HAND GESTURE TO TEXT CONVERTER – (DIVYAANUVADAK)

A Project Report

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

B.Tech. in Information Technology (VI Semester)

Submitted by

Ayushi Rajput – 2016730
Deepali – 2016736
Shivalik – 2016821
Sneha Poddar – 2016837

Under the supervision of

Dr. Nisheeth Joshi
Associate Professor
Department of Computer Science

Name of the Mentor	**Name of Coordinator(s)**
Dr. Nisheeth Joshi Associate Professor Department of Computer Science	Prof. Saurabh Mukherjee Dr. Neelam Sharma Dr. Pooja Asopa Dr. Deepak Kumar Dr. Sneha Asopa Dr. Ashok Kumar Dr. Rahul Vijay

Department of Computer Science
Banasthali Vidyapith

Banasthali - 304022

Session: 2022-23

Certificate

Certified that **Ayushi Rajput, Deepali, Shivalik, Sneha Poddar** has carried out the project work titled **"Gesture to Text Converter - DivyaAnuvadak"** from **29-12-2022** to **13-04-2023** for the award of the **B.Tech (IT) VI Semester** from **Department of Computer Science** under my supervision. The thesis embodies result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Nisheeth Joshi

Designation: Associate Professor

Place: Department of Computer Science

Date: 13-04-2023

ABSTRACT

“APAJI’s vision to educate girls,

why not to extend it to specially-abled!”

Some people affected by speech and hearing impairment cannot express their feelings and emotions, hence are not able to communicate with the people. They rely on sign language which most of the people find hard to understand. Because of this, even if they are capable, they lack opportunities.

Thus, we came up with the idea of ‘DivyaAnuvadak’ which takes hand gesture as input and translate it to equivalent text which could be understood by common people. This would further help both our community and our university build a startup to educate specially-abled girls. This would eventually boost up their self-esteem and can make them independent.

We are trying to widen Apaji’s vision to educate every girl (specially-abled) that would ultimately help these Shantas to shine too.

ACKNOWLEDGEMENT

We express our sincere indebtedness towards our guide and mentor **Dr. Nisheeth Joshi**, Associate Professor, Department of Computer Science for his valuable guidance, suggestions and supervision throughout the work. Without his kind patronage and guidance the project would not have taken shape. We would also like to express our gratitude and sincere regards for his kind approval of the project, time to time counseling and advices.

We would also like to thank **Prof. CK Jha,** Head of Department, Department of Computer Science for providing us the opportunity to build this project. We would also like to acknowledge with much appreciation the crucial role of the project coordinators, teachers and specially lab technicians for guiding and helping throughout the project.

Lastly, kudos to the team work, relentless efforts and dedication for successful completion of the project.

Chapter 1 SRS

TABLE OF CONTENTS

1. Introduction

2. General Description

3. Specific Requirements

1. INTRODUCTION

Problem Definition

The main problem that our project aims to solve is the communication problem between speech-impaired people and the others. As those people cannot express themselves with the words, they have many difficulties during their daily life. Since almost all of the normal people do not know sign language and cannot understand what speechless people mean by their special language, tasks such as shopping, settling affairs at a government office are so difficult that speech-impaired people cannot handle by their own.

This problem is very broad and many solutions and implementation can be raised. A solution could be teaching sign language to everyone, yet it is very obvious to be an inefficient and even non-applicable one. Since speechless people can understand other people by lip-reading (or even hearing since being speechless does not mean being deaf also) the main problem is that normal people do not understand them. Thus, a more suitable solution should be in the manner that makes speechless people's language understandable by the other people. For the language of speechless people to be understood by others, we need the gestures performed by speech-impaired people to be recognized and turned into a form that the others can understand.

Purpose

The aim of this document is to specify the features, requirements of the final product and the interface of ISL recognition using Tensorflow Object Detection API.

It will explain the scenario of the desired project and necessary steps in order to succeed in the task. To do this throughout the document, overall description of the project, the definition of the problem that this project presents a solution and definitions and abbreviations that are relevant to the project will be provided.

The preparation of this SRS will help consider all of the requirements before design begins, and reduce later redesign, recoding, and retesting. If there will be any change in the functional requirements or design constraint's part, these changes will be stated by giving reference to this SRS in the following documents.

Scope

The project which is going to be presented in this document is called Divya-Anuvadak. This application is planned to be used by speechless people in order to ease their life, and also government offices that should serve all of its citizens equally, private companies that want to reach and serve speechless people as well, co-operations and foundations which aims to help speech- disordered people.

Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specification

ISL: Indian Sign Language

XML: Extensible Markup Language (XML) which is a programming language with markersVS

Code: Visual Studio Code

IDE: Integrated Development Environment

RAM: Random Access Memory

HDD: Hard Disk Drive

GPU: Graphics Processing Unit

Overview

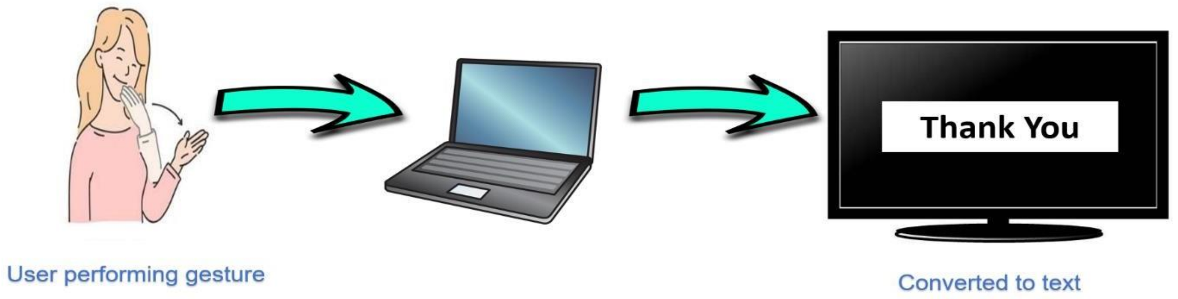
In the following section of this document, we will focus on overall description of the system. This part explains the product perspective, product functions, characteristics, constraints and technologies used in the desired application. Third part of the document explains the specific requirements of the system. Specific requirements are divided into two parts namely functional requirements and non-functional requirements of the system.

2. THE OVERALL DESCRIPTION

Product Perspective

Divya-Anuvadak will serve to speech-impaired people so that they won’t face difficulty in communication by providing their gestures converting to the text for people who does not know ISL. Our project is different from the existing systems because it focuses on the word recognition through gestures while the existing systems focuses on letter recognitions through hand signs which is very slow and makes having an actual conversation quite difficult.

This system will consist of one web portal. This web portal will be used for converting gestures to its corresponding text. The predefined gestures will be stored in the database and will be compared with the original gesture performed by the user.



Product Function

- Capturing the gestures made by the sign languages user through an image sensor.
- Tracking the gestures through OpenCV by identifying feature points.
- Pre-Processing the captured data.
- Feeding the data to the model.
- Predicting the word based on processed model.
- Selecting the word of highest possibility.
- Converting the word into text and displaying it on the screen.

Hardware Interface

- **SERVER SIDE**:

RAM	**:**	8GB (minimum)
HDD	**:**	1 TB or more
Processor	**:**	Intel® i3 or faster (2–4 GHz)
GPU	**:**	2 GHz or more

- **CLIENT SIDE**:

RAM	**:**	4 GB (minimum)
HDD	**:**	512 GB or more
Processor	**:**	Intel® Pentium or faster (2-4 GHz)
Camera	**:**	720p

Software Interface

- **SERVER SIDE**:

Operating System	**:**	Windows 10 or 11
Web Server	**:**	Apache Tomcat (v10)
IDE	**:**	Jupyter Notebook (v6.4.8), VS Code (v1.74.3)
Programming Language	**:**	Python

- **CLIENT SIDE**:

Operating System	**:**	Windows 7 or higher versions
Web Browser	**:**	Any Browser
Library	**:**	Streamlit

Communication Interface

HTTP and TCP/IP Protocol is used as a Communication Interface.

User Characteristics

- User is required to have basic knowledge of computing and internet surfing.
- User should have good knowledge of ISL and English.

General Constraints

- Hardware limitation on mobile devices as mobile devices have very limited hardware power.
- Full-fledged translation is not possible because the English language has more than 1,000,000 words.
- Only one-way communication is possible through this project.
- Fast-paced communications are not possible as the data captured requires some time to process and predict the words and the hardware is not possible to process that fast.

Technologies Used

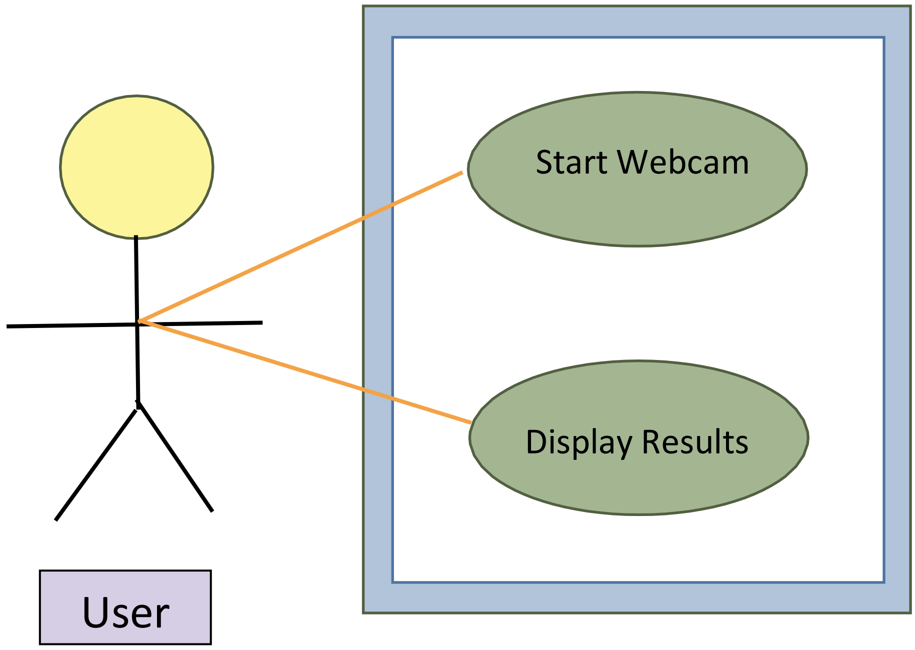
- Python
- Jupyter Notebook
- Cv2(OpenCV)
- Keras
- VS Code
- TensorFlow (as Keras uses TensorFlow in backend and for image pre-processing)
- Streamlit

3. SPECIFIC REQUIREMENTS

Functional Requirements

We describe the functional requirements by giving various Use Cases:

- **USE CASE 1:** Webcam started by user.
- **USE CASE 2:**Text is displayed on the screen.



Use Case Diagram

Non-Functional Requirements

Availability

This web application will be available free on the internet. The users do not require to login so it is user friendly.

Security

This web application does not ask for any personal information from the user. Hence it is a secure application.

Reliability

The application will be 99% reliable and will produce correct speech for the corresponding gestures.

Portability

This project is a web application so it will be available online and can be accessed from any device having internetconnectivity, e.g.: laptops, computers, mobile devices etc.

Chapter 2 SDS

TABLE OF CONTENTS

1. Introduction

Purpose

This Software Design Document provides the design details of **“DivyaAnuvadak”**.

This would help speech and hearing impaired people to communicate with theworld.

Scope

“DivyaAnuvadak.com” is a web-based service that would allow speech andhearing impaired people to convert their hand gestures to equivalent text.

This would not merely help them in communication but also in their educationwhich would further boost their self-esteem.

This would ultimately aid Banasthali to take a step ahead and inaugurate a newdepartment for such specially-abled people.

Terms,Definitions, Acronyms, and abbreviations

Admin – Administrator DFD –

Data Flow Diagram

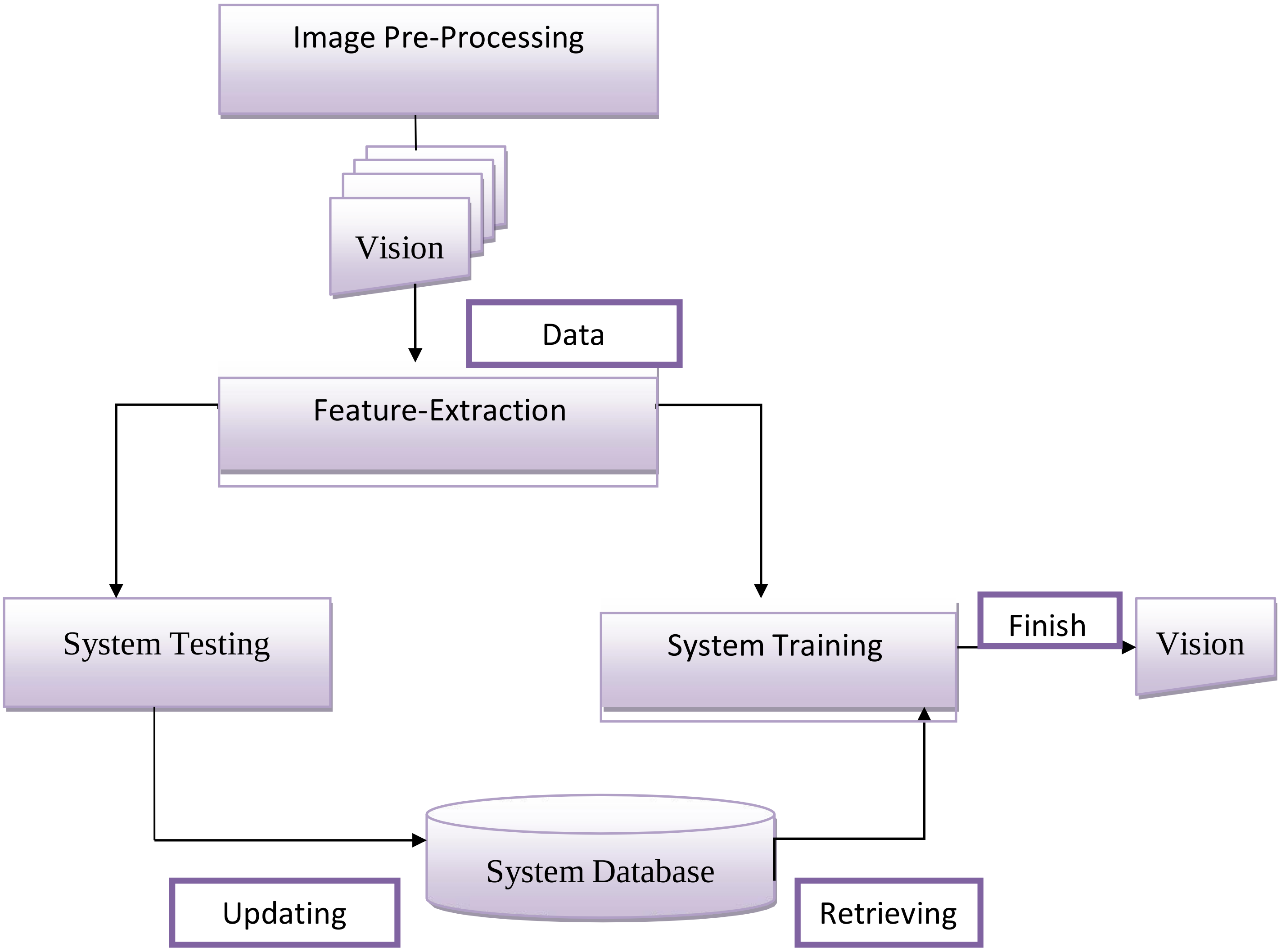
Overview

The rest of the SDS consists of the following parts –

- System Architectural Design
  - High Level Design
  - Structure Chart
  - DFD
  - Usecase Diagram
  - Sequence Diagram
  - Activity Diagram
- Data Design includes Dataset Description
- Detailed description of components
- User Interface Designs
- Testing Strategies

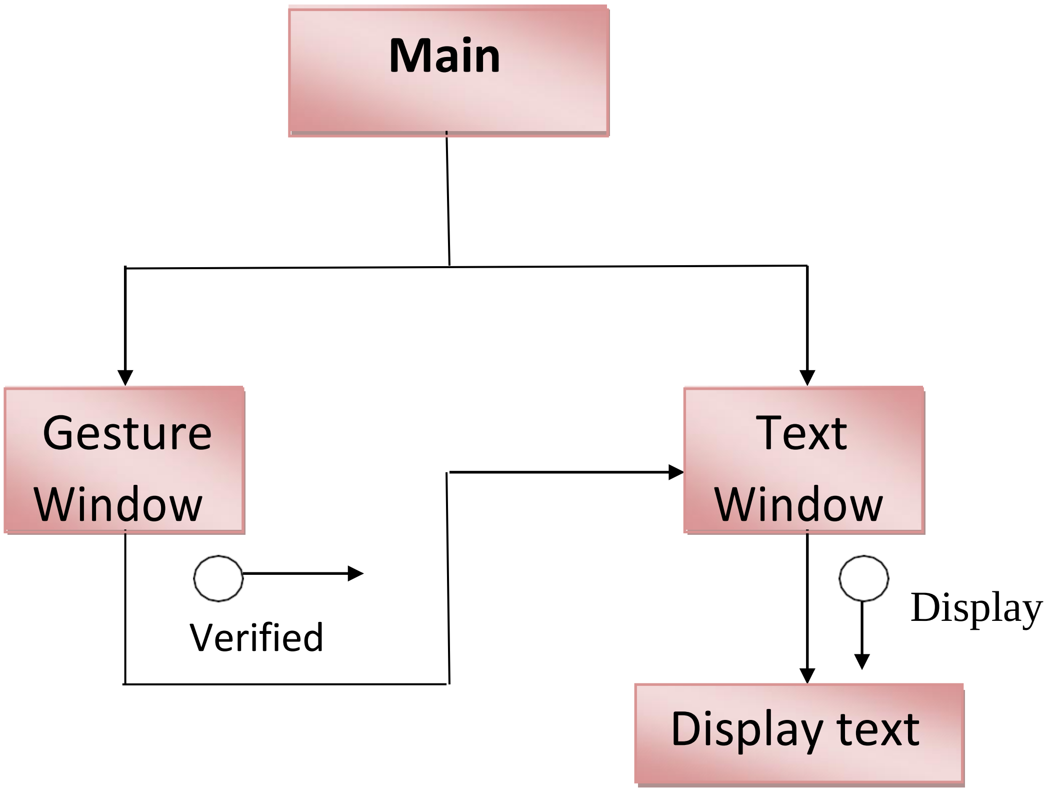
2. System Architectural Design

High – Level Design

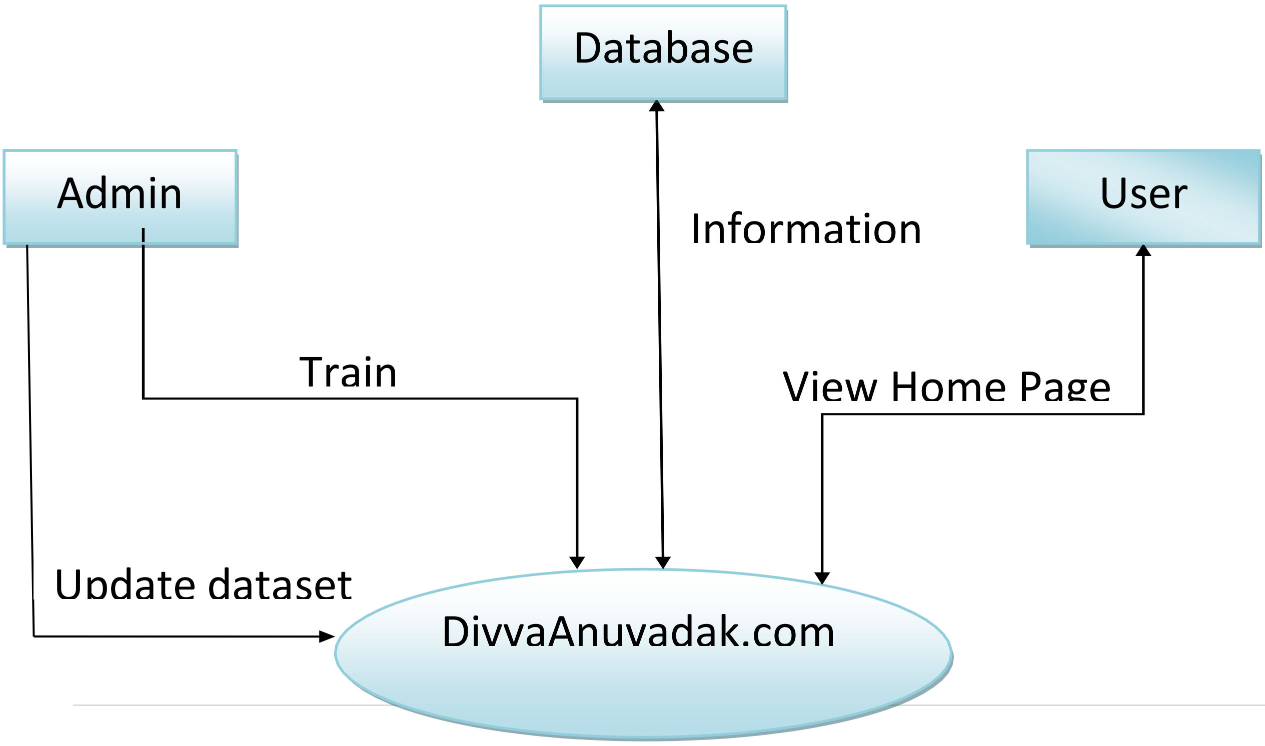


Detailed description of components

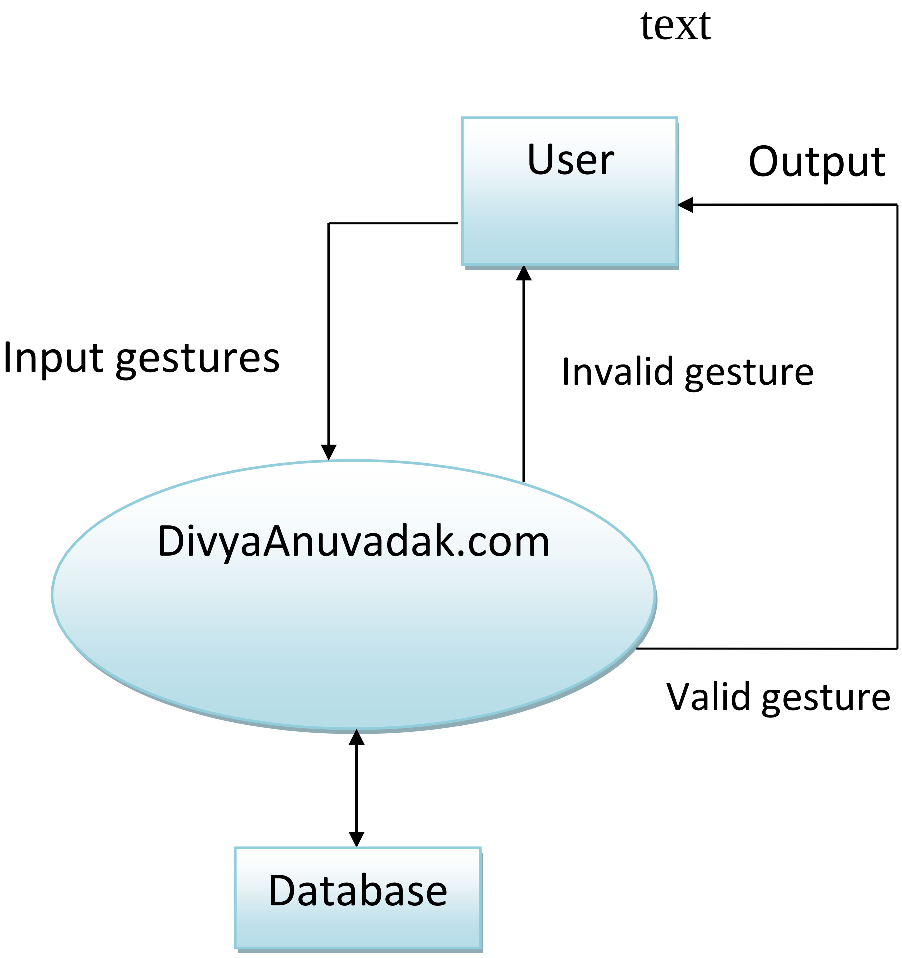
Structure Chart



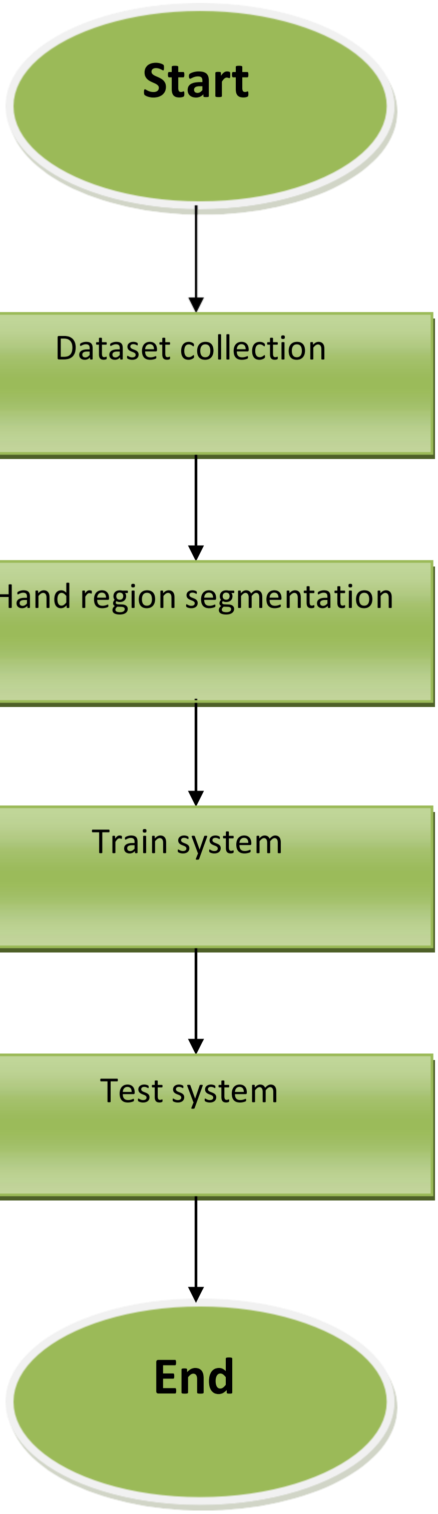
a **0-Level DFD**



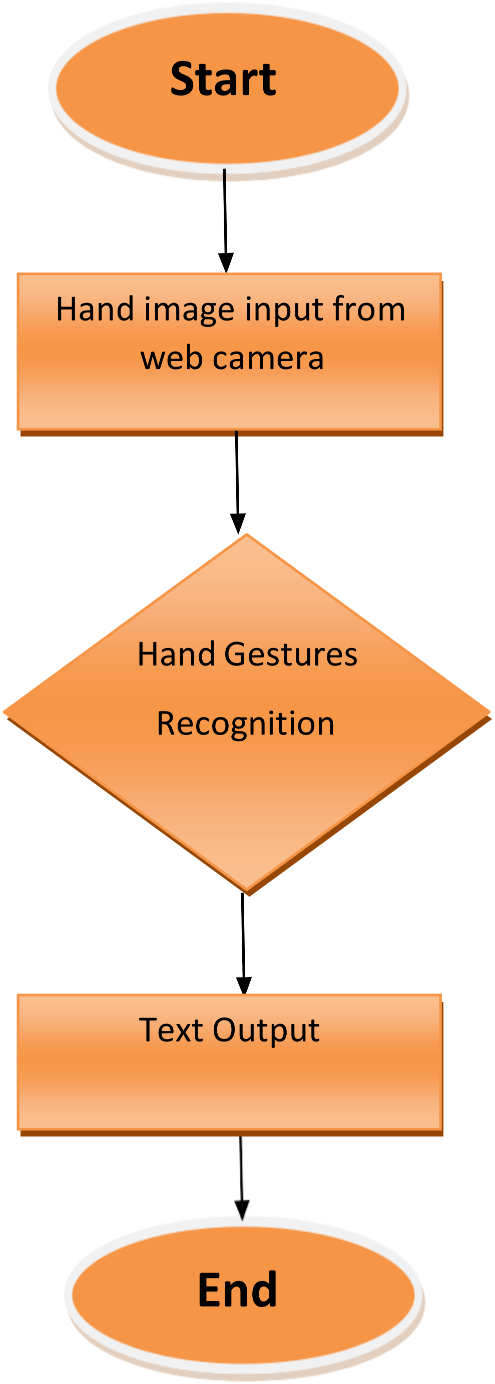
b **1-Level DFD**


text
User
Output
Input gestures
Invalid gesture
DivyaAnuvadak.com
Valid gesture
Database

a **Backend Flowchart**



b **Frontend Flowchart**



Usecase Diagram

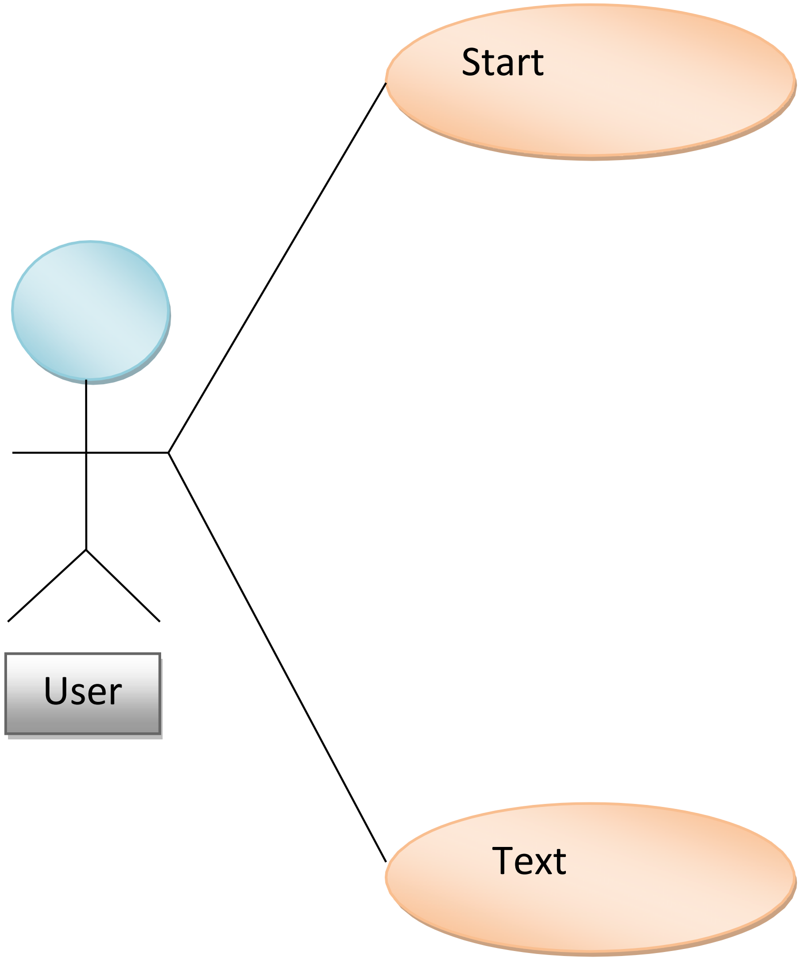
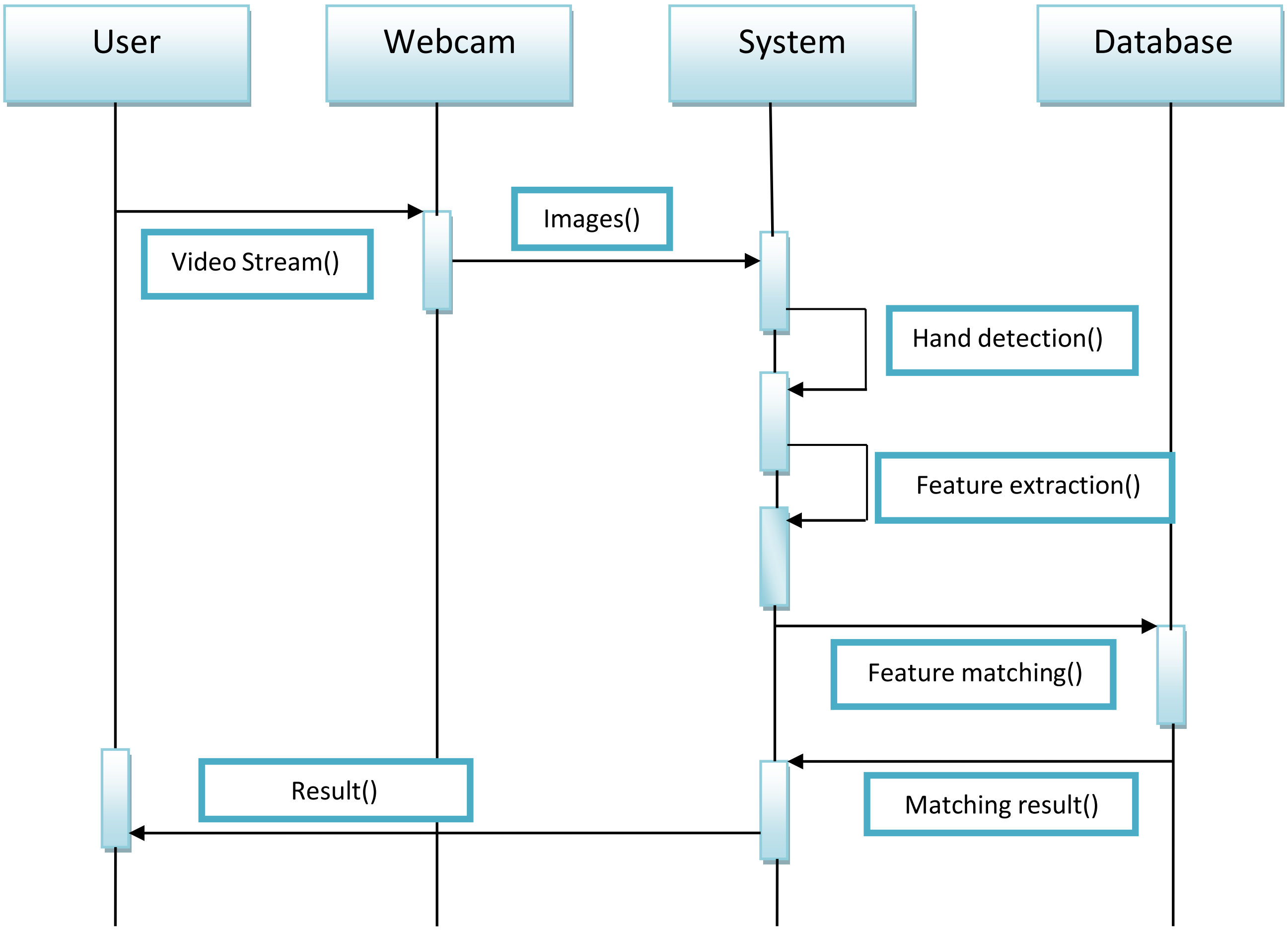


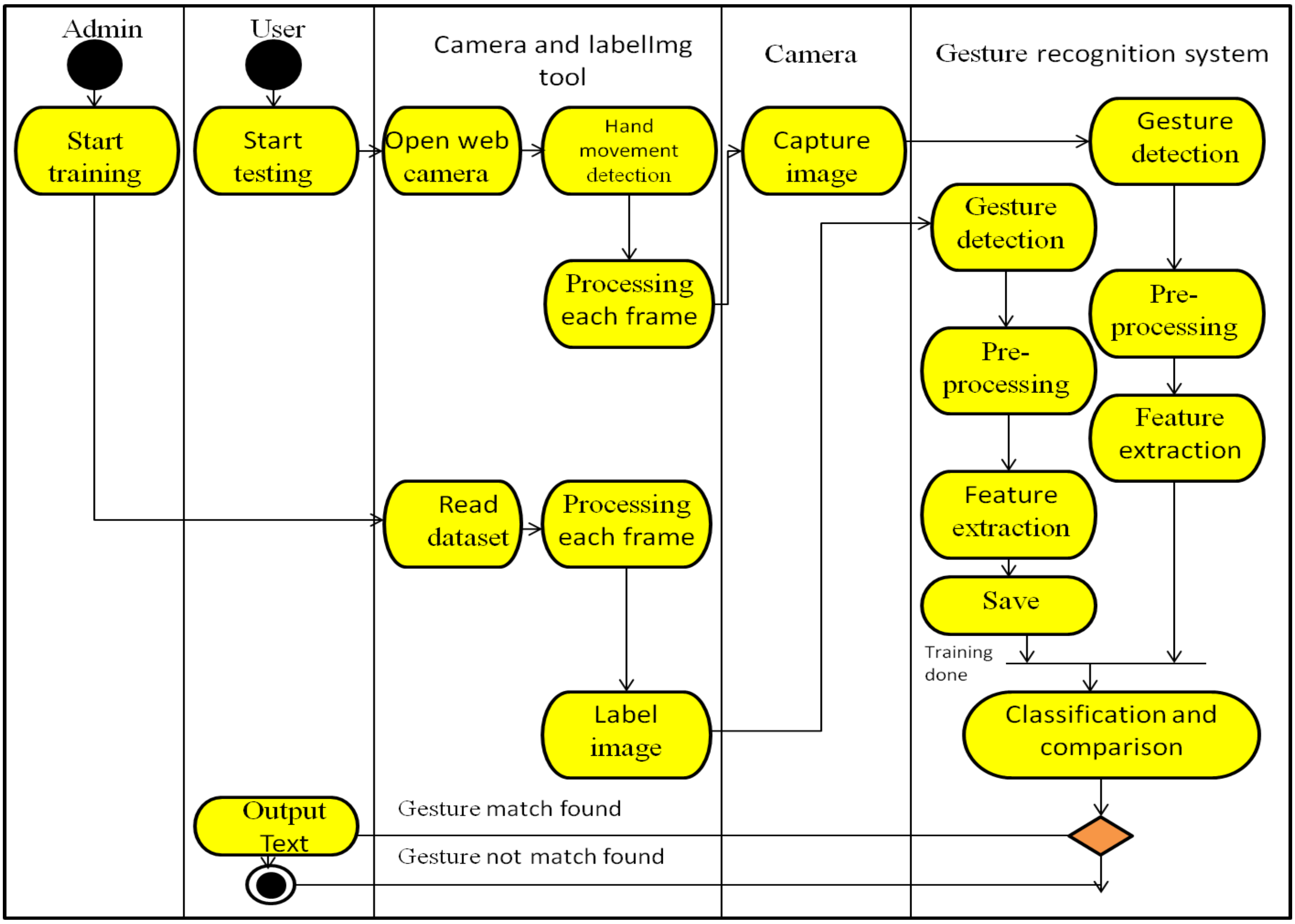
Table : **Usecase Scenario**

Usecase Name	**Sign Language Recognition**
Participating actors	User
Flow of events	Start the system(u) Capturing video(s) Capture gestures(s) Translate gestures(s) Extract features(s) Match features(s) Recognizing gestures(s) Display result
Entry condition	Run the code
Exit condition	Displaying the text label
Quality requirements	Cam pixels clarity, good light condition

Sequence Diagram



Activity diagram



3. Data Design

Dataset description

- ✓ Hello
- ✓ Thank You
- ✓ Yes
- ✓ No
- ✓ I love you

CODE

Capture Images

```
import cv2

import os

import time

import uuid

IMAGES_PATH='Tensorflow/workspace/images/collectedimages'

labels=['hello','iloveyou','no','thankyou','yes']

number_imgs=30

for label in labels:

    !mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}

    cap=cv2.VideoCapture(0)

    print('Collecting images for {}'.format(label))

    time.sleep(10)

    for imgnum in range(number_imgs):

        ret, frame=cap.read()

        imgname=os.path.join(IMAGES_PATH, label,   label+'.'+'{}.jpg'.format(str(uuid.uuid1())))

        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) and 0xFF==ord('q'):
            break
    cap.release()
```

0. Setup Paths

```
WORKSPACE_PATH = 'Tensorflow/workspace'
SCRIPTS_PATH = 'Tensorflow/scripts'
APIMODEL_PATH = 'Tensorflow/workspace/models'
ANNOTATION_PATH = WORKSPACE_PATH+'/annotations'
IMAGE_PATH = WORKSPACE_PATH+'/images'
MODEL_PATH = WORKSPACE_PATH+'/models'
PRETRAINED_MODEL_PATH = WORKSPACE_PATH+'/pre-trained-models'
CONFIG_PATH = MODEL_PATH+'/my_ssd_mobnet/pipeline.config'
CHECKPOINT_PATH = MODEL_PATH+'/my_ssd_mobnet/'
```

1. Create Label Map

```
labels = [
    {'name':'hello', 'id':1},
    {'name':'iloveyou', 'id':2},
    {'name':'no', 'id':3},
    {'name':'thank you', 'id':4},
    {'name':'yes', 'id':5}
]

print(ANNOTATION_PATH)

with open(ANNOTATION_PATH + '\label_map.pbtxt', 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
import pandas as pd
import tensorflow as tf   #.compat.v1
```

2. Create TF records

```
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x {IMAGE_PATH + '/train'} -l {ANNOTATION_PATH +
'/label_map.pbtxt'} -o {ANNOTATION_PATH + '/train.record'}
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x {IMAGE_PATH + '/test'} -l {ANNOTATION_PATH +
'/label_map.pbtxt'} -o {ANNOTATION_PATH + '/test.record'}
```

3. Download TF Models Pretrained Models from Tensorflow Model Zoo

```
!cd Tensorflow && git clone https://github.com/tensorflow/models
```

4. Copy Model Config to Training Folder

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'

!mkdir {'Tensorflow\workspace\models\\'+CUSTOM_MODEL_NAME}
!cp {PRETRAINED_MODEL_PATH+'/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config'}
{MODEL_PATH+'/'+CUSTOM_MODEL_NAME}

!cp{PRETRAINED_MODEL_PATH+'/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config'}
{MODEL_PATH+'/'+CUSTOM_MODEL_NAME}
```

5. Update Config For Transfer Learning

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

```
CONFIG_PATH = MODEL_PATH+'/'+CUSTOM_MODEL_NAME+'/pipeline.config'
```

```
config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
```

```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(CONFIG_PATH, "r") as f:
 proto_str = f.read()                                    text_format.Merge(proto_str, pipeline_config)
```

```
pipeline_config.model.ssd.num_classes = 5
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint =
PRETRAINED_MODEL_PATH+'/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0'
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= ANNOTATION_PATH + '/label_map.pbtxt'
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [ANNOTATION_PATH + '/train.record']
pipeline_config.eval_input_reader[0].label_map_path = ANNOTATION_PATH + '/label_map.pbtxt'
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [ANNOTATION_PATH + '/test.record']
```

```
config_text = text_format.MessageToString(pipeline_config)


with tf.io.gfile.GFile(CONFIG_PATH, "wb") as f:


    f.write(config_text)
```

6. Train the model

```
print("""python {}/research/object_detection/model_main_tf2.py --model_dir={}/{} --
pipeline_config_path={}/{}/pipeline.config --num_train_steps=10000""".format(APIMODEL_PATH,
MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAME))
```

7. Load Train Model From Checkpoint

```
import os
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
```

```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH, 'ckpt-6')).expect_partial()
```

```
@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

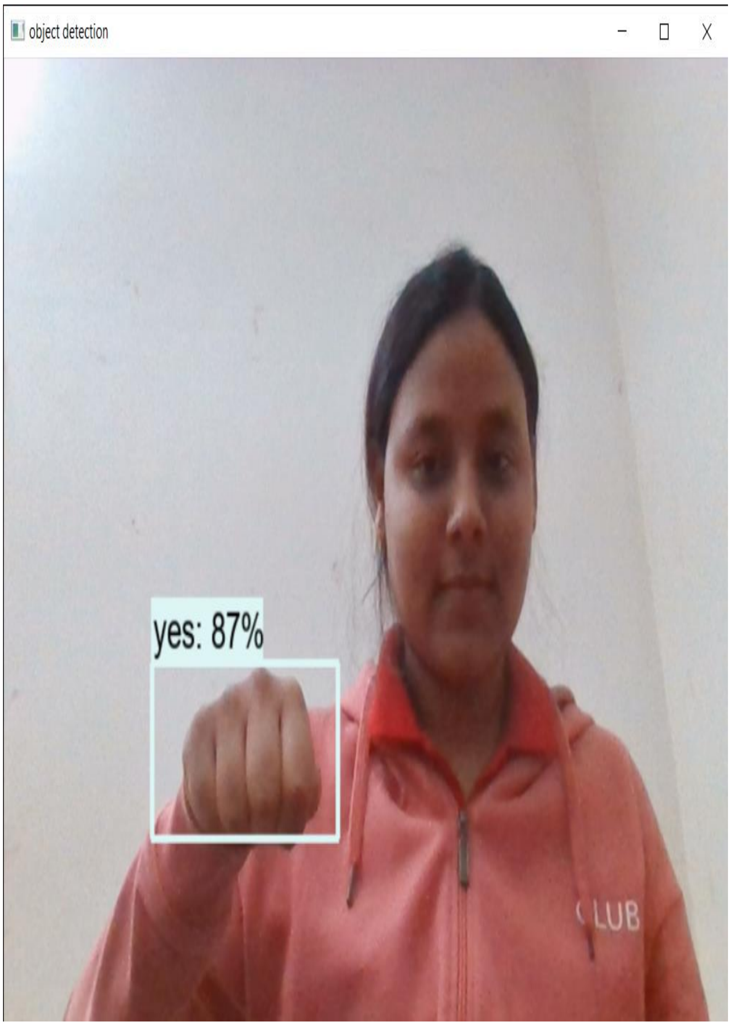
8. Detect in Real-Time

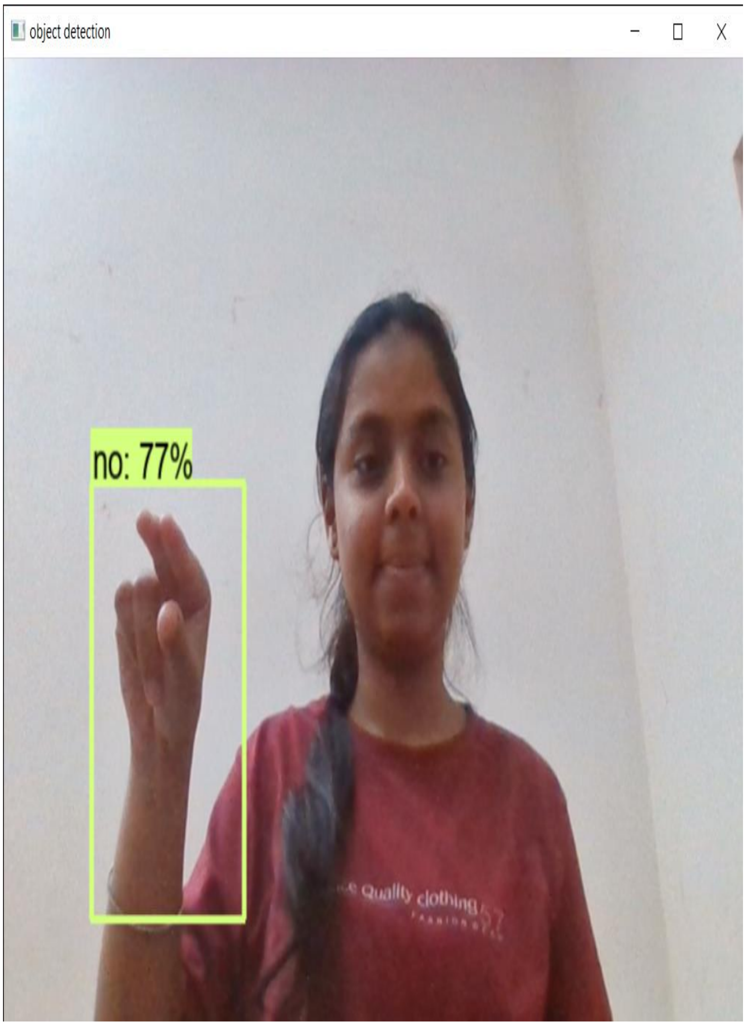
```
import cv2
import numpy as np
category_index = label_map_util.create_category_index_from_labelmap(ANNOTATION_PATH+'/label_map.pbtxt')
# Setup capture
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
while True:
```

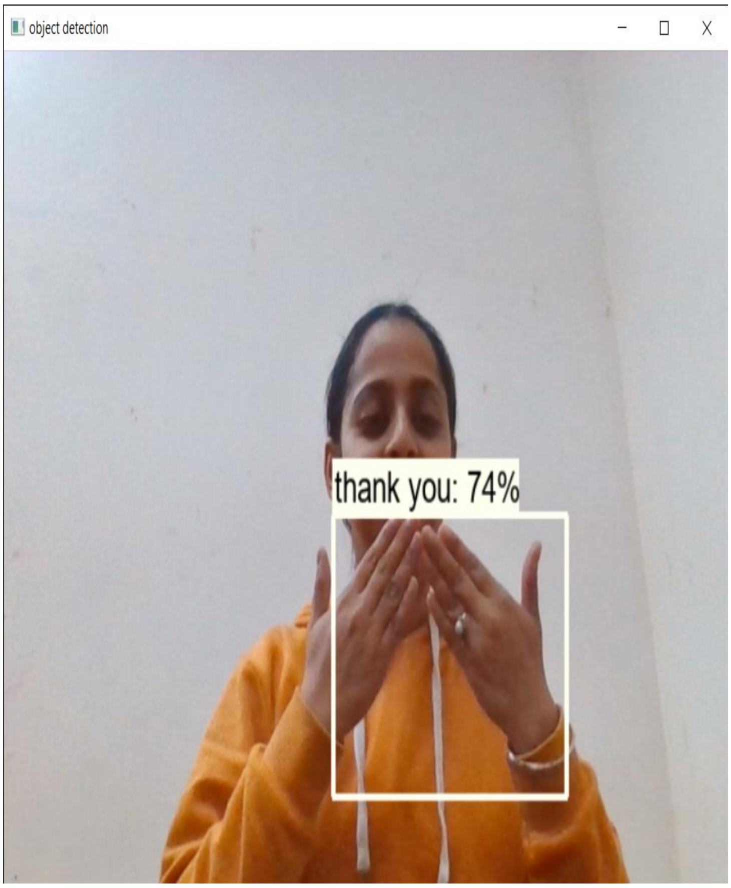
```
ret, frame = cap.read()
image_np = np.array(frame)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=1,
            min_score_thresh=.5,
            agnostic_mode=False)

cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 600)))

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

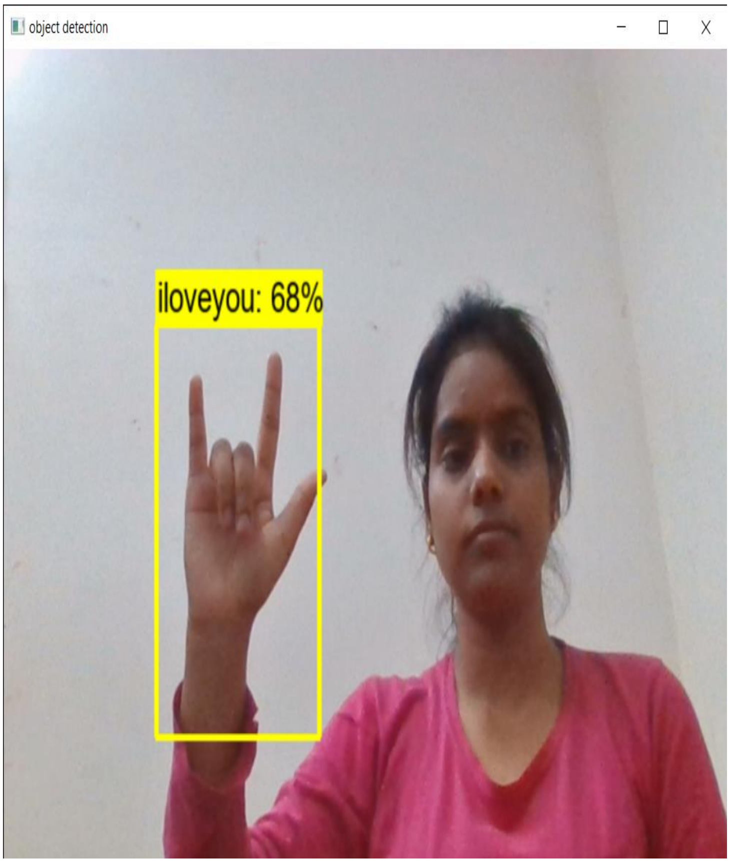
```
cap.release()
cv2.destroyAllWindows()
```

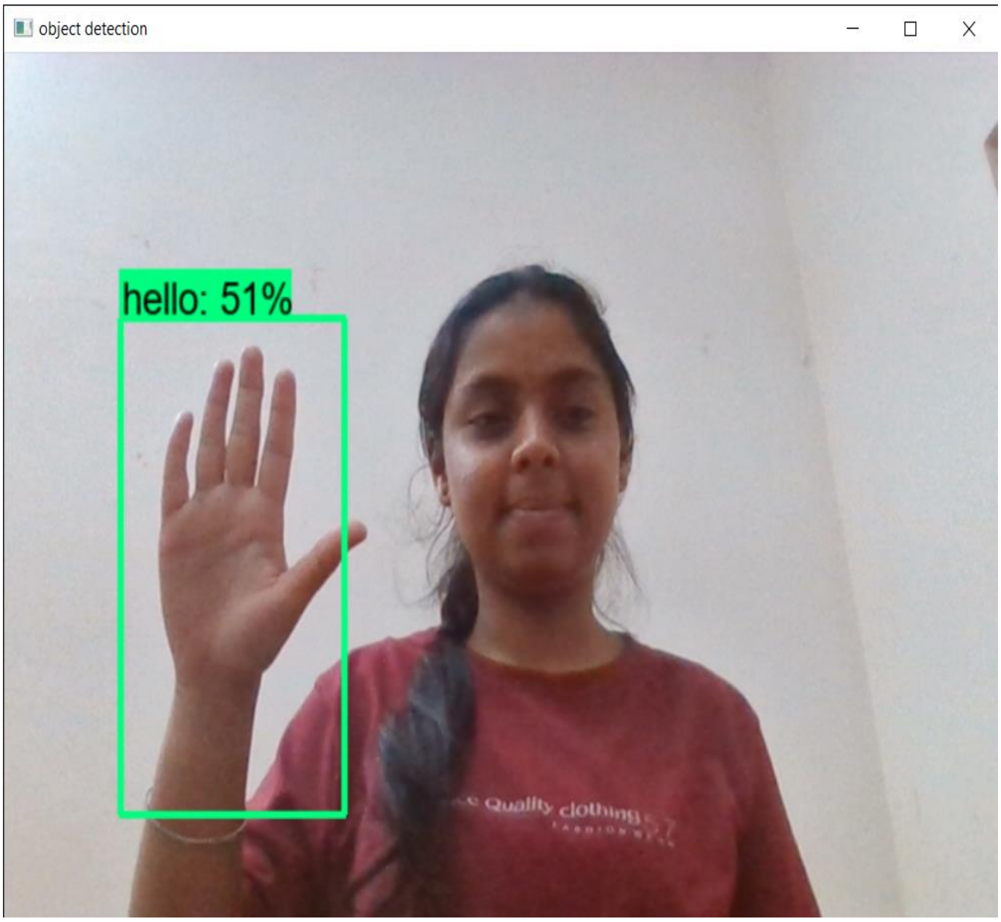
Test Cases






object detection
thank you: 74%


object detection
iloveyou: 68%


object detection
hello: 51%

4. User Interface Design

Description of the user interface

Input – Hand Gesture

Output – Corresponding Text.

Processing Details

- The hand gestures given by the user through web-camera will be processed.
- Then it will be checked whether the input is valid or not.
- Eventually, the system would generate the corresponding output text in the textwindow.

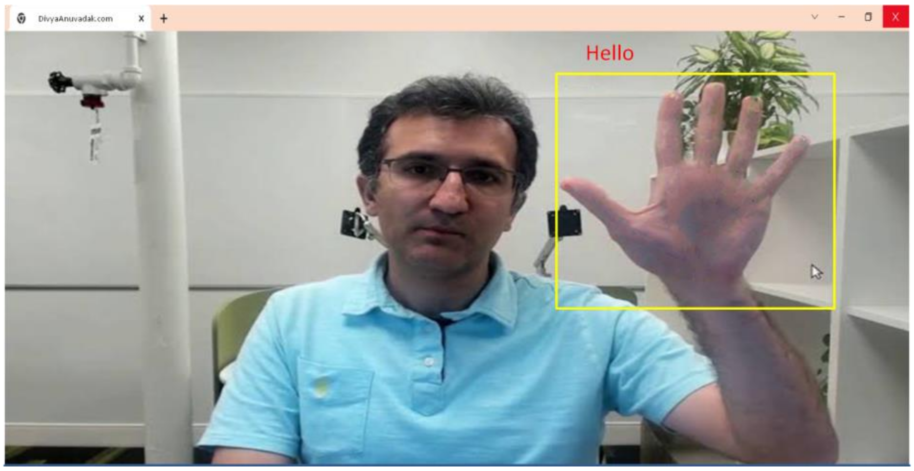
Description of ComponentGesture Window

In this window, user would provide hand gesture as input which would befurther verified and processed by the system.

Text Window

In this window, corresponding text will be displayed with respect to the validhand gestures.

Screen Image



Objects and actions

Object – Textbox

Purpose – Hand Gesture to text

Action – Gesture

Response – Corresponding text output of gesture

5. Types of Tests

Unit Tests

This test is applied on each of the module to find whether or not each module is properly working or not.

System Tests

Black box testing or system testing involves the external working of the software opposite to white box testing which checks for the internal working or code of a software application.

Integration or Regression Tests

After unit test integration testing is done to test if the entire system works together correctly, as a whole. It tries to detect integration related defects by ensuring that the different modules of the software work together correctly and provides the expected results when they are used with each other.

While regression test which is used as synonym of integration test involves retesting the application using the previously executed test data to check for any defects. In short, regression testing tries to ensure that a newly added feature or modified code does not break any functionality already working in the existing system. It is done on each build after it passed all unit tests.

Stress Tests

Here we provide our software with some unfavorable conditions to check how they perform in such conditions. It is particularly used to determine error handling under extremely heavy load conditions to ensure that the system wouldnot crash under crunch situations.

Acceptance Tests and Staging

This test will ensure that the quality of the product is not compromised and also helps in determining to what degree this software meets end users’ approval.

Its implementation can be done by engaging end users’ in the testing process an gather their feedback.

6. References

[1] Pressman Roger S., Software Engineering “A Practitioner’s Approach” FifthEdition, McGraw-Hill Publication, 2000.

[2] IEEE STD 830-1998, IEEE Recommended Practice for Software Requirement Specifications.

[3] "A Concise Introduction to Software Engineering" by Pankaj Jalota

[4] https://ieeexplore.ieee.org/document/720574