longitude coordinates of the location using `geocoder.geocode(query)`.
- The latitude and longitude coordinates are extracted from the geocoder result using `lat = result[0]['geometry']['lat']` and `lng = result[0]['geometry']['lng']`.
- A map object is created using `folium.Map()` with the center location set to the obtained coordinates and a zoom level of 9.
- A marker is added to the map using `folium.Marker()` at the specified latitude and longitude coordinates. The location information is displayed as a popup when the marker is clicked.

- The map is saved as an HTML file named "myLocation.html" using `myMap.save("myLocation.html")`.

Discussion:

The provided code showcases a simple implementation of tracking a phone number's location and visualizing it on a map using Python. By leveraging the `phonenumbers` library, the location and service provider associated with the phone number are retrieved. The obtained location is then geocoded using the OpenCage Geocoding service, and the resulting latitude and longitude coordinates are used to display the location on a map generated with `folium`.

However, it's important to note that the accuracy of the results heavily relies on the data provided by the phone number's service provider and the geocoding service used. Variations in the data quality and coverage can affect the precision of the location information.

Additionally, the project requires an API key from the OpenCage Geocoding service (`Key='cb1381ee4a9f451996f44cab9930287f'`) or other key. Users should ensure they have a valid API key to access the geocoding service.

Overall, this project provides a foundation for tracking the location of a phone number and visualizing it on a map, offering potential applications in various domains such as telecommunications, geolocation-based services, and fraud prevention.

Source Code:-

#main.py

```
import phonenumbers
import folium

from phonenumbers import geocoder
from number import num

Key='cb1381ee4a9f451996f44cab9930287f'
from phonenumbers import geocoder

sanNumber=phonenumbers.parse(num)

yourLocation=geocoder.description_for_number(sanNumber,"en")

print('Location:',yourLocation)

#service provider

from phonenumbers import carrier

service_provider=phonenumbers.parse(num)

print('Service provider:',carrier.name_for_number(service_provider,"en"))

from opencage.geocoder import OpenCageGeocode

geocoder=OpenCageGeocode(Key)

query=str(yourLocation)

result=geocoder.geocode(query)

print(result)

lat=result[0]['geometry']['lat']

lng=result[0]['geometry']['lng']

print(lat,lng)

myMap=folium.Map(location=[lat,lng],zoom_start=9)

folium.Marker([lat,lng],popup=yourLocation).add_to(myMap)

#save map in html

myMap.save("myLocation.html")
```

#number.py

```
num="+911234567890"
```

#our phone number in above ‘num’

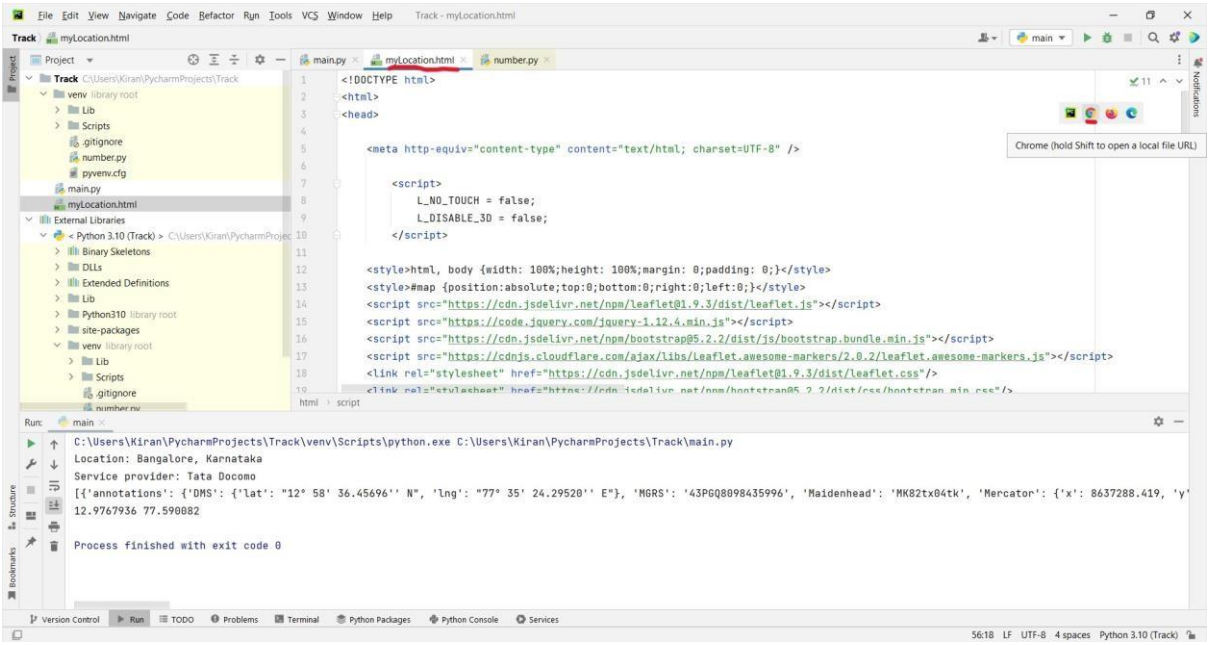
#myLocaotion.htm - ->Auto generated code after run the #main.py program.

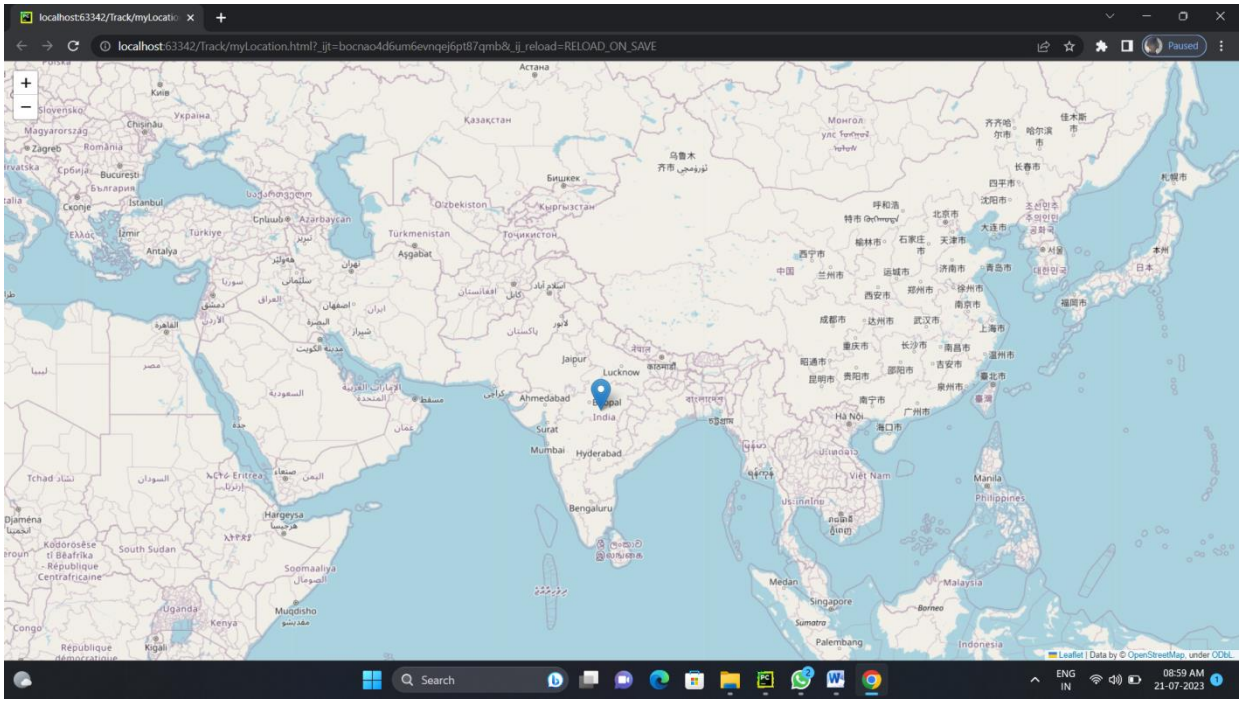
```
<!DOCTYPE html>
<html>
<head>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />

        <script>
            L_NO_TOUCH = false;
            L_DISABLE_3D = false;
        </script>

    <style>html, body {width: 100%;height: 100%;margin: 0;padding:
0;}</style>
    <style>#map {position:absolute;top:0;bottom:0;right:0;left:0;}</style>
    <script
src="https://cdn.jsdelivr.net/npm/leaflet@1.9.3/dist/leaflet.js"></script>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.mi
n.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-
markers/2.0.2/leaflet.awesome-markers.js"></script>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/leaflet@1.9.3/dist/leaflet.css"/>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css
"/>
    <link rel="stylesheet"
href="https://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css"/
>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-
free@6.2.0/css/all.min.css"/>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-
markers/2.0.2/leaflet.awesome-markers.css"/>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/python-
visualization/folium/folium/templates/leaflet.awesome.rotate.min.css"/>

            <meta name="viewport" content="width=device-width,
                initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
            <style>
                #map_76b1b9dd33e69182b872669d891394b6 {
                    position: relative;
                    width: 100.0%;
                    height: 100.0%;
                    left: 0.0%;
                    top: 0.0%;
                }
                .leaflet-container { font-size: 1rem; }
            </style>

</head>
<body>


            <div class="folium-map" id="map_76b1b9dd33e69182b872669d891394b6"
></div>
```

```
</body>
<script>


            var map_76b1b9dd33e69182b872669d891394b6 = L.map(
                "map_76b1b9dd33e69182b872669d891394b6",
                {
                    center: [12.9767936, 77.590082],
                    crs: L.CRS.EPSG3857,
                    zoom: 9,
                    zoomControl: true,
                    preferCanvas: false,
                }
            );




            var tile_layer_c6cf42d2182d7385a85fb810fb1a5aac = L.tileLayer(
                "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
                {"attribution": "Data by \u0026copy; \u003ca
target=\"_blank\"
href=\"http://openstreetmap.org\"\u003eOpenStreetMap\u003c/a\u003e, under
\u003ca target=\"_blank\"
href=\"http://www.openstreetmap.org/copyright\"\u003eODbL\u003c/a\u003e.",
"detectRetina": false, "maxNativeZoom": 18, "maxZoom": 18, "minZoom": 0,
"noWrap": false, "opacity": 1, "subdomains": "abc", "tms": false}
            ).addTo(map_76b1b9dd33e69182b872669d891394b6);


            var marker_a124d7e8dc8443cffb4c9c2f5b57f249 = L.marker(
                [12.9767936, 77.590082],
                {}
            ).addTo(map_76b1b9dd33e69182b872669d891394b6);


        var popup_79d9fbae28e4d9f2c7448aed0c587f10 = L.popup({"maxWidth":
"100%"});



                var html_5907122bc86d1a1c3d8e7c4e2f8bd598 = $(`<div
id="html_5907122bc86d1a1c3d8e7c4e2f8bd598" style="width: 100.0%; height:
100.0%;">Bangalore, Karnataka</div>`)[0];

popup_79d9fbae28e4d9f2c7448aed0c587f10.setContent(html_5907122bc86d1a1c3d8e7c
4e2f8bd598);




marker_a124d7e8dc8443cffb4c9c2f5b57f249.bindPopup(popup_79d9fbae28e4d9f2c7448
aed0c587f10)
        ;



</script>
</html>
```

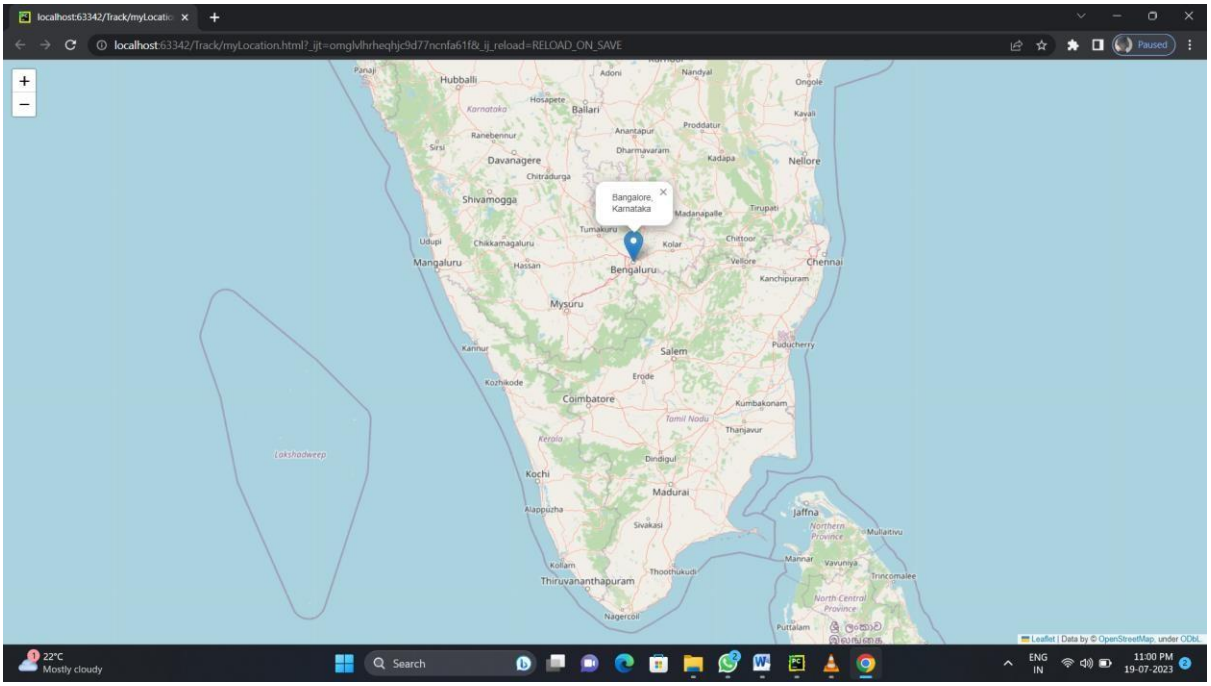
Project Snapshot:

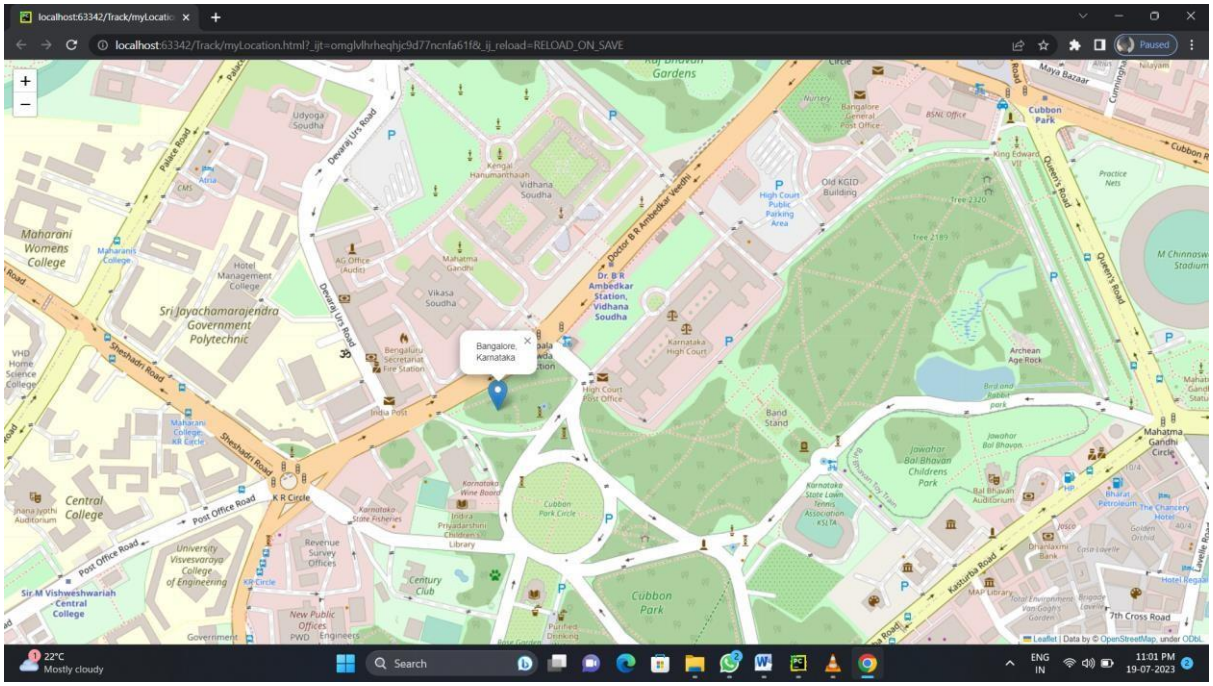
Select myLocation.html and click on chrome to view the location of phone number in map.



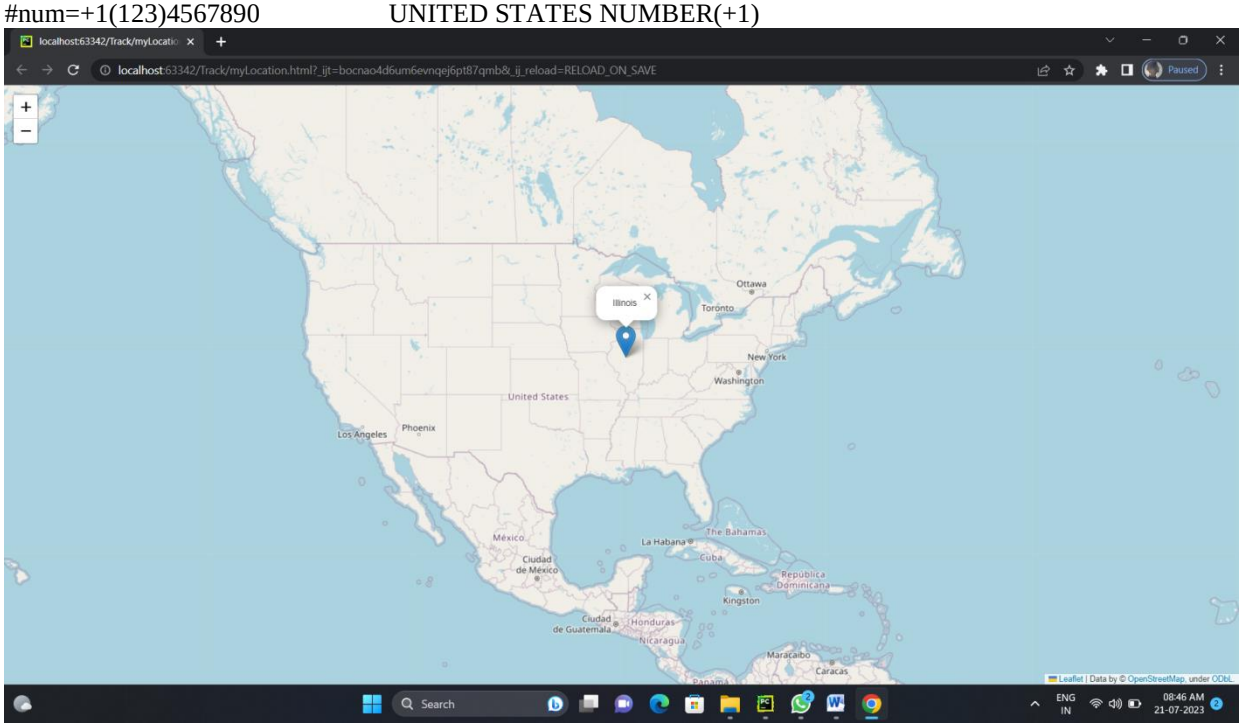


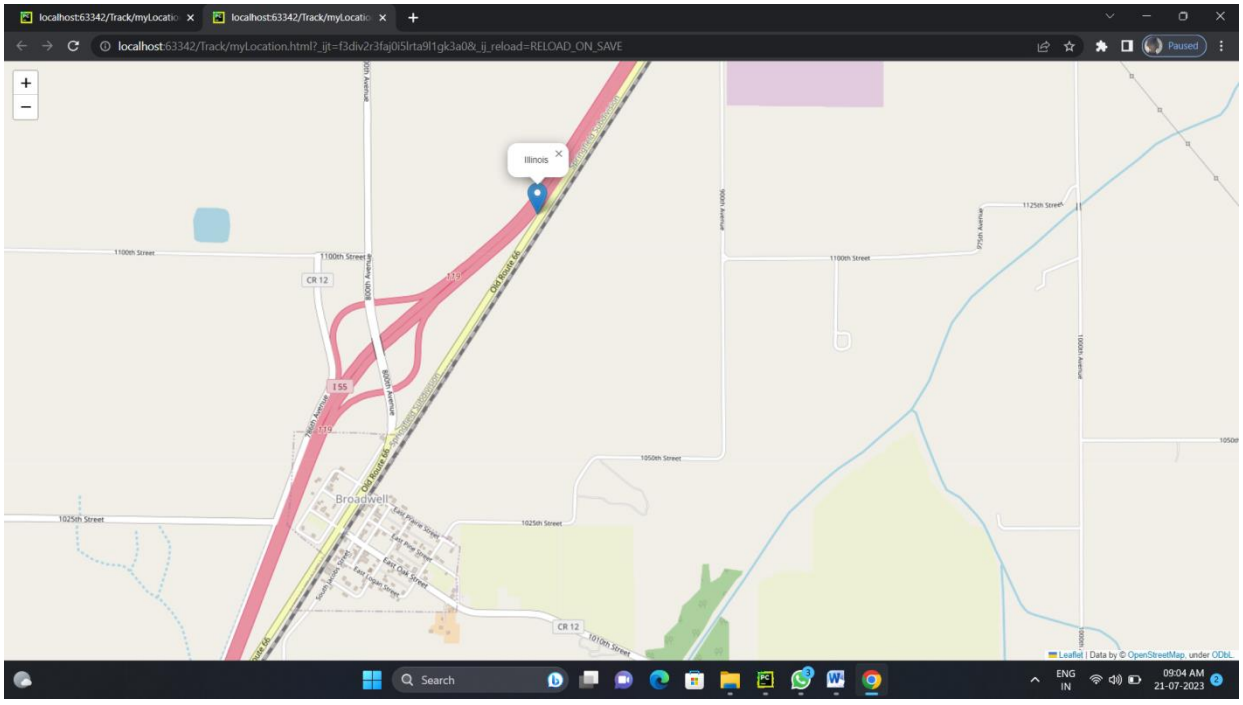
#num='+911234567890' INDIAN NUMBER(+91)





#num=+1(123)4567890 UNITED STATES NUMBER(+1)





Conclusion:

In this project, we have utilized the ‘phonenumbers’ library and the ‘folium’ library to create a program that takes a phone number as input and retrieves the location and service provider associated with that phone number. We have also obtain the ‘OpenCageGeocode’ API to obtain the latitude and longitude co-ordinates of the location.

Future Enhancement:-

One possible future enhancement for this project is to implement a web-based interface where users can input a phone number and view the location and service provider information along with the corresponding map. This can be achieved by integrating the project into a web framework like Flask or Django.

Here are the steps to implement this enhancement:

1. Set up a web framework: Install Flask or Django and set up a basic web application.

2. Create a web form: Design a web form where users can enter a phone number.

3. Handle form submission: Implement the necessary logic to handle the form submission in the web application. Retrieve the phone number entered by the user.

4. Perform phone number lookup: Use the existing code for retrieving the location and service provider information based on the phone number. Adapt the code to accept the user's input phone number.

5. Generate the map: Generate the map and marker using the retrieved location coordinates.

6. Display the results: Render the location and service provider information along with the map on a new web page or as a part of the existing page.

7. Improve the user interface: Enhance the user interface by adding styling, improving the layout, and making it more user-friendly.

8. Deploy the web application: Deploy the web application to a hosting platform or server so that it can be accessed by users.

By implementing this enhancement, users will be able to access the phone number lookup functionality through a web interface, making it more convenient and accessible. They can

Overall,this project allows us to input a phone number, retrieve its associated location and service provider , and visualize the location on an interactive map.

easily enter a phone number, view the location, service provider, and visualize the location on a map directly from their web browser.

Reference

https://youtu.be/Dz3rSZHnKkM