SRI VENKATESWARA COLLEGE OF ENGINEERING

Tirupati

INTERSHIP DOCUMENTATION

ON

"TransLingua : Al - Powered Multi - Language Translator"

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

MOGALARI MANEESHA (22BFA02067)

KORAKALA SREE PRIYA (22BFA02058)

KADALURU HANUMANTHA (22BFA02054)

TEAM ID:

LTVIP2026TMIDS79655

1.Project Overview

An AI-powered multi-language translator is a system that uses Artificial Intelligence, Natural Language Processing (NLP), and Neural Machine Translation (NMT) to automatically translate text or speech from one language to another. The system accepts user input in the form of text or voice, detects the source language, processes the content using trained AI models, and generates an accurate translation in the target language in real time. This technology helps overcome language barriers and is widely used in education, travel, business communication, government services, and social media platforms. Future improvements include offline translation, real-time speech translation, and support for more regional languages to make communication faster and more accessible worldwide.

An AI-powered multi-language translator is a smart system that uses Artificial Intelligence and Natural Language Processing (NLP) to translate text or speech from one language to another automatically. It detects the input language, understands the meaning using trained AI models, and provides accurate translation in the selected language within seconds. This technology helps people communicate easily across different languages and is widely used in education, travel, business communication, and mobile applications.

1.1 PURPOSE :

An **AI-powered multi-language translator** is designed to break language barriers by enabling fast, accurate, and natural communication between people who speak different languages. Its main purpose is to support global interaction in areas like education, business, healthcare, travel, and digital communication by translating text or speech in real time while preserving meaning, context, and tone. By using artificial intelligence and machine learning, it continuously improves accuracy, understands cultural nuances, and handles multiple languages efficiently, helping people access information, collaborate globally, and communicate inclusively without the limitations of language differences.

1.2 FEATURES :

1. ***Multi-Language Support:***

***Feature*:** Can translate many languages, including local and international ones.

***Explanation*:** This allows people from different countries to communicate easily, making global interaction possible.

2. ***Real-Time Translation:***

***Feature*:** Translates text or speech instantly.

Explanation: You don't have to wait for translations; conversations happen naturally and quickly, which is useful in meetings, travel, or online chats.

3. ***Accurate and Context-Aware:***

Feature: Understands sentence meaning, tone, and context.

***Explanation*:** Unlike simple word-for-word translators, AI ensures the translation conveys the original message correctly, avoiding confusion.

4. ***Text-to-Speech and Speech-to-Text:***

Feature: Converts spoken words to text and reads translations aloud.

***Explanation*:** Helps people who prefer listening or speaking instead of typing, making it easier to communicate in real life or online.

5. Offline Translation

Feature: Works without an internet connection for some languages.

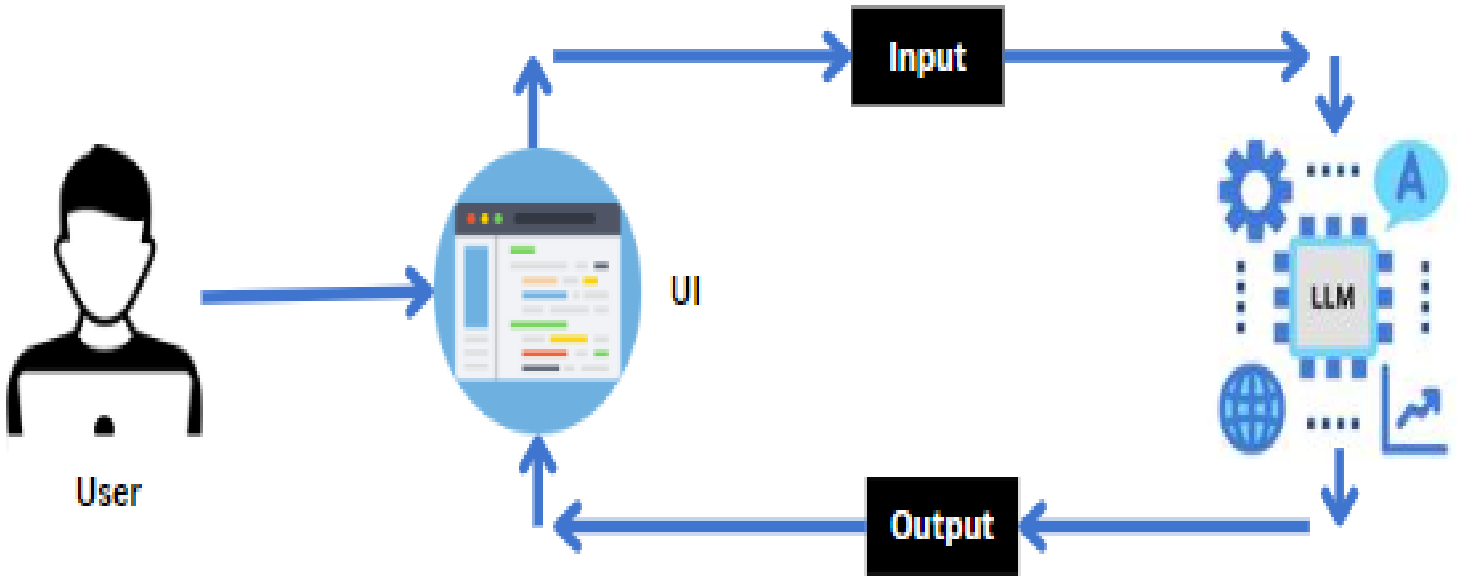
***Explanation*:** Useful for travelers or areas with limited internet, so language is never a barrier.

6. User-Friendly Interface

Feature: Simple, easy-to-use design.

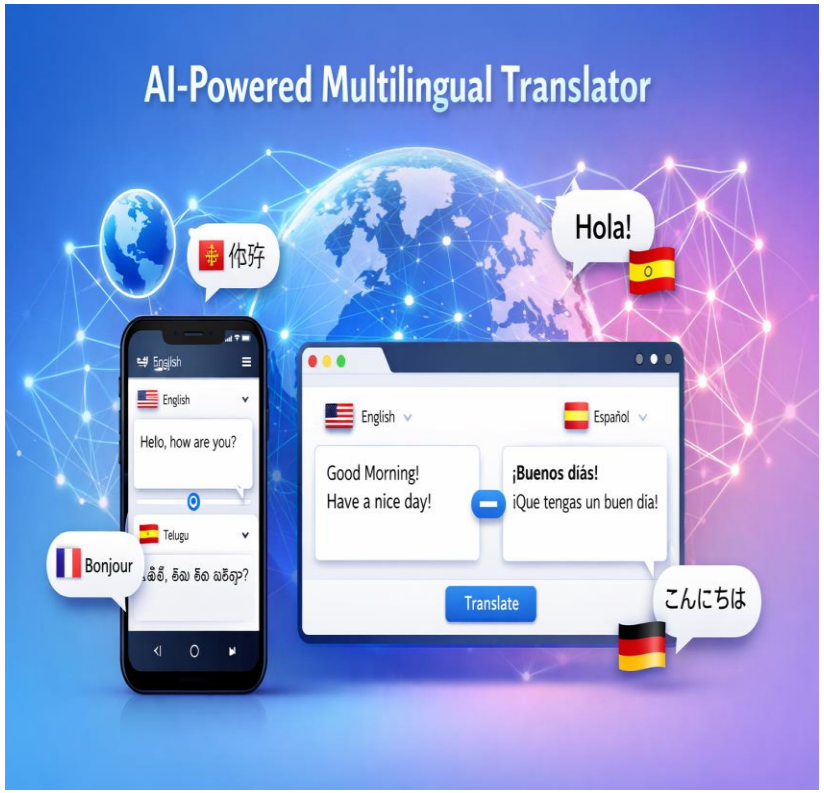
***Explanation*:** Anyone, even beginners or students, can use it without difficulty, making communication accessible to all.

2.ARCHITECTURE



2.1 FRONTEND (user Interface)

Technology: Streamlit (python-based UI Framework)



- ***Input Area***
  - *Users can type text or speak using a microphone
  - * For speech, it connects to a **speech-to-text** module.
- ***Language Selection***
  - * Dropdown menus or buttons to choose the **source language** and **target language**.
  - * Allows easy switching between multiple languages.
- ***Translation Display***
  - *Shows translated text clearly.
  - *Optionally, provides **text-to-speech** so users can hear the translation.
- ***Additional Features***
  - *Copy, share, or save translations.
  - *Simple, clean, and responsive design for desktop and mobile.

Streamlit:

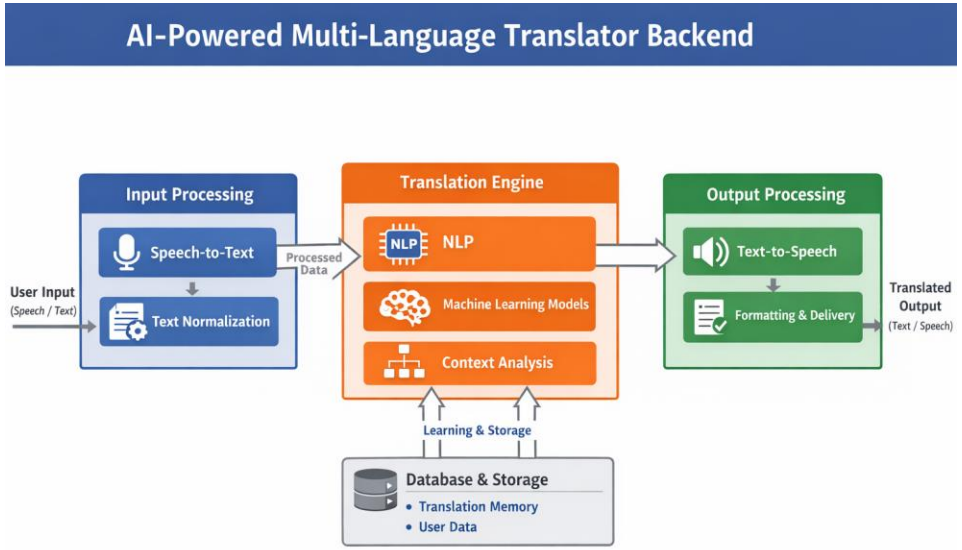
Basic knowledge of building interactive web applications using Streamlit.

Understanding of Streamlit's UI components and how to integrate them with backend logic.

2.2 BACKEND

Technology : python

- ***Input Processing Module***
  - *Converts user input into a format the AI can understand.
  - *For speech input, it uses **Speech-to-Text** to get the text version.
  - *Handles text cleaning, normalization, and tokenization.
- ***Translation Engine (Core AI)***
  - *Uses **Natural Language Processing (NLP)** and **Machine Learning models**.
  - *Understands the meaning, context, and grammar of the input.
  - *Produces accurate translations in the target language.
- ***Output Processing Module***
  - *Converts AI-generated translations to text or speech.
  - *Ensures formatting, punctuation, and tone are correct.
  - *Sends the translation back to the frontend for display.
- ***Database & Storage (Optional)***
  - *Stores user preferences, frequently used phrases, or translation history.
  - *Helps improve AI over time with **continuous learning**.



2.3 DATABASE (Data Management Layer)

1 *Language Data / Translation Corpus*

Stores all text data needed for translation.

Could include:

- Parallel corpora (same sentence in multiple languages)
- Dictionaries or word mappings
- Common phrases & idio

***Database type*:** Can be **SQL** (like PostgreSQL, MySQL) for structured sentences, or **NoSQL** (like MongoDB) for flexible storage of JSON objects with multiple translations.

2 *User Inputs / Translation History*

Stores what users input and the translations generated.

Useful for:

- Analytics (most translated languages, phrases)
- Improving model via feedback
- Caching frequent translations to reduce AI computation

Database type: **NoSQL** (MongoDB, Firebase) works well for storing dynamic user queries and logs.

3 *AI Model Metadata*

Stores model versions, fine-tuning parameters, or API logs.

***Database type*:** Lightweight **SQL** or **NoSQL**, or even a **file-based system** if small-scale.

3.SETUP INSTRUCTIONS

3.1 Pre-requisites:

1.***Python 3.10+*** – the programming language to run AI models.

2.***AI Libraries*** – transformers and torch or tensorflow for model inference.

3.***Pre-trained Translation Model*** – like Hugging Face's MarianMT or M2M100.

4.***Text Data / Datasets*** – parallel corpora for source and target languages.

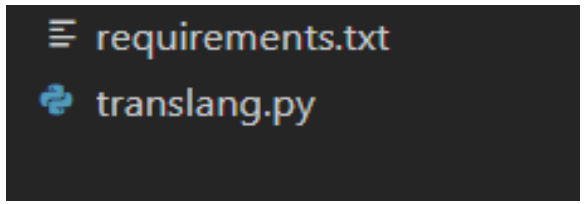
5.***Hardware*** – computer with at least 8GB RAM; GPU recommended for faster processing.

3.2 Installation :

1. *Generate PALM API*
2. *Initialize the pre-trained model*

4. PROJECT STRUCTURE

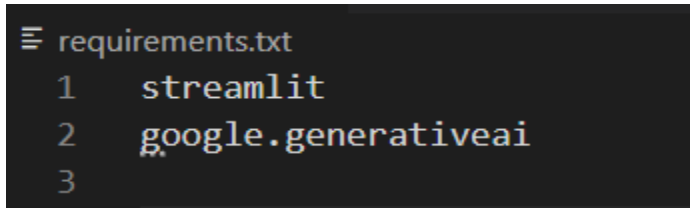
4.1 Create the Project folder which contains application file as shown below



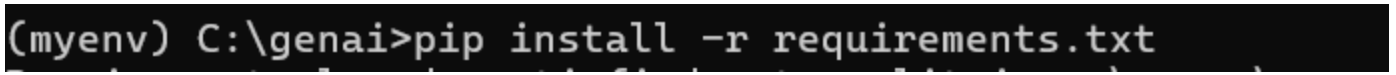
4.2 *Requirements Specification*

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

- ***Create a requirements.txt file to list the required libraries***



- ***Install the required libraries.***

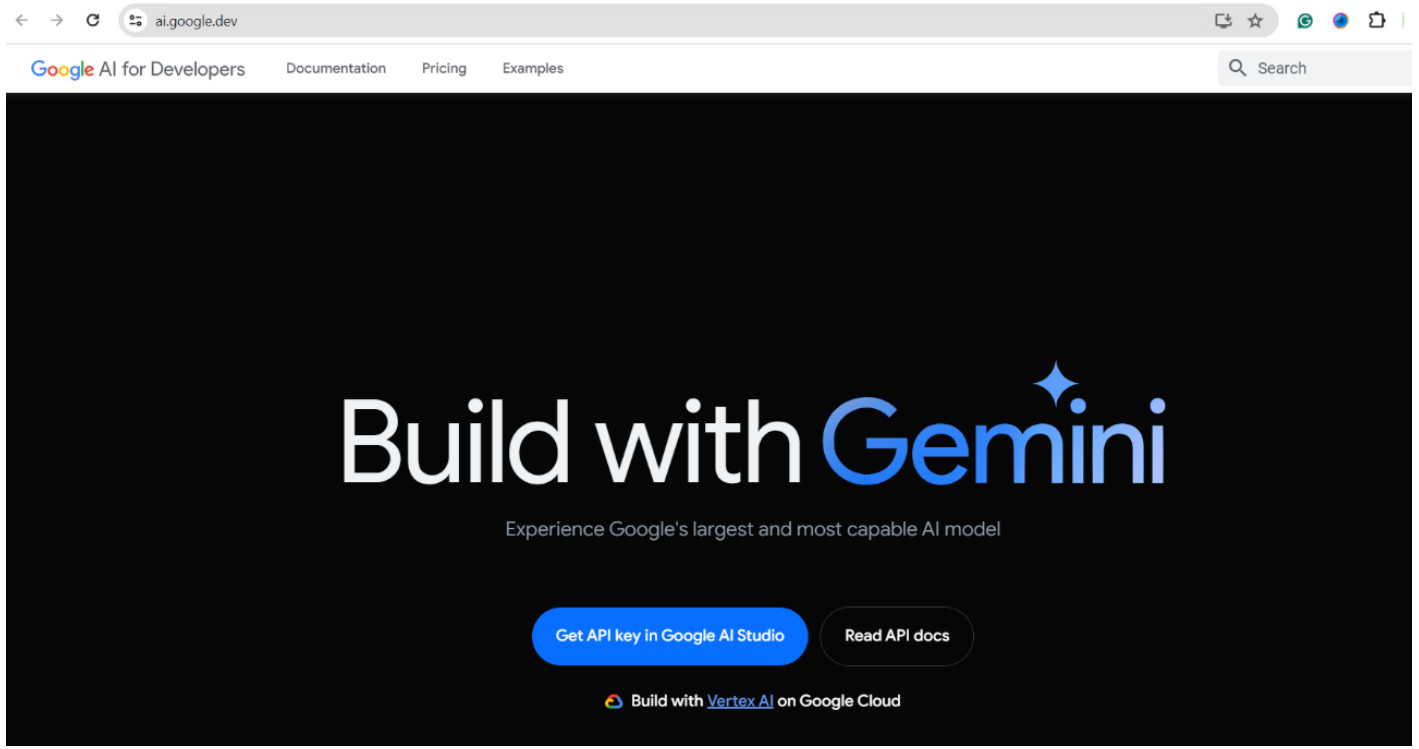


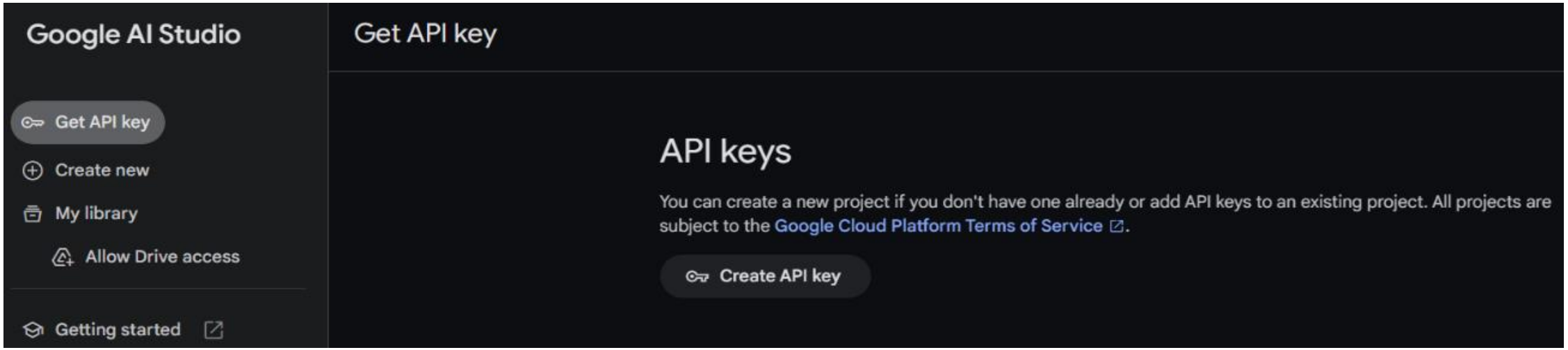
5. Initializing the Models

For initializing the model we need to generate PALM API.

5.1 Generate PALM API

Click on the link (https://developers.generativeai.google/). Then click on “Get API key in Google AI Studio”. Click on “Get API key” from the right navigation menu . Now click on “Create API key”. (Refer the below images)Copy the API key.





5.2 Initialize the pre-trained model : Import necessary files

```
from dotenv import load_dotenv # typ
import streamlit as st
import os
import google.generativeai as genai
```

- Streamlit , a popular Python library, is imported as st, enabling the creation of user interfaces directly within the Python script.
- Google Generative AI (genai): Imported to interact with the Gemini Pro model.

- The ‘load_dotenv’ function loads key-value pairs from a `.env` file into environment variables.
- The `os` module is then used to access and manage these variables.
- Activity 2.2: Configuration of the Gemini Pro API

```
# Load environment variables
load_dotenv()

api_key = "AIzaSyB5U5-f1edVl99djSKEcqDoFLcI2l6uYyI"
genai.configure(api_key=api_key)
```

- Configuring the API key: The configure function is used to set up or configure the Google API with an API key. The provided API key, in this case, is " AIzaSyB5U5-f1edVl99djSKEcqDoFLcXXXXXX".

5.3 Define the model to be used

```
model = genai.GenerativeModel('gemini-1.5-flash')
```

- Created an instance of Generative Model with the model name set to "gemini-1.5-flash".

6. MODEL DEPLOYMENT

In this milestone, we are deploying the created model using streamlit . Model deployment using Streamlit involves creating a user-friendly web interface, enabling users to interact with the model through a browser. Streamlit provides easy-to-use tools for developing and deploying data-driven applications, allowing for seamless integration of models into web-based applications.

6.1 Give the project title

```
# Initialize Streamlit app
st.set_page_config(page_title="AI-Powered Language Translator", page_icon="🌐")
st.header("🌐 AI-Powered Language Translator")
```

- The main function to start the Streamlit application, allowing users to interact with the web app.
- st.set_page_config(page_title="AI-Powered Language Translator", page_icon="????"): Sets the app's page title and icon for the browser tab.
- st.header ("???? AI-Powered Language Translator"): Displays a prominent header with the app's title, clarifying its purpose.

6.2 Create fields for user to input data for generating blog

```python
# User input for text, source language, and target language
text = st.text_area("📝 Enter text to translate:")
source_language = st.selectbox("🌍 Select source language:", ["English", "Spanish", "French", "German", "Chinese"])
target_language = st.selectbox("🌎 Select target language:", ["English", "Spanish", "French", "German", "Chinese"])

# Translate text button
if st.button("🔄 Translate"):
    if text and source_language and target_language:
        try:
            translated_text = translate_text(text, source_language, target_language)
            st.subheader("🗣️ Translated Text:")
            st.write(translated_text)
        except Exception as e:
            st.error(f"⚠️ Error: {str(e)}")
    else:
        st.warning("⚠️ Please fill in all fields.")
```

```python
if __name__ == "__main__":
    main()
```

- Finally, the main() function is called to execute the Streamlit app.

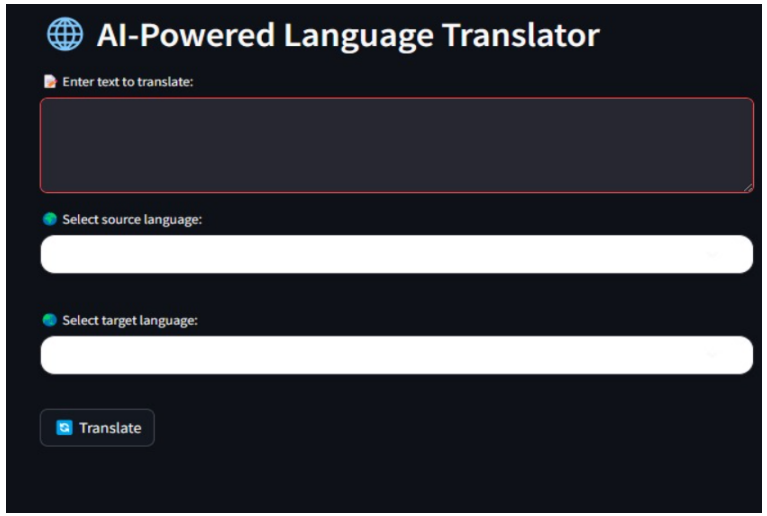
7. RUNNING THE WEB APPLICATIONS

- Open the anaconda prompt from the start menu
- Navigate to the folder where your Python script is.
- Now type “streamlit run app.py” command
- Navigate to the localhost where you can view your web page

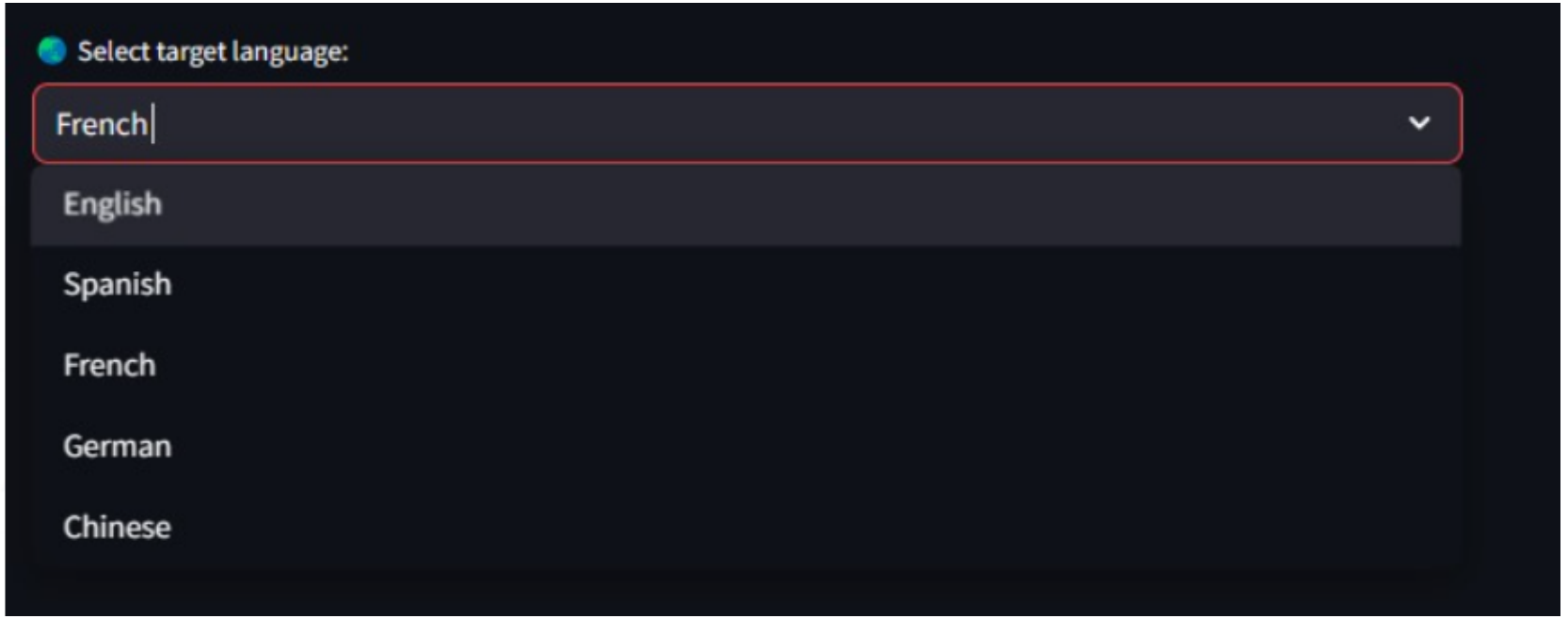
```
(s1) D:\smart bridge\Gen AI\translator gen ai>streamlit run translator.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.68.52:8501
```

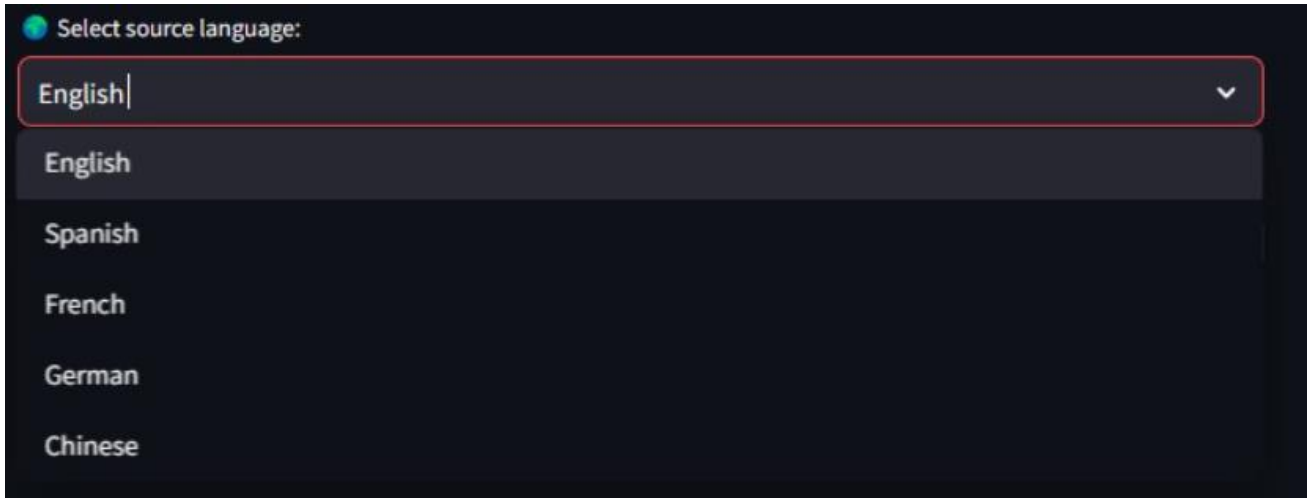
Now, the application will open in the web browser,

Now, the application will open in the web browser,



After giving the input:





The Output generated: