Meeting Analysis App — Main Project Documentation

GITHUB REPOSITORY: https://github.com/Kgoliath/meeting_summarizer

1. Introduction

1.1 Problem Statement

Meetings often generate large amounts of audio data that are difficult to review and extract actionable insights from efficiently. Manual note-taking is time-consuming and error-prone.

1.2 Purpose of the System

The Meeting Analysis App automates transcription, summarization, and sentiment analysis of meeting audio files, providing users with concise, actionable reports to improve productivity and accessibility.

1.3 Objectives

- Automate transcription of meeting audio.
- Generate summaries and extract action items.
- Analyze overall sentiment of meetings.
- Provide downloadable reports.
- Store meeting data for retrieval and history tracking.

2. Key Features

- Audio upload support for common formats.
- Automated transcription using Assembly AI.
- Summarization and action item extraction via Cohere AI.
- Sentiment analysis using Hugging Face models.
- PDF export of transcription, summary, and sentiment.
- Persistent storage in SQLite database.

3. Technology Stack

- **Backend:** Python, Flask
- **Database:** SQLite (via SQLAlchemy ORM)

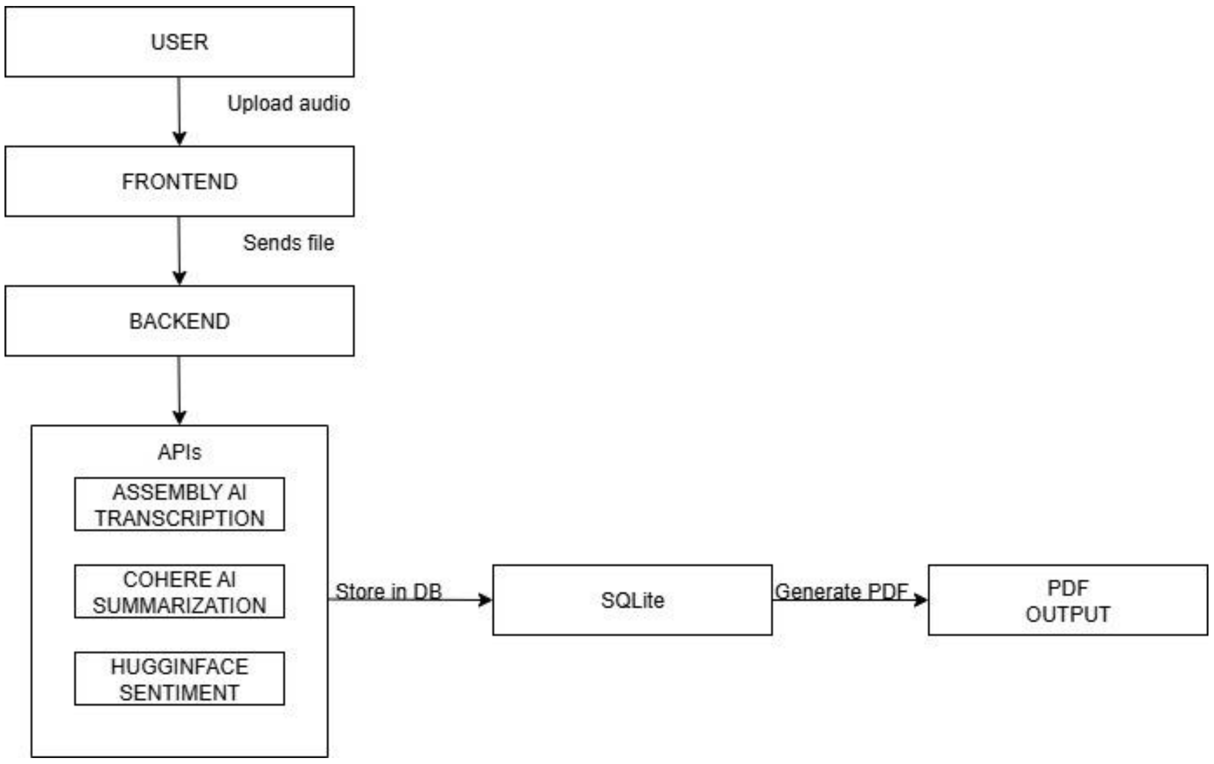
- **Frontend:** HTML templates (Jinja2)
- **APIs:**
    - Assembly AI (Audio transcription) -
    - Cohere (Text summarization) -
    - Hugging Face (Sentiment analysis) -
- **PDF Generation:** ReportLab
- **Environment Management:** .env for API keys

4. System Workflow

4.1 Step-by-Step Flow

1. User uploads an audio file via the web interface.
2. Audio file is temporarily stored in the uploads/ directory.
3. Audio is uploaded to Assembly AI for transcription.
4. Transcription status is polled until completion.
5. Transcription text is sent to Cohere Chat API for summarization and action item extraction.
6. Transcription text is sent to Hugging Face sentiment model for sentiment analysis.
7. Meeting data (filename, transcription, summary, sentiment, timestamp) is stored in SQLite.
8. A multi-page PDF report is generated with transcription, summary, and sentiment.
9. User can download the PDF report via a secure route.

4.2 System Workflow Diagram



5. Database Schema

5.1 Meeting Table

Column	**Type**	**Description**
id	Integer	Primary key
filename	String	Name of uploaded audio file
transcription	Text	Full transcribed text
summary	Text	Summarized text and action items
sentiment	Text	Overall sentiment (positive/neutral/negative)
timestamp	DateTime	Record creation timestamp

5.2 ERD Diagram

Meeting
id (PK)
filename
transcription
summary
sentiment
timestamp

6. Security & Error Handling

- API keys stored securely in .env file, never hard-coded.
- File uploads use secure_filename() to prevent path traversal attacks.
- Uploaded files are deleted after processing to prevent storage bloat.
- Errors such as missing files or API failures are logged and user-friendly error messages displayed.
- Local SQLite database used; for production, consider hardened security.

7. API Trade-off Analysis

7.1 APIs Considered

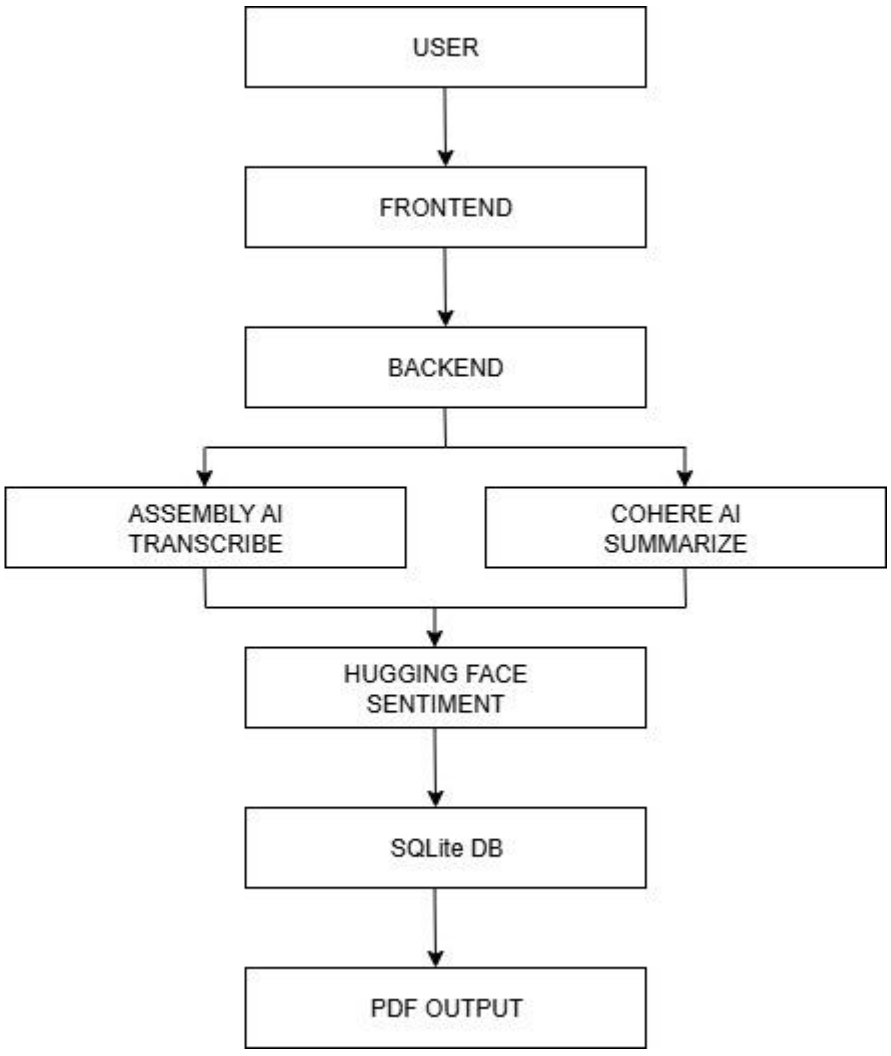
- Assembly AI (Audio transcription)
- Cohere AI (Summarization & action items)
- Hugging Face (Sentiment analysis)
- Optional alternatives: OpenAI GPT, Google Speech-to-Text

7.2 Comparison Table

API	**Accuracy**	**Latency**	**Cost**	**Security/Privacy**	**Scaling Challenges**	**Decision Notes**
Assembly AI	High	Low	Medium	HTTPS, secure	Can handle multiple requests	Selected for transcription accuracy

Cohere AI	Medium	Low	Low	HTTPS	Limited request size	Chosen for summarization efficiency
Hugging Face	Medium	Low	Free/Open	HTTPS		

8. System Architecture & Data Flow



10. Setup & Installation Guide

Prerequisites

- Python 3.10+
- Virtual environment recommended
- SQLite installed

Installation Steps

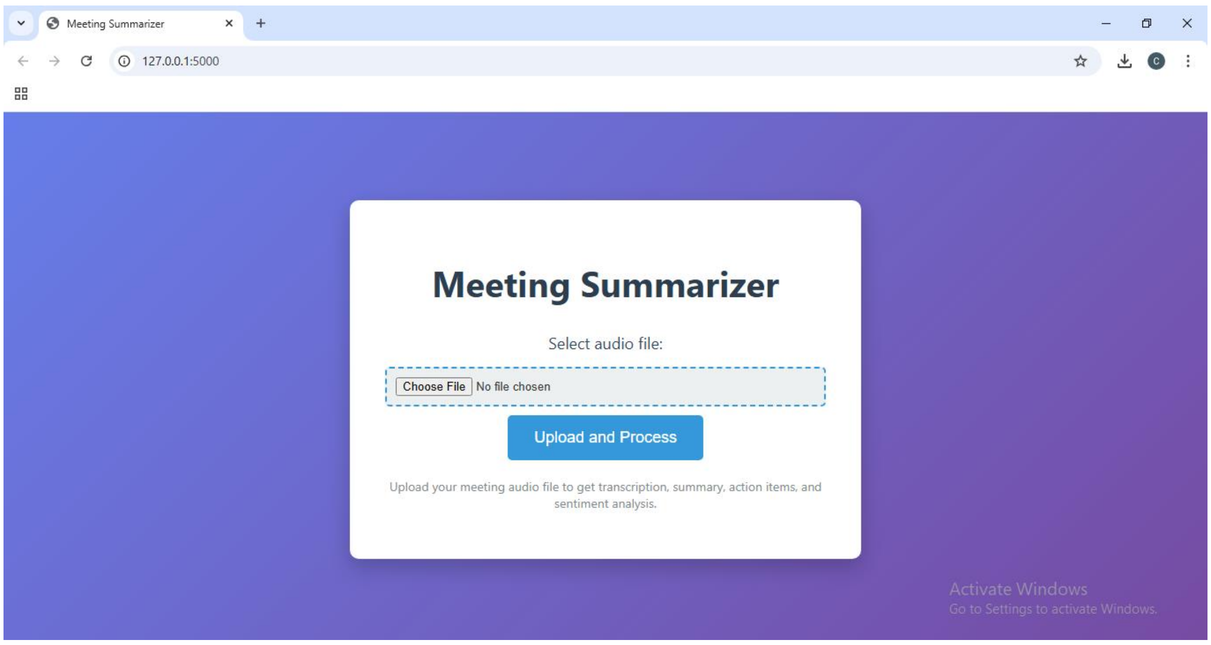
1. Clone repository: git clone <repository_url>
2. Navigate to folder: cd meeting-analysis-app
3. Install dependencies: pip install -r requirements.txt
4. Create a .env file with API keys:

```
ASSEMBLYAI_API_KEY="296211cf5ce942e0ab69db9f7ba95ef4"
COHERE_API_KEY="Ti6onZZQRJDPFXO2a8PUAdma56dUpvLzvEJn2Ki0"
HUGGINGFACE_API_KEY="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"
```

Do not commit .env to GitHub.

5. Run the app:

- python app.py (Flask)
- streamlit run app.py (Streamlit)



Upload your meeting audio file to get transcription, summary, action items, and sentiment analysis.

11. Future Enhancements

- Support more audio formats.
- Speaker identification for multi-participant meetings.
- User authentication and access control.
- Granular sentiment analysis per action item.
- Integration with productivity tools (Google Calendar, Notion).
- Real-time transcription and live meeting analysis.
- Dashboard for meeting history and analytics.

12. Conclusion

The Meeting Analysis App integrates multiple AI technologies to transform meeting audio into actionable insights. By automating transcription, summarization, and sentiment analysis, it enhances productivity, accessibility, and decision-making. The system is designed for scalability, security, and extensibility, making it suitable for professional and educational use.