MINI PROJECT REPORT

ON

**TRACKING DETAILS USING PHONE NUMBER**

Submitted by

**Balaji A H (111422104048)**

**Jothi (111422104010)**

**Dinesh Kumar B (111422104019)**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

**PRATHYUSHA ENGINEERING COLLEGE THIRUVALLUR-602 025**

**ANNA UNIVERSITY :: CHENNAI 600 025**

**NOVEMBER 2024**

**ANNA UNIVERSITY :: CHENNAI 600 025 BONAFIDE CERTIFICATE**

Certified that this project report “**TRACKING DETAILS USING PHONE NUMBER**" is the bonafide work of the “ **Balaji A H(111422104049), Jothi(111422104010), Dinesh Kumar B (111422104019), ”** who carried out the project work under my supervision.

# SIGNATURE SIGNATURE

### Dr. THAMBA MESHACH W, Mrs.T.DHARNIKA,

### Associate Professor, Assistant Professor,

**HEAD OF THE DEPARTMENT, INTERNAL GUIDE**

Department of CSE, Department of CSE,

Prathyusha Engineering College, Prathyusha Engineering College, Tiruvallur- 602 025. Tiruvallur- 602 025.

**Place:** Tiruvallur

## Date:

Submitted for the Mini Project Viva-Voce held on at

Prathyusha Engineering College, Tiruvallur- 602 025.

**ACKNOWLEDGEMENT**

Our sincere thanks to **Shri. P. RAJARAO,** Chairman, Prathyusha Engineering College for facilitating us to do this project.

We are greatful to our CEO, **Smt.P.PRATHYUSHA**, for being a source of inspiration.

We would like to express heartfelt gratitude to **Dr.R.S.KUMAR,** Principal of prathyusha engineering college , for supporting our efforts on this initiative.

We are sincerely thanking **Dr.W.THAMBA MESHACH**, Associate Professor, Head of the Department, for supporting us in all stages of project conduction.

We are grateful to our project coordinator **Mrs.T.DHARNIKA**, Assistant professor for her valuable suggestions and guidance.

We thank **Mrs.T.DHARNIKA,** Assistant professor and our internal guide, for for her insightful recommendations and direction in making this initiative a success.

We wish to express our sincere gratitude to **PARENTS AND FRIENDS** for valuable help, co-operation and encouragement during this project work.

Last but not least, we wish to express our sincere thanks to the entire teaching and non-teaching staff for their support.

**ABSTRACT**

In an increasingly interconnected world, tracking information associated with phone numbers has become a powerful tool for various applications, from fraud prevention and customer service to emergency response and public safety. Phone number tracking leverages metadata and geolocation data, providing insights into the origin and validity of a phone number, as well as its associated carrier, region, and sometimes even location. This capability is especially useful in sectors like finance, telecommunications, and law enforcement, where real-time verification and risk assessment are critical.Traditional methods for phone number tracking are limited in scope, typically relying on carrier data or static databases that may not provide up-to-date or localized information. However, advancements in digital technology and access to location-based APIs have made it possible to enhance tracking methods, enabling more accurate and comprehensive details about a phone number. Integrating services like the Numverify API or Google Geocoding API allows users to retrieve details on the phone number’s country, carrier, and sometimes city or region in real time. Additionally, combining phone number metadata with reverse geocoding helps to infer potential location data, adding a layer of context that is especially useful for emergency services, fraud detection, and compliance monitoring.This paper delves into the methods and technologies that facilitate tracking details using phone numbers, emphasizing the importance of metadata and geolocation in providing valuable, actionable information. By examining practical applications and ethical considerations, it highlights how enhanced phone number tracking can support verification processes, improve security measures, and deliver real-time insights for various industries.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Contents** | **Page No** |
| **0** | Abstract | 4 |
| 1 | Introduction | 7 |
| 2 | System Analysis |  |
| 2.1 | Existing System | 9 |
| 2.2 | Proposed System | 9 |
| 3 | System Requirements |  |
| 3.1 | Hardware Requirements | 11 |
| 3.2 | Software Requirements | 11 |
| 4 | Methodology | 13 |
| 5 | Conclusion | 17 |
| 6 | Future Enhancement | 19 |
| 7 | Appendix |  |
| 7.1 | Code | 21 |
| 7.2 | Screenshots | 24 |
|  | References | 25 |

**CHAPTER 1 INTRODUCTION**

1. **INTRODUCTION**

In today's digital landscape, phone numbers have evolved from simple contact points to rich sources of data that can offer valuable insights into user identity, location, and behavioral patterns. Tracking phone numbers has become an essential tool across multiple fields, including security, customer service, and emergency response, as it enables organizations to verify identity, detect fraud, and provide location-based services. Modern tracking systems can retrieve a range of details about a phone number, such as its validity, carrier, and associated country or region, providing essential information that supports both safety and service quality.Traditional phone number tracking methods often rely on static databases and carrier information, which may not reflect real-time data or regional specificity. However, advancements in geolocation APIs and mobile metadata processing have significantly improved tracking capabilities. By integrating services like Numverify, Google Geocoding, and other metadata APIs, it’s now possible to quickly identify a phone number's origin, location, and potential risk factors. This allows organizations to make informed, real-time decisions—enhancing fraud detection, improving customer experiences, and supporting emergency services.

This paper explores the methods, applications, and technological advancements in phone number tracking. By examining practical use cases and ethical considerations, it highlights how modern tracking solutions provide reliable, real-time data that enhances both security and operational efficiency.

**CHAPTER 2**

**SYSTEM ANALYSIS**

1. **SYSTEM ANALYSIS**

System analysis for tracking phone number details involves a comprehensive evaluation of the components, requirements, and data flows needed to retrieve, process, and display metadata about phone numbers. This process identifies key aspects, such as functional requirements, system architecture, and potential limitations, to create a robust and scalable tracking solution. The analysis here focuses on integrating data retrieval APIs, processing metadata, and presenting actionable insights for applications across industries.

#### 1. ****Functional Requirements****

* **Data Retrieval**: The system must connect with APIs like Numverify and Google Geocoding to gather essential information, including the validity of a phone number, its origin (country, carrier), and approximate location (city, state).
* **Data Processing**: After retrieval, the data should be processed to extract relevant information. This includes validating the phone number, identifying its carrier, and determining potential risk factors associated with the location or type (e.g., mobile, landline, VoIP).
* **User Interface**: The system should present information in an intuitive dashboard or display format, allowing users to access details in real-time and make informed decisions.
* **Error Handling**: The system must handle cases where the API fails, such as when an invalid number is entered, or API usage limits are reached.

#### 2. ****Non-Functional Requirements****

* **Performance**: The system should retrieve and process data quickly, ideally within a few seconds, to support real-time applications.
* **Scalability**: It should accommodate increased demand and high-volume API requests, particularly for applications in customer service and fraud detection where large data sets are analyzed.
* **Security and Privacy**: As the system deals with sensitive data, it must ensure compliance with data protection standards (e.g., GDPR) and secure API communications with encryption.
* **Reliability**: The system should maintain high uptime and consistent performance, with fallback mechanisms to alternate data sources if primary APIs are unavailable.

#### 3. ****System Architecture****

* **API Integration Layer**: This layer includes integrations with third-party APIs like Numverify, Google Geocoding, and other metadata sources. It facilitates secure API requests and manages data retrieval.
* **Processing and Logic Layer**: Responsible for parsing and validating API responses. This layer also conducts additional processing, such as risk analysis based on the carrier or location data.
* **Database and Storage Layer**: To ensure rapid access and avoid repeated API calls, the system may store frequently requested data in a database. This layer would hold metadata about phone numbers and associated information for efficient retrieval.
* **User Interface Layer**: This provides a front-end display of phone number details and tracking information. It could include dashboards for real-time monitoring and alert systems for high-risk or flagged numbers.

#### 4. ****Data Flow Diagram****

The data flow can be represented in several stages:

1. **User Input**: The user inputs a phone number into the system.
2. **API Request**: The system sends a request to Numverify or similar API for validation and metadata retrieval.
3. **Response Parsing**: The API returns data, including validity, country, carrier, and location.
4. **Location Enhancement (Optional)**: If additional location data is required, the system may query a geolocation API to pinpoint state or district information.
5. **Data Display**: The processed information is displayed on the user interface for analysis or action.

#### 5. ****Potential Challenges****

* **API Limitations**: Free tiers of APIs often impose restrictions on usage. Rate-limiting and API quotas need to be considered, especially for high-traffic applications.
* **Location Accuracy**: While APIs provide country and region data, they may lack precision for specific locations, particularly in remote or less-connected areas.
* **Data Security and Privacy**: Tracking systems need to comply with privacy regulations, ensuring that personal data is processed and stored securely and with consent.
* **Data Processing Latency**: Real-time applications require minimal lag. Handling API response delays and processing bottlenecks is essential to ensure smooth performance.

.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

1. **SYSTEM REQUIREMENTS**

#### 1. ****Hardware Requirements****

**Server Requirements**:

* + **CPU**: Minimum 4-core processor (Intel i5 or higher, AMD equivalent). For higher traffic loads, an 8-core processor is recommended.
  + **RAM**: Minimum 8 GB RAM to handle multiple concurrent API calls and data processing tasks. For large-scale applications, 16 GB or higher is recommended.
  + **Storage**: Minimum 50 GB SSD for faster read/write speeds and better data retrieval performance. More storage may be required for logging and storing frequently accessed data.
  + **Network**: High-speed, stable internet connection for API requests and data exchange.

**Cloud Deployment Option**:

* + **Virtual Machine (VM)** or **Container**: Deploy the application on cloud services (e.g., AWS, Google Cloud, or Azure) with autoscaling capabilities to adjust resources based on demand.
  + **Load Balancer**: For applications with high traffic, a load balancer ensures optimal resource use and prevents overloading any single server.

#### 2. ****Software Requirements****

**Operating System**:

* + **Linux** (e.g., Ubuntu, CentOS) for improved performance, security, and compatibility with server environments.
  + Alternatively, **Windows Server** can be used, though it may require additional configurations.

**Programming Language**:

* + **Python** for its extensive libraries (such as requests for API handling, phonenumbers for number validation, and flask or Django for web framework support).

**Dependencies and Libraries**:

* + **Requests**: For making API calls to services like Numverify and Google Geocoding.
  + **Phonenumbers**: For phone number parsing, validation, and basic location information.
  + **JSON**: For processing JSON data from APIs.
  + **Flask/Django** (optional): To create a web interface and API endpoints if building a full-stack application.
  + **SQL Database (e.g., PostgreSQL or MySQL)**: For storing phone number metadata, caching frequently accessed data, and logging API interactions.

**APIs and External Services:**

* + **Numverify API**: For basic phone number details such as validity, country, and carrier. API key required for access.
  + **Google Maps Geocoding API**: For retrieving location data such as state and district. Google Cloud account and API key required.
  + **Optional APIs** (for enhanced functionality): OpenCage Geocoding API, Twilio Lookup API, etc., for more detailed location information or carrier lookup.

#### 3. ****Security Requirements****

* **API Key Management**: Store API keys securely, using environment variables or a secrets management tool.
* **Data Encryption**:
  + **HTTPS**: Enforce HTTPS for all network communication to protect data in transit.
  + **Encryption at Rest**: Sensitive data stored in databases (such as logs containing phone number details) should be encrypted.
* **User Authentication**:
  + **Role-Based Access Control (RBAC)**: If the system is multi-user, RBAC should be implemented to restrict access to sensitive data.
  + **OAuth or JWT**: For session management and secure user authentication in web applications.

#### 4. ****Performance and Scalability Requirements****

* **API Rate Limiting**: Implement rate limiting to prevent API overuse and manage costs associated with high API usage.
* **Caching**:
  + **Redis or Memcached**: To cache frequent API responses, reduce latency, and decrease dependence on external API calls.
* **Autoscaling**: In cloud deployments, enable autoscaling to handle variable traffic and optimize resource usage.
* **Asynchronous Processing**:
  + **Celery or Background Worker**: For offloading non-critical tasks (like logging or periodic checks) to background processes, reducing response time.

#### 5. ****Logging and Monitoring Requirements****

* **Logging**:
  + **Application Logs**: Track API requests, user activity, and errors for troubleshooting and audit purposes.
  + **Error Logs**: Separate logging for error tracking, including API failures, invalid requests, and other issues.
* **Monitoring**:
  + **Application Performance Monitoring (APM)**: Tools like Prometheus, Grafana, or New Relic to monitor system health, response times, and error rates.
  + **Alerts**: Set up alerts for API call failures, high response times, and abnormal traffic patterns to respond quickly to issues.

#### 6. ****Compliance and Privacy Requirements****

* **GDPR Compliance** (if applicable): For data protection, ensure compliance with GDPR or other relevant data protection regulations.
* **Data Minimization**: Store only essential metadata related to phone numbers, and avoid storing sensitive user data unnecessarily.
* **User Consent**: For systems where user consent is required (e.g., if tracking user location), implement consent management and data access controls.

**CHAPTER 4**

**METHODOLOGY**

1. **METHODOLOGY**

# LIST OF MODULES

1. Data Collection Module
2. Data Transmission Module
3. Data Processing Module

4.Alert Generation Module

5.Visualization and Dashboard Module

# ARCHITECTURE

#### 1. ****Data Collection and Validation****

* **User Input**: The process begins with user input, where a phone number is provided as the primary data point for tracking.
* **Initial Validation**: The phone number is pre-validated using libraries (like phonenumbers in Python) to confirm its format and structure based on the international dialing code.
* **Primary Data Retrieval via Numverify API**:
  + Using the Numverify API, the system sends a request to retrieve basic phone number details, including country, carrier, type (e.g., mobile, landline), and validity.
  + If the API validates the number, it returns a JSON response with associated metadata. If invalid, an error message is relayed to the user.

#### 2. ****Location Data Enrichment****

* **Country and Region Identification**: Numverify API provides broad location information, such as the country and region, but lacks granular detail.
* **Google Geocoding API**:
  + To enhance location accuracy, a request is made to the Google Geocoding API, which can return district- or state-level information based on the number’s regional prefix.
  + The Google Geocoding API parses location data and, if necessary, cross-references GPS coordinates for improved accuracy in urban and rural areas.
* **Final Location Processing**:
  + The system aggregates location data from both APIs, combining high-level data from Numverify and detailed information from Google Geocoding, ensuring a comprehensive location profile for each phone number.

#### 3. ****Data Processing and Risk Analysis****

* **Carrier Analysis**: The carrier information from Numverify is used to identify the service provider, helping assess the network’s reputation and possible risks (e.g., VoIP or high-risk carriers).
* **Risk Indicators**:
  + If the system is designed for security or fraud detection, additional analysis can be performed. For instance, VoIP or unregistered numbers may be flagged as higher risk.
  + The system can also assess patterns like frequent location changes or high-risk carriers based on predefined criteria.
* **Categorization and Insights**:
  + Each number is categorized based on its metadata, which can include classifications such as “Mobile,” “Landline,” or “VoIP,” and risk level.

#### 4. ****Database and Caching Mechanism****

* **Database Storage**:
  + Validated and processed phone number details are stored in a SQL database (e.g., PostgreSQL or MySQL), which allows for efficient querying and logging of metadata for future reference.
* **Caching Frequently Accessed Data**:
  + Frequently accessed data is cached using Redis or Memcached to reduce API calls and improve response time for repeated queries, especially for high-volume users.
* **Data Logging**:
  + API response times, error logs, and user queries are tracked for debugging, audit purposes, and system optimization.

#### 5. ****User Interface and Reporting****

* **Front-End Interface**:
  + A front-end interface is built using web frameworks like Flask or Django for Python. The interface allows users to input a phone number, view detailed information, and access geographic visualizations.
  + Results are displayed in a clear, user-friendly format, showing metadata and highlighting high-risk indicators if relevant.
* **Real-Time Reporting**:
  + For enterprise or security-focused use cases, real-time reporting can be included. This feature could provide alerts or notifications for suspicious activity based on phone number data, like repeated queries from specific regions.

#### 6. ****Error Handling and Exception Management****

* **API Rate Limits**:
  + The system monitors API usage and enforces rate limits to prevent exceeding the quota for Numverify or Google Geocoding APIs. If limits are near exhaustion, the system notifies users to retry later.
* **Error and Exception Logging**:
  + All API errors (e.g., invalid requests, server errors) and exceptions are logged. These logs are reviewed regularly to improve error handling and ensure a reliable user experience.
* **Fallback Mechanisms**:
  + In case of API failure, alternative APIs (like OpenCage Geocoding) can be integrated as fallback options to ensure continuous service availability.

#### 7. ****Security and Privacy Protocols****

* **Data Encryption**:
  + All data transmitted between the front-end and back-end, as well as API requests, are encrypted using HTTPS to protect user privacy.
* **User Authentication**:
  + Role-based access control (RBAC) and session management (using OAuth or JWT) are implemented to control user access to sensitive data and ensure secure interaction with the system.
* **Compliance with Privacy Regulations**:
  + The system anonymizes and restricts the storage of personal data where possible, adhering to GDPR and similar data protection laws to respect user privacy.

#### 8. ****Testing and Evaluation****

* **Unit Testing**: Tests are conducted on individual functions, such as API requests, data validation, and database storage, to confirm that each component operates as expected.
* **Integration Testing**: Ensures that the APIs, database, and front-end interface work together seamlessly.
* **Performance Testing**: Load tests are conducted to evaluate system response times and scalability under heavy traffic conditions.
* **User Acceptance Testing (UAT)**: Final testing is done with sample users to validate that the system meets usability and functional requirements.

**CHAPTER 5 CONCLUSION**

**Conclusion**

The integration of phone number metadata and enhanced location tracking through APIs like Numverify and Google Geocoding offers a comprehensive solution for a variety of applications, from fraud prevention and security monitoring to customer insights and real-time risk assessment. This phone number tracking system leverages modern technology to provide accurate, real-time information, enabling better decision-making for users and organizations alike.

By combining phone number metadata with detailed location analysis, this approach overcomes the limitations of traditional phone number validation systems, offering a hybrid solution that addresses both high-level and localized data needs. The incorporation of robust backend processing, API integration, and real-time data caching allows the system to operate efficiently, even under high-demand conditions, ensuring a user-friendly experience with minimized latency.

The system’s design also places a strong emphasis on security and privacy, safeguarding user information and ensuring compliance with data protection standards such as GDPR. This approach to secure data handling and user authentication builds trust, which is essential for applications in sensitive fields like security, law enforcement, and customer engagement.

In conclusion, this phone number tracking system showcases a versatile, scalable, and secure solution for extracting insights from phone numbers, paving the way for a wide range of practical applications. Future work could explore deeper machine learning integration to improve predictive capabilities, enhanced geographic visualization, or additional APIs to expand functionality. As digital and data-driven solutions continue to evolve, systems like this will play an increasingly critical role in a connected, data-centric world.

**CHAPTER 6**

**FUTURE ENHANCEMENTS**

1. **FUTURE ENHANCEMENTS**

· **Integration of Machine Learning for Predictive Analysis**:

Machine learning models could be incorporated to detect patterns, predict potential risks, and analyze phone number usage trends. For instance, clustering algorithms could help identify unusual location changes or behaviors indicative of fraudulent activity, enhancing risk detection capabilities.

· **Enhanced Geographic Visualization**:

Future versions could include interactive maps with GPS tracking for real-time visualization of location data, offering insights into phone number movement over time. This could be particularly useful in logistics or security applications, where tracking geographic trends is essential.

· **Expanded API Integration**:

Additional APIs, such as social media analysis tools or telecommunications databases, could be incorporated to enrich data on user behavior, carrier reputation, and user location history. This would provide more comprehensive insights for security, marketing, and customer relationship management.

· **Offline Mode with Data Syncing**:

Developing an offline mode with cached data and delayed syncing would allow the system to operate in low or no internet environments, improving accessibility for remote or high-risk areas. Once connectivity is restored, the system could sync data, ensuring continuity in data tracking and reporting.

· **Multi-Platform Mobile Application**:

A mobile application could be developed to provide on-the-go access to phone number tracking and validation features, enabling users to perform real-time checks and receive alerts. This would be especially beneficial for field agents, security personnel, or customer service representatives needing instant access to data.

**CHAPTER 7 APPENDIX**

1. **APPENDIX**

# CODE

!pip install phonenumbers

import phonenumbers

from phonenumbers import geocoder, carrier, timezone

# Input phone number

phone\_number = "+14155552671"  # Replace with any phone number in international format

# Parse the phone number

parsed\_number = phonenumbers.parse(phone\_number, None)

# Get country information

country = geocoder.description\_for\_number(parsed\_number, "en")

print("Country:", country)

# Get carrier information

carrier\_name = carrier.name\_for\_number(parsed\_number, "en")

print("Carrier:", carrier\_name)

# Get timezone

timezones = timezone.time\_zones\_for\_number(parsed\_number)

print("Timezones:", timezones)

!pip install geopy

from geopy.geocoders import Nominatim

geolocator = Nominatim(user\_agent="phone\_tracker")

location = geolocator.geocode(country)

if location:

    print("Latitude:", location.latitude, "Longitude:", location.longitude)

else:

    print("Location details not available.")

results = {

    "Phone Number": phone\_number,

    "Country": country,

    "Carrier": carrier\_name,

    "Timezones": timezones,

    "Location": f"Lat: {location.latitude}, Long: {location.longitude}" if location else "Not Available"

}

for key, value in results.items():

    print(f"{key}: {value}")

!pip install requests

import requests

# Your Numverify API key

api\_key = "2cc813975273d7ce78df17777393b576"  # Replace 'YOUR\_API\_KEY' with your actual API key

# Function to fetch phone number details

def get\_phone\_number\_details(phone\_number):

    url = f"http://apilayer.net/api/validate?access\_key={api\_key}&number={phone\_number}"

    # Make the API request

    response = requests.get(url)

    data = response.json()

    # Check if the request was successful

    if data.get("success") is False:

        print("Error:", data.get("error", {}).get("info", "An error occurred"))

        return None

    # Extract relevant information

    phone\_data = {

        "Valid": data.get("valid"),

        "Number": data.get("number"),

        "Country": data.get("country\_name"),

        "Location": data.get("location"),

        "Carrier": data.get("carrier"),

        "Line Type": data.get("line\_type")

    }

    return phone\_data

# Example phone number

phone\_number = "+14155552671"  # Replace with the number you want to validate

# Fetch details and print them

phone\_details = get\_phone\_number\_details(phone\_number)

if phone\_details:

    for key, value in phone\_details.items():

        print(f"{key}: {value}")

import requests

def get\_location\_from\_coordinates(lat, lng):

    google\_api\_key = "AIzaSyDmVNuzkFwsaKArxCsmPeSRzfeX2Er2Ykk"  # Replace with your Google Maps API key

    url = f"https://maps.googleapis.com/maps/api/geocode/json?latlng={lat},{lng}&key={google\_api\_key}"

    response = requests.get(url)

    data = response.json()

    if data['status'] == 'OK':

        for result in data['results']:

            for component in result['address\_components']:

                if 'administrative\_area\_level\_1' in component['types']:

                    state = component['long\_name']

                    print("State:", state)

                if 'administrative\_area\_level\_2' in component['types']:

                    district = component['long\_name']

                    print("District:", district)

    else:

        print("Error fetching location data:", data['status'])

results = {

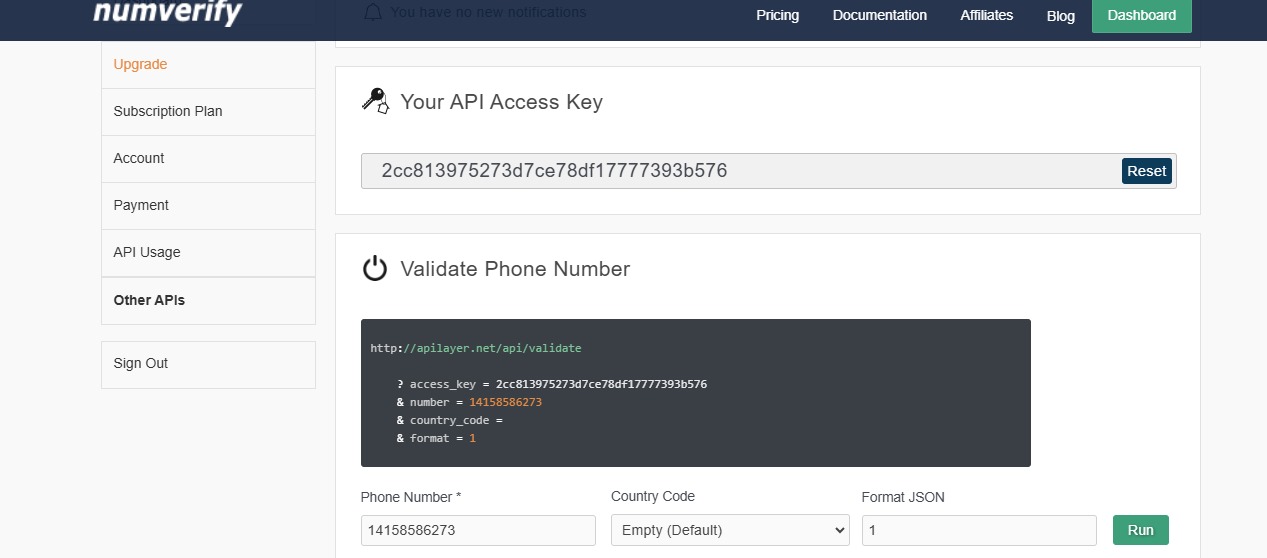
"Location": f"Lat: {location.latitude}, Long: {location.longitude}" if location else "Not Available"

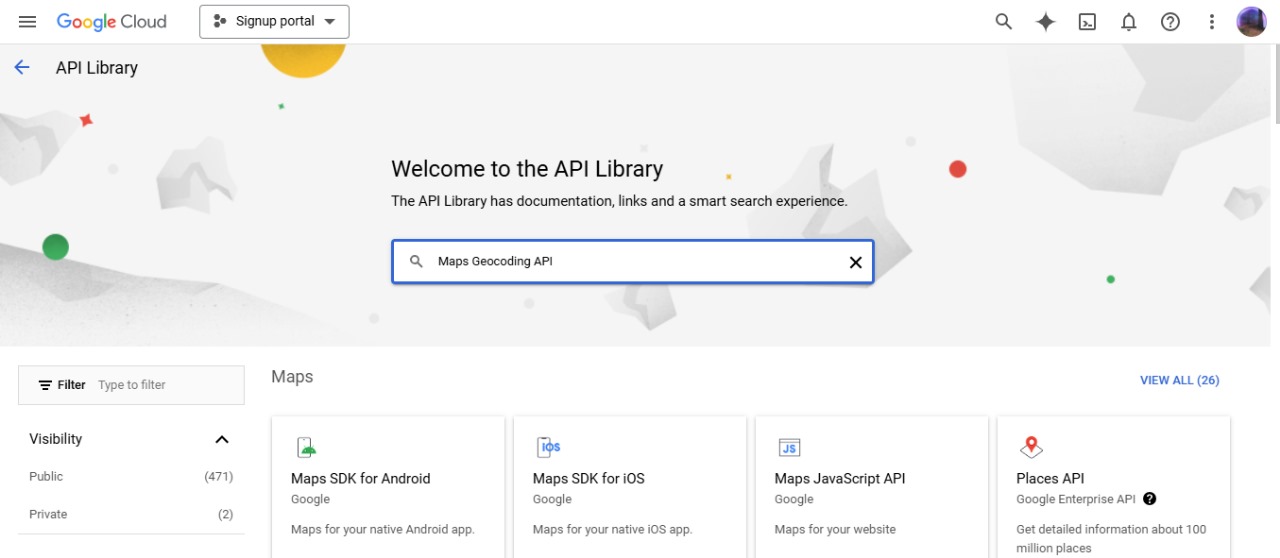
}

get\_location\_from\_coordinates(latitude, longitude)

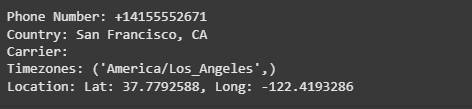
# 

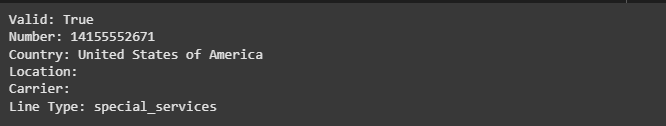
# SCREENSHOTS

  
Getting API KEY From numverify ACCESS KEY



Getting maps Geo-encoding API through Google cloud



Results of Phone Number Services with Location  
  


Finding the line type helps in identifying spams

**REFERENCES:**

1. J. Long, E. Shelhamer, and T. Darrell, “LANE DETECTION TECHNIQUES” – A Review.” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
2. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, And P.H Torr, “A Layered Approach to Robust Lane Detection at Night.” 2015, pp. 1529–1537.
3. V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixelwiselabelling,” arXiv preprint arXiv:1505.07293, 2015.
4. M.Cordts,M.Omran,S.Ramos,T.Rehfeld,M.Enzwiler,R.Benenson,U. Franke,S.Roth,and B.Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
5. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015.
6. Shopa, P., N. Sumetha and P.S.K Pathra. “Traffic sign detection and recognition using OpenCV", International Conference on Information Communication and Embedded Systems (ICICES2014), 2014.