COMPUTER INFORMATION SYSTEMS DEPARTMENT

&

INFORMATION TECHNOLOGY DEPARTMENT

KING ABDULLAH II SCHOOL OF INFORMATION TECHNOLOGY

THE UNIVERSITY OF JORDAN

HANDSPEAK

HandSpeak

King Abdullah II School of Information Technology (KASIT)

Departments of Computer Information Systems and

Information Technology

Project Supervisors: Dr. Hamad Alsawalqah, Dr. Bashar Alshboul, Dr. Reem Alfayez and Dr. Hamed Al-Bdour

Name	ID
Ammar Awad Mohammad Hasan	0212135
Dana Mohammad Sabri Qattom	0215345
Duaa Ayed Othman Al-Qudimat	0216235

ABSTRACT

HandSpeak is an innovative initiative aimed at bridging the communication gap between Arabic spoken language and Arabic sign language by leveraging artificial intelligence. The project utilizes advanced machine learning techniques, particularly MediaPipe for extracting hand and body key points and a Long Short-Term Memory (LSTM) neural network for gesture recognition and real-time translation. The primary objective is to develop a model that translates sign language gestures into text in Arabic, enhancing accessibility and inclusivity for the Deaf community.

The system design includes data preprocessing, model training, and evaluating. MediaPipe provides efficient, real-time detection of hand and pose landmarks, while the LSTM model processes sequential data to ensure accurate recognition of static and dynamic gestures. The model also supports offline functionality, prioritizing usability in regions with limited internet connectivity.

Key features include real-time translation, and accessibility options tailored for diverse users. Evaluation of the system demonstrated high accuracy, with a categorical accuracy of 98.67% during training and 97.06% during validation. This project not only promotes social inclusion by empowering the Deaf community but also lays a robust foundation for future enhancements such as gesture-to-speech translation and 3D visualizations of sign language gestures.

ACKNOWLEDGMWNTS

First, we would like to thank Allah for everything, and we would also like to thank our graduation project supervisors (Dr. Bashar Alshboul, Dr. Hamad Alsawalqah, and Dr. Reem Alfayez), who guided us working on this project (HandSpeak), and provided us with information, and a great environment for professional work, and encouraged us to do a lot of research.

Also, we would like to thank our families for helping us working on this project's requirements within the requested period, and our friends for being helpful.

Table of Content

LIST OF FIGURES

LIST OF TABLES

RAM	Random Access Memory
DHH	Deaf and Hard-of-Hearing
ROI	Return On Investment
NFR	Non-Functional Requirement
DFD	Data Flow Diagram
AI	Artificial Intelligence
LSTM	Long Short-Term Memory (neural network architecture)
BiLSTM	Bidirectional Long Short-Term Memory
RNN	Recurrent Neural Network
YOLO	You Only Look Once
FPS	Frames Per Second
OpenCV	Open-Source Computer Vision Library

LIST OF SYMBOLES AND ACRONYMS

1.0 CHAPTER ONE: INTRODUCTION

1.1 Project Motivation

Due to the small number of people who can read sign language, deaf people frequently experience severe communication difficulties. This leads to difficulties in accessing essential services, social exclusion, and a lack of equal opportunities. An AI-powered model that translates sign language into natural language can bridge this communication gap, fostering better interaction and inclusivity. Such a model would empower deaf people by enabling smooth communication, allowing them to access necessary resources and engage more fully in society.

1.2 Problem Statement

Significant obstacles still exist when interpreting sign language, despite technological advancements, a lack of sign language expertise and a slow turnaround time for translators. The process of learning sign language can take a while. By using machine learning algorithms, computer vision to develop a model that interprets and recognizes sign language gestures, this project seeks to address these problems. By enhancing digital accessibility and providing real-time translation, this solution will ensure equal communication access for the hearing-impaired community.

1.3 Project Aim and Objectives

The main aim of HandSpeak is to develop an artificial intelligence model to bridge the communication gap between Arabic spoken language and Arabic sign language by translating sign language gestures into text in real-time.

The project has several key **objectives**:

1. Understand how sign language works, and study existing solutions and algorithms related to this project.
2. Collect and prepare a reliable dataset of sign language gestures. Then, preprocess the data to ensure it is ready for training the model.
3. Choose and train a machine learning model to ensure accurate and consistent predictions.
4. Evaluate the system by testing its accuracy, usability, and real-world performance to ensure it meets user needs and expectations.
5. Document the entire development process, including design, implementation, and testing, to provide a clear and organized reference for future improvements and similar projects.

1.4 Project Scope

The scope of this project is to develop an artificial intelligence model that translates sign language into words in real-time, enhancing communication and accessibility for the hearing-impaired community.

1. **Ensure Real-Time Translation:** Implement real-time processing to provide immediate translation of sign language to natural language.
2. **Support Arabic Language**: Focus on providing accurate translation for gestures specific to Arabic sign language, catering to the needs of the Arabic-speaking community.
3. **Bridging the communication barriers** between deaf people and the community.

1.5 Project Software and Hardware Requirements

1.5.1 Software Requirements

Table 1: Software Requirements

Software	Requirements
Development Tools	Jupyter Notebook, Figma
Libraries	TensorFlow, Keras, MediaPipe, OpenCV
Programming Languages	Python 3.10
Other Tools	JSON Viewer

1.5.2 Hardware Requirements

Table 2: hardware Requirements

Hardware	Requirements
CPU	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
Hard Disk	512GB SSD
Memory (RAM)	16 GB
Laptop	HP laptop

1.6 Project Limitations

• Language-Specific

This system will be designed to translate Arabic sign languages.

• Environmental Factors

The accuracy of image recognition can be significantly influenced by environmental conditions such as lighting, background distractions, and camera angles. Variations in these factors can affect the system's performance.

• Real-Time Processing Constraints

Balancing the need for rapid processing with the demand for precise recognition can be challenging.

• Complexity of Sign Language Dynamics

Sign languages involve dynamic movements. Systems that do not fully account for these small differences may fail to accurately capture and translate the full meaning of signs.

• Data Collection

Gathering a comprehensive dataset that includes diverse and representative examples of Arabic sign languages can be challenging.

1.7 Project Expected Output

• **Sign Translation:** The system should translate recognized signs into corresponding written words in real-time.

• **Accurate Model:** The machine learning model should achieve high accuracy in recognizing and translating sign language gestures.

• **Fully Documented Process:** The entire development process, from data collection to deployment, should be thoroughly documented to provide a reference for future improvements or similar projects.

1.8 Project Schedule

Table 3:Project Schedule

Task	Description	Start time	End time	Duration	Dependency
T1	**Planning**	14/10/24	17/10/24	4	
T2	**Information gathering**	18/10/24	23/10/24	6	T1
T3	**Analysis**	24/10/24	05/11/24	12	T2
T4	**Design**	06/11/24	25/11/24	20	T3
T5	**Development & Implementation**	26/11/24	25/12/24	30	T4
T6	**Testing**	26/12/24	30/12/24	5	T5
T7	**Documentation**	31/12/24	09/01/25	10	T1,T2,T3,T4,T5
T8	**Prepare presentation**	10/01/25	11/01/25	2	T7
T9	**Delivery**	12/01/25	12/01/25	1	T7,T8

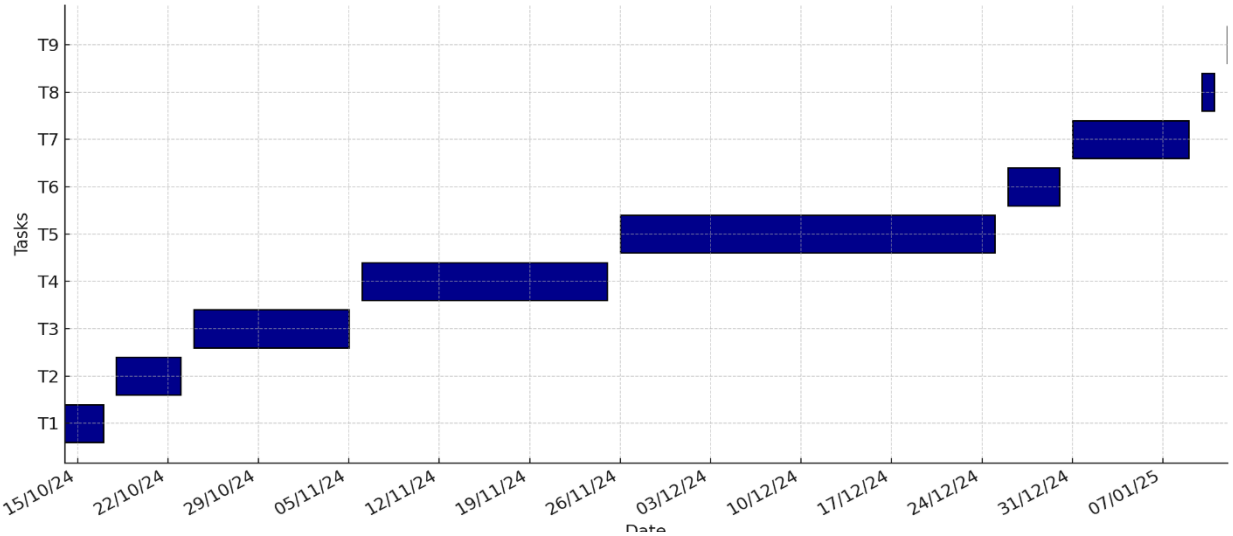

Figure 1: Gantt Chart

1.9 Dataset

After extensive research and evaluation of various datasets, we concluded that KArSL is the most comprehensive and reliable dataset available for Arabic Sign Language (ArSL). KArSL is the largest video dataset specifically designed for word-level ArSL recognition, widely used across Arab countries. It contains 502 isolated sign words, each performed by three professional signers and repeated 50 times per signer[1].

1.10 Report Outline

Chapter one introduces the project in Section 1.1, elaborating on the project motivation in Section 1.2. The problem statement is presented in Section 1.3, followed by the project aim and objectives in Section 1.4. Section 1.5 outlines the project scope, while Section 1.6 discusses the software and hardware requirements. Additionally, Section 1.7 highlights the project limitations. The project's expected output is detailed in Section 1.8, and the project schedule is described in Section 1.9. Finally, the report outline is presented in Section 1.10.

Chapter two begins with an introduction in Section 2.1, elaborates on existing systems in Section 2.2, and identifies the overall problems of the current systems in Section 2.3. The general solution approach is provided in Section 2.4, while the summary of the chapter is included in Section 2.5.

Chapter three introduces the system requirements and analysis in Section 3.1. Requirements elicitation techniques are explained in Section 3.2, and the targeted users and stakeholders are addressed in Section 3.3. Functional requirements are detailed in Section 3.4, while Section 3.5

[1] (KArSL Dataset, n.d.)

highlights the non-functional requirements. The chapter concludes with the summary output in Section 3.8.

Chapter four provides an introduction in Section 4.1, followed by a detailed explanation of the system's design components. The context diagram is elaborated in Section 4.2, and the data flow diagram (DFD) is presented in Section 4.3. Section 4.4 discusses the entity relationship diagram (ERD), while Sections 4.5 and 4.6 present the UML use case diagram and sequence diagram, respectively. The UML class diagram is detailed in Section 4.7, and the graphical user interface (GUI) design prototype is included in Section 4.8. Finally, the chapter concludes with a summary in Section 4.9.

Chapter five focuses on system implementation, starting with an introduction in Section 5.1. The process of model building is described in Section 5.2, while Section 5.3 explains code implementation, including computer vision and Flutter code. Database implementation is discussed in Section 5.4, and Section 5.5 addresses the graphical user interface implementation. The chapter concludes with a summary of the implementation in Section 5.6.

Chapter six highlights system testing and installation, beginning with an introduction in Section 6.1. System testing is detailed in Section 6.2, which includes model evaluation metrics, optimization techniques, and recognition testing. Heuristics evaluation is discussed in Section 6.3, while cooperative evaluation procedures are elaborated in Section 6.4, including pre-evaluation, evaluation, and post-evaluation procedures. Section 6.5 provides details on system installation, and Section 6.6 includes a user manual. The chapter concludes with a summary in Section 6.7.

Chapter seven concludes the report with an introduction in Section 7.1, summarizing the key outcomes of the project. The overall weaknesses of the project are outlined in Section 7.2, while Section 7.3 highlights its strengths. Future work and potential improvements are discussed in Section 7.4, and the chapter concludes with a summary in Section 7.5.

2.0 CHAPTER TWO: RELATED EXISTING SYSTEMS

There are two main approaches to hand gesture recognition: Contact-based and Vision-based. Contact-based methods involve the use of sensors on a glove to extract information about hand rotations, acceleration projections, and finger bending angles. This approach can achieve high accuracy, especially after a calibration process to adapt the sensors to the user's hand. However, it can be costly and may not lead to a natural interaction. On the other hand, Vision-based methods use visual devices such as stereo cameras, time of flight cameras, or Kinect sensors to extract depth information and create a 3D representation of the scene. Monocular systems with a single RGB camera have also been used in recent periods. These methods are generally cheaper and more adaptable than contact-based methods. Moreover, a relevant number of studies are tackling the problem from the point of view of behavioral analysis and theory of mind. Over the years, various methods have been proposed for hand gesture recognition. These range from the simplest method of wearing a colored glove that is recognized by a video camera, to methods that use skin color recognition followed by hand shape recognition. More advanced methods involve the use of machine learning, such as Skeleton-Based Recognition and Deep-Learning Based Recognition. Both contact based and vision-based methods have their advantages and disadvantages, and the choice of which method to use depends on the specific application and environment. Vision-based methods are typically used in human-computer interaction and human-robot interaction applications, while contact-based methods are more commonly used in wearable devices for control purposes.

Hand gesture technology has two primary areas of application, which are **sign language recognition and video gaming**. Sign language is a means of communication for individuals who are unable to speak, and it involves a sequence of hand gestures that represent letters, numbers, and expressions. Researchers have proposed several approaches for sign language recognition, including the use of gloves or uncovered hand interaction with a camera using computer vision techniques to identify gestures. In contrast, video gaming utilizes hand and body movements to interact with the game. **The Microsoft Kinect Xbox is an excellent example of gesture interaction for gaming purposes,** as it employs a camera placed over the screen that connects with the Xbox device through the cable port to track the user's hand and body movements.

2.1 Existing systems

1. **Data Gloves**

A data glove is supplied with sensors and can directly acquire important data, such as finger bent degree, wrist orientation, and hand motion. The data glove is the input channel in these systems and transmits data to a mobile phone. There are some limitations regarding these systems. For instance, it is difficult for a user wearing a data glove to capture hand and finger movements, and the gloves cannot acquire data such as the expressions of the face, lip-perusing, and the movements of the eyes. Moreover, a data glove can be affected by the environment, like user's location, background conditions, and collected data. In the data-glove method, the signer wears an electronic glove with sensors which detect and transmit information. Most sign language Translation systems on the market use the data-glove form because it is easy to acquire information about the degree of finger flexing of the hand.

Advantages

- Data gloves can perceive fingerspelling and sign motions, which include **static and dynamic** signs.
- Smart gloves involve **wireless** mode, **lightweight**, **reliable**, and **easy to use**, **need less computational power**, and **translation is easier to accomplish.**

Disadvantage

- **expensive** costing over $9000 US Dollars, and a less expensive glove can be more **vulnerable** to noise. Also, a cheaper data glove may have a limited number of sensors.

- **data-glove** is somehow **less comfortable** for the signers because the hands of each user are different sizes.

2. **Visual Based methods**

YOLO is a state-of-the-art object detection framework that excels in speed and accuracy by processing an entire image in a single forward pass through a neural network. Unlike traditional object detection methods that involve region proposal generation and classification in separate steps, YOLO unifies these into a single neural network model, enabling real-time performance. YOLO divides an input image into a grid and predicts bounding boxes, class probabilities, and confidence scores simultaneously for each grid cell. This architecture makes it ideal for real-time applications, such as hand gesture recognition, where it can detect and localize hands or other objects quickly and accurately. YOLO models, like YOLOv4, YOLOv5, and YOLOv8, have been widely adopted for their robustness and flexibility, and they can be fine-tuned for specific tasks using custom datasets.

Advantages

- **Real-Time Performance**: YOLO achieves high frame rates (30+ FPS on modern hardware), making it suitable for real-time applications like gesture recognition, gaming, and surveillance.
- **Unified Architecture**: Its single-step detection pipeline is simpler and more efficient compared to multi-step methods like Faster R-CNN.
- **High Accuracy**: YOLO achieves strong localization and classification performance, especially when trained on high-quality datasets.
- **Flexibility**: It can be fine-tuned for various tasks, including hand gesture recognition, sign language interpretation, and AR/VR applications.
- **Wide Adoption**: Extensive documentation, pretrained models, and support across deep learning frameworks make YOLO easy to use and integrate.

Disadvantages

- **Struggles with Small Objects**: YOLO's grid-based approach can miss small objects, as their features may get lost during down-sampling.
- **Fixed Number of Predictions**: YOLO can predict only a limited number of bounding boxes per grid cell, which might not handle crowded scenes effectively.
- **Dependency on Dataset Quality**: Like all deep learning models, YOLO heavily relies on the quality and diversity of the training data. Poorly annotated or imbalanced datasets can lead to suboptimal performance.
- **Overlapping Objects**: YOLO can struggle with accurately detecting overlapping or occluded objects, which is common in hand gesture scenarios.
- **Post-Processing Complexity**: While fast, YOLO often requires additional post-

processing, such as Non-Maximum Suppression (NMS), to refine its detections.

2.2 Overall Problems of Existing Systems

challenges in sign language detection include **the lack of large-scale labeled datasets** that accurately represent real-world settings, such as variability in signers, background, lighting, and inter-signer variation. This limitation hinders the development of accurate sign language recognition models. Additionally, the **complexity of translating** gestures based on acquired sensor data poses a challenge in machine learning-based sign language recognition using wearables. Conventional methods that utilize data fusion and mapping/lookup tables increase programming complexity. Furthermore, the recognition of visual sign languages requires **addressing the variability in hand shape, motion profile, and position of the hand, face, and body parts** contributing to each sign.

2.3Overall Solutions Approach

Our project **will improve simplicity, accuracy, and variability in datasets** for effective sign language detection and recognition in real time by using machine learning algorithms through our trained model. Most sign language translators support only small, specific domains, so we will work hard to include a broader scope so that more people can benefit from our project. In addition, our project will provide several features, including high sensitivity, high stretchability, and low cost.

Comparison of Hand Gesture Recognition Systems: Data Glove, YOLO, and MediaPipe

Table 4:Comparison of Hand Gesture Recognition Systems: Data Glove, YOLO, and MediaPipe

Aspect	YOLO	MediaPipe
Advantages	- General-purpose object detection, suitable for diverse tasks.	- Specialized for hand tracking and pose estimation.
	- Real-time performance on modern GPUs (~30+ FPS).	- Extremely fast and efficient, even on mobile devices (~30-60 FPS).
	- High accuracy for detecting objects in complex scenes.	- No training required; works out of the box for hands and poses.
	- Extensive customization for detecting various gestures.	- Robust to occlusion, lighting changes, and complex backgrounds.
Disadvantages	- Struggles with small or overlapping objects.	- Limited to hands, poses, and face tracking without further extension.
	- Requires labeled datasets and fine-tuning for hand gestures.	- Not versatile for general object detection beyond its predefined tasks.
	- Higher computational requirements; depends on GPUs for real-time use.	- Limited flexibility for integrating with custom gesture recognition tasks.
	- Needs additional models for detailed gesture classification.	- May not handle multi-object tracking as efficiently as YOLO.

3.0 CHAPTER THREE: SYSTEM REQUIREMENTS ENGINEERING AND ANALYSIS

3.1 Feasibility Study

3.1.1 Technical Feasibility

Technical Feasibility assesses whether the technology needed to develop the AI model for translating sign language is available, reliable, and capable of meeting the project's requirements.

1. **Availability of Technology:**
    - The model will leverage existing technologies such as computer vision and machine learning for gesture recognition and language translation.
    - These technologies are mature and widely used in various applications, ensuring that the necessary tools and frameworks are available for development.
2. **Integration Capabilities:**
    - The model must integrate various components, including gesture recognition, language translation, and user interface design, into a seamless system.
3. **Development Expertise:**
    - The development team should possess expertise in AI, machine learning, which are essential for building a robust and efficient system.
    - Access to skilled developers and AI researchers will ensure that the project is technically feasible and that any challenges can be effectively addressed.

3.1.2 Operational Feasibility

Operational Feasibility evaluates whether the proposed AI model can be successfully operated and maintained within the intended environment, considering the needs of users and stakeholders.

1. **User Adoption:**

- The model is designed to meet a clear and significant need within the Deaf community and among those who interact with them, making it likely that users will adopt and continue to use the model.

2. **Support and Maintenance:**

    - A robust support system will be established to address user issues, provide updates, and maintain the model's functionality.
    - Regular updates will be rolled out to improve performance, add new features, and ensure compatibility with the latest devices and operating systems.

3. **Operational Efficiency:**

    - The model's offline mode will allow users to continue using its core functions without requiring constant internet access, enhancing its operational feasibility in areas with limited connectivity.

3.1.3 Economic Feasibility

Item	Cost
Server Hosting	600$/year
Customer Support	300$/year
Maintenance	200$/year
Total	**1100$/year**

Development Costs:

Table 5: Development Cost

Employee Position	Cost
Machine Learning Engineer	2500$
Data analyst	500$
QA Tester	1200$
Total	**4200$**

Hardware and Software Costs:

Table 6: hardware & Software Costs

Item	Cost
Development Hardware	1400$
Testing Devices	800$
Total	**2200$**

Development Cost = 4200+2200 =6400$

Table 7: Operational Costs

Operational Costs:

Personal Costs:

Table 8: Personal Costs

Employee	Salary per Hour
Developers	35$/ hour
Tester	40$/hour
Technical Support	15$/hour

Tangible Benefits:

Table 9: Tangible Benefits

Benefit	Year 1	Year 2	Year 3	Year 4	Year 5
Improved Communication	2000$	2100$	2200$	2300$	2400$
Increased Efficiency	3000$	3100$	3200$	3300$	3400$
Cost Reduction	1500$	1600$	1700$	1800$	1900$
Error Reduction	1200$	1300$	1400$	1500$	1600$
Total Tangible Benefits	**7700$**	**8100$**	**8500$**	**8900$**	**9300$**

Intangible Benefits:

Table 10: Intangible Benefits

Benefit	Description
Social Inclusion	Greater integration of deaf people in society
User Satisfaction	Improved quality if life satisfaction for users

Payback Analysis:

Table 11: Payback Analysis

Year	Development Cost ($)	Operating Cost ($)	Benefits ($)	Discount Factor	Adjusted Cost ($)	Adjusted Benefits ($)
0	6400	0	0	1.0000	6400	0
1	0	1100	7700	0.9091	1000.01	6999.99
2	0	1100	8100	0.8264	908.04	6694.44
3	0	1100	8500	0.7513	826.43	6386.05
4	0	1100	8900	0.6830	751.30	6088.70

1. **Payback Year**: Between the 2nd and 3rd years because the total benefits exceed the total costs in Year 3.
2. **Lifetime ROI**: (Total Benefits - Total Costs)/Total Costs= (31943.55−10568.77)/10568.77= 202%
3. **Annual ROI**: Lifetime ROI/Project Lifetime =202%/5=40.4%
4. **Net Present Value (NPV)**:
   Cumulative Benefits−Cumulative Costs=31943.55−10568.77=21374.78

3.2 Requirements Gathering Techniques

Requirements for the HandSpeak project were gathered through a survey inspired by the study Development Web-based Arabic Assessments for DHH (Deaf and Hard-of-Hearing) Students[2]. This survey involved 30 participants, including students and educators, and focused on

[2] (Atwan, Wedyan, Abbas, & Gazzawe, 2022)

evaluating system usability, accessibility, and user satisfaction. Participants were asked to assess features such as gesture recognition, accuracy, ease of use, and offline functionality. The results demonstrated high satisfaction, with an average usability score of 4.86, highlighting the importance of intuitive interfaces and dependable performance. These insights directly influenced the development of HandSpeak to ensure it meets user needs effectively.

3.3 Targeted Users and Stakeholders

3.3.1 Targeted Users

The **primary** users of the HandSpeak model are deaf people.

Secondary users include family members, friends, educators, and service providers who interact with the primary users.

3.3.2 Stakeholders

Table 12: Stakeholders

Stakeholders	Role
Deaf Community	Primary users, providing feedback and testing
Family Members	Secondary users, assisting primary users
General Users	Using the model to translate sign language to Arabic
Developers	Developing and maintaining the model
Educators	Using the model as a learning tool

3.4 Functional Requirements

Table 13: Functional Requirements

FR	Requirement	Description	Priority	Stakeholders
R1	Real-time Translation	Translate sign language to text and speech in real-time.	High	Deaf Community, Developers, General Users
R2	OpenCV	Reads video frames from the webcam and convert them to be compatible with MediaPipe and other libraries .	High	Deaf Community, General Users
R3	Gesture Recognition	Accurate recognition of hand gestures and movements.	High	Developers, General Users
R4	Text Output	Display recognized signs as text on the screen in Arabic and English.	High	Deaf Community, General Users
R5	Offline Mode	Provide functionality without requiring an internet connection.	Medium	Deaf Community, Users
R6	Accessibility Features	Include a word list that educates users about various sign language gestures.	Medium	General Users
R7	Debugging	Showing logs on the screen for feedback.	High	Developers
R8	Detect Keypoints	Detects keypoints of the body, hands, and other landmarks in real-time using computer vision.	High	Developers

R9	Draw Styled Landmarks	Visualizes detected keypoints by drawing landmarks on the screen for better clarity and debugging.	High	Developers
R10	Adjust Landmarks	Aligns and normalizes detected landmarks based on a center point for consistency in data analysis.	High	Developers
R11	Extract keypoints	Extracts keypoints for the body's pose, left hand, and right hand for further processing.	High	Developers
R12	Generate Keypoint	Generates keypoint coordinates in a structured format for input into the recognition model.	High	Developers
R13	Data Processing	Prepares and preprocesses the raw data for use in training the machine learning model.	High	Developers
R14	Splitting Training and Validation Data	Divides the dataset into training and validation sets to evaluate model performance.	High	Developers
R15	Train The Model	Trains the machine learning model using the preprocessed dataset to learn sign language recognition.	High	Developers
R16	Compile the model	Configures the model for training by defining the optimizer, loss function, and evaluation metrics.	High	Developers
R17	Early stopping	Stops the training process when the model performance stops improving to prevent overfitting.	High	Developers
R18	Model Fitting	Fits the model to the training data and monitors its performance on the validation set.	High	Developers

R19	Model Evaluation	Tests the trained model on unseen data to evaluate its accuracy, precision, and recall.	High	Developers, General Users
R20	Save Model	Saves the trained model for future use and deployment in the application.	High	Developers

3.5 Non-Functional Requirements

Table 14: Non-Functional Requirements

NFR	Requirement	Description	Priority
NFR1	Accuracy	The system should achieve at least 95% accuracy in sign language recognition.	High
NFR2	Response Time	The model should provide real-time translations with a response time of less than 2 seconds.	High
NFR3	Reliability	The system should have an uptime of 99.9% to ensure consistent availability.	High
NFR4	Usability	The model should be intuitive, with a user satisfaction rating of at least 90%.	High
NFR5	Compatibility	The model must be compatible with the latest versions of TensorFlow, MediaPipe and OpenCV.	High
NFR6	Maintainability	The system should be designed for easy updates and maintenance, with minimal downtime required for upgrades.	Medium

3.6 Usability and User Experience Goals

Usability Goals:

- **Real-Time Performance:** Deliver seamless and real-time gesture recognition to avoid delays, ensuring smooth communication without interruptions.
- **Error Tolerance:** Implement robust algorithms that can adapt to variations in gestures, lighting, and camera quality, minimizing errors in recognition.
- **Efficiency:** Optimize the model for real-time translation, ensuring fast and accurate recognition of gestures without noticeable delays.
- **Integration:** Enable seamless integration with commonly used communication tools like video conferencing platforms, making the solution more versatile.

User Experience Goals:

- **Educational Value:** Incorporate features that allow non-signers to learn basic sign language gestures, fostering inclusivity and understanding.
- **Transparency:** Provide clear feedback to users about recognized gestures, enhancing confidence in the system's accuracy.

4.0 Chapter Four: System Design

4.1 Introduction

This chapter includes many important figures that describe our model process; it will include a context diagram, data flow diagram (DFD), entity relation diagram (ERD), use cases diagrams, sequences diagrams, class diagrams.

4.2 Context Diagram:

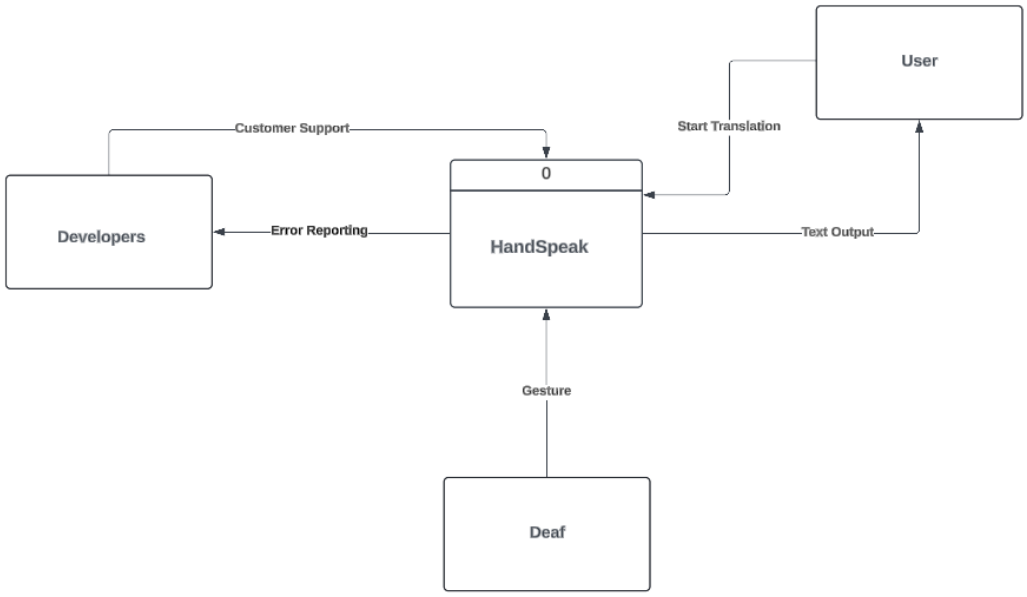

Figure 2:Context Diagram

4.3 Data Flow Diagram (DFD)

HandSpeak Model

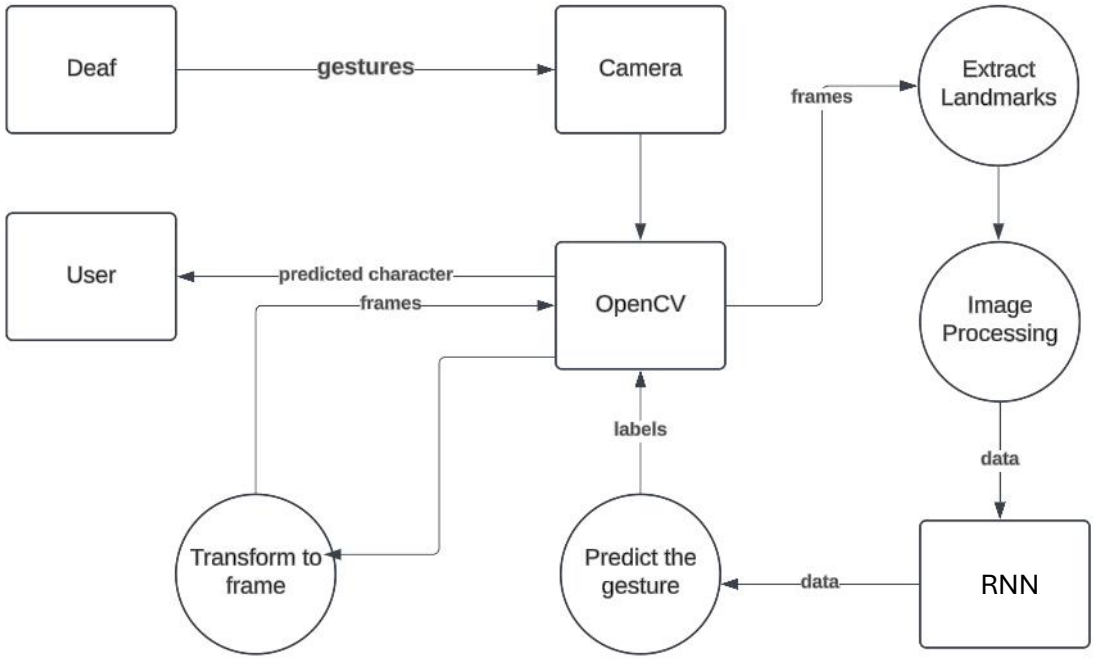

Figure 3:Data Flow Diagram (DFD)

4.4 Entity Relationship Diagram (ERD)

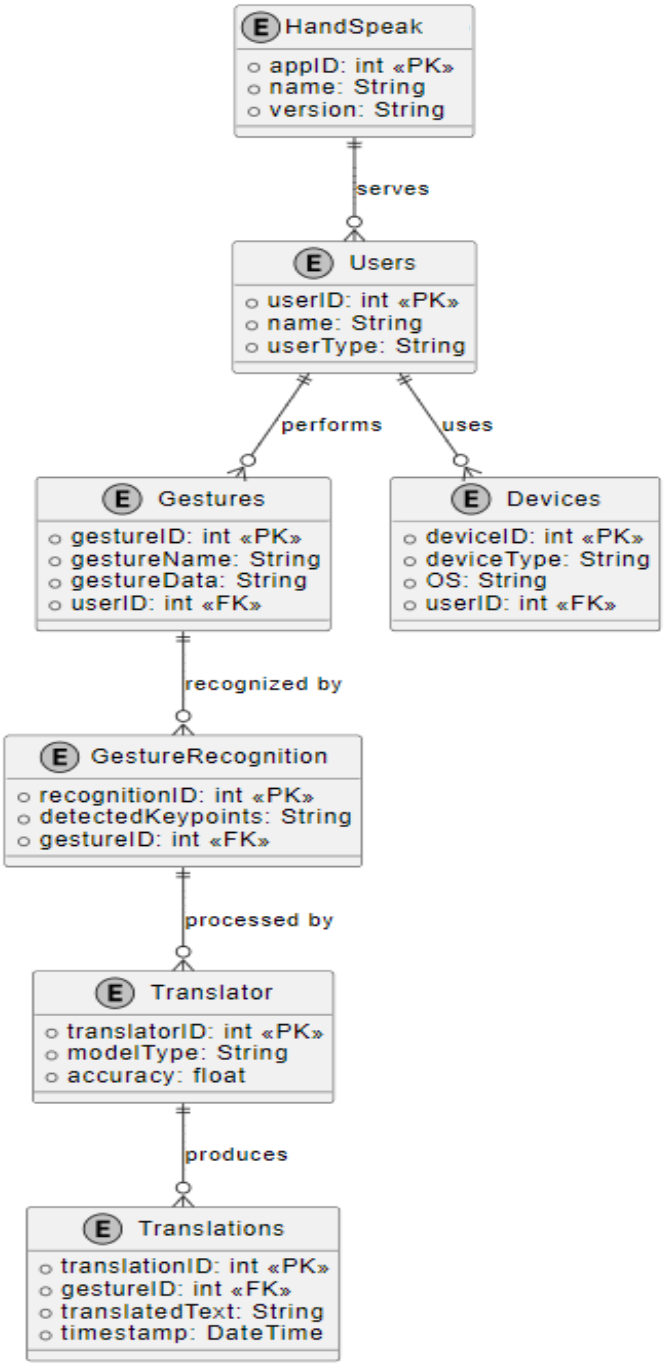

Figure 4: ERD Diagram

4.5 UML Use Case Diagram

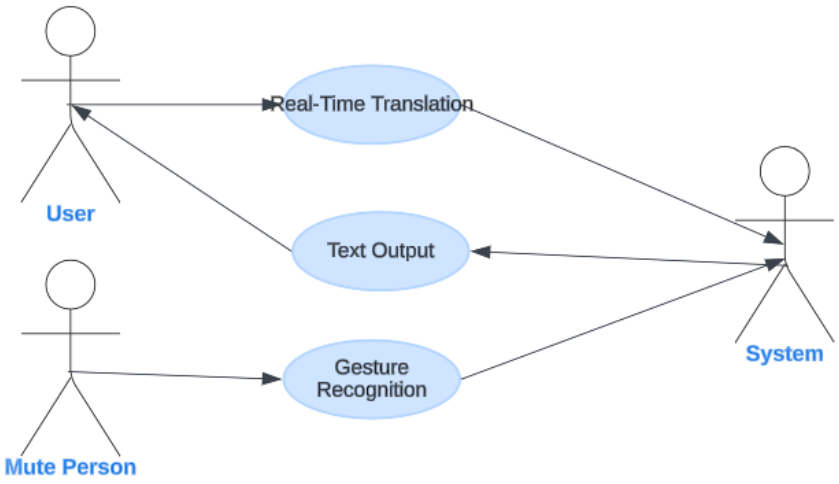

Figure 5:System Use Case Diagram1

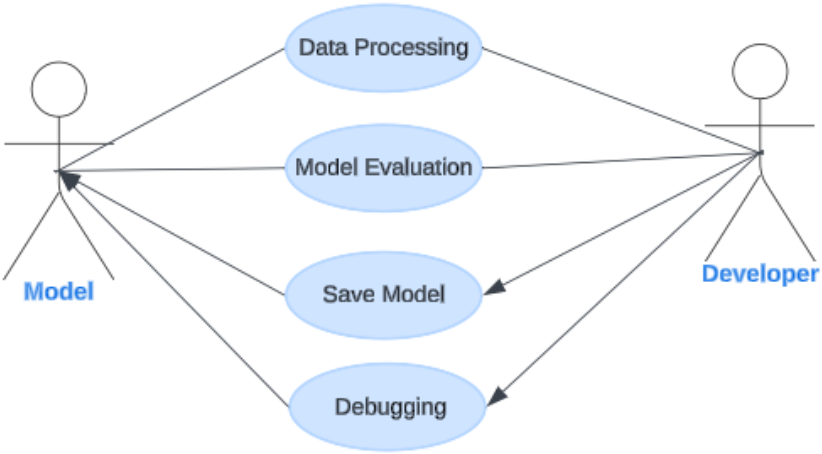

Figure 6: System Use Case Diagram2

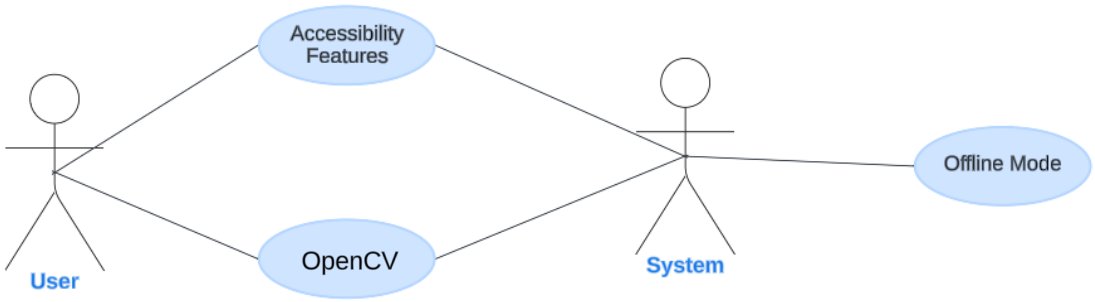

Figure 7: System Use Case Diagram3

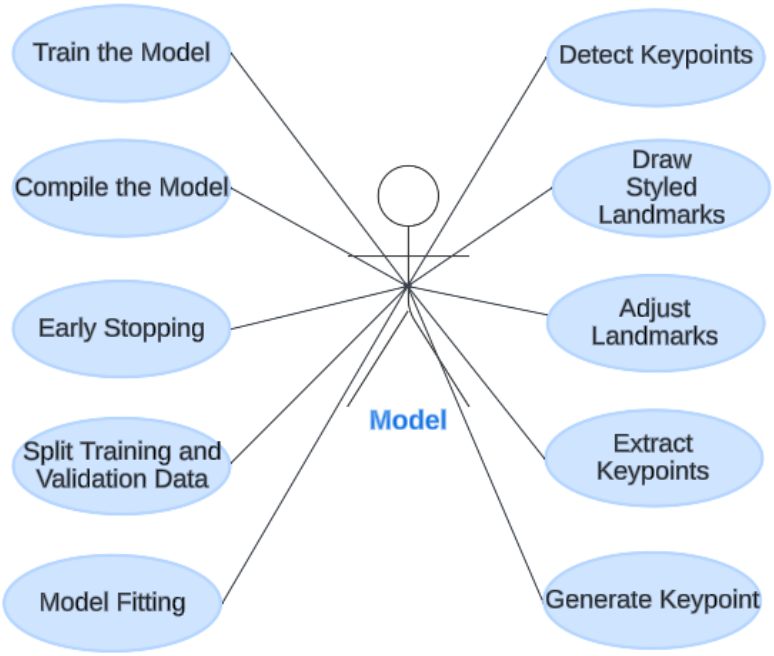

Figure 8:System Use Case Diagram5

4.6 UML Sequence Diagram

- FR1: Real-time Translation

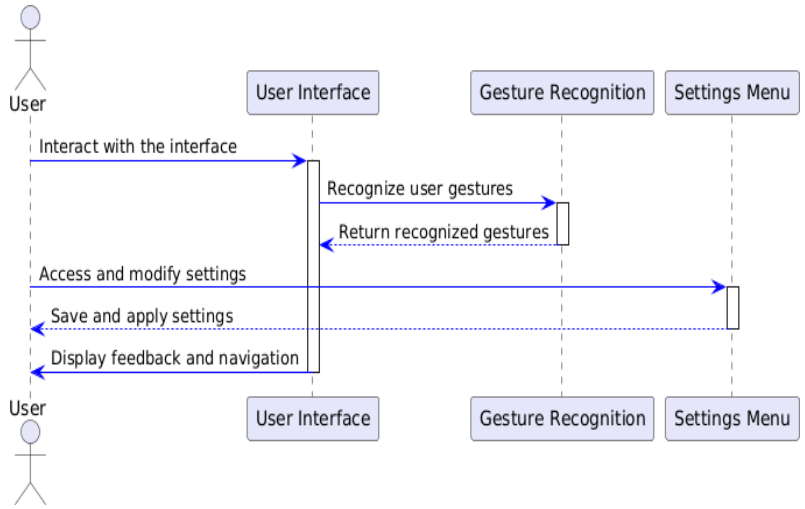

Figure 9: FR1: Real-time Translation

- FR2: OpenCV

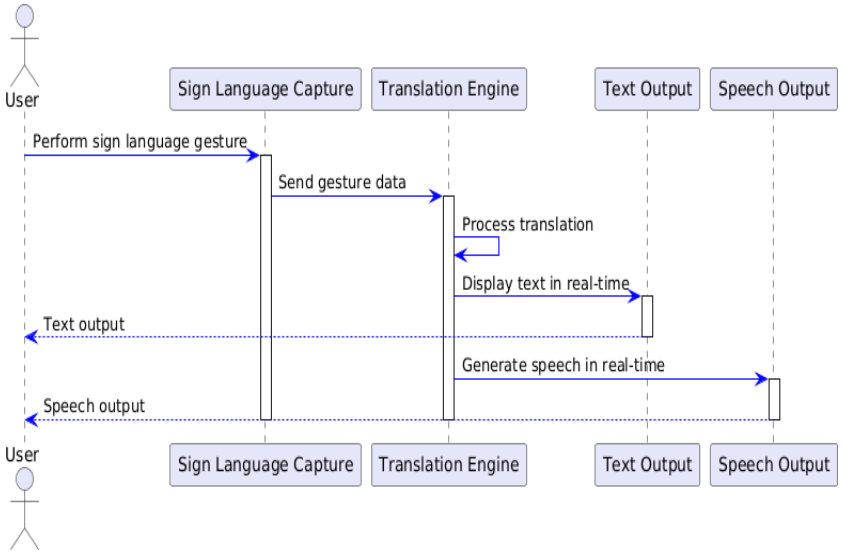

Figure 10:FR2: OpenCV

- FR3: Gesture Recognition

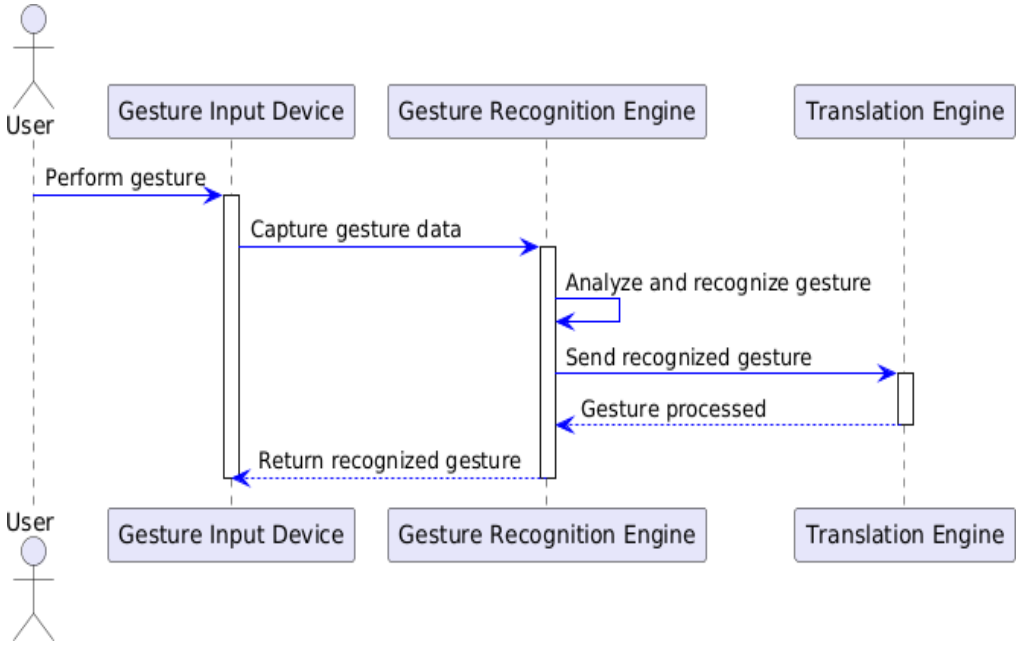

Figure 11: Gesture Recognition

- FR4: Text Output

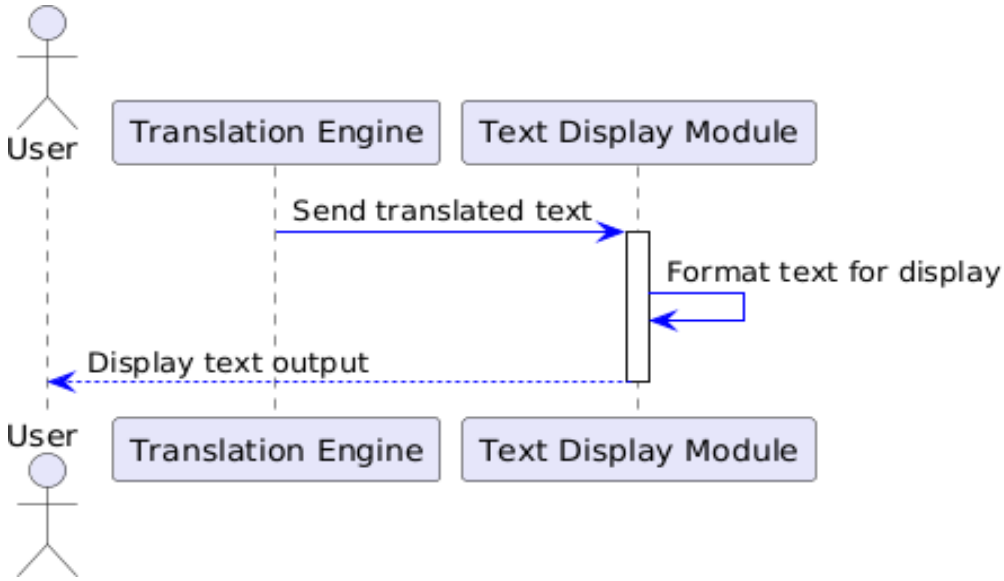

Figure 12:Text output

- FR5: Offline Mode

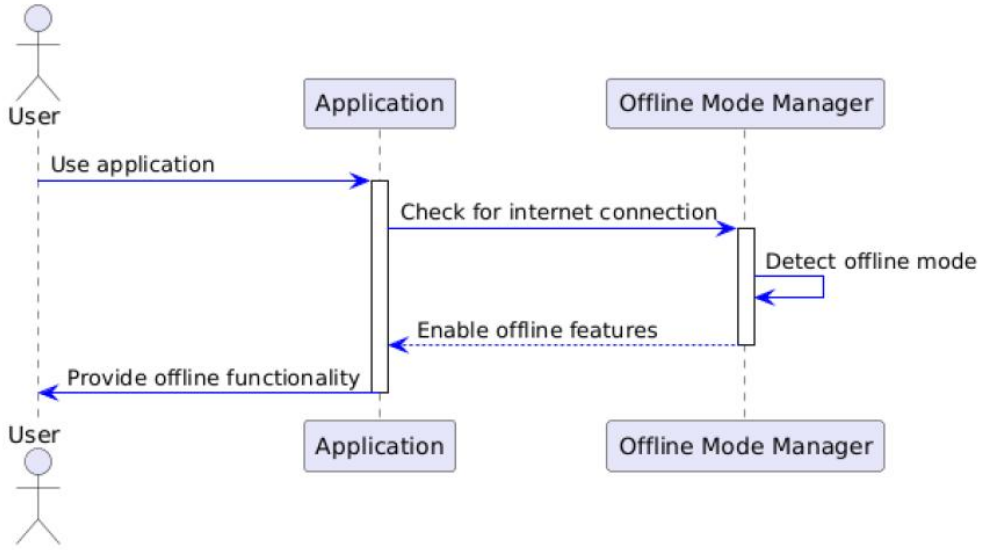

Figure 13: FR5: Offline Mode

- FR6: Accessibility Features

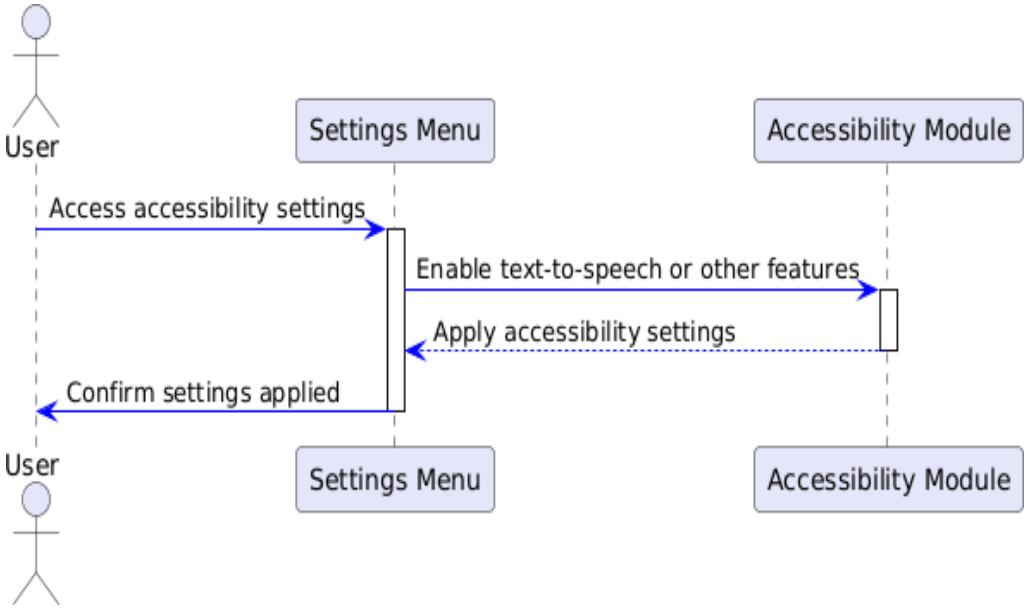

Figure 14:FR6: Accessibility Features

- FR7: Debugging

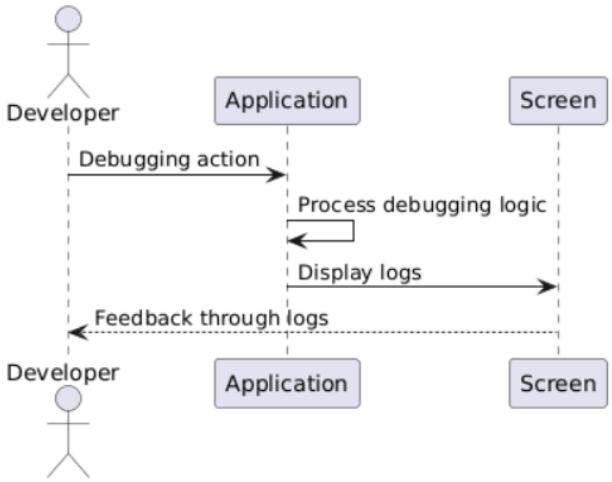

Figure 15:FR7: Debugging

- Preprocessing Workflow: FR8, FR9, FR10, FR11, FR12, FR13

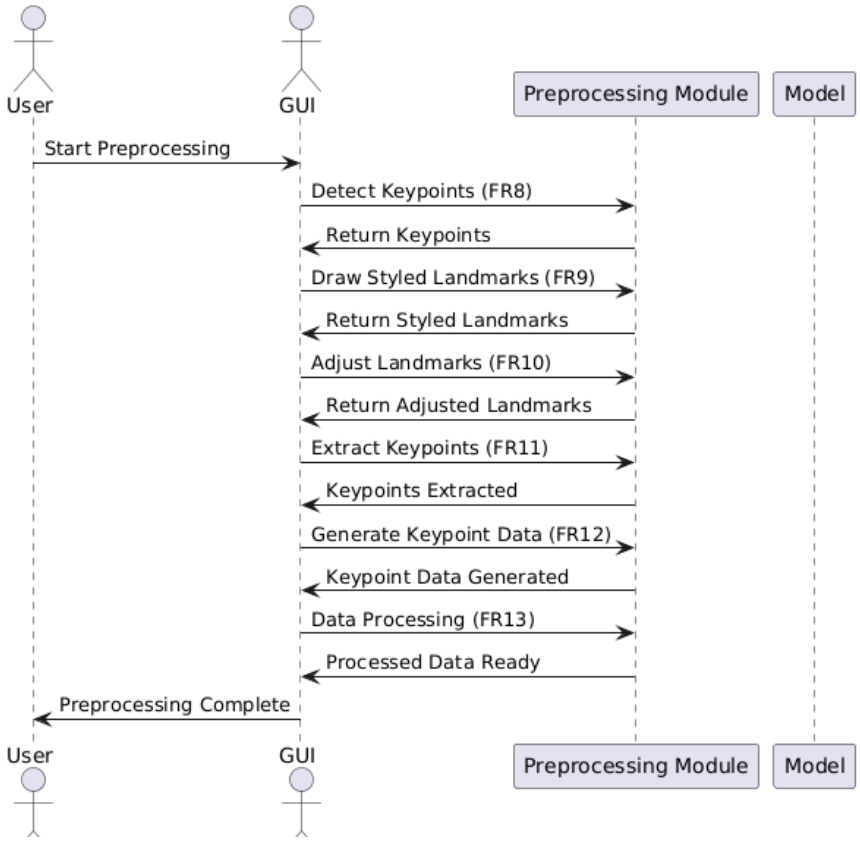

Figure 16:Preprocessing Workflow: FR8, FR9, FR10, FR11, FR12, FR13

- Training Workflow: FR14, FR15, FR16, FR17, FR18

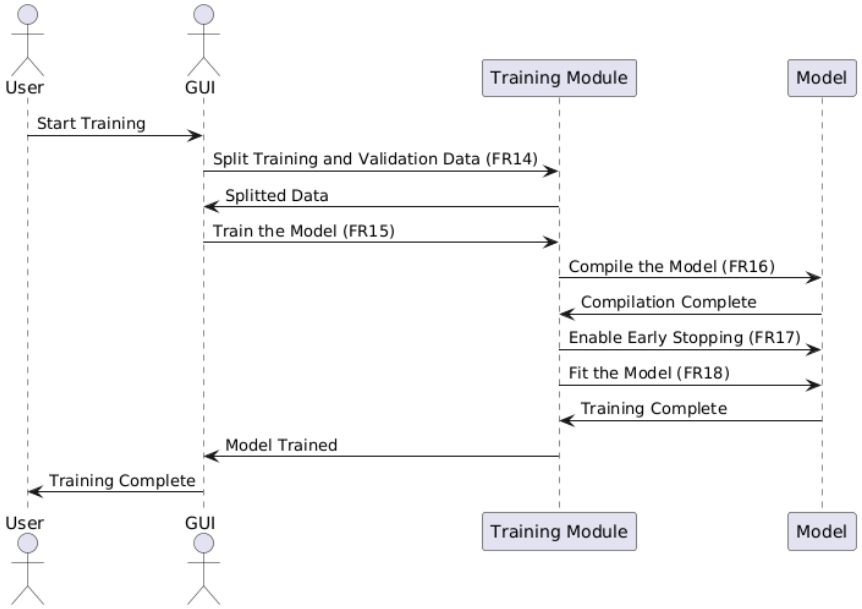

Figure 17:Training Workflow: FR14, FR15, FR16, FR17, FR18

- Evaluation Workflow: FR19, FR20

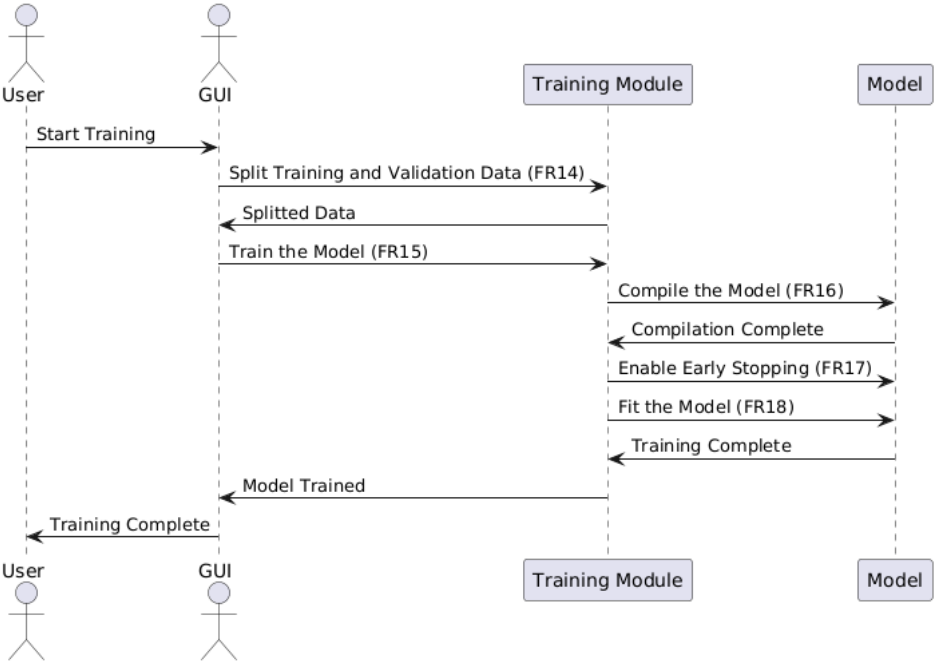

Figure 18:Evaluation Workflow: FR19, FR20E

4.7 UML Class Diagram

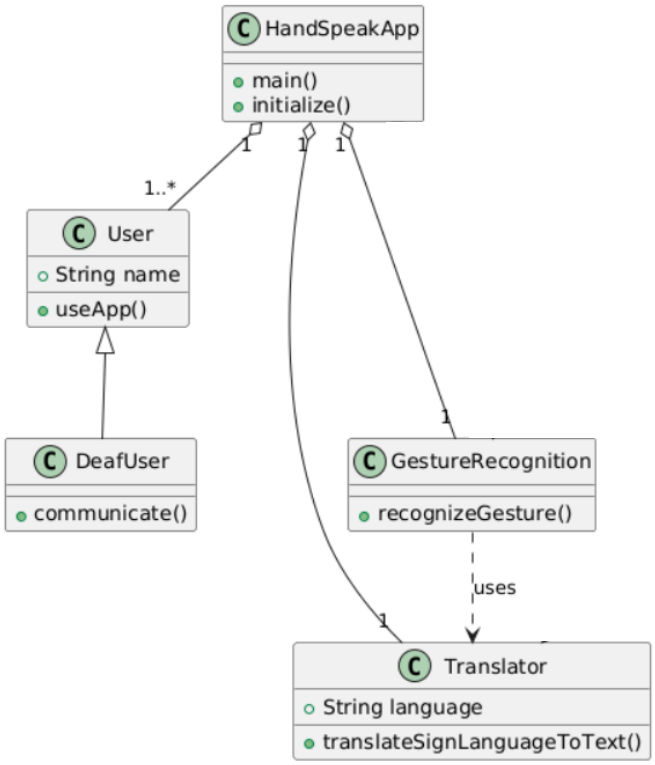

Figure 19: UML Class Diagram

4.9 Summary

In this chapter, we navigated through the essential elements of system design, employing a range of diagrams and models to articulate the architecture and functionality of the proposed system.

The journey began with an exploration of the system's context through a Context Diagram, followed by the visualization of data flow in the Data Flow Diagram (DFD). For systems with databases, the Entity Relationship Diagram (ERD) offered insights into the relationships between entities. UML diagrams, including Use Case, Sequence, and Class diagrams, provided a detailed view of system behavior and structure.

5.0 Chapter Five: System Implementation

5.1 Introduction

System implementation is an important phase in the system, at this stage designer start design the system from begging to end of the system implementation. Therefore, this chapter describes the database implementation relations, also the graphical user interface implementation figures.

- **Dataset intro**

In our project, we will be working on 502 classes, performed by 2 signers. Figure 21 presents an overview of 502 classes of KArSL[3].

Our goal is to generate 3 arrays: pose-adjusted, lh-adjusted, rh-adjusted which present respectively the adjusted keypoints for the pose, the left hand and the right hand in the input frame. Figure 22 displays the detected features by Mediapipe.

	SignID	Sign-Arabic	Sign-English
0	71	هيكل عظمي	Skeleton
1	72	جمجمة	skull
2	73	عمود فقري	Backbone
3	74	قفص صدري	Chest
4	75	جهاز تنفسي	Respiratory device
...	...	...	...
95	166	يشم	inhale
96	167	يصعد	rise
97	168	ينزل	descend
98	169	يفتح	open
99	170	يقفل (يغلق)	close

Figure 20: Label Map

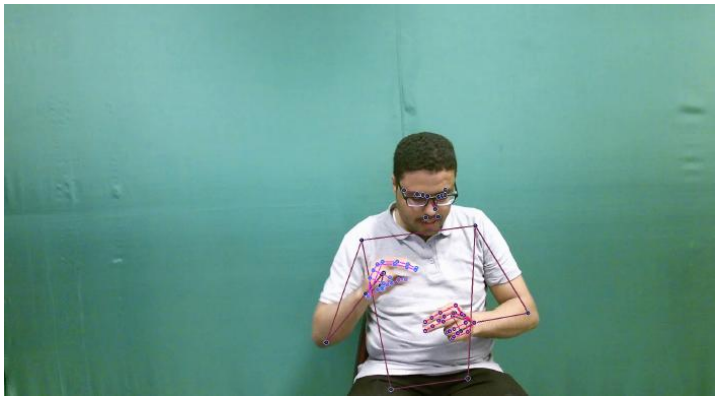

Figure 21: Sample Frame

[3] (KArSL Dataset, n.d.)

5.2 Model Development

To build our Arabic sign language model, we used BiLSTM which is based on LSTM.

In fact, LSTM is a type of recurrent neural network (RNN) architecture to model and analyze sequential data. LSTM networks can effectively handle long sequences by selectively remembering or forgetting information at each time step. This enables them to capture dependencies over longer time lags and make accurate predictions.

In our context of study, we notice that many signs in our dataset could have the same start or could end the same way, hence, we choose the BiLSTM model.

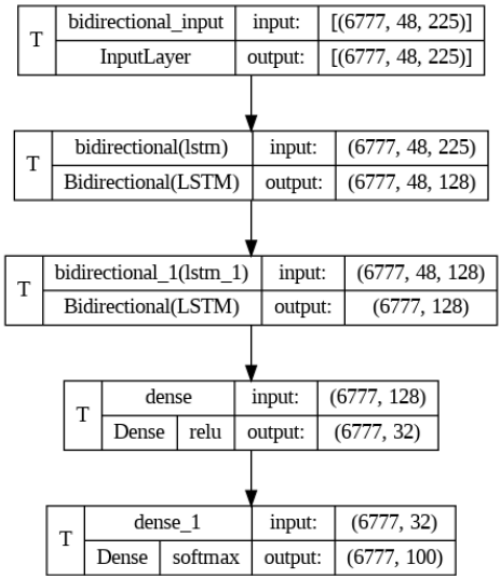

Figure 22: BiLSTM model architecture

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 48, 128)          148480
 l)

 bidirectional_1 (Bidirectio  (None, 128)              98816
 nal)

 dense (Dense)               (None, 32)                4128

 dense_1 (Dense)             (None, 100)               3300

=================================================================
Total params: 254,724
Trainable params: 254,724
Non-trainable params: 0
_________________________________________________________________
```

Figure 23: Model Summary

5.3 Code Implementation

- **Python Code**

Importing essential libraries for data processing, model building, and visualization

```
import mediapipe as mp
import tensorflow as tf
import keras
import numpy as np
import pandas as pd
import os
import shutil
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import glob
import cv2
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Bidirectional, LSTM, Dense
from keras.callbacks import EarlyStopping
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
np.random.seed(42)
```

Figure 24: Python Code Implementation1

Function to Detect Keypoints Using MediaPipe

```
def mediapipe_detection(image, model):
    """
    Perform landmark detection on an image using a specified MediaPipe model.
    """
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False                  # Image is no longer writeable ; to improve performance during processing
    results = model.process(image)
    image.flags.writeable = True                   # Image is now writeable for further processing
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results
```

Figure 26:Python Code Implementation2

Function to Extract and Adjust Keypoints from Detected Landmarks

```
def extract_keypoints(results):

    """
    Extracts and adjusts keypoints for the pose, left hand, and right hand from the results object.
    """

    pose = np.array([[res.x, res.y, res.z] for res in results.pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33*3)
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if results.left_hand_landmarks else np.zeros(21*3)
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() if results.right_hand_landmarks else np.zeros(21*3)

    # Set reference points for centering landmarks
    nose=pose[:3]
    lh_wrist=lh[:3]
    rh_wrist=rh[:3]
```

Figure 25:Python Code Implementation3

Function to Generate Keypoint Arrays for Videos

```python
def make_keypoint_arrays(path,signer,split):
    """This function generates numpy arrays of keypoints for each video in the specified folder location.
    Args:
      signer(int): the signer of interest. Could be 01 or 02 or 03
      split(str): can be 'train', 'test' or 'val'
    """
    #Creates folders to store the generated keypoints for each signer and dataset split.
    os.makedirs('working/npy_arrays',exist_ok = True)
    os.makedirs(f'working/npy_arrays/{signer}',exist_ok = True)
    os.makedirs(f'working/npy_arrays/{signer}/{split}',exist_ok = True)
```

Figure 27:Python Code Implementation4

Model

```python
# Define the Bidirectional LSTM model with Attention


model = tf.keras.Sequential([
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(len(words), activation='softmax')
])

# Compile the model

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])

# Set up early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',  # Metric to monitor for early stopping
    mode='min',  # Set mode to 'min' for minimizing the metric
    patience=5,  # Number of epochs with no improvement before stopping
```

Figure 28:Python Code Implementation5

Computer Vision Code

```python
import cv2
import numpy as np
import mediapipe as mp
from keras.models import load_model
from PIL import Image, ImageFont, ImageDraw
import arabic_reshaper
from bidi.algorithm import get_display
import json

# Load the trained model
model = load_model('working/LSTM_Model_1.h5')

# Load label map
with open("working/label_map.json", "r", encoding="utf-8") as file:
    label_map = json.load(file)
reverse_label_map = {v: k for k, v in label_map.items()}

# Path to Arabic font file
font_path = "working/Adobe Arabic Regular.ttf"
font = ImageFont.truetype(font_path, 32)

# Initialize MediaPipe Holistic
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
```

```python
# Real-time gesture recognition with toggle functionality
sequence = []
sequence_length = 48
last_prediction = ""
collecting = False  # Flag to track whether keypoint collection is active

cap = cv2.VideoCapture(0)

with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = holistic.process(frame_rgb)

        mp_drawing.draw_landmarks(frame, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)
        mp_drawing.draw_landmarks(frame, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
        mp_drawing.draw_landmarks(frame, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)

        # Handle key press to toggle collection
        key = cv2.waitKey(10) & 0xFF
        if key == ord('s'):  # Press 's' to toggle collection
            collecting = not collecting
            if not collecting:
                # Reset sequence and clear prediction when stopping collection
                sequence = []
                last_prediction = ""
                print("Keypoint collection stopped.")
            else:
                print("Keypoint collection started.")
```

```python
        # Collect keypoints if collection is active
        if collecting:
            keypoints = extract_keypoints(results)
            sequence.append(keypoints)

            if len(sequence) > sequence_length:
                sequence = sequence[-sequence_length:]

            if len(sequence) == sequence_length:
                input_sequence = np.expand_dims(np.array(sequence), axis=0)
                prediction = model.predict(input_sequence)
                predicted_class = np.argmax(prediction, axis=1)[0]
                confidence = np.max(prediction)

                if confidence > 0.7:  # Confidence threshold
                    last_prediction = reverse_label_map[predicted_class]
                    print(f"Prediction: {last_prediction} (Confidence: {confidence:.2f})")

                # Reset sequence after prediction
                sequence = []

        # Display feedback
        status_text = "Collecting" if collecting else "Idle"
        reshaped_text = arabic_reshaper.reshape(last_prediction)
        bidi_text = get_display(reshaped_text)
        frame_pil = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        draw = ImageDraw.Draw(frame_pil)
        draw.text((10, 50), f"Status: {status_text}", font=font, fill=(0, 255, 0))
        draw.text((10, 100), bidi_text, font=font, fill=(255, 0, 0))
```

Figure 29:Python Code Implementation 6

5.4 Database Implementation

Currently, there is no database implemented in the system. However, a database may be integrated in the future as part of system improvements to enhance functionality and scalability.

5.5 Graphical User Interface Implementation

Figure 30: Graphical User Interface Implementation

5.6 Summary

In this chapter shows all implementation details for HandSpeak including the database and graphical user interface implementation.

The output of this chapter is very necessary for the next chapter, whereby the testing and evaluation take place.

6.0 Chapter Six: System Testing and Evaluation

6.1 Introduction

This chapter provides a comprehensive overview of the testing, deployment, and usability aspects of the system, also ensures that the system is robust, reliable, and user-friendly.

6.2 System Testing

6.2.1 Model Evaluation Metrics

- **loss:** 0.0453
- **categorical accuracy:** 0.9867
- **validation loss:** 0.1356
- **validation categorical accuracy:** 0.9706
- **Confusion Matrix**

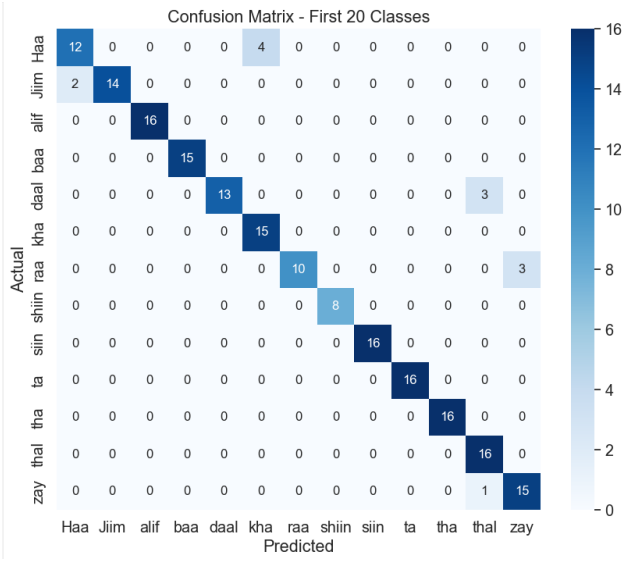

Figure 31: Confusion Matrix

6.2.2 Model Parameters

Table 15: Model Parameters

Argument	Type	Default	Description
layers	list	2 Bidirectional LSTM, 2 Dense	Defines the model architecture: 2 Bidirectional LSTM layers followed by Dense layers for classification.
units	int	64	Number of units (hidden neurons) in each LSTM layer.
activation_dense	str	relu and softmax	Activation functions for Dense layers: relu for the hidden layer and softmax for the output layer.
optimizer	str	adam	Optimization algorithm for training.
loss	str	categorical_crossentropy	Loss function for multi-class classification tasks.
metrics	list	['categorical_accuracy']	Metrics to evaluate the model's performance during training.
batch_size	int	32	Number of samples per batch for training and validation.
epochs	int	50	Maximum number of training iterations.
early_stopping	dict	See description	Early stopping configuration to prevent overfitting. Monitors val_loss, with patience of 5 epochs.
validation_split	float	0.2	Fraction of training data used for validation (20%).
input_shape	tuple	(frames, features)	Input shape for the LSTM model, determined by the preprocessed sequence length and features.
output_classes	int	len(words)	Number of classes in the output layer, based on the unique words/signs in the dataset.
loss	str	categorical_crossentropy	Loss function for multi-class classification tasks.
metrics	list	['categorical_accuracy']	Metrics to evaluate the model's performance during training.
batch_size	int	32	Number of samples per batch for training and validation.
epochs	int	50	Maximum number of training iterations.
early_stopping	dict	See description	Early stopping configuration to prevent overfitting. Monitors val_loss, with patience of 5 epochs.
validation_split	float	0.2	Fraction of training data used for validation (20%).
input_shape	tuple	(frames, features)	Input shape for the LSTM model, determined by the preprocessed sequence length and features.
output_classes	int	len(words)	Number of classes in the output layer, based on the unique words/signs in the dataset.

6.2.3 Model Optimization techniques

• Early Stopping Technique

The early stopping technique is used to reduce overfitting without compromising on the model's accuracy. It consists of stopping training the model when the validation loss begins to increase while the training loss continues to decrease.

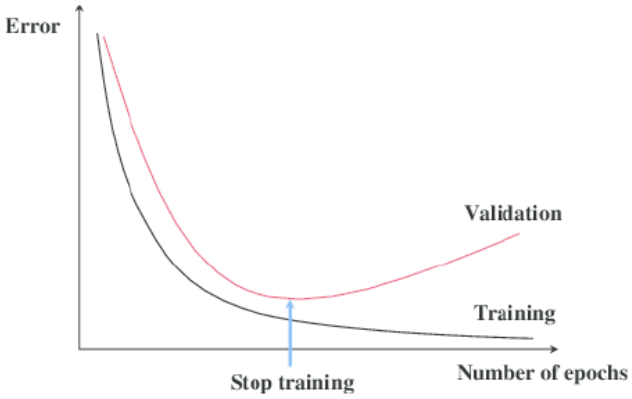

Figure 32: Early Stopping Technique

• Adam Optimization Algorithm

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.[14]

It offers adaptive learning rates, which dynamically adjust the learning rate for each parameter based on past gradients. This adaptivity provides faster convergence and improves performance across various data and model architectures.

6.2.4 Recognition Testing

To ensure the accuracy of predictions, we tested 10 samples from the dataset multiple times. The table below shows the probability of accuracy:

Table 16:Recognition Testing

Dataset Sample	Probability of Accuracy
ثقيل	98%
معجب	98%
ماء زمزم	97%
بعيد	97%
كرسي	95%
طفل	85%
خفيف	75%
مخدر	70%

6.3 Heuristics Evaluation:

Table 17: Summary of Violations by Heuristics

Heuristic Numbering Scheme	Frequency	Ratio (%)
H1	5	7.576
H2	5	7.676
H3	5	7.676
H4	10	15.152
H5	7	10.606
H6	6	9.09
H7	8	12.121
H8	5	7.576
H9	5	7.576
H10	10	15.152
Total	**66**	**100%**

Table 18: Summary of Violations by Severity Rating for Participant (1)

e	Frequency	Ratio (%)
0	15	0.23
1	12	0.181
2	10	0.151
3	9	0.14
4	20	0.30
Total	**66**	**100%**

- For more details, refer to Appendix A.

Table 19: Summary of Violations by Severity Rating for Participant (2)

Severity Rating	Frequency	Ratio (%)
0	9	0.14
1	12	0.181
2	15	0.23
3	10	0.151
4	20	0.30
Total	**66**	**100%**

- For more details, refer to Appendix A.

Table 20: Summary of Violations by Severity Rating for Participant (3)

Severity Rating	Frequency	Ratio (%)
0	12	0.181
1	10	0.151
2	15	0.23
3	9	0.14
4	20	0.30
Total	**66**	**100%**

6.4 Cooperative Evaluation:

After using the tool and answering the HandSpeak test, please indicate the extent to which you agree or disagree with each of the following statements regarding your experience with the system.

- **Cooperative Evaluation:**

Table 21: Participants Details

No.	Criteria	Participant 1	Participant 2	Participant 3
1.	**Gender**	Male	Female	Female
2.	**Age**	24	21	20
3.	**Educational Level**	BA	BA	BA
4.	**Programmer Taken**	Ammar	Dana	Duaa
5.	**Institution**	University of Jordan	University of Jordan	University of Jordan

6.4.1 Pre-Evaluation Procedures:

Participants were contacted through telephone conversations asking them the possibility to participate in the co-operative evaluation. A brief introduction to HandSpeak was given to the participants 10 minutes before they started the evaluation, and participants were asked to read that introductory document. The document also has a list of tasks which will be performed by the participants throughout the co-operative evaluation. Users were told that they need to think aloud when facing any problem in the system. They were also told that

each task they perform is monitored and timed.

6.4.2 Evaluation Procedures:

During the evaluation session, a moderator accompanied the users to do the co-operative evaluation. A comment shown in Appendix B was used by the moderator to write down the comments of each user for each task. Users were helped when they really face serious problems performing the tasks. The following tables show the comments pre-pared by the moderator for each participant.

Table 22 :Cooperative Evaluation for HandSpeak for Participant (1)

Task No.	Test	Time Taken (seconds)	Comments
A. Developer Activity			
1	Data Preprocessing	4,800	Participant followed the preprocessing steps effectively; ensured data quality for cooperative evaluation.
2	Keypoints Extraction	288,000	Keypoints extracted collaboratively; process completed accurately with attention to evaluation goals.
3	Model Training	900	Successfully trained model; minor tuning required for optimization.
4	Gesture Recognition Calibration	300	Calibration successful; required slight adjustments for accuracy.

5	Real-time Translation Setup	3600	Achieved seamless integration with minimal latency issues.
6	Code Deployment	1000	Deployment smooth; encountered minor compatibility issues resolved quickly.
B. User Activity			
1	Real-time Gesture Translation	150	High accuracy in most cases; occasional misinterpretations.
2	Feedback Submission	45	Feedback system simple and quick to use.
C. Deaf People Activity			
1	Gesture Recognition	180	Accurate for most gestures; struggled with complex signs occasionally.

Table 23 :Cooperative Evaluation for HandSpeak for Participant (2)

Task No.	Test	Time Taken (seconds)	Comments
A. Developer Activity			
1	Data Preprocessing	5,400	Participant followed the preprocessing steps effectively; ensured data quality for cooperative evaluation.
2	Keypoints Extraction	234,000	Keypoints extracted collaboratively; process completed accurately with attention to evaluation goals.
3	Model Training	900	Successfully trained model; minor tuning required for optimization.
4	Gesture Recognition Calibration	300	Calibration successful; required slight adjustments for accuracy.
5	Real-time Translation Setup	2800	Achieved seamless integration with minimal latency issues.
6	Code Deployment	2000	Deployment smooth; encountered minor compatibility issues resolved quickly.
B. User Activity			
1	Real-time Gesture Translation	180	High accuracy in most cases; occasional misinterpretations.

2	Feedback Submission	60	Feedback system simple and quick to use.
C. Deaf People Activity			
1	Gesture Recognition	210	Accurate for most gestures; struggled with complex signs occasionally.

Table 24 :Cooperative Evaluation for HandSpeak for Participant (3)

Task No.	Test	Time Taken (seconds)	Comments
A. Developer Activity			
1	Data Preprocessing	3,600	Participant followed the preprocessing steps effectively; ensured data quality for cooperative evaluation.
2	Keypoints Extraction	252,000	Keypoints extracted collaboratively; process completed accurately with attention to evaluation goals.
3	Model Training	900	Successfully trained model; minor tuning required for optimization.
4	Gesture Recognition Calibration	300	Calibration successful; required slight adjustments for accuracy.
5	Real-time Translation Setup	2300	Achieved seamless integration with minimal latency issues.

6	Code Deployment	1800	Deployment smooth; encountered minor compatibility issues resolved quickly.
B. User Activity			
1	Real-time Gesture Translation	160	High accuracy in most cases; occasional misinterpretations.
2	Feedback Submission	40	Feedback system simple and quick to use.
C. Deaf People Activity			
1	Gesture Recognition	240	Accurate for most gestures; struggled with complex signs occasionally.

- It is important to compare the time taken by each participant to complete each single task compared to the default time allocated by the moderator as shown in next table.

Table 25 :Task Completion Times in Minutes and Seconds

Task No.	Test	Default	Participant 1	Participant 2	Participant 3
A. Developer Activity					
1	Data Preprocessing	4300s	4800s	5400s	3600s
2	Keypoints Extraction	250000s	288000s	234000s	252000s
3	Model Training	900s	900s	960s	920s
4	Gesture Recognition Calibration	300s	300s	360s	400s
5	Real-time Translation Setup	3000s	3600s	2800s	2300s

6	Code Deployment	1200s	1000s	2000s	1800s
B. User Activity					
1	Real-time Gesture Translation	150s	150s	180s	160s
2	Feedback Submission	45s	45s	60s	40s
C. Deaf People Activity					
1	Gesture Recognition	180s	180s	210s	240s

6.4.3 Post-Evaluation Procedures:

After completing the co-operative evaluation, participants were given a post-test questionnaire to fill in, which is shown in Appendix C.

This questionnaire was important to capture their thoughts and feelings about HandSpeak while they were still fresh. The questionnaire was then followed by a short interview and discussion, which mainly focused on the initial modified design of HandSpeak. Table (26) shows the responses of the 3 participants to the post-test questionnaire.

Table 26 :Participants' Responses to the Post-Test Questionnaire

No.	Statement	Participant 1	Participant 2	Participant 3	Average
1	Is the system stable?	5	4	4	**4.3**
2	Is the system ease of use?	5	5	5	**3**
3	Are the functionality of the system achieve user's needs?	5	4	5	**4.6**
Average		**5**	**4.3**	**4.6**	

6.5 System Installation

- **6.3.1 Jupyter notebook**

  https://jupyter.org/

- **Operating System (Microsoft Windows 8)**

  http://windows.microsoft.com/en-us/windows/downloads

- **Microsoft Office 365**

  http://www.microsoftstore.com/store/msusa/en_US/cat/All-Office/categoryID.69403900

- **Visual Studio**

  https://code.visualstudio.com/

- **JDK (Java Development Kit)**

  https://www.oracle.com/java/technologies/javase-downloads.html

- **Python**

  https://www.python.org/downloads/

- **Libraries and Packages**

1. TensorFlow
2. NumPy
3. Matplotlib
4. OpenCV

6.6 User Manual

Predicted sign is shown here

Words List: A dictionary of sign words

Press and hold the 'Translate' button to

6.7 Summary

Chapter Six evaluated the system's performance, detailed the installation process, and provided a user manual to ensure the system is reliable, easy to deploy, and user-friendly.

7.0Chapter Seven: Project Conclusion and Future Work

7.1 Introduction:

This section summarizes the key outcomes of the project, reflecting on its objectives, results, and contributions. Additionally, it highlights potential areas for improvement, further development, and new directions for future research and application. By acknowledging current limitations and identifying opportunities for innovation, this section aims to provide a clear pathway for building upon the project's foundation.

7.2 Overall Weaknesses:

HandSpeak faces a few challenges that may limit its effectiveness. First, it only supports Arabic sign language, which reduces its usefulness for people who use other languages. Second, the

system's performance can be affected by environmental factors like poor lighting, busy backgrounds, or camera angles, which might make it less reliable in real-world situations. Third, collecting a large and diverse dataset for both English and Arabic sign languages is difficult, and without enough high-quality data, the system's accuracy could suffer. Finally, ensuring the model provides accurate translations in real-time may be hard, especially on devices with limited processing power, leading to delays and impacting the user experience.

7.3 Overall Strengths:

HandSpeak effectively addresses the communication challenges faced by the Deaf community, enhancing accessibility, and promoting inclusivity. It offers real-time translation of sign language into text, ensuring smooth and quick communication. By integrating advanced technologies like computer vision and machine learning, the model provides accurate translations and operates offline, making it accessible even in areas without internet. Additionally, it benefits not only the Deaf community but also their families, educators, and service providers, fostering social inclusion and improving communication efficiency.

7.4 Future Work:

- **Advanced Sentence Generation**: We aim to integrate state-of-the-art NLP models to translate gestures into full, grammatically correct sentences, enhancing communication fluidity.

- **Mobile Deployment**: deploy the HandSpeak model directly within the Flutter application. This integration will enhance accessibility and ensure seamless, on-the-go functionality for users, removing the need for external systems.
- **Educational Features**: To make learning sign language accessible and fun, the app will include lessons, quizzes, and tutorials. These features will empower users to gain proficiency at their own pace.
- **3D Avatars for Sign Language**: Using tools like Blender or collaborating with VSL labs, we plan to create dynamic 3D avatars to visually demonstrate gestures. This will allow users to translate words or sentences into sign language with an interactive and engaging visual representation.
- **3D Avatars for Sign Language Visualization**: Using tools like Blender or collaborating with VSL labs, we plan to create dynamic 3D avatars to visually demonstrate gestures. This will allow users to translate words or sentences into sign language with an interactive and engaging visual representation.

7.5 Summary:

This chapter highlights HandSpeak's progress in translating sign language into text and speech, making communication easier for the Deaf community. Looking ahead, the team plans to improve by adding features like gesture-to-speech, 3D avatars, and tools to teach sign language. Users will also be able to add new gestures, making the app more flexible and inclusive.

References

- Gary B. Shelly, Thomas J. Cashman, Harry J. Rosenblatt. (2008). Systems Analysis and Design, 7th Edition.

- Jeffrey A. Hoffer, J. G. (2011). Modern System Analysis and Design. Pearson Higher Education.

- Zukhriddin. (2023, Jan 26). RGB vs BGR | Relationships between color depth, pixel, and bytes. Retrieved from Medium: https://zrruziev.medium.com/rgb-vs-bgr-relationships-between-color-depth-pixel-and-bytes-7821fa9c6320

- Lucid Chart: https://www.lucidchart.com/

- Atwan, J., Wedyan, M., Abbas, A., & Gazzawe, F. (2022, May). Development Web-based Arabic Assessments for Deaf and Hard-of- Hearing Students. Retrieved from

ResearchGate:

https://www.researchgate.net/publication/362125870_Development_Web-based_Arabic_Assessments_for_Deaf_and_Hard-of-_Hearing_Students

- *KArSL Dataset*. (n.d.). Retrieved from https://hamzah-luqman.github.io/KArSL/

Appendix A

Heuristic Evaluation – A System Checklist

Disclaimer: This list is a simplified one of the original list which was developed by Xerox Corporation (© Usability Analysis & Design, Xerox Corporation, 1995) and was downloaded from ftp://cs.uregina.ca/pub/class/305/lab2/example-he.html. It has been simplified to suite the purpose it is used for, which is to evaluate the HandSpeak in order to identify current problems as experienced by the users, which is part of our graduation project that is submitted to King Abdullah II School for Information Technology, The University of Jordan. The number of questions was reduced; however, the individual questions were left intact.

Please fill in the evaluation form below, which is a form of checklist, by writing "X" in the appropriate place which mostly describes the best answer to the corresponding criterion. This form is to be filled after you have investigated the system interface i.e. have looked at, and examined the interface. The answer to each criterion is either:

- "0" which means “I don’t agree that this is a usability problem at all”.
- "1" which means “Cosmetic problem only: need not be fixed unless extra time is available on project”.
- "2" which means “Minor usability problem: fixing this should be given low priority”.
- "3" which means” Major usability problem: important to fix, so should be given high priority”.
- "4" which means” Usability catastrophe: imperative to fix this before product can be released”.

Thank you for your willingness to evaluate this system. Your time and effort are highly appreciated.

H1. Visibility of System Status

The system should always keep user informed about what is going on, through appropriate feedback within reasonable time.

Number	Review Checklist	0 1 2 3 4	Comments
1.1	Does every display begin with a title or header that describes screen contents?	( ) ( ) ( ) ( ) ( )	
1.2	Do menu instructions, prompts, and error messages appear in the same place(s) on each menu?	( ) ( ) ( ) ( ) ( )	
1.3	Is there some form of system feedback for every operator action?	( ) ( ) ( ) ( ) ( )	
1.4	Are responses times appropriate to the users cognitive processing?	( ) ( ) ( ) ( ) ( )	

1.5	Is there visual feedback in menus or dialog boxes about which choices are selectable?	( ) ( ) ( ) ( ) ( )	

H2. Match between System and the Real World

The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Number	Review Checklist	0 1 2 3 4	Comments
2.1	Are icons concrete and familiar?	( ) ( ) ( ) ( ) ( )	
2.2	Are menu choices ordered in the most logical way, given the user, the item names, and the task variables?	( ) ( ) ( ) ( ) ( )	
2.3	Do related and interdependent fields appear on the same screen?	( ) ( ) ( ) ( ) ( )	

2.4	When prompts imply a necessary action, are the words in the message consistent with that action?	( ) ( ) ( ) ( ) ( )	
2.5	On data entry screens, are tasks described in terminology familiar to users?	( ) ( ) ( ) ( ) ( )	

H3. User Control and Freedom

Users should be free to select and sequence tasks (when appropriate), rather than having the system does this for them. Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Users should make their own decisions (with clear information) regarding the costs of exiting current work. The system should support undo and redo.

Number	Review Checklist	0 1 2 3 4	Comments
3.1	When a user's task is complete, does the system wait for a signal from the user before processing?	( ) ( ) ( ) ( ) ( )	

3.2	Are users prompted to confirm commands that have drastic, destructive consequences?	( ) ( ) ( ) ( ) ( )	
3.3	Are character edits allowed in data entry fields?	( ) ( ) ( ) ( ) ( )	
3.4	If menu lists are long (more than seven items), can users select an item either by moving the cursor or by typing a mnemonic code?	( ) ( ) ( ) ( ) ( )	
3.5	If the system uses a pointing device, do users have the option of either clicking on menu items or using a keyboard shortcut?	( ) ( ) ( ) ( ) ( )	

H4. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Number	Review Checklist	0 1 2 3 4	Comments
4.1	Has a heavy use of all uppercase letters on a screen been avoided?	( ) ( ) ( ) ( ) ( )	
4.2	Are icons labeled?	( ) ( ) ( ) ( ) ( )	
4.3	Are there no more than twelve to twenty icon types?	( ) ( ) ( ) ( ) ( )	

4.4	Does each window have a title?	( ) ( ) ( ) ( ) ( )	
4.5	Is vertical and horizontal scrolling possible in each window?	( ) ( ) ( ) ( ) ( )	
4.6	Are menu choice lists presented vertically?	( ) ( ) ( ) ( ) ( )	
4.7	Are menu titles either centered or left-justified?	( ) ( ) ( ) ( ) ( )	
4.8	Are menu items left-justified, with the item number or mnemonic preceding the name?	( ) ( ) ( ) ( ) ( )	
4.9	Do embedded field-level prompts appear to the right of the field label?	( ) ( ) ( ) ( ) ( )	
4.10	Are attention-getting techniques used with care?	( ) ( ) ( ) ( ) ( )	

H5. Help Users Recognize, Diagnose, and Recover from Errors

Error messages should be expressed in plain language (NO CODES).

Number	Review Checklist	0 1 2 3 4	Comments
5.1	Is sound used to signal an error?	( ) ( ) ( ) ( ) ( )	
5.2	Are error messages worded so that the system, not the user, takes the blame?	( ) ( ) ( ) ( ) ( )	

5.3	Do error messages suggest the cause of the problem?	( ) ( ) ( ) ( ) ( )	
5.4	Do error messages indicate what action the user needs to take to correct the error?	( ) ( ) ( ) ( ) ( )	
5.5	If the system supports both novice and expert users, are multiple levels of error-message detail available?	( ) ( ) ( ) ( ) ( )	
5.6	If an error is detected in a data entry field, does the system place the cursor in that field or highlight the error?	( ) ( ) ( ) ( ) ( )	
5.7	Do error messages inform the user of the error's severity?	( ) ( ) ( ) ( ) ( )	

H6. Error Prevention

Even better than good error messages are a careful design which prevents a problem from occurring in the first place.

Number	Review Checklist	0 1 2 3 4	Comments
6.1	Are menu choices logical, distinctive, and mutually exclusive?	( ) ( ) ( ) ( ) ( )	

6.2	Are data inputs case-blind whenever possible?	( ) ( ) ( ) ( ) ( )	
6.3	Does the system prevent users from making errors whenever possible?	( ) ( ) ( ) ( ) ( )	
6.4	Does the system warn users if they are about to make a potentially serious error?	( ) ( ) ( ) ( ) ( )	
6.5	Do data entry screens and dialog boxes indicate the number of character spaces available in a field?	( ) ( ) ( ) ( ) ( )	
6.6	Do fields in data entry screens and dialog boxes contain default values when appropriate?	( ) ( ) ( ) ( ) ( )	

H7. Recognition Rather Than Recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Number	Review Checklist	0 1 2 3 4	Comments

7.1	For question and answer interfaces, are visual cues and white space used to distinguish questions, prompts, instructions, and user input?	( ) ( ) ( ) ( ) ( )	
7.2	Are inactive menu items grayed out or omitted?	( ) ( ) ( ) ( ) ( )	
7.3	Do data entry screens and dialog boxes indicate when fields are optional?	( ) ( ) ( ) ( ) ( )	
7.4	Are prompts, cues, and messages placed where the eye is likely to be looking on the screen?	( ) ( ) ( ) ( ) ( )	
7.5	Are field labels close to fields, but separated by at least one space?	( ) ( ) ( ) ( ) ( )	
7.6	Have items been grouped into logical zones, and have headings been used to distinguish between zones?	( ) ( ) ( ) ( ) ( )	
7.7	Are borders used to identify meaningful groups?	( ) ( ) ( ) ( ) ( )	
7.8	Is color coding consistent throughout the system?	( ) ( ) ( ) ( ) ( )	

H8. Flexibility and Minimalist Design

Accelerators-unseen by the novice user-may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. Provide alternative means of access and operation for users who differ from the "average" user (e.g., physical or cognitive ability, culture, language, etc.)

Number	Review Checklist	0 1 2 3 4	Comments
8.1	If menu lists are short (seven items or fewer), can users select an item by moving the cursor?	( ) ( ) ( ) ( ) ( )	
8.2	If the system uses a pointing device, do users have the option of either clicking on fields or using a keyboard shortcut?	( ) ( ) ( ) ( ) ( )	
8.3	On data entry screens, do users have the option of either clicking directly on a field or using a keyboard shortcut?	( ) ( ) ( ) ( ) ( )	
8.4	On menus, do users have the option of either clicking directly on a menu item or using a keyboard shortcut?	( ) ( ) ( ) ( ) ( )	
8.5	In dialog boxes, do users have the option of either clicking directly on a dialog box option or using a keyboard shortcut?	( ) ( ) ( ) ( ) ( )	

H9. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Number	Review Checklist	0 1 2 3 4	Comments
9.1	Are all icons in a set visually and conceptually distinct?	( ) ( ) ( ) ( ) ( )	
9.2	Does each icon stand out from its background?	( ) ( ) ( ) ( ) ( )	
9.3	Does each data entry screen have a short, simple, clear, distinctive title?	( ) ( ) ( ) ( ) ( )	
9.4	Are field labels brief, familiar, and descriptive?	( ) ( ) ( ) ( ) ( )	
9.5	Are there pop-up or pull-down menus within data entry fields that have many, but well-defined, entry options?	( ) ( ) ( ) ( ) ( )	

H10. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Number	Review Checklist	0 1 2 3 4	Comments
10.1	Are on-line instructions visually distinct?	( ) ( ) ( ) ( ) ( )	
10.2	If menu choices are ambiguous, does the system provide additional explanatory information when an item is selected?	( ) ( ) ( ) ( ) ( )	
10.3	Is the help function visible; for example, a key labeled help or a special menu?	( ) ( ) ( ) ( ) ( )	
10.4	Navigation: Is information easy to find?	( ) ( ) ( ) ( ) ( )	
10.5	Presentation: Is the visual layout well designed?	( ) ( ) ( ) ( ) ( )	
10.6	Conversation: Is the information accurate, complete, and understandable?	( ) ( ) ( ) ( ) ( )	
10.7	Is the information relevant?	( ) ( ) ( ) ( ) ( )	
10.8	Can users easily switch between help and their work?	( ) ( ) ( ) ( ) ( )	
10.9	Is it easy to access and return from the help system?	( ) ( ) ( ) ( ) ( )	
10.10	Can users resume work where they left off after accessing help?	( ) ( ) ( ) ( ) ( )	

Appendix B

Cooperative for HandSpeak

Task No.	Test	Time Taken (seconds)	Comments
A. Developer Activity			
1	Data Preprocessing		
2	Keypoints Extraction		
3	Model Training		
4	Gesture Recognition Calibration		
5	Real-time Translation Setup		
6	Code Deployment		
B. User Activity			
1	Real-time Gesture Translation		
2	Feedback Submission		
C. Deaf People Activity			
1	Gesture Recognition		

Appendix C

HandSpeak Usability Test (Post-test Questionnaire)

Gender: M / F **Age: _____**

Educational Level: _____

Programmer Taken: _____

Institution: _____

After using the system and answering HandSpeak test, please indicate the extent to which you agree or disagree with each of the following statements regarding to your experience with the system.

No.	Statements	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1	2	3	4	5
1.	Is the app stable?					
2.	Is the app ease of use?					
3.	Are the functionality of the app achieve user's needs?					