KINGDOM OF SAUDI ARABIA
Ministry of Education
Taibah University
College of Computer Science and Engineering
(Female Section)

المملكة العربية السعودية
وزارة التعليم
جامعة طيبة
كلية علوم و هندسة الحاسب الآلي
(قسم الطالبات)

Esharati-إشارتي: A Machine learning-based System for the Saudi Sign Language learning and Recognition

Graduation Project (1)

by

Ahad Saad Altalhi 4150538

Fatmah Talal Merghelani 4152189

Maria Ashraf Afifi 4150523

Moroj Hasan Abualkahir 3979831

Rafa Abdulmhsen Alhejaili 4150699

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science (Computer Science)

Supervised by

Dr. Samah Aloufi

2nd Semester - Academic Year 1445 (2023-2024)

Abstract

Sign languages are used by deaf and hard-of-hearing people as a means of communication and self-expression. Similar to any spoken language, sign language has its own rules and also vary across different countries. In order to help people learning sign language, especially in Saudi Arabia, we aimed to develop an application for learning, recognizing, and translating the Saudi Sign Language using machine learning approach. Our application has two main functionalities: teaching Saudi Sign Language and translating signs into text. The application aims for promulgates sign language and involved its users more in our community.

Keywords Saudi Sign Language; Sign Language Recognition; Machine learning; Sign Translation; Sign to text

Acknowledgement

First and foremost, praises and thanks to Allah, the Almighty, for his numerous blessings that have enabled us to successfully finish the project. We appreciate Dr. Samah Al-Aloufi, our supervisor, for her valuable advice and insights during the course of this project. She has shown us how to complete our tasks accurately and present the work of the project in the most understandable way possible. Being able to work and learn under her direction was a huge honor. We also thank our parents and siblings for their support, love, and prayers as well as for sharing their insights and ideas to help us work on our project better. Lastly, our thanks go to all the people who have supported us in completing the project directly or indirectly.

Table of Content

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ArSL	Arabic Sign Language
CNN	Convolutional Neural Network
DL	Deep Learning
DT	Decision Tree
EHD	Edge Histogram Descriptor
FN	False Negative
FP	False Positive
HOG	Histograms of Oriented Gradients
ML	Machine Learning
RFM	Random Forest Machine
SL	Sign Language
SLR	Sign Language Recognition
SSL	Saudi Sign Language
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
UML	Unified Modeling Language

Chapter 1: Introduction

1.1 Introduction

According to the World Federation of the Deaf, approximately 70 million people are deaf [1]. These individuals rely on sign language as their primary means of communication among themselves and with other people. While many deaf people are proficient in sign language, many ordinary people do not understand sign language. This communication barrier leads to isolation of deaf people. To address this issue, it is important to develop a system that teaches sign language to the general and translates it into written text.

Many studies have focused on developing a sign language recognition system for English, but there is a scarcity of research on Arabic sign language. Similar to spoken Arabic, sign language also has various dialects across different countries. Therefore, our aim is to develop a system specifically for learning, recognizing, and translating Saudi sign language. The proposed system will utilize machine learning-based methods to recognize and translate the Saudi sign language. In the learning phase, users will be learning general vocabulary that are in the Saudi sign language, categorized according to pre-defined categories. After completing each category, users will be able to assess their proficiency level which will provide them with a sense of accomplishment. In addition to teaching the sign language, the system includes the translation component which will enable real-time translation of Saudi sign language into written text, making it easier for non-signers to understand and communicate with deaf people. This component will rely on a machine learning model to accurately capture and interpret the signs. By developing this system, we can empower ordinary people to learn and understand Saudi sign language which will lead to better communication and integration of deaf people into our society.

1.2 Problem Motivation

The lack of knowledge and usage of sign language among ordinary people creates a barrier to effectively communicate with deaf people. This leads to isolation of deaf people. In light of this, we are motivated to address this challenge by developing a system that teaches sign language and provides real-time translation from sign to text. By bridging the communication gap, we aim to enable deaf to be actively participate and engage with their society.

1.3 Project Objectives

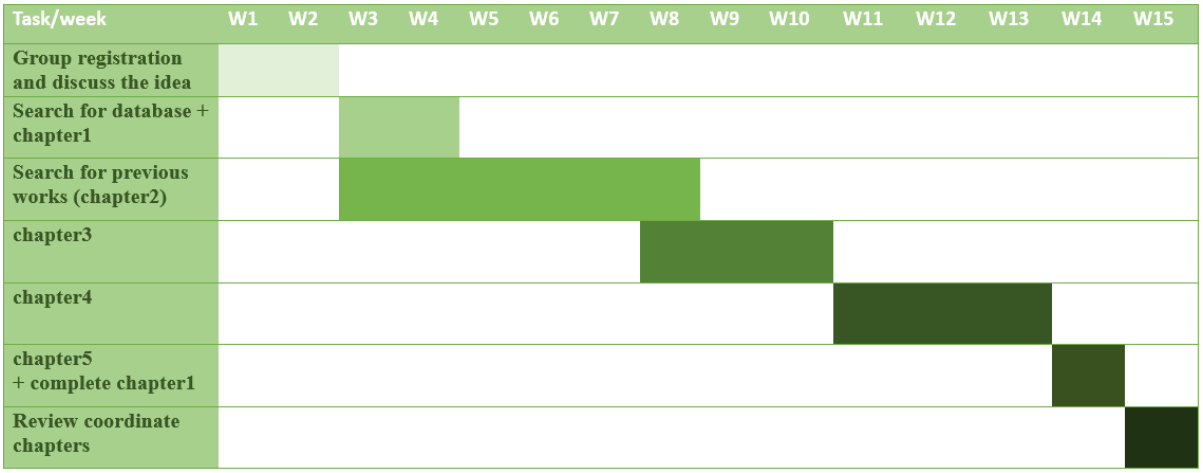
This project aims at developing a machine learning-based system that is able to recognize Arabic Sign Language in the Saudi dialect (SSL) as well as teaching users the sign language in an interactive way. The following list outlines the main objectives of the project:

- To study related work and existing applications that have common context and purpose to our project.
- To identify the proper machine learning algorithms and architectures for recognizing and detecting images.
- To identify useful methods for implementing the algorithms.
- To identify the database that will be utilized in the project for the sign language learning and recognizing purposes.
- To develop a machine learning-based model to detect the user movement and recognize the sign language.
- To evaluate the performance of the developed model using evaluation metrics.

1.4 Project Scope

This project aims to develop an application to help the deaf community to participate more in our society by teaching Arabic sign language specifically in Saudi dialect. The users have the option to learn the Arabic sign language or test their learning level.

1.5 Project Timeline



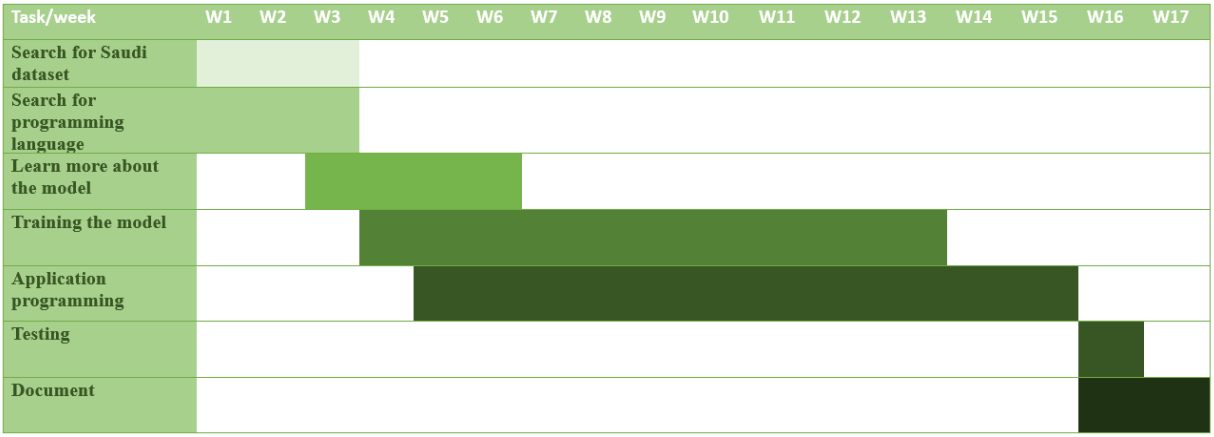


Figure 1.1 Project Timeline

1.6 Document Organization

This project consists of five chapters. In the first chapter we introduced the project objectives, the motivation of the project, the scope of the work, and lastly, we presented the timeline of our work. In Chapter 2, we provide background information about the Sign Languages and the techniques that are used for sign language recognition. Also, we reviewed and discuss related works to our work. In Chapter 3, we analyze the system

requirements including both functional and non-functional requirements. Also, we discussed the system development methodology. In Chapter 4, we present the system architectural design and illustrate both the structural static model and the dynamic models. Also, we present our system's data modeling, and an initial prototype of the system user interface. In Chapter 5, We discuss the implementation phase of the proposed application, describing in detail the utilized tools and programming languages, the model training and testing steps as well as the system development. Lastly, in Chapter 6, we summarize the document and discuss the limitations and the direction of future works.

2 Chapter 2: Literature Review

2.1 Introduction

In this chapter, we will provide a background of Sign Languages (SL) and the differences between them. In addition, we will cover the techniques utilized to recognize the sign language (SLR) and the criteria to measure the effectiveness of these different methods. After that, we will discuss available datasets for Arabic sign language. Next, we survey existing related work to our project and differentiate between them and our proposed.

2.2 Background

Sign language is a visual language used by hearing-impaired people in order to communicate with others through hand gestures, body language, and facial expressions [2- 4]. Sign languages are used by deaf people in a similar way to natural languages, but with their own grammar and syntax. Sign language is revered among the deaf community as a distinctive and vital means of communication. There are many different sign languages in use around the world, and each one is specific to its country or region. Similarly, in Arabic countries, there are variations of Arabic sign languages such as Saudi, Jordanian, Egyptian, etc. While there are similarities among them, various grammatical and linguistic rules in Arabic sign languages may differ depending on the local culture and community [5].

To communicate with others, deaf people use words and sentences, unless the word does not have a predefined sign, in which case they use finger spelling [2]. Although deaf using sign language to communicate with others, few normal people are using or even understanding the sign languages which result in isolating deaf people. Hence, to engage the deaf community within their society, we need a system to recognize the sign language and translate it into text or audio. This type of system is known as Sign Language Recognition (SLR) systems.

Sign language recognition (SLR) is a process of detecting and translating sign language gestures and movements from image or video into text or other forms of communication [6]. Generally, the SLR systems have three levels of recognition: recognizing alphabet, recognizing

isolated word, recognizing continuous words (sentence level). In order to recognize the sign languages, there are two approaches which are commonly followed: vision-based systems or device-based systems. The first line of systems utilized sensors and hardware such as gloves, Microsoft Kinect, and motion sensors for gesture tracking. On other hand, vision-based systems rely on artificial intelligence and computer techniques for analyzing images and videos to track hand gesture used in the sign language. In our work, we are focusing on the vision-based approach. Various machine learning and deep learning algorithms are employed to recognize sign languages.

Machine Learning (ML)

Machine learning (ML) is a branch of artificial intelligence (AI) that focuses on creating computer systems that can automatically learn from data, without being explicitly programmed for each task [7]. The field of machine learning is divided into several subfields, including supervised and unsupervised learning, reinforcement learning, and deep learning. Algorithms used in machine learning are trained to identify hidden patterns and relationships in data, predict the output, and improve the performance from experiences on their own [8]. Different algorithms can be used in machine learning for different tasks, such as support vector algorithm (SVM), and the Decision Trees (DT) algorithm [7-11]. ML uses data as an input, along with the output, which is fed into the learning algorithm during the learning phase, and it works out a program for itself as we see in Figure 2.1.

Figure 2.1 Machine Learning Process [8].

i. Support Vector Machines (SVM)

A support vector machine (SVM) is a supervised learning algorithm that can also be used for classification and regression problems. The goal of SVM is to create an optimal hyperplane or decision boundary that can segregate datasets into different classes. The data points that help to define the hyperplane are known as support vectors or support vector machine algorithms [11, 12].

ii. Decision Tree (DTs)

A decision tree is a supervised learning algorithm for solving the classification and regression problem. It can work with both categorical variables and continuous variables. It shows a tree-like structure that includes nodes and branches as shown in figure 2.2. It provides models that are easy to understand. It is used in decision support to represent decisions and their possible results. This algorithmic model makes use of conditional control statements. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves [11, 13].

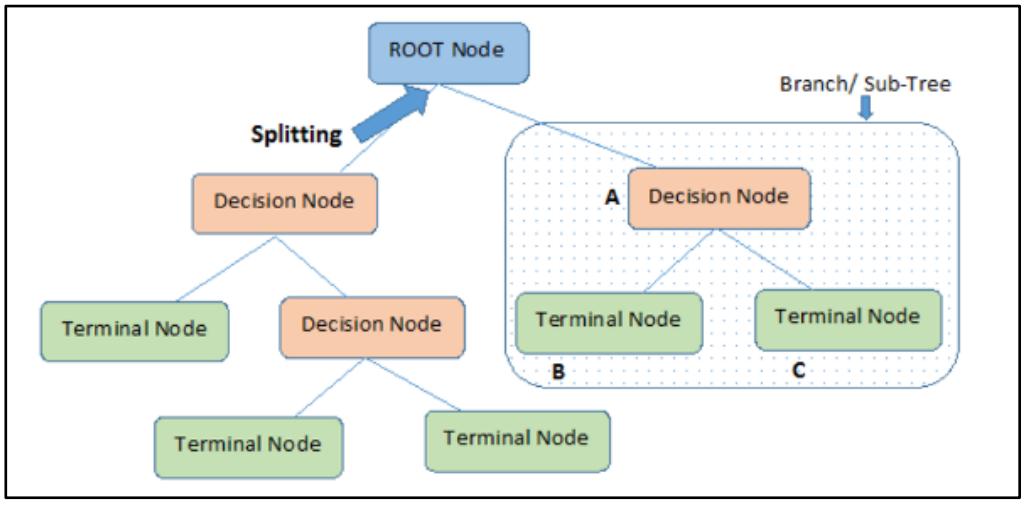


Figure 2.2 Decision Tree algorithm [13].

iii. Random Forest (RFM)

An effective tree learning method in machine learning. During the training stage, it generates many Decision Trees. To measure a random subset of characteristics in each partition, a random subset of the data set is used to build each tree. Because each tree is more variable as a result of the randomization, there is less chance of overfitting and overall prediction performance is enhanced. When making predictions, the algorithm averages or votes output of each tree. This cooperative decision-making process yields an example of steady and accurate findings, aided by the insights of several trees Since random forests can handle complex data, minimize overfitting, and produce accurate forecasts in a variety of settings, they are frequently employed for regression and classification tasks [14].

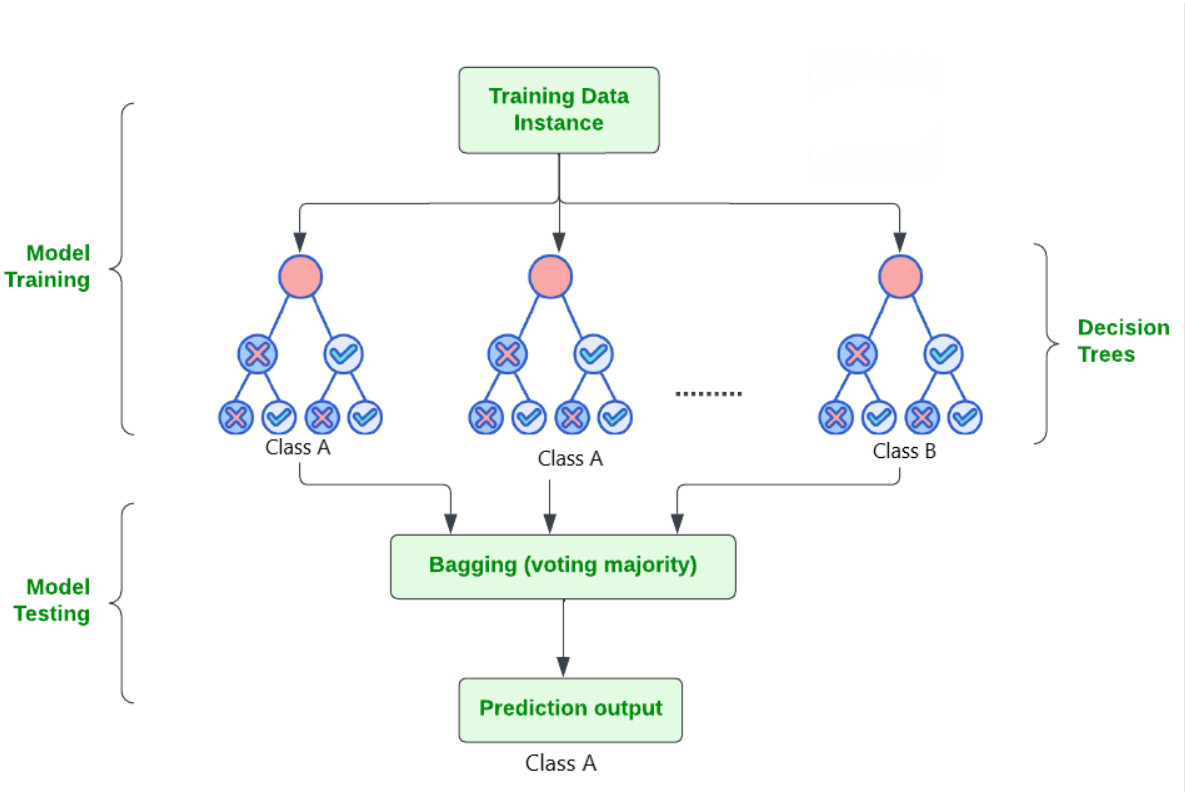


Figure 3 Random Forest algorithm [14].

- **Deep Learning (DL)**

Deep learning (DL) is a type of machine learning (ML) that focuses on deep neural networks, which have several layers. Artificial neural networks are composed of interconnected layers of artificial neurons. They are intended to learn hierarchical features automatically from raw data. Images, music, text, and video are examples of unstructured data that deep learning is particularly adept at handling. Some of the tasks that have excelled at are natural language processing, computer vision, and image and sound recognition. Deep learning models can have millions or even billions of parameters, making them extremely complicated. This complexity makes them suited for tasks with high-dimensional input since it enables them to catch subtle patterns and representations in large-scale data. However, deep learning models are computationally demanding and often benefit significantly from GPUs or specialized hardware for efficient training and inference [15].

i. Artificial Neural Networks (ANN)

Inspired by biological neural networks, ANN is a computational system that contains high numbers of interconnected nodes which work in a distributed fashion to learn from the input in order to optimize its final output. Figure 2.3 shows the structure of ANN, the input usually upload in the form of a multidimensional vector, then it distributed to the hidden layers which will then make the decisions from the previous layer and weigh up how a stochastic change within itself detriments or improves the final output, and this is referred to as the process of learning [16]

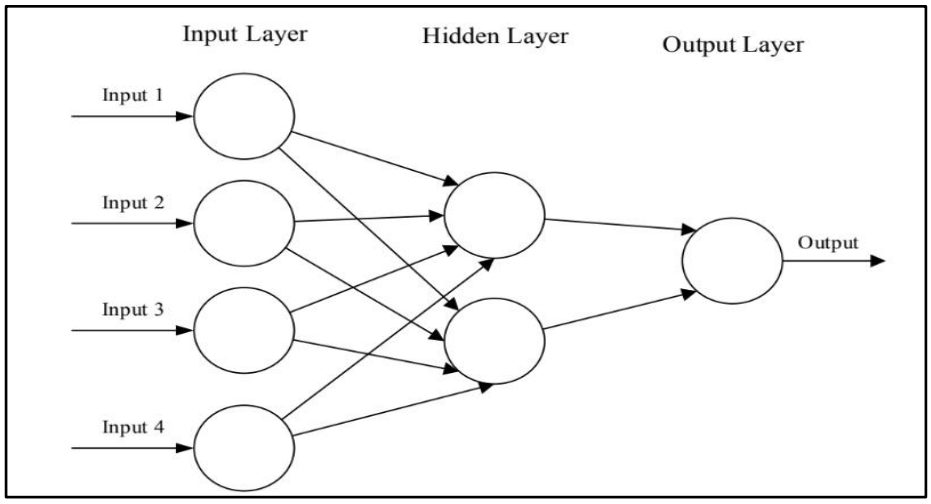


Figure 2.4 Artificial Neural Networks [15].

ii. Convolutional Neural Network (CNN)

CNN is a network in the deep learning field that has shown very impressive achievements especially in computer vision and natural language processing [17]. CNN primarily focuses on the basis that they have images as their input. It contains three types of layers: Convolutional layers, Pooling layers, and fully connected layers. A simplified CNN architecture for MNIST classification is illustrated in Figure 2.4.

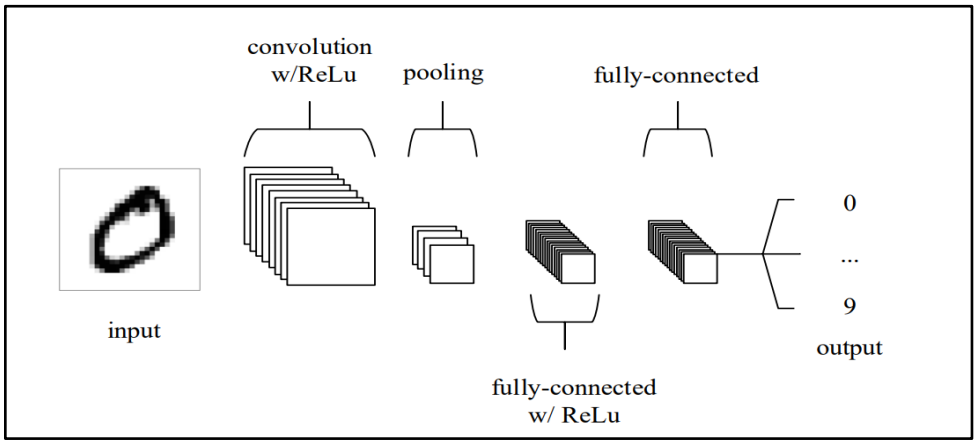


Figure 2.5 CNN architecture for MNIST classification [15].

The main and only difference between ANN and CNN is that CNN is primarily used for recognizing the pattern within images, while ANN tends to struggle with the computational complexity required to compute image data [15].

Datasets

In the stage of searching for a dataset in research or even some specialized association, we found many datasets for Arabic sign language (ArSL), some of which are specifically in the Saudi dialect. In the following, we describe the publicly available datasets for Arabic sign languages:

- The Saudi association for hearing impairment dataset contains popular words in a Saudi dialect. The dataset includes 31 categories like "religion", "home" and "directions", etc. Each category has 20 words; in total the dataset contains 620 words [18].

- In the Al-Anoud Center for the Development of the Disabled dataset, there was a different way of word classification. There are 5 catogravies: letters (الحروف), pronouns (الضمائر), verbs (الأفعال), nouns (الأسماء), and adjectives (الصفات). All Signs of Alanood Center are represented in Saudi dialect [19].

- Ala Sidig search introduces KArSL dataset for Arabic sign language (ArSL) that consists of 500 signs of numbers, letters, and words related to different domains such as “health”, “religion”, and “common verbs” [20].

- The Arabic Dictionary of Gestures for The Deaf was prepared by the League of Arab States. There are two ways to display the words: the first according to their categories, such as “Family” and "Food”, or according to the order of the letters of the alphabet. The method of representing signs is presented as a gif [21].

Evaluation Metrics

Evaluation metrics are used to measure the performance of a system. For SLR systems which is considered a classification problem, popular evaluation metrics include accuracy, precision, Recall (Sensitivity), F1-score [22, 23].

- **Accuracy**

Accuracy is a measure of all correctness of a classification model. This is determined by dividing the proportion of valid predictions by the total number of instances evaluated. Accuracy is calculated as shown in equation in 2.1 [24].

$$Accuracy = \frac{TrueNegatives + TruePositives}{TruePositives + FalsePositives + TrueNegatives + FalseNrgatives} \quad (2.1)$$

Where True Positives (TP) is the case where the model predicted Yes and the real output was also Yes", True Negatives (TN) = "It is the case where the model predicted No and the real output was also No", False Positives (FP) ="It is the case where the model predicted Yes but it was actually No", False Negatives (FN) ="It is the case where the model predicted No but it was actually Yes"

- **Precision**

How many of the model's positive predictions were actually correct is a measure of the model's performance known as Precision. It is calculated as the number of true positive predictions divided by the total number of true or false positive predictions [24]. Precision is defined in equation in 2.2 [21].

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (2.2)$$

- **Recall**

Recall calculates the percentage of actual positives a model correctly identifies (True Positive). The number of true positives (TP) divided by the number of positive outcomes actually predicted (TP) by the model and the number of positive outcomes mistakenly predicted by the model as negative outcomes (FN). Recall is defined in equation 2.3[26].

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (2.3)$$

- **F-score (F-measure)**

A machine learning (ML) model's performance is assessed using a metric called the F-score (F1 score or F-measure). It produces a single score that includes recall and precision. It is frequently used when precision and recall must be balanced, and it is especially useful when the positive class is rare. The F-score ranges from 0 to 1, with higher values indicating better performance. F-score is defined as can be seen in equation 2.4 [28].

$$F1\ Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (2.4)$$

2.3 Related Work

2.3.1 Review of Relevant Work

In the following section, we present a survey of some previously used approaches in Arabic sign language recognition (ArSL) and translation to text. Alzohairi et al. [2], developed a recognizer for Arabic Sign Language (ArSL) by capturing gestures representing the ArSL alphabet from images. The approach involved utilizing the extracted visual descriptors conveyed to a One-Versus-All Support Vector Machine (SVM), Histograms of Oriented Gradients (HOG), and Edge Histogram Descriptor (EHD). Notably, the highest performance was achieved for 27 letters using HOG, while the remaining 3 letters were recognized using EHD. The system demonstrated an accuracy of 63.5% in recognizing Arabic Alphabet gestures.

Using deep learning approach, Al-Obodi et al. [3] proposed a sign language recognition system designed for Saudi deaf people. Using the Convolutional Neural Networks (CCN) model to construct a dataset of 40 Saudi signs, each with 700 images with various backgrounds and conditions. The proposed model achieved an accuracy of 97.69% for training data and 99.47% for testing data. In the same context, the study by Hayani et al. [29], developed an Arab sign language recognition system (ASLR) that was proposed to recognize 28 letters and numbers ranging from 0 to 10. The system was built using Convolutional Neural Networks (CCN) and RGB images. The system achieved an accuracy of 90.02% when trained using 80% of the images from the database. Al-hammadi et al. [30], used 3DCNN for hand gesture recognition. In the first approach, a single 3DCNN instance was trained to extract the hand gesture features from the entire video. In the second approach, three instances of the 3DCNN structure were trained to extract the hand gesture features from the beginning, middle, and end of the video sample. These region-based features were then fused before being fed to the classifier, they use 3D Dataset, KSU-SSL dataset, ArSL dataset, and Purdue RVL-SLLL American sign language dataset, the accuracy achieved is 98.12%, 100%, and 76.67% on the three datasets, respectively for the signer-dependent mode. For the signer-independent mode, it achieved 84.38%, 34.9%, and 70% on the three datasets.

Alsulaiman et al. [31], introduced a dataset for sign language under the name King Saud University Saudi Sign Language (KSU-SSL) using the CNN. This dataset was recorded by three varieties of recording sources: fast RGB camera, an infrared (IR) camera, and a mobile camera. Additionally, the dataset was built using the Center-Connected Graph Convolutional Network (CGCN) architecture for sign language recognition. The said architecture is constructed of a small number of separable 3DGCN layers. The proposed architecture achieved an average accuracy of 97%. Moreover, Podder et al. [23], presented an Arabic sign language recognition system that records sign gestures through RGB videos. They used two datasets: the raw datasets and the face-hand region-based segmented dataset produced from the raw dataset. The authors construct six different models of CNN-LSTM- SelfMLP architecture, by using MobileNetV2 and ResNet18-based CNN backbones and three SelfMLPs. The accuracy of MobileNetV2-LSTM-SelfMLP on the segmented dataset achieved an accuracy score of

87.69%. In summary, the face-hand region-based segmentation and SelfMLP-infused MobileNetV2-LSTM-SelfMLP demonstrated significant advancement in Arabic Sign Language (ArSL) recognition.

Moreover, Tolentino et al. [33], developed a system that translates static American sign language into its corresponding English word using deep learning. Their system achieved an accuracy score of 90.04% for letter recognition, 93.44% for number recognition, and 97.52% for static word recognition. They used 2400 images with the size of 50x50 for each letter/number/word gesture for training.
For translating sign language, Tharwat et al. [5] proposed an alphabetic Arabic sign language dataset and developed a method for recognition systems (AArSLRS) using KNN. The proposed approach achieved an accuracy rate of 99.5%.

In Nada Ibrahim et al. [6], developed an automatic visual Sign Language Recognition System that translates isolated Arabic word signs into text. Their dataset was made of 30 isolated words, the results showed that the system has a recognition rate of 97%.

To translate sign language Sanmitra et al. [32], built a real time machine learning system for sign language detection that uses the PC camera to capture the images. The system recognizes the signs as words instead of single alphabets to avoid the slowness, then translates it into a text. The model has been trained using the SSD ML Algorithm with a dataset of 20 images, which obtained an accuracy of 85%.

2.3.2 Review of Relevant Applications

- **Tarjuman application**(الترجمان)

Tarjuman application is an educational application. There are virtual characters that represent the chosen word using Arabic Sign Language (ArSL). The user can select the word in three ways: writing the word as text, choosing a word from the dictionary, or choosing a word using the voice recorder.

- **Sign Language** (لغة الإشارة)

Sign language application is an educational application that provides training programs in Emirate's sign language for more than one category. The user chooses the word he wants to learn either by searching the list or typing the word in the search option. When choosing a word, a short video appears to teach the user how to represent it. There is also a one test for all categories by showing a video representing a specific word, and the user must choose the appropriate word.

- **sign dictionary** (القاموس الإشاري)

The sign dictionary application is an educational application specialized in the Kuwaiti sign language. It has many categories such as "*الأمثال الشعبية*" and "*سور القرآن*". When you choose a word from one of the categories, a video appears showing how the word is represented.

2.3.3 Relationship Between the Relevant Work and Our Own Work

Table 2.1 relationship between our work and other papers

Research Papers			
	Dataset	**Approach**	**Translation**
Our work	1- SSL Library[18]. 2- ArSLR [20].	Machine learning	text-based Voice-based
SSLR System based on CNN [3]	The standard Saudi sign dictionary.	Deep learning (CNN)	No
ArSLR with CNN [29]	ArSL.	Deep learning (CNN)	No
Image-based ArSLR system [2]	the ArSL 30 letters	Machine learning (SVM, HOG)	No

Building a largest SSL dataset [31]	KSU-SSL database.	Deep learning (CNN)	No
ArSLR System for Alphabets Using ML Techniques [5]	Arabic Alphabet Sign Language	Machine learning (KNN)	No
Hand Gesture Recognition for SL Using 3DCNN [30]	1- KSU-SSL Dataset 2- ArSL Dataset 3- Purdue RVL-SLLL American Sign Language Dataset	Deep learning (3DCNN)	No
An Automatic ArSLR System (ArSLRS)[6]	ArSL database, Benha University.	Machine learning, Deep learning	Yes (signs to text)
Signer-Independent ARSLR System Using Deep Learning Model [23]	ArSL.	Deep learning	No

Table 2.2 relationship between our work and other applications

Applications			
	Dataset	**Translation**	**Educational application**
Our work	1- SSL Library.[18] 2- ArSLR..[20]	text-based Voice-based	Yes
Tarjuman application	ArSL.	Yes (text to signs)	Yes
Sign Language application	ArSL.	No	Yes
sign dictionary application	ArSL dictionary.	No	Yes

2.4 Summary

In this chapter we discussed the concepts related to the sign language (SL) and the technique used to recognize the sign language including machine learning and deep learning. Also, we reviewed the existing work for Arabic sign language recognition and translation and discussed some similar applications to our work. Finally, we compare previous studies our own work.

Chapter 3: System Analysis

3.1 Introduction

In this chapter, we discuss the analysis of our system. Section 3.2 presents the analysis of an existing system, discussing its most important features as well as its drawbacks. Then, section 3.3 defines the requirements of our system including functionals and non-functionals requirements and the domain requirements. After determining the requirements, sections 3.4 demonstrate a UML use case diagram and descriptions of each use case. Finally, section 3.5 discusses in detail the methodology we have chosen to develop our proposed system.

3.2 Analysis of Existing Systems (Translate from text to sign language not the other way)

In this section, we analyze an existing system which is developed for recognizing Arabic sign language. The Tarjuman application ***(تُرجمان)*** [34] is an application that translate Arabic Language to Sign Language, it was designed with two goals in mind: first, to make it easier for deaf people to communicate with the rest of society, and second, to make it possible for non-deaf individuals to communicate with the deaf community. To accomplish these goals, the application offered three practical ways for translating words into sign language: Typing the word, Select the word from an existing dictionary, or Using voice to choose the word. Tarjuman allows users to pick from four virtual characters (Saad, Nada, Sultan, and Osama). The application gives users control over the character's speed and customizes the background color.

Figure 3.1 Main interface show the two ways to translate via text or voice audio.[34]

Figure 3.2 Dictionary page.[34]

Figure 0.3 Translate the word "أبيض" to sign language.[34]

Figure 0.4 Represent the result of the translation.[34]

3.3 Requirements Elicitation

In this section, we will identify the functional and non-functional requirements of our system based on our analysis and examination of existing studies, along with our review of the literature.

3.3.1 Functional Requirements

The functional requirements describe the functions and the behavior of the system under specific conditions [35]. Based on our analysis, the functional requirements of our system are defined as user requirements and system requirement as follows:

3.3.1.1. The User Requirements

The user shall be able to:

- Create an account.
- Sign into the application.
- Learn words and phrases in Saudi delicate sign language based on various categories.
- Take a test after completing every category.
- Review the results of the test, including the mistakes.
- Open the camera and capture the hand gestures video.

3.3.1.2. The System Requirements

The system shall be able to:

- Access the user's device camera.
- Display learning lessons videos to the user based on the selected category.
- Recognize the signs from the user using the camera and detecting the movements.
- Displays the result of the translation on the screen.

- Display the result of learning the Saudi sign language test after completing the category's lessons.

3.3.2 Non-Functional Requirements

Non-functional requirements are known as quality requirements, and they describe how the system should perform [35]. The non-functional requirements of our system are defined as follows:

Accuracy: The system should calculate the data correctly and show the accurate output.[36]

Scalability and Performance: The performance of the application should not be affected when the model captures high speed of hand's gestures especially in translating process.

Usability: The system's interface should be friendly, and easy to use.

Speed: The system should return the result as fast as it could. In addition, it should take little time to translate each hand gesture.[37]

Reliability: The system should be able to work and show the results without any errors.

3.3.3 User Requirements or Domain Requirements

The domain requirements describe the application environment in which the system operates and system properties.[38]

- The user should have an internet connection to access our application.
- The user should grant our application permission to access the device camera.
- The application supports only the Arabic language and the Saudi sign language (SSL).

3.4 Requirements Specification

3.4.1 **Use Case Diagram**

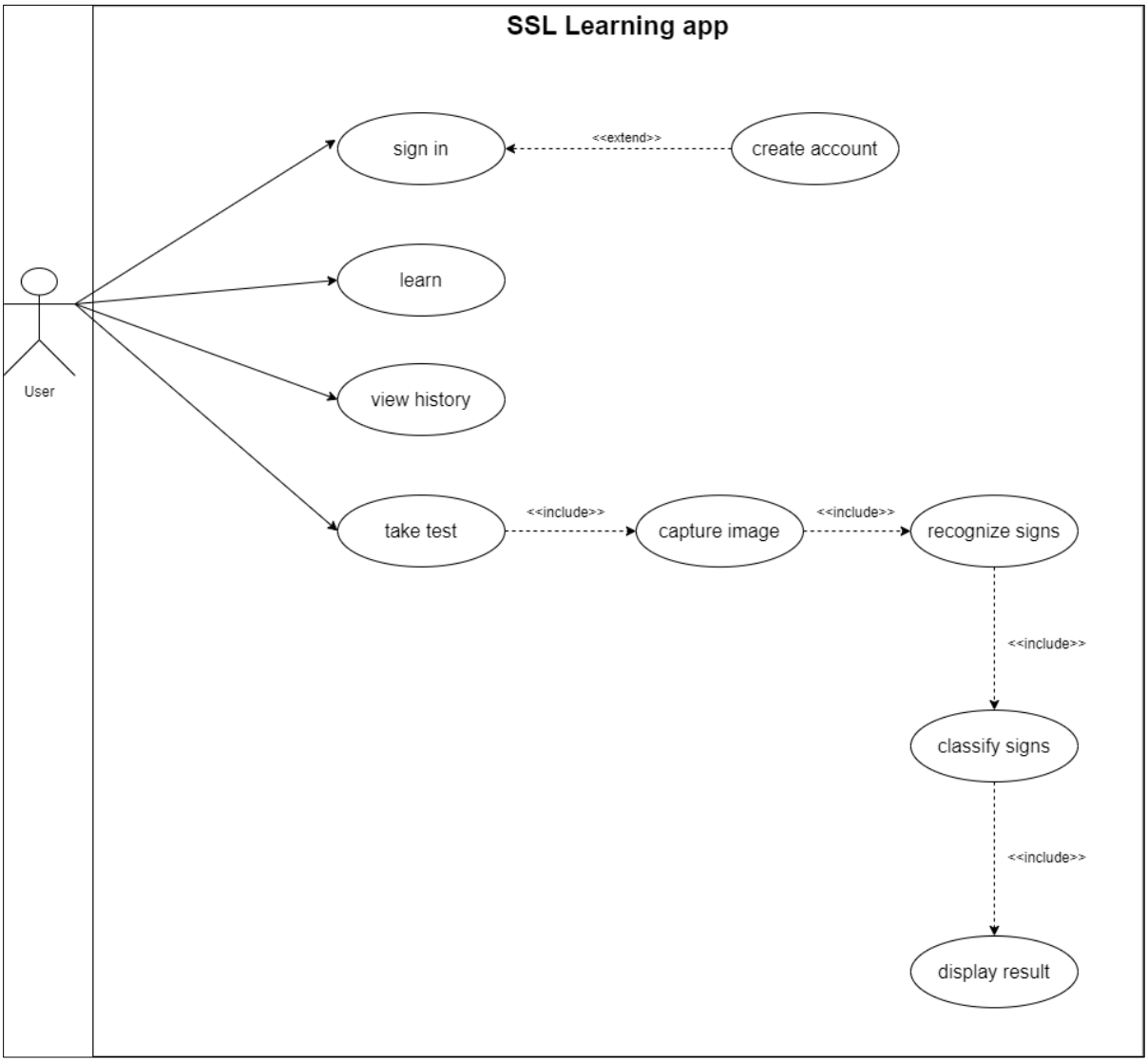


Figure 0.5 Use case diagram.

3.4.2 **The use case description**

Table 0.1 Description of Learn process use case.

Use Case Name:	Learn
Participation actor:	User
Entry condition:	Users have an active account.
Exit condition	Users can start learning SSL.
Flow of event	• Users sign into his/her account. • Users select LEARN, then choose a category they want. • Display lessons available in the selected category.

Table 0.2 Description of Take test process use case.

Use Case Name:	Take test
Participation actor:	User
Entry condition:	• Users have an active account. • Users start learning in specific categories. • Users must start taking tests.
Exit condition:	Users finish the test, and the result will be displayed to them.

Flow of event:	1. Users sign into their account. 2. Users have completed learning all lessons available in the selected category. 3. Users select TEST option. 4. User starts capturing sign language image for each word in the test. 5. Users must submit the record. 6. The system displays the test results to the user.

Table 0.3 Description of View history process use case.

Use Case Name:	View history
Participation actor:	User
Entry condition:	1. Users have an active account. 2. Users must take at least one test.
Exit condition:	The user can see the score of the tests, find out where there are mistakes (if any) and relearn.
Flow of event:	1. Users sign into their account. 2. View the history of all previous tests.

Table 0.4 Description of Recognize and Classify signs process use case.

Use Case Name:	Recognize and Classify signs
Participation actor:	User
Entry condition:	1- Users have an active account. 2- Allow access to the camera. 3- Users must start translating or taking tests. 4- Users must finish a captured.
Exit condition:	1- Signs have been recognized and classified.
Flow of event:	1. Users sign into their account. 2. Users select the TRANSLATE or TEST sections. 3. Users finish capturing. 4. The recording will be recognized and classified by the system.

Table 0.5 Description of Display result process use case.

Use Case Name:	Display result
Participation actor:	User
Entry condition:	1- Users have an active account. 2- Allow access to the camera. 3- Users must start translating or taking tests. 4- Users must finish a captured. 5- System finish recognizing signs
Exit condition:	The result will be displayed.

Flow of event:	1. Users sign into their account. 2. Users select the TRANSLATE or TEST sections. 3. Users finish capturing. 4. The capturing will be recognized and then classified by the system. 5. Display the results.

3.5 Developmental Methodology

Developmental Methodology is used for systematically organizing the best ways to develop systems in an efficient way. It may include the descriptions of work that will be performed at each stage of the development process [39]. To develop our system, we decided to use Agile Methodology.

The agile methodology is an approach for managing projects which involves dividing the project into phases [40]. The 'Agile' methods were formally defined by Edmonds in 1974, It involves multiple iterative development schedules that aim for improving the output with every iteration, and each iteration goes through all the steps of design as shown in Figure 3.7 [41, 42]. Agile methodology includes six phases as follows: [43]:

1- Requirements gathering: Define the requirements and plan the time and effort that needed for the project.

2- Design the requirements: After identifying the project, the stakeholders become involved to define requirements.

3- Construction/ iteration: Designers and developers start working on the project.

4- Testing: Test the performance of the product.

5- Deployment: Publish the product to the work environment of user.

6- Feedback: Receive feedback from users and the team would work through it.

We found this method is the most suitable one since we need to test the models in each designing and developing step so flexibility and ability to move around between steps is needed.

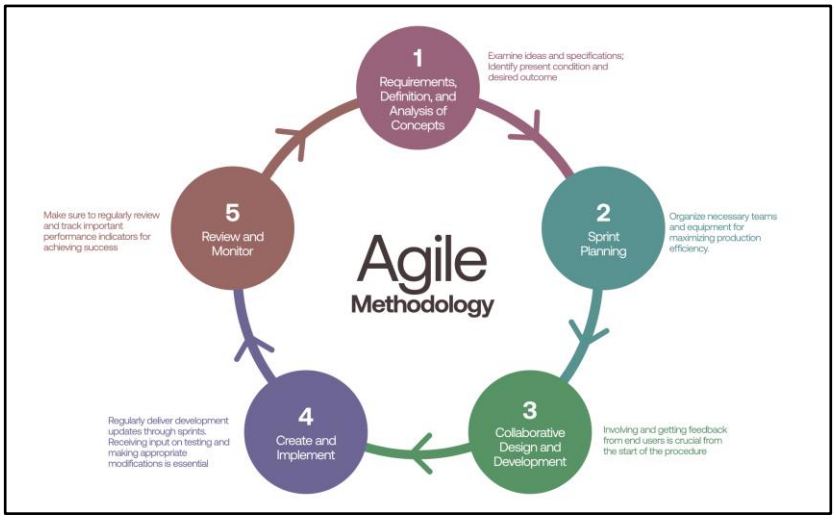


Figure 0.6 Agile methodology process [42].

3.6 Summary

In this chapter, we discussed and analyzed the system requirement. We started by analyzing an existing application and based on the analysis, we have defined the functional and non-functional requirements of our system. The functional specifications have been illustrated using the UML use-case diagram along with its description table. Finally, we discussed the developmental methodology that we utilized to develop our proposed system.

Chapter 4: System Design

4.1 Introduction

System design is a phase where the software system's overall architecture and individual components are specified. In this chapter, section 4.2 provides the architecture of our proposed system. Section 4.3 presents the structural static and dynamic models of the system, and how each module interacts with one another. Additionally, section 4.4 depicts the database design and describes how the system's algorithm is implemented. Section 4.5 illustrates the user interface design, and lastly, section 4.6 summaries the chapter.

4.2 Architectural Design

Architectural Design is concerned with defining the top-level structure of the software system, this includes making decisions of how the system will be organized and how the components inside the system will interact. Figure 4.1 illustrates the proposed system architecture and shows the relationship and the interaction between its three main components. The data pre-processing component processes the recorded videos uploaded by the user in order to be transmitted to the Recognition or translation models. The recognition model receives the processed videos from the previous module and to perform the recognition task and then return the results to the user. if the user wants to translate the recorded video, the preprocessed data will be transmitted to the translation model to recognize the sign and translate it into written text or recorded audio and displays the results to the user.

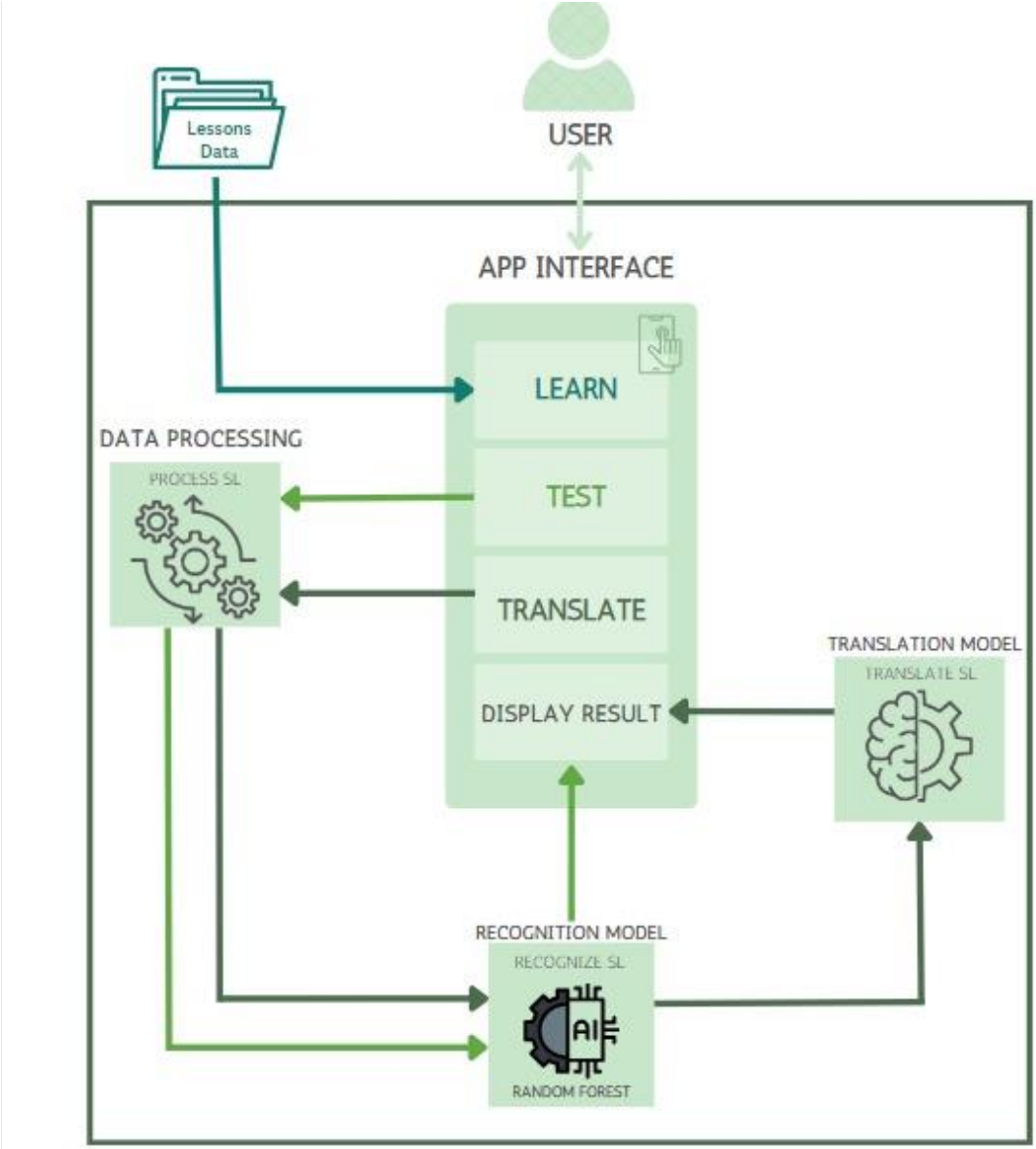


Figure 4.1 SSL Architecture Diagram.

4.3 Object Oriented Design

Object-Oriented Design (OOD) assists the writing of programs by organizing and modeling the system as a collection of interacting objects and classes [44].

4.3.1 Structural Static Models

This Section provides an overview of the Structure of the system using the class diagram. Figure 4.2 shows the Class diagram of the proposed SSL system.

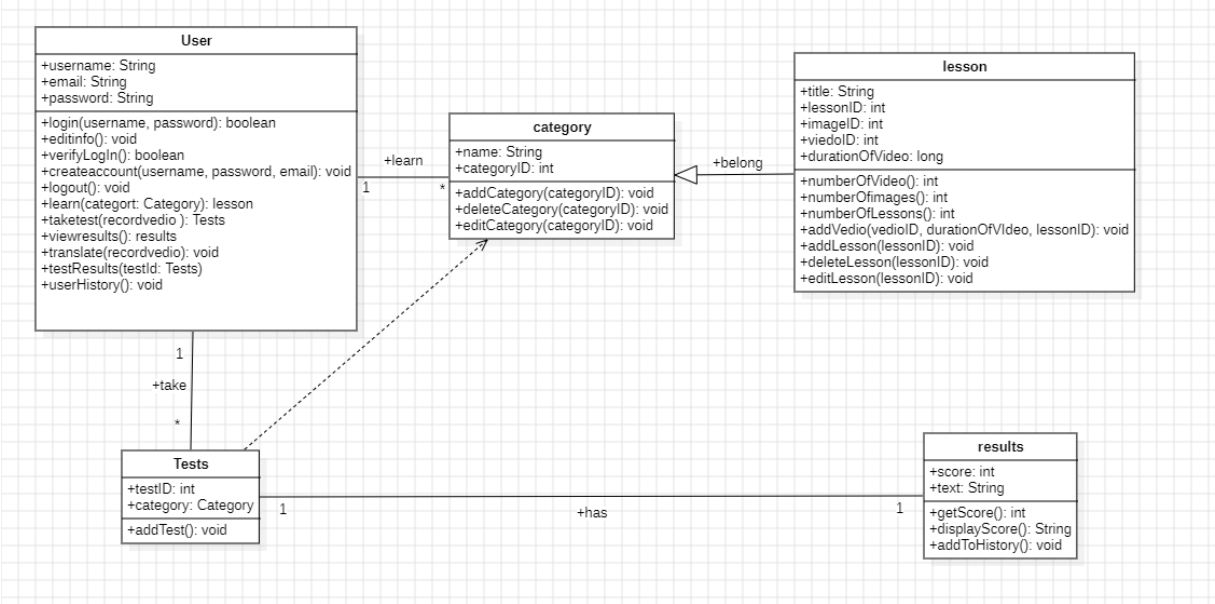


Figure 4.2 SSL Class Diagram.

Figure 4.3 shows the Flow Chart structure of our system which displays the process of the SSL system from the start of opening the program till the user gets the results.

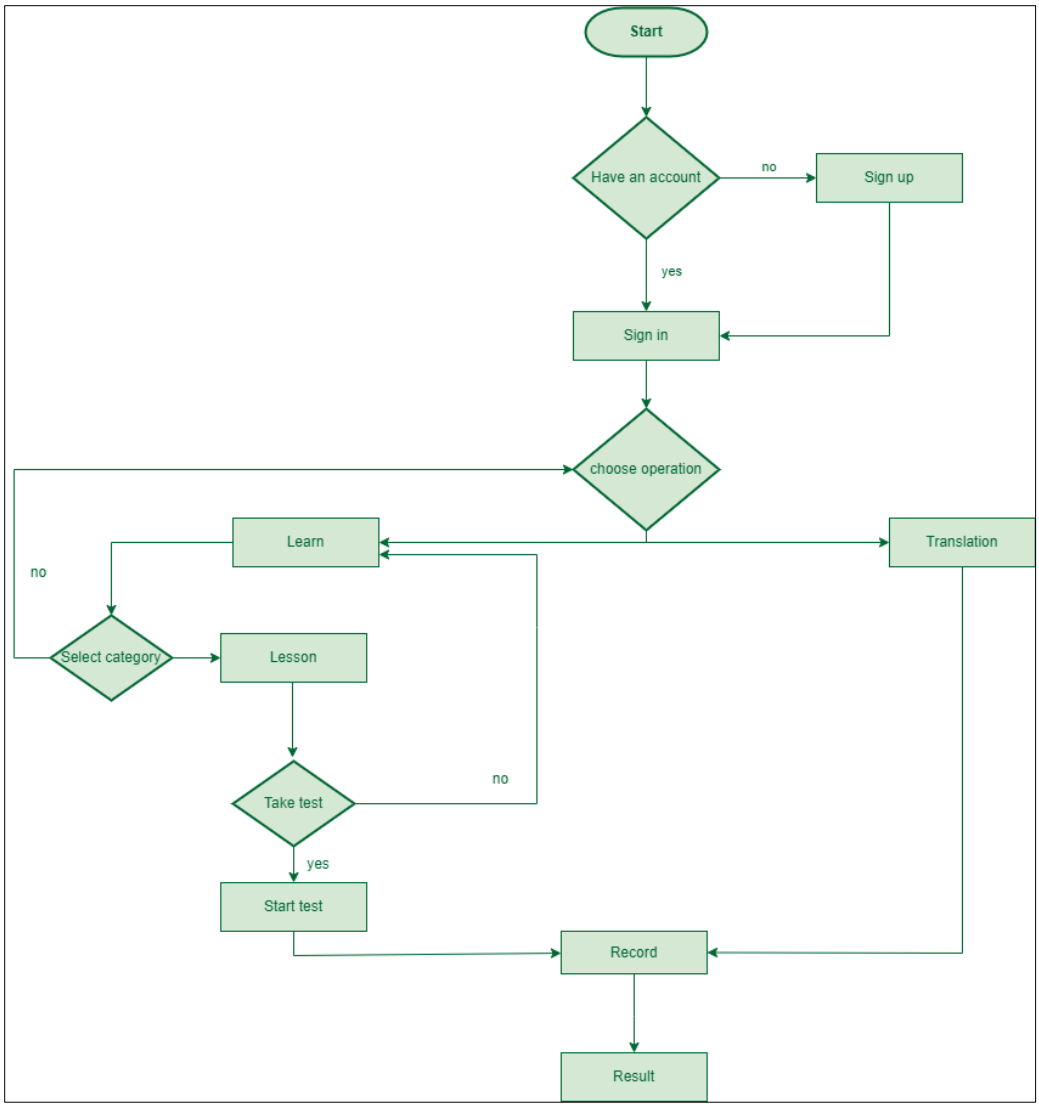


Figure 4.3 SSL Flowchart Diagram.

4.3.2 Dynamic Models

This section demonstrates the Dynamic aspect of the system using the sequence diagram.

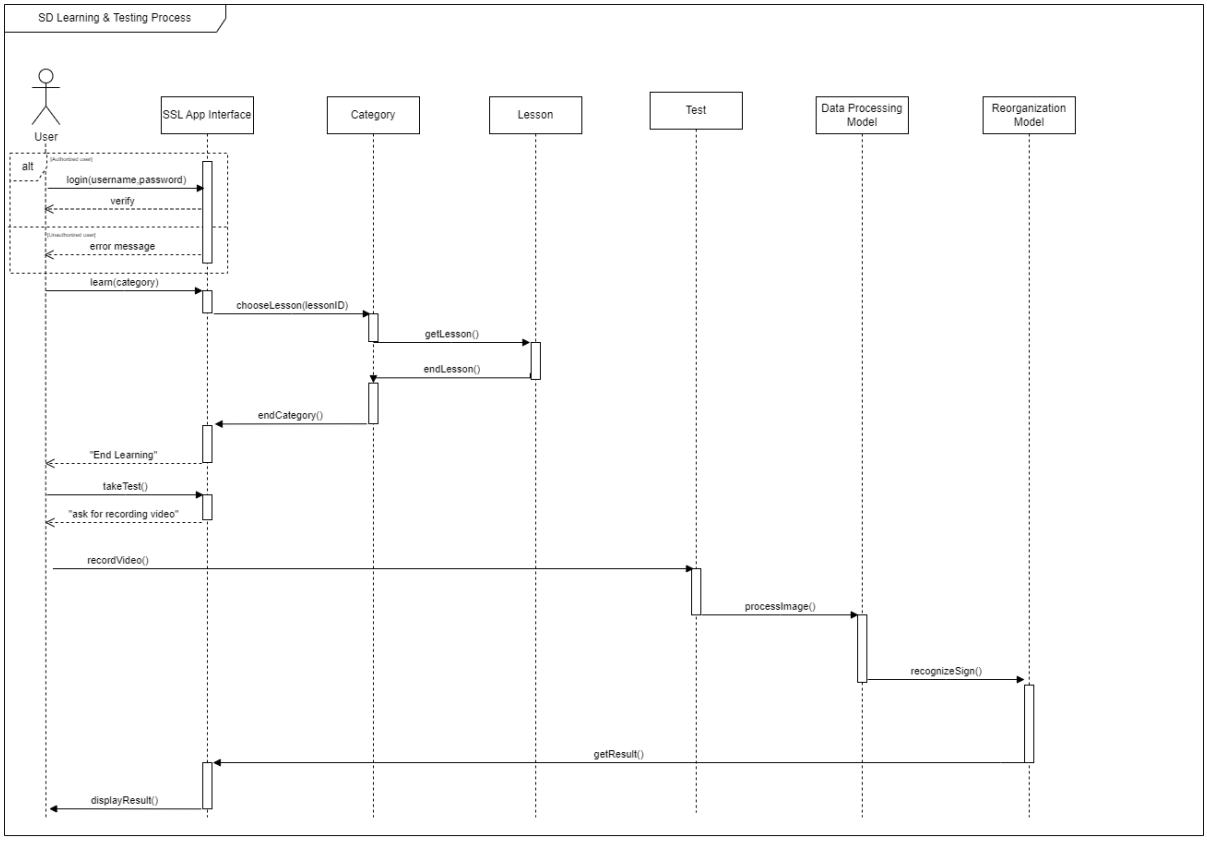


Figure 4.4 SSL Sequence Diagram for Learning/Testing.

4.4 Data Modeling

This section shows the main entities of the system, its attributes, and the relationships between them.

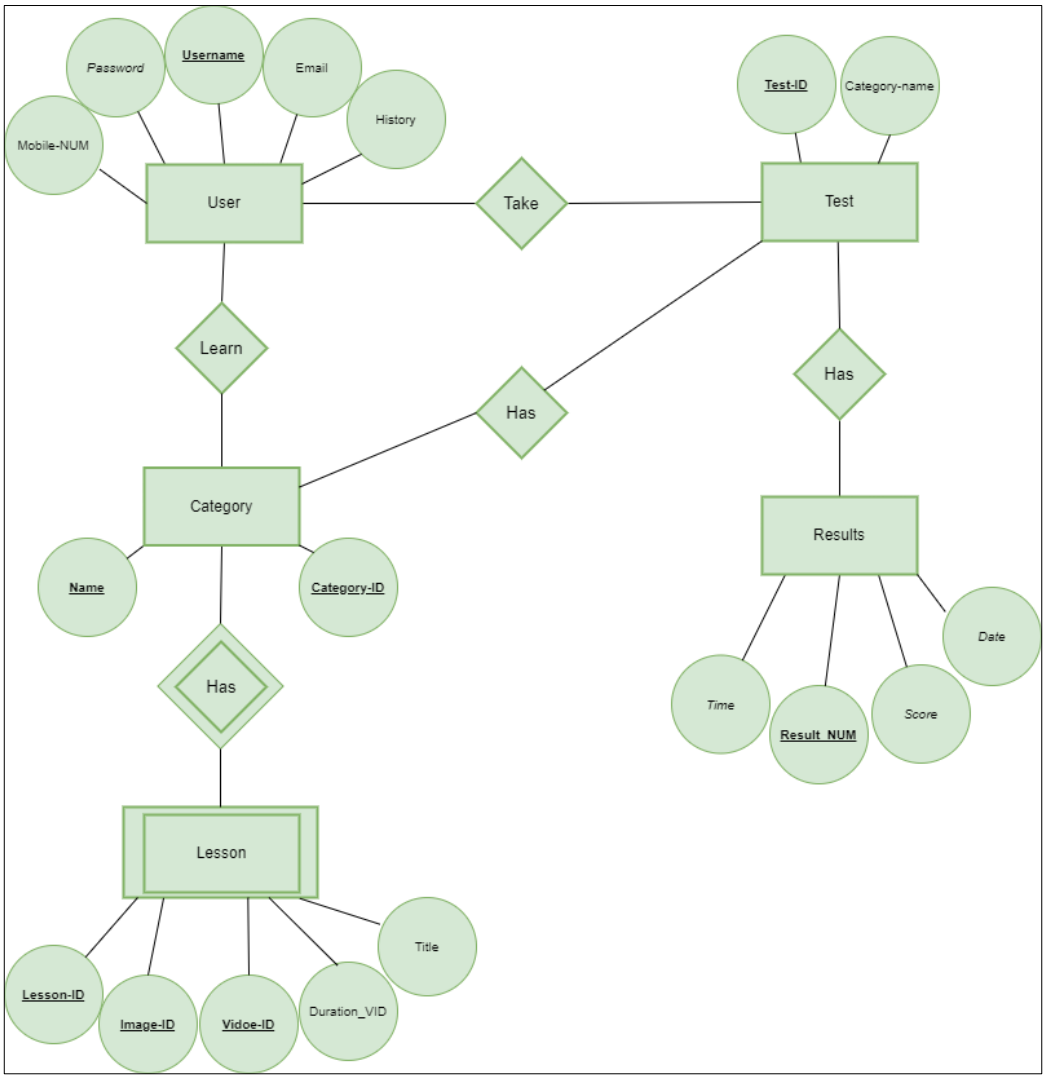


Figure 4.5 SSL ER Diagram.

4.5 User Interface Design

The system user interfaces are illustrated in Figures 4.7- 4.18.

Figure 4.6 Our application.

Figure 4.7 Sign up page.

Figure 4.8 Login page.

Figure 4.9 Homepage.

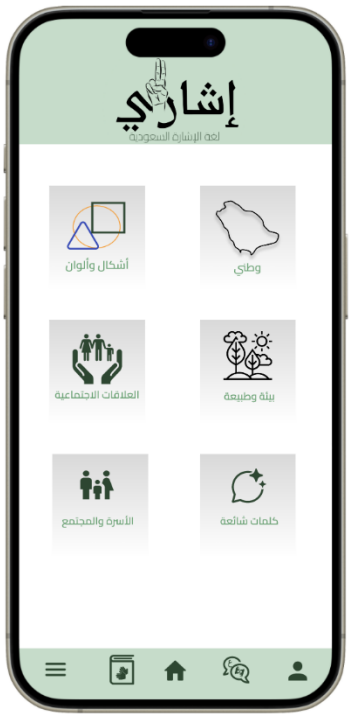


Figure 4.10 Categories page.

Figure 4.11 Lessons page.

Figure 4.12 Lesson page.

Figure 4.13 Test page.

Figure 4.14 Result of test page.

Figure 4.15 Translate page.

Figure 4.16 Record video page.

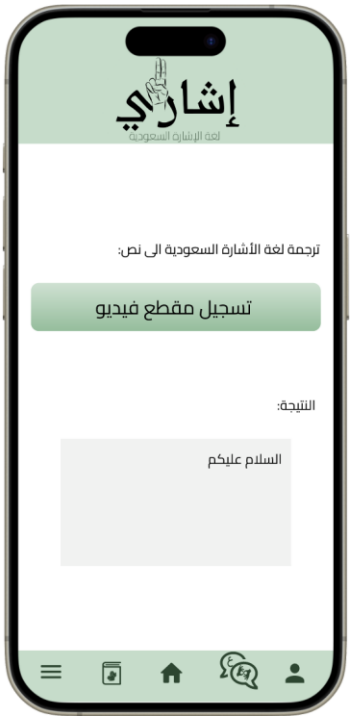


Figure 4.17 Result of translate page.

4.6 Summary

This chapter introduced the architecture of the SSL system by utilizing the architectural design. Both static and dynamic models were designed to describe the system from different aspects. Finally, the chapter presented an initial design of the system's user interface.

Chapter 5: System Implementation

5.1 Introduction

This chapter discusses the implementation phase of Esharati application. Section 5.2 presents the tools and languages used in developing the proposed application. Section 5.3 explains how the system design in chapter 4 relates to the implementation. Section 5.4 introduces the most important codes used in developing the recognition model and the application. Section 5.5 illustrates the system results and provides the unit and system testing results. Section 5.6 summarizes the main highlights discussed in the chapter.

5.2 Tools and Languages

To develop our Esharati app, we conducted research to identify the best tools and languages that serve our system.

Table 5.1 Tools and Languages.

Tools and Languages	Purpose
	**Python** is a high-level programming language used to devolve the application**.**
	**PyCharm** is a code editor that was used to develop the application.
	**Canva** is a graphic design and template editor app used for creating the logo and other images.

	**GitHub** is a platform for online software development, used to share our updates on the code and keep tracking our progress as a team.
	**Google Colab** is an online cloud-based used to training the model.
	**Google Drive** is a file storage and synchronization service used to store datasets.
	**Drawio** is a website for drawing software diagrams used to design the diagrams**.**
	**SQLite** is used to manage the database**.**

5.3 Mapping Design to Implementation

In this section, we will describe how the designs of our proposed system in Chapter 4 align with the implementation. Most of the system front-end and back-end mentioned in the design chapter have been implemented in our system. Few changes have been made due to the resources and time limitations including the following:

An attempt has been made to develop a deep learning-based model. However, due to the limitation of the dataset we have utilized classical machine learning algorithms. Also, in our proposed applications we intend to record video where the user is performing sign language but due to the quality of the data, we have access to, we decided to capture images from the users instead of videos. The translation feature of the sign language into text has been

considered as future work due to the challenges we have faced in finding datasets. Regarding the system user interface, we have added new interfaces to enhance the user experience such as profile and history interfaces as shown in figures 5.1-5.4.

Figure 5.1 new sign in/up page

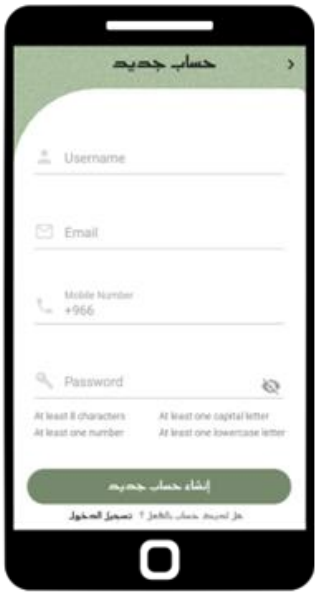


Figure 5.2 new Sign-up page

Figure 5.3 User account page

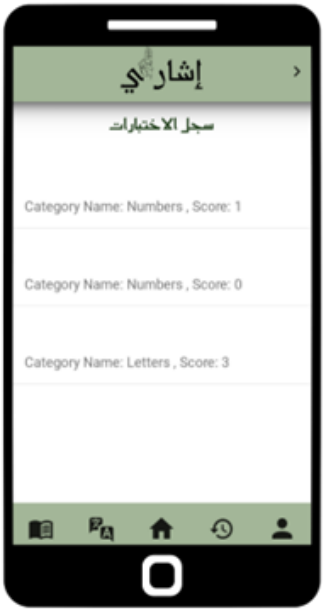


Figure 5.4 history page

5.4 Main Important Codes

5.4.1 Model Building and Training

5.4.1.1 Dataset Collection.

In order to develop sign language recognition model specifically for Saudi sign language, we need a Saudi sign language training and testing sets. During the search phase of a suitable dataset, we faced a critical challenge in finding public Saudi sign language either in video or image format. From this point, we have decided to compare the Arabic sign language and the Saudi sign language to find the similarity between the two languages. Accordingly, we have found that there are 83 similar signs between the two languages including the 28 Alphabets and the 10 numbers.

As a result, we have filtered out the Arabic sign language (ArSL) datasets provided by In Ala Sidig [20] to extract the alphabet and number signs records to use them in training our model. The dataset includes ten classes with a total of 8,162 training images for recognizing sign language of numbers and 2,132 images testing. For the alphabet, we have 28 classes with 25,047 images for training and 13,382 for testing. Each class in the number categories, contains approximately 200-250 images, while each class in alphabet categories contains approximately 500 images. All the images in both training and testing sets are colored images and are segmented from recorded videos by [20]. Figures 5.5-5.7 show examples of the images in the dataset.

Figure 5.5 Number nine by signer 1[20].

Figure 5.6 Number by signer 2[20]

Figure 5.7 Letter "تاء" by signer 1[20].

5.4.1.2 Model Development.

In order to start the data preparation and model development process, we need to import the necessary libraries and framework. The libraries utilized are the follows:

- **OS library:** is a library to interact with the operating system and manage folders. We used the library to access the data folder .

- **Matplotlib library** is a plotting library for creating a visualization, used to visualize our data.
- **Mediapipe library:** A Google framework often used for detection. We used it to detect the landmark of hand.
- **CV2:** A library in Python that is often used for process image and video. We used it to convert images from BGR into RGB.

After setting the environment, we downloaded the data in JPG format and read the data using *imread* function which is provided by CV2 module from the OpenCV library which is a library developed specifically for computer vision purposes as shown in Figure 5.8. After uploading the data, we process the images, define the images width and height, and apply hand marking to detect the hand in the images as well as the fingers positions in order to extract features to train the model as illustrated in Figure 5.9.

For training and validation, we have divided the dataset into 80% for training and 20% for validation. This is applied for both numbers and alphabet recognition models. To start the model training phase, we firstly imported the necessary libraries for machine learning algorithms. The built-in library are as follows:

- **sklearn.metrics.confusion_matrix:** It is a function from scikit-learn used to determine the performance of the model.
- **sklearn.model_selection.train_test_split:** It is a function used to split the dataset into training and testing.
- **google.colab.drive:** it is used to provide access to Google Drive in order to access the dataset.
- **pickle:** It is a function used to save the trained model.
- **numpy:** is a library used for working with arrays, we used it to convert data from list to array.

Figure 5.8 shows these built-in libraries that are important for machine learning-based models.

```
[2]  import os
     import matplotlib.pyplot as plt
     import mediapipe as mp
     import cv2
     from sklearn.metrics import confusion_matrix
     from sklearn.model_selection import train_test_split
     from google.colab import drive
     import pickle
     from sklearn.ensemble import RandomForestClassifier
     import numpy as np

     drive.mount('/content/drive')
```

Figure 5.8 important libraries

We have utilized Random Forest (RF) and Support Vector Machines (SVM) algorithms to compare their performance in recognizing the Saudi sign language. The extracted features in the previous step were fed into the algorithm to learn the pattern in order to identify the correct class as depicted in Figure 5.9. This step was applied to both RF-based model and SVM-based model.

```
for dir_ in os.listdir(traindatasets):
    for img_path in os.listdir(os.path.join(traindatasets, dir_)):
      data_aux = []
      img = cv2.imread(os.path.join(traindatasets, dir_, img_path))

      img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

      results = hands.process(img_rgb)
      if results.multi_hand_landmarks:
          for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    #print(x)
                    data_aux.append(x)
                    data_aux.append(y)
                    print(x, y)
          data.append(data_aux)
          labels.append(dir_)
```

Figure 5.9 processing images.

After that, we fit the model using the fit () function and use the predict () function for predicting the class of a given image as shown in Figure 5.9. After that, we have saved the model using as shown in Figure 5.10.

```
# Train the model
model.fit(x_train, y_train)

# Predict on the test set
y_predict = model.predict(x_test)
```

Figure 5.10 Predict the labels.

```
# Save the model
f = open('/content/drive/MyDrive/train datasets/LETTERS/lettersRFM2.pickle', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

Figure 5.11 Save the model as pickle format.

5.4.2 Model Testing (Experimental results)

We have tested the performance of both RF and SVM models using the independent testing set. We calculate the model's accuracy, precision, and recall which are commonly used to evaluate the performance of machine learning -based models. Table 5.2 shows the results of both developed modes for numbers and alphabet categories.

Table 5.2 Models performance results

Metric	**Number Category**		**Letter Category**	
	RFM	SVM	RFM	SVM
Accuracy	99%	88%	100%	87%
Precision	100%	88%	100%	87%
Recall	100%	88%	100%	87%

5.4.3 System Development.

We have used Python language and KivyMD library to develop our application, we had made classes for each category in the application, mainly there were three classes: Alphabet, Numbers, and Words which contain the most used words in the language, here are some snapshots of the main parts of the code.

5.4.3.1 Learning sign language.

To help the user in learning sign language in our application, we had utilized the dataset of SSL library by Saudi Association for Hearing Impairment [18] and classified it into three categories: Alphabet, Numbers, and Words. Videos belong to each category where organized and then uploaded it into our application as show in Figure 5.13.

```
<Happy>:
    name: "happy"
    MDFloatLayout:
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1  # Background color
            Rectangle:
                pos: self.pos
                size: self.size
#TOPBAR
    MDNavigationLayout:
        MDScreenManager:
            MDScreen:
                MDTopAppBar:
                    elevation: 4
                    pos_hint: {"top": 1}
                    md_bg_color: "#a3b698"
                    specific_text_color: "#4a4939"
#LOGO
                Image:
                    size_hint:.7, .1
                    source:'zz1-1.png'
                    pos_hint: {"center_x": 0.5, "center_y": .95}
    MDFloatLayout:
```

Figure 5.12 upload lesson word "سعيد" screen.

```
BoxLayout:
    orientation: 'vertical'
    size_hint: 0.88, 1
    pos_hint: {'center_x': 0.5, 'center_y': 0.5}
    Video:
        id: happy_player
        source: 'happy.mp4'
        state: 'play'
        options: {'eos': 'loop'}
        size_hint: None, None
        size: self.parent.size
```

Figure 5.13 uploading the lesson video to the screen.

Figure 5.14 the lesson video played on the user interface.

All the lessons are placed in different screens/classes based on their category.

```
# LEARNING NUMBERS
<Lessons3>:
    name: 'lessons3'
    MDFloatLayout:
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1  # Background color
            Rectangle:
                pos: self.pos
                size: self.size
#TOPBAR
    MDNavigationLayout:
        MDScreenManager:
            MDScreen:
                MDTopAppBar:
                    elevation: 4
                    pos_hint: {"top": 1}
                    md_bg_color: "#a3b698"
                    specific_text_color: "#4a4939"
#LOGO
                Image:
                    size_hint:.7, .1
                    source:'zz1-1.png'
                    pos_hint: {"center_x": 0.5, "center_y": .95}
```

Figure 5.15 the lesson screen for numbers in kivy file.

Figure 5.16 the lesson screen for numbers in the application.

5.4.3.2 Learning-level Evaluation.

In order to evaluate the user learning process, we use test after each category to assess the level of the user. We defined classes for each category in the application, in Figure 5.17 shows the code of class *"Numbers"* where all the lessons of this class are implemented along with its test questions. We imported Arabic ReShaper and Get Display libraries so we can display the questions in Arabic since kivyMD default font doesn't support Arabic language.

```
class Q_numbers(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.username = None
        self.question_index = 0  # current question index
        self.score = 0  # user's score
        self.total_questions = 0  # total number of questions
        self.incorrect_questions = []

        # convert to arbic
        res0 = get_display(arabic_reshaper.reshape("    صفر "))
        res1 = get_display(arabic_reshaper.reshape("   واحد "))
        res2 = get_display(arabic_reshaper.reshape("  اثنين "))
        res3 = get_display(arabic_reshaper.reshape("   ثلاثة "))
        res4 = get_display(arabic_reshaper.reshape(" اربعة "))
        res5 = get_display(arabic_reshaper.reshape("   خمسة "))
        res6 = get_display(arabic_reshaper.reshape("    ستة "))
        res7 = get_display(arabic_reshaper.reshape("   سبعة "))
        res8 = get_display(arabic_reshaper.reshape(" ثمانية "))
        res9 = get_display(arabic_reshaper.reshape("   تسعة "))

        # Define the questions data
        self.questions = [
            {
                "question": res0,
                "correct_answer": "0001"
            },
            {
                "question": res1,
                "correct_answer": "0002"
            },
            {
                "question": res2,
                "correct_answer": "0003"
            },
            {
                "question": res3,
                "correct_answer": "0004"
            },
```

Figure 5.17 Numbers lessons class.

In Figure 5.18, we put the questions in a list containing the question and the correct answer. we assigned each letter to its correct answer that is defined by its label in the data set, e.g. the letter “أ” has the label “0032” and so on for the other letters.

```
        # Define the questions data
        self.questions = [
            {
                "question": res0,
                "correct_answer": "0001"
            },
            {
                "question": res1,
                "correct_answer": "0002"
            },
            {
                "question": res2,
                "correct_answer": "0003"
            },
            {
                "question": res3,
                "correct_answer": "0004"
            },
            {
                "question": res4,
                "correct_answer": "0005"
            },
            {
                "question": res5,
                "correct_answer": "0006"
            },
```

Figure 5.18 Define the correct answers with the same labels in the dataset.

Figure 5.19 Test page, ask the user to sign number three.

In Figure 5.20, we defined a function that captures the user's signs by accessing the user device camera and sending the captured image to another function called *"preprocess_image"* to preprocess it before sending it to the model, we used the CV2 library for this task. When 5 seconds have passed, the image will be captured and sent to "*preprocess_image"* function.

```python
def image_capture_num(self):
    # Check if there are more questions to process
    if self.question_index < len(self.questions):
        # question_data = self.questions[self.question_index]
        # correct_answer = question_data.get("correct_answer", "")

        # Set up video capture
        window_name = "Image Capture"
        cv2.namedWindow(window_name)

        camera_window_x = 470
        camera_window_y = 230

        # Position the camera feed window above the app window
        cv2.moveWindow(window_name, camera_window_x, camera_window_y)
        cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)



        # Start recording
        start_time = time.time()
        while True:
            # Capture frame-by-frame
            ret, frame = cap.read()
            if not ret:
                print("Error: Failed to capture frame.")
                break


            # Display the resulting frame
            cv2.imshow(window_name, frame)  # Use window_name here

            # Check if 5 seconds have elapsed
            if time.time() - start_time >= 5:
                # Capture a picture after 5 seconds
                image_path = f'num_pic_{self.question_index + 1}.jpg'
                cv2.imwrite(image_path, frame)
                self.preprocess_image(image_path)
                break  # Exit the loop after capturing the picture
```

Figure 5.20 Number capture image method

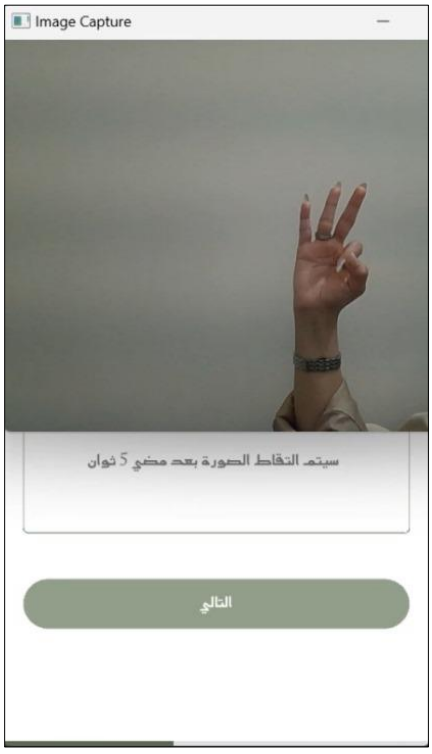


Figure 5.21 Start capturing the user's sign.

In Figure 5.22, the *"predict_number*" has received the preprocessed image from. Here we upload the model and wrote the prediction command *"model.predict"* to predict the signs in image. The prediction result will be sent to the function *"check_answer"* to use it later for checking the user answer.

```
1 usage
def predict_number(self, preprocessed_image):
    # Load the trained model from the pickle file
    with open(r'C:\Users\ahads_q4dtp7v\PycharmProjects\EsharatiNew\modelRFM2_num.p', 'rb') as file:
        data = pickle.load(file)

    # Extract the model from the dictionary
    model = data['model']
    # Debugging: Print the shape or type of preprocessed_image
    print("Preprocessed image data:", preprocessed_image)

    # Now you can use the predict method
    prediction = model.predict(preprocessed_image)
    print("Model prediction:", prediction)  # Debugging: Check what the model predicts
    self.check_answer(prediction)
```

Figure 5.22 Integrate the model in the prediction method.

To verify the correctness of the user answer, the correct answer will be compared with the prediction in the *"check_answer"* method as shown in Figure 5.23. If both values are equal, the question will be calculated, and the score will be increased. If they are not equal, the question will be placed in the list of incorrect answers to be passed later to the results page, and the score will not change. This ensures that if the user skips taking images. The question is considered an incorrect answer.

```
def check_answer(self, prediction):
    # if self.question_index < len(self.questions):
    question_data = self.questions[self.question_index]
    correct_answer = question_data.get("correct_answer", "")
    # len_ques = len(self.questions);
    # Print the data types of prediction and correct_answer
    print("Data type of prediction:", type(prediction))
    print("Data type of correct_answer:", type(correct_answer))

    print("Correct answer:", correct_answer)
    if prediction == correct_answer:
        self.score += 1
    else:
        # If the answer is incorrect, append the question data to the list
        self.incorrect_questions.append(question_data)

    # Navigate to the next question or the result screen
    self.question_index += 1
    if self.question_index < len(self.questions):
        self.display_current_question()
    else:
        # Calculate the total score
        total_score = self.score

        # Generate a unique ID for the test
        test_id = str(uuid.uuid4())

        # Store the total score in the database
        self.store_test_score(test_id, total_score)
        # Navigate to the result screen
        self.manager.current = 'result_num'
        result_screen = self.manager.get_screen('result_num')
        result_screen.update_result(self.score, self.total_questions, self.incorrect_questions)
```

Figure 5.23 check answer method.

```
1 usage
def store_test_score(self, test_id, total):
    # Connect to the SQLite database
    conn = sqlite3.connect('users_data.db')
    cursor = conn.cursor()
    res1 = get_display(arabic_reshaper.reshape("أرقام"))
    # Insert the user's test score into the 'test' table
    cursor.execute( _sql: "INSERT INTO tests (id, score, username, testName) VALUES (?, ?, ?, ?)",
                    _parameters: (test_id, total, self.username, "Numbers"))

    # Commit changes to the database
    conn.commit()

    # Close the connection
    conn.close()
```

Figure 5.24 store the results of tests in the database.

5.4.3.3 Result class

Finally, we send the variables grade, number of questions, and incorrect answers to the results page using *Result_num* class Figure 5.25, so it will be displayed to the user after finishing the test and update the results in the database.

```
4 usages
class Result_num(Screen):
    2 usages (2 dynamic)
    def update_result(self, score, total_questions, incorrect_questions):

        # Calculate and display the final score
        res1 = get_display(arabic_reshaper.reshape(" نتيجتك"))
        self.ids.score_label.text = f"{score}/{total_questions} :{res1} "

        # Provide feedback based on the score
        if score == total_questions:
            feedback = get_display(arabic_reshaper.reshape(" رائع! لقد أجبت على جميع الإسئلة بشكل صحيح "))

        elif score >= total_questions * 0.5:
            feedback = get_display(arabic_reshaper.reshape(" عمل عظيم! لقد اجتزت الاختبار. "))

        else:
            feedback = get_display(arabic_reshaper.reshape(" يمكنك القيام بعمل أفضل. استمر بالتدريب! "))

        self.ids.feedback_label.text = feedback

        # Navigate to the IncorrectNumScreen and pass the list of incorrect questions
        incorrect_num_screen = self.manager.get_screen('incorrect_num')
        incorrect_num_screen.incorrectnum(incorrect_questions)
        self.manager.get_screen('incorrect_num')
```

Figure 5.25 Result class to get the users answers.

Figure 5.26 Display the user result.

5.4.3.4 History class

The History class Figure 5.27 contains two methods, the first is to update the data every time the page is accessed, while the method *populate_data_list* first, we called Database to obtain the score and the name of the category using *Username* after that we prepared the statement that will be repeated as much as user take the test.

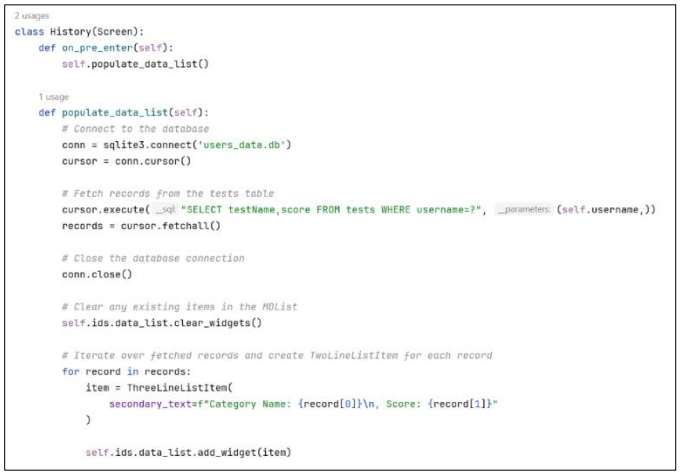


Figure 5.27 History class to record users test results.

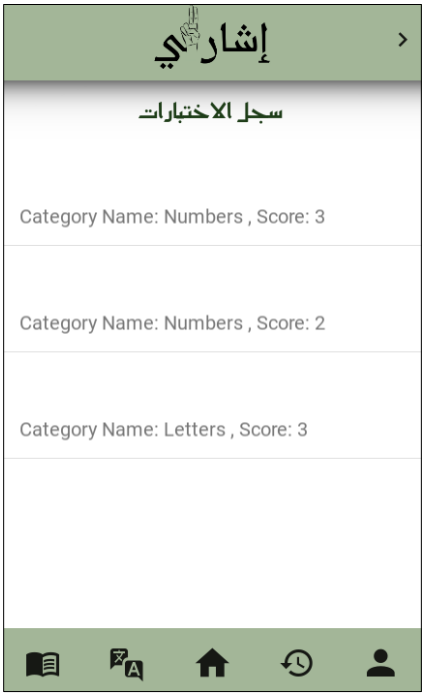


Figure 5.28 user history.

5.5 System Testing.

To check if we met the expected requirements of our application, we use two types of testing unit testing and system testing.

5.5.3 Unit testing

Is the first level in testing hierarchy, it involves testing the individual component of software application [45].

Table 5.3 Unit testing.

Unit Test ID	Method	Input	Output	Test Result
1	Calculates the score of the learning process evaluation	images	score	pass
2	Check the validation of user sign up information	Username, Email, mobile number, password	Valid or invalid	pass
3	Check if login information is valid	username, password	Valid or invalid	pass
4	Reprocessing images	image	Reprocessing image	pass
5	Store test score	Test ID, total score, username, category name	Save test score in database	pass
6	Update the result	Total score, total number of questions	Score of the test	pass
7	See incorrect answer button	incorrect answer	List of questions that answering incorrect	pass

8	History button	Username, category name, score	List of all tests	pass
9	Edit information button	Email, mobile number, password	Save new information	pass
10	Display the lessons	Lesson ID	Lesson's videos/images	pass
11	Start test button	Test ID	Questions of the test	pass

5.5.4 System testing

After testing each unit and test integration between them, now it is time to evaluate the overall functionality and performance of an entire system [46].

Table 5.4 System testing.

Scenario	System responds
System Testing Scenario: User Sign-up **1- Input Validation:** a. Enter a valid username (e.g., "yara"). b. Enter a valid email address(e.g.,"yara35@gmail.com"). c. Enter a valid Saudi phone number (e.g., "+966501234567"). d. Enter a valid password (e.g., "P@ssw0rd123"). **2- Submit Sign-up Form:** a. Click on the "Sign Up" button. **3- Expected Results (Successful Sign-up):**	• **Successful sign-up**: The system signs up Successfully without error messages and redirects the user to the home page. Figure 5.31. • **Invalid inputs**: The system will display an error message if

a. The system accepts the inputs without displaying any error messages. b. Upon successful sign-up, the user is redirected to the home screen. **4- Negative Testing (Incorrect Credentials):** a. Repeat the sign-up process with invalid inputs: • usernames already exist (e.g., "yara" containing invalid or more than 10 characters). • Invalid email address (e.g., "invalid_email.com" missing "@" symbol). • Invalid Saudi phone number (e.g., "12345678" not in the correct format). • Weak password (e.g., "password" not meeting complex requirements). **5- Expected Results (Negative Cases):** a. The system will not accept the inputs and display error messages. b. The system displays appropriate error messages under each field indicating the validation rules that were violated. c. The "Sign Up" button remains disabled until all inputs are valid. d. No sign-up process is completed with invalid inputs.	the username is already existed and clear error messages under each field explaining the issue (e.g., "Invalid username format", "Invalid email format", "Invalid phone number format", "Password must contain at least one uppercase letter, one lowercase letter, one digit, and one special character"). Figure 5.32.
System Testing Scenario: User Sign-in **1- Input Validation:** a. Enter a valid username associated with the user's account. b. Enter the correct password for the corresponding account. **2- Submit Sign-in Form:** a. Click on the "Sign In" button.	• **Successful sign-in:** The system log in Successfully without error messages and redirects the user to the home page. Figure 5.27.

3- Expected Results (Successful Sign-in): a. The system verifies the credentials and grants access to the user's account. b. The user is redirected to the home page after successful log-in. **4- Negative Testing (Incorrect Credentials):** Repeat the sign-in process with incorrect credentials: • Incorrect username. • Incorrect password. **5- Expected Results (Negative Cases):** a. The system displays a clear error message indicating the login failure. b. The error message specifies whether the username or password is incorrect. c. The user is not granted access to the account and remains on the sign-in page.	• **Invalid sign-in:** The system displays an error message "Login failed. the username or password you entered is incorrect please try again." Figure 5.33.

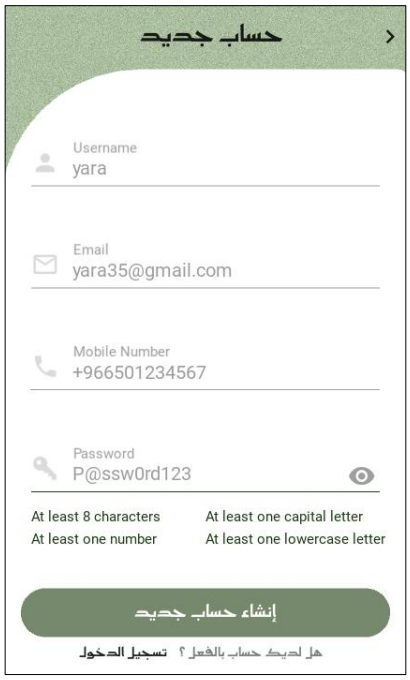


Figure 5.29 Input validation.

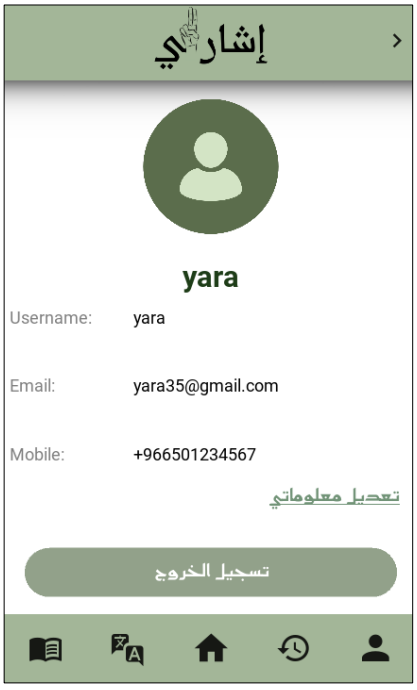


Figure 5.30 Successful sign-up/sign-in.

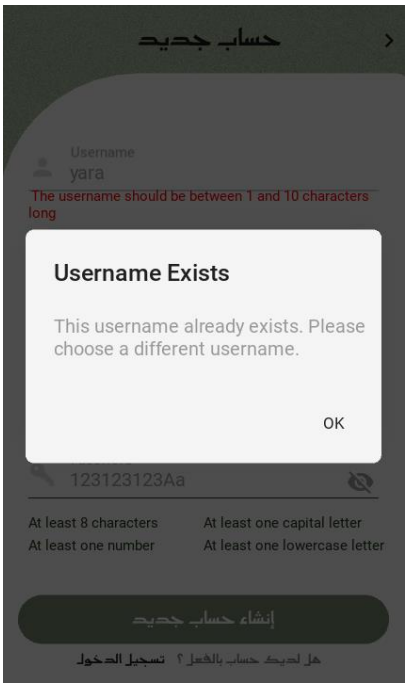


Figure 5.31 Invalid inputs.

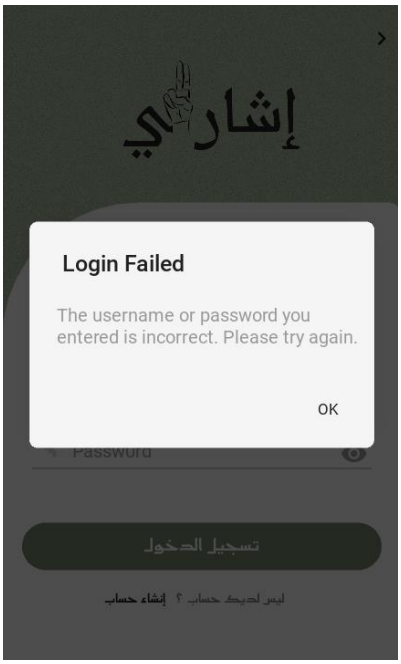


Figure 5.32 Invalid sign-in.

5.6 Conclusion

This chapter describes the tools used to build the system, and discusses the changes made from the design phase in Chapter Four. It also explains three different model types and their results, provides code samples showing key functions of our app, and discusses the datasets used. Two methods were used to test the system for error-free operation. Finally, the chapter presents the system's results and discusses its overall functionality.

Chapter 6: Conclusion and Future Work

~~5.5~~6.1 Conclusion

Sign language serves as an important means of communication for the deaf community which allow them to actively engage within society. Unfortunately, many individuals lack an understanding of sign language which results in difficulties when interacting with deaf people. To address this issue and contribute to the education of people in sign language, this project proposes the development of a system that teaches Saudi Sign Language (SSL) As future work, the option of translating will be available to assists deaf individuals in effectively conveying their messages and enables better understanding and interaction with the wider population. This objective will achieve by utilizing machine learning-based techniques to recognize sign language from captured images by the user and then translate it into written text.

~~6.1~~6.2 Goals Achieved

In the first semester, we have studied and reviewed the literature on sign language recognition and translation, exploring various techniques employed in this field. Also, we have identified and analyzed the requirements for our proposed system by defining both the functional and non-functional requirements. In addition to that, we have designed the system architecture which highlights the major components of our system. Furthermore, we have designed the system's static and dynamic models using class diagram and sequence diagram. Then, in the second semester, as we were building our Esharati app, we successfully implemented secure authentication, allowing users to edit their personal information with updates reflected in the database. We also provided Saudi Sign Language lessons via locally stored media, categorized for easy access and superior performance compared to cloud storage. Additionally, we introduced the 'Test' feature to assess users' sign language understanding by mimicking displayed text using the camera. Our content development was sourced from King Fahad University for Arabic Sign Language, focusing on 'numbers' and 'letters' categories due to data limitations. We integrated Machine Learning to enhance the app's accuracy, outperforming Deep Learning methods. Overall, we met both functional and non-functional requirements,

prioritizing user satisfaction and continuous improvement in features and usability to facilitate better communication within the deaf community.

~~6.2~~6.3 Limitations and Future work

Various Arabic sign language dialects exist, such as, Emirati, Qatari, Jordanian, and Egyptian dialects. However, due to time limitation and a lack of publicly available datasets, we were not able to include them in our work. Also, our system currently only supports static gestures. As future direction, we are planning to extend our work by supporting dynamic signs which include facial expressions and body posture and adding more categories to enrich our dataset and improve the recognition system. We will also develop a translation option that translates signs into text from the recorded video.

References

[1] 'Our Work', WFD. [Online]. Available: https://wfdeaf.org/our-work/

[2] R. Alzohairi, R. Alghonaim, W. Alshehri, S. Aloqeely, M. Alzaidan, and O. Bchir, 'Image based Arabic Sign Language Recognition System', *ijacsa*, vol. 9, no. 3, p. 10, 2018, doi: 10.14569/IJACSA.2018.090327.

[3] A. H. Al-Obodi, A. M. Al-Hanine, K. N. Al-Harbi, M. S. Al-Dawas, and A. A. Al-Shargabi, 'A Saudi Sign Language Recognition System based on Convolutional Neural Networks', *International Journal of Engineering Research and Technology*, vol. 13, no. 11, p. 7, Nov. 2020, doi: 10.37624/IJERT/13.11.2020.3328-3334.

[4] S. S and S. S, 'American Sign Language Recognition System: An Optimal Approach', *IJIGSP*, vol. 10, no. 8, pp. 18–30, Aug. 2018, doi: 10.5815/ijigsp.2018.08.03.

[5] G. Tharwat, A. M. Ahmed, and B. Bouallegue, 'Arabic Sign Language Recognition System for Alphabets Using Machine Learning Techniques', *Journal of Electrical and Computer Engineering*, vol. 2021, pp. 1–17, Oct. 2021, doi: 10.1155/2021/2995851.

[6] N. B. Ibrahim, M. M. Selim, and H. H. Zayed, 'An Automatic Arabic Sign Language Recognition System (ArSLRS)', *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 470–477, Oct. 2018, doi: 10.1016/j.jksuci.2017.09.007.

[7] L. Tucci, 'What is Machine Learning and How Does It Work? In-Depth Guide', Enterprise AI. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

[8] G. L. Team, 'What is Machine Learning? Definition, Types, Applications, and more', Great Learning Blog: Free Resources what Matters to shape your Career! [Online]. Available: https://www.mygreatlearning.com/blog/what-is-machine-learning/

[9] B. Muhammad, 'Machine Learning And Its Disciplines', NF AI. [Online]. Available: https://www.nfaicompany.com/machine-learning-and-its-disciplines/

[10] V. Kanade, 'What is machine learning? Understanding types & applications. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/

[11] ‘Machine Learning Algorithms - Javatpoint’, www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/machine-learning-algorithms

[12] ‘SVM Machine Learning Tutorial – What is the Support Vector Machine Algorithm, Explained with Code Examples’, freeCodeCamp.org. [Online]. Available : https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/

[13] A. Saini, ‘Decision Tree Algorithm - A Complete Guide - Analytics Vidhya’. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/

[14] 'Random Forest Algorithm in Machine Learning ' GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[15] ‘Introduction to Deep Learning’, GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/introduction-deep-learning/

[16] K. O’Shea and R. Nash, ‘An Introduction to Convolutional Neural Networks’. arXiv, Dec. 02, 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[17] Z. Li, F. Liu, W. Yang, and S. Peng, ‘A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects’, *IEEE Trans. Neural Netw. Learning Syst.*, vol. 33, no. 12, p. 21, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[18] ‘مكتبة لغة الاشارة السعودية’. [Online]. Available: https://sshi.sa/

[19] Al-Anoud Center for the Development of the Disabled.

[20] Arabic Sign Language Recognition by Ala addin Ismaeil Sidig.

[21] ‘القاموس الإشاري العربي للصم’. [Online]. Available: https://menasy.com/

[22] H. M and S. M.N, ‘A Review on Evaluation Metrics for Data Classification Evaluations’, *IJDKP*, vol. 5, no. 2, p. 11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[23] K. K. Podder *et al.*, ‘Signer-Independent Arabic Sign Language Recognition System Using Deep Learning Model’, *Sensors*, vol. 23, no. 16, p. 7156, Aug. 2023, doi: 10.3390/s23167156.

[24] Amsten, ‘Evaluation Metrics in Machine Learning’, GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/

[25] A. Lekhtman, ‘Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Towards Data Science’. [Online]. Available: https://towardsdatascience.com/should-i-look-at-precision-recall-or-specificity-sensitivity-3946158aace1

[26] E. D, ‘Accuracy, Recall & Precision’, Medium. [Online]. Available: https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d

[27] A. Mitrani, ‘Evaluating Categorical Models II: Sensitivity and Specificity’, Medium. [Online]. Available: https://towardsdatascience.com/evaluating-categorical-models-ii-sensitivity-and-specificity-e181e573cff8

[28] ‘What is F-score’, Deepchecks. [Online]. Available: https://deepchecks.com/glossary/f-score/

[29] S. Hayani, M. Benaddy, O. El Meslouhi, and M. Kardouchi, ‘Arab Sign language Recognition with Convolutional Neural Networks’, in *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*, Agadir, Morocco: IEEE, Jul. 2019, pp. 1–4. doi: 10.1109/ICCSRE.2019.8807586.

[30] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, M. A. Bencherif, and M. A. Mekhtiche, ‘Hand Gesture Recognition for Sign Language Using 3DCNN’, *IEEE Access*, vol. 8, pp. 79491–79509, 2020, doi: 10.1109/ACCESS.2020.2990434.

[31] M. Alsulaiman *et al.*, ‘Facilitating the communication with deaf people: Building a largest Saudi sign language dataset’, *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 8, p. 16, Sep. 2023, doi: 10.1016/j.jksuci.2023.101642.

[32] P. Rishi Sanmitra, V. V. Sai Sowmya, and K. Lalithanjana, ‘Machine Learning Based Real Time Sign Language Detection’, IJRESM, p. 5.

[33] L. K. S. Tolentino, R. O. S. Juan, A. C. Thio-ac, M. A. B. Pamahoy, J. R. R. Forteza, and X. J. O. Garcia, ‘Static Sign Language Recognition Using Deep Learning’, *IJMLC*, vol. 9, no. 6, p. 9, Dec. 2019, doi: 10.18178/ijmlc.2019.9.6.879.

[34] Turjeman app.

[35] ‘Functional and Non-functional Requirements: Specification an’, AltexSoft. [Online]. Available: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/

[36] ‘Nonfunctional Requirements’. [Online]. Available: https://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html

[37] ‘9 Nonfunctional Requirements Examples | Indeed.com’. [Online]. Available: https://www.indeed.com/career-advice/career-development/non-functional-requirements-examples

[38] ‘Domain requirements. [Online]. Available: https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Requirements/DomainReq.html

[39] H. Ltd, ‘System development methodology: Research & Development: Hitachi’. [Online]. Available: https://www.hitachi.com/rd/glossary/s/system_development_methodology.html

[40] ‘What Is Agile Methodology in Project Management?’ [Online]. Available: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

[41] M. McCormick, ‘Waterfall vs. Agile Methodology’, p. 8.

[42] ‘How and what of Retrospectives in Agile Methodology’. [Online]. Available: https://www.wednesday.is/writing-articles/retrospectives-in-agile-methodology

[43] ‘Agile Model (Software Engineering) - javatpoint’. [Online]. Available: https://www.javatpoint.com/software-engineering-agile-model

[44] ‘OOPs | Object Oriented Design’, GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/oops-object-oriented-design/

[45] “Unit Testing - javatpoint” javatpoint. [Online]. Available: https://www.javatpoint.com/unit-testing

[46] “System Testing - Software Engineering” GeeksforGeeks. [Online]. Available: httpfuns://www.geeksforgeeks.org/system-testing/