

Automatic Meeting Minutes Generation Using Multi-Speaker Dialogue Summarization

Manthan Gawas¹, Sanket Salokhe², Nehal Kambale³, Suraj Hamidwade⁴, Govind Gaikwad⁵ Prof. D. A. Jadhav⁶

^{1,2,3,4,5} Department of Computer Science and Engineering, Ashokrao Mane Group Of Institute, Vathar tarf Vadagaon.

ABSTRACT

Meetings remain one of the primary mechanisms for decision-making, collaboration, progress tracking, and knowledge exchange in academic and professional environments. However, the manual preparation of meeting minutes is time-consuming, inconsistent, and highly dependent on the attention and writing ability of a human note-taker. Important decisions, ownership details, action items, deadlines, and participant contributions may be missed when meetings are long, multi-speaker, or conducted in hybrid mode. The Automatic Meeting Minutes Generation system, named MeetMinutes in the implementation, addresses this problem by combining live meeting-room management, browser-based speech capture, transcript segmentation, Web RTC video communication, Web Socket event synchronization, and rule-based natural language processing for structured minutes generation.

The proposed system enables users to register, schedule meetings, join secure video rooms, capture spoken input as transcript segments, identify speakers, detect action-oriented and decision-oriented statements, summarize speaker contributions, produce person-to-person interaction summaries, and export the generated minutes in document formats. The backend is implemented in Java with service layers for authentication, meeting-room handling, summary generation, and database synchronization. The frontend provides a browser-based dashboard, live room interface, transcript controls, summary review page, and export buttons. The system uses practical NLP heuristics such as action markers, decision markers, speaker grouping, duplicate-line filtering, sentiment tagging, and due-date inference to convert raw conversational text into usable meeting records.

This paper presents the background, literature foundation, proposed methodology, architecture, UML-level design, sequence flow, activity flow, results, discussion, conclusion, and key takeaways of the project. The research emphasizes that even a lightweight and transparent NLP pipeline can provide meaningful automation for meeting documentation when it is integrated with collaboration workflows. The results indicate that the system improves documentation speed, organizes participant-wise accountability, and creates a foundation for future integration with advanced automatic speech recognition and transformer-based abstractive summarization models.

1. INTRODUCTION

Modern organizations conduct meetings across physical, remote, and hybrid settings. These meetings generate important information such as agenda discussions, decisions, risks, deadlines, responsibilities, blockers, and follow-up tasks. Traditional manual minute-taking often interrupts participation because one person must listen, understand, summarize, and format the meeting at the same time. In fast-moving conversations, it becomes difficult to capture all speaker turns accurately. After the meeting, reconstructing decisions from memory or chat messages can lead to incomplete records and weak accountability.

Automatic meeting minutes generation is an applied natural language processing problem that sits at the intersection of speech recognition, speaker identification, dialogue understanding, information extraction, summarization, and document generation. A complete solution must not only transcribe speech but also transform raw text into structured knowledge. For example, the sentence 'I will share the deployment report by Friday' must be understood as an action

item with an owner, task, and due date. Similarly, the phrase 'we agreed to finalize the release scope' should be treated as a decision marker.

The Meet Minutes project focuses on this practical transformation. It provides a web-based application where users can create meetings, join live rooms, communicate through audio/video, capture transcripts, generate summaries, and export the output. The implemented Java services include meeting analysis, room summary generation, action item extraction, decision highlighting, speaker summary construction, and person-to-person dialogue summarization. The frontend supports dashboard management, meeting history, live meeting controls, speech capture, chat, recording, and summary download.

The objective of this research paper is to document the design and working of the Automatic Meeting Minutes Generation project in a formal academic style. The paper studies the problem, reviews relevant literature, explains the proposed method, presents UML and behavioral diagrams, discusses results, and identifies future improvements. The research demonstrates that structured meeting minutes can be generated through a modular architecture even when using lightweight NLP methods, and that the same architecture can later support deeper machine learning or transformer-based models.

2. LITERATURE REVIEW

Early work in meeting summarization established that meetings are more complex than ordinary documents because they contain interruptions, informal language, multiple speakers, incomplete sentences, overlapping goals, and context-dependent references. The AMI Meeting Corpus is widely recognized as an important benchmark because it provides multimodal meeting recordings, transcripts, annotations, and summaries for research on meeting understanding. Such corpora helped researchers evaluate automatic summarization, dialogue act recognition, speaker activity, and meeting browser systems.

Automatic speech recognition has improved significantly over time, enabling meeting systems to convert speech into text with increasing accuracy. However, transcription alone does not solve the minutes-generation problem. A transcript may be long, repetitive, noisy, and difficult to read. Meeting minutes require compression, organization, and task-oriented extraction. Therefore, many studies separate the problem into stages: speech-to-text conversion, speaker diarization, segmentation, important utterance selection, action item detection, decision extraction, and final summary generation.

Research on extractive summarization selects important utterances or sentences directly from the transcript. This approach is easier to interpret and usually preserves factual correctness because it reuses original statements. Abstractive summarization attempts to generate new sentences that capture the meaning of the conversation. Neural abstractive models and transformer architectures have improved fluency, but they can introduce factual errors if not carefully grounded. For academic and enterprise meeting minutes, reliability and traceability are important; therefore, a hybrid approach is often preferred.

Action item extraction is a specialized information extraction task. It identifies statements that express responsibility, future work, ownership, and deadlines. Common linguistic signals include modal verbs, future-tense statements, command phrases, and project-management expressions such as 'follow up', 'send', 'prepare', 'submit', 'complete', or 'next step'. Decision detection similarly relies on markers such as 'agreed', 'approved', 'decided', 'confirmed', and 'finalized'. The Meet Minutes implementation uses this insight through predefined action and decision hint lists.

Speaker-aware summarization is another important area. Meeting summaries are more useful when they show who contributed what, who owns which task, and which participants interacted around a topic. Person-to-person summaries can expose collaboration patterns by grouping exchanges between pairs of speakers. This supports accountability and helps absent members quickly understand the meeting dynamics. MeetMinutes includes speaker accumulators and interaction accumulators that group transcript segments by participant and pairwise exchange.

The literature shows that a useful automatic minutes system should combine transcription, structured extraction, user interface support, and exportable reports. The proposed project follows this direction by integrating live meeting management with automatic summary generation rather than treating summarization as an isolated offline task.

3. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

Proposed Methodology:

The proposed methodology follows a pipeline-oriented design. First, authenticated users create or join a meeting room. The live room captures participation data, chat messages, media state, and transcript segments. Speech capture is performed in the browser through speech recognition support, while the meeting interface allows participants to manually or automatically add transcript text. Each transcript segment contains the speaker name, participant identity when available, text, source, and timing context.

Second, the backend normalizes transcript text by trimming whitespace, removing duplicate speaker-text combinations, and canonicalizing speaker names against the participant list. This improves consistency when the same participant is written with different capitalization or when simulated speaker controls are used for testing. The system then analyzes every transcript segment for action and decision signals.

Third, speaker-wise accumulation is performed. For each participant, the system stores spoken-line count, highlighted contributions, action count, and decision count. A short speaker summary is generated by selecting the first important contribution snippets and combining them with the participant's speaking statistics. This design gives a fast overview of each participant's role in the meeting.

Fourth, action items are extracted using a hint-based classifier. If a transcript statement contains action-oriented expressions such as 'will', 'follow up', 'send', 'prepare', 'submit', 'deliver', 'complete', or 'next step', it is converted into an action item. The owner is the speaker, the task is the normalized sentence, the due label is inferred from terms such as 'today', 'tomorrow', or 'Friday', and the default status is OPEN. Decision highlights are extracted using markers such as 'decided', 'approved', 'agreed', 'finalized', 'confirm', and 'confirmed'.

Fifth, the system creates the final minutes. The output includes an overall summary, decision highlights, key discussion points, action items, speaker summaries, person-to-person dialogue summaries, transcript records, and export-ready content. The frontend presents the generated summary and provides buttons for DOCX/PDF download and email sharing.

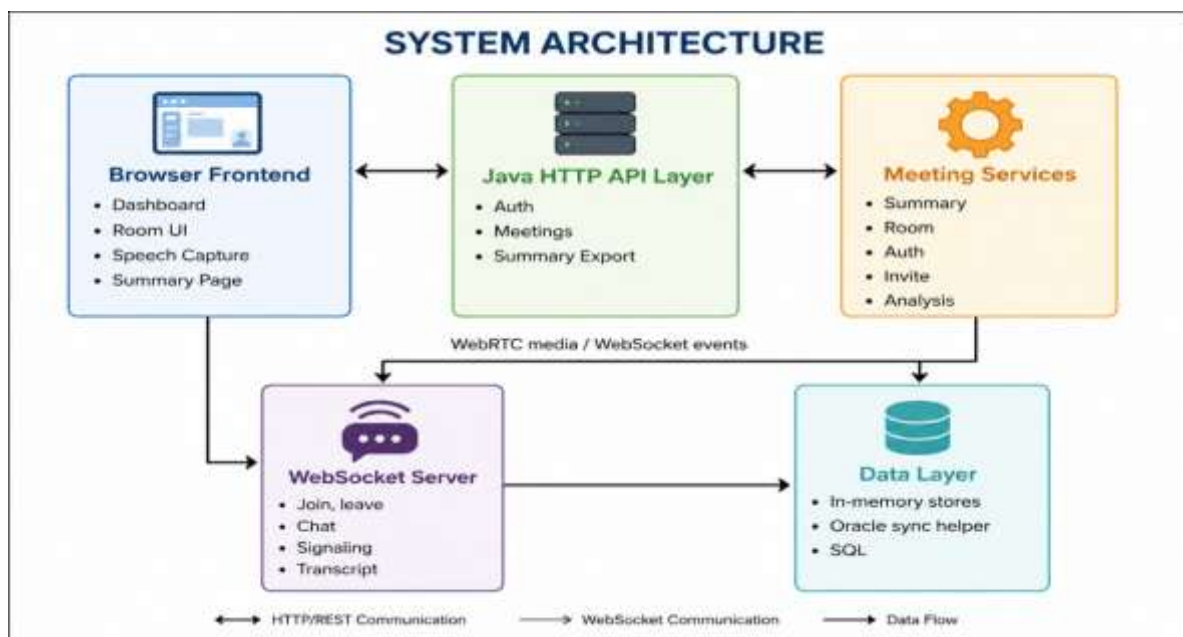


Figure 1 : System Architecture Diagram for Automatic Meeting Minutes Generation.

The architecture separates user interaction, real-time communication, API processing, business logic, and data persistence. The browser frontend manages meeting workflows and live collaboration. The Java API layer handles form-based requests, authentication, room operations, summary generation, and export endpoints. The WebSocket server manages real-time room events, including participant presence, chat, media state, transcript updates, and WebRTC signaling. The service layer applies the summarization logic, and the data layer stores users, meetings, participants, transcripts, summaries, and generated insights.

UML Diagrams:

The use case model shows that the system supports both general participant activities and host-specific controls. All users can join meetings and contribute to transcripts, while the host can manage room-level control and final minutes distribution.

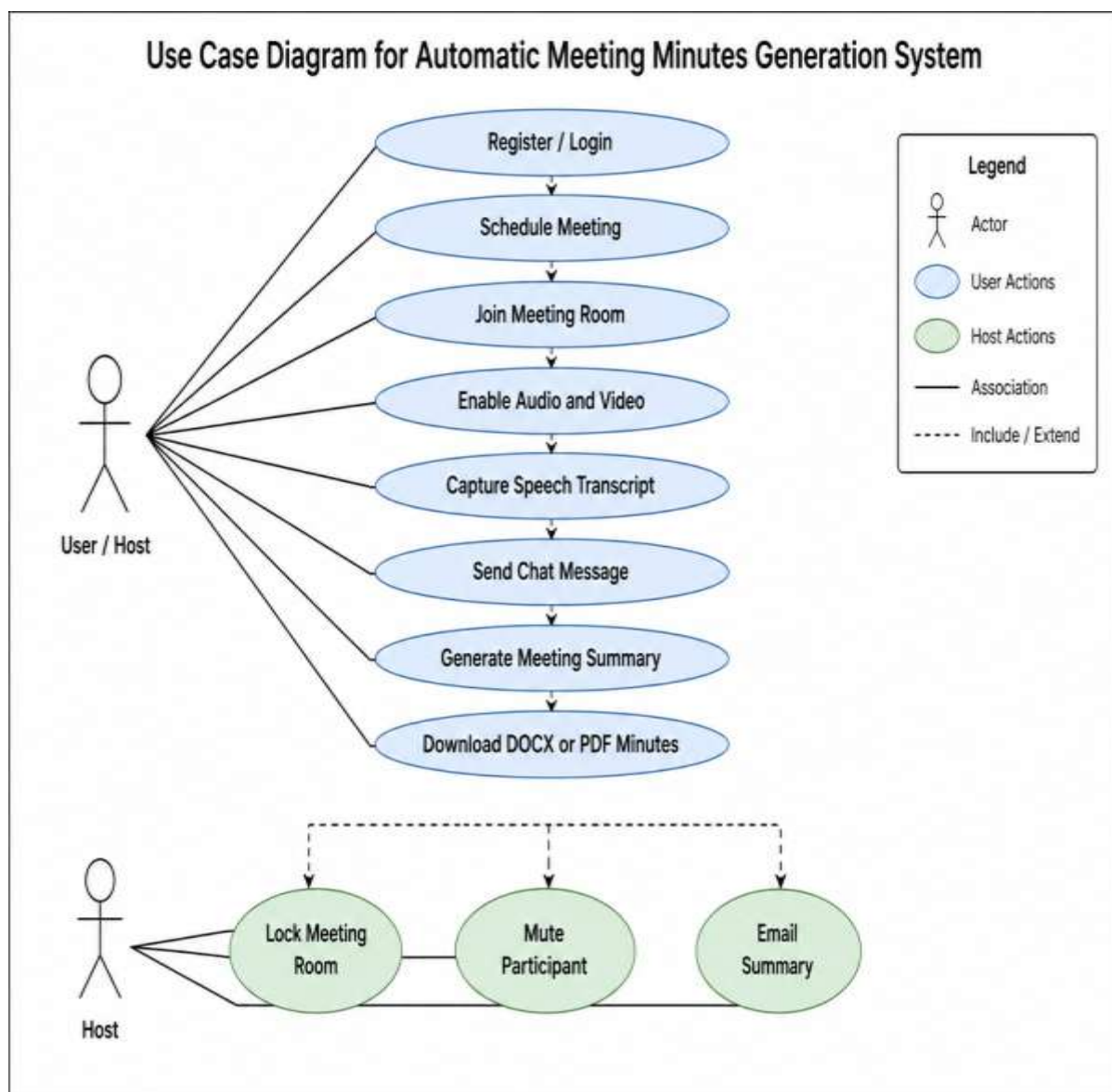


Figure 2: Use Case Diagram for Automatic Meeting Minutes Generation System .

Class Diagram:

The class model reflects the core entities visible in the project source. A meeting room contains participants and transcript segments. The summary service transforms transcript segments into a meeting summary containing action items, speaker summaries, key points, decision_highlights_and_person-to-person_summaries.

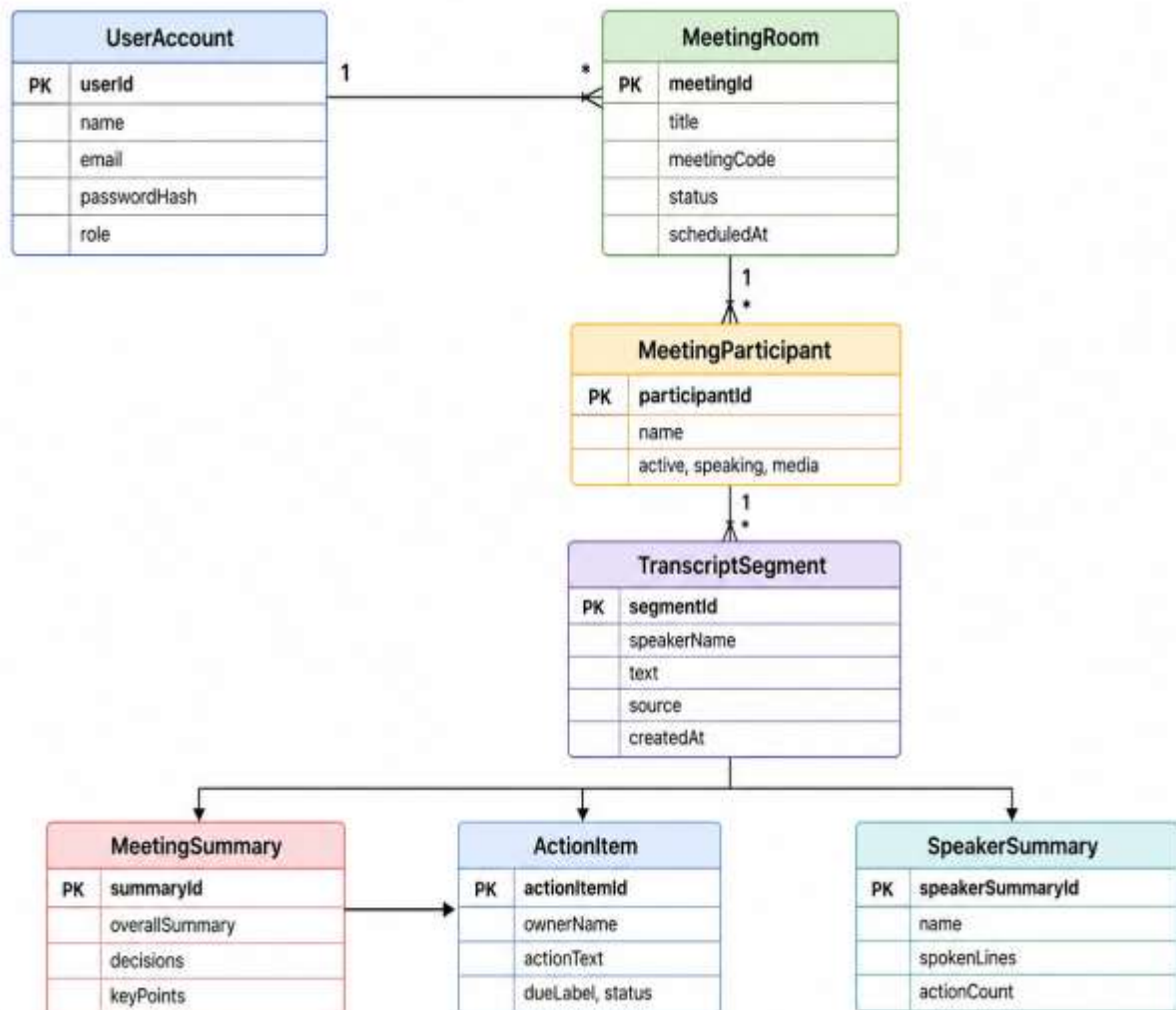


Figure 3: ER Diagram for Automatic Meeting Minutes Generation System.

Component Diagram

Component Diagram for Automatic Meeting Minutes Generation System

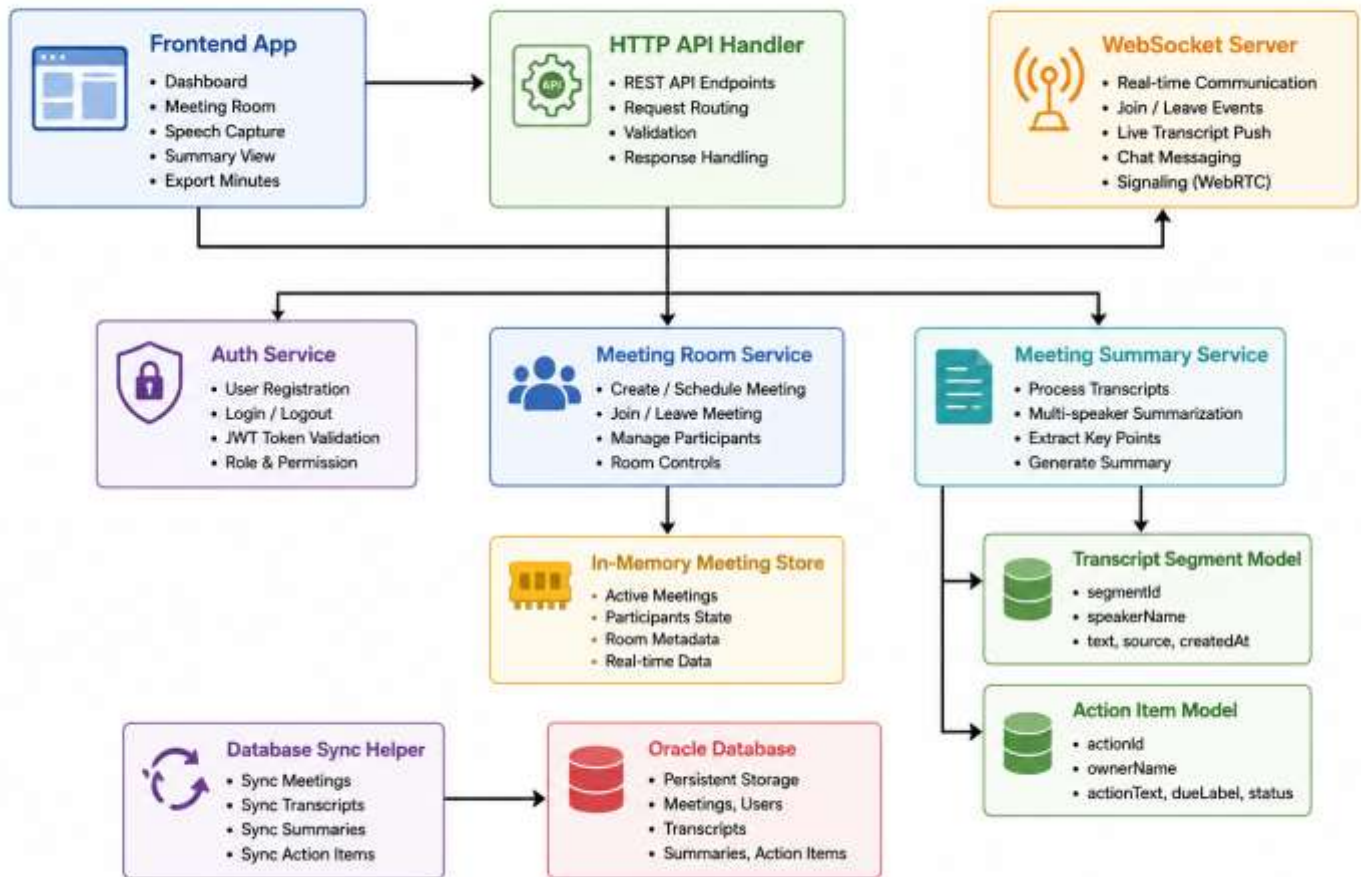


Figure 4: Component Diagram for Automatic Meeting Minutes Generation System.

Sequence Diagram

The sequence diagram shows the main runtime behavior from user authentication to minutes export. The most important design idea is that transcript generation and summary generation are separated: transcript segments can be captured continuously, while the summary can be generated on demand after or during the meeting.

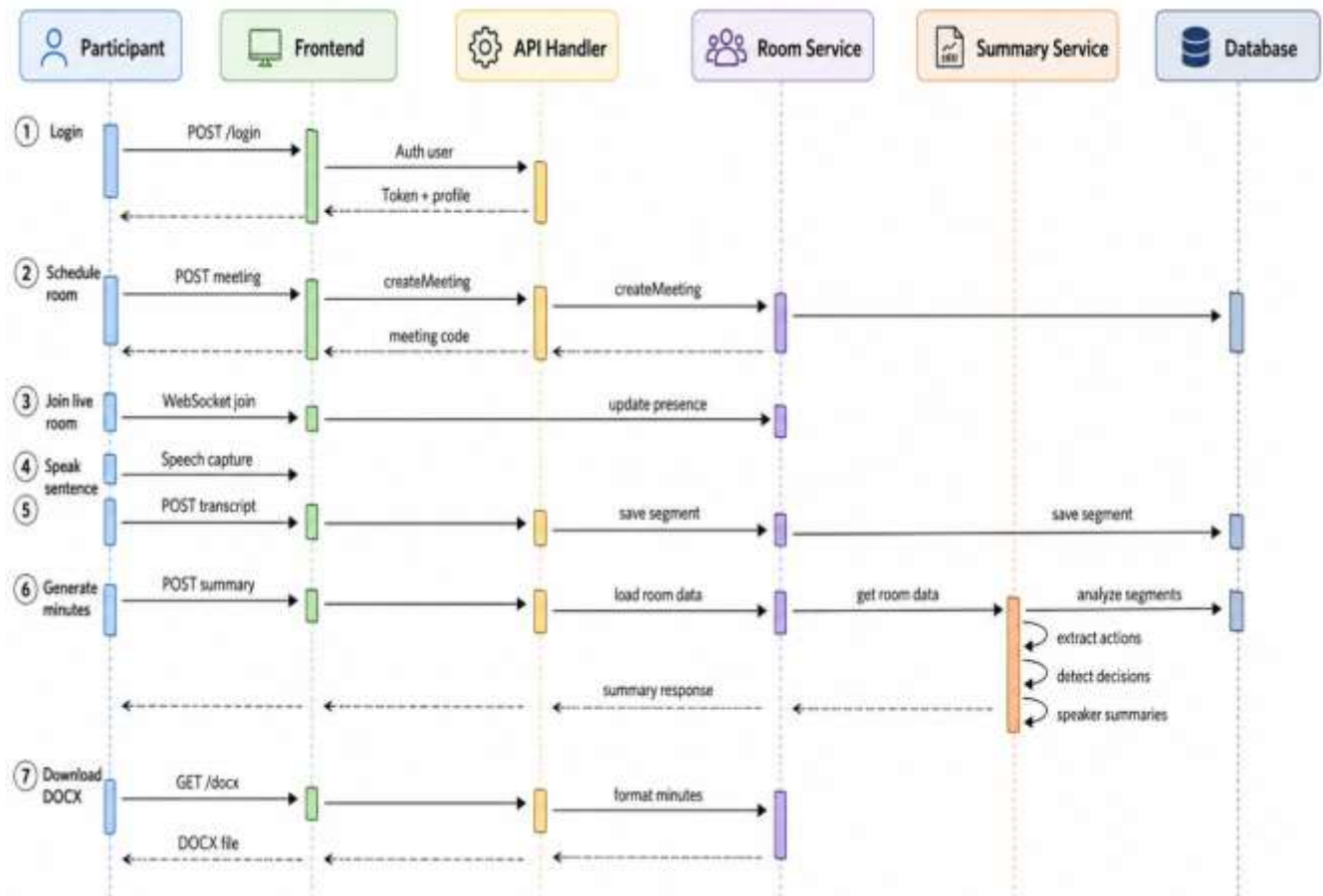


Figure 5: Sequence Diagram for Automatic Meeting Minutes Generation System.

Activity Diagram (Short Description)

In short, the activity flow begins with authentication and meeting creation, continues through real-time participation and transcript capture, and ends with automatic generation of structured minutes. The decision nodes represent lightweight NLP classification for action items and decision highlights. The flow supports both live and post-meeting usage.

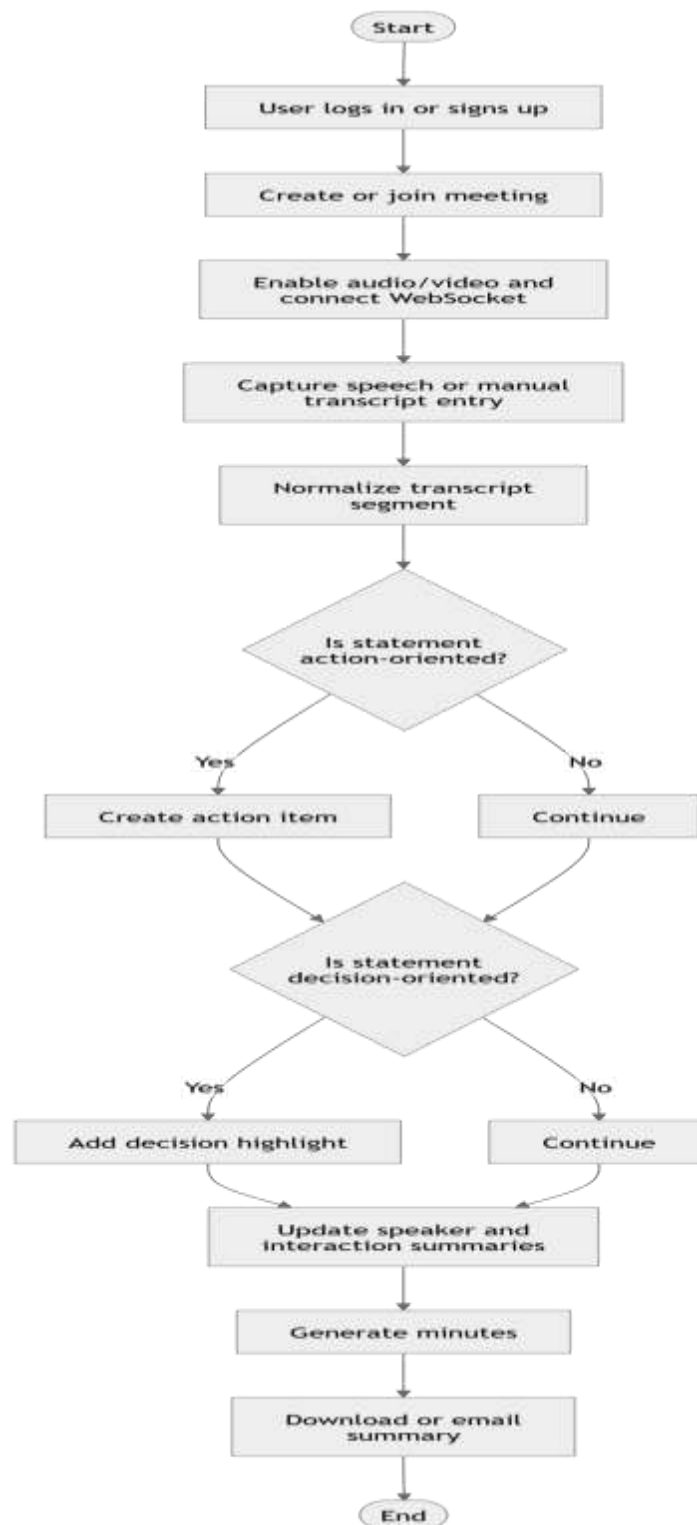


Figure 5: The Activity Diagram for Sequence Diagram for Automatic Meeting Minutes Generation System.

4. RESULTS AND DISCUSSION

□ The implemented system demonstrates that automatic meeting minutes can be generated from multi-speaker transcripts using a modular Java and browser-based architecture. The dashboard supports meeting scheduling, quick joining, meeting history, and summary review. The live room supports participant state, Web RTC media, Web Socket synchronization, chat, speech capture, screen sharing, recording, and host controls. The summary page presents the overall meeting summary, key discussion points, action items, speaker-wise notes, person-to-person dialogue summaries, and download actions.

□ The summary generation service produces useful structured output by applying deterministic rules. The action item extractor identifies future work by scanning for common action phrases such as 'will', 'follow up', 'share', 'send', 'prepare', 'submit', 'deliver', 'complete', and 'next step'. The decision detector searches for expressions such as 'decided', 'approved', 'agreed', 'finalized', 'confirm', and 'confirmed'. The due-date inference module detects words such as 'today', 'tomorrow', and 'Friday' and otherwise assigns 'Next checkpoint'. These rules are simple, transparent, and easy to adjust for project-specific vocabulary.

□ Speaker summaries are created by counting spoken lines, collecting representative highlights, and tracking action and decision counts. This is valuable because meeting minutes often need to identify responsibility and contribution, not only summarize the general topic. The person-to-person summary feature adds an additional analytical layer by examining exchanges between adjacent speakers. This can help users understand collaboration patterns and participant alignment.

□ The major strength of the system is its end-to-end integration. Instead of requiring users to upload a transcript after a meeting, the system captures meeting context during the meeting itself. The output is therefore tied to participants, meeting identifiers, meeting status, timestamps, and room activity. Another strength is explain ability: because action and decision detection are rule-based, users can understand why a sentence was selected.

□ The main limitation is that rule-based NLP cannot understand every possible sentence structure. For example, indirect commitments such as 'the deployment report should be ready soon' may not be assigned to the correct owner. Similarly, decision extraction may miss implied decisions that do not contain clear decision markers. Browser speech recognition quality can also vary depending on microphone quality, accent, background noise, browser support, and network conditions. Future versions can address these limitations through advanced ASR, speaker diarization, named entity recognition, dependency parsing, transformer-based summarization, and human-in-the-loop editing.

□ Overall, the project achieves its intended purpose as an automatic meeting minutes generation platform. It reduces manual documentation effort, improves meeting traceability, and provides a strong base for more advanced AI-assisted meeting intelligence.

5. CONCLUSION

The Automatic Meeting Minutes Generation project provides a practical solution for converting live meeting conversations into structured, exportable minutes. By integrating authentication, meeting scheduling, live room management, Web RTC communication, Web Socket synchronization, speech capture, transcript storage, NLP-based extraction, and document export, the system covers the full workflow of modern meeting documentation.

The research shows that useful meeting minutes do not depend only on transcription. They require participant-aware organization, action item identification, decision highlighting, summary generation, and readable output formatting.

The Meet Minutes implementation addresses these needs through a transparent service-based architecture and a lightweight NLP pipeline. The generated minutes include overall summary, decisions, key points, action items, speaker summaries, person-to-person summaries, and transcript-based evidence.

The system is especially suitable for academic project demonstrations, small teams, classroom discussions, project reviews, and remote collaboration scenarios. Its modular architecture allows future enhancement with machine learning models, cloud speech recognition, advanced diarization, semantic search, calendar integration, role-based access control, and analytics dashboards.

In conclusion, the project successfully demonstrates an end-to-end approach to automatic meeting minutes generation. It reduces manual effort, improves accountability, and transforms unstructured meeting dialogue into organized knowledge that can be reviewed, shared, and archived.

Key Takeaways

1. Automatic meeting minutes generation is more than speech-to-text; it requires summarization, extraction, speaker attribution, and document formatting.
2. The Meet Minutes project provides an end-to-end workflow from meeting scheduling to live participation and summary export.
3. The system uses Java backend services, a browser-based frontend, WebRTC media, WebSocket synchronization, and transcript-based NLP processing.
4. Action items are extracted using future-work and task-oriented markers, while decisions are detected through agreement and approval markers.
5. Speaker-wise summaries improve accountability by showing contribution counts, highlights, action counts, and decision counts.
6. Person-to-person summaries provide insight into dialogue flow and collaboration patterns.
7. Rule-based NLP is transparent and easy to customize, but future versions can improve accuracy with transformer models and advanced speech recognition.
8. The architecture is modular, making it suitable for extension into enterprise meeting intelligence, analytics, and searchable knowledge repositories.

6. REFERENCES

1. Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., and Wellner, P. (2005). The AMI Meeting Corpus: A pre-announcement. Machine Learning for Multimodal Interaction.
2. McCowan, I., Carletta, J., Kraaij, W., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., Post, W., Reidsma, D., and Wellner, P. (2005). The AMI Meeting Corpus. International Conference on Methods and Techniques in Behavioral Research.
3. Murray, G., Renals, S., and Carletta, J. (2005). Extractive summarization of meeting recordings. Proceedings of Interspeech.
4. Xie, S., and Liu, Y. (2008). Using corpus and knowledge-based similarity measure in maximum marginal relevance

for meeting summarization. IEEE International Conference on Acoustics, Speech and Signal Processing.

5. Purver, M., Ehlen, P., and Niekrasz, J. (2007). Detecting action items in multi-party meetings: Annotation and initial experiments. Proceedings of the SIGdial Workshop on Discourse and Dialogue.
6. Janin, A., Baron, D., Edwards, J., Ellis, D., Gelbart, D., Morgan, N., Peskin, B., Pfau, T., Shriberg, E., Stolcke, A., and Wooters, C. (2003). The ICSI Meeting Corpus. Proceedings of ICASSP.
7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.
8. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. ACL.
9. Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. ICML.
10. Jurafsky, D., and Martin, J. H. (2023). Speech and Language Processing. Draft textbook, Stanford University.
11. Oracle. Oracle Database Documentation. Oracle Corporation.
12. Mozilla Developer Network. WebRTC API and Web Speech API documentation.