TeamSync: A Smart Meeting Agent

Vrinda Ahuja
MS Computer Science
Columbia University
vva2113@columbia.edu

Sachi Kaushik
MS Computer Science
Columbia University
sk5476@columbia.edu

Akshara Pramod
MS Computer Science
Columbia University
ap4613@columbia.edu

***Abstract*—TeamSync is an AI-powered framework that can handle an entire meeting process on its own. The agent intends to join and transcribe live meetings to summarize discussions, create relevant sequential tasks, and schedule follow-ups. It uses LiveKit and Whisper for audio processing, Retrieval-Augmented Generation (RAG) for context-based reasoning, and Model Context Protocol (MCP) to interact with tools like Jira and Google Calendar.**

This system will be built as a multi-agent setup, where each agent is assigned a specific job: the Listener records and processes meetings, the Summarizer creates concise reports, the Retriever answers questions about past meetings, and the Action and Scheduler agents handle tasks and follow-ups. These agents work together using a ReAct-inspired model, which helps them reason, take actions, and learn from the results.

By combining open-source tools such as LangChain, vLLM, FAISS, and FastAPI, we aim at creating a reliable and efficient workflow that transforms meeting conversations into actionable results [9]. The main idea is to create a system that demonstrates how AI can make meetings more productive by reducing manual effort, improving collaboration, and ensuring that decisions quickly translate into concrete actions.

***Index Terms*—Agentic AI, Multi-Agent Systems, Large Language Models (LLMs), Meeting Automation, Retrieval-Augmented Generation (RAG), Model Context Protocol (MCP), Workflow Automation, ReAct Framework, Tool-Augmented Agents.**

I. Introduction and Problem Definition

Meetings are the backbone of team coordination, but the work that follows the meeting is a universal bottleneck. We are all drowning in follow-ups from meetings. Trying to remember who agreed to what, manually writing accurate minutes, and ensuring critical action items are not forgotten is slow, tedious, and notoriously error-prone. This manual gap is where key tasks fall and project momentum is lost.

Although modern AI tools like Otter.ai and Microsoft Copilot have emerged to offer transcription and basic summaries, they are not a complete solution. Their biggest flaw is that they are "single-pass" systems; they generate a summary in one shot without ever stopping to critique their own output. This "fire and forget" approach is why they still struggle with factual inaccuracies, miss the real context of a conversation, and fail to reliably extract all action items. We simply cannot trust them to run our workflows.

To solve this, we are proposing **TeamSync**, an autonomous AI assistant designed to manage the entire meeting lifecycle with a new level of reliability. This system will autonomously:

- Join and transcribe live meetings in real-time (using LiveKit and Whisper).
- Answer questions during the meeting about past decisions (using a RAG pipeline).
- Generate comprehensive Minutes of Meeting (MoM), summaries, and action lists.
- Autonomously create Jira tickets for tasks and schedule follow-ups in Google Calendar.

The system is designed as a multi-agent team (Listener, Summarizer, Knowledge Retriever, Action Executor, etc.) coordinated via the Model Context Protocol (MCP) [4].

However, the key problem this work addresses is not just automation, but trust. The central research innovation is the inclusion of a **Self-Reflection Agent** whose entire job is to critique and improve the work of the other agents before a human ever sees it. This agent creates a self-correcting evaluation loop, asking: "Is this summary factually consistent with the transcript?" "Did the Action Agent miss a task that was assigned?"

This leads to the core research question: **Can adding a self-reflection mechanism within a multi-agent LLM system significantly improve the factual consistency and completeness of meeting summaries and action items compared to traditional, non-reflective methods?** This project aims to build a truly autonomous and trustworthy AI assistant, moving beyond simple summarization to create a reliable partner in our daily workflows.

II. Proposed Approach

We are proposing a multi-agent framework where specialized agents collaborate through the Model Context Protocol (MCP) to manage the complete meeting lifecycle. This approach is inspired by the ReAct framework [1], which enables agents to reason about situations, take actions, observe results, and iteratively refine their outputs.

A. System Architecture

The system employs a distributed agent topology with six specialized agents, each responsible for a distinct aspect of meeting intelligence:

- **Listener Agent**: Captures real-time audio streams via LiveKit, performs transcription using WhisperX [10], and conducts speaker diarization with pyannote.audio [7]. Outputs structured JSON transcripts with speaker labels and timestamps.

- **Knowledge Agent (RAG)**: Maintains a vector database (FAISS) of historical meeting transcripts embedded using sentence-transformers. Responds to queries during meetings by retrieving relevant context from past discussions and generating grounded answers.
- **Summarizer Agent**: Processes complete transcripts to generate Minutes of Meeting (MoM), extract key decisions, identify action items with assignees, and highlight unresolved questions.
- **Action Agent**: Translates extracted action items into concrete workflow tasks. Creates Jira tickets with proper fields (summary, description, assignee, due date) and sends notifications to relevant team members via MCP tool integrations.
- **Self-Reflection Agent**: This is the critical innovation that differentiates this system. Reviews outputs from the Summarizer and Action agents, verifying factual consistency against source transcripts, checking for missed action items, and validating logical coherence. Implements a critique-revise evaluation loop before finalizing outputs.
- **Scheduler Agent**: Analyzes action item dependencies and meeting outcomes to propose follow-up meetings. Integrates with Google Calendar API to schedule events and send invitations to participants.

B. Technical Pipeline

The end-to-end workflow operates as a cohesive pipeline that transforms live meeting audio into actionable outcomes through four interconnected stages.

Stage 1: Real-Time Capture and Processing. LiveKit establishes a WebRTC connection to join the meeting room as a virtual participant. As the meeting progresses, audio streams are continuously processed by WhisperX for real-time transcription, while Pyannote.audio simultaneously performs speaker diarization to identify individual speakers. The transcripts are stored as vector embeddings for semantic search.

Stage 2: Intelligent Retrieval. Building on this knowledge base, participants can query past decisions during meetings. The Knowledge Agent embeds queries using sentence-transformers, performs similarity search across FAISS, and retrieves the top-k relevant segments. LangChain [3] then orchestrates retrieval-augmented [2] generation to produce contextual answers with citations to original meeting sources.

Stage 3: Self-Reflective Summarization. After the meeting concludes, the Summarizer Agent generates an initial Minutes of Meeting document with extracted action items. The Self-Reflection Agent then validates this output by checking for factual consistency with the transcript, completeness of action items with assignees and deadlines, and absence of contradictions. When issues are detected, the system enters a revision loop until the output passes all validation checks and is ready for human review.

Stage 4: Autonomous Execution. With validated outputs, the Action Agent autonomously executes workflows through the Model Context Protocol. It creates Jira tickets and maintains audit trails for accountability. Simultaneously, the Scheduler Agent analyzes task dependencies to propose follow-up meetings and integrates with Google Calendar API to schedule events and send invitations to participants.

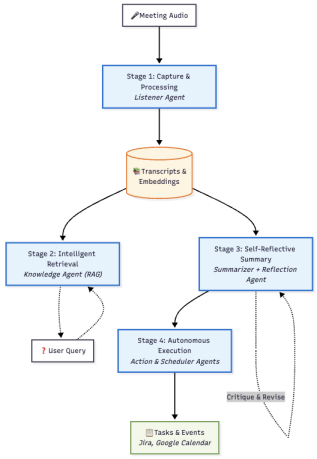

Fig. 1. Flow Diagram

C. Implementation Framework

The system is built using open-source components to ensure reproducibility:

- **Backend**: FastAPI provides REST APIs for agent coordination, Redis manages job queues, PostgreSQL stores structured data
- **LLM Serving**: vLLM enables efficient batched inference with continuous batching for low-latency responses
- **Orchestration**: LangGraph [6] or CrewAI coordinates multi-agent workflows with state management

III. Feasibility and Plan

Our project is planned for completion within a six-week timeframe, utilizing exclusively open-source technologies and free-tier cloud services to ensure feasibility. We will leverage Google Cloud Platform (GCP) credits and resources to eliminate implementation costs, focusing initially on developing a functional prototype with provisions for future scalability. The next steps for development and research are as follows:

Week 1: Foundation Setup. We begin by deploying the LiveKit server and integrating WhisperX for real-time transcription with pyannote.audio for speaker diarization. The database infrastructure will be configured concurrently. Next, we will implement our *Listener Agent* to generate structured JSON outputs with speaker labels and timestamps.

Week 2: RAG Pipeline. We aim at implemneting the RAG pipeline using sentence-transformers for embeddings and LangChain for orchestration. Also, we will work on our next agent, the *Knowledge Agent* to create an interface between user queries and the retrieval system, enabling natural language questions about past meetings with citations.

Week 3-4: Multi-Agent Architecture. In this step we plan to develop our *Summarizer Agent*, followed by the critical *Self-Reflection Agent* with its three-pass verification protocol. The

Action and *Scheduler Agents* will be developed in parallel. We plan to use CrewAI/LangGraph for orchestrating agent coordination.

Week 5: Tool Integration via MCP. We implement the MCP server for secure tool calling, beginning with Jira API integration for automated ticket creation. Next, we will add Google Calendar API for enabling automated actions including user confirmation workflows before execution.

Week 6: Evaluation and Optimization. As a last step, we plan to measure transcription WER, RAG precision/recall, ROUGE scores, and action item extraction F1. We will test the self-reflection agent's contribution but running out flow once with and then without that agent.

IV. Innovation and Relevance

A. Novel Contributions

Our project's core innovation is a **self-reflective multi-agent architecture** designed to solve the unreliability of current AI assistants. We've all seen LLMs hallucinate, and in an organizational setting, a "single-pass" summary that's factually wrong is worse than no summary at all, especially when it's based on an imperfect transcription. Our novel contribution is a full team of specialized agents, with the most critical being the **Self-Reflection Agent**. This agent functions as a dedicated "editor," auditing the outputs of the Summarizer and Action Agents. Instead of the risky "fire-and-forget" approach, this creates a self-correction loop, critically checking for factual consistency against the transcript. This agentic, self-verifying design is our direct solution to the "garbage in, garbage out" problem and builds the foundational trust currently missing from automated tools.

B. Relevance and Research Significance

This agent-based system is highly relevant as it tackles the universal, practical bottleneck of post-meeting manual work and dropped tasks. From an academic standpoint, this project contributes to the emerging field of *autonomous, self-verifying AI systems*. Our research will empirically test the core question: can a self-reflection mechanism *measurably* improve the factual accuracy and completeness of a multi-agent system's output? The reflective agent is the "brain" of a complete, end-to-end team: the Listener Agent transcribes (LiveKit/Whisper), the Knowledge Agent provides memory (RAG), and the Action Agent autonomously executes tasks (Jira/GCal). Ultimately, this work is a step toward building an AI system that is not just a tool, but a trustworthy, collaborative partner in a team's workflow.

V. Conclusion

TeamSync represents a paradigm shift from passive meeting transcription tools to active, autonomous workflow partners. By addressing the fundamental trust gap in current AI meeting assistants through a self-reflective multi-agent architecture, our work aims at tackling a critical bottleneck in organizational productivity: the manual cognitive load of post-meeting processing.

We wish to explore and understand whether self-reflection mechanisms can measurably improve the factual consistency and completeness of AI-generated meeting intelligence. Unlike existing single-pass systems that operate on a "fire-and-forget" basis, our approach introduces a supervisory Self-Reflection Agent that critiques and refines outputs before human review, creating a self-correcting reasoning loop that builds trust and reliability.

Beyond just summarization, our system is designed to autonomously execute workflows by creating Jira tickets, scheduling follow-ups, and maintaining a queryable team memory through RAG. This tool integration, handled via the Model Context Protocol, transforms the assistant from a passive observer into an active, engaged team member. Our ambitious 6-week implementation plan is grounded in this feasible, open-source approach, ensuring reproducibility. We will systematically evaluate our success by measuring transcription accuracy, RAG performance, and the F1-score of our action item extraction. This evaluation will focus on testing the self-reflection agent, allowing us to generate clear empirical evidence for its value in building more trustworthy, self-verifying AI systems.

This work contributes to the emerging field of autonomous, self-verifying AI systems while delivering immediate practical value: reducing administrative overhead, preventing dropped tasks, and ensuring that meeting decisions reliably translate into concrete actions. The success of this system will demonstrate that AI can be not just a productivity tool, but a trustworthy partner in collaborative work.

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. 11th International Conference on Learning Representations (ICLR)*, 2023.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] H. Chase, "LangChain," *GitHub Repository*, 2022. [Online]. Available: https://github.com/langchain-ai/langchain

[4] Anthropic, "The Model Context Protocol," 2024. [Online]. Available: https://modelcontextprotocol.io/

[5] J. Moura, "CrewAI: Framework for orchestrating role-playing, autonomous AI agents," *GitHub Repository*, 2023. [Online]. Available: https://github.com/joaomdmoura/crewAI

[6] LangChain, "LangGraph: Building stateful, multi-actor applications with LLMs," 2023. [Online]. Available: https://github.com/langchain-ai/langgraph

[7] H. Bredin, "Pyannote.audio: Neural building blocks for speaker diarization," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7124–7128, 2020.

[8] S. Madaan, N. Tandon, P. Gupta, H. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-Refine: Iterative Refinement with Self-Feedback," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] R. Zhang, J. Guo, L. Chen, Y. Fan, and E. Choi, "Automatic Meeting Summarization: A Survey," *ACM Computing Surveys*, vol. 55, no. 2, pp. 1–36, 2022.

[10] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-Accurate Speech Transcription of Long-Form Audio," *arXiv preprint arXiv:2303.00747*, 2023.