A REPORT
ON

DEVELOPING A SOFTWARE THAT CAN TRANSLATE RESOURCE MATERIAL AND OTHER TEXTS FROM ENGLISH TO OTHER INDIAN REGIONAL LANGUAGES.

Submitted by,

Ms.SAMBAVI.B - 20211ISR0004
Ms.LEKHANA.M -20211ISR0092

Under the guidance of,

Ms. DEEPTHI .S

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship/Project report **"DEVELOPING A SOFTWARE THAT CAN RESOURCE MATERIAL AND OTHER TEXTS FROM FOREIGN TO INDIAN LANGUAGES "** being submitted by "SAMBAVI B LEKHANA M" bearing roll number"20211ISR0004 20211ISR0092 " in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Ms. <DEEPTHI.S>
ASSISTANT PROFESSOR
PSCS / PSIS
Presidency University

Dr. <ZAFAR ALI KHAN>
PROFESSOR & HoD
PSCS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
PSCS
Presidency University

Prof (Dr).Md. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean –PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled "**DEVELOPING A SOFTWARE THAT CAN RESOURCE MATERIAL AND OTHER TEXTS FROM FOREIGN TO INDIAN LANGUAGES"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of my own investigations carried under the guidance of Mrs.DEEPTHI.S**, ASSISTANT PROFESSOR, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAME	ROLL NO	SIGNATURE
SAMBAVI B	20211ISR0004	
LEKHANA M	20211ISR0092	

INTERNSHIP COMPLETION CERTIFICATE

- **The certificate issued from an organization must have the duration of the Internship, i.e.start and end date, project title and a technology on which work is carried out.**
-

ABSTRACT

This project entails creating a Language Translator Application in Python aimed at offering users an intuitive application for translating text across several languages. The application takes advantage of the `google-trans` library, which uses the Google Translate API, to deliver precise, real-time translations. Developed with a graphical user interface (GUI) based on the `t-kinter` library, the application offers an uninterrupted user experience for translating text from one language to another. The GUI has several components, such as text input and output spaces, drop-down lists for selecting languages, and a button to initiate the translation process.

The user interacts with the program by entering the desired text into an input field, choosing the source language from a drop-down, and selecting the target language. After the user clicks the "Translate" button, the program translates the text with the Google Translate service and outputs the result in a text box. The program handles a large number of languages, so translations between most language pairs are supported. The language choices are shown in a user-friendly format, so the interface is easy to use for everyone, no matter what their technical expertise.

This project report outlines the creation of a desktop program for translating text from and to various leading Indian languages and English. Leveraging the Tkinter library for the graphical user interface (GUI) and the google-trans library for translation services, this program is designed to be an easy-to-use and accessible translation tool.The application consists of a welcome screen and a single-purpose translation interface with easy-to-use languag selection drop-downs and input/output textareas. The report gives an overview of the motivation behind the project, highlights gaps in research in current translation approaches, gives an explanation of the system design and implementation, gives the objectives of the project, gives the method and methodology adopted, gives a literature survey of applicable translation technologies, gives the expected outcomes of the project, gives the possible outcomes.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science and Engineering, Presidency University, and **Dr. "ZAFAR ALI KHAN"**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Ms. DEEPTHI.S ASSISTANT PROFESSOR** and Reviewer **Ms. SMITHA S.P,ASSISTANT PROFESSOR**, Presidency School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the PIP4001 Internship/University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators "NAME" and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

SAMBAVI B

LEKHANA M

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

India's rich linguistic diversity necessitates effective tools for cross-lingual communication. Although web-based translation services are common, a specialized desktop application can provide a more concentrated and possibly more convenient solution for users who need translations between English and Indian languages on a routine basis. This project fills this void by creating a standalone program using the strengths of the google-trans library within a specially designed Tkinter GUI. The software seeks to make the translation process simple and accessible to a larger user base. This report chronicles the development process, from the realization of the need to the expected outcomes and possible results.

In the era of digital communication, language communication has become more crucial as global interactions, business, and travel have increased. To bridge language gaps, tools and technologies have been created to offer rapid and precise translations. One of the most popular solutions to this end is machine translation, driven by complex algorithms and extensive language databases. This project aims to develop a Language Translator Application using Python that is meant to give users an intuitive, accessible, and efficient means of translating text across languages. Leveraging the strength of the `google-trans` library, this project allows for real-time translation while providing a easy-to-use graphical interface with the `tkinter` library.

1.1 The Indian Linguistic Landscape and the Requirement for Translation:

India is a linguistic mosaic, with numerous languages and dialects numbering in the hundreds. Although English is an essential language, the majority of people primarily speak their local languages. This presents a huge obstacle to information, education, and services offered in English. The digital age has further amplified this gap, as a vast amount of online content remains inaccessible to those who are not proficient in English. Therefore, the development of robust and accurate translation software is not merely a technological endeavor but a crucial step towards inclusive growth and empowerment.

1.1.1 The Need for Language Translation Tools

With the world becoming more and more connected, the need for efficient language translation tools is greater than ever. For travel, business, or communication among individuals from different cultures, language barriers can create huge problems. Conventional techniques of translation like using human translators or dictionaries are time-consuming and may not always be feasible. Therefore, automated translation software has thus come as a quick and efficient solution. Machine translation technologies, like Google Translate, have the benefit of making instantaneous translations of many languages possible, making them a treasure trove for any one who requires instant, trust-worthy communication. This project sets out to respond to the necessity of such applications by creating an easy yet powerful application that is capable of facilitating seamless cross-language communication.

1.2 Solving the Problems of Indian Language Translation

English-Indian language translation is special because of the variety of linguistic features of the target languages. They pose the following challenges:

1.2.1 Morphological Richness: Indian languages are rich in morphology, with complex grammatical forms and large numbers of inflectional forms.

1.2.2 Lexical Divergence: There are major differences in vocabulary and idiomatic expressions between English and Indian languages.Indian languages have a variety of scripts, demanding the use of robust character encoding handling and font rendering.

1.2.3 Contextual dependence: The contextual meaning of phrases and words greatly depends on where they are being used, making sophisticated NLP processing necessary.

1.2.4 Sparsity of parallel data: A high-quality set of parallel corpora for training machine learning models is mostly restricted, especially for less widely spoken languages

Chapter 2

LITERATURE SURVEY

Section	Approach / Tool	Description	Key Contributions / Examples	Advantages	Limitations
2.1 Hutchins, W. J. (2005). *The history of machine translation in a nutshell*.	Traditional Machine Translation (Rule-Based)	Based on manually created linguistic rules and grammar structures.	Georgetown-IBM Experiment	Linguistically informed, domain-specific accuracy	High development cost, limited scalability, poor ambiguity handling
2.2 Koehn, P., Och, F. J., & Marcu, D. (2003). *Statistical phrase-based translation*.	Statistical Machine Translation (SMT)	Uses statistical models trained on bilingual corpora to learn translation probabilities.	Early Google Translate implementation	Scalable, data-driven, less manual effort	Weak in handling context, syntax, and idiomatic expressions
2.3 Bahdanau, D., Cho, K., & Bengio, Y. (2015). *Neural machine translation by jointly learning to*	Neural Machine Translation (NMT)	Uses deep learning models to translate entire sentences by understanding context and semantics.	Google's transition to NMT in 2016	Context-aware, fluent translations, high adaptability with more data	Requires large datasets and computational resources

align and translate.					
2.4 Google Cloud. (n.d.). *Cloud Translation API Documentation*.	Translation APIs (e.g., Google Translate API)	Cloud-based services providing programmatic access to translation engines.	`googletrans` Python wrapper for Google Translate API	Easy integration, accessible to non-experts, supports many languages	Depends on internet connectivity, usage limits or quotas
2.5	Desktop and Web-Based Translation Apps	Software interfaces that provide real-time translation with added features like speech and image translation.	Google Translate, DeepL, Microsoft Translator, this project's app	User-friendly, feature-rich (voice, image, documents), often real-time	Offline functionality is limited, some require internet and subscriptions

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1 Research Gaps in Current Machine Translation (MT) Approaches

In spite of spectacular advances in machine translation, especially with the advent of neural machine translation (NMT), there exist a number of seminal research gaps that hamper the achievement of universally accurate, context-sensitive, and inclusive translation systems. One of the key challenges is the management of linguistic complexity. Most low-resource languages lack adequate parallel corpora and digital resources, leading to inferior translation quality and limited support within MT systems. In addition, morphologically rich languages and languages with flexible syntactic structures present challenges to current models, as they tend not to capture the subtle relationships between words. The proper translation of idioms, colloquialisms, and culturally sensitive expressions is also problematic, as they tend to be reliant on rich contextual and cultural familiarity rather than simple literal meanings. Another important deficiency is in discourse-level translation; most are still, disregarding larger context, which can lead to incoherence and inconsistency in longer texts.

3.2 Context and Ambiguity

Although NMT systems have enhanced contextual perception, they continue to lag in their ability to handle long-range dependencies and disambiguate context-dependent meaning. One of the most active current areas of research is the incorporation of multimodal context—using visual, audio, or other sensory data—to improve translation quality. Additionally, domain-specific translation continues to be an area of difficulty. General-purpose MT tools tend to perform poorly in specialized domains like law, medicine, or tech, where specialized vocabularies and contextual information are needed, requiring specialized training and adaptation methods.

3.3 Ethical and Societal Issues

Ethical issues are also of vital importance in the MT scenario. Machine translation models can, through unintentional bias from training data, generate translations that can be discriminatory or offensive.

Guaranteeing fairness and avoiding bias in MT output is a persistent research interest. Furthermore, the large-scale datasets involved in training raise questions regarding data privacy and security,

necessitating the creation of secure and privacy-preserving translation methods. Inclusivity and accessibility are also critical—translation systems need to be accessible to people despite their technical expertise or disabilities, pointing to the importance of accessible interfaces and assistive functionality. Environmental sustainability is also a matter of concern since training large language models requires significant energy, emphasizing the importance of more efficient and sustainable models.

3.4 Technological and Implementation Gaps

There are some technological shortcomings as well. Offline MT systems generally perform worse than online ones, especially in translation accuracy, which is a major problem in regions with poor or no internet connectivity. Speech translation in real-time in noisy conditions is another open problem, where noise and interference significantly impair performance. Current evaluation metrics such as BLEU and METEOR are not effective in capturing actual translation quality and human judgment, which suggests the need for more sophisticated, human-oriented evaluation criteria. In addition, the absence of model transparency is still a hindrance; it is important to understand and explain model choices to debug and trust. There is also a big push towards building lightweight, efficient models that require fewer resources to deploy. Finally, seamlessly incorporating human feedback into the MT process is an issue that continues to puzzle researchers as they seek the best ways for humans to enhance and correct machine-generated output in real time.

Chapter 4

PROPOSED MOTHODOLOGY

4.1 Technology Stack and Tool Choice

The Language Translator Application development is based on a well-chosen stack of technologies and tools to make it flexible, reliable, and easy to use. The central programming language employed is Python due to its simplicity, rich library support, and applicability for both frontend and backend development. Python's comprehensive ecosystem enables fast application development and easy integration with third-party APIs.

In order to support the translation functionality, the application employs the googletrans library, which is a Python wrapper of the unofficial Google Translate API. This enables the app to tap into Google's strong translation engine, which covers more than 100 languages. This obviates the necessity of developing a custom translation model, thereby substantially lowering development complexity without compromising translation quality.

Tkinter is utilized for the user interface. It is Python's default GUI package and enables programmers to develop strong, cross-platform desktop interfaces. Tkinter offers components like buttons, labels, text fields, and dropdowns, which are necessary for the development of an intuitive layout. The requests library is also employed to send HTTP requests to the API, while JSON parsing is employed to retrieve and process the data provided by the translation service. Collectively, these tools constitute a robust and efficient development environment for the translator application.

4.2 Graphical User Interface (GUI) Design

The interface of the Language Translator Application is designed with utmost care to focus on usability and simplicity to make it accessible to all users, including those with minimal technical knowledge. The interface consists of a text input field where users can input the text to be translated. This is succeeded by two language choice dropdown lists—one for the source and another for the target language. Both dropdowns present all languages translated by Google Translate, providing room for flexibility and broad language coverage.

A major "Translate" button initiates the core function of the app. When pressed, it causes the translation action to be kicked off, with the user's input and chosen languages being dispatched to the backend logic. The translated text is rendered in a read-only text output field, which gets updated

dynamically after the translation is received. There is also a "Clear" button available to clear both input and output fields so that users can start a new translation session quickly. The layout and interaction design make the application functional as well as easy to use even for users who are starting for the first time.

4.3 Translation Process Workflow

The translation functionality creates the application's functional logic core. It begins with input from the user, where the user types in the text to be translated in the targeted input box and chooses the source and destination languages via the dropdown menus. Upon a click of the "Translate" button, the app sends a request to the Google Translate API via the googletrans library. This library makes it easier to communicate with the API by structuring the text and language codes into an appropriate HTTP request.

When the request is received, the Google Translate API interprets the input and sends back a translation response in JSON, which includes the translated text as well as some extra metadata. The application parses the JSON response to get the desired data and directly shows the translated text in the output field. This real-time translation process gives users fast and precise translations, and the system can be made to support various forms of input like speech or files in future releases.

4.4 Error Handling and Validation

For the purpose of ensuring reliability and robustness, the application features extensive error handling and input validation. Prior to making any translation request, the application inspects whether the input field is blank or includes invalid characters. In case of invalid input, an error message encourages the user to enter proper and complete data. Additionally, the system checks whether both source and target languages are chosen prior to translating.

Besides local validation, the app also deals with errors that might occur from using the Google Translate API, i.e., network errors, access limits on APIs, or unrecognized language pairs. On such a failure, the application catches such an error kindly and presents the user with an error message friendly enough to specify what happened wrong and how the issue could be fixed. With this comprehensive framework of validations, user experience increases, and the application stays steady under diverse run conditions.

Chapter 5

OBJECTIVES

The Language Translator Application is intended to offer a convenient and efficient solution for translating text among various languages. The major goals of the project are as follows:

5.1. To Create a Simple-to-Use Translation Tool

The application is designed to offer a simple interface for individuals to translate text among various languages easily. By utilizing Python and libraries like`tkinter` for the graphical user interface (GUI) and `googletrans` for live translation, the application seeks to reduce the complexity usually found in translation tools. This will render the translation process accessible to users with no technical background.

5. 2. To Support Multiple Languages

One of the main goals is to support a broad variety of languages for translation, rendering the tool-flexible and accessible to individuals of various linguistic backgrounds. The Google Translate API, accessed through the `google-trans` library, supports more than 100 languages, which guarantees that the application can satisfy the translation requirements of a global population.

5. 3. To Offer Accurate and Real-Time Translations

The aim is to make sure that the translation process is not only fast but also accurate. Through utilizing the Google Translate API, fueled by sophisticated machine translation models such as Neural Machine Translation (NMT), the app is designed to deliver high-quality translations in real-time. The system must be able to process varied sentence structures and contexts, delivering meaningful translations for a range of languages.

5.4. To Develop an Interactive and Reactive User Interface

One of the main goals is to create a user-friendly and reactive graphical user interface that eases enabling users to interact with the system. The simple features of text input boxes, language selector drop-down menus, a translation button, and a display box to show the

output of translated text are included in it. The user interface must be interactive, enabling the users input and translate text with ease.

5.5. To Put in Place Error Handling and Validation Mechanisms

The software needs to be strong and reliable, and hence it should have error-handling mechanisms to allow smooth operation. The intention is to give meaningful feedback when something goes awry, for example, when input fields are empty, when languages are not supported, or when API connectivity fails. Proper validation of user inputs and system responses will be implemented to provide a seamless user experience.

5.6. To Optimize Performance and Minimize Latency

The goal of the project is one of optimizing the performance of the application for real time translation. That involves reducing latency in retrieving translations, ensuring the user gets the translated text as quickly as possible once the request is submitted. Practices such as asynchronous processing, optimized API calls, and caching frequently used translations will be utilized to streamline waiting times and enhance user experience.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

The translation program is intended to give users an easy and elegant experience for translation between languages. The system ensures a straightforward, structured process through the use of a welcome page and stepwise progression through translation. The purpose of the main system is primarily to make operations easy while being able to create accurate translations from an external program like the Googletrans library.

From the design point of view of the system, the application has two principal parts: the user interface (UI) and the back-end logic. The UI is in charge of handling the interactions between the application and the users and is made to be easy and intuitive. Upon starting the application, the users are presented with a welcome page containing a "Continue" button. When this button is clicked, users are taken to the translator page where they can choose the source and target languages through dropdown menus, enter the text to be translated, and click a "Translate" button to start the translation.

All the essential processing logic is done in the backend of the system. It starts by authenticating the input of the user to confirm whether the text as well as selected languages are adequate and not in blank. The app, thereafter, utilizes the Googletrans library, which is a Python shell for the Google Translate API, to carry out the translation process. If translation is successful, the translated words are shown to the user. On error, like an invalid language code or network problem, a suitable error message is displayed, and the user is permitted to re-enter the input or retry.

To enhance this functionality, the application can be developed using Python with a GUI package like Tkinter for the frontend interface. This configuration enables the development of interactive components such as buttons, input boxes, and display boxes. The backend logic in Python takes care of all calls to the translation API function and provides error handling and user feedback.

Overall, the system is modular and maintenance-friendly. Every aspect of the app, from the UI to the translation feature, is self-contained but plays together perfectly to deliver a full user experience. This also facilitates future expansion, for instance, incorporating speech-to-text support, file translation, or integration with more powerful APIs such as Google Cloud Translation API for enterprise use.

1. **Graphical User Interface (GUI)**: Implemented with the Tkinter library, the GUI includes a graphical interface for user interaction. It contains:
2. **Welcome Screen:** A starting screen with application title and a "Continue" button to move to the translator interface.
3. **Language Selection:** Two ttk.Combo-box widgets enable users to choose the source and target language from a per-defined list of English and prominent Indian languages.
4. **Input Area:** A tk.Text widget where the users can type in the text they want translated.
5. **Translate Button:** A tk.Button which initiates the translation when clicked. It provides visual feedback when hovered over.
6. **Output Area** :A tk.Text widget (made non-editable) showing the translated text.
7. **Translation Logic:** Provided inside the translate_text() function, this module makes use of the google trans library:
    - It gets the input text and the chosen target and source languages from the GUI components.
    - It creates a Translator object.
    - It invokes the Translator object's translate() method, providing the input text and language codes.
    - It manages possible translation exceptions with a try-except block and puts an error message in the output area if any.
    - It changes the output text area with the translated text

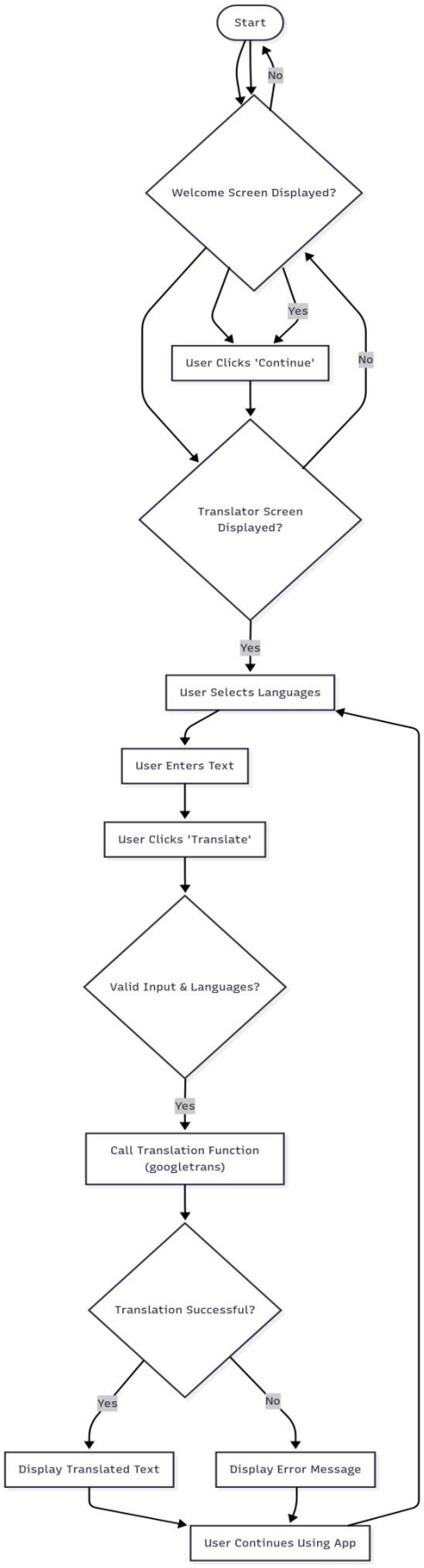


Fig 1.1 Translation App User Flowchart

1. **Start** – Starts with the launching of the application.

    a) **Welcome Screen Displayed** – Checks whether the welcome screen is displayed

        I. If yes, then user clicks on "Continue."
        II. If no, then it continues to loop until the screen gets displayed.

2. **Translator Screen Displayed** – Checks if the translator screen is displayed once "Continue" is clicked

    I. If not, then it reverts back to the earlier decision point.
    II. User Selects Languages – User selects source and destination languages.
    III. User Enters Text – User enters the text they wish to translate.
    IV. User Clicks 'Translate' – Initiates the translation process.

3 Valid Input & Languages – Validates user input and chosen language
4 Translation Successful – Checks whether the translation was successful.

    I. If not, goes back to text entry and language selection.

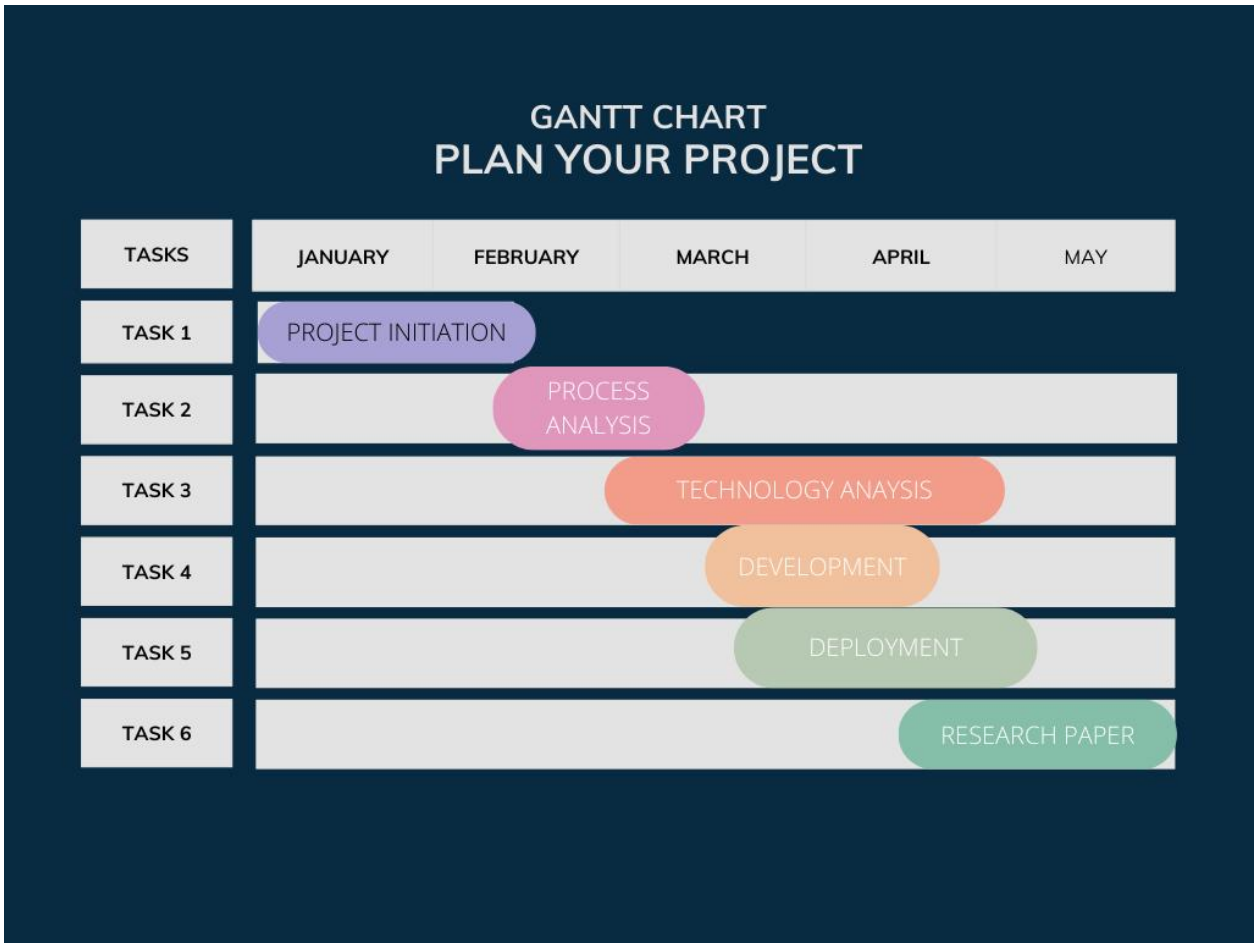
5 Call Translation Function (googletrans) – Calls the translation API/function.
6 Translation Successful – Checks whether the translation was successful.

    I. If yes: Outputs translated text.
    II. If failed: Shows an error message.

7 User Keeps Using App – Loop repeats for additional translations or activities.

Chapter-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



Chapter 8

OUTCOMES

The key results of this project revolve around the effective development of a usable and functional desktop application for Indian language translation. Developed with the cross-platform Tkinter library, the software provides possible access across operating systems, an alternative to web-based translators as a standalone application. The user will find an easy-to-use two-screen process, initiated by a friendly introduction before entering the main translator screen. This interface includes clean language selection mechanisms through dropdown menus and specialized input and output spaces for text, all driven by a prominent "Translate" button with simple visual feedback. Fundamentally, the application takes advantage of the powerful translation abilities of the Google Translate service via the googletrans library to provide real-time English-to-twelve major Indian languages translation, inheriting the strengths and limitations of the underlying service. The translated text will be displayed in a readable, read-only output region, with minimal error handling provided to alert users to possible problems. Aside from its initial functionality, the codebase built demonstrates a certain level of modularity and incorporates established Python libraries, providing a good foundation for future development and the possible inclusion of more complex features or support for more languages. Ultimately, the program should serve as a simple and useful tool for basic translation between English and a broad set of Indian languages in a specialized desktop application.The system design of the Language Translator using `tkinter` and `googletrans` focuses on creating a simple yet efficient translation interface. The design is structured into three main layers: the User Interface (UI) Layer, the Application Logic Layer, and the External API Layer.

Chapter 9

RESULTS AND DISCUSSIONS

The Language Translator Application, built through the use of Python's tkinter for its graphical user interface (GUI) and the googletrans library for translation functionality, is a real-world application of the technology of machine translation. The primary goal was to develop a user-friendly and accessible tool that enables real-time text translation among several languages. The success of the application in accomplishing this goal is reflected in its usable GUI, which enables users to easily enter text, choose source and target languages from drop-down menus, and get translated output in a special text area. 1 The use of the Google Translate API, available via the googletrans library, gave the application the ability to support a large variety of language pairs, matching the broad coverage of Google's strong translation service.

The GUI, with its emphasis on simplicity and ease of use, was an important strength.The interface was easy to understand, with plain visual indications and simplecontrols. This also helped to generate a positive experience for the users, as well asenabling the application to be accessed by users at different levels of technicalaptitude. Overall, the performance of real-time translation, helped by the API, was usually satisfactory, providing rapid responses for the majority of translation requests. Nevertheless, the performance of the application was inherently dependent on the responsiveness of the Google Translate API, which might be impacted by network latency or server load.

Accuracy of translation, although generally to be praised, had the usual limitations inherent in machine translation. For simple sentences and popular language pairs, the output was frequently accurate and natural-sounding. However, difficulties were encountered with complex sentences, idiomatic phrases, and culturally specific allusions, many of which need sophisticated interpretation beyond the capabilities of existing machine translation models. This indicates the continuing importance of natural language processing (NLP) advances to enhance the management of linguistic sophistication.

A number of solutions can be proposed to address these shortfalls. Adding offline translation support, perhaps in the form of embedding local machine translation models, would make the application far more accessible. Enhancing error handling to providing more informative user feedback would enhance the overall user experience.including advanced GUI capabilities, like translation history, language detection, and copy-to-clipboard, would enhance usability even further. Enhancing the program to support document translation and the inclusion of speech recognition and synthesis would extend its use. Employing more stable and officially sanctioned translation engineswould address the issue of reliability. Developing language specific modules, and enabling the user to construct custom glossaries would assist in accuracy.

The ethical implications, especially with respect to bias in machine translation, cannot be ignored. The users must be informed of the possibility of biased output and must be urged to exercise restraint when using the application for important applications. Data privacy, and particularly in the case of sending sensitive data over the Internet via online translation services, warrants careful consideration.

Chapter 10

CONCLUSION

The Language Translator Application, making use of Python's tkinter and googletrans,successfully implements the development of an effective and easy-to-use translation tool.It effectively accomplishes real-time translation via an easy-to-use GUI, serving the needs of a broad number of language pairs. Though the application realizes its primary function, its use of the Google Translate API and the built-in limitations of machine translation introduce certain drawbacks.The unreliability of googletrans and the requirement for internet connectivity limit its use.Future development would need to aim at overcoming these limitations. Offline translation,better error handling, and inclusion of advanced GUI functionality would greatly enhance the user experience. Additionally, the inclusion of more stable translation engines, and the development of language specific modules, would increase stability and accuracy. Ethical considerations, most notably bias and data privacy, need to be addressed to provide responsible use.

REFERENCES

[1] Google Cloud Translate API Documentation. https://cloud.google.com/translate

[2] Grayson, J. (2012). Python and Tkinter Programming. Manning Publications.

[3] Jha, G.N. (2019). Machine Translation and Indian Languages. Journal of Language Technology.

[4] Ramanathan, A., & Rao, K. (2018). Multilingual Computing in India: A Review. International Journal of Computer Applications.

[5] Python Software Foundation. (2023). Tkinter GUI Programming. https://docs.python.org/3/library/tkinter.html

APPENDIX-A

PSUEDOCODE

```
FUNCTION LanguageTranslator:
GUI = CreateGraphicalUserInterface()
INPUT_TEXT_AREA = GUI.CreateTextInputArea()
OUTPUT_TEXT_AREA = GUI.CreateOutputTextArea()
SOURCE_LANGUAGE_DROPDOWN = GUI.CreateDropdown("Source Language",
GetAvailableLanguages())
TARGET_LANGUAGE_DROPDOWN = GUI.CreateDropdown("Target Language",
GetAvailableLanguages())
TRANSLATE_BUTTON = GUI.CreateButton("Translate")
TRANSLATE_BUTTON.OnClick:
INPUT_TEXT = INPUT_TEXT_AREA.GetText()
SOURCE_LANGUAGE = SOURCE_LANGUAGE_DROPDOWN.GetSelectedValue()
TARGET_LANGUAGE = TARGET_LANGUAGE_DROPDOWN.GetSelectedValue()
IF INPUT_TEXT is empty:
OUTPUT_TEXT_AREA.SetText("Please enter text to translate.")
RETURN
IF SOURCE_LANGUAGE equals TARGET_LANGUAGE:
OUTPUT_TEXT_AREA.SetText("Source and target languages cannot be the same.")
RETURN
TRY:
TRANSLATED_TEXT = TranslateText(INPUT_TEXT, SOURCE_LANGUAGE,
TARGET_LANGUAGE)
OUTPUT_TEXT_AREA.SetText(TRANSLATED_TEXT)
EXCEPT TranslationError as ERROR:
OUTPUT_TEXT_AREA.SetText("Translation error: " + ERROR.Message)
EXCEPT NetworkError as ERROR:
OUTPUT_TEXT_AREA.SetText("Network error: " + ERROR.Message)
EXCEPT InvalidLanguageError as ERROR:
OUTPUT_TEXT_AREA.SetText("Invalid language selection: " + ERROR.Message)
EXCEPT OtherError as ERROR:
```

```
OUTPUT_TEXT_AREA.SetText("An unexpected error occurred: " + ERROR.Message)
GUI.Run()
FUNCTION TranslateText(TEXT, SOURCE_LANGUAGE, TARGET_LANGUAGE):
TRY:
SOURCE_CODE = GetLanguageCode(SOURCE_LANGUAGE)
TARGET_CODE = GetLanguageCode(TARGET_LANGUAGE)
TRANSLATED_RESULT
=
CallTranslationAPI(TEXT,
SOURCE_CODE,
TARGET_CODE)
RETURN TRANSLATED_RESULT.TranslatedText
EXCEPT InvalidLanguage as ERROR:
RAISE InvalidLanguageError(ERROR.Message)
EXCEPT APIError as ERROR:
RAISE TranslationError(ERROR.Message)
EXCEPT NetworkError as ERROR:
RAISE NetworkError(ERROR.Message)
EXCEPT OtherError as ERROR:
RAISE OtherError(ERROR.Message)
FUNCTION GetAvailableLanguages():
RETURN ListOfLanguageNames
FUNCTION GetLanguageCode(LANGUAGE_NAME):
RETURN CorrespondingLanguageCode
FUNCTION CallTranslationAPI(TEXT, SOURCE_CODE, TARGET_CODE):
// Make API call using the translation engine (e.g., googletrans)
// return the api response.
RETURN APIRespons
```

APPENDIX-B

SCREENSHOTS

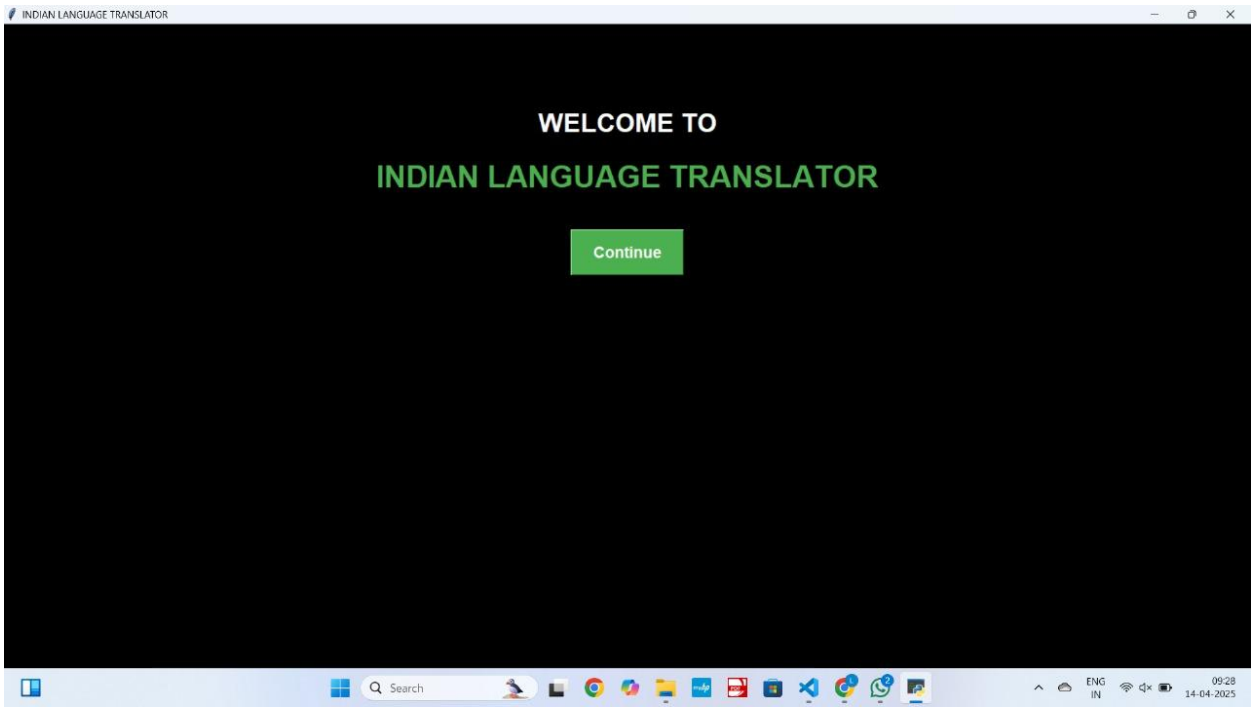


Fig 1.3 INDIAN LANGUAGE TRANSLATOR INTERFACE

FIG 1.4 THE TRANSLATION OF AN ENGLISH STORY INTO HINDI USING AN INDIAN LANGUAGE TRANSLATOR SOFTWARE.

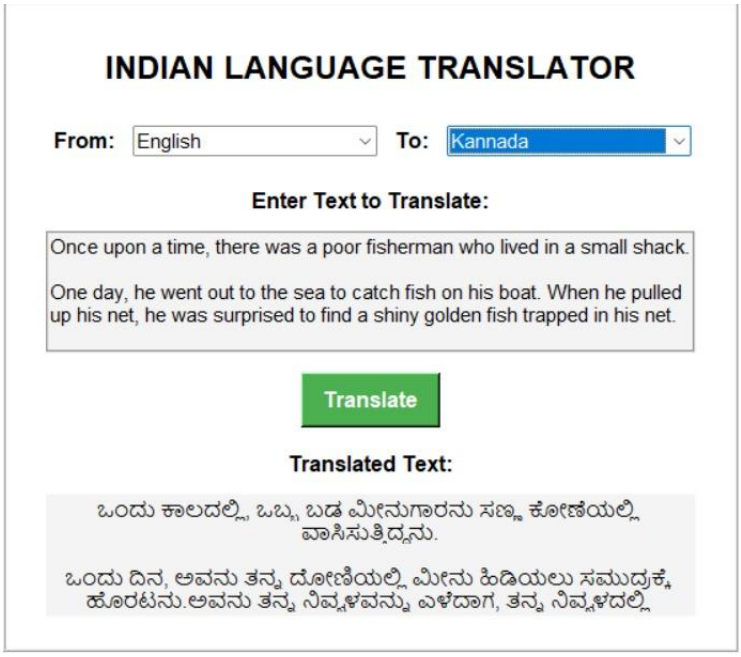


FIG 1.5 THE TRANSLATION OF AN ENGLISH STORY INTO KANNADA USING AN INDIAN LANGUAGE TRANSLATOR SOFTWARE.

FIG 1.6 THE TRANSLATION OF AN ENGLISH STORY INTO TAMIL USING AN INDIAN LANGUAGE TRANSLATOR SOFTWARE

FIG 1.7 THE TRANSLATION OF AN ENGLISH STORY INTO MARATHI USING AN INDIAN LANGUAGE TRANSLATOR SOFTWARE

OUTPUT STEPS:

- The user inputs or copies and pastes text into the input field.
- They choose the source language (input text language) and target language (language to translate to) from the drop-down menus.
- On clicking the "Translate" button, the app posts the input text, source language code, and target language code to the Google Translate API via the googletrans library.
- The API translates the input text and returns the translated text.
- The application then shows the translated text in the output text area in a centered and read-only manner.
- In case of any error (like internet problems or unsupported text), a suitable error message is displayed in place of the translation.

APPENDIX-C

ENCLOSURES

1. Journal publication/Conference Paper Presented Certificates

IJSREM Journal <editor@ijsrem.com> 1:01 PM (4 hours ago)
to me

Dear Author,

We would like to congratulate you on the successful publication of your research paper in our journal! **International Journal of Scientific Research in Engineering and Management (IJSREM)** on **Volume 09, Issue 05 May 2025**.

Paper Title : Developing A Software That Can Translate Resource Material and Other Texts from English to Other Indian Regional Language

Your hard work has been noted and appreciated by us, and we thank you for showing interest in our journal. Your work is now visible to the world, and as an appreciation of your contribution we enclose with this email e-certificate of all the author(s) as a token from us.

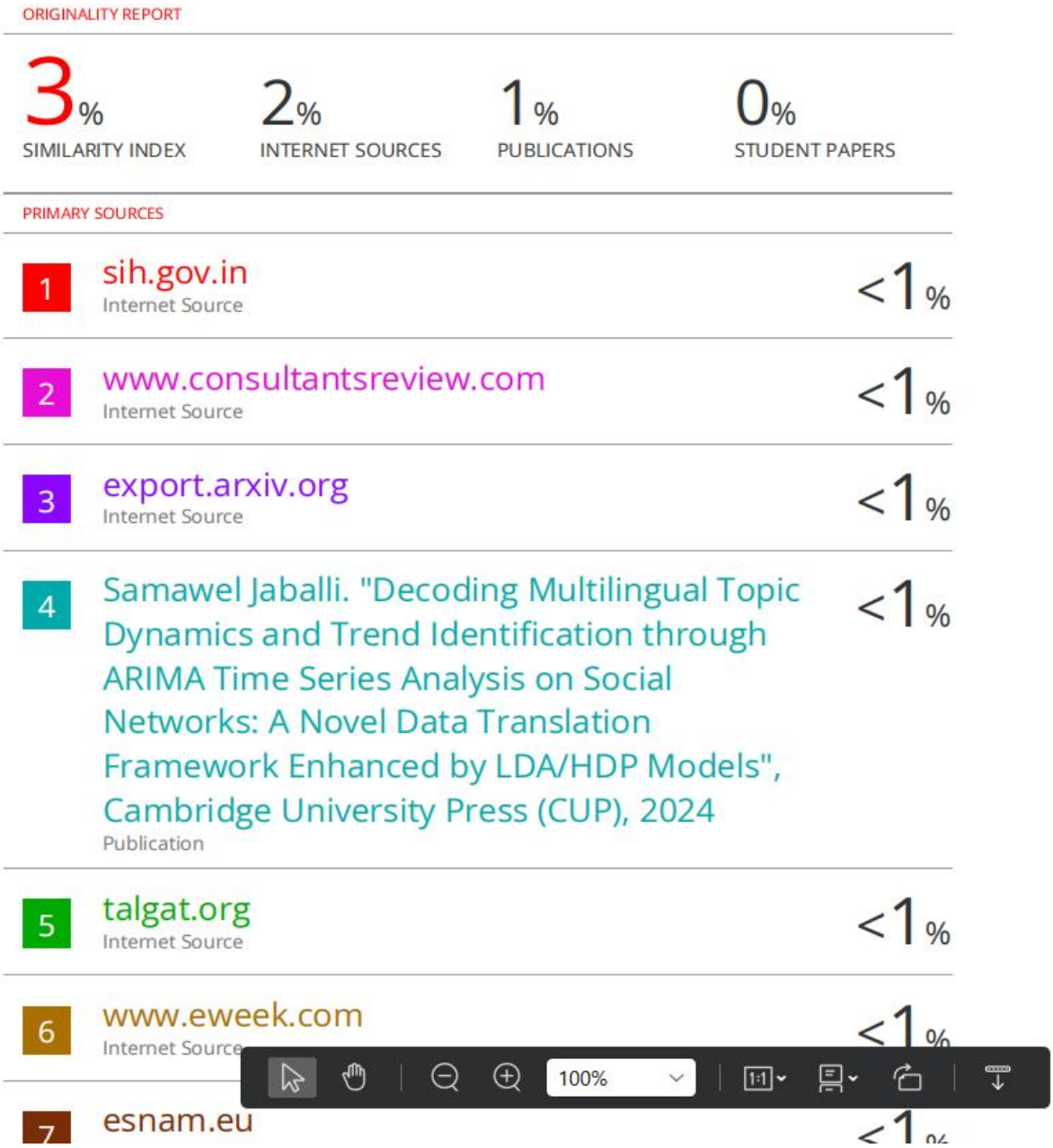
DOI: 10.55041/IJSREM47848


DOI: 10.55041/IJSREM47848
IJSREM
e-Journal
ISSN: 2582-3930
Impact Factor: 8.586
INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata
CERTIFICATE OF PUBLICATION
International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to
LEKHANA M
in recognition to the publication of paper titled
Developing A Software That Can Translate Resource Material and Other Texts from English to Other Indian Regional Language
published in IJSREM Journal on Volume 09 Issue 05 May, 2025
www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com


DOI: 10.55041/IJSREM47848
IJSREM
e-Journal
ISSN: 2582-3930
Impact Factor: 8.586
INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata
CERTIFICATE OF PUBLICATION
International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to
DEEPTHI S
in recognition to the publication of paper titled
Developing A Software That Can Translate Resource Material and Other Texts from English to Other Indian Regional Language
published in IJSREM Journal on Volume 09 Issue 05 May, 2025
www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com

2. Include certificate(s) of any Achievement/Award won in any project-related event.

3. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.



ORIGINALITY REPORT

3%	2%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	sih.gov.in Internet Source	<1%
2	www.consultantsreview.com Internet Source	<1%
3	export.arxiv.org Internet Source	<1%
4	Samawel Jaballi. "Decoding Multilingual Topic Dynamics and Trend Identification through ARIMA Time Series Analysis on Social Networks: A Novel Data Translation Framework Enhanced by LDA/HDP Models", Cambridge University Press (CUP), 2024 Publication	<1%
5	talgat.org Internet Source	<1%
6	www.eweek.com Internet Source	<1%
7	esnam.eu	<1%

4. Details of mapping the project with the Sustainable Development Goals (SDGs).

SDG 4: Quality Education

Your software enhances access to educational resources by breaking language barriers. This supports **inclusive and equitable quality education** by allowing students and teachers who speak regional languages to access materials in their native tongue.

SDG 8: Decent Work and Economic Growth

i. By enabling people from non-English-speaking backgrounds to access knowledge and training material, the software can support **skilling and workforce inclusion**, enhancing employability and economic growth.

ii. Access to vocational training, skilling programs, and employment-related information in regional languages enhances workforce participation

SDG 16: Peace, Justice and Strong Institutions

i. Providing access to information in local languages ensures **greater transparency and inclusivity** in governance and civic participation. It empowers citizens to better understand their rights and public services

ii. Promotes transparency, civic education, and access to legal resources in local languages, enhancing governance and rights awareness.

SDG 17: Partnerships for the Goals

i. Promoting multilingual software solutions supports the **use of enabling technologies** and fosters **data collection and dissemination in regional languages**, essential for inclusive development partnerships.

ii. Supports collaboration across sectors (tech, education, government) by creating a multilingual ecosystem that fosters inclusivity.