Automated Meeting Minutes Generator for Insurance Claims Committee

1. Project Overview

Objective:

To develop a secure, web-based GenAI application that automatically generates structured and accurate Minutes of Meeting (MoM) from insurance claim committee meetings using audio, transcript, or image inputs.

Scope:

- Accepts audio, text, PDF, and image uploads.
- Converts audio and scanned documents to text (Whisper, OCR).
- Generates MoM using Gemini Pro API.
- Outputs downloadable and editable MoM files.
- Streamlit-based UI with tagging, logs, and secure handling.

2. High-Level Architecture

[Input Layer] --> [Preprocessing Layer] --> [GenAI Summarizer] --> [Postprocessing + Output Layer] --> [UI Layer]

3. Module Breakdown

🔷 Module 1: Input Layer

Purpose: Accept user inputs (Audio, Text, PDF, Image)

Features:

- Upload interface in Streamlit
- Accepts .mp3, .wav, .txt, .pdf, .jpg, .png
- Optional: Participant tagging dropdown or manual entry

Tools:

- Streamlit file_uploader
- Secure file storage (temp or encrypted directory)

🔷 Module 2: Preprocessing Layer

Purpose: Normalize all inputs into clean text

Submodules:

- Audio Transcription (Whisper API)
  - Input: .mp3, .wav → Output: Clean text
  - Tool: OpenAI Whisper
- OCR for PDFs and Images
  - Input: .pdf, .jpg, .png → Output: Extracted text
  - Tool: pytesseract, pdfplumber or PyMuPDF
- Plain Text Bypass
  - Input: .txt → Direct to processing
- Text Consolidation
- Store all extracted/transcribed text in /data/transcripts/meeting_<timestamp>.txt

🔷 Module 3: GenAI Summarization Layer

Purpose: Generate structured MoM using Gemini Pro

Steps:

1. Read consolidated transcript
2. Prompt Gemini Pro API with a summarization template:
   Summarize the following meeting transcript into sections:
   - Agenda
   - Discussions
   - Decisions Taken
   - Action Items

3. Store summary in /data/outputs/meeting_<timestamp>.md/.docx
   API Used:
   Gemini Pro (via REST or SDK)

🔷 Module 4: Postprocessing & Output Layer

Purpose: Format, display, and export MoM

Features:

- Render MoM in Streamlit preview window
- Download as .docx or .md or .txt
- Option to edit live inside Streamlit
- Maintain logs of edits with timestamps

Tools:

- python-docx for Word file creation
- Editable text widgets in Streamlit
- Local JSON or SQLite for change logs

🔷 Module 5: UI/UX Layer (Streamlit)

Features:

- Home screen: Project info + Upload form
- Participant tagging interface
- Preview and Download section
- Logs viewer
- Secure login (optional)

🔷 Module 6: Data Security & Compliance

Measures:

- Encrypted file storage or automatic cleanup after session
- HTTPS-enabled deployment
- No external transmission of files except API calls
- Role-based access control (Optional for Phase 2)
- Logs: Store participant tag, file uploaded, time, and changes

Tools:

- cryptography for file encryption (optional)
- sqlite3 or local logging
- hashlib for file integrity

4. Folder Structure

```
project-root/
│
├── app/
│   ├── main.py
│   ├── utils/
│   │   ├── ocr.py
│   │   ├── whisper_transcribe.py
│   │   ├── summarizer.py
│   │   ├── file_handler.py
│   │   └── logger.py
│   └── components/
│       ├── uploader.py
│       └── editor.py
│
├── data/
│   ├── transcripts/
│   ├── outputs/
│   └── logs/
│
├── models/
│   ├── whisper_model.py
│   └── gemini_prompt.py
│
├── requirements.txt
├── README.md
└── .env
```

5. Deliverables

Deliverable	Description
Streamlit Web App	Frontend to handle uploads, view summaries
GenAI Summary Engine	Structured MoM generator using Gemini API
Whisper Integration	Audio to text converter
OCR Integration	Image/PDF to text converter
Editable Output	Editable + downloadable summary
Compliance Module	Logs + secure storage
Documentation	README + API usage guide

6. Final Integration Flow

1. User uploads file → Audio/Text/PDF/Image
2. Preprocessor:
   - •Audio → Whisper
   - •PDF/Image → OCR
   - •Text → Bypass
3. Text Consolidation → Save to /data/transcripts/
4. GenAI Summarization → Gemini generates MoM
5. Streamlit UI:
   - •Displays structured MoM
   - •Allows editing
   - •Exports .docx
6. Logs → Stored per session with timestamps

7. Security Checklist

- HTTPS for deployment
- File cleanup after session ends
- No external file storage (except processing)
- Use API keys via .env
- Optional: Login for access control
- Edit logs with timestamp and IP/user info

8. Tech Stack

Layer	Tools/Tech
Frontend	Streamlit
Summarization	Gemini Pro API
Audio Transcribe	OpenAI Whisper
OCR	Tesseract + pdfplumber
Backend Logic	Python
Output Formats	python-docx, Markdown
Storage/Logs	Local storage, SQLite/JSON logs

9. Optional Enhancements (Phase 2) //Ignore for now

- Multilingual summarization (Hindi, Tamil)
- GPT-4/Gemini-1.5 for deeper summaries
- User authentication & roles
- Cloud deployment (Streamlit Sharing / GCP)