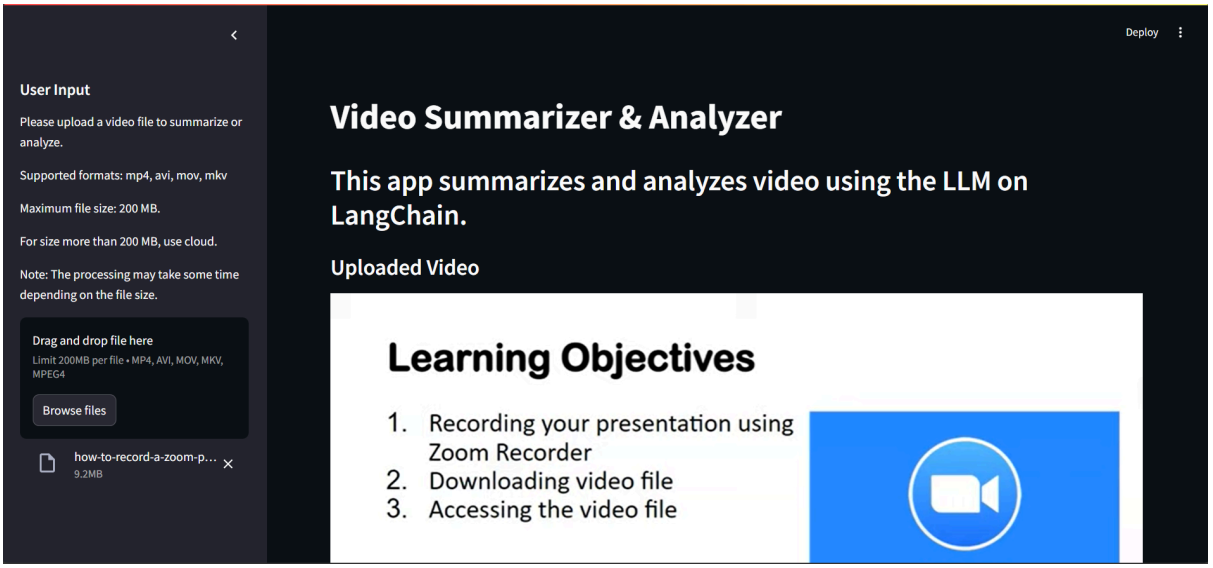
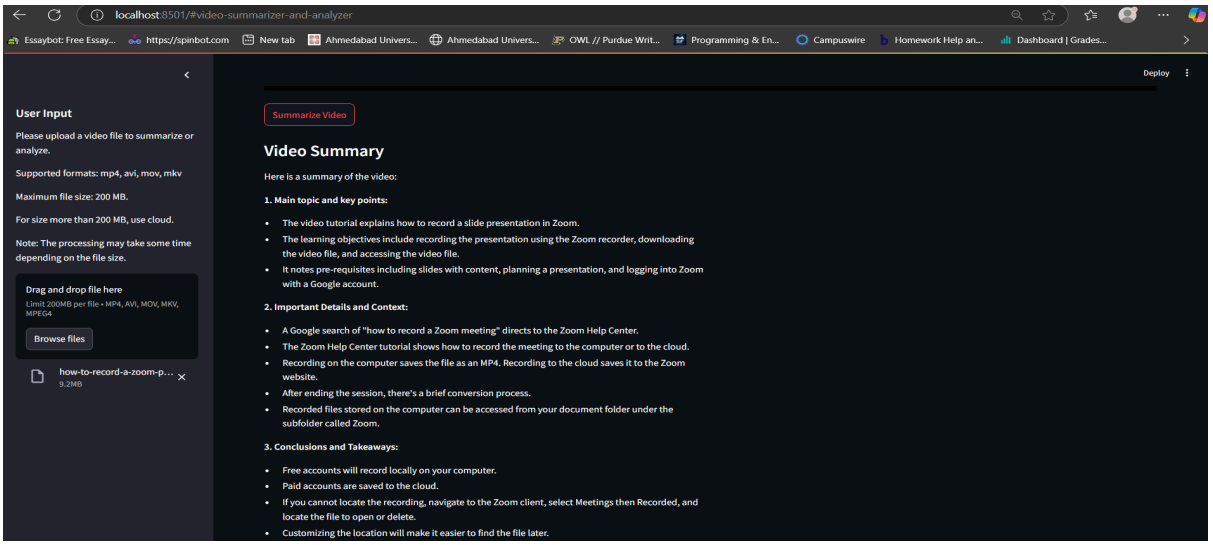
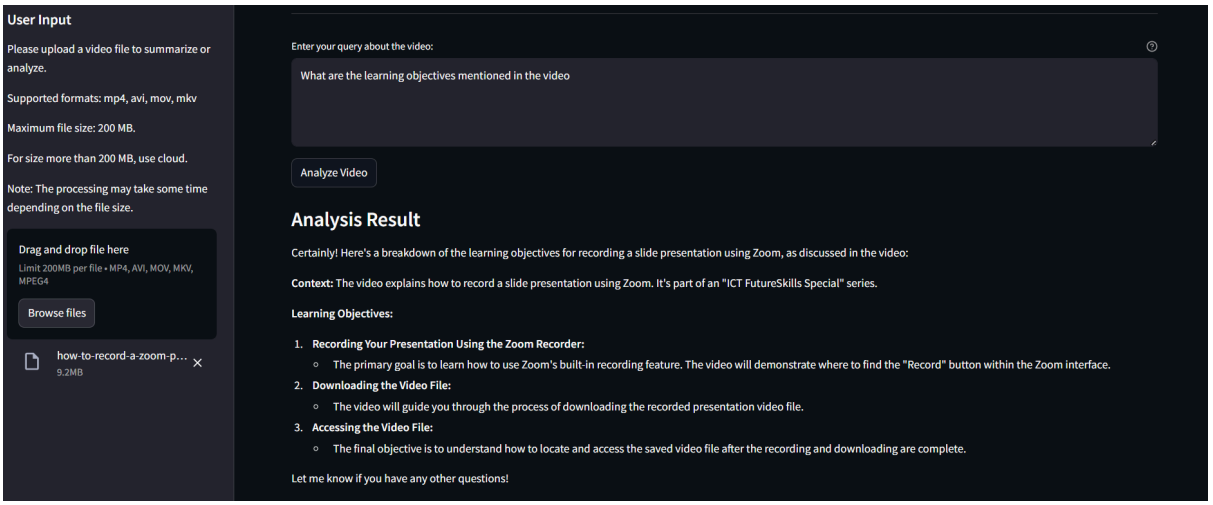
Streamlit - View Model for Input



Summarizer



Analyzer based on user's request



Project Overview
This application processes video files and provides comprehensive summaries and targeted analysis based on user queries. Based on the principles of langchain and RAG.

Technical Implementation
The application was implemented using:

Streamlit: For the web interface
Google Gemini API: For multimodal AI processing (video understanding)
Phidata: For agent creation and management
DuckDuckGo Search: For RAG (Retrieval-Augmented Generation) capabilities
Python with various supporting libraries

The core functionality involves:
Secure API integration with Google's Gemini model
Video file handling with temporary storage
Prompt engineering for effective summarization and analysis
Exception handling for robust operation

Challenges Faced:
Google Colab Implementation Failures

Initial attempts to implement this project in Google Colab were unsuccessful due to:

Ngrok Tunnel Errors: The ngrok tunneling service, commonly used to expose Colab-hosted applications to the public internet, encountered persistent connection issues.

TCP Errors with Local Tunnel: Alternative tunneling via Local Tunnel also failed with TCP errors, suggesting networking restrictions within the Colab environment.

File Size Limitations: Colab's free tier has memory constraints that affected video processing capabilities.

These issues necessitated moving development to a local environment where the application could be properly tested and deployed.

Model Selection Considerations:
While initially exploring Hugging Face models for video summarization:

HF models produced shorter, less informative summaries
The quality did not match requirements for comprehensive analysis
The summaries lacked sufficient context and detail

Google's Gemini model was ultimately selected because:
It has significantly more parameters and a more extensive training dataset
It demonstrated superior performance in understanding video content
It provided more detailed, contextually rich summaries

Future Improvements
Enhanced Interaction Features

Q&A from Summary: Add functionality to allow users to ask specific questions about the generated summary without reprocessing the video, improving efficiency and user experience.

Summary Quality Evaluation: Implement automatic evaluation metrics such as:
ROUGE scores to assess summary quality against benchmarks
Classification system to rate summaries as "worth/not worth" based on content relevance and comprehensiveness
Customizable Summary Length: Allow users to specify desired summary length (brief, standard, comprehensive)
Provide options for summary: summary 1, summary 2… and and option to select it.

Technical Enhancements:
Cloud Integration: Support for larger videos (>200MB) through cloud storage integration

Batch Processing: Enable processing of multiple videos in sequence

Persistent Storage: Option to save summaries and analyses for future reference

Export Functionality: Allow export of summaries in various formats (PDF, DOCX, TXT)

Model Improvements:
Model Switching: Option to choose between different AI models based on user needs
Implementing **openCV and ffmpeg** for uploading, processing, and analysing.

Fine-tuning: Create domain-specific fine-tuned models for specialized content types (educational, entertainment, technical). In the case of fine tuning for https://medflix.app/

1) Curated Medical dataset creation
2) Pre-train on medical terminology: enhancement
3) Structured Output Templates: Clinical case, procedural summaries
4) Test summaries based on summary materials available on web

Multi-language Support: Extend functionality to non-English videos

Conclusion:

The Video Summarizer & Analyzer represents a functional implementation of multimodal AI for video content understanding: uploading, processing, and analysing.

Also looking forward to finding a solution for a cloud based application.

Links:
Video: Video_Summarizer
Git: https://github.com/NeelB28/Video_Summarizer