



TRUECALLER API

PROJECT REPORT

Submitted by

Akshat Sharma

Bhav Goyal

Shrey Sharma

Kritika Gaur

Koyna Pandey

Purshottam Sharma

Under the guidance of

Dr. Rakshit Tandon

Gurugram Police Cyber Security Summer Internship

GPCSSI 2022

TABLE OF CONTENTS

CONTENT	Page No.
Declaration	03
Certificate	04
Acknowledgment	05
Abstract	06
Introduction	07-08
Description of Tools	09-11
Coding Part	12-15
Conclusion	16
References	17

DECLARATION

We certify that the work contained in this report is original and has been done by us under the guidance of **Dr. Rakshit Tandon.**

- a. The work has not been submitted to any other sources.
- b. We have followed the guidelines provided by this Internship.

Name and Signature of Project Team Members:

Sr. No.	Name of Team Member	Signature
1.	Akshat Sharma	
2.	Bhav Goyal	
3.	Shrey Sharma	
4.	Kritika Gaur	
5.	Koyna Pandey	
6.	Purshottam Sharma	

BONAFIDE CERTIFICATE

This is to certify that the project report titled “Truecaller API” is the bonafide work of Team 9 (members –Akshat, Bhav, Shrey, Kritika, Koyna, Purshottam) under the guidance of Dr. Rakshit Tondon, submitted as a final project in GPCSSI 2022, and is their original work.

Signature

Dr. Rakshit Tandon

Head Trainings - Cyber Peace Foundation, Advisor-Cybercrime,
Director/CoFounder-HackerShala,

Consultant – IAMAI

ACKNOWLEDGEMENT

The success and result of this project requires a lot of guidance and endorsement from many people and we are fortunate to get all of these throughout our entire internship project.

We were able to accomplish this project only with such assistance and supervision; therefore, we will never forget to thank them.

We respect and thank **Dr. Rakshit Tandon** who allowed us to work on this specific project at the **10th Gurugram Police Cyber Security Summer Internship 2022** and gave us all the support and guidance that motivated us to complete the project correctly.

Despite being busy dealing with corporate matters we are very grateful to him for such admonition.

In addition, we would like to express our heartfelt gratitude to all the **Commissionerate police staff** for their timely support.

Yours Sincerely,

Bhav Goyal
Akshat Sharma
Shrey Sharma
Kritika Gaur
Koyna Pandey
Purshottam Sharma

ABSTRACT

Truecaller is a mobile application with over 200 million unique users worldwide. Every day truecaller stores over 1 billion rows of new data which they use to analyze and improve their product. The app works as a kind of community where all people with a smartphone can enter their information to enhance the truecaller's database, view who's the unknown caller calling them, and flag many spam and fraud calls to make the user cautious of the caller.

The goal of this project is to help the Law Enforcement Agencies to identify possible suspects from a large no. call record in a CDR (Call Details Record). This project helps in taking a large set of numbers from cdr to be run through truecaller API, then the results are taken from the API response and saved in an excel file.

In this project, the above-mentioned process was first programmed in a python script. Then the python script was then included in a flask server, HTML, and CSS. The flask server was used to handle GET/POST requests, and the execution of python scripts to handle the API Request and response.

INTRODUCTION

TrueCaller is a smartphone application that has features of caller-identification, call-blocking, flash-messaging, call-recording, Chat & Voice by using the Internet. It requires users to provide a standard cellular mobile number for registering with the service. The app is available for Android and IOS.

The goal of this project is to help the Law Enforcement Agencies to identify possible suspects from a large no. call record in a CDR (Call Details Record). This project helps in taking a large set of numbers from cdr to be run through truecaller API, then the results are taken from the API response and saved in an excel file.

How will TRAI's caller ID work?

In theory, TRAI's caller ID will work just the way Truecaller's service works. If you get a call from an unknown number, you will be able to see KYC-based information about the person calling you on your screen, if you have the program turned on. TRAI has said that it is not going to enforce its caller ID on users without their permission, it is going to be consent-based and voluntary, and users will also have the authority to decide whether or not to have their names displayed, as per reports, to help keep spam away.

The information you will see is based solely on KYC data that users have shared with network providers while getting the number/connection they are calling from and this data is going to come from ID proofs like Aadhaar Card, PAN Card, passports, whatever the person has submitted as ID proof.

The aim is to help users evade spam calls and phishing attacks, and TRAI's caller ID is expected to make things more 'accurate' and 'transparent'. While TRAI's method of identification will help ensure that at least the names of users are accurate, the problems with this system lie elsewhere. We'll get to that soon.

How is this different from Truecaller?

What Truecaller already does, and what TRAI's service intends to do is the same - identify the callers for you so you can evade spam calls. Now, Truecaller identifies users based on crowdsourced data. So, for example, if you receive a call from an unknown number and you accept it via the Truecaller app or search for it on the app later, the information you will see has been culled from other Truecaller users who might have the number saved in their contacts. Mind you, Truecaller cannot look through your contacts unless you permit it.

Truecaller also allows users to suggest names for unknown callers when they receive a call for the first time. You can also report a number and block it on the platform and it will be acknowledged and marked as such based on the number of reports it gets - that is essentially how

spam and phishing calls are flagged off on Truecaller.

Now, TRAI's method of sourcing information about a caller is devoid of any human intervention. This is both a good thing and a bad thing. Any person could use their IDs to get several connections, one of which might be used for spam/phishing calls. Now, you would not know what a call is regarding unless you specifically entertain it. It is also not clear whether there will be an option to flag calls on TRAI's platform, as there is on Truecaller, or suggest a name change.

For all practical purposes, Truecaller offers more on its platform when it comes to dealing with unwanted calls. Especially if you are an Android user, Truecaller is all the more handy and effective because of the less complicated permissions issue on the platform. So, you don't need TRAI's service, unless there is a way to make the two services work together effectively.

Truecaller has "welcomed" TRAI's proposal "in the mission to make communication safe and efficient" and "appreciated" the move as a supportive one to help end the "menace of spam and scam calls". But essentially, Truecaller has nothing to worry about. TRAI's new mechanism, if included with what Truecaller is very efficiently already doing, should help clean up the data that's available to crowdsourcing apps like itself with KYC linkages.

By itself, TRAI's caller ID is perhaps going to be only half as effective as it can be if it works in tandem with companies like Truecaller.

DESCRIPTION OF TOOL

- **TruecallerJs:** An NPM Package used as an unofficial Truecaller API for phone number searching and the response was then requested in an XML format using the –XML attribute in the command used for the API request.
- **IDE:**
 - Visual Studio Code
- **Language:**
 - Python
 - HTML
 - CSS
- **Libraries:**
 - Flask
 - Flask_login
 - __init__
 - Werkzeug.utils
 - Pandas
 - Xml.dom
 - Openpyxl
 - Cgi
 - cgitb

Login

☐ Remember me

Login

Sign Up

Email

Name

Password

Sign Up

Welcome, Bhav!

select xlsx file with numbers to be searched.

[Browse...](#) No file selected.

[view](#) [submit](#)

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	9212433299	Bhav Goyal	919212433299	Airtel	Delhi	NaN	NaN	NaN	<class 'int'>
1	8587873172	Akshat Sharma	918587873172	Vodafone Idea	Delhi	NaN	NaN	NaN	<class 'int'>
2	8592952341	Shrey	918592952341	Vodafone Idea	Kerala	NaN	NaN	NaN	<class 'int'>

	A	B	C	D	E	F	G	H	I	J
1										
2	9212433299	Bhav Goyal	+919212433299	Airtel	Delhi	null	null	null	<class 'int'>	
3	8587873172	Akshat Sharma	+918587873172	Vodafone Idea	Delhi	null	null	null	<class 'int'>	
4	8592952341	Shrey	+918592952341	Vodafone Idea	Kerala	null	null	null	<class 'int'>	
5										
6										
7										

CODING PART

```
from flask import Blueprint, render_template, flash
from flask_login import login_required, current_user
from __init__ import create_app, db
from werkzeug.utils import secure_filename
import pandas as pd
from xml.dom import minidom
import openpyxl as xl
import cgi, os
import cgi; cgi.enable()
from flask import Flask, render_template, redirect, url_for, request
# our main blueprint
main = Blueprint('main', __name__)

@main.route('/') # home page that return 'index'
def index():
    return render_template('index.html')

@main.route('/profile') # profile page that return 'profile'
@login_required
def profile():
    return render_template('profile.html', name=current_user.name)

@main.route('/data', methods = ['GET', 'POST'])

def data():
    if request.method == 'POST' and request.form['submit'] == 'submit':
        f = request.files['upload']
        f.save(secure_filename(f.filename))
        print(f.filename)
        wb = xl.load_workbook(f.filename)
        sheet = wb['Sheet1']

        def cellEntry(row, column, attribute ):
            new_cell = sheet.cell(row, column)
            new_cell.value = str(attribute)
            #enters cell data in specific cell, attribute is the data entered
        def trial():
            #takes input str and the row where the corresponding data has to be
            printed
```

```

for row in range(2, sheet.max_row+1):
    data = sheet.cell(int(row), 1)
    print(str(type(data.value)))
    if(str(type(data.value)) != "<class 'int'>"):
        cellEntry(row, 2 , "invalid character")
    else:
        if(len(str(data.value)) != 10):
            cellEntry(row, 2 , "invalid number")
        else:

            # data = whatever is in the sheet at row,column specified...in
this case it would be our phoennumber
            cmd = "truecallerjs -s " + str(data.value) + " --xml | sed
'id'"

            input = os.popen(cmd).read()
            # os lib used to run truecallerjs command with a python loop
for every number from excel

            p3: None = minidom.parseString(input)
            #parsing xml into dom
            print(type(p3))
            #check type of p3 LINE NOT NEEDED
            def getTagElement(tag_name, row , column):
                #gets tag name and prints the tag data in the cell
specified

                try:

                    if (tag_name=='id'):
                        element = p3.getElementsByTagName(tag_name)
                        print(element[1].firstChild.data)
                        cellEntry(row, column,
element[1].firstChild.data)

                    else:
                        #if tag is not id print element [0] ie. the first
instance of the tag

                        element = p3.getElementsByTagName(tag_name)
                        #gets of data all instances of the tag in an array
element

                        print(element[0].firstChild.data)
                        #prints the first data at element[0]
                        cellEntry(row, column,
element[0].firstChild.data)

```

```

#enters it in the excel sheet

    except IndexError:
        cellEntry(row, column, "null")
    try:
        if
(p3.getElementsByTagName("errorResp")[0].firstChild.data == "Request failed with status
code 429"):
        cellEntry(row,2, "you are logged out")
    except IndexError :

        cellEntry(row, 9 , type(data.value))
        getTagElement("name" , row , 2)
        getTagElement("e164Format" ,row, 3)
        getTagElement("carrier" ,row, 4)
        getTagElement("city" , row, 5)
        getTagElement("image" , row , 6)
        getTagElement("id" , row , 7)
        getTagElement("caption" , row, 8)

        # if (p3.getElementsByTagName("errorResp")[0].firstChild.data
=="Request failed with status code 429"):
        #     cellEntry(row,2, "you are logged out")
        # else:
        #     cellEntry(row, 9 , type(data.value))
        #     getTagElement("name" , row , 2)
        #     getTagElement("e164Format" ,row, 3)
        #     getTagElement("carrier" ,row, 4)
        #     getTagElement("city" , row, 5)
        #     getTagElement("image" , row , 6)
        #     getTagElement("id" , row , 7)
        #     getTagElement("caption" , row, 8)
        #calling get element function defined above

trial()
wb.save('result.xlsx')
data= pd.read_excel('result.xlsx')
return render_template('data.html' , data =data.to_html())

```

```

















app = create_app() # we initialize our flask app using the __init__.py function
if __name__ == '__main__':
    db.create_all(app=create_app()) # create the SQLite database
    app.run(debug=True) # run the flask app on debug mode







```

```

w3rew01f@DESKTOP-A4KSJMN:/mnt/c/users/goyal/Downloads/Flask_truecaller_final_UI/Create_LoginPage-main$ python3 main.py
* Serving Flask app '__init__' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 529-344-625

```

	.idea	15-07-2022 11:15 AM	File folder	
	__pycache__	15-07-2022 08:15 PM	File folder	
	templates	15-07-2022 11:12 AM	File folder	
	utils	07-05-2022 02:24 AM	File folder	
	.DS_Store	15-07-2022 11:30 AM	DS_STORE File	7 KB
	.gitignore	07-05-2022 02:24 AM	Git Ignore Source File	2 KB
	__init__.py	07-05-2022 02:24 AM	Python Source File	2 KB
	auth.py	07-05-2022 02:24 AM	Python Source File	4 KB
	db.sqlite	15-07-2022 12:49 PM	SQLITE File	12 KB
	LICENSE	07-05-2022 02:24 AM	File	2 KB
	main.py	15-07-2022 01:19 PM	Python Source File	6 KB
	models.py	07-05-2022 02:24 AM	Python Source File	1 KB
	README.md	07-05-2022 02:24 AM	Markdown Source File	1 KB
	requirements.txt	07-05-2022 02:24 AM	Text Document	1 KB
	result.xlsx	15-07-2022 01:17 PM	Microsoft Excel Work...	6 KB
	transcript.xlsx	15-07-2022 01:17 PM	Microsoft Excel Work...	8 KB

	base.html	15-07-2022 01:07 PM	Brave HTML Docume...	2 KB
	data.html	14-07-2022 11:19 PM	Brave HTML Docume...	1 KB
	index.html	15-07-2022 01:09 PM	Brave HTML Docume...	1 KB
	login.html	07-05-2022 02:24 AM	Brave HTML Docume...	2 KB
	profile.html	15-07-2022 11:11 AM	Brave HTML Docume...	1 KB
	signup.html	07-05-2022 02:24 AM	Brave HTML Docume...	2 KB

CONCLUSION

Truecaller is **making your dialing experience smarter and easier to use**. By using Truecaller as your default dialer you'll be able to quickly and easily access all your existing contacts, see more relevant information about them and also identify unknown numbers and block unwanted callers.

Truecaller made a significant impact on someone's life. These range from **fighting crime with law enforcement agencies, catching a harasser, preventing financial fraud - and even saving lives**.

REFERENCE

1. ["Truecaller - Android-apps op Google Play"](#). Retrieved 10 October 2012.
2. ["Truecaller - worldwide number search and spam filter for iPhone, iPod touch, and iPad"](#). *iTunes App Store*. 27 September 2012.
3. <https://github.com/sumithemmadi/truecallerjs>
4. <https://www.w3schools.com/html/default.asp>
5. <https://www.w3schools.com/w3css/default.asp>
6. <https://www.w3schools.com/python/default.asp>
7. <https://www.tutorialspoint.com/flask/index.htm>