CODING:

Backend:

MServer.py

```
from typing import Union

from tld import get_tld

from urllib.parse import urlparse

import tldextract

import re

import os

import joblib

import pandas as pd

import uvicorn

from fastapi import FastAPI, File, UploadFile, Form

import aiofiles

import os

import time

import math

import validators

app = FastAPI()

@app.get("/")

def read_root():
```

```
    return {"Hello": "World1111"}
def fd_length(url):
    urlpath = urlparse(url).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0
def tld_length(tld):
    try:
        return len(tld)
    except:
        return -1
def digit_count(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
def letter_count(url):
    letters = 0
    for i in url:
        if i.isalpha():
```

```
        letters = letters + 1
    return letters
def no_of_dir(url):
    urldir = urlparse(url).path
    return urldir.count('/')
def having_ip_address(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4 with port
        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'  # IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}|'
        '([0-9]+(?:\.[0-9]+){3}:[0-9]+)|'
        '((?:(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d\d|\d)(?:\/\d{1,2})?)', url)  # Ipv6
    if match:
        return -1
    else:
        return 1
```

```
def shortening_service(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                  'tr\.im|link\.zip\.net',
                  url)
    if match:
        return -1
    else:
        return 1
def feature_engineering(ds):
    ds['url_length'] = ds['url'].apply(lambda i: len(str(i)))
    ds['hostname_length'] = ds['url'].apply(lambda i: len(urlparse(i).netloc))
    ds['path_length'] = ds['url'].apply(lambda i: len(urlparse(i).path))
    ds['fd_length'] = ds['url'].apply(lambda i: fd_length(i))
```

```
    ds['tld'] = ds['url'].apply(lambda i: get_tld(i, fail_silently=True))
    ds['tld_length'] = ds['tld'].apply(lambda i: tld_length(i))
    ds = ds.drop("tld", 1)
    ds['count_dash'] = ds['url'].apply(lambda i: i.count('-'))
    ds['count_at'] = ds['url'].apply(lambda i: i.count('@'))
    ds['count_question_mark'] = ds['url'].apply(lambda i: i.count('?'))
    ds['count_perc'] = ds['url'].apply(lambda i: i.count('%'))
    ds['count_dot'] = ds['url'].apply(lambda i: i.count('.'))
    ds['count_equals'] = ds['url'].apply(lambda i: i.count('='))
    ds['count_http'] = ds['url'].apply(lambda i: i.count('http'))
    ds['count_https'] = ds['url'].apply(lambda i: i.count('https'))
    ds['count_www'] = ds['url'].apply(lambda i: i.count('www'))
    ds['count_digits'] = ds['url'].apply(lambda i: digit_count(i))
    ds['count_letters'] = ds['url'].apply(lambda i: letter_count(i))
    ds['count_dir'] = ds['url'].apply(lambda i: no_of_dir(i))
    ds['use_of_ip'] = ds['url'].apply(lambda i: having_ip_address(i))
    ds['short_url'] = ds['url'].apply(lambda i: shortening_service(i))
    return ds
@app.get("/GetURL",response_model=dict,
response_model_exclude_unset=True)
async def get_url(url: str):
     print("get_url executed")
```

```
    murl = url
    if not validators.url(murl):
        return {"Prediction": "Not a Valid URL"}
    rf_model_1=joblib.load("./Detection/RF-Malicious-URL-Detect-Model.joblib")
    single_df = pd.DataFrame([murl], columns=['url'])
    single_df = feature_engineering(single_df)
    print(single_df)
    single_df_html = single_df.T.to_html(header=False, classes="table table-striped, table-bordered")
    single_df.drop(['url'], inplace=True, axis=1)
    single_df_y_pred = rf_model_1.predict(single_df.to_numpy())
    print(single_df_y_pred[0])
    pred = single_df_y_pred[0]
    malicious_type = {0: "Benign", 1: "defacement", 2: "phishing", 3: "malware"}
    prediction_type = malicious_type[pred]
    print(prediction_type)
    result = {}
    print("url_length" , single_df.iloc[0]["url_length"])
    result['murl'] = murl
    result['Prediction'] = prediction_type
    result["url_length"] = str(single_df.iloc[0]["url_length"])
```

```
    result["hostname_length"] = str(single_df.iloc[0]["hostname_length"])
    result["path_length"] = str(single_df.iloc[0]["path_length"])
    result["fd_length"] = str(single_df.iloc[0]["fd_length"])
    result["tld_length"] = str(single_df.iloc[0]["tld_length"])
    result["count_dash"] = str(single_df.iloc[0]["count_dash"])
    result["count_at"] = str(single_df.iloc[0]["count_at"])
    result["count_question_mark"]=
str(single_df.iloc[0]["count_question_mark"])
    result["count_perc"] = str(single_df.iloc[0]["count_perc"])
    result["count_dot"] = str(single_df.iloc[0]["count_dot"])
    result["count_equals"] = str(single_df.iloc[0]["count_equals"])
    result["count_http"] = str(single_df.iloc[0]["count_https"])
    result["count_www"] = str(single_df.iloc[0]["count_www"])
    result["count_digits"] = str(single_df.iloc[0]["count_digits"])
    result["count_letters"] = str(single_df.iloc[0]["count_letters"])
    result["count_dir"] = str(single_df.iloc[0]["count_dir"])
    result["use_of_ip"] = str(single_df.iloc[0]["use_of_ip"])
    result["short_url"] = str(single_df.iloc[0]["short_url"])
    return result

if _name_ == '_main_':
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Frontend:

mainactivity.java

```
package com.example.malicious_url_detection;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.content.ContextCompat;

import android.Manifest;

import android.app.AlertDialog;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.database.Cursor;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.provider.OpenableColumns;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import java.io.ByteArrayOutputStream;

import java.io.File;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import com.android.volley.AuthFailureError;
import com.android.volley.NetworkResponse;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
public class MainActivity extends AppCompatActivity {

    public String ipAddr = "10.0.2.2:8000";
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        System.out.println("111111111111111111111");
        findViewById(R.id.buttonProcess).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

System.out.println("OnClickCalled"+Manifest.permission.WRITE_EXTERNA
L_STORAGE);
                if ((ContextCompat.checkSelfPermission(getApplicationContext(),
                        Manifest.permission.WRITE_EXTERNAL_STORAGE)          !=
PackageManager.PERMISSION_GRANTED) &&
                        (ContextCompat.checkSelfPermission(getApplicationContext(),
                                Manifest.permission.READ_EXTERNAL_STORAGE)    !=
PackageManager.PERMISSION_GRANTED)) {
                    System.out.println("If Part Called");
                    sendUrlToServer();
                } else {
                    System.out.println("Else Part Called");
                }
            }
```

```
        });
    }
    public void sendUrlToServer(){
        RequestQueue queue = Volley.newRequestQueue(this);
        EditText editurl = findViewById(R.id.editurl);
        String url ="http://10.0.2.2:8000/GetURL?url="+editurl.getText();
        //String url ="http://192.168.17.193:8000/GetURL?url="+editurl.getText();
        MainActivity amain =this;
// Request a string response from the provided URL.
        JsonObjectRequeststringRequest=new
JsonObjectRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        System.out.println(response);
                        Intent intent = new Intent(amain, MaliciousOutcome.class);
                        intent.putExtra("MaliciousOutcomeData", response.toString());
                        startActivity(intent);
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
```

```
            System.out.println("That didn't work!"+" ");
        }
    });
    queue.add(stringRequest);
  }
}
```

maliciousoutcome.java

```
package com.example.malicious_url_detection;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import org.json.JSONException;
import org.json.JSONObject;
public class MaliciousOutcome extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_malicious_outcome);
    MaliciousOutcome actvy = this;
    Bundle extras = getIntent().getExtras();
```

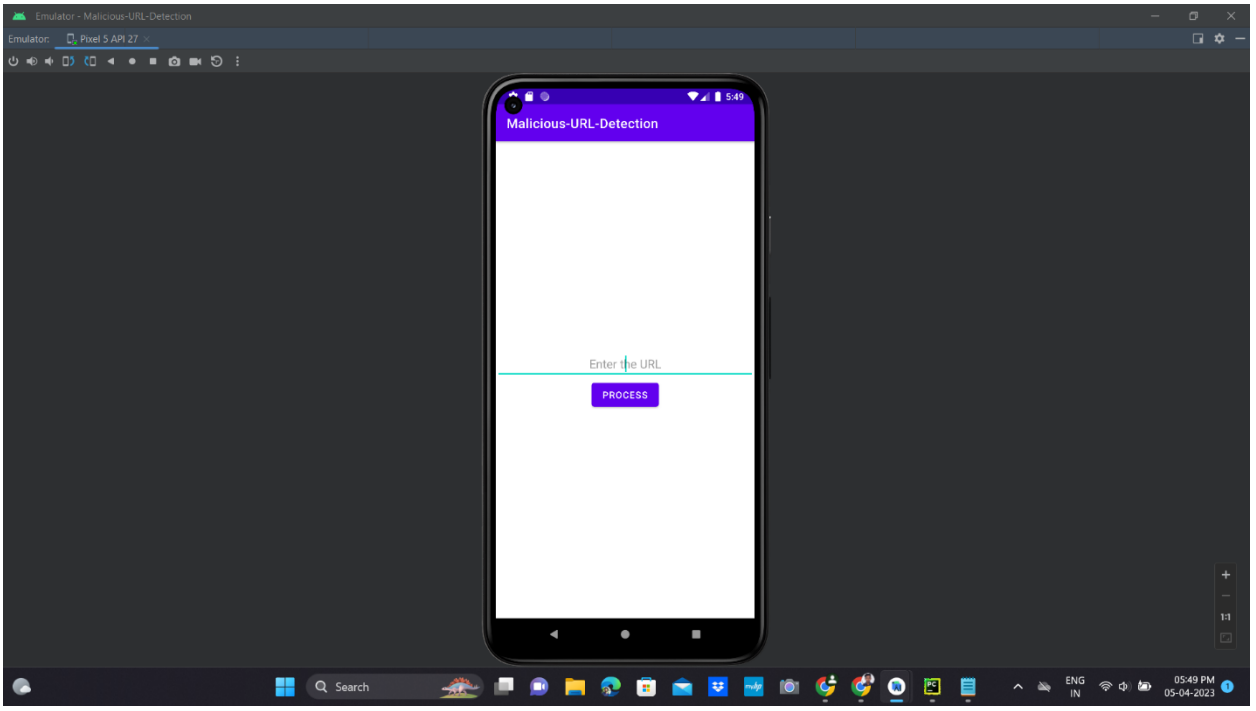
```
        JSONObject jsonObject = null;
        try {
            jsonObject=new
JSONObject(extras.getString("MaliciousOutcomeData"));
            System.out.print(jsonObject.getString("url_length"));
            TextView turl = findViewById(R.id.turl);
            turl.setText(jsonObject.getString("murl"));
            TextView tprediction = findViewById(R.id.tprediction);
            tprediction.setText(jsonObject.getString("Prediction"));
            TextView turllength = findViewById(R.id.turllength);
            turllength.setText(jsonObject.getString("url_length"));
            TextView thostnamelength = findViewById(R.id.thostnamelength);
            thostnamelength.setText(jsonObject.getString("hostname_length"));
            TextView tpathlength = findViewById(R.id.tpathlength);
            tpathlength.setText(jsonObject.getString("path_length"));
            TextView tfdlength = findViewById(R.id.tfdlength);
            tfdlength.setText(jsonObject.getString("fd_length"));
            TextView ttldlength = findViewById(R.id.ttldlength);
            ttldlength.setText(jsonObject.getString("tld_length"));
            TextView tcountdash = findViewById(R.id.tcountdash);
            tcountdash.setText(jsonObject.getString("count_dash"));
```

```
        TextView tcountat = findViewById(R.id.tcountat);
        tcountat.setText(jsonObject.getString("count_at"));
        TextView tcountquestionmark = findViewById(R.id.tcountquestionmark);
    tcountquestionmark.setText(jsonObject.getString("count_question_mark"));
        TextView tcountperc = findViewById(R.id.tcountperc);
        tcountperc.setText(jsonObject.getString("count_perc"));
        TextView tcountdot = findViewById(R.id.tcountdot);
        tcountdot.setText(jsonObject.getString("count_dot"));
        TextView tcountequals = findViewById(R.id.tcountequals);
        tcountequals.setText(jsonObject.getString("count_equals"));
        TextView tcounthttp = findViewById(R.id.tcounthttp);
        tcounthttp.setText(jsonObject.getString("count_http"));
        TextView tcountwww = findViewById(R.id.tcountwww);
        tcountwww.setText(jsonObject.getString("count_www"));
        TextView tcountdigits = findViewById(R.id.tcountdigits);
        tcountdigits.setText(jsonObject.getString("count_digits"));
        TextView tcountletters = findViewById(R.id.tcountletters);
        tcountletters.setText(jsonObject.getString("count_letters"));
        TextView tcountdir = findViewById(R.id.tcountdir);
        tcountdir.setText(jsonObject.getString("count_dir"));
```

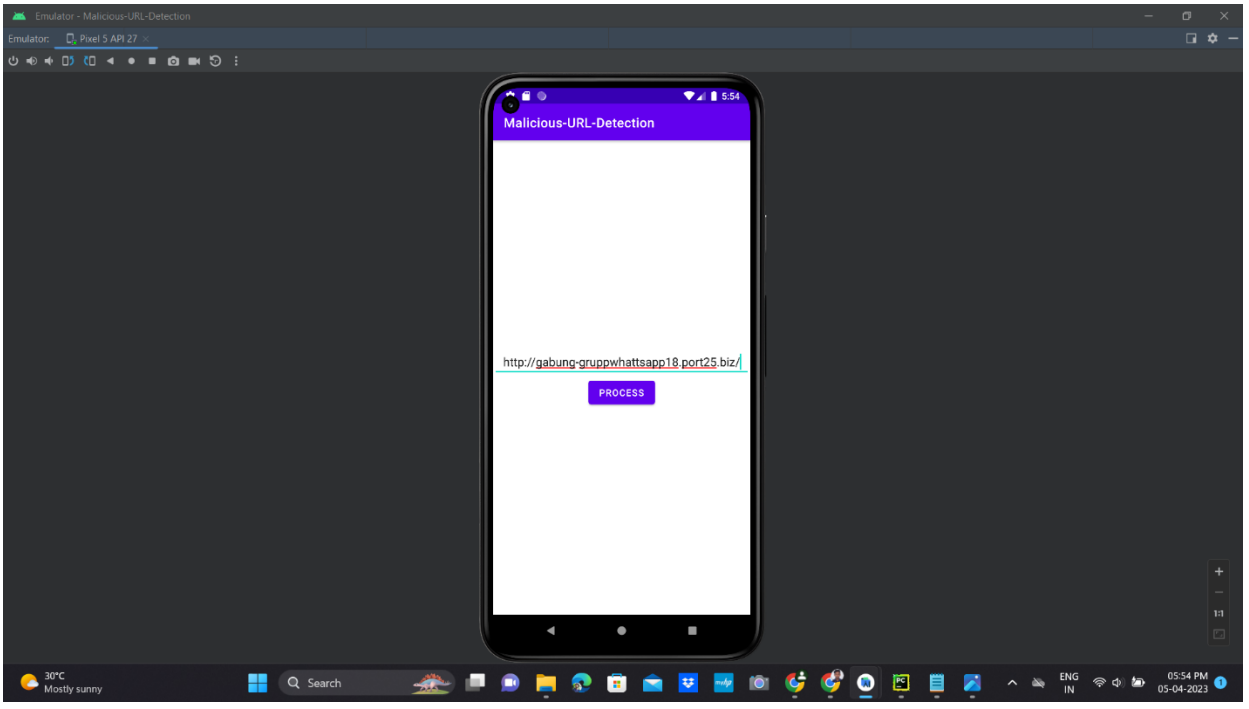
```
            TextView tuseofip = findViewById(R.id.tuseofip);
            tuseofip.setText(jsonObject.getString("use_of_ip"));
            TextView tshorturl = findViewById(R.id.tshorturl);
            tshorturl.setText(jsonObject.getString("short_url"));
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        findViewById(R.id.Close).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(actvy, MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

SAMPLE OUTPUTS:

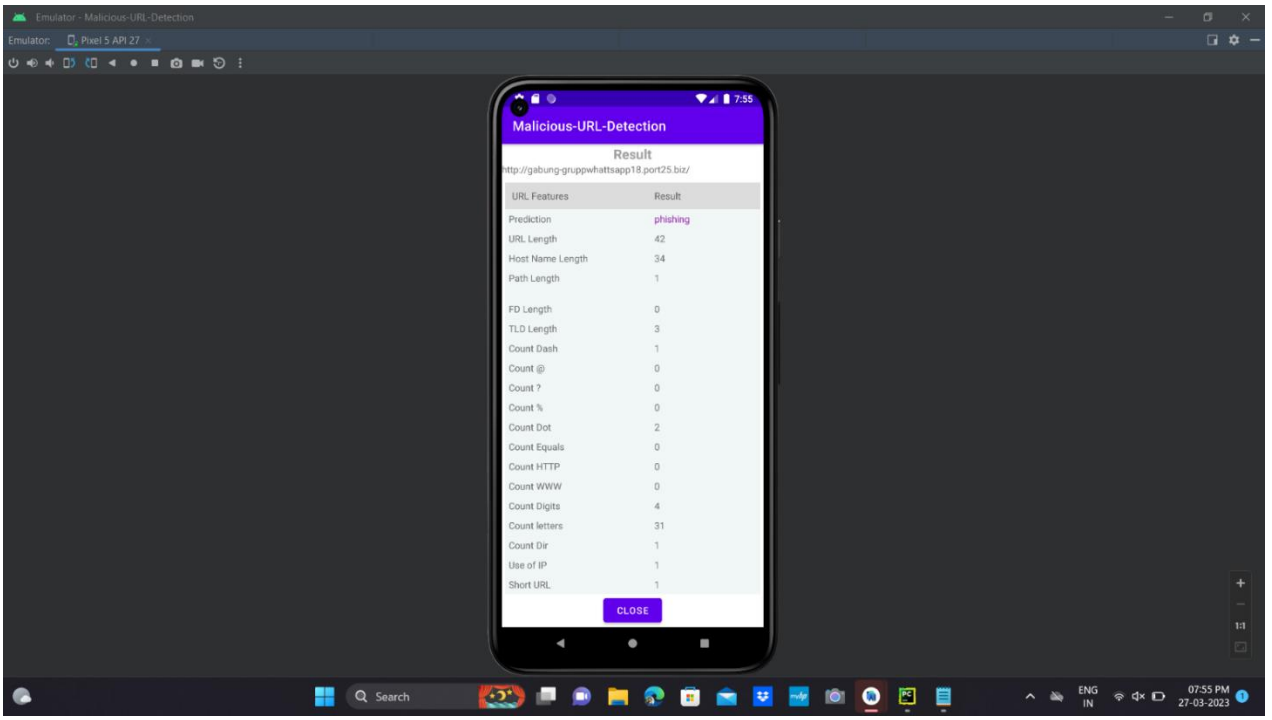
The pictures that are attached below represent the output of our project:



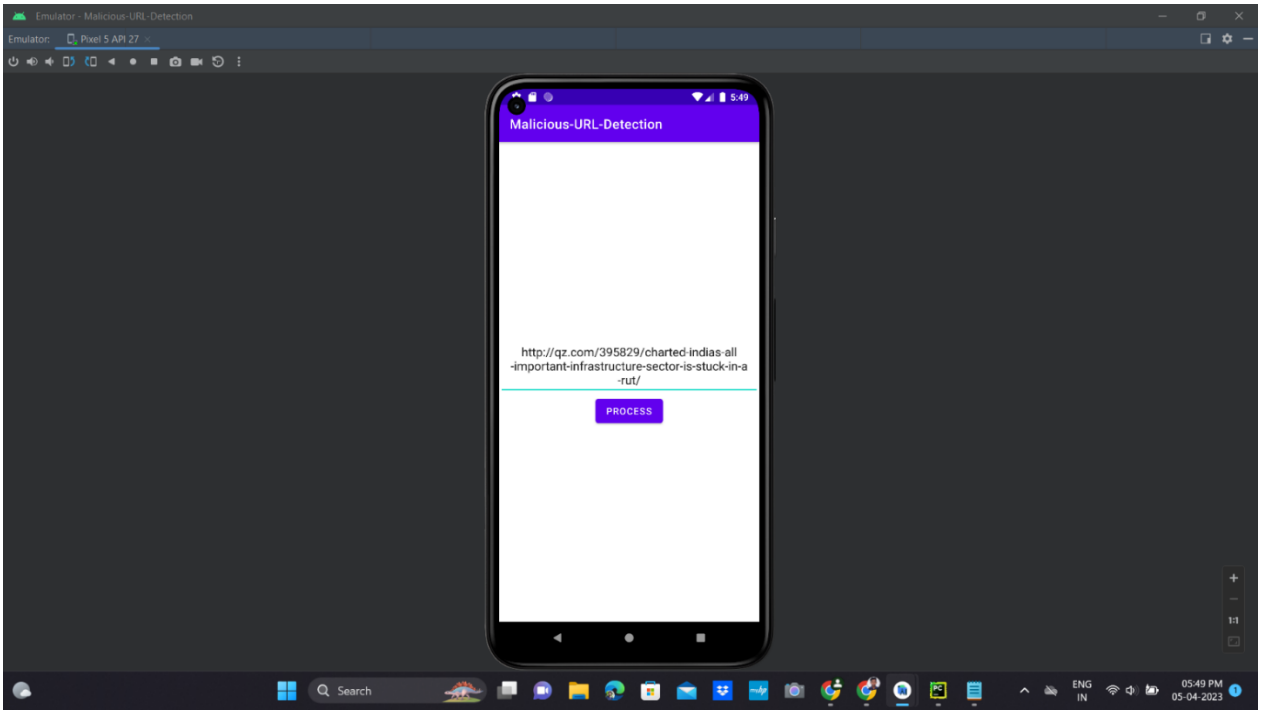
Getting the URL from the user



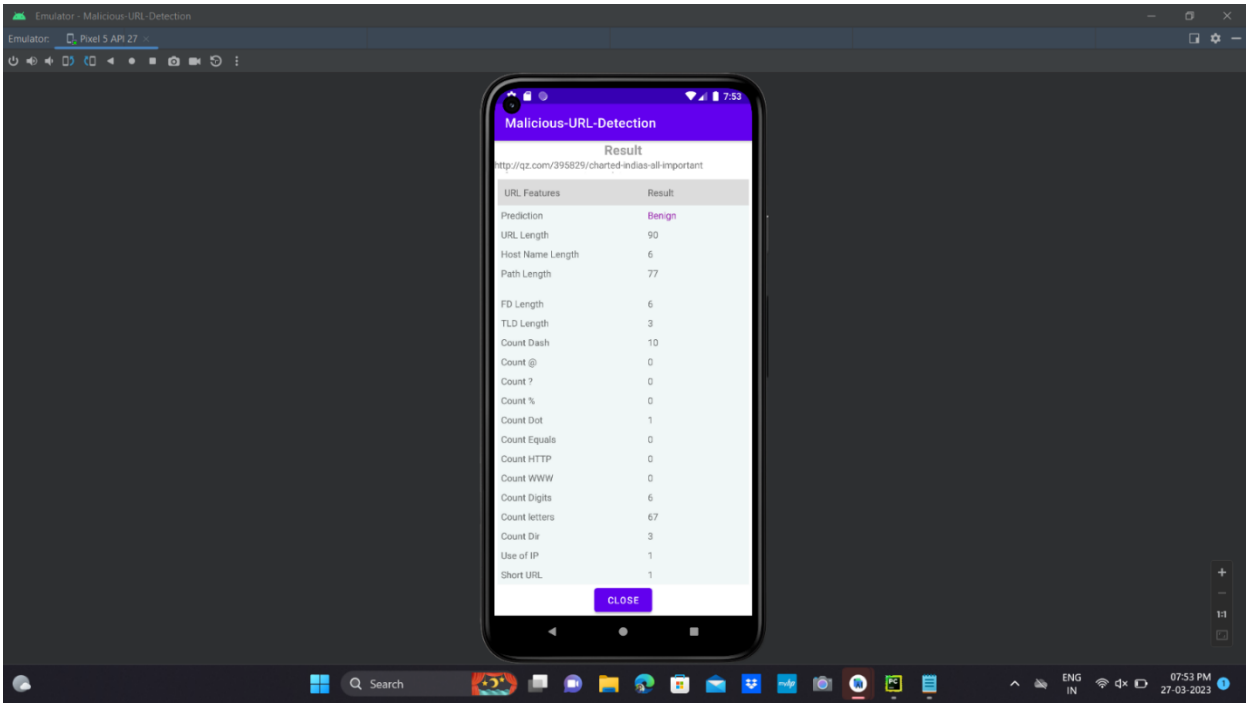
Paste the URL 1 in the textbox



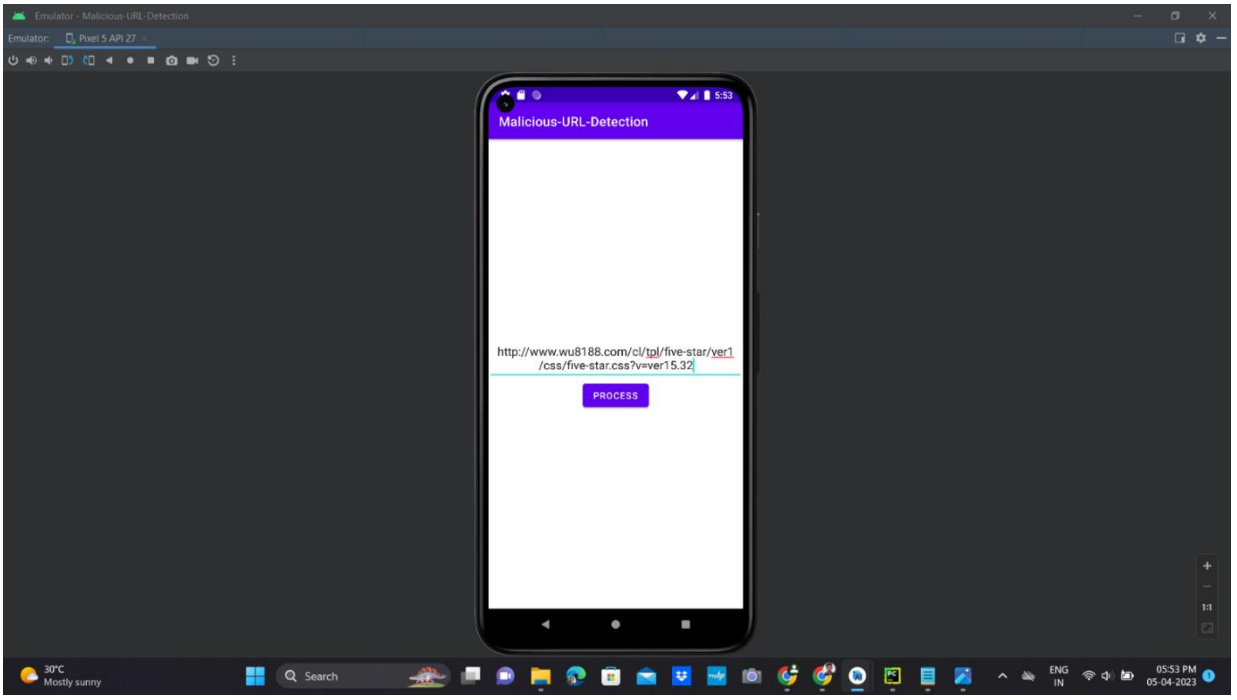
The URL 1 type is displayed



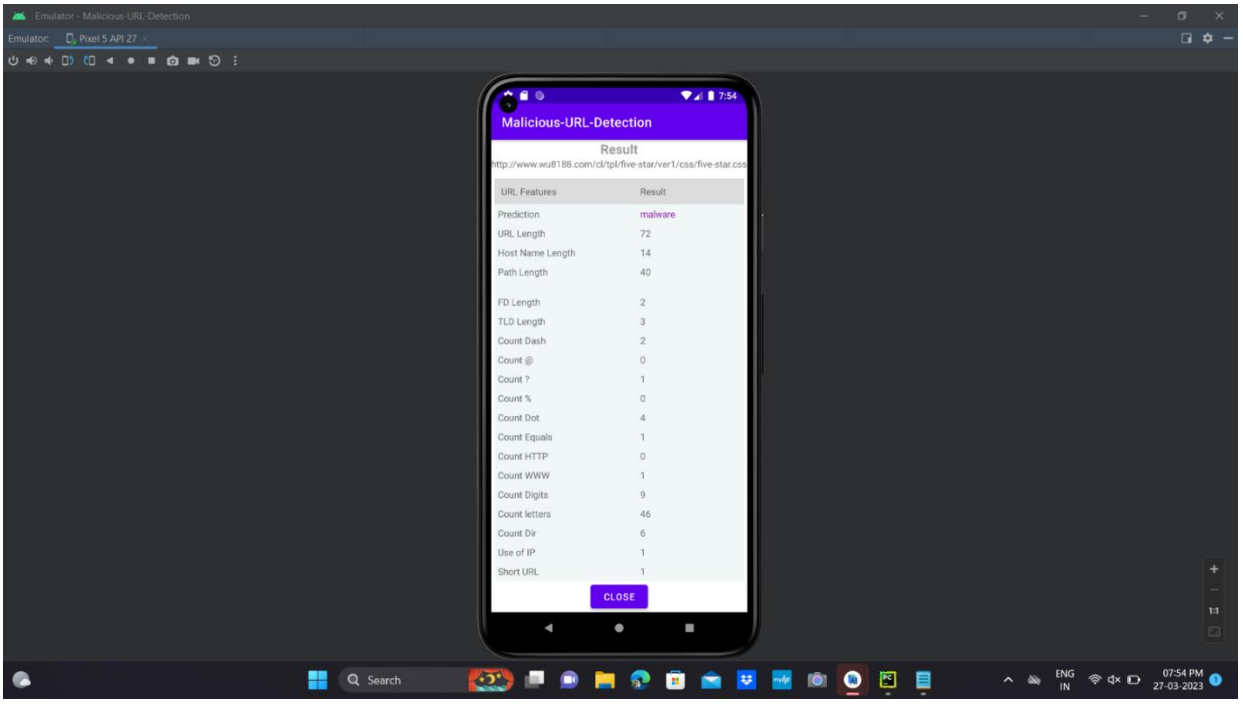
Paste the URL 2 in the textbox



The URL 2 type is displayed



Paste the URL 3 in the textbox



The URL 3 type is displayed