

Low Level Document

COMPUTER VISION SOLUTION FOR HEARING-IMPAIRED

Written By	Niranjana Shrestha, Prajin Bajracharya
Document Version	1.1
Last Revised Date	24-March-2022

Document Control**Change Record:**

Version	Date	Author	Comments
V 1.0	3/21/2022	Niranjana Shrestha, Prajin Bajracharya	Unit Test Cases Technology Stack
V 1.1	3/24/2022	Niranjana Shrestha, Prajin Bajracharya	Directory Tree Structure

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Table of Contents

1. Introduction	4
1.1 What is Low-Level design document?	4
1.2 Scope	4
1.3 Constraints	4
2. Architecture	5
3. Architecture Description	6
3.1 Data Collection and Preprocessing	6
3.2 Data Cleaning and Dataset Creation	6
3.3 Model Creation	6
3.4 Model Training	7
3.5 Model Evaluation	7
3.6 Hyper parameter Tuning	8
3.7 Model Dump	8
3.8 Model and Metrics Visualization	8
3.9 Model call for Specific input	8
3.10 Database Connection	9
3.11 User Interface	11
3.12 Deployment	14
4. Technology Stack	15
5. Directory Tree Structure	16
6. Unit Test Case	18

1. Introduction

1.1 What is Low-Level design document?

The goal of LLD or Low-level design document (LLDD) is to give the internal logical design of the actual program code for Computer Vision Solution for Hearing-Impaired. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

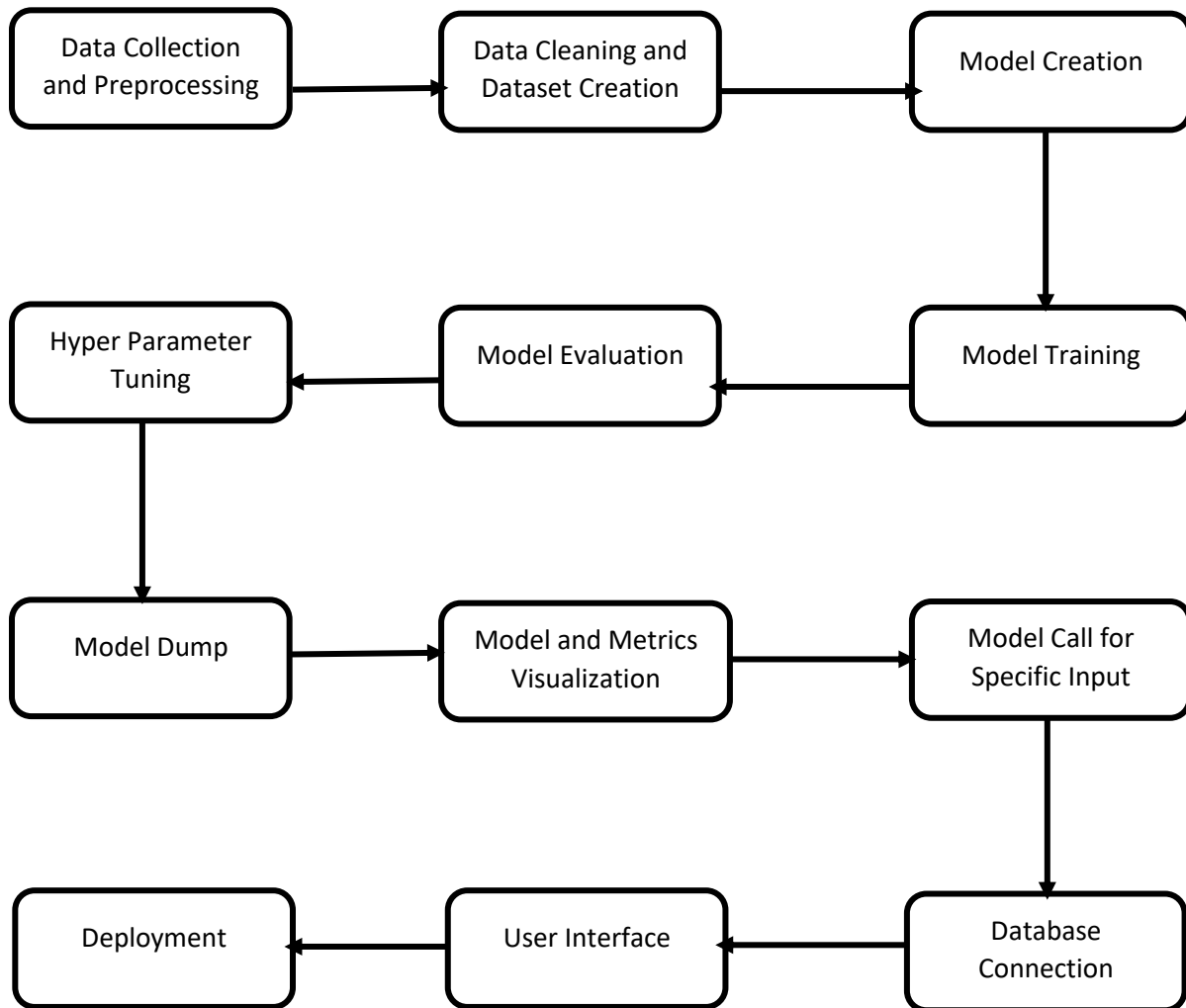
1.2 Scope

Low-Level Design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

1.3 Constraints

Background of the real time capturing video is the main constraint for the application. As the application starts to detect multiple number of objects such that the prediction is misleading and the application can't focus in constant accurate output. Similarly, the user must wear glove such that it would reduce lighting effects from bare palm of the hand.

2. Architecture



3. Architecture Description

3.1 Data Collection and Preprocessing

Since the dataset provided by kaggle, google and any other sites aren't that appropriate for our model. So, we decided to create a simple python program that can capture desired number of images from a real time video through the use of webcam for each letters or information and separate each of them in folder. However for that, we have used various functions of OpenCV which are listed below:

- Transformation
- Smoothing
- Edge Detection
- Segmentation
- Thresholding
- Contours Detection

3.2 Data Cleaning and Dataset Creation

In the Cleaning process, the captured images are represented in in the form of array as well as appropriate label are also converted in the form of array and those array of images and labels didn't needed any kind of additional change or transformation because while capturing those images it had captured appropriate images following the methods and operations of OpenCV. So for the data validation we created the pickle file which contained dataset that was splitted in different dataset with different numbers such as training, testing and validation dataset so that it could be further processed in the neural net.

3.3 Model Creation

Creating the convolutional neural network model involved making choices about various parameters and hyper-parameters. We used TensorFlow as main python library to implement CNN We took decisions about the number of layers to use in our model. The model used 8*2 layers including number of repeated max pooling and convolution layer.

In first step we have used the layer Conv2d which is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs. Thus, we have used 16 filter along with kernel size of 2*2 with input shape of

256*256*1 dimension where none determines the parameter can be in variable state with activation function as relu. Then after this we have used Maxpooling2D with pool size of 2*2 pooling windows along with 2*2 strides with padding of same size. Similarly the Conv2d and Maxpooling 2d is repeated again 2 times with 32 and 64 filters along with same activation function and also pooling size 3*3,5*5 and strides also 3*3 , 5*5 with the padding same respectively. Then the output from here goes to the flatten layer which reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension. Thus, the output goes to the dense layer containing 128 neurons which feeds back the output to the neurons with activation function relu. Then the output from it goes to the dropout layer which select 20 % of the neurons and set their weights to zero. At last, we use again the dense layer with number of classes we have used that is 34 as neurons to thus help reach the weights to appropriate class.

3.4 Model Training

After we have created our model, we simply created an instance of the model and fit it with our training data. The biggest consideration when training a model is the amount of time the model takes to train. To consume less time, we used various parameters for training the model. Such as we used Adam as an optimizer, learning rate as 3e-4, loss as categorical cross entropy etc. We can specify the length of training for a network by specifying the number of epochs to train over. Thus, we specified 15 epochs. The longer we train a model, the greater its performance will improve, but too many training epochs and you risk overfitting. But we were able to train for limited time only due to lack of time and for risk of overfitting we used dropout layer as well. Currently we used only 26 letters, 2 special function and 6 words for training each with dataset about 2000.

3.5 Model Evaluation

For model evaluation, the model's performance was compared against a validation dataset, a data set that the model hasn't been trained on. The model's performance was compared against this validation set and analyzed its performance through different metrics. The most common metric is "accuracy", the amount of correctly classified images divided by the total number of images in your data set. After viewing the accuracy of the model's

performance on a validation dataset, we trained the network again using slightly tweaked parameters. Finally, the network's performance was tested on a testing set.

3.6 Hyper parameter Tuning

For Hyper Parameter tuning, we used Manual Search algorithm because it was easy for us changing the parameter manually and by changing little we got the required point where bias was low and variance was low according to bias-variance tradeoff.

3.7 Model Dump

After comparing all accuracies and checking the ROC as well as AUC curve accuracy we saved or dumped the model in the form of .h5 extension which is regarded as default model extension from keras.

3.8 Model and Metrics Visualization

Model Visualization provides reason and logic behind to enable the accountability and transparency on the model. Machine Learning models considered as Black Box Models due to complex inner workings. Data Scientists deliver a model with high accuracy. There are some scenarios where models cannot be explained to the public because the system may hack. For the Model Visualization, we used confusion matrix as it is regarded as one of the main method for model visualization.

Visualization metrics can be defined as the metrics that are calculated to measure the attributes and capture the properties of visualization, to extract the meaningful information of data. We used python library known as Tensorboard to analyze and visualize the different metrics such as loss and accuracy of Training and Validation phases

3.9 Model call for Specific input

Model Call is done by loading the model and opening the web cam. Then hand signs were waved in the dedicated Region of Inference (ROI) so that how much the model is accurate. After the model evaluation, we used OpenCV to capture the real time video through various OpenCV programming codes to create a window that can capture the real time video along with ROI (Region of Interest) where the hand sign is predicted from that area only and we also used the methods of OpenCV just like in data collection which are:

- Transformation

- Smoothing
- Edge Detection
- Segmentation
- Thresholding
- Contours Detection

Thus these methods help to increase more accuracy of predicting the sign of the hand. We also created another window to show the letter that has been detected. Currently due to low accuracy from training we were able to predict these hand signs:

- 26-class fingerspelling recognition
- 2-extra function recognition
- 6-word class fingerspelling recognition

3.10 Database Connection

For the different purposes of User Interface such as login, signup and other authorization processes we used Cassandra database for storing the emails, usernames and passwords of various user so that our web app is much more secure and much more interactive. So for the initiation we made a database named flask in datastax astra which creates databases that runs Multi-cloud DBaaS Built on Apache Cassandra. After that, we wanted to connect the flask database that we named, to our flask application. Hence, we connected the database using the steps mentioned in datastax astra website with proper guidance and steps in it. Then after successful connection we used CQLAlchemy one of the library of Flask for helping Cassandra database synchronization in flask environment. Then we used varieties of functions related to CQLAlchemy for getting and sending the data into and from the database respectively. The database mainly contains following type of data:

SN	Records	Type	Key
1	id	UUID	Primary
2	first_name	Text	-
3	Last_name	Text	
4	username	Text	Primary
5	email	Text	Primary
6	password	Text	Primary

The screenshot shows the DataStax Astra dashboard. The left sidebar contains navigation links: Dashboard, Databases (with a 'Create Database' button), Streaming (with a 'Create Streaming' button), and Sample App Gallery. The main content area is titled 'Dashboard / flask' and includes tabs for Overview, Health, Connect, CQL Console, and Settings. Under the Overview tab, there's a section 'Usage For Current Billing Period for flask' showing metrics: Read Requests (121), Write Requests (8), Storage Consumed (155.67 KB), and Data Transfer (44.07 MB). Below this is a 'Regions' section with a table listing available regions. The table has columns: Provider, Area, Region, Datacenter ID, and Region Availability. One region is listed: Google Cloud, North America, us-east1, 7f3e974e-593e-4870-84cf-cd369a3604ae-1, Online. There's also a 'Keyspaces' section with a table listing 'flaskspace'.

Provider	Area	Region	Datacenter ID	Region Availability
Google Cloud	North America	us-east1	7f3e974e-593e-4870-84cf-cd369a3604ae-1	Online

Keyspace
flaskspace

```
cloud_config= {
    'secure_connect_bundle': 'secure-connect-flask.zip'
}

auth_provider = PlainTextAuthProvider(username='weUCOUqsJFMQzUX1Iu0BQaw', password='iZr5Kst00A01fs1KMlb1f76laxAAIxPu70vx9N, L1l0e3duihCio5Z217bn.0tyVw_ W1H8HAjqU307ERTzZugv6x0LW8q, b0Rjt4gvmf7H, pJ544D0bc, Z4ugPbYrH')

app.config['CASSANDRA_HOSTS'] = ['7f3e974e-593e-4870-84cf-cd369a3604ae-us-east1.db.astra.datastax.com']
app.config['CASSANDRA_SETUP_KWARGS'] = dict(cloud=cloud_config, auth_provider=auth_provider)
app.config['CASSANDRA_KEYSPACE'] = "flaskspace"
db = CQLAlchemy(app)
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()
```

3.11 User Interface

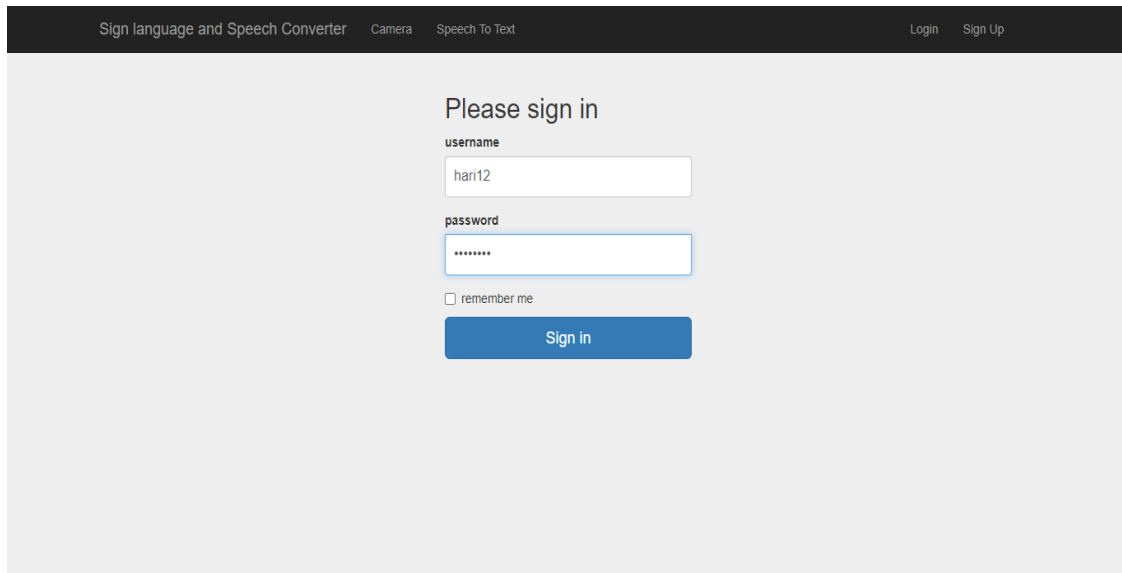
Flask is a web framework. This means flask provides us with tools, libraries and technologies that allow us to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

So to access our system through a web page we have integrated our system to the flask frontend as flask is the web framework of python as well so to implement the system build in python wasn't quite difficult. So our flask web app is made simple which consists of simple home page, login page, signup page, profile page and camera page hosted at port 8000. We have loaded the model 'handsign.h5' with the help keras library successfully in flask frontend. Then we have created all the templates, interfaces, small database using CQLAlchemy i.e. Cassandra library for python for the web app. Also we have created various functions for login, signup, etc. Finally to get the result from the camera page, we have converted the OpenCV camera output to an image bytes and hence it becomes suitable to be implemented in the flask frontend and thus real time video can be generated in the camera page. Thus the camera will detect the Hand signs and display result in same camera page.



WELCOME TO COMPUTER VISION SOLUTION FOR HEARING-IMPAIRED





Sign language and Speech Converter Camera Speech To Text Login Sign Up

Please sign in

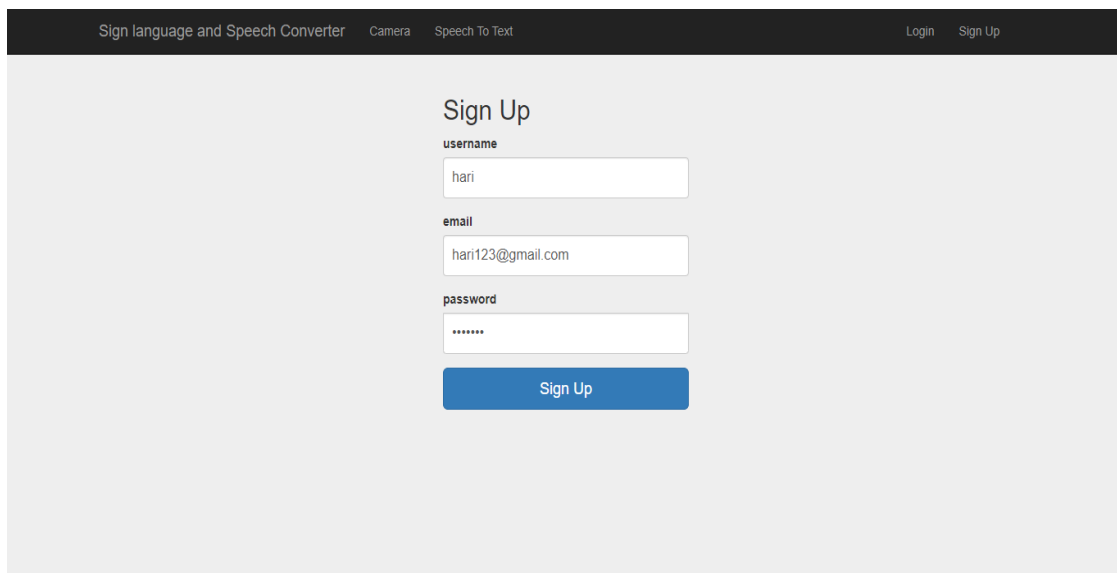
username

hari12

password

☐ remember me

Sign in



Sign language and Speech Converter Camera Speech To Text Login Sign Up

Sign Up

username

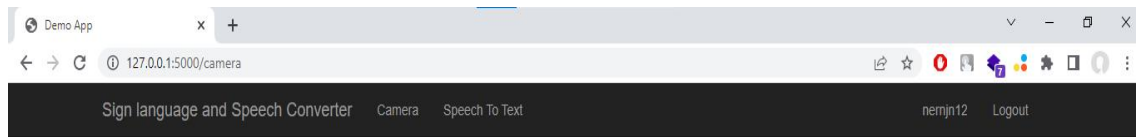
hari

email

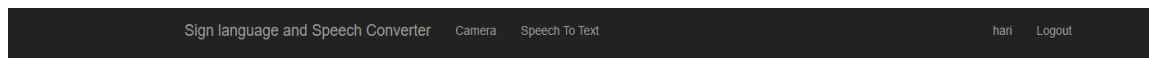
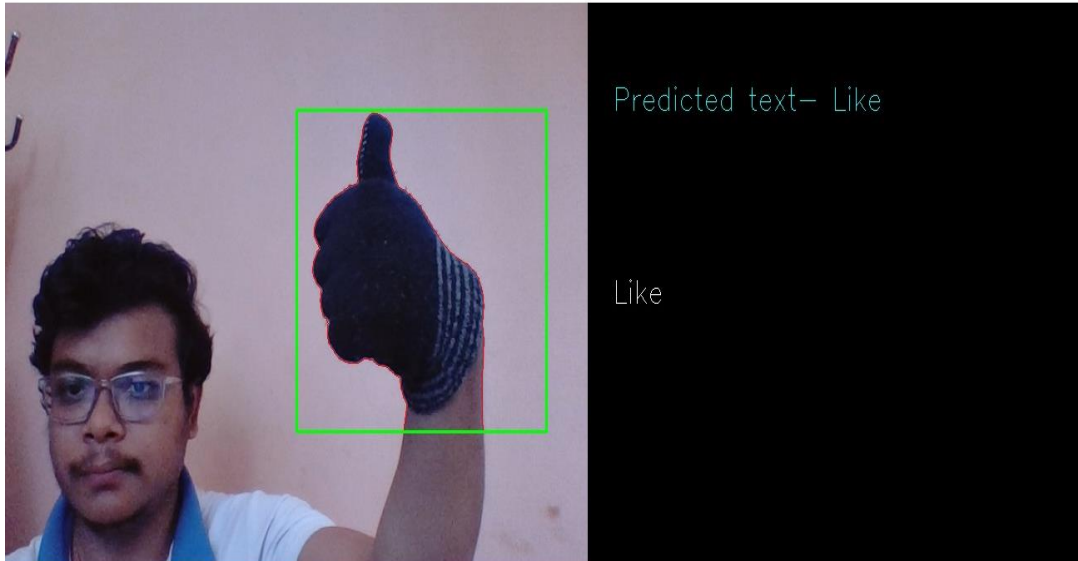
hari123@gmail.com

password

Sign Up



Streaming



Convert speech to text

Step 1 - Record your speech

Convert recorded audio to wav:

Waveform Audio (.wav) ▼

Record

Stop

Log

Step 2 - Download your record

Step 3 - Upload your .wav file

Choose File No file chosen

Step 4 - Convert audio to text

Convert

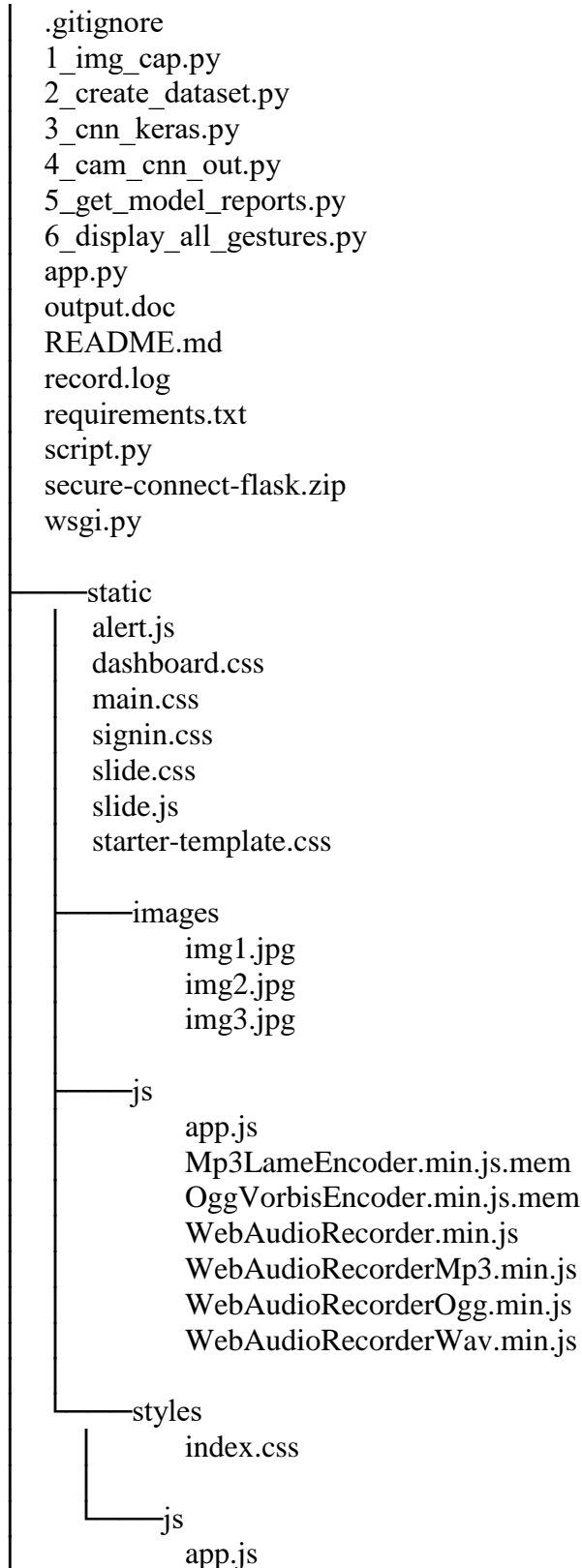
3.12 Deployment

As it was difficult for deploying our model in the docker or any other cloud deployment apps as it required camera to load through OpenCV and required heavy resources with paid version that we couldn't afford it, so we deployed in the local host for our ease. Due to these undeniable circumstances we were compelled to deploy in local host as output with a proper justification of our system design.

4. Technology Stack

Frontend	HTML, CSS, Bootstrap
Backend	Python, Tensor flow, Keras, Numpy ,Flask, Tensorboard,Pyttsx3,Pygame, Scikit-learn etc
Database	Cassandra
Deployment	Localhost

5. Directory Tree Structure



- Mp3LameEncoder.min.js.mem
- OggVorbisEncoder.min.js.mem
- WebAudioRecorder.min.js
- WebAudioRecorderMp3.min.js
- WebAudioRecorderOgg.min.js
- WebAudioRecorderWav.min.js

- templates

- camera.html
 - dashboard.html
 - index.html
 - layout.html
 - login.html
 - signup.html
 - speech_text.html

- __pycache__

- app.cpython-37.pyc
 - camera.cpython-37.pyc
 - cnn_sgn.cpython-37.pyc
 - script.cpython-37.pyc
 - test.cpython-37.pyc
 - wsgi.cpython-37.pyc

6. Unit Test Case

Test Case Description	Pre – Requisite	Expected Resulted
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify Response time of URL from backend model.	1. Application is accessible	The latency and accessibility of application is very faster we got in Local host.
Verify whether user is giving standard input.	1. Handled test cases at backend.	User should be able to see successfully valid results.
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. User is logged in to the application	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Prediction Webcam	1. Application is accessible 2. User is logged in to the application	User should get view of Webcam real time video
Verify whether user gets text from Speech To Text web page	1. Application is accessible 2. User is logged in to the application	User should be able to see converted text from the audio/speech uploaded