MINI - PROJECT REPORT ON

"IMAGE COMPONENT DETECTOR"

BY

1. Aman Raj (2193033)
2. Gaurav Kumar (2193104)
3. Aman Bharti (2193032)
4. Chinmay Mandavkar (2193088)

Under the Guidance of

Prof. Hanumant Pawar

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MIT SCHOOL of Engineering

Loni Kalbhor Pune

UNDERTAKING

We declare that the work presented in this project titled “**IMAGE COMPONENT DETECTOR**”, submitted to “**MIT ADT UNIVERSITY, SCHOOL OF ENGINEERING”** is our original work. Carried out from “October 2020 to December 2020” under the guidance of **Prof. Hamumant Pawar**.

Group Members

Aman Raj,

Gaurav Kumar,

Aman Bharti,

Chinmay Mandavkar

M.I.T. SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

LONI – KALBHOR PUNE

CERTIFICATE

This is to certify that the Mini- Project report entitled

“*Image Component detector*”

Submitted By

Aman Bharti, Gaurav Kumar, Aman Raj, Chinmay Mandavkar

is a record of Bonafide work carried out by them, under my guidance, in partial fulfillment of the requirement for the Second Year of Engineering (Computer) at M.I.T. School of Engineering, Pune under MIT Art, Design & Technology University.

Date: 29/12/2020

Place: Pune

Prof. Hanumant Pawar
Guide,
Department of CSE
M.I.T. School of Engineering
Loni Kalbhor, Pune

Dr. RajneeshKaur Sachdeo
Dean Engineering,
Head, Department of CSE
MIT School of Engineering
Loni-Kalbhor, Pune

ACKNOWLEDGEMENT

We take upon this opportunity to acknowledge the many people whose efforts and support helped me complete this project. We are deeply indebted to our mentor **Prof**. **Hanumant Pawar**. We would also like to thank our **Mini Project guide Asha ma'am** for providing us this opportunity. We also express our deepest gratitude to our mentor **Prof. Hanumant Pawar** once again for providing such meaningful and efficient knowledge. Finally, I would like to wind up by paying my heartfelt thanks all my colleagues who were there for helping each other.

Group Members

Aman Raj,

Gaurav Kumar,

Aman Bharti,

Chinmay Mandavkar

CONTENTS

ABSTRACT

Real time object detection and object segmentation methods have helped in many computer vision areas, such as scene representation & interpretation, content based image retrieval, object tracking in videos, medical applications, video surveillance, robot navigation and vehicle navigation. Object tracking is the process of locating an object or multiple objects using either a static or dynamic camera. The Image Component Detector allows the user to identify and analyse the pictorial information of images. In the paper we give you a brief about this Image Component Detector which includes its working, future scope, functions etc.

INTRODUCTION

Our project detects objects as well as the faces of people that are present in the image using Object recognition. We can detect almost every type of object by using our project. We can also detect the famous person through our project. Our project also detects any random person and for that we have to prepare a dataset and train it by using Haarcascade classifier.

- For any object detection: Choose object detection option
- For any celebrities detection: Choose celebrities detection option

We have used Django - A web framework to make the backend of our website and the front end is made through HTML and CSS. The Database we have used is Amazon AWS cloud to generate the output analyzing the information fed by the user.

PROJECT OBJECTIVE

The objective for this semester is to extract important data from images. Using this extracted information description, interpretation and understanding of the scene can be provided to the user.

REQUIREMENTS

1. Working with Python Django (Backend)
2. HTML, CSS, JS (front end)
3. Amazon web services
4. SQLite (Database)
5. Virtual Environment
6. Internet Connection

Working of Image Component Detector

In this section we describe the basic working of our project.
Firstly, we discuss all the elements used to make the project: -

Django: - Django is a Python-based free and open-source web framework that follows the model template-views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization established as a 501(c)(3) non-profit. Thus, it is used to make the backend of the website

HTML, CSS: - Hypertext Mark-up Language (HTML) is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS). Together they are used to make the front end of the website

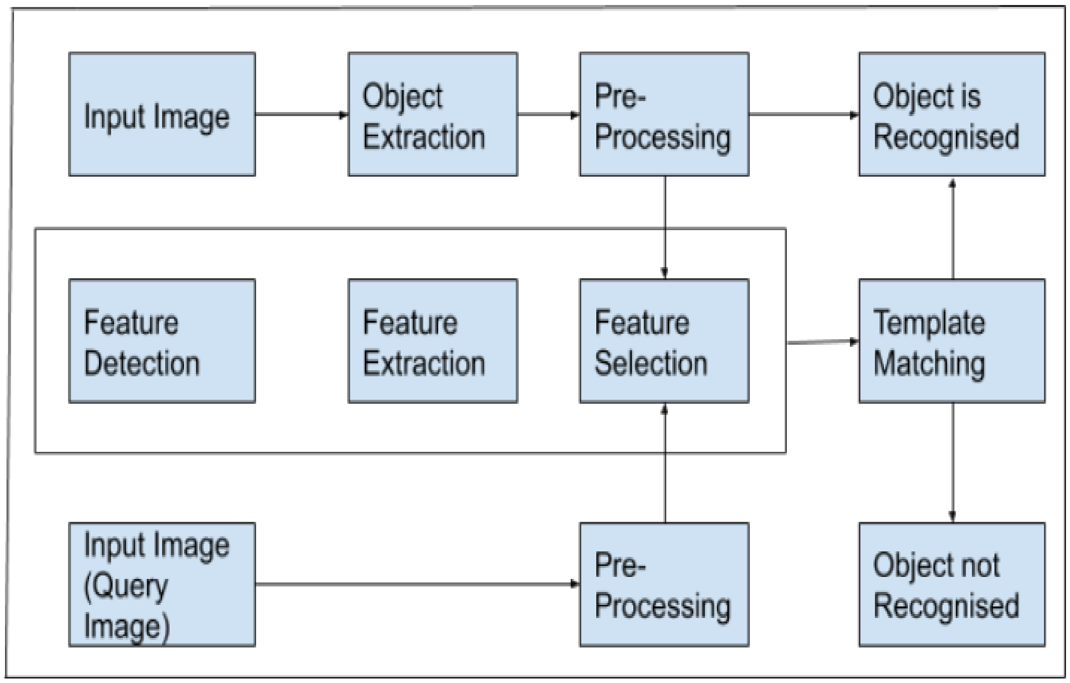
AWS Cloud: - Aws provides object recognition feature. In this project aws api is used for object recognition.

SQLite: - SQLite works great as the database engine for most low to medium traffic websites (which is to say, most websites). The amount of web traffic that SQLite can handle depends on how heavily the website uses its database. Generally speaking, any site that gets fewer than 100K hits/day should work fine with SQLite.

Working as a whole

Users have to select the image that they want to detect. After selecting the image they need to select the option whether they want to perform object detection or celebrities detection. Suppose they select object detection after that they need to click on the detect button. By clicking on the detect button the image gets refreshed and after a few seconds the new detected image replaces the previous image. Almost every component in the image recognised. SQLite database is used to store the image behind the scene.

Working Flowchart of Image Component Detector

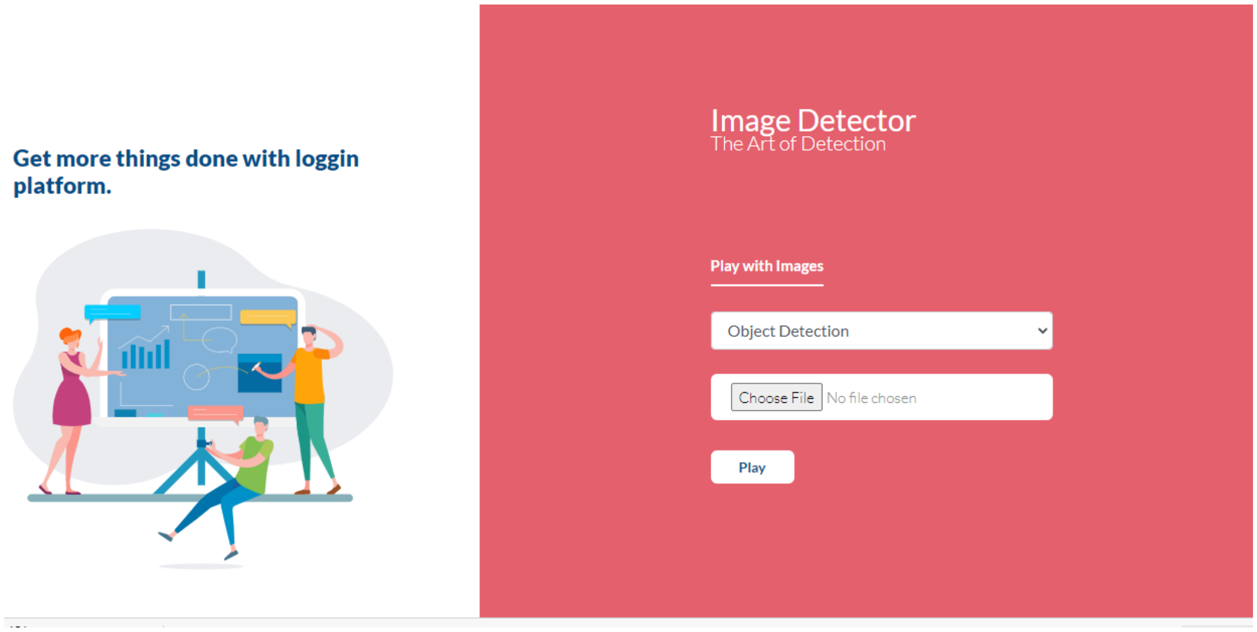


User: He/she can perform actions according to their need and the website will flow in that following direction only.

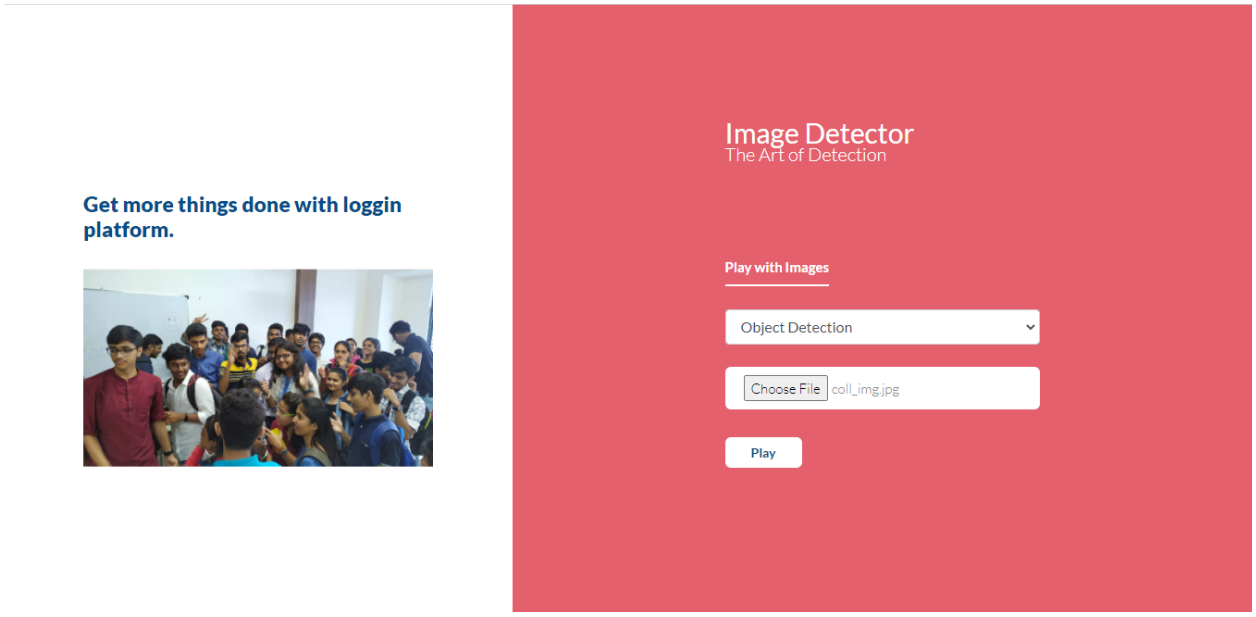
Database: Here the data entered by the user is stored to perform further calculations and display the result.

Webpage Explanation

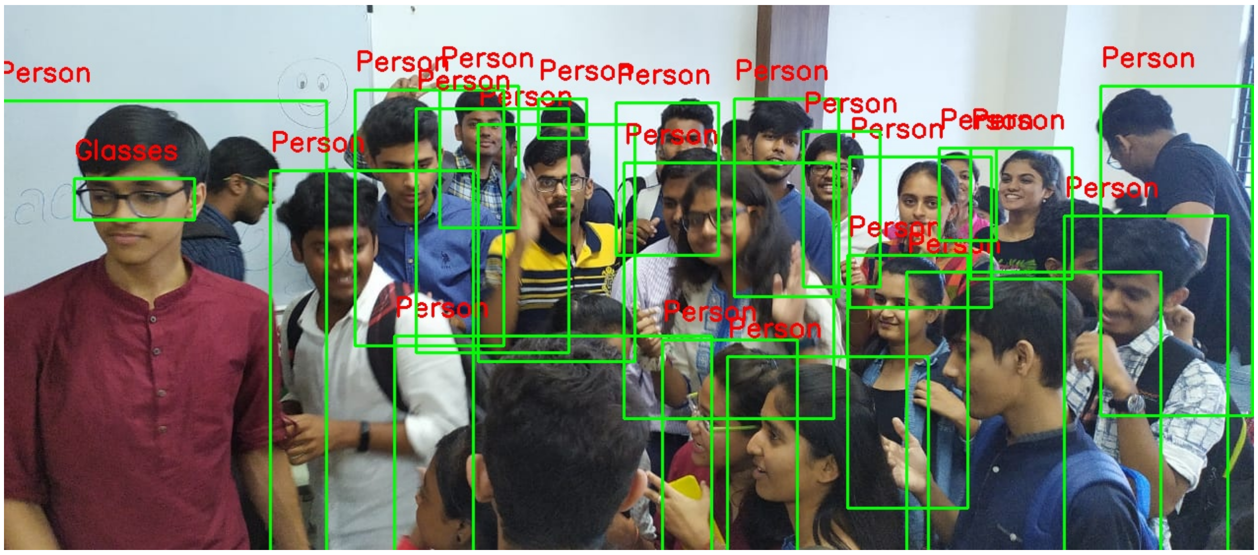
1.This is the first page of our website.



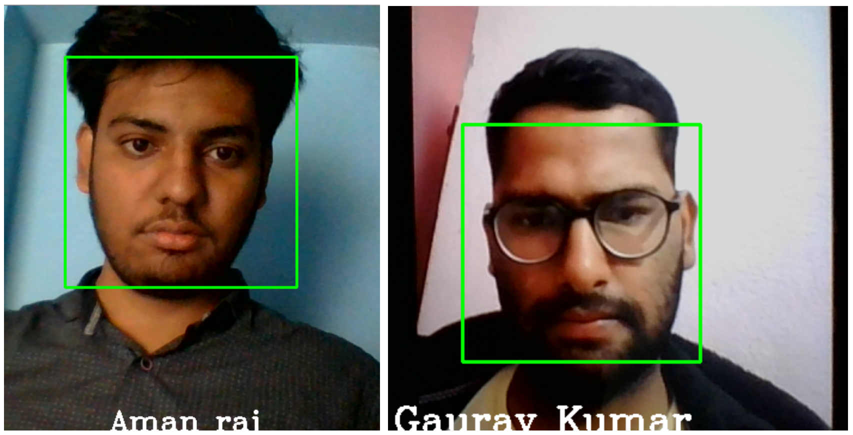
2. This will appeared when we select any image for recognition.

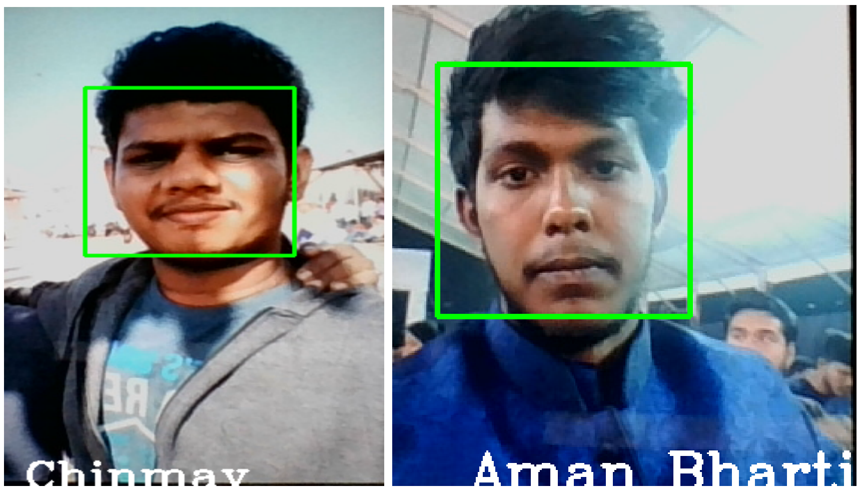


3. **This is the image recognised page appeared after clicking the detect0 button.**



4. **This is facial recognition done using Ml algorithm.**





Scope of the project

The present interface of the project is kept very simple to use. Any type of user can be familiar with it and can use it either for detecting any object or any image containing objects.

Present features

• Object Recognition .
• Face Detection and Face Recognition.
• Celebrity Recognition.
• Animal Recognition.
• Database can be modified according to the user and store information about them.
• The website is very user friendly and not at all complicated

Limitations

• There are not many websites to extract the pictographic information from the images .
• The current system doesn't work with video inputs.

Future Scope

• speech recognition software can be added to provide the user with ease.
• project can lead to an online attendance system based on face recognition.
• Detecting static objects in video sequences has a high relevance in many surveillance applications, such as the detection of abandoned objects in public areas.
• a system for the detection of static objects in crowded scenes.

Code

ImageProcessing_By_AWS.py

```
import boto3
import requests
import cv2
```

```
session = boto3.Session(profile_name="default")
Service = session.client("rekognition")


def ObjectDetection(imagePath, Service):
    session = boto3.Session(profile_name="default")
    Service = session.client("rekognition")
    image = open(imagePath, "rb").read()
    imgH, imgW = cv2.imread(imagePath).shape[:2]
    MyImage = cv2.imread(imagePath)
    response = Service.detect_labels(Image = {"Bytes": image})
    for objects in response["Labels"]:
        if objects["Instances"]:
            objectName = objects["Name"]
            for boxs in objects["Instances"]:
                box = boxs["BoundingBox"]
                x = int(imgW * box["Left"])
                y = int(imgH * box["Top"])
                w = int(imgW * box["Width"])
                h = int(imgH * box["Height"])

                MyImage = cv2.rectangle(MyImage, (x, y), (x + w, y + h), (0, 255, 0), 2)
                MyImage = cv2.putText(MyImage, objectName, (x, y - 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, [0, 0, 255], 2)
    while True:
        cv2.imshow("Detected Image", MyImage)
        if cv2.waitKey(1) == ord("q"):
            break
    cv2.imwrite("detected.jpg", MyImage)


def CelebritiesDetection(imagePath, Service):
    session = boto3.Session(profile_name="default")
    Service = session.client("rekognition")
    image = open(imagePath, "rb").read()
    imgH, imgW = cv2.imread(imagePath).shape[:2]
    MyImage = cv2.imread(imagePath)
    response = Service.recognize_celebrities(Image={"Bytes": image})
    for objects in response["CelebrityFaces"]:
        CelName = objects["Name"]
        Face = objects["Face"]
        box = Face["BoundingBox"]
        x = int(imgW * box["Left"])
        y = int(imgH * box["Top"])
        w = int(imgW * box["Width"])
        h = int(imgH * box["Height"])

        MyImage = cv2.rectangle(MyImage, (x, y), (x + w, y + h), (0, 255, 0), 2)
        MyImage = cv2.putText(MyImage, CelName, (x, y ),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, [0, 0, 255], 2)
    while True:
        cv2.imshow("Detected Image", MyImage)
        if cv2.waitKey(1) == ord("q"):
            break



image = "image11.jpg"
CelebritiesDetection(image, "Object Detection")
```

Views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from MyApi.models import MyFile

from django.conf import settings
import boto3
import requests
import cv2

def ObjectDetection(imagePath):
    session = boto3.Session(profile_name="default")
    Service = session.client("rekognition")
    image = open(imagePath, "rb").read()
    imgH, imgW = cv2.imread(imagePath).shape[:2]
    MyImage = cv2.imread(imagePath)
    response = Service.detect_labels(Image = {"Bytes": image})
    for objects in response["Labels"]:
        if objects["Instances"]:
            objectName = objects["Name"]
            for boxs in objects["Instances"]:
                box = boxs["BoundingBox"]
                x = int(imgW * box["Left"])
                y = int(imgH * box["Top"])
                w = int(imgW * box["Width"])
                h = int(imgH * box["Height"])

                MyImage = cv2.rectangle(MyImage, (x, y), (x + w, y + h), (0, 255, 0), 2)
                MyImage = cv2.putText(MyImage, objectName, (x, y - 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, [0, 0, 255], 2)
        cv2.imwrite(imagePath, MyImage)

def Celebrities_Detection(imagePath):
    session = boto3.Session(profile_name="default")
    Service = session.client("rekognition")
    image = open(imagePath, "rb").read()
    imgH, imgW = cv2.imread(imagePath).shape[:2]
    MyImage = cv2.imread(imagePath)
    response = Service.recognize_celebrities(Image={"Bytes": image})
    for objects in response["CelebrityFaces"]:
        CelName = objects["Name"]
        Face = objects["Face"]
        box = Face["BoundingBox"]
        x = int(imgW * box["Left"])
        y = int(imgH * box["Top"])
        w = int(imgW * box["Width"])
        h = int(imgH * box["Height"])

        MyImage = cv2.rectangle(MyImage, (x, y), (x + w, y + h), (0, 255, 0), 2)
        MyImage = cv2.putText(MyImage, CelName, (x, y ),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, [0, 0, 255], 2)


    cv2.imwrite(imagePath, MyImage)
```

```
def Home(request):
    if request.method == "POST":
        img = request.FILES["image"]
        service = request.POST["service"]

        print(service)
        data = MyFile.objects.create(image = img)
        path = str(settings.MEDIA_ROOT) + "/" + data.image.name
        if service == "Object Detection":
            ObjectDetection(path)

        if service == "Cel.. Det...":
            Celebrities_Detection(path)

        url = "http://127.0.0.1:8000" + data.image.url
        return redirect(url)
    return render(request, "index.html")
```

Admin.py

```
from django.contrib import admin
from MyApi.models import MyFile

admin.site.register(MyFile)
```

Apps.py

```
from django.apps import AppConfig


class MyapiConfig(AppConfig):
    name = 'MyApi'
```

models.py

```
from django.db import models

class MyFile(models.Model):
    image = models.ImageField()
```

migrations.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='MyFile',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('image', models.ImageField(upload_to='')),
            ],
        ),
    ]
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    {% load static %}
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Component Detector</title>
    <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome-all.min.css'
%}">
```

```
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/theme.css' %}">

</head>
<body>
<div class="form-body">
    <div class="row">
        <div class="img-holder">
            <div class="bg"></div>
            <div class="info-holder">
                <h3>Get more things done with loggin platform.</h3>
                <img id = "outImage" src="{% static 'images/graphic.svg' %}" alt="">
                </div>
            </div>
            <div class="form-holder">
                <div class="form-content" style="background-color: #e4606d">
                    <div class="form-items">
                        <div class="website-logo-inside">
                            <a href="">
                                <div class="logo">
                                    <p style="color: white; font-size: 30px;"><b>Image
Detector</b></p>
                                    <p style="color: white; font-size: 20px; margin-top: -27px;">The
Art of Detection</p>
                                </div>
                            </a>
                        </div>
                        <div class="page-links">
                            <a href=" " class="active">Play with Images</a>
                        </div>
                        <form method="post" enctype="multipart/form-data">
                            {% csrf_token %}
                            <select class="form-control" name="service">
                               <option>Choose Your Services</option>
                               <option>Object Detection</option>
                               <option>Face Detection</option>
                               <option>Cel.. Det...</option>

                            </select><br>
                            <input class="form-control" id ="inputImage" type="file"
onchange="ChangeImage(event)" name="image" required>
                            <div class="form-button">
                                <button id="submit" type="submit" class="ibtn">Play</button>
                            </div>
                        </form>


                    </div>
                </div>
            </div>
        </div>
    </div>

<script>
    function ChangeImage(e) {
        var image = document.getElementById("outImage");
        image.src = URL.createObjectURL(e.target.files[0]);

    }
</script>

<script src="{% static 'js/jquery.min.js' %}"></script>
<script src="{% static 'js/popper.min.js' %}"></script>
<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/main.js' %}"></script>
</body>
</html>
```

Dataset.py

```
import cv2
import numpy as np

face_classifier =
cv2.CascadeClassifier('C:/Users/rajan/anaconda3/Lib/site-packages/cv2/data/haar
cascade_frontalface_default.xml')

def face_extractor(img):
   gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   faces = face_classifier.detectMultiScale(gray,1.3,5)

   if faces is():
       return None

   for(x,y,w,h) in faces:
       cropped_face = img[y:y+h, x:x+w]

   return cropped_face


cap = cv2.VideoCapture(0)
id = 1
count = 0

while True:
   ret, frame = cap.read()
   if face_extractor(frame) is not None:
       count+=1
       face = cv2.resize(face_extractor(frame),(200,200))
       face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

       file_name_path = 'C:/Users/rajan\Desktop/face_detection/dataset/1/'+
str(id)+"."+str(count)+'.jpg'

       cv2.imwrite(file_name_path,face)


cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
       cv2.imshow('Face Cropper',face)
   else:
       print("Face not found")
       pass

   if cv2.waitKey(1)==13 or count==100:
       break

cap.release()
cv2.destroyAllWindows()
print('Samples Colletion Completed ')
```

Training.py

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
```

```
data_path = 'C:/Users/rajan/Desktop/face_detection/dataset/1/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))]

Training_Data, Labels = [], []

for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)

model = cv2.face.LBPHFaceRecognizer_create()

model.train(np.asarray(Training_Data), np.asarray(Labels))

print("Dataset Model Training Completed ")
```

Detection.py

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

data_path = 'C:/Users/rajan/Desktop/face_detection/dataset/1/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))]
Training_Data, Labels = [], []

for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)

model = cv2.face.LBPHFaceRecognizer_create()

model.train(np.asarray(Training_Data), np.asarray(Labels))

print("Dataset Model Training Complete!!!!!")

face_classifier =
cv2.CascadeClassifier('C:/Users/rajan/anaconda3/Lib/site-packages/cv2/data/haar
cascade_frontalface_default.xml')
```

```
def face_detector(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return img,[]

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,0),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))

    return img,roi

cap = cv2.VideoCapture(0)
while True:

    ret, frame = cap.read()

    image, face = face_detector(frame)

    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        result = model.predict(face)

        if result[1] < 500:
            confidence = int(100*(1-(result[1])/300))



        if confidence > 82:
            cv2.putText(image, "Aman raj", (250, 450), cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 255), 2)
            cv2.imshow('Face Cropper', image)

        else:
            cv2.putText(image, "Unknown", (250, 450), cv2.FONT_HERSHEY_COMPLEX,
1, (0, 0, 255), 2)
            cv2.imshow('Face Cropper', image)


    except:
        cv2.putText(image, "Face Not Found", (250, 450),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)
        cv2.imshow('Face Cropper', image)
        pass

    if cv2.waitKey(1)==13:
        break
```

```
cap.release()
cv2.destroyAllWindows()
```

Conclusion

The basic idea of the project is to provide the best analysis of the input provided by the user with a very few clicks. The Project we plan for in the future has a much larger scope than this but undoubtedly the present website will fulfill the basic functions on which our concepts are based on. There are a few cons in the project, most of them will be optimized to the best in the future.

Reference

- YouTube
- **Hanumant Pawar Sir**
- Google