# Speech-to-Text & Real-Time Transcription Project

## 1. Approach & Methodology

This project focuses on implementing a real-time speech-to-text system using Python, leveraging pre-trained Whisper ASR models. The system supports both real-time transcription and audio file-based transcription, integrating an interactive frontend for user-friendly accessibility.

## 2. Data Preprocessing & Selection

For optimal speech recognition, audio data needs preprocessing. This includes:  
- \*\*Resampling\*\* audio to 16kHz (required for Whisper models).  
- \*\*Noise Reduction\*\* for improving transcription accuracy.  
- \*\*Conversion to Mel-Spectrograms\*\* (handled internally by the model).  
- \*\*Dataset Selection:\*\* Audio samples were either recorded in real-time or uploaded for transcription.

## 3. Model Architecture & Tuning Process

The model used for transcription is OpenAI's Whisper ASR. The approach involved:  
- \*\*Choosing Whisper's Pre-Trained Model:\*\* Different model sizes (tiny, base, small, medium, large) were tested.  
- \*\*Fine-Tuning (Optional):\*\* The model was fine-tuned with a custom dataset if necessary.  
- \*\*Error Correction:\*\* Basic text processing was applied to refine the output and fix punctuation inconsistencies.

## 4. Performance Results & Next Steps

### Performance Results:  
- \*\*Accuracy:\*\* The system performs well on clear speech, but struggles slightly with noisy environments.  
- \*\*Latency:\*\* Near real-time performance is achieved for short utterances.  
- \*\*Limitations:\*\* The model may misinterpret words in dialects it wasn’t trained on.  
  
### Next Steps:  
- \*\*Enhancing real-time processing\*\* by optimizing the inference pipeline.  
- \*\*Improving dialect adaptation\*\* by training on region-specific datasets.  
- \*\*Deploying as a web app\*\* with Streamlit or Flask for broader accessibility.