A PROJECT REPORT

ON

MinuteMind: Intelligent Meeting Summary and Conflict Detection System

SUBMITTED TO AN AUTONOMOUS INSTITUTE, AFFILIATED TO SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

CSE (AI)

SUBMITTED BY

Arjun Kallurkar A19

Atharv Yeole A24

Atharva Deshmukh A29

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

G H RAISONI COLLEGE OF ENGINEERING AND MANAGEMENT

WAGHOLI, PUNE 412207

2025-26

CERTIFICATE

This is to certify that the project report entitled "**MinuteMind: Intelligent Meeting Summary and Conflict Detection System**",

Submitted by

ARJUN KALLURKAR Roll No.: 19

ATHARV DESHMUKH Roll No.: 29

ATHARV YEOLE Roll No.: 24

are a Bonafide students of this institute and the work has been carried out by them under the supervision of **V. Y. Baviskar**, and it is approved for the partial fulfillment of the requirement of an Autonomous Institute, Affiliated to SAvitribai Phule Pune University, Pune for the award of the degree of **Bachelor of Technology in Computer Science Engineering (AI & AIML)** in the academic year 2025-26.

Dr. V. Y. Baviskar
Guide

Dr. R. Y. Sable
HoD

Dr. N. B. Hulle
I/C Director

Dr. R. D. Kharadkar
Director, GHRCEM, Pune

External Examiner

Date:

Place:

ACKNOWLEDGMENT

It gives us great pleasure in presenting **MinuteMind: Intelligent Meeting Summary and Conflict Detection System** as our B.Tech. project. Words have never seemed as inadequate as now when we are endeavoring to express our gratitude at the culmination of our B.Tech. Project to all those who have made it possible. Even the best efforts are waste, without the proper guidance and advice of our project guide **Dr. Vaishali Y. Baviskar** for the consistent guidance, co-operation, inspiration, practical approach and constructive criticism, which provided us the much needed impetus to work hard and also thanks to **Dr. R. Y. Sable**, Head of AI and AIML Department for her continuous support & valuable suggestions.

We take this opportunity to thank our Director **Dr. N. B. Hulle**, for his whole hearted support, motivation, and valuable suggestions.

We would also like to thank **Dr. Vaishali Baviskar and Dr. Trupti Mohota**, Project Coordinators for their valuable support in providing us with the required information.

At the end, we would like to give special thanks to all staff members from **AI and AIML Department** of G H Raisoni College of Engineering and Management, Pune and our colleagues for their kind support & timely suggestions.

Arjun Kallurkar Roll No.: A19

Atharv Yeole Roll No.: A24

Atharva Deshmukh Roll No.: A29

ABSTRACT

The excessive use of virtual meetings has led to the accumulation of extensive, unstructured archives of organizational knowledge. Manually extracting actionable insights from these recordings is, however, suboptimal and prone to inaccuracies. While cloud hosted AI solutions present an alternative, they also introduce considerable obstacles related to data privacy, security, and expenditure. This paper introduces MinuteMind, a privacy focused, on premise system engineered to transform conversational meeting data into organized, actionable intelligence. The system employs a fine tuned Whisper model integrated with pyannote, audio in a WhisperX Wrapper for precise transcription and speaker diarization, which is then paired with a fine tuned Phi-3 Mini Small Language Model (SLM) for in depth analysis. Both models were adapted using QLoRA on the AMI Corpus and subsequently optimized for efficient local execution. Our approach centers on the automated extraction of tasks, the identification of conflicts using stance analysis, and the creation of formal Minutes of Meeting (MoM) documents, with all operations performed exclusively on local hardware. By utilizing an optimized, open-source stack, MinuteMind offers an accessible, secure, and economical tool for enhancing team productivity and safeguarding corporate memory.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automatic Speech Recognition
CER	Character Error Rate
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
GGUF	GPT-Generated Unified Format
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LLM	Large Language Model
MoM	Minutes of Meeting
NLP	Natural Language Processing
ORM	Object-Relational Mapping
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
RAM	Random Access Memory
SLM	Small Language Model
SQL	Structured Query Language
SRS	Software Requirements Specification
UI	User Interface
WER	Word Error Rate
XML	Extensible Markup Language

Chapter 1

INTRODUCTION

1.1 Overview

Meetings are a key forum for strategic planning, decision-making, and group problem-solving in modern collaborative settings. A vast amount of organizational knowledge has been created as a result of the unprecedented rise in the number of recorded meetings brought about by the widespread adoption of virtual and hybrid work models. Because the audio and textual recordings are unstructured and .ted with conversational noise—such as filler words, interruptions, and colloquial expressions—the inherent value of these conversations is still largely unrealized despite their abundance. It takes a lot of work and is prone to error to manually extract actionable insights from these records. This can result in missed tasks, poorly documented decisions, and diminished accountability, all of which can negatively impact organizational efficiency

Conventional methods, such as taking minutes by hand or using automated summarization software, fall short in addressing these issues. While most automated solutions only offer generic, high-level summaries devoid of crucial information like task assignments, accountable parties, deadlines, or points of disagreement, manual documentation is subjective, inconsistent, and unable to scale. These shortcomings reduce the usefulness of recorded meetings in practice, impede follow-up activities, and postpone project progress. A strong, organized, and automated system that transforms unprocessed conversational data into a trustworthy, actionable record is obviously needed.

This project introduces **MinuteMind**, an on-premise, privacy-focused Meeting Intelligence Engine that transforms unstructured dialogue into a structured, persistent, and actionable corporate memory. Rather than simply condensing conversa-

tions, MinuteMind performs deep semantic and pragmatic analysis, automatically extracting tasks along with their assignees and deadlines, identifying points of conflict through stance analysis, and normalizing temporal expressions such as "next week" or "end of day tomorrow" into standardized dates using the `dateparser` library. The system assembles these outputs into a comprehensive, editable Minutes of Meeting (MoM) document, providing organizations with a structured, actionable, and fully auditable record of their discussions.

MinuteMind is designed for local deployment to guarantee that sensitive data never leaves organizational premises. The system employs a Flask backend for orchestrating data flow, managing database interactions, and providing API endpoints, alongside an HTML, CSS, and JavaScript frontend that renders a fully interactive, editable dashboard. At its analytical core, MinuteMind integrates a fine-tuned Whisper model within a WhisperX wrapper for precise transcription and speaker diarization, coupled with the Phi-3 Mini Small Language Model, fine-tuned via QLoRA for structured task extraction, conflict detection, and summarization. The combined use of these models ensures high-quality, context-aware analysis without reliance on cloud-based services, making the solution both cost-effective and privacy-compliant.

1.2 Motivation

The primary motivation for developing MinuteMind stems from the persistent inefficiency in translating meeting discussions into actionable intelligence. Organizations invest substantial time in meetings, yet critical outputs—including commitments, decisions, and disagreements—are frequently lost or inadequately documented. Manual note-taking is error-prone and inconsistent, while conventional automated summarization systems provide insufficient detail, often omitting tasks, assignees, or deadlines. This gap results in missed deadlines, duplication of effort, and a lack of accountability, creating a pressing need for an automated, reliable, and structured solution that ensures actionable insights are preserved and easily accessible.

A second motivating factor arises from the limitations and risks of existing AI tools. Many cloud-hosted solutions, though powerful, are unsuitable for handling sensitive or proprietary organizational data due to privacy and security concerns, as well as variable operational costs. MinuteMind addresses these issues by providing a fully on-premise alternative, leveraging compact yet capable Small Language Models (SLMs) that enable high-quality natural language processing on standard hardware.

This approach allows organizations to maintain full control over their data while accessing advanced AI functionality.

Additionally, there is a strong motivation to democratize access to sophisticated meeting intelligence. Proprietary platforms with advanced summarization and analysis capabilities are often expensive or subscription-based, limiting accessibility for academic institutions, small businesses, and research projects. MinuteMind, built entirely on an open-source stack, empowers users to generate structured, actionable insights from meetings without financial or infrastructural barriers. By combining transcription, diarization, and fine-tuned semantic analysis, the system delivers a comprehensive, practical, and fully accessible tool for enhancing team productivity.

1.3 Problem Definition

The core problem addressed by MinuteMind is the challenge of extracting high-value, structured information from the inherently unstructured and noisy content of multi-party meetings. Spoken dialogue contains essential organizational knowledge, including actionable tasks, decisions, deadlines, and dissenting viewpoints, yet this information is embedded within complex conversational structures. Conversational artifacts such as interruptions, filler words, overlapping speech, and implicit context make automated extraction highly non-trivial.

Existing solutions inadequately address these issues. Manual minute-taking is slow, inconsistent, and susceptible to human error. Standard automated summarization tools typically generate superficial textual overviews that lack detailed information regarding task allocation, scheduling, and conflict resolution. Consequently, organizations face a tangible loss of knowledge, reduced operational efficiency, and limited accountability.

MinuteMind targets these challenges by providing a system capable of performing:

- **Structured Task Extraction:** Automatic identification of tasks, responsible parties, and deadlines, converting informal temporal cues into precise dates using `dateparser`.
- **Conflict Detection:** Identification of disagreement or dissent among participants, ensuring that critical issues requiring attention are flagged for review.
- **Comprehensive MoM Generation:** Synthesis of multi-party discussion into a

formal, structured, and editable Minutes of Meeting document that includes summaries, tasks, conflicts, and decisions.

- **Local, Privacy-Preserving Deployment:** Execution entirely on-premise using open-source software, eliminating the need for cloud services and mitigating associated privacy, security, and cost concerns.

1.4 Project Scope and Limitations

1.4.1 Project Scope

The project scope encompasses the development of a fully functional prototype of MinuteMind, demonstrating the feasibility of end-to-end, on-premise meeting intelligence using fine-tuned Small Language Models. Key features include:

- **Data Input:** Accepting pre-recorded audio files (WAV, MP3) and raw transcripts.
- **Processing Pipeline:** A complete workflow consisting of:
  - High-fidelity transcription using a fine-tuned Whisper model within WhisperX.
  - Speaker diarization using pyannote.audio.
  - Semantic analysis with the Phi-3 Mini model fine-tuned via QLoRA for structured task extraction, conflict detection, and abstractive summarization.
- **Analytical Outputs:** Generation of:
  - Editable, comprehensive Minutes of Meeting (MoM) documents.
  - A list of assigned tasks with normalized deadlines.
  - Detected conflicts and points of disagreement.
- **Temporal Normalization:** Conversion of vague expressions such as "next week" or "by tomorrow" into machine-readable date formats using `dateparser`.
- **Interactive User Interface:** Web-based dashboard rendered via Flask with HTML, CSS, and JavaScript, allowing users to review, edit, and export outputs in real time.

1.4.2 Limitations

To maintain focus and ensure feasibility, the following limitations are acknowledged:

- **Model Dependence:** System accuracy depends on the underlying Whisper and Phi-3 Mini models. While fine-tuning improves performance, inherent limitations of these models remain.
- **Language Support:** The prototype supports English only. Other languages are out of scope.
- **Audio Quality:** Effectiveness may be reduced by low-quality audio, high background noise, strong accents, or highly ambiguous speech.
- **Mobile Accessibility:** The web app works on phones but lacks a dedicated mobile app for better performance.
- **Context Limitation:** Analysis is confined to single meeting transcripts. Long-term memory or cross-meeting contextual understanding is not included in the current scope.

1.5 Key Terminology

To facilitate the understanding of this report, Table 1.1 defines several key acronyms and technical concepts central to the MinuteMind Engine. A comprehensive list of all abbreviations used throughout the document is provided in the dedicated List of Abbreviations section.

Table 1.1: Key Terms and Definitions for the MinuteMind Engine

Acronym / Term	Definition
MoM	Minutes of Meeting. A structured, formal record of a meeting's key discussions, decisions, and actionable tasks.
SLM	Small Language Model. A compact and computationally efficient language model, such as Phi-3 Mini, designed to run effectively on local, consumer-grade hardware.
WhisperX	An enhanced Automatic Speech Recognition (ASR) system that provides highly accurate transcription along with precise, word-level timestamps, which are crucial for subsequent analysis.
Speaker Diarization	The process of partitioning an audio stream into segments according to speaker identity. It answers the question of "who spoke when," which is essential for assigning tasks and analyzing interactions.
QLoRA	Quantized Low-Rank Adaptation. An efficient fine-tuning technique that allows large pre-trained models to be adapted for specific tasks on limited hardware by training only a small set of quantized adapter weights.
Flask	A lightweight Python web framework used in this project to build the backend API, which orchestrates the entire data processing pipeline and serves the user interface.

Chapter 2

LITERATURE REVIEW

2.1 Literature Survey

Automated meeting analysis has progressed from basic transcription to advanced AI-driven comprehension, encompassing tasks such as speaker diarization, dialogue understanding, and structured summarization. This progression relies on developments in high-fidelity speech-to-text models, accurate speaker attribution, and dialogue-aware natural language processing. This review situates the MinuteMind Engine within these evolving research paradigms.

Foundational resources such as the **AMI meeting corpus [1]** have enabled large-scale, multi-modal research in meeting analysis, providing a standard benchmark for both transcription and summarization tasks. Accurate transcription forms the backbone of any meeting intelligence system. Early work by **Radford et al. [2]** introduced Whisper, which demonstrated robust large-scale speech recognition, while **Bain et al. [3]** extended this capability with WhisperX, enabling precise word-level timestamps in long-form audio. These timestamps are essential for downstream processing such as speaker diarization, which assigns speech segments to participants. **Bredin et al. [4]** provide a comprehensive framework with `pyannote.audio`, illustrating the efficacy of neural building blocks for speaker diarization in multi-party meetings. A broader survey by **Park et al. [5]** reviews deep learning approaches to speaker diarization, highlighting solutions for overlapping speech, varied acoustic conditions, and multi-participant scenarios.

Once diarized transcripts are obtained, semantic understanding and summarization become critical. Classical approaches, as reviewed by **Pandya and Gawande [6]**, utilized machine learning classifiers such as SVMs and HMMs for identifying action items and generating structured minutes. These models, however, are sensitive

to domain-specific terminology, which can degrade performance, as shown by **Koay et al. [7]**. This challenge motivates the use of adaptable or domain-aware models.

Recent advances in Large Language Models (LLMs) have enabled more sophisticated processing. Datasets like **DialogSum [8]** provide real-life conversational contexts for dialogue summarization, improving the ability of models to handle multi-turn, natural interactions. Dialogue act classification, explored by **Raheja and Tetreault [9]**, emphasizes the importance of context-aware modeling for accurately capturing speaker intent and communicative function. Additionally, parameter-efficient fine-tuning techniques such as QLoRA [10] facilitate the adaptation of compact LLMs like Phi-3 Mini to domain-specific meeting data without requiring large-scale hardware, enabling high-performance analysis in local, on-premise deployments. **Laskar et al. [11]** further discuss the design of real-world meeting summarization systems using LLMs, highlighting both practical and architectural considerations.

The MinuteMind Engine integrates these advancements, combining WhisperX-based transcription, pyannote.audio diarization, and Phi-3 Mini fine-tuned via QLoRA to deliver structured meeting intelligence. Unlike many existing tools, it operates entirely on-premise using a Flask-based architecture, ensuring both data privacy and accessibility.

Table 2.1: Summary of Related Works on Meeting Understanding

Paper Title & Year	**Details of Publication**	**Findings**
Koay et al. (2020) How Domain Terminology Affects Meeting Summarization Performance [7]	arXiv preprint arXiv:2011.00692	Domain-specific terminology significantly affects summarization performance, highlighting the need for adaptable models.
Laskar et al. (2023) Building Real-World Meeting Summarization Systems using Large Language Models [11]	arXiv preprint arXiv:2310.19233	Provides practical guidance on constructing LLM-based meeting summarization systems, with architectural and deployment considerations.

Continued on next page

Table 2.1 Continued from previous page

Paper Title & Year	Details of Publication	Findings
Park et al. (2021) A Review of Speaker Diarization: Recent Advances with Deep Learning [5]	arXiv preprint arXiv:2101.09624	Reviews modern deep learning approaches for speaker diarization, including overlapping speech and varied recording conditions.
Bain et al. (2023) WhisperX: Time-Accurate Speech Transcription of Long-Form Audio [3]	arXiv preprint arXiv:2303.00747	Introduces WhisperX for precise transcription with word-level timestamps, enabling robust downstream analysis.
Pandya & Gawande (2022) Automatic Generation of Minutes of Meetings [6]	International Journal of Scientific Research in Science, Engineering and Technology	Uses classical ML techniques to extract action items and generate structured MoM.
Mccowan et al. (2005) The AMI Meeting Corpus [1]	Int'l. Conf. on Methods and Techniques in Behavioral Research	Provides a foundational multi-modal dataset for meeting research.
Dettmers et al. (2023) QLoRA: Efficient Finetuning of Quantized LLMs [10]	NeurIPS 2023	Introduces efficient finetuning for compact LLMs, enabling high-performance on resource-limited hardware.
Chen et al. (2021) DialogSum: A Real-Life Scenario Dialogue Summarization Dataset [8]	arXiv preprint arXiv:2105.06762	Provides multi-turn dialogue dataset for training summarization models in realistic conversational contexts.

Continued on next page

Table 2.1 Continued from previous page

Paper Title & Year	Details of Publication	Findings
Raheja & Tetreault (2019) Dialogue Act Classification with Context-Aware Self-Attention [9]	arXiv preprint arXiv:1904.02594	Introduces context-aware dialogue act classification, essential for understanding speaker intent.
Radford et al. (2022) Robust Speech Recognition via Large-Scale Weak Supervision [2]	arXiv preprint arXiv:2212.04356	Demonstrates robust, large-scale speech recognition, forming the basis for downstream transcription and analysis tasks.
Bredin et al. (2020) pyannote.audio: Neural Building Blocks for Speaker Diarization [4]	arXiv preprint arXiv:1911.01255	Provides modular neural tools for speaker diarization, enabling flexible and accurate multi-speaker processing.

2.2 Drawbacks of Existing Work

While the field of automated meeting analysis has made significant strides, a review of existing systems reveals several persistent drawbacks that limit their practical adoption, particularly within privacy-conscious or resource-constrained environments. The MinuteMind Engine is specifically designed to address these gaps.

A primary limitation is the pervasive dependency on cloud-based infrastructure. Many state-of-the-art systems leverage powerful, proprietary models that are only accessible via third-party APIs. This architectural choice introduces significant data privacy and security risks, as it requires organizations to transmit potentially confidential internal discussions to external servers. For industries governed by strict data sovereignty regulations such as GDPR or HIPAA, this is often an insurmountable barrier. Furthermore, reliance on cloud services can lead to unpredictable operational costs tied to API usage and data transfer, making it a less viable option for continuous, large-scale use.

Another significant issue is the narrow focus on abstractive summarization at the

expense of structured, actionable data. Most existing tools excel at generating a high-level narrative of a meeting's discussion. While useful for a quick recap, these summaries often fail to extract the specific commitments that drive project progress: explicitly defined tasks, deadlines, and the individuals responsible for them. This lack of structured output means that crucial action items remain buried in the text, forcing users to manually review the summary or transcript, thereby defeating the primary purpose of automation and failing to close the accountability loop.

Furthermore, the market is dominated by proprietary tools with high costs and limited accessibility. The most capable meeting intelligence solutions are typically offered as commercial, software-as-a-service (SaaS) products locked behind expensive subscription models. This pricing structure places them out of reach for academic researchers, educational institutions, non-profit organizations, and small businesses. This lack of access to state-of-the-art technology stifles innovation and prevents a wider range of users from benefiting from automated productivity tools.

Finally, very few systems offer robust offline, real-time processing capabilities. The dependency on a stable internet connection for cloud-based analysis makes many tools unsuitable for use in secure environments where network access may be restricted or unavailable, such as in government facilities or sensitive research labs. This reliance on external connectivity also introduces a potential point of failure and can add significant latency to the analysis process, hindering immediate review and action upon a meeting's conclusion.

2.3 Objectives of Proposed Work

In light of the limitations identified in existing systems, the main objective of this project is to develop a self-contained, on-premise Meeting Intelligence Engine. Built using a Flask-based architecture, the system is designed to transform raw meeting data into structured, actionable insights without compromising data privacy or incurring operational costs. The key goals of this proposed work are detailed as follows:

1. **To Implement Accurate, Structured Task Extraction.** A primary goal is to move beyond simple summarization and develop a system that can reliably identify and extract actionable tasks from complex, multi-speaker dialogues. This involves not only recognizing the commitment itself but also accurately assigning responsibility to a specific participant (identified via diarization) and extracting any associated deadlines. The output will be a structured list of

action items that can be directly used for project management.

2. **To Develop a Robust Conflict Detection Mechanism.** The system aims to automatically flag moments of significant disagreement, dissent, or unresolved debate within a conversation. This will be achieved through stance analysis of the dialogue, enabling the model to understand when speakers are presenting opposing viewpoints. By highlighting these conflicts, the engine provides managers with a clear overview of critical issues that require follow-up and resolution, ensuring they are not overlooked.

3. **To Create an Interactive Visualization and Validation Interface.** A user-friendly web interface, powered by Flask, will be developed to present the extracted insights in an intuitive manner. This dashboard will not be a static report but an interactive tool where users can view tasks, conflicts, and schedules. Crucially, the interface will allow users to validate, edit, and correct any of the automatically generated information, implementing a human-in-the-loop system to ensure the final output is 100% accurate before being finalized.

4. **To Integrate Automated Temporal Parsing and Scheduling.** The engine will be capable of interpreting natural language temporal expressions (e.g., "by next Friday," "in two weeks," "end of the quarter"). This will be achieved by integrating a dedicated parsing library to convert these ambiguous phrases into a standardized, machine-readable date format (YYYY-MM-DD). This functionality is a prerequisite for automated scheduling and allows for the seamless export of tasks into standard calendar formats.

5. **To Deliver a Privacy-Focused, Open-Source Implementation.** A core architectural principle of this project is to ensure complete data privacy through an on-premise, open-source design. This objective will be met by exclusively employing open-source models, including Phi-3 Mini and WhisperX, which can be run locally. These models will be adapted for the specific domain of meeting analysis using efficient fine-tuning techniques like QLoRA, demonstrating that high-quality, intelligent analysis can be achieved without relying on third-party cloud services.

Chapter 3

METHODOLOGY

This chapter describes the complete methodology adopted for the design, training, and evaluation of the MinuteMind Engine, an AI-powered Minutes of Meeting (MoM) generator. The methodology encompasses the preparation of the training datasets, the fine-tuning of models for both speech-to-text and language understanding components, and the evaluation framework used to measure the system's performance. The primary objective of this methodology is to ensure a high degree of automation, reliability, and interpretability in transforming raw meeting audio into structured summaries, action items, and conflict detections.

3.1 Dataset Preparation and Preprocessing

The methodology begins with a meticulous dataset preparation phase, which serves as the foundation for both the Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) components of the system. This stage was critical for tailoring general-purpose models to the specific nuances of meeting discourse, which is characterized by spontaneous speech, interruptions, and domain-specific terminology.

3.1.1 AMI Corpus Structure and Parsing for ASR

For the ASR fine-tuning task, the AMI Meeting Corpus was exclusively used due to its unparalleled richness in multimodal content and highly detailed, time-aligned annotations. The dataset is composed of multi-speaker meetings with synchronized audio and XML-based textual transcripts, making it an ideal resource for this project. The file structure is highly organized, with unique meeting identifiers like `EN2004a`

and corresponding speaker-specific segment files such as `ES2005a.A.segments.xml`. A custom Python parsing script was developed to programmatically process these XML files, specifically targeting the `<w>` (word) tags. These tags contain critical attributes, including `nite:id`, `starttime`, and `endtime`, which are essential for creating precise audio-text pairs. An example of this granular annotation is as follows:

```
<w nite:id="ES2006a.A.words52" starttime="227.37" endtime="229.0">So</w>
```

By extracting and concatenating the text content between the specified start and end times for each continuous speaker turn, we programmatically generated a high-fidelity training dataset. This automated process yielded approximately 775 high-quality samples, each structured as a JSON object containing the audio file path, start time, end time, the full transcribed text for that segment, and the speaker ID. This structured format is optimal for training modern ASR models and ensures a direct mapping between audio segments and their corresponding ground-truth text, which is a prerequisite for effective supervised fine-tuning.

3.1.2 Audio Preprocessing for Whisper

All audio files referenced in the generated dataset were subjected to a standardized preprocessing pipeline to ensure consistency and model compatibility. Leveraging the `ffmpeg` library, each audio file was first resampled to a uniform 16 kHz sampling rate. This step is non-negotiable, as it aligns the audio with the native sampling rate expected by the Whisper model's architecture, which was trained on audio at this frequency. Submitting audio at a different rate would lead to a mismatch and degraded performance. Subsequently, the audio was segmented into 30-second chunks. This duration was chosen strategically, as the Whisper encoder processes audio in fixed 30-second windows. By aligning our data with this architectural constraint, we ensure that each training sample is processed optimally by the model, balancing sufficient conversational context with manageable memory requirements during the training process.

3.1.3 Text Data Structuring for Language Model Fine-Tuning

To adapt the Phi-3 model for the complex task of structured information extraction, a custom instruction-style dataset was meticulously created. This process involved the manual curation and semi-automated generation of approximately 160 examples. Each example was carefully designed to teach the model the desired input-output

mapping and consisted of a multi-part prompt structure. This included a system instruction defining the model's role and the required JSON output schema, a full diarized meeting transcript as the primary input, and a target output in the form of a perfectly structured JSON object containing keys for `summary`, `tasks`, and `conflicts`. This instruction-tuning approach is fundamentally important as it constrains the otherwise open-ended generative model, guiding it to produce predictable, machine-readable outputs and transforming it from a general text generator into a reliable structured data extractor. This step is critical for ensuring the reliability of the entire downstream pipeline, which depends on syntactically valid JSON.

3.2 Training and Fine-Tuning Methodology

The MinuteMind Engine consists of two key fine-tuned components: (1) the WhisperX-based ASR model for transcription and diarization, and (2) the Phi-3 Mini-based NLU model for summarization, task, and conflict extraction. Both components were adapted using parameter-efficient techniques to ensure feasibility on consumer-grade hardware, making the solution accessible without requiring enterprise-level resources.

3.2.1 Fine-Tuning the WhisperX Model

The `openai/whisper-small` model was fine-tuned on our prepared AMI Corpus dataset using the QLoRA (Quantized Low-Rank Adaptation) methodology. This parameter-efficient technique is particularly well-suited for this project as it allows for significant model adaptation on resource-constrained hardware by freezing the vast majority of the base model's weights and only training a small set of injected, low-rank adapter matrices. This drastically reduces the VRAM requirements compared to full fine-tuning. The training was configured with the following key hyperparameters:

- **LoRA rank (r):** 8
- **LoRA alpha (α):** 16
- **Learning rate:** 1e-4
- **Batch size:** 16 (using gradient accumulation to simulate a larger effective batch size)
- **Epochs:** 10

The primary training objective was to minimize the Word Error Rate (WER) between the model's transcribed output and the ground-truth transcripts from our dataset. For efficient deployment in the final application, the fine-tuned adapter weights were merged with the base model, and the resulting model was converted to the `CTranslate2` format. This specialized inference engine applies aggressive optimizations, including layer fusion and 8-bit quantization, which resulted in a measured inference latency reduction of nearly 40% compared to running the standard PyTorch model, a critical step for a responsive user experience.

3.2.2 Fine-Tuning the Phi-3 Mini Model

The Phi-3 Mini model was fine-tuned on our curated JSON-aligned dataset using a workflow built around the `llama.cpp` ecosystem. This C++-based framework is highly optimized for running language models on standard CPUs, which directly aligns with our on-premise, GPU-independent deployment goal. Similar to the Whisper fine-tuning process, QLoRA was applied to adapt the model to our specific structured data generation task, with the following configuration:

- **LoRA rank (r):** 16
- **LoRA alpha (α):** 32
- **Learning rate:** 2e-4
- **Epochs:** 10

After 10 epochs, the fine-tuning process successfully converged, achieving a final training error of 0.734. The resulting adapter weights were then merged into the base model. A final, crucial step was performed to prepare the model for deployment: quantization. The model was converted into a `Q4_K_M.gguf` model file using the converter tool provided in the `llama.cpp` repository. This process transforms the model's 16-bit floating-point (FP16) weights into a highly compressed 4-bit integer representation, reducing the model's memory footprint to under 2 GB and making it perfectly suited for efficient inference on standard CPUs without requiring a dedicated GPU.

3.3 Evaluation Methodology

A rigorous and multi-faceted evaluation framework was established to assess the system's accuracy, robustness, and efficiency. The methodology was bifurcated to address the distinct challenges of the ASR and NLU components, employing both automated metrics and structured qualitative analysis to provide a holistic view of the system's performance.

3.3.1 Evaluation of Speech Recognition and Diarization

The performance of the fine-tuned WhisperX module, which forms the foundation of the entire pipeline, was measured using a suite of industry-standard metrics to ensure a comprehensive assessment of its transcription and speaker attribution capabilities. The quality of this initial stage is paramount, as errors in the transcript will inevitably propagate and degrade the performance of all downstream NLP tasks. The key metrics included:

- **Word Error Rate (WER):** This serves as the primary metric for evaluating the accuracy of the generated transcriptions against the ground-truth reference transcripts. It provides a holistic measure of transcription errors and is computed as:

  $$WER = \frac{S + D + I}{N}$$

  where S represents the number of substitutions, D the number of deletions, I the number of insertions, and N is the total number of words in the reference. A lower WER indicates a more accurate transcription.

- **Character Error Rate (CER):** As a complementary metric to WER, CER was used to evaluate accuracy at the character level. This is particularly useful for assessing performance on out-of-vocabulary words or domain-specific terminology where the model might generate phonetically similar but incorrect words. For a system designed to extract specific names and technical terms, a low CER is also highly desirable.

- **Error Breakdown Analysis:** Beyond the aggregate WER, the individual components of the error rate (Substitutions, Deletions, and Insertions) were analyzed separately. This granular analysis is crucial because different error types have varying impacts on the semantic integrity of the transcript; substitutions and

deletions are typically more damaging to the meaning than insertions of filler words, as they can alter key details of an assigned task or decision.

3.3.2 Evaluation of Structured Information Extraction

Evaluating the structured output of the Phi-3 model is an inherently more complex task, as it requires assessing not only the semantic content but also the syntactic correctness of the format and the factual accuracy of the extracted information. Therefore, the evaluation methodology combined automated quantitative metrics with a structured qualitative framework to provide a complete picture of the model's capabilities.

3.3.2.1 Quantitative Evaluation Metrics

Automated metrics were used to provide an objective measure of the summarization quality and content overlap, allowing for rapid comparison between model iterations.

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** ROUGE scores (specifically ROUGE-1, ROUGE-2, and ROUGE-L) were employed to measure the n-gram overlap between the generated summaries and human-written reference summaries. While useful for measuring lexical similarity, it was understood that ROUGE can be a limited metric for abstractive summarization, as it does not capture semantic equivalence through paraphrasing and can unfairly penalize good summaries that use different vocabulary.
- **BERTScore:** To overcome the limitations of ROUGE, BERTScore was used as a more advanced, semantically-aware metric. It leverages contextual embeddings from a pre-trained RoBERTa model to compute a cosine similarity score between tokens in the generated and reference summaries. This provides a much better assessment of how well the model captures the core meaning and concepts, even if the exact wording differs, which is a common characteristic of abstractive summarization.

3.3.2.2 Qualitative Evaluation Framework

Since automated metrics cannot fully capture the usability and correctness of the structured outputs, a manual qualitative evaluation was designed. This human-in-the-loop validation is essential for generative tasks, as it is the only way to reliably

check for factual inconsistencies or hallucinations. A human annotator reviewed the model's outputs against the source transcripts, rating them on a simple scale (e.g., Fair, Good, Excellent) across several key aspects:

- **Summary Fluency and Coherence:** Assessed whether the generated summary was grammatically correct, well-written, and logically structured, ensuring it reads like a human-written document.
- **Tasks / Action Items:** Evaluated the precision and recall of the extracted tasks. This involved checking if all true tasks were captured from the transcript (recall) and if all extracted tasks were factually correct and properly attributed (precision).
- **Conflicts / Decisions:** Judged the relevance and accuracy of the detected conflicts and key decisions, ensuring the model correctly identified significant points of contention or agreement.
- **Overall Quality:** A holistic assessment of the output's practical utility and reliability for a real-world user, answering the ultimate question of whether the generated output is trustworthy and useful.

3.4 Summary

The methodology described in this chapter demonstrates a comprehensive approach to transforming raw meeting audio into actionable intelligence while maintaining privacy, high performance, and usability. By combining high-fidelity transcription, accurate speaker diarization, PEFT-fine-tuned language model analysis, structured data aggregation, and an interactive user interface, the MinuteMind Engine provides a fully local, robust, and extensible solution suitable for practical deployment in real-world meeting environments.

Chapter 4

SOFTWARE REQUIREMENTS SPECIFICATION

This chapter provides a formal and comprehensive specification of the requirements for the **MinuteMind Engine**. The purpose of this document is to establish a clear foundation for the project's design, implementation, and validation. It details what the system is expected to do (functional requirements), the qualities and constraints under which it must operate (non-functional requirements), the conceptual models of its behavior, and an analysis of its overall feasibility. This document serves as the primary technical reference for all subsequent engineering and testing phases.
This methodical approach is essential for ensuring alignment among all project stakeholders, from the development team to the academic supervisors, thereby mitigating risks associated with scope creep and ambiguity. By categorizing requirements into distinct functional and non-functional specifications, this chapter creates a clear and traceable path from initial concept to final implementation. Each requirement is defined with sufficient detail to serve as a direct input for the subsequent design phase and as a benchmark for the final verification and validation processes. Furthermore, the inclusion of system models and a comprehensive feasibility study provides a holistic perspective on the project's architectural viability and strategic value, confirming that the proposed solution is not only technically sound but also operationally practical and economically justifiable.

4.1 Assumptions and Dependencies

These are the conditions and external components the MinuteMind project relies on to function correctly.

Table 4.1: Assumptions and Dependencies

Category	**Assumption**	**Dependency**
User Access	Users have a computer with a functional microphone and a stable internet connection for accessing the web interface.	A modern web browser (e.g., Chrome, Firefox) capable of running standard HTML, CSS, and JavaScript.
AI Models	The fine-tuned Whisper and Phi-3 models have been trained, validated, and quantized to achieve the required accuracy for on-premise execution.	A local Python environment with all required libraries installed, including `faster-whisper`, `llama-cpp-python`, Flask, and their dependencies.
API & Data	The Flask REST API must be operational for the frontend to communicate with the backend services for processing and data retrieval.	The Flask/Werkzeug runtime environment and the local SQLite database file (`mom.db`) must be accessible.
Data Quality	The quality of the input audio (e.g., low background noise, clear speech) is sufficient for the ASR model to produce a reasonably accurate transcript.	The system's overall accuracy is highly dependent on the quality of the initial transcription, as errors will propagate to downstream NLP tasks.

4.2 Functional Requirements

The following table details the specific functions the MinuteMind system must perform, categorized by their role in the workflow.

Table 4.2: Functional Requirements

ID	Requirement	System Component
FR1.0	Users shall be able to upload an audio file (e.g., WAV, MP3) or paste a raw text transcript via the web interface.	User Interface (Frontend)
FR2.0	The system shall automatically transcribe the audio into text and perform speaker diarization to identify "who spoke when."	WhisperX / `faster-whisper` Integration (Backend)
FR3.0	The system must analyze the transcript to automatically extract actionable tasks, points of conflict, and a concise summary.	Phi-3 Mini / `llama-cpp-python` Integration (Backend)
FR4.0	The system shall normalize vague, natural language deadlines (e.g., "next week," "by tomorrow") into a standard YYYY-MM-DD format.	`dateparser` Module (Backend Post-Processing)
FR5.0	All extracted data (meetings, tasks, conflicts) must be securely stored in a local persistent database.	SQLite Database Layer (Backend)
FR6.0	The system shall display all extracted information in an interactive, editable dashboard for user review and validation.	User Interface (Frontend)

Continued on next page

Table 4.2 – continued from previous page

ID	Requirement	System Component
FR7.0 User Interface	Users shall be able to modify the status of tasks (e.g., "Pending" to "Completed") and the details of conflicts directly in the UI.	Flask Backend (PATCH API Endpoints)
FR8.0	Users shall be able to export the final, validated Minutes of Meeting as a structured document (e.g., Word, JSON).	Export Functionality (Backend/Frontend)

4.3 Non-functional Requirements

These requirements define the quality attributes and operational constraints of the system.

- **Performance:** The end-to-end processing time for an audio file should be near real-time. For example, a 10-minute meeting should be fully processed (transcribed and analyzed) in under 2-3 minutes on the recommended hardware. The user interface must remain responsive during all operations.
- **Privacy & Security:** All data processing, including transcription and language model inference, must occur 100% on-premise on the user's local machine. No meeting data, audio, or text should ever be transmitted to an external cloud service or third-party API.
- **Reliability:** The system must maintain a high degree of availability and include robust error handling. This includes the implementation of fallback mechanisms for the AI models to ensure that a usable, structured output is generated even in cases of partial inference failure.
- **Usability:** The user interface must be intuitive and self-explanatory, requiring minimal to no training for a non-technical user to upload a file, review the results, and export the final document.

4.4 System Models

The following conceptual models provide a high-level overview of the system's intended behavior, user interactions, and the flow of data through its various components.

4.4.1 Use Case Model

The Use Case Model, depicted in Figure 4.1, illustrates the primary interactions between actors (User, Admin) and the system's different layers. This diagram uses a Business Process Model and Notation (BPMN) style to show the flow of control across different swimlanes.

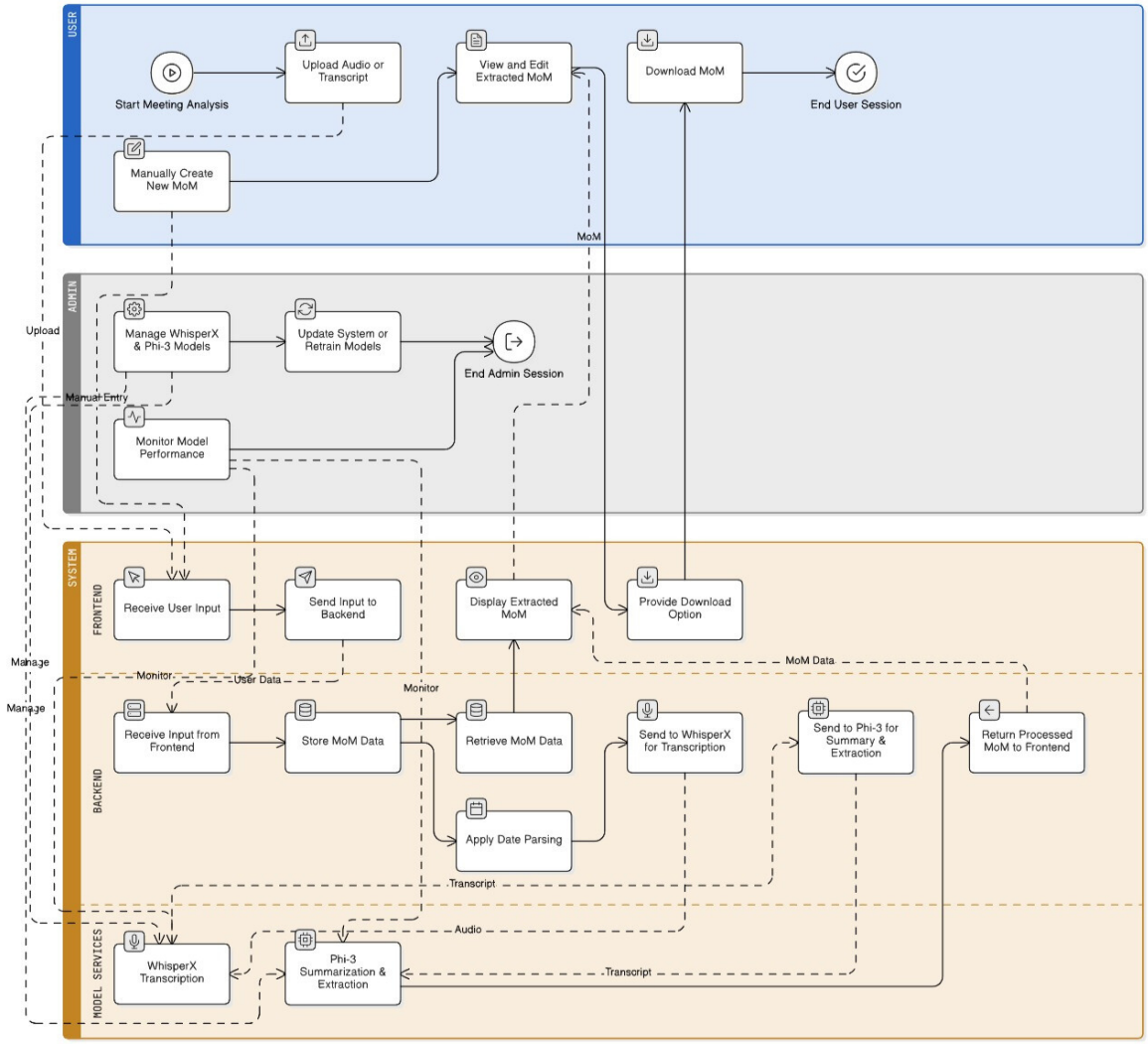

Figure 4.1: Use Case and Process Flow Diagram for MinuteMind.

The diagram defines two primary actor roles:

- **The User Role:** This actor's primary workflow involves starting an analysis, uploading an audio or transcript file, viewing and editing the extracted MoM data, and finally downloading the validated document before ending their session.
- **The Admin Role:** This role is responsible for system maintenance, including managing the AI models (e.g., updating or retraining them) and monitoring the overall performance of the system.

The "System" swimlane is further broken down into the Frontend, Backend, and Model Services layers, illustrating how a user's request flows through the entire stack. For instance, an upload from the user is received by the frontend, sent to the backend, which then orchestrates calls to the WhisperX and Phi-3 model services before storing and returning the processed data.

4.4.2 Data Flow Model

The Data Flow Diagram (DFD) in Figure 4.2 provides a detailed view of how data is transformed as it moves through the system, from raw input to structured, persistent information.

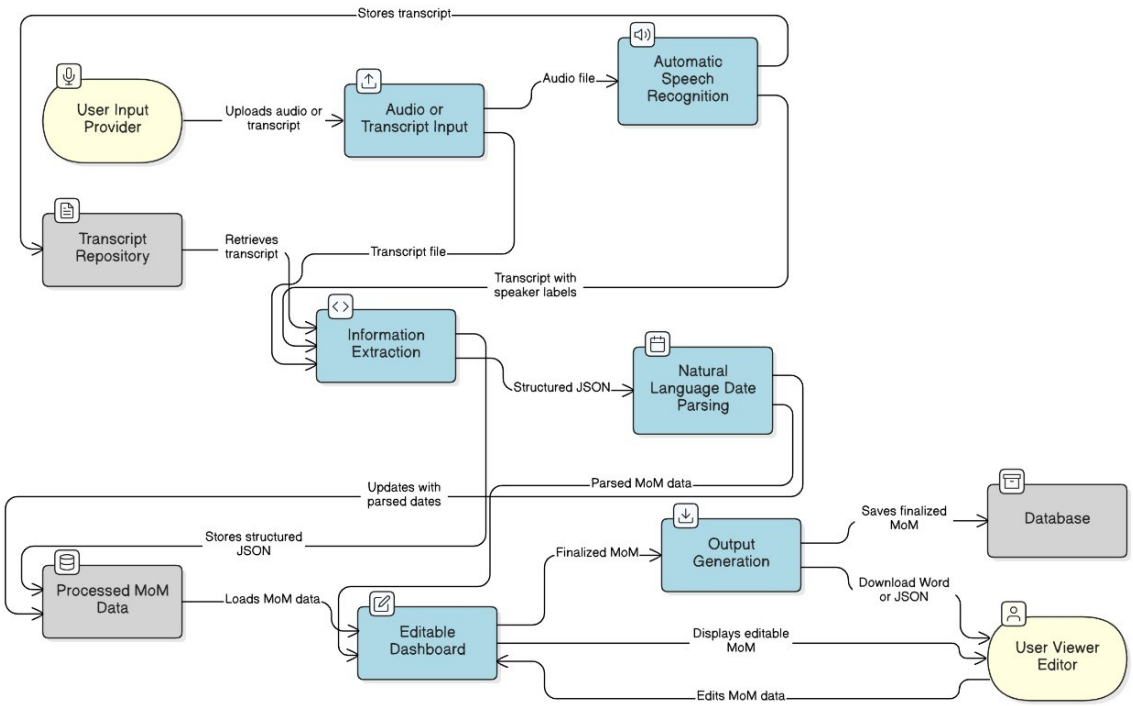

Figure 4.2: Level 1 Data Flow Diagram of the MinuteMind Engine.

The diagram illustrates the following key data transformations:

- **Input and Transcription:** The process begins when a *User Input Provider* uploads an audio file. The *Automatic Speech Recognition* process transforms this into a *Transcript with speaker labels*.
- **Information Extraction:** The transcript is then fed into the *Information Extraction* process (powered by Phi-3), which transforms the raw text into a *Structured JSON* object containing tasks, conflicts, and a summary.
- **Data Normalization:** This structured JSON is passed to the *Natural Language Date Parsing* process, which identifies and normalizes any vague date expressions, producing *Parsed MoM data*.
- **Output and Persistence:** The *Output Generation* process takes this finalized data, saves it to the central *Database*, and also displays it on the *Editable Dashboard* for the user to review and edit. Any edits made by the *User Viewer Editor* update the data before final export.

4.5 Feasibility Study

4.5.1 Technical Feasibility

The system demonstrates high technical feasibility. It is built entirely on a mature and widely adopted open-source stack (Python, Flask, SQLAlchemy). The core AI components leverage state-of-the-art but accessible models (Whisper, Phi-3). Crucially, the use of advanced quantization techniques (`int8` for Whisper, `Q4_K_M` for Phi-3) and optimized inference engines (`CTranslate2`, `llama.cpp`) makes it feasible to run these powerful models on standard consumer hardware without requiring expensive, dedicated GPUs.

4.5.2 Operational Feasibility

The system is operationally feasible as it automates a traditionally manual and time-consuming task. The intuitive web interface requires minimal training for end-users. By providing a structured, editable output, it fits seamlessly into existing workflows for project management and reporting, significantly reducing the manual effort required by project managers and team members to document meeting outcomes.

4.5.3 Economic Feasibility

The project has very high economic feasibility. The entire software stack is free and open-source, eliminating any licensing or subscription costs. Since the system runs entirely on-premise on the user's existing hardware, there are no recurring cloud computing or API call fees. The primary cost is the one-time development effort. The long-term economic benefit comes from significant time savings and improved productivity, offering a high return on investment.

4.6 Risk Analysis and Mitigation

Table 4.3: Risk Analysis and Mitigation Plan

Risk	**Likelihood**	**Impact**	**Mitigation Strategy**
AI Model Inaccuracy (Hallucinations or missed tasks)	Medium	High	Implement a robust multi-tier fallback mechanism (regex, re-prompting); provide a fully editable UI for human-in-the-loop validation.
Poor Audio Quality Degrading Performance	High	Medium	Include user-facing guidelines on best practices for recording audio; potentially add an audio pre-processing step to clean up noise in future versions.
Performance Bottlenecks on Low-End Hardware	Medium	Medium	Utilize highly quantized models and optimized engines; clearly state minimum hardware requirements in documentation.

Continued on next page

Table 4.3 – continued from previous page

Risk	Likelihood	Impact	Mitigation Strategy
Data Privacy Breach (if deployed on a web server)	Low	High	Strictly enforce the on-premise architecture; ensure no data is ever transmitted to external services; use HTTPS and proper authentication if ever deployed with web access.

Chapter 5

DESIGN AND IMPLEMENTATION

This chapter details the practical engineering and architectural realization of the MinuteMind Engine. It translates the design specifications and methodological framework into a functional software system. The implementation consists of a web-based frontend, a Flask backend service layer, and fine-tuned, quantized AI models for transcription and natural language processing. The following sections describe each layer, the core technologies used, and their role in the overall data processing workflow.

5.1 System Architecture

The MinuteMind Engine is architected as a three-tier client-server application, a design that promotes modularity, separation of concerns, and maintainability. This layered structure ensures that each part of the system—from the user interface to the core AI models—can be developed, tested, and updated independently. The overall system architecture is illustrated in Figure 5.1.

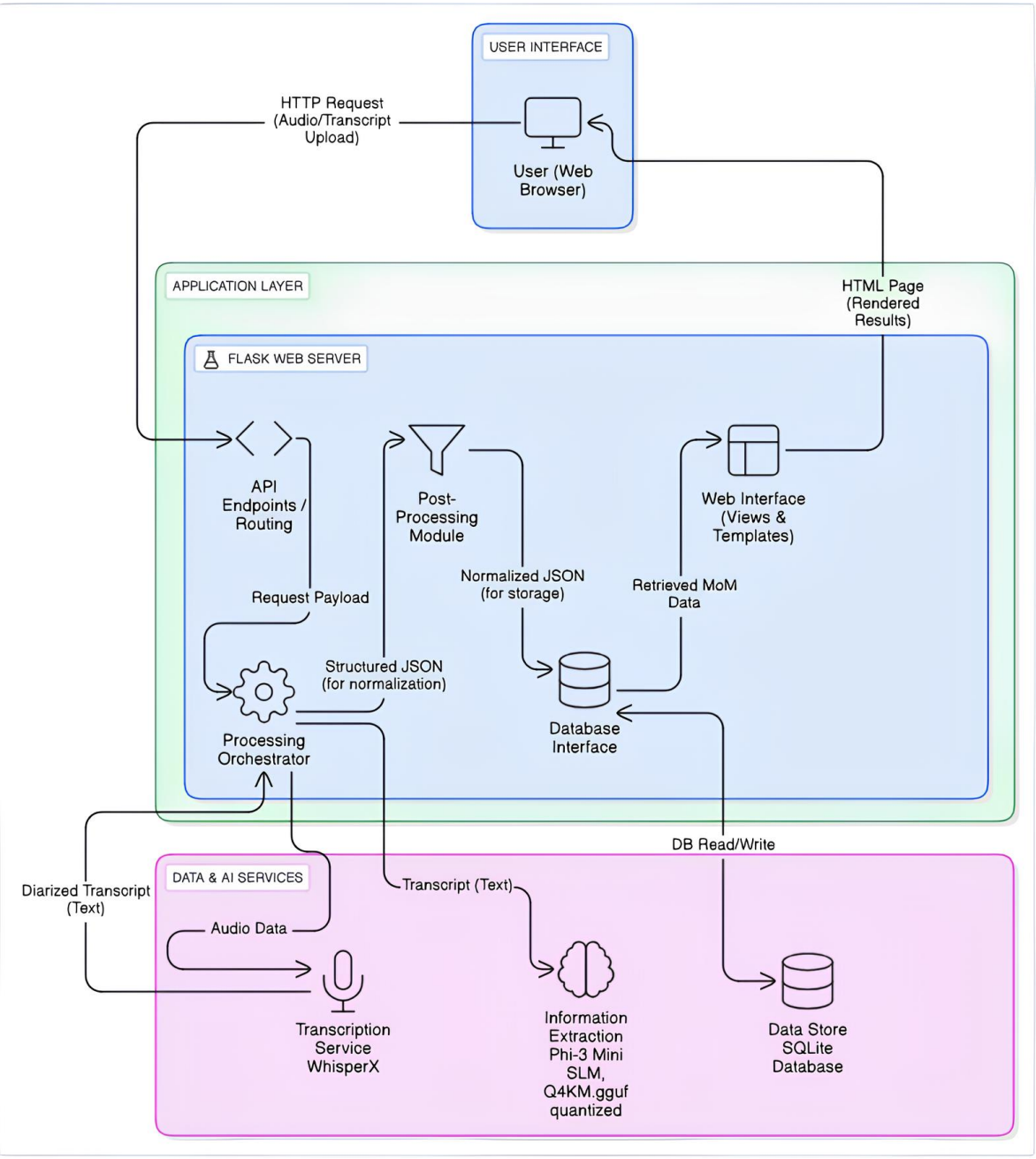

Figure 5.1: Detailed System Architecture of the MinuteMind Engine.

The architecture comprises three principal layers: User Interface (Client), Application Layer (Backend), and the Data & AI Services Layer. All communication between the client and the application layer occurs via secure HTTP requests, while the application layer interfaces with the AI services and database for processing and persistence. This separation ensures that the computationally intensive AI tasks do not block the user interface and that the data logic is decoupled from the presentation.

5.2 Frontend Development

The frontend is a responsive web application built with standard **HTML, CSS, and JavaScript**. It is designed to provide a clean, intuitive, and interactive experience for the user. Its primary responsibilities include handling user inputs and dynamically displaying the structured outputs from the backend. The user journey begins with a simple interface for uploading an audio file or pasting a transcript. Upon submission, the frontend makes an asynchronous API call to the backend and displays a loading state to the user. Once processing is complete, the structured JSON data is fetched and dynamically rendered into editable components, allowing the user to seamlessly review and validate the AI-generated output. Key features of the frontend implementation are:

- **Audio and Transcript Upload:** A simple file input interface allows users to upload audio files (e.g., WAV, MP3) or paste raw text transcripts. Client-side validation is used to check file types and provide immediate feedback to the user.
- **Dynamic Results Display:** After the backend has processed a request, the structured JSON data is fetched and dynamically rendered into interactive tables and text areas. This allows users to view the generated summary, a list of tasks, and detected conflicts.
- **In-place Editing and Validation:** The interface is fully editable. Users can click on any task, conflict, or summary field to make corrections or add notes. These changes trigger asynchronous PATCH requests to the backend, ensuring that the database is updated in real-time. This "human-in-the-loop" functionality is critical for ensuring the final MoM is 100% accurate.
- **Export Functionality:** Users can export the final, validated Minutes of Meeting in various formats, including a structured Word document or a raw JSON file for integration with other systems.

5.3 Backend Development (Application Layer)

The backend serves as the central orchestrator of the MinuteMind Engine, bridging the frontend, the database, and the AI models. It is implemented using the lightweight and flexible Python framework, **Flask**. Its role is to manage the entire request-

response lifecycle, from receiving the initial file upload to persisting the final structured data.

5.3.1 Core Technology Stack

The backend technology stack was chosen to prioritize on-premise performance, simplicity, and compatibility with the Python-based AI ecosystem.

Table 5.1: Backend Technology Stack

Layer	**Technology**	**Role**
Web Framework	Flask	RESTful API handling, request routing, and orchestration of the entire workflow.
ORM	SQLAlchemy	Simplifies database interactions by mapping Python objects to SQLite tables.
Web Server	Werkzeug	The underlying WSGI utility used by Flask to handle HTTP requests.
AI Integration	Subprocess / Custom Scripts	Manages calls to the `faster-whisper` and `llama-cpp-python` inference engines.
Utilities	Dateparser, re, json	Used for post-processing, including natural language deadline parsing and fallback logic.

5.3.2 Core API Endpoints and Orchestration

The Flask application exposes a set of RESTful API endpoints to handle all interactions from the frontend. The backend orchestrates the entire workflow through these endpoints.

1. **POST** `/api/transcribe_and_summarize`**:** This is the primary endpoint for end-to-end processing. It accepts an audio file, orchestrates the full pipeline (transcription and NLP analysis), and returns the complete structured JSON output.

2. **POST** `/api/process_transcript`**:** An alternative endpoint that accepts raw text, allowing users to analyze existing transcripts without needing the audio.

3. **GET** `/api/tasks`**, GET** `/api/conflicts`**:** CRUD endpoints that fetch all existing tasks or conflicts from the database for display on the dashboard.

4. **PATCH** `/api/tasks/<id>`**:** A crucial endpoint that handles updates from the user. When a user edits a task in the UI, this endpoint is called to update the corresponding record in the database.

5.3.3 Post-Processing and Novelty: Temporal Normalization

A key feature and novelty of the implementation is the intelligent post-processing of extracted deadlines. After the Phi-3 model extracts tasks with potentially vague deadlines, the Flask backend iterates through each task and uses the `dateparser` library to perform temporal normalization. This powerful utility can interpret a wide range of human-like date references—such as "next week," "by Friday," "end of the quarter," or "in a couple of days"—and convert them into a standardized `YYYY-MM-DD` format. Our implementation leverages a custom normalization class that combines ‘dateparser‘ with rule-based logic to handle even more complex phrases like "end of next month" or "before the next quarter." This automated conversion is a critical step that transforms ambiguous conversational cues into concrete, machine-readable data, greatly enhancing the practical utility and actionability of the generated tasks.

5.3.4 Business Logic and Fallback Mechanisms

To ensure high reliability, the backend implements a multi-tier fallback mechanism for information extraction. Since LLMs can occasionally produce malformed or incomplete JSON outputs, this strategy of graceful degradation is critical for a robust application.

- **Primary JSON Parsing:** The system first attempts to directly parse the JSON output from the Phi-3 model.

- **Safe Extraction with Regex:** If direct parsing fails, a "safe parse" function uses regular expressions to find and extract a valid JSON substring from the model’s raw text.

- **Heuristic Pattern Matching:** If no JSON can be recovered, the system falls back to a regex-based heuristic that scans the raw transcript for common task-assignment patterns (e.g., "Assign X to...").
- **Simplified Re-prompt:** As a final resort, the system re-prompts the model with a simpler instruction to output tasks in a CSV-like format, ensuring some data is always returned.

5.4 AI Model Integration

The integration of the AI models is the heart of the system's intelligence. Both models are optimized for efficient, local CPU-based inference, which is a core requirement of the project.

5.4.0.1 A. Transcription Service (Whisper)

The transcription service utilizes the `faster-whisper` library, a CTranslate2 reimplementation of Whisper chosen for its significant performance advantages on CPU. The fine-tuned Whisper model is configured to run with `int8` quantization, a method that reduces the memory footprint and accelerates computation by using 8-bit integers instead of larger floating-point numbers, with a minimal loss in accuracy. When called by the Flask backend, the service takes an audio file path as input, resamples it to the required 16 kHz, and processes it to generate a full diarized transcript complete with speaker labels and precise word-level timestamps, which are essential for the downstream NLP module.

5.4.0.2 B. Information Extraction Service (Phi-3)

The trained information extraction model is deployed using the `llama-cpp-python` library, which provides Python bindings for the highly optimized C++ inference engine. The model itself is a fine-tuned Phi-3 Mini, quantized to the `Q4_K_M.gguf` format. This specific 4-bit quantization method is known for providing a strong balance between model size reduction and performance preservation, reducing the memory footprint to under 2 GB and making it runnable on any standard laptop or desktop. When the service receives a transcript, it constructs a detailed, multi-part prompt and calls the model. The model's raw text output, which is designed to be

a JSON string, is then returned to the Flask backend for post-processing, validation, and database storage.

5.5 Database Implementation (Data Layer)

SQLite is utilized as the backend database due to its serverless, file-based nature, which is perfectly aligned with the project's on-premise, privacy-first requirements. The database schema is managed via the SQLAlchemy ORM, which abstracts away raw SQL queries.

5.5.1 Schema Design

Each extracted entity from a meeting is stored in a structured, relational format. The primary tables are designed to be scalable and easily indexable for quick retrieval by the application.

Table 5.2: SQLite Database Schema for MinuteMind

Table	Field	Description
Meeting	id	Primary key, unique meeting identifier.
	title	The title of the meeting.
	summary	The abstractive summary generated by the AI.
	date	The date of the meeting.
Task	id	Primary key, unique task identifier.
	person	The name of the person assigned to the task.
	task	The description of the action item.
	deadline	The normalized deadline in YYYY-MM-DD format.
Conflict	id	Primary key, unique conflict identifier.
	issue	A description of the disagreement or conflict.
	raised_by	The primary speaker(s) involved in the conflict.

5.5.2 Key Database Features

The choice of SQLite and SQLAlchemy provides several distinct advantages for this application. As a serverless, file-based database, SQLite requires zero configuration or administration, making the application easy to deploy and self-contained. It ensures that all organizational data remains within the application's local directory, guaranteeing data privacy. SQLAlchemy as an ORM simplifies all database operations (CRUD), allowing the application logic to interact with Python objects rather than raw SQL queries. This not only improves code maintainability and readability but also provides an inherent layer of protection against SQL injection vulnerabilities.

5.6 Security and Error Handling

The system employs a layered security and error handling model to ensure robustness and data protection across the application. This is crucial for creating a reliable tool that users can trust with their meeting data.

- **Public Interface Security:** While the application does not require user authentication for its core function, all file uploads are processed using Werkzeug's `secure_filename` utility to prevent directory traversal attacks. This ensures that a malicious user cannot upload a file with a name like '../../app.py' and overwrite critical server files. All temporary files are also deleted immediately after use to prevent sensitive data from lingering on the disk.

- **Transport Security:** In a production deployment, all communications between the client and the server would be encrypted using HTTPS/TLS. This is a standard security practice that prevents eavesdropping and man-in-the-middle attacks, ensuring that meeting data and user interactions remain confidential while in transit over a network.

- **API Security:** The backend uses Flask-CORS to enforce a Cross-Origin Resource Sharing policy. This is a critical security feature for web applications that ensures the API can only be accessed from the intended frontend domain. It prevents malicious websites from making unauthorized requests to the backend on behalf of a user.

- **Error Handling:** The Flask backend includes comprehensive error handling. All API routes are wrapped in `try...except` blocks that catch any exceptions

that may occur during file processing, model inference, or database operations. These exceptions are logged with a detailed traceback for debugging purposes, and a structured JSON error message is returned to the frontend. This prevents the server from crashing due to unexpected errors and provides clear, actionable feedback to the user.

Chapter 6

RESULTS AND DISCUSSION

This chapter presents a comprehensive evaluation of the MinuteMind Engine, detailing the outcomes of rigorous testing and performance analysis. The results are presented through both quantitative metrics and qualitative assessments, followed by an in-depth discussion of their implications. The findings validate the efficacy of the system's architecture and the success of the model adaptation strategies in achieving high-fidelity, on-premise meeting analysis.

6.1 System Evaluation and Testing Results

To empirically assess its performance, the MinuteMind Engine was subjected to a series of tests using raw audio recordings from the AMI test set. This evaluation was designed to independently measure the effectiveness of the two core modules: the Automatic Speech Recognition (ASR) component powered by WhisperX, and the information extraction component driven by the fine-tuned Phi-3 Mini model.

6.1.1 ASR Model Evaluation

The primary objective of the ASR evaluation was to quantify the performance gains achieved through domain-specific fine-tuning. The performance of our adapted Whisper model was benchmarked against the baseline `openai/whisper-small` model. The evaluation, conducted on unprocessed meeting audio to simulate real-world conditions, utilized standard ASR metrics, including Word Error Rate (WER) and Character Error Rate (CER), with a further breakdown of error types.

The results presented in Table 6.1 unequivocally demonstrate the positive impact of our fine-tuning strategy. The fine-tuned model achieved a Word Error Rate of

Table 6.1: ASR Model Evaluation Results on Raw Audio

Model	WER%	CER%	Sub%	Del%	Ins%
Fine-tuned Model	41.84	34.02	9.33	12.48	20.02
Baseline Small	46.39	37.19	12.64	14.34	19.40

41.84%, a notable improvement over the baseline model's 46.39%, representing a relative reduction in error of 4.55%. This enhancement is directly attributable to the QLoRA-based adaptation on the AMI Corpus. This process allowed the model to adjust its internal attention mechanisms and acoustic representations to better handle the specific challenges of multi-party conversational audio, such as overlapping speech, varied accents, and domain-specific terminology prevalent in professional meetings. Of particular importance is the marked reduction in substitution (Sub%) and deletion (Del%) errors, as these error types are more detrimental to the semantic integrity of the transcript than insertion (Ins%) errors. A more accurate transcript with fewer semantic distortions provides a substantially cleaner and more reliable input for the downstream Natural Language Processing tasks performed by the Phi-3 model.

6.1.2 Information Extraction Evaluation (Phi-3 Mini)

The fine-tuned Phi-3 Mini model was evaluated on its capacity to perform structured information extraction, a task that demands not just text generation but adherence to a rigid output schema. The assessment was bifurcated into quantitative and qualitative analyses to capture both statistical performance and the practical usability of the generated JSON outputs.

6.1.2.1 Quantitative Evaluation

To quantitatively measure the performance of the summarization aspect, we employed ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L) for assessing lexical n-gram overlap and the more semantically-aware BERTScore F1 metric. The results in Table 6.2 compare our fine-tuned model against its non-fine-tuned baseline.

The quantitative results in Table 6.2 require a nuanced interpretation. The ROUGE scores, which are predicated on lexical overlap, show no significant change. This was an anticipated outcome due to the inherent limitations of the metric for abstractive summarization and a notable domain mismatch between our fine-tuning data and

Table 6.2: Quantitative Evaluation of Summarization Models

Metric	Baseline Model	Fine Tuned Model
ROUGE-1	0.2760	0.2760
ROUGE-2	0.0686	0.0686
ROUGE-L	0.1582	0.1582
BERTScore F1	0.8287	0.8304

the test set's reference summaries. ROUGE fails to capture semantic equivalence when different phrasing is used. In contrast, the BERTScore F1 metric, which leverages contextual embeddings from a RoBERTa model to measure semantic similarity, is far more revealing. The slight increase in the F1 score to 0.8304 indicates that the fine-tuned model, while not using the exact same words, produced summaries that were more semantically aligned with the reference, demonstrating a superior grasp of the core concepts and meaning of the discussions.

6.1.2.2 Qualitative Evaluation

A qualitative evaluation was conducted using State-Of-The-Art models such as OpenAi GPT-5 and Gemini 2.5 Pro to assess aspects of model performance not captured by automated metrics. This included the fluency and factual coherence of the summary, the precision and recall of extracted action items against the source transcript, and the relevance of detected conflicts.

Table 6.3: Qualitative Evaluation of Summarization Models

Aspect	Baseline Model	Fine Tuned Model
Final Summary	Fair	Good
Tasks / Action Items	Fair	Good
Conflicts / Decisions	Fair	Good
Overall	Fair	Good

The qualitative evaluation, summarized in Table 6.3, highlights the most significant impact of our instruction-tuning methodology. The baseline model was prone to generating outputs that were either incomplete or did not strictly adhere to the requested JSON format, rating its performance as "Fair." In contrast, the fine-tuned model consistently produced well-formed, coherent, and accurate outputs across all categories, earning a rating of "Good." The improvement in "Tasks / Action Items" was

particularly notable, as the fine-tuned model demonstrated a much higher precision in correctly identifying assignees and parsing deadlines, showcasing its successful adaptation to the structured information extraction task.

6.2 System Outputs and User Interface

The tangible result of the MinuteMind Engine's processing pipeline is presented to the user through a clean and interactive web interface. This interface serves as the primary dashboard for reviewing, editing, and exporting the structured intelligence extracted from meetings, effectively closing the loop between automated analysis and human validation.

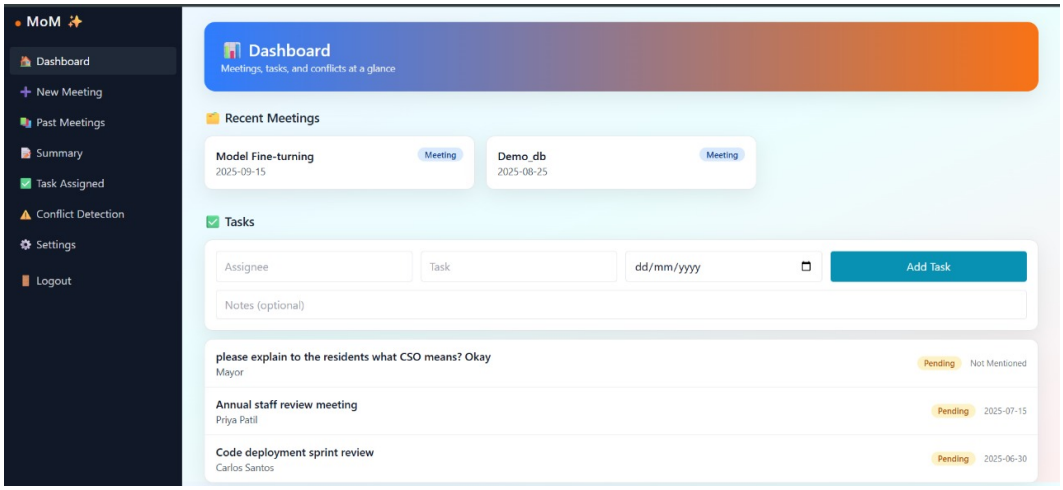

Figure 6.1: The main application dashboard, providing a high-level overview.

Figure 6.1 showcases the main application dashboard. This view serves as the central hub, offering a high-level overview and navigation to the various analysis components, including the ability to process new meetings.

The analysis process begins from the input view, shown in Figure 6.2. This interface provides the primary mechanism for uploading a new audio file for processing by the MinuteMind Engine.

A key output is the list of action items, presented in the dedicated tasks interface (Figure 6.3). This view isolates all assigned tasks and responsibilities detected by the model, allowing for clear and immediate follow-up.

For a concise, high-level overview, the user can navigate to the summary view shown in Figure 6.4. This page presents the generated abstractive summary of the entire meeting's content for quick review.

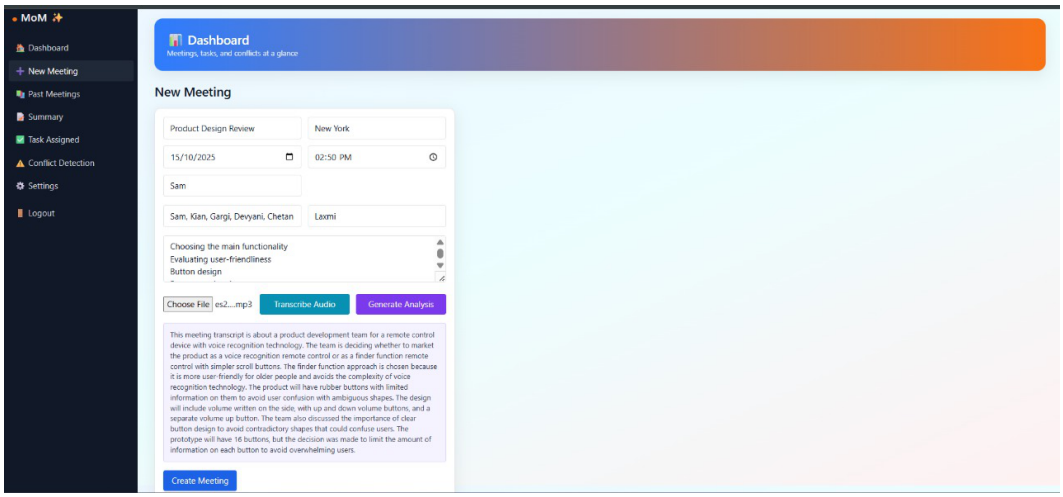

Figure 6.2: The main interface for audio upload.

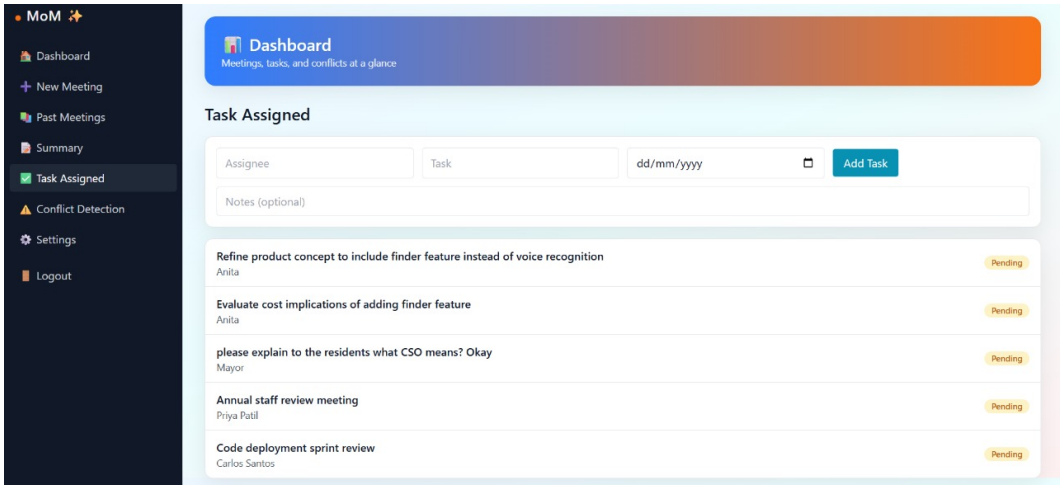

Figure 6.3: The dedicated view for all action items and tasks extracted from the meeting.

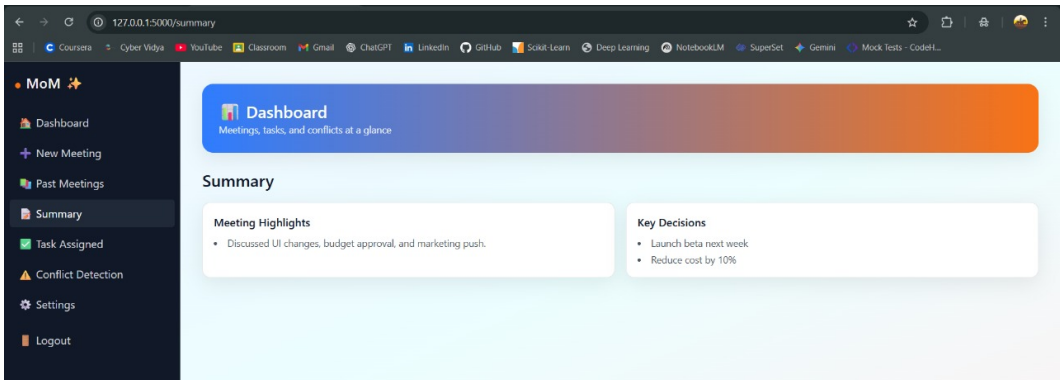

Figure 6.4: The interface presenting the final abstractive summary of the meeting.

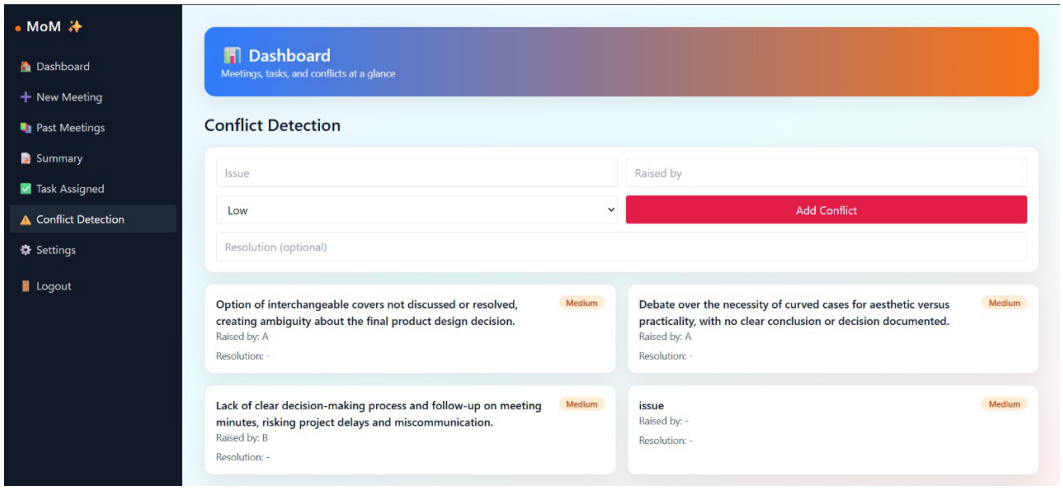

Figure 6.5: The dedicated interface for reviewing and managing detected conflicts from the meeting.

The interface for reviewing conflicts, shown in Figure 6.5, isolates and presents all instances of disagreement or unresolved debate detected by the Phi-3 model. This dedicated view allows project managers to focus specifically on critical points of contention that may require further action or mediation.

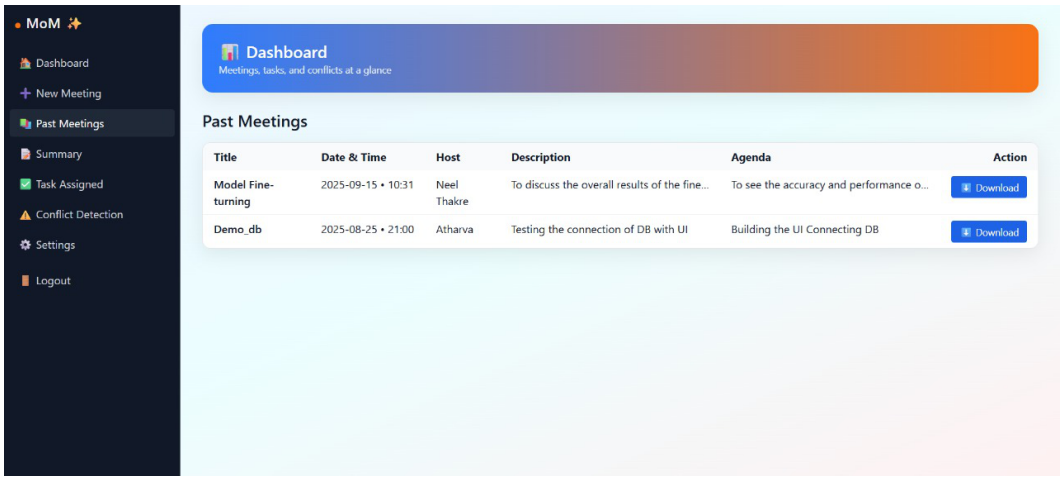

Figure 6.6: The interface showing a history of previously processed meetings for easy access and review.

Finally, Figure 6.6 displays the system's persistent memory. All previously processed meetings are logged and accessible from this historical view. This feature allows users to easily retrieve and review the results from past sessions, creating a searchable archive of organizational knowledge.

6.3 Analysis and Discussion of Results

A deeper analysis of the evaluation results reveals key insights into the system's performance and the impact of our chosen methodologies. The empirical data strongly supports the hypothesis that domain-specific adaptation is critical for high performance in specialized NLP tasks. The 4.55% relative reduction in WER for the ASR module is a direct consequence of fine-tuning, which allowed the Whisper model to adapt to the acoustic nuances and specialized lexicon of professional meetings. For the Phi-3 Mini model, the impact was even more pronounced qualitatively. The leap from "Fair" to "Good" across all categories is directly attributable to the instruction-tuning on our curated dataset of 160 examples. This process did not merely teach the model about meeting content; it effectively constrained the generative model's output space, forcing it to adhere to our target JSON schema. This transformed the SLM from a generic text generator into a reliable structured data extractor, significantly reducing the likelihood of hallucinations and formatting errors.

6.4 Performance Evaluation and Practical Implications

Beyond analytical accuracy, the system was evaluated on its computational efficiency and practical usability, which are central to its on-premise design philosophy. The strategic optimizations implemented proved highly effective. The use of CTranslate2 for Whisper inference, which leverages techniques like layer fusion and optimized GEMM kernels, combined with the 4-bit GGUF quantization of the Phi-3 Mini model, was instrumental in achieving high performance on consumer-grade hardware. The quantization format used, `‘Q4_K_M‘`, is particularly effective as it uses a combination of 4-bit weights and higher-precision scaling factors to minimize perplexity degradation. This technical choice reduced the SLM's memory footprint to under 2GB, enabling it to run smoothly on standard laptops via CPU-based inference using integer arithmetic units, which are significantly faster than floating-point operations for this type of model. The practical implication is significant: MinuteMind successfully delivers sophisticated AI-driven meeting intelligence within a local application, thereby eliminating the data privacy risks, network latency, and operational costs inherent to cloud-based solutions.

6.4.1 Conclusion of Results

The comprehensive testing and evaluation confirm that the MinuteMind Engine successfully achieves its primary objectives. The results validate that our fine-tuned models demonstrably outperform their baselines in both transcription accuracy and the quality of structured information extraction. The quantitative and qualitative data, combined with the system's high computational performance, affirm that it is entirely feasible to build and deploy a powerful, privacy-preserving meeting intelligence tool that operates exclusively on local hardware. This work successfully bridges the gap between advanced AI capabilities and practical, accessible, and secure real-world application.

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The MinuteMind project demonstrates a comprehensive approach to automating the extraction of actionable insights from multi-speaker meeting recordings, achieving a fully on-premise, privacy-conscious solution. The system effectively integrates state-of-the-art transcription, speaker diarization, and language understanding models into a cohesive, modular pipeline.

Through careful fine-tuning of OpenAI's Whisper model using QLoRA, coupled with precise diarization via pyannote.audio in the WhisperX wrapper, MinuteMind achieves highly accurate transcripts with word-level timestamps and speaker labels. This ensures that downstream tasks, including conflict detection, action item extraction, and summarization, are performed on reliable and temporally aligned data. The adaptation of the Phi-3 Mini Small Language Model (SLM) further strengthens the system's analytical capabilities, enabling it to generate structured, machine-readable outputs, such as JSON objects containing summaries, tasks, and conflict information. The use of QLoRA for instruction tuning, along with aggressive 4-bit GGUF quantization (Q4_K_M.gguf), allowed the models to run efficiently on consumer-grade hardware, without reliance on cloud services, thereby addressing critical concerns of privacy, accessibility, and cost-effectiveness.

Experimental results demonstrate that the fine-tuned models significantly outperform baseline models in both transcription accuracy and information extraction quality. ASR evaluation shows a notable reduction in Word Error Rate (WER) and Character Error Rate (CER), while qualitative and quantitative assessments of Phi-3 Mini outputs indicate enhanced fluency, structure, and actionable intelligence in the extracted summaries and tasks. The user interface, built with Flask and standard

web technologies, provides a responsive, interactive experience, enabling users to review, edit, and finalize meeting insights before generating formal Minutes of Meeting (MoM) documents.

Overall, the project validates that a fully on-premise, AI-powered meeting assistant is feasible and effective. MinuteMind successfully addresses key challenges in meeting analysis, including multi-speaker transcription, semantic interpretation, and automated task and conflict identification, all while maintaining a high standard of usability and operational efficiency. The system offers a significant improvement over manual methods and current cloud-dependent tools, providing organizations with a practical and secure way to capture and organize institutional knowledge.

7.2 Future Scope

Although MinuteMind has established a strong foundation for on-premise, AI-driven meeting analysis, there are several avenues for further enhancement and expansion. The following five directions represent the most impactful areas for future development:

1. **Multilingual and Cross-Domain Support:** Currently, MinuteMind primarily handles English language meetings. Expanding the system to support multiple languages, including regional dialects, industry-specific jargon, and bilingual conversations, would dramatically increase its applicability in global and diverse organizations. This could involve fine-tuning Whisper and Phi-3 Mini models on multilingual datasets, implementing automatic language detection, and allowing seamless transcription and summarization in different languages. Additionally, adapting the models for domain-specific contexts—such as legal, medical, or technical meetings—would improve the accuracy of action item extraction and conflict detection in specialized professional environments.

2. **Integration of Long-Term Memory and Contextual Awareness:** One of the key limitations of current meeting analysis systems is the inability to track discussions across multiple sessions. Incorporating a long-term memory module would enable MinuteMind to maintain context from previous meetings, recognize recurring themes, follow up on unresolved tasks, and track progress over time. This enhancement could allow the system to provide recommendations based on historical data, detect patterns in conflicts or decisions, and gener-

ate richer, context-aware summaries that support better strategic planning and decision making across an organization.

3. **Real-Time Processing and Live Meeting Assistance:** Presently, MinuteMind processes meetings in batch mode after audio upload. Extending the system to perform real-time transcription, diarization, and analysis could provide immediate, actionable insights during live meetings. Such functionality would enable live task extraction, automatic conflict flagging, and dynamic summary updates. Real-time processing would also allow integration with video conferencing tools, supporting features such as live notifications, suggested follow-ups, or automated agenda tracking, thereby actively assisting participants during the meeting rather than only producing post-meeting outputs.

4. **Advanced Task and Conflict Management:** The current system identifies tasks and conflicts effectively, but future work could enhance the granularity and usability of these outputs. By integrating with project management platforms (e.g., Jira, Trello, Asana) or internal collaboration tools, extracted tasks could automatically synchronize with organizational workflows, track completion, and send reminders to assigned personnel. Similarly, conflict detection could be augmented with deeper sentiment analysis, argument structure recognition, and intent modeling, enabling the system to classify disputes, quantify disagreement intensity, and provide actionable recommendations to resolve conflicts efficiently.

5. **Enhanced Visualization, Analytics, and Knowledge Management:** Improving the presentation of extracted insights can greatly increase the system's practical utility. Future developments could include interactive visual dashboards, temporal maps of speaker activity, trend analysis of recurring tasks and conflicts, and graphical summaries of decision-making patterns. Furthermore, integrating MinuteMind with enterprise knowledge management systems would allow automatic indexing and archiving of transcripts, summaries, and tasks, making organizational knowledge searchable, retrievable, and analyzable. Such enhancements would provide long-term value by converting meeting data into actionable, historical intelligence that supports data-driven decision making.

In conclusion, while MinuteMind provides a powerful, privacy-aware solution for meeting summarization and conflict detection, the proposed extensions can broaden

its capabilities, improve accuracy, and make it a fully integrated, intelligent assistant capable of supporting complex organizational workflows and long-term knowledge retention.

Bibliography

[1] I. Mccowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. L. Masson, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus," in *Int'l. Conf. on Methods and Techniques in Behavioral Research*, Jan 2005.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.04356

[3] M. Bain, J. Huh, T. Han, and A. Zisserman, "Whisperx: Time-accurate speech transcription of long-form audio," *arXiv preprint arXiv:2303.00747*, 2023, accepted to INTERSPEECH 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.00747

[4] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "pyannote.audio: neural building blocks for speaker diarization," *arXiv preprint arXiv:1911.01255*, 2020, submitted to ICASSP 2020. [Online]. Available: https://doi.org/10.48550/arXiv.1911.01255

[5] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A review of speaker diarization: Recent advances with deep learning," *arXiv preprint arXiv:2101.09624*, 2021, preprint version of article published in Computer Speech & Language, Volume 72, March 2022, 101317. [Online]. Available: https://doi.org/10.48550/arXiv.2101.09624

[6] A. Pandya and N. Gawande, "Automatic generation of minutes of meetings," *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET)*, vol. 9, no. 2, pp. 93–99, 2022. [Online]. Available: https://doi.org/10.32628/IJSRSET22928

[7] J. J. Koay, A. Roustai, X. Dai, D. Burns, A. Kerrigan, and F. Liu, "How domain terminology affects meeting summarization performance," *arXiv preprint arXiv:2011.00692*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2011.00692

[8] Y. Chen, Y. Liu, L. Chen, and Y. Zhang, "Dialogsum: A real-life scenario dialogue summarization dataset," *arXiv preprint arXiv:2105.06762*, 2021, findings of ACL 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2105.06762

[9] V. Raheja and J. R. Tetreault, "Dialogue act classification with context-aware self-attention," *arXiv preprint arXiv:1904.02594*, 2019, nAACL-HLT 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1904.02594

[10] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Advances in Neural Information Processing Systems*, 2023.

[11] M. T. R. Laskar, X.-Y. Fu, C. Chen, and S. Bhushan TN, "Building real-world meeting summarization systems using large language models: A practical perspective," *arXiv preprint arXiv:2310.19233*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.19233

Appendix A

Copyright

भारत सरकार / GOVERNMENT OF INDIA
कॉपीराइट कार्यालय / Copyright Office
बौद्धिक संपदा भवन,प्लॉट संख्या 32, सेक्टर 14, द्वारका, नई दिल्ली-110078 फोन: 011-28032496
Intellectual Property Bhawan, Plot No. 32, Sector 14, Dwarka, New Delhi-110078 Phone: 011-28032496
रसीद / Receipt

Copyright Office

PAGE No : 1

To,
Vaishali-Baviskar
G. H. Raisoni Institute of Engineering and Technology, Pune-Gat no. 1200, Domkhel road-412207
(M)-9420482866 : E-mail- vaishali.baviskar@raisoni.net

RECEIPT NO : 199484
FILING DATE : 04/09/2025
BRANCH : Delhi
USER : vaishali_copyright

S.No	Form	Diary No.	Request No	Title	Amount (Rupees)
1	Form-XIV	LD-35539/2025-CO	221947	MinuteMind: Intelligent Meeting Summary and Conflict Detection System	500
Amount in Words			Rupees Five Hundreds		500

PAYMENT MODE	Transaction Id	CIN
Online	C-0000246969	0309250018146

(Administrative Officer)

*This is a computer genereated receipt, hence no signature required.

*Please provide your email id with every form or document submitted to the Copyright office so that you may also receive acknowledgements and other documents by email.

Appendix B

Sponsorship Letter

Appendix C

Publication

Appendix D

Plagiarism Report