Department of Computer Science

Banasthali Vidyapith

Software Requirement Specifications (SRS) and Software Design Specifications (SDS)

Hand Gesture to Text Converter (DivyaAnuvadak)

BTI_G05

Guided By:

Dr. Nisheeth Joshi

Associate Professor

Department of Computer Science

Submitted By:

Ayushi Rajput - 2016730

Deepali - 2016736

Shivalik – 2016821

Sneha Poddar - 2016837

B.Tech. (IT) VI Semester

Chapter 1 SRS

TABLE OF CONTENTS

1. Introduction

2. General Description

3. Specific Requirements

1. INTRODUCTION

1.1 Problem Definition

The main problem that the project aims to solve is to reduce the communication gap of speech or hearing impaired people with the normal ones. As these people cannot express themselves with their words, they face many difficulties in their daily lives. Since almost all of the normal people do not know sign language and cannot understand what these people mean by their special language, tasks such as shopping, settling affairs at a government office becomes so difficult that these people cannot handle by their own.

This problem is very broad and many solutions 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 these people can understand other people by lip-reading, the main problem is that normal people do not understand them. Thus, a more suitable solution should be in the manner that makes these people's language understandable by the other people. For the language of these people to be understood by others, we need the gestures performed by them to be recognized and turned into a form that the others can understand.

1.2 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.

1.3 Scope

- This application will facilitate speech and hearing impaired people to ease their communication.
- It can be used in offices, institutions, seminars etc.
- It would help both our community and university to build a startup and educate these specially-abled people.

1.4 Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specification

ISL: Indian Sign Language

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

VS Code: Visual Studio Code

IDE: Integrated Development Environment

RAM: Random Access Memory

HDD: Hard Disk Drive

GPU: Graphics Processing Unit

1.5 Overview

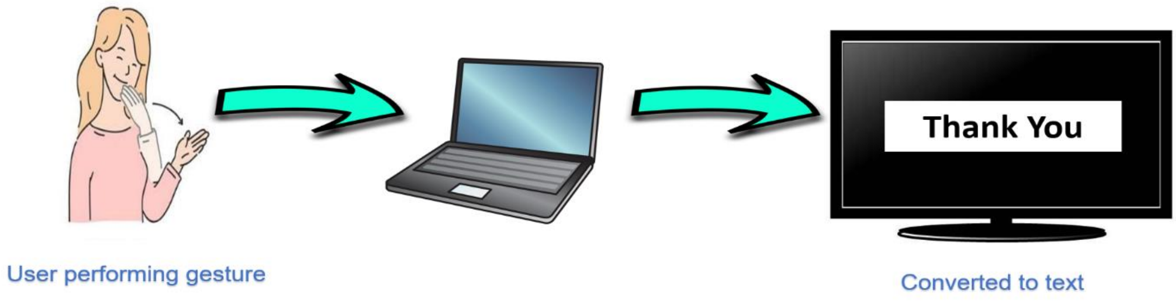
In the following section of this document, we will focus on overall description of the system. This part explains the product perspective, product functions, characteristics, constraints and technologies used in the desired application. Third part of the document explains the specific requirements of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

2. The Overall Description

2.1 Product Perspective

Divya-Anuvadak will serve the speech and hearing impaired people by providing their gestures converted to the equivalent text. Our project focuses on the word recognition through gestures.

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 memory and will be compared with the original gesture performed by the user.



2.1.1 Product Function

- Capture the gestures made by the user through a web camera.

- Recognizing the gestures through Tensorflow Object Detection API by identifying features.
- 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.

2.1.2 Hardware Interface

- **<u>SERVER SIDE</u>**:

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

- **<u>CLIENT SIDE</u>**:

RAM	**:**	8 GB (minimum)
HDD	**:**	512 GB or more
Processor	**:**	Intel® Pentium or faster (2-4 GHz)
Camera	**:**	720 pixels or more

2.1.3 Software Interface

- **<u>SERVER SIDE</u>**:

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

- **<u>CLIENT SIDE</u>**:

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

2.1.4 Communication Interface

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

2.2 User Characteristics

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

2.3 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.

2.4 Technologies Used

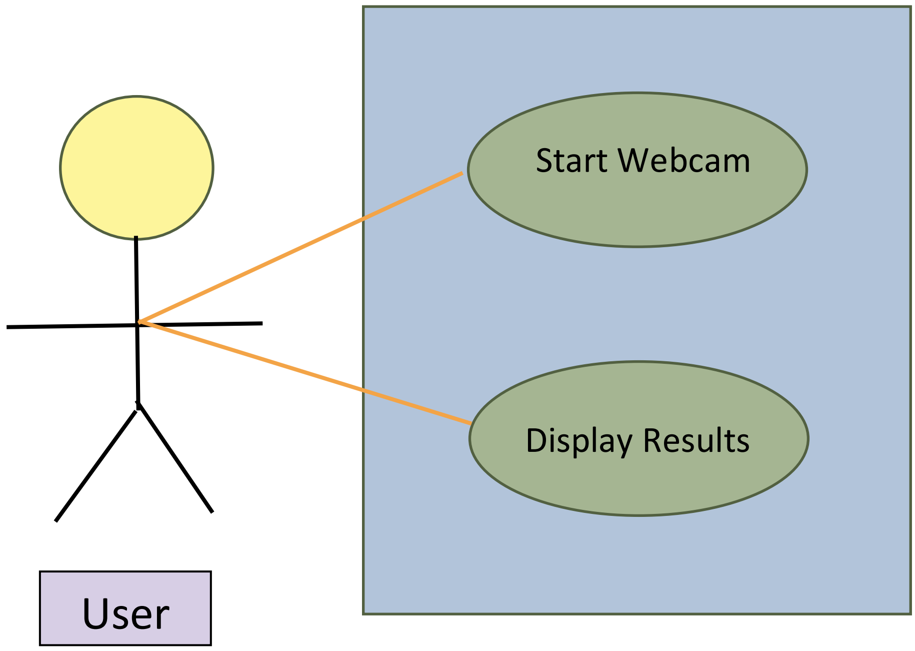
- Python
- Jupyter Notebook
- Keras
- VS Code
- TensorFlow (as Keras uses TensorFlow in backend and for image pre-processing)
- Streamlit

3. Specific Requirements

3.1 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

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

Table : Usecase Scenario

3.2 Non-Functional Requirements

3.2.1 Availability

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

3.2.2 Security

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

3.2.3 Reliability

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

3.2.4 Portability

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

Chapter 2 SDS

TABLE OF CONTENTS

1.Introduction

1.1 Purpose

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

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

1.2 Scope

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

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

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

1.3 Terms,Definitions, Acronyms, and abbreviations

Admin – Administrator

DFD – Data Flow Diagram

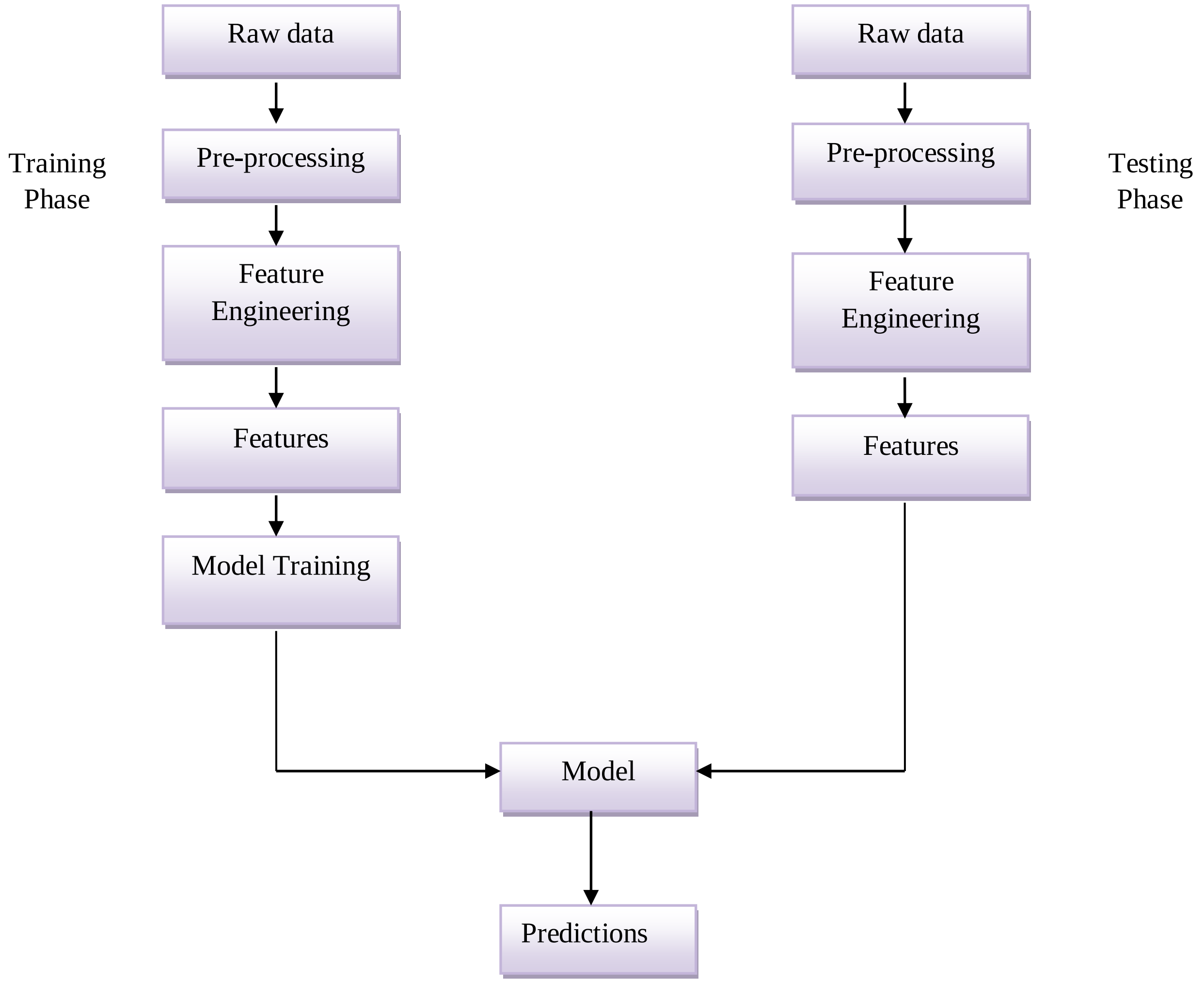
1.4 Overview

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

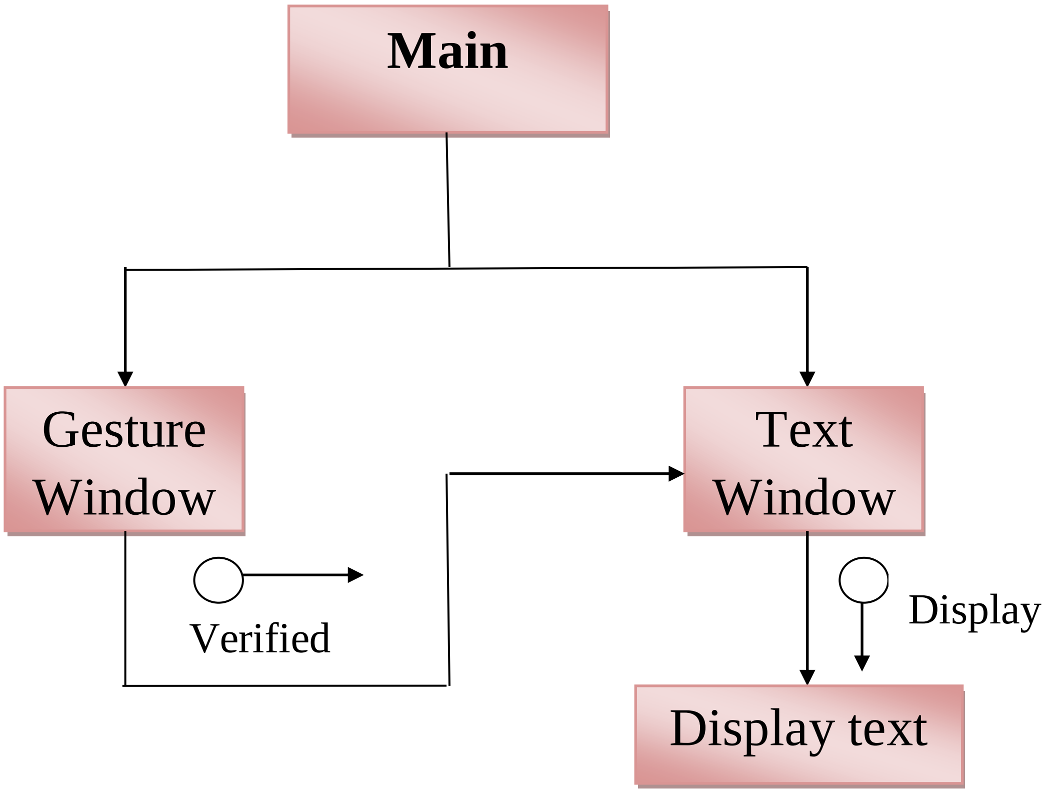
2. System Architectural Design

2.1 High – Level Design

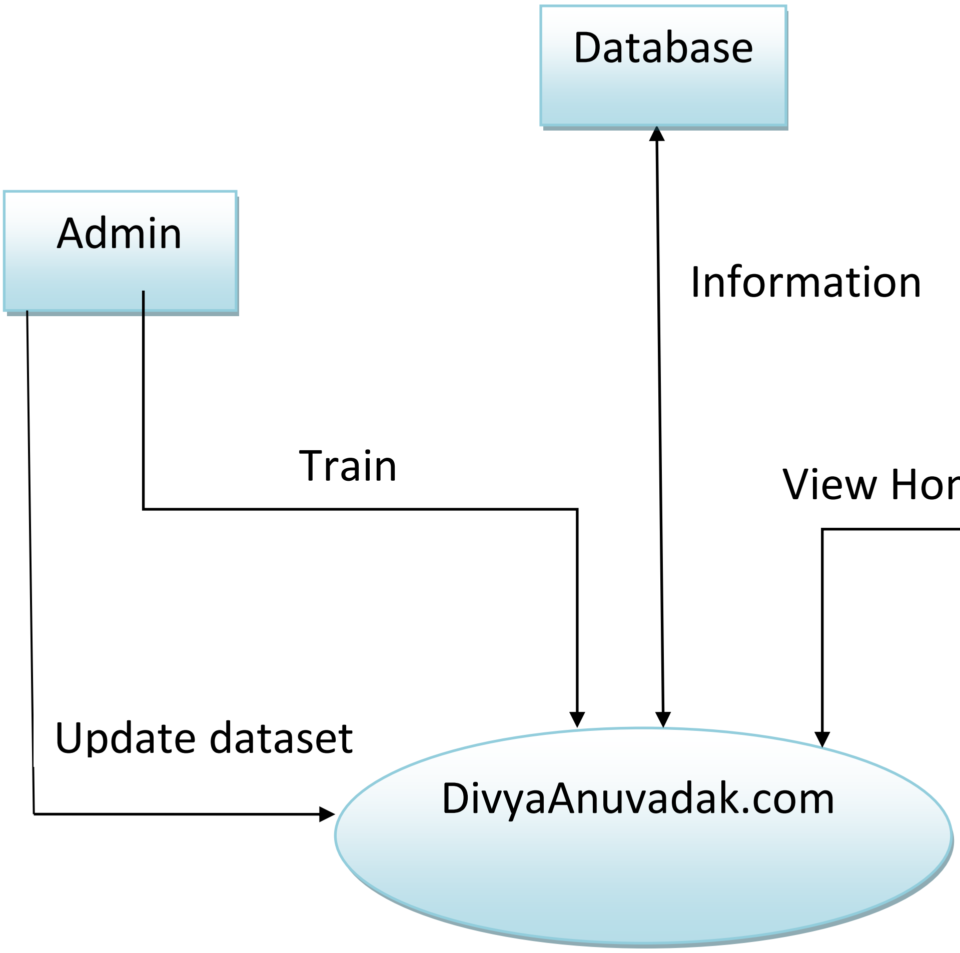


2.2 Detailed description of components

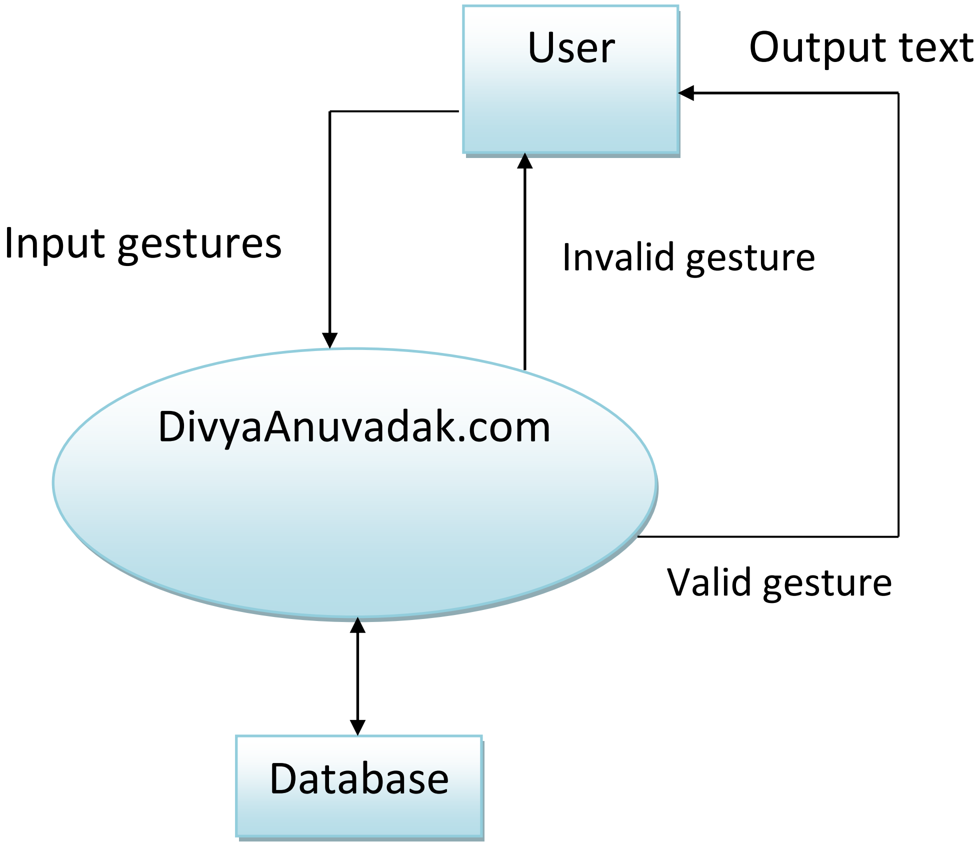
2.2.1 Structure Chart



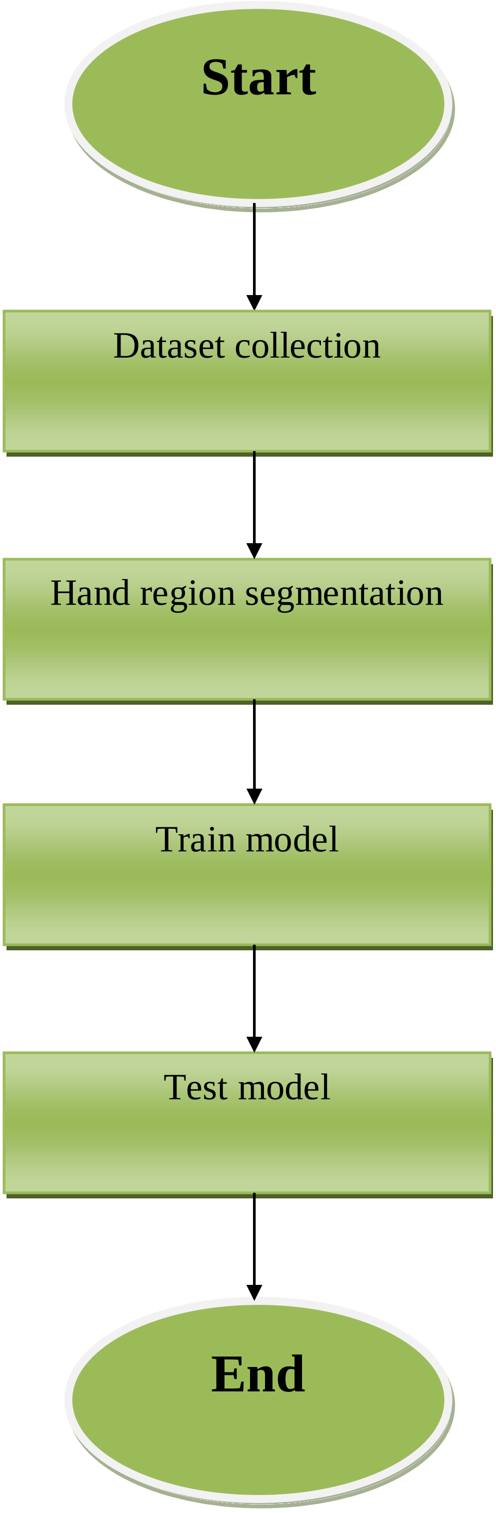
2.2.2a 0-Level DFD



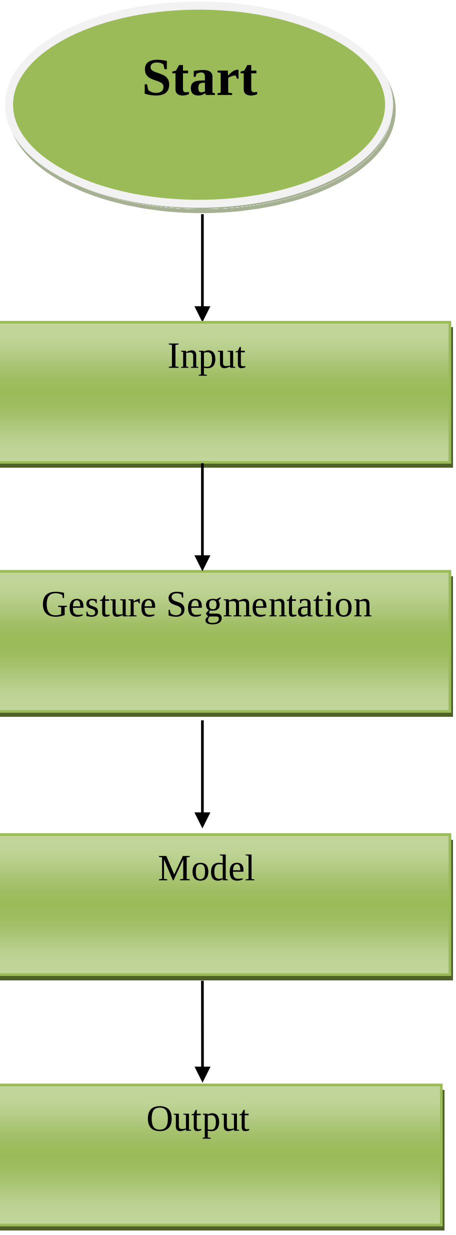
2.2.2b 1-Level DFD



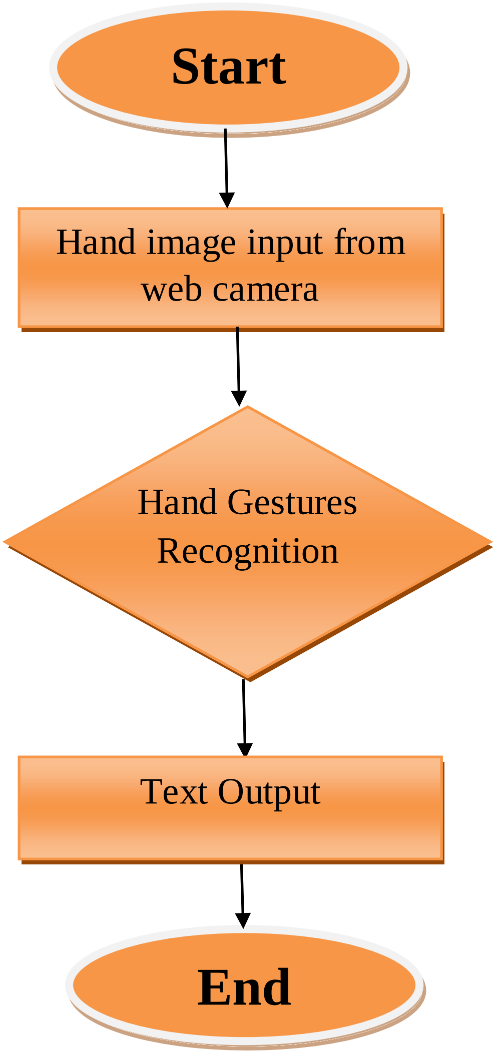
2.2.3a Backend Flowchart



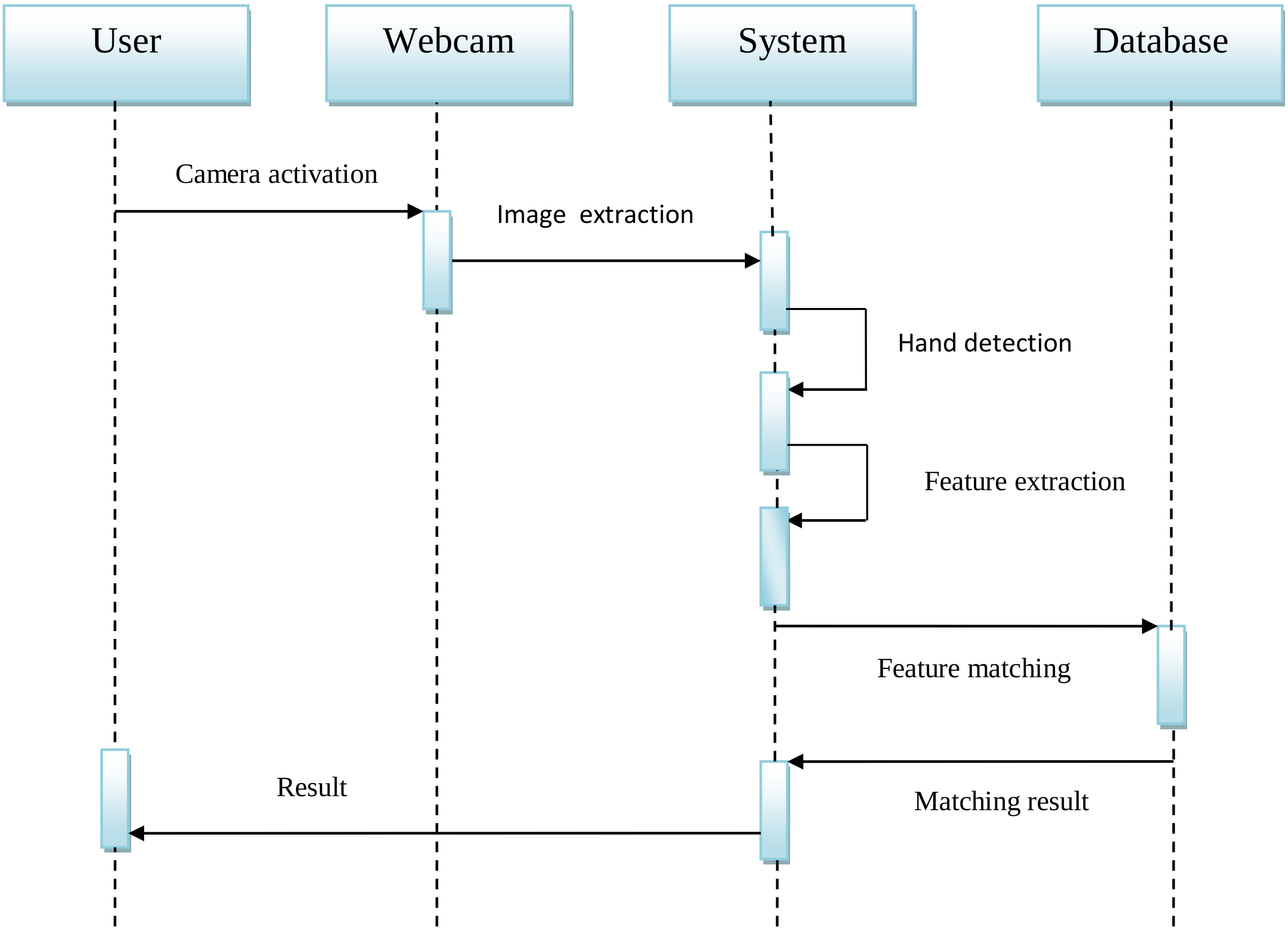
Testing



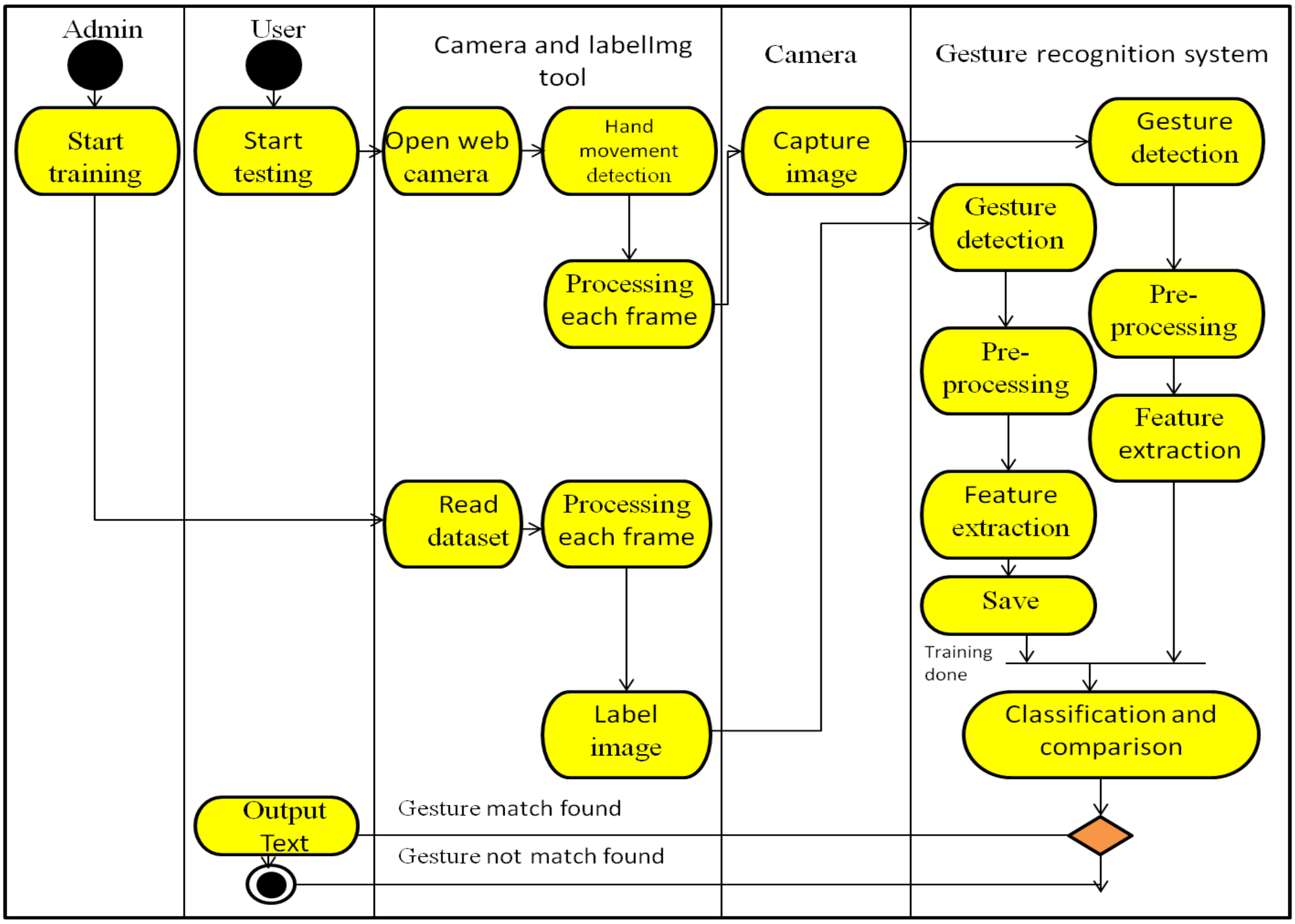
2.2.3b Frontend Flowchart



2.2.5 Sequence Diagram



2.2.6 Activity diagram



3. Data Design

3.1 Dataset description

4. User Interface Design

4.1 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 text window.

Description of Component

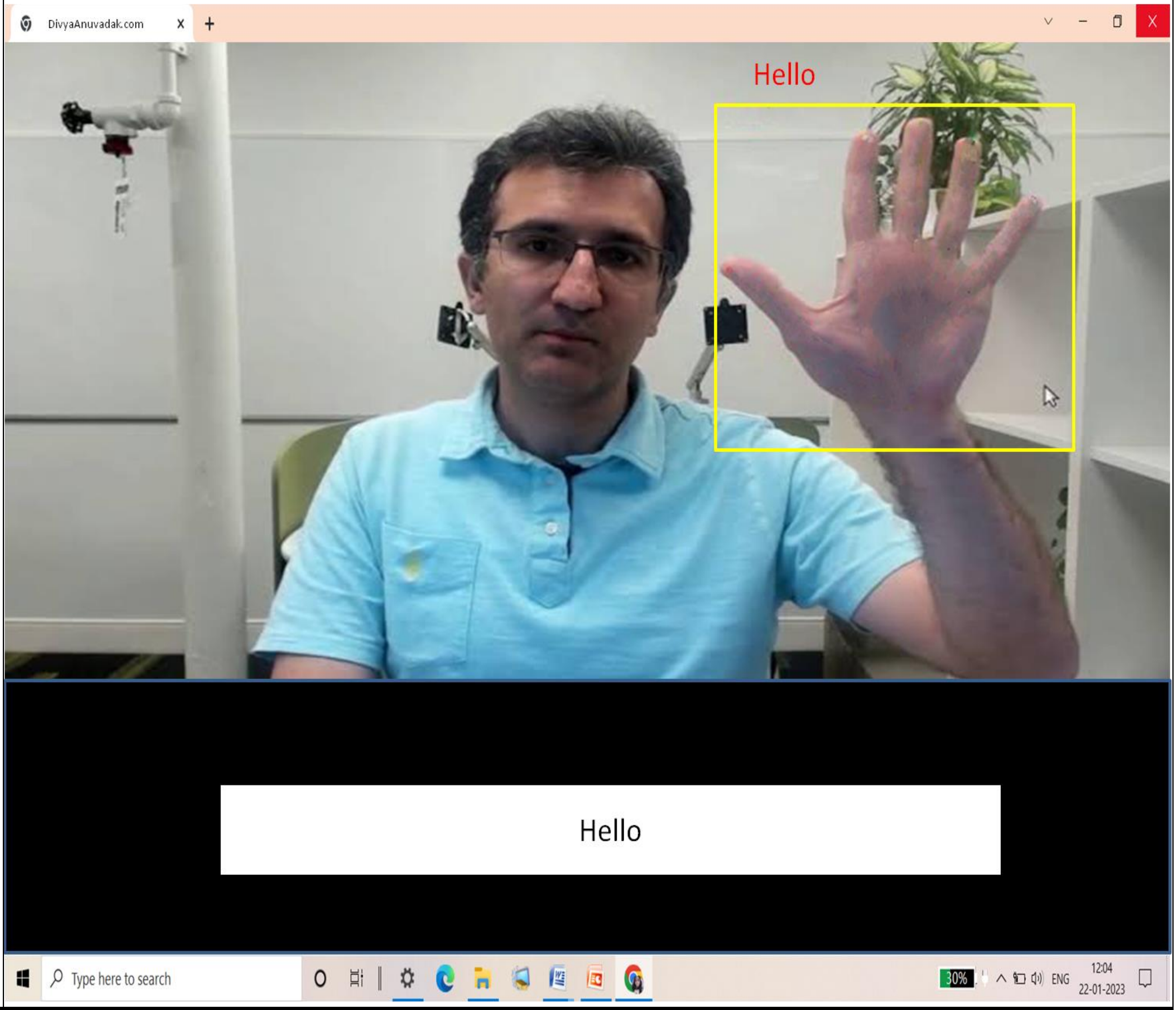
Gesture Window

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

Text Window

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

4.2 Screen Image



4.3 Objects and actions

Object – Textbox

Purpose – To convert hand gestures into text

Action – Gesture

Response – Corresponding text output of gesture

5.Types of Tests

5.1 Unit Tests

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

5.2 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.

5.3 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.

5.4 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 would not crash under crunch situations.

5.5 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" Fifth Edition, McGraw-Hill Publication, 2000.

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