IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

AI-Based Meeting Minutes Generator

Shinde Sanskruti Kailas, Shinde Nikita Rajendra, Ohal Isha Vinayak,
Garje Pooja Baban, Prof. U. B. Shelake
Department of Computer Engineering
Adsul's Technical Campus, Chas, Ahmednagar

Abstract : *Meetings are an essential means of communication in academic institutions, business organizations, and project environments. They facilitate information sharing, problem-solving, decision-making, and task allocation among participants. Despite their importance, preparing meeting minutes remains a largely manual and time-consuming activity. The process requires careful review of discussions, identification of critical decisions, extraction of action items, and accurate documentation of responsibilities and deadlines. In lengthy meetings involving multiple participants, important information may be missed, resulting in incomplete or inconsistent records.*

The motivation behind this project is to reduce the effort involved in manual documentation and improve the accuracy, consistency, and accessibility of meeting records. Recent advancements in Artificial Intelligence (AI), Natural Language Processing (NLP), and Automatic Speech Recognition (ASR) make it possible to automate several aspects of meeting analysis and summarization. These technologies can reduce repetitive documentation tasks and allow participants to focus on discussion rather than note-taking.

The proposed AI-Based Meeting Minutes Generator is a web-based application designed to generate concise and structured meeting minutes from meeting transcripts. The system accepts meeting transcripts as input and may support audio recordings through speech-to-text conversion when an appropriate component is integrated. NLP techniques are applied to analyze the transcript, generate a meaningful summary, identify key discussion points, extract decisions, and detect action items. The application attempts to identify responsible individuals and associated deadlines only when such information is explicitly available in the conversation. The system is proposed using Python and Flask, with HTML, CSS, and JavaScript for the user interface. Generated minutes can be reviewed, edited, stored, and exported by the user. Human verification remains necessary because automated systems may omit or misinterpret information.

Keywords: Artificial Intelligence, Natural Language Processing, Automatic Summarization, Speech Recognition, Meeting Minutes Generation, Action Item Extraction, Flask, Web Application

I. INTRODUCTION

Meetings are used to share information, solve problems, coordinate work, and make decisions. Meeting minutes provide a written record of important discussion points, decisions, and responsibilities. Preparing them manually can be difficult when discussions are lengthy or several topics are covered.

Natural Language Processing (NLP) can analyze conversational text, summarize lengthy content, and identify relevant information. Speech recognition can convert supported audio into text. These technologies can assist minute preparation, but generated content must be checked against the original transcript.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-41067

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

Project Idea

The project proposes a web application that accepts a meeting transcript and produces a structured draft containing a summary, decisions, and action items. The user can review, edit, and export the result. Audio input may be added through a compatible speech-to-text component.

Motivation

Reduce repetitive manual summarization.
Improve consistency in meeting records.
Make decisions and assigned tasks easier to locate.
Apply AI, NLP, and web-development concepts to a practical problem.

Existing System

In the conventional process, a participant or secretary takes notes, reviews the notes or recording, organizes the information, and prepares a document. This process depends on human attention and may require repeated listening or reading.

Limitations of Existing Approach

Manual preparation requires time and concentration.
Important decisions or tasks may be missed.
Minutes may vary in structure between note-takers.
Reviewing long recordings or transcripts is repetitive.
Searching past outcomes is difficult when records are inconsistent.

Proposed System

The proposed application validates transcript input and processes it using NLP. It generates a summary and identifies decisions and action items. The user reviews and edits the draft before saving or exporting it. Audio transcription is optional and depends on integration of a suitable component.

Advantages

- Produces a structured draft of meeting minutes.
- Reduces repetitive documentation work.
- Separates summary, decisions, and action items.
- Allows correction before final export.
- Uses a modular design for easier maintenance.

Limitations of Proposed System

Output quality depends on transcript quality, language, model capability, and input length. Audio transcription may introduce errors. Ambiguous statements may not identify an owner or deadline. The system cannot guarantee perfect factual accuracy and requires human review. Live meeting integration and automatic calendar updates are outside the core scope.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

II. PROBLEM DEFINITION AND SCOPE

Problem Definition

The main problem is the time and effort required to prepare accurate and consistent minutes from unstructured conversations. Long transcripts may contain repetitions, side discussions, and incomplete statements that make key outcomes difficult to identify.

Problem Statement

To design and develop a web-based application that processes meeting transcripts using NLP and generates structured, editable minutes containing a concise summary, key decisions, and action items, while allowing users to review and export the result.

Goals and Objectives

- Accept and validate meeting transcript input.
- Generate a concise summary.
- Identify explicit decisions.
- Extract action items and supported owner/deadline details.
- Provide review and editing facilities.
- Export approved minutes.
- Evaluate the system with representative test cases.

Statement of Scope

Included features are transcript entry or upload, input validation, summary generation, decision and actionitem extraction, optional owner/deadline identification, editable result sections, and document export or basic record storage when implemented.

Constraints and Feasibility

Poor audio quality may cause transcription errors; transcript correction and direct text input should be allowed. Long inputs may require limits or chunking. Ambiguous statements must not lead to invented decisions, owners, or deadlines. External service failures should produce clear errors. Sensitive data should be retained only when required.

Technical: Python, Flask, and an NLP library or hosted model are suitable for a prototype. Audio requires a compatible speech-to-text component. Operational: The workflow is submit, generate, review, edit, and export. Economic: Open-source tools and SQLite can reduce initial cost; hosted APIs and deployment may incur charges.

Resources: A standard Windows/Linux computer, Python, Flask, HTML, CSS, JavaScript, VS Code, and internet access for hosted APIs are sufficient. Audio hardware is optional.

III. SOFTWARE REQUIREMENT SPECIFICATION

This specification defines the users, inputs, processing stages, outputs, and quality requirements of the AIBased Meeting Minutes Generator. It provides a basis for implementation and testing.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

Usage Scenario

The meeting organizer opens the application, enters a meeting title and transcript or uploads a supported file, and submits the content. The system validates and processes the input. The generated summary, decisions, and action items are displayed for review. The user edits incorrect or incomplete content and exports or saves the approved minutes.

User Profiles

Meeting Organizer: Submits content, generates minutes, reviews, edits, and exports the result.
Administrator (optional): Configures the application or manages records if these functions are implemented.
NLP/STT Component: Local or external processing service.
Database: Stores meeting records when history is enabled.

Functional Requirements

FR-01 Input Submission: The system shall accept meeting transcript text.
FR-02 Input Validation: The system shall reject empty or invalid submissions.
FR-03 File Upload: Upload shall accept supported formats and configured file sizes.
FR-04 Audio Transcription: Supported audio shall be transcribed only if audio support is implemented.
FR-05 Summarization: The system shall generate a concise summary.
FR-06 Decision Extraction: The system shall identify decisions supported by the transcript.
FR-07 Action Items: Extract tasks and include owners or deadlines only when supported by the source.
FR-08 Review and Editing: The user shall be able to inspect and edit generated minutes.
FR-09 Export: The system shall export reviewed minutes when implemented.
FR-10 Storage: The system shall save and retrieve records when persistence is enabled.

Non-Functional Requirements

Usability: Clear labels, readable text, separate result sections, and understandable error messages.
Performance: Display processing status and record actual processing time during evaluation.
Reliability: Handle invalid input, unavailable services, and export errors without crashing.
Security and Privacy: Validate uploads, protect API keys, limit access to stored records, and disclose whether data is processed locally or externally.
Maintainability: Keep interface, NLP, transcription, database, and export modules separate where practical.
Accuracy: Allow comparison with the source transcript; human verification is required.

Interface Requirements

The interface should provide a transcript input area, optional upload control, Generate button, processing status, separate output sections, edit controls, and Export button.

Database Requirements

SQLite may be used for saved history. Suggested fields include meeting_id, title, meeting_date, source_type, transcript_text, summary_text, decisions, action_items, created_at, and updated_at. If storage is not implemented, this must be stated as a limitation.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

SDLC Model

An iterative model is suitable because NLP components and output quality require testing and refinement. Stages include requirement analysis, design, implementation, testing, evaluation, and refinement.

IV. SYSTEM DESIGN

System Overview

The system uses a modular web architecture. The browser interface collects meeting content. Flask validates the request and controls processing. If audio input is enabled, a speech-to-text module converts audio into text. The NLP module generates a summary and extracts decisions and action items. Results are returned for user review. Database and export modules save records or create a downloadable document when implemented.

System Architecture

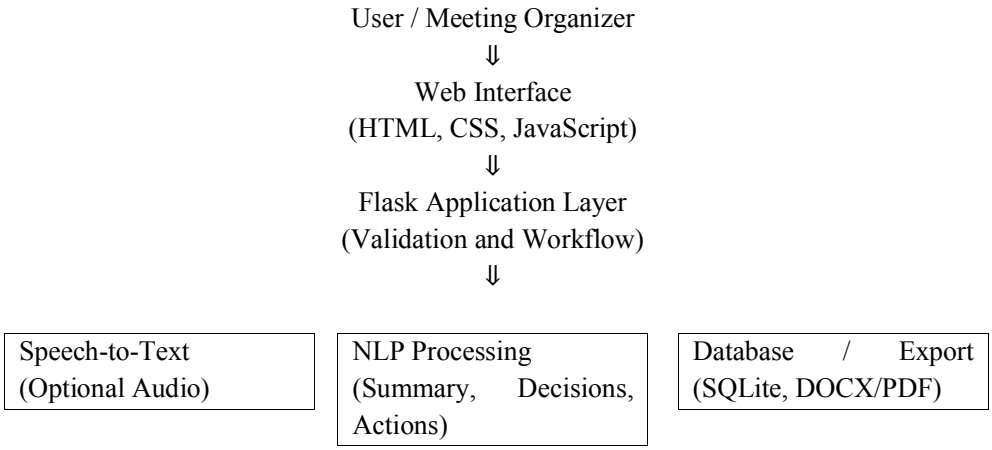


Figure 4.1: Proposed System Architecture

The modular design allows the NLP component to be replaced without redesigning the interface. Speech-to text is required only when audio input is supported.

Data Flow Diagram – Level 0

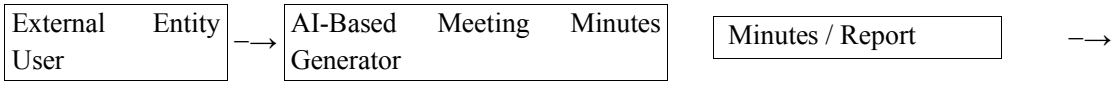


Figure 5.2: DFD Level 0 (Context Diagram)

At Level 0, the user provides meeting content and receives generated minutes. If an external service is used, the application sends the minimum required content and receives a transcription or processed result. If processing is local, the external-service entity should be removed.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

Sequence Diagram

User	Web UI	Flask	NLP / Storage
Submit transcript			
	Validate input		
		Process transcript	
			Return summary and actions
	Display result		
Review / export	Send request	Create file / save	

Figure: Sequence Diagram

The user submits content through the interface. Flask validates the request and invokes transcription or NLP processing as required. The result is displayed, optionally saved, and exported when requested.

Example Output Structure

Table: Expected Meeting Minutes Output

Section	Expected Content
Meeting details	Title, date, participants if provided
Summary	Brief overview of topics and outcomes
Key discussion points	Important topics or issues
Decisions	Decisions supported by the transcript
Action items	Task, owner, and deadline when available
Review status	Draft / reviewed by user
Export	Approved minutes document

Database Design

A single-table prototype is sufficient for a basic application. If multiple action items are stored separately, an ActionItems table can be linked to Meetings using meeting_id. A Decisions table may also be used.

Table: Proposed Database Tables

Table	Suggested Fields
Meetings	meeting_id (PK), title, meeting_date, transcript_text, summary_text, created_at, updated_at
Decisions (optional)	decision_id (PK), meeting_id (FK), decision_text
ActionItems (optional)	action_id (PK), meeting_id (FK), task_text, owner, deadline, status

Processing Modules

Input Module: Accepts transcript text and checks that the submission is valid.
Transcription Module (optional): Converts supported audio into text.
Text Processing Module: Prepares transcript text for analysis.
Summarization Module: Produces a concise overview.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, November 2026 Impact Factor: 8.2

Extraction Module: Identifies explicit decisions and action items.
Review Module: Allows users to correct and approve the draft.
Storage and Export Module (optional): Saves history and creates a document.

V. TESTING AND EVALUATION

Testing Approach

Testing should confirm that input validation, transcript processing, summarization, extraction, editing, and export work as expected. Optional audio and database functions should be tested only if implemented.

Test Cases

Test ID	Input / Action	Expected Result
TC-01	Valid transcript	System generates a draft
TC-02	Empty input	Validation message displayed
TC-03	Unsupported file	File rejected with clear message
TC-04	Explicit decision	Decision appears in decision section
TC-05	Task with stated owner or deadline	Supported details extracted
TC-06	Task with no owner or deadline	Missing details are not invented
TC-07	Edit generated minutes	User changes are retained
TC-08	Export reviewed minutes	Readable document is generated

Evaluation

Generated summaries and extracted items should be compared with manually prepared reference minutes. Report precision, recall, F1-score, processing time, and observed errors. Human review should assess whether important facts and context are preserved. Numerical results are not claimed in this proposal.

VI. SUMMARY AND CONCLUSION

Summary

The AI-Based Meeting Minutes Generator is proposed to reduce repetitive work involved in preparing meeting documentation. It accepts meeting content, processes it using NLP and optionally speech recognition, and generates an editable draft containing a summary, decisions, and action items. It is designed as a web application using Python and Flask, with a browser-based interface and optional database storage and document export.

The report defines the project objectives, scope, requirements, proposed architecture, processing workflow, database design, and testing plan. The architecture separates the interface, application workflow, transcription, NLP processing, storage, and export.

A key requirement is that the system must not invent owners, deadlines, or decisions that are not supported by the transcript. Generated minutes should be reviewed by the user before distribution.

Conclusion

The proposed project applies artificial intelligence and natural language processing to meeting documentation. Automating parts of summarization and action-item extraction may reduce drafting time and make meeting outcomes easier to review. The effectiveness depends on input quality and the selected NLP

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, November 2026
Impact Factor: 8.2

and speechrecognition components. A generated summary is not automatically an authoritative record. Human review, clear error handling, and responsible data handling are essential.

The final claim of success should be based on the implemented system and actual testing. Limitations should be acknowledged clearly, and future work should be proposed only where it extends the tested prototype.

REFERENCES

[1]. See, P. J. Liu, and C. D. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," Proceedings of ACL, 2017.
[2]. Y. Liu and M. Lapata, "Text Summarization with Pretrained Encoders," Proceedings of EMNLP-IJCNLP, 2019.
[3]. M. Zhong et al., "QMSum: A New Benchmark for Query-Based Multi-Domain Meeting Summarization," Proceedings of NAACL-HLT, 2021.
[4]. K. Gliwa et al., "SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization," 2019.
[5]. Y. Zhu et al., "DialogSum: A Real-Life Scenario Dialogue Summarization Dataset," Findings of ACLIJCNLP, 2021.
[6]. J. Carletta et al., "The AMI Meeting Corpus: A Pre-announcement," 2005.
[7]. J. Zhang et al., "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," ICML, 2020.
[8]. M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," ACL, 2020.
[9]. A Radford et al., "Robust Speech Recognition via Large-Scale Weak Supervision," 2023.
[10]. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," EMNLP: System Demonstrations, 2020.
[11]. Python Software Foundation, "Python Documentation." Available: https://docs.python.org/.
[12]. Pallets Projects, "Flask Documentation." Available: https://flask.palletsprojects.com/.