# ‘MediaVerse’

AI-Powered Platform for Seamless Content Interaction

By
Aishwarya Bhat
Chaitali Shinde
Prajakta Badgujar

# Agenda

- The Challenge, The Pain, The Gap
- What is MediaVerse?
- Related Work
- How MediaVerse stands out?
- Our Goal
- Key Features
- Technology Stack
- Processing Workflow
- App Experience
- Challenges Faced
- Future Enhancements
- Demonstration

## The Challenge

In the digital age, users deal with a flood of content in various formats — documents, audio recordings, and video lectures.

## The Pain

Manually reading, transcribing, and extracting key points from these files is slow, error-prone, and inefficient.

## The Gap

There is no unified tool that can **process**, **understand**, and allow **natural interaction** with this information — across all formats.

# ‘MediaVerse’

A Unified AI-Powered Platform for Interacting with PPTs, PDFs, Text Documents, Audios & Videos





# Related Work

- Traditional tools (e.g., PDF search, Word search) **locate keywords** but don't **understand or explain** the content.
- Most systems **cannot process audio or video** files and **don’t provide summaries or direct answers**.
- Virtual assistants like Siri or Google Assistant **only answer general questions**, not based on **your own documents**.
- Speech-to-text tools (e.g., YouTube captions) generate transcripts, but **don’t interpret** them or answer questions.
- Users still need to **manually read or listen** through content to find specific information.
- Recent AI tools (e.g., ChatGPT, BERT) show language understanding but are usually **limited to text files**.
- These bots **lack media support** and don't return **timestamped answers** from audio/video files.

# How Mediaverse Stands Out?

- Supports **multiple file types**: PDF, DOC/DOCX, PPT/PPTX, TXT, audio, and video.
- Uses **OpenAI Whisper** for accurate speech-to-text conversion from media files.
- Applies a **Large Language Model (LLM)** to extract, understand, and answer questions from content.
- Provides **direct, contextual answers** with **timestamp references** for audio and video.
- Saves users from having to **read or listen** to entire files — offering **fast, intelligent access to information**.

# Our Goal

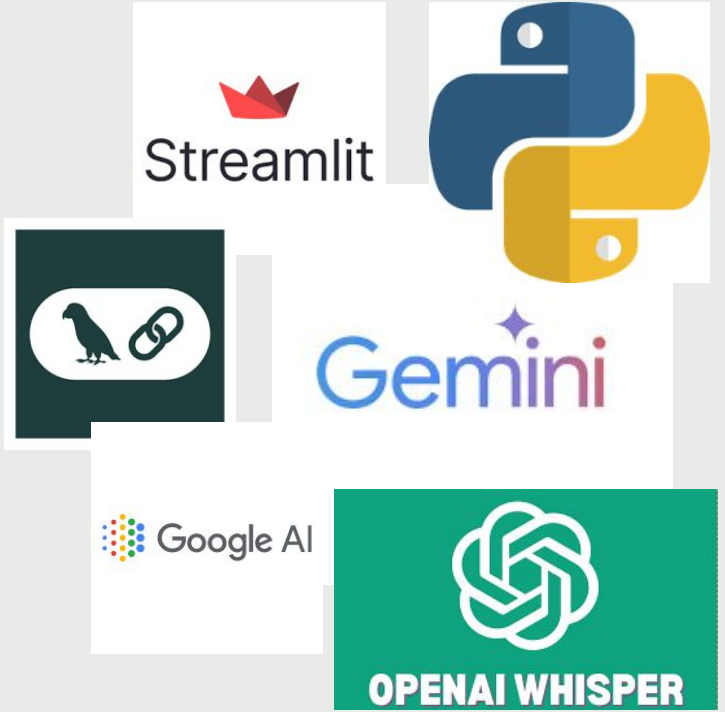
- Build a single intelligent interface to interact with diverse content formats seamlessly.
- Enable real-time interaction with text, audio and video content.
- Use AI to extract, summarize, and answer queries across formats.
- Improve workflow efficiency and comprehension.
- Ensure context-aware responses and smooth user experience.

# Key Features

- Conversational interface for user queries
- Unified multi-format interface
- No need to switch apps or tools.
- Accurate content extraction & summarization
- High relevance chatbot responses
- Audio and Video content Support
- Speech transcription with timestamps

# Technology Stack

1. **Streamlit** - UI framework for building interactive web apps
2. **OpenAI Whisper** - Transcription of audio and video files
3. **Google Gemini** - Conversational AI for intelligent responses
4. **FAISS** - Vector indexing for fast document similarity search
5. **LangChain** - Handling prompt templates and QA chain logic
6. **MoviePy** - Handling video processing
7. **PyPDF2** - Reading PDF files
8. **docx2txt/ python-docx** - Extracting content from Word documents
9. **Python-pptx** - Extracting content from PowerPoint presentations
10. **LibreOffice + subprocess** - Converting .doc files to .pdf or .txt



# Processing Workflow

## File Upload

Mediaverse starts with a seamless upload interface where users can drag and drop documents and media files. It supports multiple formats including PDFs, Word, PowerPoint, plain text, and audio/video files, making it highly versatile for different content types. Uploaded files are temporarily stored for secure and efficient processing.

## File Processing

Each file is processed according to its type using specialized tools. Text-based documents are parsed to extract content, while audio and video files are transcribed using OpenAI Whisper. This model provides accurate speech-to-text conversion along with timestamps, ensuring that all files are converted into clean, usable text.

## Text Chunking & Embedding

The extracted text is divided into overlapping chunks to maintain context across sections. These chunks are then converted into high-dimensional semantic vectors using Google Generative AI embeddings. This step ensures that the meaning and context of the content are captured for intelligent search and retrieval.

## FAISS Vector Indexing

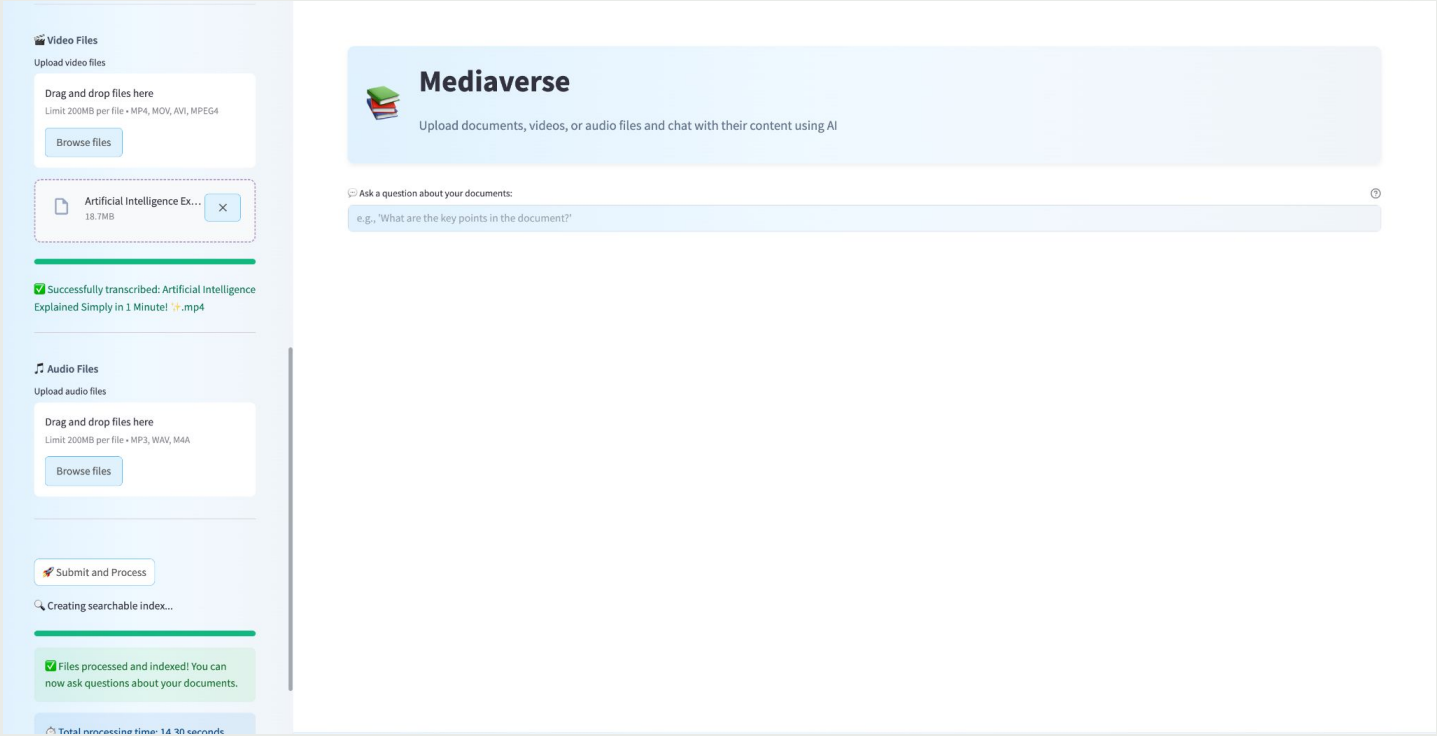
All embeddings are stored in a FAISS index, enabling fast and meaningful similarity searches. When a user asks a question, FAISS retrieves the most contextually relevant chunks of content, even if the question doesn't contain exact keywords. Timestamps are retained to provide source references for audio or video content.

## Gemini-powered Q&A

The retrieved text chunks are passed to Google Gemini, which uses its advanced language capabilities to generate natural language answers. The system delivers accurate, context-aware responses, and includes timestamp references when relevant. This turns uploaded content into an interactive, AI-driven conversation experience.

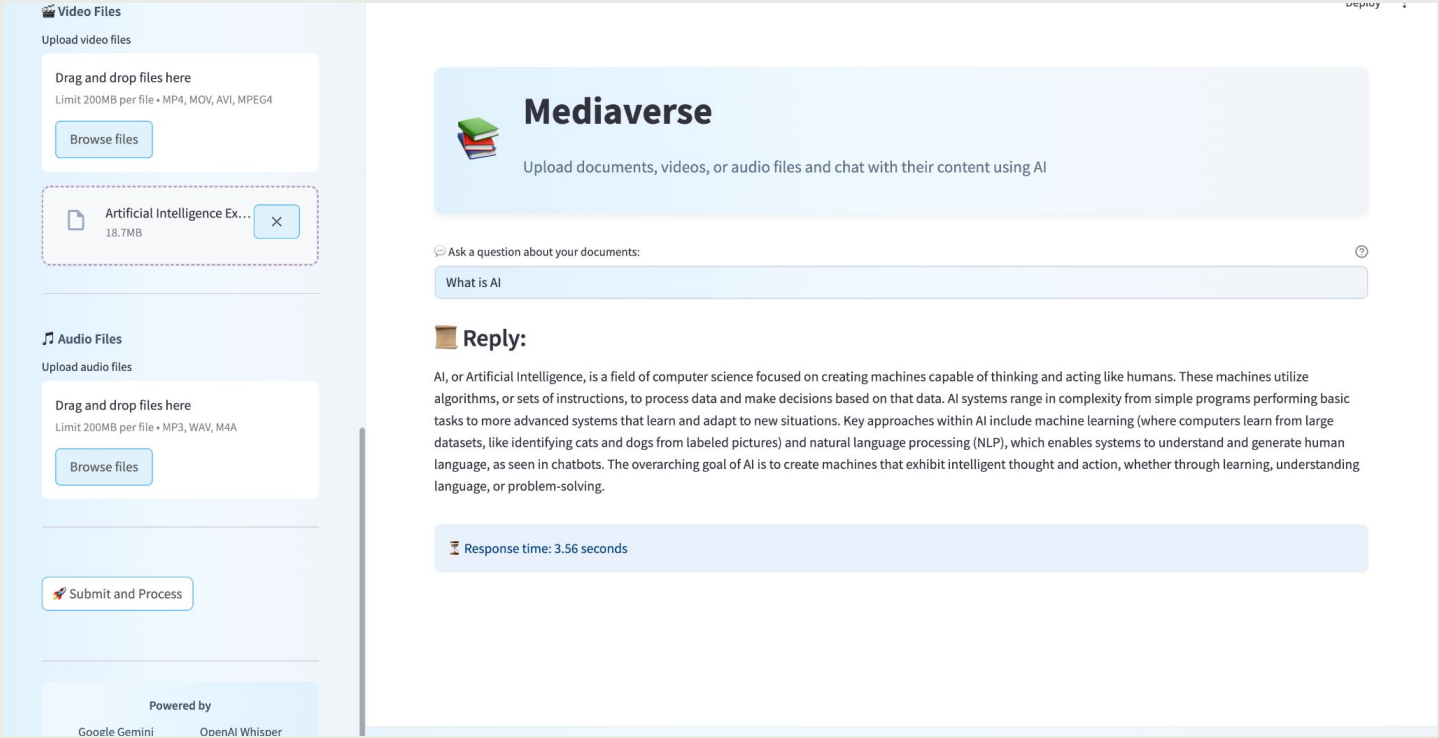
# App Experience

1. Uploads: Drag and drop files in sidebar and submit for processing.



# App Experience

2. AI Assistant: Ask questions in plain English to Get contextual, timestamped answers



# Challenges Faced

- **DOC parsing**: Solved via LibreOffice conversion to PDF/TXT
- **Large files**: Managed with smart chunking + batching
- **Accuracy in transcription**: Used Whisper base model (trade off between speed and precision)
- **Contextual answers**: Handled with prompt engineering using LangChain

# Future Enhancements

- Multilingual support
- Upload from cloud sources (Google Drive, Dropbox)
- Speaker recognition in media files
- Chat history & session saving
- OCR support for scanned PDFs
- Advanced summarization features
- Multi-user support with session-based storage.
- Integrate chunk relevance scoring before passing to the LLM to reduce token usage.
- Implement streaming responses for long answers.

# Demonstration

# Thank You