

AI-Powered Meeting Assistant for Transcription, Summarization and Action Extraction

Dr. Rupali Mishra

Coordinator, Department of Information Technology, Prahadrai Dalmia Lions College, Mumbai, India

Abstract - Meetings are an integral part of organizational communication, yet documenting them effectively remains a persistent challenge. Traditional manual note-taking is time-consuming, error-prone, and often results in incomplete information capture. This paper presents AutoScribe, an AI-powered meeting assistant designed to automate transcription, summarization, and action item extraction from meeting recordings. The system leverages state-of-the-art speech recognition using OpenAI's Whisper API and natural language processing using GPT-4-based models to transform raw meeting audio into structured, actionable outputs. A production-grade full-stack architecture comprising React for the frontend, FastAPI for the backend, and MongoDB for data storage ensures scalability and reliability. AutoScribe provides users with accurate transcripts, concise summaries, structured action item lists, and email-delivered reports, thereby enhancing productivity and decision-making. Experimental evaluation on 50 real-world meeting recordings demonstrates a Word Error Rate (WER) of 6.3% for transcription, a ROUGE-L F1 score of 0.71 for summarization, and an action extraction F1 score of 85.4%. These results confirm AutoScribe's effectiveness as a comprehensive meeting intelligence solution, particularly relevant in remote and hybrid work environments

Key Words: AI, Automatic Speech Recognition, Meeting Assistant, Natural Language Processing, GPT-4, Whisper API, Summarization, Action Item Extraction, FastAPI, MongoDB

1. INTRODUCTION

In today's dynamic workplace, meetings serve as the cornerstone of organizational communication, collaboration, and decision-making. Whether conducted in-person, remotely, or in hybrid formats, meetings facilitate knowledge exchange and strategic planning. However, a persistent challenge remains — the documentation of these meetings. Studies indicate that professionals spend approximately 31 hours per month in unproductive meetings, with a significant portion of time wasted on follow-up and re-clarification due to poor or absent documentation [1].

Traditional approaches to meeting documentation include manual note-taking, which is inherently subjective and error-prone. Note-takers often miss critical points, misinterpret statements, or fail to capture action items accurately. This problem is amplified in fast-paced, multi-speaker meetings where simultaneous conversations occur.

The resulting documentation is often incomplete, inaccurate, or overly verbose, rendering it less useful for decision-making and follow-up.

The advent of Artificial Intelligence (AI), particularly in the domains of Automatic Speech Recognition (ASR) and Natural Language Processing (NLP), has opened new avenues for automating meeting documentation. AI-powered systems can transcribe spoken language in real-time or from recordings, extract semantic meaning, and generate concise summaries — all with minimal human intervention. Such systems not only improve accuracy but also reduce cognitive load on meeting participants, allowing them to focus on the discussion rather than documentation.

AutoScribe is an AI-powered meeting assistant designed to address these challenges. The system accepts audio recordings of meetings, transcribes them using OpenAI's Whisper API, processes the transcriptions through GPT-4-based NLP models to generate structured summaries and extract action items, and delivers these outputs to users through an intuitive web interface. The system also supports email delivery of reports and provides centralized storage through MongoDB, making it suitable for team-wide deployment.

This paper presents the design, architecture, implementation, and evaluation of AutoScribe. Section 2 reviews related work. Section 3 describes the system methodology and architecture. Section 4 discusses the system modules in detail. Section 5 presents experimental results. Section 6 concludes the paper with future directions.

2. LITERATURE REVIEW

Automated meeting transcription and summarization have attracted significant research attention over the past decade. Early systems relied on rule-based approaches and HMM-based ASR engines, which suffered from poor accuracy in noisy environments and with accented speech [2]. The introduction of deep learning-based ASR systems, particularly recurrent neural networks and later transformer architectures, dramatically improved transcription quality.

Whisper, developed by OpenAI, represents the state of the art in open-vocabulary ASR. Trained on 680,000 hours of multilingual speech data, Whisper achieves near-human-level word error rates across diverse acoustic conditions and accents [3]. Its robustness makes it particularly suitable for

meeting environments where audio quality can vary significantly.

In the domain of NLP-based summarization, extractive methods — which select key sentences from the original text — were dominant in early systems. However, abstractive summarization using transformer-based language models such as BERT, T5, and GPT variants has shown superior performance in generating coherent, contextually appropriate summaries [4]. GPT-4, with its instruction-following capability and large context window, is particularly well-suited for meeting summarization and action item extraction tasks.

Several commercial meeting assistants exist, including Otter.ai, Fireflies.ai, and Microsoft Teams' built-in transcription. While effective, these tools are typically closed-source, subscription-based, and offer limited customization. Academic systems such as AMI Meeting Corpus tools and ICSI Meeting Corpus analyzers offer research value but lack production-ready interfaces [5]. AutoScribe bridges this gap by offering an open, extensible, full-stack solution with a customizable AI pipeline.

Action item extraction — identifying tasks, assignees, and deadlines from meeting transcripts — remains an understudied problem. Prior work by Purver et al. (2007) approached it as a classification task using shallow features [6]. Modern LLM-based approaches, leveraging in-context learning and prompt engineering, offer a more flexible and accurate alternative, which AutoScribe exploits.

3. SYSTEM ARCHITECTURE

AutoScribe follows a layered, modular architecture comprising a React-based frontend, a FastAPI-powered backend, and a MongoDB database. The system adopts RESTful API design principles for communication between layers, ensuring loose coupling and easy scalability. Figure 1 illustrates the high-level system architecture.

3.1 Frontend Layer: The frontend is developed using React.js, a component-based JavaScript framework. It provides an intuitive user interface for audio file upload, progress tracking, and results display. The interface includes modules for transcript viewing, summary display, action item listing, and report download. Authentication is handled via JWT tokens, ensuring secure session management.

3.2 Backend Layer: The backend is built on FastAPI, a high-performance Python web framework. FastAPI's asynchronous capabilities allow concurrent processing of multiple meeting uploads. The backend exposes endpoints for user authentication, file upload, transcription triggering, summary generation, and data retrieval. All API calls are documented automatically via FastAPI's built-in OpenAPI (Swagger) interface.

3.3 AI Processing Pipeline: The AI pipeline constitutes the core of AutoScribe. Upon receiving an audio file, the pipeline invokes the Whisper API for speech-to-text conversion. The

resulting transcript is then passed to the GPT-4 API through carefully engineered prompts for summarization and action extraction. The pipeline is designed to be modular — individual components can be swapped or upgraded independently as better models become available.

3.4 Database Layer: MongoDB, a document-oriented NoSQL database, is used for data persistence. Each meeting is stored as a document containing the raw transcript, generated summary, extracted action items, metadata (user ID, timestamp, duration), and email delivery status. MongoDB's flexible schema accommodates varying meeting structures without requiring predefined table schemas.

4. METHODOLOGY

AutoScribe's processing pipeline follows a sequential, five-stage workflow as described below.

Stage 1 — Audio Upload: Users upload meeting recordings through the React frontend. Supported formats include MP3, MP4, WAV, and M4A. The frontend validates file type and size before transmitting the file to the backend via a multipart HTTP POST request. The backend stores the file temporarily in local storage for processing.

Stage 2 — Transcription: The audio file is forwarded to the OpenAI Whisper API. Whisper performs end-to-end speech recognition, producing a raw text transcript with timestamps. The system uses the 'whisper-1' model, which supports over 50 languages. Speaker diarization (attributing speech to individual speakers) is applied in post-processing using pyannote.audio, providing speaker-labeled transcripts where possible.

Stage 3 — Summarization: The raw transcript is segmented and passed to the GPT-4 API. A custom system prompt instructs the model to generate a structured summary covering: (a) meeting agenda, (b) key discussion points, (c) decisions made, and (d) unresolved issues. The summary is formatted in markdown for easy rendering in the frontend.

Stage 4 — Action Extraction: A separate GPT-4 call is made with a prompt engineered specifically to identify action items. The model extracts tasks in a structured JSON format containing: task description, assignee (if mentioned), deadline (if mentioned), and priority level. This structured output is directly rendered as an interactive checklist in the frontend.

Stage 5 — Storage and Delivery: The transcript, summary, and action items are stored in MongoDB under the user's account. Users can optionally trigger an email delivery of the meeting report using SendGrid integration. Reports are formatted as HTML emails with inline CSS for compatibility across email clients.

5. IMPLEMENTATION DETAILS

5.1 Technology Stack: The complete technology stack for AutoScribe is summarized in Table 1. The choice of each

technology was driven by performance, ecosystem maturity, and developer productivity considerations.

Table 1: AutoScribe Technology Stack

Component	Technology
Frontend	React.js, Axios, Tailwind CSS
Backend	FastAPI (Python 3.10)
Speech Recognition	OpenAI Whisper API
NLP / Summarization	OpenAI GPT-4 API
Speaker Diarization	pyannote.audio
Database	MongoDB (Atlas)
Authentication	JWT (JSON Web Tokens)
Email Delivery	SendGrid API
Deployment	Docker, Nginx, AWS EC2

5.2 API Integration: Integration with OpenAI APIs follows a secure, server-side pattern. The backend stores API keys in environment variables, never exposing them to the frontend. Requests are rate-limited using FastAPI middleware to comply with OpenAI's usage policies. Error handling covers API timeout, token limit exceeded, and invalid audio format scenarios.

5.3 Security Implementation: AutoScribe implements OAuth 2.0-compliant authentication using JWT tokens with a 24-hour expiry. Passwords are hashed using bcrypt with a salt factor of 12. All API endpoints are protected with Bearer token authentication except the registration and login endpoints. HTTPS is enforced via Nginx reverse proxy with Let's Encrypt SSL certificates.

5.4 Frontend Design: The React frontend employs a component-based architecture with React Router for navigation. Key components include: AudioUploader (drag-and-drop file upload with progress indicator), TranscriptViewer (scrollable, searchable transcript with speaker labels), SummaryPanel (formatted meeting summary), ActionBoard (interactive task checklist), and HistoryDashboard (paginated meeting history).

6. EXPERIMENTAL EVALUATION

To evaluate AutoScribe's performance, a dataset of 50 meeting recordings was collected from consenting participants. Meetings ranged from 15 to 90 minutes, covering topics including project reviews, academic seminars, and team standups. Performance was assessed across four dimensions: transcription accuracy, summarization quality, action extraction precision, and system response time.

6.1 Transcription Accuracy: Word Error Rate (WER) was used as the primary metric for transcription quality. AutoScribe achieved an average WER of 6.3% across the test dataset, compared to 18.7% for Google Speech-to-Text (free tier) and 9.1% for Azure Cognitive Services. Performance

was consistently better in low-noise environments, with WER as low as 3.1% for studio-quality recordings.

Table 2: Transcription WER Comparison

System	Avg. WER (%)	Notes
AutoScribe (Whisper)	6.3	Best performance
Azure Speech Services	9.1	Close second
Google Speech-to-Text	18.7	Lower accuracy
Manual Transcription	2.1	Human baseline

6.2 Summarization Quality: Summarization quality was assessed using ROUGE-L scores against human-generated reference summaries. AutoScribe achieved a ROUGE-L F1 score of 0.71, indicating strong overlap with human summaries. Qualitative evaluation by domain experts rated 86% of generated summaries as 'accurate' or 'highly accurate'. Common failure modes included over-compression of technical discussions and occasional omission of nuanced context.

6.3 Action Extraction Precision: For action item extraction, precision and recall were measured against manually annotated ground truth. AutoScribe achieved a precision of 88.4% and recall of 82.7%, resulting in an F1 score of 85.4%. The model performed best when action items were explicitly stated (e.g., 'John will submit the report by Friday') and less accurately on implicit commitments.

6.4 System Performance: Average end-to-end processing time for a 30-minute meeting recording was 4.2 minutes on the deployed server (AWS EC2 t3.medium). This includes 2.8 minutes for Whisper transcription, 0.7 minutes for summarization, 0.4 minutes for action extraction, and 0.3 minutes for storage operations. For real-time applications, the system can process audio in streaming chunks with approximately 1.3x real-time speed.

7. USE CASE SCENARIOS

7.1 Corporate Team Meetings: AutoScribe is particularly valuable for weekly team standups and project review meetings. Integration with calendar systems (Google Calendar, Outlook) enables automatic meeting detection and scheduling of post-processing. Action items are synchronized with project management tools such as Jira and Trello via webhook integrations.

7.2 Academic and Research Seminars: Research group meetings and conference presentations benefit from AutoScribe's ability to handle technical vocabulary. The system's multi-language support (50+ languages) accommodates international academic collaborations. Lecture recordings can be processed to generate study notes and highlight key concepts.

7.3 Remote and Hybrid Work: In the post-pandemic era of distributed teams, AutoScribe addresses the documentation gap in virtual meetings. Unlike native conferencing tools, AutoScribe is platform-agnostic — it works with recordings from Zoom, Microsoft Teams, Google Meet, or any audio source. This flexibility makes it universally applicable across organizational IT landscapes

8. LIMITATIONS AND FUTURE WORK

Despite its strong performance, AutoScribe has several limitations. First, speaker diarization accuracy degrades in meetings with more than six participants due to voice similarity. Future work includes fine-tuning speaker embedding models on meeting-domain data. Second, the system currently does not support real-time transcription — only post-meeting processing. A streaming mode using Whisper's real-time API, currently in beta, is planned for a future release.

Third, GPT-4-based summarization introduces latency and cost per meeting. Exploring smaller, fine-tuned models (e.g., Mistral-7B or LLaMA-3 fine-tuned on meeting data) as cost-effective alternatives is a priority for future research. Fourth, the current action extraction module lacks the ability to track task completion status over time. Future iterations will include a project-level dashboard with task tracking, deadline reminders, and completion analytics.

Additionally, privacy compliance in enterprise settings requires on-premise deployment options. Future releases will support local model inference using quantized LLMs, enabling sensitive-data processing without cloud API dependencies. Integration with enterprise identity providers (LDAP, SAML 2.0) is also planned

9. CONCLUSIONS

This paper has presented AutoScribe, an AI-powered meeting assistant that automates transcription, summarization, and action item extraction from meeting recordings. By integrating OpenAI's Whisper ASR and GPT-4 NLP models within a production-grade full-stack architecture, AutoScribe delivers accurate, structured meeting documentation with minimal user effort.

Experimental evaluation on 50 meeting recordings demonstrates that AutoScribe achieves a WER of 6.3% for transcription, a ROUGE-L F1 of 0.71 for summarization, and an action extraction F1 of 85.4%. These results validate the effectiveness of combining state-of-the-art ASR and LLM technologies for meeting intelligence applications.

AutoScribe is particularly relevant in today's hybrid work environments, where efficient documentation and follow-up are critical for team productivity. By eliminating the burden of manual note-taking, AutoScribe enables participants to engage more deeply in discussions while ensuring that no critical information is lost. Future work will focus on real-time processing, on-premise deployment, and integration with enterprise productivity ecosystems.

REFERENCES

1. J. M. Perlow, C. Hadley, and E. Eun, "Stop the Meeting Madness," *Harvard Business Review*, Jul.-Aug. 2017.
2. S. Young et al., "The HTK Book (for HTK Version 3.4)," Cambridge University Engineering Department, 2006.
3. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," *arXiv preprint arXiv:2212.04356*, 2022.
4. T. Brown et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.
5. J. Carletta et al., "The AMI Meeting Corpus: A Pre-announcement," in *Machine Learning for Multimodal Interaction*, Springer, 2006.
6. M. Purver, J. Dowding, J. Niekrasz, P. Ehlen, and S. Peters, "Detecting and Summarizing Action Items in Multi-Party Dialogue," in *Proc. 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium, 2007.
7. OpenAI, "OpenAI API Documentation," [Online]. Available: <https://platform.openai.com/docs>. [Accessed: Mar. 2025].
8. FastAPI, "FastAPI Documentation," [Online]. Available: <https://fastapi.tiangolo.com>. [Accessed: Mar. 2025].
9. MongoDB, Inc., "MongoDB Documentation," [Online]. Available: <https://www.mongodb.com/docs>. [Accessed: Mar. 2025].
10. React, "React Documentation," [Online]. Available: <https://reactjs.org>. [Accessed: Mar. 2025].