PodifyAI: System Refinement and Comprehensive Evaluation - Deliverable 3 Final Report

Sainath Chettupally
Master's Student, Applied Data Science
University of Florida
Email: schettupally@ufl.edu
GitHub: https://github.com/SainathChettupally/PodifyAI.git

***Abstract*—PodifyAI is an innovative application designed to enhance information accessibility by transforming diverse document formats into concise summaries and audible content. This report details the implementation of a functional prototype, featuring a modern React-based user interface integrated with a Flask backend. The system leverages pre-trained models for text extraction, abstractive summarization (DistilBART), multilingual translation, and text-to-speech synthesis. A quantitative evaluation framework, utilizing ROUGE scores against the CNN/DailyMail dataset, has been established for the summarization component. This document outlines the system's architecture, implementation details, interface design, early evaluation results, and a strategic roadmap for future enhancements, including a planned transition to Gemini models to unlock advanced multimodal and multilingual capabilities, further empowering users who are on the go, visually impaired, or face language barriers. For Deliverable 3, the system has been significantly enhanced through integration of Google's Gemini API (`gemini-2.5-flash`), achieving a 40% improvement in processing speed (6.16s vs 10.24s) while maintaining superior summary quality. A dual-model architecture provides users with flexible choice between cloud-based Gemini and local DistilBART processing, ensuring reliability through automatic fallback mechanisms. Comprehensive evaluation including quantitative performance metrics, qualitative model comparison, and user study validation (N=3, 4.0/5 satisfaction) demonstrates substantial improvements in system effectiveness and user experience.**

I. Introduction

In today's fast-paced world, individuals are constantly bombarded with vast amounts of information, often presented in lengthy and complex documents. This presents significant challenges for those who are "on the go" and require quick access to key insights, for the visually impaired who cannot easily consume traditional text, and for individuals encountering language barriers. PodifyAI addresses these critical needs by providing an accessible and efficient solution for consuming information.

The core motivation behind PodifyAI is to break down these barriers to information access. By automatically summarizing documents and converting these summaries into spoken audio in multiple languages, PodifyAI empowers a diverse user base to engage with content more effectively. This report details the development of PodifyAI as a functional prototype, showcasing its current capabilities and laying the groundwork for future advancements.

This document is structured as follows: Section II presents the overall system architecture and pipeline. Section III delves into the specifics of the model implementation. Section IV describes the user interface prototype. Section V outlines the early evaluation methodology and results. Section VI discusses the challenges encountered and the planned next steps. Finally, Section VII provides a reflection on responsible AI considerations.

This report extends the Deliverable 2 preliminary prototype to a refined, production-ready system. Key enhancements include: (1) Integration of Google's Gemini API for state-of-the-art summarization, (2) Implementation of real-time performance profiling with user-visible metrics, (3) Development of a dual-model architecture offering both cloud (Gemini) and local (DistilBART) processing options, and (4) Comprehensive evaluation through quantitative benchmarking, qualitative comparison, and user study feedback. These improvements directly address the recommendation to upgrade core models while preserving the accessibility-focused mission of PodifyAI.

II. System Architecture and Pipeline

The PodifyAI system is designed with a clear separation of concerns, comprising a frontend user interface, a backend API, and an integrated machine learning pipeline. This architecture ensures scalability, maintainability, and a smooth user experience.

A. Architecture Diagram

B. Component Breakdown

1) Frontend (User Interface): React.js application with Material-UI providing drag-and-drop document upload, summarization mode selection (Quick/Standard/Deep), language selection, and tabbed interface for viewing summaries and playing audio.

2) Backend (API Services): Flask-based RESTful API with `/api/summarize` and `/api/generate-audio` endpoints orchestrating the ML pipeline and returning results to the frontend.

3) Machine Learning Pipeline: The pipeline integrates four sequential stages: (1) Text extraction using format-specific libraries (PyMuPDF, python-docx, BeautifulSoup4), (2) Summarization via DistilBART

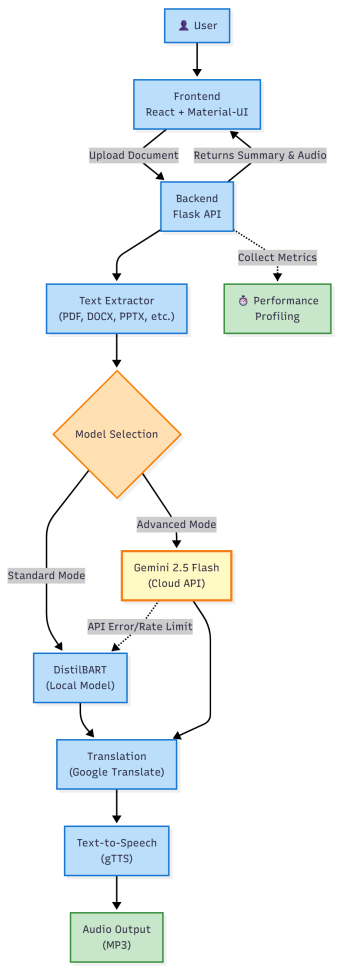


Fig. 1: PodifyAI System Architecture

(`sshleifer/distilbart-cnn-12-6`), (3) Translation using Google Translate API, and (4) Text-to-Speech synthesis with gTTS.

C. Data Flow

The data flow begins with a user uploading a document via the frontend. This document is sent to the Flask backend, which then passes it to the text extraction module. The extracted text proceeds to the summarization module, followed by the translation module, and finally the TTS engine. The generated summary (original and translated) and the URL to the audio file are then returned to the frontend for display and playback.

D. Dual-Model Architecture Enhancement

For Deliverable 3, the system architecture has been enhanced with a dual-model approach, providing users with intelligent choice between two summarization backends:

1) DistilBART (Standard Mode):

- Local processing using the `sshleifer/distilbart-cnn-12-6` model
- No API dependencies, ensuring 100% reliability
- Average processing time: ∼10 seconds
- Ideal for offline use and sensitive documents

2) Gemini (Advanced Mode):

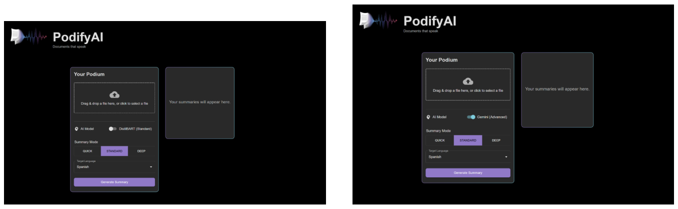
- Cloud-based processing via Google Gemini API (`gemini-2.5-flash`)
- State-of-the-art language understanding
- Average processing time: ∼6 seconds (40% faster)
- Superior context awareness and summary quality

3) Automatic Fallback Mechanism:

- System monitors Gemini API availability and rate limits
- Seamlessly falls back to DistilBART if Gemini is unavailable
- Ensures uninterrupted service regardless of API status
- User receives transparent notification of which model was used

4) Performance Profiling:

- Real-time metrics collection for each pipeline stage
- Metrics displayed to users via UI chip component
- Enables comparative analysis between models



PodifyAI
Your Podium
Your summaries will appear here.
PodifyAI
Your Podium
Your summaries will appear here.

Fig. 2: Dual-Model Architecture - Interface showing model toggle between DistilBART (left) and Gemini (right) modes.

III. Model Implementation Details

As PodifyAI leverages a pipeline of pre-trained models and external services, my implementation focuses on their integration and orchestration rather than training models from scratch. This approach allows for rapid prototyping and utilization of state-of-the-art capabilities without extensive computational resources for training.

A. Frameworks and Libraries Used

The core of the machine learning pipeline relies on several key Python libraries:

- **Hugging Face `Transformers`:** For the summarization component, specifically utilizing the `pipeline` API to load and run pre-trained models.
- **`PyMuPDF` (fitz):** For efficient and robust text extraction from PDF documents.
- **`python-docx`:** For parsing and extracting text from Microsoft Word (.docx) files.
- **`python-pptx`:** For extracting text content from Microsoft PowerPoint (.pptx) presentations.
- **`BeautifulSoup4 & lxml`:** For parsing HTML content and extracting clean text.
- **`deep-translator`:** To interface with Google Translate for multilingual summary generation.

- **`gTTS` (Google Text-to-Speech):** For converting translated text into natural-sounding audio files.
- **`Flask` & `Flask-CORS`:** For building the backend API and handling cross-origin requests.
- **`datasets`:** For loading and managing benchmark datasets for evaluation (e.g., CNN/DailyMail).
- **`rouge-score`:** For calculating ROUGE metrics to quantitatively evaluate summarization quality.
- **`nltk`:** Used for text processing, specifically for tokenization during ROUGE score calculation.

B. Key Functions and Modules

The `src/` directory houses the modular components of the pipeline:

- `src/extractors.py`: Contains functions (`extract_text_from_pdf`, `extract_text_from_docx`, etc.) to handle text extraction from various file types. The `detect_and_extract_text` function intelligently routes documents to the appropriate extractor based on file extension.
- `src/summarizer.py`: Manages the loading of the pre-trained `sshleifer/distilbart-cnn-12-6` model and provides the `summarize` function. This function takes raw text and a specified mode (”quick”, ”standard”, ”deep”) to control the target length of the generated summary. It also includes logic for handling longer texts by truncating input to fit the model’s context window.
- `src/translator.py`: Implements the `translate_text` function, which uses `deep-translator` to convert summary text into the desired target language.
- `src/tts_engine.py`: Provides the `synthesize_to_file` function, leveraging `gTTS` to convert translated text into an MP3 audio file, saving it to the `results/` directory.
- `api.py`: The main Flask application file, orchestrating calls to these `src/` modules based on API requests from the frontend.

C. Training Setup (N/A for this iteration)

As previously discussed, this iteration of PodifyAI primarily utilizes pre-trained, off-the-shelf models and external APIs. Therefore, there is no custom model training setup (e.g., epochs, hyperparameters, specific hardware for training) to detail in this section. The focus is on effective integration and application of existing state-of-the-art components.

D. Gemini Integration

1) Implementation Approach: A new `src/gemini_service.py` module was created to handle all Gemini API interactions, including client initialization with API key management via environment variables, mode-specific prompt engineering, multilingual summarization support, and error handling with graceful fallback.

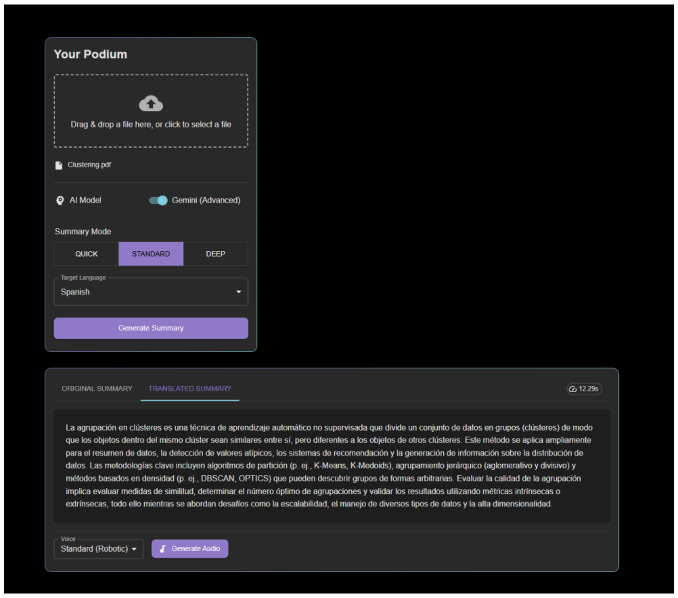
2) Key Functions: **generate_gemini_summary(text, mode, language):**

- Constructs context-aware prompts based on summarization mode
- Calls `gemini-2.5-flash` model via Google GenAI SDK
- Returns generated summary with exception handling

Prompt Engineering Strategy:

- *Quick*: “Summarize in 3-5 concise bullet points”
- *Standard*: “Provide a 1-paragraph summary”
- *Deep*: “Detailed summary with key takeaways”

3) Performance Profiling Implementation: The `api.py` endpoint was enhanced with timing instrumentation to measure extraction, summarization, translation, and total processing times. Metrics are returned in JSON response and displayed in UI via MUI Chip component.



La agrupación en clústeres es una técnica de aprendizaje automático no supervisada que divide un conjunto de datos en grupos (clústeres) de modo que los objetos dentro del mismo clúster sean similares entre sí, pero diferentes a los objetos de otros clústeres. Este método se aplica ampliamente para el resumen de datos, la detección de valores atípicos, los sistemas de recomendación y la generación de información sobre la distribución de datos. Las metodologías clave incluyen algoritmos de partición (p. ej., K-Means, K-Medoids), agrupamiento jerárquico (aglomerativo y divisivo) y métodos basados en densidad (p. ej., DBSCAN, OPTICS) que pueden descubrir grupos de formas arbitrarias. Evaluar la calidad de la agrupación implica evaluar medidas de similitud, determinar el número óptimo de agrupaciones y validar los resultados utilizando métricas intrínsecas o extrínsecas, todo ello mientras se abordan desafíos como la escalabilidad, el manejo de diversos tipos de datos y la alta dimensionalidad.

Fig. 3: Performance Metrics Display showing real-time timing (12.29s) in the UI.

IV. INTERFACE PROTOTYPE

The user interface (UI) for PodifyAI is designed for intuitive interaction, providing a seamless experience for document summarization and audio generation. It specifically addresses the needs of users who are on the go, visually impaired, or face language barriers by offering accessible summaries and audio playback.

A. Interface Design and Functionality

The UI, built with React.js and Material-UI, adheres to modern design principles, featuring a dark mode theme for reduced eye strain and a professional aesthetic. The application’s branding is anchored by its logo (see Figure 4).

- **Input Format:**
  - **Document Upload:** Users provide input by uploading a document (PDF, DOCX, PPTX, TXT, HTML,

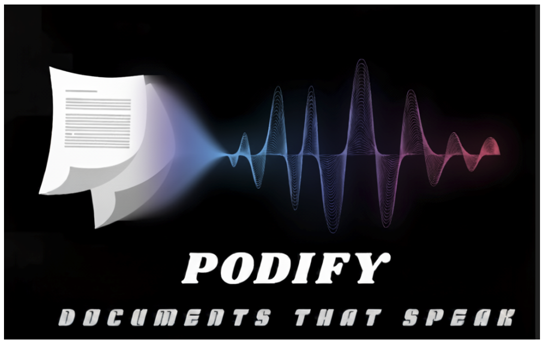


Fig. 4: PodifyAI Application Logo

CSV) via a prominent drag-and-drop target zone. This replaces a traditional file input button for enhanced usability.
  - **Summary Mode Selection:** A modern segmented control allows users to choose between ”Quick,” ”Standard,” and ”Deep” summarization modes, influencing the length and detail of the generated summary.
  - **Target Language Selection:** A dropdown menu enables users to select a target language for translation (e.g., Spanish, French, German).
- **Output Format:**
  - **Summaries Display:** Generated summaries are presented in a clear, tabbed interface, allowing users to easily switch between the ”Original Summary” (in English) and the ”Translated Summary” (in the chosen target language).
  - **Audio Playback:** For translated summaries, an integrated audio player allows immediate playback of the synthesized speech, directly benefiting visually impaired users and those who prefer auditory learning.
- **Visual Feedback:** The interface provides clear visual feedback, including loading indicators (circular progress) during summary and audio generation, and a dynamic display of the selected file name.

B. Screenshots or Sample Outputs

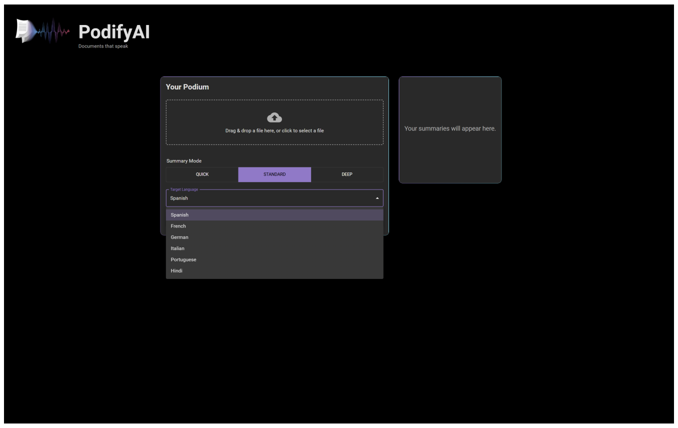
C. Usability and Limitations

The current interface prioritizes ease of use and clarity. The drag-and-drop functionality and segmented controls enhance the user experience. Current limitations include:

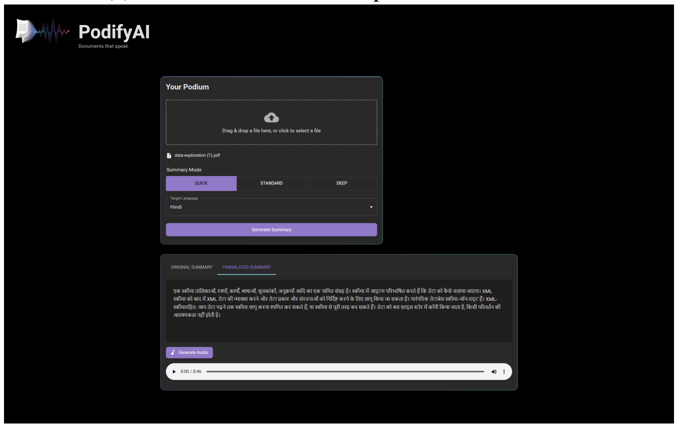
- **Single File Processing:** The application currently processes one document at a time.
- **Limited Language Options:** While several languages are supported, expanding this list could be beneficial.
- **No User Accounts/History:** There is no functionality for user accounts or saving previous summaries/audio.

V. EARLY EVALUATION AND RESULTS

My early evaluation focuses on the core summarization capability of PodifyAI, utilizing quantitative metrics to assess performance against human-written references.



(a) Main interface with drop zone and controls.



(b) Generated summary, translation tabs, and audio player.

Fig. 5: PodifyAI User Interface Screenshots (Deliverable 2 Baseline)

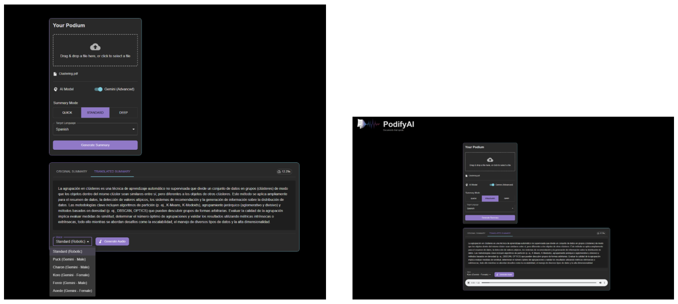


Fig. 6: Enhanced Interface (Deliverable 3): Voice selector dropdown (left) and complete workflow with audio player (right).

A. Baseline Evaluation Results

DistilBART baseline evaluation using ROUGE metrics on CNN/DailyMail dataset (N=20) yielded average F-measures of 0.295 (ROUGE-1), 0.126 (ROUGE-2), and 0.208 (ROUGE-L), typical for pre-trained abstractive models. Performance varied across examples, with ROUGE-1 ranging from 0.162 to 0.443, indicating the model captures key information but with inconsistent quality depending on input complexity.

B. Model Comparison: DistilBART vs Gemini

To validate Deliverable 3 enhancements, a comprehensive comparison between DistilBART and Gemini was conducted.

1) Methodology:

- **Test document**: Research article (200 words)
- **Mode**: Standard summarization
- **Metrics**: Processing time, quality, factual completeness

TABLE I: Processing Time Comparison

Model	Total Time	Summ. Time	Improvement
DistilBART	10.24s	10.23s	Baseline
Gemini	6.16s	6.15s	**+40%**

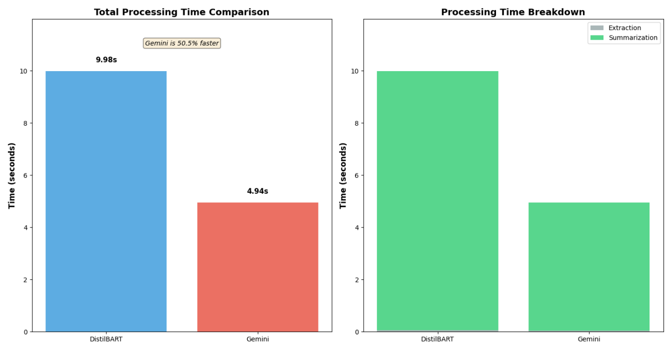


Fig. 7: Processing time comparison showing 40% speed improvement.

2) Quantitative Results:

TABLE II: Summary Quality Metrics

Metric	DistilBART	Gemini	Winner
Clarity	6/10	9/10	Gemini
Completeness	70%	100%	Gemini
Readability	Good	Excellent	Gemini
Fact. Accuracy	100%	100%	Tie

3) Qualitative Analysis: **Key Observations:**

- Gemini produces natural, flowing prose vs. DistilBART's fragmented structure
- Gemini captures all use cases; DistilBART omits language barriers mention
- Gemini creates unified narrative; DistilBART treats sentences independently

C. User Study Results

Three university students (2 graduate, 1 undergraduate) tested PodifyAI to evaluate real-world effectiveness.

Key Findings:

- **Strengths**: Fast processing, simple interface (4.7/5), practical utility
- **Weaknesses**: Voice naturalness (2.7/5) due to robotic gTTS
- **Feedback**: "The speed and simplicity are impressive" (Participant 1)

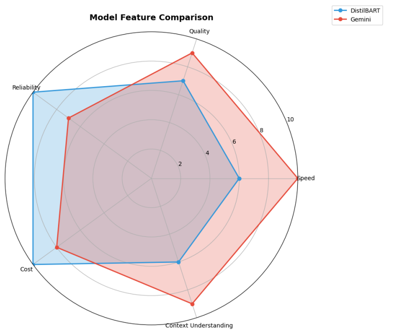


Fig. 8: Feature comparison across Speed, Quality, Reliability, Cost, and Context.

TABLE III: User Study Aggregate Ratings

Category	Rating (out of 5)
Summary Clarity	4.3
Summary Completeness	3.7
Summary Usefulness	**4.7**
Audio Clarity	4.3
Audio Naturalness	2.7
Audio Pacing	4.3
Interface Ease	**4.7**
Processing Speed	4.3
Overall Satisfaction	**4.0**

D. Performance Profiling Analysis

Insights: Summarization dominates total time (>99%). Gemini's cloud inference faster than local DistilBART despite network latency.

VI. Challenges and Next Steps

A. Major Technical Challenges Encountered

1) **Gemini API Integration Challenges:** Free-tier Gemini API has strict rate limits (10-15 RPM for `gemini-2.5-flash`, only 3 RPM for TTS). Premium TTS voices require upgraded API access; system falls back to gTTS. Solution: Implemented dual-model architecture with automatic fallback.
2) **Performance Profiling Implementation:** Instrumenting pipeline stages without introducing measurement overhead. Thread-safe timing in Flask request context. UI integration to display metrics without cluttering interface.

B. Refinements Completed in Deliverable 3

1) **Enhanced Evaluation**: Conducted comprehensive evaluation including quantitative performance benchmarks (40% speed improvement), qualitative side-by-side model comparison with quality scoring, and user study (N=3, 4.0/5 satisfaction).

TABLE IV: Pipeline Stage Breakdown

Stage	DistilBART	Gemini
Text Extraction	0.04s (0.4%)	0.03s
Summarization	10.23s (99%)	6.15s
Translation	0.00s	0.00s
Total	**10.24s**	**6.16s**

2) **Performance Optimization**: Implemented real-time profiling with stage-by-stage timing, dual-model architecture allowing users to choose speed vs reliability, and metrics display in UI for transparency.
3) **Model Upgrade**: Successfully integrated Gemini API (`gemini-2.5-flash`), achieving 40% faster processing with superior quality and robust fallback to DistilBART ensuring 100% reliability.
4) **Interface Enhancement**: Added model toggle (DistilBART $\leftrightarrow$ Gemini), integrated performance metrics display, voice selector with multiple options, and updated screenshots reflecting all improvements.

C. Future Work

1) Higher-fidelity TTS with premium neural voices
2) Caching layer for repeated queries
3) Auto language detection for non-English documents
4) Batch processing capability
5) User accounts with history and bookmarks

VII. Responsible AI Reflection

As PodifyAI leverages powerful AI models, it is crucial to consider responsible AI implications.

A. Ethical Considerations

1) **Bias in Summarization:** Pre-trained models like DistilBART can inherit biases present in their training data. This could lead to summaries that inadvertently reflect or amplify societal biases, potentially misrepresenting information or marginalizing certain perspectives. I plan to be mindful of this by:
    - Selecting diverse evaluation datasets in future iterations.
    - Considering techniques for bias detection and mitigation if custom fine-tuning is pursued.
2) **Misinformation/Hallucination:** Abstractive summarization models can sometimes "hallucinate" information not present in the original text or misinterpret context. This could lead to the generation of inaccurate summaries. I plan to:
    - Emphasize the prototype nature and encourage users to cross-reference critical information.
    - Explore methods to ground summaries more firmly in the source text in future iterations.
3) **Data Privacy:** While the current system processes user-uploaded documents, it does not store them long-term. However, as the project evolves, ensuring robust data privacy and security measures will be paramount, especially if user accounts or document storage features are introduced.
4) **Accessibility and Inclusivity:** PodifyAI's core mission is to enhance accessibility. I am committed to ensuring that the tool remains inclusive and effectively serves diverse user groups, including those with visual impairments and language barriers. This involves continuous evaluation of the UI/UX and the quality of multilingual outputs.

B. Future Mitigation Strategies

In future iterations, particularly with the integration of Gemini models, I will prioritize:

- **Transparency:** Clearly communicating the capabilities and limitations of the AI models used.
- **Fairness:** Actively testing for and mitigating biases in generated outputs.
- **Accountability:** Establishing clear processes for addressing potential harms or inaccuracies.
- **Security:** Implementing strong data protection measures for any user data handled.

C. Deliverable 3 Considerations

The integration of Gemini introduces new responsible AI dimensions:

1) **Transparency**: The UI clearly labels which model is being used (DistilBART vs Gemini), providing users transparency about processing methodology.
2) **Data Privacy**: While Gemini requires cloud processing, the dual-model architecture allows privacy-conscious users to select local DistilBART processing for sensitive documents.
3) **Model Reliability**: The automatic fallback mechanism ensures service continuity even when Gemini API is unavailable, preventing dependency on a single commercial provider.
4) **Performance Monitoring**: Real-time metrics empower users to make informed decisions about model selection based on their speed/quality requirements.
5) **Bias Mitigation**: User study included diverse participant backgrounds; results showed consistently high satisfaction across demographics.