AI-Powered Organizational Knowledge Management Platform Using Large Language Models for Intelligent Meeting Understanding and Semantic Knowledge Retrieval

Authors

Vyankatesh Talokar	Huzefa Khan	Tushar Deshmukh
Student, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.	Student, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.	Student, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.
Rajnandini Ingle	**Ankita Rathod**	**Prof. M. K. Nichat**
Student, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.	Student, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.	Assistant Professor, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India.

Abstract

Organizations conduct frequent meetings for technical discussions, project planning, decision-making, and task assignment. However, important meeting information is often scattered across recordings, notes, and documents, making it difficult to retrieve and reuse. This paper presents an AI-powered organizational knowledge management platform that converts meeting recordings into structured and searchable knowledge. The system uses OpenAI Whisper for transcription, an LLM to extract summaries, topics, decisions, and action items, PostgreSQL for structured storage, and ChromaDB with sentence embeddings for semantic retrieval. A RAG-based pipeline enables users to ask natural-language queries and receive context-grounded answers with meeting references and timestamps. The implemented platform integrates React, Spring Boot, FastAPI, Whisper, embeddings, PostgreSQL, ChromaDB, and an OpenRouter-hosted LLM, and was validated through backend, AI-service, and end-to-end testing.

Keywords: Artificial Intelligence, Organizational Knowledge Management, Meeting Understanding, Large Language Models, Speech-to-Text, Semantic Search, Embeddings, ChromaDB, Retrieval-Augmented Generation, Meeting Summarization.

1. Introduction

Organizations conduct frequent meetings for project planning, technical discussions, client requirements, progress reviews, and decision-making. These meetings generate valuable knowledge such as decisions, requirements, discussions, and task assignments. However, this information is often scattered across audio recordings, notes, documents, emails, and collaboration platforms, making it difficult to retrieve and reuse. Manual note-taking can also result in incomplete or inconsistent records, while finding specific information in lengthy past meetings can be time-consuming [5], [7]. Recent advances in Automatic Speech Recognition, Large Language Models, semantic embeddings, and Retrieval-Augmented Generation provide opportunities to automate this process [1]–[4]. The proposed AI-powered organizational knowledge management platform converts meeting recordings into timestamped transcripts, extracts summaries, topics, decisions, and action items, and indexes transcript content using embeddings. It further provides natural-language semantic search and RAG-based question answering, creating a persistent and searchable knowledge layer for retrieving information from historical meetings.

2. Aims and Objectives

The objectives of the proposed system are:

1. To develop a platform for organizing and managing meeting information.
2. To convert meeting recordings into text and extract summaries, key topics, decisions, and action items using AI

techniques.

3. To store meeting transcripts and structured knowledge in a persistent repository.
4. To implement semantic search and RAG-based question answering using vector embeddings.
5. To ensure secure access to meeting information through authentication and access control.

3. Literature Review

The development of intelligent meeting knowledge systems is supported by research in speech recognition, meeting summarization, knowledge management, sentence embeddings, and Retrieval-Augmented Generation.

3.1 Meeting Summarization

Rennard et al. presented a survey of abstractive meeting summarization techniques and discussed methods, datasets, and challenges involved in automatically summarizing meeting conversations [5]. Such research demonstrates the importance of automatically converting lengthy meeting discussions into concise information. However, summarization alone does not provide a complete long-term organizational knowledge retrieval mechanism.
The proposed system extends the concept of meeting summarization by extracting not only summaries but also topics, decisions, and action items and by making historical meeting content searchable.

3.2 Automatic Speech Recognition

Radford et al. introduced Whisper, a large-scale weakly supervised speech recognition system capable of robust speech transcription across different languages and acoustic conditions [1].
Whisper provides the foundation for the proposed system's Speech-to-Text component. The meeting recording is converted into a transcript along with timestamped segments, which are subsequently processed by the knowledge extraction pipeline.

3.3 Sentence Embeddings and Semantic Representation

Reimers and Gurevych introduced Sentence-BERT, demonstrating the effectiveness of sentence embeddings for semantic similarity and information retrieval tasks [2].
The proposed platform applies sentence-transformer embeddings to transcript chunks. This allows the system to represent meeting content as numerical vectors and retrieve content based on semantic similarity.

3.4 Retrieval-Augmented Generation

Lewis et al. introduced Retrieval-Augmented Generation, which combines information retrieval with language generation. Instead of relying entirely on information stored within an LLM, relevant external knowledge is retrieved and supplied as context to the model [3].

The proposed platform applies this principle to organizational meeting knowledge. Relevant transcript chunks are retrieved from the vector database and supplied to the LLM to generate grounded answers.

3.5 RAG for Large Language Models

Research on RAG systems has demonstrated their usefulness for knowledge-intensive question answering. Gao et al. reviewed Retrieval-Augmented Generation techniques and identified retrieval quality and effective context construction as important factors in RAG systems [4].
In the proposed system, transcript chunks are retrieved from ChromaDB using semantic similarity, and timestamp metadata is included when constructing the context supplied to the LLM.

3.6 LLM-Based Action Item Extraction

Recent research on LLM-based meeting analysis has demonstrated the usefulness of language models for identifying tasks and action items from meeting conversations [6].
The proposed system extends this concept by extracting action-item descriptions along with available assignees and due dates and storing them as structured knowledge.

Table 01: Comparative Literature Review

Ref.	Author(s) & Year	Technique	Contribution	Limitation / Gap
[1]	Rennard et al. (2023)	Meeting summarization	Studies abstractive meeting summarization	Mainly focuses on summarization
[2]	Radford et al. (2023)	Whisper ASR	Robust speech recognition	Produces transcription rather than organizational knowledge
[3]	Reimers & Gurevych (2019)	Sentence-BERT	Semantic sentence embeddings	General semantic representation
[4]	Lewis et al. (2020)	RAG	Combines retrieval and generation	General knowledge-intensive tasks
[5]	Gao et al. (2023)	RAG survey	Reviews retrieval-augmented techniques	Not specific to meeting knowledge
[6]	Sadia et al. (2025)	LLM-based action extraction	Extracts meeting action items	Focuses primarily on action items
[7]	Alavi & Leidner (2001)	Knowledge management	Provides foundations for organizational knowledge management	Traditional knowledge-management perspective

Research Gap

The literature demonstrates strong individual solutions for speech recognition, meeting summarization, semantic representation, action-item extraction, and RAG [1]–[6]. However, these approaches are often presented as separate components.
There remains a practical need for an integrated platform that combines:

Meeting Audio → Transcription → Structured Knowledge → Semantic Indexing → Retrieval → RAG-based Answering

The proposed platform addresses this gap by combining these components into a single organizational meeting knowledge management workflow.

4. Proposed Methodology

The proposed system integrates meeting processing, AI-based knowledge extraction, structured storage, vector indexing, and semantic question answering.
The overall methodology consists of three interconnected pipelines:

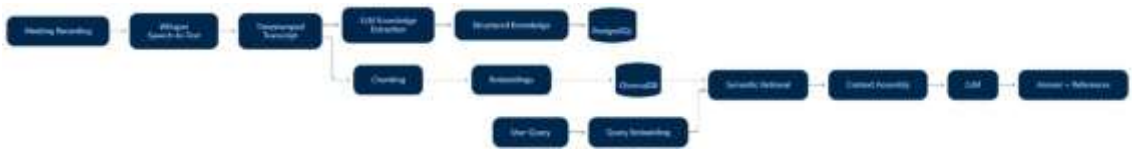


Fig. 01: Proposed AI-Powered Meeting Understanding and Semantic Knowledge Retrieval Workflow

4.1 Requirement Analysis

The first stage identifies the requirements of users who need to preserve and retrieve information generated during organizational meetings.
The system requires:

- User registration and authentication
- Meeting creation and metadata management
- Meeting audio upload

- Audio validation and storage
- Automatic transcription
- Timestamped transcript generation
- Summary and knowledge extraction
- Structured knowledge storage
- Semantic vector indexing
- Natural-language search
- RAG-based question answering
- Meeting references and timestamps
- User-specific data isolation

4.2 System Design

The proposed system follows a layered architecture that integrates the user interface, backend application services, AI processing services, relational knowledge storage, and vector-based semantic retrieval. The architecture is designed to support the complete workflow from meeting audio ingestion

to structured knowledge extraction and context-aware question answering. The workflow draws on established methods for speech recognition, semantic representation, and retrieval-augmented generation [1]–[4].

4.3 Meeting Audio Ingestion

The user first creates a meeting and uploads an audio recording.
The backend validates the uploaded file based on supported audio types and size restrictions. The audio is stored using a unique filename and associated with the corresponding meeting.
After the database transaction is successfully committed, an asynchronous processing event initiates the transcription pipeline.
This approach prevents long-running AI processing from blocking the upload request.

4.4 Speech-to-Text Processing

The transcript and its segments are stored in PostgreSQL.
The uploaded meeting recording is sent to the FastAPI AI service, where the local Whisper model converts the audio into text [1]. The system also detects the language and generates timestamped transcript segments. Each segment contains the spoken text, its start and end times, and its position in the transcript. The complete transcript and its segments are then stored in PostgreSQL for later access, search, and retrieval.

4.5 AI Meeting Understanding and Knowledge Extraction

After transcription, the transcript is passed to the LLM through the OpenRouter API. The LLM is instructed to identify structured meeting information, including:

- **Executive Summary** - A concise overview of the meeting.
- **Key Topics** - Major subjects discussed during the meeting.
- **Key Decisions** - Important decisions or agreed outcomes.
  - **Action Items** - Tasks identified from the discussion, including:
  - Task description
  - Assignee, when available
  - Due date, when available

The AI response is constrained to a structured JSON format and validated before being stored in PostgreSQL.

4.6 Knowledge Storage

Structured organizational knowledge is stored in PostgreSQL. The main entities include:

- Users
- Meetings
- Meeting Audio
- Meeting Transcripts
- Transcript Segments
- Meeting Knowledge
- Meeting Topics
- Meeting Decisions
- Action Items

This relational structure allows information to remain associated with the original meeting and user.

4.7 Semantic Indexing

The transcript is divided into smaller, overlapping chunks to make it easier to retrieve relevant information during a search. Each chunk typically contains around 700 characters, with a maximum size of 1000 characters and an overlap of 150 characters to preserve context between consecutive chunks. Metadata such as the meeting ID, transcript ID, chunk index, timestamps, and language is stored along with each chunk to help identify its source. The chunks are then converted into 384-dimensional vector embeddings using the all-MiniLM-L6-v2 sentence-transformer model, following the sentence-embedding approach described by Reimers and Gurevych [2], and stored in ChromaDB for semantic search and retrieval [8].

4.8 Semantic Search and RAG

When a user submits a natural-language query, the query is converted into an embedding. ChromaDB performs similarity-based retrieval to identify transcript chunks whose semantic meaning is closest to the query [2], [8]. The retrieval process is also restricted to meetings owned by the authenticated user.

The retrieved chunks are then assembled into a context block containing transcript text and timestamp information. The context is supplied to the LLM through a RAG prompt. The LLM generates an answer based on the retrieved meeting information rather than relying solely on its general knowledge, following the retrieval-augmented generation approach [3], [4].

The system returns:

- Generated answer
- Relevant meeting
- Transcript reference
- Chunk reference
- Start timestamp
- End timestamp

This allows the user to trace the answer back to the source meeting content.

4.9 Security and User Isolation

The platform implements stateless JWT-based authentication. Passwords are securely hashed using BCrypt. For semantic search, the authenticated user's owned meeting IDs are obtained by the Spring Boot backend and used to restrict vector retrieval. Therefore, one user cannot retrieve meeting content belonging to another user through semantic search.

4.10 Technologies Used

Table 02: Technologies used.

Layer	Technology	Role
Frontend	React 19	Web interface
Build Tool	Vite 6	Frontend development/build
Backend	Java 21 + Spring Boot	REST APIs and business logic
Security	Spring Security + JWT	Authentication and authorization
Password Security	BCrypt	Password hashing
ORM	Spring Data JPA / Hibernate	Database interaction
AI Service	Python + FastAPI	AI processing
Speech-to-Text	OpenAI Whisper	Audio transcription
LLM	OpenRouter-hosted LLM	Meeting understanding and RAG
Embeddings	Sentence-Transformers	Text-to-vector conversion

Relational DB	PostgreSQL	Structured knowledge storage
Vector DB	ChromaDB	Semantic vector storage/retrieval
Deployment	Vercel + Cloudflare Tunnel	Public frontend and backend connectivity

5. Research Methodology

The research methodology follows an **experimental and iterative software-development approach**. The system is developed and evaluated through successive stages:

Requirement Analysis → Literature Review → System Design → Module Development → AI Integration → System Integration → Testing → Validation → Deployment

5.1 Data Preparation

The system processes several types of meeting data:

- Audio recordings
- Transcript text
- Timestamped transcript segments
- Structured meeting knowledge
- Transcript chunks
- Vector embeddings
- Meeting metadata

Audio recordings are processed using Whisper to generate transcripts. The transcript is then segmented into overlapping chunks for semantic indexing.

5.2 Model and Analysis Strategy

The system uses different AI models and components for specific stages of meeting understanding and retrieval. OpenAI Whisper is used for speech recognition and converts the uploaded meeting audio into textual transcripts with timestamps. For knowledge extraction, an OpenRouter-hosted LLM processes the transcript and generates structured information such as summaries, topics, decisions, and action items, informed by research on meeting summarization and action-item extraction [5], [6].

For semantic representation, the system uses Sentence-Transformers all-MiniLM-L6-v2, which generates 384-dimensional dense embeddings from transcript chunks. These embeddings are stored in ChromaDB, where cosine-similarity-based retrieval is used to identify transcript content relevant to a user query [8]. The retrieved context is then provided to the LLM through a Retrieval-Augmented Generation (RAG) prompt, allowing the system to generate answers grounded in the relevant meeting information and provide corresponding references [3], [4].

5.3 Experimental Procedure

The experimental procedure follows the complete workflow implemented in the proposed system. A user first registers and logs into the platform and creates a meeting before uploading the corresponding recording. The uploaded audio is processed asynchronously, after which Whisper generates a timestamped transcript. The transcript is then passed to the LLM for extraction of structured meeting knowledge, including summaries, topics, decisions, and action items.

For semantic retrieval, the transcript is divided into smaller chunks and converted into embeddings that are stored in ChromaDB. When a user submits a natural-language query, the query is converted into an embedding and compared with the stored transcript embeddings to retrieve relevant content. The retrieved information is assembled as contextual input for the LLM, which generates a context-grounded response. The final response is returned along with relevant meeting and timestamp references.

5.4 Testing and Evaluation

The implemented system was evaluated at multiple levels to verify the correctness of its individual components and the complete workflow. The Spring Boot backend was tested using 61 test cases, with all 61/61 tests passing, covering APIs, business logic, security, and database interactions. The FastAPI-based AI service was independently tested using 33 test cases, with 33/33 tests passing. The production frontend build also completed successfully without build errors.
End-to-end functionality was evaluated through a complete workflow covering system health verification, user registration and login, authenticated user access, meeting creation, audio upload, asynchronous AI processing, transcript and knowledge retrieval, and semantic search with RAG. All 9/9 end-to-end stages passed successfully, confirming that the major components of the implemented system work together as intended.

5.5 Experimental Environment

The performance evaluation was conducted on a local Apple Silicon system using Python 3.12.14 and PyTorch. The speech recognition component used the locally cached OpenAI Whisper tiny model, while semantic embedding generation used the sentence-transformers/all-MiniLM-L6-v2 model. ChromaDB was used for vector storage and similarity-based retrieval.

The experiments were performed locally without external API calls. The embedding and retrieval benchmarks used the existing transcript collection containing 53 chunks. The PostgreSQL database contained 17 meetings before and after testing, and the ChromaDB collection size remained unchanged.

Table 03: Experimental configuration

Parameter	Configuration
Speech recognition model	OpenAI Whisper tiny
Embedding model	all-MiniLM-L6-v2
Embedding dimensionality	384
Vector database	ChromaDB 1.5.9
Transcript chunks evaluated	53
Meeting records in PostgreSQL	17
Embedding execution device	Apple Silicon MPS
Retrieval values ((K))	1, 3, 5, 10
Execution mode	Local, offline

5.6 Speech Recognition Performance

The Whisper transcription component was evaluated using six local audio recordings with a combined duration of approximately 527.25 seconds. Each recording was processed three times, resulting in 18 measured trials. All trials completed successfully. The measurements represent the latency of the complete transcription call and do not measure transcription accuracy.

Table 04: Whisper transcription performance

Audio file	Audio duration (s)	Mean latency (s)	Mean real-time factor
Meeting 1	65.691	1.446	0.0220
Meeting 2	90.057	2.124	0.0236
Meeting 3	102.014	2.226	0.0218
Meeting 4	83.097	1.777	0.0214
Meeting 5	83.233	1.821	0.0219

Meeting 6	103.160	2.174	0.0211
Overall mean	—	**1.928**	**0.0220**

The real-time factor (RTF) is calculated as the transcription-call latency divided by the corresponding audio duration. An RTF below 1 indicates that processing completed faster than the duration of the input audio. The measured mean RTF of 0.022 indicates that transcription was substantially faster than real time under the tested local configuration. These results describe processing speed only and do not establish the accuracy of the generated transcripts.

5.7 Embedding Generation Performance

The semantic embedding component was evaluated using five representative queries, with 30 repetitions per query, giving 150 measured trials. Model loading was measured separately and took 0.4948 seconds. The overall median embedding latency was 10.329 milliseconds.

Table 05: Embedding generation latency

Query ID	Query category	Mean latency (ms)	Median latency (ms)	Standard deviation (ms)
Q1	Architecture and services	9.408	9.295	0.781
Q2	Technology stack and AI models	15.656	9.275	26.729
Q3	Vector database and storage	14.821	9.937	26.761
Q4	Deliverables and project progress	11.688	11.627	0.350
Q5	Semantic search and retrieval	11.018	11.156	0.538
Overall	**150 trials**	**12.518**	**10.329**	**16.859**

Two queries exhibited isolated latency spikes during the first measured repetition, which increased their mean latency and standard deviation. Therefore, the median is also reported to describe typical observed latency. The results demonstrate that query embedding generation completed within milliseconds for the tested queries, although latency may vary with execution conditions and hardware.

5.8 Vector Retrieval Performance

ChromaDB retrieval was evaluated at four values of (K), representing the number of nearest transcript chunks requested. Five query vectors were reused across the retrieval conditions to avoid including embedding-generation time in the retrieval measurements. Each (K) setting was measured 150 times, resulting in 600 retrieval trials.

Table 06: ChromaDB retrieval latency by top-(K)

Top-(K)	Number of trials	Mean latency (ms)	Median latency (ms)	Standard deviation (ms)
1	150	0.428	0.350	0.188
3	150	0.423	0.384	0.093
5	150	0.455	0.447	0.062
10	150	0.552	0.542	0.061

All retrieval trials completed successfully. The median retrieval latency increased from 0.350 ms at (K=1) to 0.542 ms at (K=10). This indicates a small latency increase when requesting more results from the evaluated collection. However, the collection contained only 53 chunks; performance on substantially larger collections requires separate evaluation.

5.9 Discussion

The experiments indicate that the tested local implementation can perform speech transcription, semantic embedding generation, and vector retrieval with low measured latency under the evaluated conditions. Whisper processed the six recordings faster than real time, while embedding generation had a median latency of 10.329 ms. ChromaDB retrieval had a median latency below 0.6 ms for all tested top-(K) values.

Embedding generation took longer than the vector lookup itself in these measurements. Consequently, query encoding is a more prominent component of the measured local semantic retrieval latency than nearest-neighbour lookup. These results apply to the tested hardware, cached models, five representative queries, and 53-chunk collection. They should not be interpreted as a measure of end-to-end response time, retrieval relevance, transcription accuracy, or performance at production scale.

5.10 Overall Research Methodology

The overall research methodology followed a structured development process beginning with problem identification and literature review, followed by requirement analysis and system architecture design. The individual system modules were then developed and integrated with the speech recognition and AI components. The integrated design is informed by prior work in speech recognition, meeting summarization, sentence embeddings, and RAG [1]–[5].

After the development of semantic indexing and the RAG-based retrieval mechanism, the complete system was integrated and subjected to functional, component-level, and end-to-end testing. The validated system was then deployed and evaluated based on its ability to transform unstructured meeting recordings into structured organizational knowledge and provide semantic access to historical meeting information.

This methodology provides an end-to-end approach for converting meeting recordings into structured, searchable, and context-aware organizational knowledge.semantic access to historical meeting information.

6. Limitations and Recommendations

The proposed system relies on computationally intensive components such as Whisper, embedding models, PyTorch, and ChromaDB for speech processing and semantic retrieval. These components require more memory and processing resources than conventional web applications, particularly when larger AI models are used.

Recommendation: Future deployments can use persistent high-memory or GPU-based cloud infrastructure to improve processing performance and support larger AI models.

6.1 Speech Recognition Accuracy

The accuracy of speech transcription depends on the selected Whisper model and the quality of the meeting recording. Smaller models provide faster processing with lower resource requirements, but their performance may be affected by background noise, multiple speakers, accents, or complex conversations.

Recommendation: Larger Whisper models can be used when sufficient computational resources are available to improve transcription accuracy.

6.2 Speaker Identification

The current system generates timestamped transcript segments but does not provide complete speaker diarization. As a result, the transcript does not explicitly associate every segment with an individual speaker.

Recommendation: Future versions can integrate speaker diarization techniques to identify and distinguish individual speakers within meeting recordings.

6.3 LLM Dependency

The quality of structured knowledge extraction and RAG-based responses depends partly on the selected Large Language Model and its response behaviour. Variations in model capability can affect the consistency and quality of generated summaries, decisions, topics, and action items.

Recommendation: Future versions can support multiple LLM providers and models, along with fallback mechanisms to improve reliability and flexibility.

6.4 Semantic Retrieval

The quality of semantic retrieval depends on factors such as transcript chunking, embedding quality, and the formulation of user queries. In some cases, relevant information may not be retrieved if the selected chunks do not sufficiently represent the user's query.

Recommendation: Future work can investigate adaptive chunking, reranking models, hybrid keyword and vector search, and other advanced retrieval techniques to improve retrieval relevance.

6.5 Deployment Dependency

The current demonstration architecture uses a local backend and AI processing environment connected through a Cloudflare Quick Tunnel. This approach is suitable for development and demonstration but requires the local processing environment to remain active for continuous system availability.

Recommendation: Future deployment can move the backend and AI processing components to persistent cloud infrastructure to provide continuous availability and improve scalability.

7. Conclusion

The proposed AI-Powered Organizational Knowledge Management Platform for Intelligent Meeting Understanding and Semantic Knowledge Retrieval provides an integrated solution for converting unstructured meeting recordings into structured and searchable organizational knowledge. The system combines Speech-to-Text transcription, Large Language Model-based knowledge extraction, relational data storage, semantic embeddings, vector retrieval, and Retrieval-Augmented Generation within a unified platform.

The proposed workflow converts meeting audio into timestamped transcripts using Whisper and processes the transcripts using an LLM to extract executive summaries, topics, decisions, and action items. The transcript is also divided into chunks and converted into embeddings for storage in ChromaDB. This enables semantic retrieval of relevant meeting information when users submit natural-language queries.

The RAG component uses the retrieved transcript context to generate context-grounded responses along with relevant meeting and timestamp references. The platform also incorporates JWT-based authentication and user-specific meeting filtering to maintain separation between users and their stored meeting information.

The implemented system was evaluated through backend, AI-service, frontend build, and end-to-end testing. The results included 61/61 successful backend tests, 33/33 successful AI-service tests, and 9/9 successful end-to-end workflow stages. These results demonstrate the successful integration of the major components required for meeting transcription, knowledge extraction, semantic retrieval, and RAG-based question answering.

Overall, the proposed system transforms meeting recordings from passive stored files into structured, searchable, and reusable organizational knowledge. Future development can extend the platform through speaker diarization, multilingual processing, persistent cloud AI infrastructure, improved retrieval techniques, and integration with platforms such as Google Meet, Zoom, and Microsoft Teams.

8. References

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision,” in Proceedings of the 40th International Conference on Machine Learning (ICML), vol. 202, pp. 28492–28518, 2023. Available: https://proceedings.mlr.press/v202/radford23a.html

[2] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. Available: https://aclanthology.org/D19-1410/

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020. Available: NeurIPS paper

[4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023. doi: 10.48550/arXiv.2312.10997. Available: https://arxiv.org/abs/2312.10997

[5] V. Rennard, G. Shang, J. Hunter, and M. Vazirgiannis, "Abstractive Meeting Summarization: A Survey," Transactions of the Association for Computational Linguistics, vol. 11, pp. 861–884, 2023. doi: 10.1162/tacl_a_00578. Available: https://aclanthology.org/2023.tacl-1.49/

[6] S. Sadia, F. Adeeba, S. Shams, and S. Hussain, "Leveraging LLMs for Action Item Identification in Urdu Meetings: Dataset Creation and Comparative Analysis," Information Processing & Management, vol. 62, no. 2, Art. 104071, 2025. doi: 10.1016/j.ipm.2025.104071. Available: https://www.sciencedirect.com/science/article/pii/S0306457325000135

[7] M. Alavi and D. E. Leidner, "Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," MIS Quarterly, vol. 25, no. 1, pp. 107–136, 2001.
Available: https://aisel.aisnet.org/misq/vol25/iss1/6/

[8] Chroma, "Chroma Documentation," Chroma, Open-source AI application database. Available: https://docs.trychroma.com/

[9] Spring Boot, "Spring Boot Documentation," Spring. Available: https://docs.spring.io/spring-boot/documentation.html