PodifyAI – Technical Blueprint Report

Course: EEE 6778 – Applied Machine Learning II
Author: Sainath Chettupally
Date: October 25, 2025

Git_Repo: https://github.com/SainathChettupally/PodifyAI.git

1. Introduction and Problem Context

In today's fast-paced professional and academic environment, people face a constant flood of information long research papers, corporate reports, and meeting notes. Even with digital productivity tools, attention spans and available reading time continue to shrink. Students, researchers, and professionals often struggle to extract insights efficiently, leading to information overload, slower decision-making, and accessibility barriers for visually impaired or auditory learners.

PodifyAI addresses this challenge by transforming textual documents into short, natural-sounding audio summaries. It lets users upload any text-based file (PDF, DOCX, PPTX, TXT, MD, HTML, CSV) and choose between Quick, Standard, or Deep summary modes. Each mode balances brevity and detail to fit the listener's time and purpose. By blending advanced NLP summarization with text-to-speech synthesis, PodifyAI turns static documents into dynamic, portable podcasts bridging the gap between written and auditory learning.

2. Objectives

1. Summarize multi-format documents using a transformer-based model (DistilBART/BART-Large).
2. Provide natural audio output through a lightweight text-to-speech pipeline.
3. Offer a user-friendly interface for seamless upload, summary generation, and playback.
4. Ensure reproducibility and scalability through clean modular code and environment management.
5. Lay the foundation for future integration of personalization, multilingual voices, and Edge AI optimization.

3. Dataset and Input Handling

3.1 Sources and Formats

PodifyAI accepts user-uploaded files rather than a fixed dataset.
Supported types include:

- PDF via PyMuPDF (fitz)

- DOCX via python-docx
- PPTX via python-pptx
- TXT/MD via direct read
- HTML via BeautifulSoup + lxml
- CSV (text columns concatenated)

3.2 Pre-processing Pipeline

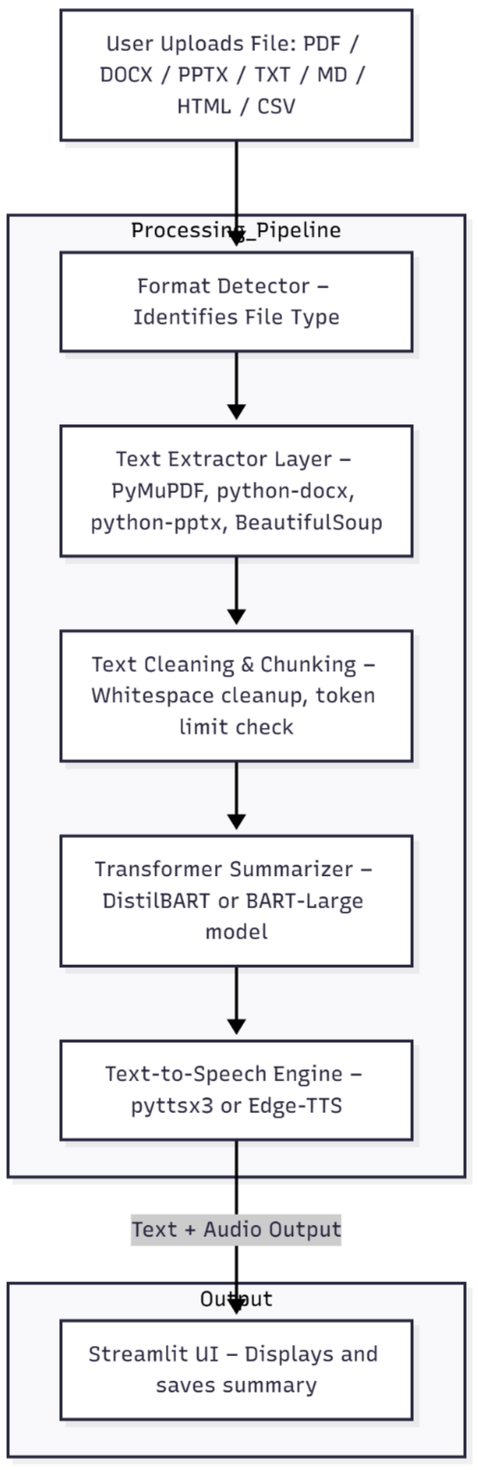
1. File Detection: Extension-based router selects the correct extractor.
2. Text Extraction: Parsers convert structured formats into clean text.
3. Cleaning: Whitespace normalization, sentence segmentation, and removal of tags or non-text tokens.
4. Truncation & Chunking: Long inputs split to stay within the model's 1 k-token limit.
5. Sanity Check: Preview - first 800 characters displayed in the UI.

3.3 Ethical and Privacy Considerations

- All processing is local; no external API calls.
- Users retain full ownership of uploaded files.
- No content storage or analytics are performed beyond the local session.

4. Planned Architecture

4.1 System Overview


User Uploads File: PDF / DOCX / PPTX / TXT / MD / HTML / CSV
Processing_Pipeline
Format Detector – Identifies File Type
Text Extractor Layer – PyMuPDF, python-docx, python-pptx, BeautifulSoup
Text Cleaning & Chunking – Whitespace cleanup, token limit check
Transformer Summarizer – DistilBART or BART-Large model
Text-to-Speech Engine – pyttsx3 or Edge-TTS
Text + Audio Output
Output
Streamlit UI – Displays and saves summary

4.2 Core Modules

Module	Purpose	Technologies
extractors.py	Handles format detection and text extraction for multiple file types	PyMuPDF, python-docx, python-pptx, BeautifulSoup
summarizer.py	Performs abstractive summarization using pre-trained transformer pipelines	Hugging Face Transformers, PyTorch
tts_engine.py	Converts summarized text into placeholder audio output	pyttsx3 (offline TTS) / Edge-TTS (optional)
app.py	Provides the interactive web UI for uploading and summarizing files	Streamlit
setup.ipynb	Verifies environment, extraction, summarization, and saving	Jupyter Notebook

4.3 Model Design

- Base Model: sshleifer/distilbart-cnn-12-6 for fast inference.
- Upgrade Path: facebook/bart-large-cnn or flan-t5-large for higher quality.
- Chunk Summarization: Documents exceeding ~1 k tokens are split into segments. Each segment is summarized independently; summaries are merged and re-compressed into a final coherent output.
- Output Length Control: Dynamic max_length and min_length adjusted by summary mode.

5. User Interface and Experience

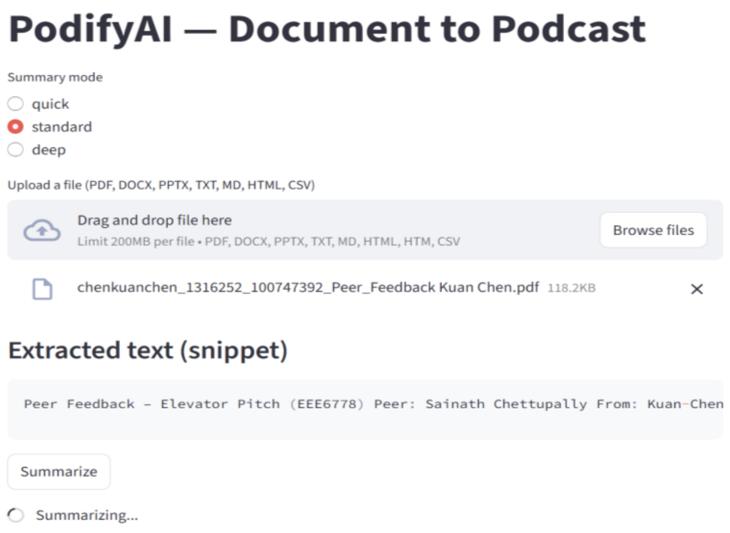
5.1 Interaction Flow

1. Upload File → User selects any supported document.
2. Select Mode → Quick (≈ short), Standard (medium), Deep (long).
3. Generate Summary → System runs extraction → cleaning → summarization.
4. View & Download → Summary displayed on-screen; text saved under results.
5. Generate Audio → TTS converts the summary text into an MP3 or placeholder TXT.

5.2 Design Philosophy

- Single-page, distraction-free layout.
- Clear progress feedback (“Summarizing…” spinner).
- Minimal clicks: one upload → one output.
- Supports quick prototyping for future Gradio or mobile versions.

5.3 Wireframe (Basic UI)



PodifyAI — Document to Podcast

Summary mode
quick
[✓ standard]
deep

Upload a file (PDF, DOCX, PPTX, TXT, MD, HTML, CSV)

Drag and drop file here
Limit 200MB per file • PDF, DOCX, PPTX, TXT, MD, HTML, HTM, CSV

chenkuanchen_1316252_100747392_Peer_Feedback Kuan Chen.pdf 118.2KB

Extracted text (snippet)

```
Peer Feedback – Elevator Pitch (EEE6778) Peer: Sainath Chettupally From: Kuan-Chen
```

6. Innovation and Anticipated Challenges

Challenge	Description	Mitigation
Long-Document Handling	Transformer models have ~1 k token limits.	Implement chunk-and-merge summarization; optional long-context models later.
Latency and Memory	BART-Large is slow on CPU.	Use DistilBART for Deliverable 1; quantization for Deliverable 2.
Audio Naturalness	Offline TTS can sound robotic.	Replace with Edge-TTS (neural) or Azure Cognitive Voices in future milestones.
Format Diversity	Parsing errors from poorly formatted PDFs/PPTs.	Fallback to plain-text and error logging.
Bias and Fairness	Summaries may reflect training-data bias.	Add transparency section and disclaimer in UI.

Unique Value

- Unified multi-format input pipeline (not restricted to PDFs).

- Modular NLP architecture ready for personalization and Edge deployment.
- Supports auditory learning and accessibility for diverse users.

7. Implementation Timeline

Week	Focus	Key Deliverable
Oct 20 – 26	Data pipeline + environment setup	Working extractor and baseline summary demo
Oct 27 – Nov 2	Summarization pipeline tuning	Stable chunking and summary quality tests
Nov 3 – 9	TTS integration and UI prototype	Functional Streamlit demo
Nov 10 – 23	Evaluation + model optimization	Improved accuracy and response time
Nov 24 – Dec 7	UX polish + documentation	Final user-ready UI and README
Dec 8 – 11	Presentation and submission	Demo + Final report upload

8. Responsible AI Reflection

PodifyAI demonstrates how machine learning can improve information accessibility when implemented responsibly.

Fairness & Bias:
The summarization model is pre-trained on large open-domain corpora that may contain social or topical biases. All outputs are clearly marked as machine-generated. Future iterations will include prompt-level fairness filters.

Transparency:
The UI discloses the model name and version. Summaries remain editable, allowing users to verify fidelity.

Privacy & Security:
No cloud storage or third-party APIs are used. All processing occurs locally; uploaded files remain private.

Environmental Impact:
To reduce compute overhead, PodifyAI employs DistilBART for inference and plans to integrate model quantization and on-device caching in later milestones.

9. Future Work

- Integrate multilingual summarization (English → Spanish, Hindi etc.).
- Add keyword-guided or topic-aware summaries.

- Deploy as a PWA or mobile app for offline audio learning.
- Integrate speaker customization and emotion tone control.
- Extend to video transcripts (YouTube/MP4 input) using Whisper ASR.

10. Conclusion

Deliverable 1 establishes PodifyAI's foundation—from data ingestion and summarization logic to an interactive UI and reproducible environment.
It demonstrates that a modular, transformer-based pipeline can effectively summarize multi-format documents and generate audio output locally.
Subsequent milestones will focus on model optimization, user evaluation, and broader deployment, transforming PodifyAI into a full-fledged intelligent audio-assistant platform for modern learners and professionals.