IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

AI Powered Meeting Summarizer and Action Item Generator

Patil Harshal Mahesh[1], Patil Diksha Sudhakar[2], Pardeshi Tushar Rajendra[3], Patil Harshal Dyaneshwar[4], Dr. Ujwala Patil[5]

Student, SES's R. C. Patel Institute of Technology, Shirpur 425405, India[1]

Student, SES's R. C. Patel Institute of Technology, Shirpur 425405, India[2]

Student, SES's R. C. Patel Institute of Technology, Shirpur 425405, India[3]

Student, SES's R. C. Patel Institute of Technology, Shirpur 425405, India[4]

Head Of Department (AIML), SES's R. C. Patel Institute of Technology, Shirpur 425405, India[5]

Abstract: Meetings are an important part of academic and professional work, but the information discussed during them is not always recorded in a consistent or useful form. Preparing minutes manually can take time, and important decisions, responsibilities, deadlines, and risks may be overlooked. This paper presents MeetMind, an AI-powered meeting summarization and action-item management system. The application accepts either a typed meeting transcript or an uploaded audio/video recording. For recordings, it generates a transcript using the OpenAI speech-to-text service when configured, or a local Whisper-based transcription option. The transcript is then processed by an offline natural language processing pipeline that uses spaCy, NLTK, and scikit-learn. TF-IDF sentence ranking supports extractive summaries, while linguistic patterns and named-entity information assist in identifying decisions, tasks, owners, deadlines, priorities, and risks. An optional locally hosted language model through Ollama can provide additional summary and insight generation.

MeetMind stores meeting records in an SQLite database with FTS5 full-text search. The stored information supports a Memory Assistant, an Accountability Center, calendar-event export, analytics, a Meeting Readiness and Dependency Radar, and report generation. The prototype was tested at module level using eight functional test cases, all of which passed. In one sample product-strategy meeting containing 617 words and eight speakers, the system identified 15 action items and eight decisions and produced a productivity score of 86.5 out of 100, displayed as 87%. These observations demonstrate the operation of the prototype; they do not represent validated accuracy measurements. The paper describes the system design, implementation, observed outputs, limitations, and directions for further evaluation.

Keywords: Meeting summarization, action-item extraction, natural language processing, TF-IDF, speech recognition, Whisper, SQLite, full-text search, meeting analytics.

I. INTRODUCTION

Meetings allow teams to exchange information, discuss problems, make decisions, and distribute responsibilities. However, the value of a meeting depends not only on the discussion itself but also on what happens afterwards. When notes are incomplete or responsibilities are not recorded clearly, participants may forget decisions, miss deadlines, or repeat discussions that have already taken place. Preparing detailed minutes manually can also be difficult when meetings are long or involve several participants.

Recent developments in natural language processing and automatic speech recognition provide practical ways to reduce this documentation effort. Speech-recognition systems can convert recorded conversations into text, while text-processing methods can identify important sentences, topics, and entities. Extractive approaches such as TF-IDF sentence ranking select informative parts of a transcript, whereas language models can generate more fluent summaries. These techniques provide useful building blocks, but a meeting-support system also needs to organize decisions and tasks and make them available for later review.

This paper introduces MeetMind, a software prototype developed to support the meeting lifecycle from input to follow-up. The system accepts a transcript or an audio/video file, processes the content, and presents structured meeting information through a Streamlit interface. Its design combines an offline rule- and statistics-based NLP pipeline with optional local language-model enrichment. Meeting records are stored in a searchable SQLite knowledge base, allowing users to revisit earlier discussions and monitor assigned work.

The main contributions of this work are: (1) a workflow that processes typed transcripts and uploaded recordings; (2) a hybrid analysis design that can operate using deterministic NLP methods without depending on an LLM; (3) structured

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151110

extraction of summaries, decisions, action items, and risks; (4) a persistent meeting knowledge base with full-text search and source-linked question answering; and (5) follow-up features, including task tracking, calendar export, analytics, readiness indicators, and report generation.

II. RELATED WORK

Automatic text summarization has been studied using both statistical and learning-based methods. TF-IDF estimates the importance of a term by considering how frequently it appears in a document and how uncommon it is across a collection. It is widely used for information retrieval and can also support extractive sentence ranking [1]. TextRank treats sentences or other text units as nodes in a graph and uses relationships between them to rank their importance [2]. Research on meeting-recording summarization has also examined how extractive methods can identify useful discussion content from spoken interactions [3]. Research has also examined meeting-specific extractive summarization, where lexical and speech-related features are used to identify useful portions of meeting recordings [3]. A more recent survey discusses neural approaches and the challenges of abstractive meeting summarization, including long conversations and multiple speakers [13].

Neural language models have expanded the methods available for language understanding and text generation. BERT introduced contextual representations that can be fine-tuned for several language-understanding tasks [4]. The LLaMA research line demonstrated a family of foundation language models intended for research and generative applications [5]. For audio input, Whisper uses large-scale weak supervision to support speech recognition across multiple languages and recording conditions [6].

MeetMind uses these established methods as building blocks; it does not claim to introduce a new summarization algorithm. Its default text-analysis path uses TF-IDF sentence ranking and rule-based extraction so that the processing remains understandable and can run without a language model. Speech recognition converts recordings into text, while an optional local language model can provide additional generated insights. Related meeting-documentation systems have also combined speech recognition, topic detection, and summarization in a single workflow [14].

Technique/ study	Role in the field	Relevance to MeetMind	Consideration
TF-IDF [1]	Estimates term importance using document frequency.	Supports keyword and sentence ranking.	Does not fully represent context or meaning.
TextRank [2]	Ranks text units using graph relationships.	Provides a related extractive summarization approach.	Graph ranking may not capture all semantic relationships.
Meeting summarization [3]	Studies extraction of important content from meeting recordings.	Motivates meeting-specific summarization and decision capture.	Performance depends on transcript quality and meeting characteristics.
BERT [4]	Produces contextual language representations.	Represents a contextual NLP approach relevant to future enhancement.	Model deployment can require additional compute.
LLaMA [5]	Provides open-weight generative language models.	Motivates optional LLM-based enrichment.	Output quality and resource use depend on model and configuration.
Whisper [6]	Performs speech-to-text transcription.	Supports transcription of uploaded recordings.	Transcription quality can vary with audio conditions and model size.

III. PROBLEM STATEMENT AND OBJECTIVES

The problem addressed in this work is the difficulty of converting meeting discussions into reliable, searchable, and actionable records. Manual note-taking can omit information, and unstructured notes make it difficult to identify who is responsible for a task, when it is due, or whether a decision has already been made. In addition, when previous meeting records are stored in separate files, retrieving relevant information across meetings becomes inconvenient.

The objective of MeetMind is to develop a software application that processes meeting transcripts and recordings and organizes the resulting information for review and follow-up. The specific objectives are:

- Generate short and detailed summaries, along with key discussion points.
- Identify decisions and extract action items, including assignees, deadlines, and priorities when these details are stated.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

- Provide sentiment, topic, keyword, and speaker-contribution information.
- Transcribe supported audio/video recordings and process long inputs in manageable chunks.
- Store meeting information in a persistent database and support full-text retrieval.
- Provide task follow-up, calendar export, analytics, readiness indicators, and downloadable reports.

IV. PROPOSED SYSTEM

A. Workflow Overview

The user begins by entering a meeting transcript or uploading an audio/video recording through the application. Text input is validated before analysis. For uploaded media, the application checks the supported file type and configured file-size limit, extracts the audio, and sends it to the selected transcription backend. The generated transcript is then passed to the NLP analysis engine.

The analysis engine produces summaries, key points, decisions, action items, risks, topics, sentiment, and meeting statistics. If the optional Ollama service is enabled and available, its output can be used to enrich the analysis. The transcript and structured results are then saved in the SQLite knowledge base. Dashboard pages and follow-up modules retrieve information from this stored record.

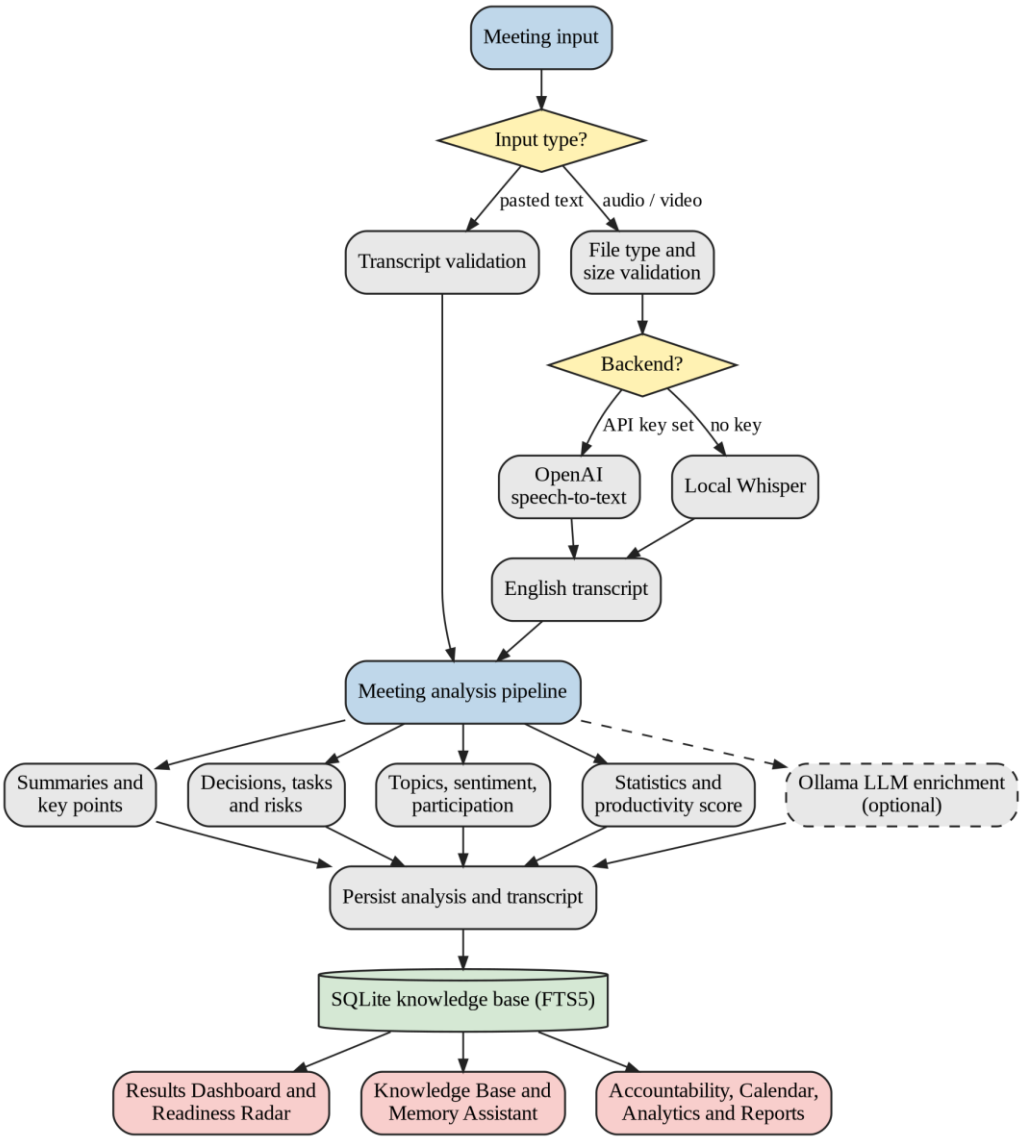


Fig. 1. End-to-end workflow of MeetMind.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

B. System Architecture

MeetMind is organized as a set of connected software modules. The Streamlit application provides the user interface, authentication flow, input controls, and navigation. The transcription module converts uploaded recordings into text, while the MeetingAnalyzer component performs NLP processing. The KnowledgeBase module stores transcripts and analysis results in SQLite and maintains an FTS5 index for text search.

The dashboard, Memory Assistant, Accountability Center, calendar functions, analytics, and report generator use the saved meeting records. This arrangement separates the processing stage from later review and retrieval. It also allows the application to continue using its rule-based analysis when optional LLM enrichment is unavailable.

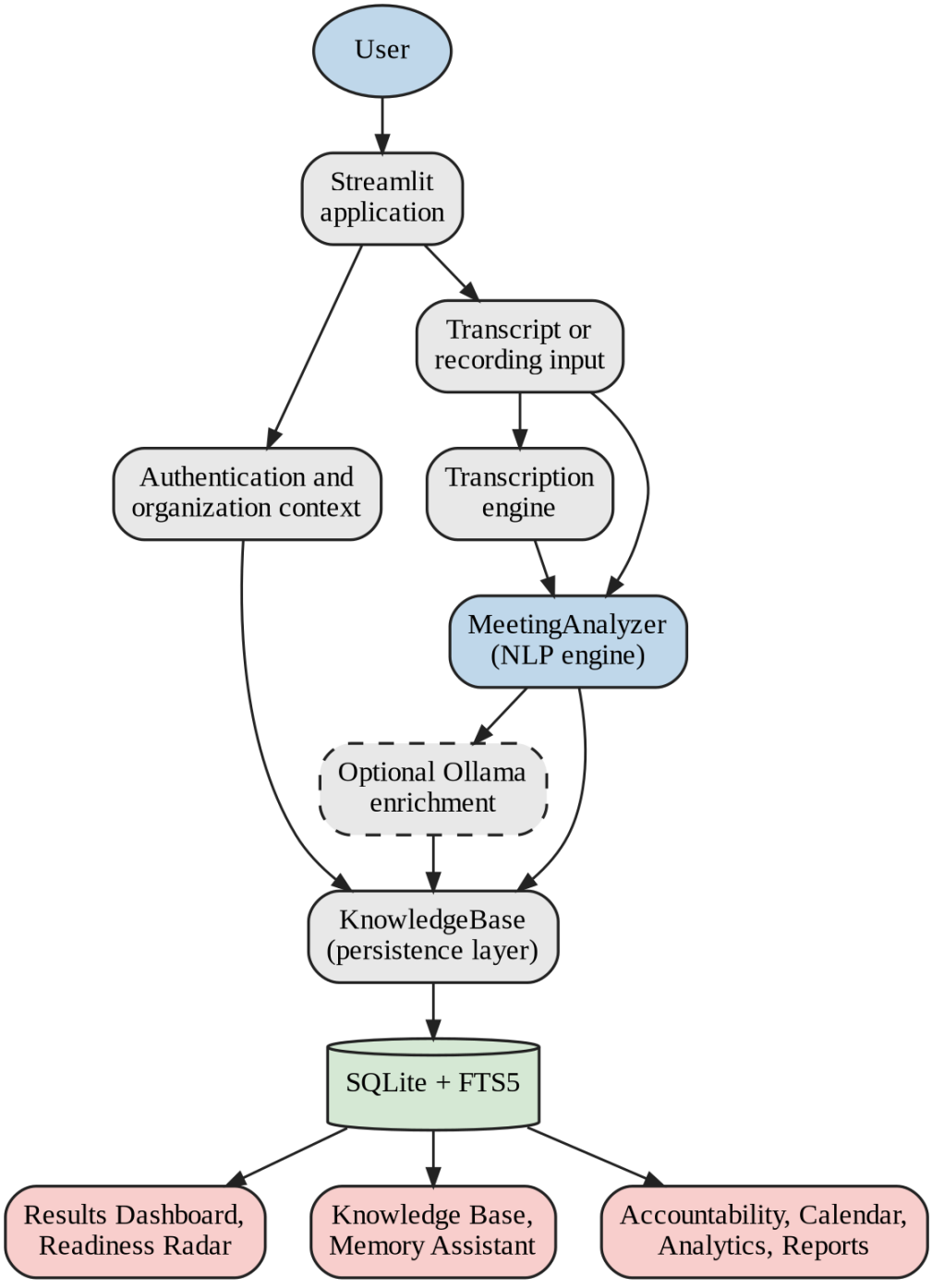


Fig. 2. Layered architecture of MeetMind.

DOI: 10.17148/IJARCCE.2026.151110

C. Input and Transcription Layer

The application accepts typed transcripts and supported audio/video files. When a speech-to-text API key is configured, the recording is submitted to the OpenAI transcription service. If the key is not available, or the API request fails, the application can use the local Whisper backend. The configured local model in the prototype is openai/whisper-tiny.en. The Whisper research describes large-scale weakly supervised training for speech recognition and multilingual audio processing [6].

To handle longer recordings, audio is divided into chronological segments, with a default chunk duration of 180 seconds. The resulting partial transcripts are joined in their original order before text analysis. MoviePy and SoundFile are used for media-duration handling and audio processing. The current application is designed for English input and produces English summaries and analysis; translation is not included.

D. NLP Analysis Engine

The default analysis engine uses statistical and rule-based processing and can run independently of the optional LLM. Transcripts exceeding the configured limit of 12,000 characters are divided into smaller sections. Each section is processed, and the extracted results are combined to form the meeting-level output. The TF-IDF implementation uses the scikit-learn toolkit [7], while the text-processing workflow draws on common NLP operations described in the NLTK literature and documentation [8], [10]. The use of scikit-learn for vectorization and statistical text processing is consistent with its documented machine-learning toolkit [7].

1) Text preprocessing

The transcript is cleaned and divided into sentences using NLTK and spaCy. Tokens are converted to lowercase, common stop words are removed, and words are lemmatized to bring related word forms to a common base. Where speaker labels are present in a format such as “Name: utterance”, the system uses them to associate statements with participants. The NLP libraries used for tokenization, linguistic processing, and related text operations are described in the NLTK and spaCy references [8]–[10].

2) Extractive summarization

[1] The system uses TF-IDF vectors from scikit-learn to estimate the importance of terms in each sentence. A sentence receives a higher ranking when it contains terms that are informative within the transcript. The general TF-IDF expression is:

$$\text{TF-IDF}(t, d) = \text{tf}(t, d) \times \log(N / \text{df}(t))$$

where tf(t, d) represents the frequency of term t in sentence d, N is the number of sentence-level documents, and df(t) is the number of sentences containing that term. The highest-ranked sentences are selected and returned in their original order. The application uses them to create a short summary, a detailed summary, and a set of key points. The user can adjust the requested summary length.

3) Decision, action-item, and risk extraction

The extraction stage combines spaCy named-entity information with linguistic patterns and predefined rules. Phrases that indicate agreement or commitment are examined as possible decisions. Statements that describe work to be completed are treated as potential action items. When the transcript explicitly provides the information, the system attempts to identify the responsible person, deadline, and priority. Risk detection looks for language associated with concerns, blockers, and dependencies and assigns a severity based on the configured rules.

4) Sentiment, topics, and speaker statistics

A lexicon-based method examines positive and negative terms to assign a positive, negative, or neutral label to statements and to summarize the overall sentiment. TF-IDF terms support keyword and topic identification. The application also calculates descriptive statistics, including word count, estimated meeting duration, number of speakers, and the contribution attributed to each speaker label.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

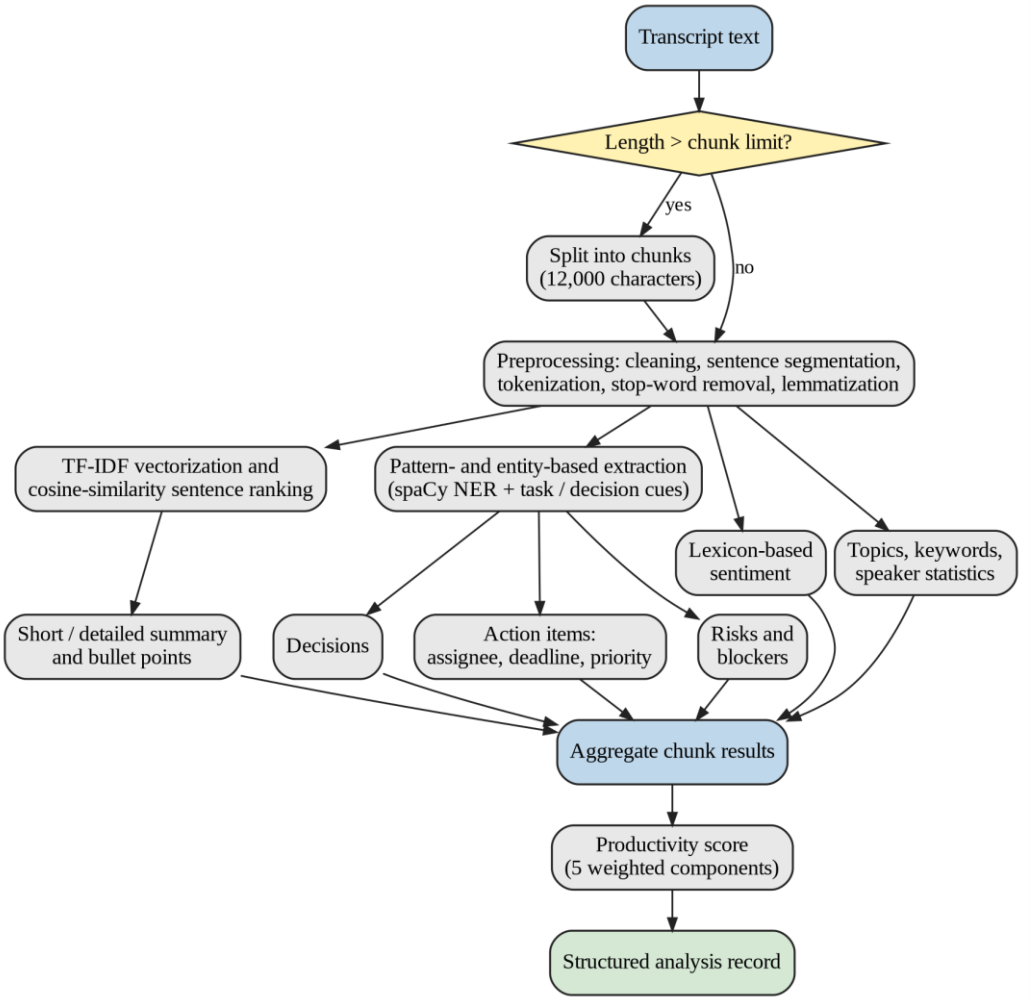


Fig. 3. NLP meeting-analysis pipeline.

E. Optional LLM Enrichment

Users can enable an additional language-model processing step through a locally hosted Ollama service. The prototype uses llama3.2 as its default model configuration. When available, the service receives the transcript and generates an additional summary or further insights. Because the configured endpoint is local, the transcript is processed on the host machine in this mode. If Ollama is unavailable, the application continues with the rule-based NLP output and indicates that the optional service is not active. Therefore, LLM enrichment is an additional capability and is not required for the core analysis pipeline.

F. Knowledge Base and Memory Assistant

After analysis, each meeting is saved in an SQLite database. The stored information includes the transcript, summaries, decisions, action items, and analytics. The database uses write-ahead logging (WAL) mode, and the application uses thread-local database connections. SQLite FTS5 provides full-text indexing over transcripts, summaries, decisions, and tasks. SQLite FTS5 provides the full-text indexing and query capability used by the knowledge-base search layer [11]. The FTS5 extension provides the full-text indexing and query capability used by the knowledge base [11].

The Memory Assistant uses the stored meeting information to retrieve context relevant to a user's question. It presents the meetings used as supporting sources so that users can check where the answer came from. User and organization information is also maintained to scope workspace records. The prototype's authentication and workspace controls should be reviewed further before use with sensitive or production data.

G. Follow-Through and Readiness Modules

The Accountability Center organizes extracted tasks by owner, due date, and status and highlights overdue work. The

DOI: 10.17148/IJARCCE.2026.151110

calendar module creates reminders and exports events in the iCalendar (.ics) format. The Meeting Readiness and Dependency Radar combines signals such as high-severity risks, blocker-related statements, and task-load pressure into a readiness indicator. This indicator is intended to help users prioritize review; it should not be interpreted as a guaranteed prediction.

The Quality Coach uses the analysis results to provide practical suggestions, such as reviewing unresolved risks or improving participation balance. These modules use the structured meeting record so that follow-up information remains connected to the discussion from which it was extracted.

H. Meeting Productivity Score

MeetMind calculates a meeting productivity score on a scale of 0 to 100. The score combines five components: action density, decision ratio, participation, risk awareness, and clarity. The overall expression is:

$$S = A + D + P + R + C,\ S \leq 100$$

Here, A represents action density, D represents the decision ratio, P represents participation, R represents risk awareness, and C represents clarity. The component limits used in the reported sample are 25, 25, 25, 15, and 10 points, respectively.

Component	Maximum points	Sample score
Action density	25	25.0
Decision ratio	25	25.0
Participation	25	20.2
Risk awareness	15	10.0
Clarity	10	6.3
Total	100	86.5 (approximately 87)

I. Report Generation

The report generator uses Jinja2 templates to prepare a structured meeting report. Depending on the available information, the report includes the meeting title and date, duration, word and speaker counts, productivity score, overall sentiment, executive and detailed summaries, key points, decisions, action items, risks, and participant analysis. Reports can be exported as Markdown, HTML, or email-style text. Calendar events can also be exported separately in .ics format.

V. IMPLEMENTATION

MeetMind is implemented in Python 3.10 or later and is designed to run on Windows, macOS, and Linux. On initial setup, the application may need to download the required NLTK resources, the spaCy en_core_web_sm model, and the selected local Whisper model. The reference configuration described for the prototype includes an Intel Core i3/i5/i7-class processor or better, at least 8 GB of RAM, and approximately 50 GB of free storage. No dedicated GPU is specified as a requirement for the basic workflow; actual processing time depends on the selected transcription model and input length.

Area	Technology	Purpose in the prototype
User interface	Streamlit	Input workflow, authentication, navigation, and dashboards
NLP	spaCy, NLTK, scikit-learn, NumPy	Text processing, TF-IDF, entity and keyword analysis
Visualization	Plotly	Interactive charts and analytics
Speech-to-text	OpenAI API, faster-whisper / Whisper	Remote or local transcription
Media processing	MoviePy, SoundFile	Audio duration handling and chunking

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

Optional language model	Ollama	Local LLM-based enrichment
Database and search	SQLite, FTS5	Persistent meeting records and full-text search
Reporting and configuration	Jinja2, python-dotenv	Report templates and environment configuration
Testing	pytest	Unit and integration tests

Configuration variable	Prototype default	Purpose
SPEECH_TO_TEXT_API_KEY	Unset	Optional key for API-based transcription
OLLAMA_URL / OLLAMA_MODEL	localhost:11434 / llama3.2	Local LLM endpoint and model
WHISPER_MODEL	openai/whisper-tiny.en	Local English transcription model
MAX_AUDIO_FILE_SIZE_MB	30	Maximum permitted upload size
AUDIO_CHUNK_SECONDS	180	Duration of each audio segment
TRANSCRIPT_CHUNK_CHARS	12000	Maximum transcript characters per analysis chunk

The application is divided into modules for the user interface, configuration, NLP processing, transcription, LLM integration, database operations, authentication, calendar management, analytics, and report generation. The prototype includes pytest test suites for transcription, knowledge-base operations, calendar functions, and security-related checks. A demonstration deployment can be made through Streamlit Community Cloud; only non-sensitive sample data should be used for such a hosted demonstration.

VI. SYSTEM USER INTERFACE

MeetMind provides a Streamlit-based interface through which users can submit meeting content, review the generated analysis, and access saved meeting information. The interface is organized around a simple sequence: users enter the workspace, provide a transcript or recording, review the generated results, and use the available follow-up modules.

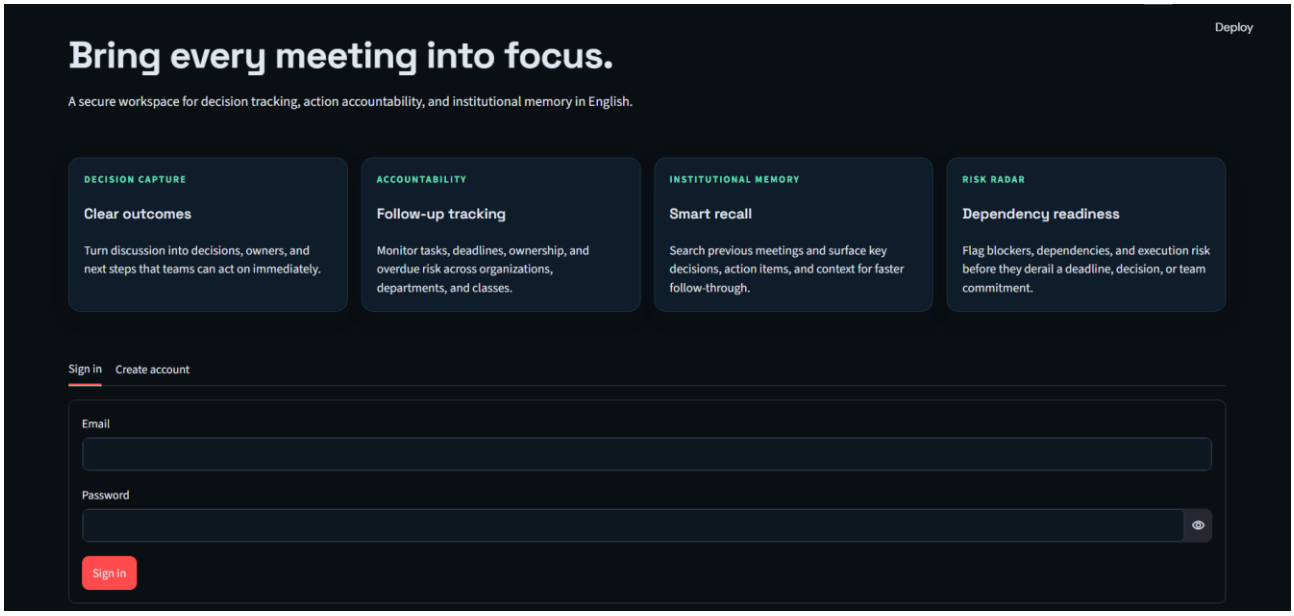


Fig. 5. MeetMind landing page showing the application overview and sign-in options.

A. Landing Page and Sign-in

The landing page introduces MeetMind and presents its main capabilities, including decision capture, accountability, institutional memory, and risk monitoring. Users can sign in or create an account to enter the workspace. After signing in, the Analyze Meeting page allows users to enter a meeting title and choose an input type: text transcript, audio

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

recording, or video recording. For text-based analysis, the user can paste the transcript into the provided text area and start processing. The page also includes a summary-length control and an optional language-model setting.

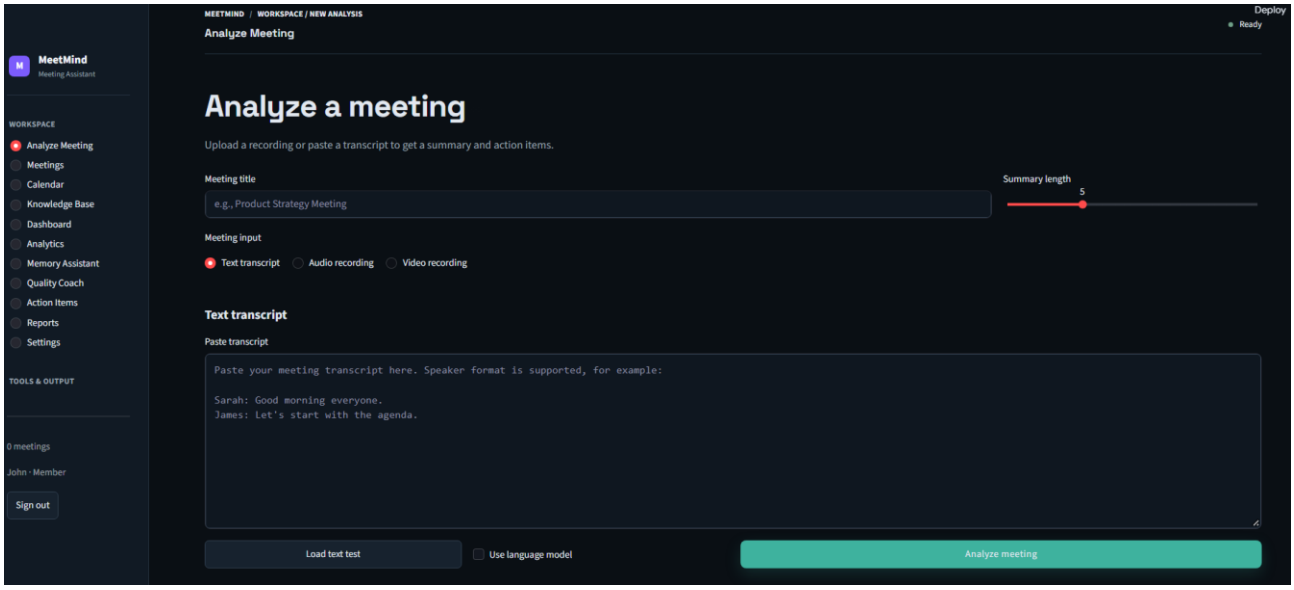


Fig. 6. Analyze Meeting screen for entering a meeting title and providing transcript or recording input.

B. Generated Meeting Summary and Action Items

After analysis, MeetMind displays the meeting summary and provides tabs for viewing the summary, detailed summary, key points, and enhanced summary. The results screen also presents extracted decisions and action items. In the example shown, the action-item panel displays task descriptions together with available ownership, due-date, and priority information. The original transcript can also be expanded for reference. The displayed content is an example output from the application and should not be interpreted as a measured accuracy result.

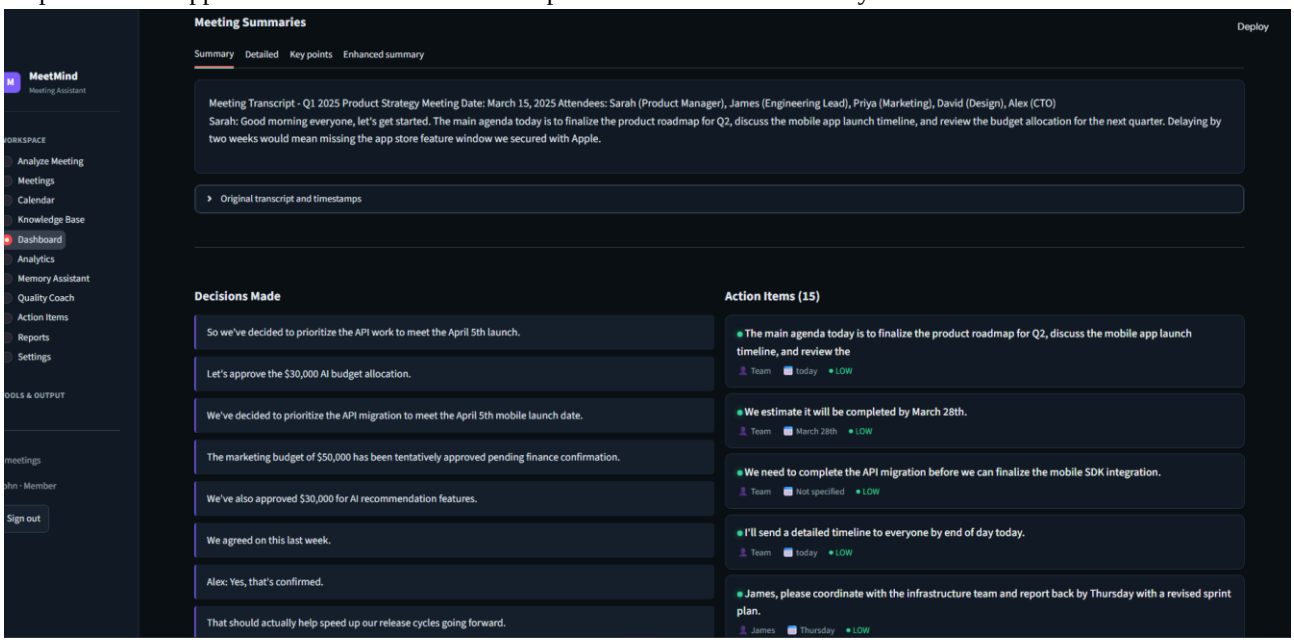


Fig. 7. Meeting analysis results showing the generated summary, decisions, and extracted action items.

C. Workspace Navigation and Additional Modules

The workspace navigation provides access to additional modules, including Meetings, Calendar, Knowledge Base, Dashboard, Analytics, Memory Assistant, Quality Coach, Action Items, Reports, and Settings. These modules support the wider meeting-management workflow described in the proposed system. The screenshots included in this section document the landing page, meeting-input screen, and analysis-results screen supplied for this study.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

VII. DATASET AND TEST MATERIAL

The application processes transcripts supplied by users or generated from uploaded recordings. The test material described for the prototype includes a sample "Q1 2025 Product Strategy Meeting" transcript with eight participants representing product, engineering, marketing, design, and technology roles. The project repository also contains five synthetic English demonstration recordings with corresponding transcripts. Four recordings are shorter than two minutes, while one is approximately ten minutes long. These materials are used to demonstrate transcription, structured analysis, and chunk-based processing.

Stored field	Description
Meeting title and date	Identifies the meeting and its date
Transcript	Text of the meeting discussion
Participants	Participant names and available speaker labels
Decisions	Statements identified as meeting decisions
Action items	Tasks with available assignee, deadline, priority, and status
Risks	Concerns, blockers, and dependencies identified in the discussion
Duration and statistics	Meeting duration, word count, and speaker count
Productivity score	Heuristic meeting score on a 0–100 scale

VIII. RESULTS AND DISCUSSION

A. Functional Testing

The prototype was checked using eight functional test cases. The tests examined whether the main analysis functions generated outputs, whether meeting records could be saved and retrieved, and whether the dashboard and report features operated as expected. All eight test cases were recorded as passed. These results indicate that the tested functions operated for the selected test inputs. They do not establish the precision or recall of the extraction algorithms.

Test case	Expected behaviour	Observed status
Meeting analysis	Generate a meeting summary	Passed
Decision detection	Identify decision statements	Passed
Action-item extraction	Identify task-related statements	Passed
Risk detection	Identify concerns and blockers	Passed
Knowledge-base storage	Save a meeting record	Passed
Analytics dashboard	Display meeting statistics	Passed
Report generation	Create a meeting report	Passed
Report download	Allow the report to be downloaded	Passed

B. Observed Output

For the sample product-strategy meeting, the transcript contained 617 words and eight speaker labels. The system estimated the meeting duration at approximately five minutes and extracted 15 action items and eight decisions. The productivity score was 86.5 out of 100, which the interface displayed as approximately 87% and assigned grade A. The identified decisions included prioritizing API work for an April 5 launch and approving an AI budget allocation of $30,000.

At the time represented by the prototype screenshots, the knowledge base contained 11 meeting records. The analytics dashboard showed 150 extracted tasks and 81 decisions across those records. The average transcript length was 563 words per meeting, and the average productivity score was 84. These are observed application outputs, not ground-truth accuracy results.

Measure	Sample meeting	Knowledge base (11 meetings)
Word count	617	Average 563
Tasks extracted	15	150 total
Decisions detected	8	81 total
Productivity score	86.5 (displayed as 87)	Average 84

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151110

C. Development Across Project Phases

The project was developed in two broad phases. The initial phase focused on transcript-based analysis. Users supplied meeting text, and the application generated summaries, decisions, and task-related information. The later phase extended the workflow to accept audio and video recordings, added API or local Whisper transcription, and introduced persistent search, workspace management, task follow-up, calendar export, and additional review tools.

Capability	Initial text-based phase	Audio/video phase
Input	Pasted transcript	Transcript or uploaded audio/video
Transcription	Not included	OpenAI API or local Whisper
Analysis	Summaries, decisions, tasks, and risks	Same analysis with chunked long transcripts
Task information	Task phrases, owners, and deadlines	Assignee, deadline, priority, and status
Storage and search	SQLite storage and basic search	SQLite with FTS5 and Memory Assistant
Follow-up	Limited	Accountability Center and calendar export
Review features	Productivity score	Readiness Radar and Quality Coach
User management	Single-user workflow	Sign-in and organization scoping
Reports	Text report	Markdown, HTML, and email-style output

D. Discussion

The observed results show that the prototype can connect transcript input, structured analysis, storage, and follow-up features within one application. The combination of deterministic NLP and optional LLM enrichment provides two processing paths: a locally executable baseline and an additional generative step when the local model service is available. The source-linked results in the Memory Assistant and the use of heuristic scores are intended to support human checking rather than replace it.The current evaluation is functional rather than quantitative. The number of extracted tasks or decisions does not by itself show whether every item is correct or whether important items were missed. A more rigorous study will require manually annotated reference data, defined evaluation procedures, and metrics such as precision, recall, and F1-score for extraction tasks. Summary quality can be assessed using suitable reference-based measures, together with human review.

IX. PRIVACY, ETHICAL CONSIDERATIONS, AND LIMITATIONS

Meeting recordings and transcripts may contain personal, organizational, or confidential information. In the default local configuration, meeting records are stored in an SQLite database on the host machine, so access to the device and database must be controlled. When API-based transcription is selected, the uploaded recording is transmitted to the configured service for processing. Where organizational policy requires local processing, users should select the local transcription option and review the handling of generated files.

The optional Ollama configuration processes transcript content through a local endpoint by default. A hosted demonstration environment should use synthetic or otherwise non-sensitive meeting data. Before production deployment, authentication, workspace isolation, retention rules, access permissions, and secure handling of recordings and transcripts require further review.

The current prototype has several limitations. It supports English only. Speaker contribution analysis depends on speaker labels present in the transcript or transcription output; the application does not perform speaker identification. Rule- and lexicon-based extraction may miss decisions, owners, dates, or risks when they are expressed indirectly or in unusual language. Sentiment labels may also fail to capture context, irony, or nuanced opinions. The readiness indicator and productivity score are heuristic measures and should not be treated as objective assessments of meeting quality. Finally, the current tests do not provide a quantitative accuracy benchmark, and the authentication and organization-scoping features require additional security evaluation.

X. CONCLUSION AND FUTURE WORK

This paper presented MeetMind, a meeting-support application that converts transcripts and uploaded recordings into structured meeting information. The system combines speech-to-text transcription, TF-IDF-based sentence ranking, rule- and entity-based extraction, optional local LLM enrichment, and persistent storage using SQLite with FTS5. It also provides tools for searching earlier meetings, reviewing decisions and tasks, monitoring follow-up work, exporting calendar events, viewing analytics, and generating reports.

The prototype passed the eight functional test cases described in this study. In the sample product-strategy meeting, it processed a 617-word transcript with eight speakers and produced 15 action items, eight decisions, and a productivity



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151110

score of 86.5. The broader knowledge-base statistics show that the application can store and summarize information across multiple meeting records. These observations demonstrate functional operation but should not be interpreted as proof of extraction accuracy.

Future work will focus on evaluating the system with a larger and more varied set of meeting recordings and manually annotated transcripts. Summary quality can be assessed using ROUGE alongside human evaluation, while action-item and decision extraction can be measured using precision, recall, and F1-score. Further development may include transformer-based abstractive summarization, speaker diarization, multilingual processing, integration with external calendar and task-management services, and stronger production-level security, access control, and durable storage. Evaluation of meeting summaries has also been studied using both automatic and human-centred assessment methods [15]. ROUGE is one established automatic measure for comparing generated summaries with reference summaries, although human review is also important for judging factual correctness and usefulness [16].

ACKNOWLEDGMENT

The authors express their gratitude to the Head of the Department, faculty members, and management of SES's R. C. Patel Institute of Technology, Shirpur, for their support and for providing the facilities required for this work. The authors also thank Dr. Ujwala Patil for her guidance and academic support during the development of the project.

REFERENCES

[1] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988, doi: 10.1016/0306-4573(88)90021-0.
[2] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in Proc. 2004 Conf. on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain, 2004, pp. 404–411. [Online]. Available: https://aclanthology.org/W04-3252/
[3] G. Murray, S. Renals, and J. Carletta, "Extractive summarization of meeting recordings," in Proc. Interspeech 2005, Lisbon, Portugal, 2005, pp. 593–596. [Online]. Available: https://www.isca-archive.org/interspeech_2005/murray05_interspeech.pdf
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, USA, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.
[5] H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv:2302.13971, 2023. [Online]. Available: https://arxiv.org/abs/2302.13971
[6] A. Radford et al., "Robust speech recognition via large-scale weak supervision," in Proc. 40th Int. Conf. on Machine Learning, PMLR, vol. 202, 2023, pp. 28492–28518. [Online]. Available: https://arxiv.org/abs/2212.04356
[7] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html
[8] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009. [Online]. Available: https://www.nltk.org/book/
[9] Explosion AI, "spaCy: Industrial-strength natural language processing in Python," documentation. [Online]. Available: https://spacy.io/
[10] NLTK Project, "Natural Language Toolkit," documentation. [Online]. Available: https://www.nltk.org/
[11] SQLite Consortium, "FTS5: Full-text search," SQLite documentation. [Online]. Available: https://www.sqlite.org/fts5.html
[12] MeetMind Project Team, "AI-Powered Meeting Summarizer and Action Item Generator," source-code repository. [Online]. Available: https://github.com/dp101-codecooks/AI-Powered-Meeting-Summarizer-Action-Item-Generator
[13] V. Rennard, G. Shang, J. Hunter, and M. Vazirgiannis, "Abstractive meeting summarization: A survey," Transactions of the Association for Computational Linguistics, vol. 11, pp. 861–884, 2023, doi: 10.1162/tacl_a_00578.
[14] T.-C. Huang, C.-H. Hsieh, and H.-C. Wang, "Automatic meeting summarization and topic detection system," Data Technologies and Applications, vol. 52, no. 3, pp. 351–365, 2018, doi: 10.1108/DTA-09-2017-0062.
[15] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in Text Summarization Branches Out, Barcelona, Spain, 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013/
[16] G. Murray, S. Renals, J. Carletta, and J. Moore, "Evaluating automatic summaries of meeting recordings," in Proc. ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, MI, USA, 2005, pp. 33–40.