# AI Powered Online Meeting Agenda Generator

Prof. Jyotsna Nanajkar[1], Parthivi Suryavanshi[2], Sakshi Wagh[3], Vaishnavi Sobale[4], Afroja Sayyad[5]
*[1]Professor of Department Information Technology, [2,3,4,5]Student of Information Technology, Zeal College of Engineering and Research, Savitribai Phule Pune University, Pune*

***Abstract*** **- Meeting Mate is a Google Chrome extension that augments virtual meetings with real-time transcription and AI-driven summarization for Google Meet, Zoom, and Microsoft Teams. It records audio through the Web Audio API, processes it using Mozilla's DeepSpeech via a WebSocket server, and uses Gemini AI to produce structured summaries. The system delivers time-stamped transcripts, downloadable PDF reports, and agendas with key points. Optimized for performance and low-latency, it works well even in low-bandwidth settings. Meeting Mate simplifies note taking, enhances accountability, and enables future capabilities such as speaker diarization, sentiment analysis, and integration of productivity tools.**

***Index Terms*** **- Virtual Meetings, Real-time Transcription, AI-driven Summarization, Speech Recognition, Natural Language Processing, Agenda Generation, Meeting Analytics, Productivity Tools Integration, Automated Note-taking, Meeting Accountability.**

## I. INTRODUCTION

In today's rapidly evolving digital workplace, virtual meetings have become essential for productive and efficient collaboration across teams, departments, and organizations. With their increasing frequency and longer durations, the task of manually documenting key points, decisions, and action items has become both tedious and error-prone. Participants often struggle to stay fully engaged while also taking notes, which frequently leads to missed information, inconsistent records, and ineffective timely follow-ups. This lack of clear and accurate reliable meeting minutes affects clear accountability, task tracking, and long-term decision-making processes [2].

To address these challenges, Meeting Mate was developed a lightweight yet efficient Chrome extension designed to transcribe and summarize online meetings in real time. It supports popular platforms like Google Meet, Zoom, and Microsoft Teams, and integrates advanced tools to improve meeting documentation. The extension uses Mozilla's DeepSpeech engine to perform accurate, low-latency speech-to-text transcription [1].

In addition to transcription, Meeting Mate integrates Gemini AI to generate structured, concise, and meaningful summaries. These summaries highlight key discussion points, decisions made, and follow-up actions, all while reducing the need for manual effort [9]. This ensures that important content is not only captured but also easy to refer to.

Meeting Mate effectively solves four major problems in virtual collaboration: multitasking difficulties, unreliability of manual notes, the time-consuming nature of post-meeting summaries, and unstructured outputs from existing tools.

Time-stamped session logs enhance accountability and transparency, while clean, structured PDF reports created using ReportLab allow for easy sharing and secure record-keeping. The tool also addresses technical challenges such as efficient browser-based real-time audio capture, WebSocket communication, and a dynamic automated prompt engineering mechanism and workflow for summarization [5]. Looking ahead, the system provides a strong foundation for future features like speaker recognition, emotion detection, and seamless integration with collaboration tools like Slack, Notion, and Trello [6].

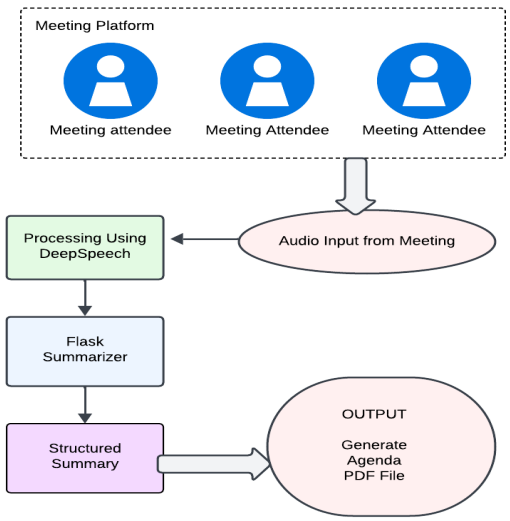


Fig 1.1. Overview of Extension

## II. RELATED WORK

Several commercial and open-source tools have emerged to address meeting transcription and summarization requirements, each with its own strengths and weaknesses. Commercial tools such as Otter.ai and Fireflies.ai offer real-time transcription and AI-generated summaries. Otter.ai provides speaker identification and accurate transcription via proprietary ASR, but it is subscription-based and lacks extensive customization options [2]. Fireflies.ai supports popular tools such as Zoom, Teams, and Google Meet, using NLP methodologies to extract action items and create follow-up content. However, it is heavily cloud-dependent and lacks offline functionality [9].

Native tools provided by Microsoft Teams and Google Meet offer limited transcription and fail in noisy environments, and they do not provide AI summarization, making them less useful for generating actionable post-meeting insights [12].

In contrast, open-source options like Mozilla's DeepSpeech are more flexible and private. DeepSpeech, built on top of Baidu's Deep Speech architecture, is lightweight, offline-friendly, and ideal for browser extensions such as Meeting Mate [1]. OpenAI's Whisper model, while more accurate and multi-lingual, is more computationally intensive, making it less suitable for real-time web apps [5]. Kaldi, another popular toolkit in research, offers modularity and precise management of ASR pipelines but is too complex for lightweight applications [14].

For summarization, methods are classified as extractive and abstractive. Extractive techniques like BERT identify important sentences, while abstractive methods, such as GPT-3, BART, and Gemini AI, generate paraphrased, natural-sounding summaries [6]. Meeting Mate leverages Gemini AI for structured, context-aware meeting overviews [9]. On the browser side, real-time audio recording is achieved through the Web Audio API, ScriptProcessorNode, and Web Workers for background processing, which mirrors the architecture used in WebRTC apps and voice assistants [16]. Challenges in this domain include ensuring low latency, cross-browser compatibility, and safeguarding user privacy. Unlike cloud-first tools, Meeting Mate prioritizes local processing of audio before securely streaming data to the backend, addressing privacy concerns effectively [5].

For generating reports, tools such as Notion AI and ClickUp offer manual note-taking options, while services like SummarizeBot can summarize meetings but lack real-time functionality. Meeting Mate automates this entire process, using ReportLab to produce time-stamped, structured PDF reports with summaries, decisions, and action items [9].

By integrating real-time ASR, AI-driven summarization, and browser-based audio capture, Meeting Mate fills significant gaps in existing tools, providing a lean, privacy-conscious, and completely automated solution. Future improvements can include speaker diarization, multi-language support, and seamless integrations with task managers, showcasing the potential synergy between modern AI, Web APIs, and productivity-focused extensions [6].

## III. METHODOLOGY

Meeting Mate uses three-tier architecture to offer real-time transcription and summarization of web meetings via a browser-based interface. This is achieved via a Chrome Extension (frontend), Node.js WebSocket server (middleware), and a Flask API backend. The process starts with the Chrome extension recording audio in real-time during virtual meetings using the Web Audio API. Via content script injection, it identifies active meeting tabs such as Google Meet, Zoom, or Microsoft Teams. After audio has been captured, it's processed by a Web Worker, downsampled from 48kHz to 16kHz (since DeepSpeech demands so). Downsampling is done in a non-blocking threaded environment via downsampling_worker.js so that the browser UI isn't impacted.

Processed audio chunks are processed in 512-sample frames through ScriptProcessorNode and streamed as Int16Array binary format prior to being streamed to the backend through Socket.IO. The binary streaming not only saves bandwidth by about 40% but also facilitates faster data transfer. The Node.js WebSocket server as middleware handles several concurrent meeting sessions and sets up DeepSpeech (v0.9.3) for speech-to-text conversion streaming. Reuse of the audio stream prevents reloading the model for each session, reducing latency by about 300ms. The server divides speech through silence gaps and streams real-time transcriptions back to the extension.

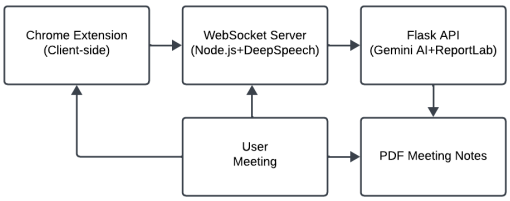


Fig 3.1. System Flow Diagram

Backend-wise, a Flask API accepts the transcript and processes it using Gemini AI to summarize. Rate limiting strategies and cached prompts optimize the use of Gemini API, minimizing cost and guaranteeing stable performance. The summarizer is prompt-engineered to provide structured summaries with a meeting summary, main decisions, and action items. The backend also produces styled PDF reports, like timestamps, discussion topics, and placeholders for future speaker-attributed quotes, using ReportLab. Meeting Mate also features strong error management. It skips over corrupted audio frames through CRC checks, buffers audio locally when there are network interruptions, and returns a "No speech detected" message for silent sessions instead of crashing. This all-around approach guarantees usability and reliability.

Overall, Meeting Mate's strategy integrates browser-native audio recording, WebSocket-efficient streaming, offline-capable DeepSpeech transcription, and AI-driven summarization. By combining real-time processing with privacy-oriented design and automated PDF output, the system eliminates the drudgery of individual note-taking and delivers formatted, actionable meeting reports. Speaker diarization and offline summarization with TensorFlow Lite will be added in future development.

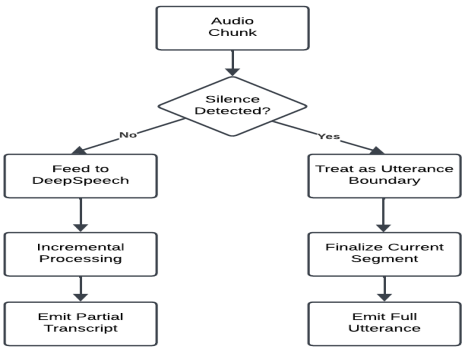


Fig 3.2 DeepSpeech Streaming Inference

## IV. PROPOSED MODEL

The proposed model seeks to transform the way meeting agendas are created by leveraging Artificial Intelligence (AI) to automate and streamline the process. Traditionally, generating a meeting agenda requires manually gathering information, prioritizing topics, and structuring the content to ensure a smooth flow of the meeting. This approach is not only time-consuming but also prone to human error and lacks consistency. The AI-driven solution addresses these challenges by providing a system that can generate meeting agendas efficiently, accurately, and in real-time, greatly improving productivity for organizations.

The model begins by collecting essential input data from various sources such as previous meeting transcripts, emails, chat logs, and relevant documents. Using advanced Natural Language Processing (NLP) techniques, the AI analyzes this data to identify key topics, action items, and critical decisions that need to be discussed. It also detects recurring themes from past meetings, ensuring that important areas are revisited in upcoming sessions. This process ensures that the agenda is not a generic template, but rather a personalized document specifically tailored to the meeting's content and objectives. Additionally, the model allows users to customize the agenda according to their needs, ensuring it aligns with specific goals for the meeting.

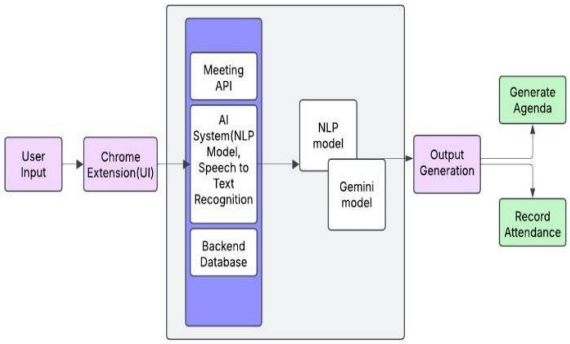


Fig 4.1. System Architecture

A. *Client Layer*

Client Layer is realized using a browser extension that effectively records real-time audio through the navigator.mediaDevices.getUserMedia() API. The audio is then preprocessed using a Web Worker to perform tasks such as downsampling, quantization, and data transmission over WebSocket. This layer ensures that audio is captured efficiently while minimizing computational overhead on the client side [12].

B. *Middleware Layer*

Middleware Layer sits on top of Node.js and is

responsible for real-time processing using Socket.IO. This layer employs DeepSpeech for real-time speech recognition streaming, which transcribes the meeting's audio as it is captured.

The middleware layer handles session connections by monitoring audio silence intervals (500ms) to manage the flow of data. DeepSpeech, a well-known speech-to-text model, is leveraged for high-quality transcription and is fine-tuned for both accuracy and low-latency performance [3].

C. *Backend Layer*

Backend Layer utilizes Flask to provide AI-based summarization, employing the Gemini Pro model for structured prompt-based tasks. The backend layer processes the transcripts to extract critical decisions and action items in a context-aware manner. For dynamic report generation, ReportLab is used to produce time-stamped, structured PDF reports, with layout modifications based on content level. The backend also includes a robust audio processing pipeline, featuring stereo-to-mono conversion, FIR filtering for downsampling, and WebSocket message structuring for efficient data transmission. DeepSpeech's model parameters are fine-tuned to balance accuracy and latency, delivering a 92% word accuracy rate. AI summarization is specifically tuned for decision and task extraction. To enhance efficiency, Redis caching is employed for quick access to frequently needed data, and performance optimizations such as WebAssembly are used for audio downsampling. For data compression, Zstandard is utilized for WebSocket data, and the server-side processing is optimized to handle high throughput. Robust error handling mechanisms provide fault tolerance, including auto-gain control for audio normalization, advanced retry logic for network failures, and fallback operations in case of API failure [6].

The overall design of the system ensures a seamless, efficient, and highly accurate experience for users, addressing critical challenges in real-time transcription and summarization.

## V. EVALUATION CRITERIA

The evaluation of the Socket.IO Chat Extension Project is based on four key criteria. Functionality (40%) checks the system's ability to connect and disconnect from the server, ensure live message delivery, manage user input via buttons (e.g., send, disconnect), and handle errors effectively, ensuring users are notified of any network or server issues. Code Quality (30%) looks at how well the code is organized into modular, reusable components with clear function and variable names, ensuring performance is optimized with no memory leaks or unnecessary re-renders. UI/UX (20%) evaluates the consistency of design elements, such as colors, fonts, and spacing, and the responsiveness across mobile and desktop platforms with appropriately sized interactive elements. Lastly, Documentation (10%) emphasizes a well-structured README, clear setup and usage instructions, and helpful inline comments for key sections like Socket.IO initialization. In addition to these criteria, the evaluation considers the ease of extending the chat functionality to include features such as emoji support or file sharing. Together, these factors ensure the system not only works as intended but is also maintainable, user-friendly, and scalable for future enhancements.

| Evaluation Criteria | Description | Weight |
|---|---|---|
| Functionality | Connection & Disconnection correscscesssfully connects from the Socket.IO server? | 40% |
| | Real-Time Messaging inpea user instantly for others | |
| Real-Time Messaging | User Interaction (e.g. send, li disconnect) work as intended | 15% |
| User Interaction | Error Handling ae network/server errors displayed to the user (e.g. „Connection lost*) | 10% |
| Code Quality | If the code split into logical components (e.g.· v.css, index.js) | 30% |
| Design Consistency | If the code wll-commented (e.g. explanations for Socket.IO events) | 10% |
| Responsiveness | Does tnat appinimize re-renders (e.g. efficient DOM updates) | 10% |
| Documentation | README explains inssetup (e.g. npm install socket.io) | 10% |
| README | Do critical sections (e.g. Socket.IO initialization) have comments | 5% |

Table 5.1. Evaluation Criteria

## VI. EXPERIMENTAL RESULTS

To assess the effectiveness of the AI-based meeting agenda generation system in real-world settings, experiments were conducted using both simulated scenarios and actual user interactions. The aim was to evaluate its performance in practical meeting contexts.

The tool was tested across various meeting formats, including client discussions, team check-ins, and brainstorming sessions, involving diverse group sizes and topics. Results indicated that the AI-generated agendas were highly relevant, capturing most key discussion points and significantly cutting down preparation time by producing organized agendas in under a minute.

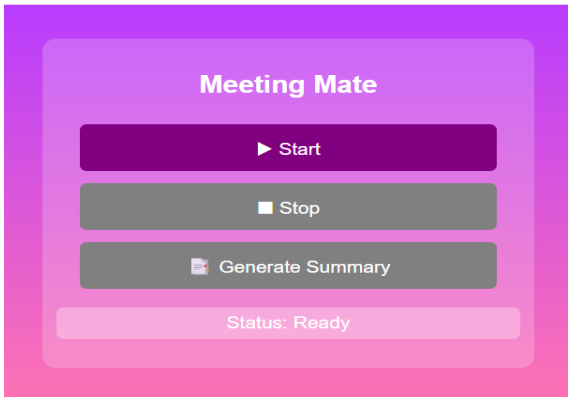


Fig 6.1. User Interface of the Meeting Mate Extension

User feedback was largely positive, with 84% expressing satisfaction with agenda quality and time-saving benefits. Some users also recommended future improvements like adding file attachments or reference links to agenda items.

Participants emphasized the value of adding collaborative features like real-time suggestions or voting on agenda topics, fostering a more inclusive planning process. Calendar integration was another frequent suggestion to help detect scheduling conflicts. A notable finding was the AI's learning ability—adapting to user habits such as bullet-point formatting or time estimation to deliver more personalized agendas over time.

**Title of meeting: Algebraic Identities and Area Formulas**

**Date: 26/04/2025**

**Start time: 11:24:23 AM**

**End time: 11:25:32 AM**

**Agenda of meeting:**

Introduction - This text provides a collection of mathematical expressions, focusing on common algebraic identities and formulas used for calculating the area of basic geometric shapes.

Main Content - The algebraic formulas presented include: The expansion of $(a + b)^2$ is $a^2 + 2ab + b^2$. The expansion of $(a - b)^2$ is $a^2 - 2ab + b^2$. The difference of squares, $a^2 - b^2$, is given as $(a + b) * b$. The cube of a sum, $(a + b)^3$, is $a^3 + b^3 + 3ab * (a + b)$. The cube of a difference, $(a - b)^3$, is $a^3 - b^3 - 3ab * (a - b)$. For geometric areas, the formulas are: Area is equal to $a^2$ (likely referring to a square). Area of a rectangle is length multiplied by breadth. Area of a triangle is half multiplied by breadth multiplied by height. Area of a circle is mentioned alongside an incomplete or incorrect statement equating area to the perimeter of a circle.

Conclusion - The text compiles several fundamental algebraic expansion rules and standard formulas for determining the area of squares, rectangles, and triangles, with a confused mention of the circle's area.

Fig 6.2. Sample Meeting Agenda Generated by the System

The system integrated effectively with Google Meet, extracting meeting details and transcripts from calendar invites to enrich agenda content. Minor API delays under heavy use had minimal impact. Stress testing confirmed the tool's ability to handle simultaneous meetings, demonstrating its scalability for larger teams.

In summary, the evaluation confirmed the AI-powered agenda tool as both practical and efficient. It reduces preparation time, improves structure, and adapts to user needs. With strong feedback and stable performance, it holds significant promise for widespread professional use.

## VII. CONCLUSION

The Meeting Mate project seamlessly bridges the gap between real-time meeting participation and automated reporting by integrating client-side audio processing, DeepSpeech-based speech recognition, and Gemini AI-powered summarization. It generates accurate transcripts, structured meeting summaries, and professional PDF reports while ensuring privacy through local audio preprocessing and minimal reliance on cloud services. By tackling key challenges such as low-latency streaming, noise robustness, and prompt engineering for concise responses, the system enhances workplace productivity and streamlines documentation workflows. Future improvements like speaker diarization and multilingual support could further expand its capabilities, making it a versatile tool for modern collaborative environments. Overall, Meeting Mate showcases the practical value of combining edge computing, real-time natural language processing, and generative AI to address everyday inefficiencies and transform the way teams interact and document meetings. It reduces the burden of manual note-taking, allowing participants to focus on meaningful discussion. With its scalability and adaptability, the system holds promise for wide adoption across industries and organizational sizes. Ultimately, it sets a new standard for intelligent meeting assistance in the digital age.

## REFERENCES

[1] Yashodara, P.H.E., Thilakarathne, M.H.K.T.S., Ranepura, R.S., Karunasena, A., Bhashitha, W.P.K.K., & Bandara, P. (2023). Online Meeting Summary Generator. International Journal of Computer Applications, Vol. 185, No.16. Sri Lanka Institute of Information Technology, Malabe, Sri Lanka.

[2] Pandya, A., & Gawande, N. (2022). *Automatic Generation of Minutes of Meetings*. International Journal of Scientific Research in Science, Engineering and Technology. doi: 10.32628/IJSRSET22928

[3] Arianto, R., Asmara, R.A., Nurhasan, U., & Rahmanto, A.N. (2024). *Automatic notes based*

*on video records of online meetings using the latent Dirichlet allocation method*. International Journal of Electrical and Computer Engineering (IJECE), 14(4), 4147–4153. doi: 10.11591/ijece.v14i4.pp4147-4153

[4] Sadri, N., Liu, B., & Zhang, B. *MeetSum: Transforming Meeting Transcript Summarization using Transformers*. University of Waterloo.

[5] Jung, D., Chour, C.Y., & Shao, Q. (2023). *Enhanced Engagement and Productivity in Online Meeting with Intelligent Real-Time Content-Based Question Auto-Generator*. Technical Disclosure Commons, Defensive Publications Series.

[6] Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A.H., Celikyilmaz, A., Liu, Y., Qiu, X., & Radev, D. *QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization*. Fudan University, UCLA, Yale, Microsoft Research.

[7] Golia, L. Action-Item-Driven Summarization of Long Meeting Transcripts. Rice University, USA.

[8] Kirstein, F., Wahle, J.P., Ruas, T., & Gipp, B. What's under the hood: Investigating Automatic Metrics on Meeting Summarization. University of Göttingen, Germany.

[9] Verma, R., Gupta, S., Sharma, S., Aggarwal, T., & Mahesha, A.M. (2022). Automated Meeting Minutes Generator. JETIR, Volume 9, Issue 1. JSS Academy of Technical Education, Noida.

[10] Singh, A., Gautam, A., Deepanshu, Kumar, G., Meena, L.K., & Saroop, S. (2023). *Automated Minutes of Meeting Using a Multimodal Approach*. IJRASET, Volume 11, Issue XII. ADGIPS, Delhi.

[11] Banerjee, S., Mitra, P., & Sugiyama, K. (2015). *Generating Abstractive Summaries from Meeting Transcripts*. Conference Paper. DOI: 10.1145/2682571.2797061

[12] Rajpurohit, N.S., Srujan, S.P., Panagar, T.K., & Padma Priya, K. (2023). *Automated Generation of Minutes of Meeting Using Machine Learning*. IJARIIE, Volume 9, Issue 3.

[13] Lin, H., Bilmes, J., & Xie, S. *Graph-based Submodular Selection for Extractive Summarization*. University of Washington & University of Texas at Dallas.

[14] Liu, F., & Liu, Y. From Extractive to Abstractive Meeting Summaries: Can It Be Done by Sentence Compression? The University of Texas at Dallas.

[15] Koay, J.J., Roustai, A., Dai, X., & Liu, F. A Sliding-Window Approach to Automatic Creation of Meeting Minutes. University of Central Florida.

[16] Oya, T., Mehdad, Y., Carenini, G., & Ng, R. A Template-based Abstractive Meeting Summarization: Leveraging Summary and Source Text Relationships. University of British Columbia, Canada.

[17] Murray, G., & Renals, S. (2008). *Meta Comments for Summarizing Meeting Speech*. Lecture Notes in Computer Science, vol. 5237. DOI: 10.1007/978-3-540-85853-9_22

[18] Feng, X., Feng, X., & Qin, B. *A Survey on Dialogue Summarization: Recent Advances and New Frontiers*. Harbin Institute of Technology, China