

Capstone Project Phase B

**SignQuest**

Project No. 24-1-D-7

**Supervisor: Dr. Reuven Cohen**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Email** |
| Guy Shlomo Banbo | 316139310 | guybanbo@gmail.com |
| Elior Malik | 204361505 | elior676@gmail.com |

[Github Link](https://github.com/guybanbo/SignQuest.git)

**Contents**

[Abstract 3](#_5blhe4s9m6zv)

[1. Introduction 4](#_26zkn5ghze4a)

[2.Project description 6](#_ekdpfi4u3eih)

[2.1 Model and Teaching method 6](#_yl5q3tbjh7ig)

[2.2 Admin dashboard 7](#_2wv1kajpb4uq)

[2.3 Use Case Diagram 8](#_wvxrqeexm3nm)

[2.4 Activity Diagram 9](#_edlqhv4hq3po)

[2.5 System architecture 10](#_oufifww55hkr)

[2.6 ASL Datasets 11](#_p7klvkgtxnd3)

[2.6.1 WLASL 11](#_6ig9kx4alcod)

[2.6.2 SignASL.org 11](#_hltqjea45tw9)

[3. Tools We Used 12](#_z7gaa1kemn9e)

[4. Description of the development process 16](#_5lzt1y5i6cqu)

[4.1 Recognition model development 17](#_xdeud5a3jxjm)

[4.1.1 Dataset 17](#_ocjba35yf3mq)

[4.1.2 Model improvement 17](#_egos44h9mcqs)

[4.2 Creating the user interface 19](#_k744q76mda0p)

[4.3 Results 20](#_p99pulmdbufc)

[4.3.1 Model Results 20](#_zadph8sx29xw)

[4.3.2 User Testing and Feedback Summary 22](#_xxmpw5bkvz4)

[4.4 Development process challenges and solutions 22](#_r58aainenysx)

[4.5 Testing 25](#_n93d737pyw63)

[5. Summary and conclusions 26](#_1u7ov4j83psi)

[5.1 Improvement suggestions 26](#_cgyiht5kb4b8)

[6. User documentation 27](#_fp71zqli8ulp)

[6.1 User’s guide operating instructions 27](#_qjvowmkf4f33)

[6.1.1 General description 27](#_l57dpp5bfxwi)

[6.1.2 The System’s Main Modules 27](#_8fu326qjujmz)

[6.1.3 Stakeholders 27](#_4wmx8fvldtjx)

[6.1.4 User Interface 28](#_jx3jsf82xvvn)

[7. Maintenance Guide 40](#_8wccdaem4v48)

[7.1 Hardware and software requirements 40](#_8a3xpq1bol44)

[7.2 Training a model 40](#_oi8doueinx6v)

[7.2.1 Data augmentation (optional) 40](#_932oga7wt77n)

[7.2.2 Collecting hand key points from the videos 40](#_wa4rwu4xt2l2)

[7.2.3 Training 41](#_yfhapbtjp2cg)

[7.3 Package Diagram 42](#_cdy76rvls5nc)

[8. References 43](#_lu2u5xfuc1an)

[Appendix 44](#_kn7ms6j91rj7)

# *Abstract*

The ability to communicate effectively is a fundamental human need, and for the deaf and the hard-of-hearing community, sign language serves as their main way of communication. However, learning sign language can be challenging and usually requires qualified sign language instructors.

Our project deals with the development of SignQuest, an application for assisting in sign language learning using a live sign video recognition by machine learning techniques.

SignQuest consists of three components: a teaching system that offers basic ASL lessons, a video dataset that contains videos of ASL words' gestures taken from various sources, and a gesture recognition model. The recognition model uses the MediaPipe framework for extracting hand landmarks, and an LSTM(Long short-term memory) neural network model to recognize the sequential landmarks of the ASL gestures.

Our model was trained on hundreds of video films of 30 different ASL words, and achieved a 98% accuracy rate, enabling a high accuracy of live hand gesture recognition.

In conclusion, SignQuest provides an effective solution for learning ASL, bridging the communication gaps for the deaf and hard-of-hearing community.

# 

# 

# 1. Introduction

Over 5% of the world’s population – approximately 430 million people – experience disabling hearing loss. Deaf individuals, who typically have profound hearing loss, use sign language for communication. Unfortunately, in the United States, about only 500,000 people use the American Sign Language (ASL), a small fraction of the overall hearing-impaired population. The problem is not only the limited availability of educational resources but also the lack of real-time ASL feedback that often requires costly private instructors.  
  
To address these gaps, we developed SignQuest, an interactive application that facilitates the learning of ASL using machine learning technology. This solution is designed not only for the hearing-impaired community but also for anyone that is interested in learning sign language to let them communicate effectively with deaf individuals.  
  
SignQuest uses modern technologies to offer a more accessible and engaging learning experience. The core of the system is a live gesture recognition system ,powered by machine learning techniques. The application uses the MediaPipe, a framework for extracting hand landmarks from real-time video input. The data is analyzed by a Long Short-Term Memory (LSTM) neural network model to recognize the sequential nature of the ASL gestures.

The application has three main components.  
The first component is the **teaching system** which provides step-by-step lessons, starting with basic ASL words and gradually advancing to more complex sentences.  
The second component is the **video dataset** that contains a rich collection of gesture videos sourced from various platforms, used to train and validate the recognition model.  
The third component is a **recognition model** built using MediaPipe and an LSTM neural network. The model was trained on hundreds of videos, representing 30 different ASL words. The result was a recognition accuracy of 98%, that ensures precise feedback for each ASL gesture of the user.

During the learning process, the user views a video demonstrating a specific ASL gesture, and he is requested to mimic the gesture. The system takes a live video of the gesture and provides instant feedback, helping users to improve their signing accuracy. Furthermore, SignQuest offers interactive exercises such as matching gestures with the correct word, filling in missing words, and constructing complete ASL sentences, making the learning process more dynamic and comprehensive.

In the following chapters we will discuss the details of the project.  
The project book is divided into 7 chapters.

Chapter 1 is the introduction of the project.  
Chapter 2 gives a general description of the project and reviews the system architecture, workflow, and use cases with UML diagrams. It also introduces the datasets used in the project.  
Chapter 3 covers the main tools and technologies used in the project.  
Chapter 4 covers the development process including the optimization and training of the sign language recognition model, the creation of the user interface, the results of the project, challenges we faced and testing of the software.  
Chapter 5 provides a summary of the project, offering key conclusions and suggestions for future improvements.  
Chapter 6 provides the user documentation for the app.  
Chapter 7 includes installation requirements, model training instructions, and the system’s package diagram.

# 2.Project description

The main goal of this project is to create an app for learning the american sign language. The system can help the deaf community to deal with communication in their daily life.

SignQuest is designed to assist in learning basic American Sign Language (ASL) through an interactive learning system. This system provides lessons on ASL fundamentals, helping users to practice and master 30 common ASL words.

The system includes a user-friendly interface for accessing the lessons developed using Tkinter- a python library for creating a GUI.

All The lessons and user information are stored on a dedicated MongoDB database.

The project includes a comprehensive video dataset featuring gesture recordings of these 30 ASL words, sourced mainly from WLASL and SignASL data sets used for training and testing the recognition model.

## **2.1 Model and Teaching method**

At the heart of SignQuest is a gesture recognition model that leverages the MediaPipe framework to extract hand landmarks. These landmarks are then fed into a Long Short-Term Memory (LSTM) neural network, which is responsible for recognizing the sequential patterns of ASL gestures.

In the system, users engage in various exercises to practice ASL. They can repeat gestures by watching a video and mimicking it in front of the camera, choose the correct word by selecting from multiple choices after watching a gesture video, or type the correct word corresponding to a shown gesture. Additionally, users can guess and sign a missing word in an incomplete English sentence or sign a full sentence based on a given ASL sentence.

The two main types of exercises in our system, repeating a gesture and signing a sentence, use the gesture recognition model to recognize signs in real-time and give the user feedback when they successfully perform a gesture.

In the sentence exercise, the user is shown an english sentence and the corresponding asl sentence and is asked to sign the gestures in the same order.

The exercise helps the user to understand the ASL syntax and improves the effectiveness of the learning process.

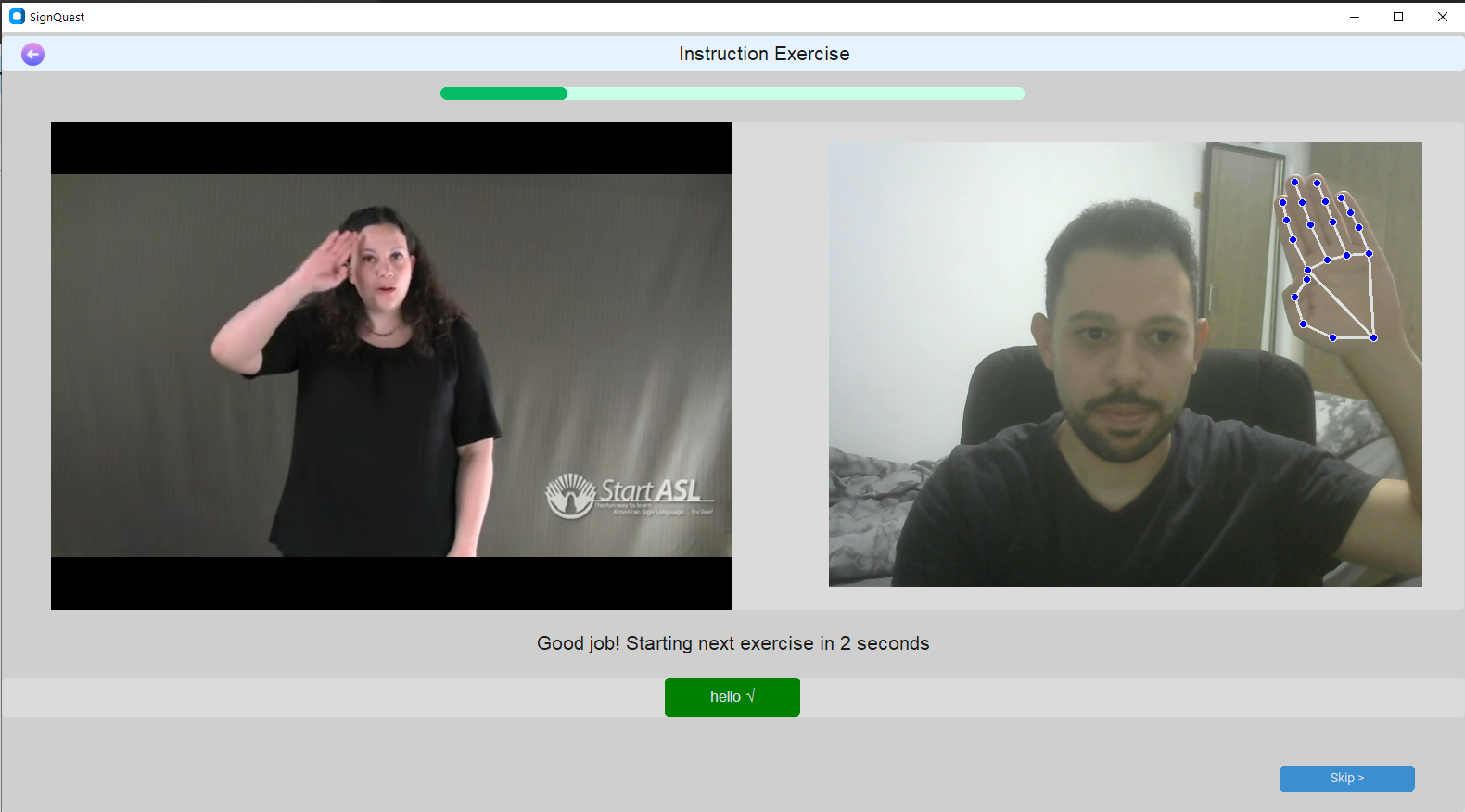


Figure 1: Repeat the sign exercise page (chapter 2)

## **2.2 Admin dashboard**

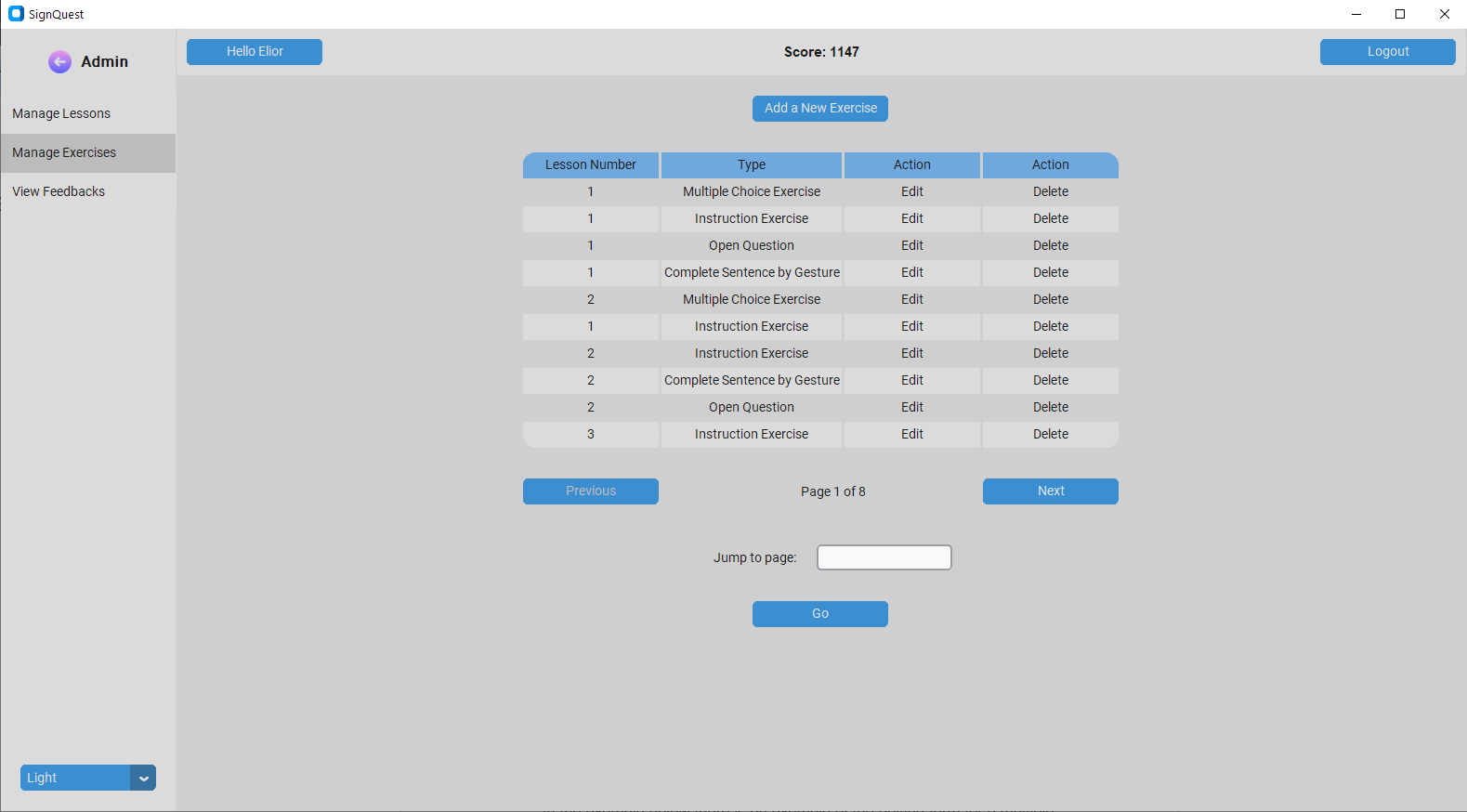
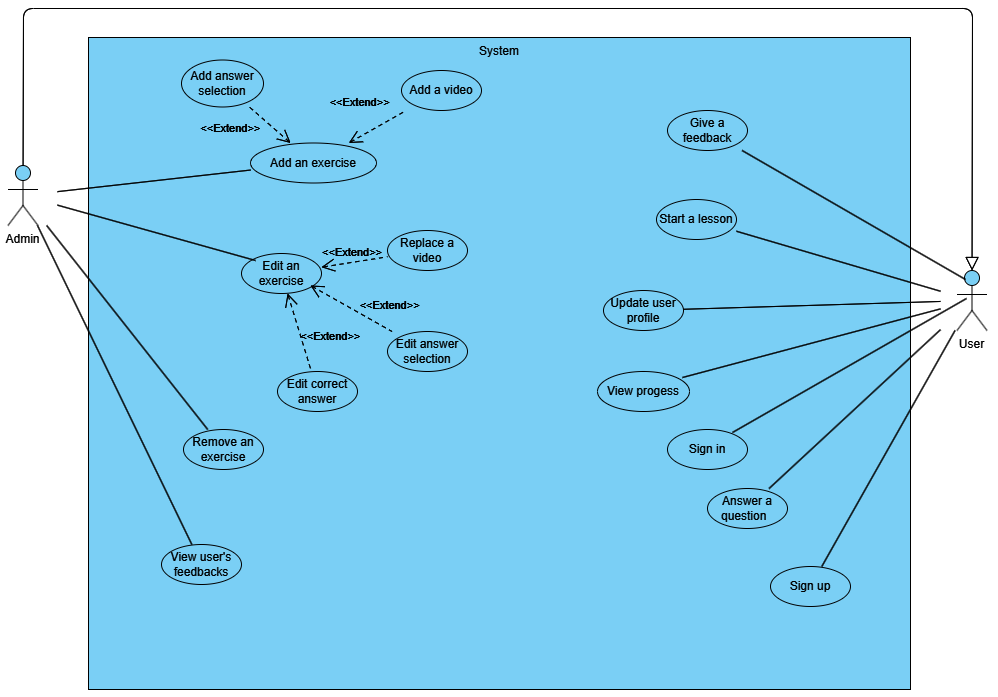
This part of the system is only accessible to the admin. The admin can do several operations in the dashboard- he can manage the lessons and their contents (list of words that appear in the lesson). In addition, the admin can add,edit and remove exercises from existing lessons and view users' feedback. 

Figure 2: Admin’s manage exercises page (chapter 2)

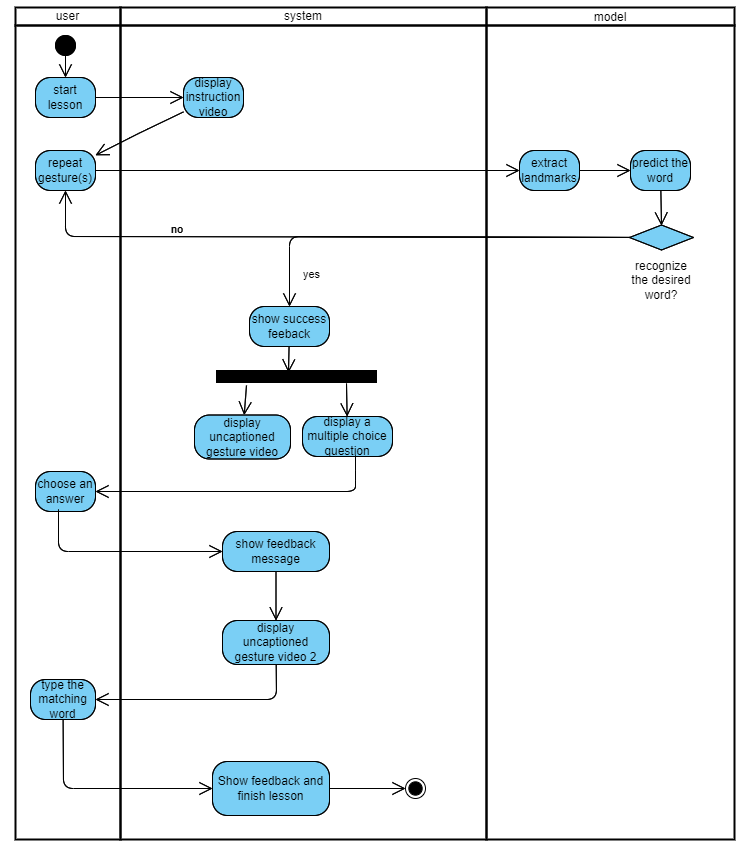
## **2.3 Use Case Diagram**

*figure 3: Use case diagram*

**Add an exercise** - adding a question (exercise) that is part of a lesson. Some of the exercises include a video or multiple-choice answer.  
  
**View User’s Feedbacks** - the admin can view users’ feedbacks on the overall user experience

**Start a lesson** - At the beginning of each lesson an instruction video appears that includes a new gesture performed by a signer  
  
**View Progress -** Each user has a progress that shows him the percentage of completed lessons ,new learned words and the scores of completed lessons.

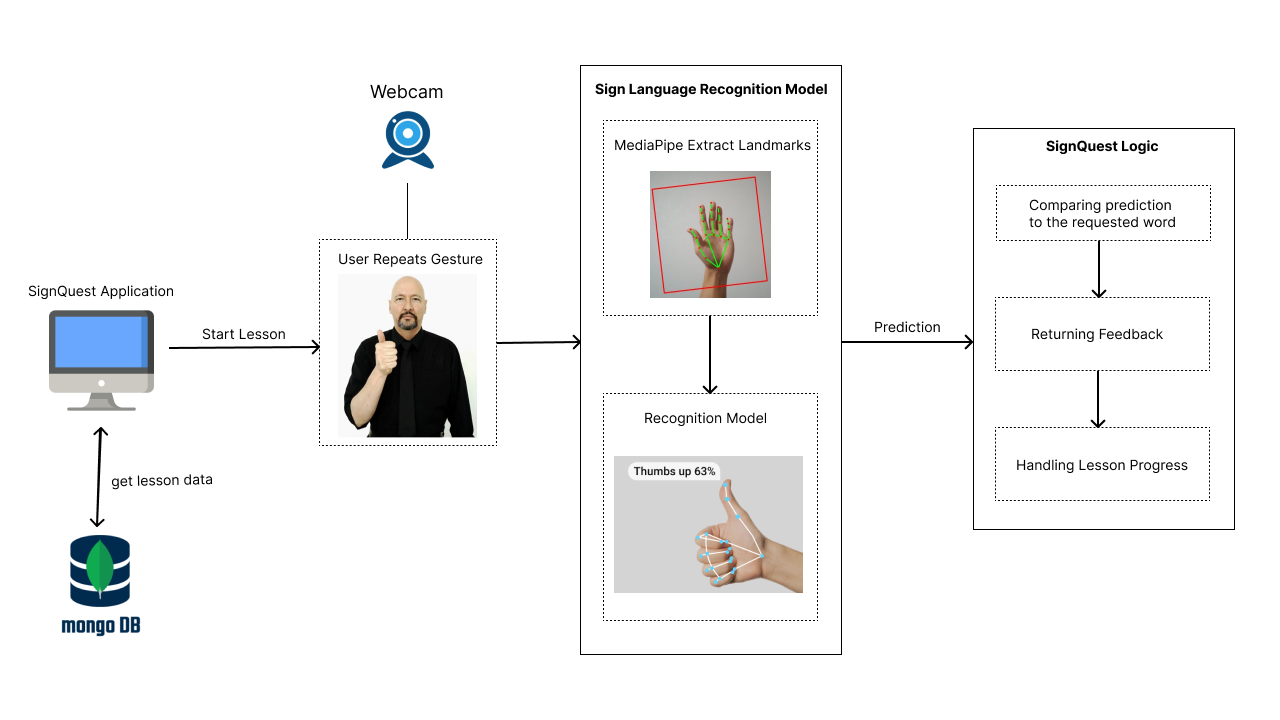
## **2.4 Activity Diagram**

*figure 4: activity diagram*

The diagram describes a typical lesson process.  
The process starts by clicking start lesson, then the system displays an instruction video on how to perform the gesture.  
Using the user’s webcam, he performs the gesture and the system extracts his landmarks and passes it to the model that predicts the word, on success the user moves to the next question until the lesson is over.

## **2.5 System architecture**

First, the SignQuest application gets the lessons data from the mongoDB database.  
At the start of a lesson the user is asked to repeat an ASL gesture that is shown in an instruction video. The sign language recognition model uses MediaPipe to extract the landmarks and predict the performed user’s gesture. Then, the prediction is compared to the requested word. In case of a match, a feedback is returned to the user and the lesson continues.

*figure 5: System Architecture*

## **2.6 ASL Datasets**

There are a few datasets of videos of ASL gestures.

In our project, we used videos from 2 datasets: WLASL and SignASL.

### 2.6.1 WLASL

Word-Level American Sign Language [[1](#kix.qduasyfvh97l)] is a large video dataset specifically designed for American Sign Language (ASL) recognition at the word level. It contains 2,000 common different words in ASL, with each video corresponding to a specific ASL word. Researchers and developers can use WLASL to train and evaluate models for accurate word-level sign language recognition

### 2.6.2 SignASL.org

The largest collection of video signs online. The collection contains videos from multiple sources and Currently there is just over 40,000 videos listed on this site, Signed by more than 150 different signers of different ages, ethnicities and genders.

## 

# 3. Tools We Used

After searching technologies for our sign language teaching application, we chose a few technologies for creating the recognition model and developing the application.

For the sign language recognition model we used MediaPipe to extract the hand landmarks. In addition, we used LSTM which is a neural networks model that captures the sequential and contextual nature of sign language gestures. The LSTM network we used was provided by the tensorflow library.

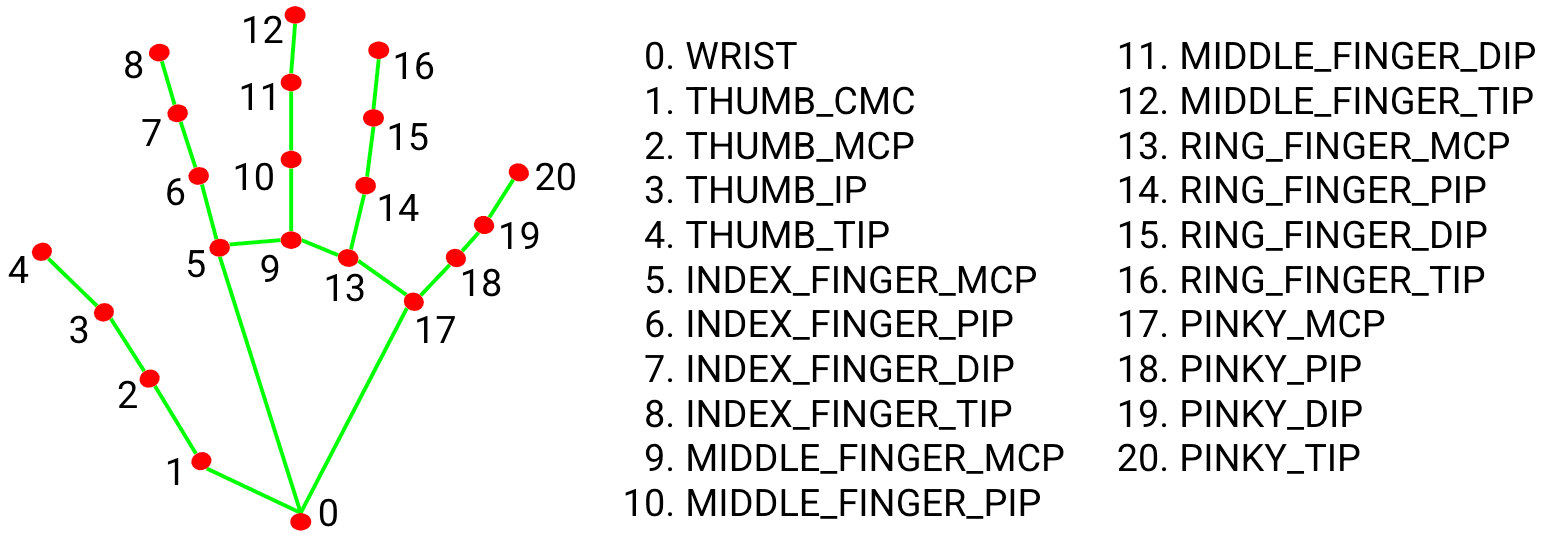
For the application development we used Python as the programming language of the application, along with Tkinter which is a GUI framework and MongoDB for storing information.

In addition,we used OpenCV for video processing and for capturing video from a webcam in real time.

**MediaPipe -** MediaPipe **[**[2](#kix.dy0zbibznwsi)**]**  is an open-source framework developed by Google that facilitates the creation of real-time multimedia applications. It provides a suite of pre-built machine learning models and tools for various tasks. MediaPipe Holistic which is a solution under the Mediapipe framework includes pose estimation, face detection, hand tracking, and more.  
MediaPipe simplifies complex computer vision tasks by using pre-trained models to analyze images or videos in real-time. It identifies key points such as body joints, facial landmarks, and hand keypoints, enabling applications like sign language recognition, augmented reality effects, and more.

The MediaPipe holistic provides the following landmarks:

* **Hand** - The hand landmarks consist of 21 3D coordinates representing the positions of key points on the hand, such as the fingertips, knuckles, and wrist. These landmarks can be used to track hand gestures, hand poses, and interactions with virtual objects.
* **Pose** - The pose landmarks consist of 33 3D coordinates representing the positions of key points on the human body, including shoulders, elbows, wrists, hips, knees, and ankles. These landmarks can be used for pose estimation, tracking body movements, and analyzing body gestures.
* **Facial** - The facial landmarks consist of 468 3D coordinates representing the positions of key points on the face, such as the eyes, eyebrows, nose, mouth, and jawline. These landmarks can be used for facial recognition, facial expression analysis, and facial motion tracking, among other applications.

**

*figure 6: Hand Landmarks representation used in Mediapipe*

We used the MediaPipe framework because it enables the extraction of the hand landmarks from a video. Those landmarks are essential for tracking and recognition of an ASL gesture.  
MediaPipe can extract the landmarks accurately even in cases of weakened hardware devices and lower quality cameras.

**LSTM (RNN)** - A recurrent neural network (RNN) [[3](#kix.4fap8rlosk0l)] is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning. The algorithms are incorporated into popular applications such as Siri, voice search, and Google Translate.

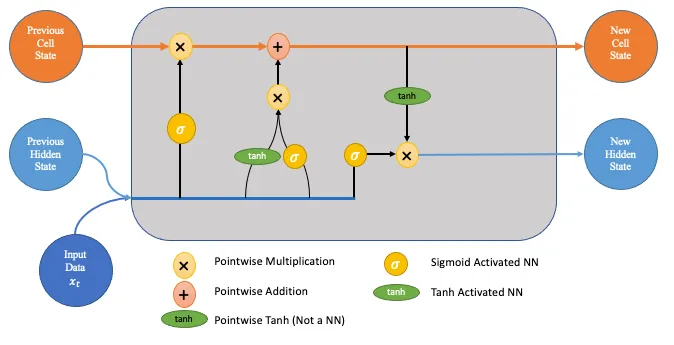
The downside of RNN is that it can only remember the immediate past input. It can’t use inputs from several previous sequences to improve its prediction.

In order to solve this problem, a variant of RNN called Long short-term memory (LSTM) that enables the model to expand its memory capacity to accommodate a longer timeline can be used.

LSTM networks add a special memory block called *cells* in the hidden layer.   
Each cell is controlled by an input gate, output gate, and forget gate, which enables the layer to remember helpful information.  
  
In the article [[3](#kix.4fap8rlosk0l)] a system for American Sign Language (ASL) alphabet recognition that integrates LSTM and MediaPipe was proposed. The system was able to recognize the letters with accuracy of 99%. The authors concluded that the work can be further extended for word recognition.

We chose to use LSTM for our sign language recognition model architecture.

Sign language involves dynamic gestures and sequences of movements.  
LSTM [[3](#h9ot5gek6z7b)] is designed to handle sequential data, making it well-suited for capturing the gestures in sign language with sequence of movements that comprise the whole gesture.  
The information stored by the LSTM model is essential for creating an   
accurate model for recognizing ASL words and even sentences.

  
*figure 7: LSTM Architecture*

**MongoDB**- We chose to store our dataset of ASL words and matching video links and the users’ information in the MongoDB database.   
We also used PyMongo, which is a Python library used to interact with MongoDB.   
  
**Tkinter** - We used Tkinter to design the app’s user interface.  
Tkinter is the most used environment for GUI generation in Python.  
It is simple and easy to use, and enables you to create a GUI relatively fast.we also used an extension of the library called CustomTkinter that has more advanced features.  
  
**Tensorflow-** tensorFlow is a powerful open-source machine learning library developed by Google. It's designed to facilitate the creation, training, and deployment of deep neural networks and other machine learning models.

We used tensorflow in our project to build our LSTM neural network.

**OpenCV -** Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. It provides a wide range of functionalities designed to help developers build real-time computer vision applications.

We used OpenCV in our project for processing and the collection of keypoints from the videos in our dataset.

In addition we used OpenCV to integrate a live video feed from the web camera to our app, allowing real time detection and recognition of gestures combined with the model.

### 

# 4. Description of the development process

In the beginning of the project we collected videos of 30 different ASL gestures from WLASL and SignASL. as we were collecting the videos from those 2 sources we noticed we get a limited amount of videos per word- less than 10 words per video.

The videos we collected were processed using Mediapipe which allowed the collecting of landmarks from each frame of the videos.

The landmarks made the data more suitable for training a LSTM neural network, which uses sequential data or time series data.

As we began to train the model we saw a difficulty in achieving a satisfying accuracy percentage and concluded it was due to the relatively small number of videos per word.

In order to solve this problem, we used data augmentation.

After the training of the model we began the development of the learning app.

The development of the app included the building of a MongoDB database that will allow the system to store and retrieve lessons and users information.

For building the GUI of the app we used Tkinter, a popular framework for creating friendly user interfaces with capabilities and functions that serve our project's needs, including the displaying of videos for the lessons in the app.

We integrated it with the model by loading its file to the app. The loaded model then enabled the real-time predicting of ASL gestures.

## **4.1 Recognition model development**

### 4.1.1 Dataset

For the training of the model, we used hand key points extracted from videos of our dataset using Mediapipe.  
During the training we divided the dataset to a training set and a test set, with 10% of the videos included in the test set.   
The dataset contains videos of 30 different ASL words we collected mainly from WLASL and SignASL, but the average number of videos per class was less than 10.  
The small amount of videos resulted in bad performance of the model, which we decided to solve using data augmentation.

In order to solve this problem, we used data augmentation- we artificially increased the number of videos per word by creating and manipulating copies of the original videos.. The data augmentation allowed us to continue training the model with enough data to get much better results.

The augmentation techniques that were used are:

* **Horizontal shifting (translation)** - each of the original videos in our dataset was shifted horizontally by various relative distances : 0.05 and 0.1 to both the left and right, resulting in four different shifted versions for each original video.
* This technique increased the number of videos by 5 times.
* **Zoom** - after the horizontal shifting, all of the videos in the dataset were zoomed by 1.1, increasing the number of videos by 2 times.
* **Mirroring** - finally, all the videos were mirrored.This resulted in the inclusion of videos of each gesture performed in both hands, which can allow better gesture identification. This technique also increased the total number of videos by 2.

At the end of the data augmentation, we got 120-220 videos per class, which were sufficient to train and test the model. In addition, we discarded videos that were too long(120 frames or more).

The total number of videos after the process was 5240,with up to 119 frames for each video and 126 different hand key points that were extracted from each frame.

### 4.1.2 Model improvement

In order to develop the best sign language recognition model, we did many experiments to get the right architecture and hyperparameters.

We used the Adam optimizer for the training since it is widely used in different machine learning models. We used Adam's default learning rate of 0.001.  
The first architecture we tried included one LSTM layer of size 16 and one dense layer for the output. This architecture performed very poorly, so we decided to gradually increase the number of LSTM and dense layers until we saw a significant improvement in the model’s performance. We also increased the size of the different layers until we got the optimal sizes.  
We saw that we still didn’t get good results and also the training results were very unstable. Therefore we tried to adjust the model's hyperparameters including the learning rate, the batch size and the number of epochs.

However, the model still wasn’t good and stable enough and often resulted in overfitting.

We tried to change the model architecture, using different layers combinations of dropout, L2 and batch normalization. The only combination that showed significant improvement was using blocks of batch normalization after every LSTM and Dense layer(except for the output dense layer). To further stabilize the model we added a clipnorm parameter with a value of 1 to the Adam optimizer. Together with batch normalization, it smoothed completely the big spikes that appeared in the loss value.

The final recognition model consists of 3 LSTM layers and 3 Dense layers.  
In addition, batch normalization is applied throughout the model for regularization.

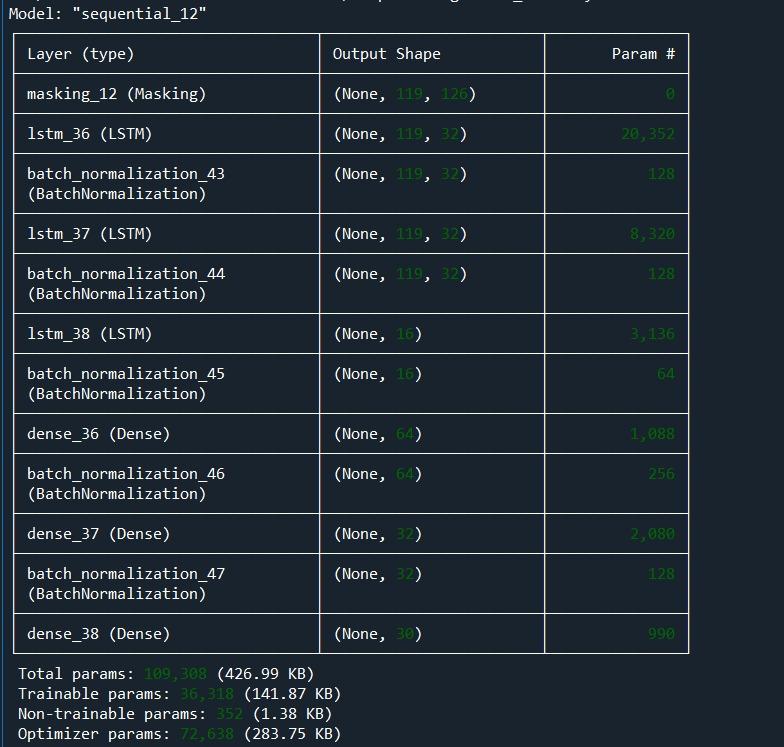


figure 8: model architecture summary

## **4.2 Creating the user interface**

In developing the GUI for our system, our primary goal was to create a user-friendly and visually appealing interface..

After evaluating different options, we concluded that Tkinter was the most appropriate choice for our project. Tkinter offers simplicity for creating functional GUIs while providing enough flexibility to create visually pleasing interfaces. Additionally, the availability of various pre-built components in Tkinter allowed us to meet key project requirements, such as integrating video content into lessons., we also utilized the CustomTkinter, an extension of Tkinter. CustomTkinter enabled us to leverage more advanced features and modern UI components, giving us greater control over the aesthetics and layout.

In conclusion, selecting and mastering the right tools for building our GUI was a critical part of our development process, and the time invested in learning Tkinter and CustomTkinter paid off in delivering a user-friendly interface that enhances the overall experience of our application.

## **4.3 Results**

### 4.3.1 Model Results

After doing the data augmentation we did a few experiments to get the best results.  
The best results were obtained for 150 epochs with a batch size of 32.  
The Adam optimizer was also used with a learning rate of 0.001.   
A clipnorm parameter with a value of 1.0 was also added to the optimizer to stabilize the training process.

10% of the dataset was used for the test set, including 524 ASL videos.

After the training, the model training reached accuracy of 99% and loss value of about 0.03. The test set accuracy was 98% and test loss was 0.07

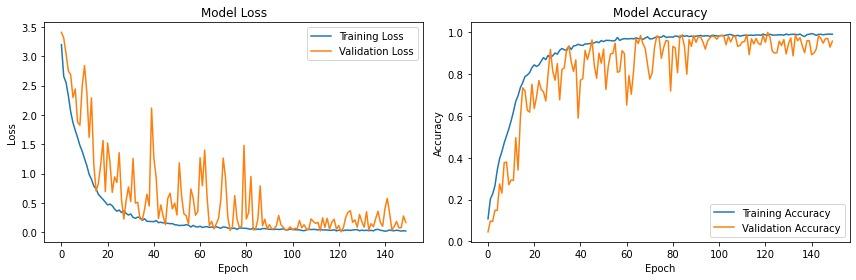


figure 9: Loss and accuracy graph

According to the results of the confusion matrix of the model, the model can correctly identify ASL gestures in most cases. Most of the classes had a 100% correct predictions rate on the test set, with all of the classes having a correct prediction rate of at least 82%.

The next page includes the confusion matrix summary table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Correct predictions** | **Incorrect Predictions** | **Test Samples** |
| bathroom | 90% | 10% | 10 |
| cost | 96.66% | 3.34% | 30 |
| deaf | 100% | 0% | 13 |
| drink | 100% | 0% | 16 |
| eat | 100% | 0% | 21 |
| fine | 100% | 0% | 19 |
| go | 100% | 0% | 18 |
| good | 100% | 0% | 13 |
| goodbye | 82.35% | 17.65% | 17 |
| hamburger | 100% | 0% | 14 |
| hello | 87.5% | 12.5% | 16 |
| home | 100% | 0% | 16 |
| how | 100% | 0% | 15 |
| israel | 100% | 0% | 11 |
| learn | 100% | 0% | 12 |
| live | 100% | 0% | 19 |
| me | 100% | 0% | 21 |
| morning | 100% | 0% | 14 |
| no | 100% | 0% | 25 |
| please | 100% | 0% | 18 |
| thank you | 100% | 0% | 20 |
| want | 100% | 0% | 15 |
| water | 100% | 0% | 5 |
| what | 100% | 0% | 29 |
| when | 95.65% | 4.35% | 23 |
| where | 92.85% | 7.15% | 14 |
| work | 100% | 0% | 21 |
| yes | 100% | 0% | 12 |
| you | 95.65% | 4.35% | 23 |
| your | 100% | 0% | 24 |

### 4.3.2 User Testing and Feedback Summary

As part of the evaluation of the system, we asked a few users to test the system.  
The users were able to effectively learn ASL gestures - all of the users got an average score of above 80% in the lessons they completed.

The users provided positive feedback on the app through a form included in the appendix.  
  
The feedback summary table:

|  |  |
| --- | --- |
| Question | Most common answer |
| 1 | Good |
| 2 | Accurate |
| 3 | Easy to use |
| 4 | Good Variety |
| 5 | Satisfied |
| 6 | Very Helpful |

## **4.4 Development process challenges and solutions**

**Hardware limitation** - To successfully train this model, it is essential to have a powerful GPU and CPU. We succeeded in training the model in our computer. However, it took a few hours to extract the landmarks from the videos. For a model with more words it is recommended to use cloud services such as Google Colab.

After the landmark extraction, the LSTM training itself took only 30 minutes.

**Dataset Collecting** - There are several ASL datasets, none of them provide a sufficient number of videos for each word, which is essential for optimal model training.

To address this issue, our approach was to merge two different ASL datasets(WLASL, SignASL) in order to create a diverse dataset.

After trying to train a model using videos collected from the two datasets, we got bad results and we solved this by artificially increasing the number of videos per word using data augmentation.

**Different length of videos of the same gesture** - When collecting the videos of different gestures, we noticed that videos of the same gesture can have different lengths. The meaning is that each video of a word includes a different number of total frames, which can confuse the model and affect its accuracy.

In order to solve this challenge we added padding of zeros for the missing frames in the shorter videos so each video will “have” the same number of frames (The padding was actually done to the values of hand landmarks collected by Mediapipe and not the videos). In addition, we used masking so the model will ignore the added frames so the model will take only real and relevant frames into account when trying to find the correct weights. It was done by the masking layer that appears in the model architecture summary.

**Creating a Teaching Method** - Finding an effective teaching method to aid users in remembering sign language poses a significant challenge.

In the process of finding the right teaching method, we drew inspiration from other language learning apps and created 5 different types of exercises:

1. **Repeat the gesture** - the system shows a video of an ASL gesture and the user is asked to repeat it in front of the camera.
2. **Choose the correct word** - The system shows a video of a gesture and the student is asked to choose the correct answer from multiple possible choices.
3. **Type the correct word** - The system shows a video of a gesture and the student is asked to type the corresponding word.
4. **Missing word** - The system shows a sentence in English with a missing word and the user is asked to guess and sign the word.
5. **Sign a sentence** - The user is shown an ASL sentence (text) and is asked to sign the corresponding gestures.

The different types of exercises are based on memory recognition and recall to ensure an effective learning process.  
At the end of each lesson, a score for the lesson is calculated based on the exercises that were completed successfully.

**Different gestures for the same word** - While collecting videos of asl gestures, we noticed that some words have more than one gesture.

We had to decide which gesture variation to choose. In some cases, We preferred variations that had more videos. However, in other cases the different versions had almost the same number of videos. In these cases we decided to choose the variations that are easier to learn and aren’t too close to other gestures in our dataset.

**Different syntax from English** - In ASL, there is a very different syntax from english. The order of words in an ASL sentence is usually different from the orders of the words in the matching English sentence.

In addition, some English words like “is” and “in” are not signed as part of a sentence.The gesture for “me” can also be used in the meaning of “I” in sentences, making the syntax a little more different.

For example, The sentence “I live in Israel” becomes “Me live Israel” in ASL.

Since most of the videos in signASL and WLASL datasets are videos of ASL words and not sentences, we had to look for ASL materials in other ASL sources including different ASL websites and Youtube channels with explanations on the syntax and sentence examples.

## **4.5 Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Action** | **Expected result** | **Actual Result** |
| **1** | Signing up | 1. Success message 2. New user created in the database. 3. page redirects to login | pass |
| **2** | Signing in | 1. Success message 2. File with user credentials created 3. page redirects to homepage | pass |
| **3** | User updating his profile | User profile is updated with the new information. | pass |
| **4** | User viewing homepage | 1. showing all available lessons 2. showing toolbar | pass |
| **5** | User answering a question | 1. giving a feedback according to the answer 2. moving to the next question | pass |
| **6** | User viewing his progress | Correctly shows the user's progress: completed lessons ,words and scores of completed lessons. | pass |
| **7** | User watching an instruction video | Correctly shows instruction video corresponding to the word | pass |
| **8** | User submit a feedback | 1. Success message 2. New feedback is added to the DB with a rating,review and date. | pass |
| **9** | Admin adding an exercise | 1. Success message 2. New exercise is added to the DB with a type, correct answer, video link and list of possible answers | pass |
| **10** | Admin editing an exercise | 1. Success message 2. the new exercise information updated in the DB with a type, correct answer, video link and list of possible answers | pass |
| **11** | Admin viewing user’s feedback | All user’s feedback will be shown, with review, rating and date. | pass |
| **12** | User performs sign language gestures | 1. accurately recognizing users’ performed gesture 2. showing feedback to the user | pass |

# 5. Summary and conclusions

The goal of our project is to create an accessible and effective solution for learning ASL and bridging communication gaps for the deaf and hard-of-hearing community.   
We also aim to create a user-friendly interface for learning the ASL.

We successfully achieved our goal by creating an ASL learning app with real-time gesture recognition capabilities and a friendly user interface.  
We successfully trained an ASL recognition model using a dataset of 5240 ASL videos, with 524 of them used for the test set.

The sign language recognition model that is used by the app reached an accuracy rate of 99% on the train set and 98% on the test set, thus exceeding our goal of 77% accuracy from the first phase of the project.

The model is also able to make predictions of live gestures within seconds, enabling a smooth learning experience.

Although our model is trained to recognize only 30 ASL words, we believe it can also be trained to recognize more words with good results and even be trained to recognize other sign languages given the right data.

We also evaluated the app by asking users to test it.

User testing showed that participants effectively learned ASL gestures, achieving an average score above 80%, and provided positive feedback on the app.

## **5.1 Improvement suggestions**

**1. Training the model with a uniform dataset** - In our project we trained the model with a dataset of ASL videos with different lengths. We achieved good results, but training the model with a dataset of videos with the same short length(30/60 frames) can yield even better results.Collecting videos of the same length eliminates the need for padding the hand landmarks data of shorter videos with zeroes and might improve the model a little.

**2.** **Developing a web based UI** - While Tkinter enabled us the development of a user-friendly and visually appealing user interface, there was a bit of a learning curve when trying to learn how to use it. Developing the UI as a website using Javascript and different web libraries and frameworks such as React can lead to a faster development process and can be a little easier to maintain.

# 6. User documentation

## **6.1 User’s guide operating instructions**

### 6.1.1 General description

The main purpose of the system is to provide an accessible and effective solution for learning ASL. The system is able to recognize ASL signs in real time and provide the user feedback upon successfully signing a requested sign.

### 6.1.2 The System’s Main Modules

The app has 2 main modules:

* **User Module -** this module is accessible to all the users. The module enables users to learn asl and participate in the different lessons that are stored in the app’s DB. Each lesson includes different types of exercises. In some of the exercises the user is asked to perform ASL gesture(s) in front of a camera and the system analyzes his gestures to check their correctness.

The user is also able to track his lessons progress, send feedback on the app and update his details.

* **Admin Module -** This module is only accessible to the admin. In this module, The admin can create new lessons and edit their contents. The admin can also edit, remove and add new exercises to existing lessons.

In addition, the admin can view the users’ feedback.

### 6.1.3 Stakeholders

The system is helpful for people from the deaf and the hard-of-hearing community who need to learn ASL and have limited access to other learning resources. In addition, the system enables ASL learning for anyone who needs to communicate with them- including relatives, teachers, health care providers etc. Finally, the system is also targeted to the general public to make ASL learning accessible for everyone, bridging communication gaps with the deaf community.

### 6.1.4 User Interface

In this part we will explain the functions of different screens in the user interface.

* **Homepage -** This is the main page that is shown to the user after logging in or starting the app. The homepage is displayed again whenever the user clicks on the SignQuest logo at the top-left corner. clicking on the button “Hello <username>” navigates to the “My Profile” screen. The “Admin Page” button is only visible to the admin and clicking it opens the admin module. The user can switch between light and dark mode with the button in the bottom-left corner of the screen. The user can logout by clicking on “Logout”. In the navigation bar on the left side of the screen the user can browse through the list of the available lessons. completed lessons are marked with a V.

Figure 10: SignQuest Homepage

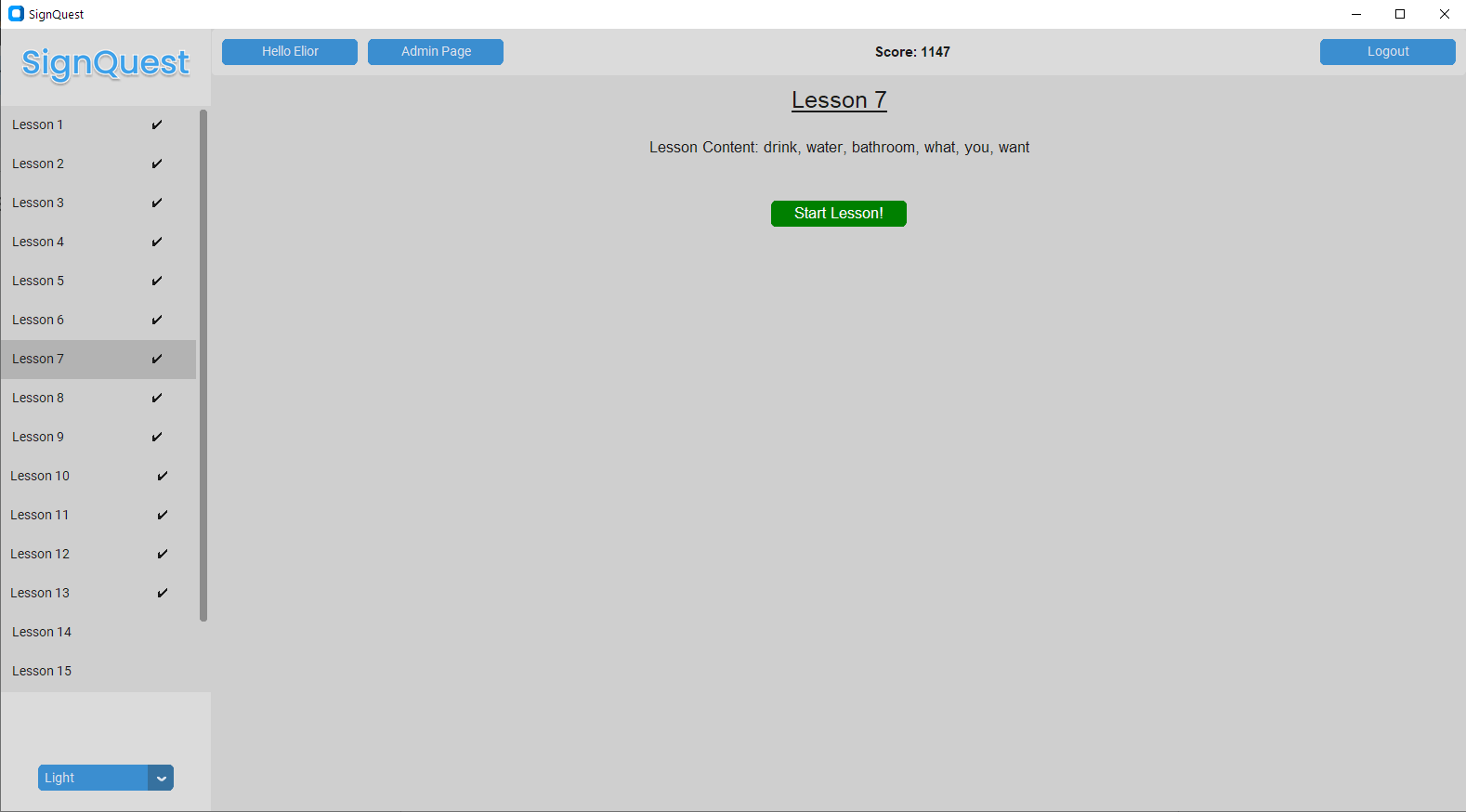
* **Start a lesson-** To start a lesson, the user can click on the lesson in the navigation bar and then click on the button “Start Lesson” that appears on the middle of the screen.

Figure 11: Start a lesson page

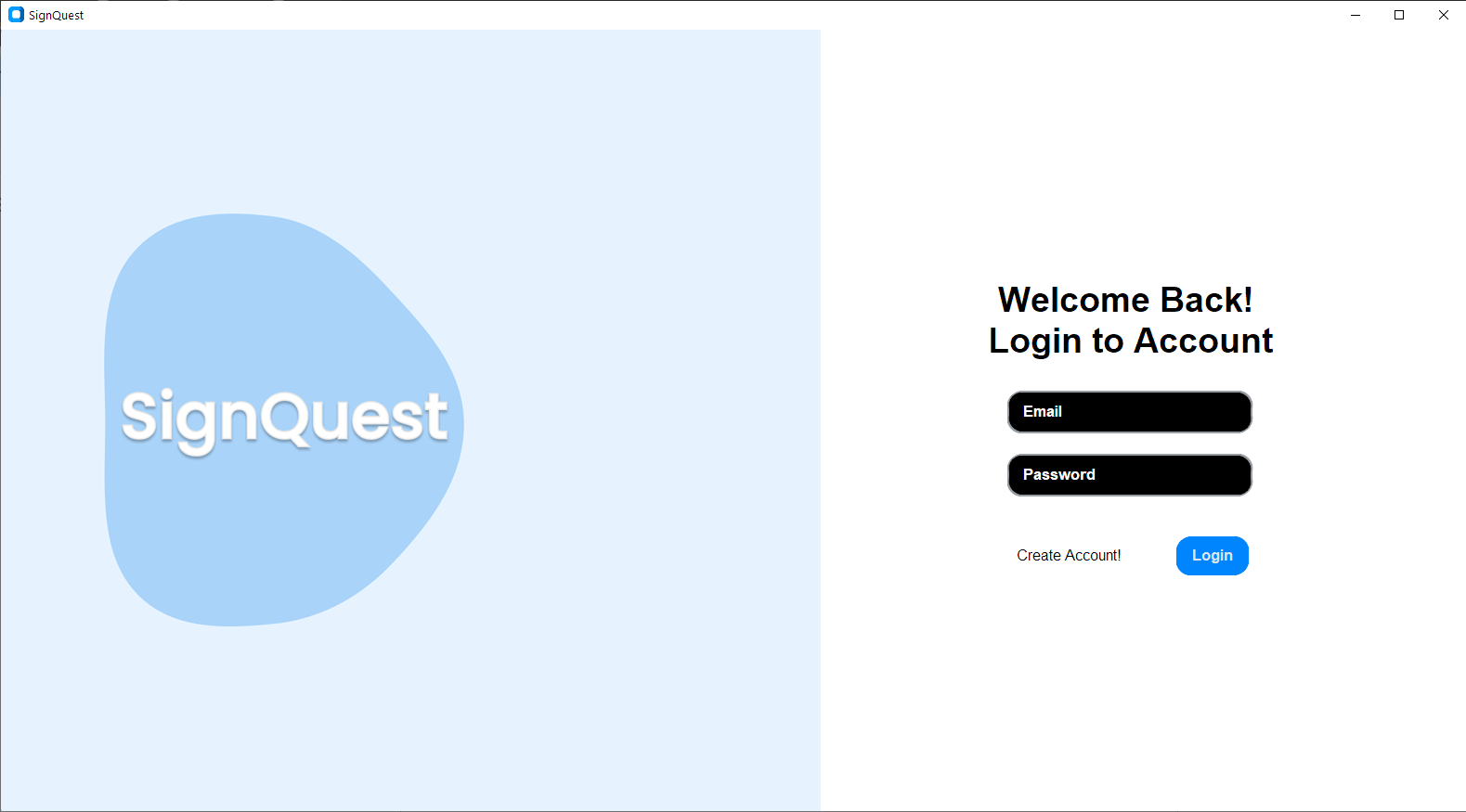
* **Login -** This is the first page of the application if you are not logged in. Login to the app requires email and password, if you don’t have one you can click “Create account” and move to the registration.  
  **the admin credentials are**: Email: [elior676@gmail.com](mailto:elior676@gmail.com) , password: 123

Figure 12: Login page

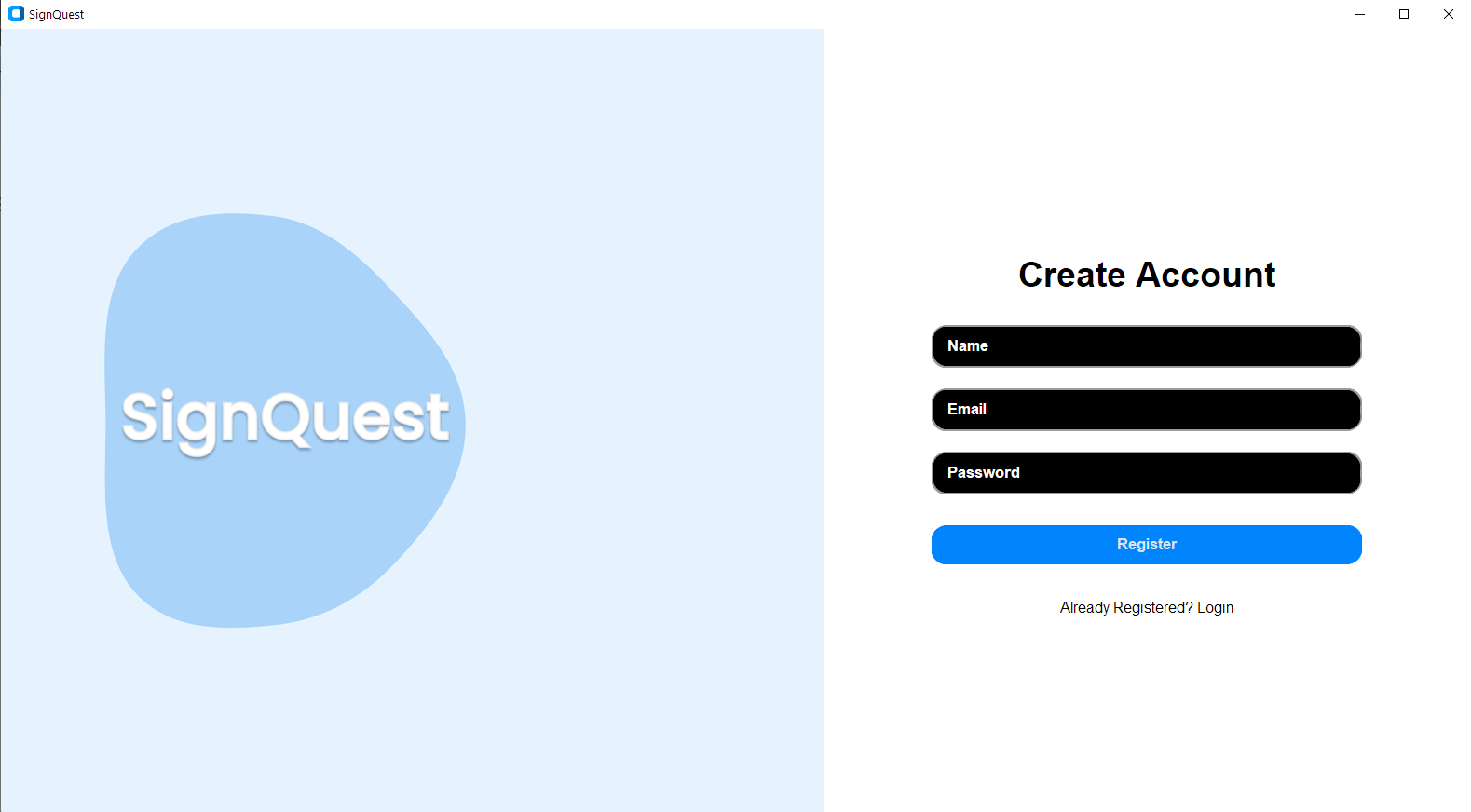
* **Register -** to register to the application you need to enter your name,email and password. After successful registration the user will be redirected to the login page.

Figure 13: Register page

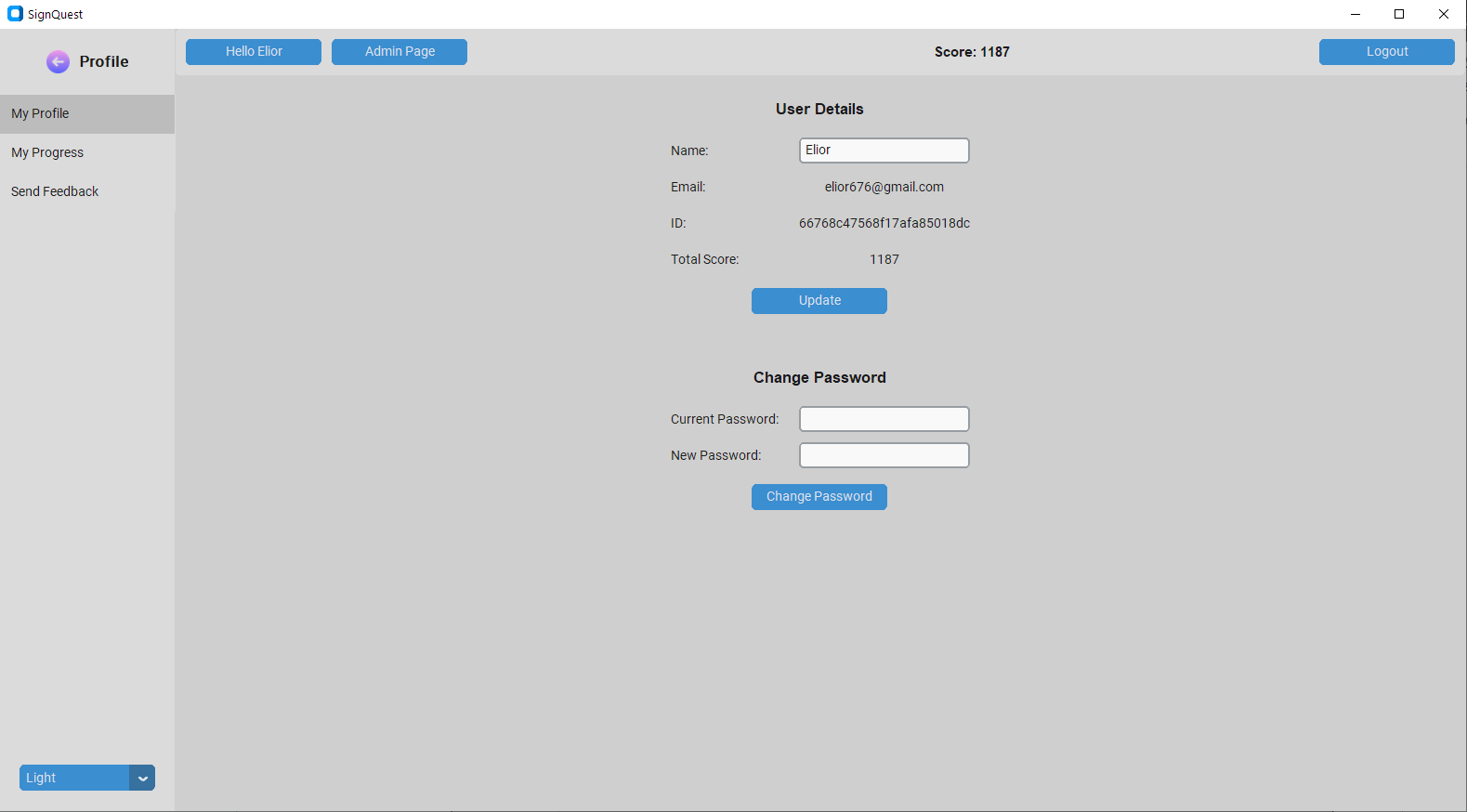
* **My Profile-** In this screen, the user can see his display name, email, user ID and Total score. The user is also able to change the display name in this screen by typing a new name and clicking on “update”.  
  In addition, the user can change his password by typing his current password and a new password and clicking on “Change Password”.

Figure 14: My profile page

* **My Progress-** In this screen the user can see the percentage of the lessons he completed, the number of words he learned and the score he achieved in completed lessons.

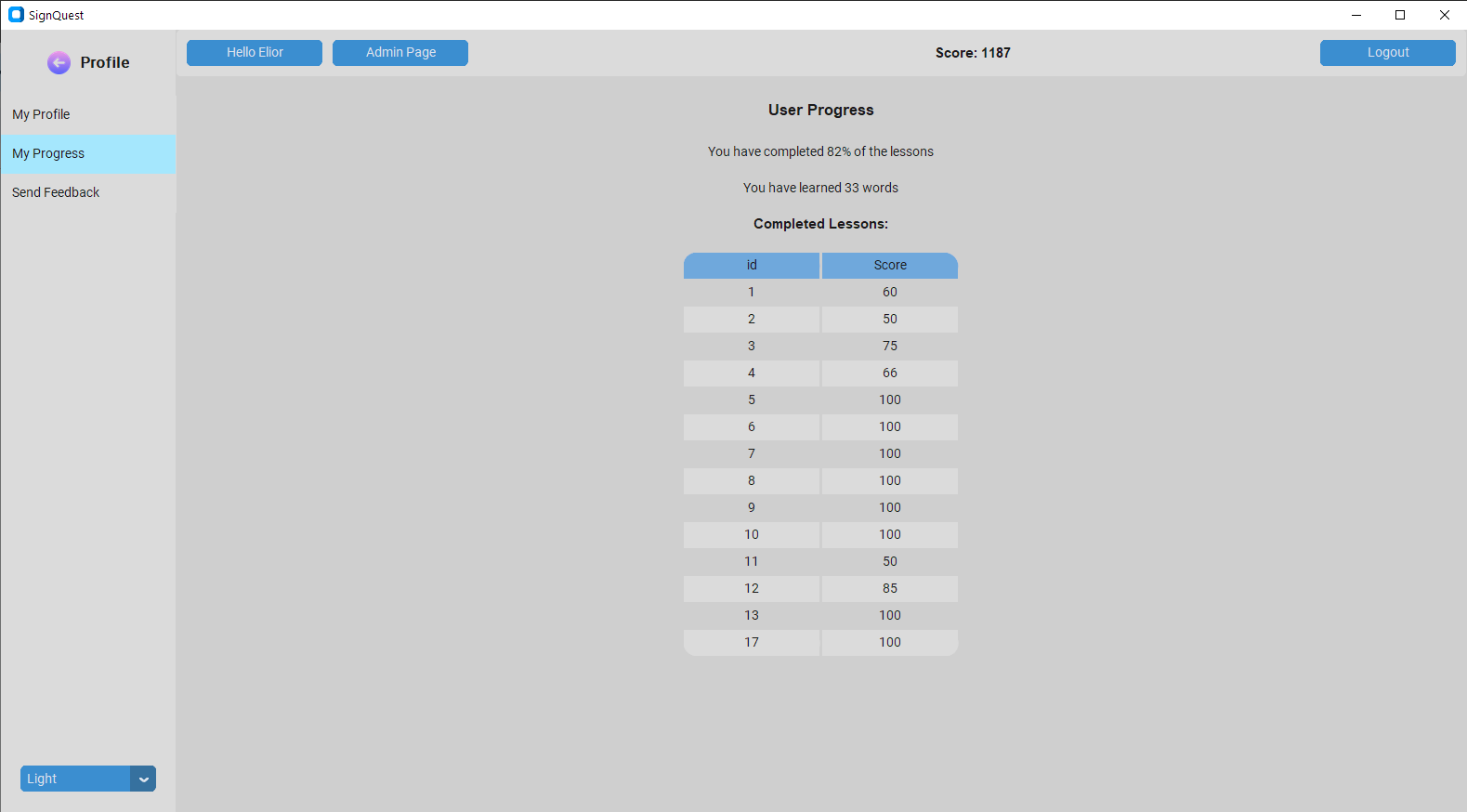


Figure 15: My progress page

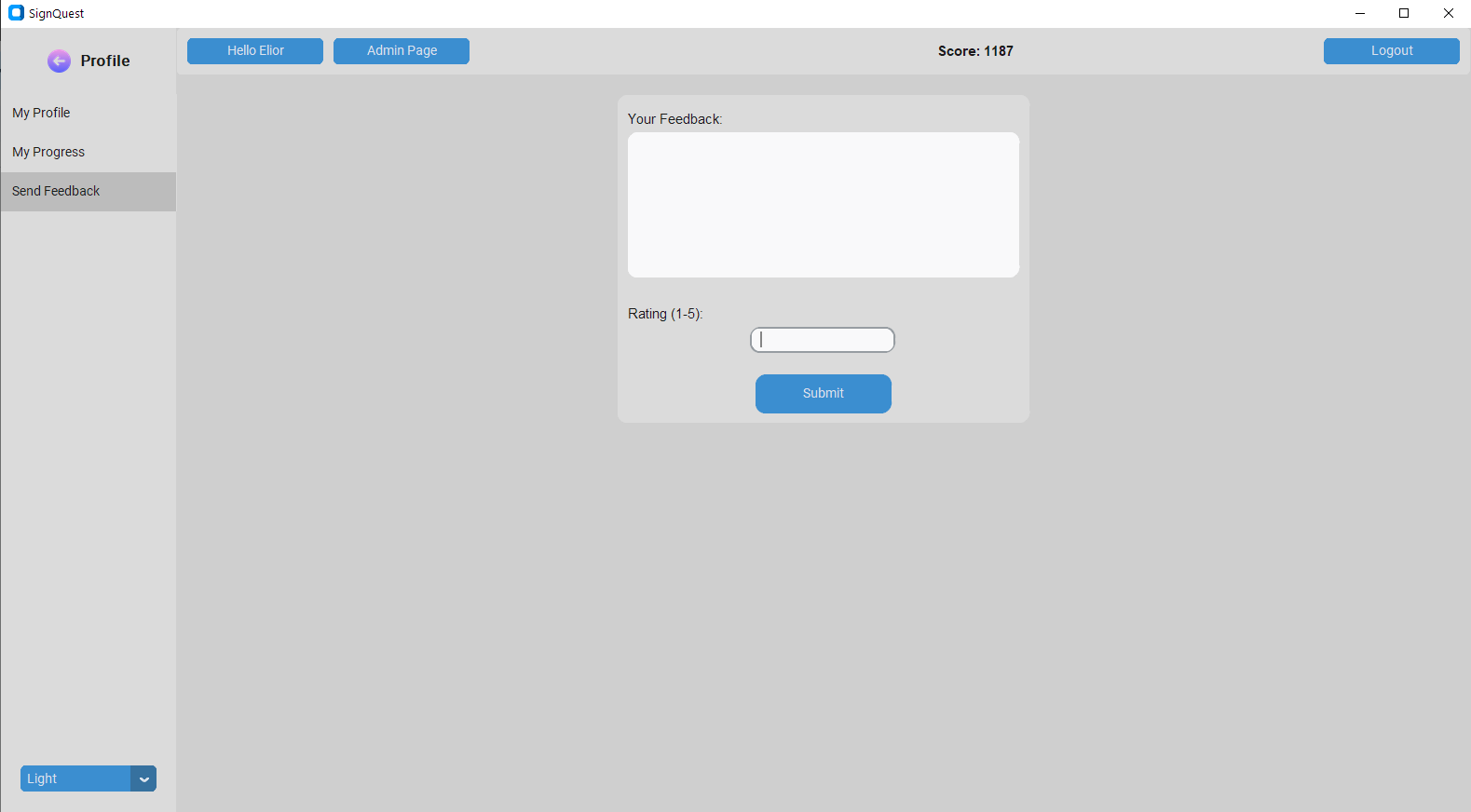
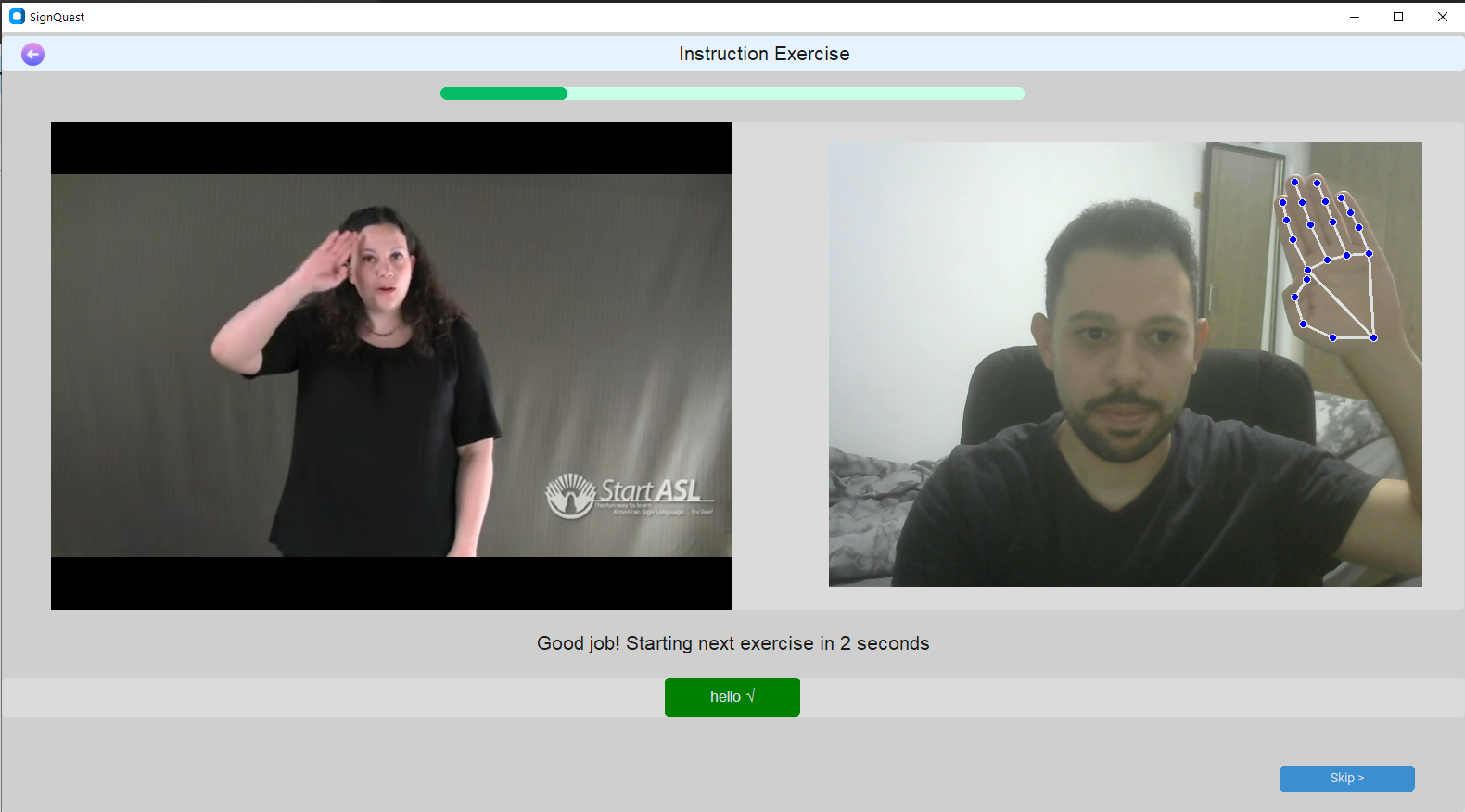
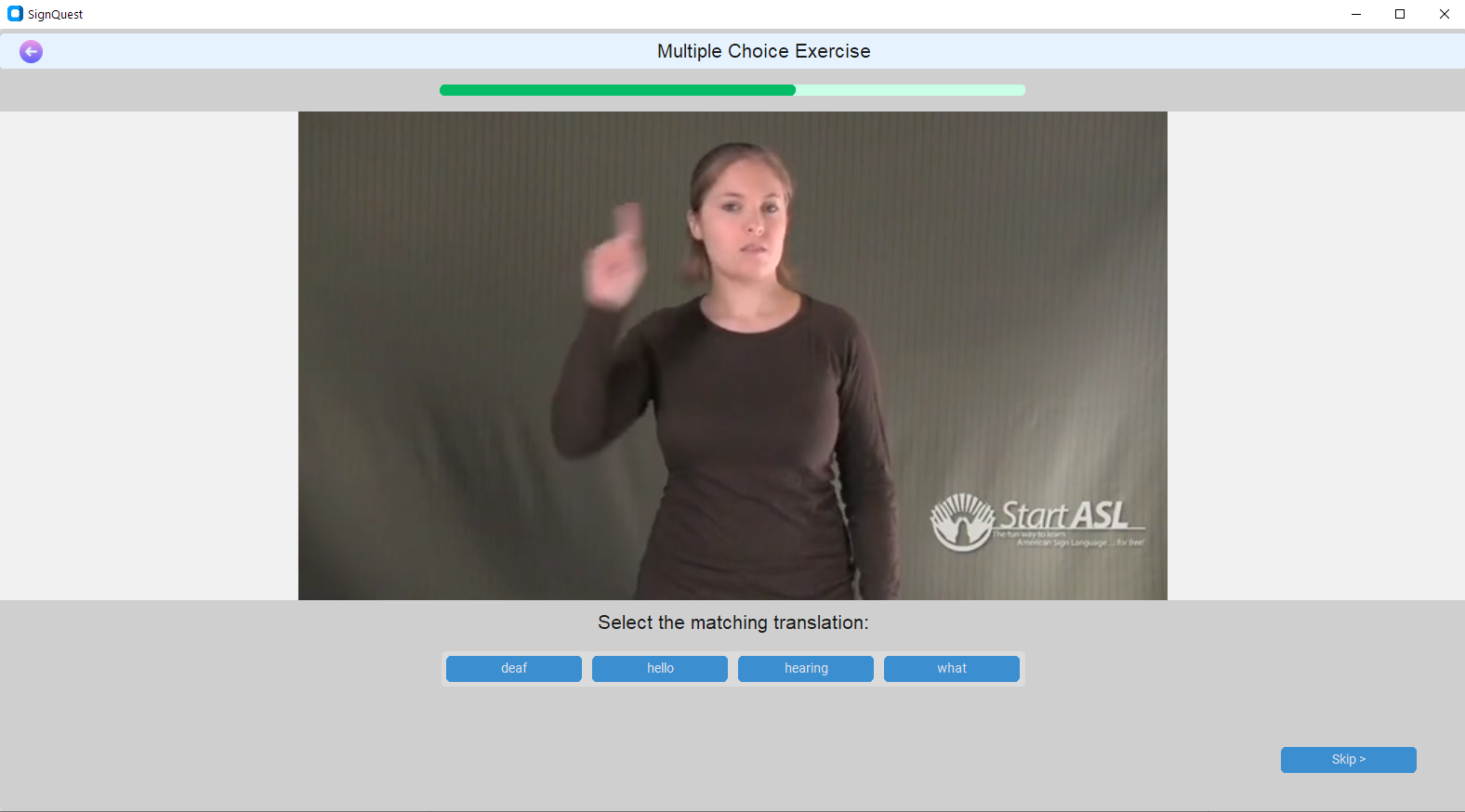
* **Send Feedback**- In this screen the user can write a feedback on the overall user experience and give the system a rating between 1 and 5. 

Figure 16: Send a feedback page

* **Repeat the sign exercise**-in this type of exercise the user is shown a video of a new asl gesture in the left square, and is asked to repeat it in front of a webcam.  
  The matching English word for the gestures appears under the video and The camera video feed appears in the right square on the screen. After correctly performing the gesture, a feedback message will appear to the user and he will be moved to the next exercise.   
  The user can see a progress bar at the top of the screen(in green) to track his progress in the whole lesson.

  
Figure 17: Repeat the sign exercise page(chapter 6)

* **Multiple Choice Exercise-** In this type of exercise. The user is shown a video of a gesture and is asked to choose the matching English word by clicking on it.After clicking on one of the words, the user’s answer is checked and he gets a feedback on his answer and is moved on to the next exercise.

  
Figure 18: Multiple choice exercise page

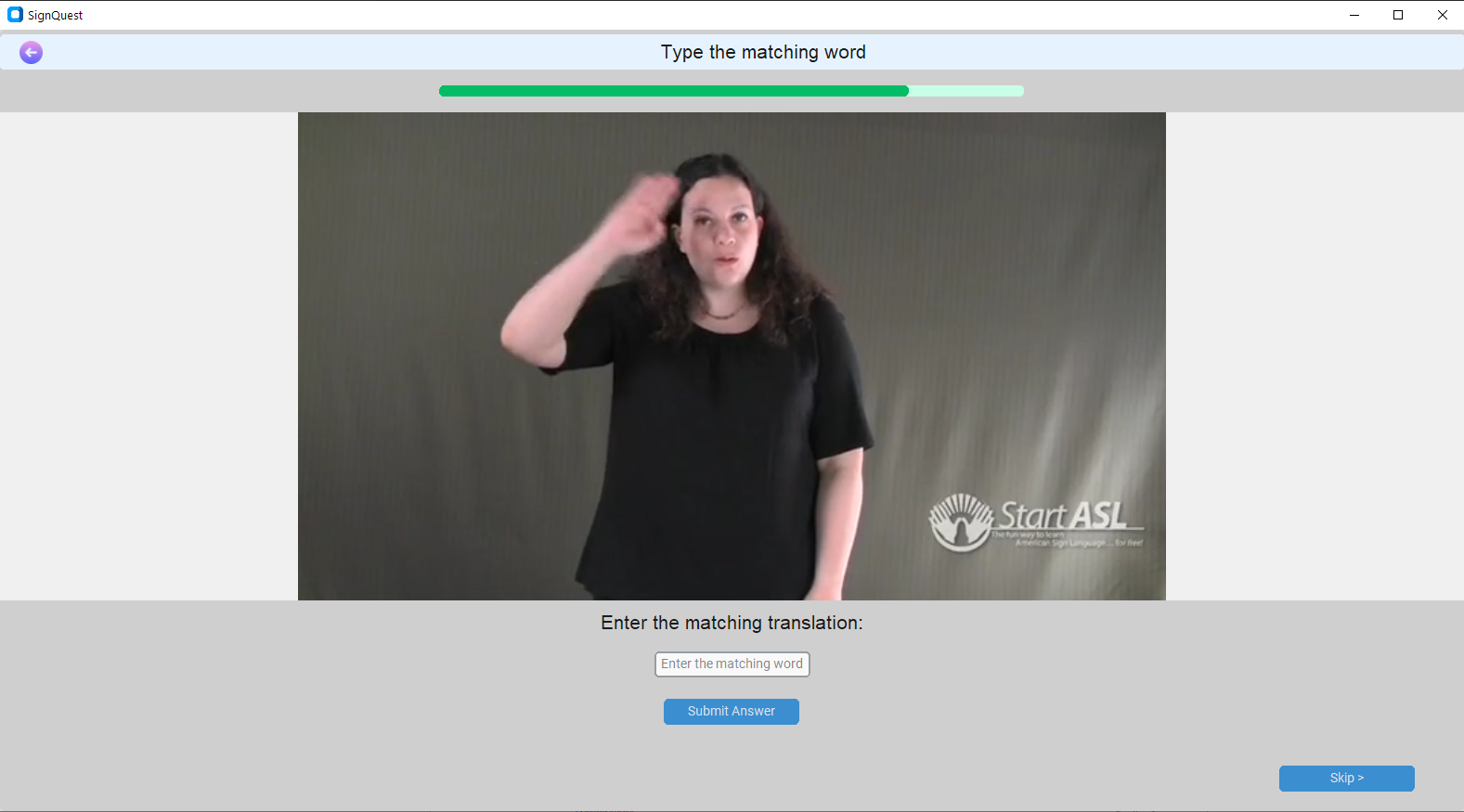
* **Type the matching word-** The user is asked to type the word that matches an ASL gesture video. After clicking on “Submit Answer” his answer is checked,he gets feedback on his answer and is moved to the next exercise

Figure 19: Type the matching word exercise page

* **Missing Word Exercise-** The user is shown a sentence with a missing word.The user needs to understand what is the missing word and then perform the matching ASL gesture in front of a camera.

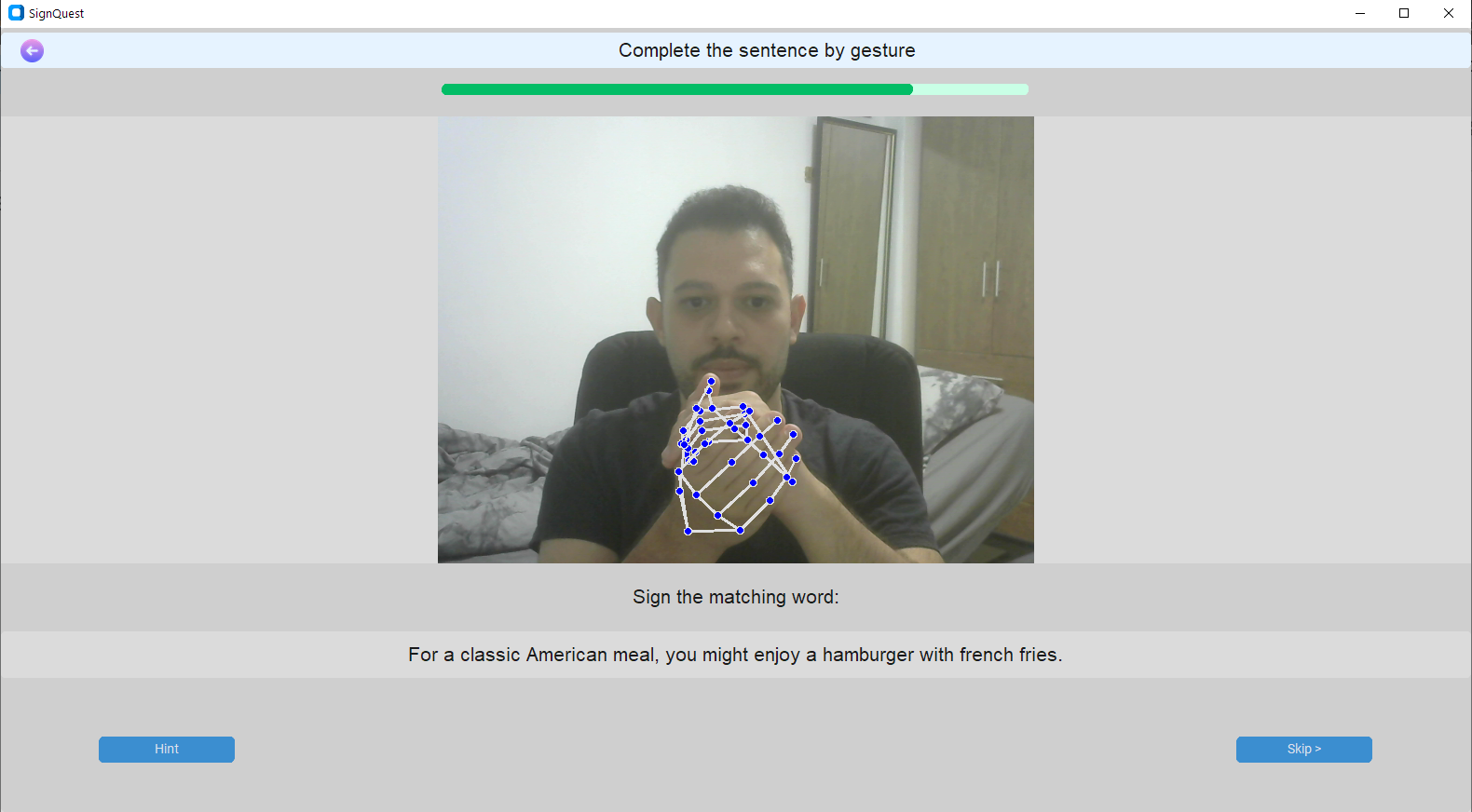
****

Figure 20: Missing word exercise page

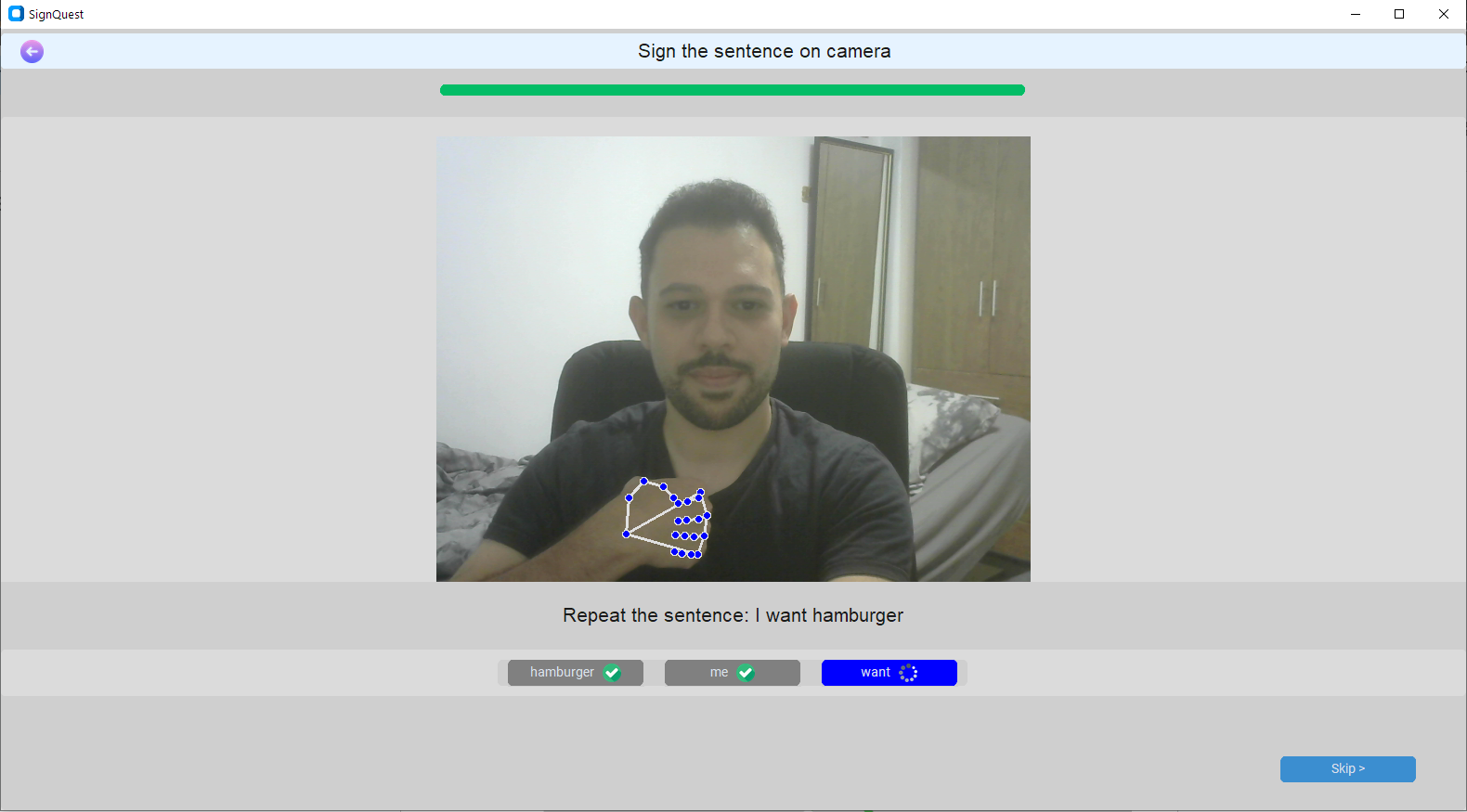
* **Sentence Exercise-** The user is shown a sentence and is asked to perform a number of ASL gestures that form the matching ASL sentence.The gestures that the user needs to perform are listed under the sentence. To help the user keep track of his progress in the exercise, the current gesture that the user needs to perform is marked in blue. The Order of words in the ASL Sentence might be different than in the English sentence and some words can be missing/altered due to the difference in the syntax between the languages(see the example below).****

Figure 21: Sentence exercise page

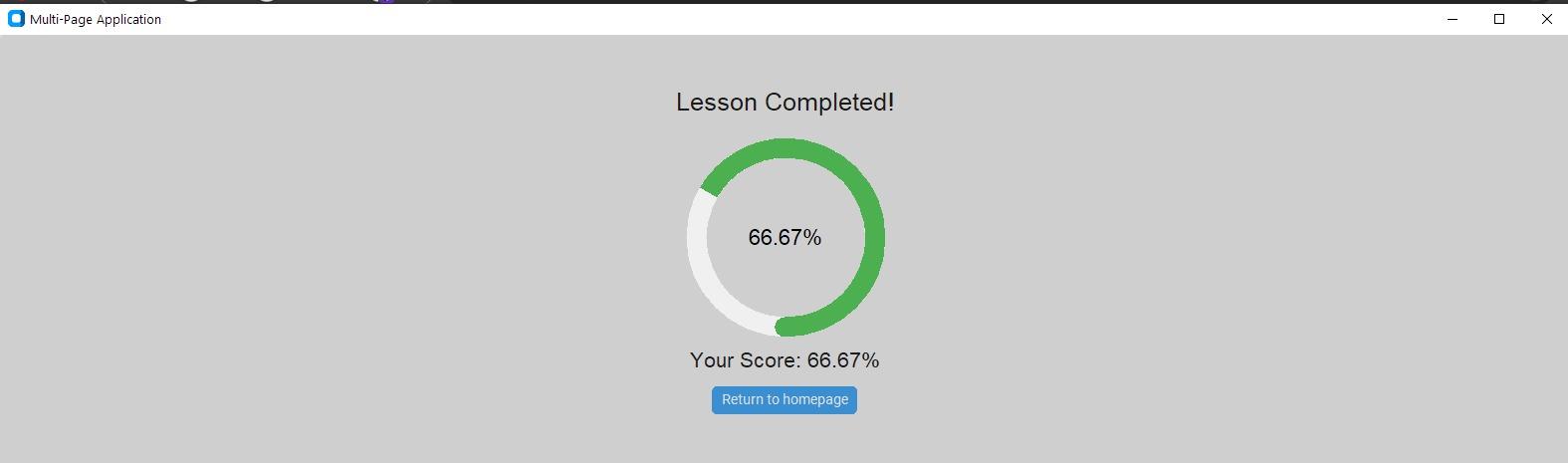
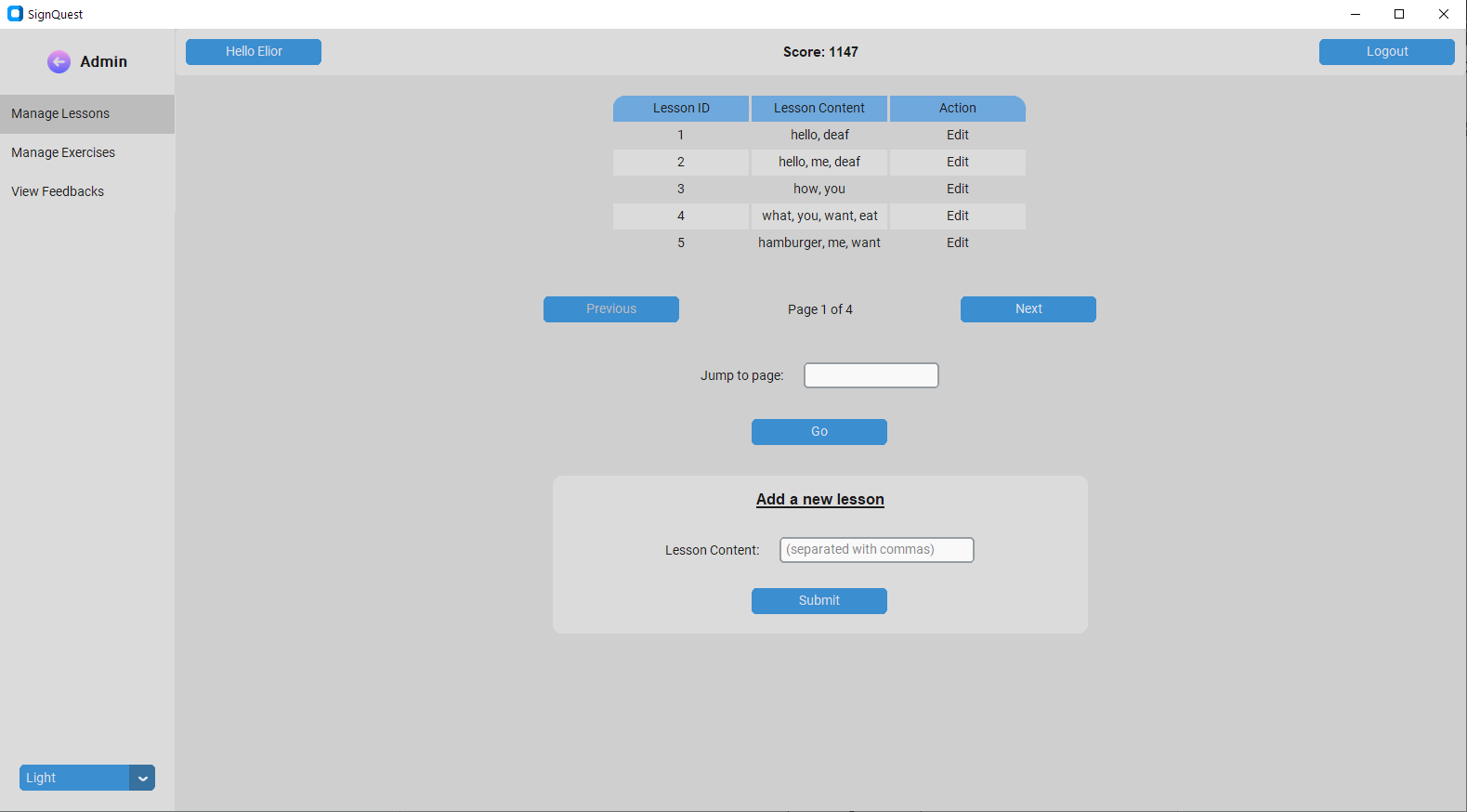
* **lesson summary-** after completing all the exercises in a lesson, the user’s score in the lesson is calculated and displayed. the user can return to the homepage from this screen by clicking on the button at the bottom.  
  ****

Figure 22: Lesson summary page

* **Admin’s Manage Lessons-** in this screen, the admin can see a table of the lessons that are stored in the DB and their contents(list of words that are learned and practiced in the lesson).  
  The user can edit the Lesson Content by clicking on Edit in the row of the lesson in the table.  
  The admin can navigate between the exercises pages with the next and prev buttons, or by typing a page number in the field “jump to page” and clicking on “Go”..   
  In addition, the admin can create a new lesson and add it to the DB by entering the lesson content, the list of words that will appear in the lesson in the “ADD a new lesson” section at the bottom of the screen. Clicking on “Submit” will save the lesson in the database.

  
Figure 23: Admin’s manage lesson page

* **Admin’s Manage Exercises-** In this screen the admin can view a table of the exercises that are stored on the DB, including their IDs, the IDs of the lessons they belong to and their types.  
  The admin can navigate between the exercises pages with next and prev or buttons or by typing a page number in the field “jump to page” and clicking on “Go”.   
  The admin can also Delete an exercise by clicking on the “Delete” button in the exercise’s row.  
  In addition, the Admin can edit an exercise by clicking on “Edit”.  
  The “Edit” button will open an exercise edit screen.

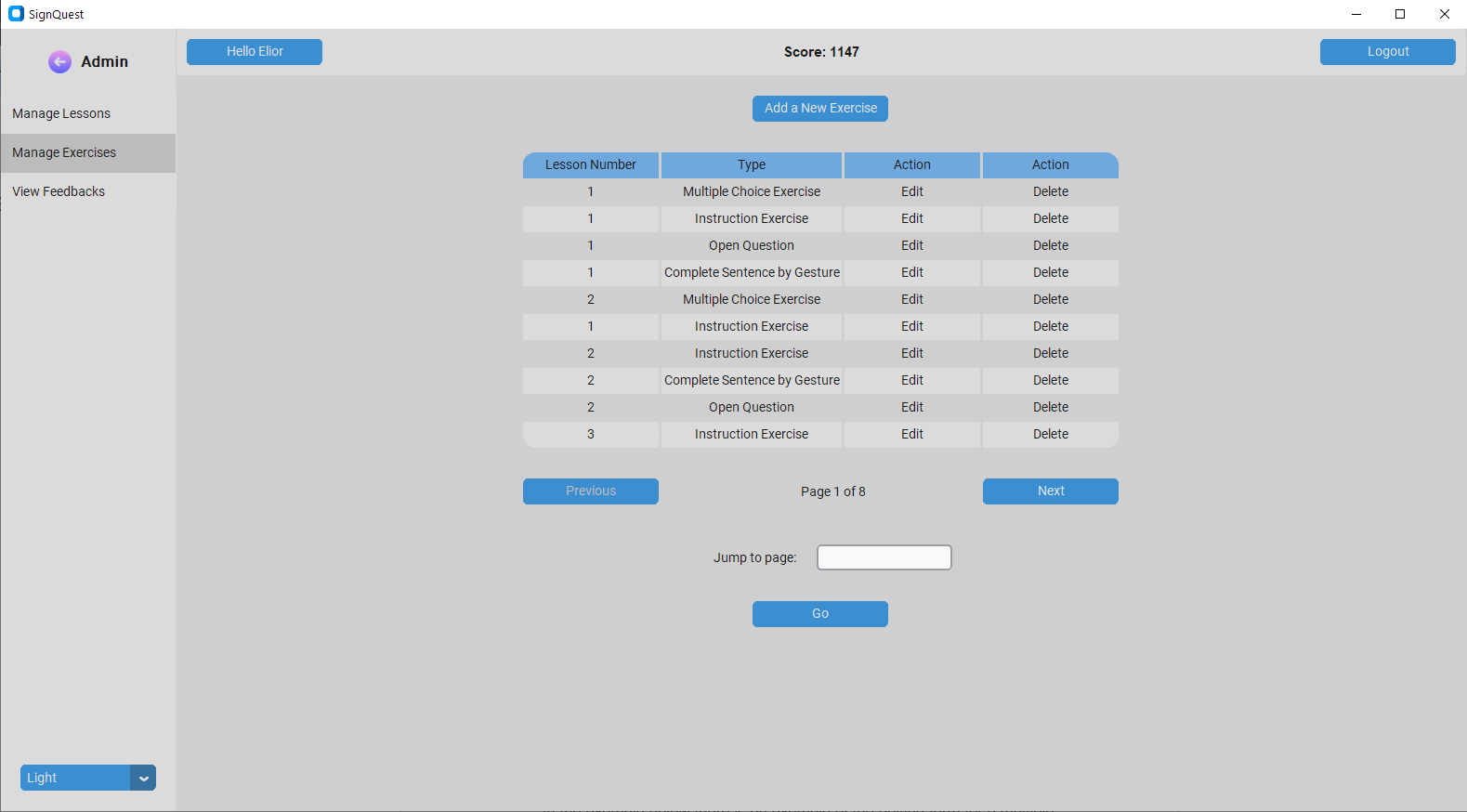
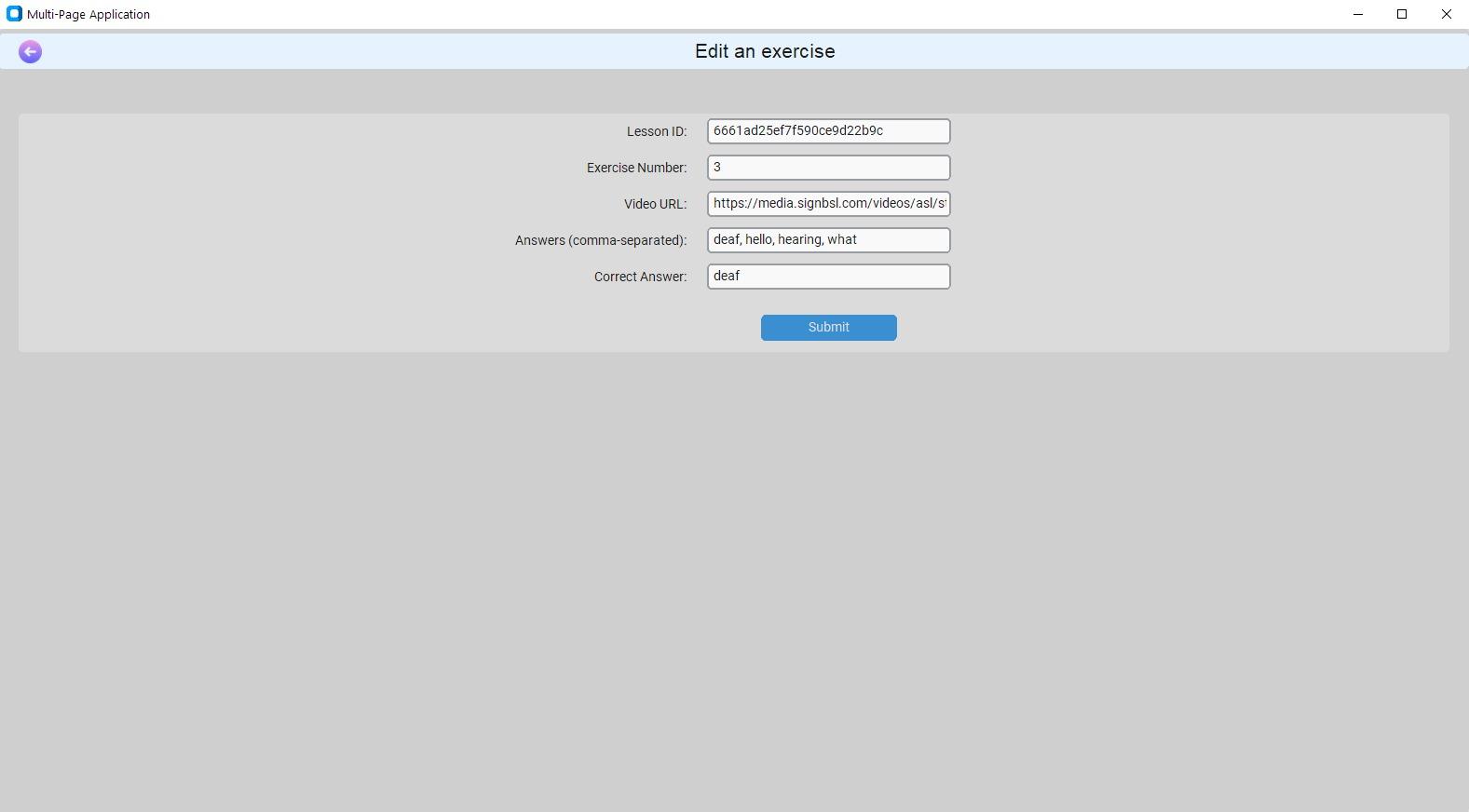
The admin can create a new exercise by clicking on ”Add a new Exercise” at the top of the screen, which will open the adding an exercise screen.

Figure 24: Admin’s manage exercises page(chapter 6)

* **Admin’s Edit Exercise -** After the admin chooses an exercise to edit in the “Manage Exercises” screen, the exercise editing screen is opened and an editing form is shown to the user according to the exercise’s type.

Below there is an example of the editing form for a multiple choice exercise. In the form the admin can edit the lesson id of the exercise, the exercise number(within the lesson order), the url of the video that is shown in the exercise, the list of possible answers and the correct answer. To save the changes the admin can click on “Submit”.

  
Figure 25: Edit an exercise page

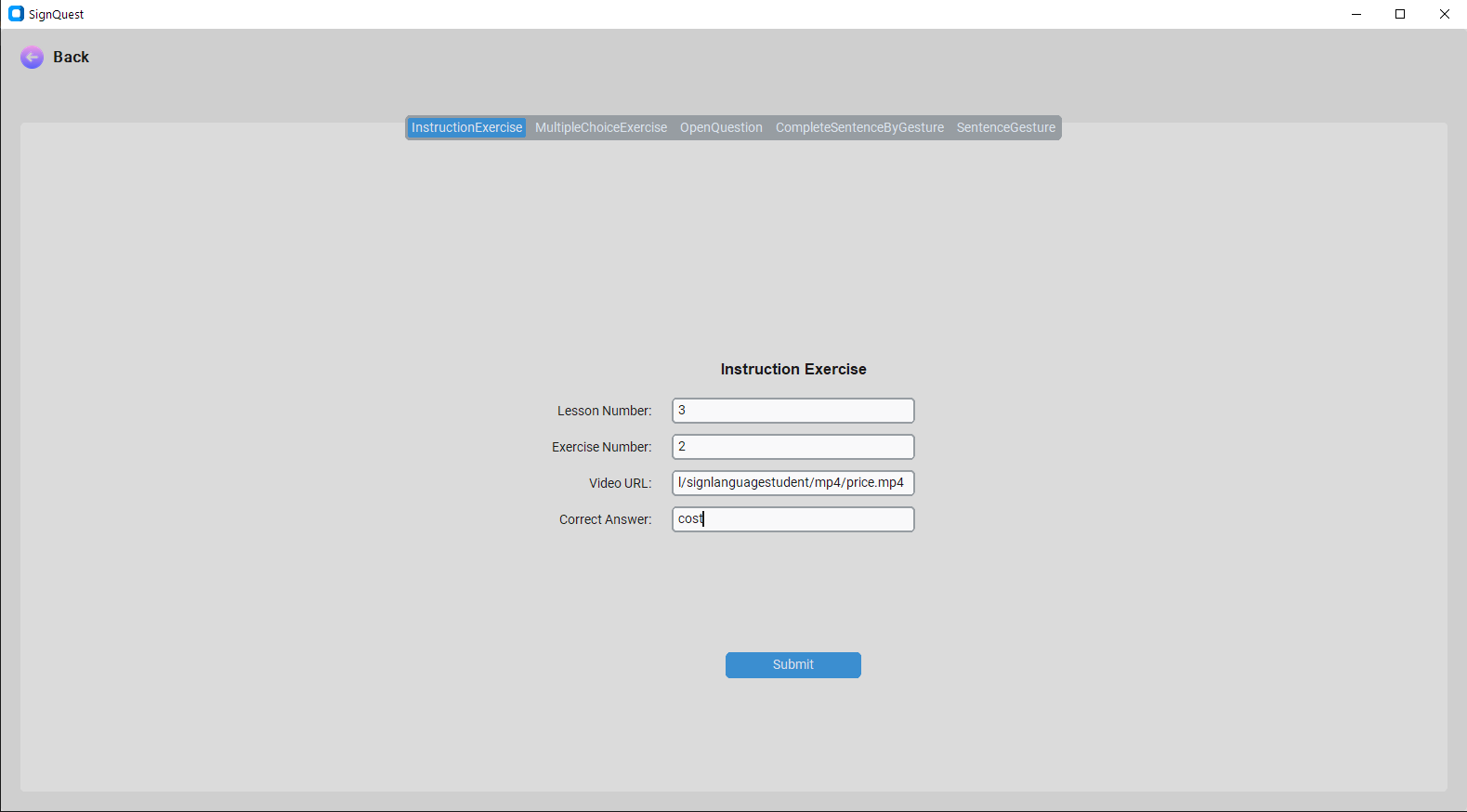
* **Adding a new exercise** - In this screen the admin can add an exercise to the DB. in the toolbar at the top of the screen the admin can choose the type of the exercise he wants to add. After clicking on the exercise type, a form for entering the details of the exercise will appear. In the example below you can see the form for creating an instruction exercise (repeat the sign) where the admin needs to enter the lesson id, exercise number, an asl gesture video url and the gestures’ matching English word (“Correct Answer”). 

Figure 26: Add a new exercise page

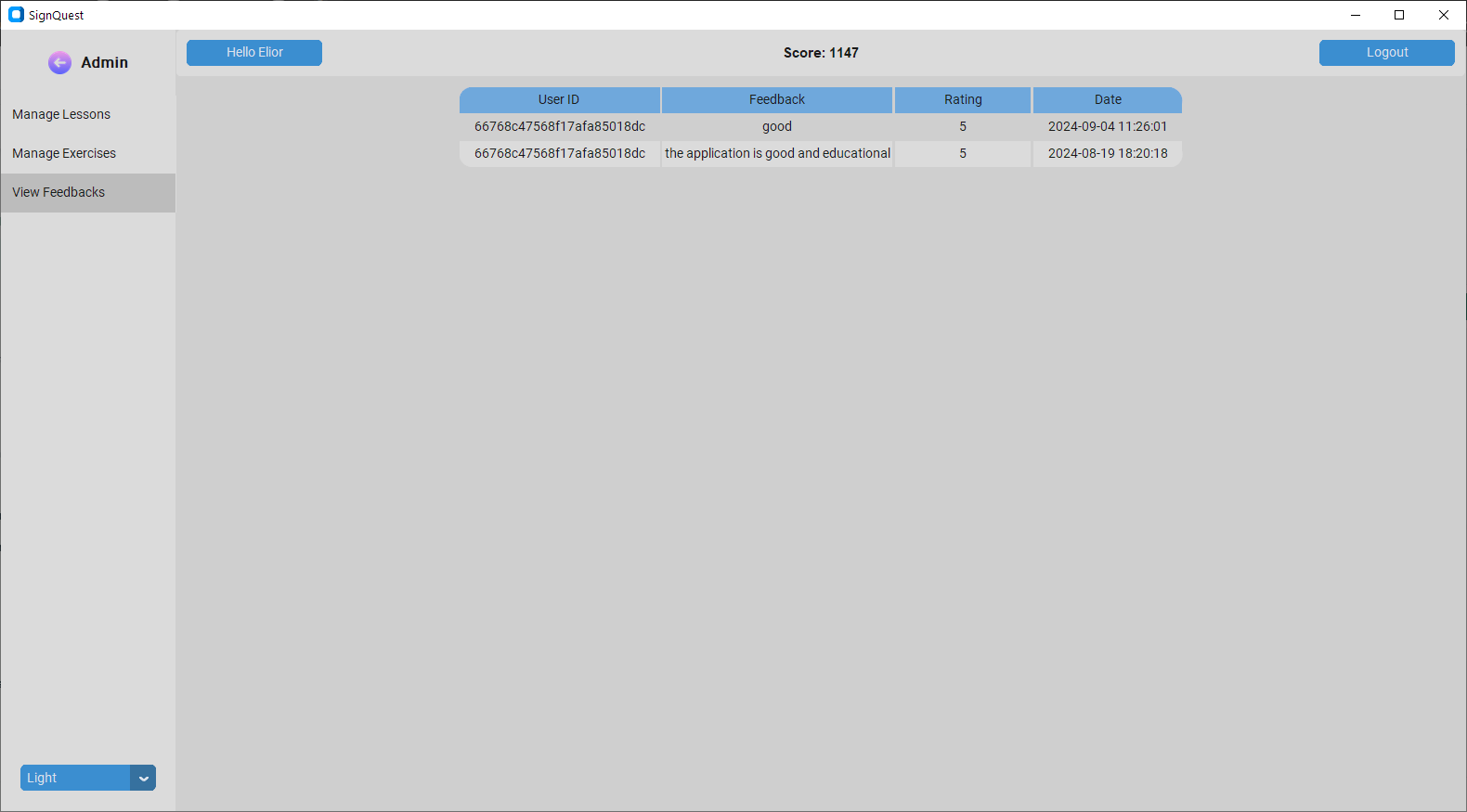
* **Admin’s view feedbacks** - In this screen the admin can view the feedbacks sent by users, including the IDs of the users that sent the feedbacks, the feedbacks, the rating each user gave the app and the dates the feedbacks were submitted

Figure 27: Admin view feedback page

# 7. Maintenance Guide

In this section we will describe what are the requirements in terms of software and hardware that must be installed for using or developing our system

## 7.1 Hardware and software requirements

* Python 3.10.9
* Windows OS
* Webcam
* Installation of the required libraries using **“pip install -r requitements.txt”**

The main file to run is **app.py**

## **7.2 Training a model**

First, you need to create a dataset.  
The dataset has to include videos that are divided into folders. The name of the folders should be the labels(words) of the gestures in the videos.

### 7.2.1 Data augmentation (optional)

if you don’t have enough videos per class(word) you can use data augmentation.

There a 3 different data augmentation techniques you can use:

* **Shifting video** - each of the original videos in the dataset is shifted horizontally by various relative distances: 0.05 and 0.1 to both the left and right.
* **Zoom video** -all of the videos in the dataset are zoomed by 1.1.
* **Mirror video** - all the videos are mirrored (flip horizontally).

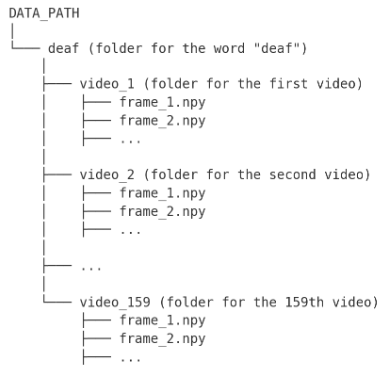
The relevant files are in the folder “**data\_augmentation**”, in each file there's a variable for the folder path of the videos, variable name is “**input\_directory**”.

### 7.2.2 Collecting hand key points from the videos

Aftercollecting the dataset, you need to collect the hand key points from the videos in the dataset using MediaPipe. you can do that by running the file collectKeyPoints.py. You need to update the variables **VIDEOS\_PATH** and **DATA\_PATH,** where **VIDEOS\_PATH** is the folder that stores the videos and **DATA\_PATH** is where you want to store the hand key points data.

This process creates a folder with the name in the **DATA\_PATH** variable.

The folder includes folders with the names of the words in the dataset.  
Each internal folder includes a folder for each video of the word.  
In each video folder there are NPY files that include the key points for each frame in the video.

  
Figure 28: DATA\_PATH folder diagram

### 7.2.3 Training

For training a model, open the file **train.py**,update the variable **DATA\_PATH** to the folder where you saved the hand key points data and run the file. After the training process will end, graphs of the loss and accuracy and a confusion matrix will be shown in the plots section in the dev environment. In addition, a summary of the model will be printed in the console. The weights of the model are saved in a **keras** file.

## **7.3 Package Diagram**

Figure 29: Package diagram

# 8. References

1. Li, D., Opazo, C. R., Yu, X., & Li, H. (2019). Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1910.11006>
2. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019, June). Mediapipe: A framework for perceiving and processing reality. In *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)* (Vol. 2019).‏
3. Sundar, B., & Bagyammal, T. (2022). American sign language recognition for alphabets using MediaPipe and LSTM. *Procedia Computer Science*, *215*, 642-651.‏

# Appendix

Figure 30: User feedback form