ASL Interpreter Mobile Application

Raymond Cook
Jadepan Thedthong

ASL INTERPRETER APPLICATION FINAL REPORT

Execution

Task	End Date	Owner
Build Hands keypoint estimation for upper body	10/13/2022	Jadepan Thedthong
Create Initial GUI for App	10/13/2022	Raymond Cook
Build Pose Keypoint Estimation	10/27/2022	Jadepan Thedthong
Merge two keypoint estimation models together	11/10/2022	Jadepan Thedthong
Implement Account Login into App	11/10/2022	Raymond Cook
Final Validation Checks for All Subsystems	11/24/2022	All
Final Presentation and Report	12/01/2022	All
Task	**End Date**	**Owner**
Finalize dataset for Word-Level Translation	2/6/2023	Jadepan Thedthong
Establish camera recording and playback features	2/8/2023	Raymond Cook
Launch server and initialize the model on it	2/14/2023	Raymond Cook
Create server endpoints to establish application communication	2/28/2023	Raymond Cook
Data Augmentation process to expand dataset	3/17/2023	Jadepan Thedthong
Develop video upload feature for application	3/21/2023	Raymond Cook
Finalize the ML model and deploy on server	4/10/2023	Jadepan Thedthong
Incorporate lesson plan and tracking onto application	4/12/2023	Raymond Cook
Final Validations	4/21/2023	All
Final Presentation, Report, and Showcase	4/30/2023	All

ASL Interpreter Mobile Application

Raymond Cook
Jadepan Thedthong

CONCEPT OF OPERATIONS

REVISION – 2
30 April 2023

Concept of Operations
for
ASL Interpreter Mobile Application

Team 33

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2022/09/15	Raymond Cook		Draft Release
1	2022/10/01	Raymond Cook		Updated to match with FSR/ICD
2	2023/04/30	Raymond Cook		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

List of Figures

No table of figures entries found.

1) Executive Summary

The project sponsor is looking to provide a streamlined approach to facilitating conversations between deaf and/or mute people that must communicate using American Sign Language (ASL), who will be referred to as signers for the remainder of this report. This approach should be easily accessible, require minimal training, and provide accurate results quickly. The goal is to use a trained machine learning model to effectively translate ASL and expedite conversations between signers and the general population. The desired benefits of this project are to allow more efficient conversations between people that do and do not know ASL as well as increase basic comprehension of ASL among the general populace.

2) Introduction

According to the United States Census, approximately 11.5 million people are deaf which constitutes roughly 3.6 percent of the American population. The standard form of communication for signers in the English-speaking North American region is sign language, specifically ASL. However, most of the general population will have minimal or zero knowledge of the language which can make conversations between signers and the general population tedious and at times, difficult. This breakdown in communication creates the need for an interpreter, whether an actual person or other tool, to bridge the gap between the two parties to increase the quality of life for signers in any general setting.

2.1) Background

The system being created will help signers by replacing alternative methods of communication, such as writing or texting to communicate with non-signers. Our system will analyze videos and translate ASL to English text to eliminate the need for a human translator or other means of translation. Additionally, it will provide an alternative to older methods of learning ASL through an in-person school or online videos by including a structured lesson plan that will provide a more intuitive and practical approach to learning. This lesson plan will not teach the nuances and specifics of ASL, but rather common words and phrases to accelerate the user in using ASL to communicate, even if on a rudimentary level. Ultimately, the system aims to improve a signer's quality of life by improving their ability to have conversations with non-signers as well as increase the accessibility of education of ASL.

2.2) Overview

The system will be created as a mobile application for Android devices and utilize an account system to keep track of user's preferences and progress in their lessons. This app will consist of two major features to assist users in translating and learning ASL, the first of which being the camera scanning module. It will take and send a video to a server that will serve as the machine learning core. The server will then send back a written translation in English of whatever word is in the video. The second core feature is the previously mentioned structured lesson plan to help users learn ASL and will use the account database to carry the user's progress across multiple devices.

2.3) Referenced Documents and Standards

Document Title	Release Date	Source
S1810 | Disability Characteristics	2021	American Community Survey
Python Library Reference	3.8	Python Software Foundation
TensorFlow Documentation	2.1	TensorFlow.org

Table 1: Reference Documents and Standards

3) Operating Concept

3.1) Scope

The application mentioned in this report is designed for people that need to communicate effectively with people who can not communicate verbally. Ideally, this should improve the daily lives of those who can only communicate through ASL by making everyday conversations and interactions easier for everybody involved. The app will be portable and accessible so that users can use it anywhere as long as they have an internet connection.

3.1.1) Scope Modifications

The original scope of the project was to provide an application that could translate ASL on a sentence-level to act as an interpreter for entire conversations. Unfortunately, we ran into two main issues that lead to us receiving a reduction in scope from our sponsor.

The first issue was devising a method that would allow us to splice a video into smaller windows in order to capture each individual word being signed in a sentence. After that, combining the words back together successfully without any errors such as incorrectly translated words, missed words, duplicated words, and processing time, proved to be an elusive task. Our initial plan to implement this would take too long to process given current hardware limitations and defeat the original goal of providing relatively quick translations.

The second issue was that even if the first one were to be resolved, the matter of translating a raw ASL translation to proper English was much more difficult than originally anticipated. The grammatical structure of ASL is drastically different from proper English and as such, would require its own ML model or separate algorithm to try and convert from the raw ASL translations to English text. Another difficulty of translating from ASL to English is that ASL is a much more contextual language so much of the vocabulary is slimmed down to accelerate their conversations and reduce physical movement, which requires intuition and context to successfully translate back into English.

As such, our sponsor approved a reduction in scope for this system to be a word-level translation to act as a proof-of-concept of what could be given more time and resources.

3.2) Operational Description and Constraints

Upon launching the application, users will be asked to log-in to or sign-up for an account. This will allow them to access their data across multiple devices so that they may continue their lessons and modify their settings anywhere. If the user does not sign in, they will not be able to access the rest of the app.

Once logged into the app, users will have two options to choose from - the translator portion and the ASL lesson plan. If the user selects the Translator module, they will be asked to capture at least the upper body of the signer using the back camera of their device. After the signer is recorded signing, the user will be allowed to playback the video to check its quality, upload it, and then the app will retrieve the translation and display that to the user, allowing them to effectively communicate without knowing ASL.

If the user selects the second option, they will be introduced into the educational aspect of the app. A series of lessons will be implemented, starting with learning the alphabet in ASL and progressing through to basic words and phrases. Included in each lesson are clips of the major word or phrase being taught so that the user can have a visual representation of how to sign it.

3.3) System Description

The system will contain two main subsystems: the Translator module and the Application and Server subsystem. The Translator module will use the camera on the smartphone to record a video and then send it to our server, which will detect the signer, analyze their motions, and then translate the sign language into a readable text which is sent back to the application for the user to read. To train the model, a dataset containing multiple videos of each word being signed was used alongside a corresponding translation model. As a result, our machine learning model can compare the prediction with the expected translation during the training process.

The application and server module comprises the app’s navigation as well as a lesson plan to teach ASL. Here, the user will be able to select the user profile page, the translator module, or the ASL lesson plan. The translator module allows the user to record a video of a signer, play it back to verify quality, upload it to the server, and receive a translation generated by the machine learning model. Data tracking is accomplished using our server-side storage to maintain the user’s progress and tie it to their credentials. The lesson plan consists of a series of lessons to help the user learn ASL which includes a pre-recorded video to demonstrate ASL for the user to have a visual representation.

3.4) Modes of Operations

The application will have two main areas for the user to operate in: the Translator module and the Education module. The Translator module utilizes the machine learning model in the server to detect and translate ASL into English text for the user. Users will also be able to choose to participate in a program to learn ASL within the Education module.

3.5) Users

The target users for this application are people who interact with signers often. To use this product, all users will need an Android device running Android OS version 9 (Pie) or later and basic knowledge of how to navigate and use apps. To maintain the service and push updates, the developer should have intermediate knowledge in Kotlin (Android development language), Jetpack Compose for the application interface, PHP for the server backend, as well as Python, TensorFlow, Keras, MediaPipe, and OpenCV to work with the machine learning program.

3.6) Support

Most buttons will be labeled with a brief one to two word description to make navigation through the app intuitive without the need for detailed instruction. Buttons not explicitly labeled will have icons that express the purpose of the button. A code repository will also be available with documentation for users who wish to contribute to future development.

4) Scenarios

4.1) Single Person In a Frame

In an ideal situation only one person would be in view of the camera, simplifying the video that the machine learning model has to process by having a sole target to focus on. In this situation, the user of the app would point the camera at the user to record a video with the Translator module being used to convert the signer's ASL into English text to translate the words being signed.

4.2) Multiple People In a Frame

As our application is intended to be portable and used in any setting, we expect other people and objects to be in the view of the camera during usage of the app other than the signer. The initial design for the Translator module is to let the machine learning algorithm try to translate everyone in the frame. Obviously, not everyone in the frame will be signing so the algorithm will send back the response associated with the highest probability of signing as we expect that person to be the one communicating. A threshold is included in the model so that if the model is unable to detect any one person signing anything, an error message will be displayed to the user so that they know how to remedy the situation and avoid false positives.

4.3) Learning ASL

For users who would like to learn ASL to reduce their dependence on the Translator module, the app also offers a method for users to begin learning ASL. This lesson plan begins with the basics and teaches the user the alphabet in ASL. As the user becomes more familiar with the alphabet and the language itself, they will progress to the next set of lessons to begin learning common words and phrases. A pre-recorded video is used to demonstrate these letters, words, and phrases to the user to provide a more intuitive learning experience rather than reading descriptions of motion or seeing a series of screenshots.

5) Analysis

5.1) Summary of Proposed Improvements

Only one server is planned to be used for this product. Future improvements could introduce additional servers in more locations for both redundancy and to reduce latency in different geographical areas.

As with any machine learning model, to expand the vocabulary that it is able to translate we will need a larger dataset with samples per word in the magnitude of ten thousands in order to account for various lighting situations, camera angles, speed, and so forth.

5.2) Disadvantages and Limitations

The translation's accuracy may not be entirely accurate due to errors within the machine learning model. Also, the video uploading process could impact performance as missing/corrupted packets would result in data that can not be analyzed. In addition, without an active internet connection the app loses its functionality due to being unable to communicate with the server. Lastly, The lesson plan may have inaccuracies due to our limited knowledge in structuring a curriculum for teaching ASL to newcomers.

5.3) Alternatives

Current plans involve using an account provider such as Auth0. Future implementations may change this to meet development needs or develop an in-house system to reduce operational costs. The current machine learning model is a Long Short-Term Memory (LSTM) model, which may be changed in the future depending on more time being invested in studying different models and deciding on one that works best for this approach.

5.4) Impact

This project seeks to provide an accessible and reliable method for deaf and/or mute people to communicate with people who are not familiar with ASL. The ethical concerns brought on by this project are the widespread use of cameras in public settings being used to train a machine-learning algorithm as well as the proper storage of our user's data to ensure its safety.

ASL Interpreter Mobile Application

Raymond Cook
Jadepan Thedthong

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 2
30 April 2023

FUNCTIONAL SYSTEM REQUIREMENT
FOR
ASL Interpreter Mobile Application

TEAM 33

APPROVED BY:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2022/10/01	Jadepan Thedthong		Draft Release
1	2022/10/01	Jadepan Thedthong		Semester Final Report
2	2023/04/30	Raymond Cook		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

List of Figures

6) Introduction

6.1) Purpose and Scope

The ASL interpreter is a tool to help people communicate with deaf people and raise awareness for those who can not communicate with verbal English. In order to ease communication between the two parties, our application will translate the sign language using the camera on the user's phone. The system will then send this video to our cloud server that hosts our machine learning Translator which will translate the signed word and then send the translation back to the application. In order to help raise awareness among the general public and bridge the gap between the two parties, our application will also offer lessons complete with a user account system to keep track of the user's progress.

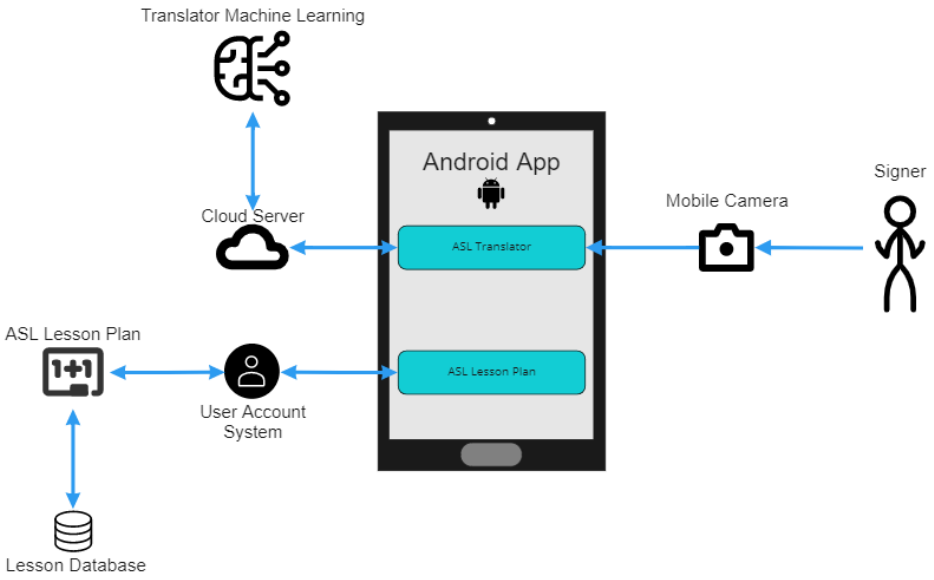

Figure 1: Project Conceptual Diagram

6.2) Responsibility and Change Authority

Progress on the project will be evaluated by the team leader, Raymond Cook, for quality and timeliness and to ensure the original requirements of the project are being met. The project will consist of two main components: the machine learning-driven translator as well as the application and server code. Jadepan Thedthong will be in charge of the machine learning translator, including body and extremity detection, LSTM model development, and final translation from ASL to English. Raymond Cook will be responsible for developing the Android application, designing a basic curriculum to help teach ASL, and development and maintenance of the server where both the machine learning algorithm and the server database for user accounts as well as materials for the curriculum will reside.

7) Application and Reference Documents

7.1) Application Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein.

Document Name	Revision/Release Date	Publisher
Python Library Reference	3.8	Python Software Foundation
TensorFlow Documentation	2.1	TensorFlow.org
Kotlin Documentation	v1.7.20	Jetbrains
YOLOv5	2020/05/18	Glenn Jocher
MediaPipe	0.8.11	Google LLC
Auth0		Okta
Keras API reference		Keras
Scikit-Learn Documentation	0.21.3	Scikit-Learn
Retrofit Library (Android HTTP Communication)	2013	Square, Inc.

Table 1: Application Documents

7.2) Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Name	Revision/Release Date	Publisher
3D Image Classification from CT Scans	2020/09/23	Keras
Step by Step Implementation: 3D CNN in Keras	2020/03/28	Towards Data Science
OpenPose	1.7.0	CMU Perceptual Computing Lab
LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past	2022/02/06	Towards Data Science

Table 2: Reference Documents

7.3) Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions. All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable document are for guidance and information only, apart from ICD’s that have their applicable documents considered to be incorporated as cited.

8) Requirements

8.1) System Definition

As shown in Figure 1, our ASL interpreter system is divided into the following key subsystems:

a. **ASL Translation Module**

   i. **Detection and Key Points Estimation Model**: This model will detect the essential key points of the signer from a video taken by the mobile camera with the key points being the shoulders, elbows, wrists, and finger joints. This model will then export the estimated location of the key points into our translation model.
   ii. **Translation Model**: This model is a Long Short-Term Memory (LSTM), a type of Recurrent Neural Network, where the data from the Detection and Key Points Estimation Model will be processed and the word associated with the sign predicted. Then, it will output the translation as text to the user through the application.

b. **Lesson Plan Module**

   i. **User Account System**: The user's progress on their lessons will be stored on a server instead of the device itself so that in the event the user changes or reformats their device, they will be able to continue where they left off.
   ii. **ASL Lesson Plan**: This lesson plan will provide the user with educational lessons teaching ASL. It will include illustrations and quizzes so that the user can follow along and actively learn ASL.

c. **User Interface**
   i. One of our goals is to be accessible to the general public; therefore, an easy-to-use interface will help users navigate our application with ease. This interface allows the user to go to one of three main screens; the user profile page, the translator module, and the lesson plan.

d. **Remote Server**
   i. The remote server has been set up with multiple endpoints to allow the application to communicate with the server to send and retrieve data when necessary. This has been accomplished by the use of multiple HTTP POST and GET requests that are processed using PHP. The server also stores user data for

basic metric tracking, such as words translated and lesson progress.

8.2) Characteristics

8.2.1) Performance Requirements

8.2.1.1) ASL Translation Accuracy

To prevent mistranslations, a high level of confidence is needed before sending the words back to the user. To ensure an accurate and reliable translation, the machine learning translation model shall have at least 60% accuracy when compared to a separate validation dataset and real-world testing. Even though we could not find any source that specifies a guideline for recommended accuracies for this kind of model, we have decided that an accuracy of 60% is a good foundation that serves as a proof-of-concept and can still be improved.

8.2.1.2) ASL Translation Vocabulary

Operating under the assumption that with 30 words basic communication can be performed, the model shall have the ability to translate at least 30 words from ASL to English.

8.2.2) Software Requirements

8.2.2.1) Programming Language - Machine Learning Model

To complete the application in the given timeframe, Python 3.8 will be used due to our familiarity with it as well as the vast amount of open-source libraries available. One such library is TensorFlow which shall be used to implement our convolutional neural network algorithm. All dependencies in the project also are compatible with version 3.8, making it an ideal development version due to lack of workarounds needed.

8.2.2.2) Use of TensorFlow

TensorFlow is a free library with extensive documentation and support. Due to this, we shall be using it to power our LSTM model. We have also decided to use Keras to construct our model as it is a high-level API of the TensorFlow platform. Thus, TensorFlow is most suitable for this project.

8.2.2.3) Programming Language - Android Application

The Android application will be built using Kotlin as that is the standard put out by Google for development. The actual interface will be built using Jetpack Compose, another tool put out by Google to accelerate development. In addition, many resources are available as well as documentation.

8.2.3) Interface Requirements

8.2.3.1) HTTP GET Server Interaction: Load Data

In order to test the server hosting the assets, it must have a functional HTTP GET endpoint to act as an interface. Using this, we can load lessons from the server to a device and update certain items in the lesson plan without pushing an update to the application itself.

8.2.3.2) HTTP POST Server Interaction: Translation Model

In addition, the server shall also host the machine learning model and have an additional HTTP POST endpoint to receive videos from the application. Then, the Translator can process the video and output a translated message. This is done to make the app as universal as possible by minimizing the required processing power of the device and thus making it accessible from both the newest as well as older Android devices with similar performance across all of them..

8.2.3.3) Android Application

In order to meet the set goal by our sponsor to be portable and accessible, our application shall be developed on the Android platform, specifically Android OS V9 (Pie). This is due to the widespread use of Android as well as its distribution store having less strict standards.

ASL Interpreter Mobile Application

Raymond Cook
Jadepan Thedthong

INTERFACE CONTROL DOCUMENT

REVISION – 2
30 April 2023

Interface Control Document
for
ASL Interpreter Mobile Application

Team 33

Approved by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2022/10/01	Raymond Cook		Draft Release
1	2022/12/04	Jadepan Thedthong		Semester Final Report
2	2023/04/30	Raymond Cook		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

List of Figures

9) Overview

This document aims to provide a brief description of how the subsystems in this application will interact with each other. The application is being designed to serve as a basic interface to allow users to quickly access what they need from it upon launch. The machine learning subsystem will be a back-end system that requires little to no direct interaction from the user. It will work similarly to an API where the app will take a video as an input and produce a translated message as an output. Screenshots of the UI will be shown to demonstrate how the user will navigate and use the application.

10) References and Definitions

10.1) References

Document Title	Revision/Release Date	Publisher
Kotlin Documentation	v1.7.20	Jetbrains
RFC 9110 HTTP Semantics	June 2022	IETF
LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past	Feb 2022	Towards Data Science

Table 1: References

10.2) Definitions

Graphical User Interface (GUI)	What users will use to interact with the application
Internet Engineering Task Force (IETF)	Working group to establish open standards for developing internet interfaces
Machine Learning (ML)	Computer-driven models that can take in an input, compare it to a known database, and generate the most likely output based on similarity
Key Points	Targets of interest for the ML model, namely the hands, fingers, elbows, shoulders, and facial expressions
Long Short-Term Memory (LSTM)	LSTM stands for Long Short-Term Memory which is a type of Recurrent Neural Network.

11) Android Application Interface

The application will contain an intuitive, low-profile UI to allow users to quickly access what they need. From the main menu, they will be able to access the two features of the app, the Translator and the lesson plan, as well as the user profile page. Upon loading the app, users will be greeted with a basic home screen after they log in that will showcase some basic stats of their usage as shown in Figure 1 below.

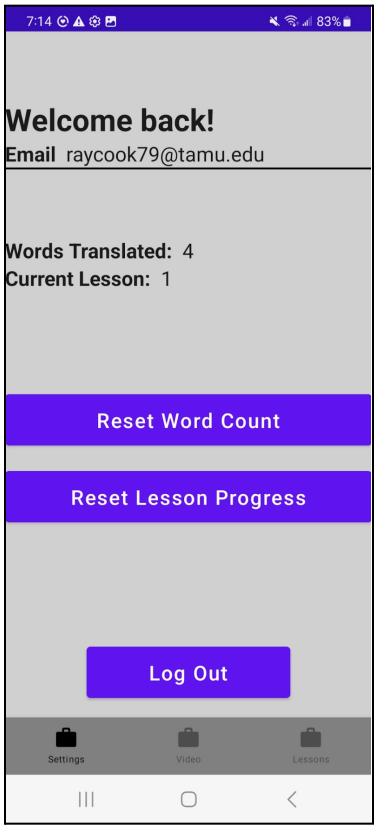

Figure 1: User Profile Screen

11.1) ASL Lesson Plan

When users tap the Lessons icon at the bottom navigation bar, they will see the following screen as shown in Figure 2. The following list shows all of the lessons that are available to the user as they swipe through it. However, availability is locked until the previous lesson has been completed. Each lesson's title will refer to the content contained within and once a user clicks "Begin Lesson", they will be taken to the lesson screen and be presented with a little written description and explanation of the topic. Additionally, each lesson will include visual aid in the forms of images and

video clips that will showcase the important points of the lesson as shown in Figure 3.

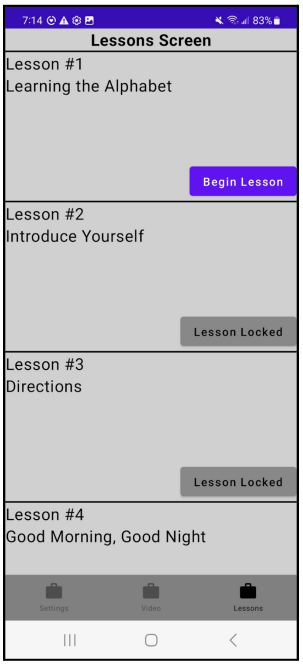

Figure 2: Lesson Plan Page

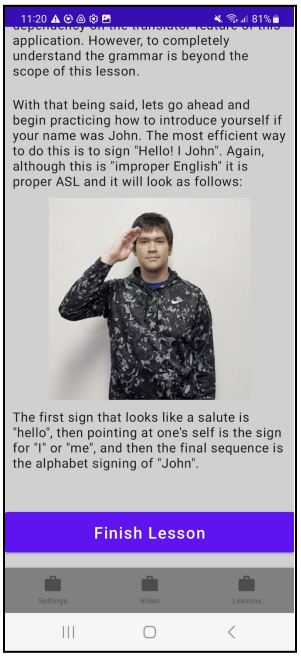

Figure 3: Content of Lesson #2

11.2) Application-Side Translator Module

For users wishing to translate signers in a quick amount of time, they will access the Translator portion of the app. From here, users will be prompted to record the signer so that the app can upload the video to the ML server for off-site processing. After the model has been run, the ML server will then respond with a string of the corresponding English translation that the app will display to the user. Since each video will only consist of a single sign, the processing time of the server will be kept to a minimum and the network bandwidth needed to upload should also be minimal. Per our test, the average sign results in a 3-4MB video file which should be able to be transmitted over most internet connections fairly quickly.

Figure 4: Image of the Translator Module

12) Overall ASL Translator Interface

The Translator screen will provide an interface between our backend server hosting the machine learning models and our Android application through the utilization of the HTTP POST method. This POST will be used to send a video to the server and then use a second HTTP GET request to receive the translated result from the server.

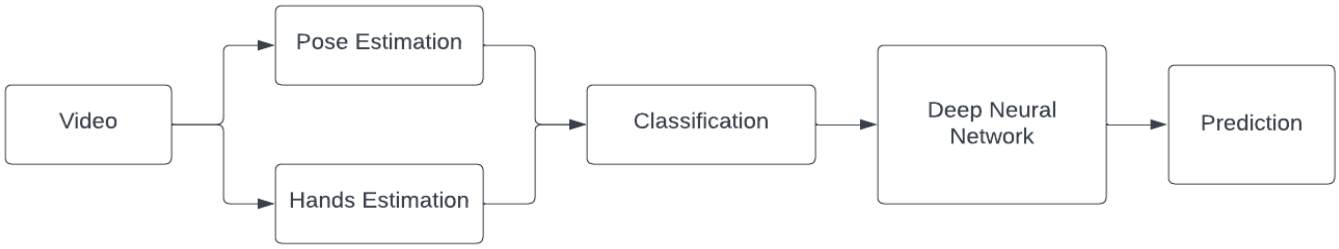

Figure 5: Word-Level Machine Learning Subsystem

12.1) Key Points Detection

This is the first step of the translation process where the server receives a video via HTTP POST and can estimate the key points. The workflow uses the mediaPipe library to detect the key points and output the appropriate key points relative to the frame dimensions: x-axis, y-axis, and z-axis in a numpy format. The x-axis and y-axis are the keypoint coordinates normalized to the range of 0 and 1 by the frame width and height respectively. The z-axis is the key-point depth relative to the midpoint of the hips.

For our model, some examples are the first keypoint being 0 for the nose, 12 being the left shoulder, and 13 being the right shoulder. It will then stream a series of these key points into our LSTM neural network for further processing and eventual translation.

12.2) Word-Level Translation Model

This model, as stated, will receive the key points from the Key Points Detection system. It will then feed these key points as an input to our deep neural network. After performing the translation and prediction, it will transmit the translated sign to the application via HTTP GET. The output of the model itself is an integer that is mapped to its associated word. For example, if the output is 0 the corresponding word is “hi” while 1 stands for “bye.”

ASL Interpreter Mobile Application

Jadepan Thedthong

MACHINE LEARNING SUBSYSTEM REPORT

REVISION – 1

30 April 2023

Machine Learning Subsystem Report
for
ASL Interpreter Mobile Application

Team 33

Approved by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2022/12/4	Jadepan Thedthong		Draft Release
1	2023/04/30	Jadepan Thedthong		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

List of Figures

1) Introduction

The machine learning subsystem is designed to receive a video input, specifically an MP4 file, and then translate the video into a string of text stored for the server to retrieve when needed. The model works by extracting important elements such as the body, arms, hands, and face and uses these key points to predict a word associated with the respective sign.

2) Machine Learning Subsystem

2.1.1) Key Points Extraction and Classification

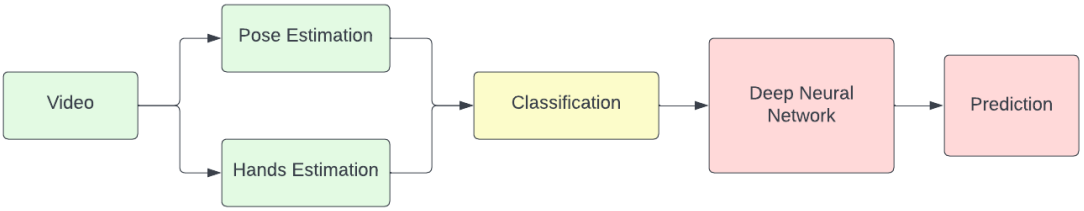

Figure 1: Word-Level Machine Learning Subsystem

The goal of the subsystem is to translate ASL at a word-level basis. Initially, the model loads a video using OpenCV which is analyzed frame-by-frame for key points that are then extracted. These extracted points are full-body key points, including for example facial key points and limb key points using the MediaPipe from Google. In addition, the mediaPipe also uses frame timestamps to ensure that the key points are synchronized with the input video stream. Then it goes into the classification by retaining only the essential key points by ignoring key points like feet and legs that do not have an impact on ASL. The detailed classification is listed below in the following figures and tables.

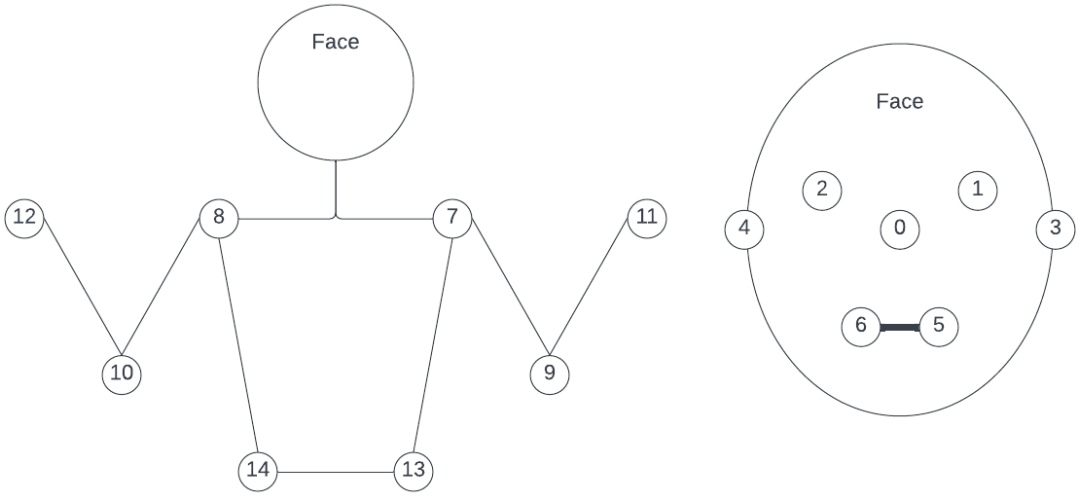

Figure 2: Pose Key-Points

0. Nose	8. Right Shoulder
1. Left Eye	9. Left Elbow
2. Right Eye	10. Right Elbow
3. Left Ear	11. Left Wrist
4. Right Ear	12. Right Wrist
5. Mouth (Left Side)	13. Left Hip
6. Mouth (Right Side)	14. Right Hip
7. Left Shoulder	

Table 1: Pose Key-Points Labels

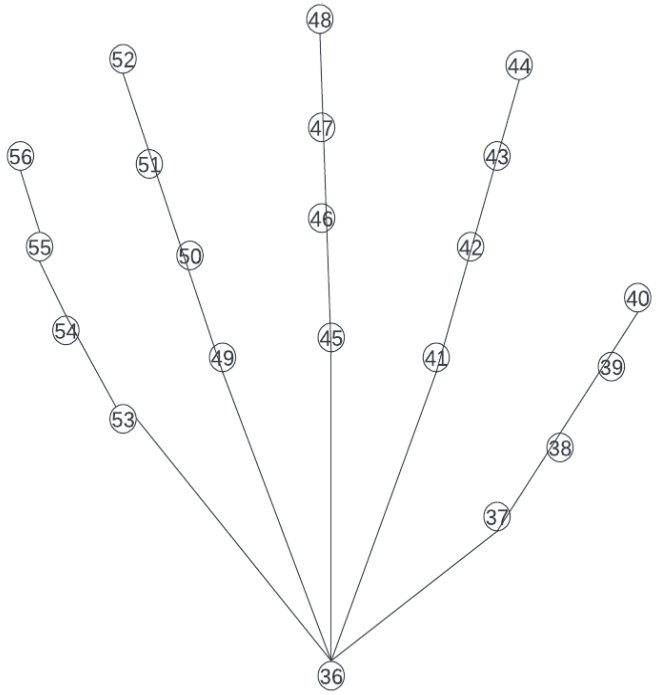

Figure 3: Right-Hand Key Points

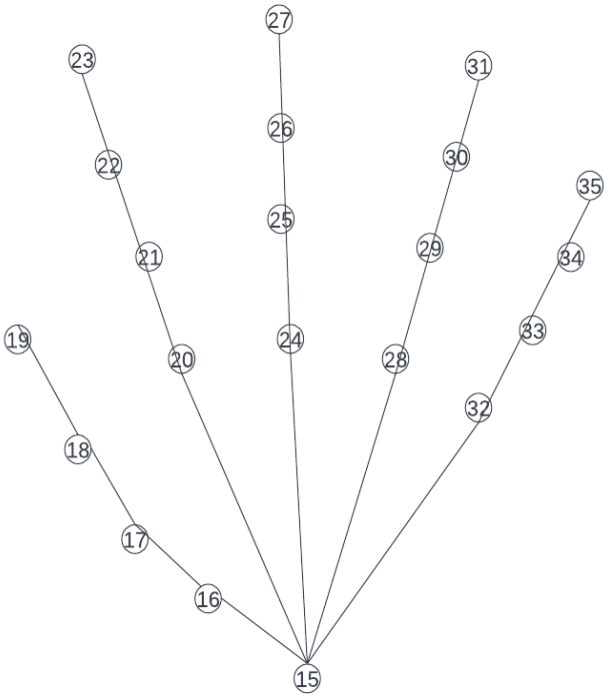

Figure 4: Left-Hand Key Points

15. LH_wrist	29. LH_ring_pip	43. RH_index_dip
16. LH_thumb_cmc	30. LH_ring_dip	44. RH_index_tip
17. LH_thumb_mcp	31. LH_ring_tip	45. RH_middle_mcp
18. LH_thumb_ip	32. LH_pinky_mcp	46. RH_middle_pip
19. LH_thumb_tip	33. LH_pinky_pip	47. RH_middle_dip
20. LH_index_mcp	34. LH_pinky_dip	48. RH_middle_tip
21. LH_index_pip	35. LH_pinky_tip	49. RH_ring_mcp
22. LH_index_dip	36. RH_wrist	50. RH_ring_pip
23. LH_index_tip	37. RH_thumb_cmc	51. RH_ring_dip
24. LH_middle_mcp	38. RH_thumb_mcp	52. RH_ring_tip
25. LH_middle_pip	39. RH_thumb_ip	53. RH_pinky_mcp

26. LH_middle_dip	40. RH_thumb_tip	54. RH_pinky_pip
27. LH_middle_tip	41. RH_index_mcp	55. RH_pinky_dip
28. LH_ring_mcp	42. RH_index_pip	56. RH_pinky_tip

Table 2: Hands Key-Points Labels

Since our model does not need all of the landmarks generated from the MediaPipe, above is the list of the key points that are essential for ASL.

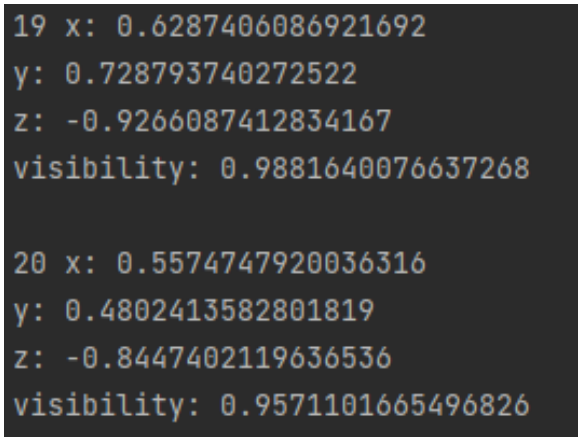

Figure 5: Example Output

As stated in the ICD, The key point coordinates are normalized to the range of 0 and 1 by the frame width and height, respectively, and are represented by the x-axis and y-axis. The key-point depth in relation to the hips' midpoint is the z-axis.

The key points are extracted and stored as a 2D numpy array where each line represents a frame of the video and in each line is an array of key points. This process helps accelerate training and lower memory consumption during the training period. Instead of loading all the video samples, extracting the key points, and using those key points to train, the model will train on the pre-processed key points instead.

2.1.2) Translation Model Implementation

The machine learning model consists of three layers of bidirectional LSTM, two dropout layers, one dense layer, and one output layer. The model is trained using a collection of .npy files that stores the key points of the signer as detailed above in section 2.1.1. The dataset used to train the model is built by recording two hundred

videos per word with a total of forty words being trained for the project. Then, a data augmentation process I developed is used to increase the sample size to five hundred videos per word. This data augmentation process includes modifications to each video such as horizontal flips, randomly rotating -5 to 5%, and modifying the video's speed by a factor ranging from 70-130% speed. This helps expand the model's dataset with 'new' data that helps in preventing overfitting and reducing data bias that we have encountered before with a lower sample number. The final model used in the project was its 12th iteration and took forty hours to train, with around 15 minutes per epoch for eighty epochs. We also implemented early termination and checkpoints procedures to reduce wasteful processing if the validation accuracy was not improving for ten successive iterations.

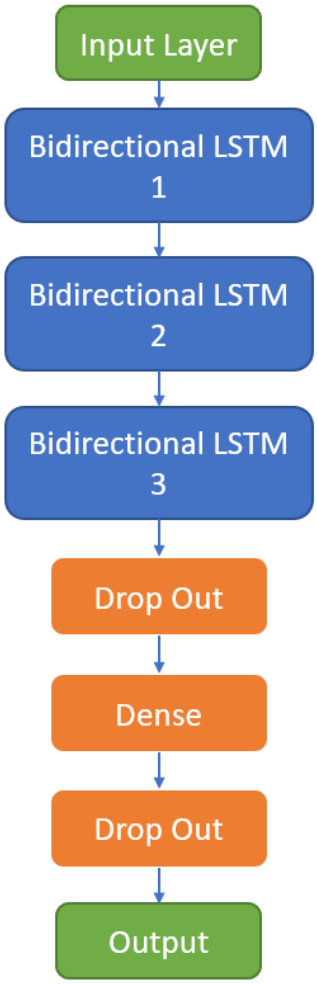

Figure 6: Model Layout

The input layer takes in the key points numpy array with the shape of (93, 186) where 93 is the maximum number of frames in a video in the dataset and 186

represents the key points dimensions. The first bidirectional LSTM layer contains 32 LSTM units within a bidirectional wrapper, which helps the model process sequence data in both forward and backward directions. The second bidirectional LSTM layer is similar to the previous layer but instead with 64 LSTM units. This additional layer helps the model learn more complex patterns that are present in the data. The third and final bidirectional LSTM layer again contains 32 LSTM units, like the first layer and serves to help the model capture and learn complex patterns, reinforcing the second layer. The dropout layers help reduce overfitting by preventing the model from relying heavily on individual neurons. The dense layer is a connected layer of 32 units with a Rectified Linear Unit (ReLU) activation function and helps to introduce non-linearity into the model. Finally, the output layer produces a probability of the translation. Finally, we use the soft-max activation function to select the translation with the highest probability.

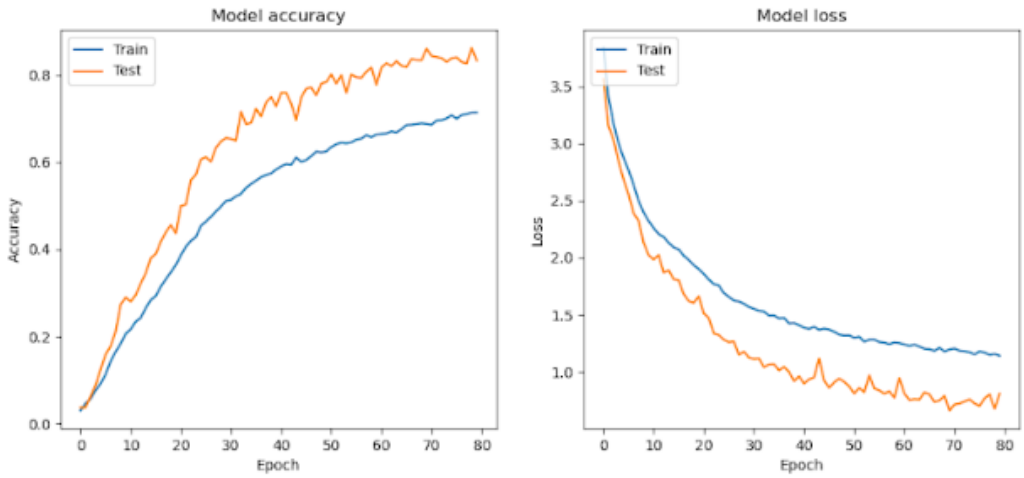

Figure 7: Training Process

2.2) Translation Integration

The translation algorithm is a Python script that takes in 2 arguments: an input file and an output directory for use in integrating with the server. The program will read the video using OpenCV and extract the key points using the MediaPipe library. The extracted key points will then be passed to the machine learning model which will save the prediction to a text file in the desired destination with the same file name as the input video to allow the application to pull the proper translation when needed.

2.3) Word-Level Machine Learning Validation

2.3.1) Evaluation Metrics

The required metric is the accuracy of the model but however, it is not the best sole representation of the model's performance. Thus, we have also included the precision, recall, and F1 scores as well.

The precision score tells us about the accuracy of the positive predictions made by the model. It measures how many predicted positive instances are actually positive when compared to false positives, which in turn means that high precision scores also indicate a model with low false positive predictions.

The recall score tells us the ability of a model to correctly identify positive instances. It does this by measuring how many of the actual positive instances in the dataset are correctly identified by the model.

The F1 score is the harmonic mean of the precision and recall scores and provides a score that balances both metrics. Generally, a high F1 score indicates that the model performs well on both the precision and recall tests. Thus, it is another good metric to consider when evaluating the performance of the model.

2.3.2) Evaluation and Validation Process

The algorithm will run against a testing dataset consisting of 50 samples per word. This dataset has not been used during the training process, ensuring that a brand new dataset is used on the model. This method ensures that the model performs well with real-world data outside the training dataset used to build the model itself. The test bench will compare the predicted results with the actual label and store those scores to calculate the metrics mentioned above. The library used to evaluate the model is scikit-learn with its precision_score, recall_score, f1_score, classification_report, and confusion_matrix imports.

To evaluate the script used by the server, we ran the script 20 times with different videos to test the correctness of the prediction and the functionality of the script. We also tested with different resolutions and aspect ratios to ensure that the model would work on a variety of phones. We also tested the edge cases such as if a signer is even present in the frame to reduce the load of the server and ensure that the machine learning model only runs when there is a valid video being sent to the server. Also, if the confidence level is lower than a set threshold, the model would output an error back to the user stating as such and preventing false translations from being sent to the user.

2.3.3) Results

Metric	Specification	Result
Accuracy	60%	85.30%
Word Count	30	39

Table 3: Specifications and Results

Metric	Result
Precision	85.28%
Recall	85.30%
F1 Score	84.6%

Table 4: Additional Metrics

The validation results for the machine learning model show that our model is doing really well in the control environment with around 85.30% accuracy which meets the specification by our sponsor. Also, our model can translate thirty-nine words which surpasses the system requirement of thirty words. An interesting note is that the model was originally trained to translate forty words. However, there was some confusion during the dataset-building process. For example, the signs for “eat” and “food” are extremely similar and thus, the predictions became skewed and showed only “eat” when encountering them which explains the 0 F1-score for “food” and a low F1-score for “eat” in Figure 8.

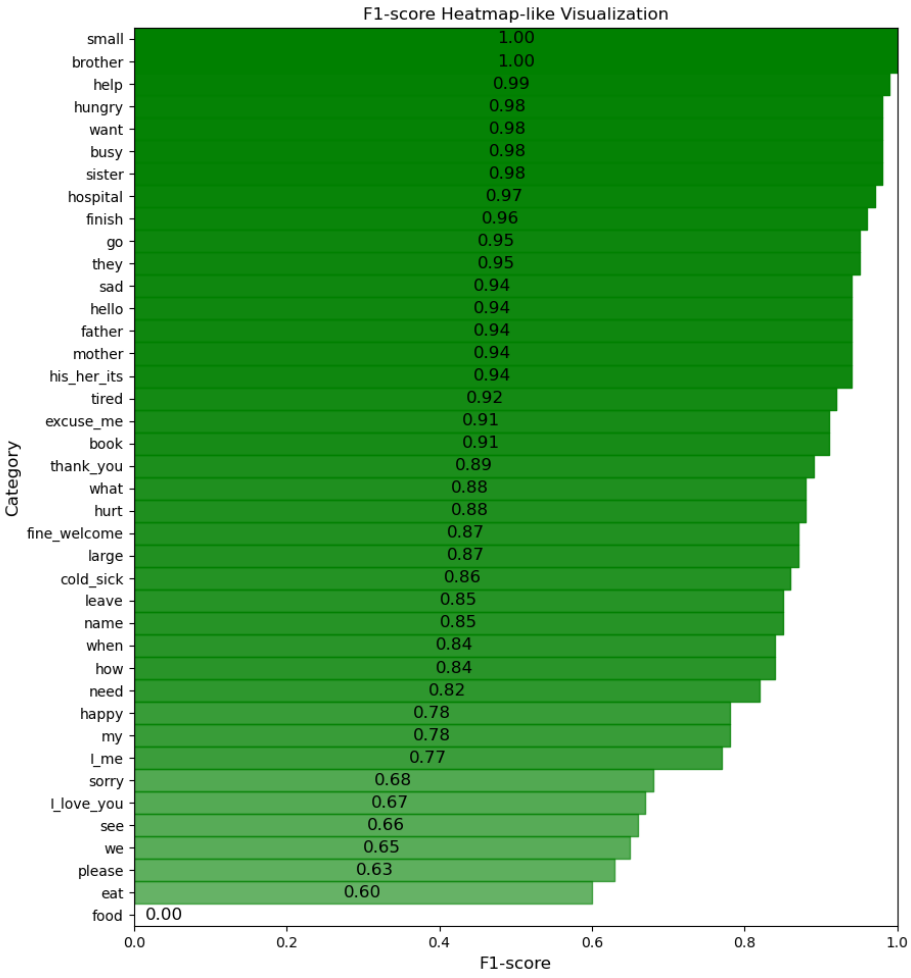

Figure 8: F1 Score Heatmap

3) Translation Machine Learning Conclusion

In conclusion, the machine learning model works in the control environment and meets all the specifications; however, the performance is slightly degraded in real-world applications. Thus, a larger dataset with more variety and adjustments to the model are needed for the future. We plan to implement a feedback option from users that will allow the dataset to expand with the user base. For example, when a user receives an incorrect translation, they

can input the correct translation into our application which can then be used with the extracted key points to train the model further.

ASL Interpreter Mobile Application

Raymond Cook

ANDROID APPLICATION/SERVER SUBSYSTEM REPORT

REVISION – 1

30 April 2023

Android Application/Server Subsystem Report
for
ASL Interpreter Mobile Application

Team 33

Approved by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2022/12/04	Raymond Cook		Draft Release
1	2023/04/30	Raymond Cook		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

List of Figures

1) Introduction

The Android Application subsystem allows the user to navigate to each element of the system, which is the user profile page, translator module, and the ASL lesson plan. The application was written in Kotlin, in large part due to the use of ‘Composables’ from Jetpack Compose, through the Android Studio IDE. Kotlin was chosen for this application due to Android encouraging developers to transition their applications to Kotlin from Java. Android has also created a new UI toolkit named ‘Jetpack Composables’ that allows developers to quickly create efficient and effective UI’s without the need for XML, simplifying a project’s workflow.

2) GUI Implementation

With Composables, XML layout files are no longer needed and creating applications adaptable to varying screen sizes and orientations is a very routine process. A navigation graph is created that helps the application organize which view to display to the user by passing its pointer to each Composable function as shown in Figure 1. By doing this, the graph has essentially become a ‘global variable’ and further navigation throughout the app was easy to accomplish using this approach.

```
@Composable
internal fun RecordingScreen(
    navController: NavHostController,
    factory: ViewModelProvider.Factory,
    viewModel : MainViewModel,
    recordingViewModel: RecordingViewModel = viewModel(factory = factory),
    onShowMessage: (message: Int) -> Unit
) {
    val state by recordingViewModel.state.collectAsState()
    val context = LocalContext.current
    val lifecycleOwner = LocalLifecycleOwner.current

    val listener = remember(recordingViewModel) {
        object : VideoCaptureManager.Listener {
            override fun onInitialised(cameraLensInfo: HashMap<Int, CameraInfo>) {
                recordingViewModel.onEvent(RecordingViewModel.Event.CameraInitialized(cameraLensInfo))
            }
            override fun onProgress(progress: Int) {
                recordingViewModel.onEvent(RecordingViewModel.Event.OnProgress(progress))
            }
            override fun recordingCompleted(outputUri: Uri) {
                recordingViewModel.onEvent(RecordingViewModel.Event.RecordingEnded(outputUri))
            }
            override fun onError(throwable: Throwable?) {
                recordingViewModel.onEvent(RecordingViewModel.Event.Error(throwable))
            }
        }
    }
```

Figure 1: Code Section Demonstrating Navigation Controller

For example, each tab on the bottom is implemented as its own Composable function and the main program loop works by having the navigation controller call each function when tapped. The images in Figure 2 show what is currently being used in the final product for this project. The GUI is finished with all features meeting the original requirements of the project. That includes being intuitive and easy to navigate, as well as leading the user to each page freely and incorporating the translator module and ASL lesson plan seamlessly.

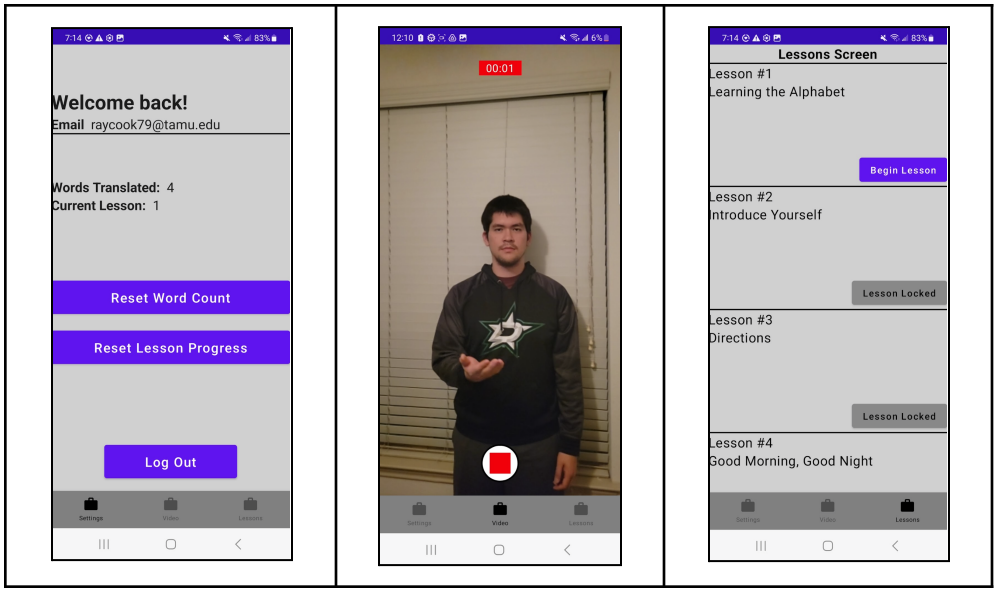

Figure 2: Each Screen as Seen when Authorized

3) Account Login System and Database

The account login and database system is currently being implemented through the use of a third-party service named Auth0. This service provides a complete account registration and login system along with metadata storage which was not used for this project. Figure 3 displays the workflow and appearance of the application as a user logs in.

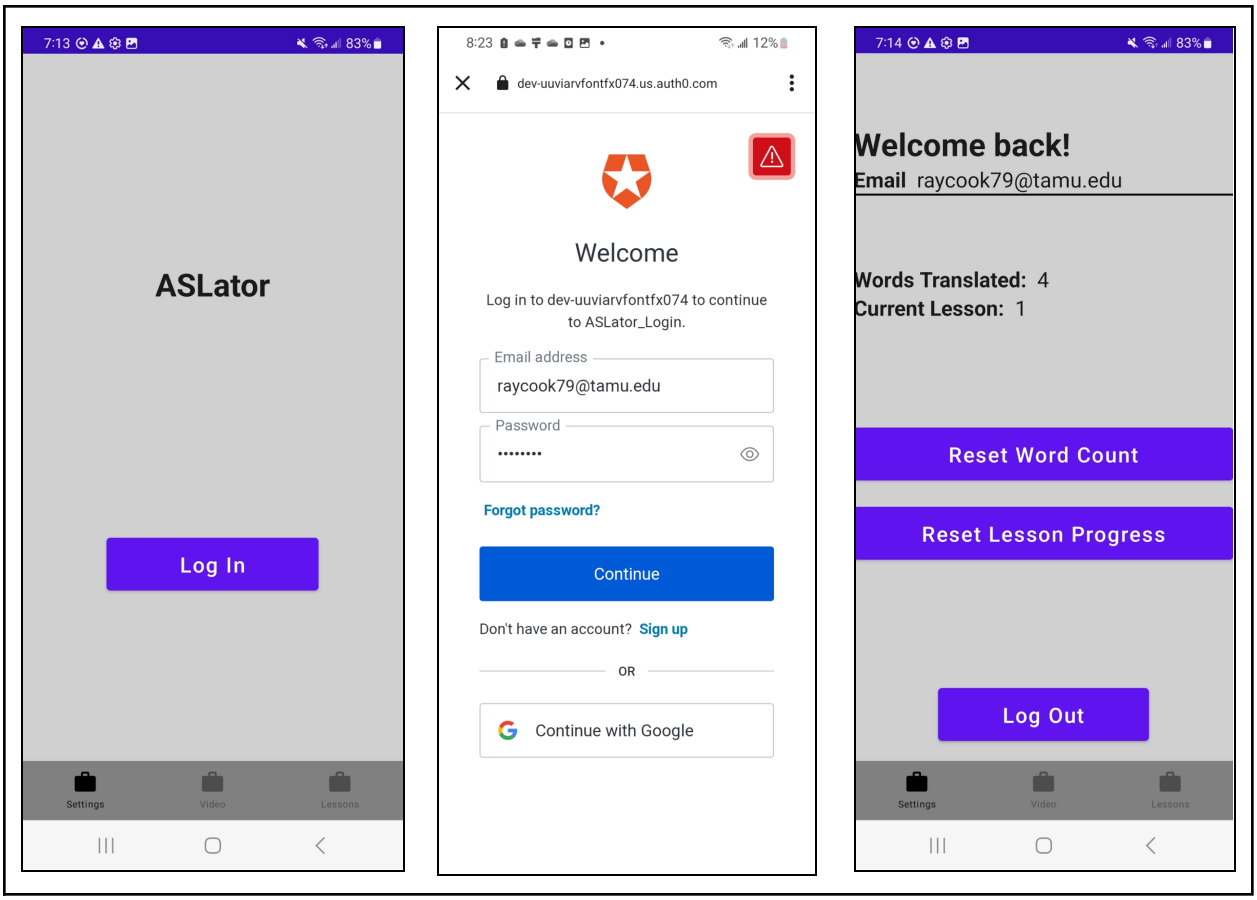

Figure 3: Application Login Workflow

Once a user enters the application, they will be brought to the ‘Settings’ tab which has the login page. If a user does not have an account, they will be able to register for one here as well. After they login, basic information is shown here, such as what lesson they are on and how many words they have translated total. These metrics are stored on the server and updated using HTTP GET requests from the application.

4) Video Recording and Uploading

The Translator screen utilizes the camera on the back of the device to record videos of signers and upload them to the server. As long as the video includes the upper half of the signer, the model will be able to process the video and attempt to generate a translation. The application also allows users to playback the video to ensure they are satisfied with the quality of the video and avoid having to generate a translation and try again after realizing the video did not capture the signer correctly. This video is uploaded to the server using a HTTP POST method and retrieves the translation using a HTTP GET method. All endpoints on the server were created to facilitate this design and guarantee that user’s videos would not get mixed with others.

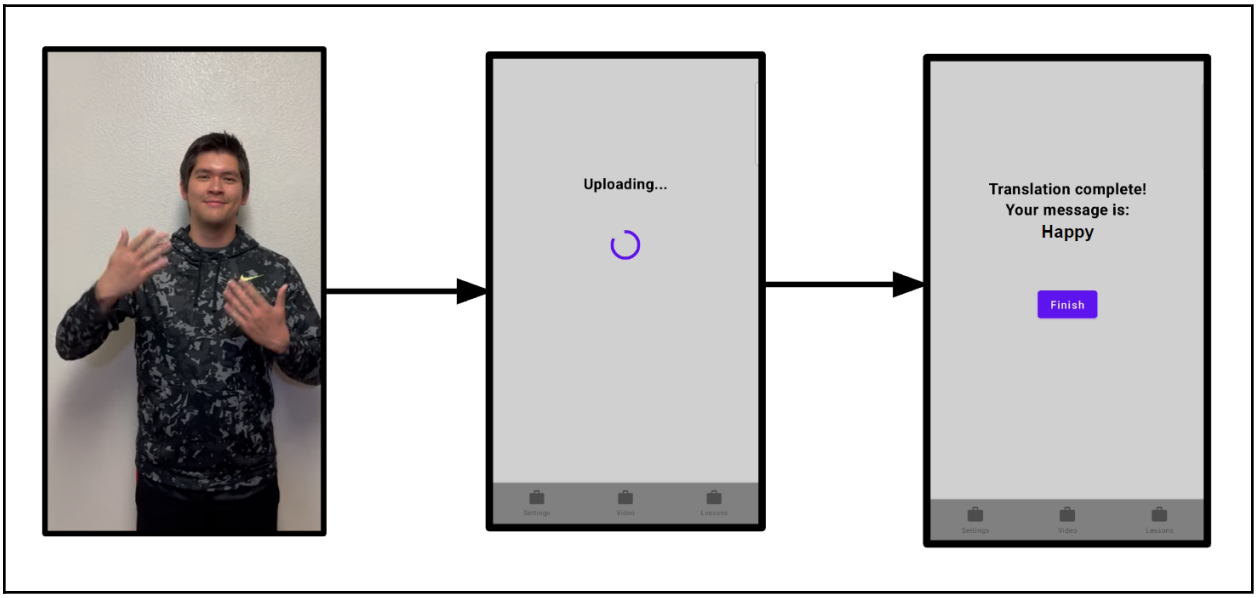

Figure 4: Workflow of the Video Upload Process

Ideal positioning of the signer would be to capture their entire body in a well-lit area from a frontal view. Once the signer is ready, the person recording the video can then signal the signer to start, begin the recording, finish the recording when the signer is done, and then upload it to the server. A tap-to-start and tap-to-finish was decided as the better solution versus a hold-to-record due to convenience and flexibility in holding the mobile device.

5) ASL Lesson Plan and Curriculum

The ASL lesson plan has been developed to help the user learn the basics in an effort to allow them to communicate on a fundamental level. The goal is not to educate a user to a high level of proficiency in a short period of time but rather to allow the user to know enough to be somewhat independent of the application. In order to achieve this goal, material was selected that would help someone to participate in general conversation, such as asking for directions, introducing oneself, ordering food, and so forth. The general flow of the material was influenced by previous language courses I had taken, particularly in Spanish. Ten lessons have been completed in the ‘Compose’ style with the ability to easily create more lessons if desired. All assets in the lessons are loaded from the server when selecting a lesson to reduce the installation size needed on the application.

Lesson	Purpose
Lesson 1: Learning the Alphabet	ASL relies on the alphabet to sign names and other indicators. In the event that a signer does not know the specific sign for a word, they can always fall back to spelling it.
Lesson 2: Introduce Yourself	The most important part of a conversation is knowing who you are talking to. This builds on Lesson 1 by showing how to introduce yourself by spelling your name as well as teaching the user their first signs to complete the sentence.
Lesson 3: Directions	Since the purpose of these lessons are to provide the user with a general sense of ASL, learning directions made logical sense to be the next topic. It once again expands on the concept of names as well as expanding the user's sign vocabulary.
Lesson 4: Good morning, Good Night	This topic is the first to not rely upon the alphabet and instead will be purely conversational using signs.
Lesson 5: When Duty Calls	An important thing to know no matter where you go is where the nearest restroom is. Combining this with material learned in Lesson 3 leaves the user with enough knowledge to find a bathroom.
Lesson 6: General Emotions	A common topic for conversations might revolve around how one is feeling. This lesson shows the signs for a couple of emotions to get the user started on recognizing a few of these.
Lesson 7: Talking About the Weather	Weather is useful to communicate to anyone, whether it's cold, rainy, sunny, or anything else. This lesson showcases a few of these signs and we begin to see the 'contextual' part of ASL in a more highlighted fashion.
Lesson 8: Ordering Food	Food is needed by everyone, so knowing how to ask people for what you want is important to recognize.
Lesson 9: Lets Talk Sports	Another common pastime for most people is to either participate in or spectate different sports.

Lesson 10: Graduation	A fitting end to the lesson plan, this final segment highlights the word graduation as well as how to specify what degree a person obtains.

Table 1: Current ASL Curriculum

When a user is on the Lessons tab, they will be presented with a scrolling list of lessons. Access to later lessons will be denied until they have completed the previous lessons to ensure that the user is learning the material properly. Once a user starts a lesson, they will be presented with the information pertaining to that particular lesson's topic as well as questions and illustrations to help the user visualize and digest the information. Once the user completes the lesson, their user metadata will update accordingly and the next lesson will now be available to them.

6) Server

Arguably the most important part of the entire project, the server ties everything together and allows communication between the application and the machine learning model. For this project, the server was built using a WAMP (Windows, Apache, MySQL, PHP) stack hosted on a local machine. Python was also implemented into the backend in order to support and run the machine learning model to generate the translations.

Multiple endpoints were created to ensure smooth communication between the application and the server. Instead of trying to create an application that required constant internet connection, I instead developed an application that only needed to check in with the internet connection at certain times in the application. For example, when logging in, an initial check is performed and the proper credentials and user data returned to the application. Afterwards however, the application does not have to connect to the internet again until loading in assets for the lesson plan or uploading a video to the server for translation purposes. See Table 2 below for a list of the endpoints and their purpose.

Endpoint	**Type**	**Parameter(s)**	**Purpose**
/set_lessons.php	HTTP GET	file = Email of user count = New value of lesson progress	Used to update the user's progress in the lesson plan using the email as a unique identifier (UID) and count being the new lesson progress

/get_lessons.php	HTTP GET	file = Email of user	Used to retrieve the lesson progress upon logging in to re-establish lesson progress
/set_words.php	HTTP GET	file = Email of user count = New value of words translated	Used to update the total number of words translated by the user using email as the UID and count as the new value
/get_words.php	HTTP GET	file = Email of user	Used to retrieve the number of words translated by the user to display it to the user
/translation.php	HTTP GET	file = Email of user	Used to retrieve the translation of the signed word after being run through the model using the user's email and timestamp of the video to retrieve the proper translation
/upload.php	HTTP POST	file = Multipart name = Name of video tmp_name = UID generated when video is recorded	Used by the application to upload the signer's video to the server to be processed through the ML model

Table 2: Endpoints of the Server

In order to process the signer's video and return the translation to the application, I developed a workflow that utilizes a user's email to serve as a UID and the timestamp to ensure uniqueness across the user's videos.

To begin with, the application uploads the video to the server using the upload.php endpoint which receives the video as an MP4 file in byte form. This video is stored in the server using a combination of the user's email and the timestamp at which it was sent from the application. The application then uses the translation.php endpoint to request the translation back from the server, which is stored in a generic text file with the same title as the video. The application then receives the translation stored in the text file and can display it to the user, completing the process.

7) Validation Plan

The following criteria and test methods were used to measure and validate the progress of the project as described in Tables 2 and 3.

Criteria	Test Method	Result
Effective GUI that is intuitive and stable	Navigate to each tab and explore all options	Tested on a physical Samsung S21 and a virtual Google Pixel 3 with no crashes, all options less than two taps away
Account System Implemented with Associated Data	Log-in to an existing and register a new account	Logging in through Auth0 initiates a connection with the local server to transmit data to and from the application
Video Recording	Record a video and play it back to the user before uploading (if desired)	Fully implemented
Video Uploading	Upload video to server and verify proper transmission	Implemented using HTTP POST
ASL Lesson Plan Curriculum	Teach the user basic ASL from the alphabet to general topics common in daily life	All lessons have been fully created along with accompanying visual aids

Table 3: Validation Methodologies

Items	Specification	Actual
Space Required on Disk	< 100MB	22.70MB
RAM Usage	< 200MB	157MB (Maximum)
Time For Translation	<= 15 Seconds	9.87 Seconds (Average)
ASL Lesson Plan	10 Lessons	Accomplished
User Data Storage	Server-based, cross-device	Accomplished

Table 4: Application Criteria

8) Conclusion

The application and server subsystem have reached a functional stage that is worthy of an initial deployment to the general public. All features have been successfully implemented and integration with the machine learning model has been successfully tested. The design of having the server store the assets allows the application itself to maintain a very small installation size while still providing the visual aids necessary to make the lessons efficient. This also allows future updates to be made to the assets as well as the machine learning model without impact to the application itself.

9) Future Plans

As with any product, future improvements can always be made. The server is currently running off of a WAMP stack on a desktop computer. Transitioning to a cloud-based service with scalable processing power will increase the reliability of the system as well as give it the resources necessary to handle the high spikes in computational resources needed when being fed videos to translate.

Another improvement is to implement a user feedback option when receiving translations. For instance if the translation received is incorrect, the user would be able to report it as incorrect and send the proper word back to the server. Once verified by staff, this video and word could be stored and used to expand the dataset as the user base grows to evolve the model as time goes on. This would be the best way to naturally grow the dataset to include different camera angles, lightings, backgrounds, number of people, and even the ‘accents’ of signers because similar to how people from different backgrounds may say certain things slightly differently, the same difference exists in ASL.

ASL Interpreter Mobile Application

Raymond Cook
Jadepan Thedthong

SYSTEM REPORT

REVISION – 1
30 April 2023

System Report
FOR
ASL Interpreter Mobile Application

Team 33

Approved by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	2023/03/15	Raymond Cook		Draft Release
1	2023/04/30	Raymond Cook		Revisions for Spring 2023 Final Report

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1) Overview

The project sponsor's original request was that of a system that would utilize machine learning to scan and translate anyone signing ASL to English text in order to increase the quality of life for people that must use ASL to communicate on a daily basis. One of the main constraints the sponsor presented was that the solution must be widely accessible to the general public and provide relatively quick translations. With that in mind, the final version of the system developed can be seen in Figure 1 below as a fully developed Android application that communicates with a remote server that serves as data storage as well as houses the machine learning model.

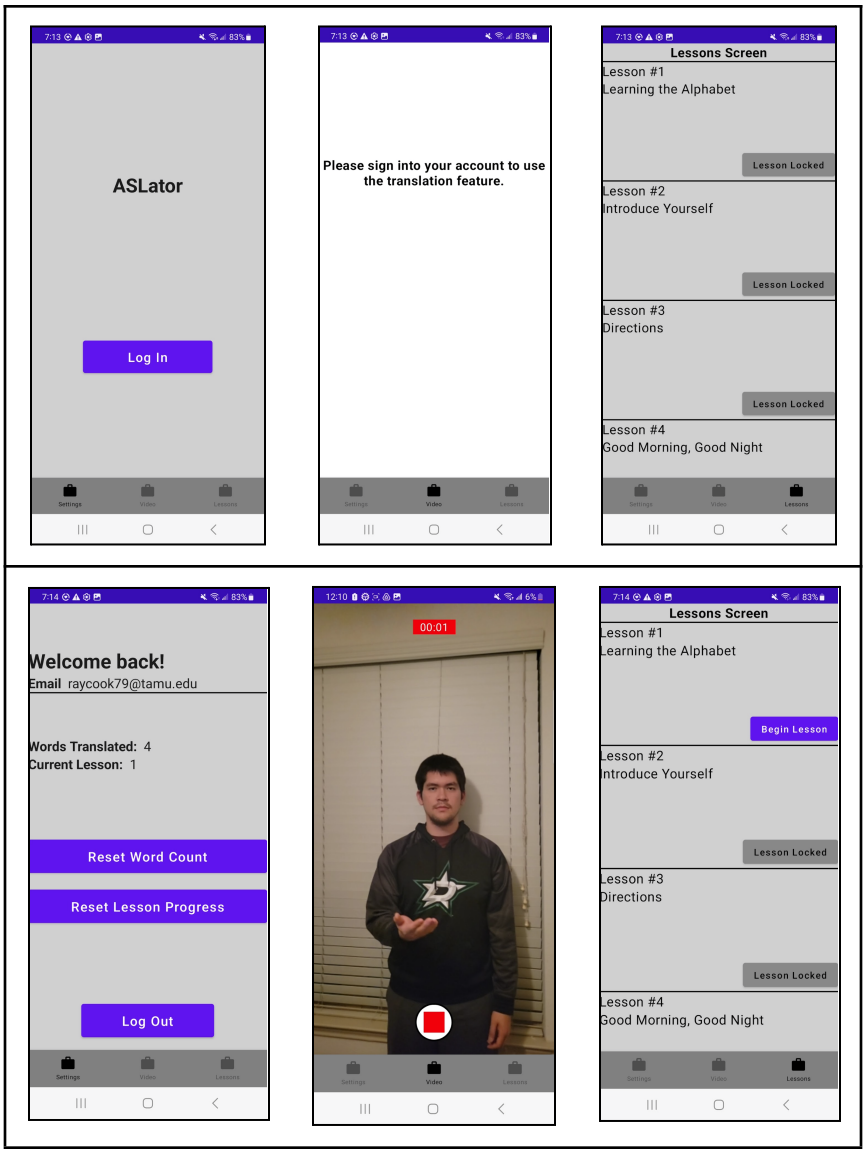

Figure 1: Images of Application when Unauthorized and Authorized

There are three main portions to the application, the first of which being the account login/registration page. This is required to ensure that the user's data can be stored and transferred across multiple devices as well as send videos to and from the server which is where the machine learning model resides.

The second main portion of the application is the camera recording portion which is the main feature of the application. Here a user can take a video of a signer, playback the video to ensure quality, and then upload it to the remote server where it is then run through the machine learning model to generate an English translation that is then sent back to the user on the application.

The final portion of the application is the ASL lesson plan which provides basic information to the user in an attempt to teach them to recognize and be able to perform common words and phrases. The goal was not to teach the entirety of ASL since it was designed only to be able to increase effectiveness quickly without any foundational knowledge of ASL.

The final implementation of the project consisted of two main subsystems: the Android Application and Remote Server subsystem and the machine learning model subsystem. The Android Application provided the user with an intuitive interface that was simple and quick to navigate, allowing them to be able to use it effectively in the world. The server handled all data storage and processing, such as storing account data (words translated, lesson progress) and running uploaded videos through the machine learning model to be processed and translated. The machine learning model subsystem runs each video through two passes, the first of which extracts the wireframe skeleton of a signer as a collection of key points and the second that analyzes said collection to recognize gestures and produce a translation from ASL to English. This system report will go over the main design plan, major changes and decisions that were made, and overall critical accomplishments both for the project in terms of integration as well as breakthroughs in each subsystem.

2) Development Plan and Execution

2.1) Design Plan

The original design for the system was to create a product that could translate ASL at a sentence-level to accelerate entire conversations. Unfortunately as development continued on the project, we realized that this would be too difficult of a task for the time given on the project. After talking to our sponsor, a scope reduction was agreed upon that lowered the goal from sentence-level translation to word-level translation. The overall navigation map of the application as well as how it interfaces with the server can be seen in the figure below.

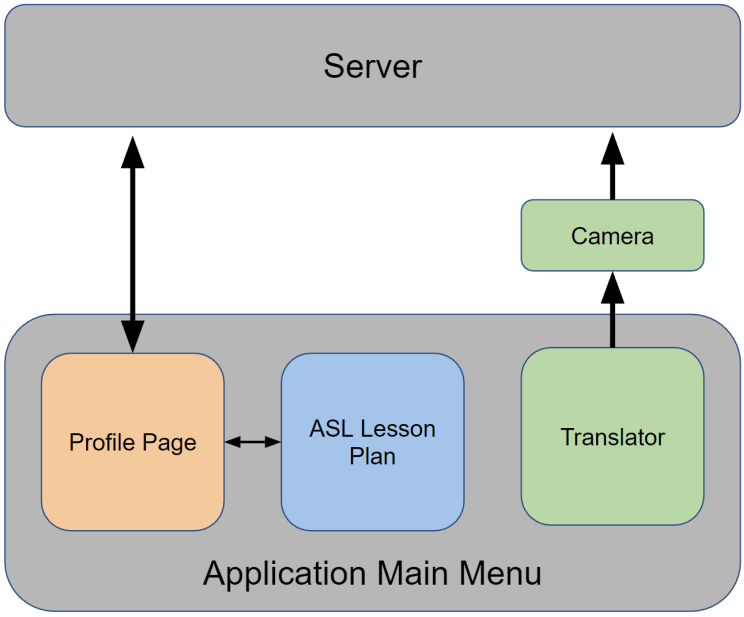

Figure 2: Basic Diagram of the System

Integration of the subsystems was fairly simple due to the structure of the backend of the server with only a few minor setbacks.The most challenging part of developing this system was the creation and polishing of each individual subsystem.

2.2) Execution Plan

The first three-quarters of the project focused exclusively on development of each of the subsystems. It was during this time where we realized that some of our initial goals and approaches were going to cause our project to be behind on schedule,

leading to our scope reduction with the sponsor. After the scope reduction, the subsystems were finalized quickly and work began on integration.

Much of the time spent in the first half of the project was researching the fundamentals of ASL, Android application development, as well as the different machine learning model networks that could be used. After much trial and error, the final decisions were made to use Kotlin/Compose for the application and a LSTM model for the machine learning subsystem. Throughout this process, constant assistance and advice from the sponsor and TA's helped us to make critical design decisions in a timely manner to where we could still complete the project in the original timeframe.

The second half of the project is where most of the development was done and subsystems were finalized. After developing the proper network functions into the application, the rest of the work was more of the research and development line of work, especially to design the ASL lesson plan. For the machine learning model, most of the work was dedicated to creating a dataset capable of training a model for the ASL translations. Since it proved rather difficult to find an open-source dataset that contained enough samples of each word to train a model sufficiently, the decision was made to lower the vocabulary count so that the dataset could be made in-house.

Due to the nature of being a software project, integration was fairly seamless once the proper endpoints were established on the server. All of the validation checks aside from uploading the video from the application to the server were independent of the cohesive system, allowing independent development for much of the project.

2.3) Validation Plan

As mentioned earlier, most of the validation checks happened in each subsystem individually. However, there were still a few key components of validation that had to occur on a system or rather, a product level. Since this application relies heavily upon a network connection, it was imperative that we include a few screens so the user could troubleshoot issues if any arose. The following figure showcases the current error screens that we have in the app that can help determine whether the error came from the client side, ML model side, or server side.

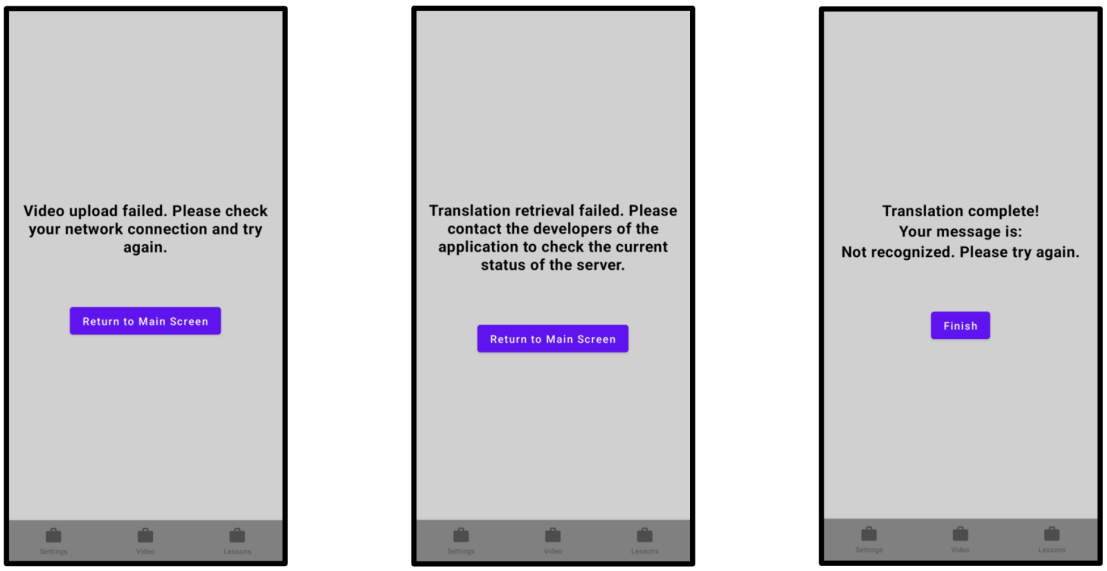

Figure 3: Basic Error Messages

This error handling system, although simple, provides a clean way for the app to fail gracefully without crashing and still be informative to either the user themself or the developer team. Other validation methods included checking the video files received on the server for resolution, frame rate, and file name to ensure that no quality was lost during the HTTP POST process.

3) Conclusion

The final product developed is an Android application that can run on most devices given that they are running at least Android OS 9 (Pie), have a functional camera, and an active internet connection. This application allows the user to log in to their account or register if needed, generate translations from signers, and gain access to an ASL lesson plan. During the course of this project, we both learned many valuable lessons about planning, flexibility, and perseverance due to the unique nature of the project which, beyond the engineering aspect, required us to also have enough knowledge of ASL to produce a product that would be beneficial to the real world users.

3.1) Future Work

One of the first things to improve the project if we were to continue development would be to generate a more varied dataset. The current datasets we have for both training and validation were made in a static environment with one camera angle, one lighting situation, and so forth. While this works really well in a “studio” approach, it does not have as much accuracy in the real world. To fix this, a photogrammetry rig with at least nine cameras attached to take videos from nine different angles simultaneously would assist in expanding the dataset. Other ways to improve the variety of the data would be to include multiple signers to develop the dataset as well as different lightings and backgrounds.

Another feature that would greatly benefit the product moving forwards would be to incorporate user feedback on the translation module. There is no better way to collect real world data to train the model than to collect data from the real world. By allowing users to add feedback, they could correct erroneous translations as well as add words that are not officially supported by the model yet. After going through a verification process to prevent falsely training the model, these new videos sent by users would allow us to continuously train and evolve the model to increase its accuracy and vocabulary.

The most important improvement would be to shift the server backend from a local machine to a cloud-based service. This would provide the system with redundancy in the event of hardware failure and the ability to scale processing power as needed if multiple people were to request translations at once.