Foss Fest Hackathon 2025

Problem Statement Title:

Speech Recognition

Team Name:

Smarticle

Problem Statement

- Many users struggle with typing due to physical limitations, multitasking, or language barriers.
- Existing voice-to-text tools are often embedded in complex ecosystems or lack a simple, chat-style interface.
- There's a need for an intuitive, lightweight desktop app that converts speech to text in real time.

Project Introduction

- This project is a **desktop-based voice-to-text chat application** built using Python and Tkinter.
- It captures spoken input via microphone and transcribes it using Google's Speech Recognition API.
- The transcribed text is displayed in a scrollable chat interface, mimicking a conversation flow.

Relevance

- Enhances accessibility for users with motor impairments or dyslexia.
- Useful for note-taking, journaling, or real-time transcription during meetings.
- Aligns with the goals of the Lomiri Hackathon by promoting inclusive and open-source solutions.

Solution Approach

- Use speech_recognition to capture and transcribe audio.
- Display transcribed text in a chat-style GUI using tkinter.
- Run recognition in a separate thread to keep the UI responsive.
- Provide real-time feedback via a status bar and system messages.

Security Measures

- Microphone access is limited to runtime only; no audio is stored locally.
- API errors and recognition failures are handled gracefully to avoid crashes.
- No user data is transmitted beyond the Google API for transcription.

Technology Stack

- **Programming Language**: Python
- **GUI Framework**: Tkinter
- **Speech Recognition**: speech_recognition library (Google Web Speech API)
- **Audio Input**: PyAudio
- **Threading**: Python threading module for non-blocking recognition
- **Styling**: Custom style.py module for colors, fonts, and emojis
- **Text Display**: ScrolledText widget for chat-style interface

Advantages

- Simple and intuitive interface.
- Real-time transcription with minimal latency.
- Lightweight and cross-platform (runs on any OS with Python).
- Modular styling for easy customization.

Challenges & Solution

•**UI Freezing During Recognition**
Solution: Used multithreading to run speech recognition in the background, keeping the GUI responsive.
•**Ambient Noise Interference**
Solution: Applied adjust_for_ambient_noise() to calibrate the microphone before listening.
•**Unrecognized Speech or Silence**
Solution: Handled UnknownValueError with friendly system messages and retry prompts.
•**Continuous Listening Loop Management**
Solution: Designed a loop with graceful exit conditions and clear user feedback.
•**Visual Clarity in Chat Display**
Solution: Implemented role-based color tagging and emoji cues for better readability.

Conclusion

This project demonstrates how voice technology can be integrated into desktop applications to improve accessibility and productivity. It’s a lightweight, user-friendly tool that bridges the gap between speech and text in real time.

Future Improvements

- Add multilingual support and language selection.
- Integrate offline recognition using CMU Sphinx.
- Enable saving transcripts to file (e.g., .txt or .docx).
- Add voice command triggers (e.g., "Stop listening", "Save note").
- Enhance UI with themes and dark mode.

References

- Speech recognition library
- Google Speech-to-Text API
- Tkinter documentation
- PyAudio