Audio Transcription Application Utilizing WhisperAI

Rishit Mohan- 22BCE2422
Rhea Bhatia- 22BCE2657
Arin Parmod Singla- 22BDS0212

1. Problem Statement

In a world dominated by digital audio—ranging from podcasts to recorded lectures and meetings—manual transcription is time-consuming and error-prone. Users require an efficient, accurate, and user-friendly transcription tool. The challenge is to create a web-based application capable of accepting audio files, processing them with high accuracy, and producing text output with minimal latency and technical setup. OpenAI's Whisper, a powerful speech-to-text model, provides a potential solution when paired with an intuitive front-end like Streamlit.

2. Goals and Scope

Primary Goal:
To develop a user-centric web application that performs accurate and high-fidelity speech-to-text conversion.

Scope:

- Support for common audio formats (MP3 and WAV)
- Built-in audio preprocessing (resampling and conversion)
- Integration with Whisper's inference pipeline
- Option to download the resulting transcript (e.g., .srt format)

Target Audience:
Content creators, educators, researchers, journalists, accessibility advocates, and professionals seeking fast, reliable audio transcription with minimal setup.

3. Requirements

Functional Requirements

- Upload `.mp3` or `.wav` files
- Preprocess audio into 16 kHz mono using Librosa
- Transcribe audio using OpenAI Whisper
- Display transcribed text in the UI
- Provide option to download transcript as `.srt`
- Show spinner/progress during transcription

Non-Functional Requirements

- **Performance:** Transcribe ~5-minute files in under 1 minute on standard systems

- **Usability:** Interface accessible to non-technical users
- **Compatibility:** Cross-platform support (Windows, macOS, Linux)
- **Security:** Temporary files auto-deleted post-processing
- **Maintainability:** Modular, documented codebase
- **Scalability:** Extensible to cloud-based/multi-user setups
- **Reliability:** Conflict-free temporary file handling using Python's tempfile module

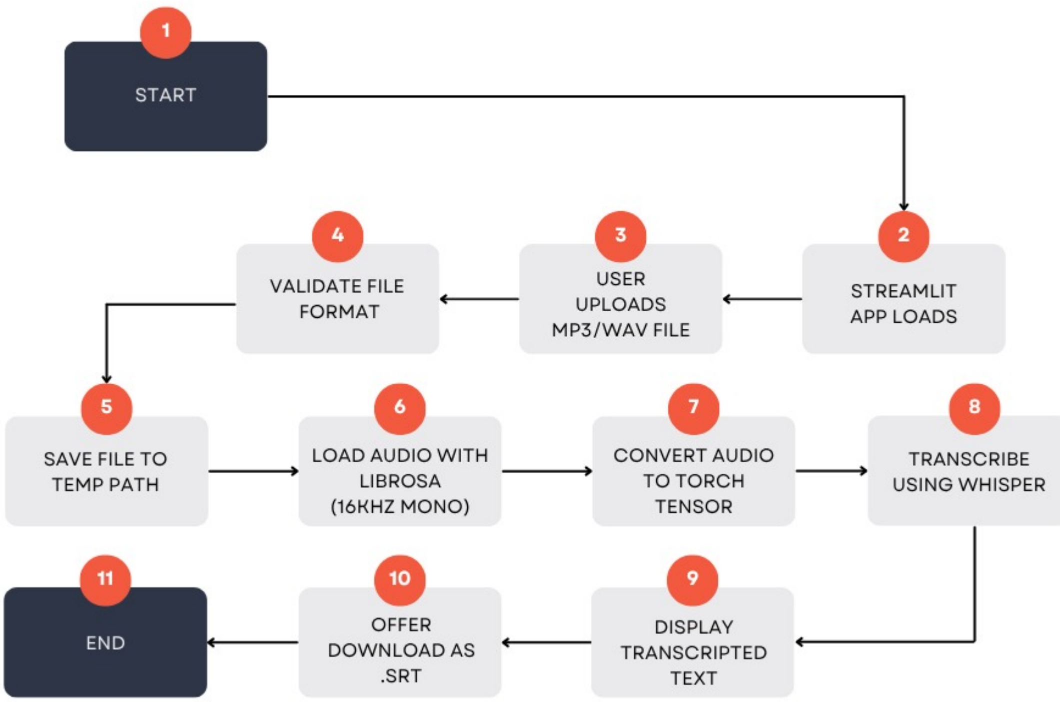
4. Project Planning and Scheduling

Day	Phase	Tasks
1	Requirement & Setup	Define scope, install Python & FFmpeg, create venv, install dependencies
2	UI & Upload Handling	Build Streamlit UI, file upload & validation, add spinner
3	Transcription Logic	Load model, integrate Librosa, run transcription, display transcript
4	Download & Testing	Add download feature, test edge cases, validate with various inputs
5	Documentation & Cleanup	Modularize code, comment codebase, create README, UI polish

5. Technology Stack

- **Python 3.8+** – Programming language
- **Streamlit** – Interactive web UI framework
- **OpenAI Whisper** – Speech recognition model
- **Librosa** – Audio resampling & preprocessing
- **PyTorch** – Model backend
- **tempfile** – Secure, temporary file handling

6. System Workflow



7. Prerequisites

- Python 3.8 or higher
- FFmpeg installed and added to system PATH
- IDE like VS Code or PyCharm for development

8. Setup Instructions

- `Create virtual environment`
  ```
  python -m venv venv
  ```
- `Activate environment`
  ```
  On macOS/Linux: source venv/bin/activate
  On Windows:venv\Scripts\activate
  ```

- `Install dependencies`

  ```
  pip install streamlit openai-whisper librosa torch
  ```

- `Run application`

  ```
  streamlit run app.py
  ```

9. Core Features

- **Multi-format Upload**: Supports MP3 & WAV
- **Live Transcription Feedback**: Real-time spinner during processing
- **Accurate Transcription**: High precision, even in noisy inputs
- **Downloadable Output**: Generate `.srt` subtitle files for reuse

10. Code (Simplified Example)

```
import streamlit as st
import whisper
import librosa
import torch
import tempfile

model = whisper.load_model("base")

def transcribe(audio_file):
    file_ext = audio_file.name.split('.')[-1].lower()
    if file_ext not in ("mp3", "wav"):
        st.error("Unsupported format. Please upload MP3 or WAV.")
        return None

    with tempfile.NamedTemporaryFile(delete=False, suffix=f".{file_ext}")
as tmp:
        tmp.write(audio_file.read())
        temp_path = tmp.name

    y, _ = librosa.load(temp_path, sr=16000, mono=True)
    audio = torch.from_numpy(y).float()
    result = model.transcribe(audio)
    return result["text"]
```

```
def main():
    st.title("Whisper AI Transcription App")
    uploaded_file = st.file_uploader("Upload MP3 or WAV", type=["mp3",
"wav"])

    if uploaded_file:
        with st.spinner("Transcribing..."):
            transcript = transcribe(uploaded_file)
            if transcript:
                st.success("Transcription complete!")
                st.write(transcript)
                st.download_button("Download Transcript", transcript,
file_name="transcript.srt")

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

11. Testing

Test Case	Expected Result	Status
Upload valid `.mp3` file	Successful transcription	✅
Upload valid `.wav` file	Successful transcription	✅
Upload unsupported format	Error message shown	✅
No FFmpeg installed	Show installation error	✅
Upload empty file	Graceful error handling	✅
Verify download option	File downloads successfully	✅

12. Advantages and Disadvantages

Advantages

- High transcription accuracy
- Simple, user-friendly UI
- Supports MP3 & WAV
- Downloadable subtitle transcripts
- Secure, temp file handling

Disadvantages

- Local deployment doesn't scale well
- Requires setup of Python and FFmpeg
- Not ideal for real-time or very large files
- Performance depends on hardware

13. Challenges Encountered

- **Whisper Installation:** Couldn't be installed via basic `pip install whisper`; required PyPI or GitHub methods.
- **Format Compatibility:** Whisper only accepts 16 kHz mono audio; required Librosa integration.
- **Latency Issues:** Long files caused delays; solved using progress indicators.
- **File Handling:** Used `tempfile` to avoid conflicts and ensure cleanup.

14. Conclusion

The WhisperAI Transcription Application provides a powerful, easy-to-use platform for audio-to-text conversion. With a blend of OpenAI Whisper and Streamlit, users can reliably transcribe audio with high accuracy and minimal effort. This project serves as a foundation for future enhancements like cloud hosting or real-time streaming support.

15. References

- OpenAI Whisper GitHub
- Streamlit Documentation
- Librosa Audio Library
- https://openai.com/index/whisper/
- https://docs.pytorch.org/docs/stable/index.html