International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04879
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

# NUMBER TRACKER MAIN USING PYTHON

M.SATISH, ALLANKI.OOHA
Assistant Professor, MCA Final Semester, Master of Computer Applications,
Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India.

**ABSTRACT:**

This project presents a Phone Number Tracker application built using Python and Tkinter. The system enables users to track essential details of any entered phone number in a user-friendly GUI. It integrates the phonenumbers library to parse and extract the country, operator, and timezone information of the provided phone number. The application utilizes geopy and Timezone Finder to determine the latitude, longitude, and local time of the phone number's location. The tool fetches geolocation data accurately using Nominatim geocoder, ensuring real-world mapping consistency. The GUI design includes a clean entry field, search button, and dynamic display of country, SIM carrier, timezone, time, longitude, and latitude. The system icon and structured layout enhance usability for non-technical users. Error handling is incorporated to handle invalid numbers gracefully. The system can aid in basic telecom analysis and learning projects. It visually demonstrates how Python GUI and external APIs can be integrated effectively. Users gain experience with libraries like pytz, phonenumbers, and tkinter. The project also demonstrates practical handling of geolocation APIs. This tool can be extended for spam analysis, telecom audits, or law enforcement utilities. It emphasizes how real-time data can be fetched using minimal resources. The design aligns with standard event-driven programming structures in Python. The system is lightweight and suitable for academic demonstrations. It also highlights modular code structuring for clarity and reusability. Overall, the Phone Number Tracker showcases practical, clear, and impactful implementation of phone number geolocation in Python.

**Key Words:** Tkinter, Phone Numbers, Carrier, Geocoder, Timezone, Timezone Finder, Geopy/Nominatim, Pytz, Datetime.

## I.INTRODUCTION

In today's digitally connected world, identifying the location, network provider, and timezone of a phone number can be essential for security, logistics, and user support services. The Phone Number Tracker application is a Python-based desktop tool developed using the Tkinter library for its graphical user interface (GUI). It leverages several Python packages, including phonenumbers, geopy, timezonefinder, and pytz, to extract comprehensive details about any valid phone number entered by the user [5]. This application parses the entered phone number and retrieves its geographical location using the phonenumbers and geopy libraries, identifies the carrier network, and determines the time zone associated with the number. Additionally [6], it fetches the longitude and latitude of the location, displaying this data visually within the GUI. To enhance user experience, the application dynamically fetches and displays the current local time at the phone number's location, ensuring users can track and communicate effectively across different regions [15]. The interface is designed to be user-friendly, providing a clear entry field for input and displaying results categorically, including the country [21], SIM carrier, time zone, longitude, latitude, and local time. This project demonstrates practical integration of multiple Python libraries for real-time data extraction and visualization within a single desktop application, making it a useful tool for learning about geolocation services and GUI development using Python.

### 1.1 Existing System

In the existing system, users who wish to identify details about a phone number typically need to search manually across various online platforms to find the carrier, region, or time zone associated with the number [13]. This process can be time-consuming, inaccurate, and fragmented, requiring the user to visit multiple websites and enter the phone number repeatedly to gather different pieces of information like the location, SIM operator, timezone, and geographical coordinates. Additionally, the traditional manual method does not provide live local time based on the phone number's location [6], nor does it present this information in a unified [20], user-friendly graphical interface. Users may also need to interpret raw location coordinates themselves if any online service provides them, which can be difficult for non-technical users.

#### 1.1.1 Challenges

During the development of the Phone Number Tracker application, several challenges were encountered:

- **Accurate Parsing of International Numbers:** Handling different formats of phone numbers from various countries required ensuring the correct use of the phonenumbers library while managing invalid or incomplete entries to avoid crashes.

- **Geolocation Accuracy:** Obtaining precise longitude and latitude using the geopy and Nominatim services was sometimes inconsistent, particularly for numbers linked to regions without clear geolocation data, requiring fallback handling and user feedback when coordinates were unavailable [7].
- **Timezone Identification:** Mapping retrieved coordinates to accurate timezones using timezonefinder posed challenges due to potential mismatches or the absence of timezone data for certain coordinates [18], necessitating appropriate exception handling to prevent application failure.
- **Real-Time Time Display:** Displaying the current local time for the tracked location demanded precise timezone handling and synchronization with system time [1], which required additional care to avoid showing incorrect times due to timezone misalignments.

### 1.2 Proposed System

- **Automatic Number Parsing:** Accepts the phone number entered by the user and automatically parses it to extract structured information.
- **Country Detection:** Identifies and displays the country associated with the phone number [6].
- **SIM Operator Detection:** Retrieves the carrier/operator information for the phone number.
- **Longitude and Latitude Retrieval:** Uses geolocation services to obtain and display the precise coordinates of the detected region.
- **Live Local Time Display:** Displays the current local time of the phone number's location, helping users know the time before calling.
- **Graphical User Interface:** Uses Tkinter for a clean, user-friendly interface with input fields [10], labels, and graphical buttons, making it easy for non-technical users to operate.
- **Error Handling:** Displays user-friendly error messages if invalid numbers are entered or network issues occur while fetching data.
- **Compact Layout:** All details are presented within a single window [22], ensuring clarity and ease of reading.

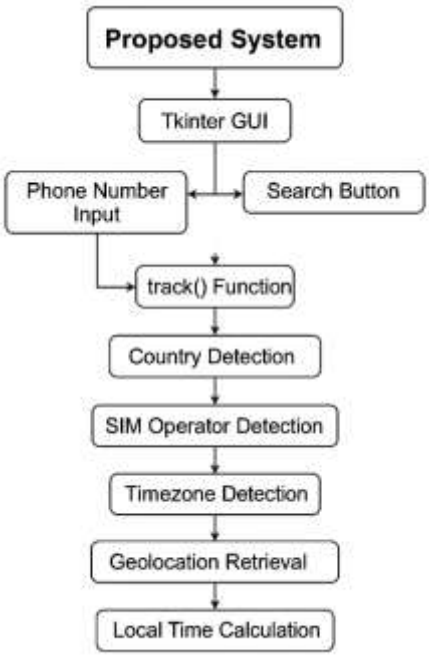


**Fig: 1 Proposed System**

#### 1.2.1Advantages

- **Centralization:** All essential phone number details are retrieved and displayed in one place.
- **Time-Efficient:** Reduces user effort by automating the information retrieval process [6].
- **User-Friendly:** The intuitive GUI makes the tool accessible for all types of users, including those with limited technical skills.
- **Accuracy:** Uses reliable APIs and libraries to fetch accurate carrier, location, and timezone data.
- **Educational Use:** Helps students understand practical implementation of phone number parsing, API usage, and GUI application development in Python [4].
- **Extendable:** Can be extended to include features such as call validation, spam detection, or integration with additional APIs for advanced features.

## II. LITERATURE REVIEW

### 2.1 Architecture

The Phone Number Tracker system architecture is designed as a layered desktop application that uses a Python Tkinter GUI and integrates multiple libraries to automate phone number tracking.

**Architectural Flow:**

- **User Input:** Phone number entered in the Tkinter entry field.
- **Event Trigger:** User clicks the Search button.
- **Processing:**
    - Phone number parsed and validated.
    - Country [19], SIM operator, and timezone retrieved.
    - Location geocoded to obtain longitude and latitude.
    - Local time calculated using timezone data [14].
- **Display:** All details are shown in the GUI for the user to view in a clear, organized format.

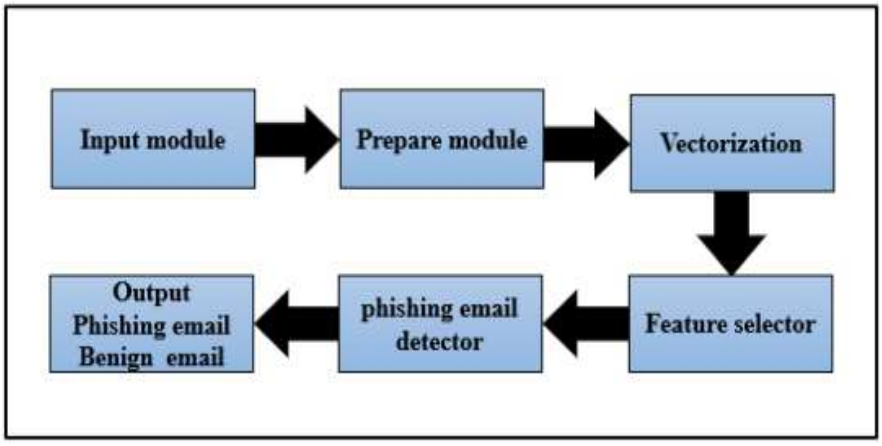


**Fig: 2 Architecture Diagram**

### 2.2 Algorithm:

The Phone Number Tracker uses straightforward, library-based algorithms to automate phone number detail extraction and display [6]. The following algorithms are implemented in the system:

- **Phone Number Parsing Algorithm:** The phone numbers library parses the input string into a structured phone number object, enabling further data extraction.
    - **Country Detection Algorithm:**
        - **Objective:** To detect the country associated with the parsed phone number [19].
        - Returns a string indicating the country or region associated with the phone number using phone numbers.geocoder.
    - **Timezone Detection Algorithm**:
        - **Objective:** To retrieve the timezone(s) associated with the phone number's location.
        - Fetches a list of potential time zones for the phone number.

### 2.3 Techniques:

- In the proposed Phone Number Tracker system, various programming and library-based techniques are used to achieve automated phone number detail extraction and display within a user-friendly GUI.

- The system uses Tkinter to build a graphical user interface where users can enter a phone number and view the extracted details clearly [23].
- The phone number parsing technique is applied using the phonenumbers library to parse and validate the entered phone number, ensuring that the system only processes valid numbers.
- To identify the time zone of the phone number's location [6], the system applies the timezone detection technique using timezone. timezones for number, providing the user with accurate timezone information.
- For obtaining the geographical coordinates, the geocoding technique is used with the geopy library and Nominatim service to convert the detected location into latitude and longitude values.

### 2.4 Tools:

To develop the Phone Number Tracker system [15], the following tools and libraries were used:

- **Python:**
  - The core programming language used for implementing the application logic [6], GUI, and integration with external libraries.
  - Provides simplicity and flexibility for rapid development.
- **Tkinter:**
  - Used to create the Graphical User Interface (GUI) of the system.
  - Helps in designing input fields, labels, buttons, and layout management for user interaction.
  - Makes the application user-friendly and visually organized.
- **Phonenumbers:**
  - A Python library used to parse [8], validate, and extract information from phone numbers.

### 2.5 Methods:

- The Phone Number Tracker system uses a systematic approach to extract, process, and display phone number details efficiently. The following methods are implemented in the system.
- The system first accepts a phone number input from the user using a Tkinter-based graphical interface. A dedicated Search button allows the user to initiate the tracking process easily [13].
- Once the user clicks the Search button, the system applies the phone number parsing method using the phone numbers library, which helps in validating and converting the entered number into a structured format for further processing.
- To identify the SIM operator [7], the SIM operator detection method is used with the help of the carrier.name for number function, allowing the system to display the service provider of the phone number.
- To accurately determine the geographical position of the detected location, the geolocation retrieval method using the geopy library and Nominatim is applied [19]. This method converts the detected location name into latitude and longitude, which are displayed in the GUI for the user's reference.

## III. METHODOLOGY

### 3.1 Input:

The Phone Number Tracker system requires specific inputs to process and extract relevant details effectively. The main inputs used in the system are:

- **Phone Number:**
  - The primary input for the system.
  - Entered by the user in the Tkinter Entry field within the GUI [12].
  - Can include country code and number (e.g., +1XXXXXXXXXX).
- **User Interaction (Button Click):**
  - The Search button in the GUI is used to trigger the track () function.
  - Acts as an input event to initiate data processing for the entered phone number [3].
- **Internet Connectivity:**
  - Accessing geopy's Nominatim service to retrieve latitude and longitude.

- Fetching accurate location data for the detected region.

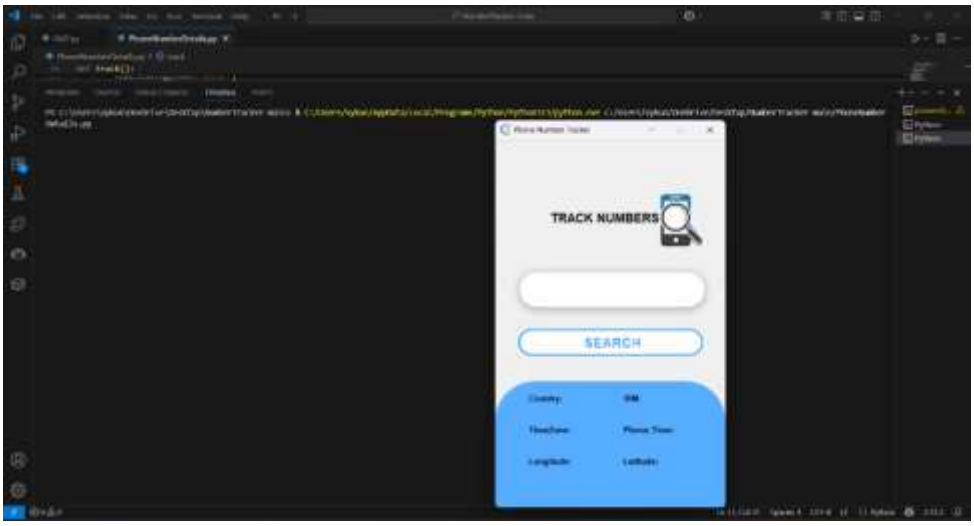


**Fig: 3 Input screen**

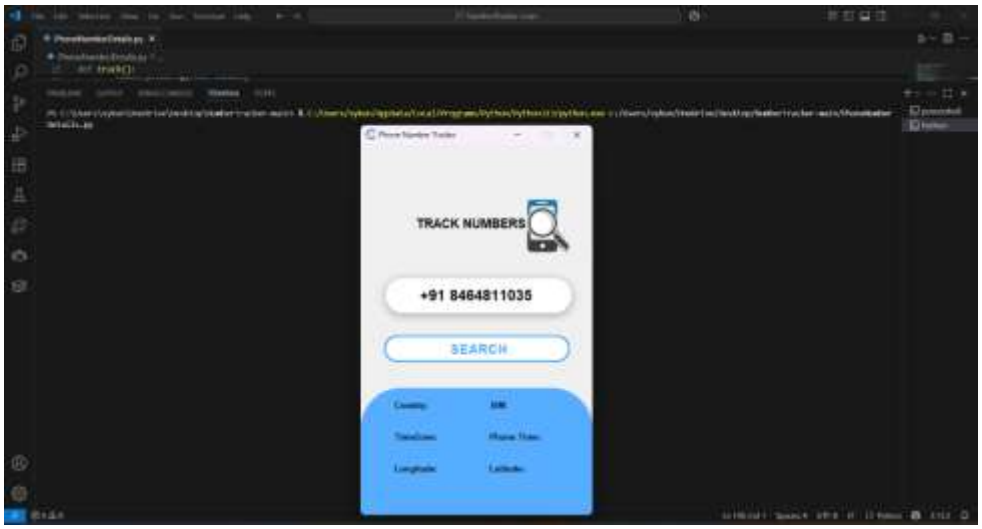


**Fig: 4 Input screen**

### 3.2 Method of Process

The Phone Number Tracker system follows a systematic method of process to extract and display phone number details efficiently and accurately [5]. The step-by-step workflow is as follows:

**1. User Input:**

- The user opens the Phone Number Tracker application.
- The user enters the phone number they wish to track in the Tkinter Entry field provided in the GUI.

**2. Phone Number Parsing:**

- The system uses the phonenumbers library to parse and validate the entered phone number.

**3. Country and SIM Carrier Extraction:**

- Using geocoder. description for number [16], the system retrieves the country or region associated with the phone number.

**4. Timezone Detection:**

- The system retrieves the timezones associated with the phone number using timezone.timezones for number.

**5. Geolocation Retrieval:**

- The system uses the geopy library with Nominatim to convert the detected location name into geographical coordinates (latitude and longitude).

**6. Local Time Calculation:**

- The system then uses pytz and datetime to calculate and display the current local time of the detected location.

**7. Display Results:**

- The extracted details, including:
    - Country/Region
    - SIM Operator
    - Timezone(s)
    - Longitude
    - Latitude
    - Local Time
- are displayed in the GUI using Tkinter Labels [23], updating dynamically as soon as data is retrieved.

**8. Error Handling:**

- If the user enters an invalid number or if there is a network failure:
    - The system uses a try-except block to handle exceptions gracefully.
    - Appropriate “Error” messages are displayed in the GUI [15], ensuring the application remains stable.

**9. Completion:**

- The user can view all details clearly on a single screen and repeat the process for other numbers if needed.

### 3.3 Output:

**Output Display Format:**

- All outputs are shown clearly on the GUI labels under respective headings:
    - Country: [Country Name]
    - SIM: [Carrier Name]
    - Timezone: [Detected Timezones]
    - Phone Time: [Current Local Time]
    - Latitude: [Latitude Value]
    - Longitude: [Longitude Value]
- Users can easily read and record the displayed outputs for analysis and reference.

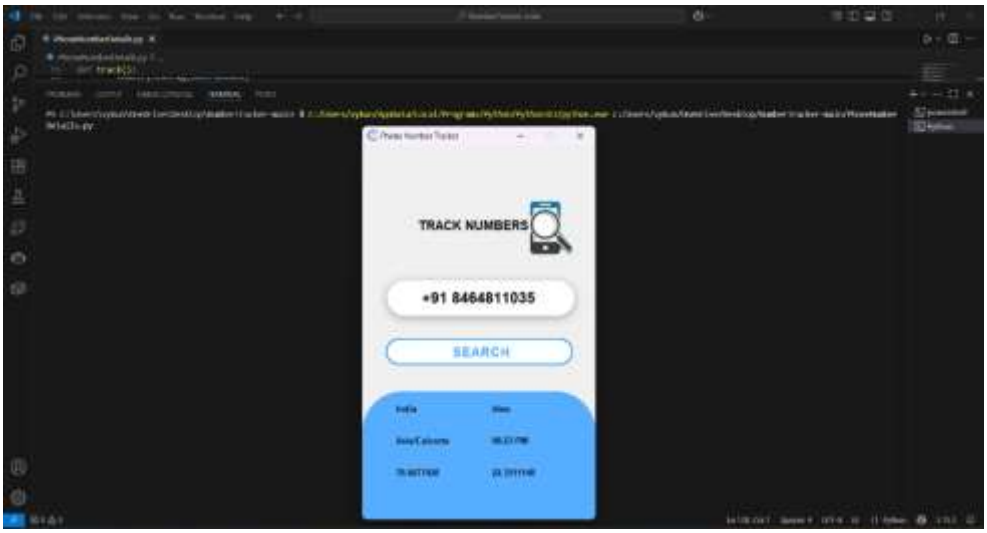


**Fig: 5 Output screen**

## IV. RESULTS

The Phone Number Tracker system was successfully developed and tested using Python and Tkinter to automate the process of tracking phone number details with a clear and user-friendly graphical interface. The results obtained after testing the system with various phone numbers are as follows:

- Country/Region Detection: The system successfully identifies and displays the country or region associated with the entered phone number.
- SIM Operator Detection: Displays the SIM carrier/operator linked with the phone number accurately.
- Timezone Detection: Retrieves and displays the timezone(s) associated with the phone number's region.
- Local Time Display: The system correctly calculates and displays the current local time based on the detected location.
- Latitude and Longitude Retrieval: The system provides precise latitude and longitude values for the detected region.
- Error Handling: When invalid numbers are entered or if there are network failures, the system displays clear error messages without crashing, ensuring stability during usage.

## V. DISCUSSIONS

The Phone Number Tracker system was designed to automate the extraction of detailed information about phone numbers using Python and Tkinter while ensuring user-friendly operation and real-time accuracy. During the development and testing phases, the system demonstrated effective performance in parsing and validating phone numbers using the phonenumbers library, ensuring that only correct formats are processed. This reduced potential errors and helped in maintaining the system's reliability. The integration of geolocation and timezone detection using the geopy, timezonefinder, and pytz libraries allowed the system to accurately identify the location, coordinates, and the local time of the phone number's region. This feature is particularly useful for users who need to know the appropriate time before making calls, enhancing the practicality of the system.

## VI. CONCLUSION

The Phone Number Tracker system was successfully developed using Python and Tkinter to automate the process of retrieving detailed information about any phone number efficiently and accurately. The system effectively integrates multiple libraries, including phonenumbers, geopy, timezonefinder, pytz, and datetime, to extract and display relevant details such as country, SIM carrier, timezone, local time, longitude, and latitude of the entered phone number. Through a user-friendly graphical interface, the system allows users to easily input a phone number and receive clear, organized outputs without requiring technical expertise. This reduces manual effort and eliminates the need to search across multiple platforms to gather information about a phone number.

## VII. FUTURE SCOPE

The Phone Number Tracker system has the potential for several enhancements in the future to increase its usability and effectiveness. Features such as spam call and fraud number detection can be added to help users identify unwanted calls. The system can also be extended to display the detected location on a map, providing a better visual understanding of the phone number's region. The application can be developed into a mobile app for Android and iOS, allowing users to track numbers conveniently on their phones. Additionally, offline tracking using preloaded country and carrier data can reduce internet dependency. By implementing these features, the Phone Number Tracker can become a more powerful, practical, and user-friendly tool for daily use and learning.

## VIII. ACKNOWLEDGEMENT

M. Satish is an enthusiastic and committed faculty member in the Department of Computer Science. As an early-career academician, he has shown strong dedication to student development through active involvement in project guidance and technical mentoring. Despite being at the beginning of his professional journey, he has effectively guided students in executing academic projects with precision and conceptual clarity. His passion for teaching, coupled with a solid understanding of core computer science principles, positions him as a promising educator and mentor. Mr. Satish continues to contribute meaningfully to the academic environment through his proactive approach to learning and student engagement.

Allanki Ooha is pursuing her final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. . With interest in Ptyhon Allanki Ooha has taken up her PG project on Number tracking main by using Python and published the paper in connection to the project under the guidance M. Satish, Assistant Professor, SVPEC.

## REFERENCES

**[1] Python Official Documentation**
https://docs.python.org/

**[2] Tkinter GUI Documentation**
https://docs.python.org/3/library/tkinter.html

**[3] phonenumbers Library Documentation**
https://github.com/daviddrysdale/python-phonenumbers

**[4] geopy Library Documentation**
https://geopy.readthedocs.io/

**[5] timezonefinder Documentation**
https://timezonefinder.readthedocs.io/

**[6] pytz Library Documentation**
https://pytz.sourceforge.net/

**[7] Python datetime Module**
https://docs.python.org/3/library/datetime.html

**[8] Nominatim API Documentation**
https://nominatim.org/release-docs/latest/api/Search/

**[9] Stack Overflow – Python Tag**
https://stackoverflow.com/questions/tagged/python

**[10] Tutorials Point Python Tutorials**
https://www.tutorialspoint.com/python/index.htm

**[11] Geeks for Geeks Python Programming**
https://www.geeksforgeeks.org/python-programming-language/

**[12] Real Python – Python Tutorials**
https://realpython.com/

**[13] PyPI – Python Package Index for library installation**
https://pypi.org/

**[14] W3Schools Python Reference**
https://www.w3schools.com/python/

**[15] GitHub Discussions on phonenumbers issues**
https://github.com/daviddrysdale/python-phonenumbers/issues

**[16] PyCharm IDE Documentation**
https://www.jetbrains.com/pycharm/documentation/

**[17] VS Code Python Extension Guide**
https://code.visualstudio.com/docs/python/python-tutorial

**[18] IEEE Xplore – Mobile Tracking Papers**
https://ieeexplore.ieee.org/

**[19] MDN Web Docs – APIs and HTTP concepts**
https://developer.mozilla.org/

**[20] JSON.org – Understanding JSON used in API parsing**
https://www.json.org/