Face Recognition using API

In this session we'll learn how to develop api and use it's features in Android Studio.
Flow of session:

A. Api development.
B. Hosting api on Local Server.
C. Using api in android studio.

A. Api Development:

An api is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. In this session we are going to use python for developing api.

Api Development in Python:

Modules required:
1. Flask
2. Jsonpickle
3. numpy
4. opencv
5. face_recognition

Python IDE version: Python 3.6

For understanding structure of Api in python let's try a simple code:

```
from flask import Flask
app = Flask(__name__)


@app.route('/hello')
def hello_world():
      return "Hello World"


app.run()
```

app = Flask(__name__)
creates object of Flask by calling its constructor

@app.route('/')
App routing is used to map the specific URL with the associated function that is intended to perform some task. In this case function 'hello_world' will be called when url contains '/hello'.

To see the result run the python code on terminal and open link: ‘127.0.0.1:5000/hello’

Moving forward to face recognition api, to keep things simple we’ll be writing 4 methods/ functions:
Updating dataset image, Updating dataset name, Recognizing the face, Returning the frame

Follow the following steps:
1. import all the necessary modules

```
from flask import Flask, request, Response, send_file
import numpy as np
import base64
import cv2
import jsonpickle
import face_recognition
from os import listdir
from os.path import isfile, join, dirname, realpath
```

2. initialize require variables and objects

```
imageDir = dirname(realpath(__file__)) + "\\static\\dataset\\"
app = Flask(__name__)
known_face_encodings = []
known_face_names = []
dataName = "UserName"
```

3. writing method to update the dataset name

```
@app.route('/api/dataset/name',methods = ['POST'])
def updateName():
    r=request                                                    #request recieved by
the api
    global dataName
    dataName=r.data.decode('ascii')                              #converting data from
array of bits to string
    response = {'message':'success'}                             #creating return
response
    response_pickled = jsonpickle.encode(response)
    return Response(response=response_pickled, status=200, mimetype="application/json")
```

4. writing method for updating dataset image and saving it in directory ‘imageDir’

```
@app.route('/api/dataset/image',methods = ['POST'])
def image():
    r=request                                                    #request recieved by the
api
    data = base64.b64decode(r.data)                              #converting data from
array of bits to 3D array(frame)
    f=open(imageDir+dataName+".jpg",'wb')
    f.write(data)                                                #saving the image
    f.close()
    frame = cv2.imread(imageDir+dataName+".jpg",1)
    reduced_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    cv2.imwrite(imageDir+dataName+".jpg",reduced_frame)

    #creating face encoding
    dataset=face_recognition.load_image_file(imageDir+dataName+".jpg")
    frame_encoding = face_recognition.face_encodings(dataset)[0]
    known_face_encodings.append(frame_encoding)
    known_face_names.append(dataName)
```

```
    response = {'message':'success'}
    response_pickled = jsonpickle.encode(response)
    return Response(response=response_pickled, status=200, mimetype="application/json")
```

5. writing method for recognizing faces from dataset

```
@app.route('/api/detect',methods = ['POST'])
def test():
    r=request                                                    #request recieved by the
api
    data = base64.b64decode(r.data)
    f=open(dirname(realpath(__file__))+"Input.jpg",'wb')
    f.write(data)
    f.close()

    global known_face_encodings
    global known_face_names
    frame=cv2.imread(dirname(realpath(__file__))+"Input.jpg",1)


    #image manipulation
    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]




    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # Or instead, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        face_names.append(name)


    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 60), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left, bottom - 6), font, 3.0, (255, 255, 255), 3)
        print(left,right)


    cv2.imwrite(dirname(realpath(__file__))+"Output.png",frame)
    response = {'message':'success'}
```

```
    response_pickled = jsonpickle.encode(response)
    return Response(response=response_pickled, status=200, mimetype="application/json")
```

6. writing method to return the image

```
@app.route('/api/download')
def download():
    return send_file(dirname(realpath(__file__))+'Output.png')      #returning the edited frame
```

7. if the dataset already has some images then they need to be encoded first before hosting the api

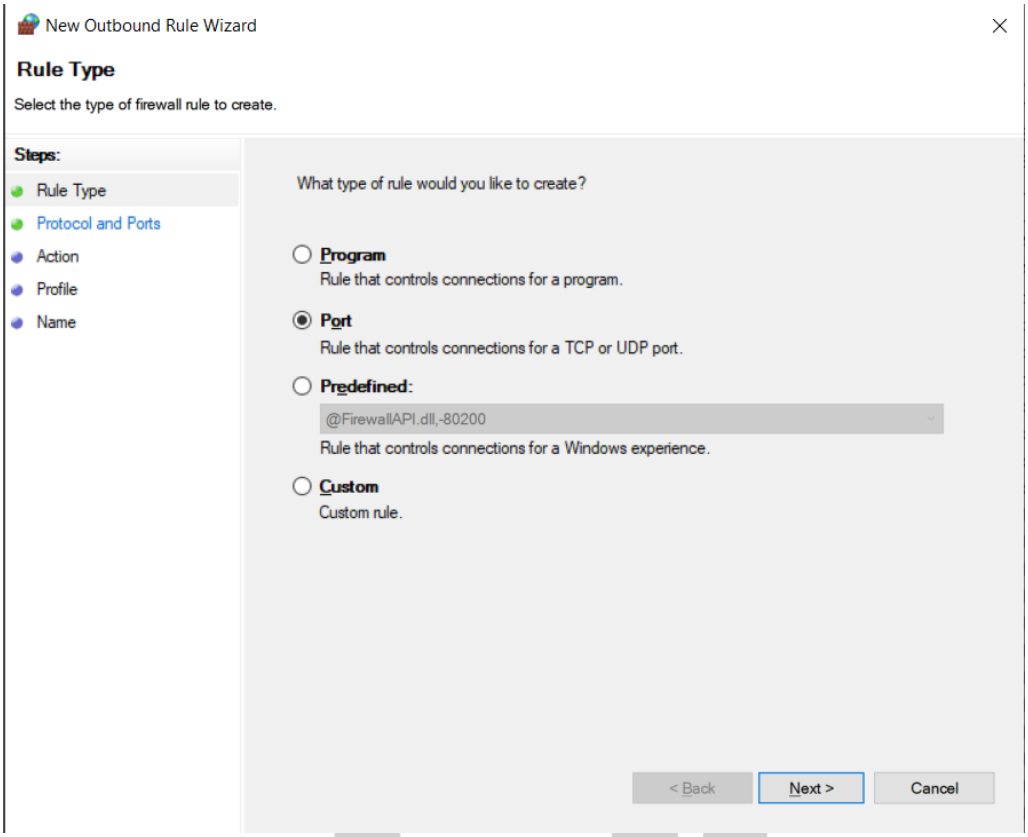
```
imageDir = dirname(realpath(__file__)) + "\\static\\dataset\\"
onlyfiles = [f for f in listdir(imageDir) if isfile(join(imageDir, f))]

for i in onlyfiles:
    dataset=face_recognition.load_image_file(imageDir+i)
    print(dataset[0])
    frame_encoding = face_recognition.face_encodings(dataset)[0]
    known_face_encodings.append(frame_encoding)
    known_face_names.append(i[:-4])
host = input("Enter the IP address of this PC: ")
port = int(input("Enter the Port number: "))
print("Hosting API on",host+":"+str(port))
app.run(host=host, port=port)
```

B. hosting api on local server

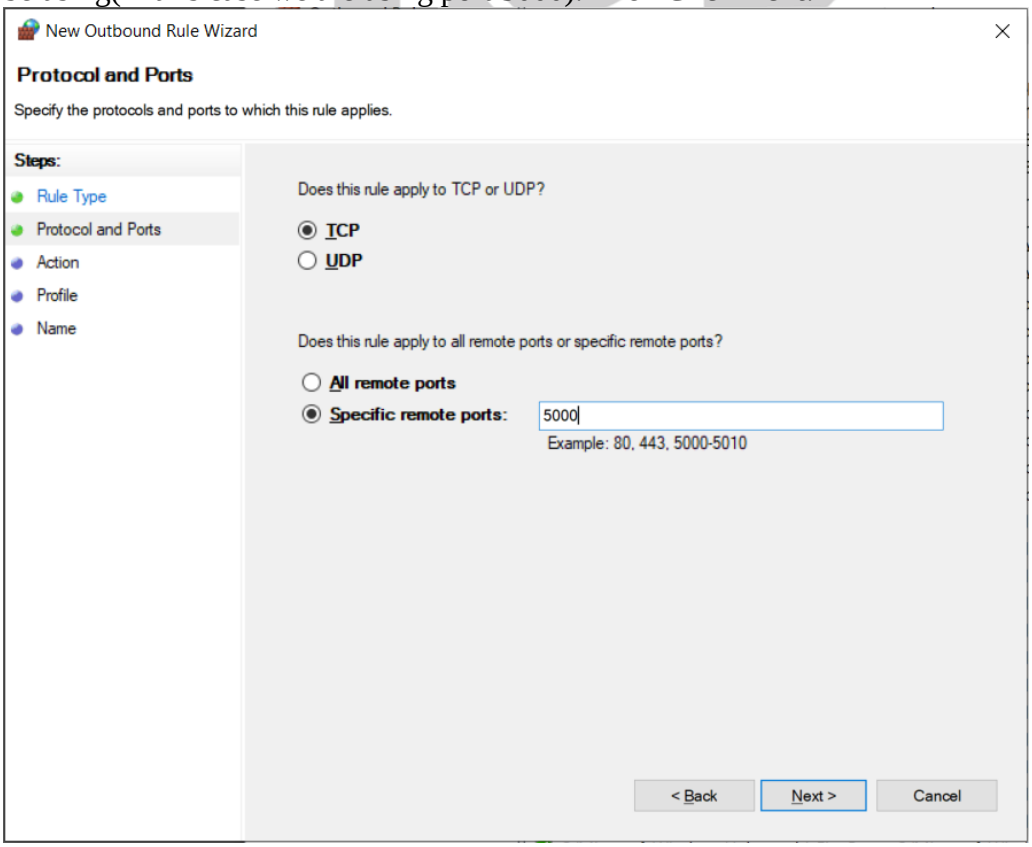
This is by far the most important and sometimes most difficult process. The api we developed will be hosted locally on your laptop and android app will be sending the api requests via hotspot.
For accessing ports and ip addresses of the devices on the same network some changes has to done in ‘Windows firewall defender’
Simplest (risky) way :
Launch Windows Firewall and turn it off.

Safer way:

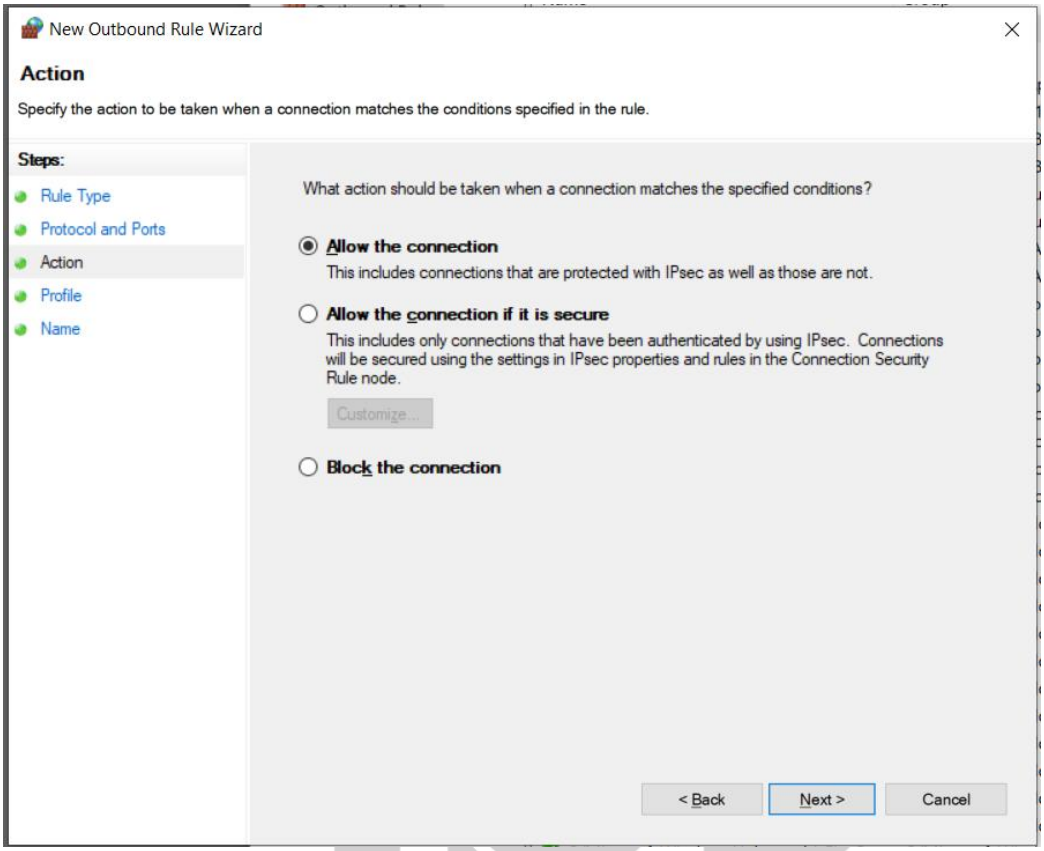
1. Launch Windows Firewall. Go to advanced settings.
2. In Inbound Rules select New Rule
3. Select the **port** option and click **Next**.



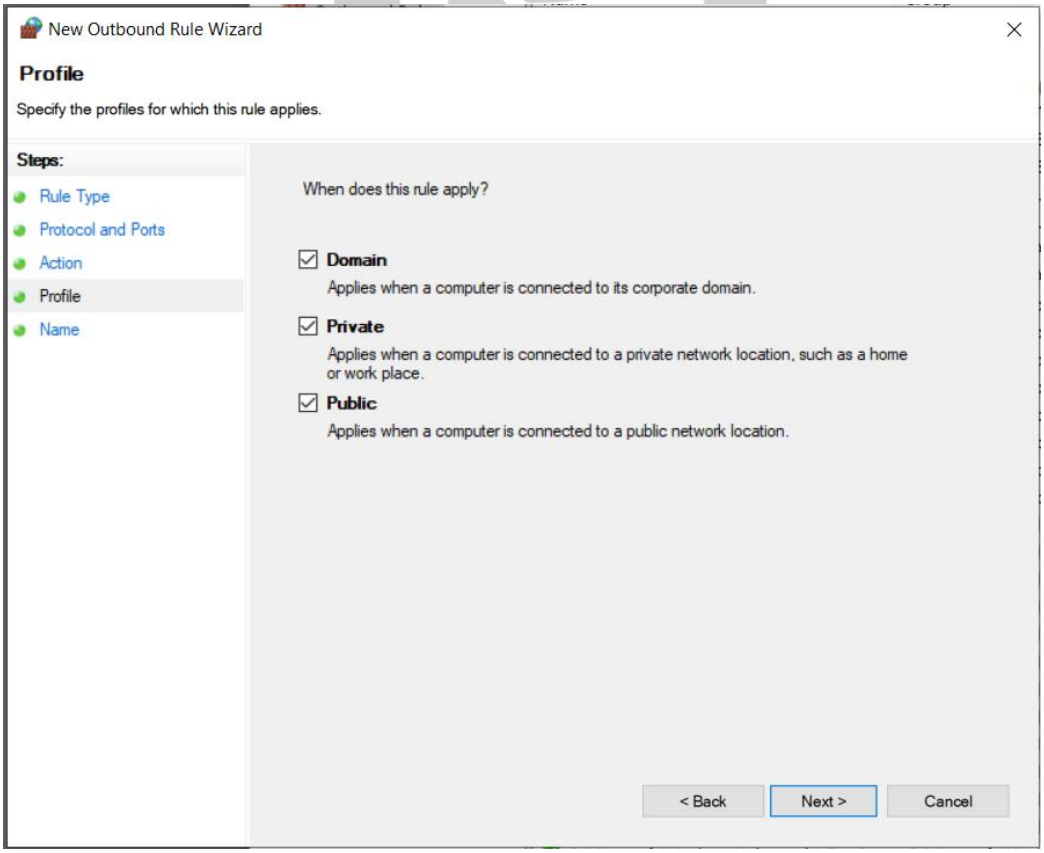
4. Select **TCP** option . Then Select **Specific remote ports** and enter the port number that we'll be using(in this case we are using port 5000). Then Click **Next**.



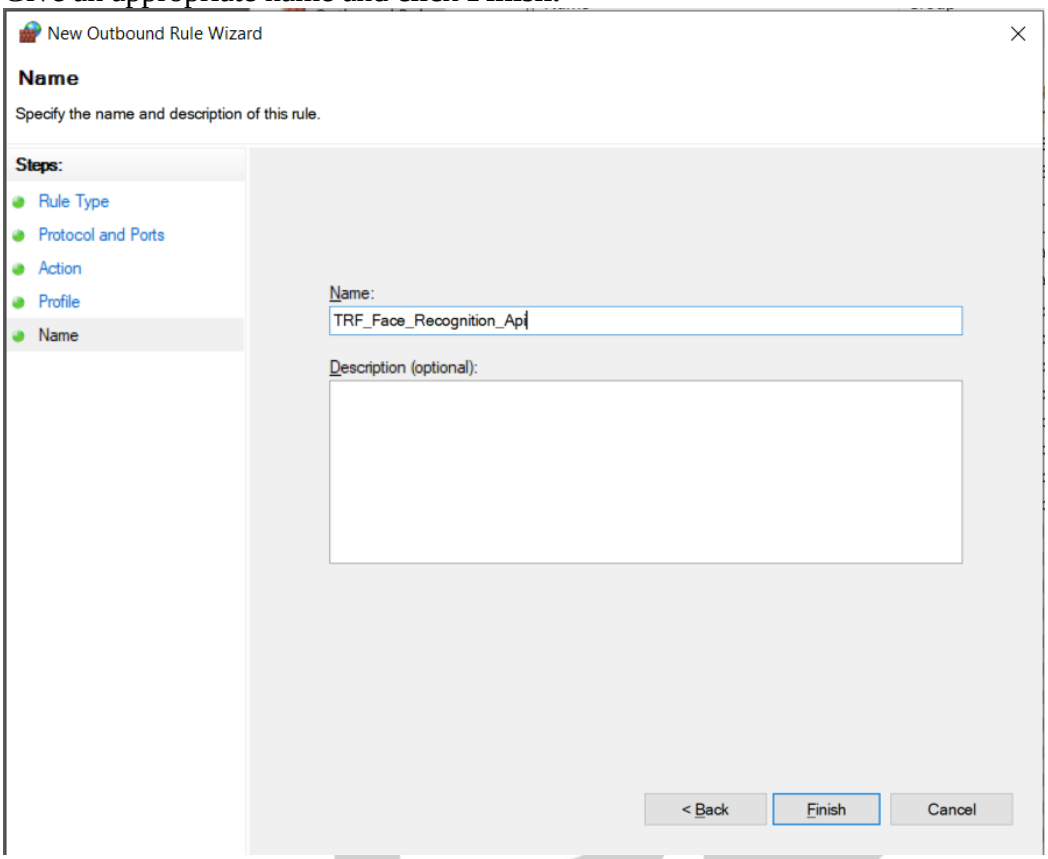
5. Select **Allow the connection** and click **Next.**



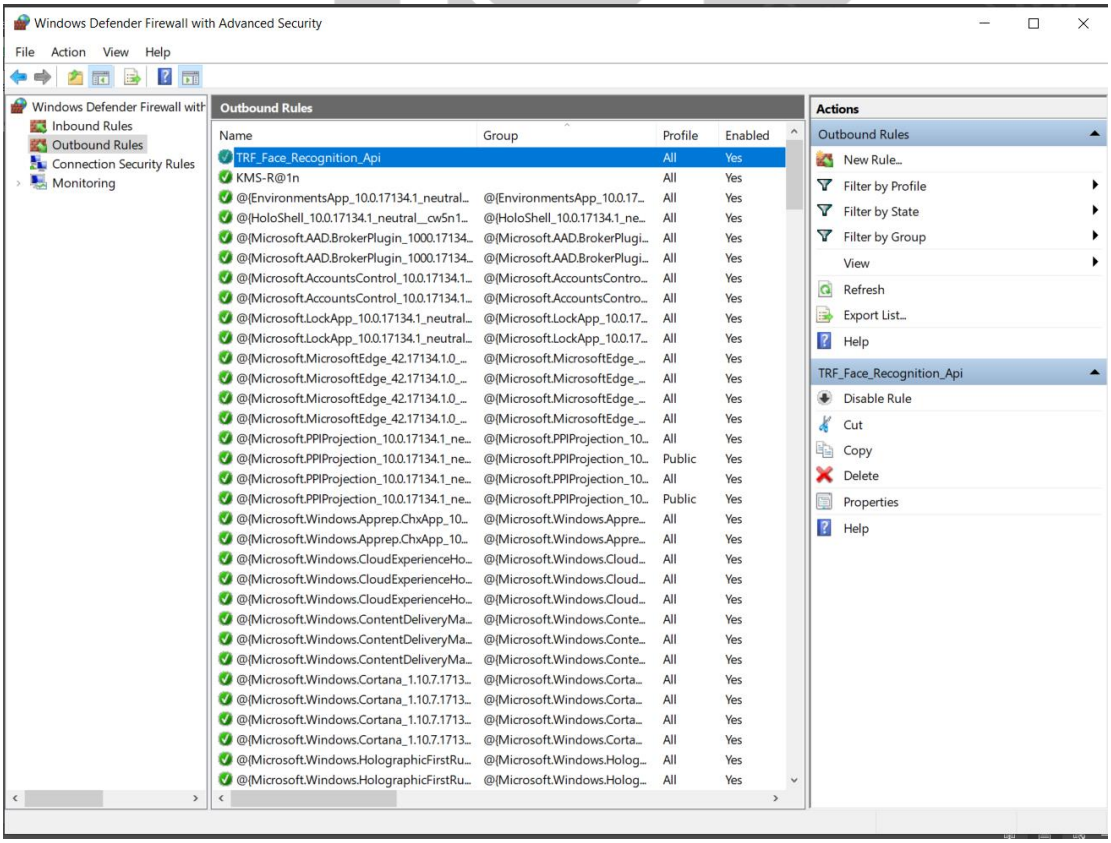
6. Tick **all** the boxes and click **Next.**



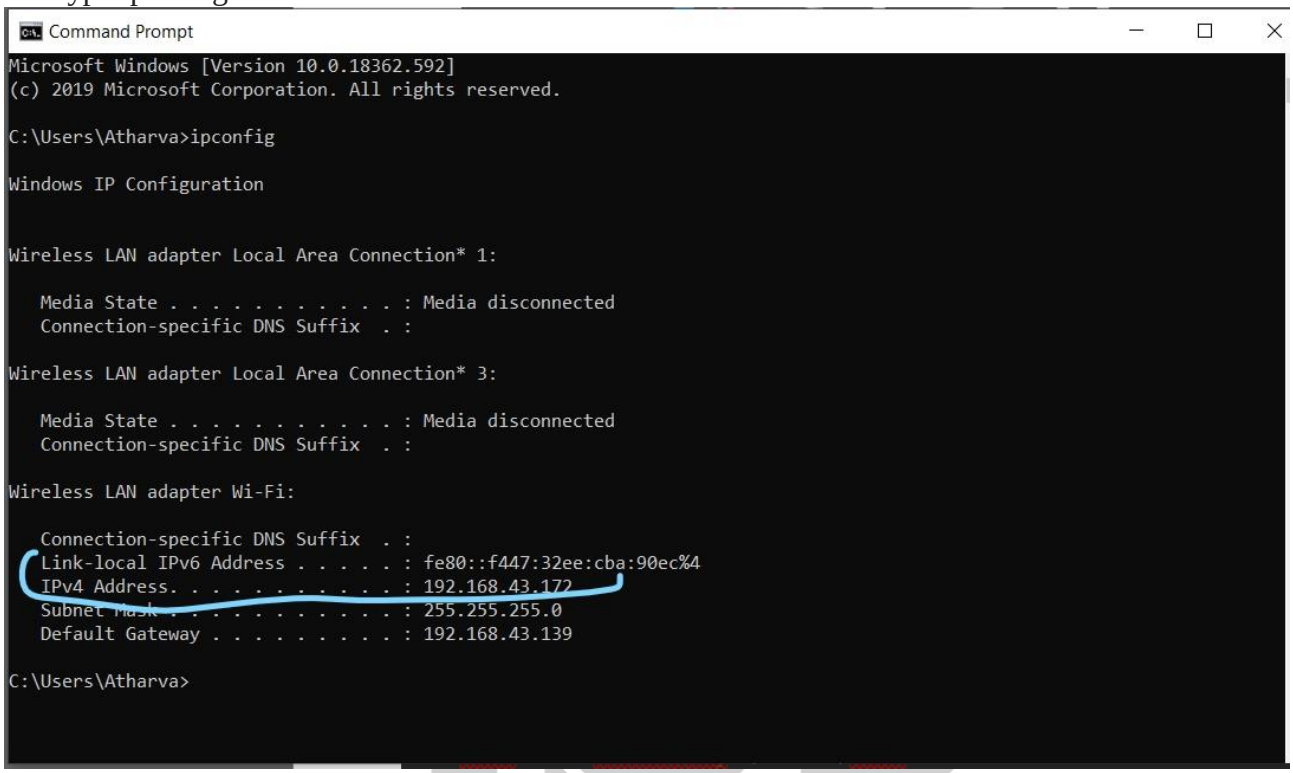
7. Give an appropriate name and click **Finish.**



8. Your rule is created

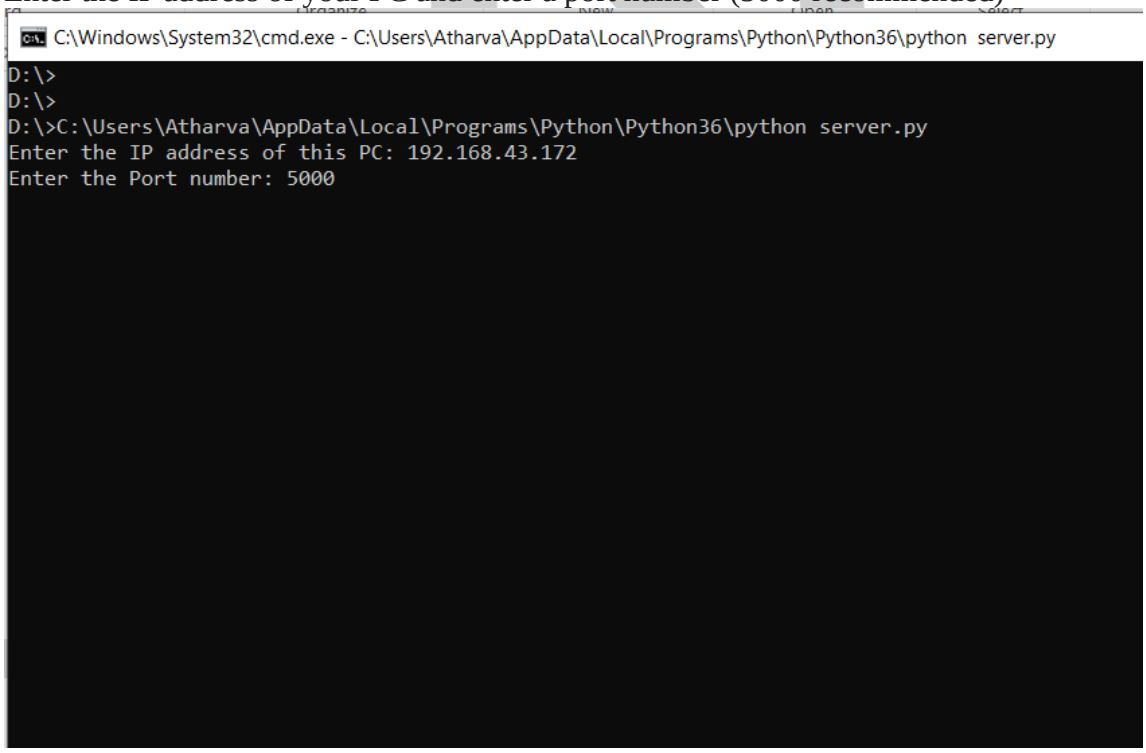


Now the port on your laptop is free to be accessed from outside.
Next, we'll be hosting api on our laptops IP address
For this start hotspot of your android device and connect your laptop. Then open command prompt and type ipconfig.



that's your IP address.

Now run the python code we just wrote in command prompt.
Enter the IP address of your PC and enter a port number (5000 recommended)



And your Api is ready to accept requests.

Android Application:

1. Create a New Project in Android Studio.
2. In Choose your project window, select Empty Activity in Phone and Tablet section and click next.
3. In Configure your project window, enter the project name, package name and save location.
4. Select API 21: Android 5.0 (Lollipop) as the minimum SDK level and click Finish.

A new project will be created by Android Studio for you. Wait for the Gradle Build to complete the syncing. You can view gradle build progress at the bottom of Android Studio.

5. Add the following into the string.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
    <color name="opacityBlack">#8F000000</color>
    <color name="yellow">#DBCA33</color>
    <color name="black">#0C0C0C</color>
    <color name="trflogo_black">#0B0A0B</color>
    <color name="white">#ffffff</color>
    <color name="color1">#1B51D8</color>
    <color name="color2">#8BC34A</color>
    <color name="color3">#A78408</color>
    <color name="color4">#5BEB0D</color>
    <color name="color5">#E10FF8</color>
    <color name="color6">#F10808</color>
    <color name="color7">#EBFC05</color>
    <color name="color8">#17E9E1</color>
    <color name="color9">#0019AA</color>
    <color name="color10">#FABD06</color>
    <color name="color11">#F00050</color>
    <color name="color12">#4409AC</color>
    <color name="color13">#022B26</color>
    <color name="color14">#26045F</color>
    <color name="color15">#79260C</color>
    <color name="clicked">#505050</color>
</resources>
```

6. Add the following to the string.xml

```
<resources>
    <string name="app_name">api_getandpost</string>
    <string name="enter_the_ip_address_of_server">Enter the ip address of
Server</string>
    <string name="ping">PING</string>
    <string name="next">next</string>
    <string name="click">Click</string>
    <string name="enter_the_name_of_the_person">Enter the name of the
person.</string>
    <string name="getting_your_image_ready">Getting your image ready !</string>
    <string name="upload">UPLOAD</string>
    <string name="detect">DETECT</string>
    <string name="name">NAME</string>
    <string name="enter_the_ip_address_of_the_server">Enter the IP address of the
server below:</string>
    <string name="enter_the_port_number_of_the_host">Enter the PORT number of the
```

```
Host below:</string>
    <string name="proceed">Proceed</string>
    <string name="port_number">PORT Number</string>
    <string name="ipv4_address">IPv4 Address</string>
    <string name="api_server_details">API Server Details</string>
</resources>
```

Global

***THIS IS NOT AN ACTIVITY. THIS IS A SIMPLE JAVA CLASS.**

1. Right click on your package name and select, New>Java class.
2. Name the class as Global.

Create the following variables:

```
    public static Bitmap mBitmap = null;
    public static String imageEncoded;
    public static String userUrl;
```

ClickImage

1. Right click on your package name and select, New>Activity>Empty Activity.
2. Name the new activity as ClickImage. Make sure the Generate Layout File option is checked.
3. Declare the following global variables.

4. Define all the widgets globally. Initialize the widgets in OnCreate() method. Set onClick listener to the click button. Call the captureImage() method in onClick. Complete OnCreate method is given below.

```
CameraView mCameraView;
Button click;
Boolean capture = false;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
    setContentView(R.layout.activity_clickimage);
    click = (Button)findViewById(R.id.click);

    mCameraView = (CameraView)findViewById(R.id.pic);
    click.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mCameraView.captureImage();
        }
    });
}
```

5. Add CameraKitListener to mCameraView. We need to add our code in onImage overriden method. i.e save the frame in Bitmap format. This method is called when a frame is capture.

```
mCameraView.addCameraKitListener(new CameraKitEventListener() {
    @Override
    public void onEvent(CameraKitEvent cameraKitEvent) {

    }

    @Override
    public void onError(CameraKitError cameraKitError) {

    }

    @Override
    public void onImage(CameraKitImage cameraKitImage) {
        mBitmap=cameraKitImage.getBitmap();

        Intent i = new Intent(getApplicationContext(),LoadingScreen.class);
        mCameraView.stop();
        startActivity(i);
    }

    @Override
    public void onVideo(CameraKitVideo cameraKitVideo) {

    }
});
```

6. Moving to UI part. There are Two widgets- CameraView, Button

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ClickImage">
    <com.wonderkiln.camerakit.CameraView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/pic">
    </com.wonderkiln.camerakit.CameraView>

    <Button
        android:id="@+id/click"
        android:layout_margin="20dp"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:elevation="100dp"
        android:background="@drawable/capture"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

    </Button>

</androidx.constraintlayout.widget.ConstraintLayout>
```

LoadingScreen

In this activity the bitmap frame captured is converted into a string for sending it to Our api.

1. Right click on your package name and select, New>Activity>Empty Activity.
2. Name the new activity as LoadingScreen. Make sure the Generate Layout File option is checked.
3. Create a variable for ImageView. Load the Currently saved Bitmap image into the ImageView.

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.util.Base64;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;

import java.io.ByteArrayOutputStream;

import static com.example.api_getandpost.Global.imageEncoded;
import static com.example.api_getandpost.Global.mBitmap;

public class LoadingScreen extends AppCompatActivity {
    ImageView loadingImage;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_loading_screen);
        loadingImage = (ImageView)findViewById(R.id.loadingInmage);
        loadingImage.setImageBitmap(mBitmap);
    }
}
```

4. The conversion will be carried out on another Thread (not UI thread as it is not for processing data). After the conversion call the Option Activity.

```
Thread encoding = new Thread(){
    public void run(){
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        mBitmap.compress(Bitmap.CompressFormat.PNG, 100, baos);
        byte[] b = baos.toByteArray();
        imageEncoded = Base64.encodeToString(b, Base64.DEFAULT);
        Intent i = new Intent(getApplicationContext(),Option.class);
        startActivity(i);
        finish();
    }
};
encoding.start();
```

5. UI of LoadingScreen only contains an ImageView and a TextView.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoadingScreen">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/loadingInmage"
        android:background="@color/yellow">
    </ImageView>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/opacityBlack"
        android:textColor="#ffffff"
        android:gravity="center"
        android:textSize="20sp"
        android:fontFamily="sans-serif-light"
        android:text="@string/getting_your_image_ready">
    </TextView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Option

1. Right click on your package name and select, New>Activity>Empty Activity.
2. Name the new activity as Option. Make sure the Generate Layout File option is checked.
3. Create one variable for ImageView and two variables for Button.

```
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;

import static com.example.api_getandpost.Global.mBitmap;

public class Option extends AppCompatActivity {
    Button upload,detect;
    ImageView userData;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_option);
        upload = (Button) findViewById(R.id.uploadScreen);
        detect = (Button) findViewById(R.id.detectScreen);
        userData = (ImageView) findViewById(R.id.selectorDisplay);
        userData.setMaxHeight(mBitmap.getHeight());
        userData.setMaxWidth(mBitmap.getWidth());
        userData.setImageBitmap(mBitmap);
    }
}
```

4. Add setOnClickListener to both the buttons and call the respective activity.

```
detect.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onClick(View v) {
        detect.setBackgroundColor(getColor(R.color.clicked));
        Intent i =new Intent(getApplicationContext(), Detection.class);
        startActivity(i);
        finish();
    }
});
upload.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onClick(View v) {
        upload.setBackgroundColor(getColor(R.color.clicked));
        Intent i =new Intent(getApplicationContext(),uploadData.class);
        startActivity(i);
        finish();
    }
});
```

5. UI of Option:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/yellow"
    tools:context=".Option">
    <ImageView
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:id="@+id/selectorDisplay"
        android:foregroundGravity="top"
        android:background="#00ffffff">
    </ImageView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:orientation="horizontal"
        android:weightSum="2"
        app:layout_constraintBottom_toBottomOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:padding="15dp">
            <Button
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/uploadScreen"
                android:background="@color/black"
                android:textColor="@color/white"
```

```
                android:elevation="20dp"
                android:text="@string/upload">
            </Button>
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:layout_margin="15dp">
            <Button
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@color/black"
                android:textColor="@color/white"
                android:elevation="20dp"
                android:text="@string/detect"
                android:id="@+id/detectScreen">
            </Button>
        </LinearLayout>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

url

***THIS IS NOT AN ACTIVITY. THIS IS A SIMPLE JAVA CLASS.**

1. Right click on your package name and select, New>Java class.
2. Name the class as url.

3. This class inherits methods from AsyncTask<String, void, String>. The parameters are:
a. Params, the type of the parameters sent to the task upon execution.
b. Progress, the type of the progress units published during the background computation.
c. Result, the type of the result of the background computation.

We'll be overriding two functions:

a. onPostExecute() – This method is called when the api request is terminated successfully.
b. doInBackground() – This method is called when the execute method is called. It communicates with the api.

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.util.Log;

import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;

import javax.net.ssl.HttpsURLConnection;

public class url extends AsyncTask<String, Void, String> {


    @Override
```

```
    protected void onPostExecute(String result) {

    }

    @Override
    protected String doInBackground(String... requestURL) {

    }
}
```

4. Create an Interface inside the url class for communication between the class and the activity where it is called.

```
public urlExecption execution = null;

public interface urlExecution{
    void processFinish(String output);
}
```

5. Call the processFinish() method in onPostExecute() method.

```
@Override
protected void onPostExecute(String result) {
    execution.processFinish(result);
}
```

6. Add the following code in the doInBackground() method.

```
@Override
protected String doInBackground(String... requestURL) {
    URL url;
    String response = "";
    try {
        url = new URL(requestURL[0]);

        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(15000);
        conn.setConnectTimeout(15000);
        conn.setRequestMethod("POST");
        conn.setDoInput(true);
        conn.setDoOutput(true);
        conn.setRequestProperty("Content-Type", "Detection/jpeg");


        OutputStream os = conn.getOutputStream();
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(os, "UTF-8"));
        writer.write(requestURL[1]);

        conn.connect();
        writer.flush();
        writer.close();
        os.close();
        conn.connect();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

uploadData

This activity upload the bitmap and a string (name) as Dataset needed for face recognition.

1. Right click on your package name and select, New>Activity>Empty Activity.
2. Name the new activity as uploadData. Make sure the Generate Layout File option is checked.
3. Create one variable for ImageView, one variable for EditText and one variable for Button. The EditText will take input(name of the person in the image) from the user.

```
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Bitmap;
import android.os.Build;
import android.os.Bundle;
import android.util.Base64;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;

import java.io.ByteArrayOutputStream;

import static com.example.api_getandpost.Global.imageEncoded;
import static com.example.api_getandpost.Global.mBitmap;
import static com.example.api_getandpost.Global.userUrl;

public class uploadData extends AppCompatActivity implements AsyncResponse {

    ImageView userData;
    Button upload;
    EditText name;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_upload_data);

        userData = (ImageView) findViewById(R.id.userData);
        userData.setImageBitmap(mBitmap);
        upload = (Button) findViewById(R.id.upload);
        name = (EditText) findViewById(R.id.name);
        upload.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.M)
            @Override
            public void onClick(View v) {
                String dataName = String.valueOf(name.getText());
                upload.setBackgroundColor(getColor(R.color.clicked));
                finish();
            }
        });
```

```
    }
}
```

4. Create an object of class url(which we created in the last step). This object will communicate with our python API. Call the method ‘execute’ which takes two argument:
a. http address
b. Data to be transferred in string format
we’ll be creating object apiUrlName for sending the name of the person in the image and apiUrlImage for sending the bitmap image itself(in String format which we generated in the LoadingScreen Activity).

Declare the following objects globaly.

```
url apiUrlName = new url();
url apiImageUrl = new url();
```

Add the following lines in OnCreate function

```
apiUrlName.execution = this;
apiImageUrl.execution = this;
```

Add the following lines in the onClickListener

```
apiUrlName.execute(userUrl+"api/dataset/name",dataName);
apiImageUrl.execute(userUrl+"api/dataset/image",imageEncoded);
```

Implement the url.urlExecution interface and add the @Override method processFinish. Keep this method empty.

5. UI of this activity consists of one ImageView, one editText and one Button.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="0dp"
    android:orientation="vertical"
    android:layout_margin="0dp"
    android:background="@color/yellow"
    tools:context=".uploadData">
<ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/userData"
        android:background="#00000000"
        app:layout_constraintTop_toTopOf="parent" >
    </ImageView>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:gravity="center"
        android:text="@string/enter_the_name_of_the_person"
        android:background="#5F000000"
        app:layout_constraintBottom_toTopOf="@+id/uploadLayout" />
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:id="@+id/uploadLayout"
        android:layout_height="100dp"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        android:weightSum="10">
        <EditText
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/name_border"
            android:focusable="true"
            android:fontFamily="sans-serif"
            android:gravity="center"
            android:textColor="#ffffffff"
            android:hint="@string/name"
            android:layout_weight="2"
            android:inputType="text">
        </EditText>
        <Button
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/upload"
            android:background="@color/black"
            android:text="@string/upload"
            android:textColor="#ffffff"
            android:layout_gravity="center"
            android:layout_weight="8">
        </Button>
    </LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Detection

1. Right click on your package name and select, New>Activity>Empty Activity.
2. Name the new activity as Detection. Make sure the Generate Layout File option is checked.
3. Create a single variable for ImageView to display the detected faces.

```
import androidx.appcompat.app.AppCompatActivity;

import com.squareup.picasso.Picasso;

import android.os.Bundle;
import android.util.Base64;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.ByteArrayOutputStream;

import static com.example.api_getandpost.Global.imageEncoded;
import static com.example.api_getandpost.Global.mBitmap;
import static com.example.api_getandpost.Global.userUrl;

public class Detection extends AppCompatActivity {
    ImageView detected;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_detection);
        detected = (ImageView) findViewById(R.id.detected);
    }


    @Override
    protected void onResume() {
        super.onResume();
    }

 }
```

4. Implement the url.urlExecution interface and add the @Override method processFinish().

```
public class Detection extends AppCompatActivity implements  {
    ImageView detected;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_image);
        detected = (ImageView) findViewById(R.id.detected);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    public void processFinish(String output) {

    }
}
```

5. Similar to the uploadData, create an object mUrl of class url and pass the http address and the String for detection.

```
url mUrl = new url();
mUrl.execution = this;
mUrl.execute(userUrl+"api/detect",imageEncoded);
Toast.makeText(getApplicationContext(),"Sending Frame for
Detection",Toast.LENGTH_LONG).show();
```

6. When the process is finished the overridden method processFinish() is called. In this method we'll load the detected frame into the ImageView. For this, we'll use Picasso as it makes things simple. It requires minimum two arguments:
a. http address
b. ImageView where the image will be loaded

Add the following code in processFinish() method

```
Toast.makeText(getApplicationContext(),"Frame recieved",Toast.LENGTH_LONG).show();
Picasso.get().load(userUrl+"api/download").into(detected);
```

UI of this Activity only contains one ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="0dp"
    android:orientation="vertical"
    android:layout_margin="0dp"
    tools:context=".Detection">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/detected"
        android:background="@color/yellow"
        app:layout_constraintTop_toTopOf="parent" >
    </ImageView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Here we end making of FACE RECOGNITION app.

- You can find all the code on this GitHub repo - https://github.com/sau22rane/TRF_api_based_Face_Recognition/tree/master

If you have any doubts, errors or problems regarding the app, please drop an email at saurabh.rane18@vit.edu describing the issue with screenshots.