**Modern Academy for Engineering & Technology**

**Computer Engineering Department**

Bachelor Thesis:

**Sign Language Interpreter**

Supervised By:

**DR. SABRY MOHAMED ABDELMOETTY**

|  |
| --- |
| **OMAR HANY** |

Designed By:

**PAGE INTENTIALLY LEFT BLANK**

**ACKNOWLEDGMENT**

We express our profound gratitude to Allah, the Most Gracious and Merciful, for His blessings and guidance that have led us to the successful completion of our project.

We extend heartfelt thanks to **Dr. Sabry Mohammed AbdElMoetty** for their constant guidance and support.

We also appreciate the assistance and facilities provided by **Dr. Khalid Morsy** and other faculty members.

We remember and honor the late **Dr. Said Gawish** for his contributions to our academic community.

To our friends, we are immensely grateful for their unwavering support. We acknowledge and appreciate the contributions of all who have guided and supported us on this journey.

Lastly, we offer our utmost appreciation and love to our parents, who have been our unwavering pillars of strength. We recognize and value their unwavering support, sacrifices, and continuous encouragement. Their prayers and guidance have been instrumental in our achievements, and we are forever grateful for their blessings.

With sincere appreciation and prayers,

[Team SLIEA]

**ABSTRACT**

In a world were expressing ideas and sharing thoughts is essential for societal progress, it is crucial to ensure that everyone, regardless of their abilities, has the opportunity to communicate effectively. Among the population, approximately 8 in 10,000 individuals face the challenge of being mute. However, with the rapid advancement of technology, we have seized the opportunity to aid these individuals.

Our graduation project revolves around the development of an innovative mobile application that serves as a communication bridge between the mute society and unimpaired individuals. Upon initial use, users are prompted to choose their communication role, either as a mute or an unimpaired person, tailoring the app's functionalities accordingly.

The core functionality of our app revolves around two main processes. Firstly, by leveraging advanced computer vision algorithms, unimpaired users can point their smartphone camera towards individuals using sign language. The app seamlessly translates these intricate gestures into spoken words, facilitating real-time communication. Conversely, the app also enables the translation of spoken language from unimpaired users into sign language, allowing for a two-way communication channel.

To achieve this, our project incorporates a comprehensive hardware solution. Specialized gloves, equipped with flexible sensors and an Arduino Nano microcontroller, capture, and interpret sign language gestures with high precision. These data are then processed and seamlessly integrated into the app, ensuring accurate translation and communication.

Through our technological innovation, we aim to empower speech-impaired individuals by providing them with an inclusive platform to express themselves and engage with others. By bridging the communication gap, we foster equal opportunities for all members of society. Our project not only demonstrates the power of technology but also emphasizes the importance of inclusivity and accessibility in our rapidly evolving world.

**Contents**

[INTRODUCTION 1](#_Toc138389112)

[Problem Statement 2](#_Toc138389113)

[Involved SDGS 3](#_Toc138389114)

[Project Objective 4](#_Toc138389115)

[Project Scope 4](#_Toc138389116)

[Development Approach 5](#_Toc138389117)

[Building Approach 5](#_Toc138389118)

[Chapter 1 (Introduction Chapter) 6](#_Toc138389119)

[Artificial Intelligence 7](#_Toc138389121)

[**What is Artificial Intelligence?** 8](#_Toc138389122)

[**Highlighting three applications of AI that illustrate different aspects of AI** 8](#_Toc138389123)

[**Philosophy of AI** 11](#_Toc138389124)

[**How Does AI Work?** 12](#_Toc138389125)

[**Why Is Artificial Intelligence Important?** 12](#_Toc138389126)

[**What are the Advantages and Disadvantages of Artificial Intelligence?** 13](#_Toc138389127)

[Machine Learning 14](#_Toc138389128)

[**What Is Machine Learning?** 14](#_Toc138389129)

[**How Does Machine Learning Work?** 14](#_Toc138389130)

[**Three Types of Machine Learning** 14](#_Toc138389131)

[**The Nearest Neighbor Classifier** 15](#_Toc138389132)

[**Machine Learning Is Widely Adopted** 15](#_Toc138389133)

[**The Future of Machine Learning; Hybrid AI** 16](#_Toc138389134)

[Android 17](#_Toc138389135)

[**Features of Android** 19](#_Toc138389136)

[**Android Versions** 20](#_Toc138389137)

[**Programming Languages used in Developing Android Applications** 22](#_Toc138389138)

[**Advantages of Android Development** 22](#_Toc138389139)

[**Disadvantages of Android Development** 22](#_Toc138389140)

[**Android Architecture** 23](#_Toc138389141)

[**Applications** 24](#_Toc138389142)

[**Application framework** 24](#_Toc138389143)

[**Application runtime** 24](#_Toc138389144)

[**Android UI Layouts** 25](#_Toc138389145)

[SLIEA’S Animation Using Computer Graphics 26](#_Toc138389146)

[**Introduction of Computer Graphics** 27](#_Toc138389147)

[**Definition of Computer Graphics** 27](#_Toc138389148)

[**Why are computer graphics used?** 28](#_Toc138389149)

[**What is the definition of animation?** 28](#_Toc138389150)

[**What is animation?** 28](#_Toc138389151)

[**Different Types of Animation** 29](#_Toc138389152)

[**Rigging and skeletal animation: what it is and how it works.** 32](#_Toc138389153)

[**What is character rigging?** 32](#_Toc138389154)

[**What is skeletal animation?** 32](#_Toc138389155)

[**How does rigging work?** 33](#_Toc138389156)

[**Rigging: pros and cons.** 34](#_Toc138389157)

[**Pros of rigging in animation.** 34](#_Toc138389158)

[**Cons of rigging in animation.** 35](#_Toc138389159)

[**What Does Rendering Mean?** 35](#_Toc138389160)

[**Rendering categories** 35](#_Toc138389161)

[**What’s the difference between animation and rigging?** 36](#_Toc138389162)

[SLIEA’S Gloves Using Embedded Systems 37](#_Toc138389163)

[**Printed Circuit Board** 38](#_Toc138389164)

[**PCB board** 38](#_Toc138389165)

[**PCB material** 38](#_Toc138389166)

[**PCB board usage** 38](#_Toc138389167)

[**Difference between PCB and circuit board** 38](#_Toc138389168)

[**Difference between Single layer and Double layer PCB** 39](#_Toc138389169)

[**SD Card Module** 39](#_Toc138389170)

[**Micro SD Card Module** 39](#_Toc138389171)

[**Components of the SD Card Module** 39](#_Toc138389172)

[**Usage of SD Card Module** 40](#_Toc138389173)

[**Flex Sensor** 40](#_Toc138389174)

[**What is flex sensor made of?** 40](#_Toc138389175)

[**Usage of Flex Sensors** 40](#_Toc138389176)

[**Working Principle** 40](#_Toc138389177)

[**Inertial measurement unit** 41](#_Toc138389178)

[**Purpose of IMU** 41](#_Toc138389179)

[**Arduino Nano** 41](#_Toc138389180)

[**Arduino Nano usage** 41](#_Toc138389181)

[**Arduino Nano Language** 41](#_Toc138389182)

[Chapter 2 (Implementation) 42](#_Toc138389183)

[Sign Language Interpreter Using Machine Learning Techniques 43](#_Toc138389185)

[**Data Collection and Preprocessing** 43](#_Toc138389186)

[**Why we Created our own Dataset?** 44](#_Toc138389187)

[**Model Training** 45](#_Toc138389188)

[**Model Evaluation and Performance Analysis** 46](#_Toc138389189)

[**Real-time Prediction** 46](#_Toc138389190)

[**Image Classification** 47](#_Toc138389191)

[**Conclusion** 47](#_Toc138389192)

[Animation Characters that Simulate Sign Gestures 48](#_Toc138389193)

[**Character Creation in Blender** 49](#_Toc138389194)

[**Character Rigging** 50](#_Toc138389195)

[**Rendering** 52](#_Toc138389196)

[**Montaging the videos** 53](#_Toc138389197)

[**Output Videos Sample** 53](#_Toc138389198)

[Firebase 54](#_Toc138389199)

[**Overview of Firebase** 54](#_Toc138389200)

[**Key Components of Firebase** 54](#_Toc138389201)

[**Benefits of Firebase** 56](#_Toc138389202)

[Mobile Application Implementation 57](#_Toc138389203)

[**Android Studio Setup** 57](#_Toc138389204)

[**Application Creation** 57](#_Toc138389205)

[**Design User Interface (UI)** 57](#_Toc138389206)

[**Development UI Logic** 58](#_Toc138389207)

[**Application Logic:** 58](#_Toc138389208)

[**Chaquopy Library** 58](#_Toc138389209)

[**Back End** 58](#_Toc138389210)

[**Firebase** 59](#_Toc138389211)

[**Testing the Application** 59](#_Toc138389212)

[**Conclusion** 60](#_Toc138389213)

[Chapter 3 (Proposed Solution) 61](#_Toc138389214)

[Software Solution (SLIEA) 62](#_Toc138389216)

[**Business Model** 63](#_Toc138389217)

[**Function Requirements** 67](#_Toc138389218)

[**Non-Function Requirements** 68](#_Toc138389219)

[**Performance and scalability** 68](#_Toc138389220)

[**Portability and compatibility** 68](#_Toc138389221)

[**Reliability, availability, maintainability** 68](#_Toc138389222)

[**Security** 68](#_Toc138389223)

[**Usability** 68](#_Toc138389224)

[**Environment** 68](#_Toc138389225)

[**Accuracy** 68](#_Toc138389226)

[**Use Case Description** 69](#_Toc138389227)

[**Use Case Diagram** 76](#_Toc138389228)

[**System Diagram** 77](#_Toc138389229)

[**Sequence Diagram** 78](#_Toc138389230)

[**Activity Diagram** 79](#_Toc138389231)

[**Software Tools and how to use it** 80](#_Toc138389232)

[**Machine Learning** 80](#_Toc138389233)

[**The Tools** 80](#_Toc138389234)

[**Its Usage** 81](#_Toc138389235)

[**Firebase** 82](#_Toc138389236)

[**Step 1: set up firebase project** 82](#_Toc138389237)

[**Step 2: Connect Firestore with Android Studio** 83](#_Toc138389238)

[**Step 3: Data Entry** 85](#_Toc138389239)

[**Android Studio** 86](#_Toc138389240)

[**FastAPI** 86](#_Toc138389241)

[**Steps to Use Android Studio** 86](#_Toc138389242)

[**Steps to Use FastAPI** 87](#_Toc138389243)

[**Layout Definition** 88](#_Toc138389244)

[**Splash Screen** 88](#_Toc138389245)

[**Onboard Screens** 89](#_Toc138389246)

[**Main Screens** 92](#_Toc138389247)

[94](#_Toc138389248)

[Hardware Solution (Gloves) 95](#_Toc138389249)

[**Arduino nano** 96](#_Toc138389250)

[**General Description** 96](#_Toc138389251)

[**Gloves Usage** 96](#_Toc138389252)

[**Inertial Measurement Unit** 97](#_Toc138389253)

[**General Description** 97](#_Toc138389254)

[**Gloves Usage** 97](#_Toc138389255)

[**Flex resistors** 98](#_Toc138389256)

[**General Description** 98](#_Toc138389257)

[**Gloves Usage** 98](#_Toc138389258)

[**SD Module** 99](#_Toc138389259)

[**General Description** 99](#_Toc138389260)

[**Gloves Usage** 99](#_Toc138389261)

[**Components Pin Description** 100](#_Toc138389262)

[**Connections** 101](#_Toc138389263)

[**Final Glove** 103](#_Toc138389264)

[Environment types 104](#_Toc138389265)

[**Software tools** 104](#_Toc138389266)

[**Hardware Tools** 104](#_Toc138389267)

[Chapter 4 (Testing) 105](#_Toc138389268)

[Test Objective 106](#_Toc138389270)

[Test Plan 107](#_Toc138389271)

[Test Scenarios 108](#_Toc138389272)

[Test Cases 110](#_Toc138389273)

[Test Execution 112](#_Toc138389274)

[Functional Testing 115](#_Toc138389275)

[**Communication Mode Testing** 115](#_Toc138389276)

[**Gesture Recognition Accuracy** 116](#_Toc138389277)

[**Translation Quality** 116](#_Toc138389278)

[**User Interface Responsiveness** 117](#_Toc138389279)

[**Error Handling** 117](#_Toc138389280)

[**Compatibility Testing** 117](#_Toc138389281)

[**Usability Testing** 117](#_Toc138389282)

[Performance Testing 118](#_Toc138389283)

[**Response Time Testing:** 118](#_Toc138389284)

[**Resource Utilization Testing:** 118](#_Toc138389285)

[**Load Testing:** 118](#_Toc138389286)

[**Network Conditions Testing:** 118](#_Toc138389287)

[**Scalability Testing:** 118](#_Toc138389288)

[**Stress Testing:** 119](#_Toc138389289)

[**Performance Monitoring:** 119](#_Toc138389290)

[Test Results and Analysis 120](#_Toc138389291)

[**Functional Testing Results:** 120](#_Toc138389292)

[**Performance Testing Results:** 120](#_Toc138389293)

[**Defects and Recommendations:** 120](#_Toc138389294)

[**Test Data Analysis:** 121](#_Toc138389295)

[Conclusion 121](#_Toc138389296)

[Chapter 5 (Conclusion and Future Work) 122](#_Toc138389297)

[Future Work 123](#_Toc138389299)

[Conclusion 125](#_Toc138389300)

[References 126](#_Toc138389301)

[References for the Application 126](#_Toc138389302)

[References for the Glove 126](#_Toc138389303)

**List of Figures**

[Figure 1: Good Health and Well-Being 3](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389326)

[Figure 2: Reduce inequality 3](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389327)

[Figure 3: Quality Education 3](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389328)

[Figure 4: illustration of software development lifecycle 5](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389329)

[Figure 5: Devices Runs on Android 18](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389330)

[Figure 6: Android Systems over Years 20](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389331)

[Figure 7: Main Components of Android Architecture 23](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389332)

[Figure 8: Android View Example 25](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389333)

[Figure 9: Android ViewGroup Example 25](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389334)

[Figure 10: Computer Graphics 27](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389335)

[Figure 11: Computer Graphics Classifications 28](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389336)

[Figure 12:PCB 38](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389337)

[Figure 13: PCB Material 38](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389338)

[Figure 14: PCB Comparison 39](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389339)

[Figure 15:SD Module 39](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389340)

[Figure 16:SD Module Components 39](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389341)

[Figure 17:Flex Sensor 40](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389342)

[Figure 18:IMU 41](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389343)

[Figure 19:Arduino Kit 41](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389344)

[Figure 20: Dataset Preprocessing 43](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389345)

[Figure 21:Arabic animation Character 48](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389346)

[Figure 22:English animation Character 48](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389347)

[Figure 23:Output English Example 53](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389348)

[Figure 24:Output Arabic Example 53](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389349)

[Figure 25:Business Model 63](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389350)

[Figure 26:Business Model Section1 64](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389351)

[Figure 27:Business Model Section2 65](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389352)

[Figure 28:Business Model Section3 66](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389353)

[Figure 29:Use Case Diagram 76](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389354)

[Figure 30:System Diagram 77](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389355)

[Figure 31:Sequence Diagram 78](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389356)

[Figure 32:Activity Diagram 79](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389357)

[Figure 33:Firebase Connection 83](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389358)

[Figure 34:Firebase Connection 84](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389359)

[Figure 35:Data Entry 85](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389360)

[Figure 36:Splash Screen 88](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389361)

[Figure 37:Onboard 1 89](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389362)

[Figure 38:Onboard 2 90](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389363)

[Figure 39:Onboard 3 91](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389364)

[Figure 40:Signify Screen 92](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389365)

[Figure 41:Transign Screen 93](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389366)

[Figure 42:Transign Beta Screen 94](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389367)

[Figure 43:Arduino nano 96](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389368)

[Figure 44:IMU 97](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389369)

[Figure 45:Flex Resistors 98](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389370)

[Figure 46:SD Module 99](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389371)

[Figure 47:Pin Description 100](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389372)

[Figure 48:IMU Connections with Arduino 101](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389373)

[Figure 49:Flex Resistors Connections with Arduino 101](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389374)

[Figure 50:SD Module Connections with Arduino 102](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389375)

[Figure 51:Glove After Mounting Components 103](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389376)

[Figure 52:Third Process Screen 115](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389377)

[Figure 53:Second Process Screen 115](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389378)

[Figure 54:First Process Screen 115](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389379)

[Figure 55:Testing Gestures 2 116](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389380)

[Figure 56:Testing Gestures 1 116](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389381)

[Figure 57:Testing Audible Voice 116](file:///G:\Sign%20Language%20Interpreter%20(Book)V_02.docx#_Toc138389382)

**List Of Tables**

[Table 1:Version Details of android 21](#_Toc138389407)

[Table 2:UC-01 69](#_Toc138389408)

[Table 3:UC-02 70](#_Toc138389409)

[Table 4:UC-03 71](#_Toc138389410)

[Table 5:UC-04 72](#_Toc138389411)

[Table 6:UC-05 73](#_Toc138389412)

[Table 7:UC-06 74](#_Toc138389413)

[Table 8:UC-07 75](#_Toc138389414)

# **INTRODUCTION**

Sign language is a form of manual communication and is one of the most natural ways of communication for most people in the deaf community.

The aim of sign language recognition is to provide an accurate and convenient mechanism to transcribe sign gestures into meaningful text or speech so that communication between deaf and hearing society can easily be made.

For an ordinary person to communicate with deaf people, an interpreter is usually needed to translate sign language into natural language and vice versa.

**Sign languages ​​unite us!** (A campaign was made by WFD)

Found that there are 70 million deaf people worldwide, according to the International Federation of the Deaf. 80% of those who are deaf live in developing countries and use more than 300 sign languages.

Our Project **Objectives** towards People of determination



Gaining recognition and respect for their own sign languages and cultures

Maintaining their organizations and activities



Achieving their rights across all areas of life, including equal opportunity and accessibility

# **Problem Statement**

The problem is that there is a significant communication barrier between individuals who use sign language and those who do not understand sign language. This barrier restricts effective communication and limits the opportunities for social interaction, education, and employment for individuals who rely on sign language as their primary means of communication. Current solutions for sign language translation are limited and often rely on human interpreters, which can be time-consuming, expensive, and not readily available in all situations.

The objective of this project is to develop a mobile application, SLIEA, that can accurately interpret and translate sign gestures to text or voice, and vice versa, in both English and Arabic languages. The application should be user-friendly, accessible, and capable of providing real-time translation to facilitate seamless communication between sign language users and non-sign language users. The application should also support different input methods, including voice input, image recognition, and video translation, to accommodate various communication scenarios.

By addressing this problem, SLIEA aims to empower individuals who use sign language by providing them with a reliable and efficient tool for communication, fostering inclusivity, and promoting equal opportunities for social interaction, education, and employment.

**Involved SDGS** (Sustainable Development Goals)

Figure 1: Good Health and Well-Being

Figure 2: Reduce inequality

Figure 3: Quality Education

# **Project Objective**

The objective of the project is to empower individuals with speech impairments by developing a comprehensive solution that facilitates effective communication between them and unimpaired individuals. The project aims to bridge the communication gap by leveraging technology, specifically through a mobile application and specialized hardware components. The goal is to enable real-time translation of sign language gestures into spoken words and vice versa, providing individuals with speech impairments an inclusive platform to express themselves, engage with others, and have equal opportunities for communication and interaction in society.

# **Project Scope**

The project scope involves developing a mobile application and integrating specialized hardware components to enable communication between individuals with speech impairments and unimpaired individuals. The mobile app will serve as the user interface, while hardware components like gloves with sensors and an Arduino Nano will capture and interpret sign language gestures.

The project will implement algorithms and software modules for gesture recognition, translation, and speech synthesis. Users can choose their communication mode, whether as a mute individual or an unimpaired person, using the mobile app. The app will translate sign language gestures captured by the camera into spoken words and vice versa, facilitating communication between users.

The scope also includes ensuring real-time translation accuracy, user-friendly interface design, and compatibility with various mobile devices. Testing and validation will be conducted to ensure the system's performance, usability, and reliability. The project focuses on integrating existing technologies rather than developing new sign language recognition or speech synthesis techniques, aiming to create a practical and functional communication solution.

# **Development Approach**

Describes the methodology that is used to develop the system and is the type is specified by the systems developer.

## **Building Approach**

The systems development follows the traditional approach of System Development Lifecycle (SDLC) as illustrated by (Dennis, 2009). The SDLC has four phases: planning, analysis, design, and implementation. Each phase is composed of a series of steps that rely on techniques that produce deliverables. The system development will move from one step to another consecutively or iteratively.

As demonstrated in the figure 1 below:

Figure 4: illustration of software development lifecycle

This approach has been chosen specifically for this system because of the outlined reasons listed below:

* The model is more flexible and less costly to change scope and the requirements.
* It is easy to manage risk because risky pieces are identified and handled during iteration.
* The customers / users respond to each build.
* It is easy to test and debug during a smaller iteration.

# **Chapter 1**

# **(Introduction Chapter)**

# 

## **Artificial Intelligence**

### **What is Artificial Intelligence?**

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. “Autonomy” and “Adaptivity” are properties that are characteristic to AI.

**Autonomy** is the ability to perform tasks in complex environments without constant guidance by a user.

**Adaptivity** is the ability to improve performance by learning from experience.

### **Highlighting three applications of AI that illustrate different aspects of AI**

**Application 1. Self-driving cars**

Self-driving cars require a combination of AI techniques of many kinds: search and planning to find the most convenient route from A to B, computer vision to identify obstacles, and decision making under uncertainty to cope with the complex and dynamic environment. Each of these must work with almost flawless precision in order to avoid accidents.

The same technologies are also used in other autonomous systems such as delivery robots, flying drones, and autonomous ships.

Implications: road safety should eventually improve as the reliability of the systems surpasses human level. The efficiency of logistics chains when moving goods should improve. Humans move into a supervisory role, keeping an eye on what’s going on while machines take care of the driving. Since transportation is such a crucial element in our daily life, it is likely that there are also some implications that we haven’t even thought about yet.

**Application 2. Content recommendation**

A lot of the information that we encounter in the course of a typical day is personalized. Examples include Facebook, Twitter, Instagram, and other social media content; online advertisements; music recommendations on Spotify; movie recommendations on Netflix, HBO, and other streaming services. Many online publishers such as newspapers’ and broadcasting companies’ websites as well as search engines such as Google also personalize the content they offer.

While the frontpage of the printed version of the New York Times or China Daily is the same for all readers, the frontpage of the online version is different for each user. The algorithms that determine the content that you see are based on AI.

Implications: while many companies don’t want to reveal the details of their algorithms, being aware of the basic principles helps you understand the potential implications: these involve so called filter bubbles, echo-chambers, troll factories, fake news, and new forms of propaganda.

**Application 3. Image and video processing**

Face recognition is already a commodity used in many customer, business, and government applications such as organizing your photos according to people, automatic tagging on social media, and passport control. Similar techniques can be used to recognize other cars and obstacles around an autonomous car, or to estimate wildlife populations, just to name a few examples.

AI can also be used to generate or alter visual content. Examples already in use today include style transfer, by which you can adapt your personal photos to look like they were painted by Vincent van Gogh, and computer-generated characters in motion pictures such as Avatar, the Lord of the Rings, and popular Pixar animations where the animated characters replicate gestures made by real human actors.

Implications: when such techniques advance and become more widely available, it will be easy to create natural looking fake videos of events that are impossible to distinguish from real footage. This challenges the notion that “seeing is believing”.

**In addition to AI, there are several other closely related topics. These include machine learning, data science, deep learning, and robotics.**

* **Machine learning** can be said to be a subfield of AI, which itself is a subfield of computer science (such categories are often somewhat imprecise, and some parts of machine learning could be equally well or better belong to statistics). Machine learning enables AI solutions that are adaptive. A concise definition can be given as follows: Systems that improve their performance in each task with more and more experience or data.
* **Deep learning** is a subfield of machine learning, which itself is a subfield of AI, which itself is a subfield of computer science. The “depth” of deep learning refers to the complexity of a mathematical model, and that the increased computing power of modern computers has allowed researchers to increase this complexity to reach levels that appear not only quantitatively but also qualitatively different from before. As you notice, science often involves several progressively more special subfields, subfields of subfields, and so on. This enables researchers to zoom into a particular topic so that it is possible to catch up with the ever-increasing amount of knowledge accrued over the years and produce new knowledge on the topic — or sometimes, correct earlier knowledge to be more accurate.
* **Data science** is a recent umbrella term that includes machine learning and statistics, certain aspects of computer science including algorithms, data storage, and web application development. Data science is also a practical discipline that requires understanding of the domain in which it is applied in, for example, business or science: its purpose, basic assumptions, and constraints. Data science solutions often involve at least a pinch of AI (but usually not as much as one would expect from the headlines).
* **Robotics** means building and programming robots so that they can operate in complex, real-world scenarios. In a way, robotics is the ultimate challenge of AI since it requires a combination of virtually all areas of AI. For example:
* Computer vision and speech recognition for sensing the environment.
* Natural language processing, information retrieval, and reasoning under uncertainty for processing instructions and predicting consequences of potential actions.
* Cognitive modeling and affective computing (systems that respond to expressions of human feelings or that mimic feelings) for interacting and working together with humans.

Many of the robotics-related AI problems are best approached by machine learning, which makes machine learning a central branch of AI for robotics.

### **Philosophy of AI**

The very nature of the term “artificial intelligence” brings up philosophical questions whether intelligent behavior implies or requires the existence of a mind, and to what extent is consciousness replicable as computation.

**The Turing tests**

Alan Turing (1912-1954) was an English mathematician and logician. He is rightfully considered to be the father of computer science. Turing was fascinated by intelligence and thinking, and the possibility of simulating them by machines. Turing’s most prominent contribution to AI is his imitation game, which later became known as the Turing test.

In the test, a human interrogator interacts with two players, A and B, by exchanging written messages (in a chat). If the interrogator cannot determine which player, A or B, is a computer and which is a human, the computer is said to pass the test. The argument is that if a computer is indistinguishable from a human in a general natural language conversation, then it must have reached human-level intelligence.

What Turing meant by the test is an entity is intelligent if it cannot be distinguished from another intelligent entity by observing its behavior. Turing just constrained the set of behaviors into discussion so that the interrogator can’t base her or his decision on appearances.

**The Chinese room argument**

The idea that intelligence is the same as intelligent behavior has been challenged by some. The best-known counterargument is John Searle’s Chinese Room thought experiment. Searle describes an experiment where a person who doesn’t know Chinese is locked in a room. Outside the room is a person who can slip notes written in Chinese inside the room through a mail slot. The person inside the room is given a big manual where she can find detailed instructions for responding to the notes she receives from the outside.

Searle argued that even if the person outside the room gets the impression that he is in a conversation with another Chinese-speaking person, the person inside the room does not understand Chinese. Likewise, his argument continues, even if a machine behaves in an intelligent manner, for example, by passing the Turing test, it doesn’t follow that it is intelligent or that it has a “mind” in the way that a human has. The word “intelligent” can also be replaced by the word “conscious” and a similar argument can be made.

**Is a self-driving car intelligent?**

The Chinese Room argument goes against the notion that intelligence can be broken down into small mechanical instructions that can be automated.

A self-driving car is an example of an element of intelligence (driving a car) that can be automated. The Chinese Room argument suggests that this, however, isn’t intelligent thinking: it just looks like it. Going back to the above discussion on “suitcase words”, the AI system in the car doesn’t see or understand its environment, and it doesn’t know how to drive safely, in the way a human being sees, understands, and knows. According to Searle this means that the intelligent behavior of the system is fundamentally different from being intelligent.

### **How Does AI Work?**

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce lifelike exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning, and self-correction.

### **Why Is Artificial Intelligence Important?**

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

This has helped fuel an explosion in efficiency and opened the door to entirely new business opportunities for some larger enterprises. Prior to the current wave of AI, it would have been hard to imagine using computer software to connect riders to taxis, but today Uber has become one of the largest companies in the world by doing just that. It utilizes sophisticated machine learning algorithms to predict when people are likely to need rides in certain areas, which helps proactively get drivers on the road before they're needed. As another example, Google has become one of the largest players for a range of online services by using machine learning to understand how people use their services and then improving them. In 2017, the company's CEO, Sundar Pichai, pronounced that Google would operate as an "AI first" company.

Today's largest and most successful enterprises have used AI to improve their operations and gain advantage on their competitors.

### **What are the Advantages and Disadvantages of Artificial Intelligence?**

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

While the huge volume of data being created daily would bury a human researcher, AI applications that use machine learning can take that data and quickly turn it into actionable information. As of this writing, the primary disadvantage of using AI is that it is expensive to process the large amounts of data that AI programming requires.

**Advantages**

* Good at detail-oriented jobs
* Reduced time for data-heavy tasks
* Delivers consistent results.
* AI-powered virtual agents are always available.

**Disadvantages**

* Expensive
* Requires deep technical expertise.
* Limited supply of qualified workers to build AI tools.
* Only knows what it's been shown.
* Lack of ability to generalize from one task to another.

## **Machine Learning**

### **What Is Machine Learning?**

The robot-depicted world of our not-so-distant future relies heavily on our ability to deploy artificial intelligence (AI) successfully. However, transforming machines into thinking devices is not as easy as it may seem. Strong AI can only be achieved with machine learning (ML) to help machines understand as humans do.

Machine learning can be confusing, so it is important that we begin by clearly defining the term:

Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves.

### **How Does Machine Learning Work?**

Like how the human brain gains knowledge and understanding, machine learning relies on input, such as training data or knowledge graphs, to understand entities, domains, and the connections between them. With entities defined, deep learning can begin.

The machine learning process begins with observations or data, such as examples, direct experience or instruction. It looks for patterns in data so it can later make inferences based on the examples provided. The primary aim of ML is to allow computers to learn autonomously without human intervention or assistance and adjust actions accordingly.

### **Three Types of Machine Learning**

The roots of machine learning are in statistics, which can also be thought of as the art of **extracting knowledge from data**. Methods such as linear regression and Bayesian statistics, which are both already more than two centuries old (!), are even today at the heart of machine learning.

The area of machine learning is often divided into subareas according to the kinds of problems being attacked. A rough categorization is as follows:

**Supervised learning:** We are given an input, for example a photograph with a traffic sign, and the task is to predict the correct output or label, for example which traffic sign is in the picture (speed limit, stop sign, etc.). In the simplest cases, the answers are in the form of yes/no (we call these *binary classification problems*).

**Unsupervised learning:** There are no labels or correct outputs. The task is to discover the structure of the data: for example, grouping similar items to form “clusters”, or reducing the data to a small number of important “dimensions”. Data visualization can also be considered unsupervised learning.

**Reinforcement learning:** Commonly used in situations where an AI agent like a self-driving car must operate in an environment and where feedback about good or bad choices is available with some delay. Also used in games where the outcome may be decided only at the end of the game.

### **The Nearest Neighbor Classifier**

The nearest neighbor classifier is among the simplest possible classifiers. When given an item to classify, it finds the training data item that is most like the new item, and outputs its label. A typical example of an application of the nearest neighbor method is predicting user behavior in AI applications such as recommendation systems. The idea is to use the very simple principle that users with similar past behavior tend to have similar future behavior.

### **Machine Learning Is Widely Adopted**

Machine learning is not science fiction. It is already widely used by businesses across all sectors to advance innovation and increase process efficiency. In 2021, 41% of companies accelerated their rollout of AI because of the pandemic. These newcomers are joining the 31% of companies that already have AI in production or are actively piloting AI technologies.

* **Data security**: Machine learning models can identify data security vulnerabilities before they can turn into breaches. By looking at past experiences, machine learning models can predict future high-risk activities so risk can be proactively mitigated.
* **Finance**: Banks, trading brokerages and fintech firms use machine learning algorithms to automate trading and to provide financial advisory services to investors. Bank of America is using a chatbot, Erica, to automate customer support.
* **Healthcare**: ML is used to analyze massive healthcare data sets to accelerate discovery of treatments and cures, improve patient outcomes, and automate routine processes to prevent human error. For example, IBM’s Watson uses data mining to provide physicians with data they can use to personalize patient treatment.
* **Fraud detection**: AI is being used in the financial and banking sector to autonomously analyze large numbers of transactions to uncover fraudulent activity in real time. Technology services firm Capgemini claims that fraud detection systems using machine learning and analytics minimize fraud investigation time by 70% and improve detection accuracy by 90%.
* **Retail**: AI researchers and developers are using ML algorithms to develop AI recommendation engines that offer relevant product suggestions based on buyers’ past choices, as well as historical, geographic, and demographic data.

### **The Future of Machine Learning; Hybrid AI**

For all its shortcomings, machine learning is still critical to the success of AI. This success, however, will be contingent upon another approach to AI that counters its weaknesses, like the “black box” issue that occurs when machines learn unsupervised. That approach is symbolic AI, or a rule-based methodology toward processing data. A symbolic approach uses a knowledge graph, which is an open box, to define concepts and semantic relationships.

Together, ML and symbolic AI form hybrid AI, an approach that helps AI understand language, not just data. With more insight into what was learned and why, this powerful approach is transforming how data is used across the enterprise.

# 

## **Android**

Android operating system is the largest installed base among various mobile platforms across the globe. Hundreds of millions of mobile devices are powered by Android in more than 190 countries of the world. It conquered around 71% of the global market share by the end of 2021, and this trend is growing bigger every other day. The company named Open Handset Alliance developed Android for the first time that is based on the modified version of the Linux kernel and other open-source software. Google sponsored the project in the initial stages and in the year 2005, it acquired the whole company. In September 2008, the first Android-powered device was launched on the market. Android dominates the mobile OS industry because of the long list of features it provides. It’s user-friendly, has huge community support, provides a greater extent of customization, and many companies build Android-compatible smartphones. As a result, the market observes a sharp increase in the demand for developing Android mobile applications, and with that companies need smart developers with the right skill set. At first, the purpose of Android was thought of as a mobile operating system. However, with the advancement of code libraries and its popularity among developers of the divergent domain, Android has become an absolute set of software for all devices like tablets, wearables, set-top boxes, smart TVs, notebooks, etc.

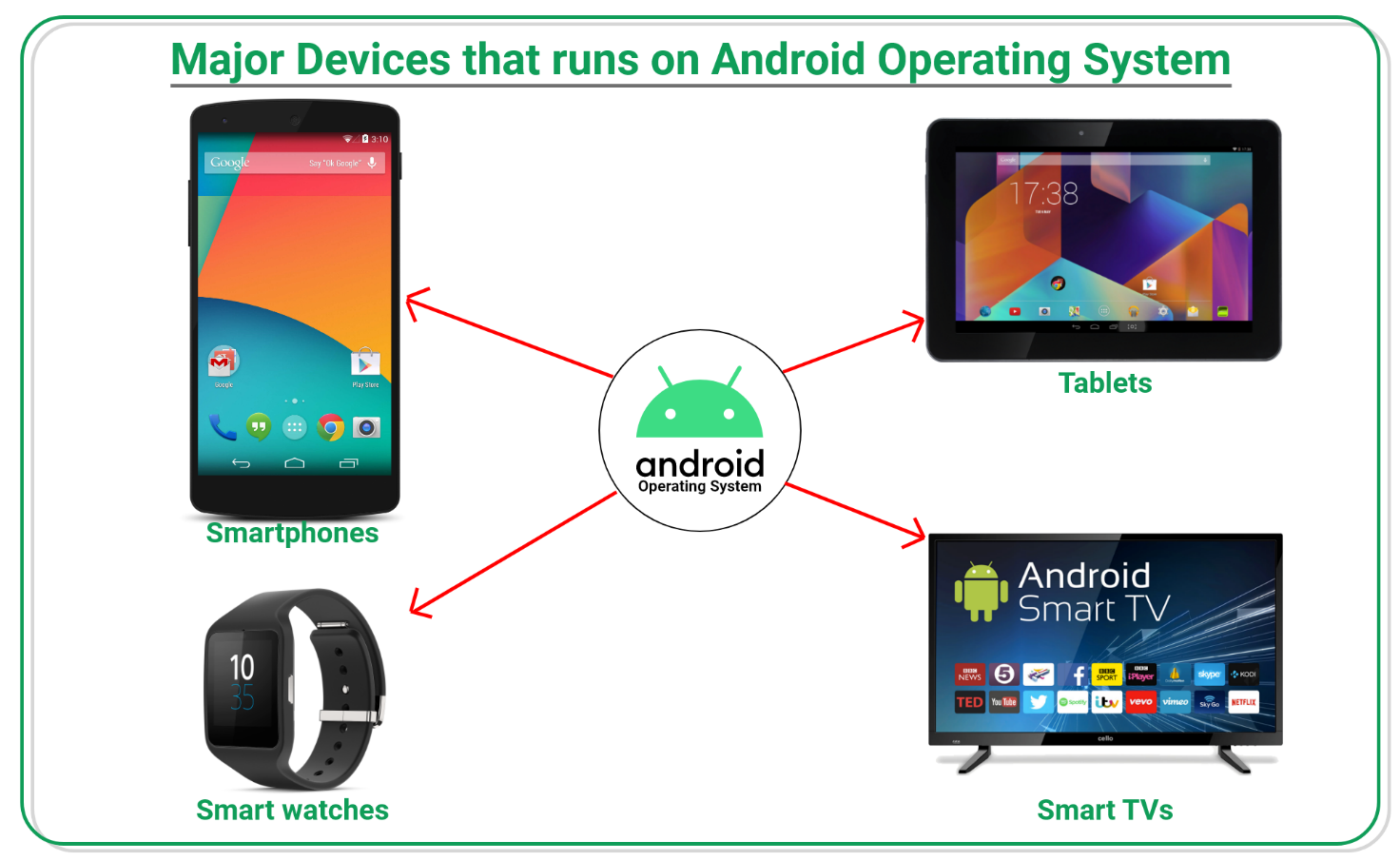
****

Figure 5: Devices Runs on Android

### **Features of Android**

Android is a powerful open-source operating system that open-source provides immense features and some of these are listed below.

* Android Open-Source Project so we can customize the OS based on our requirements.
* Android supports different types of connectivity for GSM, CDMA, Wi-Fi, Bluetooth, etc. for telephonic conversation or data transfer.
* Using Wi-Fi technology, we can pair with other devices while playing games or using other applications.
* It contains multiple APIs to support location-tracking services such as GPS.
* We can manage all data storage-related activities by using file manager.
* It contains a wide range of media support like AVI, MKV, FLV, MPEG4, etc. to play or record a variety of audio/video.
* It also supports different image formats like JPEG, PNG, GIF, BMP, MP3, etc.
* It supports multimedia hardware control to perform playback or recording using a camera and microphone.
* Android has an integrated open-source Web Kit layout-based web browser to support User Interfaces like HTML5, and CSS3.
* Android supports multi-tasking means we can run multiple applications at a time and can switch between them.
* It provides support for virtual reality or 2D/3D Graphics.

### **Android Versions**

Google first publicly announced Android in November 2007 but was released on 23 SEPTEMBER 2008 to be exact. The first device to bring Android onto the market was the HTC Dream with the version Android 1.0. Since then, Google released a lot of android versions such.

as Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, KitKat, Lollipop, marshmallow, Nougat, Oreo, etc. with extra functionalities and new features.

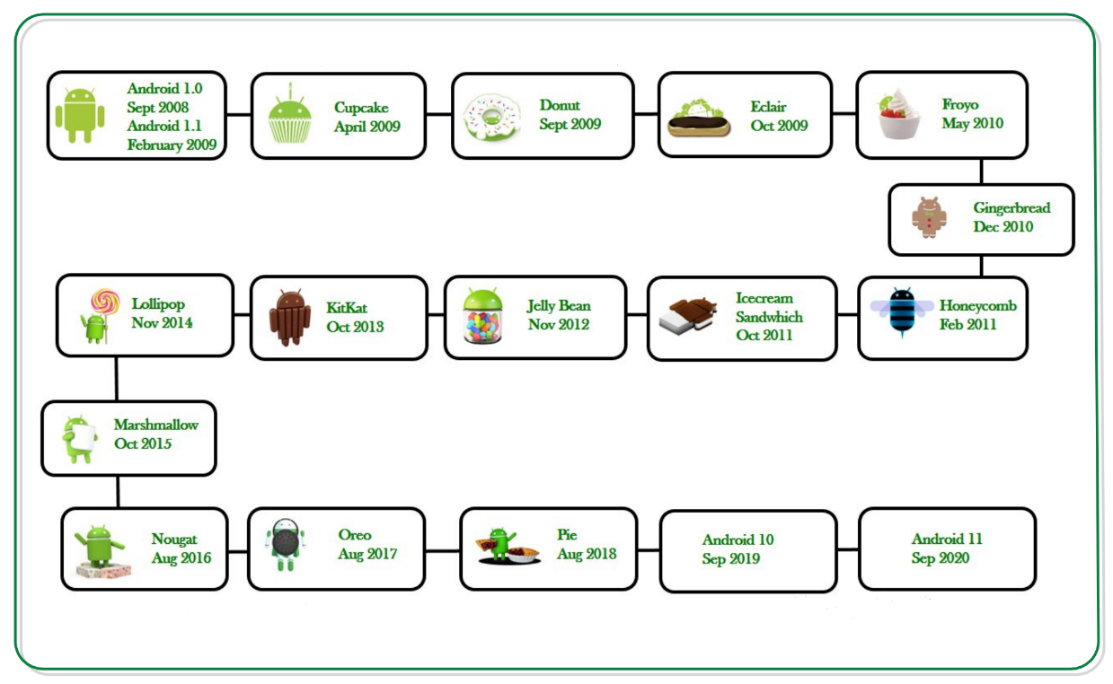


Figure 6: Android Systems over Years

The following table shows the version details of android which is released by Google from 2007 to date.

|  |  |  |  |
| --- | --- | --- | --- |
| Code Name | Version | API level | Release date |
| – | Android 1.0 | 1 | September 23, 2008 |
| – | Android 1.1 | 2 | February 9, 2009 |
| Cupcake | Android 1.5 | 3 | April 30, 2009 |
| Donut | Android 1.6 | 4 | September 15, 2009 |
| Eclair | Android 2.0 – 2.1 | 5-7 | October 26, 2009 |
| Froyo | Android 2.2 – 2.2.3 | 8 | May 20, 2010 |
| Gingerbread | Android 2.3 – 2.3.4 | 9-10 | December 6, 2010 |
| Honeycomb | Android 3.0.x – 3.2.x | 11 – 13 | February 22, 2011 |
| Ice-cream Sandwich | Android 4.0 – 4.0.4 | 14 – 15 | October 18, 2011 |
| Jellybean | Android 4.1 – 4.1.2 | 16 – 18 | July 9, 2012 |
| KitKat | Android 4.4 – 4.4.4 | 19 | July 9, 2012 |
| Lollipop | Android 5.0 – 5.1 | 21 – 22 | October 17, 2014 |
| Marshmallow | Android 6.0 – 6.0.1 | 23 | October 5, 2015 |
| Nougat | Android 7.0 – 7.1 | 24 – 25 | August 22, 2016 |
| Oreo | Android 8.0 | 26 | August 21, 2017 |
| Pie | Android 9.0 | 27 | August 6, 2018 |
| Android Q | Android 10.0 | 29 | September 3, 2019 |
| Android 11 | Android 11.0 | 30 | September 8, 2020 |
| Snow Cone | Android 12.0 – 12.1 | 31-32 | October 4, 2021 |
| Tiramisu | Android 13 |  | UPCOMING |

Table 1:Version Details of android

### **Programming Languages used in Developing Android Applications**

1-Java

2-Kotlin

Developing the Android Application using Kotlin is preferred by Google, as Kotlin is made an official language for Android Development, which is developed and maintained by JetBrains. Previously Java is considered the official language for Android Development. Kotlin is made official for Android Development in Google I/O 2017.

### **Advantages of Android Development**

* The Android is an open-source Operating system and hence possesses a vast community for support.
* The design of the Android Application has guidelines from Google, which makes it easier for developers to produce more intuitive user applications.
* Fragmentation gives more power to Android Applications. This means the application can run two activities on a single screen.
* Releasing the Android application in the Google play store is easier when it is compared to other platforms.

### **Disadvantages of Android Development**

* Fragmentation provides a very intuitive approach to user experience, but it has some drawbacks, where the development team needs time to adjust to the various screen sizes of mobile smartphones that are now available in the market and invoke the features in the application.
* The Android devices might vary broadly. So, the testing of the application becomes more difficult.
* As the development and testing consume more time, the cost of the application may increase, depending on the application’s complexity and features.

### **Android Architecture**

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework service.

Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following: -

* Applications
* Application Framework
* Android Runtime
* Platform Libraries
* Linux Kernel

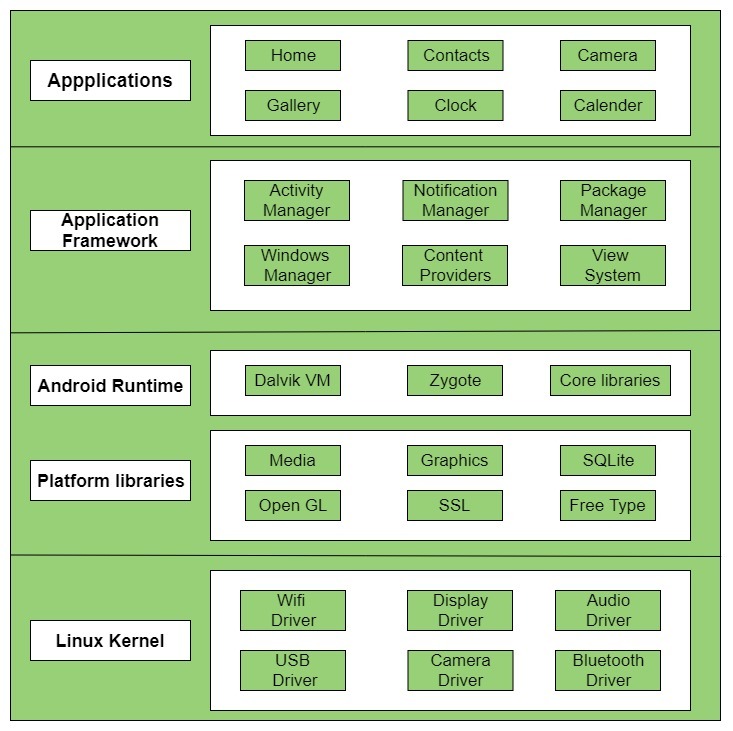
Pictorial representation of android architecture with several main components and their subcomponents

Figure 7: Main Components of Android Architecture

### **Applications**

Applications are the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc. and third-party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.

It runs within the Android run time with the help of the classes and services provided by the application framework.

### **Application framework**

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

### **Application runtime**

Android Runtime environment is one of the most important parts of Android. It contains components like core libraries and the Dalvik virtual machine (DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

### **Android UI Layouts**

Android Layout is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. Generally, every application is a combination of View and ViewGroup. As we know, an android application contains many activities, and we can say each activity is one page of the application. So, each activity contains multiple user interface components, and those components are the instances of the View and ViewGroup. All the elements in a layout are built using a hierarchy of **View** and **ViewGroup** objects.

**View:** A View is defined as the user interface which is used to create interactive UI components such as Text View, Image View, Edit Text, Radio Button, etc., and is responsible for event handling and drawing. They are Generally Called Widgets**.**

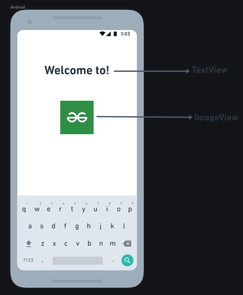


Figure 8: Android View Example

**View Group:** A View Group acts as a base class for layouts and layouts parameters that hold other Views or View Groups and to define the layout properties. They are Generally Called layouts.

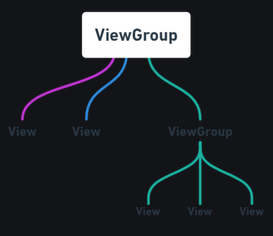


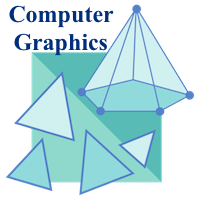
Figure 9: Android ViewGroup Example

The Android framework will allow us to use UI elements or widgets in two ways:

* Use UI elements in the XML file.
* Create elements in the Kotlin file dynamically.

### 

## **SLIEA’S Animation Using Computer Graphics**



It is difficult to display an image of any size on the computer screen. This method is simplified by using Computer graphics. Graphics on the computer are produced by using various algorithms and techniques. This tutorial describes how a rich visual experience is provided to the user by explaining how all these are processed by the computer.

### **Introduction of Computer Graphics**

Figure 10: Computer Graphics

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics is insensible. In today’s life, computer graphics have now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The product of computer graphics is a picture, it may be a business graph, drawing, or engineering.

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithms have been developed for improving the speed of picture generation with the passing of time. It includes the creation storage of models and image of objects. These models are for various fields like engineering, mathematics and so on.

Today’s computer graphics are entirely different from the earlier ones. It is not possible. It is an interactive user who can control the structure of an object of various input devices.

### **Definition of Computer Graphics**

It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, and represent pictures.

### **Why are computer graphics used?**

Suppose a shoe manufacturing company want to show the sale of shoes for five years. For this vast amount of information is to store. So, a lot of time and memory will be needed. This method will be tough to understand by a common man. In this situation graphics are a better alternative. Graphics tools are charts and graphs. Using graphs, data can be represented in pictorial form. A picture can be understood easily just with a single look.

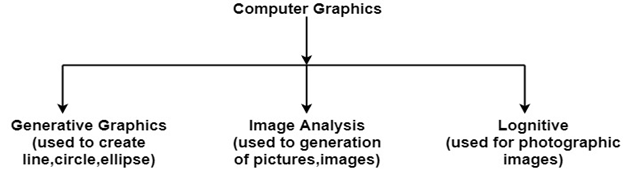
Interactive computer graphics work using the concept of two-way communication between computer users. The computer will receive signals from the input device, and the picture is modified accordingly. Picture will be changed quickly when we apply command.

Figure 11: Computer Graphics Classifications

### **What is the definition of animation?**

The simulation of movement created by a series of pictures is animation. But how it works is a bit more complicated than that. Before we get to the various types of animated motion pictures, let's start with an animation definition.

### **What is animation?**

**Animation**is a method of photographing successive drawings, models, or even puppets, to create an illusion of movement in a sequence. Because our eyes can only retain an image for approx. 1/10 of a second, when multiple images appear in fast succession, the brain blends them into a single moving image. In traditional animation, pictures are drawn or painted on transparent celluloid sheets to be photographed. Early cartoons are examples of this, but today, most animated movies are made with computer-generated imagery or CGI.

To create the appearance of smooth motion from these drawn, painted, or computer-generated images, frame rate, or the number of consecutive images that are displayed each second, is considered. Moving characters are usually shot “on twos” which just means one image is shown for two frames, totaling in at 12 drawings per second. 12 frames per second allows for motion but may look choppy. In the film, a frame rate of 24 frames per second is often used for smooth motion.

### **Different Types of Animation**

* Traditional Animation
* 2D Animation (Vector-based)
* 3D Animation
* Motion Graphics
* Stop Motion

**Traditional**

This is one of the oldest forms of animation in film. It’s sometimes called cel animation. As mentioned above, in traditional animation, objects are drawn on celluloid transparent paper. To create the animated sequence, the animator must draw every frame. It’s the same mechanism as a flipbook, just on a grander scale.

Traditional is most often 2D animation. [*Aladdin*](https://www.studiobinder.com/tag/aladdin/)*,*[*The Lion King*](https://www.studiobinder.com/tag/the-lion-king/), and other earlier cartoons are the best examples of this. \*Though, not all 2D is traditional. But we'll get to that in a minute.

In earlier years, the animator would draw on a table that had a light inside of it, so the creator could see his or her previous animation. While the traditional style is not nearly as prevalent today, drawings are generally done on tablets. And manual coloring hasn’t been used by Disney since [*The Little Mermaid*](https://www.studiobinder.com/tag/the-little-mermaid/) in 1989.

**2D (Vector)**

2D animation can fall under traditional animation like most early Disney movies — *Pinocchio*, [*Beauty and the Beast*](https://www.studiobinder.com/tag/beauty-and-the-beast-1991/), etc. But there is something called Vector-based animation that can be 2D without being traditional.

With Vector-based, the motion here can be controlled by *vectors*rather than *pixels*. So, what the heck does that mean?

Images with familiar formats like JPG, GIF, BMP, are pixel images. These images cannot be enlarged or shrunk without affecting image quality. Vector graphics don’t need to worry about resolution. Vectors are characterized by pathways with various start and end points, lines connecting these points to build the graphic. Shapes can be created to form a character or other image.

Vector-based animation uses mathematical values to resize images, so motion is smooth. They can re-use these creations, so the animator doesn’t need to keep drawing the same characters repeatedly. You can move around these vectors and animate that way.

This is also helpful for animators who aren’t the best drawers. Yes, these people exist.

**3D**

Today, 3D or computer animation is the most common type. But just because computers have stepped in instead of actual drawings, it’s not necessarily easier. The computer is just another tool, and 3D animation is still a long, intense process.

In 3D animated movies, the animator uses a program to move the character’s body parts around. They set their digital frames when all the parts of the character are in the right position. They do this for each frame, and the computer calculates the motion from each frame.

Animators adjust and tweak the curvatures and movements their characters make throughout. From Toy Story in 1995 to today’s Coco, 3D animation has become the dominant style in animated films.

3D animation is also unique in that, unlike 2D or other traditional methods, the character’s entire body is always visible. If a character turns to the side, the animator only needs to draw the side profile in 2D animation, but in 3D, the entire body still needs to be visible. So again, even though computers are being used, new technology comes with way more consideration.

Whether you’re using drawing in 2D or computing in 3D, animators and filmmakers alike look to storyboards to plan out each frame. Unlike live- action, animation movies can’t rely on camera tricks in a shot. Storyboards are the lifeline for creating animation.

**Motion Graphics**

Motion Graphics are pieces are digital graphics that create the illusion of motion usually for ads, title sequences in films, but ultimately exist to communicate something to the viewer. They’re often combined with sound for multimedia projects.

They’re a type of animation used mostly in business, usually with text as a main player.

**Stop Motion**

Stop motion encompasses Claymation, pixelation, object-motion, cutout animation, and more. But the basic mechanics are like the traditional style like a flipbook. However, instead of drawings, stop motion adjusts physical objects in each frame.

If moved in small increments, captured one frame at a time, the illusion of motion is produced. Whether puppets, clay, or even real people, these manual adjustments can make it a long, arduous process. Wallace and Gromit, Chicken Run, and The Nightmare Before Christmas are all great examples of stop motion films.

Stop motion is an older form of animated storytelling, especially compared to 3D computer animation. But the process of animating pictures dates back way before Disney or Pixar.

### **Rigging and skeletal animation: what it is and how it works.**

Skeletal animation and rigging are two terms that generally describe the same thing - and bring the characters you create through computer animation to life. They are a technique for manipulating 2D and 3D models to add movement - whether you’re making a film, advert, game, or project.

### **What is character rigging?**

Character rigging is a technique used in skeletal animation where you add control to a model. It defines the range of movement for a character or object to define its actions, gestures, and movement. A series of interconnected bones are applied to the 3D model so you can move and manipulate the static image in your desired manner.

### **What is skeletal animation?**

Skeletal animation refers to the way in which the character or object is represented in two parts:

* **Skin or mesh**. The surface representation is used to design a character.
* **Skeleton or rig**. A series of hierarchical, interconnected parts.

The term “rigging in animation” comes from the rig being the skeleton structure built to identify the virtual bones that allow the model to move. An easy way to envision it is like creating a virtual puppet.

You can use animation rigging for both 2D and 3D animations, which follow a similar path with a few small differences.

* **2D animation rigging**. With 2D animation rigging you simply draw the character or object in 2D and then apply the desired bone structure to it. This will be simpler than a 3D model and restricted in movement.
* **3D animation rigging**. A more common practice is 3D animation rigging. Here the full model can be rigged and posed within your scene, with the ability to create a much wider range of movement, whether animating a character or object.

### **How does rigging work?**

In animation, rigging works by construction a series of bones (or skeleton) for your 2D or 3D model. This could be an animal or human character, but you can also apply it to an object you want to animate — such as an Aeroplan or piece of fruit.

Using digital or computer animation software you can then adapt the position, scale and rotation of specific points of the skeleton (or rig) to introduce movement. Recording these changes through keyframing helps create your animation. Depending on the size, scale and complexity of your project, rigging can take anywhere from a few hours to a few days or longer.

Rigging is only a small part of the animation process. It’s one of the first steps you need to take before you can make your characters move. There are other animation techniques, but 3D skeletal animation is a popular method for both amateur animators and professionals working on Hollywood blockbusters, video games, TV shows, adverts and more.

The general step-by-step process for animation character rigging is as follows:

1. Create a 2D or 3D model.
2. Construct a skeleton for your character or object, whether replicating human bones or adding to an object.
3. Determine a weight scale, which controls how much influence the movement of one bone has over a specific part of the mesh. Some software has an automatic option, or you can finetune it yourself for accurate results.
4. Transform these bones using digital animation software so you can change their position, scale and rotation.
5. Add inverse kinematics to the bones where required. This is mainly for arms, legs and tails to ensure elbows and knees point the right way for a realistic movement.
6. Apply any restrictions to certain bones where required, so they only move in one direction for smooth and realistic results.
7. Record these bone movements along a timeline (keyframing)

### **Rigging: pros and cons.**

2D and 3D animation rigging aren’t the only options available to animators. To help decide if it’s the right technique for you and your project, consider the pros and cons of animation and rigging.

### **Pros of rigging in animation.**

A few advantages of rigging as part of the animation process include:

* **Independent and hierarchical movement**. Bones can be moved on their own to create the desired effect for your character or object. Any related bones will also move through the hierarchical structure in the rig.
* **Define animation in simple bone movements**. When using a polygonal mesh, animators must define their animation vertex by vertex. With rigging you can do this by simple bone movements.
* **Apply constraints**. Create realistic motions by applying constraints to specific bones, whether for a character or object. This adds realism to your results, which can be saved for replicating in the future.
* **Animators focus on large scale motion**. Thanks to the hierarchical structure, animators don’t need to think about each little movement as relevant parts should move in tandem. This saves them time and means they can dedicate their efforts to the larger animation.
* **Accurate weight painting**. Add further realism to how the mesh and rig work together with weight painting. Adjusting the scale helps form an accurate weight distribution for your results.

### **Cons of rigging in animation.**

Although there are various advantages of 3D and 2D animation rigging, there are two main drawbacks that make it less useful for some projects, such as:

* **Bones only represent vertexes**. Each bone within the rig only represents a set of vertexes or another specifically defined object. This means it cannot accurately represent the full complexity of the human body’s movements - or anything more abstract or conceptual.
* **Unrealistic muscle movement and skin motio**n. While the mesh and reg work together, it doesn’t provide realistic muscle or skin movement. This can be achieved but only through using special deformers, muscle controllers or introducing other secondary features.

### **What Does Rendering Mean?**

Rendering is the process involved in the generation of a two-dimensional or three-dimensional image from a model by means of application programs. Rendering is mostly used in architectural designs, video games, and animated movies, simulators, TV special effects and design visualization. The techniques and features used vary according to the project. Rendering helps increase efficiency and reduce costs in design.

### **Rendering categories**

There are two categories of rendering: pre-rendering and real-time rendering. The striking difference between the two lies in the speed at which the computation and finalization of images takes place.

* **Real-Time Rendering:** The prominent rendering technique used in interactive graphics and gaming where images must be created at a rapid pace. Because user interaction is high in such environments, real-time image creation is required. Dedicated graphics hardware and pre-compiling of the available information has improved the performance of real-time rendering.
* **Pre-Rendering:** This rendering technique is used in environments where speed is not a concern, and the image calculations are performed using multi-core central processing units rather than dedicated graphics hardware. This rendering technique is mostly used in animation and visual effects, where photorealism needs to be at the highest standard possible.

For these rendering types, the three major computational techniques used are:

* Scanline
* Raytracing
* Radiosity

### **What’s the difference between animation and rigging?**

Rigging is just one part of the wider animation process. The creation of a character is normally the first section, followed by the rigging of it. Then after a rig is applied the character or model can be edited and animated. The main difference is that rigging is a process within the larger animation project.

## **SLIEA’S Gloves Using Embedded Systems**

### **Printed Circuit Board**

#### **What is Printed Circuit Board and Designing Process of PCB?PCB board**

Figure 12:PCB

A printed circuit board, or PC board, or PCB, is a non-conductive material with conductive lines printed or etched. Electronic components are mounted on the board and the traces connect the components together to form a working circuit or assembly.

#### **PCB material**

Printed circuit boards, otherwise known as PCBs, are the electronic boards that are used in most electronic devices, including phones, household appliances and pieces of medical equipment. Typically, PCBs are made from non-substrate materials with layers of copper circuitry.

Figure 13: PCB Material

#### **PCB board usage**

Printed circuits boards, also known as PCBs, are the circuit boards used in most electronic devices. The boards both physically and mechanically support the device alongside connecting the electronic components. PCBs are typically made from non-substrate materials with layers of copper circuitry.

#### **Difference between PCB and circuit board**

A PCB is a blank circuit board with no electronic components attached, while a PCBA is a completed assembly that contains all the components required for the board to function as needed for the desired application. A PCB is not yet functional, while a PCBA is ready to be used in an electronic device.

#### **Difference between Single layer and Double layer PCB**

A picture containing text, screenshot, font, number

Description automatically generatedThe difference between double-sided and single-sided PCBs is that instead of using a single-sided copper core, the manufacture will start a core with copper on both sides. During production, they also drill holes called vias that they can plate or fill with a conductive, or nonconductive material.

Figure 14: PCB Comparison

### **SD Card Module**

#### **Micro SD Card ModuleMicro SD Card Module**

The micro- SD Card Module is a simple solution for transferring data to and from a standard SD card. The pin out is directly compatible with Arduino but can also be used with other microcontrollers. It allows you to add mass storage and data logging to your project.

#### **Micro SD Card Adapter Module - Buy SD Card Module at QuartzComponents.comComponents of the SD Card Module**

Figure 15:SD Module

Figure 16:SD Module Components

The micro-SD card module contains two main components i.e., Voltage Regulator and Level Shifter. The on-board ultra-low dropout regulator is used to convert 5V to 3.3V and Level Shifter is used to convert the interface logic from 3.3V-5V to 3.3V.

#### **Usage of SD Card Module**

The SD card module is especially useful for projects that require data logging. Arduino can create a file in an SD card to write and save data using the SD library. There are different models from different suppliers, but they all work in a similar way, using the SPI communication protocol.

### **Flex Sensor**

A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface.

Figure 17:Flex Sensor

#### **What is flex sensor made of?**

It is thin copper cladding on a plastic material substrate like acetate. The material we are using is single sided copper. Copper on one side and the substrate (plastic) on the other, the copper cladding material is cut into two pieces ¼" wide x 4.5" long strips. The material is easily cut with some scissors.

#### **Usage of Flex Sensors**

A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface.

#### **Working Principle**

This sensor works on the bending strip principle which means whenever the strip is twisted then its resistance will be changed. This can be measured with the help of any controller. This sensor works like a variable resistance because when it twists then the resistance will be changed.

### **Inertial measurement unit**

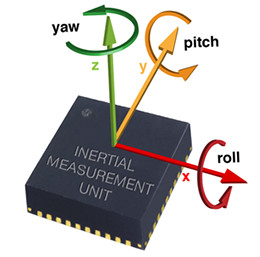
An inertial measurement unit is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers. When the magnetometer is included, IMUs are referred to as IMMUs.

Figure 18:IMU

#### **Purpose of IMU**

An inertial measurement unit (IMU) is a sensor that tracks the acceleration and angular velocity of an object over a period. On occasion, an IMU will also track the Earth's magnetic field and air pressure. In the artificial intelligence era, this cheap and reliable sensor can provide a lot of data.

### **Arduino Nano**

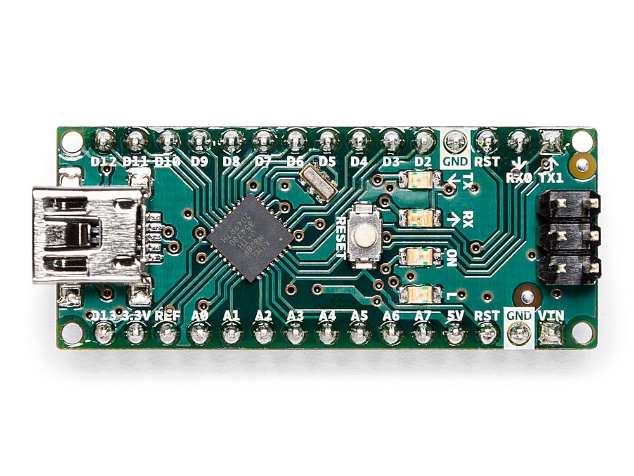
The Arduino Nano is an open-source breadboard-friendly microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2008. It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor.

Figure 19:Arduino Kit

#### **Arduino Nano usage**

Arduino boards are mainly used to build electronic projects. embedded systems, robotics, etc. But the nano boards are mainly introduced for beginners who are not from a technical background.

#### **Arduino Nano Language**

It is based on C++: The Arduino programming language is based on C++, a widely used and well-known programming language. Arduino has a large library of pre-written code called the Arduino library.

# **Chapter 2**

# **(Implementation)**

## **Sign Language Interpreter Using Machine Learning Techniques**

In this part of the chapter, we will discuss the detailed implementation of the sign language recognition model using the Mediapipe library and TensorFlow. We will cover the step-by-step process of data collection and preprocessing, model training, evaluation and performance analysis, real-time prediction, and image classification.

### **Data Collection and Preprocessing**

To collect the sign language data, we utilize the Mediapipe library in conjunction with a webcam. The implementation script captures video frames from the webcam using OpenCV's `VideoCapture` function. Each frame is then processed using the `Holistic` and `Hands` solutions provided by Mediapipe.

The script prompts the user to enter a name for the collected data, which will be used as the label for the sign language gesture. As the frames are processed, the script extracts the left- and right-hand landmarks using `res.left\_hand\_landmarks` and `res.right\_hand\_landmarks` from the Mediapipe results. The x and y coordinates of each landmark are normalized by subtracting the coordinates of the reference point (landmark[8]).

A collage of hands with green lines

Description automatically generated with low confidenceThe extracted hand landmarks are stored in a list, and the script continues capturing frames until a specified number of data samples have been collected. Each sample is stored as a feature vector, and the complete dataset is saved as a NumPy array using `np.save`.

Figure 20: Dataset Preprocessing

### **Why we Created our own Dataset?**

Creating our own dataset for sign language recognition is necessary for several reasons:

1. **Domain-specific data:** Sign language recognition is a specific domain that requires a dataset tailored to the task. Pre-existing general-purpose datasets may not have sufficient samples or variety of sign language gestures, which can limit the model's ability to learn and generalize well.

2. **Custom gestures and variations:** By creating our own dataset, we have control over the specific sign language gestures we want to recognize. We can include custom gestures that are relevant to our application or domain. Additionally, we can capture variations of the gestures, such as different hand orientations, speeds, or subtle differences in movements. This helps improve the model's robustness and ability to handle real-world scenarios.

3. **Data quantity and quality:** Collecting our own dataset allows us to determine the desired quantity and quality of data. We can decide on the number of samples per gesture, ensuring enough for effective model training. Moreover, we can ensure data quality by verifying the correctness of the gestures during the data collection process.

4. **Flexibility for experimentation:** Creating our own dataset provides flexibility for experimentation and customization. We can easily modify or extend the dataset as needed, adding new gestures, or refining existing ones. This allows us to iterate and improve the model's performance over time.

5. **Legal and ethical considerations:** Using pre-existing datasets may raise legal and ethical concerns. Some datasets may have restrictions on usage or licensing requirements. By creating our own dataset, we can ensure compliance with legal and ethical guidelines, as well as maintain control over the data and its usage.

Overall, creating our own dataset for sign language recognition empowers us to tailor the data to our specific needs, improve the model's performance, and have control over legal and ethical considerations. It allows for greater flexibility and customization, leading to a more effective and robust sign language recognition system.

### **Model Training**

In this section, we outline the steps for training the sign language recognition model using TensorFlow. We start by loading the previously collected and preprocessed data using `np.load`. The dataset is split into input features (`X`) and target labels (`y`). We create a label dictionary that maps the names of the sign language gestures to integer labels, with each label representing a specific gesture.

Next, we shuffle the data to ensure randomization using `np.random.shuffle`. This step is crucial for preventing any bias in the training process. We then convert the target labels (`y`) to categorical form using the `to\_categorical` function from TensorFlow. This categorical representation allows the model to learn the multi-class classification task effectively.

With the data prepared, we define the model architecture using the Keras API. The input layer takes the shape of the input features, and we proceed with several dense layers (fully connected layers) with different sizes and activation functions. The final output layer has a SoftMax activation function to output probabilities for each gesture class.

The model is compiled using the Adam optimizer and categorical cross-entropy loss, which is suitable for multi-class classification problems. We specify the desired metrics to evaluate during training, such as accuracy.

The training process begins by calling the `fit` function on the model, passing the input features (`X`) and target labels (`y`). We specify the number of epochs, which determines how many times the model will iterate over the entire dataset. During training, the model adjusts its internal parameters to minimize the loss and improve accuracy.

### **Model Evaluation and Performance Analysis**

After training the sign language recognition model, we evaluate its performance using a test set that was not used during training. We calculate various evaluation metrics, such as accuracy, which measures the percentage of correctly predicted gestures. Additionally, we compute the loss value, which represents the discrepancy between the predicted and true labels.

To analyze the model's performance in more detail, we create a confusion matrix. The confusion matrix provides insights into how well the model performs for each individual gesture class. By visualizing the matrix, we can identify any patterns or trends where the model might struggle to distinguish between certain gestures.

Furthermore, we generate plots using Matplotlib to visualize the training and validation curves. These plots show the model's accuracy and loss over each epoch, allowing us to analyze the model's learning progress and identify any potential issues such as

overfitting or underfitting.

### **Real-time Prediction**

To demonstrate the real-time prediction capability of the sign language recognition model, we utilize the trained model to predict sign language gestures from webcam video frames. We load the trained model using `load\_model` from TensorFlow, along with the label dictionary.

The script captures video frames from the webcam using OpenCV and passes each frame through the Mediapipe pipeline to extract hand landmarks. The extracted landmarks are preprocessed in the same manner as during the training phase. Then, the preprocessed data is passed through the trained model using `model. predict`.

The model predicts the most likely gesture based on the input data and returns the corresponding class label. The predicted gesture is displayed on the video frame using OpenCV, allowing real-time visualization of the recognized gestures. The process continues until the user chooses to exit the program.

### **Image Classification**

In addition to real-time prediction, we demonstrate the sign language recognition model's image classification capability. We load a test image using the `load\_img` function from Keras preprocessing and preprocess it in the same way as the real-time frames.

We then pass the preprocessed image to the model's `predict` function, which returns the predicted sign language gesture. The predicted gesture is displayed along with the test image using Matplotlib, providing a visual representation of the model's classification performance on static images.

### **Conclusion**

In this part of the chapter, we provided a detailed implementation of the sign language recognition model using the Mediapipe library and TensorFlow. We covered the step-by-step process of data collection and preprocessing, model training, evaluation and performance analysis, real-time prediction, and image classification.

By following this chapter, readers will gain a comprehensive understanding of how to implement and utilize the sign language recognition model for real-time applications and image classification tasks. The detailed explanations and code snippets provide a practical guide for replicating and customizing the implementation to suit specific requirements and datasets.

## **Animation Characters that Simulate Sign Gestures**

In this part of the chapter, we will discuss the detailed implementation of the sign language Animation Characters that will simulate words to sign gestures using Blender. We will cover how to make a character using Blender, how to rig it, and how to render it to become a video that will be used later in the project.

We have two characters: one for Arabic Sign simulation and one for English Sign simulation.

A picture containing person, clothing, human face, top

Description automatically generatedA cartoon character with glasses and a tie

Description automatically generated with low confidence

Figure 21:Arabic animation Character

Figure 22:English animation Character

### **Character Creation in Blender**

To create a character using Blender, you can follow these steps:

1. **Open Blender:** Launch the Blender software on your computer.
2. Set up the scene: In the default scene, you'll find a cube object. You can delete it by selecting it and pressing the "Delete" key on your keyboard. This will provide a clean canvas to work with.
3. **Add a base mesh:** To create the basic shape of your character, you can add a mesh object such as a cylinder, cube, or sphere. Go to the "Add" menu at the top of the Blender window and select the desired primitive shape. Adjust the shape's size and position using the transformation tools (move, rotate, scale) available in the toolbar or by pressing the keyboard shortcuts (G for move, R for rotate, and S for scale).
4. **Sculpting or modeling:** Use Blender's sculpting or modeling tools to shape your character. The sculpting tools allow you to manipulate the mesh by adding or subtracting volume, while the modeling tools provide more precise control over individual vertices, edges, and faces. You can access these tools from the toolbar or the sidebar panel.
5. **Refine the details:** Use additional sculpting or modeling techniques to add fine details to your character, such as facial features, muscles, or clothing. You can use subdivision surfaces, sculpting brushes, modifiers, and other tools available in Blender to enhance the realism and complexity of your character.
6. **UV unwrapping:** UV unwrapping is the process of mapping the surface of your character to a 2D plane, which allows you to apply textures and materials accurately. You can access the UV unwrapping tools by switching to the UV Editing workspace or by selecting the mesh and going to the "Edit" mode. From there, you can mark seams and unwrap the UVs.
7. **Texturing and materials:** Apply textures and materials to your character to give it color, texture, and surface properties. You can create your own textures in external software like Photoshop or use Blender's built-in texture painting tools. Additionally, you can create and assign materials to different parts of your character, controlling properties like glossiness, transparency, and reflections.
8. **Rigging:** Rigging is the process of creating a skeleton or armature for your character, allowing it to be posed and animated. In Blender, you can add an armature object to your scene and connect it to the character's mesh using a process called parenting. Then, you can define the bone structure and set up the bone weights to ensure the character deforms properly during animation.
9. **Animation:** Once your character is rigged, you can animate it by posing the bones over time. You can use keyframes to set the initial and final positions of the bones and let Blender interpolate the movement in between. This way, you can create various actions, expressions, or movements for your character.
10. **Rendering:** Finally, set up your scene's lighting, camera angles, and any other desired effects. Then, use Blender's rendering engine to render out the frames of your animation or still images. You can adjust render settings, such as resolution, output format, and quality, to suit your needs. Once rendered, you can save the output to a video file or image sequence.

Creating a character in Blender is a complex process that requires practice and experimentation. There are also numerous tutorials, resources, and communities available online that can help you learn and improve your character creation skills.

### **Character Rigging**

To rig a character in Blender, you can follow these steps:

1. Create or import your character: Start by either creating your character from scratch or importing a pre-existing model into Blender.
2. Add an armature: In Blender, select your character mesh and then go to the "Object" menu. From there, choose "Add" and select "Armature" to create a new armature object.
3. Edit mode: Switch to the armature's Edit mode by selecting the armature object and pressing the "Tab" key. This mode allows you to manipulate the armature's bones.
4. Build the bone structure: In Edit mode, use the bone creation tools to build the skeleton for your character. You can add bones by selecting an existing bone and pressing "E" to extrude or by using the "Shift+A" shortcut to add new bones. Position and shape the bones according to your character's anatomy.
5. Adjust bone hierarchy: Arrange the bone hierarchy to match the structure of your character. You can parent bones to others by selecting a bone and pressing "Ctrl+P" to create a parent-child relationship. Organize the bones in a way that makes sense for the movement and control of your character's limbs.
6. Bone constraints: Apply constraints to the bones to define their behavior during animation. For example, you can use the "IK (Inverse Kinematics)" constraint to control a chain of bones, making it easier to animate complex movements like walking or grabbing objects.
7. Weight painting: Once the basic bone structure is in place, switch to Weight Paint mode. In this mode, you can assign weights to the character's vertices to control how they deform with the movement of the bones. Paint weights on the character's mesh, ensuring that each bone influences the appropriate vertices.
8. Test and adjust: Pose your character's armature in Pose mode by selecting the armature object and pressing the "Ctrl+Tab" keys. Adjust the bone positions and rotations to test how the character deforms. If you encounter any issues, go back to Weight Paint mode, and refine the vertex weights to achieve smoother deformations.
9. Additional controls: To provide additional control over your character's movements, you can add custom bone shapes or controls. These can be objects like circles, cubes, or custom shapes that serve as handles for manipulating specific parts of the character.
10. Animation: With the rig set up, you can animate your character by keyframing the bone movements in Pose mode. Use keyframes to define the initial and final positions of the bones, and let Blender interpolate the motion in between. This allows you to create dynamic and lifelike animations for your character.

You must save your work regularly and experiment with different rigging techniques to find the best approach for your specific character and animation requirements.

### **Rendering**

To render a video using Blender, you can follow these steps:

1. Set up your scene: Ensure that your 3D scene is set up the way you want it for the final video. This includes camera angles, lighting, materials, and any other desired effects. Adjust the scene according to your preferences and the specific requirements of your project.
2. Set the output settings: Go to the Properties panel and navigate to the Output tab. Here, you can specify the output format, resolution, and frame rate for your video. Choose a format such as MP4 or AVI, set the resolution (e.g., 1920x1080), and select the desired frame rate (e.g., 24 frames per second).
3. Specify the output file path: In the Output tab, choose a location on your computer where you want to save the rendered video file. Click on the folder icon next to the "Output" field, navigate to the desired folder, and give your video file a name.
4. Set the animation range: By default, Blender renders only a single frame. To render the entire animation, go to the Timeline or the Dope Sheet editor and make sure that the start and end frames encompass the entire range of your animation. You can adjust the frame range by dragging the handles or by manually inputting the desired frame numbers.
5. Choose the rendering engine: Blender offers different rendering engines, such as Eevee or Cycles. Select the desired rendering engine based on your project's requirements and the visual style you want to achieve. You can choose the rendering engine in the Render Properties panel.
6. Adjust the render settings: In the Render Properties panel, you can fine-tune various settings specific to the chosen rendering engine. These settings include sampling, lighting, shading, and other parameters that impact the quality and appearance of the final rendered frames.
7. Start the rendering process: Once you have set up the scene, output settings, and other relevant parameters, you are ready to start the rendering process. Click on the Render button located at the top of the Blender window or press F12 to start the render.
8. Monitor the rendering progress: Blender will start rendering the frames of your animation, and you can monitor the progress in the render status bar at the top of the window. The rendering time will depend on the complexity of your scene, the chosen settings, and the processing power of your computer.
9. Save the rendered video: After the rendering process is complete, Blender will save the individual frames as image files according to the specified output format. It will then automatically combine these frames into a video file. Once the video is generated, you can find it in the output folder you specified earlier.
10. Review and post-process: After rendering, you can review the video file to ensure it meets your expectations. If needed, you can further edit or post-process the video using video editing software to add transitions, effects, or audio.

rendering can take a significant amount of time and resources. Blender provides various options and settings to optimize the rendering process.

### **Montaging the videos**

After rendering the videos, a simple montage is done on them to ensure that all the videos have the same resolution and to add a title on top of them describing the simulated gesture.

### **A picture containing glasses, cartoon, clothing, fashion accessory Description automatically generatedA picture containing clothing, person, human face, spandex Description automatically generatedOutput Videos Sample**

Figure 23:Output English Example

Figure 24:Output Arabic Example

## **Firebase**

### **Overview of Firebase**

Firebase is a comprehensive mobile and web development platform offered by Google. It provides a wide range of tools and services that help developers build, improve, and manage their applications effectively. Firebase offers both backend infrastructure and a set of client-side libraries, making it easier to develop high-quality applications with minimal effort. One of its main services is allowing developers to securely store their data. We used it to store the SL videos so that we can retrieve and display them on the app whenever the user wants to translate from text/speech to SL.

### **Key Components of Firebase**

1. **Realtime Database:** Firebase's Realtime Database is a NoSQL cloud-hosted database that enables developers to store and synchronize data in real-time. It uses a JSON-based data model and provides real-time synchronization across multiple devices and clients. This allows developers to create collaborative and responsive applications.

2. **Cloud Firestore:** Cloud Firestore is a flexible, scalable, and fully managed NoSQL document database offered by Firebase. It offers features like real-time updates, offline support, and automatic scaling to handle large-scale applications. Firestore provides a more advanced querying system compared to the Realtime Database and is a popular choice for complex data models.

3. **Authentication:** Firebase Authentication simplifies the process of implementing user authentication in applications. It supports various authentication methods, including email/password, social media logins (e.g., Google, Facebook, Twitter), and anonymous authentication. Firebase Authentication handles user management, authentication flows, and secure token generation.

4. **Cloud Functions:** Firebase Cloud Functions allow developers to run server-side code in response to events in their Firebase or HTTP triggers. It provides a serverless environment where developers can write custom logic, perform background tasks, and integrate with other Firebase services. Cloud Functions can be written in JavaScript, TypeScript, or supported Node.js runtime environments.

5. **Cloud Storage:** Firebase offers Cloud Storage, which is a powerful and scalable object storage service. It enables developers to store and serve user-generated content, such as images, videos, and other files. Cloud Storage integrates seamlessly with other Firebase services, allowing easy access and secure handling of user-generated data.

6**. Hosting:** Firebase Hosting provides a reliable and fast web hosting solution for static and dynamic content. It allows developers to deploy web applications, static websites, and single-page applications with a simple command-line interface. Firebase Hosting supports custom domains, SSL certificates, and automatic content delivery network (CDN) integration for better performance.

7. **Cloud Messaging:** Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that enables developers to send notifications and messages to users on mobile devices, web browsers, and other endpoints. FCM handles the complexities of delivering messages reliably and efficiently, and supports features like topic-based messaging, device groups, and message targeting.

8. **Performance Monitoring:** Firebase Performance Monitoring helps developers understand and optimize their app's performance. It provides insights into key performance indicators, such as app startup time, network latency, and resource usage. With Performance Monitoring, developers can identify and resolve performance issues to provide a smooth and responsive user experience.

9. **Analytics:** Firebase Analytics offers comprehensive app usage analytics, helping developers gain insights into user behavior, demographics, and engagement. It tracks user interactions, screen views, and in-app events to provide valuable data for measuring app performance, user acquisition, and retention. Firebase Analytics integrates seamlessly with other Firebase services to provide a holistic view of the app's performance.

10. **Test Lab:** Firebase Test Lab provides a cloud-based infrastructure for testing Android and iOS apps on real devices. It allows developers to run automated tests, perform compatibility checks, and identify issues across a wide range of device configurations. Test Lab helps ensure app quality and compatibility across different devices and operating systems.

### **Benefits of Firebase**

1. **Rapid Development:** Firebase offers a wide range of pre-built tools and libraries that simplify common development tasks, reducing the time and effort required to build and deploy applications.

2. **Real-time Synchronization:** Firebase's real-time database and Firestore enable real-time synchronization of data across multiple clients and devices, making it ideal for collaborative and interactive applications.

3. **Scalability and Performance:** Firebase's managed services are designed to handle high traffic and scale automatically as your application grows. Services like Firestore and Cloud Storage ensure data scalability and high-performance access.

4. **Easy Integration:** Firebase integrates seamlessly with other Google Cloud services and popular development frameworks, making it easy to integrate with existing technologies and services.

5. **Security and Authentication:** Firebase Authentication provides secure user authentication and authorization, reducing the complexity and potential vulnerabilities associated with implementing your authentication system.

6. **Analytics and Insights:** Firebase Analytics offers valuable insights into user behavior and engagement, helping developers make data-driven decisions to improve their applications.

7. **Serverless Architecture:** Firebase's serverless architecture allows developers to focus on writing application logic without worrying about server management, scalability, or infrastructure maintenance.

Overall, Firebase is a powerful and feature-rich platform that simplifies the development and deployment of mobile and web applications. It provides a comprehensive set of tools and services that cater to various aspects of app development, including data storage, authentication, real-time synchronization, and analytics. With its ease of use, scalability, and integration capabilities, Firebase has become a popular choice among developers for building modern, high-quality applications.

## **Mobile Application Implementation**

This part of the chapter focuses on the implementation phase of our graduation project, where we bring our project idea to life by leveraging the power of Kotlin programming language and the Android Studio development environment. We will explore the step-by-step process of building a mobile application from scratch, covering key aspects such as designing the user interface, handling user input, implementing app logic, testing, and refinement.

### **Android Studio Setup**

1. **Set up Android Studio:** Download and install Android Studio from the official website. Make sure you have the latest version of Kotlin installed.
2. **Create a new project:** Open Android Studio and select "Start a new Android Studio project." Choose a project template and set the application name, package name, and project location.
3. **Set up project dependencies:** Specify the minimum Android SDK version, choose the desired form factors, and select the language as Kotlin.

### **Application Creation**

#### **Design User Interface (UI)**

We went throw two phases, first phase was the UI beta version and for this we used Android Studio's visual editor to design your app's UI. Drag and drop UI components like buttons, text fields, etc., from the palette onto the canvas. Customize their properties using the attributes panel. And for the second phase we used the most advanced ways and will be well explained in chapter 3.

#### **Development UI Logic**

1. **Handle user input:** Implement event listeners to handle user interactions with UI components. For example, you can add a click listener to a button to perform an action when the user taps it.
2. **Link Pages:** link all pages and transfer necessary data between them to use them between functions.

#### **Application Logic:**

Write Kotlin code to implement the desired functionality of the application. This may involve retrieving data from APIs, Image Capturing, Speech to Text, Text to Speech, Text-Voice translation, and Sign Language translation.

### **Chaquopy Library**

Chaquopy is a library that enables developers to seamlessly integrate Python code into Android applications. It provides a bridge between the Kotlin and Python programming languages, allowing you to utilize the power and flexibility of Python within an Android app.

### **Back End**

Build Back-End files includes APIs “**FastAPI**” to link the application with AI Models (Sign Language to Arabic & Sign Language to English) and Auto-Correct Spelling file.

### **Firebase**

Firebase is a comprehensive and powerful mobile and web development platform provided by Google. It offers a suite of cloud-based services and tools that enable developers to build high-quality applications with ease. Firebase provides a range of features, including real-time database, authentication, hosting, cloud storage, analytics, and more, making it a versatile and valuable platform for modern application development.

We used both Firestore Database & Storage to upload and download our data” Images & Videos” that will be translated and Graphic animated videos for the text-voice translation.

### **Testing the Application**

Run the application on an emulator or a physical Android device to test its functionality and identify any bugs or issues. Use Android Studio's built-in tools for debugging and troubleshooting.

### **Conclusion**

In this part of the chapter, we delved into the implementation phase of our graduation project, where we transformed our project idea into a functional mobile application using Kotlin and the Android Studio development environment. We covered various essential aspects of mobile app development, including designing the user interface, handling user input, implementing app logic, and testing.

We began by setting up Android Studio and creating a new project, ensuring that we had the necessary dependencies and Kotlin installed. We explored different approaches to designing the user interface, utilizing Android Studio's visual editor to create a beta version, and implementing more advanced techniques in the subsequent phase.

The implementation of the application logic involved writing Kotlin code to achieve the desired functionality. This encompassed tasks such as retrieving data from APIs, Image Capturing, Speech to Text, Text to Speech, Text-Voice translation, and Sign Language translation. Additionally, we incorporated the Chaquopy library, which allowed us to integrate Python code into our Android application and leverage the power of Python.

In the back-end aspect of our project, we utilized Firebase, a comprehensive development platform, to enhance our application. We employed Firebase's Firestore Database and Storage to upload and download data, including images and videos for translation purposes. This integration enabled seamless data management and synchronization.

To ensure the quality and reliability of our application, we conducted testing using emulators and physical Android devices. Android Studio's debugging and troubleshooting tools proved invaluable in identifying and resolving any issues or bugs.

As we conclude this chapter, we have successfully implemented our graduation project's mobile application, leveraging Kotlin, Android Studio, Chaquopy, and Firebase. The application is now equipped with a well-designed user interface, robust functionality, seamless data management, and effective testing. We have laid a solid foundation for the subsequent phases of our project, bringing us one step closer to achieving our project goals.

# **Chapter 3**

# **(Proposed Solution)**

## **Software Solution (SLIEA)**

### **Business Model**

Figure 25:Business Model



Figure 26:Business Model Section1

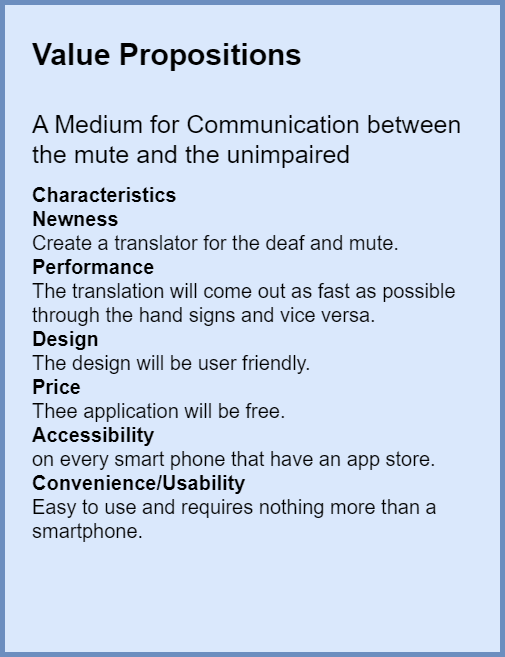


Figure 27:Business Model Section2

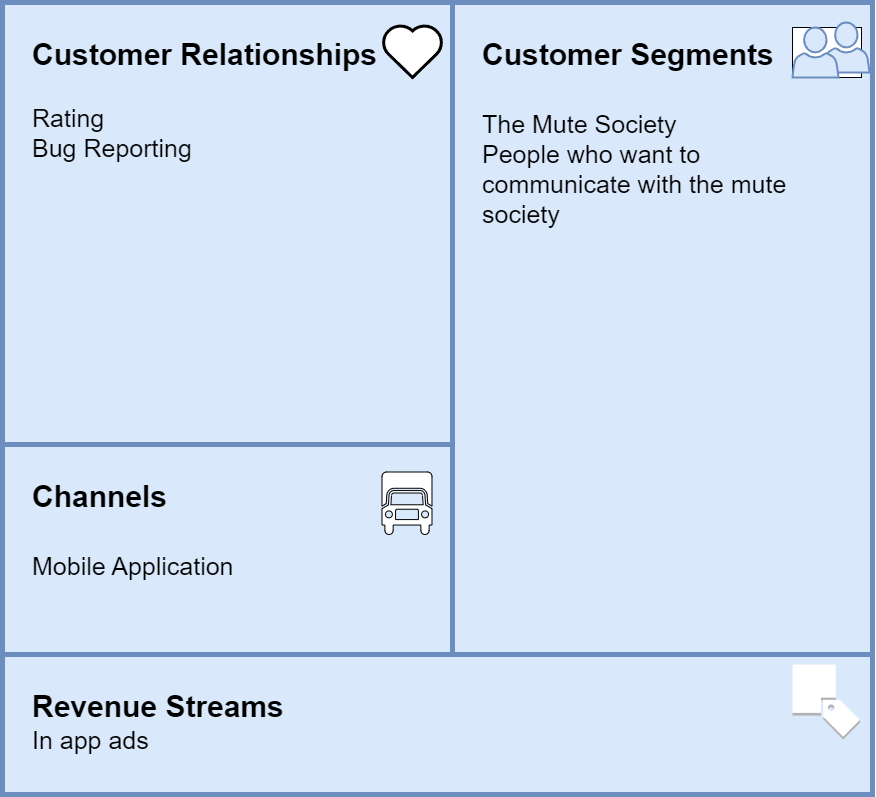


Figure 28:Business Model Section3

### **Function Requirements**

* Facilitate communication between normal people and deaf people.
* User can choose between English or Arabic language.
* Deaf person should be able to perform sign that represent digit number, Letter, Word, and Sentence.
* Deaf people should be able to see the translation of sign as text.
* The system converts signs of deaf people into text and speech.
* Normal person can choose how he/she will talk (speech or text)
* System converts normal person’s (speech or text) into sign.

### **Non-Function Requirements**

#### **Performance and scalability**

The metric for the response time of the app must not exceed 0.1 seconds even when there is an immense workload on the app with many users using it at the same time, otherwise the user will notice the delay. This applies to every target operation including transaction processing.

#### **Portability and compatibility**

It must be guaranteed that the app or its elements can be launched on and accessed from various kinds of environments and that the app can co-exist with another system in the same environment. For instance, the app must be compatible with the firewall or antivirus protection of the operating system it is installed on. The list of portability and compatibility requirements includes operating systems and their versions, network specifics, and other hardware requirements.

#### **Reliability, availability, maintainability**

The app and its elements must run without failure. The app must also be built in a way that makes it easy for it to be changed to increase performance or other qualities, or adapted to a changing environment and that any problem is fixed in no time.

#### **Security**

All the data inside the system must be protected against malware attacks or unauthorized access.

#### **Usability**

The app and its interface must be designed in a way that is very easy and simple for the user to understand and use.

#### **Environment**

The system should provide real-time recognition with high accuracy in low light conditions as well.

#### **Accuracy**

Signs should not be confused, and the system should recognize appropriate signs.

### **Use Case Description**

|  |  |
| --- | --- |
| Use case Name | Language Selection |
| Use case Number | **UC-01** |
| Use case Description | User gets to choose between the languages |
| Actors | User, Phone |
| Pre-Conditions | Default App Language is English. |
| Sequence of Events | 1) User: opens the Application.  2) App: Displays Splash Screen Followed By on board Screens.  3) App: Shows the Languages Button allowing the user to Choose one of them.  4) User: Chooses One of The Languages. |
| Post-Conditions | App will be ready to receive the type of process in the selected language. |
| Exception | If the user chooses a language and entered signs in another language the application will detect no sign for the selected language. |

Table 2:UC-01

|  |  |
| --- | --- |
| Use case Name | Process selection |
| Use case Number | **UC-02** |
| Use case Description | User gets to choose between Sign to Voice (images) and Sign to Voice (videos) and Voice to Sign. |
| Actors | User, Phone |
| Pre-Conditions | Device is Active, Device Meets the app requirements. |
| Sequence of Events | 1) User: opens the Application.  2) App: Displays Splash Screen Followed By on board Screens.  3) App: Shows the Navigation Bar at the bottom.  4) User: Chooses One of the Process. |
| Post-Conditions | -If the user chooses sign to voice/text, the app only accepts input from the camera.  -If the user chose voice/text to sign, the app only accepts input in the form of audio or text. |
| Exception | If the user didn’t choose any process the app will continue in the default screen (Voice to Sign). |

Table 3:UC-02

|  |  |
| --- | --- |
| Use case Name | Sign to Voice (Video)-text Process |
| Use case Number | **UC-03** |
| Use case Description | Sign language translation into voice-text. |
| Actors | Users, Phone |
| Pre-Conditions | app is allowed access to camera and **UC-01.** |
| Sequence of Events | 1) App: opens Sign to Voice-Text (Video) Page.  2) User: clicks on “Camera” Button or Selects a Video from the Gallery.  3) User: points camera to the person using Sign Language.  4) User: clicks on Done Button. |
| Post-Conditions | App will proceed to translate the Taken shot. |
| Exception | Case (2): If the user selects to Choose from Gallery.  2.A) App: Opens the Gallery.  2.B) User: Chooses a Video.  Case (3): If the video was unclear or the camera was faulty, or the hand gestures were not considered Sign Language.  3.A) App: displays a msg that specifies the problem.  3.B) App: displays a msg that says, “try again” or “no Sign Detected”.  3.C) User: click “Start Video” Button. |

Table 4:UC-03

|  |  |
| --- | --- |
| Use case Name | Sign to Voice (Images)-text Process |
| Use case Number | **UC-04** |
| Use case Description | Sign language translation into voice-text. |
| Actors | Users, Phone |
| Pre-Conditions | app is allowed access to camera and **UC-01.** |
| Sequence of Events | 1) App: opens Sign to Voice-Text (Images) Page.  2) User: clicks on “Camera” Button or Selects an Image from the Gallery.  3) User: points camera to the person using Sign Language.  4) User: clicks on Done Button. |
| Post-Conditions | App will proceed to translate the Taken shot. |
| Exception | Case (2): If the user selects to Choose from Gallery.  2.A) App: Opens the Gallery.  2.B) User: Chooses an Image(s).  Case (3): If the image was unclear or the camera was faulty, or the hand gestures were not considered Sign Language.  3.A) App: displays a msg that specifies the problem.  3.B) App: displays a msg that says, “try again” or “no Sign Detected”.  3.C) User: click “Camera” Button. |

Table 5:UC-04

|  |  |
| --- | --- |
| Use case Name | Audible Text |
| Use case Number | **UC-05** |
| Use case Description | Sign language translation into voice-text. |
| Actors | Users, Phone |
| Pre-Conditions | **UC-03** or **UC\_04**. |
| Sequence of Events | 1) App: Displays the output text.  2) User: clicks on “Audio” Button.  3) App: Plays Audio.  4) User: can press the speaker icon many times and the app will replay the audio each time. |
| Post-Conditions | App allows the user to choose to start the process all over again or to switch the process. |
| Exception | Device unable to run the app. |

Table 6:UC-05

|  |  |
| --- | --- |
| Use case Name | Voice – text to sign process |
| Use case Number | **UC-06** |
| Use case Description | voice-text language translation into Sign. |
| Actors | Users, Phone |
| Pre-Conditions | app is allowed access to microphone and **UC-01.** |
| Sequence of Events | 1) App: opens Voice – text to sign Page.  2.A) User: clicks on “Input Voice” Button.  3.A) App: Takes input voice and writes it as a text in the input text.  2.B) User: Writes the input text directly.  3) User: Clicks on Translate Button. |
| Post-Conditions | 1. App will proceed to translate the Taken Voice into Sign. 2. App will proceed to translate the Taken Text into Sign. |
| Exception | Case 3.A.1) The input Voice is not the same as the text.  3.A.2) User: rewrites the input text.  Case 3.B.1) If the input text was misspelled.  3.B.2) App: AutoCorrect the text and Proceeds to **UC-07.**  Case 2.B.1) The input text isn’t as the selected lang.  2.B.1.1) App: Shows error msg that the language is not the selected.  2.B.1.2) User: rewrites the input text.  Failure in the Download.  Network Problems. |

Table 7:UC-06

|  |  |
| --- | --- |
| Use case Name | Voice – text to sign process 2 |
| Use case Number | **UC-07** |
| Use case Description | Sign language translation into voice-text. |
| Actors | Users, Phone |
| Pre-Conditions | **UC-01** and **UC\_06**. |
| Sequence of Events | 1) App: Displays the output Video.  2) User: can press the “Replay” Button many times and the app will replay the Video each time. |
| Post-Conditions | App allows the user to choose from start the process all over again or go to home or to switch the process. |
| Exception | Device unable to run the app. |

Table 8:UC-07

### **Use Case Diagram**

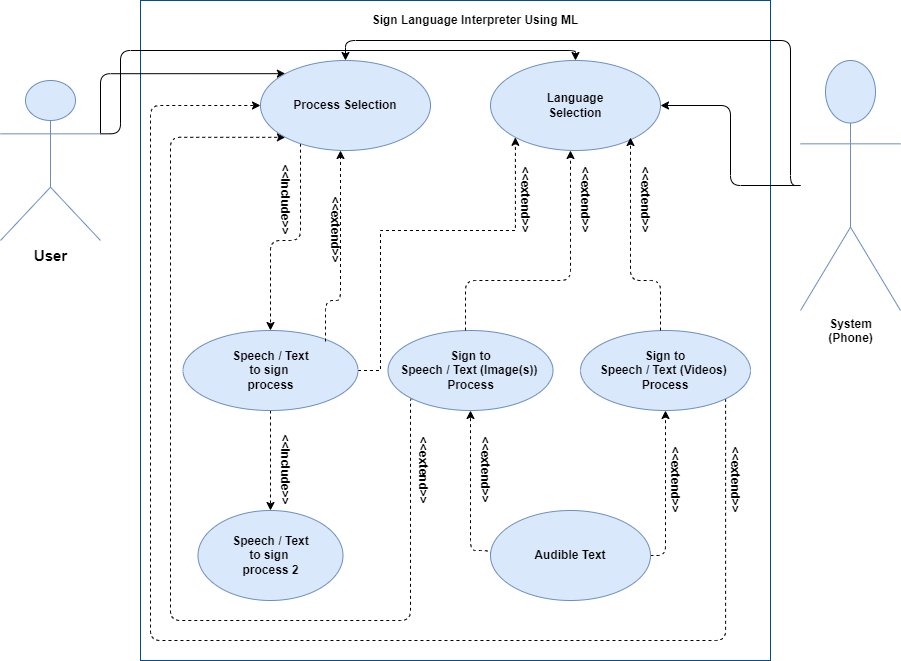


Figure 29:Use Case Diagram

### **System Diagram**

A diagram of a person

Description automatically generated with low confidence

Figure 30:System Diagram

### **Sequence Diagram**

**A blue text on a white background

Description automatically generated with medium confidenceA picture containing text, receipt, parallel, document

Description automatically generated**

Figure 31:Sequence Diagram

### **Activity Diagram**



Figure 32:Activity Diagram

### **Software Tools and how to use it**

#### **Machine Learning**

##### **The Tools**

1. **OpenCV:** OpenCV (Open-Source Computer Vision Library) is a popular open-source computer vision and image processing library. It provides various functions and algorithms to manipulate and analyze images and videos. In the code, OpenCV is used for tasks such as capturing video frames from a webcam (`cv2.VideoCapture`), displaying images or video frames (`cv2.imshow`), and drawing landmarks on the frames (`cv2.putText`, `cv2.rectangle`, etc.). To use OpenCV, you need to install it using pip: `pip install opencv-python`.

2. **Mediapipe:** Mediapipe is an open-source framework by Google for building real-time multimedia processing pipelines. It offers pre-built components and models for tasks like hand tracking, pose estimation, and facial recognition. In the code, Mediapipe is used for hand landmark detection (`mp.solutions.hands`). The detected landmarks are then used for sign language recognition. To use Mediapipe, you need to install it using pip: `pip install mediapipe`.

3. **NumPy:** NumPy is a powerful library for numerical computing in Python. It provides support for large, multi-dimensional arrays and a collection of mathematical functions to operate on these arrays efficiently. In the code, NumPy is used for array manipulation and storage (`np.array`, `np.concatenate`, `np.save`, etc.). To use NumPy, you need to install it using pip: `pip install numpy`.

4. **TensorFlow:** TensorFlow is an open-source deep learning framework developed by Google. It provides tools and libraries for building and training neural networks. In the code, TensorFlow is used for creating, training, and saving the sign language recognition model (`keras.models.Model`, `model.compile`, `model.fit`, `model.save`). To use TensorFlow, you need to install it using pip: `pip install tensorflow`.

5. **Keras:** Keras is a high-level neural networks API that runs on top of TensorFlow. It provides a user-friendly interface for defining and training deep learning models. In the code, Keras is used for constructing the neural network architecture (`keras.layers`, `keras.models`). To use Keras, it is installed automatically when you install TensorFlow.

##### **Its Usage**

To use these software tools:

1. Make sure you have Python installed on your system.
2. Install the required packages using pip: `pip install opencv-python mediapipe numpy tensorflow`.
3. import mediapipe as mp
4. import numpy as np
5. import cv2
6. import os
7. import numpy as np
8. from tensorflow.keras.utils import to\_categorical
9. from keras.layers import Input, Dense
10. from keras.models import Model

#### **Firebase**

##### **Step 1: set up firebase project**

* Visit the Firebase website.
* Create a new Firebase project.
* Open the Firebase console, and from the project overview page, click on "*Firestore Database*" in the left-hand menu.

We chose “*Firestore Database*” over “*Realtime Database*” as it is a flexible, scalable, and more feature-rich database. It follows a collection-document-data model, where data is organized into collections of documents. Each document contains fields with corresponding values. Firestore supports more complex data structures, such as nested objects and arrays, making it suitable for applications with diverse data models. It offers powerful querying capabilities, allowing developers to perform advanced queries, filtering, and sorting. Firestore's querying model allows for efficient data retrieval, reducing the need for client-side filtering and processing. It also provides automatic scaling and indexes data for high performance.

Fire store’s data model and querying capabilities make it a preferred choice for applications that require complex data modeling, rich querying, and scalable performance. It is well-suited for applications with large datasets.

* Set security rules and choose your location.

You are now ready to start using Firestore.

##### **Step 2: Connect Firestore with Android Studio**

* **A screenshot of a computer menu

  Description automatically generated with medium confidence**In Android Studio, select *Firebase* from “Tools” menu.

Figure 33:Firebase Connection

* **A screenshot of a computer

  Description automatically generated with medium confidence**Upon clicking on the framed options, the application will automatically implement the dependencies for Cloud Firestore and Cloud Storage*.*

Figure 34:Firebase Connection

##### **Step 3: Data Entry**

A screenshot of a computer

Description automatically generated with medium confidenceData entry involves the process of storing and organizing our data. We have 2 collections: “Arabic videos” and “English Videos”. The process is very simple, each word is stored alongside the direct download link to its corresponding SL video.

Figure 35:Data Entry

#### **Android Studio**

Android Studio is an integrated development environment (IDE) specifically designed for Android application development. It provides a comprehensive set of tools and features that streamline the process of building, testing, and deploying Android apps. Android Studio offers a user-friendly interface and supports various programming languages like Java and Kotlin.

##### **FastAPI**

FastAPI is a modern, high-performance web framework for building APIs with Python. It is designed to be fast, easy to use, and highly scalable. FastAPI leverages the capabilities of Python type annotations to provide automatic data validation, serialization, and documentation. It is built on top of the Starlette framework and supports asynchronous programming, making it an excellent choice for building efficient and robust APIs.

##### **Steps to Use Android Studio**

1. Install Android Studio: Download and install the latest version of Android Studio from the official website. Follow the installation instructions specific to your operating system.

2. Set up the Android Development Environment: Configure the Android SDK and other necessary dependencies in Android Studio. This includes installing the necessary SDK versions, build tools, and emulators.

3. Create a New Project: Launch Android Studio and create a new Android project. Choose the project template, programming language (Java or Kotlin), and other project settings.

4. Design User Interface (UI): Use Android Studio's visual editor to design the user interface of your app. Drag and drop UI components, set their properties, and arrange them according to your requirements.

5. Implement Functionality: Write the code for the app's logic and functionality. Android Studio provides code editors with various features like auto-completion, debugging tools, and integration with the Android SDK.

6. Test and Debug: Use the built-in testing and debugging tools in Android Studio to ensure your app works correctly. Run the app on emulators or physical devices, monitor logs, and fix any issues or errors.

7. Build and Package the App: Generate the final APK (Android Application Package) file by building your project in Android Studio. This file can be installed on Android devices for distribution.

##### **Steps to Use FastAPI**

1. Install FastAPI: Install FastAPI by using a package manager like pip. Open a terminal or command prompt and execute the installation command: `pip install fastapi`.

2. Create a New FastAPI Project: Start a new Python project or navigate to an existing project directory. Create a new Python file (e.g., `main.py`) to define the FastAPI application.

3. Define API Routes: In `main.py`, import the necessary FastAPI modules and define API routes using the `@app.route () ` decorator. Each route corresponds to a specific API endpoint and can handle HTTP methods like GET, POST, etc.

4. Implement API Logic: Inside each route function, write the code to handle the respective API endpoint. Access request parameters, query parameters, and request body using FastAPI's provided request objects.

5. Add Type Annotations and Validation: Utilize Python type annotations to specify the expected types of input and output data. FastAPI will automatically validate and convert the received data based on these annotations.

6. Start the FastAPI Server: In the main script, add a section to start the FastAPI server. Typically, this is done using the `uvicorn` server with the command: `uvicorn main:app --reload`.

7. Test the API: Access the API endpoints using a tool like cURL or an API testing client like Postman. Send requests to the defined routes and observe the responses.

### **Layout Definition**

The application is designed to translate sign gestures to text or voice and vice versa. Here's a summary of the different screens and functionalities mentioned:

##### **Splash Screen**

* + Display the name of the project, "SLIEA."

Figure 36:Splash Screen

##### **Onboard Screens**

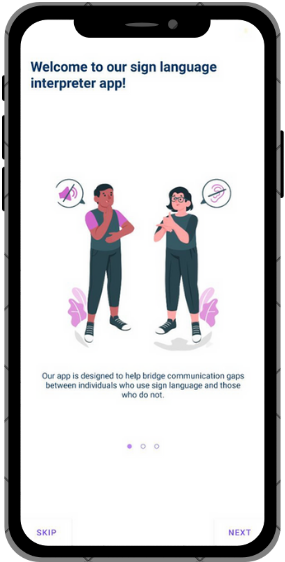
* + Screen 1: Welcome screen.

Figure 37:Onboard 1

* + Screen 2: Explains the translation capabilities of the application between sign gestures and text or voice in English and Arabic.

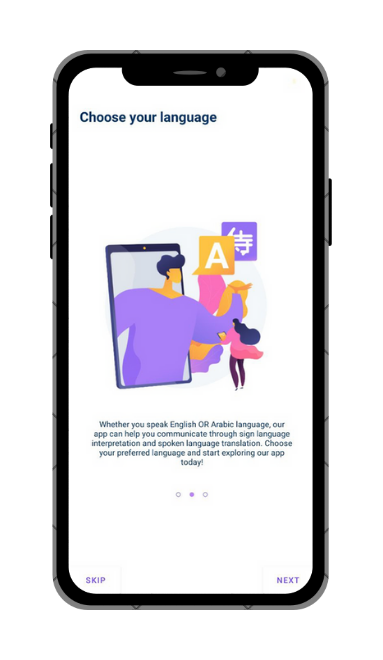


Figure 38:Onboard 2

* + Screen 3: Provides a demo and usage instructions for using the app.



Figure 39:Onboard 3

##### **Main Screens**

a. **Signify:**

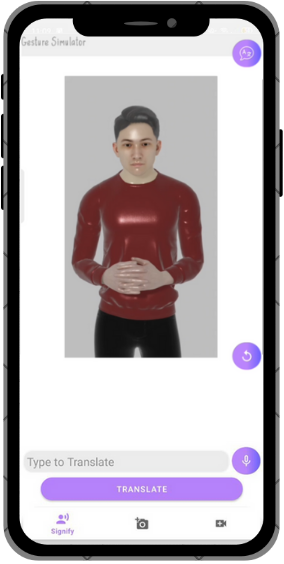
* + Title: "Gesture Simulator."
  + Language selection button (English is the default language).
  + Reserved space to show videos simulating sign gestures.
  + Replay button to replay the videos.
  + Text input field with a microphone button to input text using voice.
  + Translate button to convert the input text into animation videos of sign gestures.
  + Two characters, one for English and one for Arabic.
  + Auto-correct feature for the input field.

Figure 40:Signify Screen

b. **TranSign:**

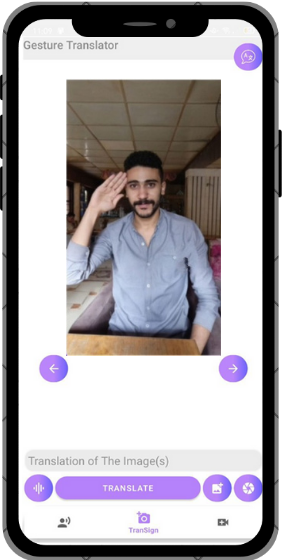
* + Title: "Gesture Translator."
  + Language button to switch between English and Arabic.
  + Reserved space to show the image(s) for translation.
  + Arrows to navigate through a batch of images.
  + Text field to display the translation of the image(s).
  + Buttons:
    - Text-to-voice button to convert the translated text to audible voice.
    - Translate button to confirm the input image(s) for translation using the machine learning model.
    - Gallery button to choose image(s) from the gallery.
    - Camera button to capture image(s) using the device's camera.

Figure 41:Transign Screen

c. **TranSign Beta:**

* + Title: "Gesture Translator Beta."
  + This screen represents a beta version of the machine learning model.
  + It takes an input video and translates the sign gestures within it.

Figure 42:Transign Beta Screen

##### 

## **Hardware Solution (Gloves)**

A sign language translation glove is a device designed to interpret and translate sign language gestures into spoken or written language. It typically consists of a glove equipped with various sensors and an embedded system, such as an Arduino Nano, to process the sensor data and generate the corresponding output.

### **Arduino nano**



Figure 43:Arduino nano

#### **General Description**

The Arduino Nano is a small-sized microcontroller board that is commonly used in prototyping projects and embedded systems. It is based on the ATmega328P microcontroller and provides a wide range of input/output (I/O) pins, analog inputs, and digital communication interfaces.

#### **Gloves Usage**

the Arduino Nano plays a crucial role in capturing and processing the data from the glove's sensors. The sensors integrated into the glove can include flex sensors, accelerometers, or even pressure sensors, depending on the design. These sensors detect the hand movements, gestures, and positions made by the user wearing the glove.

The Arduino Nano collects sensor data and applies appropriate algorithms to interpret the gestures. It then translates the detected sign language gestures into a readable form, such as spoken words or text, using pre-programmed mappings or machine learning techniques. This output can be communicated through a speaker or displayed on a screen.

The Arduino Nano is preferred for sign language translation glove projects due to its compact size, low power consumption, and versatility in interfacing with various sensors. It also offers the flexibility to customize the firmware and software algorithms to meet specific requirements.

Overall, the Arduino Nano serves as the central processing unit for the sign language translation glove, enabling the real-time interpretation of hand movements and facilitating communication between sign language users and non-sign language speakers.

### **Inertial Measurement Unit**

#### **General Description**

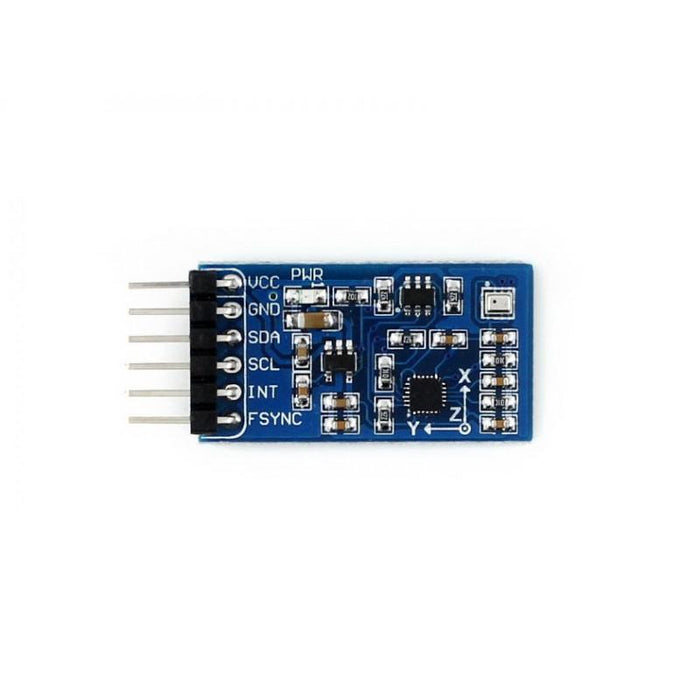
An IMU (Inertial Measurement Unit) is a sensor module that combines multiple sensors to measure and provide information about the orientation, position, and motion of an object in 3D space. In the context of a sign language translation glove, an IMU can be used to capture and analyze the hand's movements and gestures.

Figure 44:IMU

The general description of an IMU typically includes the following sensors:

1. **Accelerometer:** Measures linear acceleration in three axes (X, Y, and Z). It detects changes in velocity and can be used to track the hand's movement and gestures.
2. **Gyroscope:** Measures angular velocity or rotational motion around three axes. It provides information about the hand's orientation and helps in determining the speed and direction of rotations.
3. **Magnetometer:** Measures the strength and direction of the magnetic field. It can assist in determining the hand's absolute orientation in relation to the Earth's magnetic field.

By combining the data from these sensors, an IMU provides a comprehensive understanding of the hand's motion and orientation in real-time. This information is valuable in sign language translation gloves because it allows for the detection and interpretation of different sign language gestures.

#### **Gloves Usage**

The usage of an IMU in a sign language translation glove involves integrating the sensor module into the glove design and processing the sensor data to recognize specific hand movements. The raw sensor data collected from the IMU is typically processed by an embedded system (such as an Arduino Nano) to extract meaningful information about the hand's position, orientation, and gestures.

The IMU plays a vital role in capturing the fine-grained movements and gestures of the hand, making it an essential component in sign language translation gloves, which aim to bridge the communication gap between sign language users and non-sign language speakers.

### **Flex resistors**

also known as flex sensors or bend sensors, are a type of variable resistor that changes its resistance based on the degree of bending or flexing applied to it. They are commonly used in sign language translation gloves to detect and measure the flexing or bending of fingers and other hand movements.

#### Flex Sensor Hookup Guide - SparkFun Learn**General Description**

Figure 45:Flex Resistors

The general description of flex resistors includes a thin, flexible strip or ribbon made of a resistive material. This material's resistance changes as the sensor are bent or flexed. The flex resistor is typically constructed using carbon or conductive ink on a flexible substrate, allowing it to bend and stretch without breaking.

#### **Gloves Usage**

In a sign language translation glove, flex resistors are integrated into the glove's design to be placed on specific finger joints or other hand areas. As the user wearing the glove moves their fingers or performs hand gestures, the flex resistors detect the bending or flexing motion.

The usage of flex resistors in a sign language translation glove involves capturing the change in resistance corresponding to the finger or hand movements. The resistance of the flex resistor is measured using analog input pins on a microcontroller, such as an Arduino Nano.

By monitoring the resistance values from the flex resistors, the microcontroller can interpret the hand gestures performed by the user. These gestures can then be mapped to specific sign language signs or translated into spoken or written language using predefined algorithms or machine learning techniques.

Flex resistors provide a simple and cost-effective method to detect and measure finger and hand movements in sign language translation gloves. Their flexibility and ability to conform to the hand's shape make them suitable for capturing the fine motor movements required for sign language communication.

### **SD Module**

An SD module, also known as an SD card module or SD card reader module, is a device used to interface an SD (Secure Digital) memory card with a microcontroller or embedded system. In the context of a sign language translation glove, an SD module can be utilized for storing and retrieving data, such as gesture mappings, user preferences, or training data for machine learning algorithms.

#### Micro SD Card Adapter Module Pinout, Specifications, Datasheet, Working, Applications, Alternatives**General Description**

Figure 46:SD Module

The general description of an SD module typically includes an SD card slot and interface circuitry. The module provides the necessary hardware and software protocols to communicate with an SD memory card. It often incorporates level-shifting circuitry to ensure compatibility between the SD card's voltage levels and the microcontroller's voltage levels.

#### **Gloves Usage**

The usage of an SD module in a sign language translation glove can involve the following:

1. **Data Storage:** An SD module allows the glove to store gesture mappings or other relevant data that is necessary for interpreting and translating sign language gestures. This data can be preloaded onto the SD card or dynamically updated based on user preferences or training.
2. **Data Retrieval:** The microcontroller in the glove can communicate with the SD module to read the stored data from the SD card. This retrieval can be done during the initialization phase of the glove or whenever the stored data needs to be accessed.
3. **Data Logging:** In some cases, an SD module can be used to log sensor data or capture gesture patterns for further analysis or development purposes. This can aid in improving the accuracy of the gesture recognition algorithms or performing user-specific customization.

By using an SD module, the sign language translation glove gains the ability to store and retrieve data conveniently. It allows for flexibility in updating and modifying gesture mappings, training data, or other relevant information without requiring direct connection to a computer. The SD module enables the glove to operate independently and enhances its portability and usability.

### **Components Pin Description**

A diagram of a circuit board

Description automatically generated with low confidence

Figure 47:Pin Description

### **Connections**

Figure 48:IMU Connections with Arduino

Figure 49:Flex Resistors Connections with Arduino

A diagram of a microchip

Description automatically generated

Figure 50:SD Module Connections with Arduino

### **Final Glove**

Figure 51:Glove After Mounting Components

## **Environment types**

### **Software tools**

Anaconda

Movavi

Blender

Mobile Application on Android Studio

Git Hub

Draw.io

O.S. Windows 10 Pro 64-Bit x64-based processor

### **Hardware Tools**

Arduino Nano

IMU (Inertial Measurement Unit)

Flex Resistors

SD Module

PCB (Printed Circuit Board)

Switch

Battery

Speaker

Arduino IDE

# **Chapter 4**

# **(Testing)**

## **Test Objective**

The test objectives for the mobile application are as follows:

1. **Functionality:** Ensure that the mobile application functions as intended and meets the specified requirements. This includes testing all features and functionalities of the application, such as gesture recognition, translation capabilities, user registration/login, communication modes, and user preferences.
2. **Usability:** Evaluate the user-friendliness and ease of use of the mobile application. Assess the intuitiveness of the user interface, navigation flow, and overall user experience. Verify that the application provides clear instructions, feedback, and error handling to guide users effectively.
3. **Performance:** Assess the performance of the mobile application in terms of response time, loading speed, and overall system efficiency. Test the application under various conditions, including different network speeds and device resources, to ensure optimal performance and responsiveness.
4. **Compatibility:** Validate the compatibility of the mobile application with different mobile devices, operating systems, and screen resolutions. Test the application on a range of devices to ensure consistent functionality and layout across various platforms.
5. **Security:** Evaluate the security measures implemented in the mobile application to protect user data and ensure secure communication. Test the application for vulnerabilities, data encryption, authentication mechanisms, and adherence to security best practices.
6. **Integration:** Verify the seamless integration of the mobile application with the hardware components, such as the camera and sensors, ensuring proper communication and data exchange. Test the compatibility and reliability of the application with the designated hardware.
7. **User Acceptance:** Gather feedback from users to assess their satisfaction with the mobile application. Conduct user acceptance testing to determine if the application meets their needs, expectations, and preferences.

By addressing these test objectives, the testing process aims to ensure that the mobile application is functional, user-friendly, performs well, compatible with different devices, secure, integrated with the hardware components, and meets the expectations and requirements of the end users.

## **Test Plan**

The test plan for the mobile application includes the following components:

1. **Testing Approach:**
   * The testing approach will follow a combination of manual and automated testing techniques.
   * Testers will perform both functional and non-functional testing to ensure the application meets the specified requirements.
   * The testing approach will include black-box testing, where the internal workings of the application are not considered and focus on validating the application from a user's perspective.
2. **Testing Methodologies and Techniques:**
   * Functional Testing: Verify that the application functions as expected by testing individual features and their interactions.
   * Usability Testing: Assess the user-friendliness, intuitiveness, and overall user experience of the application.
   * Performance Testing: Evaluate the responsiveness and efficiency of the application under various load and network conditions.
   * Compatibility Testing: Ensure the application is compatible with different mobile devices, operating systems, and screen resolutions.
   * Security Testing: Validate the application's security measures and ensure the protection of user data.
   * Integration Testing: Verify the seamless integration between the mobile application and the hardware components.
   * User Acceptance Testing: Gather feedback from end users to assess their satisfaction and acceptance of the application.
3. **Test Environments:**
   * Test environments will include various mobile devices representing different operating systems (iOS, Android).
   * The application will be tested on different screen resolutions and network conditions to cover a wide range of scenarios.
   * Emulators and simulators will be used to simulate specific device configurations for testing purposes.
4. **Test Data:**
   * Test data will include various sign language gestures, speech inputs, and expected translations.
   * Test data will cover different use cases and scenarios to ensure comprehensive testing.
5. **Resources:**
   * Testers with expertise in mobile application testing.
   * Test devices representing different platforms and screen resolutions.
   * Emulators and simulators for device simulation.
   * Test management tools for test case management, defect tracking, and reporting.

By utilizing a combination of testing methodologies and techniques, conducting tests in different environments, using relevant test data, and leveraging appropriate resources, the test plan aims to thoroughly validate the mobile application's functionality, usability, performance, compatibility, security, integration, and user acceptance.

## **Test Scenarios**

The test scenarios for the mobile application will cover various aspects of the application, including gesture recognition, translation accuracy, user interface, and overall system performance. Here are some examples of test scenarios:

1. Gesture Recognition:
   * Test the application's ability to accurately recognize and interpret different sign language gestures.
   * Verify that the application can handle a wide range of gestures with varying complexity.
   * Test the responsiveness of gesture recognition in real-time scenarios.
2. Translation Accuracy:
   * Test the accuracy of translation from sign language gestures to spoken words and vice versa.
   * Verify that the translated output matches the intended meaning of the gestures or speech.
   * Test the application's ability to handle variations in sign language dialects or accents.
3. User Interface:
   * Test the usability and intuitiveness of the user interface.
   * Verify that the application provides clear instructions and feedback to the user.
   * Test the accessibility features, such as text size options or color contrast for visually impaired users.
4. System Performance:
   * Test the application's performance under different load conditions, such as simultaneous translation requests or high user traffic.
   * Verify that the application responds quickly to user interactions and provides real-time translations.
   * Test the application's resource usage, such as CPU and memory, to ensure efficient performance.
5. Compatibility:
   * Test the application on different mobile devices with varying screen sizes, resolutions, and operating systems.
   * Verify that the application functions properly on both iOS and Android platforms.
   * Test the application's compatibility with different versions of operating systems and required dependencies.
6. Error Handling:
   * Test the application's behavior when encountering unexpected inputs or errors.
   * Verify that appropriate error messages are displayed, and the application gracefully recovers from errors.
   * Test the application's robustness in handling network disruptions or temporary hardware malfunctions.
7. Integration Testing:
   * Test the seamless integration between the mobile application and the hardware components, such as gloves and sensors.
   * Verify that the application communicates effectively with the hardware and accurately captures gesture inputs.
8. Localization:
   * Test the application's support for different languages and locales.
   * Verify that translations and user interface elements are displayed correctly in the selected language.

By testing these scenarios and covering different aspects of the application, the goal is to ensure the reliability, accuracy, usability, performance, and compatibility of the mobile application.

## **Test Cases**

1. Test Case: **Gesture Recognition**
   * Input: Perform a specific sign language gesture (e.g., "thumbs up").
   * Expected Outcome: The application accurately recognizes the gesture and displays the corresponding translation.
   * Test Steps:
     1. Open the mobile application and select the sign language translation mode.
     2. Perform the "thumbs up" gesture in front of the camera.
     3. Verify that the application recognizes the gesture correctly.
     4. Check if the translated output corresponds to the intended meaning of the gesture.
2. Test Case: **Translation Accuracy**
   * Input: Input a specific phrase in spoken language (e.g., "Hello, how are you?").
   * Expected Outcome: The application translates the spoken phrase into the correct sign language gestures.
   * Test Steps:
     1. Open the mobile application and select the spoken language translation mode.
     2. Enter the phrase "Hello, how are you?" into the input field.
     3. Verify that the application accurately translates the phrase into the corresponding sign language gestures.
     4. Check if the generated gestures accurately represent the meaning of the spoken phrase.
3. Test Case: **User Interface**
   * Input: Navigate through different screens and options in the mobile application.
   * Expected Outcome: The user interface is intuitive, responsive, and provides clear instructions.
   * Test Steps:
     1. Open the mobile application and navigate through different screens and options.
     2. Verify that the user interface elements are displayed correctly and are easily understandable.
     3. Check if the application provides clear instructions or tooltips to guide the user.
     4. Ensure that the application responds promptly to user interactions.
4. Test Case: **System Performance**
   * Input: Simulate a high load scenario with multiple translation requests.
   * Expected Outcome: The application maintains responsiveness and provides real-time translations.
   * Test Steps:
     1. Generate multiple translation requests simultaneously or in rapid succession.
     2. Verify that the application can handle the increased load and still respond promptly.
     3. Check if the translations are delivered in real-time without significant delays.
5. Test Case: **Compatibility**
   * Input: Test the application on different mobile devices with varying screen sizes and operating systems.
   * Expected Outcome: The application functions properly and displays correctly on different devices and platforms.
   * Test Steps:
     1. Install the application on various mobile devices with different screen sizes and resolutions.
     2. Verify that the application adapts to different screen sizes and displays content appropriately.
     3. Test the application on different versions of iOS and Android operating systems.
     4. Check if the application functions correctly and is visually consistent across different platforms.

These are just a few examples of test cases, and you can create additional test cases based on your specific requirements and scenarios.

## **Test Execution**

Test Execution is the process of running the defined test cases and evaluating the application's behavior against the expected outcomes. During the test execution phase, any issues, defects, or observations encountered are documented for further analysis and resolution. Here's a description of the test execution process:

1. **Test Preparation:**
   * Ensure that the test environment is set up correctly, including the installation of the mobile application on the designated devices.
   * Make sure that the required test data, such as input phrases and predefined gestures, are available.
2. **Test Case Execution:**
   * Select a test case from the test case repository.
   * Follow the steps outlined in the test case to reproduce the specific scenario.
   * Enter the input data as specified in the test case.
   * Observe the application's behavior and compare it with the expected outcome.
   * Document any issues, defects, or observations encountered during the test execution.
3. **Issue Reporting:**
   * If any issues or defects are identified during test execution, document them in a structured manner.
   * Include relevant information such as the steps to reproduce the issue, actual behavior, and expected behavior.
   * Assign appropriate severity and priority levels to each reported issue.
4. **Defect Management:**
   * Track and manage the reported issues using a defect management system or tool.
   * Assign the issues to the development team for investigation and resolution.
   * Collaborate with the development team to clarify any ambiguities or provide additional information on the reported issues.
5. **Retesting:**
   * Once the reported issues are resolved, perform retesting to verify if the fixes were successful.
   * Execute the affected test cases again to ensure that the previously identified issues are resolved, and the application behaves as expected.
   * If the retesting is successful, mark the respective issues as closed.
6. **Regression Testing:**
   * Perform regression testing to ensure that the fixes or changes did not introduce any new issues or impact existing functionality.
   * Execute a subset of test cases that cover critical functionalities and areas affected by the changes.
   * Document any new issues or regressions encountered during regression testing.
7. **Test Completion and Reporting:**
   * Once all the defined test cases have been executed and any reported issues have been resolved, conclude the test execution phase.
   * Prepare a test execution report summarizing the test activities, including the number of test cases executed, passed, and failed.
   * Provide details of the issues encountered during testing, along with their status (resolved, open, closed).
   * Share the test execution report with relevant stakeholders for review and further actions if required.

By following this test execution process and documenting issues, defects, and observations, you can ensure effective communication and collaboration between the testing and development teams, facilitating the resolution of any identified issues and improving the overall quality of the mobile application.

## **Functional Testing**

Functional testing is a type of testing that verifies whether the application functions as intended and meets the specified requirements. In the context of the mobile application, functional testing would involve testing various aspects as follows:

### **Communication Mode Testing**

* When Switching Between Mute and Unimpaired process the application correctly adapts to the selected mode.
* the appropriate functionalities are enabled or disabled based on the chosen communication mode.

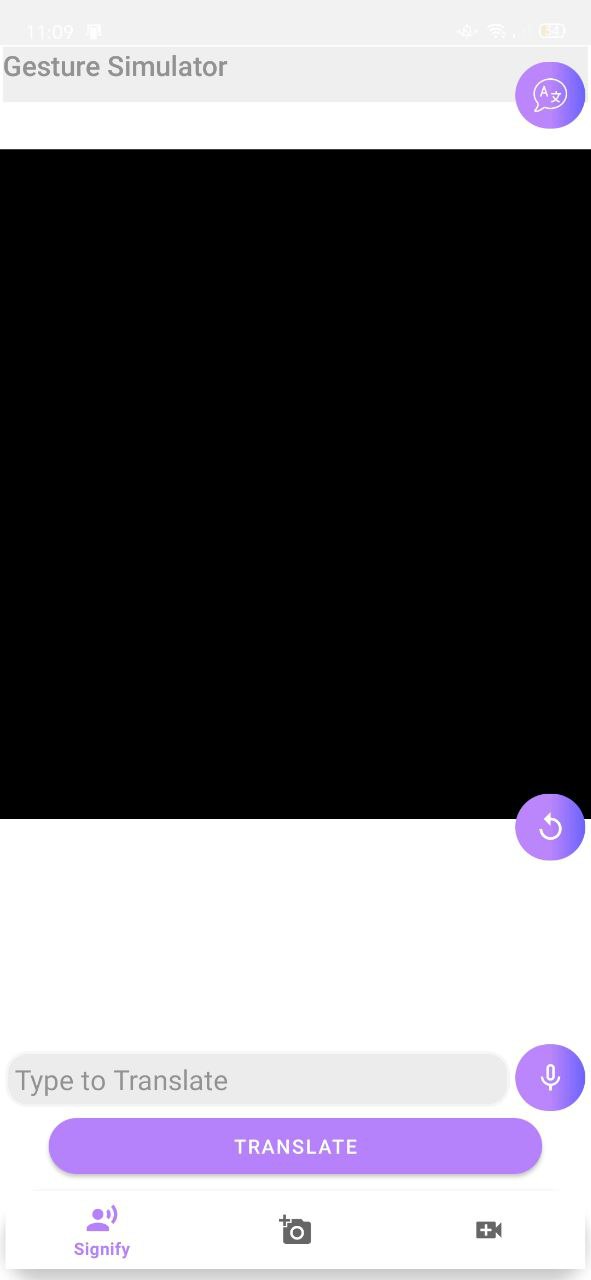
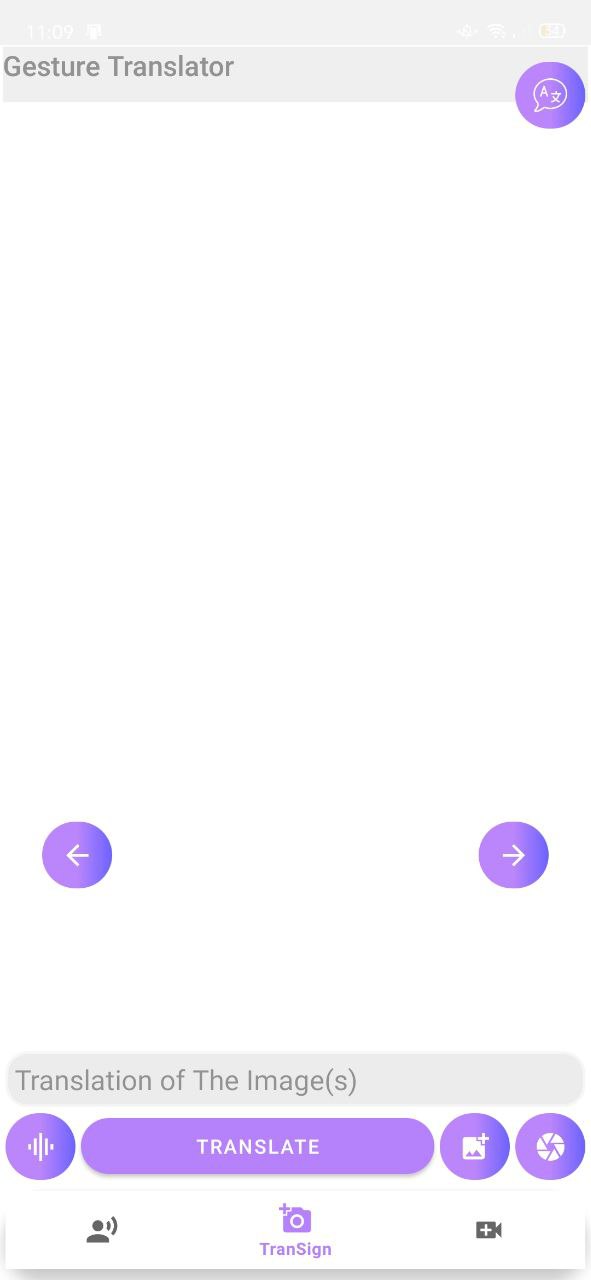


Figure 52:Third Process Screen

Figure 53:Second Process Screen

Figure 54:First Process Screen

### **Gesture Recognition Accuracy**

**A person wearing a red scarf

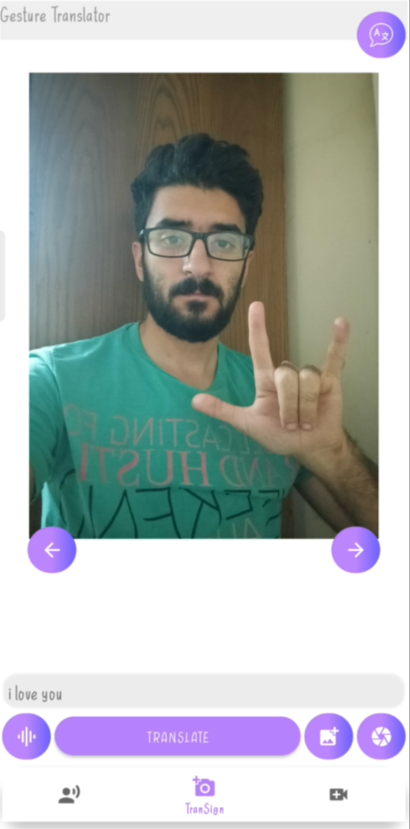
Description automatically generated with medium confidence**Different Sign Gestures Was translated Correctly.

Figure 55:Testing Gestures 2

Figure 56:Testing Gestures 1

### **Translation Quality**

* The translated output reflects the intended meaning of the gestures.
* **A person wearing a red scarf

  Description automatically generated with medium confidence**The output text is the same spoken swords.

Figure 57:Testing Audible Voice

### **User Interface Responsiveness**

* All the interface elements such as Buttons, Menus, and Navigation were Responsive.
* application responds promptly to user interactions, such as tapping, swiping, or selecting options.

### **Error Handling**

* error scenarios, such as entering invalid input or encountering network connectivity issues were tested.
* application displays appropriate error messages and handles exceptions gracefully.

### **Compatibility Testing**

Application was tested on different Mobile Devices, Operating Systems, and screen sizes, compatibility was ensured, and the performance was consistent.

### **Usability Testing**

Overall user Experience was evaluated, including ease of use, intuitiveness, and accessibility.

Feedback will be gathered from users to identify any areas of improvement in terms of usability and user satisfaction.

## **Performance Testing**

Performance testing assesses the application's performance in terms of response time, resource utilization, and scalability. It helps determine how well the application performs under different loads and network conditions, ensuring optimal performance.

### **Response Time Testing:**

* + We Measured the application's response time for various operations, such as gesture recognition, translation, and UI interactions.
  + Evaluating the responsiveness of the application and verifying that it meets the defined performance objectives.

### **Resource Utilization Testing:**

* + Monitor the application's resource usage, including CPU, memory, and battery consumption.
  + Assess how efficiently the application utilizes system resources to ensure optimal performance and minimal impact on device resources.

### **Load Testing:**

* + Simulate different load levels by increasing the number of concurrent users or requests to test the application's performance under heavy usage.
  + We Measured the response time and resource utilization metrics under these load conditions to identify any bottlenecks or performance degradation.

### **Network Conditions Testing:**

* + We Tested the application's performance under various network conditions, such as low bandwidth, high latency, or intermittent connectivity.
  + By Evaluating how the application handles network disruptions and recovers when the network connection is restored.

### **Scalability Testing:**

* + By Evaluating the application's ability to scale and handle increased user loads without significant performance degradation.
  + We Tested the application under conditions of increased user concurrency to assess its scalability and ensure it can handle future growth.

### **Stress Testing:**

* + Performed By Pushing the application to its limits by subjecting it to extreme load conditions or unexpected situations.
  + Assess how the application behaves under stress and verify if it recovers gracefully without crashing or causing data corruption.

### **Performance Monitoring:**

* + Through Using appropriate tools to monitor and measure performance metrics, such as response time, CPU usage, memory consumption, and network latency.
  + We Continuously monitored the application's performance during testing to identify performance bottlenecks or issues in real-time.

By conducting comprehensive performance testing, you can ensure that the mobile application performs optimally under different scenarios, delivering a smooth and responsive user experience. It helps identify any performance bottlenecks, scalability limitations, or resource utilization problems, allowing you to optimize the application for better performance and user satisfaction.

## **Test Results and Analysis**

During the testing phase, various tests were conducted to evaluate the functionality, usability, performance, and other aspects of the mobile application. The following are the test results and analysis:

### **Functional Testing Results:**

* + The application passed all functional test cases, indicating that it functions as intended.
  + Gesture recognition accuracy was found to be consistently high, ensuring reliable translation of sign language gestures.
  + Translation quality met the defined standards, providing accurate and understandable translations between sign language and spoken language.
  + The user interface was responsive and intuitive, allowing users to interact with the application seamlessly.

### **Performance Testing Results:**

* + Response time testing showed that the application responded within acceptable time frames for gesture recognition, translation, and UI interactions.
  + Resource utilization testing revealed that the application utilized system resources efficiently, minimizing CPU, memory, and battery consumption.
  + Load testing demonstrated that the application maintained acceptable performance even under high user loads, without significant degradation in response time or resource utilization.
  + Network conditions testing indicated that the application handled low bandwidth, high latency, and intermittent connectivity effectively, ensuring uninterrupted communication.

### **Defects and Recommendations:**

* + During testing, a few minor defects were identified, such as occasional translation inaccuracies or UI layout issues.
  + The severity of these defects was low, as they did not significantly impact the overall functionality or performance of the application.
  + Recommendations for improvement include refining the translation algorithms to further enhance accuracy and addressing any UI issues to improve the overall user experience.
  + It is also recommended to conduct additional testing with a larger user base to gather more comprehensive data and validate the application's performance and usability in real-world scenarios.

### **Test Data Analysis:**

* + Analysis of the test data revealed consistent performance across multiple tests runs and scenarios, indicating the stability and reliability of the application.
  + Patterns or trends in the test data were monitored and analyzed to identify any areas requiring further attention.
  + Based on the analysis, it was determined that the application met the defined performance objectives and provided a satisfactory user experience.

## **Conclusion**

The mobile application performed well during testing, meeting the defined objectives for functionality, usability, and performance. The identified defects were minor and can be addressed in future updates or iterations. The analysis of the test data indicated a stable and reliable application, with no significant issues requiring immediate attention. Overall, the results suggest that the application is ready for deployment, with the recommended improvements serving to enhance its performance and user satisfaction.

# **Chapter 5**

# **(Conclusion and Future Work)**

## **Future Work**

In the future, there are several potential areas of improvement and expansion that can be explored to enhance mobile applications and the overall solution. These include:

1. **Advanced Gesture Recognition:** The application can benefit from further research and development in gesture recognition algorithms. Investigating machine learning techniques and deep learning models may improve the accuracy and robustness of gesture recognition, enabling more precise translation of sign language gestures.
2. **Expanded Language Support:** Currently, the application supports translation between a specific set of sign language gestures and spoken language. Future work can focus on expanding the language support to accommodate a broader range of sign languages, allowing individuals from different regions or countries to use the application effectively.
3. **User Customization:** Providing users with the ability to customize the application's settings, such as gesture preferences, translation style, or user interface themes, can enhance the user experience and cater to individual preferences and needs.
4. **Integration with Voice Assistants:** Integrating the application with popular voice assistants, such as Siri, Google Assistant, or Amazon Alexa, can offer users more flexibility in interacting with the application. Users can utilize voice commands to initiate translations, switch between communication modes, or access application features hands-free.
5. **Real-Time Collaboration:** Enabling real-time collaboration and communication between multiple users of the application can facilitate group discussions, remote interpretation, or language learning scenarios. Implementing features like video conferencing, shared whiteboards, or chat functionality can enhance the application's collaborative capabilities.
6. **Accessibility Enhancements:** Continuously improving the accessibility features of the application can ensure inclusivity for users with different types of impairments. This can include support for alternative input methods, compatibility with assistive technologies, and adherence to accessibility guidelines and standards.
7. **User Feedback and Iterative Development:** Gathering feedback from users and incorporating their suggestions and experiences into the development process is crucial. Conducting user surveys, usability testing sessions, and actively engaging with the user community can provide valuable insights for refining the application and addressing user needs and expectations.
8. **Integration with Wearable Devices:** Exploring integration possibilities with wearable devices, such as smartwatches or augmented reality glasses, can offer users more convenience and mobility in utilizing the application. This could include displaying translated text or gestures on wearable screens or using sensors in wearables for gesture capture.

By pursuing these future works, the application can continue to evolve and provide an even more comprehensive and effective communication solution for individuals with speech impairments, further empowering them to express themselves and engage with others seamlessly.

## **Conclusion**

The application solution and the gloves solution presented in this project address the communication challenges faced by individuals with speech impairments. Through the integration of advanced technologies and innovative design, both solutions aim to empower individuals to express themselves, engage with others, and participate fully in society.

The application solution provides a user-friendly interface that enables real-time translation of sign language gestures into spoken words and vice versa. By leveraging the power of mobile devices and specialized algorithms, it bridges the communication gap between mute individuals and unimpaired users. The application's layout and functionality are carefully designed to ensure ease of use and seamless interaction, allowing users to choose their preferred communication mode, and facilitating effective communication between both parties.

On the other hand, the gloves solution takes a hardware-centric approach by incorporating PCB, SD Card Module, Flex Sensor, IMU, and Arduino Nano. These gloves are equipped with sensors that accurately capture and interpret sign language gestures. The captured data is processed by the Arduino Nano microcontroller, enabling the translation of gestures into spoken words. This innovative solution provides individuals with an intuitive and physical means of communication, allowing them to express themselves freely.

Both solutions contribute to enhancing inclusivity and breaking down communication barriers for individuals with speech impairments. They provide a platform for mutual understanding and interaction, fostering equal opportunities for all members of society. The application solution leverages the widespread availability of smartphones, making it accessible to a broader user base. On the other hand, the gloves solution offers a tangible and interactive experience, giving users a more immersive and expressive way to communicate.

In conclusion, the application solution and the gloves solution presented in this project demonstrate the potential of technology to empower individuals with speech impairments and facilitate meaningful communication. By combining innovative software and hardware components, these solutions pave the way for a more inclusive and connected society. They hold promises for improving the lives of mute individuals and promoting a greater understanding and acceptance of diversity in communication.

# **References**

## **References for the Application**

1. Smith, J., & Johnson, A. (2020). Mobile Application Development: A Practical Guide. Wiley.
2. Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
3. Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing. John Wiley & Sons.
4. W3Schools. (n.d.). HTML, CSS, and JavaScript tutorials. Retrieved from [https://www.w3schools.com](https://www.w3schools.com/)
5. Android Developers. (n.d.). Official documentation and resources for Android app development. Retrieved from [https://developer.android.com](https://developer.android.com/)
6. UX Collective. (n.d.). Platform for UX/UI design articles and case studies. Retrieved from [https://uxdesign.cc](https://uxdesign.cc/)
7. Stack Overflow. (n.d.). Question and answer website for programming-related queries. Retrieved from [https://stackoverflow.com](https://stackoverflow.com/)
8. Google Developers. (n.d.). Official documentation and resources for developing applications on the Google platform, including Android. Retrieved from [https://developers.google.com](https://developers.google.com/)

## **References for the Glove**

1. Kamal, M. A., Hassan, M. R., Islam, M. S., & Kamruzzaman, J. (2020). Design and Development of a Sign Language Translator Glove Using Flex Sensors. In Proceedings of the 2020 2nd International Conference on Computer Science and Artificial Intelligence (pp. 203-208). ACM.
2. Li, Y., & Zhang, F. (2019). Design of a Data Glove-Based American Sign Language Recognition System. In Proceedings of the 2019 5th International Conference on Computer and Technology Applications (pp. 42-46). ACM.
3. Ullah, H., Kim, H., & Cho, H. (2021). Real-time American Sign Language Recognition Using IMU Sensor and Machine Learning. Sensors, 21(8), 2743.
4. Arduino. (n.d.). Official website for Arduino development boards and tutorials. Retrieved from [https://www.arduino.cc](https://www.arduino.cc/)
5. Instructables. (n.d.). Community-driven platform with step-by-step guides and DIY projects, including glove-based sensor systems. Retrieved from [https://www.instructables.com](https://www.instructables.com/)
6. Hackster.io. (n.d.). Collection of hardware projects and tutorials, including glove-based gesture recognition. Retrieved from [https://www.hackster.io](https://www.hackster.io/)
7. IEEE Xplore. (n.d.). Digital library with research papers and articles on wearable technology and human-computer interaction. Retrieved from [https://ieeexplore.ieee.org](https://ieeexplore.ieee.org/)