

I) Introduction

The Report given below is a culmination of our Literature Survey, Project Requirement Specification Document, High Level Design Document and the Implementation that were written as a part of the Capstone Project Phase -1. The final submission of the Project is a Partial Implementation of the Project i.e is the Preprocessing Phase that would be submitted along with the project report. This Report consists of a Problem Statement, Literature Survey, Project Requirement Specification, High Level Design Document, Detailed System Design, Implementation, a Conclusion of Capstone Phase -1 and a Plan of what would happen in phase 2.

II) PROBLEM STATEMENT:

To build a Indian Sign Language Translator i.e to convert the signs shown by a person in a video to a meaningful caption.

Such a model is very useful when integrated with a mobile app. This not only helps the deaf-dumb people but also the people who interact with them

The model must be robust to variations in video quality, variations in lightning, distance from the camera to the subject and camera angle

III) LITERATURE SURVEY:

3.1 PAPER1:

Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison

Link: <https://arxiv.org/abs/1910.11006>

Based on a new word level american sign language(WLASL). There are 2 different models that have been used:

- (i) holistic visual appearance based approach
- (ii) 2D human pose based approach.

Proposed : novel pose-based temporal graph convolutional networks (Pose-TGCN)

Pose-TGCN :to boost the accuracy of pose-based methods the authors have used spatial and temporal dependencies in human pose trajectories.

Dataset and baseline deep models : <https://dxli94.github.io/WLASL/>

Sign language can be classified into 2 types:

1. Word Level (Isolated SL translation)
2. Sentence Level (Continuous SL Translation)

3.1.1 Challenges:

1. The exact spatio-temporal signs must be captured so that different body movement combinations can be used to establish a relation among successive words..
2. The vocabulary of the english language is in a tune of a million words whereas there are very few gestures- in tune of hundreds. This imposes this huge challenge of scalability.
3. A gesture in sign language can have multiple meanings in regular english vocabulary. The gestures for “wish” and “hungry” are the same. They must be contemplated based on the situation

3.1.2 Methods(brief):

(i)For *appearance-based methods*, The authors provided a baseline by retraining the classic attention model with VGG-net for extracting features from the images and GRU for predicting the words. A 3D convolutional networks have been used which can be observed to be better than VGG-GRU baseline.

(ii)For *pose-based methods*, They use human poses extracted from original videos as input features. They use GRU to model the temporal movements of the poses as a baseline.

3.1.3 Dataset:

Temporal boundary: The start and end frames of a sign are indicated by a temporal boundary. The boundaries are labelled as the first and last frames of the signs whenever the videos will not include sign repetitions. Otherwise, the repetition boundaries are manually

labelled.

Body Bounding-box: The authors have used YOLOv3 as an individual detection tool to isolate and create bounding boxes around the body of signers in videos in order to minimise the disturbances and irregularities caused by backgrounds and let models concentrate on the signers.

Signer Diversity: The model must be immune to signer diversity which include color , gender and other inherent bias.

3.1.4 Methods:

- Image-appearance based Baselines

- a. **2D Convolution with Recurrent Neural Networks** : 2D RNNs are used to capture long-term temporal dependencies among inputs, while CNNs are used to extract spatial features from input images. A CNN and an RNN were used to create the first baseline, which captured spatio-temporal features from input video frames.

- b. **3D Convolutional Networks** : In a hierarchical fashion, 3D convolutional networks can create both holistic representation of each frame and can also capture the temporal relations between various frames. The author claims that this method captures spatiotemporal features better than 2D CNN with RNN.

- Pose-based Baselines

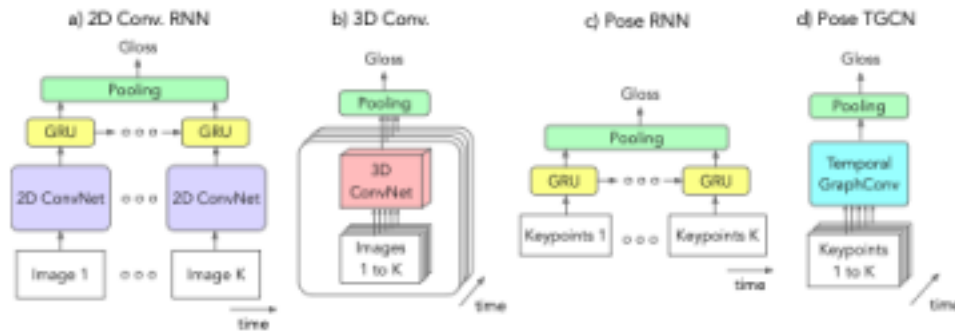
a. **Pose based Recurrent Neural Networks** : RNN is used to model the temporal and sequential details of the pose movements in this pose-based baseline, and the output of the RNN is used for sign recognition. Using OpenPose, the authors extract 55 body and hand 2D key points from a WLASL frame. As described in, these key points include 13 upper-body joints and 21 joints for both the left and right hands. Then, as an input function, they concatenate all of the 2D coordinates of each joint and feed it to a two-layer stacked GRU.

b. **Pose Based Temporal Graph Neural Networks**: The authors use Temporal Graph Convolution Networks to implement a novel pose-based approach for sign language recognition (TGCN). They suggest a graph network-based architecture for modelling the pose sequence's spatial and temporal

dependencies. The temporal motion is encoded as a holistic representation of the various body keypoint trajectories captured from OpenPose, unlike previous work where typically models motions using 2D joint angles. They view the human body as a completely connected graph with K vertices and describe the edges in the graph as a weighted adjacency matrix, inspired by recent work on human pose forecasting.

Paper: H.-k. Chiu, E. Adeli, B. Wang, D.-A. Huang, and J. C. Niebles.

Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1423–1432. IEEE, 2019.



3.1.5 Training and Testing:

3.1.5.1 Data Preprocessing and Augmentation: The authors have resized the resolution of all original video frames so that the person bounding-box has a diagonal dimension of 256 pixels. The various models (VGG-GRU, Pose-GRU, Pose-TGCN) are trained by randomly selecting 50 frames from each video and hence the models predict labels based only on the partial observations of the input video.

3.1.5.2 Implementation details: In PyTorch, the models VGG-GRU, Pose-GRU, Pose-TGCN, and I3D were implemented. For I3D, pre-trained weights were used. The training phase included the use of Adam Optimizer or SGD. On each subset, all models are trained with 200 epochs. When the validation accuracy stops improving, they interrupt the training process. The ratio in which training:validation:testing is 4:1:1.

3.1.5.3 Evaluation Metric: They assess the models by averaging the top-K classification accuracy scores ($K = 1, 5, 10$) across all sign instances.

3.2 PAPER 2:

Neural Sign Language Translation based on Human Keypoint Estimation Link:

<https://arxiv.org/abs/1811.11436>

3.2.1 KETI Sign Language Dataset:

The KETI dataset was created to help hearing-impaired people understand Korean sign language in a variety of emergency situations. They carefully reviewed cases of relatively general emergency conversations and selected 105 sentences and 419 terms that can be applied in a variety of emergency situations. All sign videos have been annotated with gloss sequences, where a gloss is a single word that refers to a unit sign and is used to transcribe sign language. For example, a sign that says, "I am burning," can be annotated with the following glosses: ('FIRE', 'SCAR'). Similarly, ('Home', 'FIRE') annotates the sentence 'A house is on fire.' Glosses seem to be a better choice for annotating a symbol because they can be represented in a variety of natural sentences or terms of the same context.

3.2.2 Our Approach:

The authors suggest a sign recognition framework based on pre-existing libraries such as OpenPose's estimation of human keypoints. They're using OpenPose, an open source toolkit for real-time multi-person keypoint detection, to build their framework. OpenPose estimates 137 keypoints in total, with 25 key points from body pose, 42 key points from both hands, and 70 key points from the human face. The main reason for using OpenPose as a function extractor for sign language recognition is that it is resistant to a wide range of changes.

To understand a signer's sign language, they use the approximate coordinates of 124 keypoints, with 12 key points from the human body, 21 key points from each hand, and 70 key points from the face.

First, since it only recognises the human body, the recognition device should be reliable in

a variety of cluttered environments. Second, since the variation of extracted keypoints is small, the method based on human keypoint detection works well regardless of the diversity of the signer. Furthermore, we use the function normalisation technique to further reduce the signer-dependent variance.

After extracting human key-points the vector is normalized with mean and standard deviation in order to limit the variance observed in the data. This normalisation aids in the uniformity of dataset inputs while also reducing training time.

The key challenge with training neural networks with limited datasets is that the trained models do not generalise well to data from validation and test sets. We use random frame skip sampling, which is widely used to process video data such as video classification, to complement training data because the dataset is even smaller than the usual cases in our issue.

3.2.3 Attention-based Encoder-Decoder Network:

Since it successfully replaces statistical machine translation methods, the encoder-decoder paradigm based on RNN architectures such as LSTMs or GRUs is being increasingly used for NMT.

An encoder RNN plays the following function given an input sentence $x = (x_1, x_2, \dots, x_{Tx})$: $h_t = \text{RNN}(x_t, h_{t-1})$

at time t , where h_t is a secret state. The encoder produces a fixed-size background vector after processing the entire input sentence, which represents the sequence as follows:

$$c = q(h_1, h_2, \dots, h_{Tx}),$$

Assume that $y = (y_1, y_2, \dots, y_{T_y})$ is an output sentence that corresponds to the training set's input sentence x . The decoder RNN is then equipped to predict the next word based on all of the previously predicted words as well as the encoder RNN's context vector. To put it another way, the decoder calculates the probability of translation y by decomposing the joint probability into the ordered conditional probabilities as follows:

$$p(y) = p(y_1 | \{y_1, y_2, \dots, y_{i-1}\}, c).$$

$$i=1$$

The probabilities for the RNN's are computed as follows

$$p(y_i | y_1, y_2, \dots, y_{i-1}, c) = \text{softmax}(g(s_i)),$$

where s_i is the decoder RNN's hidden state at time i and g is a linear transformation that produces a vocabulary-sized vector.

$$s_i = \text{RNN}(y_{i-1}, s_{i-1}, c),$$

where y_{i-1} is the previously predicted letter, s_{i-1} is the decoder RNN's last hidden state, and c is the encoder RNN's context vector.

Different attention models mentioned are:

- Bahdanau attention.
- Luong attention.
- Multi-head attention (Transformer)

3.2.4 Accuracy:

Attention type	Validation Set				Test Set			
	ROUGE-L	METEOR	BLEU	CIDEr	ROUGE-L	METEOR	BLEU	CIDEr
Vanilla seq2seq [51]	90.03	62.66	87.79	3.838	62.93	38.03	50.80	2.129
Bahdanau et al. [2]	94.72	67.44	94.03	4.264	71.45	44.06	63.38	2.616
Luong et al. [37]	96.63	72.24	95.86	4.322	73.61	46.52	65.26	2.794
Transformer [53]	94.14	69.27	92.90	4.227	73.18	47.03	66.58	2.857

Table 3. Performance comparison of sign language translation on different types of attention mechanisms.

3.2.5 Conclusions:

Sentence-level vs Gloss-level training:

Annotation	Validation Set				Test Set			
	Accuracy	ROUGE-L	METEOR	BLEU	Accuracy	ROUGE-L	METEOR	BLEU
Sentence-level	82.07	94.42	67.35	90.57	45.56	66.10	41.09	57.37
Gloss-level	93.28	96.03	71.04	93.85	55.28	63.53	38.10	52.63

Table 4. Comparison of sign language translation performance on different types of annotations.

Different Model Architectures:

Feature type	Validation Set				Test Set			
	ROUGE-L	METEOR	BLEU	CIDEr	ROUGE-L	METEOR	BLEU	CIDEr
VGGNet-16 [48]	66.85	41.92	56.75	2.369	44.75	25.79	27.88	1.016
VGGNet-19 [48]	61.77	38.72	50.95	2.060	42.75	24.81	24.27	0.839
ResNet-50 [20]	62.26	38.79	51.99	2.124	38.76	21.85	19.45	0.664
ResNet-101 [20]	66.28	41.81	56.26	2.368	40.10	22.68	21.86	0.772
ResNet-152 [20]	74.10	48.03	66.73	2.841	38.44	22.71	20.78	0.753
OpenPose [5, 47, 56]	96.92	72.14	96.11	4.380	73.95	46.66	64.79	2.832

Effect of feature normalization methods:

Method	ROUGE-L	METEOR	BLEU	CIDEr
Feature Normalization	66.28	40.94	56.91	2.401
2D Normalization	72.05	44.98	62.69	2.706
Object Normalization	64.16	38.84	53.83	2.235
Object 2D Normalization	73.61	46.52	65.26	2.794

Table 6. Effect of different feature normalization methods on the translation performance. The results are obtained on the test set.

Effect of augmentation factor:

Augmentation factor	Validation Set				Test Set			
	ROUGE-L	METEOR	BLEU	CIDEr	ROUGE-L	METEOR	BLEU	CIDEr
100	96.63	72.24	95.86	4.322	73.61	46.52	65.26	2.794
50	96.92	72.14	96.11	4.380	73.95	46.66	64.79	2.832
10	95.69	70.14	94.46	4.227	71.40	45.10	62.95	2.662

Effect of the number of sampled frames:

Number of frames	Validation Set				Test Set			
	ROUGE-L	METEOR	BLEU	CIDEr	ROUGE-L	METEOR	BLEU	CIDEr
50	96.63	72.24	95.86	4.322	73.61	46.52	65.26	2.794
40	96.52	72.36	95.96	4.327	72.71	46.18	64.35	2.757
30	95.88	70.60	94.87	4.281	73.35	46.46	64.48	2.778
20	94.38	68.40	92.98	4.181	72.37	45.37	62.19	2.693
10	83.26	55.81	78.43	3.427	65.89	40.65	54.01	2.308

Effect of batch size:

Batch size	Validation Set				Test Set			
	ROUGE-L	METEOR	BLEU	CIDEr	ROUGE-L	METEOR	BLEU	CIDEr
128	96.63	72.24	95.86	4.322	73.61	46.52	65.26	2.794
64	96.94	73.20	96.35	4.332	72.93	46.06	63.91	2.725
32	95.65	70.68	94.57	4.231	71.94	45.30	62.71	2.673
16	93.74	67.58	92.74	4.118	70.63	43.86	61.94	2.571

3.3 PAPER 3

The expertise in a wide range of fields like natural language Processing, computer vision, human computer interaction, computer graphics, deaf culture and linguistics is required for developing successful sign language language recognition, generation and translation systems. An interdisciplinary workshop was conducted to review the current state-of-the art technology, to understand the current pressing challenges that are often overlooked by computer scientists. This paper presents the results of that workshop. The 3 main questions answered by this paper are:-

- 1) What are the biggest challenges that we face in this field
- 2) What immediate actions(calls to action) must people working in the field act upon?
- 3) What does the interdisciplinary view of the current problem reveal?

1) We focus on the current Datasets, Recognition & Computer Vision, Modelling & Natural Processing, Avatars & Computer Graphics, UI/UX Design as a part of the current Landscape and the challenges to be faced

2) A few calls to action that we need to keep in mind while implementing the project are :-

- a) Deaf members should be involved throughout :- Involvement of Deaf people in such projects is necessary since this is mainly targeted to help them and the people who respect the language and communicate with them.
- b) Real-world applications :- Since technology has limitations, only specific domains are appropriate for Sign language processing. To serve real-world use cases, and account for real-world constraints algorithms, interfaces datasets and overall systems should be built.
- c) Develop UI guidelines :- We lack a systematic understanding of how people interact with it because sign language processing is still developing. Error metrics and Guidelines would support the creation of consistently effective interfaces for effective system design.
- d) Create representative, larger, public video datasets :- Big datasets consisting of

diverse signers are essential for training software to perform well for diverse users. Ensuring that datasets are publicly available is important for spurring developments, and for ensuring that the Deaf community has equal ownership.

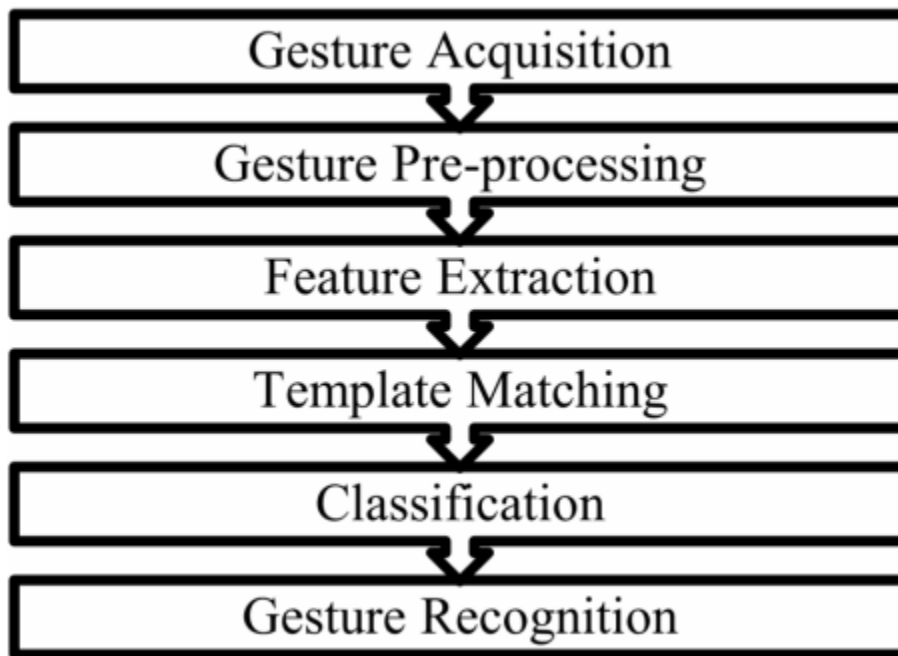
e) Develop software for annotation support and standardize the annotation system :- Annotations are essential to training recognition systems, providing inputs to NLP and MT software, and generating signing avatars. Data sharing, expanding software compatibility, and helping control quality will ensure Standardization . Accuracy, reliability, and cost would be improved upon annotation

3.4 PAPER 4

Indian Sign Language Translator Using Gesture Recognition Algorithm

Block diagram of Gesture recognition system includes:

- 1) Acquisition :- To capture gestures , sensor devices such as data gloves or motion tracker or markers are used. To capture based on vision , cameras are used.
- 2)Pre-processing :- This part is mainly used to extract the gestural features from the whole set of features present in the whole video. The gestural features are extracted and the others are left out
- 3) Feature Extraction :- All the features collected in the previous step are stored in a code vector.
- 4) Template Matching:- In this block, the code vector is compared with the existing codebook vectors in the database.
- 5) Classification:- A classification of gestures will be done, based on the output of template matching,. This block will classify the gesture as per the nearest match found in template matching.
- 6) Gesture Recognition:-The gestures will be recognised completely and the appropriate will be generated



Implementation of the Algorithm

It includes Data Acquisition, Pre-Processing, Feature Extraction, Template Matching and Gesture Recognition. To create a database, the input videos are firstly converted into frames which are pre-processed to get the enhanced features. These are then extracted and saved in a codebook. This code vector is matched with the existing reference codebook and gesture is recognized. After trying variations of multiple algorithms for Pre-processing, Feature extraction and vector quantization, the best performing algorithm has to be shortlisted and an algorithm has to be generated upon that and an output has to be sorted out.

3.5 PARER 5

NEURAL SIGN LANGUAGE TRANSLATION

This paper starts with an introduction to Sign Language Recognition and explains it as a naive gesture recognition problem. SLT is much more than that, it tries to understand various grammatical, linguistic patterns of sign language that differ from native language. Neural Sign Language Translation uses sequence-to-sequence based deep learning techniques, spatio-temporal representation of signs, relation between them and learns how these signs relate to spoken and written language. For that activity new vision techniques, which copy the tokenization and embedding steps of Neural Machine Translation are introduced. The paper suggests to use CNN with attention based encoder, decoder models to model required outcome. Target sequences are projected into continuous space using word embeddings. Sign videos are encoded using spatial embeddings using 2D CNNs. Talking about tokenization on input both frame level and gloss level tokenizations are used with gloss

level using RNN-HMM forced alignment approach. The output is tokenized at word level.

Attention Based Encoder -Decoder Networks

- * Various generic tokenization schemes are used on spatio temporal embeddings of input
- * During encoding phase the feature vectors are fed to encoder one by one in reverse. This models temporal changes in video frames and compress their cumulative representation into hidden states.
- * The final result produced is the latent embedding of the sequence, which is passed to the decoder
- * The decoding phase which started by initializing of h_{sign} , which in classic encoder would be the only information source is took along with previous hidden state, word embedding and previously predicted word to predict next word and update the hidden state
- * But using the model above can't handle long term dependencies so Attention The mechanism of Bahdanau Lu-ong et al is used.

Dataset: RWTH-PHOENIX-Weather 2014T

Build: All encoder decoder networks are built using four stacked layers of residual recurrent units, with each layer containing 1000 hidden units. AlexNet as a spatial embedding layer, weights initialized based on training on ImageNet Sign language of America which contains 26 hand signs, one delete, one space sign which are Translated using CNNs.

Convolution neural networks: CNN learns input as a series of sequential numbers. CNN has many layers. Layers are usually convolutional, pooling and fully-connected layers. The convolution layer consists of a kernel / filter with a designated size that slides or converts, over and over again and in short pixels, finally extracted from a small, simple matrix. The filter moves over all the pixels in the image to form a new matrix called the feature matrix. This new matrix highlights the most important features.

Pooling Layer: The next layer is a pooling layer. Pooling layer passes a pooling core over the feature matrix which either takes the highest valued pixel (max-pooling) or the average of it (average pooling). This layer helps to further reduce the size of the matrix.

Fully Connected Layer: Fully connected Layer helps to classify the images and it's the final layer. The matrix produced from previous layers is first vectorized, further sent through a neural net. The Neural net is similar to an ANN which passes the vector through it by applying the weights, applying the bias and finally resulting in classification.

Unit Type:	DEV SET					TEST SET				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
LSTM	41.69	41.54	27.90	20.66	16.40	41.92	41.22	28.03	20.77	16.58
GRU	43.85	43.71	30.49	23.15	18.78	43.73	43.43	30.73	23.36	18.75

Table 3. G2T: Attention Mechanism Experiments.

Attention:	DEV SET					TEST SET				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
None	40.32	40.45	27.19	20.28	16.29	40.71	40.66	27.48	20.40	16.34
Bahdanau	42.93	42.93	29.71	22.43	17.99	42.61	42.76	29.55	22.00	17.40
Luong	43.85	43.71	30.49	23.15	18.78	43.73	43.43	30.73	23.36	18.75

Table 4. G2T: Batch Size Experiments.

BS:	DEV SET					TEST SET				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
128	43.85	43.71	30.49	23.15	18.78	43.73	43.43	30.73	23.36	18.75
64	43.78	43.52	30.56	23.36	18.95	44.36	44.33	31.34	23.74	19.06
32	44.63	44.67	31.44	24.08	19.58	44.52	44.51	31.29	23.76	19.14
16	44.87	44.10	31.16	23.89	19.52	44.37	43.96	31.11	23.66	19.01
1	46.02	44.40	31.83	24.61	20.16	45.45	44.13	31.47	23.89	19.26

3.6 PAPER 6

Translating Sign Language In Real Time With AI

Sign language of America which consists of 26 hand signs and one delete and one space sign are Translated using CNNs .

Convolution neural networks: CNNs read input as a series of numbers arranged in an array. A CNN consists of multiple layers. The layers are usually convolutional, pooling and fully-connected layers. The convolution layer consists of a kernel/filter with a designated size that slides or convolves over the pixels multiplying and summing values ,finally outputting into a smaller , simpler matrix. The filter travels over every pixel in the photo creating a new matrix called feature matrix. This new smaller matrix highlights the most important features.

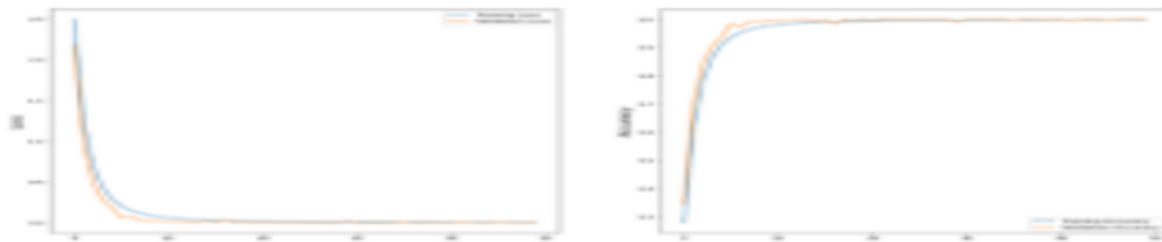
Pooling Layer: The next layer is a pooling layer. The pooling layer further reduces the matrix size. It passes a pooling kernel over the feature matrix and takes either the highest pixel value(max-pooling) or the average(average-pooling).

Fully Connected Layer: Here's where the classification happens. The matrix is first flattened into a vector and then passed through a neural net. The neural net it passes through is similar to an Artificial Neural Net in that it passes the vector through, applying weights and biases finally ending up with a classification.

The CNN classifies the image by using a softmax activation function which gives the probability the input is from a certain class.

After convolution, it's time for the fully-connected layers. But before that happens, the data is flattened into a single column vector.

1. Now that the data can pass through the neural net, a dense layer is used. A dense layer is pretty much the neural net. It passes the input from the previous steps an outputs all of it to it's neurons. The neurons are connected and pass data from one tothe next layer. In this case, there are 256 neurons.
2. Then, it goes through batch normalization.
3. Then a ReLU function is used for activation.
4. Finally, 25% of the nodes are once again dropped out using dropout.
5. There are two fully connected layers, so the code repeats with 512 nodes.
6. Once past the second fully connected layer, the output is put through a softmax function which is used to give the probability the image belongs to one of the 29.



IV) PROJECT REQUIREMENT SPECIFICATION

4.1 Introduction

This document presents the Project Requirement Specification of the Indian Sign Language Translation Project that would be implemented in coming months. In this document we would present various requirements of the project starting with Scope followed by the perspective, features, Constraints, assumptions and dependencies, risks,

Operating environment on which the project would run. We will continue specifying the Functional requirements, External Interface Requirements and the Non-Functional Requirements. We will finish with Appendixes.

4.1.1 Project Scope

To build a Indian Sign Language Translator i.e to convert the signs shown by a person in a video to a meaningful caption. This translator is highly beneficial for the not just the deaf and dumb people, but also for the people trying to communicate with them. The model must be robust to variations in video quality, variations in lightning, distance from the camera to the subject and camera angle etc. The goal of this project is to provide(coverage) a means of communication for deaf and dumb people living in India to communicate with anyone in ISL. This Application provides support for only one sign Language (i.e ISL).

4.2 Product Perspective

This project started with the idea of providing a Standard of communication for the specially abled people living in India (and a standard language (i.e ISL) too because the signs used in the country vary from region to region. To make this systemised, several attempts are being made to make a standardised language i.e Indian Sign Language).

4.2.1 Product Features

The two major features of the product are:-

- 1) Converting a video of a person showing Sign Gestures to a sentence.
- 2) Converting a sentence into a video of a person showing Sign Gestures

4.2.2 Operating Environment

This Product would be hosted on Android Operating System, on any mobile device supporting the OS.

4.2.3 General Constraints, Assumptions and Dependencies

The Constraints for the project are:-

- 1) Generation and Augmentation of data :- There are a large number of variations of ISL across India, and robust translation will require accounting for these variations. Hence there are almost no publicly available datasets for ISL. So, data collection for variations should happen too or a standardised ISL must be set up.
- 2) Segmentation:- There is no demarcated boundary when transitioning from one sign to the next. Several signs are combinations of other signs and these will have to be taken care of as well.
- 3) Grammar :- English Grammar rules are more stringent than ISL rules of grammar. The same set of signs can be permuted in any way, resulting in the sentence having the same meaning.
- 4) Speed and Variations :- Just like every other language, different sign language users speak at different speeds. There is also a large variation from region to region as well. These should be accounted for.
- 5) Regulatory Policies :- Video Data might be generated with faces which were used for video captioning. Hence it is necessary to avoid that circumstance.

Assumption :-

- 1) We assume that there is no external disturbance in the videos that are being captured.

Dependency :-

For both the above mentioned functionalities we implement the finish-to-start dependency (wherein the predecessor task must be finished before another

task, the successor, can start.)since the video has to be processed after capturing it and only then the output can be generated.

4.2.4 Risks

We are currently trying to concatenate frames of each video that are generated using 3 preprocessing methods(i.e cropped image, human pose estimation output and finger keypoint detection frames). We are still unsure about the accuracy this method would give out on passing it to an CNN - LSTM network.

4.3 Functional Requirements

Validity tests on inputs: The basic constraints on the input are that there must be a human in front of the camera. He must do the gestures before the camera.

Sequence of operations: Firstly a video is processed as a set of images since it is divided as frames. The individual frames are preprocessed in 3 ways. Firstly they are cropped up to the human part , Secondly we use human pose estimation and lastly we use finger keypoint detection. Then these inputs are together concatenated and then sent to the CNN - LSTM network which then determine what word is predicted. Since sign language have no vocabulary we take all the words we receive in the form of an array and then design another context based LSTM vector which takes paragraphs as input. This then predicts the best possible meaning from the given words and the context.

Error handling and recovery:

Consequences of parameters : All the parameters and weights are adjusted as they are in the system. They are updated in a way to achieve maximum accuracy. We have weights updated through backpropagation. If needed we can allot more weights to preprocessing

layers if it gives more accuracy.

Relationship of outputs to inputs: The output must be a sentence based on the context whereas input must be a video. There must be a unique output for every unique input.

4.4. External Interface Requirements

4.4.1 User Interfaces

The screen must be such that it must fit to the average mobile screen. There must be a basic signup and login page and once a user logs in to his account. Then the application asks permission to use the camera and then when the camera is faced and a button on the screen is pressed the visual input must be simultaneously translated and displayed to the user on his screen. There must also be a help function that must be able to help all users on how to operate the app and also another option to report issues so that they can be resolved in subsequent versions. The complete input is taken and processed and then the output is returned. An error must be displayed in case there are issues like the video is running too long or the user is holding the camera in the wrong direction or the hands are not visible.

4.4.2 Hardware Requirements

Databases must be needed to store the user data in the backend since only registered users can access the features in the application. There must be a phone that is connected to a camera for live input stream. Minimal RAM is needed for processing the requests. Most requests are made through HTTP, FTP etc

4.4.3 Software Requirements

Python , Tensorflow , 2.4.1 , Colab and Windows are used as interface , pandas , numpy , Pytorch, OS module , Open CV . etc.

4.4.4 Communication Interfaces

NIC Card , Internet

4.5 Non-Functional Requirements

4.5.1 Performance Requirement

To make app available all the time , reliable and have multiple points of failure we are planning to host the app in AWS . With the help of AWS various quality attributes are met like

Reliability: By monitoring various KPIs which reflect business value automatic triggers are activated when the threshold is breached . This automation helps to anticipate failures before they occur.

Robustness: Having multiple points of failures , Horizontal scaling of resources and automated monitoring of resources helps the app to be robust and make it available all the time

Application usability is made easy as it just contains three options of help , camera recording , record history of user translations.

4.5.2 Safety Requirements

Identity Access Control of AWS helps in safe management of resources and control over user Access

4.5.3 Security Requirements

All the communication between app and server is encrypted and is done through HTTPS protocol. Various security schemes of aws like security engineering, encryption and protection of data, centralized logging and monitoring helps in safe and secure deployment of application , transmission of user data.

V) HIGH LEVEL DESIGN DOCUMENT

5.1 Introduction

This is a High Level Design Document for the Project “Indian Sign Language Translation”. This is documented as a requirement for Review - 3 as a part of our Capstone Project Phase - 1. This document is based on the requirements specified in the Project Requirement Specifications document. This document would help the reader understand the Current Systems (Existing systems with the same functionality) , Design Considerations (Design Goals , Architecture Choices , Constraints , Assumptions and Dependencies) , High level System Design , Design Description (Master Class Diagram , reusability considerations) , State Diagrams, UI Diagrams , External Interfaces , Packaging and Deployment Diagram , help gathered from external sources and the Design Details. This would really help the reader to gain insights into the thought process and the planning that was undergone to develop the Application.

This whole idea is to build an Android App , that could capture video of any person signaling in sign language and translate it english that would enable anyone to communicate with any person with speech disability.

5.2 Current System [if applicable]

A few apps such as [Nirvatha Vadathi](#) (An application that uses Google Speech API to convert Speech to text or the other way to help communicate with challenged people) have come up with the similar idea of assisting challenged people to communicate. But none have come to public use. Our idea is that capturing vision , emotion , poses is also a very important aspect in capturing the perfect essence in the thought/ idea that the person is trying to convey.

5.3 Design Considerations

5.3.1 Design Goals

Our goal is to implement a fully automated sign language translator that can effectively translate a sequence of signs into a relevant sentence. Various designs have been considered for the project. After research and observing various state of the art benchmarks we have adopted methods that yield the highest level of accuracy. Large portion of our designs is inspired from the industry level implementations of Sign language translation for American , Korean , German Sign Languages. Our principles are to keep design a simple yet effective model for Indian sign language translation that can effectively capture all the different movements done by the signer also not compromising on the context the words have been used.

Availability : As our application process the video offline availability is achieved and all the modules are downloaded from the play store .

Privacy : As user stores his videos in his mobile and our application doesn't store video in our server privacy depends on user pc

Speed : As it is an DL application Speed depends on the user PC

processor. **5.3.2 Architecture Choices**

Our data was from the INCLUDE dataset from IIT madras. It is set of 52 files ranging from Adjectives to society. Upon detailed Literature survey we have come to the conclusion that pose estimation methods like OpenPose and dense Pose and finger keypoint detection help in the best identification of the Video. Hence we have implemented these methods. We plan on making a concatenated image of 225 X 225 X 9 as the input to the next layer. We are implementing a CNN-RNN based attention model for prediction. This model is simple and the majority of the research journals have used these. We are also exploring the upcoming field of transformers since they can give better accuracy. The major pros are that these models are reliable and fairly accurate. The cons include that the model could have some inherent assumptions.

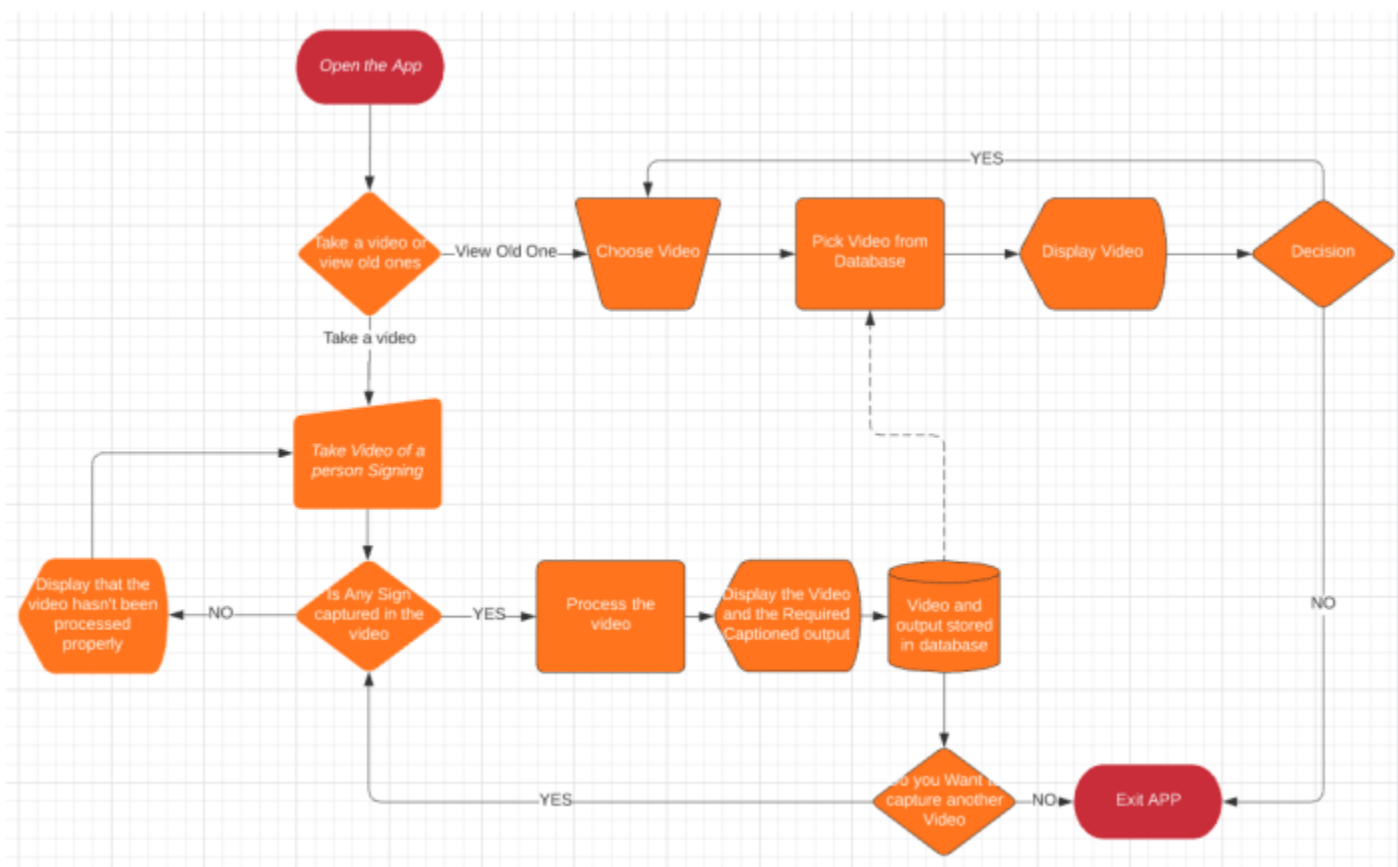
5.3.3 Constraints, Assumptions and Dependencies

The dependencies made on the dataset include signer diversity and that the information is correct. The constraints include the similar symbols for numerous words.

The communication is similar to interprocess communications seen in the various operating systems. The dependency is that the user's phone has sufficient RAM for the application to run. The user is expected to have a working camera for live video recording or real time translation. The interface used is Android. We require the user to have an android phone for application to run. We expect a performance of 70%. The main issue is that there are similar expressions for the same words. The symbols for FOOD and FIRE are very similar hence they can be similar in nature.

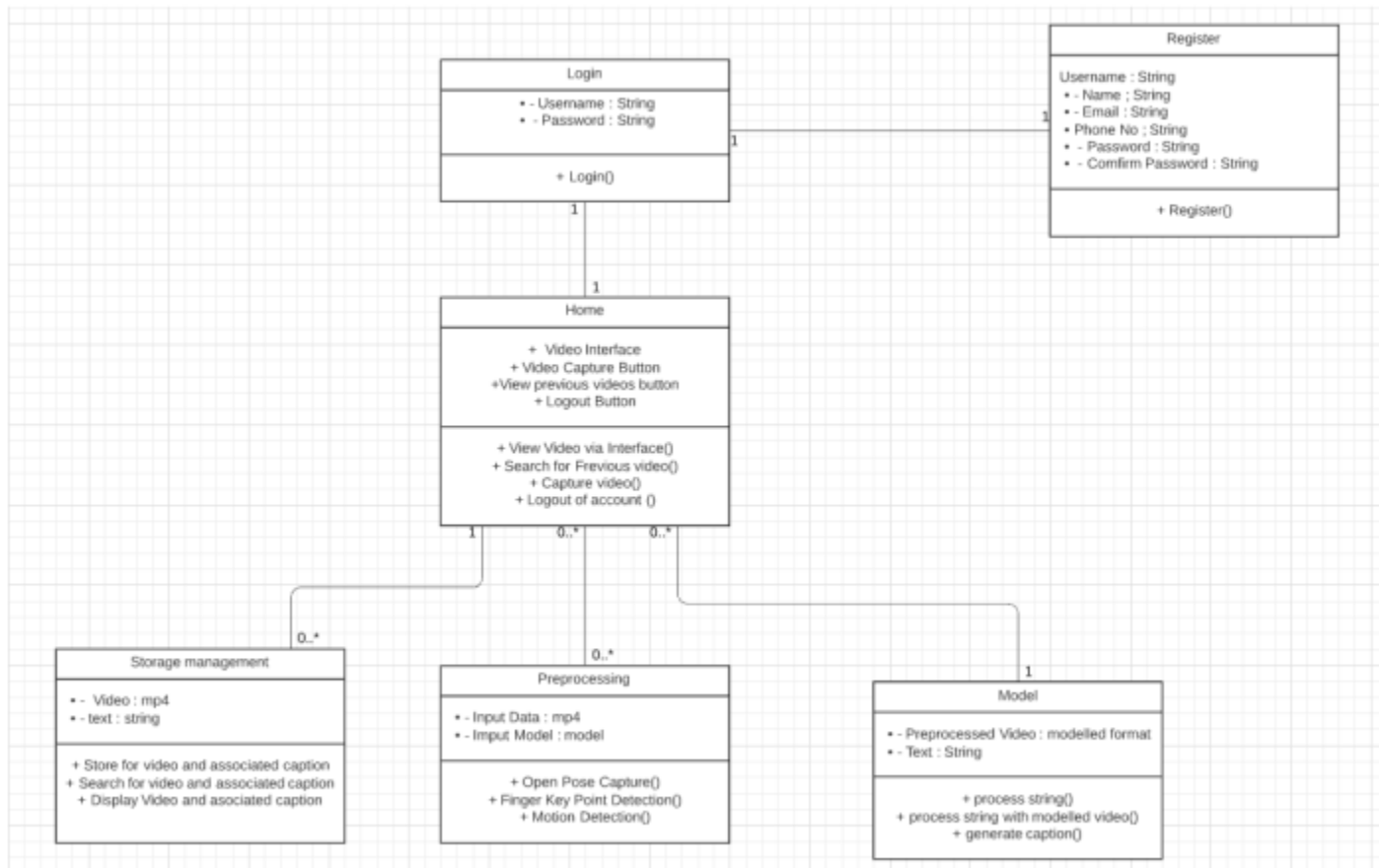
The issues of scalability and is managed in such a way as the basic packages are installed in the users end system. We also maintain availability through pre installed repositories installed. The maintainability is done through periodic updates. This constantly improves the performance of the application.

5.4. High Level System Design



5.5 . Design Description

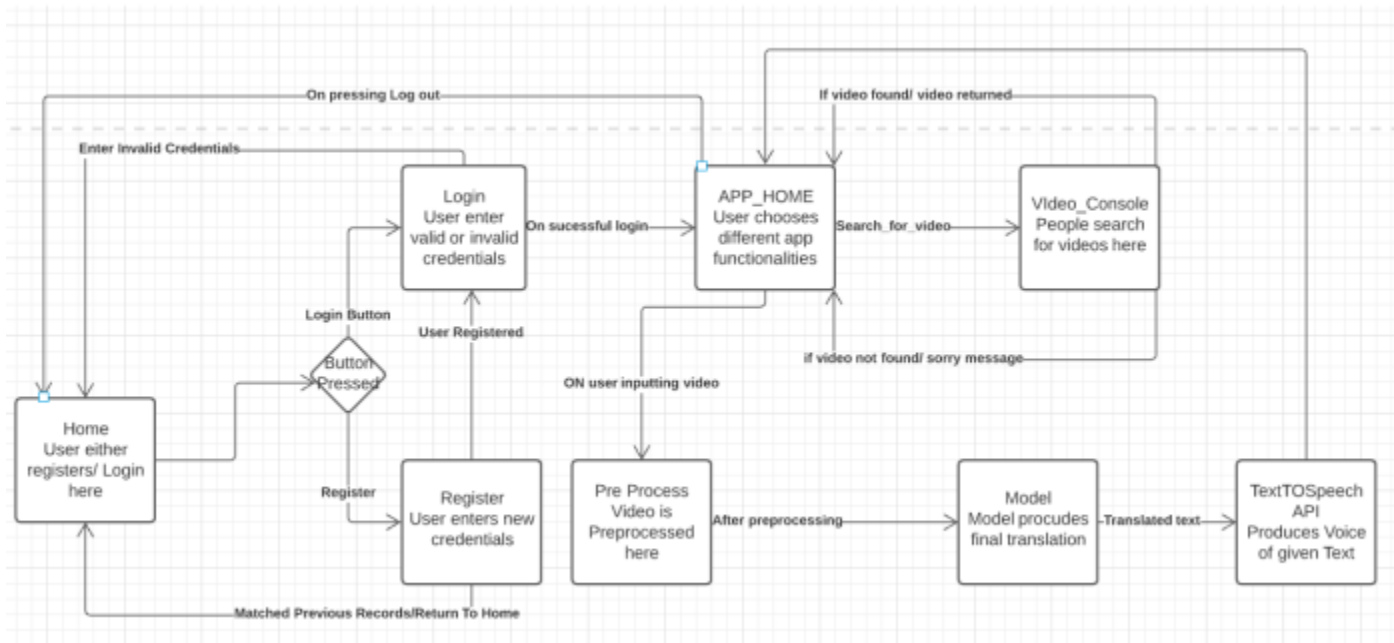
5.5.1 Master Class Diagram



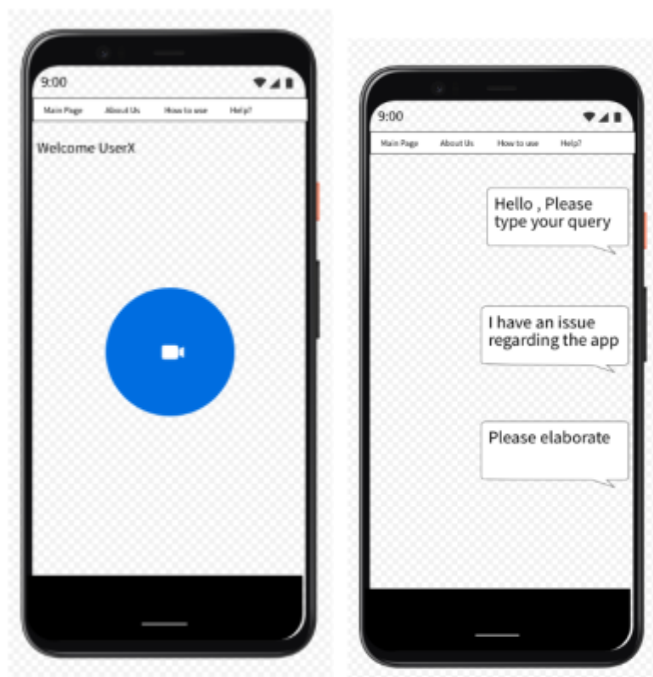
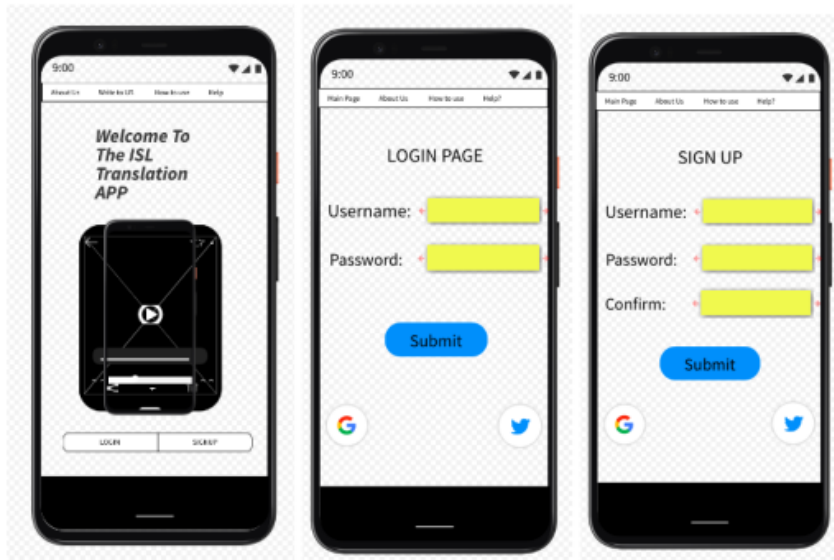
Reusability Considerations

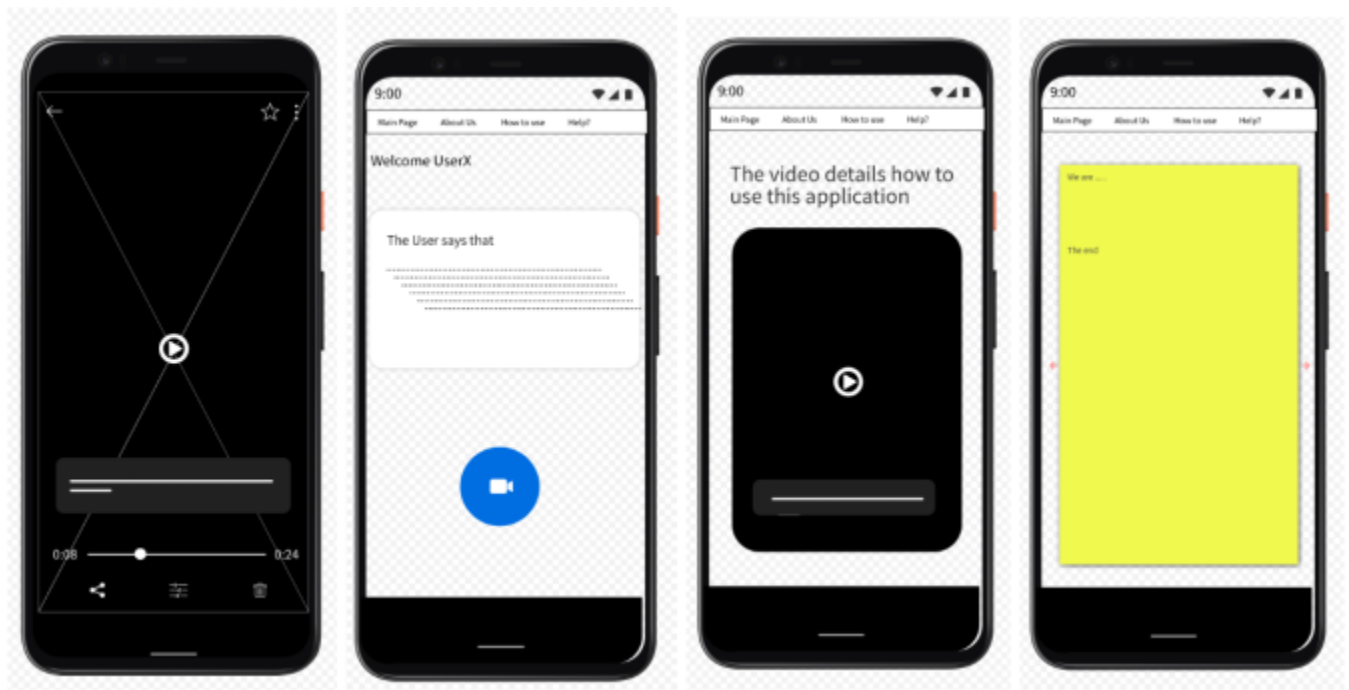
We are reusing Open Pose Detection as well as Finger key Point detection algorithm for the preprocessing steps.

5.5.2 STATE DIAGRAM

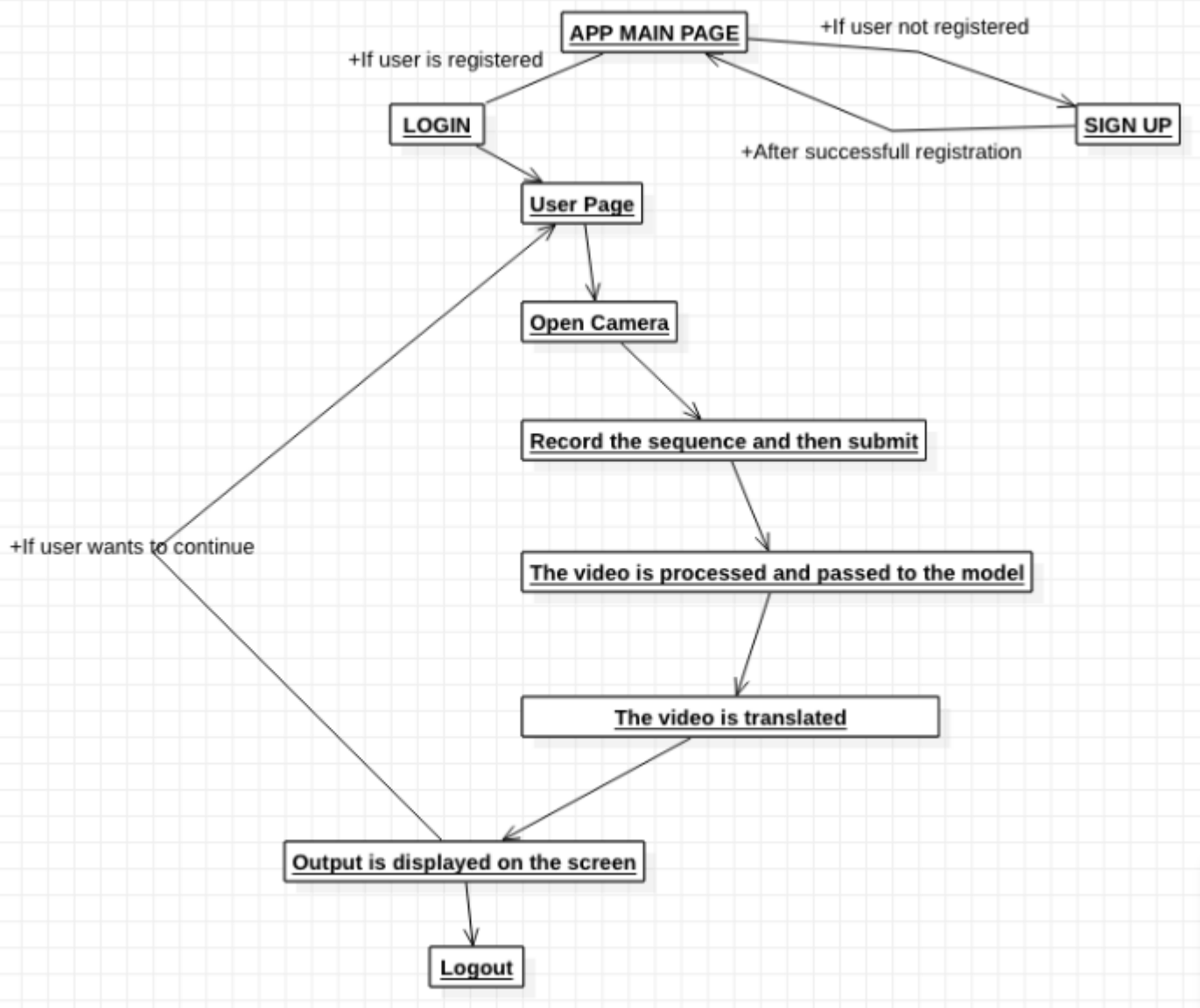


5.5.3 User Interface Diagrams

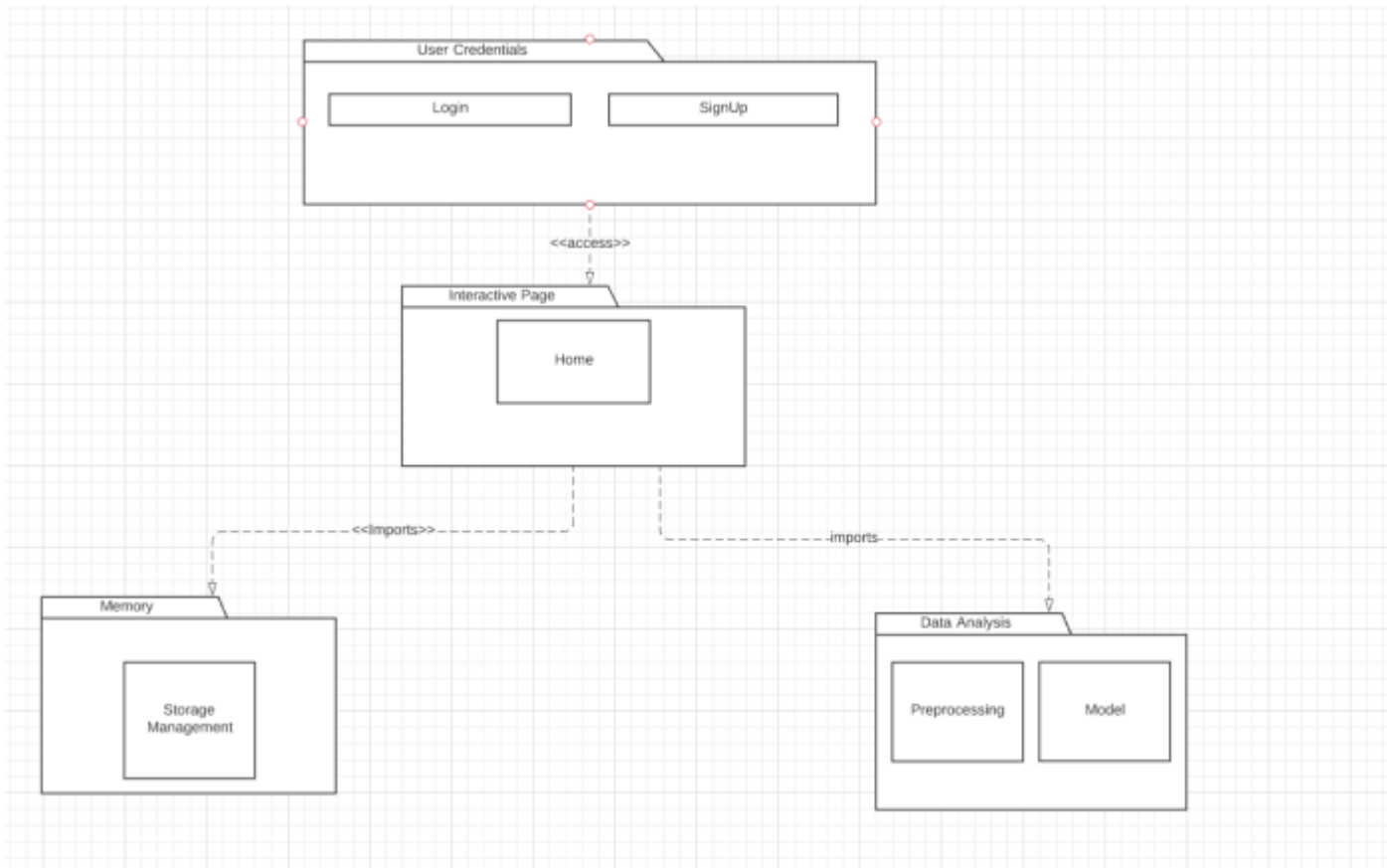




5.5.4 External Interfaces



5.5.5 Packaging and Deployment Diagram



5.6 Help

Some of the help i.e the softwares & User Manuals that would aid us in building the soft are

- 1) [Sign Language Translation from AI4 Bharat](#) - The user manual that gave us the idea of Indian Sign Language Translation.
- 2) Google Colab - It is a software that allows anybody to write and execute python code through the browser, and is well suited to data analysis, machine learning and education.
- 3) [Open Pose](#) - **OpenPose** is a real-time system to jointly detect the hand, facial, and foot (overall the entire human body)key-points of multiple people in a single image. 4) Android Studio - It is the Official IDE for Android App development.

5.7 Design Details

5.7.1 Novelty: The application that is designed is novel in nature. There is only one research paper that has been written on the Indian sign language translation. There was no such development before since there was no comprehensive dataset for ISL. Hence the application is one of its kind.

5.7.2 Innovativeness : We have used innovative and modern methods inspired from the state of the art converters from American and Korean sign language papers.

5.7.3 Interoperability: We plan on restricting the use of the device to working only on Android phones. There is no need for exchange of information apart from the initial download of modules for prediction

5.7.4 Performance:- We want to keep the performance to be above 70%. We have used various methods in preprocessing like Openpose and finger keypoint detection. We are also planning to use state of the art technology like transformers for better accuracy.

5.7.5 Maintainability :- Maintainability is the measure of how the software is maintained from time to time. It is related to the consistency, complexity

structure and size of the codebase .The code would be maintained in the GitHub Application so that different versions of the code can be maintained during the different phases of the project development lifecycle.

5.7.6 Portability :- Portability is the measure of how the same software can be used in different environments. Since we are initially planning to build an Android App , the application would run on any mobile device built on the Android Operating System.

5.7.7 Legacy to modernization :- Legacy Modernization aims at transforming existing systems into web based thin client systems and integrating multiple related systems. It is basically the process of reducing the complexity of the code. It can be done by following a simple set of steps.

- a) The cost, complexity, and the risk involved in the code must always be kept in check and must be minimised as much as possible on moving forward through various versions.
- b) The business perspective of the code must also be checked i.e all the growing business requirements must be met on time. The business fit , business value and agility requirements of the consumer of the code must be met.

5.7.8 Reusability :- Code reusability is basically the practice of using existing code in new softwares. We plan to use code reusability for functionalities that have been developed and are used extensively. We are mainly focussed on working upon codes and functionalities that are not used extensively.

5.7.9 Application compatibility :- Application compatibility is listing the versions of operating systems upon which the application being developed works. Since the camera module works from Android 5.0 (API level 21) and higher we would develop application on versions 5.0 and above

5.7.10 Resource utilization :- Resource utilization is organisation and utilization

of various resources (Resource include not just the space consumed the data, the hardware and the cost of managing these but also the time and effort of the people working for the project). On the data utilization perspective we are initially trying to capture videos and then preprocess them initially , but we will move on to real time captioning so that it will be fast, reliable and space saving. From the people's perspective, we have created a gantt chart so that the project can be planned according to schedule and the team can allocate their time accordingly.

VI) Low level System Design:

6.1 Overview

This is a Low Level Design and Implementation Document for the Project “Indian Sign Language Translation”. This is documented as a requirement for Review - 2 as a part of our Capstone Project Phase - 2. This document is based on the requirements specified in the Project Requirement Specifications document. This document would help the reader understand the Design Considerations (Design Constraints, Assumptions and Dependencies) and certain diagrams such as Use Case diagram, master class diagram (also the data members and methods of each class present in the master class diagram), Sequence diagram and Packaging Diagram. This would really help the reader to gain insights into the thought process and the planning that was undergone to develop the application.

The whole idea is to build an Android App, that could capture video of any person signalling in sign language and translate it to english captions that would enable anyone to communicate with any person with speech disability.

6.2 Purpose

The purpose of this project is to create an application that enables any person with or without speech or hearing disabilities to communicate with any other person in our country. Models and Applications have been developed for ASL(American Sign Language), but no Application has been developed to help people communicate in ISL(Indian sign Language). This application is an effort to enable people to use such an application.

6.3 Scope

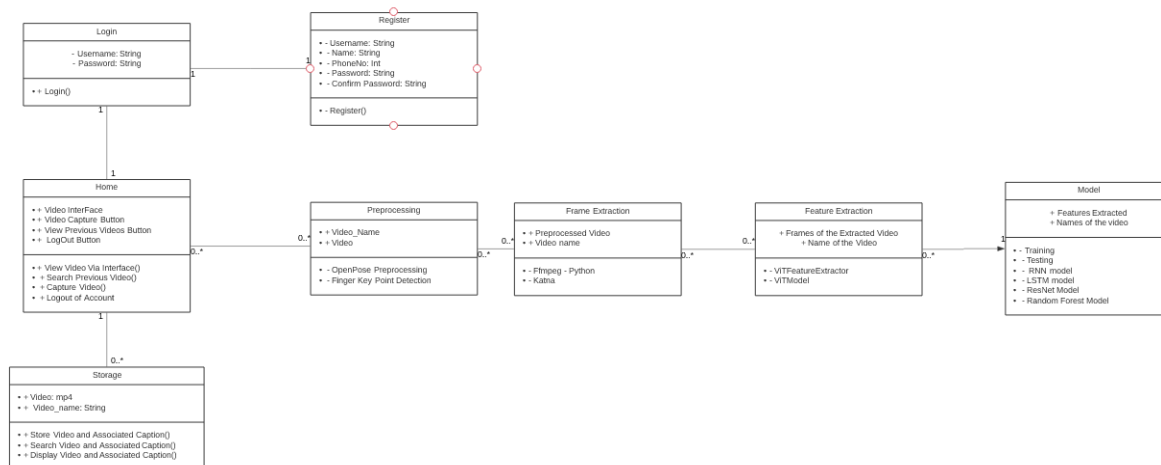
The aim of this project is to build an automatic ISL translator - i.e. given an input video of a specially abled person signing a word, the model should be able to predict which word is being signed. This translator is highly beneficial for the not just the deaf and dumb people, but also for the people trying to communicate with them. The model built should be reasonably robust to variations in camera angle, video quality, distance from the camera to the subject, variations in lighting etc. The goal of this project is to provide(coverage) a means of communication for deaf and dumb people living in India to communicate with anyone in ISL.This Application provides support for only one sign Language (i.e ISL).

The dependencies made on the dataset include signer diversity and that the information is correct. The constraints include the similar symbols for numerous words. The communication is similar to interprocess communications seen in the various operating systems. The dependency is that the user's phone has sufficient RAM for the application to run. The user is expected to have a working camera for live video recording or real time translation. The interface used is Android. We require the user to have an android phone for the application to run. We expect a performance of 70%. The main issue is that there are similar expressions for the same words. The symbols for FOOD and FIRE are very similar hence they can be similar in nature. The issues of scalability are managed in such a way as the basic packages are installed in

the users end system. We also maintain availability through pre-installed repositories installed. The maintainability is done through periodic updates. This constantly improves the performance of the application.

6.4 Design Description

6.4.1 Master Class Diagram

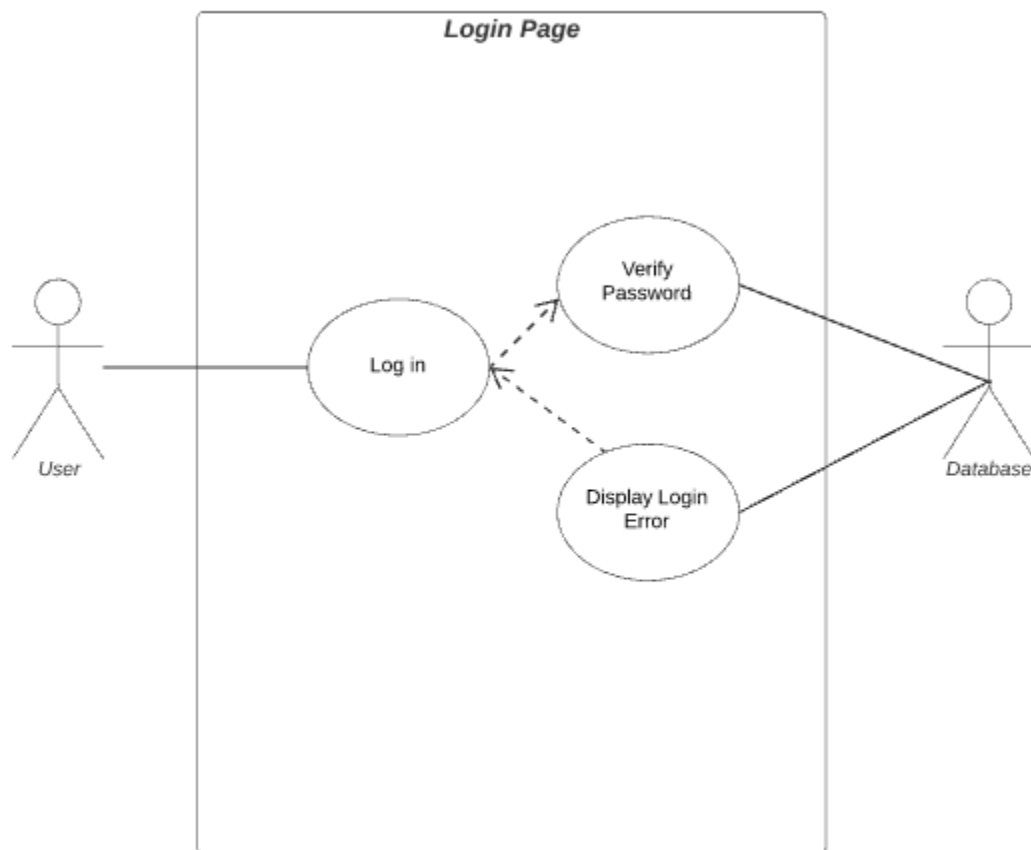


6.4.2 Login

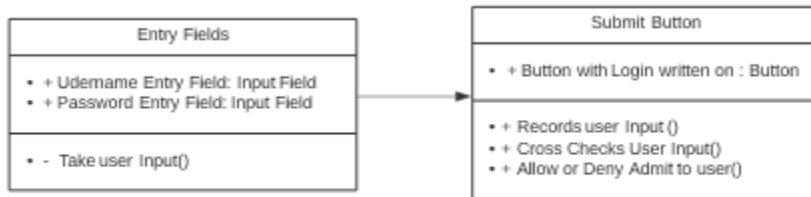
6.4.2.1 Description

This Module is used to help a user log into their account. Using this module every user can Login to their account so that they can access the videos that they recorded. This contains the username and Password Fields.

5.4.2.2 Use Case Diagram



6.4.2.3 Class Diagram



6.4.2.3.1 Entry Fields

6.4.2.3.1.1 Entry Fields - Description

The fields in which the user enters the username and password.

6.4.2.3.1.2 Entry Fields - Data members

Data Type	Data Name	Initial Value	Description
Input Field	Username Field	""	Records UserName
Input Field	Password Field	""	Records Password

6.4.2.3.1.3 Entry Fields - Function 1 - Take User Input

This take username and password of the user

6.4.2.3.2 Submit Button

6.4.2.3.2.1 Submit Button Description

This button is used to login a user into the software.

6.4.2.3.2.2 Submit Button - Data members

Data Type	Data Name	Initial Value	Description
Button	Login	NULL	Records , checks users credentials

6.4.2.3.2.3 Submit Button - Function 1 - Records User Input

This take username and password of the user

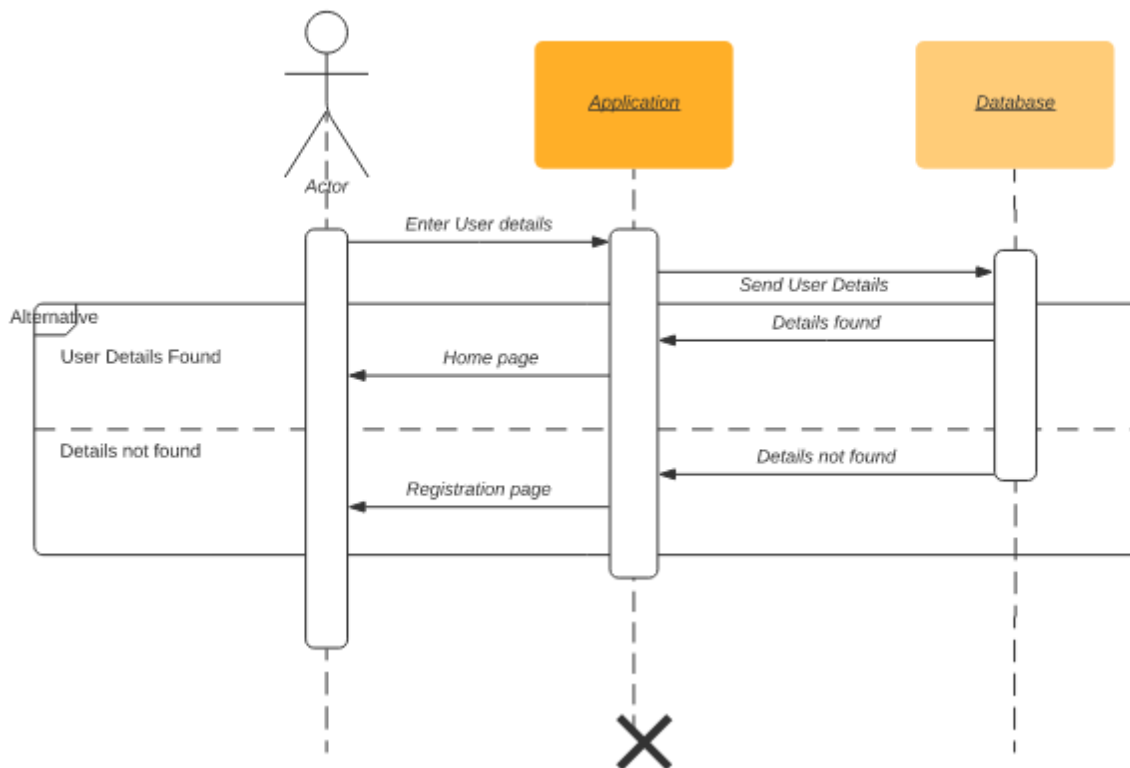
6.4.2.3.2.4 Submit Button - Function 1 - Verifies User Input

This verifies username and password of the user with the database

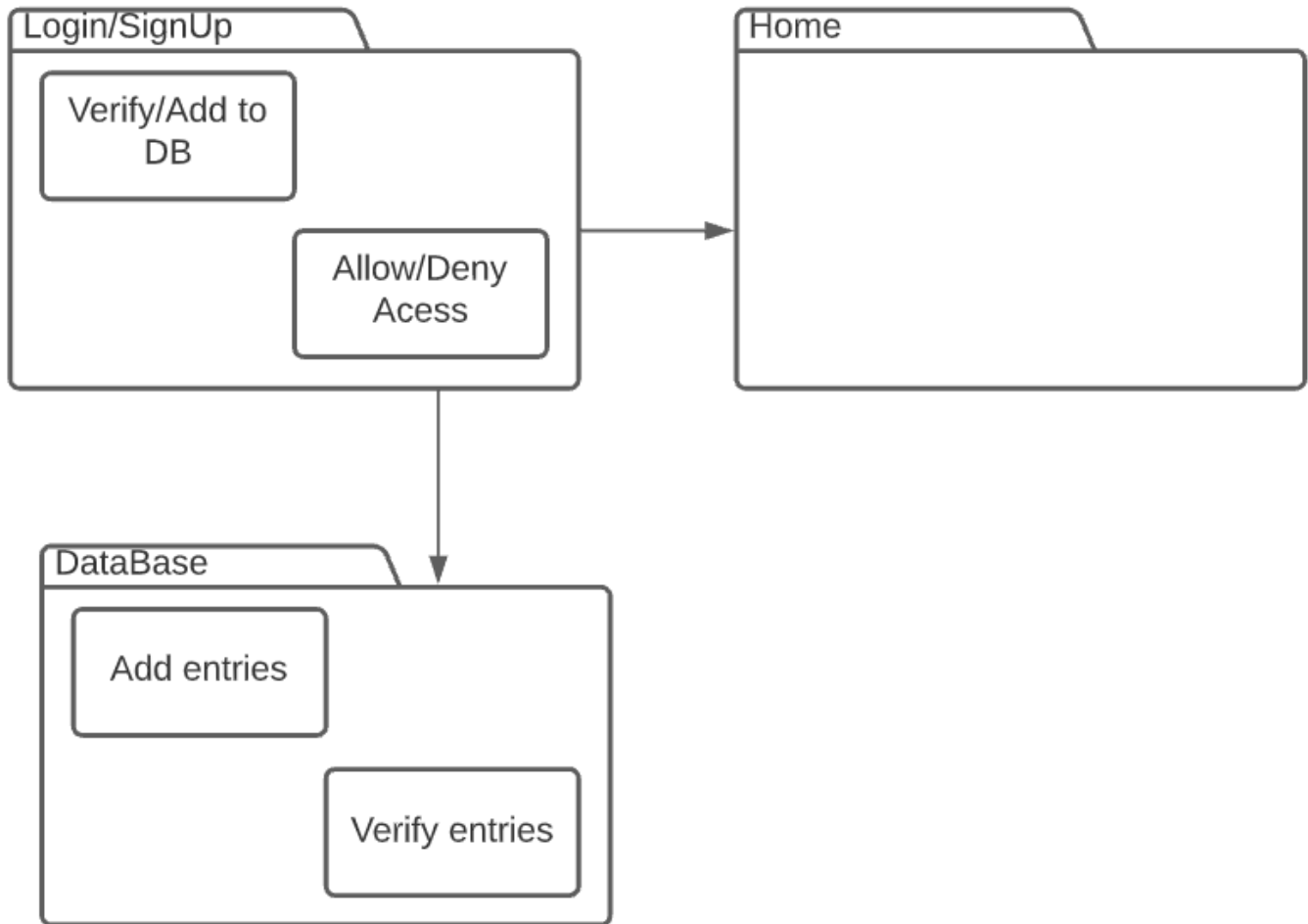
6.4.2.3.2.5 Submit Button - Function 1 - Allows or denies user

After Verification, the user is either allowed or denied.

6.4.2.4 Sequence Diagram



6.4.2.5 Packaging and Deployment Diagrams

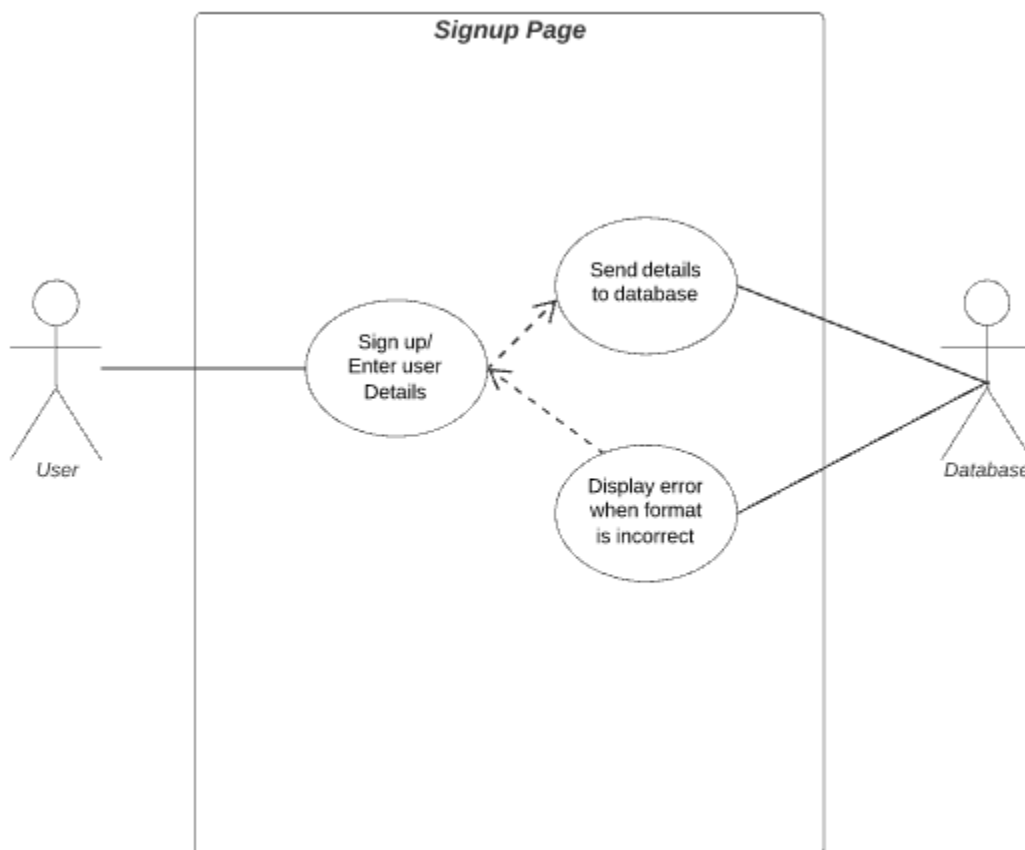


6.4.3 Signup

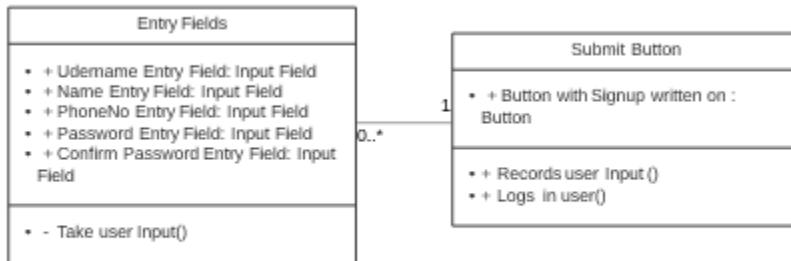
6.4.3.1 Description

This module is used to take in the credentials of new users and store it to the database, so that they can be cross verified when a user logs in.

6.4.3.2 Use Case Diagram



6.4.3.3 Class Diagram



6.4.3.3.1 Entry Fields

6.4.3.3.1.1 Entry Fields - Description

The fields in which the user enters the username, name, Phno and password

6.4.3.3.1.2 Entry Fields - Data members

Data Type	Data Name	Initial Value	Description
Input Field	Username Field	""	Records UserName
Input Field	Name Field	""	Records name
Input Field	PhoneNo Field	""	records password

Input Field	Password Field	“”	Records Password
Input Field	Confirm Password Field	“”	Records Password and checks with above password

6.4.3.3.1.3.1 Entry Fields - Function 1 - Take User Input

This takes the username, name, PhoneNo and password of the user.

6.4.3.3.2 Submit Button

6.4.3.3.2.1 Submit Button Description

This button is used to Signup a user into the software.

6.4.3.3.2.2 Submit Button - Data members

Data Type	Data Name	Initial Value	Description
Button	Signup	NULL	Records user, Logs in

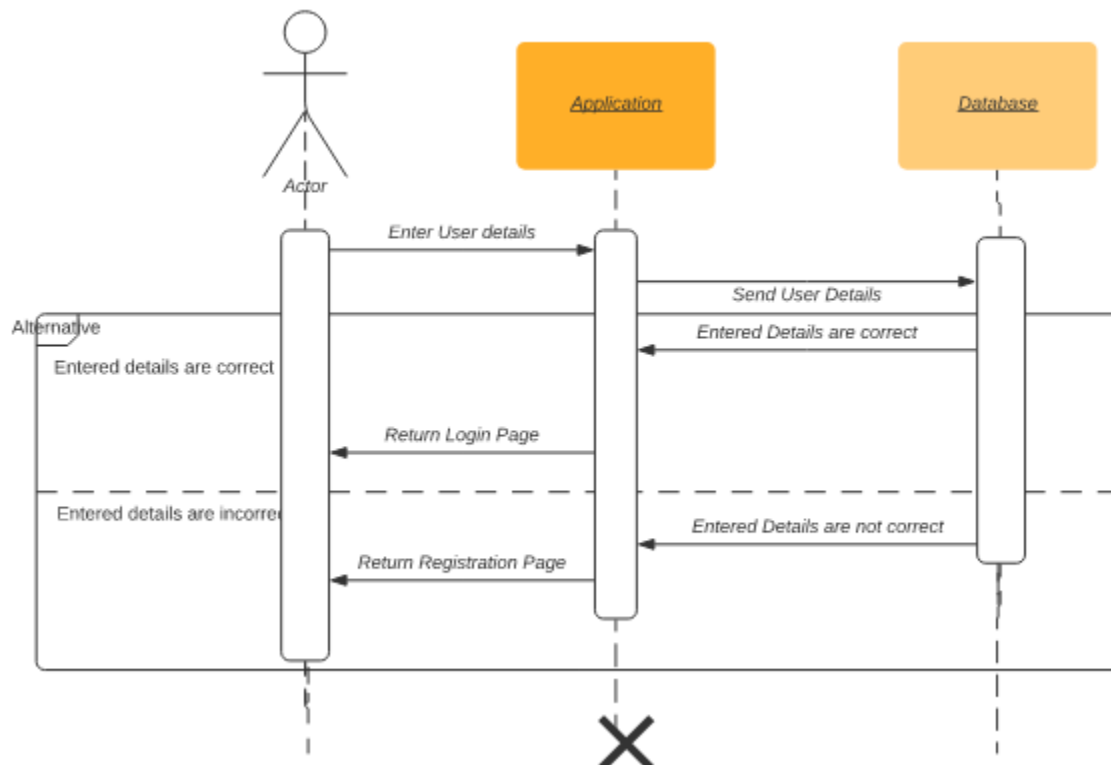
6.4.3.3.2.3.1 Submit Button - Function 1 - Records User Input

This take username and password of the user

6.4.3.3.2.3.1 Submit Button - Function 2 - Logs in user

The user is Logged in into the account

6.4.3.4 Sequence Diagram

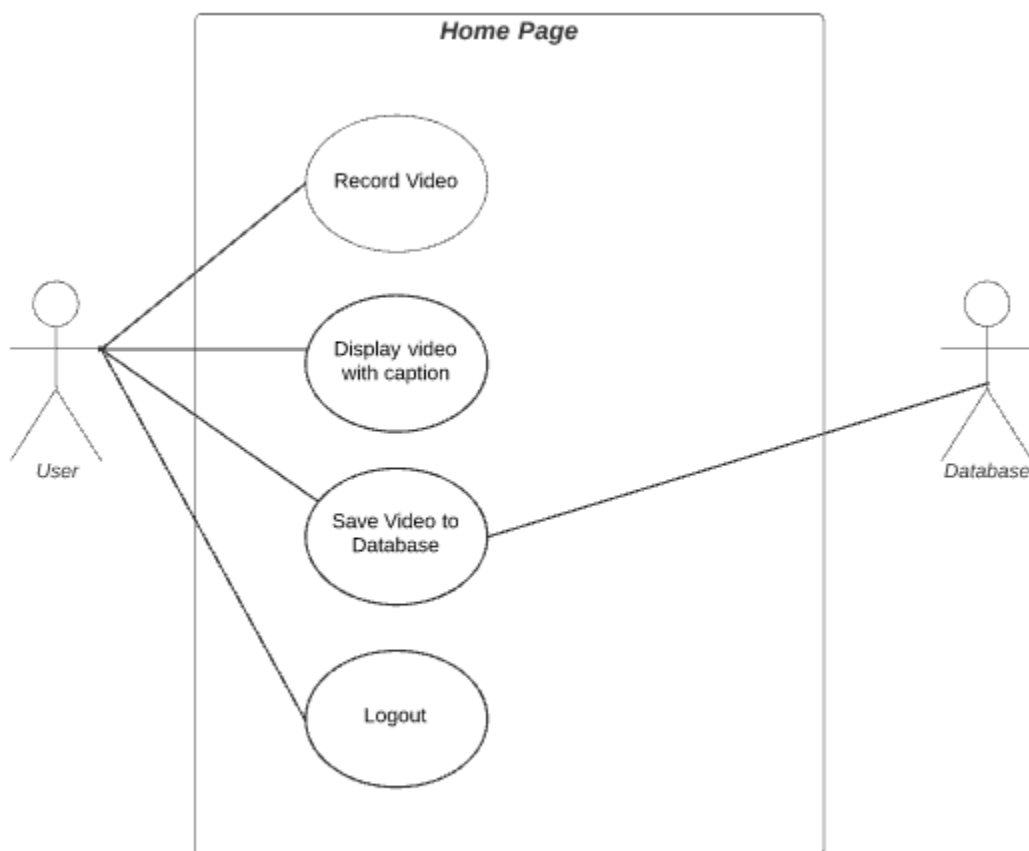


6.4.4 Home

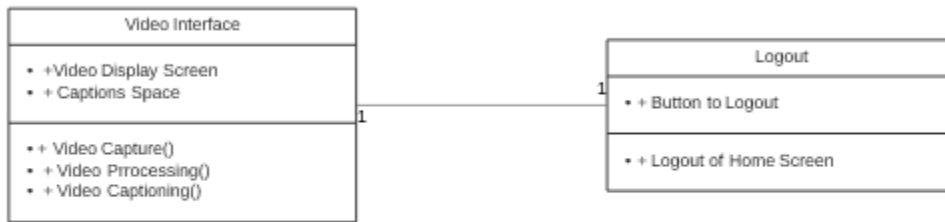
6.4.4.1 Description

This is the main page where video recording and captioning happens. A video interface and a capture button is present where the user can capture their video and

6.4.4.2 Use Case Diagram



6.4.4.3 Class Diagram



6.4.4.3.1 Video Interface

6.4.4.3.1.1 Video Interface - Description

The fields in which the user enters the username, name, Phno and password

6.4.4.3.1.2 Video Interface - Data members

Data Type	Data Name	Initial Value	Description
mp4	Video Display	NULL	Displays Video
text	Caption Space	""	Displays Captions

6.4.4.3.1.3.1 Video Interface - Function 1 - Video Capture

This captures Videos from the camera of the phone.

6.4.4.3.1.3.2 Video Interface - Function 2 - Video Processing

It processes the video using the model that we have built

6.4.4.3.1.3.3 Video Interface - Function 3 - Video Captioning

The processed caption along with the video are being displayed.

6.4.4.3.2 Logout

6.4.4.3.2.1 Logout - Description

The button that helps a user to logout of their account.

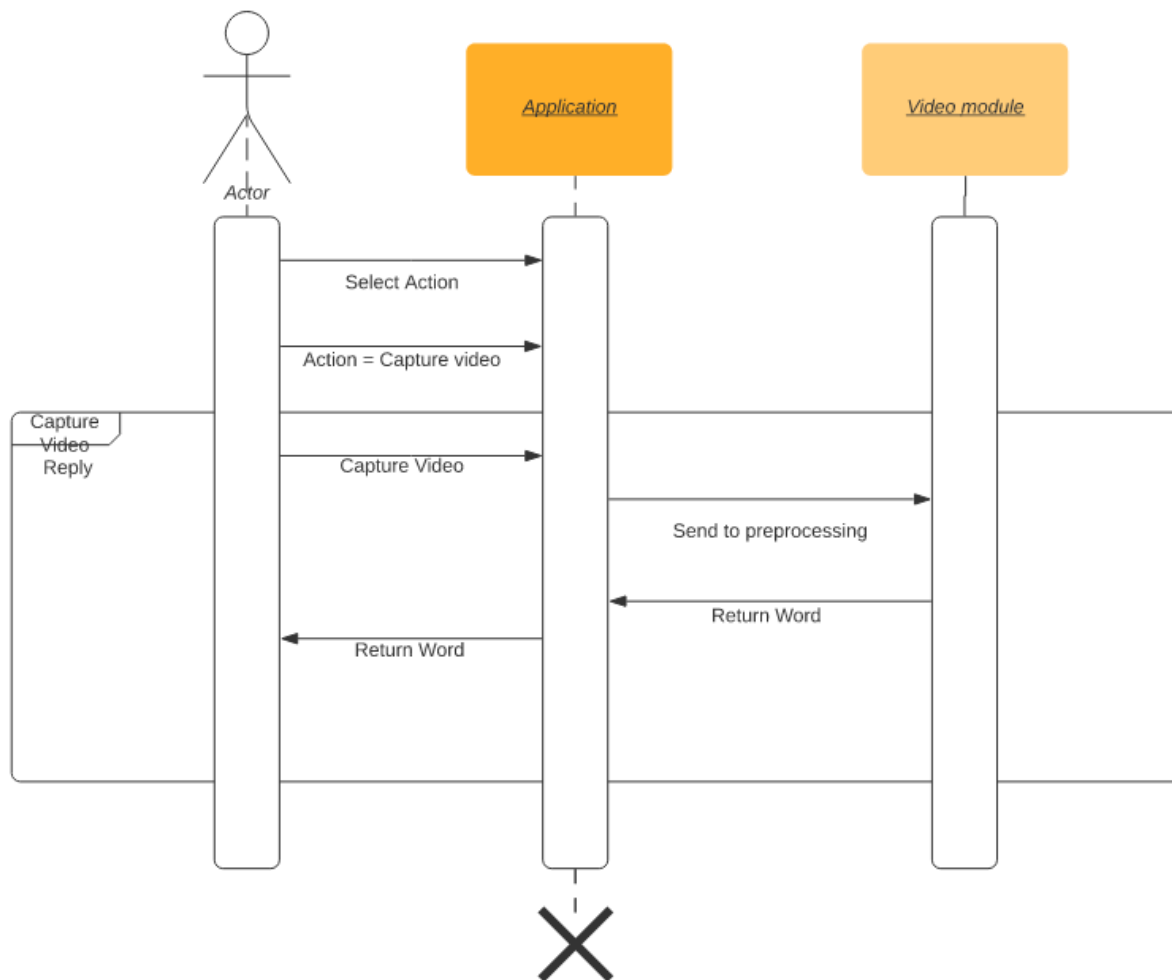
6.4.4.3.2.2 Logout - Data members

Data Type	Data Name	Initial Value	Description
Button	Logout	NULL	Logout User

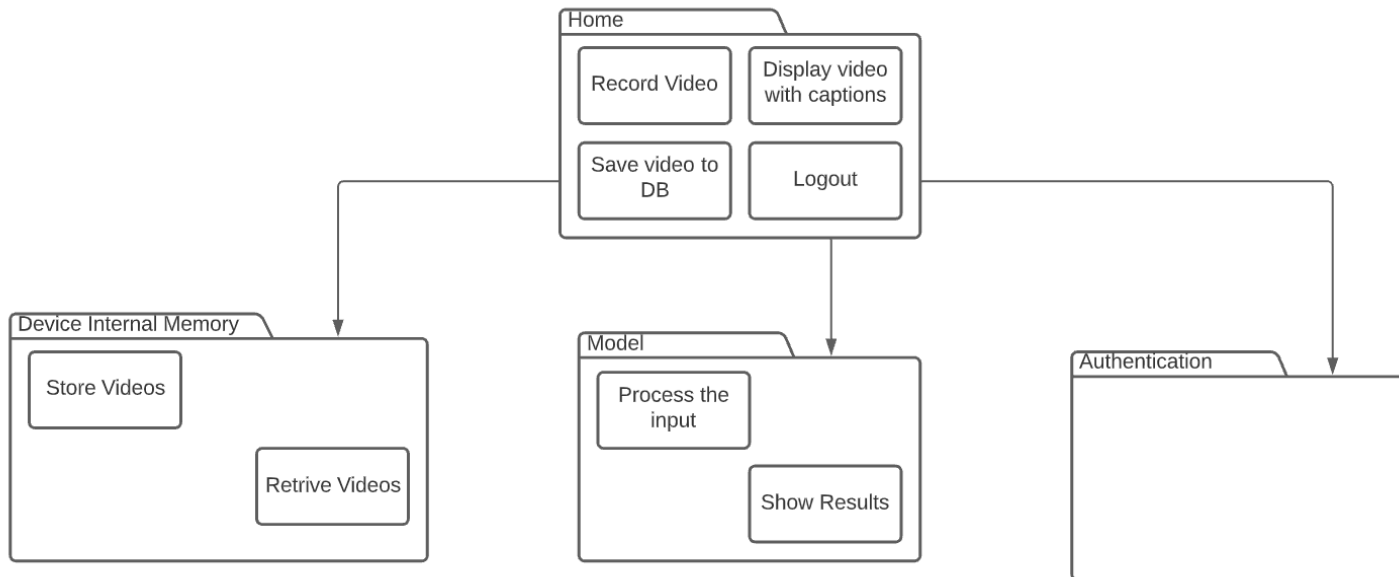
6.4.4.3.2.3.1 Logout - Function 1 - Logout of account

On clicking on the logout button the user is able to logout of their account.

6.4.4.4 Sequence Diagram



6.4.4.5 Packaging and Deployment Diagrams

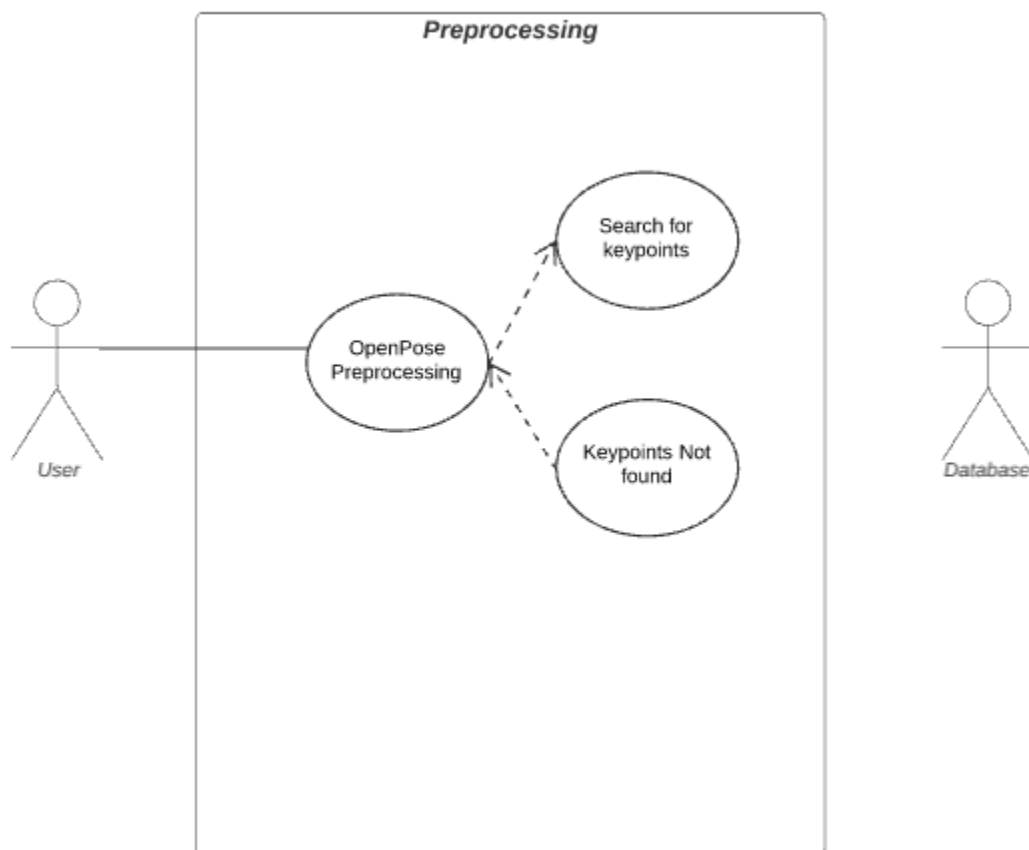


6.4.5 Preprocessing

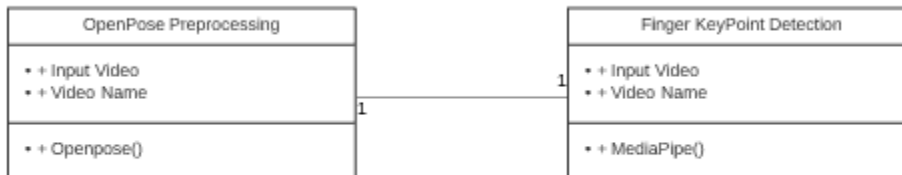
6.4.5.1 Description

In this module the video that is captured from the home screen is preprocessed and the output i.e a openpose Stick Diagram with keypoints is pushed to the next phase

6.4.5.2 Use Case Diagram



6.4.5.3 Class Diagram



6.4.5.3.1 Openpose Preprocessing

6.4.5.3.1.1 Openpose Preprocessing - Description

The preprocessing method wherein the video can be converted to video with skeletal structures where the movement of frames can be recorded.

6.4.5.3.1.2 Open Pose Preprocessing - Data members

Data Type	Data Name	Initial Value	Description
mp4	Video Input	NULL	The video that is sent for preprocessing
String	Video Name	Name of each video	Name that can be stored with preprocessed result

6.4.5.3.1.3.1 Openpose Preprocessing - Function 1 - Openpose

The main function of this module is to convert the video of a person to a video of a skeletal structure where the movement of various key points can be recorded.

6.4.5.3.2 Finger Keypoint Detection

6.4.5.3.2.1 Finger Keypoint Detection - Description

The preprocessing method wherein the fingers in a video can be converted into Finger keypoints

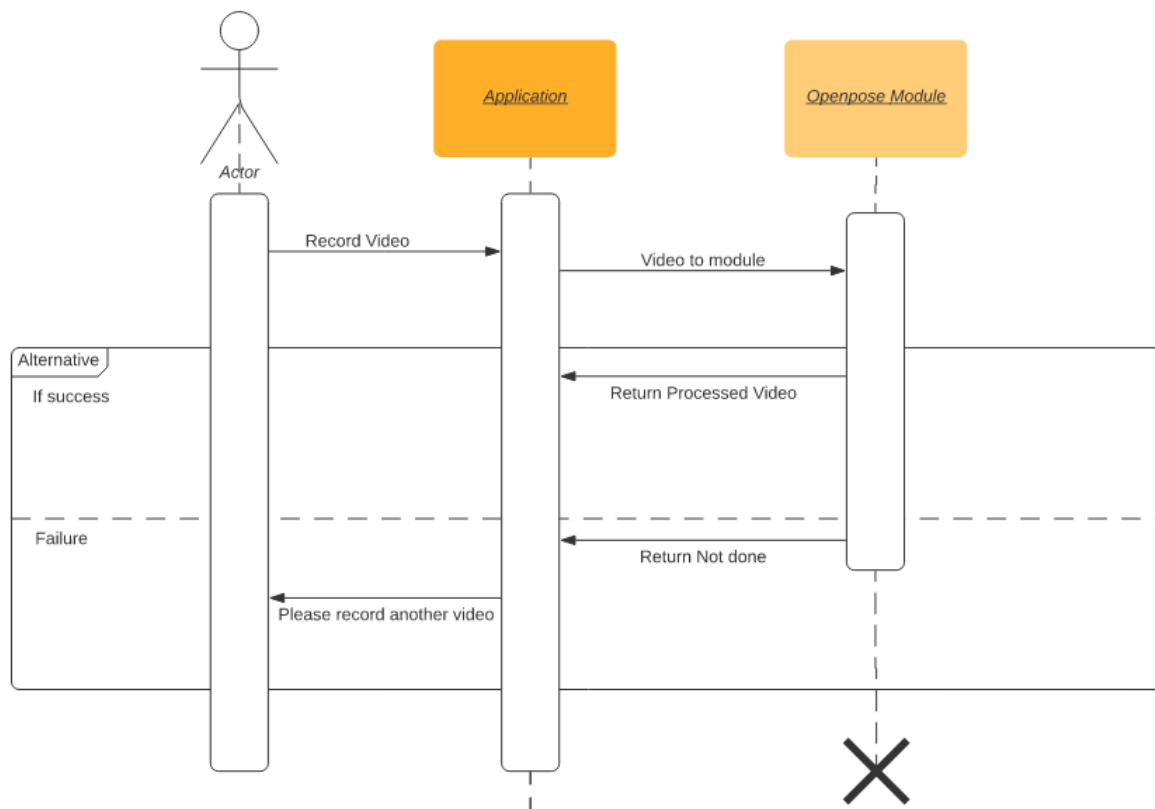
6.4.5.3.2.2 Finger Keypoint Detection - Data members

Data Type	Data Name	Initial Value	Description
mp4	Video Input	NULL	The video that is sent for preprocessing
String	Video Name	Name of each video	Name that can be stored with preprocessed result

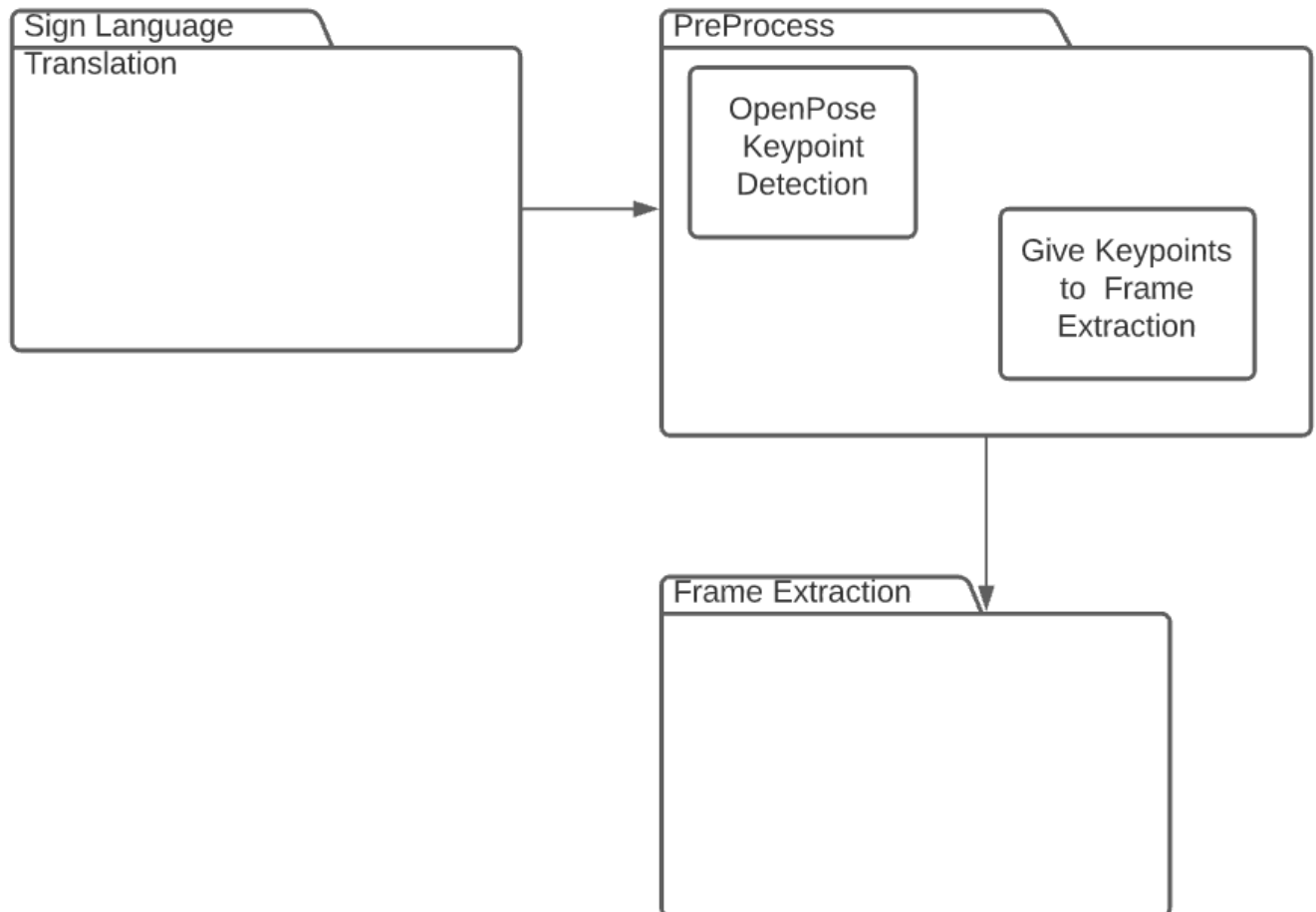
6.4.5.3.2.3.1 Finger Keypoint Detection - Function 1 - Mediapipe

The main function of this module is to convert the video of a person to a video of a skeletal structure where the movement of various keypoints can be recorded.

6.4.5.4 Sequence Diagram



6.4.5.5 Packaging and Deployment Diagrams

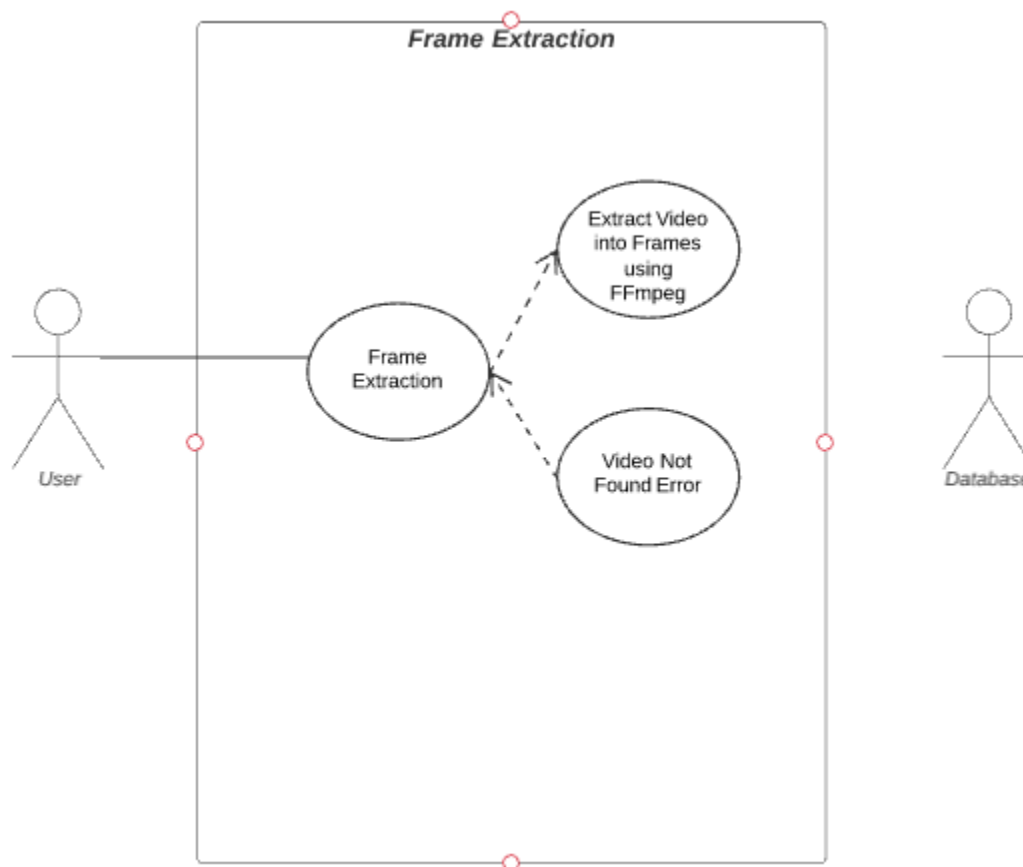


6.4.6 Frame Extraction

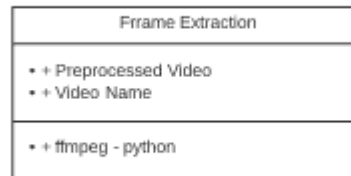
6.4.6.1 Description

The preprocesses video files along with the names are bought into this module where the video is divided into 15 frames.

6.4.6.2 Use Case Diagram



6.4.6.3 Class Diagram



6.4.6.3.1 Frame Extraction

6.4.6.3.1.1 Frame Extraction - Description

In this step the Video which was preprocessed in the previous step is taken and it is divided into 15 frames and each frame is named after the video.

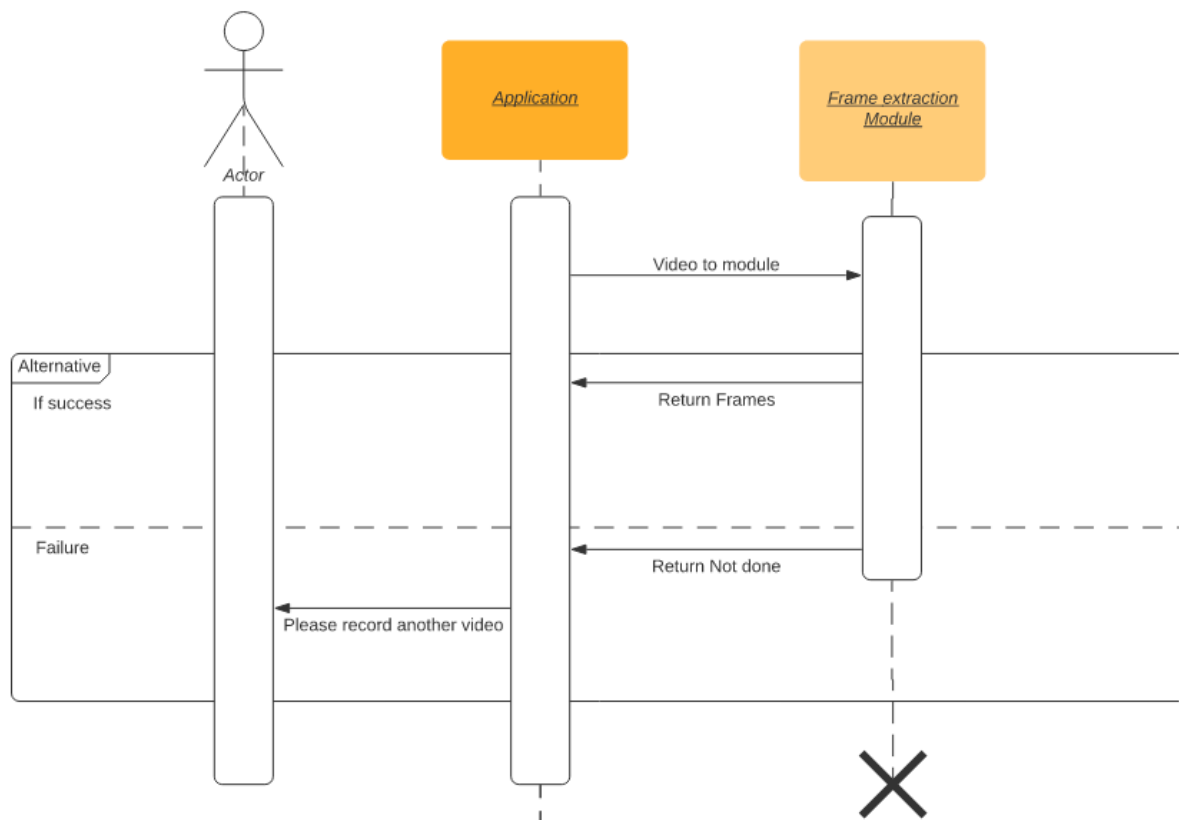
6.4.6.3.1.2 Frame Extraction - Data members

Data Type	Data Name	Initial Value	Description
mp4	Preprocessed Video	NULL	The video that is sent for preprocessing
String	Video Name	Name of each video	Name that can be stored with preprocessed result

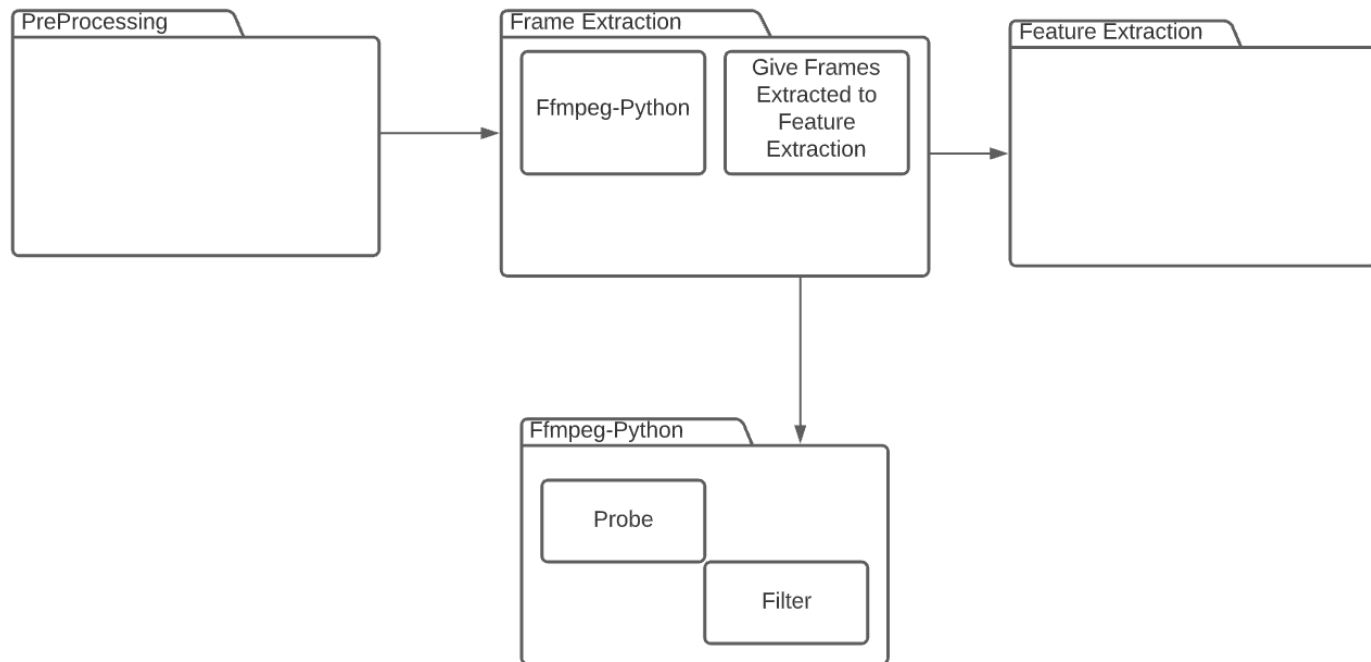
6.4.6.3.1.3.1 Frame Extraction - Function 1 - FFmpeg - python

The main function of this module is to convert the video of a person to a video of a skeletal structure where the movement of various key points can be recorded.

6.4.6.4 Sequence Diagram



6.4.6.5 Packaging and Deployment Diagrams

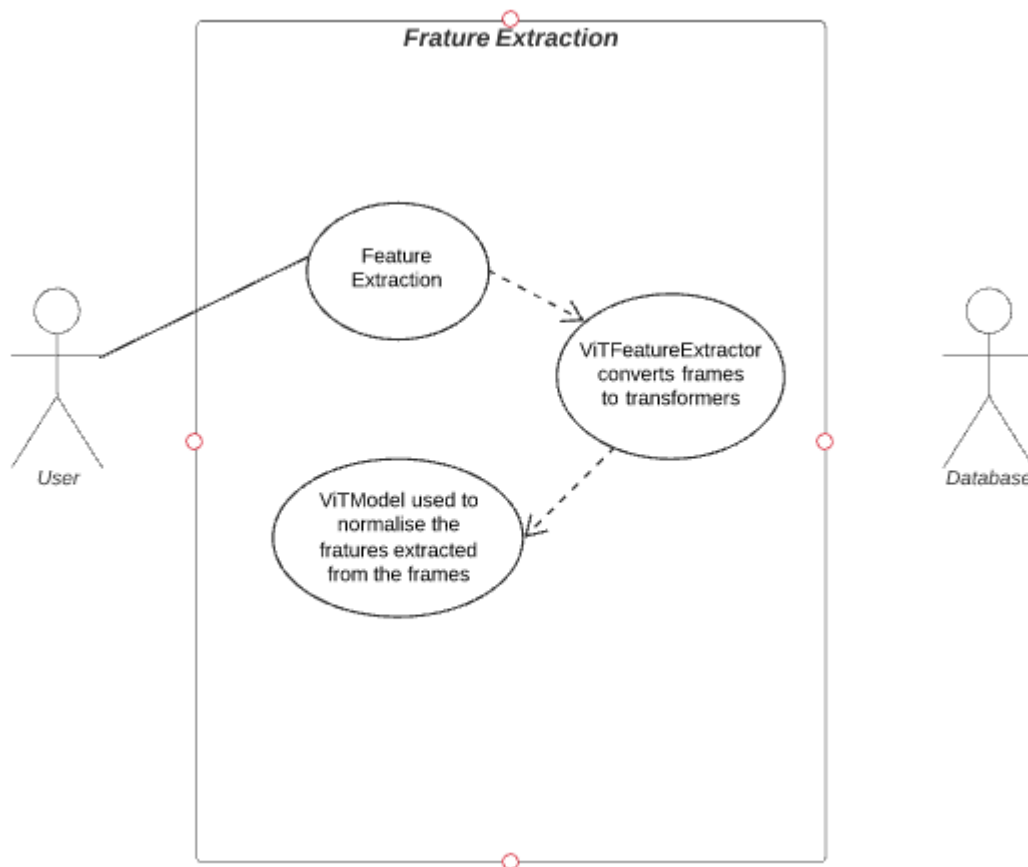


6.4.7 Feature Extraction

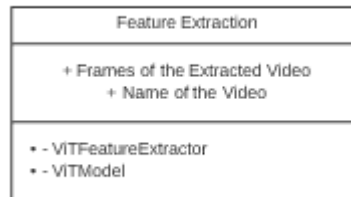
6.4.7.1 Description

The features are extracted from the frames and are saved in a dictionary with the name of the video as its key, so that classification can be done successfully.

6.4.7.2 Use Case Diagram



6.4.7.3 Class Diagram



6.4.7.3.1 Feature Extraction

6.4.7.3.1.1 Feature Extraction - Description

In this step the Video which was preprocessed in the previous step is taken and it is divided into 15 frames and each frame is named after the video.

6.4.7.3.1.2 Feature Extraction - Data members

Data Type	Data Name	Initial Value	Description
jpg	Frames of the preprocessed video	NULL	The frames extracted from the preprocessed video
String	Video Name	Name of each video	Name that can be stored with preprocessed result

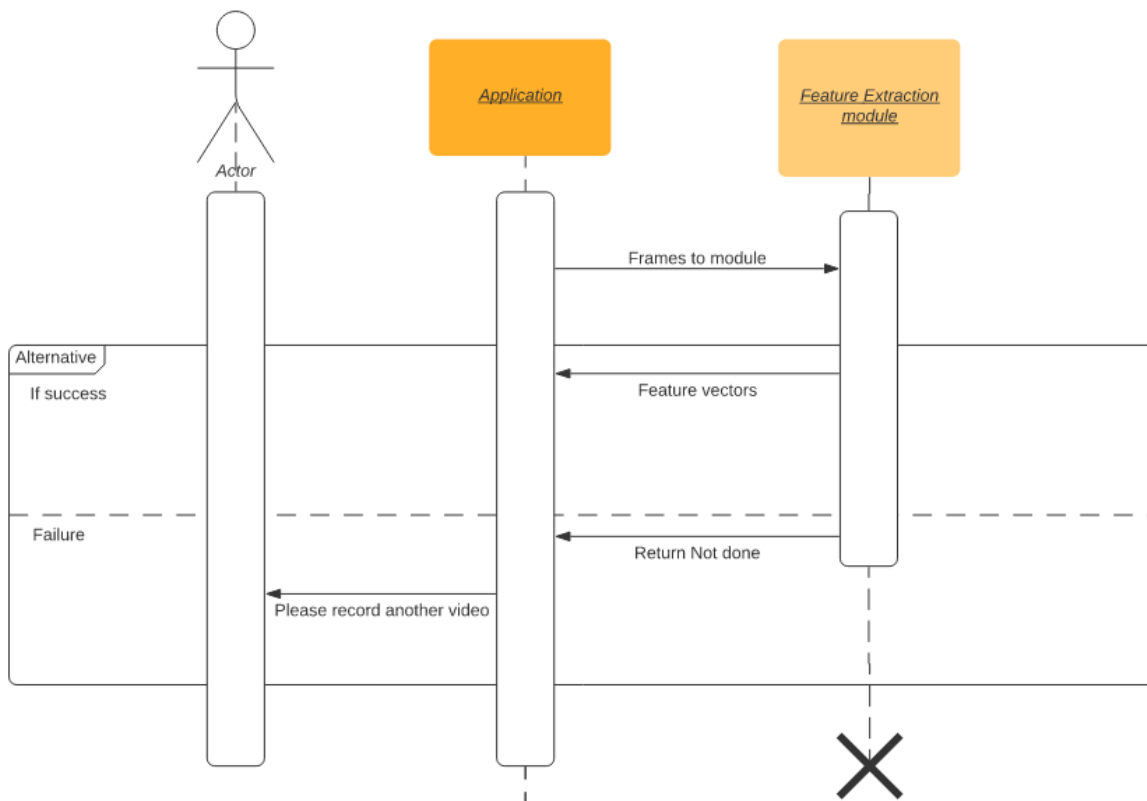
6.4.7.3.1.3.1 Feature Extraction - Function 1 - VIFeatureExtractor

This module takes in the frames and converts them into transformers

6.4.7.3.1.3.2 Feature Extraction - Function 1 - VIModel

This takes the transformers as the input and converts them into features/numbers which can later be used for training.

6.4.7.4 Sequence Diagram

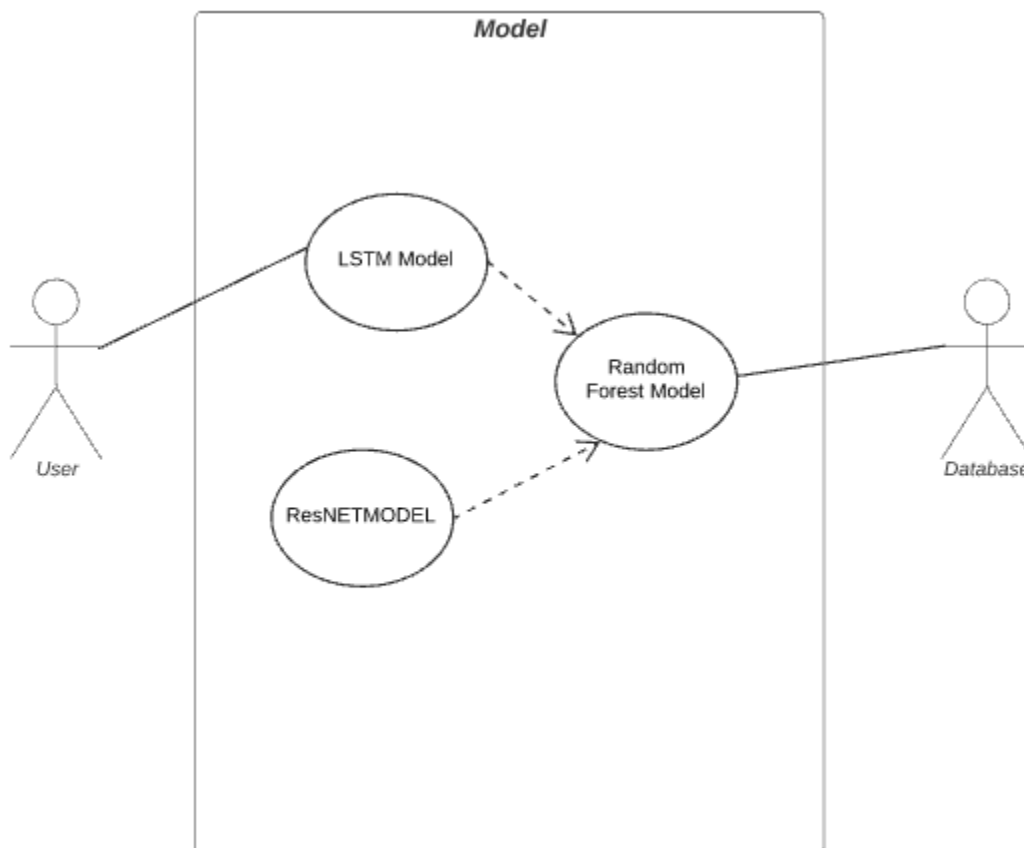


6.4.8 Model

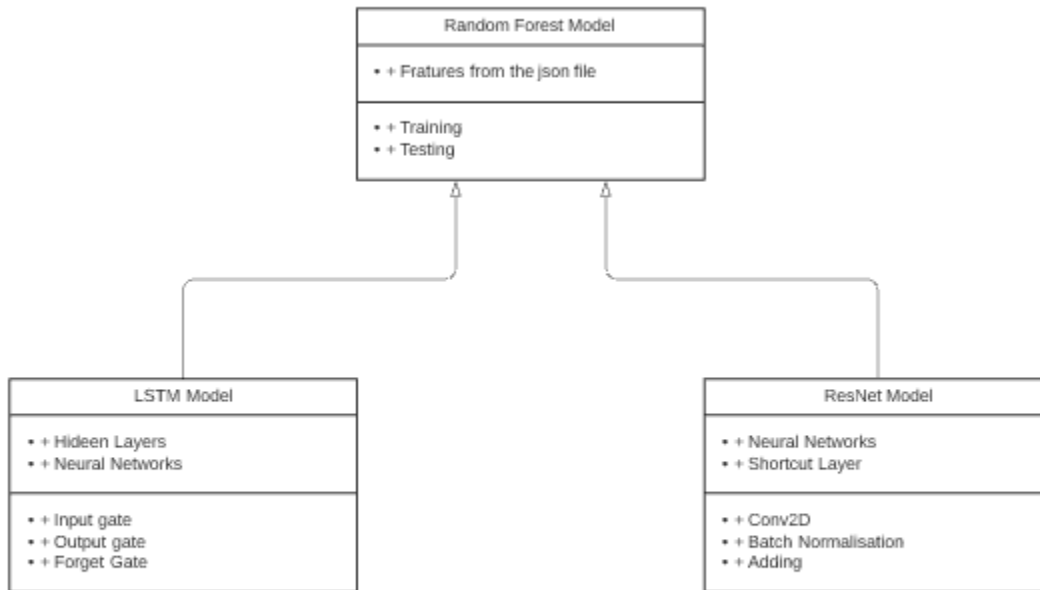
6.4.8.1 Description

This is the most important model of the entire project. In this step the features are being trained. The methods of modelling used here are RNN, LSTM, ResNet and Random Forest Model wherein the model all the models are combined together and are

6.4.8.2 Use Case Diagram



6.4.8.3 Class Diagram



6.4.8.3.1 LSTM Model

6.4.8.3.1.1 LSTM Model - Description

LSTM(Long Short Term Memory) models are those models that, unlike the feedforward neural networks, also have a feedback mechanism which helps the networks to also learn

from the subsequent networks. It has Encoders, Decoders and Attention mechanisms which will be explained in the next section.

6.4.8.3.1.2 LSTM Model - Data members

Data Type	Data Name	Initial Value	Description
Layers	Frames of the preprocessed video	NULL	The network of layers formed from the perceptron
Perceptron	Video Name	Name of each video	A basic unit of neural network that consists of input values, weights, bias and activation values.

6.4.8.3.1.3.1 LSTM Model - Function 1 - Input Gate

The input gate controls the information flow to the current cell state using a pointwise multiplication operation of 'sigmoid' and 'tanh' respectively

6.4.8.3.1.3.2 LSTM Model - Function 2 - Output gate

the output gate decides which information should be passed on to the next hidden state

6.4.8.3.1.3.3 LSTM Model - Function 3 - Forget Gate

The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function.

6.4.8.3.2 ResNet Model

6.4.8.3.2.1 ResNet Model - Description

ResNet Model is a model that was designed to Uphold many layers so that training can happen better and at the same time the vanishing gradient problem could also be solved.

6.4.8.3.2.2 LSTM Model - Data members

Data Type	Data Name	Initial Value	Description
Network of layers	Neural Network	NULL	The network of perceptrons
Layer	Shortcut	NULL	A shortcut that can skip over multiple layers to avoid overtraining

6.4.8.3.2.3.1 ResNet Model - Function 1 - Conv2D

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

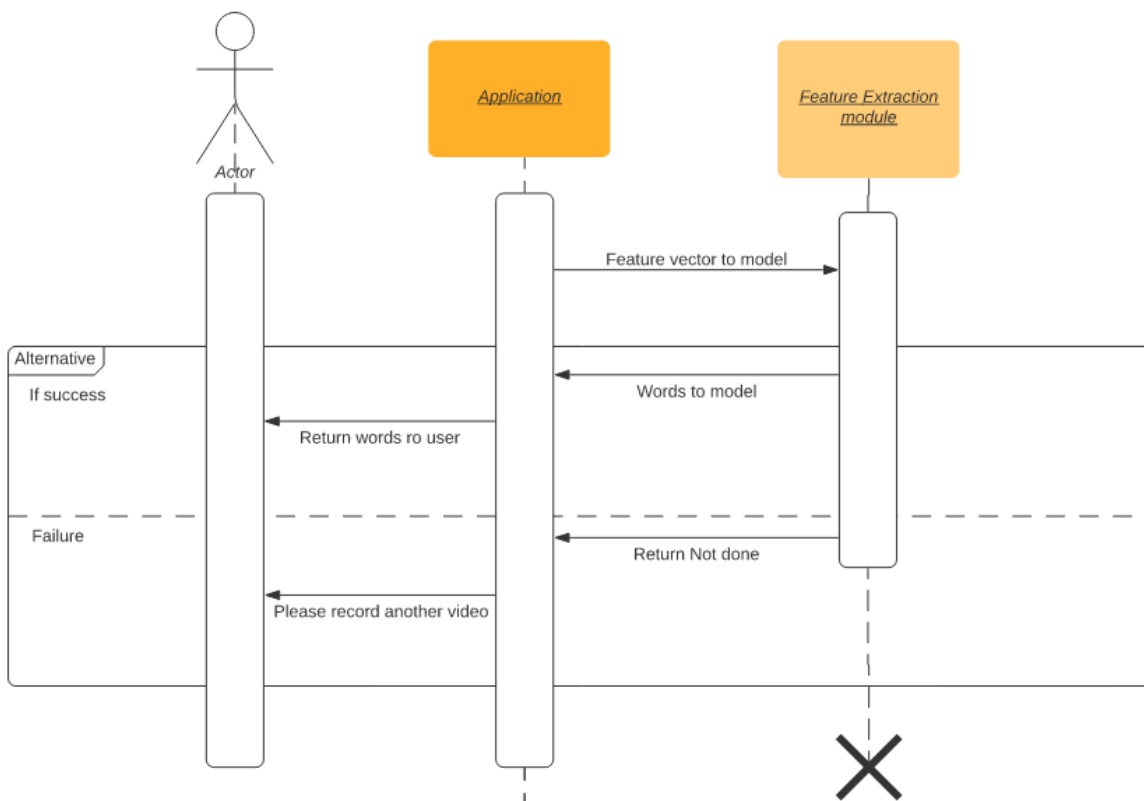
6.4.8.3.2.3.2 ResNet Model - Function 2 - Batch Normalisation

It is a training layer that normalises its inputs

6.4.8.3.2.3.3 ResNet Model - Function 3 - Adding

Adding the resultant Matrix of the Shortcut iteration and the normal path iteration

6.4.8.4 Sequence Diagram

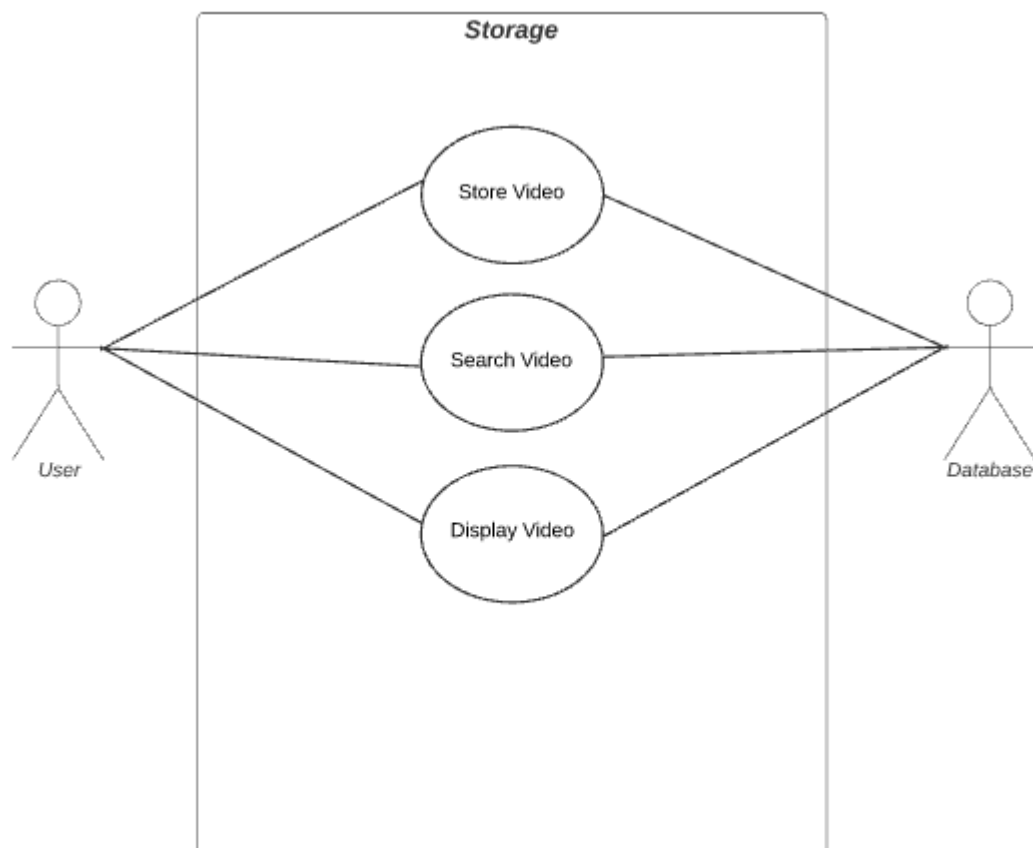


6.4.9 Storage

6.4.9.1 Description

This module is responsible for the backend saving of data where the unprocessed, preprocessed videos along with their captions are stored and retrieved.

6.4.9.2 Use Case Diagram



6.4.9.3 Class Diagram

Storage
<ul style="list-style-type: none"> + Video: mp4 + Video_name: String
<ul style="list-style-type: none"> + Store Video and Associated Caption() + Search Video and Associated Caption() + Display Video and Associated Caption()

6.4.9.3.1 Storage

6.4.9.3.1.1 Storage - Description

In this step the Video which was preprocessed in the previous step is taken and it is divided into 15 frames and each frame is named after the video.

6.4.9.3.1.2 Storage - Data members

Data Type	Data Name	Initial Value	Description
mp4	Video	NULL	The video that is being stored
String	Video Name	Name of each video	Name that can be stored with preprocessed result

6.4.9.3.1.3.1 Storage - Function 1 - Store Video and associated Caption()

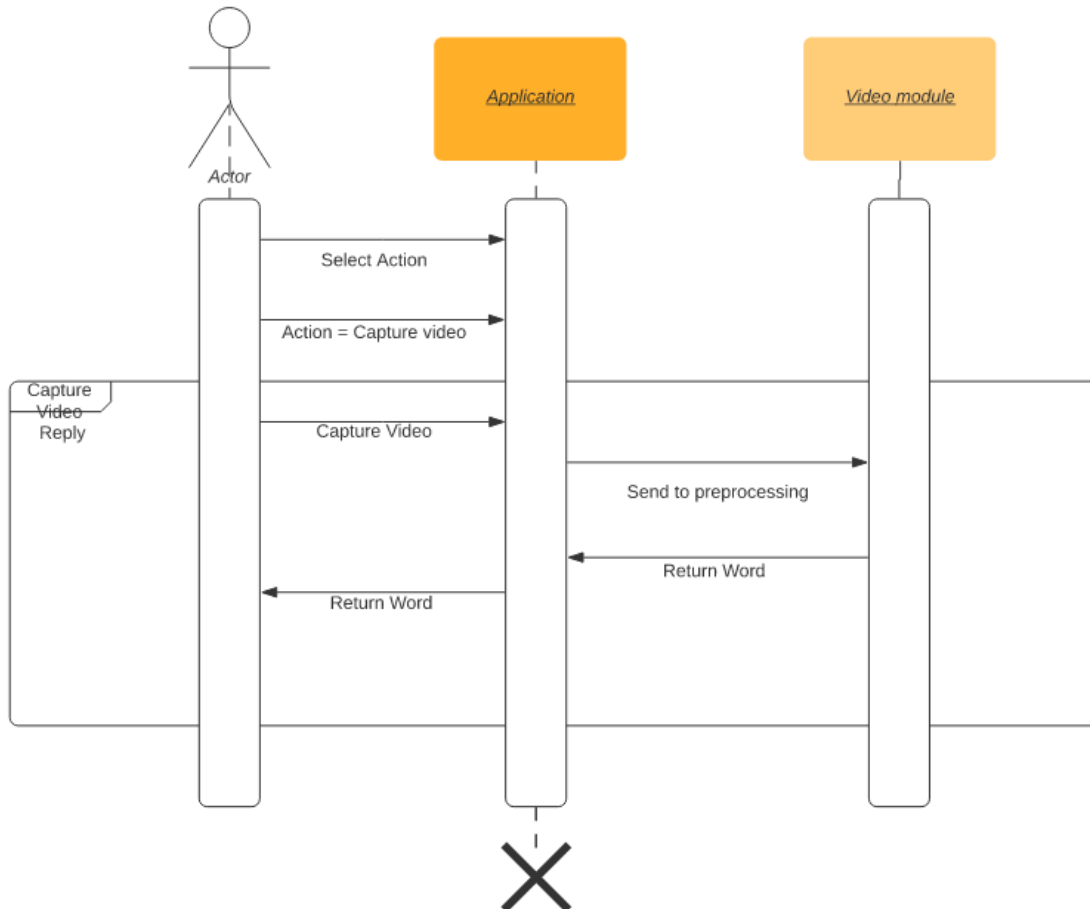
This function is used to store the Video and caption

6.4.9.3.1.3.2 Storage - Function 2 - Search Video and associated Caption()

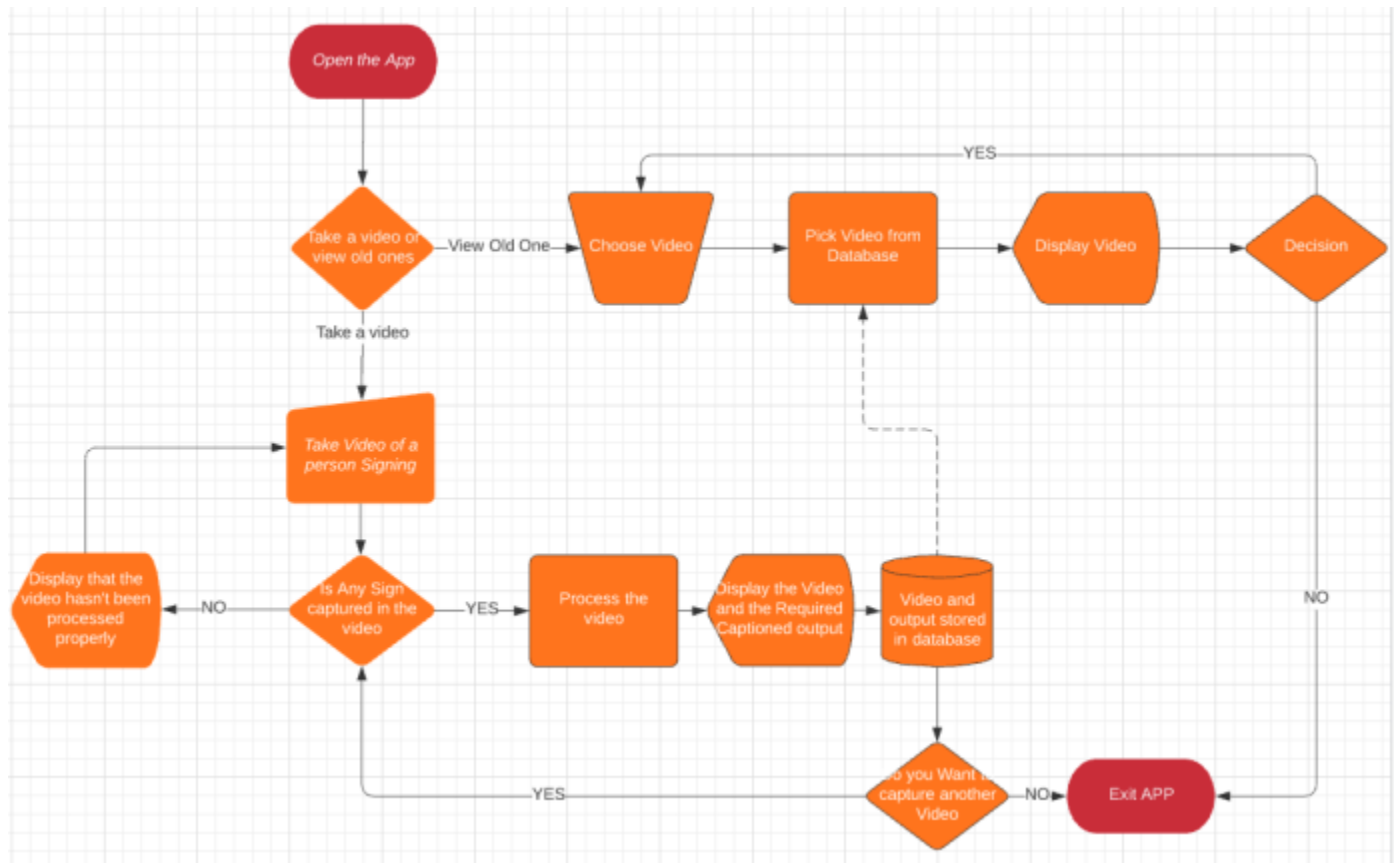
This function is used to search for the video and its associated caption

6.4.9.3.1.3.3 Storage - Function 3 - Display Video and associated Caption()

This function is used to display the video and its associated caption

6.4.9.4 Sequence Diagram

VII) System Design (Detailed):

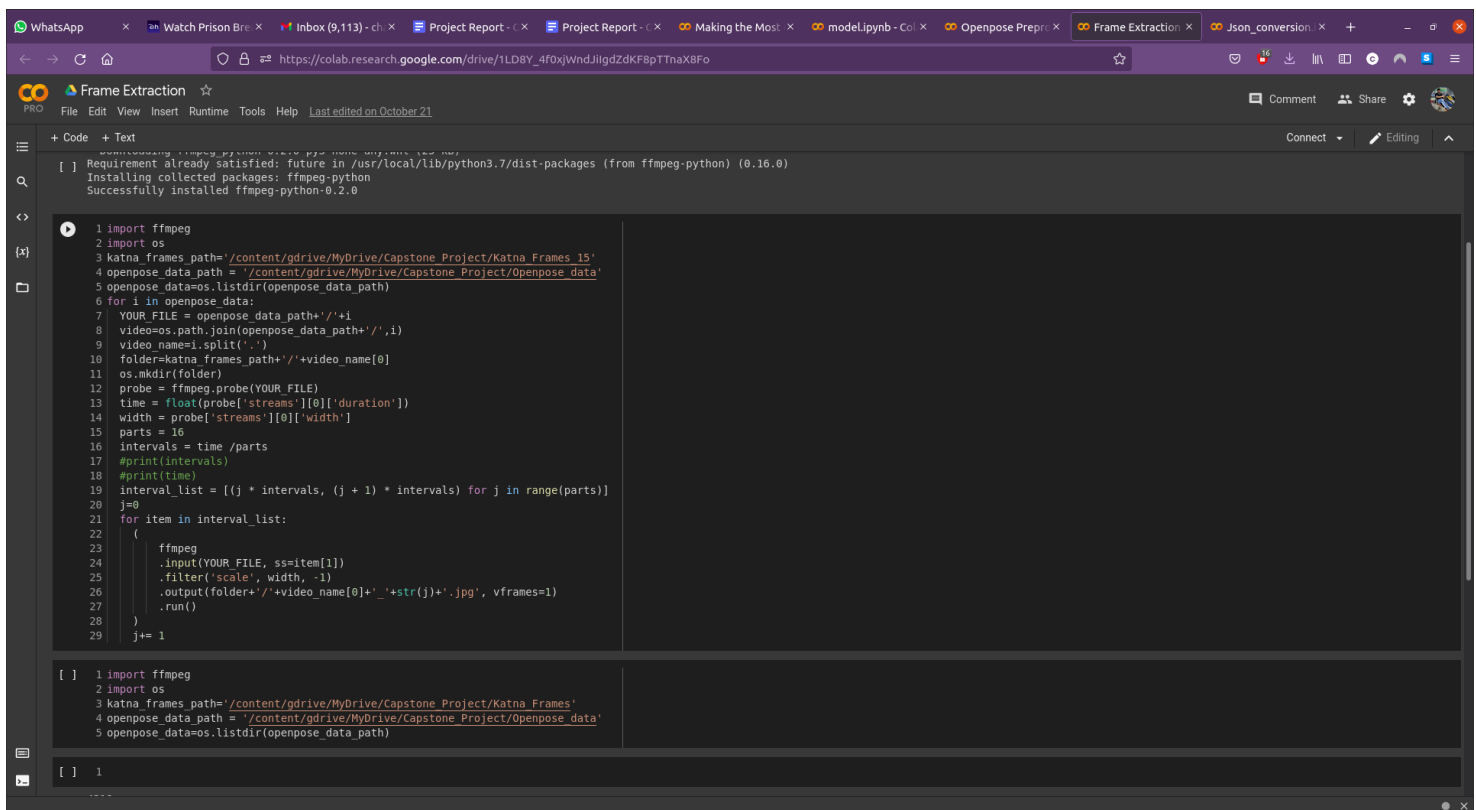


The System as a whole is divided into 3 parts- Front-end, core code and Backend.

Firstly we will introduce the core code. Here we have a dataset called INCLUDE which has 4292 video's. Each video has 1 sign. We have used preprocessing in 3 ways - The image is first made cropped in the part as a bounding box , The second Open pose implementation and third finger keypoint detection. All three images are concatenated into X3 images. Each video is divided into frames and each frame is then passed to the

pre-processing layer and X9 images are gathered. These images are passed through a spatial CNN and the output of the CNN is passed through the RNN which outputs a one hot vector which predicts the words. The words that we get are then formed as a sentence based on a context based model.

VIII) PSEUDO CODE:



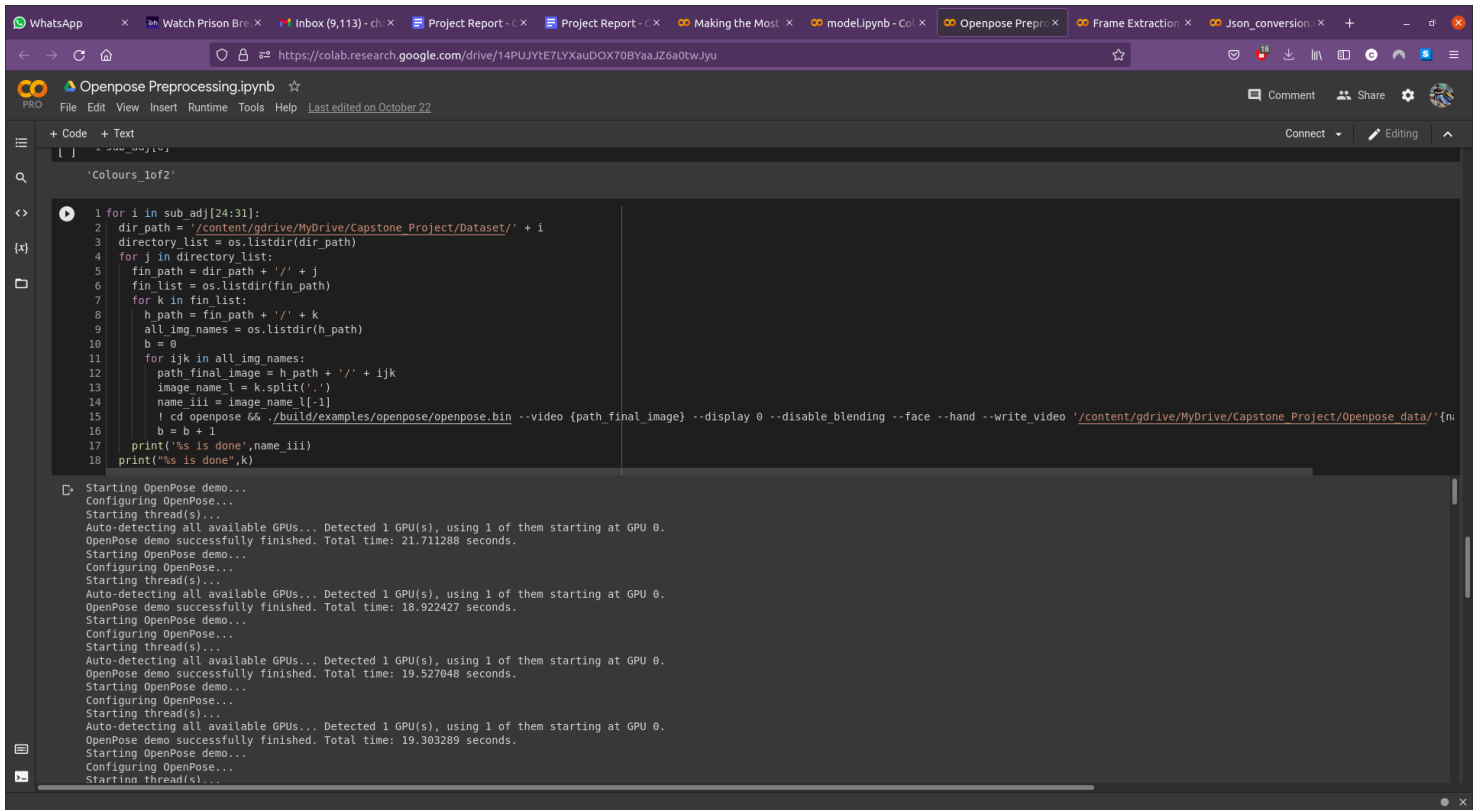
```
[ ] Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from ffmpeg-python) (0.16.0)
Installing collected packages: ffmpeg-python
Successfully installed ffmpeg-python-0.2.0

1 import ffmpeg
2 import os
3 katna_frames_path="/content/gdrive/MyDrive/Capstone Project/Katna Frames 15"
4 openpose_data_path = "/content/gdrive/MyDrive/Capstone Project/Openpose_data"
5 openpose_data=os.listdir(openpose_data_path)
6 for i in openpose_data:
7     YOUR_FILE = openpose_data_path+'/'+i
8     video=os.path.join(openpose_data_path+'/'+i)
9     video_name=i.split('.')
10    folder=katna_frames_path+'/'+video_name[0]
11    os.mkdir(folder)
12    probe = ffmpeg.probe(YOUR_FILE)
13    time = float(probe['streams'][0]['duration'])
14    width = probe['streams'][0]['width']
15    parts = 16
16    intervals = time / parts
17    #print(intervals)
18    #print(time)
19    interval_list = [(j * intervals, (j + 1) * intervals) for j in range(parts)]
20    j=0
21    for item in interval_list:
22        (
23            ffmpeg
24            .input(YOUR_FILE, ss=item[1])
25            .filter('scale', width, -1)
26            .output(folder+'/'+video_name[0]+'_'+str(j)+'.jpg', vframes=1)
27            .run()
28        )
29        j+= 1

[ ] 1 import ffmpeg
2 import os
3 katna_frames_path="/content/gdrive/MyDrive/Capstone Project/Katna Frames"
4 openpose_data_path = "/content/gdrive/MyDrive/Capstone Project/Openpose_data"
5 openpose_data=os.listdir(openpose_data_path)

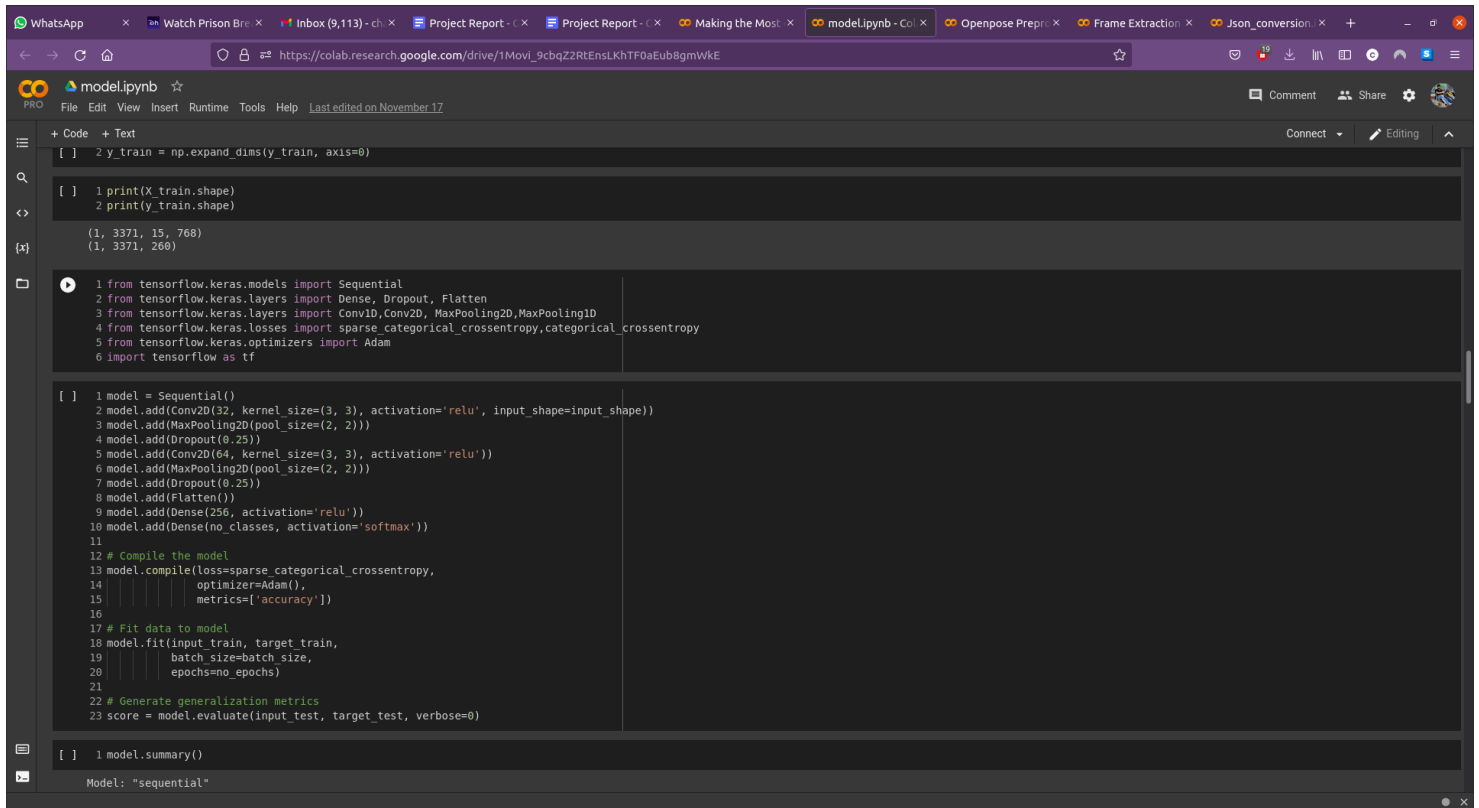
[ ] 1
```

```
1 import numpy as np
2 import json
3 def encoding(final_path_list):
4     a = []
5     for path in final_path_list:
6         image = Image.open(path)
7         inputs = feature_extractor(images=image, return_tensors="pt")
8         outputs = model(*inputs)
9         output_state = outputs.pooler_output
10        k = output_state.detach().numpy().reshape([1,768])
11        a.append(k)
12    return a
13
14 def image_concatenate(image_emb_arr):
15     k = image_emb_arr[0]
16     for j in image_emb_arr[1:]:
17         k = np.concatenate((k, j), axis=0)
18     return k
19
20 data final = {}
21 name_1 = '/content/gdrive/MyDrive/Capstone Project/Katna Frames 15'
22 for file_name in frames_file:
23     name_2 = name_1 + '/' + file_name
24     image_name_path = os.listdir(name_2)
25     if len(image_name_path)!=15:
26         continue
27     path_1 = name_2 + '/' + image_name_path[0]
28     path_2 = name_2 + '/' + image_name_path[1]
29     path_3 = name_2 + '/' + image_name_path[2]
30     path_4 = name_2 + '/' + image_name_path[3]
31     path_5 = name_2 + '/' + image_name_path[4]
32     path_6 = name_2 + '/' + image_name_path[5]
33     path_7 = name_2 + '/' + image_name_path[6]
34     path_8 = name_2 + '/' + image_name_path[7]
35     path_9 = name_2 + '/' + image_name_path[8]
36     path_10 = name_2 + '/' + image_name_path[9]
37     path_11 = name_2 + '/' + image_name_path[10]
38     path_12 = name_2 + '/' + image_name_path[11]
39     path_13 = name_2 + '/' + image_name_path[12]
40     path_14 = name_2 + '/' + image_name_path[13]
41     path_15 = name_2 + '/' + image_name_path[14]
42     final_path_list = []
43     final_path_list.append(path_1)
44     final_path_list.append(path_2)
```



```
1 for i in sub_adj[24:31]:
2   dir_path = '/content/gdrive/MyDrive/Capstone Project/Dataset/' + i
3   directory_list = os.listdir(dir_path)
4   for j in directory_list:
5     fin_path = dir_path + '/' + j
6     fin_list = os.listdir(fin_path)
7     for k in fin_list:
8       h_path = fin_path + '/' + k
9       all_img_names = os.listdir(h_path)
10      b = 0
11      for ijk in all_img_names:
12        path_final_image = h_path + '/' + ijk
13        image_name_l = k.split('.')
14        name_iii = image_name_l[-1]
15        ! cd openpose && ./build/examples/openpose/openpose.bin --video {path_final_image} --display 0 --disable_blending --face --hand --write_video '/content/gdrive/MyDrive/Capstone Project/Openpose_data/' + name_iii
16        b = b + 1
17      print('%s is done' % name_iii)
18      print('%s is done' % k)
```

Starting OpenPose demo...
Configuring OpenPose...
Starting thread(s)...
Auto-detecting all available GPUs... Detected 1 GPU(s), using 1 of them starting at GPU 0.
OpenPose demo successfully finished. Total time: 21.711288 seconds.
Starting OpenPose demo...
Configuring OpenPose...
Starting thread(s)...
Auto-detecting all available GPUs... Detected 1 GPU(s), using 1 of them starting at GPU 0.
OpenPose demo successfully finished. Total time: 18.922427 seconds.
Starting OpenPose demo...
Configuring OpenPose...
Starting thread(s)...
Auto-detecting all available GPUs... Detected 1 GPU(s), using 1 of them starting at GPU 0.
OpenPose demo successfully finished. Total time: 19.527048 seconds.
Starting OpenPose demo...
Configuring OpenPose...
Starting thread(s)...
Auto-detecting all available GPUs... Detected 1 GPU(s), using 1 of them starting at GPU 0.
OpenPose demo successfully finished. Total time: 19.303289 seconds.
Starting OpenPose demo...
Configuring OpenPose...
Starting thread(s)...



The screenshot shows a Google Colab notebook with the following code:

```
[ ] 2 y_train = np.expand_dims(y_train, axis=0)

[ ] 1 print(X_train.shape)
    2 print(y_train.shape)

(1, 3371, 15, 768)
(1, 3371, 260)

1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Dense, Dropout, Flatten
3 from tensorflow.keras.layers import Conv1D, Conv2D, MaxPooling2D, MaxPooling1D
4 from tensorflow.keras.losses import sparse_categorical_crossentropy, categorical_crossentropy
5 from tensorflow.keras.optimizers import Adam
6 import tensorflow as tf

[ ] 1 model = Sequential()
    2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    3 model.add(MaxPooling2D(pool_size=(2, 2)))
    4 model.add(Dropout(0.25))
    5 model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    6 model.add(MaxPooling2D(pool_size=(2, 2)))
    7 model.add(Dropout(0.25))
    8 model.add(Flatten())
    9 model.add(Dense(256, activation='relu'))
   10 model.add(Dense(no_classes, activation='softmax'))
   11
   12 # Compile the model
   13 model.compile(loss=sparse_categorical_crossentropy,
   14               optimizer=Adam(),
   15               metrics=['accuracy'])
   16
   17 # Fit data to model
   18 model.fit(input_train, target_train,
   19           batch_size=batch_size,
   20           epochs=no_epochs)
   21
   22 # Generate generalization metrics
   23 score = model.evaluate(input_test, target_test, verbose=0)

[ ] 1 model.summary()

Model: "sequential"
```

IX) Conclusion and Future Work

We really are looking forward to working on improving the services provided in the App. We plan to simultaneously generate captions for the signs so that the process would appeal to more real time usage. Also instead of using stick figures for converting the captions to video we would like to use GAN's to generate an artificial human so that it would be more appealing to the end users.

X) Appendices

Appendix A: Definitions, Acronyms and Abbreviations

- 1) ML :- Machine Learning
- 2) DL : Deep Learning
- 3) ISL :- Indian Sign Language
- 4) CNN :- Convolutional Neural Network
- 5) RNN :- Recurrent Neural Network
- 6) UML :- Unified Machine Language
- 7) INCLUDE:- Indian Lexicon Sign Language Dataset
- 8) UI : User Interface

Appendix B: References

- 1) Danielle Bragg, Oscar Koller, Mary Bellard, Larwan Berke, Patrick Boudreault, Annelies Braffort, Naomi Caselli, Matt Huenerfauth, Hernisa Kacorri, Tessa Verhoef, Christian Vogler, and Meredith Ringel Morris. 2019. Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective. In The 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19). Association for Computing Machinery, New York, NY, USA, 16–31. DOI: <https://doi.org/10.1145/3308561.3353774>

- 2) P. C. Badhe and V. Kulkarni, "Indian sign language translator using gesture recognition algorithm," 2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS), Bhubaneswar, India, 2015, pp. 195-200, doi: 10.1109/CGVIS.2015.7449921.
- 3) Parton, Becky. (2006). Sign Language Recognition and Translation: A Multidisciplined Approach From the Field of Artificial Intelligence. Journal of deaf studies and deaf education. 11. 94-101. 10.1093/deafed/enj003.
- 4) N. C. Camgoz, S. Hadfield, O. Koller, H. Ney and R. Bowden, "Neural Sign Language Translation," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 7784-7793, doi: 10.1109/CVPR.2018.00812.
- 5) Akyol, S., & Alvarado, P. (2001). Finding relevant image content for mobile signlanguage recognition. Retrieved June 17, 2005, from http://www.techinfo.rwth-aachen.de/Veroeffentlichungen/V001_2001.pdf
- 6) Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison DONGXU LI, Cristian Rodriguez, Xin Yu, HONGDONG LI; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 1459-1469

Appendix C: Record of Change History

#	Date	Document Version No.	Change Description	Reason for Change
1.	1/11/2021	1	-	-
2.				
3.				

Appendix D: Traceability Matrix

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.	LLD Reference Section No. Name
Login page	1	2
Signup Page	2	3
Home page	3	1
Help Page	4	8
How to login page	5	9
Logout Page	6	-