

**University Department of Computer Science**

**University of Mumbai**

**Project Proposal**

Of

***CLOUD BASED FACIAL RECOGNITION SYSTEM***

*By*

***AJHARALI SHAIKH***

**INDEX**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NUMBER** |
| **Background** | **3** |
| **Requirements** | **8** |
| **Design** | **11** |
| **Coding and Implementation** | **13** |
| **Test cases** | **19** |
| **Deployment** | **21** |
| **Client-Server** | **22** |
| **Application flow** | **24** |

Department of computer science, university of mumbai  
faceclick

5/2/2021

# Overview

## Project Background and Description

|  |  |
| --- | --- |
|  | Describe how this project came about, who is involved, and the purpose.  Note: To delete any tip (such as this), select it and start typing. If you’re not yet ready to add your own text, select a tip and press spacebar to remove it. |

**BACKGROUND:-**

Automated facial recognition was pioneered in the 1960s. [Woody Bledsoe](https://en.wikipedia.org/wiki/Woody_Bledsoe), [Helen Chan Wolf](https://en.wikipedia.org/wiki/Helen_Chan_Wolf), and Charles Bisson worked on using the computer to recognize human faces. Their early facial recognition project was dubbed "man-machine" because the coordinates of the facial features in a photograph had to be established by a human before they could be used by the computer for recognition. On a [graphics tablet](https://en.wikipedia.org/wiki/Graphics_tablet) a human had to pinpoint the coordinates of facial features such as the pupil centers, the inside and outside corner of eyes, and the [widows peak](https://en.wikipedia.org/wiki/Widows_peak) in the hairline. The coordinates were used to calculate 20 distances, including the width of the mouth and of the eyes. A human could process about 40 pictures an hour in this manner and so build a database of the computed distances. A computer would then automatically compare the distances for each photograph, calculate the difference between the distances and return the closed records as a possible match.[[3]](https://en.wikipedia.org/wiki/Facial_recognition_system#cite_note-3)

In 1970 [Takeo Kanade](https://en.wikipedia.org/wiki/Takeo_Kanade) publicly demonstrated a face matching system that located anatomical features such as the chin and calculated the distance ratio between facial features without human intervention. Later tests revealed that the system could not always reliably identify facial features. But interest in the subject grew and in 1977 Kanade published the first detailed book on facial recognition technology.

**DESCRIPTION:-**

Using Computer Vision we can perform a variety of facial applications, including **facial recognition**, **building a virtual makeover system**(i.e., makeup, cosmetics, eyeglasses/sunglasses, etc.), or even **aiding in law enforcement** to help detect, recognize, and track criminals.

Computer Vision is powering facial recognition at a massive scale — just take a second to consider that **over**350 million**images are uploaded to Facebook**every day.

For each of those images, Facebook is running **face detection** (to detect the presence) of faces followed by **face recognition** (to actually tag people in photos).

**Face detection is**different**than face recognition.**

During face detection we are simply trying to locate where in the image faces are.

Our face detection algorithms do not know who is in the image, simply that a given face exists at a particular location.

Once we have our detected faces, we pass them into a facial recognition algorithm which outputs the actual identify of the person/face.

Thus, all Computer Vision and facial applications must start with face detection.

OpenCV’s face detector is **accurate** and able to **run in real-time** on modern laptops/desktops.

Human face detection and recognition play important roles in many applications such as video surveillance and face image database management. In our project, we have studied worked on both face recognition and detection techniques and developed algorithms for them.

Algorithms give different rates of accuracy under different conditions as experimentally observed. In face detection, we have developed an algorithm that can detect human faces from an image.

Facial recognition is an enhanced application [bio-metric software](https://en.wikipedia.org/wiki/Biometrics) that uses a deep learning algorithm to compare a live capture or digital image to the stored face print to verify individual identity. However, **deep learning** is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

Facial detection is the process of identifying a human face within a scanned image; the process of extraction involves obtaining a facial region such as the eye spacing, variation, angle and ratio to determine if the object is human.

we wanted to localize various facial structures, including:

* Nose
* Eyes
* Mouth
* Jawline

Using facial landmarks we can do that.

A facial recognition system is a technology capable of matching a [human face](https://en.wikipedia.org/wiki/Human_face) from a [digital image](https://en.wikipedia.org/wiki/Digital_image) or a [video frame](https://en.wikipedia.org/wiki/Film_frame) against a [database](https://en.wikipedia.org/wiki/Database) of faces, typically employed to [authenticate](https://en.wikipedia.org/wiki/Authenticate) users through [ID verification services](https://en.wikipedia.org/wiki/ID_verification_service), works by pinpointing and measuring facial features from a given image.

While initially a form of computer [application](https://en.wikipedia.org/wiki/Application_software), facial recognition systems have seen wider uses in recent times on [smartphones](https://en.wikipedia.org/wiki/Smartphone) and in other forms of technology, such as [robotics](https://en.wikipedia.org/wiki/Robotics). Because computerized facial recognition involves the measurement of a human's physiological characteristics facial recognition systems are categorised as [biometrics](https://en.wikipedia.org/wiki/Biometrics). Although the accuracy of facial recognition systems as a biometric technology is lower than [iris recognition](https://en.wikipedia.org/wiki/Iris_recognition) and [fingerprint recognition](https://en.wikipedia.org/wiki/Fingerprint), it is widely adopted due to its contactless process Facial recognition systems have been deployed in advanced [human-computer interaction](https://en.wikipedia.org/wiki/Human-computer_interaction), [video surveillance](https://en.wikipedia.org/wiki/Video_surveillance) and automatic [indexing](https://en.wikipedia.org/wiki/Search_engine_indexing) of images.[[2]](https://en.wikipedia.org/wiki/Facial_recognition_system#cite_note-:8-2) They are also used widely by law enforcement agencies.

## Project Scope

|  |  |
| --- | --- |
|  | Project scope defines the boundaries of a project. Think of the scope as an imaginary box that will enclose all the project elements/activities. It not only defines what you are doing (what goes into the box), but it sets limits for what will not be done as part of the project (what doesn’t fit in the box). Scope answers questions including what will be done, what won’t be done, and what the result will look like. |

The objective of the program given is to detect object of interest(face) in real time and to keep tracking of the same object.This is a simple example of how to detect face in Python. You can try to use training samples of any other object of your choice to be detected by training the classifier on required objects.

The workflow can be broken down into following basic steps:

Training a machine learning model on a local system.

A web interface application for to capture users face/image.

• The image of the person is captured and sent to the backend service.

• Build a backend process for handling requests for your web application.

• Search in the database table that the image of the person is present or not.

1. If the image is already present in the backend send the alert to the user that the face is recognized.

2. If the image is not present in the database send the alert to the user that face is not recognized. Do you want to store the image in the database.

## High-Level Requirements

|  |  |
| --- | --- |
|  | Describe the high level requirements for the project. For example: |

* System Requirement: - 1. Operating System.  
   2. 8GB RAM Minimum.  
   3. 500GB HDD Minimum.  
   4. Core I3 processor Minimum.

5. Camera

* Software Needed: - 1. Python.  
   2. Anaconda/PyCharm.  
   3. Web Browser
* Technical Requirement: - 1. OpenCV.

## Deliverables

|  |  |
| --- | --- |
|  | List agencies, stakeholders or divisions which will be impacted by this project and describe how they will be affected by the project. |

* Web based Application for Face Recognition.
* A web-based application for users to check the project.
* A backend process for handling requests for web application.
* Restful API service to communicate between front end and back end.

## Affected Parties

|  |  |
| --- | --- |
|  | List business processes or systems which will be impacted by this project and describe how they will be affected. |

* Government.
* Mobile phone makers in products.
* Colleges in the classroom.
* Social media companies on websites.
* Businesses at entrances and restricted areas.
* Retailers in stores.
* Airlines at departure gates.

## Affected Business Processes or Systems

|  |  |
| --- | --- |
|  | Describe any specific components that are excluded from this project. |

Security Sector.

## Specific Exclusions from Scope

|  |  |
| --- | --- |
|  | Describe how you plan to implement the project. For example, will all parts of the project be rolled out at once or will it be incremental? What will be included in each release? |

The project will be on Incremental model.

Project will be limited on certain picture database.

## Implementation plan

|  |  |
| --- | --- |
|  | Include recommendations that lead to your proposed solution. Summarize what you’re proposing to do and how you’re going to meet the goals. You’ll be able to expand on the details within the ‘Our Proposal’ section. |

The workflow can be broken down into following basic steps:

Training a machine learning model on a local system.

Wrapping the inference logic into a Django application.

Using docker to containerize the Django application.

Hosting the docker container on an AWS ec2 instance and consuming the web-service.

**Step 1**. A picture of your face is captured from a photo or video. Your face might appear alone or in a crowd. Your image may show you looking straight ahead or nearly in profile.

**Step 2**. Facial recognition software reads the geometry of your face. Key factors include the distance between your eyes and the distance from forehead to chin. The software identifies facial landmarks — one system identifies 68 of them — that are key to distinguishing your face. The result: your facial signature.

**Step 3**. Your facial signature — a mathematical formula — is compared to a database of known faces.

**Step 4**. A determination is made. Your faceprint may match that of an image in a facial recognition system database.

## Design

HOW IT WORKS?

The system is made into python which consist some python library for working as suggested via instruction,

* First the flask framework is used for the frontend process.
* Second the System captures the image frame with the help of web camera.
* Third OpenCV is used for computing the images into encodings and checking the backend for the match.
* Fourth the system matches the images encoding if any existing image encodings matches with the captured image encodings.
* It shows the matched name on the screen of the video stream.

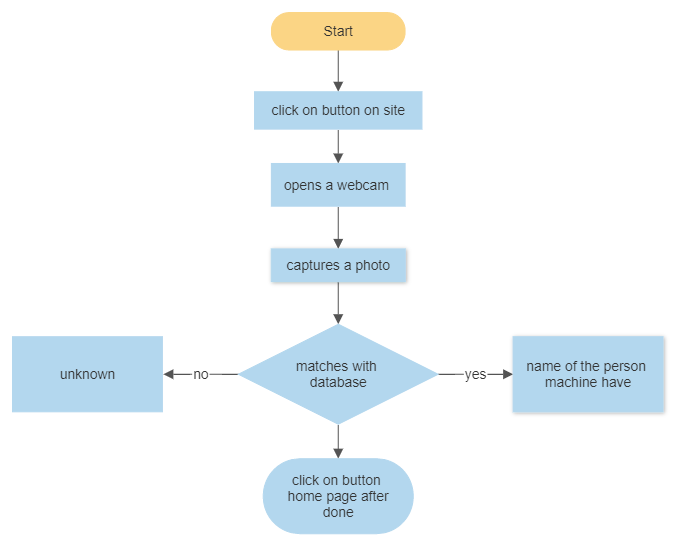
**Application Development.**

1.Frontend Development

choose a language in which this idea is going to implemented, so here I have choose Python its simple to understand and a high level language compatible to make my system as described. It is compatible to OpenCV library which is a essential part of the project and Face-recognition library which is going to work on background for detection of faces and encoding the face image and matching with the existing image encodings. Flask is a framework which is going to be the face of the application using basic Web technologies.

2.Backend Development

Python is used for the while project, for Frontend part we have used python web framework named Flask it is lightweight framework. now for backend part we are going to use python library Face-recognition for the algorithm used in face detection and face recognition. Opencv library uses the camera to capture the image and the face-recognition library encodes that captured image using certain algorithm used in that library and checks for the database matching encoding if the encodings match with the existing image encoding. Face detection algorithm detects the face on the video stream which captures the images in 5 seconds each make a rectangle on the face and displays the name of the image encoding matched name (ex. Shahrukh) and if the encoding doesn’t matches it shows unknown on the face rectangle of that person.



As shown in the above flowchart the system works exactly like that.

## Coding and Implementing phase

1.implementing the Flask Framework

from flask import Flask, render\_template, Response

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='192.168.29.127',port='8080', debug=True)

2.implementing OpenCV and face recognition

from flask import Flask, render\_template, Response

import face\_recognition

import cv2

import numpy as np

app = Flask(\_\_name\_\_)

# video\_capture = cv2.VideoCapture(0)

# camera = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

obama\_image = face\_recognition.load\_image\_file("obama.jpg")

obama\_face\_encoding = face\_recognition.face\_encodings(obama\_image)[0]

# Load a second sample picture and learn how to recognize it.

biden\_image = face\_recognition.load\_image\_file("biden.jpg")

biden\_face\_encoding = face\_recognition.face\_encodings(biden\_image)[0]

# Load a second sample picture and learn how to recognize it.

ali\_image = face\_recognition.load\_image\_file("selfphoto.jpg")

ali\_face\_encoding = face\_recognition.face\_encodings(ali\_image)[0]

salman\_image = face\_recognition.load\_image\_file("salman.jpg")

salman\_face\_encoding = face\_recognition.face\_encodings(salman\_image)[0]

amirkhan\_image = face\_recognition.load\_image\_file("amirkhan.jpg")

amirkhan\_face\_encoding = face\_recognition.face\_encodings(amirkhan\_image)[0]

shahrukh\_image = face\_recognition.load\_image\_file("shahrukh.jpg")

shahrukh\_face\_encoding = face\_recognition.face\_encodings(shahrukh\_image)[0]

amitabh\_image = face\_recognition.load\_image\_file("amitabh.jpg")

amitabh\_face\_encoding = face\_recognition.face\_encodings(amitabh\_image)[0]

irfankhan\_image = face\_recognition.load\_image\_file("irfankhan.jpg")

irfankhan\_face\_encoding = face\_recognition.face\_encodings(irfankhan\_image)[0]

john\_image = face\_recognition.load\_image\_file("john.jpg")

john\_face\_encoding = face\_recognition.face\_encodings(john\_image)[0]

mahesh\_image = face\_recognition.load\_image\_file("mahesh.jpg")

mahesh\_face\_encoding = face\_recognition.face\_encodings(mahesh\_image)[0]

nawazuddin\_image = face\_recognition.load\_image\_file("nawazuddin.jpg")

nawazuddin\_face\_encoding = face\_recognition.face\_encodings(nawazuddin\_image)[0]

tarique\_image = face\_recognition.load\_image\_file("tarique.jpeg")

tarique\_face\_encoding = face\_recognition.face\_encodings(tarique\_image)[0]

tiger\_image = face\_recognition.load\_image\_file("tiger.jpg")

tiger\_face\_encoding = face\_recognition.face\_encodings(tiger\_image)[0]

varun\_image = face\_recognition.load\_image\_file("varun.jpg")

varun\_face\_encoding = face\_recognition.face\_encodings(varun\_image)[0]

vicky\_image = face\_recognition.load\_image\_file("vicky.jpg")

vicky\_face\_encoding = face\_recognition.face\_encodings(vicky\_image)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [

obama\_face\_encoding,

biden\_face\_encoding,

ali\_face\_encoding,

salman\_face\_encoding,

amirkhan\_face\_encoding,

shahrukh\_face\_encoding,

amitabh\_face\_encoding,

irfankhan\_face\_encoding,

john\_face\_encoding,

mahesh\_face\_encoding,

nawazuddin\_face\_encoding,

tarique\_face\_encoding,

tiger\_face\_encoding,

varun\_face\_encoding,

vicky\_face\_encoding

]

known\_face\_names = [

"Barack Obama",

"Joe Biden",

"Ajharali Shaikh",

"salman khan",

"amir khan",

"shahrukh khan",

"amitabh",

"irfankhan",

"john",

"mahesh",

"nawazuddin",

"tarique",

"tiger",

"varun",

"vicky"

]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

def gen\_frames():

# video\_capture = cv2.VideoCapture("http://192.168.29.236:8080/video")

video\_capture = cv2.VideoCapture(0)

process\_this\_frame = True

while True:

# Grab a single frame of video

ret, frame = video\_capture.read()

# 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]

# Only process every other frame of video to save time

if process\_this\_frame:

# Find all the faces and face encodings in the current frame of video

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"

# # If a match was found in known\_face\_encodings, just use the first one.

# if True in matches:

# first\_match\_index = matches.index(True)

# name = known\_face\_names[first\_match\_index]

# 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)

process\_this\_frame = not process\_this\_frame

# Display the results

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 - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image

# cv2.imshow('Video', frame)

ret, buffer = cv2.imencode('.jpg', frame)

frame = buffer.tobytes()

yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

# Hit 'q' on the keyboard to quit!

# if cv2.waitKey(1) & 0xFF == ord('q'):

# break

# # Release handle to the webcam

# video\_capture.release()

@app.route('/')

def index():

# rendering webpage

return render\_template('index.html')

@app.route('/new')

def new():

# rendering webpage

return render\_template('new.html')

@app.route('/video\_feed')

def video\_feed():

return Response(gen\_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/delete')

def delete():

cv2.destroyAllWindows()

return render\_template('index.html')

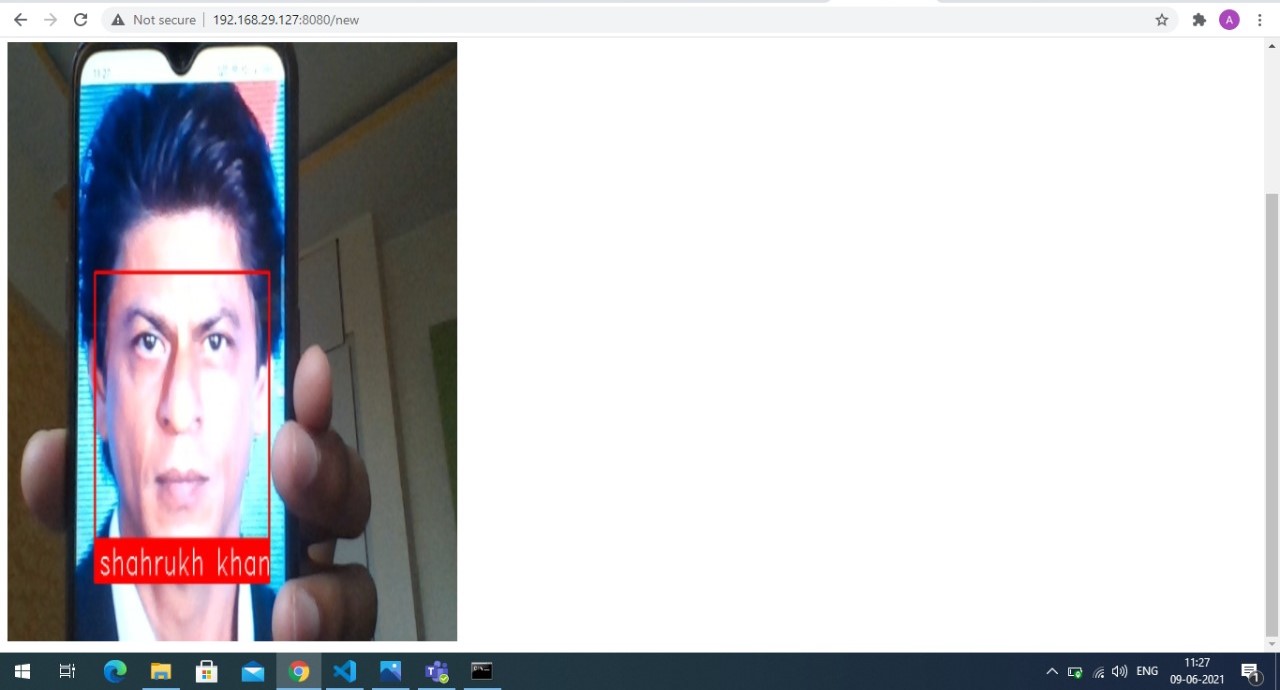
if \_\_name\_\_ == '\_\_main\_\_':

# defining server ip address and port

app.run(host='192.168.29.127',port='8080', debug=True)

## Test Cases

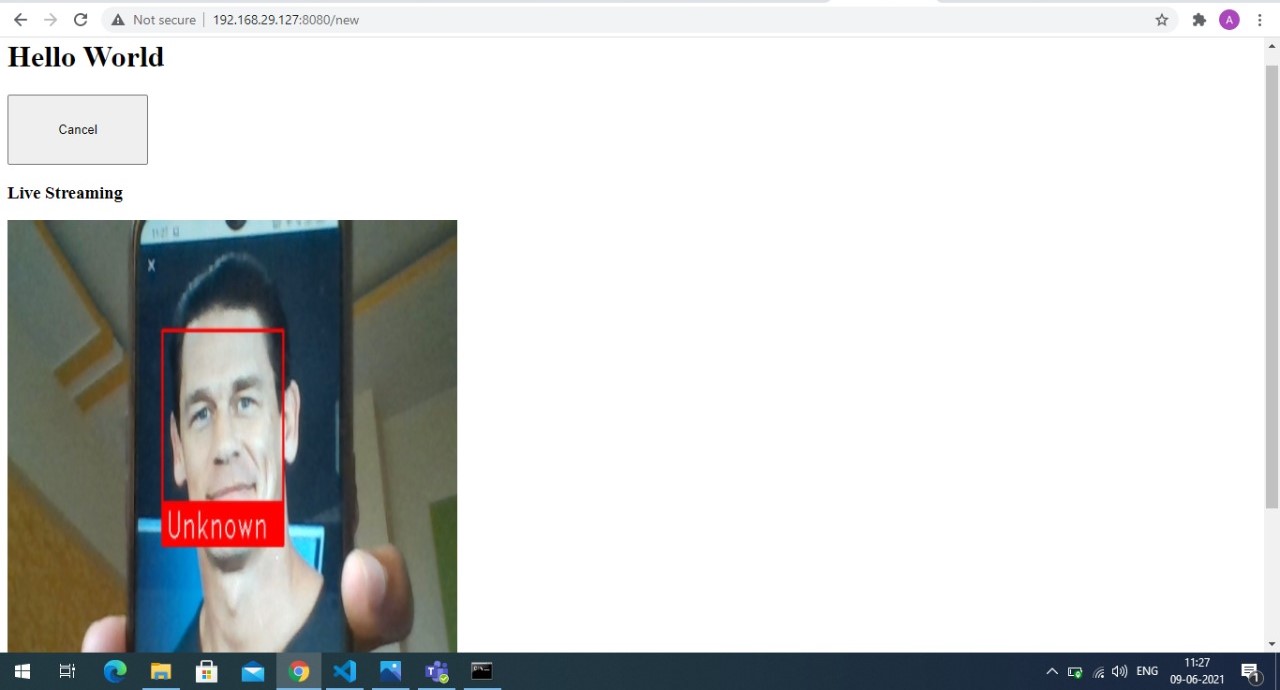
1. Checking the names of the encoding images matching



The code checks the database and matches the captured image encodings with the database encodings and displays the name of the person in front of camera.

There are total 15 faces in the database for use cases using celebrities photos like Shahrukh, salman, amir, Irfan and more.

2.Checking for the unknown faces



As we can see this image depicts the unknown face on the system because this face is not encoded in the system.

## Deployment

1.Deploying on cloud:

While deploying on cloud I faced many challenges all the free hosting and cloud storage giving company doesn’t support the opencv and Face-recognition library to use on free tires because the project uses more computing power and storage companies like.

1.Pythonanywhere

2.Heroku

3.Aws

All of these cloud providing services have been tried that’s why I have to use the other technique of cloud computing to upload my project so I have access the project via cloud.

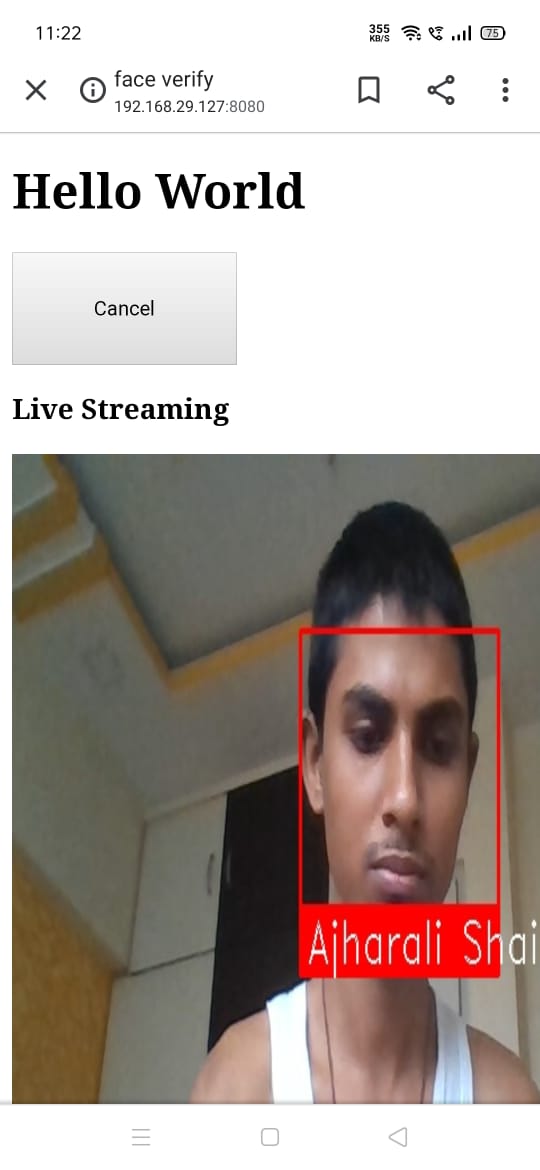
2.Deployed on Server using local network:

After getting the setback from the cloud service providers I have used this technique and I worked now my project is deployed on my laptop and my laptop acts as a server and that can be accessed using the mobile phone it acts as a client. The architecture is shown below

## Client-Server

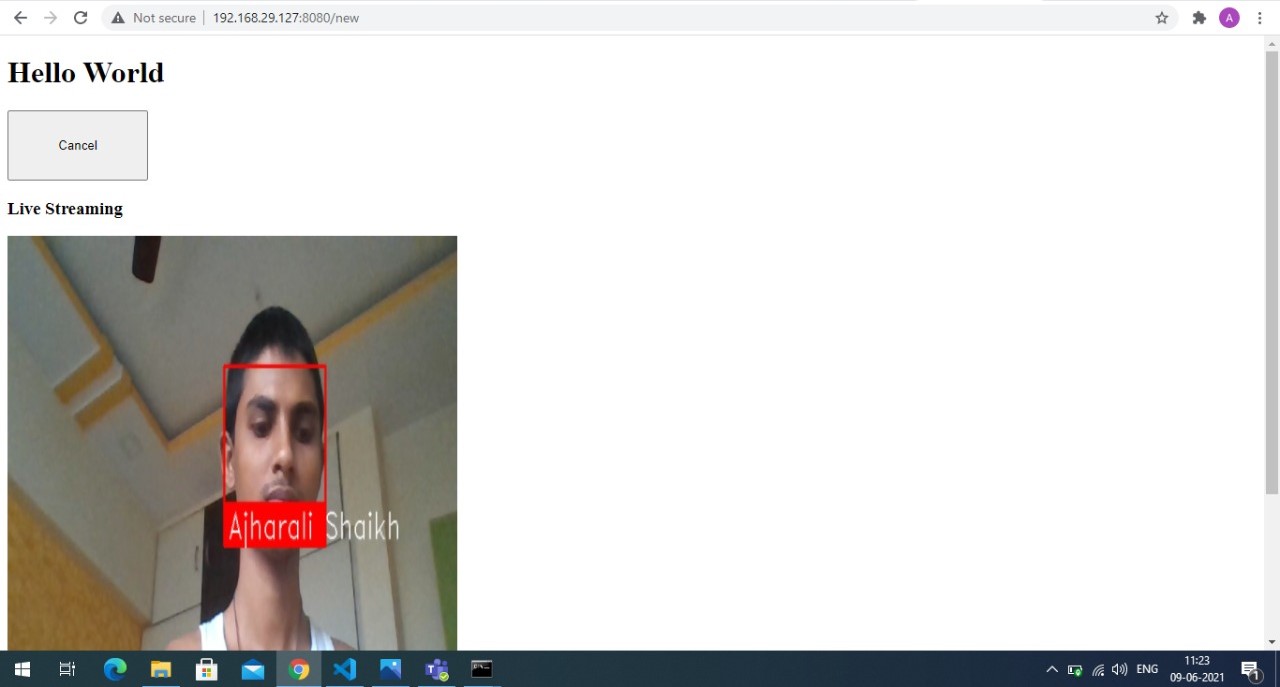
Client:-

As this project is on cloud computing specialization, I have used my laptop as a server and my mobile phone as a client.



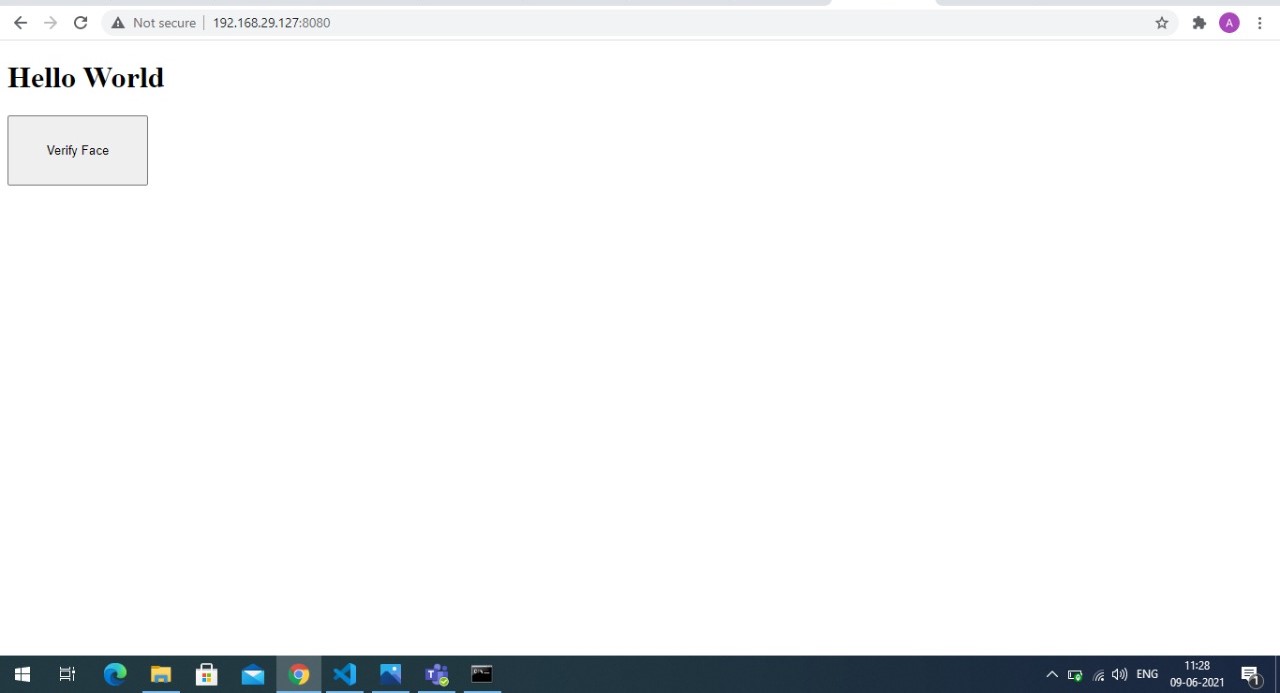
Server:-

Here my Laptop is acting like a server.

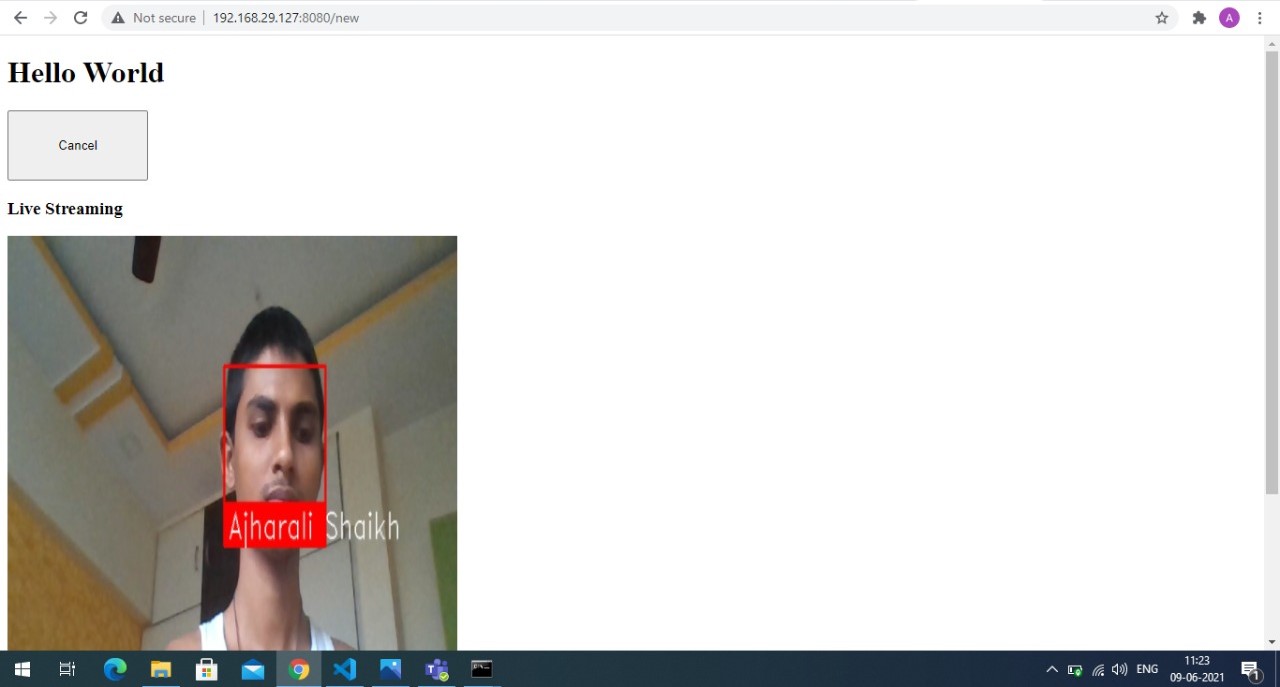


## Application flow

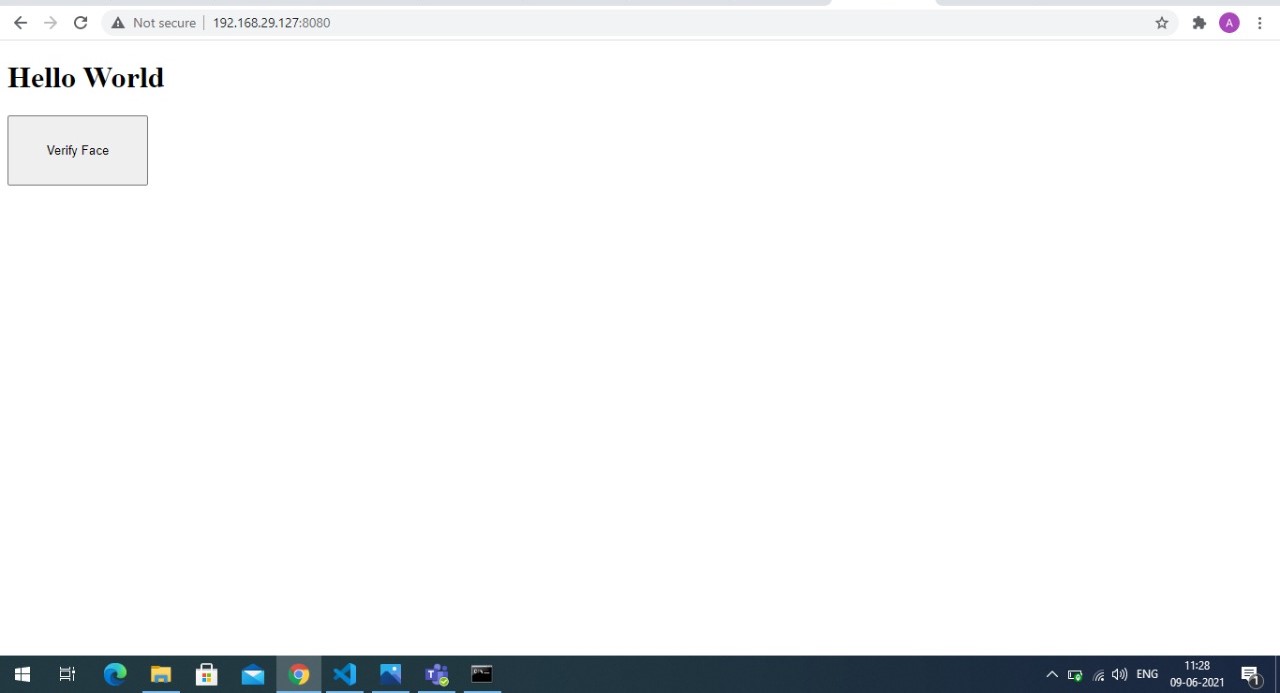
1.Start



2. Verifying faces



3.After clicking on cancel it will redirect to home page



Test Result: Success

# Approval and Authority to Proceed

We approve the project as described above, and authorize the team to proceed.

|  |  |  |
| --- | --- | --- |
| Name | Title | Date |
| Mr. Prashant Londhe | Visiting Faculty- Cloud Computing |  |
| Dr. Jyotshna Dongardive | Head of Department – Computer Science |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| Approved By |  |  | Date |  | Approved By |  |  | Date |