# Software Requirements Specification (SRS)


## Real Time Language Translator

### 1. Introduction

#### 1.1 Purpose

The purpose of this software is to provide a real-time multilingual translation tool that allows users to speak in one language, have their speech translated into another, and hear the translated text. This tool integrates speech recognition, text translation, and text-to-speech functionalities to enable seamless multilingual communication.

#### 1.2 Scope

The program is designed for real-time language translation, targeting use cases such as:

- International meetings
- Language learning
- Multilingual customer support

The application utilizes Python libraries such as `SpeechRecognition`, `Googletrans`, `gTTS`, and `Streamlit` for processing speech, translating text, and delivering audio output. The system is accessible via a simple user interface and can handle multiple languages.

#### 1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **TTS**: Text-to-Speech
- **GTTS**: Google Text-to-Speech
- **UI**: User Interface
- **SpeechRecognition**: A Python library for capturing and transcribing speech.
- **Googletrans**: A library for translating text using Google Translate.

#### 1.4 References

- Python Documentation: https://www.python.org/doc/
- SpeechRecognition Documentation: https://pypi.org/project/SpeechRecognition/
- Googletrans Documentation: https://pypi.org/project/googletrans/
- Streamlit Documentation: https://docs.streamlit.io/

---

## 2. Overall Description

#### 2.1 Product Perspective

The system is a standalone application built using Python and deployable on local machines or web servers. It incorporates real-time audio processing, translation, and playback.

#### 2.2 Product Features

1. **Real-Time Speech Recognition:**
   - Converts spoken input into text using the SpeechRecognition library.
2. **Language Translation:**
   - Translates the transcribed text into a target language using Googletrans.
3. **Text-to-Speech Conversion:**
   - Converts translated text into speech using gTTS.
4. **User Interface:**
   - Provides a simple interface for selecting source and target languages and starting/stopping the translation process.

#### 2.3 User Classes and Characteristics

- **General Users:**
  - Require real-time language translation for conversations.
  - Non-technical, relying on a simple UI for interaction.
- **Developers:**
  - Use the program as a base for integrating advanced translation features.

#### 2.4 Operating Environment

- **Software:**
  - Python 3.8.5 or later
  - Required Libraries: `SpeechRecognition`, `pygame`, `gTTS`, `Streamlit`, `googletrans`
- **Hardware:**
  - Microphone for capturing speech.
  - Speakers for audio output.
- **Platform:**
  - Works on Windows, MacOS, and Linux.

#### 2.5 Design and Implementation Constraints

- Requires an active internet connection for Google Translate and gTTS services.
- Limited by the accuracy of third-party libraries for speech recognition and translation.
- Dependency on the performance of the user's hardware (microphone and speakers).

#### 2.6 Assumptions and Dependencies

- The user has a working microphone and speakers.
- Users will have a basic understanding of how to select source and target languages.

---

## 3. Specific Requirements

#### 3.1 Functional Requirements

1. The system must recognize spoken input using the SpeechRecognition library.
2. The system must translate text using the Googletrans module.
3. The system must convert translated text into audio using gTTS.
4. The system must allow users to:
   - Select source and target languages.
   - Start and stop the translation process via UI buttons.

#### 3.2 Non-Functional Requirements

1. **Performance:**
   - The translation process, from speech input to audio output, should occur within 5 seconds.
2. **Usability:**
   - The user interface must be simple and intuitive, with dropdowns for language selection and clear instructions.
3. **Reliability:**
   - The system must handle errors gracefully, such as microphone unavailability or internet disconnection.
4. **Scalability:**
   - The system should support additional language modules without major code changes.

#### 3.3 External Interface Requirements

1. **User Interface:**
    - Dropdown menus for language selection.
    - Start and Stop buttons for controlling the translation process.
    - A placeholder to show system status (e.g., "Listening," "Processing").
2. **Hardware Interfaces:**
    - Microphone for input.
    - Speakers or headphones for output.

#### 3.4 System Features

1. **Speech Input:**
    - Captures speech input for up to 10 seconds per phrase.
2. **Translation:**
    - Supports all languages available in Google Translate.
3. **Audio Output:**
    - Plays translated speech using the pygame mixer module.

---

## 4. System Architecture

The system follows a modular architecture:

1. **Speech Recognition Module:**
    - Captures and converts speech to text.
2. **Translation Module:**
    - Translates the text into the desired language.
3. **Text-to-Speech Module:**
    - Converts the translated text into audio.
4. **UI Module:**
    - Provides controls for user interaction.
5. **Integration Module:**
    - Handles the sequence of tasks (speech recognition → translation → audio playback).

---

## 5. Validation and Testing

#### 5.1 Test Cases

1. **Speech Recognition:**
    - Test if the system accurately captures and transcribes speech.
2. **Translation Accuracy:**
    - Verify if translations are correct for multiple languages.
3. **Audio Output:**
    - Check if the audio output matches the translated text.
4. **Error Handling:**
    - Simulate scenarios such as no internet connection or unrecognized speech and ensure the system handles them gracefully.

#### 5.2 User Acceptance Testing

- The system will be tested with real users speaking different languages to verify its usability and accuracy.

---

### 6. Future Enhancements

1. Support offline speech recognition and translation.
2. Add functionality for text input (in addition to speech).
3. Include a feedback mechanism for users to report inaccuracies.
4. Improve performance by integrating multi-threading or asynchronous processing.

---

## 7. Appendices

- **Third-Party Libraries:**
  - `SpeechRecognition` for speech-to-text.
  - `googletrans` for translation.
  - `gTTS` for text-to-speech.
  - `pygame` for playing audio.
  - `Streamlit` for the user interface.