Real-Time Meeting Intelligence and Task Automation Using Edge-based AI and LLM

Dr. R. Anitha[1], Shivam Manoj Shukla[2], C. Sujay[3], Sanjay Kumar S[4]
[1]*Professor, Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Chennai, India*
[2,3,4]*Undergraduate Students, Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Chennai, India*

***Abstract*—In today's professional environment, virtual meetings have become an essential part of communication and teamwork. However, participants often struggle to capture important points, track decisions, and remember assigned tasks during continuous discussions. Traditional meeting assistant platforms mainly depend on cloud-based services for speech processing and analysis, which may cause delays, raise privacy concerns, and require constant internet connectivity. This paper proposes an intelligent edge-based meeting assistant system that provides live transcription, contextual analysis, and automated task handling during meetings. The system integrates real-time speech recognition, a locally deployed Large Language Model (LLM), and an agent-driven automation framework to understand ongoing conversations and respond intelligently. Based on the meeting context, the system can automatically generate summaries, extract action items, send follow-up emails, and perform information retrieval tasks. Unlike conventional cloud-centric approaches, the proposed framework performs most computations directly on edge devices. This reduces latency, enhances data security, and limits the transfer of sensitive meeting information to external servers. The modular architecture also allows easy scalability and integration with different collaboration platforms. Experimental evaluation demonstrates that the proposed system delivers accurate real-time transcription and efficient meeting assistance with minimal delay. Comparative analysis with cloud-based solutions highlights the advantages of the edge-oriented design in terms of responsiveness, privacy, and operational efficiency.**

***Index Terms*—Edge Computing, Agentic AI, Large Language Models, Meeting Intelligence, Real-Time Transcription, VOSK, LLaMA, Ollama, Speech Recognition, Task Automation**

I. INTRODUCTION

Meetings play a vital role in modern organizations by supporting communication, collaboration, and decision-making among teams. However, managing the large amount of information generated during meetings is often challenging. Participants are expected to actively contribute to discussions while also remembering important decisions, tracking responsibilities, and recording follow-up actions. As a result, important details are frequently missed or documented inaccurately. Conventional methods such as handwritten notes or manual documentation are time-consuming and unreliable, especially in long or fast-paced meetings. To address these issues, automated meeting transcription systems have gained popularity in recent years. Although these systems help convert spoken conversations into text, most of them rely heavily on cloud-based infrastructure for processing and storage. This dependence on cloud services introduces several drawbacks, including network delays, privacy and security concerns, and increased operational costs due to continuous use of remote APIs and servers. In today's professional environments, there is a growing demand for intelligent systems that not only record meetings but also analyze discussions, identify tasks, monitor workflows, and improve productivity. Organizations require solutions capable of generating summaries, detecting important events, predicting potential issues, and assisting in task management automatically. The integration of edge computing, artificial intelligence, and smart automation technologies provides an effective way to achieve these objectives while maintaining faster response times and better data

privacy. This work focuses on developing an edge-based intelligent meeting assistant that combines local processing capabilities with advanced AI reasoning. The proposed system integrates real-time speech recognition, locally deployed Large Language Models (LLMs), and an agent-oriented automation framework within a unified architecture. By processing most operations directly on edge devices, the system reduces dependency on external cloud platforms and enables low-latency meeting assistance. Even though recent advancements in speech processing and AI have improved meeting automation technologies, there is still a lack of efficient integration between live transcription, contextual understanding, and autonomous task execution in edge environments. The proposed architecture addresses this gap through a modular and layered framework designed for real-time collaboration support.

The major contributions of this work include:

1. Designing a layered architecture for an edge-based intelligent meeting assistant.
2. Implementing a locally hosted LLM pipeline for contextual understanding and insight generation.
3. Developing an agent orchestration mechanism capable of identifying conversational triggers and executing automated tasks.
4. Evaluating the performance of edge-based processing in comparison with traditional cloud-based systems across latency, privacy, and efficiency metrics.

The overall objective of the proposed system is to provide a scalable, privacy-focused, and low-latency meeting intelligence solution that enhances productivity and improves the efficiency of professional collaboration.

II. RELATED WORK

The study of intelligent meeting assistant systems has gained significant attention in recent years due to the increasing demand for automated collaboration tools in professional environments. Researchers have explored this domain through various technologies such as Automatic Speech Recognition (ASR), Natural Language Processing (NLP), edge computing, and multi-agent artificial intelligence systems. Early research primarily focused on meeting transcription and structured documentation, providing the foundation for automated summarization and information extraction techniques [1]. With the advancement of cloud computing technologies, commercial platforms such as Otter.ai, Microsoft Teams Transcription, and Zoom AI Companion introduced real-time transcription and meeting assistance features for enterprise users. Although these systems improved accessibility and usability, they mainly depend on cloud-based infrastructures for processing. Such dependence creates several limitations, including increased network latency, continuous internet requirements, recurring service costs, and privacy risks caused by transmitting sensitive meeting conversations to external servers. Recent developments in edge computing and lightweight AI models have opened new possibilities for deploying intelligent applications directly on local devices. The emergence of compact and optimized language models such as Meta LLaMA and Mistral AI Mistral has demonstrated that advanced language understanding tasks can now be executed efficiently on commodity hardware. Local inference platforms like Ollama further support the deployment of LLMs in edge environments. Previous studies have also shown that edge-based speech recognition systems can provide acceptable transcription accuracy while reducing dependence on cloud services [4]. Another important advancement in this field is the integration of agentic AI frameworks into intelligent automation systems. The development of reasoning-based architectures such as React and frameworks including LangChain and AutoGen enabled AI agents to plan tasks, analyze context, and interact with external tools dynamically. These frameworks introduced autonomous decision-making capabilities where language models can perform multi-step operations using feedback loops. However, most existing implementations are designed for large-scale cloud environments and are not optimized for low-resource edge devices operating in real-time meeting scenarios. Commercial meeting intelligence solutions mainly focus on transcription and summarization functionalities. Earlier research explored systems capable of identifying entities, decisions, and action items from live conversations to automate meeting documentation. These approaches contributed significantly toward reducing manual effort in professional workflow management. Some prior works also proposed digital knowledge

representations for meeting content, where extracted information such as participant notes, assigned tasks, and schedules could be stored and retrieved dynamically during collaboration sessions. Despite these advancements, there remains a lack of unified architectures that effectively combine streaming speech recognition, local LLM reasoning, and autonomous agent orchestration within a lightweight edge-computing framework. Existing systems often address these components independently rather than integrating them into a cohesive real-time solution. The proposed work addresses this research gap by presenting a modular edge-based meeting intelligence architecture that combines real-time ASR, edge-local LLM inference, and agent-driven automation into a single deployable framework. The system utilizes open communication standards and scalable design principles to ensure compatibility with modern collaborative environments while maintaining low latency, enhanced privacy, and reduced cloud dependency.

III. SYSTEM METHODOLOGY

The proposed system architecture is designed as a modular layered framework consisting of four interconnected sections, as illustrated in Figure 1. Each layer is responsible for a specific set of operations and communicates with neighbouring components through structured interfaces. Together, these layers create a complete real-time processing pipeline that captures live meeting audio, generates transcripts, performs AI-based analysis, and executes intelligent automated actions.

As shown in Figure 1, the architecture is divided into four primary sections:

1. The Edge Layer, responsible for audio acquisition and user interaction.
2. The Core OS Layer, which manages communication and data routing.
3. The Live Transcription Pipeline and Agentic AI Brain, forming the central intelligence engine of the system.
4. The Agentic Capabilities and Output Layer, which handles automation services and external integrations.

The Edge Layer functions directly on the user's local device and acts as the front-end interface of the system. This layer captures live meeting audio through the microphone and streams it continuously for processing. To achieve low-latency performance, the incoming audio is divided into small chunks, generally between 250 ms and 500 ms in duration. These smaller audio segments are transmitted in real time to the backend through a persistent WebSocket communication channel. Streaming audio in this manner allows the system to produce partial transcriptions almost instantly during conversations. In addition to audio capture, the Edge Layer hosts a browser-based dashboard that displays live transcripts, AI-generated suggestions, meeting insights, and overall system status. Users can monitor ongoing processing activities through this interface and may also provide prompts or custom instructions to the AI assistant whenever required. Figure 2 illustrates the real-time transcript visualization within the user interface.

The Core OS Layer serves as the communication backbone of the entire architecture. It manages data exchange between all major modules and ensures smooth real-time interaction across the system. A WebSocket-based gateway maintains continuous bidirectional communication between the client device and backend processing services. Through this persistent connection, audio chunks are forwarded from the Edge Layer to the transcription engine, while generated transcripts and status responses are returned to the frontend dashboard. Unlike traditional request-response communication models, persistent WebSocket connections significantly reduce latency by eliminating repeated HTTP connection overhead. An additional component within this layer is the Express API Gateway, which processes structured frontend requests and coordinates backend services. The gateway manages operations such as routing audio streams, handling user requests, forwarding prompts to the AI reasoning module, and securely communicating with storage components. It also provides an abstraction layer that prevents sensitive backend APIs and internal processing logic from being directly exposed to end users.

The Live Transcription Pipeline represents the first stage of the system's intelligence module. Its primary responsibility is to transform incoming speech into readable and time-synchronized text suitable for downstream AI analysis. The architecture utilizes VOSK as the core Automatic Speech Recognition

(ASR) engine because of its lightweight design and offline processing capabilities. Audio chunks received from the Core OS Layer are continuously processed by VOSK, which produces both partial and finalized transcription outputs in real time. Since transcription occurs locally, sensitive audio information never needs to be transmitted to external cloud servers. To improve transcript consistency across streaming segments, a dedicated Transcription Buffer is integrated into the pipeline. This component temporarily stores intermediate transcription outputs and maintains contextual continuity between consecutive audio chunks. Once a transcript segment is finalized, it is simultaneously forwarded to the Agentic AI Brain for contextual analysis and stored within the data persistence layer for future retrieval. This parallel-processing design enables AI reasoning and data storage to occur concurrently, reducing system bottlenecks and improving overall efficiency.

The Agentic AI Brain constitutes the cognitive engine of the system. A locally deployed LLM specifically LLaMA running via the Ollama inference runtime processes final transcripts and contextual prompts. Running the model entirely on-device ensures that no conversational data leaves the local environment. The Agent Orchestrator receives parsed responses from the Local LLM and determines whether additional actions are required.

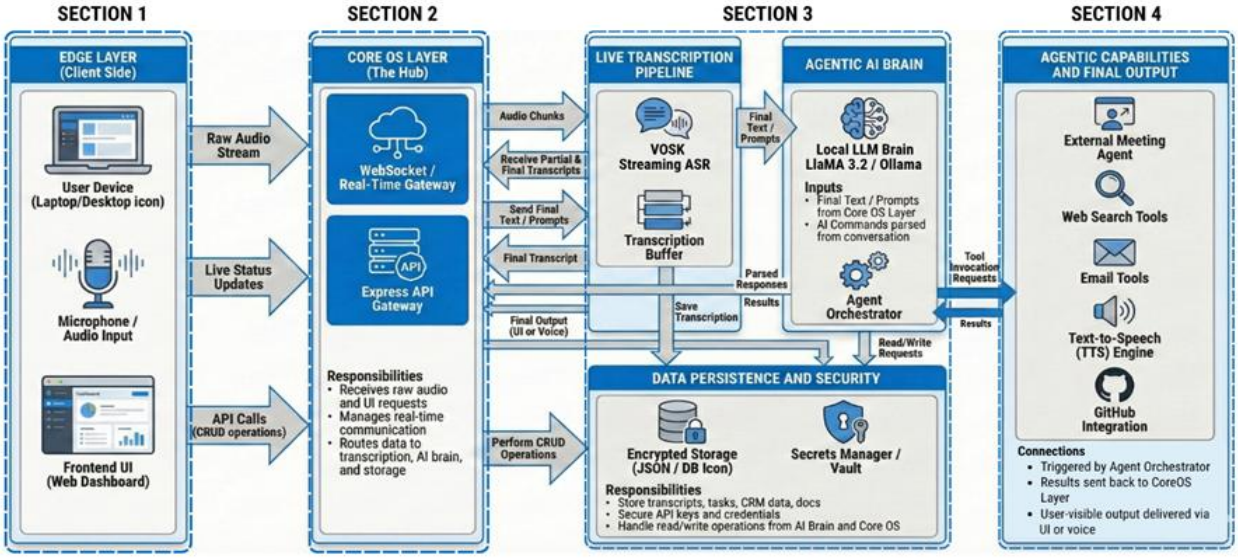

Figure 1. End to End Architecture of the Edge-Based Agentic Meeting Assistant

IV. DESIGN OF THE SYSTEM

4.1. VOSK Streaming ASR

VOSK is an offline speech recognition framework widely used for real-time transcription applications on desktop and edge devices. It supports continuous speech processing with low computational requirements, making it suitable for real-time meeting intelligence systems. The toolkit performs speech recognition using acoustic and language modelling techniques and supports customizable vocabularies for domain-specific applications. In the proposed system, the VOSK streaming API was configured with a 16 kHz mono-channel audio input, which matches standard microphone capture settings commonly used in conferencing environments. To improve recognition performance in professional meeting scenarios, the vocabulary and language model were optimized for technical and business-oriented English terms frequently used in enterprise discussions. For real-time processing, incoming audio was divided into streaming chunks of approximately 400 milliseconds. This chunk size was selected to maintain a balance between low-latency response and transcription stability. As audio data is processed continuously, the transcription engine generates both partial and finalized text outputs in real time, allowing users to follow conversations with minimal delay. The streaming approach also enables synchronization between spoken dialogue and displayed transcripts within the meeting dashboard.

4.2. Backend Communication Framework (Express + WebSocket)

The backend infrastructure of the proposed system was developed using Express.js and WebSocket-based communication services to support low-latency real-time interaction. The backend acts as the central coordination layer responsible for managing communication between the frontend interface, transcription engine, AI reasoning modules, and external services. A persistent WebSocket connection is used to stream audio data and system responses continuously between the client device and backend modules. This approach avoids the repeated overhead associated with conventional HTTP request-response communication and significantly improves real-time performance. The Express.js API gateway manages structured requests from the frontend dashboard, including transcript retrieval, AI prompt handling, session management, and tool invocation requests. The gateway also routes finalized transcripts to the AI processing layer and storage modules while maintaining secure communication between system components. To ensure secure operation, authentication and credential validation mechanisms are integrated into the backend infrastructure. Sensitive configuration data and API credentials are protected using secure storage and secret management techniques. This design prevents unauthorized access and minimizes exposure of backend processing logic to end users.

4.3. Local LLM Integration Using LLaMA via Ollama

The contextual intelligence component of the system is powered by locally deployed Large Language Models using Ollama and Meta LLaMA models. Local deployment enables AI inference directly on edge devices without depending entirely on external cloud-based AI services. This improves privacy, reduces latency, and lowers operational costs. The LLM module is responsible for performing advanced contextual analysis on meeting transcripts. It generates meeting summaries, identifies action items, extracts decisions, analyzes intent and sentiment, and produces structured outputs for downstream automation. The system was designed to support multiple output formats, including bullet-point summaries, categorized task lists, JSON-based structured responses, and contextual recommendations. Prompt engineering techniques were applied to guide the LLM toward generating consistent and structured outputs. For example:

- Meeting summaries were formatted as concise bullet points.
- Task extraction outputs included assignee names and deadlines.
- Action triggers followed predefined JSON schemas for easier orchestration.

To support multiple concurrent meeting sessions, each session is assigned a unique identifier token. These identifiers allow the orchestration layer to maintain contextual separation between parallel meetings and associated users. The generated outputs are visualized directly within the meeting dashboard, enabling users to monitor summaries, decisions, and AI-generated insights in real time. One major advantage of this approach is its simplicity and scalability. Since meeting sessions are managed through lightweight identifiers, the system can operate efficiently without requiring specialized hardware or complex infrastructure. Any modern computing device capable of running the local AI stack can host the proposed application.

4.4. Proposed Meeting Intelligence Architecture

The proposed meeting intelligence framework follows an edge-oriented and IoT-inspired architecture where multiple software components interact through lightweight communication interfaces. The architecture combines streaming speech recognition, local AI reasoning, and autonomous tool orchestration into a unified real-time collaboration system.

At the core of the architecture is the Agent Orchestrator, which acts as the central decision-making controller of the system. Similar to a relay mechanism in IoT systems, the orchestrator receives contextual information from the transcription and AI reasoning modules and determines which external tools or services should be activated. The orchestrator is responsible for handling higher-level reasoning tasks that extend beyond basic speech transcription. Based on meeting context and detected conversational triggers, it can invoke different automation modules such as Email generation, Task scheduling, Reminder creation, Web search operations, Knowledge retrieval services. Communication between the orchestrator and external modules is performed through predefined API interfaces, ensuring modularity and extensibility.

This design allows additional tools and automation capabilities to be integrated into the system without modifying the core transcription or AI processing pipeline. By combining edge computing, local LLM inference, and agent-driven automation, the proposed architecture provides a scalable and privacy-focused meeting intelligence solution capable of supporting real-time collaborative environments efficiently.

4.5. Performance Comparison: Cloud Vs. Edge

To contextualize the advantages of the proposed edge-based architecture, Table 1 provides a systematic comparison across ten key operational criteria against a representative cloud-based meeting assistant combining a cloud ASR service such as Google Speech-to-Text with a hosted LLM API.

Table 1. Performance Comparison: Cloud-Based vs. Edge-Based Meeting Assistant

Criterion	Cloud-Based System	Proposed Edge-Based System
Transcription Latency	800–2000ms (network-dependent)	80–250ms (local processing)
Data Privacy	Data sent to external servers; breach risk	All data on-device; no external transmission
Internet Dependency	Requires stable broadband; fails offline	Functional without internet connection
Infrastructure Cost	Per-minute/per-API pricing; scales with use	One-time setup; no recurring cloud charges
Scalability	Horizontally scalable via cloud provider	Limited by local HW; modular design aids scaling
LLM Processing	Cloud-hosted (GPT-4/Gemini); opaque	Local LLM (LLaMA/Ollama); private & customizable
Agent Capabilities	Dependent on external API services	Integrated orchestrator with direct tool invocation
Insight Quality	High (large-scale frontier models)	Comparable for summarization & task extraction
Setup Complexity	Low (SaaS subscription)	Moderate (requires local model deployment)
Compliance	Varies; depends on vendor agreements	High; GDPR/HIPAA-friendly by design

As shown in Table 1, the edge-based system provides substantially lower transcription latency (80–250 ms versus 800–2000 ms for cloud systems), owing to the elimination of network round-trips for audio transmission and inference. The privacy advantage of the edge approach is perhaps the most strategically significant. Cloud-based systems necessarily transmit meeting audio and transcripts to external infrastructure, creating exposure under data protection frameworks such as GDPR and HIPAA. The proposed system's entirely local processing model makes it inherently compliant with these frameworks without requiring additional contractual controls.

V. AGENT ORCHESTRATOR

The Agent Orchestrator serves as the core control and decision-making component of the proposed meeting intelligence system. It continuously operates in the background during meetings, receiving finalized transcript segments and contextual information from the Local LLM module. Based on the detected conversational intent, the orchestrator determines which automated action or external tool should be executed. The architecture is designed to support non-blocking operations so that transcription, reasoning, and task execution can occur simultaneously without interrupting the live meeting experience. The orchestration process begins whenever a finalized transcript segment is received along with the current conversation context. Before forwarding the data to the Local LLM Brain, the orchestrator checks the size of the active context buffer. If the stored conversation history exceeds the predefined token limit, older transcript segments are compressed into a shorter summary and removed from active memory. This context management mechanism ensures efficient memory usage and allows the system to support long-duration meetings without exceeding the processing limitations of the local language model.

5.1. Context Management and LLM-Based Reasoning

After validating the context window, the orchestrator prepares a structured prompt for the Local LLM Brain. This prompt combines the ongoing conversation history with predefined system instructions that specify:

- Role of the AI model
- Expected response structure
- Supported intent categories.

The prepared prompt is then submitted to the local language model for reasoning and contextual analysis. The model processes the transcript data and returns a structured response containing detected actionable intents. To simplify downstream processing, the output is generated in JSON format, where each object represents an intent along with its associated parameters and metadata. This structured reasoning approach allows the orchestrator to identify relevant actions automatically while maintaining consistency across different meeting scenarios.

5.2. Intent Detection and Tool Invocation

After receiving the structured intent list, the orchestrator processes each detected intent sequentially. The current implementation supports five primary intent categories. A SUMMARIZE intent instructs the system to generate a concise summary of the ongoing discussion. The summary is displayed directly within the meeting dashboard for participant review. A SEND_EMAIL intent triggers the email automation workflow. In this process, the orchestrator extracts the recipient address, subject details, and message content from the generated intent payload. The required credentials are validated through the secure credential manager before the Email Tool automatically composes and sends the message. For a WEB_SEARCH intent, the orchestrator extracts the user query from the conversation context and forwards it to the integrated web search module. The retrieved search results are then passed back to the Local LLM Brain for contextual summarization before being displayed to the user. This additional reasoning step helps present meaningful information related to the meeting discussion instead of raw search outputs. An ASSIGN_TASK intent enables automatic task management. The orchestrator extracts task details, responsible participants, deadlines, and related metadata from the transcript context and stores them in the encrypted storage system. If external integrations are enabled, the system can additionally create project management entries such as repository issues or workflow tickets automatically. A TTS_RESPONSE intent activates the Text-to-Speech module, allowing the AI assistant to provide spoken responses or alerts during the meeting session. This feature enables more interactive communication between users and the meeting assistant.

5.3. Data Persistence and Asynchronous Processing

All tool execution operations within the orchestration pipeline are performed asynchronously. This design choice is important because it prevents slow external operations from blocking the primary transcription and reasoning workflow. As a result, the system can continue processing live meeting audio while background tasks such as email delivery, web searches, or database updates are executed independently. Once all intent-processing tasks are completed, the finalized transcript segments are securely stored in the encrypted persistence layer. This guarantees reliable preservation of meeting data and ensures that no important conversational information is lost during long or complex sessions.

VI. RESULTS AND SYSTEM EVALUATION

A prototype implementation of the proposed architecture was deployed in a controlled meeting environment to evaluate system performance and usability. Test sessions included both one-on-one discussions and multi-participant team meetings lasting between 15 and 60 minutes.

6.1 Transcription Performance

To benchmark transcription accuracy, ten professionally phrased test utterances covering meeting-domain vocabulary were evaluated against both the Cloud STT system (Google Speech-to-Text) and the Local STT engine (VOSK). The primary accuracy metric used is Word Error Rate (WER), which is the standard evaluation measure in automatic speech recognition (ASR) research. WER is computed by comparing the system's output transcript word by word against a known reference transcript, counting the minimum number of word-level substitutions, deletions, and insertions required to convert the hypothesis into the reference.

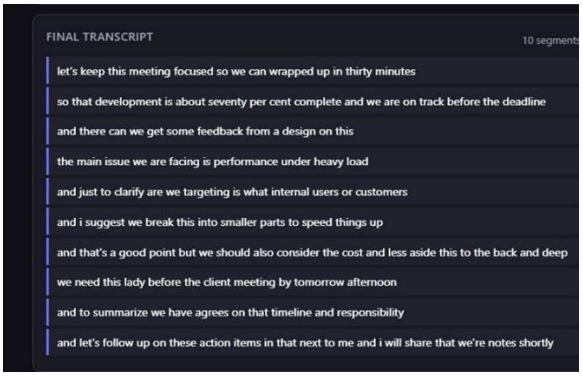

Figure 2. User Interface Displaying Real-Time Meeting Transcription and Segmentation

The frontend interface presents the four output panels through which the VOSK ASR engine surfaces transcription results in real time. Partial hypotheses generated by the acoustic model are streamed continuously into the Live Draft panel, providing immediate visual feedback as speech is recognised. Upon sentence completion, VOSK finalises the segment and the locked-in text is committed to the Final Transcript panel.

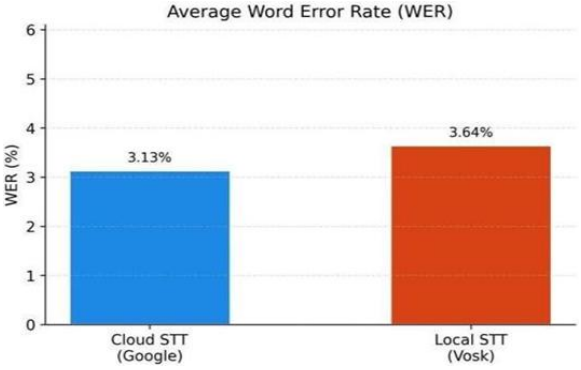

Fig. 4a: Average Word Error Rate (WER) Comparison

Figure 4a presents a bar chart comparing the average WER of both systems across all ten test utterances. Google Speech-to-Text achieved an average WER of 3.13%, while VOSK achieved 3.64%. Both values are exceptionally low, placing both systems well within the accuracy range considered acceptable for professional meeting transcription. The difference of 0.51 percentage points is negligible in practice. This result demonstrates that the locally-running VOSK model achieves nearly identical accuracy to a cloud-hosted service backed by Google's large-scale infrastructure. End-to-end transcription latency was measured as the total elapsed time from the moment an audio clip was submitted to the system until the final text transcript was returned.

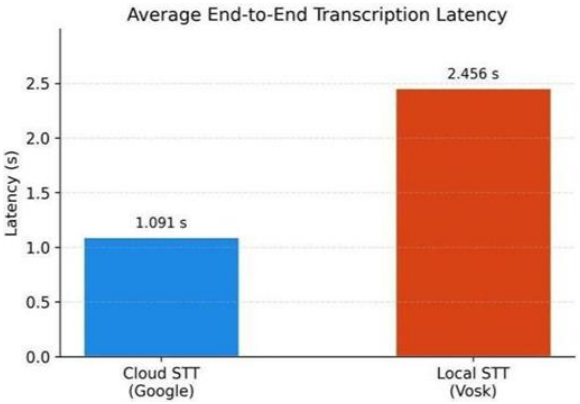

Fig. 4b: Average End-to-End Transcription Latency

Figure 4b presents a bar chart comparing the average latency of both systems. Google Speech-to-Text averaged 1.091 seconds per utterance, while VOSK averaged 2.456 seconds. Google achieves its lower latency by routing audio to optimized, GPU-accelerated servers. VOSK, running entirely on the local CPU, takes longer per utterance but eliminates any dependence on internet connectivity.

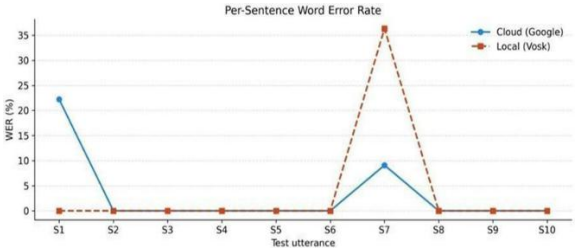

Figure 5. Per-Sentence Word Error Rate (WER) for All 10 Test Utterances (S1–S10)

Figure 5 provides a deeper per-sentence view of WER as a line plot. Both systems recorded a WER of 0% on the majority of sentences, indicating perfect transcription. The exceptions are S1, where Google recorded a 22% WER due to misrecognition of a time expression, and S7, where VOSK recorded a 36% WER on a sentence containing the technical term "large language model."

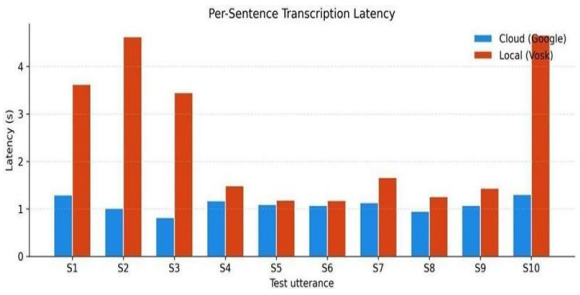

Figure 6. Per-Sentence Transcription Latency for All 10 Test Utterances (S1–S10). Google STT Latency Remains Consistently Under 1.5 s Across All Sentences

Figure 6 presents per-sentence latency as a grouped bar chart. Google's latency is remarkably stable across all sentences, staying between 0.82 and 1.31 seconds regardless of sentence length. VOSK's latency scales with audio duration: short sentences (S4 through S9) are processed in 1.2 to 1.7 seconds, while longer sentences (S1, S2, S3, S10) take 3.4 to 4.6 seconds. Critically, VOSK's latency is entirely deterministic and unaffected by network conditions, server load, or bandwidth constraints.

6.2 LLM Insight Generation

The locally hosted LLaMA 3 8B model was evaluated for its ability to generate structured meeting summaries, task lists, and key decision points from finalized transcript segments. The model demonstrated consistent capability in identifying the core content of professional discussions and formatting it into readable, action-oriented output. Summaries correctly captured the main topics discussed, decisions reached, and follow-up responsibilities assigned during each session, without requiring any manual editing by participants.

As illustrated in Figure 7, the system automatically structures the processed dialogue into a formal Minutes of Meeting (MoM) format, organizing key discussion points into distinct categories and attaching specific action items along with their assigned owners and priority levels. This structured output removes the burden of manual note-taking and ensures that no action item is lost or overlooked after the meeting concludes.

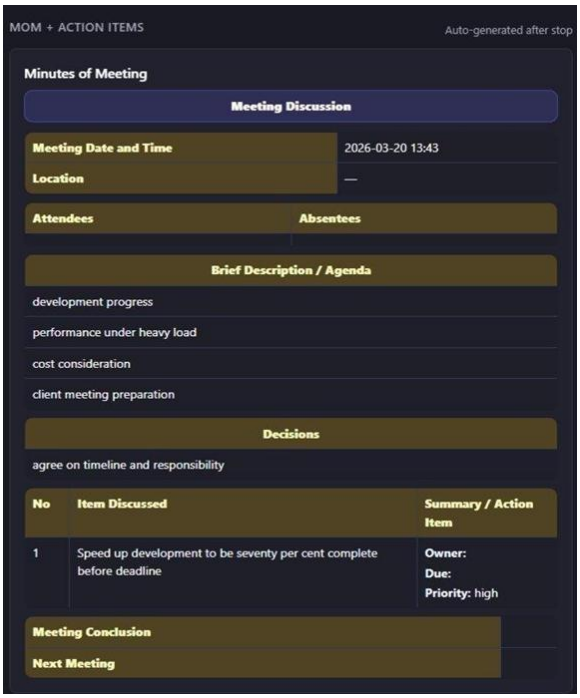

Figure 7. Structured Minutes of Meeting (MoM) Generated by the Agentic Orchestrator, Featuring Automated Action Item Extraction and Email Dispatch Options

6.3 Agent Orchestration

The agent orchestrator was evaluated across a range of meeting scenarios involving different intent types, including meeting summarization, email dispatch, web search, task assignment, and text-to-speech response. Across all observed sessions, the orchestrator demonstrated reliable intent detection and tool dispatching when conversational instructions were clearly and explicitly stated by the speaker.

Cases where the orchestrator did not successfully dispatch an action were primarily associated with ambiguously phrased instructions, where the speaker's intent was implicit or spread across multiple conversational turns without a clear trigger statement. In these situations, the orchestrator logged the segment as unresolved and continued processing the transcript without interruption, ensuring that individual parsing failures did not affect the overall pipeline.

Among the supported intent types, email generation and dispatch demonstrated strong end-to-end reliability. Web search intents performed with similar consistency: queries were extracted from the triggering utterance, submitted to the Web Search Tool, and the retrieved results were passed back

through the Local LLM Brain for contextual summarization before being surfaced on the user interface. Task assignment intents which involved writing structured task records to Encrypted Storage and optionally creating corresponding GitHub issues executed reliably in all cases where the task description and assignee were explicitly named in the conversation. Text-to-speech responses were delivered with minimal perceptible delay, as the TTS engine operated as an independent module that did not share resources with the transcription pipeline.

Meeting summarization intents consistently produced well-structured output covering the discussion points and decisions recorded up to the moment of the trigger. Across all intent types, asynchronous execution ensured that no tool invocation introduced blocking latency into the live transcription display, confirming that the orchestrator's non-blocking architecture functioned as intended under realistic meeting conditions.

6.4 User Feedback

Preliminary user feedback from ten participants indicated that the system significantly reduced the cognitive load associated with manual note-taking. Eight of ten participants reported that the automated task extraction feature saved them meaningful time in post-meeting follow-up. Six participants specifically noted the privacy advantage as a key reason they would prefer this system over cloud-based alternatives for sensitive internal discussions.

VII. THE ADVANTAGES OF CONTROL AND MONITORING OF MEETING INTELLIGENCE DEVICES

Today, retrieving information discussed in meetings in real time has become much easier than in the past. With modern meeting intelligence systems, users can access discussions from virtually anywhere using a smart device and an internet connection. In the case of the proposed edge-based solution, even constant internet connectivity is not strictly required, since most processing is handled locally on the device. These advancements have significantly improved productivity and documentation efficiency in professional environments. Meeting intelligence systems now play an important role in reducing manual effort during discussions. Automated summarization allows participants to focus more on active engagement rather than taking notes, while intelligent task extraction ensures that key action items are automatically identified, recorded, and assigned to relevant individuals. This helps improve accountability and reduces the chances of missing important follow-up actions. The actual productivity improvement depends on the nature of the meeting and the complexity of the conversation. However, in most typical business scenarios, the system is capable of generating structured Minutes of Meeting (MoM), sending follow-up emails, and creating task records without requiring manual intervention. One of the key advantages of the edge-based approach is enhanced privacy, as sensitive organizational data is processed locally and not continuously transmitted to external cloud servers. Modern systems also support remote access through dashboards, allowing users to generate summaries even if they were not actively taking notes during the meeting. With a simple interaction on the interface, structured summaries and action items can be retrieved within seconds. This makes collaboration more flexible and reduces dependency on manual documentation during live discussions.

Remote monitoring makes professional collaboration more comfortable and effective on the go. The system also provides email dispatch capability, allowing follow-up messages to be sent directly from the meeting context without switching applications.

VIII. DISCUSSION

The experimental evaluation demonstrates that the proposed edge-based meeting intelligence system is both effective and suitable for real-world professional applications. By integrating real-time speech recognition, locally deployed Large Language Models (LLMs), and an autonomous orchestration framework, the system is capable of providing intelligent meeting assistance without relying heavily on external cloud infrastructure. The architecture successfully supports live transcription, contextual understanding, and automated task execution while maintaining user privacy. The performance analysis shows that the edge-oriented design significantly reduces processing latency compared to traditional cloud-based systems. Since most computations are performed locally, the system avoids delays caused by continuous network communication and remote server processing. This

advantage becomes especially important in environments with unstable or limited internet connectivity, such as remote offices or low-bandwidth locations, where cloud-dependent solutions may suffer from interruptions and slower response times. Although the proposed system achieved promising results, certain limitations remain. One major challenge is the requirement for sufficient local computational resources to support real-time LLM inference. Devices with lower processing capabilities may experience slower response times during complex reasoning tasks or long-duration meetings. To address this issue, future improvements could introduce a hybrid processing strategy in which lightweight local models handle real-time meeting operations, while larger cloud-based models are used selectively for advanced post-meeting analysis when internet connectivity is available. Another important advantage of the proposed framework is its modular architecture. The layered design allows new services and automation tools to be integrated easily without affecting the core transcription or orchestration pipeline. Future extensions may include integration with customer relationship management (CRM) systems, enterprise workflow platforms, calendar services, and project management tools to further enhance meeting productivity and collaboration support.

IX. CONCLUSION

The proposed edge AI system integrates meeting intelligence with automated task execution in a unified and practical framework. It supports a range of flexible automation capabilities that are useful in professional environments, including email generation and dispatch, web-based information retrieval, task creation and assignment, and voice-based feedback through speech synthesis. Based on insights gathered from the literature review and existing related systems, this work also emphasizes strategies that can help organizations better understand and adopt intelligent meeting automation technologies. In addition, the system is designed to proactively notify users about follow-up actions and important updates that arise during the meeting itself, thereby reducing the need for manual tracking after the session ends. This paper introduced a modular, edge-driven agentic meeting assistant that enables real-time transcription, contextual understanding, and automated execution of meeting-related tasks in professional settings. The system combines streaming speech recognition using Vosk, a locally deployed LLM based on Meta LLaMA models, and a lightweight agent orchestration layer to deliver a privacy-focused alternative to conventional cloud-based meeting tools. The proposed architecture demonstrates that the integration of edge computing with agentic AI can significantly improve meeting efficiency while reducing latency, dependency on external services, and associated operational costs. The experimental evaluation and performance analysis summarized in Table 1, along with prototype testing results, confirm the practicality and effectiveness of the system in real-world enterprise scenarios. These findings highlight the advantages of an edge-first approach in delivering reliable and responsive meeting intelligence.

Future research directions include: (1) enhancing speech recognition performance in noisy or acoustically complex environments; (2) extending agent capabilities to support deeper integration with enterprise software ecosystems; (3) developing adaptive context management techniques to handle long-duration or multi-session meetings more efficiently; and (4) conducting large-scale empirical studies across different organizations to measure the impact of automated meeting intelligence on productivity and workflow optimization.

REFERENCES

[1] L. Lazzaroni, F. Bellotti, and R. Berta, “An Embedded End-to-End Voice Assistant,” Engineering Applications of Artificial Intelligence, vol. 136, Art. 108998, pp. 1–18, 2024.

[2] S. Asthana, S. Hilleli, P. He, and A. L. Halfaker, “Summaries, Highlights, and Action Items: Design, Implementation and Evaluation of an LLM-powered Meeting Recap System,” Proceedings of the ACM on Human-Computer Interaction, vol. 9, no. 2, Art. CSCW176, pp. 1–29, 2025.

[3] H. Chen et al., “MISP-Meeting: A Real-World Dataset with Multimodal Cues for Long-form Meeting Transcription and Summarization,” in Proceedings of the ACL, Volume 1: Long Papers, 2025, pp. 15479–15492.

[4] M. Houtti, M. Zhou, L. Terveen, and S. Chancellor, “Observe, Ask, Intervene: Designing AI Agents for More Inclusive Meetings,” in Proceedings of CHI 2025, ACM, 2025, pp. 1–29.
[5] J. Nanajkar et al., “A Survey on AI-Powered Meeting Assistants: Automating Agenda Generation, Summarization, and Record-Keeping,” International Journal of Creative Research Thoughts (IJCRT), vol. 13, no. 2, pp. 865–873, 2025.
[6] S. M. K. et al., “AI-Driven Multi-Modal Information Synthesis: Integrating PDF Querying, Speech Summarization, and Cross-Language Text Summarization,” Procedia Computer Science, vol. 258, pp. 2996–3018, 2025.
[7] Y. O. Sharrab et al., “Advancements in Speech Recognition: A Systematic Review of Deep Learning Transformer Models,” IEEE Access, vol. 13, pp. 46925–46940, 2025.
[8] S. Di Leo, L. De Cicco, and S. Mascolo, “Real-Time Speech-to-Text on Edge: A Prototype System for Ultra-Low Latency Communication with AI-Powered NLP,” Information, vol. 16, no. 8, Art. 685, pp. 1–10, 2025.
[9] H. Polat et al., “Implementation of a Whisper Architecture-Based Turkish ASR System,” Electronics, vol. 13, no. 21, p. 4227, 2024.
[10] A. Gupta et al., “Realtime Meeting Transcript Summarizer (Awaaz AI),” International Journal of Advanced Research in Science, Communication and Technology, vol. 5, no. 4, pp. 141–146, 2025.
[11] S. Chaudhari et al., “AI-Powered Virtual Meeting Summarizer,” International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 10, p. 2388, 2025.
[12] A. Kasav et al., “Smart Meeting Assistant and Performance Tracker for Online Meetings,” IJIRT, vol. 12, no. 9, p. 1698, 2026.