AEYE

A Project Report

Submitted in the partial fulfillment for the award of the degree of

B.Tech VI Semester Computer Science

Submitted by:

Vidhu Chaudhary-1812916

Under the supervision of
(Dr. Sudha Morwal, Associate Professor)

Department of computer Science

Banasthali Vidyapith

Banasthali - 304022

Session: 2020-21

ABSTRACT

Computer Vision deals with the science of computers and software systems that recognize as well as understand images and scenes. Computer Vision covers various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, pedestrian counting, vehicle detection, web images, security systems and self-driving cars.

The project aims to build a real time object detection system to detect objects in the surrounding of a user. This project is inspired by the hope to aid the visually impaired people and help them in conducting their everyday tasks with ease. In this project, we are using highly accurate object detection-algorithms such as CNN with the help of YOLOv3 model. These methods and algorithms require understanding of mathematical and deep learning frameworks by using dependencies such as OpenCV. We can detect each and every object in image by the area in a highlighted rectangular box and identify each and every object and assign its respective tag. This also includes the accuracy of each method for identifying objects. It facilitates the feature of text-to-speech conversion in order to tell the class of detected objects to the user with the help of audio. The technology of object detection and sound processing would combine together to provide a comfortable user experience.

ACKNOWLEDGEMENT

We would like to express my gratitude and appreciation to all those who gave us the possibility to complete this report. A special thanks to our final year project coordinators, Dr. Manisha Agarwal and Dr. Neelam Sharma, whose help, stimulating suggestions and encouragement, helped us to coordinate our project.

We would also like to acknowledge with much appreciation the crucial role of the supervisor, Dr. Sudha Morwal, who supported us and guided us through our project.

A special thanks goes to the team members, who helped in completing the project and implementing all the features successfully.

Last but not least, many thanks to the ones who have given their full effort in guiding the team in achieving the goal as well as their encouragement to maintain our progress in track. We would like to appreciate the guidance given by other teachers, especially in our project presentation that has improved our presentation skills by their comment and tips.

TABLE OF CONTENTS

1. Objective

The objective of this report is to provide a detailed description of a real time object detection system. This document will present the system requirements and system design. Visually impaired people confront many problems in moving from one place to another. Vision is human's power to notify him of the obstacles in his way. A solution which is easily available is needed to solve the problems of visually impaired people. This project tries to transform the visual world into the audio world with the potential to inform visually impaired people about the objects in their local environment. Objects detected from the scene are represented by their names and converted to audio. The model of a web application has been proposed which is designed to detect multiple objects in real-time with the simplest interface that can be easily used by visually impaired people i.e., the target users of our application.

2. Requirement Analysis

2.1 Requirement Specification

The below diagram is the architectural / conceptual diagram of our system. It is a four layered architecture. Through this conceptual design we are trying to show the interaction between the different layers of our system. All the requests of the users are taken by the application and are given to the API for processing. The API uses a dataset which contains thousands of labelled images and compares the current image with the images in the dataset. After classification and identification, the API sends the label of the current image to the application.

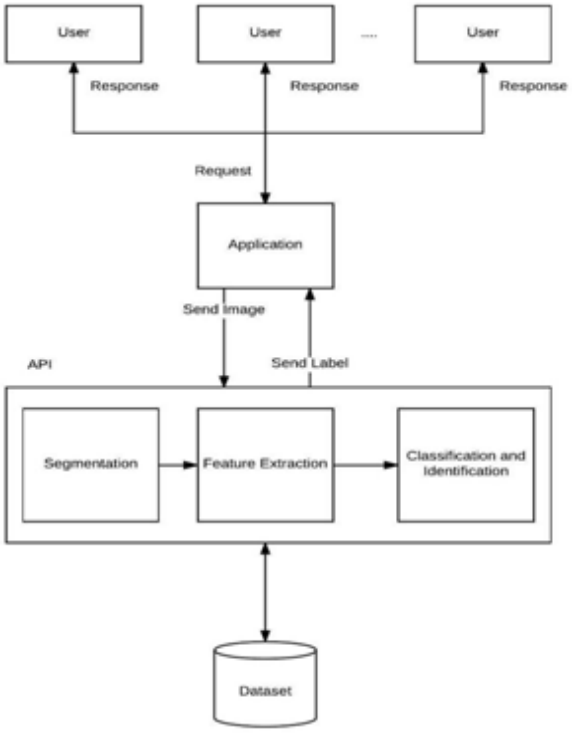


Fig.1 Architecture of the system

The output of the system is in audio form, which is easily understandable for a visually impaired user. The application being developed can detect the objects in the user's surroundings. It identifies objects familiar to the user in his daily life, and then tells the user of these detected objects to aid him in his daily life routines. For Image Processing purposes, we use the OpenCV Library. We then compare the contents in a photo to our pre-stored image dataset i.e., MS-COCO. Apart from these, Google Text To Speech API is used which allows us to give an audio output. The information about external modules used for creating the application are as follows:

A. OpenCV

OpenCV (Open Source Computer Vision) is a library of programming capacity which for the most part went for constant PC vision. Initially created by Intel, it is currently kept up by Itseez and supported by Willow Garage. The library is cross-platform and free for use under the open-source BSD license. OpenCV bolsters the Deep Learning Structures Torch/PyTorch, Caffe & TensorFlow.

B. Microsoft COCO Dataset

The COCO dataset stands for Common Objects in Context, and is designed to represent a vast array of objects that we regularly encounter in everyday life. The COCO dataset is labeled, providing data to train supervised computer vision models that are able to identify the common objects in the dataset. Of course, these models are still far from perfect, so the COCO dataset provides a benchmark for evaluating the periodic improvement of these models through computer vision research. COCO is a large-scale object detection, segmentation, and captioning dataset. It has several features such as object segmentation, recognition in context, superpixel stuff segmentation, 330K images (>200K labeled) and 1.5 million object instances.

C. Google Text To Speech API

gTTS (Google Text-to-Speech) is a Python library and CLI tool to interface with Google Translate's text-to-speech API. It writes spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation. It features flexible pre-processing and tokenizing.

2.2 Hardware and Software Requirements

2.2.1 Hardware Requirements

The system will work on laptop or a PC with stable internet connection. The device on which the system is running must have a minimum of 4GB RAM else it might face problems. The running device must have a clear pixel resolution and a speaker connected to the device. It is expected that the CPU speed of the device and the internet connection speed shall be enough to respond to the user's request quickly.

2.2.2 Software Requirements

Windows operating systems were used during the development process of this application. The text editor used by our team was Visual Studio Code version 1.53.2 and higher. The system runs best on the web browser.

2.3 Product Functions

In our project, we build a real-time object detection system with the goal of informing the user about surrounding objects. The system to be developed will be trained about object information. Feature extraction is also a part of the process. The core specialty of the system is to detect multiple objects in an image. That is, it is a system where N object detectors are trained for N different objects. When an image is sent to the system, all object detectors do their work. If an object is found by a detector, it will mark its boundary and label the object name. After the process completes for all N detectors, the image is displayed with all the tags and the user is prompted one by one with audios containing the name of the objects detected.

First, the user starts the application and captures the image of the surrounding in front of him or of the object in front of him which he wants to identify. The application digitizes and stores the captured image in the memory. It is then used to detect objects in other images. This image will be processed by using libraries like OpenCV and Google Cloud Vision API. The OpenCV library contains many Image Processing algorithms and Google Cloud Vision API has the power to compare the input image with millions of other images using Microsoft's COCO Dataset. The API receives image data, performs image segmentation, feature extraction, classification & identification functions and uses the COCO dataset to get the image label. The gTTS API then converts the label into audio and sends it via the speaker.

2.4 Use- Case Diagrams

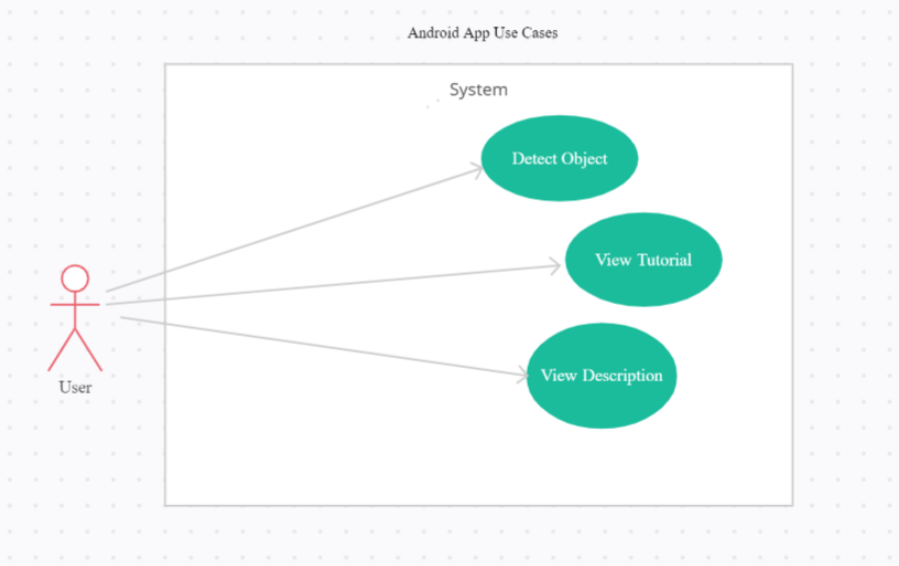


Fig.2 Use-Case Diagram

3. System Design

3.1 High Level Design

3.1.1 Structure and Relationship

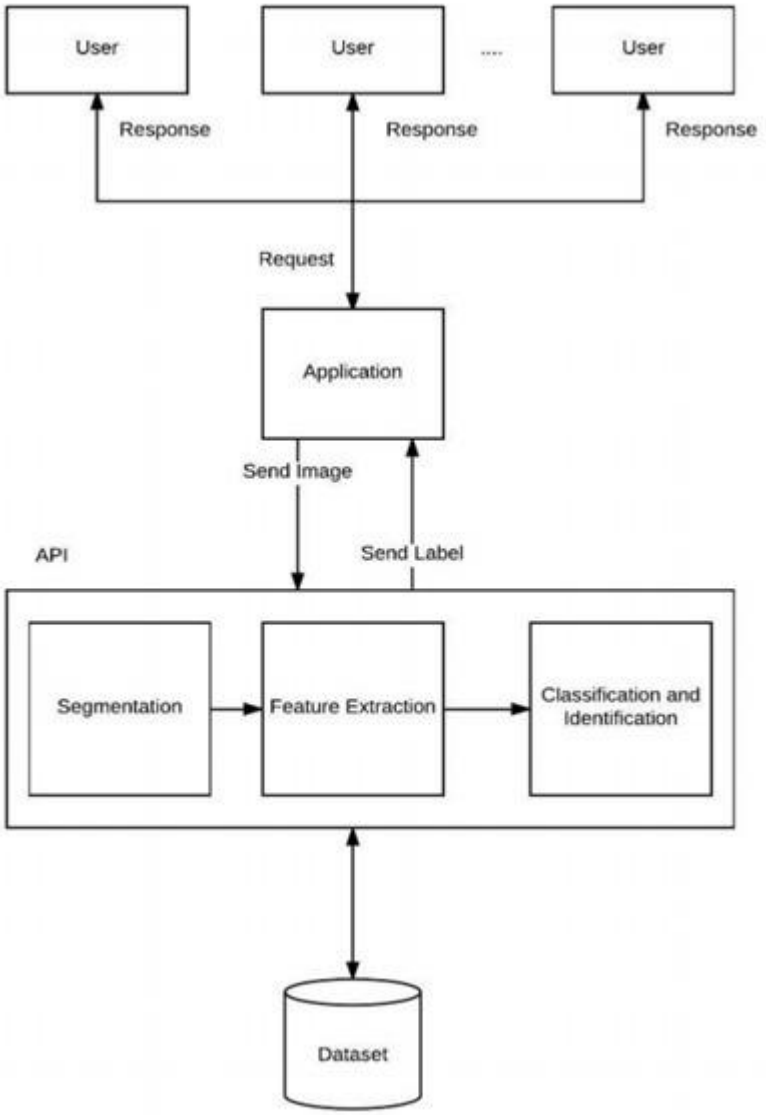


Fig.3 Architectural diagram

The above diagram is the architectural / conceptual diagram of our system. It is a four layered architecture. Through this conceptual design we are trying to show the interaction between the different layers of our system, mainly dividing into four main components as mentioned above. The first layer depicts the users, which is the external layer of our entire architecture. The figure shows an

interaction between the users and the middle layer which is the application. All the requests of the users are taken by the application and are given to the API for processing. The API uses a dataset which contains thousands of labelled images and compares the current image with the images in the dataset.

After classification and identification, the API sends the label of the current image to the application. The web camera of the laptop/ PC will be used to capture an image of the surrounding which will be stored in memory. This image will be processed by using libraries like OpenCV. It uses the COCO dataset to compare the input image with millions of other images. The process gets completed and the objects are identified in the image. The user gets informed about the identified objects present in his surroundings via an audio output.

3.2 Module Diagram

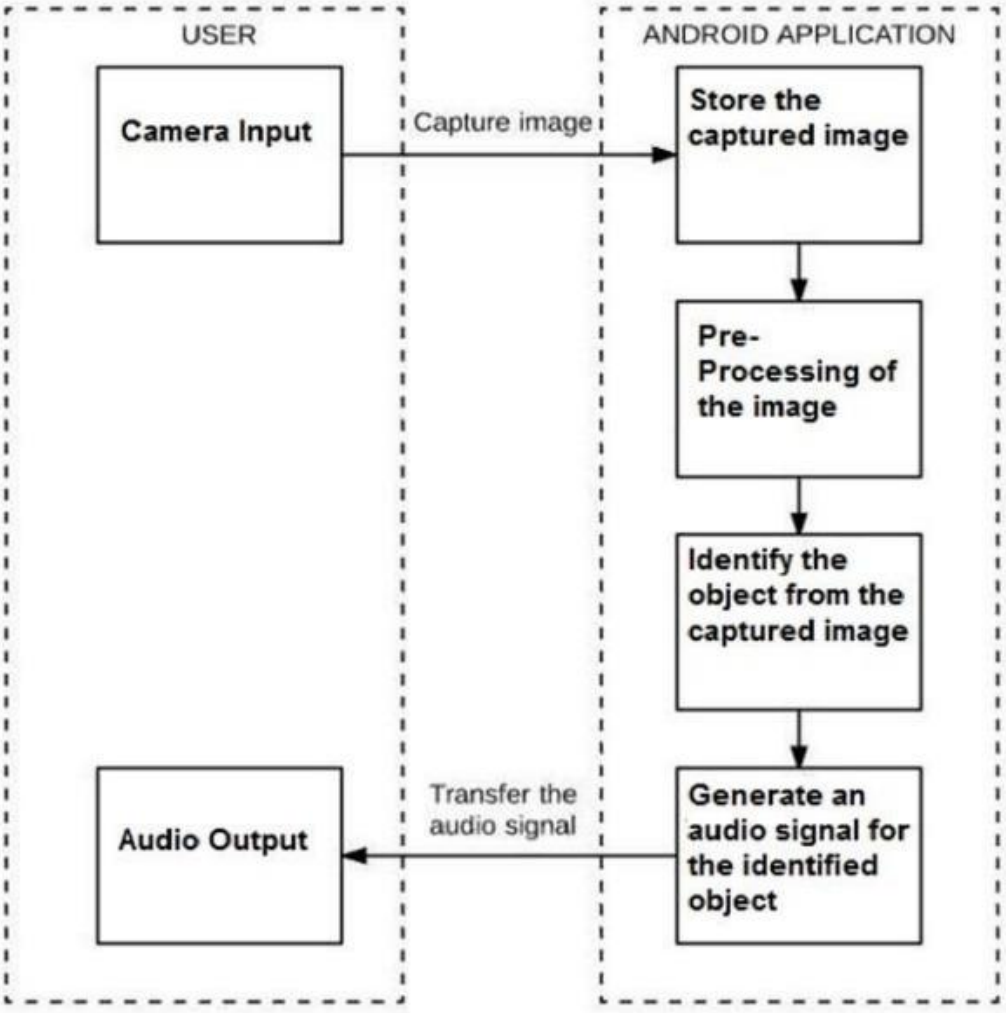


Fig.4 Module diagram

3.3 Sequence Diagram

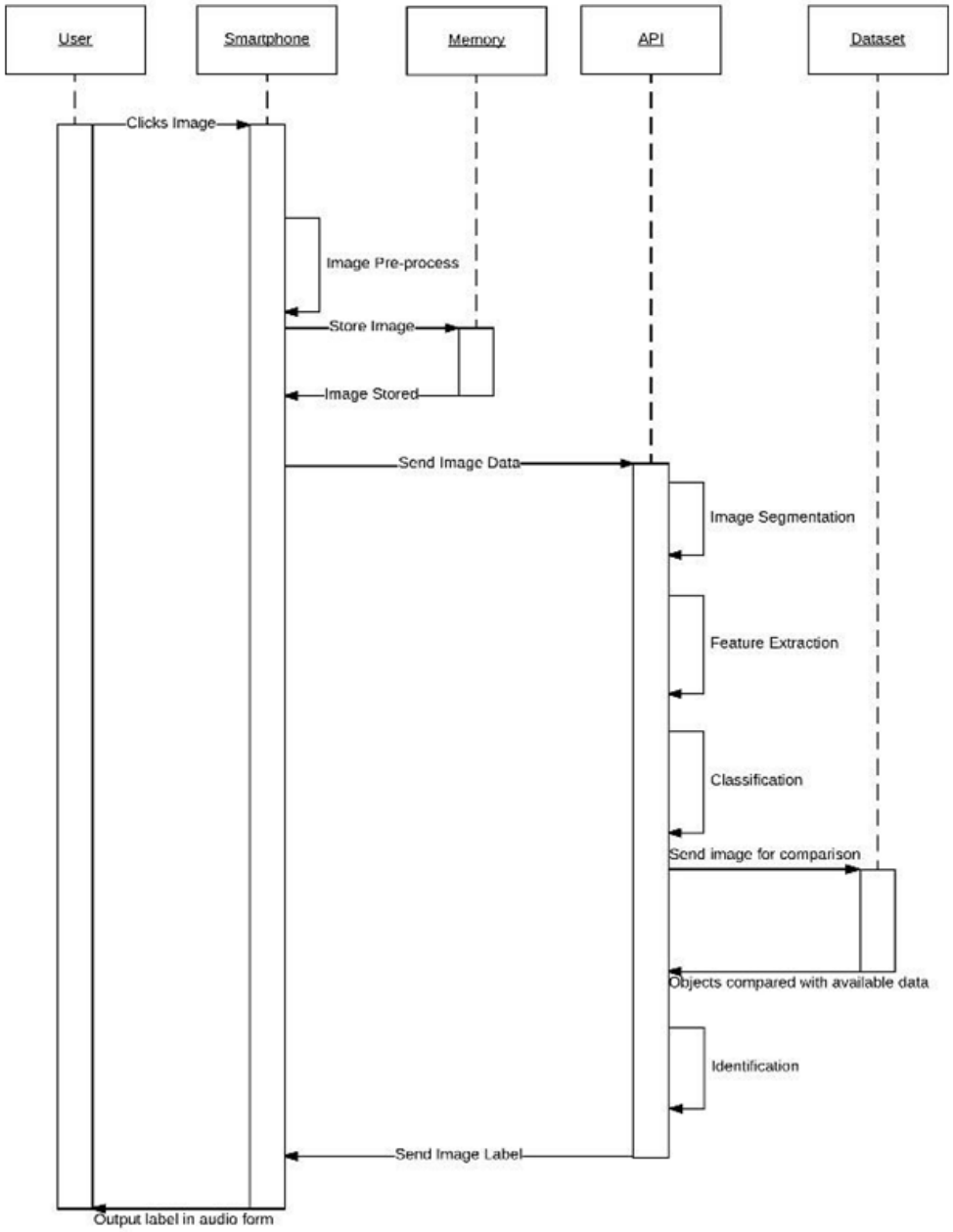


Fig.5 Sequence diagram

The above sequence diagram of our framework clarifies the stream of the framework, that is, what action takes place first and what action will follow the previous action. First, the user starts the application and captures the image of the surrounding in front of him or of the object in front of him which he wants to identify. The application digitizes and stores the captured image in the memory. It is then used to detect objects in other images. This image will be

processed by using libraries like OpenCV and Google Cloud Vision API.

The OpenCV library contains many Image Processing algorithms and Google Cloud Vision API has the power to compare the input image with millions of other images using Microsoft's COCO Dataset. The API receives image data, performs image segmentation, feature extraction, classification & identification functions and uses the COCO dataset to get the image label. The application then converts the label into audio and sends it via the speaker.

4. Coding (Main Module)

File Name: camera.py

Code:

```
import numpy as np
import argparse
import subprocess
import time
import os
import cv2 as cv
from gtts import gTTS
from playsound import playsound
face_cascade=cv.CascadeClassifier("haarcascade_frontalface_alt2.xml")
ds_factor=0.6
def audio(text):
    language = 'en'
    myobj = gTTS(text=text, lang=language, slow=False)
    myobj.save("predict.mp3")
    playsound("predict.mp3")
    os.remove("predict.mp3")
class VideoCamera(object):
    def __init__(self):
        self.video = cv.VideoCapture(0)

    def __del__(self):
        self.video.release()

    def get_frame(self):
        whT = 320
        confThreshold = 0.5
        nmsThreshold = 0.2
```

```
        classesFile = "yolov3-coco/coco-labels"
        classNames = []
        with open(classesFile, 'rt') as f:
            classNames = f.read().rstrip('\n').split('\n')

        modelConfiguration = "yolov3-coco/yolov3.cfg"
        modelWeights = "yolov3-coco/yolov3.weights"
        net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeigh
ts)
        net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
        net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)


        def findObjects(outputs, img):
            hT, wT, cT = img.shape
            bbox = []
            classIds = []
            confs = []
            for output in outputs:
                for det in output:
                    scores = det[5:]
                    classId = np.argmax(scores)
                    confidence = scores[classId]
                    if confidence > confThreshold:
                        w, h = int(det[2] * wT),int(det[3]*hT)
                        x, y = int((det[0] * wT) - w / 2), in
((det[1] * hT) - h / 2)
                        bbox.append([x, y, w, h])
                        classIds.append(classId)
                        confs.append(float(confidence))

            indices = cv.dnn.NMSBoxes(bbox, confs, confThreshold, nmsThre
shold)

            for i in indices:
                i = i[0]
                box = bbox[i]
                x, y, w, h = box[0], box[1], box[2], box[3]
                # print(x,y,w,h)
                cv.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), 2)
                cv.putText(img, f'{classNames[classIds[i]].upper()} {int(confs[i]
 * 100)}%',
                           (x, y - 10), cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0,
255), 2)
                audio(classNames[classIds[i]])

        while True:
            success, img = self.video.read()
```

```
        blob = cv.dnn.blobFromImage(img, 1 / 255, (whT, whT), [0, 0, 0],
1, crop=False)
        net.setInput(blob)
        layersNames = net.getLayerNames()
        outputNames = [(layersNames[i[0] - 1]) for i in net.getUnconnecte
dOutLayers()]
        outputs = net.forward(outputNames)
        findObjects(outputs, img)
        ret, jpeg = cv.imencode('.jpg', img)
        return jpeg.tobytes()
```

File name: index.html

Code:

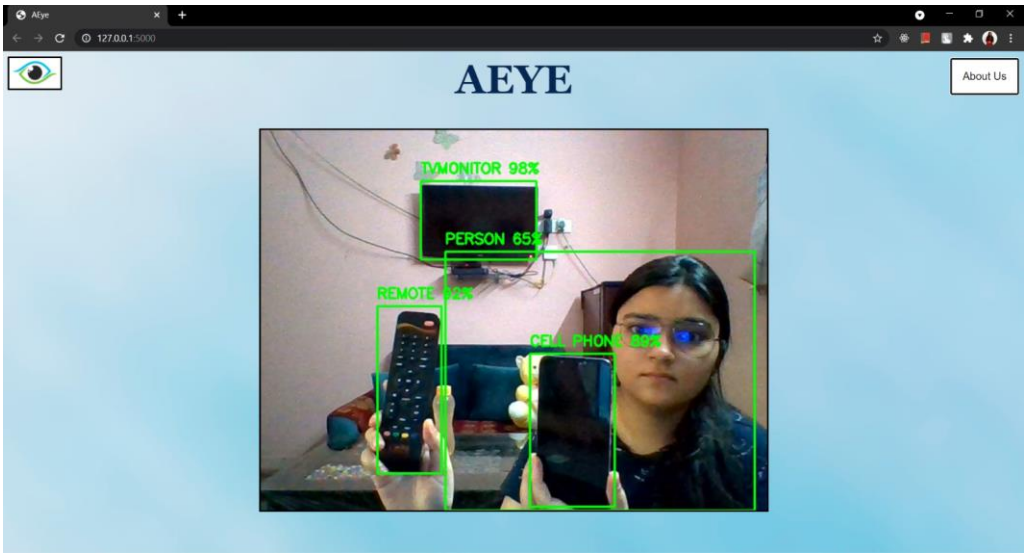
```
<html>
 <head>
  <title>AEye</title>
  <style>
   #icon{
    display: flex;
    position: fixed;
    align-content: left;
    width:5%
   }
 img {
  border: black solid;
  border-radius: 2px;
  border-style: solid;
  display: block;
  margin-top: 0px;
  margin-left: auto;
  margin-right: auto;
 width: 50%
 }
 /*body{
  margin: 0%;
  display: flex;
  flex-direction: column;
  align-content: center;
 }*/
 .header{
  display: inline;
  position: relative;
  margin: 0;
 }
 h1{
  margin-top:0;
```

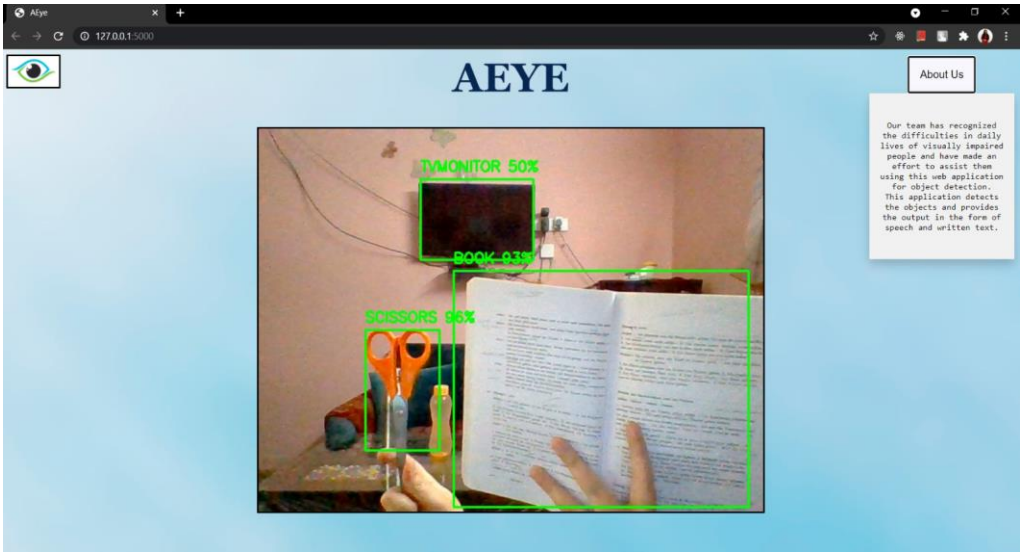
```
    text-align: center;
    font-style:inherit;
    font-size: 60px;
    display:flex;
    position: relative;
    flex-direction:column;
  }
  .abt{
    margin: 0;
    align-items: right;
    display:flex;
    position: fixed;
    flex-direction: column;
  }
  /*.a{
    margin-right: 10px;
    padding-top: 20px;
    text-align: right;
    align-items: right;
    display:flex;
    position: relative;
    flex-direction: column;
  }*/
  .dropbtn {
  background-color:white;
  color: black;
  padding: 16px;
  font-size: 16px;
  border: black solid;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
  padding: 10px;
    display:block;

    color: black;
    text-align:center;
    position: absolute;
    top: 0px;
    right: 0px;
}

/* Dropdown Content (Hidden by Default) */
.dropdown-content {
  display: none;
  position: relative;
  background-color: #f1f1f1;
```

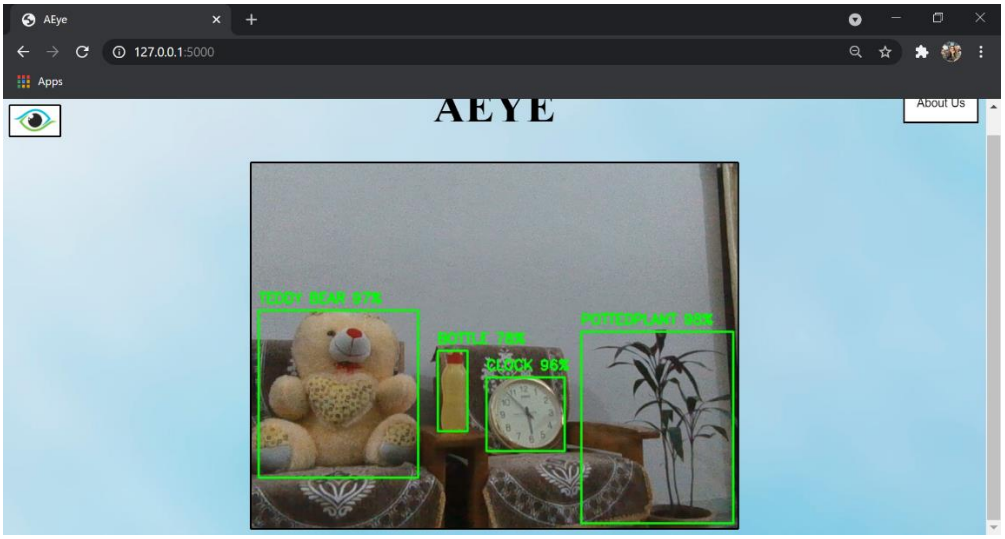
```
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

/* Links inside the dropdown */
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {background-color: #ddd;}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {display: flex;}

/* Change the background color of the dropdown button when the dropdo
wn content is shown */
.dropdown:hover .dropbtn {background-color:white;}
    </style>
  </head>
  <body style="background-
image: url('https://cdn.hipwallpaper.com/i/18/5/1Ox6pE.jpg');">
     <img id="icon" src="https://lirp.cdn-
website.com/379be215/dms3rep/multi/opt/logo+new+2-480w.png">
    <div class="header">
      <h1>AEYE</h1>
    </div>
    <img id="bg" class="center" src="{{ url_for('video_feed') }}">
    <div class="dropdown">
      <button class="dropbtn">About Us</button>
      <div class="dropdown-content">
        <pre>
          <a>
Our team has recognized
the difficulties in daily
lives of visually impaired
people and have made an
effort to assist them
using this web application
for object detection.
This application detects
the objects and provides
the output in the form of
audio and written text.
```

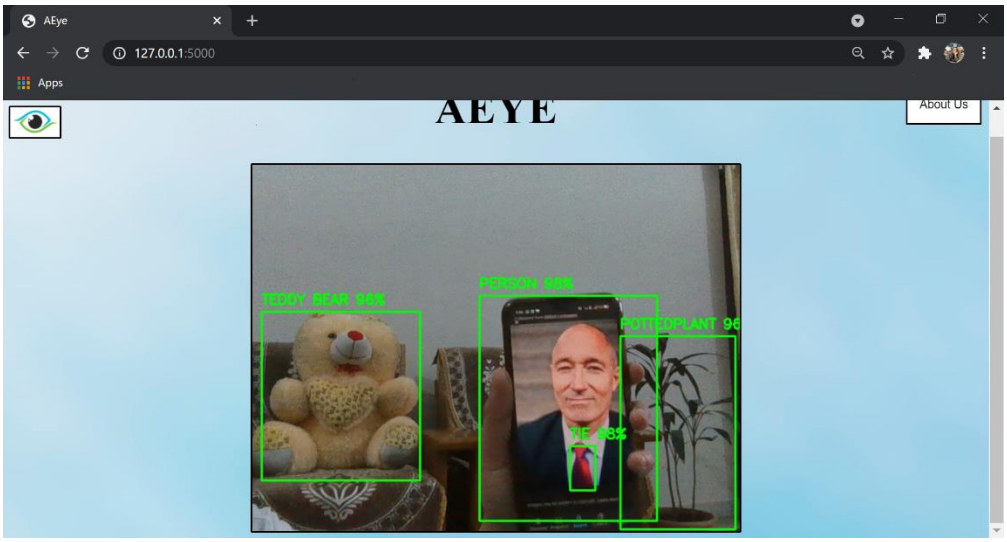
```
</a>
      </pre>
    </div>
  </div>
 </body>
</html>
```

5. Testing







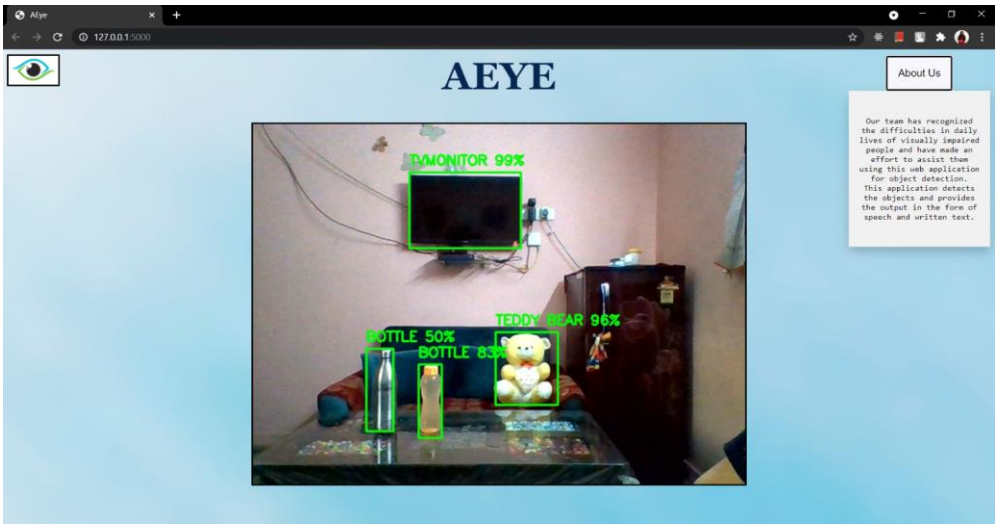


6. User Interfaces

There will be one graphical user interface for the use of people who are not visually impaired there will be one basic page belonging to this GUI. This GUI will be responsible for providing communication between the user and the application.

- The contents of the pages will be displayed in english language. The users of the system are assumed to understand the supported language by the application i.e. English.
- AEye will depend on the internet connection. Without an internet connection, the application will not be started

- Keeping in mind the convenience of the users and to minimize eye strain for the visually impaired people we have kept the background of the user interface as blue.
- The logo of our application is placed at the top left corner of the main page and a hypertext ‘About Us’ is displayed at the top right corner.



7. Appendices

There is no available appendix for this report.

8. References

[1] Pressman Roger S.,Software Engineering “A practitioner’s Approach” Fifth Edition , Publication

[2] R. Rajwani , D. Purswani , P. Kalinani , D. Ramchandani, I. Dokare, “Proposed System on Object Detection for Visually Impaired People”, *International Journal of Information Technology*, vol. 4, no. 1, Mar ,2018. [Online serial]. Available: http://www.ijitjournal.org/volume-4/issue-2/IJIT-V4I2P1.pdf

[3] https://github.com/Garima13a/YOLO-Object-Detection

[4] Joseph Redmon and Anelia Angelova, Real-Time Grasp Detection Using Convolutional Neural Networks (ICRA), 2015.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640,2015.