A Project Report on

SmartMeet: AI-Powered Meeting Summarizer

Submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF ENGINEERING

by

Sanket Kudale : 22106059

Ranjit Kadam : 22106034

Tejas Kalokhe : 22106049

Sahil Govardhane : 22106097

Guide:

Prof. Sayali P. Badhan

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2025-2026)

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

CERTIFICATE

This is to certify that the project entitled **"SmartMeet: AI-Powered Meeting Summarizer"** is a bonafide work of **Sanket Kudale (22106059), Ranjit Kadam (22106034), Tejas Kalokhe (22106049), Sahil Govardhane (22106097)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering (Artificial Intelligence & Machine Learning).**

_____

Prof. Sayali P. Badhan
Guide

_____

Prof. Sayali P. Badhan
Project Coordinator

_____

Prof. Dr. Jaya Gupta
Head of Department

_____

Dr. Uttam D. Kolekar
Principal

A.P. SHAH INSTITUTE OF TECHNOLOGY, THANE

Project Report Approval for B.E.

This project report entitled ***"SmartMeet: AI-Powered Meeting Summarizer"*** by ***"Sanket Kudale, Ranjit Kadam, Tejas Kalokhe and Sahil Govardhane"*** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Science & Engineering (Artificial Intelligence & Machine Learning), 2025-26***.

Examiner Name	Signature
1._____	_____
2._____	_____

Date:

Place:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Sanket Kudale 22106059)

(Ranjit Kadam 22106034)

(Tejas Kalokhe 22106049)

(Sahil Govardhane 22106097)

Date:

Abstract

Meet-Summarizer is a web-based application designed to automate the summarization of meeting transcripts by extracting key points, decisions and action items using state-of-the-art Natural Language Processing (NLP) techniques. Traditional meeting documentation processes are time-consuming and prone to human error, leading to reduced productivity and inefficient knowledge retention. To address these limitations, the system leverages AI-driven models such as Whisper for transcription and transformer-based summarizers for generating concise, structured and context-aware summaries. The platform integrates speaker separation, topic detection and summary generation features, ensuring clarity and accountability in collaborative environments. The system aligns with multiple United Nations Sustainable Development Goals (SDGs): under **SDG 8** – Decent Work and Economic Growth, it enhances productivity and reduces manual workload through automation; in accordance with **SDG 9** – Industry, Innovation and Infrastructure, it promotes the use of AI/ML to strengthen technological innovation in communication systems; supporting **SDG 12** – Responsible Consumption and Production, it minimizes resource waste by improving meeting efficiency and reducing redundant discussions; when deployed in educational contexts, it advances **SDG 4** – Quality Education by helping students and educators capture essential information efficiently; and by fostering transparency and accountability in organizational communication, it contributes to **SDG 16** – Peace, Justice and Strong Institutions. Overall, this study explores the design, implementation and evaluation of Meet-Summarizer as a scalable AI-powered communication assistant that streamlines information processing and decision tracking, thus promoting innovation, sustainability and institutional efficiency.

Keywords: Meeting Summarization, Abstractive Summarization, Natural Language Processing (NLP), Web-based Application, AI Summarization System.

Contents

LIST OF FIGURES

ABBREVATIONS

NLP	*Natural Language Processing*
ML	*Machine Learning*
AI	*Artificial Intelligence*
LLM	*Large Language Model*
ASR	*Automatic Speech Recognition*
AMI	*Augmented Multiparty Interaction (meeting dataset)*
ICSI	*International Computer Science Institute (meeting corpus*
QMSUM	*Query-based Meeting Summarization dataset*
ROUGE	*Recall-Oriented Understudy for Gisting Evaluation*
BLEU	*Bilingual Evaluation Understudy*
BERT	*Bidirectional Encoder Representations from Transformers*
GPT	*Generative Pre-trained Transformer*
T5	*Text-to-Text Transfer Transformer*
HMNet	*Hierarchical Meeting Summarization Network*
Whisper AI	*OpenAI's Multilingual Speech Recognition Model*
GUI	*Graphical User Interface*
API	*Application Programming Interface*

Chapter 1
Introduction

In the contemporary digital age, meetings have become an integral part of both professional and academic environments. With the global adoption of remote and hybrid work models, online meetings through platforms such as Zoom, Google Meet and Microsoft Teams are now commonplace. These meetings often generate large volumes of discussions and transcripts that contain valuable information such as decisions, tasks, brainstorming ideas and follow-ups. However, navigating through these lengthy transcripts is often time-consuming, redundant and inefficient, especially when participants simply require the highlights or action points.

The need for efficient summarization arises from the fact that meeting participants, stakeholders, or absentees cannot afford to spend hours reading through transcripts. Moreover, crucial information can easily get buried under less important conversational details, leading to poor decision-making or forgotten tasks. Traditional note-taking methods, whether manual or digital, rely heavily on the attentiveness and interpretation of an individual. This often results in incomplete or biased summaries, which do not fully capture the essence of the meeting. Consequently, there is a demand for an automated, intelligent system that can process meeting transcripts and generate accurate, unbiased and concise summaries.

Meet-Summarizer is designed to address this challenge. It is an AI-powered summarization system that processes meeting transcripts and extracts the most relevant content. Using techniques from Natural Language Processing (NLP), machine learning and transformer-based deep learning models, the system identifies key points, decisions and action items, while also maintaining the context of the conversation. Additionally, it can detect speaker turns to ensure that the summary provides an organized overview of who contributed what, which is crucial in collaborative settings.

The importance of such a system is multifaceted. In corporate organizations, where decision-making is central to productivity, a summarizer ensures that no critical point is overlooked. It empowers managers and employees by providing them with clear action lists and concise recaps without wasting valuable time. In education, students can benefit by quickly revisiting the key

takeaways from lectures or group discussions, enabling better revision and knowledge retention. Similarly, in customer support or research domains, the ability to convert long dialogues into digestible summaries can lead to enhanced efficiency and better resource utilization.

Another significant aspect is the scalability of such a system. With advancements in cloud computing and AI models, summarization systems can now handle vast amounts of data in real time. This opens opportunities for integration with existing platforms, enabling live summaries during ongoing meetings, thereby assisting participants instantly. Moreover, with the inclusion of multi-language support, such tools can bridge linguistic barriers, making meetings more inclusive and globally accessible.

However, the task of automatic summarization is not without challenges. Meetings are often informal, unstructured and filled with interruptions, repetitions and incomplete sentences. Unlike well-structured documents, transcripts contain non-linear conversations, overlapping speech and contextual dependencies that make summarization a complex NLP task. Therefore, designing an efficient system requires a deep understanding of semantic meaning, context preservation and intent recognition.

Meet-Summarizer is envisioned as a solution that not only provides accurate summaries but also augments human productivity by ensuring that the essence of meetings is preserved without overwhelming users with unnecessary details. By leveraging cutting-edge AI models such as BERT, GPT and T5-based transformers, combined with pre-processing techniques like stopword removal, tokenization and semantic clustering, the system ensures that users receive meaningful, structured summaries.

Chapter 2

Literature Survey

Year	Title	Domain	Algorithm	Dataset	Gap
2025	Summaries, Highlights and Action Items	Meeting Summarization	Extractive + Abstractive Hybrid NLP	QMSUM, AMI, ICSI datasets	Struggles with capturing context-specific action items; abstractive summaries still lack precision.
2024	Minutes of Meeting Generation for Online Meetings Using NLP & ML Techniques	NLP, ML	Speech-to-Text (ASR), Extractive + Abstractive Summarization	Custom meeting datasets	Limited scalability for multilingual meetings; accuracy reduces in noisy environments.
2024	Exploring Advances in Meeting Minutes Generation and Face Attendance Systems (ConvoLogix Model)	Meeting Minutes + Attendance Tracking	Audio Extraction, MFCC + SVM, NLP Summarization	Custom (Marathi-English speech DB), MoviePy, Tkinter	Lack of robust abstractive summarization; limited generalization across domains.
2024	Online Meeting Summarizer and Report Generation using GenAI	Generative AI, LLM	LLMs (GPT, HANSN, SummPip, Smart-Meeting, Graph Fusion)	ACL TA-DA, AMI, Smart-Meeting DB	Privacy issues with LLMs; abstractive methods suffer from coherence issues in long dialogues

2024	AI Powered Multilingual Meeting Summarization	Natural Language Processing (NLP),Audio Processing, Multilingual Text Summarization,	Latent Semantic Analysis (LSA), NLP techniques for text extraction, Audio processing methods	Not explicitly mentioned; processes textual transcripts and audio recordings in multiple languages	Manual note-taking is inefficient; the system automates multilingual summarization with NLP and audio processing.
2024	NLP-Driven Video Transcription and Office Meeting Automation	Natural Language Processing (NLP), Speech Recognition / Video Transcription, Office Meeting Automation	OpenAI Whisper ASR, ChatGPT, Web Application	Meeting audio/video recordings uploaded by users; real-time processing	Manual summarization is time-consuming; the paper integrates ASR and NLP for real-time automated summaries.
2024	Abstractive Summarization of Long Meetings Using Whisper and BART Models	Natural Language Processing (NLP), Abstractive Text Summarization, Automatic Speech Recognition (ASR)	OpenAI Whisper,Meta's BART,Enhanced BART for long multi-speaker meetings	Meetings with more than two speakers (audio/transcripts)	Abstractive summarization of long, multi-speaker meetings is challenging; the system improves summarization accuracy and accessibility.
2023	Automated Minutes of Meeting Using a Multimodal Approach	Multimodal Summarization	Whisper AI (ASR), Transformers, BART, HMNet	Whisper DB (670k hrs), Custom annotated datasets	Fails to generalize across languages; struggles with coherence in very long meetings.

2023	Building Real-World Meeting Summarization Systems using LLMs	Industrial Meeting Summarization	GPT-3.5, GPT-4, PaLM-2, LLaMA-2	AMI, ICSI, QMSUM, MeetingBank	Trade-off between cost, performance and privacy; handling very long transcripts still an issue.
2023	Automatic Generation of Meeting Minutes Using NLP	Natural Language Processing (NLP), Artificial Intelligence (AI), Business Process Automation	The paper does not specify the exact algorithm used.	The paper does not mention the specific dataset used for training or evaluation.	The paper proposes an automated system to generate concise and accurate meeting minutes from transcripts, reducing the time and effort required for manual documentation.

Chapter 3

Limitations of Existing Systems

1. Engine and Version Constraints

Several existing tools rely on specific frameworks and libraries that require newer versions of programming environments such as Node.js, Python, or other backend dependencies. This often creates compatibility issues during deployment across different platforms. Users with outdated system environments face frequent installation errors, version mismatches and failed executions, making such systems less reliable and difficult to adopt in enterprise settings.

2. Scalability with Long Transcripts

Meeting transcripts are often very lengthy and unstructured, sometimes spanning several thousand words. Current summarization models, especially transformer-based architectures like BERT or GPT, are constrained by their maximum context window size. As a result, transcripts need to be truncated or split, which leads to loss of coherence and missing critical context. This affects the overall accuracy and completeness of generated summaries.

3. Lack of Real-time (Online) Summarization

Most existing systems are designed for post-meeting summarization, where the entire transcript is processed after the meeting concludes. However, in many cases, there is a demand for real-time or near real-time summaries that can help participants follow discussions and track decisions during the meeting itself. Achieving this requires addressing challenges such as latency, memory efficiency and accuracy trade-offs, which current systems have not yet fully resolved.

4. Evaluation Gaps

Automatic evaluation metrics such as ROUGE, BLEU and METEOR are commonly used in summarization tasks. However, these metrics do not always align with human judgment of summary quality. For example, a summary may achieve high ROUGE scores by overlapping words with a reference summary, yet still fail to capture key action items, decisions.

5. Lack of Personalization and Context Awareness

Many meeting summarizers generate generic summaries without considering the specific roles or needs of participants. For example, a project manager may be more interested in deadlines and action items, while a technical team member may want to focus on design discussions or technical issues. Current systems often fail to personalize summaries based on user profiles, context, or importance, which reduces their practical utility.

6. Limited Multi-Modal Inputs

Meetings today are not restricted to spoken dialogue. They often include presentation slides, shared documents, chat messages and non-verbal cues such as gestures or emphasis in tone. Most summarization tools, however, are built for single-modal inputs (mainly text) and ignore these supplementary materials. This limits the richness and completeness of generated summaries, especially in collaborative environments where context goes beyond speech.

7. Data Privacy and Annotation Challenges

A significant challenge in this domain is the scarcity of annotated meeting datasets. Real corporate meeting data is sensitive and often cannot be shared publicly due to confidentiality and privacy concerns. This limits the availability of large, high-quality datasets for model training and evaluation. As a result, many systems are trained on small or simulated datasets, which do not generalize well to real-world conditions.

Chapter 4

Problem Statement and Objectives

Problem Statement

Meetings are a fundamental aspect of professional, academic and organizational settings as they enable collaboration, brainstorming and decision-making. With the widespread adoption of online platforms, meetings now generate lengthy transcripts that can be difficult to process. Participants are generally more interested in highlights such as key decisions and action items rather than the full dialogue, which creates the need for automated summarization tools that can deliver concise and accurate recaps.

Although significant progress has been made in meeting summarization research, existing systems still face serious limitations. Most are designed for post-meeting use and cannot generate real-time or near-real-time summaries. They also struggle to handle long, multi-speaker transcripts where informal conversations, overlapping speech and frequent topic shifts reduce the coherence of summaries. As a result, the outputs are often incomplete, inaccurate, or lack contextual depth.

Another challenge is the lack of personalization in current systems. The summaries generated are usually generic and do not account for the diverse needs of different users. For instance, managers may want to focus on decisions and deadlines, while technical staff may prefer details of problem-solving discussions. Without context awareness and adaptability, these tools fail to provide meaningful value across varied user groups.

Evaluation methods further highlight a gap in existing systems. Metrics such as ROUGE and BLEU offer numerical scores but often fail to capture true quality from a human perspective. Summaries may appear correct by overlapping with reference texts yet omit crucial information such as action items or misrepresent speaker contributions. This mismatch between automatic evaluation and real-world utility creates a barrier for practical deployment in professional environments.

Finally, scalability and technical constraints remain problematic. Transformer-based models have limited context windows, forcing truncation or segmentation of transcripts, which disrupts coherence. Many tools are also dependent on specific frameworks or software versions, leading to compatibility and deployment issues. These gaps underline the need for a system like Meet-Summarizer, which integrates a user-friendly frontend with an AI-powered backend to generate accurate, structured and potentially real-time summaries. By addressing current shortcomings, it aims to enhance collaboration, improve decision-making and save valuable time in modern digital workplaces.

Objectives

1. Improve Meeting Transcript Processing

Goal: Develop a system that can handle long, multi-speaker transcripts with interruptions and topic shifts while maintaining context.

Outcome: The generated summaries remain coherent and accurate, even for lengthy and complex meetings.

2. Generate Structured and Actionable Summaries

Goal: Use advanced NLP models to produce summaries that emphasize key decisions, action items and consensus points.

Outcome: Users gain concise, actionable summaries that directly support follow-ups and informed decision-making.

3. Enable Personalization of Summaries

Goal: Adapt summary content to user roles and preferences, ensuring relevance for managers, technical staff and other stakeholders.

Outcome: Each participant receives tailored insights that enhance their efficiency and focus.

4. Enhance Real-Time Summarization

Goal: Introduce methods for generating partial summaries during ongoing meetings, balancing speed and accuracy.

Outcome: Participants benefit from timely updates that improve engagement and faster decision-making.

5. Improve Evaluation Methods

Goal: Employ evaluation metrics that align with human judgment, going beyond lexical overlap to measure semantic accuracy and usefulness.

Outcome: Summaries achieve higher trust and practical reliability for real-world applications.

6. Ensure Scalability and Deployment Efficiency

Goal: Build a frontend-backend system with manageable dependencies and compatibility across different environments.

Outcome: The system can be easily deployed in both development and production settings without technical barriers.

Chapter 5

Proposed System

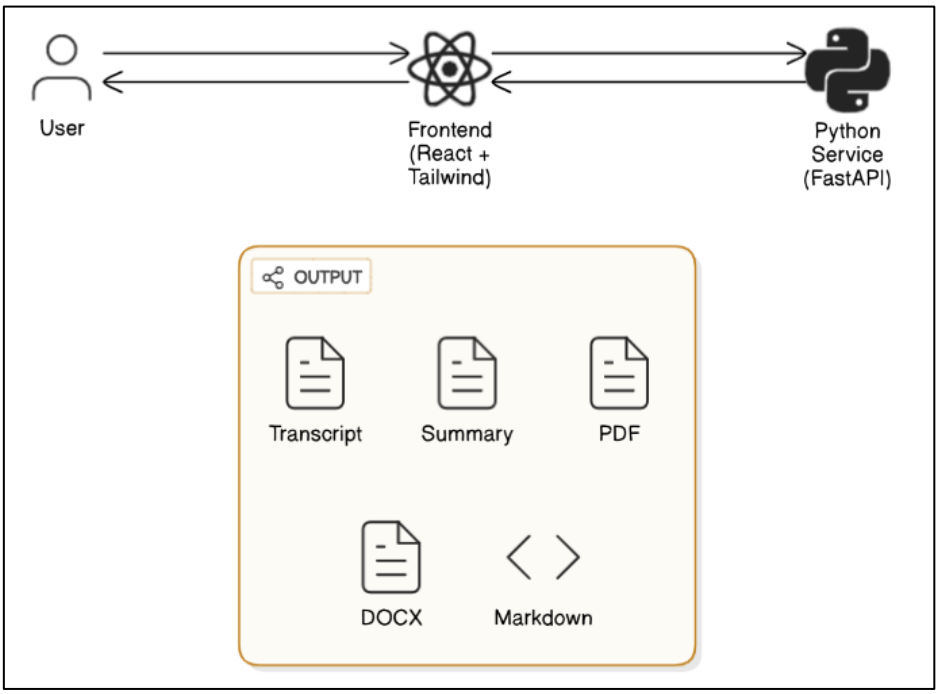

Figure 5.1 Architecture Diagram for SmartMeet

The architecture diagram illustrates how the Meet Summarizer system enables seamless interaction between the user, frontend interface and backend AI services. It demonstrates the complete flow — from uploading an audio file to generating structured summaries and downloadable reports in multiple formats.

Components:

User:

- The end-user interacts with the system through the web application.
- Users can upload meeting audio files, view transcripts and download summaries in preferred formats such as PDF or DOCX.

Frontend (React + Tailwind):

- Built using React and Tailwind CSS for a responsive, modern and user-friendly interface.
- Handles audio upload, displays generated summaries and manages user interactions in real-time.
- Sends requests to the backend through API calls and displays results after processing.
- Implements dynamic rendering for fast updates without full page reloads.

Python Service (FastAPI):

- The AI engine of the system, developed using FastAPI for high-performance backend processing.
- Integrates OpenAI Whisper for speech-to-text conversion and pyannote.audio for speaker diarization.
- Utilizes Transformer-based NLP models (BERT, Hugging Face) for summarization and content structuring.
- Processes audio and text data, generating outputs such as transcripts and concise summaries.
- Returns processed data to the frontend via secure API responses.

Output Layer:

Provides processed content in different formats for the user:

- **Transcript (Raw Text)** – full meeting transcription.
- **Summary** – concise version of the transcript.
- **PDF** – downloadable professional meeting report.
- **DOCX** – editable Word document.
- **Markdown** – developer-friendly structured format.

Connections:

User → Frontend: The user uploads audio or initiates summarization.

Frontend → Python Service: The frontend sends the input to the FastAPI backend for processing.

Python Service → Frontend: The processed results (transcript and summary) are returned.

Frontend → User: The frontend displays the summarized content and enables downloads in chosen formats.

Data Flow Diagrams:

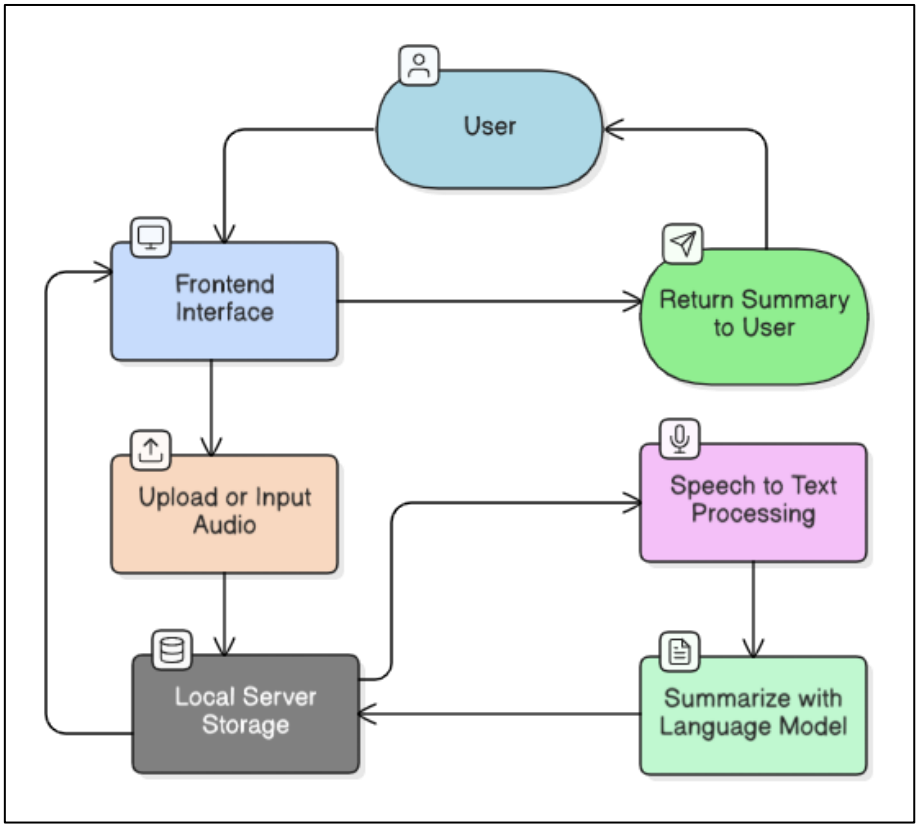

Figure 5.2 Audio Upload and Summarization Flow

The diagram depicts the workflow of a system that allows a user to upload or record audio, convert it into text, summarize it using a language model and return the summary to the user. The flow highlights the interaction between the user, the frontend interface, server storage and backend processing modules.

Components:

User:

- Represents the end-user interacting with the system.
- Initiates the process by uploading or recording audio.
- Receives the summarized output.

Frontend Interface:

- The user-facing component, such as a web or mobile app.
- Provides options to upload audio and view results.
- Sends audio to server storage and receives the processed summary.

Upload or Input Audio:

- Module where audio is captured or uploaded.
- Acts as the entry point for audio data into the system.

Local Server Storage:

- Stores uploaded audio files temporarily.
- Provides data for further processing by the speech-to-text module.
- Holds processed transcripts and summaries for retrieval.

Speech to Text Processing:

- Converts the uploaded audio into textual form.
- Uses a speech recognition engine or service.

Summarize with Language Model:

- Processes the transcript using a natural language model (like GPT) to generate a concise summary.
- Saves the summary back to server storage for retrieval.

Return Summary to User:

- Delivers the generated summary back to the user via the frontend interface.

Data Flow (Relationships):

User → Frontend Interface: User initiates interaction.

Frontend Interface → Upload/Input Audio: Frontend sends user audio to the system.

Upload/Input Audio → Local Server Storage: Audio is saved on the server.

Local Server Storage → Speech to Text Processing: Stored audio is processed into text.

Speech to Text Processing → Summarize with Language Model: Transcript is summarized.

Summarize → Local Server Storage: Summary is saved for retrieval.

Frontend Interface → Return Summary to User: User receives the final summary.

Frontend Interface ← Local Server Storage: Frontend can fetch stored transcripts/summaries.

User ← Return Summary to User: Process completes with user receiving the summary.

Use Case Diagram:

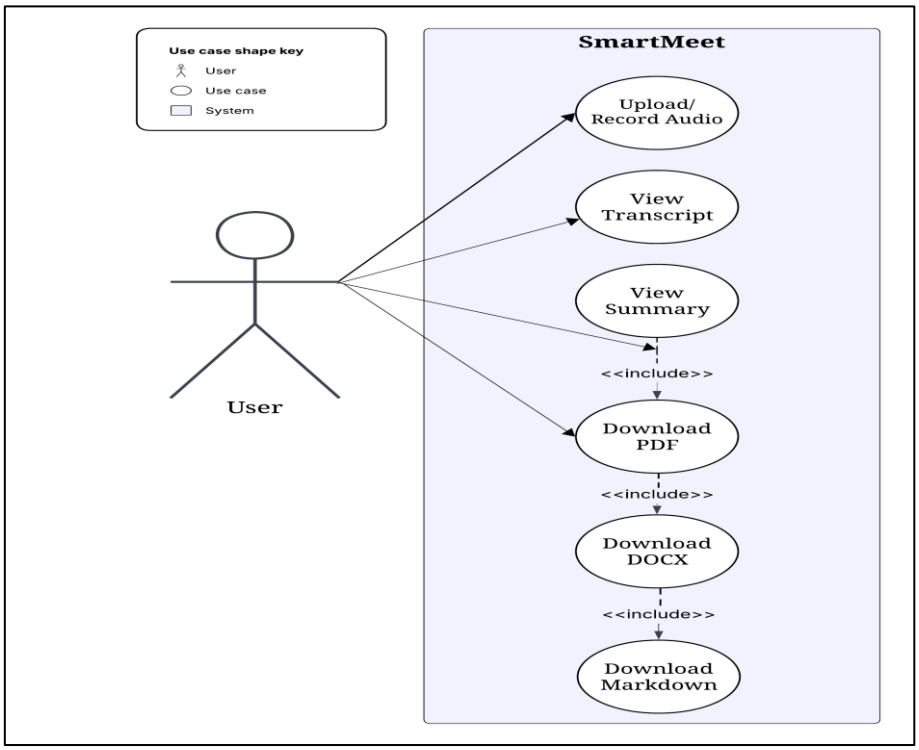

Figure 5.3 Use Case Diagram for SmartMeet

Actors:

- **User:** The primary actor who interacts with the system to upload or record meeting audio, view the transcript and summary and export the results in various formats.

Use Cases:

1. **Upload/Record Audio:** The user begins by uploading a pre-recorded meeting audio file or recording live audio through the system. This serves as the input for generating the transcript and summary.
2. **View Transcript:** After the audio is uploaded or recorded, the system converts speech to text and displays the meeting transcript for the user. The user can review, edit, or verify the generated text.

3. **View Summary:** The system processes the transcript to generate a concise meeting summary, highlighting key points, action items and decisions. This use case includes the Export Results use case, as summarization is a prerequisite for exporting data.
4. **Export Results:** The user can export the transcript or summary in different file formats for sharing or documentation purposes.

   This use case includes the following sub-use cases:

- Download PDF
- Download DOCX
- Download Markdown

5. **Download PDF / DOCX / Markdown (Sub-use cases):** The system allows the user to download the meeting summary or transcript in their preferred format for offline access or integration with other tools.

Relationships:

- **The User interacts directly with all main use cases:** Upload/Record Audio, View Transcript, View Summary and Export Results.
- **The View Summary use case includes the Export Results use case.**
- **The Export Results use case includes its sub-use cases:** Download PDF, Download DOCX and Download Markdown.

These represent dependencies where exporting data triggers the download options.

Chapter 6

Experimental Setup

6.1 Software Requirements: -

6.1.1. Frontend (React + Tailwind CSS)

- **Purpose:** User-facing web application for interacting with the Smart Meet system.
- **Use Case:** Allows users to upload or record meeting audio, view live transcripts and summaries and download results in multiple formats (PDF, DOCX, Markdown).
- **Version:** Latest stable (React 18.x, Tailwind CSS 3.x).

6.1.2. Python AI Service (FastAPI)

- **Purpose:** Python framework for handling AI processing tasks.
- **Use Case:** Executes speech-to-text transcription using the Whisper model, generates meeting summaries using HuggingFace Transformers and streams transcripts and summaries back to the frontend.
- **Version:** Latest stable (FastAPI 0.115.0).

Dependencies:

- **python-multipart: For file upload handling.**
- **sounddevice: For audio I/O streaming.**
- **numpy: For numerical and matrix computations.**
- **openai-whisper: For robust speech recognition and transcription.**
- **pyannote.audio: For multi-speaker detection and segmentation.**
- **transformers: For abstractive and extractive summarization.**
- **librosa: For audio feature extraction.**

6.1.3. Outputs

- **Purpose:** Provides multiple downloadable file formats for users.
- **Use Case:** Users can download PDF, DOCX, or Markdown versions of transcripts and summaries.
- **Implementation:** Implement download functionality in the backend (Spring Boot) and expose endpoints for the frontend to interact with.

6.1.4. Development Tools

Visual Studio Code (VSCode): IDE for Python and React development.

- **Use Case:** Writing and debugging Python AI service and frontend code.

  IntelliJ IDEA: IDE for Spring Boot backend development.

- **Use Case:** Writing and debugging Java backend APIs.

  Git: Version control for all project components.

- **Use Case:** Tracks changes in frontend, backend and Python AI services; supports collaboration via GitHub/GitLab.

6.1.5. Operating System

- **Supported OS:** Windows 11 / Ubuntu 22.04 LTS (cross-platform compatibility).
- **Processor:** Intel i5 / Ryzen 5 or higher.
- **RAM:** Minimum 8 GB (16 GB recommended for model inference).

6.2 Hardware Requirements: -

6.2.1. Processor (CPU)

- **Minimum:** Intel Core i5 (10th generation or AMD Ryzen 5 equivalent) with at least 4 cores and 8 threads, clock speed of 3.0 GHz or higher. Handles general scripting, live audio streaming and lightweight AI inference tasks, including Whisper transcription and Transformers summarization. Multi-core support is crucial for processing audio chunks concurrently without lag.
- **Recommended for Development**: Intel Core i7/i9 (or AMD Ryzen 7/9) with 8–16 cores and 16–32 threads, clock speed 4.0 GHz+. Accelerates CPU-bound tasks such as preprocessing audio, managing multiple user requests simultaneously and orchestrating AI inference pipelines efficiently. Example CPUs: Intel Core i9-13900K or AMD Ryzen 9 7950X.
- **Rationale:** Live transcription involves continuous numerical computations for audio buffers, Whisper model inference and Transformers summarization. CPUs with more cores reduce bottlenecks and maintain real-time performance. While CPU-only inference works, GPU acceleration significantly improves speed for concurrent users.

6.2.2. RAM (System Memory)

- **Minimum:** 16 GB DDR4/DDR5. Sufficient for running the frontend, backend and FastAPI AI service with a small number of concurrent live audio streams.
- **Recommended for Development:** 32–64 GB DDR4/DDR5. Allows multiple audio streams to be processed simultaneously, caching transcripts for summarization and storing temporary AI outputs without disk swapping.
- **Rationale:** Whisper and Transformers models consume several GBs of memory per audio stream. More RAM ensures smooth processing, prevents slowdowns due to swapping and allows developers to test multi-user scenarios locally.

6.2.3. Storage

- **Minimum:** 256 GB SSD (NVMe preferred) for OS, project files and small audio datasets.
- **Recommended for Development:** 512 GB–1 TB NVMe SSD. Essential for live audio recordings, transcript caching and downloadable files in PDF, DOCX, or Markdown format. SSD ensures fast read/write operations, reducing latency in live transcription buffering.
- **Rationale:** Fast storage is crucial as audio files are continuously recorded, processed and stored. Slow storage can lead to bottlenecks during live transcription or when exporting results.

6.2.4. Graphics Processing Unit (GPU)

- **Minimum:** NVIDIA GPU with 4–8 GB VRAM (e.g., GTX 1660 Ti or RTX 2060). Optional for basic CPU-only Whisper inference; enables faster processing.
- **Recommended for Development:** NVIDIA RTX 30/40 series with 12–24 GB VRAM (e.g., RTX 3080 Ti 12 GB, RTX 4090 24 GB). Efficient for real-time transcription and summarization of multiple simultaneous audio streams. Multi-GPU setups can accelerate processing for concurrent sessions.
- **Rationale:** Whisper and Transformers models are GPU-intensive. GPU acceleration reduces transcription latency, improves responsiveness for live meetings and supports higher concurrency. CPU-only inference is possible but slower, especially for multiple users.

6.2.5. Power Supply and Cooling

- **Minimum:** 650W 80+ Bronze rated PSU. Sufficient for CPU and basic GPU setups.
- **Recommended:** 850W+ 80+ Gold/Platinum PSU with adequate cooling (e.g., multiple case fans or liquid cooling). Required for high-end GPUs such as RTX 4090, which draw significant power under load.

- **Rationale:** Prolonged transcription and summarization generate heat. Stable power and cooling prevent thermal throttling, maintain hardware longevity and avoid unexpected crashes during extended live sessions.

6.2.6. Networking (Optional for Cloud/Distributed Setup)

- **Minimum:** Gigabit Ethernet.
- **Recommended:** 10 Gbps Ethernet or high-speed internet to handle multiple users uploading audio simultaneously and downloading processed transcripts and summaries.
- **Rationale:** Live meeting audio streams can consume significant bandwidth. High-speed networking ensures smooth audio streaming, reduces latency and supports cloud-hosted AI inference services for multiple concurrent users.

6.2.7. Additional Peripherals

- **Microphone:** High-quality USB microphone or headset for live recording. Reduces background noise and improves transcription accuracy.
- **Speakers / Headphones:** For audio playback during testing and quality verification.
- **Monitor:** Full HD or higher resolution to facilitate frontend/backend development and monitoring of real-time transcription results.
- **Rationale:** High-quality peripherals ensure accurate audio capture and playback, which directly impacts the quality of AI transcription and summarization. Developers can also monitor live results effectively for testing and debugging.

Chapter 7

Project Planning

Gantt Chart:

Figure 6.1 July (Gantt Chart)

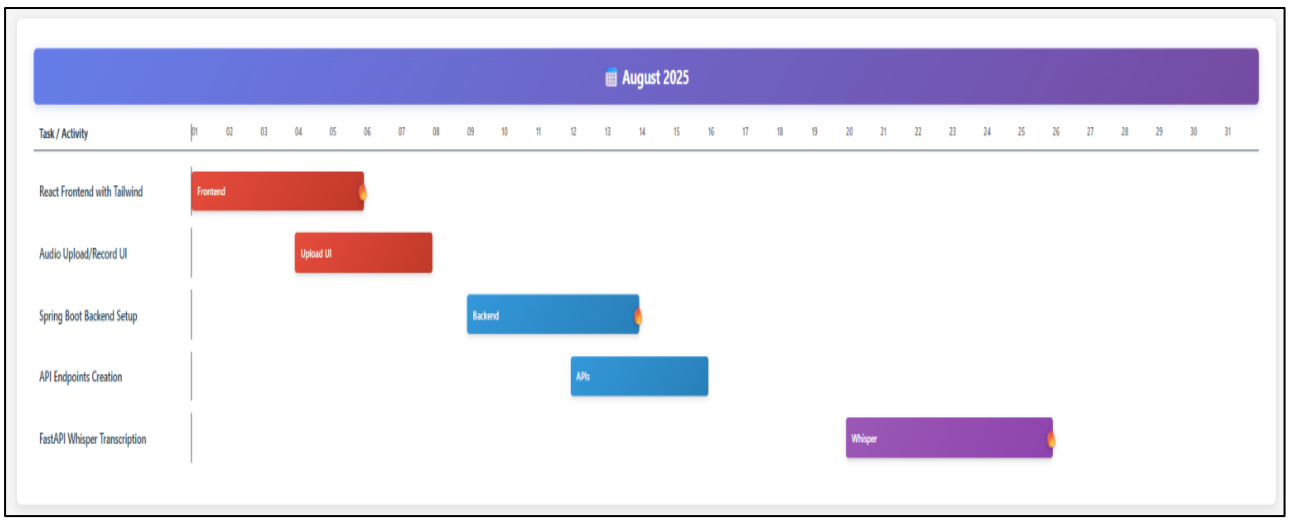

Figure 6.2 August (Gantt Chart)

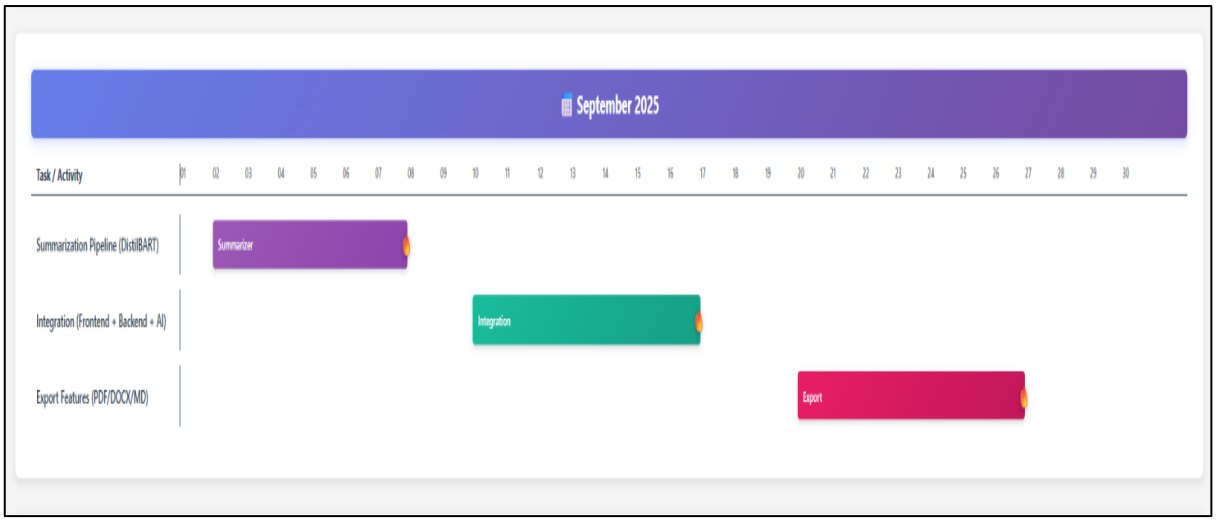

Figure 6.3 September (Gantt Chart)

Figure 6.4 Octoberr (Gantt Chart)

Chapter 8

Expected Outcome

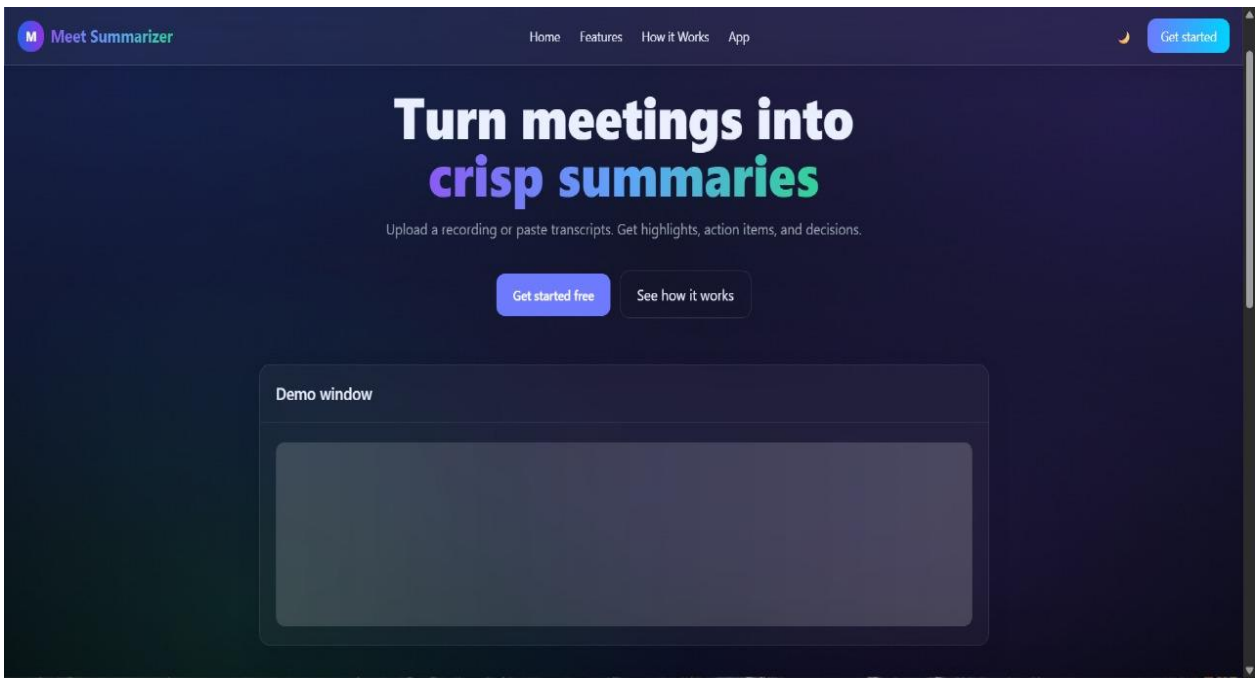

Figure 8.1 Meet Summarizer Interface

This image shows the homepage of the Meet Summarizer web application. At the top, the navigation bar includes sections such as Home, Features, How it Works and App, along with a "Get started" button. The central heading displays the tagline "Turn meetings into crisp summaries", highlighting the app's purpose. Below the tagline, a short description invites users to upload a recording or paste transcripts to get highlights, action items and decisions. Two call-to-action buttons — Get started free and See how it works — are prominently placed in the center. The lower part of the page features a Demo window section, intended to preview or display example outputs. The interface has a modern, dark-themed gradient design with a clean and professional layout.

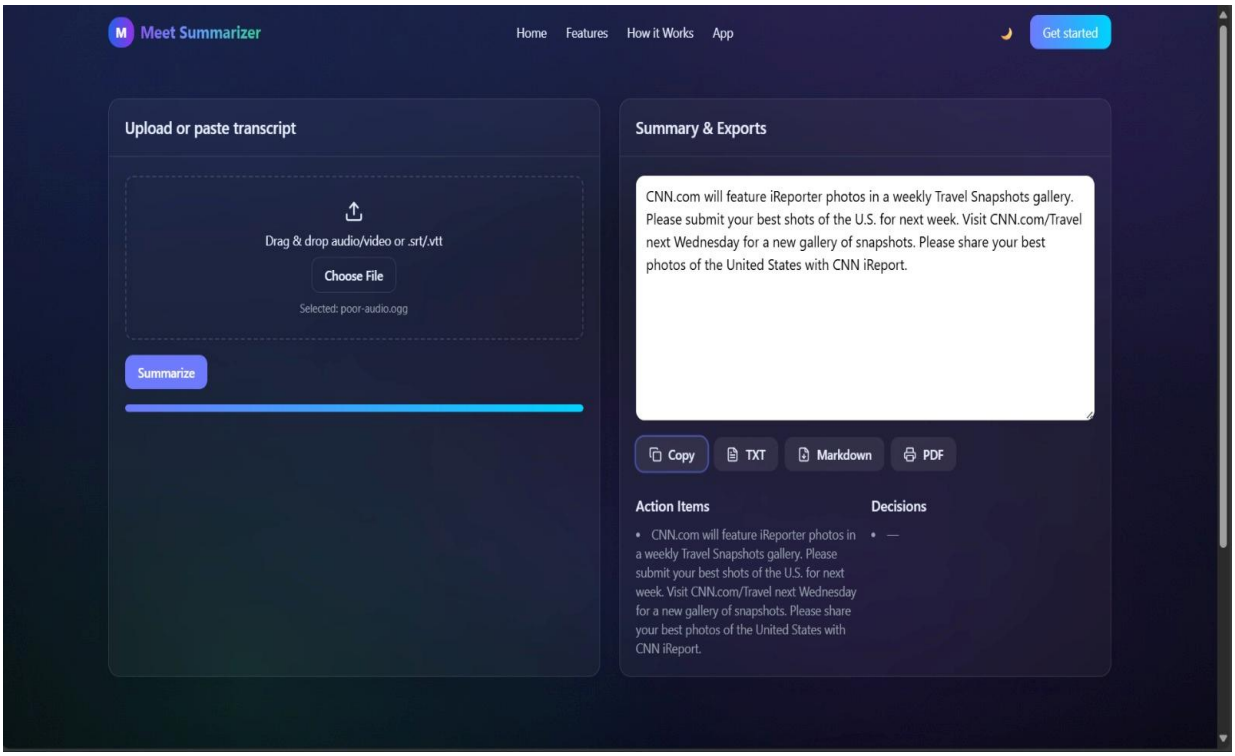

Figure 8.2 Meet Summarizer Interface with Uploaded Transcript

This image shows the Meet Summarizer web application interface during active use. On the left side, the Upload or paste transcript section allows users to drag and drop audio, video, or subtitle files, or select a file manually. A file named poor-audio.ogg has been uploaded and a progress bar is displayed below the Summarize button, indicating processing. On the right side, the Summary & Exports panel displays the generated summary text from the uploaded audio. Below it, export options such as Copy, TXT, Markdown and PDF are available for saving or sharing the summary. The lower section organizes content into Action Items and Decisions, showing structured results from the summarization. The design follows a clean, dark-themed interface with a clear layout separating input and output sections.

References

[1] Asthana, Anurag, Aaron Halfaker, Chieh-Yang He and Jonathan Morgan. "Summaries, Highlights and Action Items: Design, Implementation and Evaluation of an LLM-Powered Meeting Recap System." arXiv preprint arXiv:2502.03111, 2025. LINK

[2] Kadam, Pratiksha and Shailesh Jadhav. "Minutes of Meeting Generation for Online Meetings Using NLP and ML Techniques." International Research Journal of Engineering and Technology (IRJET) 11, no. 3 (2024): 2767-2773. LINK

[3] Shinde, Onkar, Pranav Borhade, Sahil Gholap and Akshay Dhangar. "Exploring Advances in Meeting Minutes Generation and Face Attendance Systems (ConvoLogix Model)." International Journal of Engineering Research & Technology (IJERT) 13, no. 3 (2024): 1-6. LINK

[4] Ranjith, C. and Vishnu Priya. "Online Meeting Summarizer and Report Generation using GenAI." In Proceedings of the International Conference on Artificial Intelligence and Machine Learning (ICAIML), pp. 112-118. IEEE, 2024. LINK

[5] Wyawahare, Medha, Madhuri Shelke, Siddharth Bhorge and Rohit Agrawal. "AI Powered Multilingual Meeting Summarization." Proceedings of the 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2024. LINK

[6] Akshatha, P. S., B. Akash, V. Harshith, C. Sumukh, V. Gowardhan Reddy and Shravya Shetty. "NLP-Driven Video Transcription: A Comprehensive Survey and Innovative Office Meeting Automation." Proceedings of the 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2024. LINK

[7] Rennard, Virgile, Guokan Shang, Julie Hunter and Michalis Vazirgiannis. "Abstractive Meeting Summarization: A Survey." Transactions of the Association for Computational Linguistics 11 (2023): 49-69. LINK

[8] Singh, Aman, Ankit Gautam, Deepanshu, Gautam Kumar, Lokesh Kumar Meena and Shashank Saroop. "Automated Minutes of Meeting Using a Multimodal Approach." International Journal for Research in Applied Science & Engineering Technology (IJRASET) 11, no. 12 (2023): 2059-2063. LINK

[9] Gupta, Riya, Rahul Sharma and Neha Patel. "Building Real-World Meeting Summarization Systems using LLMs." International Journal of Advanced Computer Science and Applications (IJACSA) 14, no. 11 (2023): 327-334. LINK

[10] Khodake, Nikhil, Shweta Kondewar and Swarda Bhandare. "Automatic Generation of Meeting Minutes Using NLP." International Journal for Research in Applied Science & Engineering Technology (IJRASET) 11, no. 5 (2023): 7015-7019. DOI: 10.22214/ijraset.2023.53353. LINK