**Live Meeting Summarizer Application**

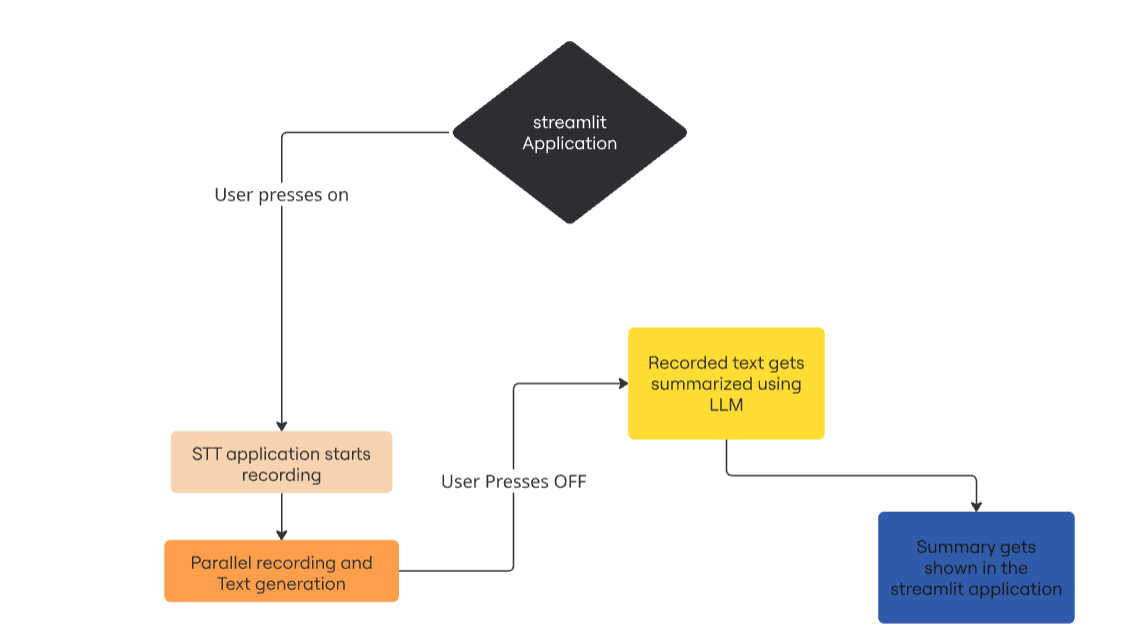
**Project Statement:**  
The objective is to develop a real-time Meeting Summarizer that leverages speech-to-text (STT) models and open source LLMs or hugging face transformers to generate structured summaries from live meeting recordings. The solution will be delivered as a fully integrated Streamlit application with backend support for transcription, speech diarization, summarization, user interaction and endpoint services like sending email or downloading the summary by 1 click.

**Outcomes:**

* Capture live audio and convert it to text using offline STT models.
* Identify and segment the audio by speaker using diarization models like pyannote
* Integrate transformer-based summarization using LLMs and transformers from Groq API or Hugging Face.
* Deliver structured summaries only after the Stop button is pressed and the complete transcript is available.
* Build an interactive UI using Streamlit with real-time control and result rendering.
* One-click export as Markdown/PDF and send via email.

**Modules to be implemented:**

* Audio Recording and STT using Vosk/Whisper with PyAudio or SoundDevice
* Integrates diarization models like pyannote-audio for speaker separation.
* Prompt updates for diarized transcript to perfom text wummarization via LLM or transformer models (Groq or Hugging Face etc)
* Streamlit Frontend with recording controls
* Backend Pipeline (multi-threaded capture, queuing, async summarization)
* Testing, Evaluation & Optimization
* Final Presentation & Documentation



**Milestone 1: Speech-to-Text System (Weeks 1–2)**

**Module 1: Project Kickoff and Audio Dataset Setup**  
Tasks:

* Define technical architecture (multi-threaded STT → diarization → summarization).
* Explore and compare STT models: Vosk and Whisper.
* Set up Python environment with pyaudio, sounddevice, vosk, or whisper.
* Download evaluation datasets like AMI Meeting Corpus or create synthetic datasets for testing and benchmarking.
* Use the dataset to evaluate baseline WER across models and to assist in model selection.

Deliverables:

* Architecture diagram and documentation
* Audio data samples for testing.
* WER benchmark report using AMI dataset.

Evaluation:

* Completion of architecture and setup
* Data readiness for transcription experiments

**Module 2: Real-Time Speech-to-Text Engine Implementation**  
Tasks:

* Implement threaded audio capture from the microphone.
* Connect STT engine (Vosk/Whisper) to transcribe in real time.
* Display transcription updates in a terminal/console.
* Evaluate performance using jiwer to calculate WER.

Deliverables:

* Python module for real-time STT
* Sample transcription logs
* Evaluation report using WER

**Evaluation:**

* WER below 15%
* Real-time transcription accuracy and stability

**Milestone 2: Speech diarization and summarization Engine (Weeks 3–4)**

**Module 3: Speaker Diarization Engine**  
Tasks:

* Integrate pyannote.audio to tag different speakers in audio.
* Process the .wav file after recording ends.
* Convert speaker segments into readable transcript like:

[Speaker 1]: Let's discuss next quarter goals.

[Speaker 2]: We should increase sales by 20%.

* Sync segments with STT output.

Deliverables:

* Diarized transcript
* Python function for diarization
* Sample AMI corpus results

Evaluation:

* Diarization Error Rate (DER) < 20%
* Accurate segmentation of turns

**Module 4: LLM-Based Summarization Logic**  
Tasks:

* Choose summarizer model (Groq LLaMA 3.1, T5, BART via HuggingFace).
* Design prompts for diarized transcripts
* Write API wrapper for summarization
* Evaluate summary quality with ROUGE, BLEU, human review.

Deliverables:

* Prompt templates for various meeting types .
* Summarizer module
* Sample summary outputs

Evaluation:

* ROUGE > 0.4.
* Speaker-based structure preserved in summary

**Milestone 3: Frontend and Backend Fusion (Weeks 5–6)**

**Module 5: STT + Diarization + Summarization Integration**

Tasks:

* Ensure summarization only triggers after Stop button.
* Build backend queue for transcript → diarization → summary.
* Merge diarized text with transcription output before summarization.

Deliverables:

* Fully integrated processing pipeline
* End-to-end transcript → summary demo.

**Evaluation:**

* Consistent structure & no race conditions

**Module 6: Streamlit UI Development**  
Tasks:

* Design real-time frontend with Start/Stop buttons
* Add transcription log and summary viewer
* Status bar: “Recording”, “Transcribing”, “Summarizing”
* There need to be real time stt on UI and after end of recording there need to be summary along with diarized transcript

Deliverables:

* Fully functional Streamlit application
* Robust integration between all backend components.
* Live feedback components.

**Evaluation:**

* Smooth end-to-end functionality from Streamlit UI
* No backend/API delays or blocking

**Milestone 4: Finalization and Delivery (Weeks 7–8)**

**Module 7: Testing, Evaluation, and Optimization**  
Tasks:

* Add download options: Export summaries as .md and .pdf
* Integrate email system using smtplib to send: Subject: “Meeting Summary – [Date/Title]”
* Implement structured logging:Save each meeting session with timestamp, transcript, summary, speaker info in .json/.parquet

Deliverables:

* Working export and email modules
* Updated UI with export/email option
* Final bugfixes

Evaluation:

* Fully functional Streamlit application.

**Module 8: Documentation and Demo Prep**  
Tasks:

* Finalize documentation covering architecture, APIs, models, and UI.
* Prepare a demo script with sample recordings.
* Package entire solution for GitHub release.
* Create a presentation deck and rehearse walkthrough.

Deliverables:

* Documented codebase with README
* Slide deck and/or recorded walkthrough
* GitHub repo with tagged release

Evaluation:

* Comprehensive documentation
* Demo-ready solution
* Presentation delivery quality

**Evaluation Criteria:**

|  |  |  |  |
| --- | --- | --- | --- |
| Milestone | Focus Area | Metric / Evaluation Method | Target/Goal |
| Milestone 1 | STT System | WER (Word Error Rate) via jiwer | < 15% |
| Milestone 2 | Diarization & Summarization | ROUGE Score, DER | ROUGE > 0.4, DER < 20% |
| Milestone 3 | UI Integration | Control Responsiveness | No lag/errors |
| Milestone 4 | Full System + Output Features | Full Pipeline Flow | Fully Working with Export/Email |
| Overall | Delivery & Presentation | Docs + GitHub + Demo | Clear, complete, and professional |

**Tech Stack:**

|  |  |
| --- | --- |
| Area | Tools / Libraries |
| STT | Vosk, Whisper, pyaudio, sounddevice |
| Diarization | pyannote.audio, torchaudio |
| Summarizer | LLaMA 3.1 (Groq API), T5, BART (Hugging Face) |
| Frontend | Streamlit |
| Backend | Python threading, asyncio, queue |
| Evaluation | jiwer (WER), rouge\_score, BLEU |
| Logging/Export | JSON, Markdown, Pandas, Parquet, smtplib |