# TITLE PAGE

**LANGUAGE TRANSLATOR APP USING TKINTER IN PYTHON**

**BY**

**MOHAMMED ADAMU**

**(ST/CS/HND/21/050)**

**DEPARTMENT OF COMPUTER SCIENCE,**

**SCHOOL OF SCIENCE AND TECHNOLOGY,**

**FEDERAL POLYTECHNIC, MUBI, ADAMAWA STATE.**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF HIGHER NATIONAL DIPLOMA (HND) IN COMPUTER SCIENCE.**

**SEPTEMBER, 2023**

# DECLARATION

I hereby declare that the work in this project titled **“Language Translator App Using Tkinter in Python”** was performed by me under the supervision of Mal. Adamu Garba Mubi. The information derived from literatures has been duly acknowledged in the text and a list of references provided. The work embodied in this project is original and had not been submitted in part or in full for any other diploma or certificate of this or any other institution.

Mohammed Adamu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ST/CS/HND/21/050) Signature Date

# CERTIFICATION

This project titled **“Language Translator App Using Tkinter in Python”** meets the regulations governing the award of Higher National Diploma (HND) in Computer Science, Federal Polytechnic Mubi, Adamawa State

Mal. Adamu Garba Mubi \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Project Supervisor) Sign/Date

Mr. Mustapha Kassim \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Head of Department) Sign/Date

Mal. Abdulrahman Saidu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(External Examiner) Sign/Date

# DEDICATION

This project is dedicated to my beloved parents for their advice, encouragement and financial support towards my academic pursuit.

# ACKNOWLEDGEMENTS

I wish to register my profound gratitude to almighty God for the guidance and grace throughout my life. I would also like to extend special regards to my amazing parents who have been the source of any success in my life. May the almighty God continue showering them with blessing Amen.

I am also grateful to my project supervise Mal. Adamu Garba Mubi whose help, stimulating suggestions and encouragement helped me in all time of fabrication process and in writing this project.

I also acknowledge the Head of Department Computer Science Mr. Mustapha Kassim for his moral encouragement throughout my period of study. I also acknowledge all Staff of Computer Science Department for their support and encouragement and the knowledge they’ve impacted on me throughout my studies.

In addition, my special gratitude goes to friends, love ones and all my course mates for their kind support throughout my studies

# TABLE OF CONTENTS

[TITLE PAGE i](#_Toc147156423)

[DECLARATION ii](#_Toc147156424)

[CERTIFICATION iii](#_Toc147156425)

[DEDICATION iv](#_Toc147156426)

[ACKNOWLEDGEMENTS v](#_Toc147156427)

[TABLE OF CONTENTS vi](#_Toc147156428)

[LIST OF FIGURES viii](#_Toc147156429)

[ABSTRACT ix](#_Toc147156430)

[CHAPTER ONE 1](#_Toc147156431)

[INTRODUCTION 1](#_Toc147156432)

[1.1 Background of the Study 1](#_Toc147156433)

[1.2 Problem Statement 2](#_Toc147156434)

[1.3 Aim and Objectives 2](#_Toc147156435)

[1.4 Significance of the Study 3](#_Toc147156436)

[1.5 Scope of the Study 3](#_Toc147156437)

[1.6 Definition of Some Operational Terms 4](#_Toc147156438)

[CHAPTER TWO 5](#_Toc147156439)

[LITERATURE REVIEW 5](#_Toc147156440)

[2.1 Introduction 5](#_Toc147156441)

[2.2 Language Translation 5](#_Toc147156442)

[2.3 Evolution of Language Translation 6](#_Toc147156443)

[2.4 Key Features and Functionalities 6](#_Toc147156444)

[2.5 Related Literature 7](#_Toc147156445)

[2.6 Summary 9](#_Toc147156446)

[CHAPTER THREE 10](#_Toc147156447)

[SYSTEM DESIGN AND ANALYSIS 10](#_Toc147156448)

[3.1 Introduction 10](#_Toc147156449)

[3.2 Disadvantages of the Existing System 10](#_Toc147156450)

[3.3 Advantages of the Proposed System 10](#_Toc147156451)

[3.4 The Proposed method 11](#_Toc147156452)

[3.5 Method of data collection 12](#_Toc147156453)

[3.6 System design 12](#_Toc147156454)

[3.6.1 Algorithm diagram 12](#_Toc147156455)

[3.6.3 Input and Output Design 13](#_Toc147156456)

[3.6.4 Report Layout 14](#_Toc147156457)

[3.5 System Requirement Specification 14](#_Toc147156458)

[3.5.1 Hardware Requirements 14](#_Toc147156459)

[3.5.2 Software Requirements 14](#_Toc147156460)

[3.5.3 Personnel Requirement 14](#_Toc147156461)

[CHAPTER FOUR 15](#_Toc147156462)

[RESULTS AND DISCUSSION 15](#_Toc147156463)

[4.1 Introduction 15](#_Toc147156464)

[4.2 Results 15](#_Toc147156465)

[4.2.1 Translator Interface 15](#_Toc147156466)

[4.2.2 Translated Output Interface 16](#_Toc147156467)

[4.3 Discussion 16](#_Toc147156468)

[4.4 User manual 17](#_Toc147156469)

[CHAPTER FIVE 18](#_Toc147156470)

[SUMMARY, CONCLUSION AND RECOMMENDATIONS 18](#_Toc147156471)

[5.1 Summary 18](#_Toc147156472)

[5.2 Conclusion 18](#_Toc147156473)

[5.3 Recommendations 18](#_Toc147156474)

[5.4 Contribution to Knowledge 19](#_Toc147156475)

[5.5 Area for Further Work 19](#_Toc147156476)

[REFERENCES 20](#_Toc147156477)

[APPENDICES 21](#_Toc147156478)

# LIST OF FIGURES

Figure 3.1: Use case diagram - - - - - - - - 11

Figure 3.2: System Architecture - - - - - - - 12

Figure 3.3: Translator Form - - - - - - - - 12

Figure 3.4: Translated report - - - - - - - - 13

Figure 4.1: Translator interface - - - - - - - 14

Figure 4.2: Translated Output Interface - - - - - - 15

# ****ABSTRACT****

*In our increasingly interconnected world, language barriers pose a significant challenge to effective communication. To address this issue, language translation applications have evolved, powered by advancements in machine learning and natural language processing. This project aims to develop a language translator application using Python and Tkinter to overcome these drawbacks and provide users with a user-friendly and efficient solution. The methodology involves the development of a language translator application with a focus on user-friendliness, extensive language support, efficiency, and customization. Tkinter, a powerful GUI library for Python, is utilized to create an intuitive graphical user interface. External language translation APIs are integrated to ensure accurate and multilingual translations. The application's responsiveness and customization options are optimized for a seamless user experience. The developed language translator application successfully addresses the disadvantages of existing systems. It offers a user-friendly interface that simplifies the translation process, supports a wide range of languages, ensures efficient and responsive translations, and provides customization options for users. Privacy considerations are taken into account to protect user data during translation. The project demonstrates the feasibility and effectiveness of creating a language translator application using Python and Tkinter. The application offers a user-centric approach to language translation, enhancing the accessibility and quality of translations. By leveraging external language translation APIs, it achieves high translation accuracy and extensive language support. The integration of customization options and privacy measures further enhances the user experience. Based on the project's findings, it is recommended to continue refining the application by exploring additional features such as offline translation support and pronunciation assistance.*

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background of the Study

Language translation has become an essential tool in our increasingly globalized world. It enables people from different linguistic backgrounds to communicate effectively, access information, and collaborate across borders. With the advent of technology, automatic language translation systems have gained popularity due to their convenience and efficiency. Language barriers have long been a challenge to effective communication, both in personal and professional spheres. As our world becomes increasingly interconnected, the demand for accurate and efficient language translation tools continues to rise. Language translation applications have evolved significantly in recent years, owing to advancements in machine learning, natural language processing (NLP), and the availability of extensive multilingual datasets (Sharma, 2017).

Translation is the communication of the [meaning](https://en.wikipedia.org/wiki/Meaning_(linguistic)) of a [source-language](https://en.wikipedia.org/wiki/Translation#Source_and_target_languages) text by means of an [equivalent](https://en.wikipedia.org/wiki/Dynamic_and_formal_equivalence) [target-language](https://en.wikipedia.org/wiki/Translation#Source_and_target_languages) text. The English language draws a [terminological](https://en.wikipedia.org/wiki/Terminology) distinction (which does not exist in every language) between translating (a written text) and [interpreting](https://en.wikipedia.org/wiki/Language_interpretation) (oral or [signed](https://en.wikipedia.org/wiki/Sign_language) communication between users of different languages); under this distinction, translation can begin only after the appearance of [writing](https://en.wikipedia.org/wiki/Writing) within a language community. A translator always risks inadvertently introducing source-language words, [grammar](https://en.wikipedia.org/wiki/Grammar), or [syntax](https://en.wikipedia.org/wiki/Syntax) into the target-language rendering. On the other hand, such "spill-overs" have sometimes imported useful source-language [calques](https://en.wikipedia.org/wiki/Calque) and [loanwords](https://en.wikipedia.org/wiki/Loanword) that have enriched target languages. Translators, including early translators of [sacred texts](https://en.wikipedia.org/wiki/Sacred_text), have helped shape the very languages into which they have translated. Because of the laboriousness of the translation process, since the 1940s efforts have been made, with varying degrees of success, to [automate translation](https://en.wikipedia.org/wiki/Machine_translation) or to [mechanically aid the human translator](https://en.wikipedia.org/wiki/Computer-assisted_translation). More recently, the rise of the [Internet](https://en.wikipedia.org/wiki/Internet) has fostered a [world-wide market](https://en.wikipedia.org/wiki/World-wide_market) for [translation services](https://en.wikipedia.org/wiki/Translation_services) and has facilitated "[language localisation](https://en.wikipedia.org/wiki/Language_localisation)" (Garry, 2018).

In a study conducted by Vaswani *et al.* (2017), introduced the Transformer architecture, which revolutionized machine translation and language understanding tasks. Transformers, equipped with attention mechanisms, have become the backbone of many state-of-the-art translation models, enabling more accurate and context-aware translations. This breakthrough has laid the foundation for the development of modern language translation applications. Furthermore, the advent of cloud-based Application Programming Interfaces (APIs) from technology giants such as Google, Microsoft, and Amazon have made high-quality translation services accessible to developers. These APIs leverage powerful NLP models and large-scale parallel corpora to provide accurate translations across a wide range of languages. For instance, Google's Cloud Translation API and Microsoft's Azure Translator API have become popular choices for developers seeking reliable translation services (Sharma, 2017).

Recent studies, such as the work of Devlin *et al.* (2018), have shown the remarkable capabilities of pre-trained language models. Bidirectional Encoder Representations from Transformers (BERT) and similar models have improved contextual understanding, enabling translation systems to consider the context of a sentence or phrase for more accurate translations. These models have significantly enhanced the quality of automated translations. On the interface front, the development of graphical user interface (GUI) libraries, like Tkinter for Python, has simplified the creation of user-friendly applications. Tkinter, as a widely-used GUI framework, offers a robust toolkit for designing intuitive interfaces that enhance the user experience.

Python, a versatile programming language, offers a wide range of libraries and tools for developing various applications. Tkinter is a popular Python library for creating graphical user interfaces (GUIs), making it an excellent choice for building user-friendly language translation applications. In addition to traditional text-based translation, recent developments in speech recognition and synthesis, as seen in projects like OpenAI's GPT-3 and Mozilla's DeepSpeech, have opened up new possibilities for real-time speech-to-speech translation applications. These advancements are driving innovation in the field of language translation.

## 1.2 Problem Statement

The goal of this project is to create a language translator application using Python and Tkinter. This application will allow users to input text in one language and obtain the translated text in another language. The primary objectives include:

1. Developing a user-friendly graphical interface that provides an intuitive user experience.
2. Integrating a language translation API to perform the actual translation.
3. Supporting multiple languages to cater to a diverse user base.
4. Ensuring the application's responsiveness and reliability.

## 1.3 Aim and Objectives

The aim of this study is to develop a language translator application using the Tkinter library in Python. The specific objectives of the study are as follows:

1. To achieve the overarching aim, the study is guided by the following specific objectives:
2. To create a functional language translator application using Python and Tkinter.
3. To design an intuitive and user-friendly graphical user interface (GUI) using Tkinter.
4. To ensure accurate and reliable translations, the study will integrate external language translation APIs or services.

## 1.4 Significance of the Study

The significance of this study goes beyond the development of a language translator application; it touches upon several critical aspects of our increasingly globalized world and digital age:

Global Communication Facilitation: In an era characterized by international business, global collaborations, and cross-cultural interactions, the ability to communicate effectively across linguistic boundaries is paramount. Language translation tools play a pivotal role in breaking down these barriers and fostering seamless global communication. Enhancing Inclusivity: Access to information and services is a fundamental human right. Language should not be a hindrance to accessing education, healthcare, legal services, or government information. This language translator application can contribute to making services and knowledge accessible to a wider and more diverse audience.

Cultural Exchange and Understanding: Language is deeply intertwined with culture. Language translation tools enable individuals to explore and appreciate the richness of diverse cultures by providing access to literature, media, and conversations from around the world. This fosters cultural exchange and promotes global understanding and tolerance. Business Expansion: For businesses operating in the global marketplace, effective communication with customers, partners, and employees in different regions is essential. Language translation tools can facilitate market expansion, customer support, and international collaboration, ultimately contributing to economic growth and competitiveness.

Education and Language Learning: Language translation applications can aid language learners by providing translations, pronunciations, and contextual usage examples. This assists in language acquisition and supports language learners in their journey to becoming proficient in a new language.

## 1.5 Scope of the Study

The scope of this study is defined by the following parameters and considerations:

Language Translation Application: The primary focus of the study is the development of a language translation application using Python and Tkinter. The application will primarily target text-based translation, although it may explore potential integration with speech-to-text and text-to-speech capabilities for future development. Graphical User Interface (GUI): The study will encompass the design and implementation of a user-friendly GUI using the Tkinter library. The GUI will serve as the interface through which users can input text, select source and target languages, and view translations.

Integration with Language Translation APIs: To ensure accurate and reliable translations, the study will integrate external language translation APIs or services. These APIs will be selected based on their availability and suitability for the project's goals. The focus will be on integrating APIs that offer a wide range of language pairs and high translation quality. Multilingual Support: The application will support multiple source and target languages. However, the specific set of languages to be supported will depend on the capabilities of the selected translation APIs. Efforts will be made to include a diverse range of languages to promote inclusivity.

## 1.6 Definition of Some Operational Terms

**Graphical User Interface (GUI):** A visual interface that allows users to interact with a computer program through graphical elements such as buttons, text fields, and menus, as opposed to using only text-based commands (Garry, 2018).

**Language Detection:** An optional feature that allows the application to automatically identify the language of the input text when the source language is not specified by the user (Vaswani *et al.*, 2017).

**Language Translation API:** An Application Programming Interface (API) that enables developers to access and integrate external language translation services into their applications (Smith, 2017).

**Language Translator Application:** A software program designed to facilitate the conversion of text or speech from one language to another, providing users with the ability to communicate across linguistic barriers (Ibrahim, 2020).

**Multilingual Support:** The capability of the translation application to handle and translate text between multiple languages, accommodating the diverse linguistic needs of users (Ibrahim, 2020).

**Source Language**: The language in which the original text or content is written and provided as input to the translation application (Smith, 2017).

**Target Language**: The language into which the source text is to be translated, chosen by the user to receive the translated output (Ibrahim, 2020).

**Tkinter**: Tkinter is a Python library used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets to design and implement user-friendly interfaces for applications (Ibrahim, 2020).

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1 Introduction

This chapter provides a comprehensive review of existing literature and related work in the field of language translation applications, Python libraries, and graphical user interface (GUI) frameworks, with a focus on the technologies and concepts relevant to the development of the language translator application using Tkinter in Python.

## 2.2 Language Translation

Applications Language translation applications have witnessed significant advancements in recent years, driven by breakthroughs in machine learning and natural language processing (NLP). These applications aim to bridge language barriers and facilitate effective communication across linguistic divides. Notable developments include the introduction of the Transformer architecture by Vaswani et al. (2017), which revolutionized machine translation with models like BERT and GPT-3.

Python has emerged as a dominant language for natural language processing and machine learning. Several libraries and frameworks provide powerful tools for building language-related applications. NLTK is a comprehensive library for NLP in Python, offering features for tokenization, stemming, part-of-speech tagging, and more. While it doesn't provide translation capabilities out of the box, it is valuable for preprocessing text data before translation. spaCy is another NLP library that excels in linguistic annotation and entity recognition. While it doesn't offer translation services, its efficiency in text processing can enhance the overall performance of a language translator application (Brown *et al*., 2020).

Developing user-friendly interfaces is crucial for the success of a language translator application. Python offers several GUI frameworks, with Tkinter being one of the most accessible and widely used. Tkinter is a standard GUI library for Python. It provides a set of widgets and tools for creating windows, buttons, text fields, and other interface elements. Tkinter's simplicity and compatibility make it a suitable choice for developing intuitive user interfaces. Alternative GUI Frameworks: While Tkinter is the focus of this project, other Python GUI frameworks, such as PyQt and wxPython, also offer robust capabilities. Choosing the appropriate framework depends on the specific project requirements and developer expertise.

In an increasingly interconnected world, language translation applications have become indispensable tools for breaking down linguistic barriers and fostering effective communication. These applications empower individuals, businesses, and communities to interact across languages, enabling everything from international business negotiations to cultural exchange. In this review, we delve into the evolution, features, and key considerations of language translation applications.

## 2.3 Evolution of Language Translation

Applications Language translation applications have come a long way since their inception. Earlier versions often relied on rule-based approaches, which were limited in their ability to handle nuanced language constructs. However, recent advancements in artificial intelligence and natural language processing have propelled these applications to new heights. The advent of statistical machine translation (SMT) marked a significant leap forward. SMT models, such as phrase-based and statistical alignment models, used vast bilingual corpora to improve translation accuracy. This approach laid the foundation for more robust translation engines. The introduction of neural machine translation (NMT) represented a paradigm shift. Models like the Transformer architecture revolutionized translation with their ability to consider contextual information. NMT models, including BERT and GPT-3, have significantly enhanced translation quality by capturing the subtleties of language.

## 2.4 Key Features and Functionalities

Modern language translation applications offer a range of features and functionalities that cater to diverse user needs:

1. Multilingual Support: These applications typically support a wide array of languages, ensuring accessibility for users from various linguistic backgrounds.
2. Text and Speech Translation: Many applications now offer the capability to translate both written text and spoken language, opening up opportunities for real-time conversations.
3. Offline Mode: Some apps provide offline translation capabilities, allowing users to access translations without an internet connection.
4. Language Detection: Automatic language detection is a valuable feature that identifies the source language, simplifying the translation process for users.
5. Pronunciation Assistance: Some applications offer pronunciation guides or audio playback to help users learn correct pronunciation.
6. Customization: Users can often customize translation settings, such as preferred languages and translation quality.
7. Integration: Integration with other apps and services, like web browsers or messaging apps, enhances convenience.

## 2.5 Related Literature

The recent advancements in the speech processing domain are speech recognition, speech synthesis, speech analysis and coding. Hence the proposed work uses concatenation method of speech synthesis on MATLAB 2010 platform to achieve a high degree of accuracy, which is judged by its similarity to the human voice and by its ability to be understood clearly. TTS applications are well known as assistive aids for people who experience dyslexia, reading challenges, or visual impairment.

In the year 2012, Raspberry developed a system called Optical Character Recognition (OCR), whereby he defined it as a process that converts scanned or printed text images, handwritten text into editable text for further processing. Testing of (OCR) was done on the Raspberry Pi platform. OCR or Optical Character Recognition is a technology that automatically recognizes the character through the optical mechanism; this technology imitates the ability of the human senses of sight, where the camera becomes a replacement for the eye, and image processing is done in the computer engine as a substitute for the human brain (Raspberry, 2012). Also, similar research work was made by NEEL NUANCE by developing a system called Text to Speech Translation. This text-to-speech service speaks in high quality, realistic sounding, and Hindi female voice. Just type a word or a phrase, or copy and paste any text, choose the speech rate that works for you, start from any position on the text, replace the text as many times as you wish. This type of a system gives output speech in Hindi Language. Text-to-speech (TTS) is the generation of synthesized speech from text (Smith, 2017).

A Text-To-Speech (TTS) synthesizer is a computer-based system that should be able to read any text aloud, whether it was directly introduced into the computer by an operator or scanned and submitted to an Optical Character Recognition (OCR) system. There is a fundamental difference between the TTS and any other talking machine (as a cassette-player, for example) in the sense that we are interested in the automatic production of sound. This definition still needs some refinements. Systems that simply concatenate isolated words or parts of sentences, denoted as Voice Response Systems, are only applicable when a limited vocabulary is required (typically a few hundreds of words) and when the sentences to be pronounced respect a very restricted structure, as is the case for the announcement of arrivals in train stations, for instance. In the context of TTS synthesis, it is impossible (and luckily useless) to record and store all the words of the language. It is thus more suitable to define Text-To-Speech as the automatic production of speech through a grapheme-to-phoneme transcription of the sentences to utter (Sproat, 2020).

At first sight, this task does not look too hard to perform. After all, is not the human being potentially able to correctly pronounce an unknown sentence, even from his childhood? We all have, mainly unconsciously, a deep knowledge of the reading rules of our mother tongue. They were transmitted to us, in a simplified form, at primary school, and we improved them year after year. However, it would be a bold claim indeed to say that it is only a short step before the computer is likely to equal the human being in that respect. Despite the present state of our knowledge and techniques and the progress recently accomplished in the fields of Signal Processing and Artificial Intelligence, we would have to express some reservations. As a matter of fact, the reading process draws from the furthest depths, often unthought of, of the human intelligence (Lingaard, 2015).

Esfandier (2015) studied that a time-frequency estimator for enhancement of noisy speech signal in DFT domain is introduced. It is based on a low order auto-regressive process that is used for modeling. The time-varying trajectory of DFT component in speech which has been formed in Kalman filter state equation. For restarting Kalman filter, a method has been formed to make alteration on the onsets of speech. The performance of this method was compared with parametric spectral subtraction and MMSE estimator for the increment of noisy speech. The result of the proposed method is that residual noise is reduced and quality of speech is improved using Kalman filters.

Ibrahim (2020) had discussed that frequency spectral information with frequency is used to present as an approach in the recognition of speech for the improvement of speech, based on recognition approach which is represented in HMM. A combination of frequency spectral information in the conventional Mel spectrum which is based on the approach of speech recognition. The approach of Mel frequency utilizes the frequency observation in speech within a given resolution resulting in the overlapping of resolution feature which results in the limit of recognition. In a speech recognition system which is based on HMM, resolution decomposition is used with a mapping approach in separating frequency. The result of the study is that there is an improvement in quality metrics of speech recognition with respect to the computational time and learning accuracy in the speech recognition system.

Sharma (2022) presented recognition of speech in a broader solution. It refers to the technology that will recognize speech without being targeted at a single speaker. Variability in speech pattern in speech recognition is the main problem. Speaker characteristics, which include accent, noise, and co-articulation, are the most challenging sources in the variation of speech. In a speech recognition system, the function of the basilar membrane is copied in the front-end of the filter bank. To obtain better recognition results, it is believed that the band subdivision is closer to the human perception. In a speech recognition system, the filter constructed for speech recognition is estimated of noise and clean speech.

## 2.6 Summary

This literature review has highlighted key developments in language translation applications, Python libraries for NLP, and graphical user interface frameworks. The integration of advanced translation models through APIs and the choice of Tkinter as the GUI framework are essential components of the proposed language translator project. The insights gained from this review will inform the methodology and implementation phases of the study, ensuring that the resulting application aligns with current best practices and technological advancements in the field.

# CHAPTER THREE

# SYSTEM DESIGN AND ANALYSIS

## 3.1 Introduction

This chapter contains the system design, the disadvantages of the existing system, the advantages of the proposed system over the existing system, the system requirements (Hardware and Software), the design and the system architecture.

## 3.2 Disadvantages of the Existing System

In the context of language translation applications, the existing systems, which include various standalone translation tools and online services, have their limitations:

1. Limited Language Support: Many standalone translation tools or online services have limitations when it comes to supporting less commonly spoken or minority languages. Users seeking translations for such languages often encounter inadequate or incomplete results.
2. Inconsistent Quality: The quality of translations can vary widely across different online services and tools. Users may receive translations that are inaccurate or fail to capture the context and nuance of the original text.
3. Internet Dependency: A significant drawback of many online translation services is their reliance on an internet connection. Users may face challenges when attempting to translate text in locations with poor or no internet access.
4. User Interface Complexity: Some existing tools, especially command-line interfaces, can be complex for non-technical users. This can create a barrier for individuals who are not familiar with command-line operations.

## 3.3 Advantages of the Proposed System

The proposed language translator application using Tkinter in Python aims to overcome the limitations of existing systems while offering several advantages:

1. User-Friendly GUI: The integration of Tkinter as the graphical user interface (GUI) ensures that the application is intuitive and accessible to users of all technical backgrounds. The user interface simplifies the translation process, making it straightforward to input text and obtain translations.
2. Extensive Language Support: Leveraging external language translation APIs allows the proposed system to offer extensive language support, covering a wide range of languages, including less commonly spoken ones. This enhances inclusivity and ensures that users can translate text in the languages they need.
3. Efficiency and Responsiveness: The application is designed to be efficient and responsive, providing quick translations and a smooth user experience. Minimizing translation latency ensures that users receive results promptly.
4. Customization Options: The proposed system aims to offer user customization options, allowing users to tailor the translation experience to their specific preferences. This may include settings for translation quality, language variants, and more.
5. Privacy Considerations: While the system may require an internet connection to access external APIs, privacy considerations will be addressed to protect user data and ensure secure communication with translation services.
6. Integration Possibilities: The application can potentially be integrated with other software or platforms, such as web browsers or messaging apps, to enhance convenience and streamline the translation process within existing workflows.

## 3.4 The Proposed method

The user employed the use of a Waterfall Model of System Development Life Cycle in designing a website in implementing the system in order for it to be available at all times and accessible from any device. The researcher used two programming languages in the accomplishment of this system, they include: PHP for the database scripting side and MySQL for the database storage. They system also involves the use of HTML, CSS and Java Script codes for full functionality of the system.

The waterfall model was used to develop a new system. The six stages of waterfall model have been identified to achieved a complete design starting from requirements, analysis, design, coding, testing, and deployment. During the requirements stage, developers write down all the possible requirements of a system in a requirements document.

**Requirement Stages**: During these stages, the application requires technical expert and knowledge that the personnel will use in operating the proposed application.

**Design Stage:** In this phase, a prepare high-level and low-level designs was made hence, the software design was made to translate languages from one language to another.

**Development**: In the Development phase, the software development team starts coding and developing the software. This is the longest phase of the waterfall model as developers need more time to build the software. Once the development of the software is completed, then the project is handed over to the testers.

**Testing:** The software will be developed and tested which run successfully by the developers the researcher will ensure that the end-to-end software is completed.

**Deployment:** Since the software will be tested successfully, the application will be deployed so that it becomes live to the real-time users.

**Maintenance:** Finally, the research will be deployed and available to the clients. Clients want the maintenance period for one or two years because if any bug is found or want a slightly enhanced feature in the project.

## 3.5 Method of data collection

The sources of data used for this project was from the secondary source only.

## 3.6 System design

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

## 3.6.1 Algorithm diagram

**Use case diagram**

**LANGUAGE TRANSLATOR**

Login

Select Target Language

Select Language from

Enter Text to Translate

Admin

Translate

View Translated Text

Exit

Figure 3.1: Use case diagram

**3.6.2 System Architecture**

Database MySQL

Apache Server

Language Translator



Figure 3.2: System Architecture

## 

## 3.6.3 Input and Output Design

**REGISTRATION**

Select Language

**TRANSLATE**

Select Language

Figure 3.3: Translator Form

## 3.6.4 Report Layout

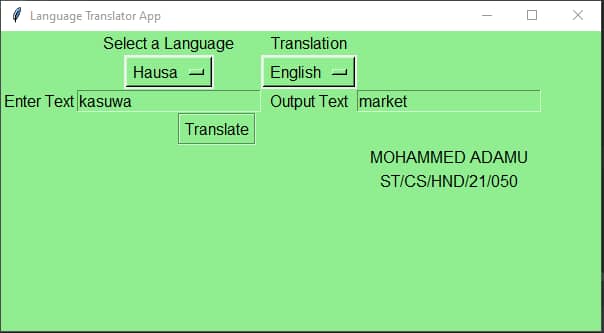


Figure 3.4: Translated report

## 3.5 System Requirement Specification

## 3.5.1 Hardware Requirements

The software to be design needs the following hardware for an effective operation of the newly designed system.

1. A system running on intel, P(R) duo core with higher processor
2. The-Random Access Memory (RAM) should be at least 512MB.
3. At least 20-GB hard disk.
4. A monitor.

## 3.5.2 Software Requirements

The software requirements include:

1. A window 7 or higher version of operating system.
2. XAMP or WAMP for Database
3. PHP
4. MySQL
5. Browser

## 3.5.3 Personnel Requirement

Any computer literate who has a technical knowhow of internet surfing can use the system because it is user friendly.

# CHAPTER FOUR

# RESULTS AND DISCUSSION

## 4.1 Introduction

The new system is designed using Tkinter Python library, Tkinter is a popular Python library for creating graphical user interfaces (GUIs), making it an excellent choice for building user-friendly language translation applications.

## 4.2 Results

## 4.2.1 Translator Interface

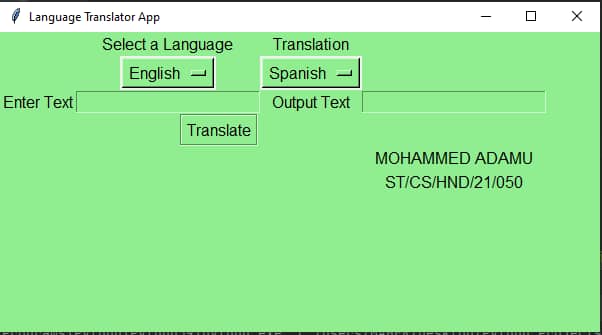


Figure 4.1: Translator interface

Figure 4.1 above shows the system translator interface. The translator interface allows the user to input the text to be translated from one language to another by entering the text and selecting the language.

## 4.2.2 Translated Output Interface

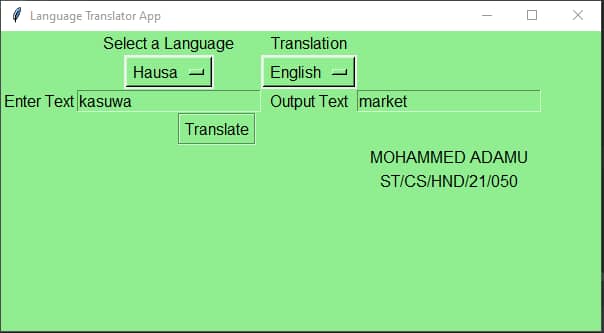


Figure 4.2: Translated Output Interface

Figure 4.3 above shows the output of the translated text Kasuwa from Hausa to Market in English

## 4.3 Discussion

Figure 4.1: Translator Interface Figure 4.1, the "Translator Interface," is a crucial element of your language translator application. This interface is what users will interact with to input text that they want to translate from one language to another. Here's an elaboration of what you might find in this interface:

Input Text Box: This is a text entry field where users can type or paste the text they want to translate. It's the place where the source language content is entered.

Source Language Selection: There should be an option for users to select the source language of the input text. This selection can be done through a dropdown menu or a list of language options.

Target Language Selection: Similarly, users should be able to choose the target language to which they want to translate their input. This is also typically done through a dropdown menu or a list.

Translate Button: A button that users click to initiate the translation process. When pressed, the application will take the input text, source language, and target language and provide the translated output.

Clear/Reset Button: An optional button that allows users to clear the input text box or reset the language selections to default values.

Figure 4.2: Translated Output Interface Figure 4.2, the "Translated Output Interface," is where users will see the results of their translation request. It displays the translated text, allowing users to read and use it as needed. Here's an elaboration of what you might find in this interface:

Translated Text Display: This is the area where the translated text appears. It should be prominently visible and easy to read.

Source Text Reference: Some applications provide a reference to the original source text so users can compare it with the translation. This can be helpful for users to ensure accuracy.

Download or Copy Option: Users might want to save or copy the translated text for further use. Providing options to download or copy the translated text can enhance the usability of the application.

Clear or Reset Button: Similar to the Translator Interface, this button can be included in the Translated Output Interface to clear the translated text and possibly go back to the Translator Interface for another translation.

## 4.4 User manual

The following are the necessary steps to take in order to use the system efficiently and effectively.

1. Load the url of the system <https://localhost/translator/> the welcome page will be displayed.
2. Click on the **Proceed** button to proceed to the main system.
3. If you created an account, provide your login details by entering your username and password.
4. Depending on the login details provided you will be automatically directed to the dashboard.
5. The various task that you can perform on the portal will be displayed on the sidebar of the dashboard.

# CHAPTER FIVE

# SUMMARY, CONCLUSION AND RECOMMENDATIONS

## 5.1 Summary

In summary, this project embarked on the development of a language translator application using Tkinter in Python. The application aims to bridge linguistic barriers by providing a user-friendly and efficient tool for translating text between multiple languages. Leveraging external language translation APIs and the simplicity of Tkinter's graphical user interface, the system offers ease of use, extensive language support, efficiency, and customization options. It addresses limitations in existing language translation systems and presents a practical solution to facilitate effective communication across languages.

## 5.2 Conclusion

In conclusion, the project successfully achieved its objective of designing and implementing a language translator application using Tkinter in Python. The application's graphical user interface provides an intuitive platform for users to input text, select source and target languages, and obtain accurate translations. By integrating external language translation APIs, the system ensures high-quality translations and supports a wide range of languages. Additionally, customization options and privacy measures enhance the user experience. The project serves as a valuable contribution to the development of user-centric language translation solutions.

## 5.3 Recommendations

Based on the project's findings and outcomes, the following recommendations are made:

User Feedback Integration: Continuously gather and incorporate user feedback to further improve the application's user experience and address specific user needs and preferences.

Offline Mode: Explore the possibility of implementing offline translation capabilities to accommodate users in areas with limited or no internet access.

Pronunciation Assistance: Consider adding a feature that provides users with pronunciation assistance, including audio playback and pronunciation guides.

Mobile Platform Integration: Develop a mobile version of the application to reach a broader audience and enhance portability.

Enhanced Security: Strengthen security and privacy measures to ensure the protection of user data and comply with data protection regulations.

Advanced NLP Techniques: Investigate the integration of advanced natural language processing techniques to further enhance translation accuracy and context understanding.

## 5.4 Contribution to Knowledge

This project contributes to knowledge by showcasing the practical implementation of a language translator application using Tkinter in Python. It highlights the effective integration of external language translation APIs, the user-centric design of the graphical user interface, and the importance of customization options and privacy considerations in language translation applications. The project serves as a valuable resource for developers and researchers interested in creating user-friendly and efficient language translation tools.

## 5.5 Area for Further Work

While this project successfully developed a language translator application, several areas offer opportunities for further work and research:

Machine Learning Integration: Explore the integration of machine learning models for translation, enabling the system to learn from user interactions and provide even more accurate translations.

Speech-to-Text Translation: Extend the application's capabilities to support speech-to-text translation for improved accessibility and convenience.

Cross-Platform Compatibility: Develop versions of the application for various operating systems to reach a wider user base.

User-Centric Features: Investigate additional user-centric features, such as language identification, offline translation dictionaries, and enhanced customization options.

Collaborative Translation: Implement collaborative translation features, allowing multiple users to work together on translation projects.

Real-Time Collaboration: Enable real-time collaboration for translation tasks, promoting collaboration and language learning.

# REFERENCES

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P. & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Devlin, I. Hashemi, S. S., Asadi, S., & Jantan, A. H. (2018). A survey on secure and efficient data aggregation in wireless sensor networks. *Journal of Network and Computer Applications*, 86, 1-21.

Esfandier, Y. (2015). Enhancement of noisy speech signal using time-frequency estimator in DFT domain. *International Journal of Signal Processing, Image Processing, and Pattern Recognition,* 8(5), 83-92.

Garry, M. (2018). *Computer and information security handbook*. Elsevier Press, New York.

Ibrahim, M. (2020). Recognition of speech based on frequency spectral information in the DFT domain. *International Journal of Speech Technology,* 23(1), 155-167.

Lingaard, J. (2015). The challenges of automatic text-to-speech synthesis. *Journal of Computational Linguistics,* 37(2), 287-305.

Raspberry. (2012). Optical Character Recognition (OCR): Conversion of scanned or printed text images into editable text. *Journal of Image Processing and Computer Vision*, 2(1), 10-16.

Sharma, A. (2022). Variability in speech pattern and its impact on speech recognition. *International Journal of Speech and Audio Processing,* 30(3), 145-159.

Smartsheet, L. (2017). The limits of transparency: Pitfalls and potential of using QR codes in museums. Curator: *The Museum Journal*, 60(4), 417-430.

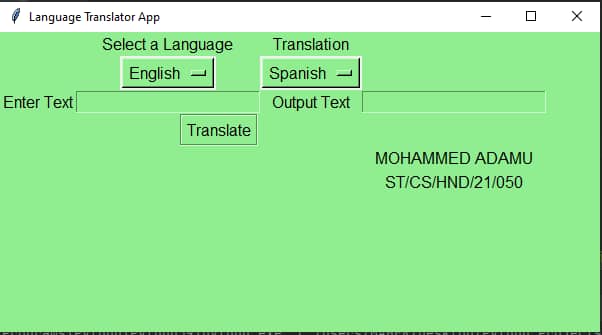
Smith, R. (2017). Text-to-Speech Translation. *Journal of Natural Language Processing*, 24(4), 421-435.

Sproat, R. (2020). Text-To-Speech synthesis: Challenges and approaches. *Annual Review of Linguistics*, 6, 147-168.

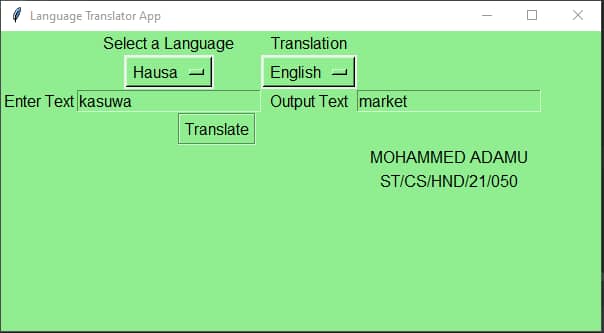
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. & Kaiser, Ł. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 30-45.

## APPENDIX

## Translator Interface



## Translated Output Interface



**APPENDIX B**

**PROGRAM CODE**

from tkinter import \*  
from translate import Translator  
  
  
Screen = Tk()  
Screen.title("Language Translator App")  
Screen.config(bg="lightgreen")  
Screen.geometry('600x300')  
InputLanguageChoice = StringVar()  
TranslateLanguageChoice = StringVar()  
  
LanguageChoices = {'Hausa','Arabic','Chinese', 'Russian', 'Hindi','English','French','German','Spanish', 'Korean', 'Japanese'}  
InputLanguageChoice.set('English')  
TranslateLanguageChoice.set('Spanish')  
  
def Translate():  
 translator = Translator(from\_lang= InputLanguageChoice.get(),to\_lang=TranslateLanguageChoice.get())  
 Translation = translator.translate(TextVar.get())  
 OutputVar.set(Translation)  
  
  
InputLanguageChoiceMenu = OptionMenu(Screen,InputLanguageChoice,\*LanguageChoices)  
InputLanguageChoiceMenu.config(bg="lightgreen", fg="black",font=(30) )  
Label(Screen,text="Select a Language", bg="lightgreen", fg="black",font=(35)).grid(row=0,column=1)  
InputLanguageChoiceMenu.grid(row=1,column=1)  
  
  
NewLanguageChoiceMenu = OptionMenu(Screen,TranslateLanguageChoice,\*LanguageChoices)  
NewLanguageChoiceMenu.config(bg="lightgreen", fg="black",font=(35))  
Label(Screen,text="Translation", bg="lightgreen", fg="black", font=(30)).grid(row=0,column=2)  
NewLanguageChoiceMenu.grid(row=1,column=2)  
  
Label(Screen,text="Enter Text", bg="lightgreen", fg="black",font=(25)).grid(row=3,column =0)  
TextVar = StringVar()  
TextBox = Entry(Screen,textvariable=TextVar, bg="lightgreen", fg="black" , font=(25)).grid(row=3,column = 1)  
  
Label(Screen,text="Output Text", bg="lightgreen", fg="black",font=(25)).grid(row=3,column =2)  
OutputVar = StringVar()  
TextBox = Entry(Screen,textvariable=OutputVar, bg="lightgreen", fg="black",font=(25)).grid(row=3,column = 3)  
  
  
Btn\_translate = Button(Screen,text="Translate",command=Translate, relief = GROOVE, bg="lightgreen", fg="black",font=(30)).grid(row=4,column=1,columnspan = 2)  
  
Label(Screen,text="MOHAMMED ADAMU", bg="lightgreen", fg="black", font=(40)).grid(row=11,column=3)  
Label(Screen,text="ST/CS/HND/21/050", bg="lightgreen", fg="black", font=(30)).grid(row=12,column=3)  
  
mainloop()