GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Blind Aid System Using Voice Based Assistance

A Report for the Evaluation 3 of Project 2

Submitted by

GAURAV PANDEY

(1613101276/16SCSE101289)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under the Supervision of

MR. SOUMALYA GHOSH

ASSISTANT PROFESSOR

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"Blind Aid System Using Voice Based Assistance"** the Bonafide work of **"Gaurav Pandey (1613101276)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Mr. SOUMALYA GHOSH
Assistant professor,
School of Computing Science & Engineering

TABLE OF CONTENTS

1. ABSTRACTION

This project presents an approach of aiding blind persons by the help of a voice-based assistance based on cloud technology. Basic Prototype consist of a wearable goggle which are connected through cloud service and based on the visuals provided by the camera, voice-based assistance will be provided to the user. What this project is heading towards is providing a solution to the blind person's vision-based decision problem. With the help of Machine Learning Algorithms, IoT devices and Internet, this project can detect things ranging from friendly faces and their current emotion to unknown person's gender and age. Prototype's compute unit will be based on a single board computer which will be connected to the cloud services for data processing loads. Friendly faces can be added to the database by voice commands. This will help disables to make decisions based on the person's face in front of him. As this project is relying on cloud services for much of its processing work, it can be made using low-power computers which will be cost efficient. This will help economically weak person afford one for themselves.

2. INTRODUCTION

2.1 Overall description

Blindness is the condition of lacking visual perception due to Physiological or Neurological factors. There are different types of Blindness where Complete Blindness is that you cannot see at all. Partial Blindness is that you have limited vision. There are many problems for Visually Impaired people in Society, they face many types of hurdles in performing every day routine works. The Blind people are like whom we can say that lost his/her way on the journey and don't know where and how to pass easy life. Various Technology have been developed to help the Visually Impaired people. Day by day advancement in the field of Science introduces new technologies for Benefits of Visually Impaired people to make their life Easy and Comfortable.

2.2 Purpose

This device once developed, will give the ability to the blind person to see the world around them and provide them with the cognitive ability. With the voice-based assistant, blind person will now be able to deal with the situation in a better way. Assisting blind person with concise and detail of the view in front of them.

2.3 Motivation and Scope:

There has some astounding breakthrough in the development of new technologies to assist the blind which aim to replicate the function of the eye, but this project takes a totally different approach. The Cognitive Aid System for Blind People (CASBLiP) uses lasers and digital video images to create a three-dimensional acoustic map which, when relayed through headphones, enables users to **"see"** the

world with sound. Laser and digital video camera images will be used to analyse the distance of obstacles and help to predict the movement of people and motor vehicles as the user nears them. This spatial information will then be transformed and presented to the user via headphones as an "audible map".

The signals received via headphones will guide and assist the user to negotiate and navigate the obstacles and dangers of the outside world.

The user will also wear glasses with miniature video cameras mounted on them. These will provide the necessary video vision.

Once developed, this is how it will work as you move around, the sounds received via headphones will alter and the stereo audio system will enable you to interpret sounds and then place them in accordance with their distance to you. For example, as you walk away from an object, the sound will decrease. Walk closer to an object and the sound will increase. If an object is on your right, you'll hear it on your right and you will also be warned via audio tone to get out of the way if something is headed straight for you.

2.4 Hardware and Software Requirements

2.4.1 Hardware Requirements

The following project comprises of different hardware components at different level:

- **Server-Side Hardware Requirement:**

  Processor: Intel Xeon or AMD Epic server processor with at least 64 cores.

  RAM: At least 128GB of ECC memory.

  Storage Device: Ultra-fast SSDs with minimum of 2TB Space for Database purpose.

  Graphic Processing Unit: Nvidia Quadro GPUs with at least 12GB of dedicated memory.

Network Card: 100Gbps server grade access card.

- **Client-Side Hardware Requirement:**

  Single Board Computer: Raspberry Pi 4, Asus Tinkerboard, Banana Pi or equivalent

  Camera Module: At least 5mega-pixel camera sensor

  GSM Module: At least 3G or Higher generation support

  Audio Interface: Bluetooth compatible headset

  Power Unit: 4000mAh or higher battery pack with voltage overload sensor

2.4.2 Software Requirement:

Software requirement will be different for both client and server system:

- **Server-Side Software Requirements:**

  Operating System: Unix based operating system such as linux.

  Dependencies and Libraries: Python3, Django, Nginx, Gunicorn, Keras, Tensorflow, OpenCV,

  Database: PostgreSQL

- **Client-Side Software Requirements:**

  Operating System: Linux Distro

  Dependencies and Libraries: Python3, Speech-Recognition(python-library), Selenium, OpenCV, gTTS.

3. LITERATURE SURVEY

1. Emotion wouldn't be emotion without the urge to act and express (or the urge not to act, in the case of sadness). At best it would be lively cognition, but not anything that truly maws us, as the term e-motion implies. Because of emotion, we may be inclined to move toward or away from someone, to act or give up, to cry or sing, to help or hide. Action tendencies may exert subtle influences on communicative behavior that are not as overt as the more prototypical expressions of emotion such as facial expressions. For instance, the tendency to move toward or away from others has its communication counterpart in responsiveness (Davis, 1982), immediacy (Andersen, 1985), interpersonal warmth (Andersen & Guerrero, 1998a), communication apprehension (McCroskey, 1982), or shyness (Crozier, 1990). We can we the tendency toward hyper- or hypo-activity in animated or relaxed communicator styles.

2. Visual impairment is the only disability category within which women are significantly more likely than men to have been victims of violent crime (especially striking because, among people with and without disabilities, women are typically less likely than men to be victimized). Females (31.9 per 1,000) had a higher rate of total violent victimization than males (22.8 per 1,000). In all other disability groups, victimization rates for males and females were similar. Visual impairment is the only disability category within which people are significantly less likely than people without disabilities to report to police when they have been the victim of a violent crime

3. The assistive device for a blind person needs to provide several features, among them: a clear and concise information within seconds, a consistent performance during day and night time, works indoors and outdoors; detects objects from close to further than 5 m; and detects the static and dynamic objects in order to handle any sudden appearance of objects; otherwise, the user's life is at risk.

4. PROBLEM STATEMENT

Making a device which will be able to help visually impaired person to know about their surrounding by the mean of audio conversion of visual details by the use of machine learning algorithms while being portable and light as well as cost efficient so that a middle-class person could afford one. Initial setup will be done by the distributor while rest of the system should be able to run on voice-based command. User interface should be based on voice. Usability of the Device greatly depends on how precisely camera feed is converted into voice-based details of the feed. Even the accuracy of machine learning model should be greater enough so that no false detection should be made as the user can not cross-validate the output generated by the device. Output should be generated in real-time so that the user can make judgement without any delay.

5. PROPOSED MODEL

This project is based in providing aid to the blind disabled. Providing details about the person who is in front of the disable and helping them by providing basic characteristics of the person. This is done by the use a wearable goggle, a computer unit and a headset unit. Providing the disable the information required to make decision based on the person's characteristics. As this solution will be cloud based this means that we can easily lower down the cost of manufacturing the wearable goggles.

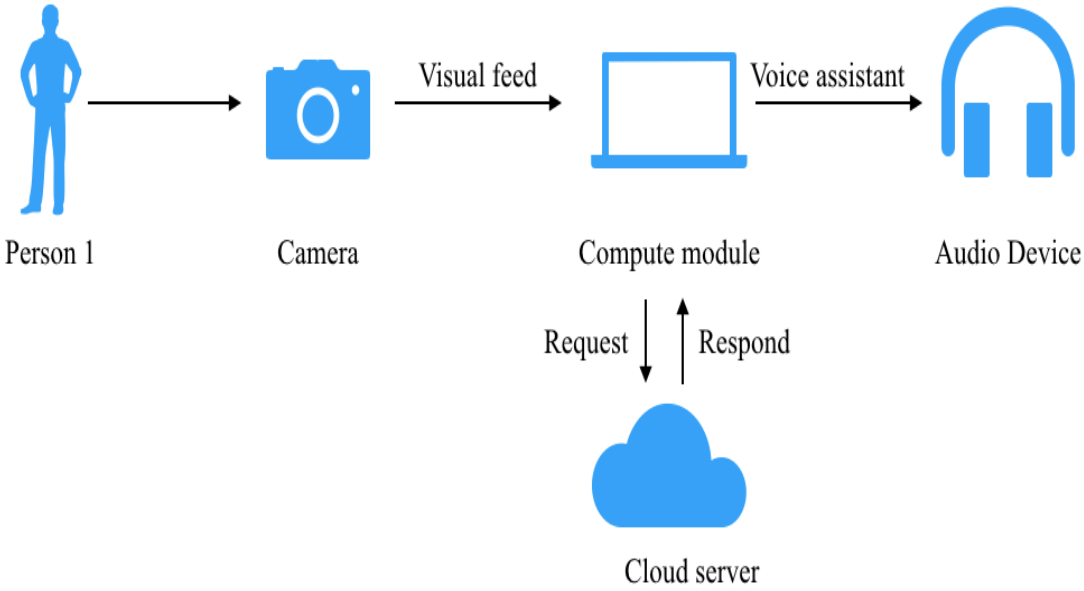


Fig. 5.1 Voice based assistant for blind people

As in the fig 5.1, we can see the basic unit of the system, i.e. compute module (SBC computer), camera, audio device and most important cloud server. These all module are either connected directly like in the case camera or by the mean of wireless technology like Bluetooth for audio device and internet for the connection with the cloud server.

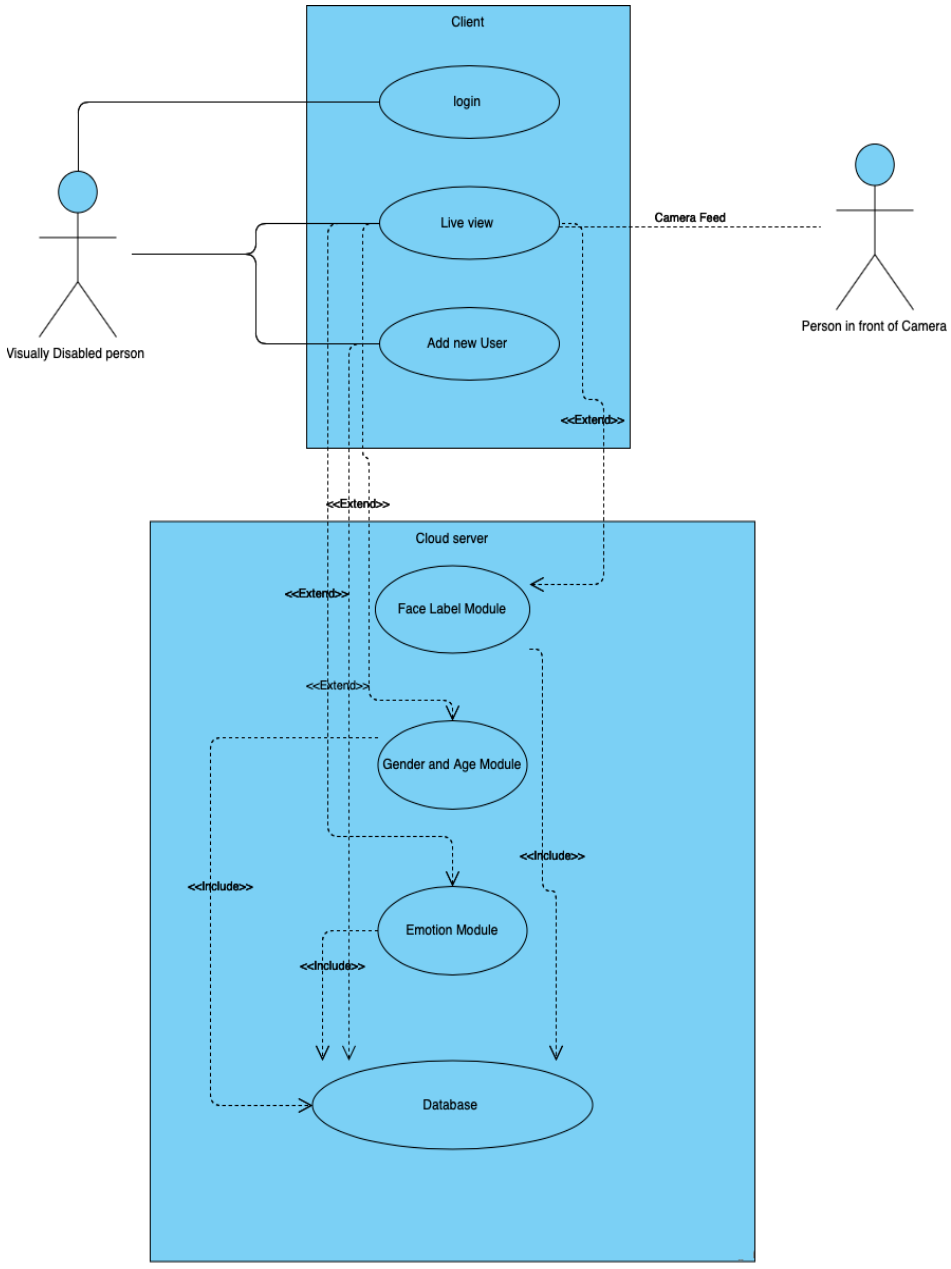


Fig. 5.2 Use-Case Diagram of the System

Basic use-case of the system can be generated out from the fig 5.2, like what happens at cloud side or at client side. User-Interface has to be voice based as per the scenario. This can be achieved by the use of various python libraries and deep learning modules. As well as for the connection between client side and cloud server can be done by the use of cloud application API or in our case, using selenium to interact with the web-application for the classification purpose and retrieving the data which further can be converted into speech and presented to the user.

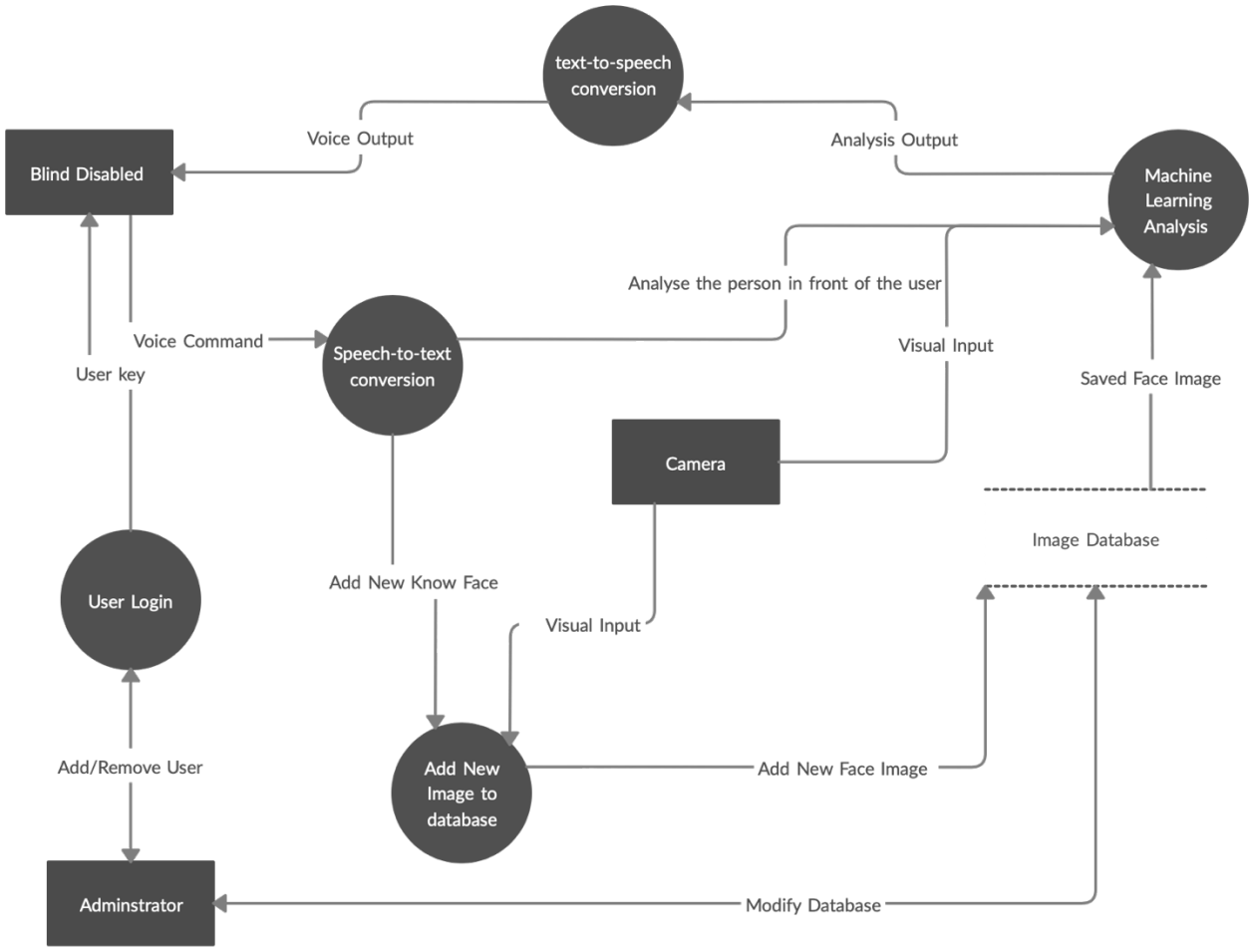


Fig 5.3 Dataflow diagram of the System

6. IMPLEMENTATION

6.1 Machine Learning Model:

This project uses the four type of Machine learning models which are as followed:

Facial-Emotion Recognition:

Dataset Used: FER2013 from Kaggle

This machine learning model was used to detect the emotion of the human from the image send to it. This model achieved the accuracy of 65% on the given dataset. Though it's accuracy can be increased but it all depends upon the quality of the dataset used. Emotion were classified into seven labels- 'happy', 'sad', 'neutral', 'surprised', 'angry', 'disgusted' and 'feared'. All the code to this model is presented in later part of this report.

Face Recognition:

Library Used: face_recognition

This module detects the face of the subject in the visual input and labels it according the name of the person. If the person data is not present in the database, module will label them unknow.

Gender Classification:

Library Used: Caffe model

This module predicts the gender of the person. This module uses Caffe model which is pretrained ML model.

Age Classification:

Library Used: Caffe Model

This module predicts the age group of the person in front the user. Predicted age group is obtained by using Caffe model. Caffe models are pretrained ML model which can be used to obtain the result.

6.2 Django Server:

- Basically, a Django server will be running on the cloud system.
- This Django server will only allow particular IP-address to connect to it.
- Django server will be running four machine learning modules.
- As server is static and therefore it can be made as powerful as possible, giving result in real-time.
- A copy of user's database is kept in the cloud encrypted by the unique key.
- This key is unique for each device connected to the server.
- All the Machine Learning Module will run on this server.

6.3 Selenium web driver with Speech Recognition:

Selenium WebDriver is the successor to Selenium RC. Selenium WebDriver accepts commands (sent in Selenese, or via a Client API) and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results. Most browser drivers actually launch and access a browser application (such as Firefox, Google Chrome, Internet Explorer, Safari, or Microsoft Edge); there is also an HtmlUnit browser driver, which simulates a browser using the headless browser HtmlUnit.

For the purpose of speech-based user interface, we are using python library, **Speech_Recognition** which with the help of google voice recognition converts the speech into text. From these text, particular words are looked after. These words are associate with particular functionality of the system. These keywords are- "add person", "remove person", "what is in front of me".

- If the person says "add person", what our script does is as followed:

  (1) First the image is captured using the camera module.

(2) Selenium web driver opens up the browser in headless mode and after login it goes to the add new image field.

(3) There it adds the image it captured using the camera. Image will be then uploaded to the database.

- If the person says “what is in front of me” :

(1) Image will be captured using camera module

(2) Selenium web driver will open up the browser in headless mode and after login, it will go into ‘see live feed’ block.

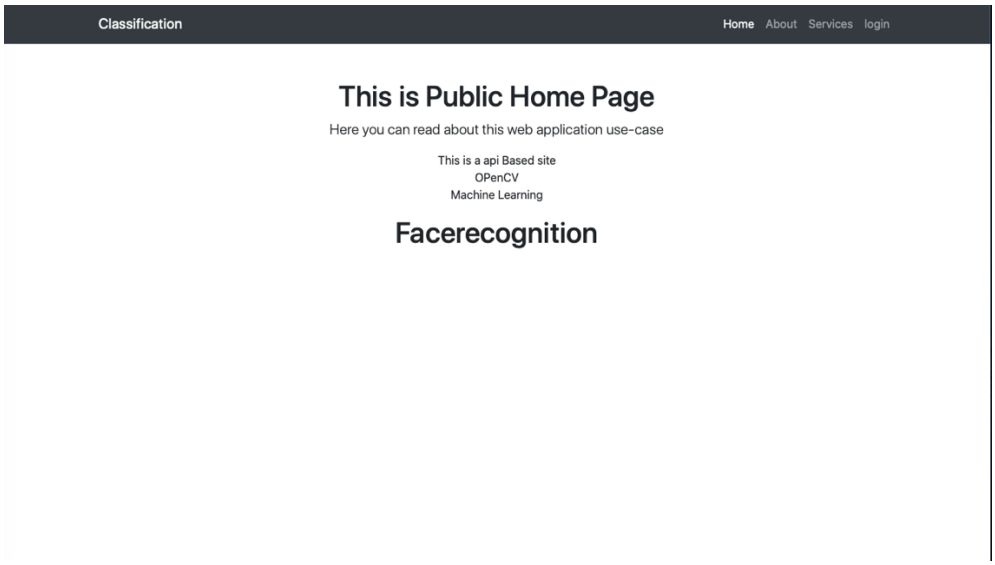
(3) Image will be uploaded there.

(4) Then classification models will work on the image and will generate the labels accordingly.

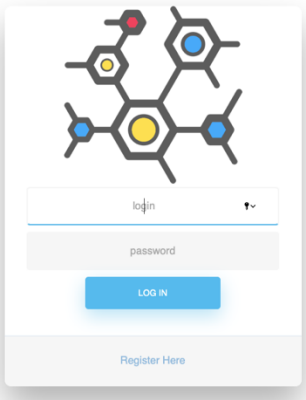
(5) Once labels are generated, selenium will pass those labels to text-to-speech module were it will be converted to speech for the blind person to hear.

7. OUTPUT

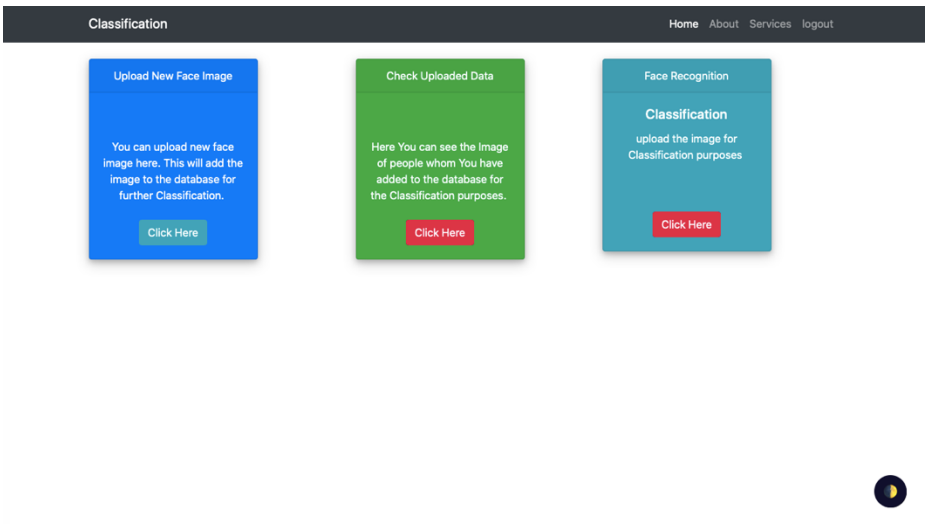
Django Web-Application:



Home Page



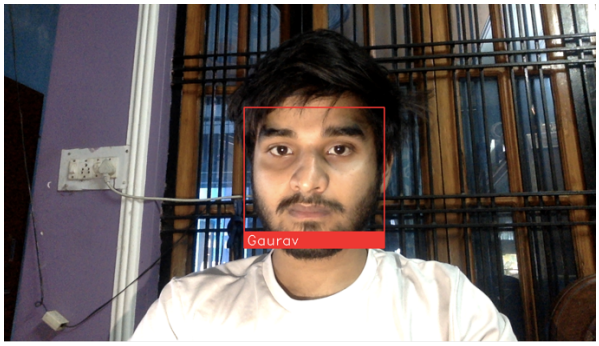
Login Page



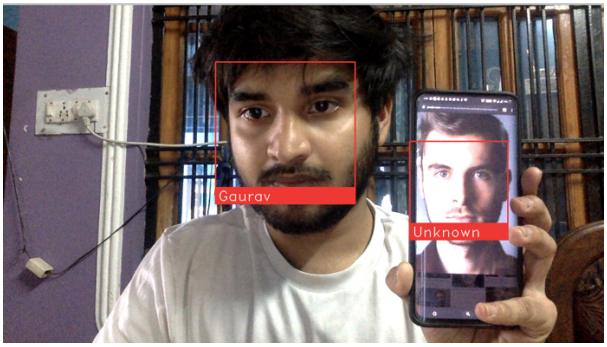
Homepage(after login)

Machine learning Classification:

Face-Recognition module:

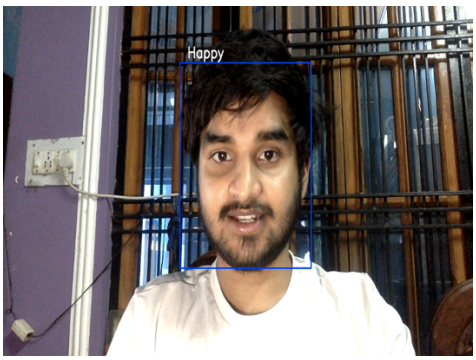


Face detected with correct label

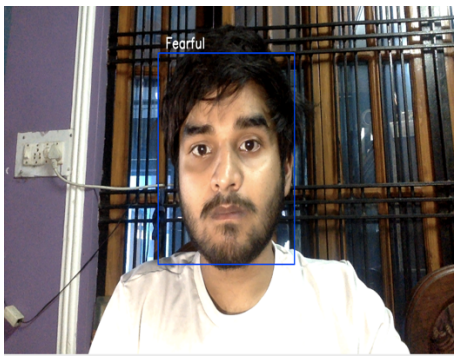


Face detected with unknown face separately

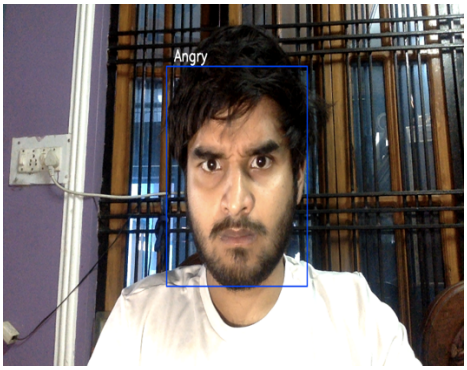
Facial-Emotion-Recognition:



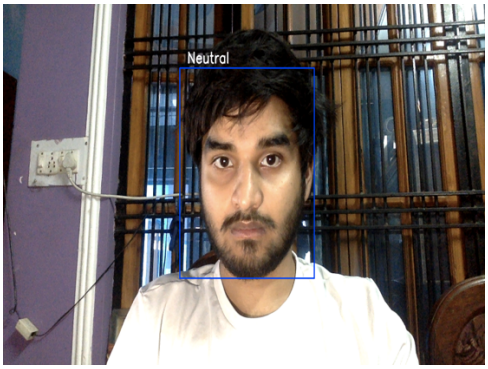
Happy Emotion



Fearful Emotion



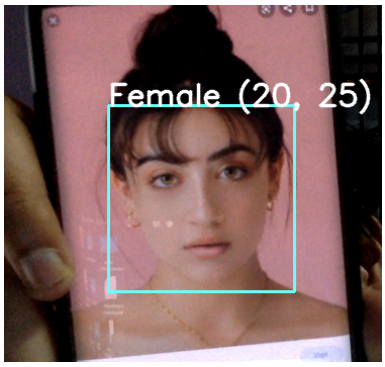
Angry Emotion



Neutral Emotion

Gender-Age Classification:

Correct gender as well as age-range for male



Correct gender as well as age-range for female

8. CODE

8.1 Django Server:

8.1.1 Base Urls.py:

```
from django.contrib import admin
from django.urls import path,include
urlpatterns = [path('admin/', admin.site.urls),
path('',include('filehandling.urls')),]
```

8.1.2 Application-Urls.py:

```
from django.contrib import admin
from django.urls import path,include
from . import views
from django.conf import settings
from django.conf.urls.static import static
app_name='filehandling'
urlpatterns=[
    path('',views.homeView,name="home"),
    path('home_auth/',views.home_authView,name="home_auth"),
    path('login/',views.loginView,name="login"),
    path('login/register',views.registerView,name="register"),
    path('logout/',views.logoutView,name="logout"),
    path('imageupload/',views.image_uploadView,name="image_upload"),
    path('viewimage/',views.imageView,name="see-image"),
    path('emotion-analysis/',views.emotionView,name="emotion-analysis"),
    path('gender-analysis/',views.genderView,name="gender-analysis"),
```

```
    path('age-analysis/',views.ageView,name="age-analysis"),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

8.1.3 Settings.py:

```
import os
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = 'v@fdh0n3s$+5tpr4f*sg@)ra*+v(3y_gqz9cij)nn#0v+x)7va'
DEBUG = True
ALLOWED_HOSTS = []
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'filehandling',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
]
ROOT_URLCONF = 'sample.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'sample.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


AUTH_PASSWORD_VALIDATORS = [
```

```
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

STATIC_URL = '/static/'
#STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

```
STATICFILES_DIRS = ( os.path.join(BASE_DIR, 'static'), )

STATICFILES_DIRS = [ os.path.join(BASE_DIR, 'static'), ]

MEDIA_URL='/media/'

MEDIA_ROOT=os.path.join(BASE_DIR,'media')
```

8.1.4 Views.py:

```
from django.shortcuts import render , redirect

from django import forms

import cv2

from django.contrib.auth import authenticate , login , logout

from django.contrib.auth.models import User

from sample import settings

from .forms import UserForm,ImageForms

from django.contrib.auth.decorators import login_required

from django.shortcuts import render, redirect

from django.contrib import messages

from .models import ImageForm

from MachineLearning.detectface import detect_face

from numpy import asarray

def homeView(request):

    return render(request,'filehandling/home.html')

def registerView(request):

    form = UserForm()

    if request.method == 'POST':

        form = UserForm(request.POST)

        if form.is_valid():
```

```
            form.save()
            return redirect('filehandling:login')
    else:
        form = UserForm()
    params = {
        'forms' : form
    }
    return render(request, 'login_page/register.html', params)
def loginView(request):
    if request.method == 'POST':
        user1 = request.POST.get('login')
        passwd = request.POST.get('password')
        user = authenticate(request , username = user1 , password = passwd )
        if user is not None:
            login(request , user)
            return redirect('filehandling:home_auth')
        else:
            messages.error(request, 'Oops, Wrong username or password, please try a diffrerent one')
    return render(request , 'login_page/login.html')
def logoutView(request):
    logout(request)
    return render(request,'filehandling/home.html')
@login_required(login_url='filehandling:login')
def home_authView(request):
    return render(request,'filehandling/home_auth.html')
@login_required(login_url='filehandling:login')
```

```
def image_uploadView(request):
    form=ImageForms()
    if request.method=='POST':
        form=ImageForms(request.POST,request.FILES)
        current_user = request.user
        if form.is_valid():
            username=form.cleaned_data['name']
            userpicture=form.cleaned_data['image']
           #if(detect_face(userpicture)):
               #print(true)

m=ImageForm(id=None,user_id=current_user,user_name=username,image_file=userpicture)
            m.save()
            """m=ImageForm(pk=current_user.id)
            m.user_name=form.cleaned_data['name']
            m.image_file=form.cleaned_data['image']
            m.save()"""
            messages.success(request,'Image was successfully uploaded')

            redirect('filehandling:home_auth')
        else:
             messages.error(request,'All fields are required')
    else:
        form = ImageForms()
    params = {
            'forms' : form
```

```
            }
    return render(request,'fileupload/fileupload.html',params)
def imageView(request):
    current_user = request.user
    img=[e.image_file for e in ImageForm.objects.filter(user_id=current_user)]
    path=settings.MEDIA_URL
    return render(request,'viewimage/viewimage.html',{"img":img, 'media_url':path})
def detectWithWebcam(request):
    # Get a reference to webcam #0 (the default one)
    video_capture = cv2.VideoCapture(0)
    # Load a sample picture and learn how to recognize it.
    images=[]
    encodings=[]
    names=[]
    files=[]
    nationalIds=[]
    prsn=Person.objects.all()
    for crime in prsn:
        images.append(crime.name+'_image')
        encodings.append(crime.name+'_face_encoding')
        files.append(crime.picture)
        nationalIds.append(crime.national_id)
    for i in range(0,len(images)):
        images[i]=face_recognition.load_image_file(files[i])
        encodings[i]=face_recognition.face_encodings(images[i])[0
    # Create arrays of known face encodings and their names
```

```
    known_face_encodings = encodings
    known_face_names = names
    n_id=nationalIds
    while True:
        # Grab a single frame of video
        ret, frame = video_capture.read()

        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_frame = frame[:, :, ::-1]

        # Find all the faces and face enqcodings in the frame of video
        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

        # Loop through each face in this frame of video
        for (top, right, bottom, left), face_encoding in zip(face_locations, 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]:
        ntnl_id = n_id[best_match_index]
        person = Person.objects.filter(national_id=ntnl_id)
        name = known_face_names[best_match_index]+', Status: '+person.get().status




        if(person.get().status=='Wanted'):
            image= ImageForm.objects.create(
                name=person.get().name,
                national_id=person.get().national_id,
                address=person.get().address,
                picture=person.get().picture,
                status='Wanted',
                latitude='20202020',
                longitude='040404040'
            )
            image.save()
    # 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)
        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()
    return redirect('/success')
```

8.1.5 Forms.py

```
from django.contrib.auth.forms import UserCreationForm
from .models import ImageForm
from django.contrib.auth.models import User
from django import forms
class UserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username' , 'email','password1' ,'password2'  ]
class ImageForms(forms.Form):
    name=forms.CharField(max_length=100,label='')
    image=forms.ImageField(label='')
    name.widget.attrs.update({'class':'form-control','placeholder':"Person's Name"})
    image.widget.attrs.update({'class':'file-upload'})
```

8.1.6 Models.py

```
from django.db import models
from django.contrib.auth.models import User
class ImageForm(models.Model):
    user_id=models.ForeignKey(User, null = True , on_delete=models.CASCADE)
    user_name=models.CharField(max_length=100)
    image_file=models.ImageField(upload_to='Images/user_images/')
```

8.1.7 Templates:

8.1.7.1 Base.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta charset="utf-8">
    <title>{% block title %}Gaurav's Site{% endblock %}</title>
  </head>
  <body>
    <div class="sidenav">
      <a href="/">Home</a>
      <a href="/create">Create</a>
      <a href="/2">View</a>
```

```
    </div>
    <div id='content', name='content',class="main">
      {%block content%}
      {% endblock %}
    </div>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
  </body>
</html>
```

8.1.7.2 Home.html:

```
{% load static %}
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

```
  <meta name="description" content="">
  <meta name="author" content="">
  <title>Classification</title>
  <!-- Bootstrap core CSS -->
  <link href="{% static 'bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
</head>
<body>
  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top">
    <div class="container">
      <a class="navbar-brand" href="#">Classification</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item active">
            <a class="nav-link" href="/">Home
              <span class="sr-only">(current)</span>
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">About</a>
          </li>
```

```
        <li class="nav-item">
          <a class="nav-link" href="#">Services</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="{% url 'filehandling:login' %}">login</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
<!-- Page Content -->
<div class="container">
  <div class="row">
    <div class="col-lg-12 text-center">
      <h1 class="mt-5">This is Public Home Page</h1>
      <p class="lead">Here you can read about this web application use-case</p>
      <div class="container-sm">This is a api Based site</div>
      <ul class="list-unstyled">
        <li>OPenCV</li>
        <li>Machine Learning</li>
      </ul>
    </div>
  </div>
</div>
<div class="container-xl">
  <div class="row">
```

```
    <div class="col-lg-12 text-center">
    <h1>Facerecognition</h1>
   </div>
</div>
  </div>
  <!-- Bootstrap core JavaScript -->
  <script src="{% static 'jquery/jquery.slim.min.js' %}"></script>
  <script src="{% static 'bootstrap/js/bootstrap.bundle.min.js' %}"></script>
</body>
</html>
```

8.1.7.3 Login.html:

```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="ltr">
  <head>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">

    <style media="screen">
      html {
  background-color: #56baed;
}
```

```
body {
  font-family: "Poppins", sans-serif;
  height: 100vh;
}

a {
  color: #92badd;
  display:inline-block;
  text-decoration: none;
  font-weight: 400;
}

h2 {
  text-align: center;
  font-size: 16px;
  font-weight: 600;
  text-transform: uppercase;
  display:inline-block;
  margin: 40px 8px 10px 8px;
  color: #cccccc;
}
/* STRUCTURE */
.wrapper {
  display: flex;
  align-items: center;
  flex-direction: column;
```

```
  justify-content: center;
  width: 100%;
  min-height: 100%;
  padding: 20px;
}
#formContent {
  -webkit-border-radius: 10px 10px 10px 10px;
  border-radius: 10px 10px 10px 10px;
  background: #fff;
  padding: 30px;
  width: 90%;
  max-width: 450px;
  position: relative;
  padding: 0px;
  -webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);
  box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);
  text-align: center;
}
#formFooter {
  background-color: #f6f6f6;
  border-top: 1px solid #dce8f1;
  padding: 25px;
  text-align: center;
  -webkit-border-radius: 0 0 10px 10px;
  border-radius: 0 0 10px 10px;
}
```

```
/* TABS */
h2.inactive {
  color: #cccccc;
}
h2.active {
  color: #0d0d0d;
  border-bottom: 2px solid #5fbae9;
}
/* FORM TYPOGRAPHY*/
input[type=button], input[type=submit], input[type=reset]  {
  background-color: #56baed;
  border: none;
  color: white;
  padding: 15px 80px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  text-transform: uppercase;
  font-size: 13px;
  -webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);
  box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);
  -webkit-border-radius: 5px 5px 5px 5px;
  border-radius: 5px 5px 5px 5px;
  margin: 5px 20px 40px 20px;
  -webkit-transition: all 0.3s ease-in-out;
  -moz-transition: all 0.3s ease-in-out;
```

```
  -ms-transition: all 0.3s ease-in-out;
  -o-transition: all 0.3s ease-in-out;
  transition: all 0.3s ease-in-out;
}
input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover  {
  background-color: #39ace7;
}
input[type=button]:active, input[type=submit]:active, input[type=reset]:active  {
  -moz-transform: scale(0.95);
  -webkit-transform: scale(0.95);
  -o-transform: scale(0.95);
  -ms-transform: scale(0.95);
  transform: scale(0.95);
}
input[type=password] {
  background-color: #f6f6f6;
  border: none;
  color: #0d0d0d;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 5px;
  width: 85%;
  border: 2px solid #f6f6f6;
```

```
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  -ms-transition: all 0.5s ease-in-out;
  -o-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
  -webkit-border-radius: 5px 5px 5px 5px;
  border-radius: 5px 5px 5px 5px;
}
input[type=text] {
  background-color: #f6f6f6;
  border: none;
  color: #0d0d0d;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 5px;
  width: 85%;
  border: 2px solid #f6f6f6;
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  -ms-transition: all 0.5s ease-in-out;
  -o-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
  -webkit-border-radius: 5px 5px 5px 5px;
```

```
  border-radius: 5px 5px 5px 5px;
}
input[type=text]:focus {
  background-color: #fff;
  border-bottom: 2px solid #5fbae9;
}
input[type=text]:placeholder {
  color: #cccccc;
}
input[type=password]:focus {
  background-color: #fff;
  border-bottom: 2px solid #5fbae9;
}
input[type=password]:placeholder {
  color: #cccccc;
}
.icon{
  max-width: 5%;
  height: auto;
}
/* ANIMATIONS */
/* Simple CSS3 Fade-in-down Animation */
.fadeInDown {
  -webkit-animation-name: fadeInDown;
  animation-name: fadeInDown;
  -webkit-animation-duration: 1s;
```

```
  animation-duration: 1s;
  -webkit-animation-fill-mode: both;
  animation-fill-mode: both;
}
@-webkit-keyframes fadeInDown {
  0% {
    opacity: 0;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }
  100% {
    opacity: 1;
    -webkit-transform: none;
    transform: none;
  }
}
@keyframes fadeInDown {
  0% {
    opacity: 0;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }
  100% {
    opacity: 1;
    -webkit-transform: none;
    transform: none;
```

```
  }
}
/* Simple CSS3 Fade-in Animation */
@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
.fadeIn {
  opacity:0;
  -webkit-animation:fadeIn ease-in 1;
  -moz-animation:fadeIn ease-in 1;
  animation:fadeIn ease-in 1;
  -webkit-animation-fill-mode:forwards;
  -moz-animation-fill-mode:forwards;
  animation-fill-mode:forwards;
  -webkit-animation-duration:1s;
  -moz-animation-duration:1s;
  animation-duration:1s;
}
.fadeIn.first {
  -webkit-animation-delay: 0.4s;
  -moz-animation-delay: 0.4s;
  animation-delay: 0.4s;
}
.fadeIn.second {
  -webkit-animation-delay: 0.6s;
  -moz-animation-delay: 0.6s;
```

```
  animation-delay: 0.6s;
}
.fadeIn.third {
  -webkit-animation-delay: 0.8s;
  -moz-animation-delay: 0.8s;
  animation-delay: 0.8s;
}
.fadeIn.fourth {
  -webkit-animation-delay: 1s;
  -moz-animation-delay: 1s;
  animation-delay: 1s;
}
/* Simple CSS3 Fade-in Animation */
.underlineHover:after {
  display: block;
  left: 0;
  bottom: -10px;
  width: 0;
  height: 2px;
  background-color: #56baed;
  content: "";
  transition: width 0.2s;
}
.underlineHover:hover {
  color: #0d0d0d;
}
```

```
.underlineHover:hover:after{
  width: 100%;
}
/* OTHERS */
*:focus {
    outline: none;
}
#icon {
  width:60%;
}
    </style>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <div class="wrapper fadeInDown">
  <div id="formContent">
    <!-- Tabs Titles -->
    <!-- Icon -->
    <div class="fadeIn first">
      <img src="{% static 'media/website/login/icon.png' %}" id="icon" alt="User Icon" />
    </div>
    <!-- Login Form -->
    <form class="login_form" action="" method="POST">
      {% csrf_token %}
      {% if messages %}
```

```
                    <div class="span12">
                        {% for message in messages %}
                            <div class="alert alert-danger">
                                {{ message|safe }}
                            </div>
                        {% endfor %}
                    </div>
                {% endif %}
        <input type="text" id="login" class="fadeIn second" name="login" placeholder="login">
        <input type="password" id="password" class="fadeIn third" name="password"
placeholder="password">
        <input type="submit" class="fadeIn fourth" value="Log In">
    </form>
    <!-- Remind Passowrd -->
    <div id="formFooter">
      <a class="underlineHover" href="{% url 'filehandling:register'  %}">Register Here</a>
    </div>
  </div>
</div>
  </body>
</html>
```

8.1.7.4 Home-auth.html:

```
{% extends 'filehandling/base_auth.html' %}
{% load static %}
    {% block content %}
```

```
<style media="screen">
  div.card{
    width:250px;
    box-shadow:0 4px 8px 0 rgba(0,0,0,0.2),0 6px 20px 0 rgba(0,0,0,0.19);
    text-align:center;
  }
</style>
 <div class="container">
   <div class="row">
     <div class="col-sm">
       <div class="mt-4">
         {% include 'card/database_upload.html' %}
       </div>
     </div>
     <div class="col-sm">
       <div class="col-sm">
         <div class="mt-4">
           {% include 'card/uploaded_pic_card.html' %}
         </div>
       </div>
     </div>
     <div class="col-sm">
       <div class="mt-4">
         {% include 'card/face_detect_card.html'%}
       </div>
     </div>
```

```
    </div>
  </div>
  {% endblock %}
```

8.1.8 Emotion-Detection.py:

```
import numpy as np
import argparse
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
# command line argument
ap = argparse.ArgumentParser()
ap.add_argument("--mode",help="train/display")
mode = ap.parse_args().mode
# plots accuracy and loss curves
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
```

```
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy
    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])
axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accur
acy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['
accuracy'])/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss']
)/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
# Define data generators
train_dir = 'data/train'
val_dir = 'data/test'
num_train = 28709
```

```
num_val = 7178
batch_size = 64
num_epoch = 50
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')
validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')
# Create the model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
# If you want to train the same model or try other models, go for this
if mode == "train":
    model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-
6),metrics=['accuracy'])
    model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
    plot_model_history(model_info)
    model.save_weights('model.h5')
# emotions will be displayed on your face from the webcam feed
elif mode == "display":
    model.load_weights('model.h5')
    # prevents openCL usage and unnecessary logging messages
    cv2.ocl.setUseOpenCL(False)
    # dictionary which assigns each label an emotion (alphabetical order)
    emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6:
"Surprised"}
```

```
    # start the webcam feed
    cap = cv2.VideoCapture(0)
    while True:
        # Find haar cascade to draw bounding box around face
        ret, frame = cap.read()
        if not ret:
            break
        facecasc = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
            roi_gray = gray[y:y + h, x:x + w]
            cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
            prediction = model.predict(cropped_img)
            maxindex = int(np.argmax(prediction))
            cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
        cv2.imshow('Video', cv2.resize(frame,(800,600),interpolation = cv2.INTER_CUBIC))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

8.1.9 Age-Gender Detection:

```
from pathlib import Path
import cv2
import dlib
import numpy as np
import argparse
from contextlib import contextmanager
from wide_resnet import WideResNet
from keras.utils.data_utils import get_file
pretrained_model = "https://github.com/yu4u/age-gender-
estimation/releases/download/v0.5/weights.28-3.73.hdf5"
modhash = 'fbe63257a054c1c5466cfd7bf14646d6'
def get_args():
    parser = argparse.ArgumentParser(description="This script detects faces from web cam input, "
                                                 "and estimates age and gender for the detected faces.",
                                     formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument("--weight_file", type=str, default=None,
                        help="path to weight file (e.g. weights.28-3.73.hdf5)")
    parser.add_argument("--depth", type=int, default=16,
                        help="depth of network")
    parser.add_argument("--width", type=int, default=8,
                        help="width of network")
    parser.add_argument("--margin", type=float, default=0.4,
                        help="margin around detected face for age-gender estimation")
    parser.add_argument("--image_dir", type=str, default=None,
```

```
                help="target image directory; if set, images in image_dir are used instead of
webcam")
    args = parser.parse_args()
    return args
def draw_label(image, point, label, font=cv2.FONT_HERSHEY_SIMPLEX,
            font_scale=0.8, thickness=1):
    size = cv2.getTextSize(label, font, font_scale, thickness)[0]
    x, y = point
    cv2.rectangle(image, (x, y - size[1]), (x + size[0], y), (255, 0, 0), cv2.FILLED)
    cv2.putText(image, label, point, font, font_scale, (255, 255, 255), thickness,
lineType=cv2.LINE_AA)
@contextmanager
def video_capture(*args, **kwargs):
    cap = cv2.VideoCapture(*args, **kwargs)
    try:
        yield cap
    finally:
        cap.release()
def yield_images():
    # capture video
    with video_capture(0) as cap:
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
        while True:
            # get video frame
            ret, img = cap.read()
```

```
            if not ret:
                raise RuntimeError("Failed to capture image")
            yield img
def yield_images_from_dir(image_dir):
    image_dir = Path(image_dir)
    for image_path in image_dir.glob("*.*"):
        img = cv2.imread(str(image_path), 1)
        if img is not None:
            h, w, _ = img.shape
            r = 640 / max(w, h)
            yield cv2.resize(img, (int(w * r), int(h * r)))
def main():
    args = get_args()
    depth = args.depth
    k = args.width
    weight_file = args.weight_file
    margin = args.margin
    image_dir = args.image_dir
    if not weight_file:
        weight_file = get_file("weights.28-3.73.hdf5", pretrained_model,
cache_subdir="pretrained_models",
                               file_hash=modhash, cache_dir=str(Path(__file__).resolve().parent))
    # for face detection
    detector = dlib.get_frontal_face_detector()
    # load model and weights
    img_size = 64
```

```
model = WideResNet(img_size, depth=depth, k=k)()
model.load_weights(weight_file)
image_generator = yield_images_from_dir(image_dir) if image_dir else yield_images()
for img in image_generator:
    input_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_h, img_w, _ = np.shape(input_img)
    # detect faces using dlib detector
    detected = detector(input_img, 1)
    faces = np.empty((len(detected), img_size, img_size, 3))
    if len(detected) > 0:
        for i, d in enumerate(detected):
            x1, y1, x2, y2, w, h = d.left(), d.top(), d.right() + 1, d.bottom() + 1, d.width(), d.height()
            xw1 = max(int(x1 - margin * w), 0)
            yw1 = max(int(y1 - margin * h), 0)
            xw2 = min(int(x2 + margin * w), img_w - 1)
            yw2 = min(int(y2 + margin * h), img_h - 1)
            cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
            # cv2.rectangle(img, (xw1, yw1), (xw2, yw2), (255, 0, 0), 2)
            faces[i, :, :, :] = cv2.resize(img[yw1:yw2 + 1, xw1:xw2 + 1, :], (img_size, img_size))
        # predict ages and genders of the detected faces
        results = model.predict(faces)
        predicted_genders = results[0]
        ages = np.arange(0, 101).reshape(101, 1)
        predicted_ages = results[1].dot(ages).flatten()
        # draw results
        for i, d in enumerate(detected):
```

```
            label = "{}, {}".format(int(predicted_ages[i]),
                                    "M" if predicted_genders[i][0] < 0.5 else "F")
            draw_label(img, (d.left(), d.top()), label)
        cv2.imshow("result", img)
        key = cv2.waitKey(-1) if image_dir else cv2.waitKey(30)
        if key == 27:  # ESC
            break
if __name__ == '__main__':
    main()
```

9. CONCLUSION

The main aspect of this report was to show the implementation of a product which can be used by the blind disabled in order to facilitate them with more knowledge of surrounding which they were earlier not able to perceive. With the implementation of cloud-based technology, newer machine learning algorithms as well as new facilities can be added to the system with ease. With better dataset, ML model's accuracy can be improved. User-Interface used in this report is based on voice, which helps to overcome the main problem when considering the system designed for blind disabled. Further we can speculate that with the help of new emerging technology in the field of computation power, these machine learning models can be made to run on edge devices which means even in the case of internet not working, our system will still be able to help the person with all the ability. We can even add the facility to call somebody by the use of telecom SIM. This project is made open-source so that people can use this in their own system. This will help the project to grow as the community will be able to add new features into it.

10. REFERENCES

- https://github.com/ageitgey/face_recognition/blob/master/examples/web_service_example.py
- https://docs.djangoproject.com/en/3.0/
- https://towardsdatascience.com/face-detection-recognition-and-emotion-detection-in-8-lines-of-code-b2ce32d4d5de
- https://towardsdatascience.com/predict-age-and-gender-using-convolutional-neural-network-and-opencv-fd90390e3ce6
- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://www.ffri.hr/~ibrdar/komunikacija/seminari/Importance%20of%20emotions%20in%20interaction.pdf