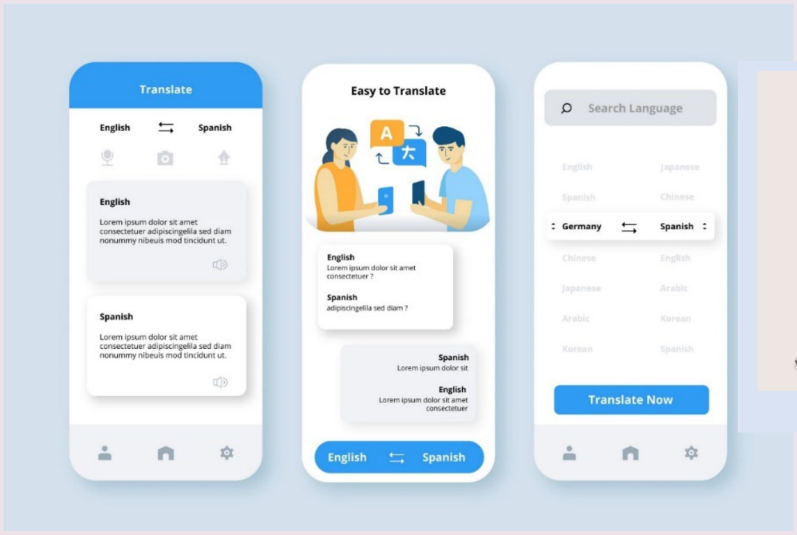
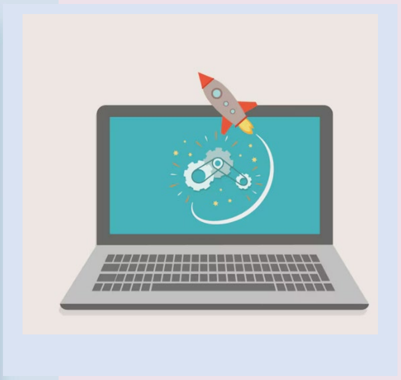




Software Requirements Specification
Version 1.0

GeoSpeak

Theme: Language Master

Project Name: GeoSpeak

Category: GenAI Smart Solutions

Contents

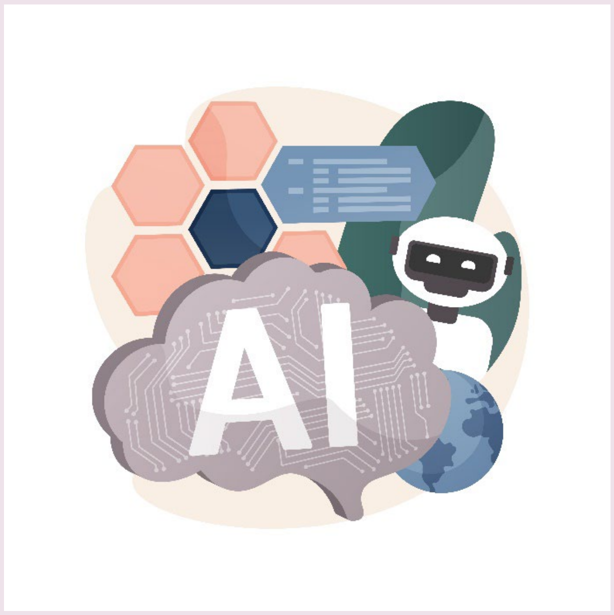
1.1 Background and Necessity for the Application

In a world that is becoming more connected, effective communication across different languages is crucial for increasing collaboration, cultural exchange, and business operations. The demand for accurate and efficient language translation services has emerged, driven by international trade, travel, online content creation, and social media interactions. Traditional translation methods, including manual translation and simple rule-based systems, often fall short in terms of speed, accuracy, and contextual relevance.

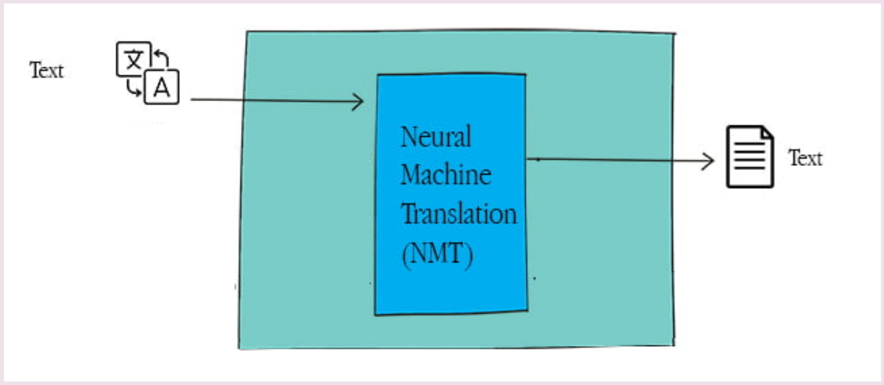
The necessity of this application '**GeoSpeak**' arises from the limitations of traditional translation methods and the growing complexity of language usage in various domains. Using OpenAI's Large Language Models (LLMs) and Generative AI (Gen AI), the application can dynamically interpret and translate text considering context, idioms, and subtle nuances. This is particularly important for applications in diplomacy, global business, education, healthcare, and customer service where precise communication can significantly impact outcomes. This application aims to bridge language barriers, enhance cross-cultural understanding, and improve access to information for users worldwide.

1.2 Proposed Solution

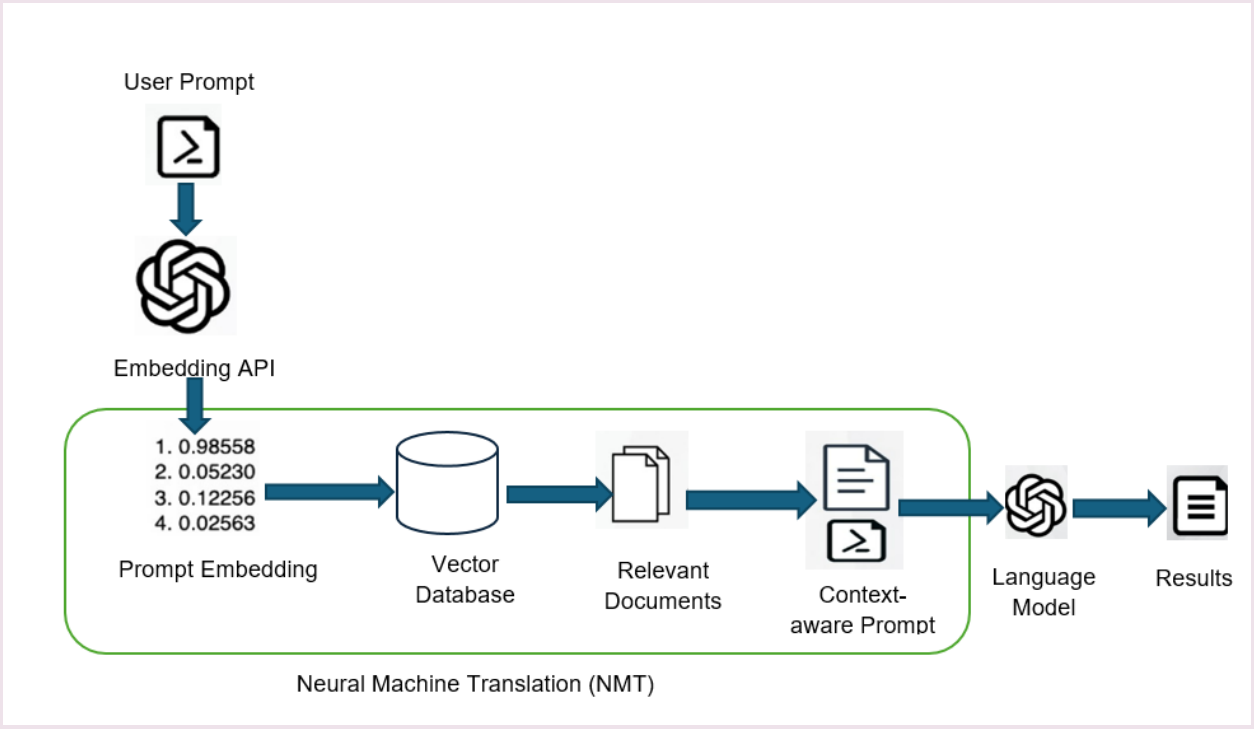
The proposed solution, **GeoSpeak**, is a comprehensive real-time Web application translation tool designed to bridge language barriers. At its core, the application will leverage advanced AI technologies, specifically a cloud-based LLM, to provide accurate and contextually appropriate translations. This GeoSpeak solution meets current requirements and evolves for future growth, becoming essential in multinational and multilingual work environments. OpenAI's models, known for their exceptional language understanding and generation capabilities, enable real-time and contextually aware translations that are far more accurate than traditional methods. By addressing both, current translation challenges and expecting future necessities, this solution has the potential to significantly improve cross-language communication in diverse workplace settings.



Workflow of the GeoSpeak application is as follows:



Sample Architecture of the Application is as follows:



Sample Architecture of the Application

Initially at the User Input stage, the user provides text in the English language and selects the desired target language from a dropdown menu. This input text is then processed by the Embedding API, which converts the text into a high-dimensional embedding vector that captures its semantic meaning. Subsequently, this embedding vector is used to query a Vector Database that contains precomputed embeddings of parallel corpora or translation examples. The system retrieves relevant documents that closely match the source text embedding. These documents consist of paired texts in both the source and target languages, which help to inform the translation process.

The application then creates a Context-aware Prompt, which integrates the source text and the specified target language, ensuring the translation model understands the context and language requirements. For instance, the prompt might be structured as "Translate following text to French: 'Hello, how are you?'". This context-aware prompt is fed into a Language Model, a sophisticated AI model trained on vast amounts of multilingual text data. The language model processes the prompt and generates the corresponding translation in the target language.

Finally, the translated text is presented to the user as output. This output can be displayed in a user-friendly format allowing the user to see the translated text directly. Throughout this process, the application leverages OpenAI LLMs to ensure accurate and contextually appropriate translations, making it a powerful tool for users requiring real-time language translation services.

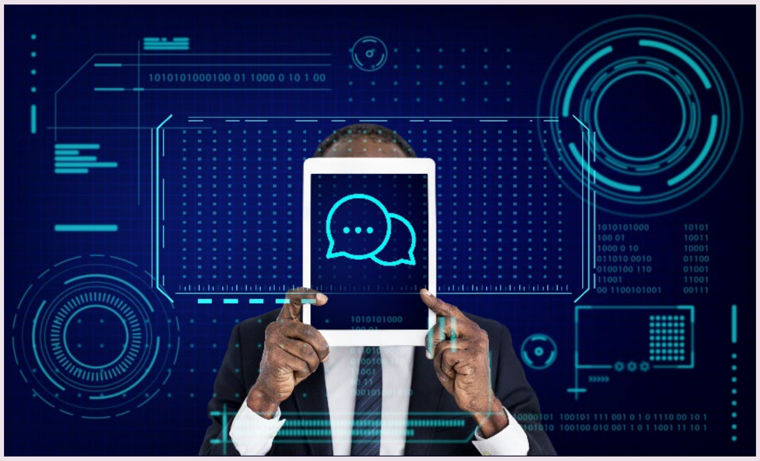
1.3 Purpose of the Document

The purpose of this document is to present a detailed description of the (Gen AI) simulating text conversion by matching user prompts to scripted responses titled **GeoSpeak**.

This document explains the purpose and features of (Gen AI) and LLM and the constraints under which it must operate. This document is intended for both stakeholders and developers of the system.

1.4 Scope of Project

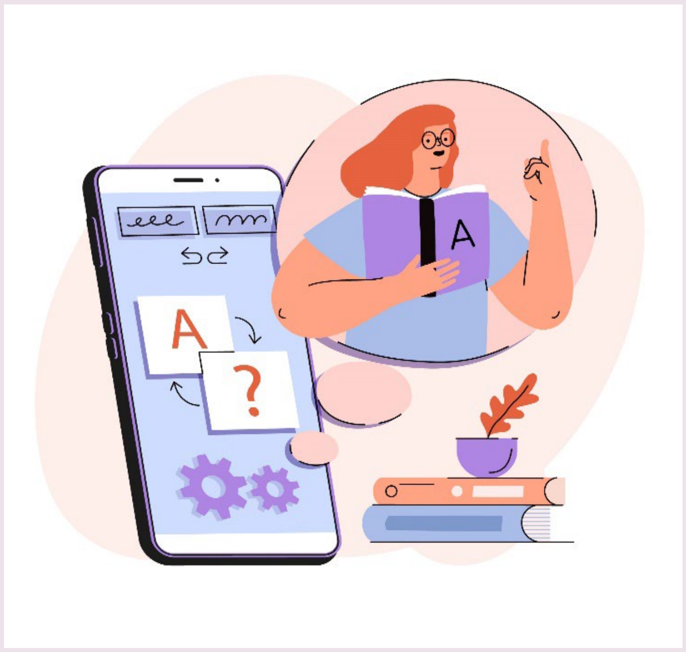
The scope of this project involves developing a language translation application that allows users to input text, select a target language, and receive a translation. It includes processing text through an Embedding API to generate a semantic vector, querying a Vector Database for relevant translation examples, and creating a context-aware prompt for a Language Model. This model then generates the translation, which is presented to the user in a clear and user-friendly format. The application leverages OpenAI's LLM to ensure accurate and contextually appropriate translations.



1.5 Constraints

GeoSpeak faces several constraints including the requirement for high-quality precomputed embeddings in the Vector Database to ensure accurate retrieval of parallel corpa/translation examples. There is also a reliance on the performance of the Language Model to provide contextually accurate translations which requires efficient integration and processing capabilities.

Additionally, the application must handle diverse languages and text inputs robustly and maintain user data security and privacy throughout the translation process. Performance and scalability are crucial, especially under high user loads. This application is not equipped to audio and speech translation.



1.6 Functional Requirements

The project involves developing an intelligent translation application utilizing (Gen AI), Large LLMs, and technologies from OpenAI or other relevant model. The application will be designed to process user input provided in the form of text input. Some of the functional requirements are as follows:

i. **Text Input** - The application must allow users to input text in a source language and select a target language from a drop-down menu.

ii. **Text Processing** - It must process the input text using an Embedding API to generate a high-dimensional semantic vector.

iii. **Document Retrieval -** The application must query a Vector Database to retrieve relevant documents based on the generated embedding vector.

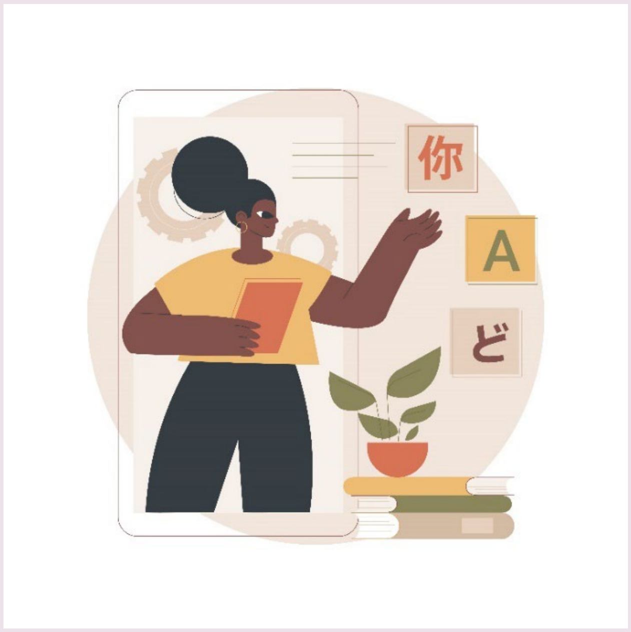
iv. **Context-aware Prompt Creation** - It must generate a context-aware Translation model.

v. **Translation Generation** - The application must use a Language Model to generate a translation of the source text based on the context-aware prompt.

vi. **Result Display** - It must present the translated text to the user in a clear and user-friendly format.

vii. **Error Handling** - The system should handle errors in text processing, document retrieval, and translation generation gracefully.

viii. **User Interface** - It must provide an intuitive and responsive user interface for input, language selection, and viewing translation results.



1.7 Non-Functional Requirements

There are several non-functional requirements that should be fulfilled by the system.

The application should be:

1. **<u>Secure</u>**: The application must ensure the secure handling of user data and interactions with APIs and databases.
2. **<u>Efficient</u>**: The application should process and return translations quickly, with minimal latency, to ensure a smooth user experience.
3. **<u>Scalable</u>**: The Application must handle varying loads efficiently, from individual users to high traffic scenarios, without degradation in performance.
4. **<u>Reliable</u>**: The application should operate consistently and correctly with minimal downtime and robust error recovery mechanisms.
5. **<u>Usable</u>**: The user interface should be intuitive and accessible, providing a seamless experience for users of varying technical skills.

These are the bare minimum expectations from the project. It is a must to implement the FUNCTIONAL and NON-FUNCTIONAL requirements given in this SRS.

Once they are complete, you can use your own creativity and imagination to add more features if required.

1.8 Interface Requirements

1.8.1 Hardware

Intel Core i5/i7 Processor or higher
8 GB RAM or higher
Color SVGA
500 GB Hard Disk space
Mouse
Keyboard

1.8.2 Software

Technologies to be used:

1. **Frontend**: HTML5 or any other scripting languages
2. **Backend**: Flask/Django
3. **Data Store**: TXT
4. **Programming/IDE**: Python, Jupyter Notebook, Anaconda, or Google Colab
5. **Libraries**: Tensorflow, Keras, OpenAI API, Python libraries, and pre-trained transformers

1.9 Project Deliverables

You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design Specifications
- Diagrams such as User Flow Diagram/User Journey Map
- Detailed steps to execute the project
- Test Data Used in the Project
- Project Installation Instructions
- Link of GitHub for accessing the uploaded project code (Link should have public access)
- Link of Published Blog

The source code, including .ipynb files for Jupyter Notebook and Google Colab, should be shared via GitHub. Appropriate access permissions should be granted to users to allow testing for Jupyter Notebook and Google Colab. The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end.

Provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project; hence, all crucial aspects of the project must be documented properly. Ensure that documentation is complete and comprehensive.

You should publish a blog of minimum 2000 words on any free blogging Website such as Blogger, Tumblr, Ghost, or any other blogging Website. The link of the published blog should be submitted along with the project documentation.

Submit a video (.mp4 file) demonstrating the working of the application, including all the functionalities of the project. This is MANDATORY.

*~~~ End of Document ~~~*