AudioBook Generator

1. Introduction / Objective

AudioBook Generator is a web application that allows users to upload one or more text documents (PDF, DOCX, TXT) and automatically converts them into high-quality audiobooks. The application leverages Large Language Models (LLMs) to rewrite extracted text in an engaging, listener-friendly "audiobook style" before using open-source Text-to-Speech (TTS) technology to produce downloadable audio files. This project enhances accessibility, productivity, and the enjoyment of written content.

---

2. Methodology / Workflow

1. **User Uploads Documents**
   - Users select and upload one or more documents through an interactive Streamlit web interface.
2. **Text Extraction**
   - The backend parses uploaded files and extracts text content:
     - PDF: `PyPDF2` or `pdfplumber`
     - DOCX: `python-docx`
     - TXT: Native file reading
3. **LLM-Based Text Enrichment**
   - Extracted text is processed by a Large Language Model (e.g., OpenAI API, Gemini API, or open-source LLM) to rewrite the text for better narration and listener experience.
   - Example LLM prompts: "Rewrite this text for an engaging audiobook narration."
4. **Text-to-Speech Conversion**
   - The enriched text is fed into an open-source TTS library (such as `pyttsx3`, `Coqui TTS`, or `Tortoise TTS`), producing a high-quality `.mp3` or `.wav` audio file.
5. **Audio Download**
   - The generated audio file is presented for immediate download within the Streamlit UI.

---

3. Modules

- **Document Upload Module:** Handles file uploads via Streamlit.
- **Text Extraction Module:** Extracts raw text from PDFs, DOCX, and TXT files.
- **LLM Enrichment Module:** Calls the LLM to rewrite and enhance extracted text.
- **Text-to-Speech Module:** Converts enriched text into audio using a TTS library.
- **Audio Delivery Module:** Provides the final audio file to the user for download.

4. Week-wise Module Implementation and High-Level Requirements

Weeks 1–2:

- Set up environment and install dependencies.
- Implement file upload and multi-format text extraction.

Weeks 3–4:

- Integrate LLM for audiobook-style text rewriting.
- Build API connection between Streamlit and backend LLM processing.

Weeks 5–6:

- Integrate and test open-source TTS conversion.
- Ensure support for different voice options and error handling.

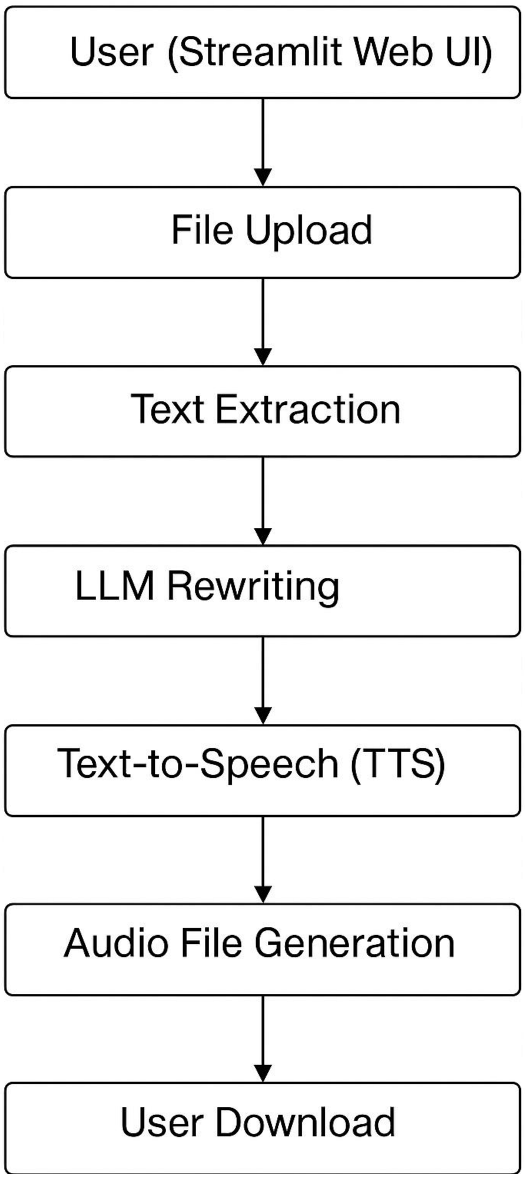
Weeks 7–8:

- Finalize UI/UX in Streamlit.
- Conduct thorough testing, optimize performance, and complete documentation.

5. Evaluation Criteria

- **Milestone 1 (Week 2):**
  File upload and accurate text extraction operational.
- **Milestone 2 (Week 4):**
  LLM-based text rewriting working and demonstrably improving narration.
- **Milestone 3 (Week 6):**
  Audio file generation (from rewritten text) stable and high-quality.
- **Milestone 4 (Week 8):**
  Full application workflow—document upload to audio download—operational, user-friendly, and documented.

6. Design / Architectural Diagram



7. Technology Stack

- **Frontend:** Streamlit
- **Backend:** FastAPI or Flask (optional, for modularity or scale)
- **Text Extraction:** PyPDF2, pdfplumber, python-docx
- **LLM Integration:** OpenAI API, Gemini API, or local open-source LLM
- **Text-to-Speech:** pyttsx3, Coqui TTS, Tortoise TTS, or gTTS
- **Programming Language:** Python 3.x