**INDEX**

1. Abstract
2. Introduction
3. System Analysis
   1. Existing System
   2. Prompted System
4. Feasibility Report
   1. Technical Feasibility
   2. Operational Feasibility
   3. Economical Feasibility
5. System Requirement Specification Document
   1. Module Description
   2. Process Flow
   3. SDLC Methodology
   4. Software Requirements
   5. Hardware Requirements
6. System Design
   1. E-R Diagram
   2. DFD
   3. UML
7. Technology Description
8. Testing & Debugging Techniques
9. Bibliography

**ABSTRACT**

In today’s fast-paced professional environment, meetings generate vast amounts of information, often leading to challenges in documentation, knowledge retention, and follow-up on key action items. The **AI Meeting Summarizer** is an intelligent solution designed to automatically transcribe spoken content from meeting recordings, summarize the key discussion points, and extract actionable insights. Leveraging advanced speech recognition (OpenAI Whisper) and natural language processing (Hugging Face Transformers), the system delivers concise and structured summaries that save time and enhance productivity. With a simple user interface, users can upload recordings, receive accurate transcripts, and download professionally formatted reports. This project aims to reduce manual effort in note-taking, increase information accessibility, and improve organizational efficiency through automation and AI-driven analysis.

OBJECTIVE

The primary objective of the **AI Meeting Summarizer** project is to develop an intelligent, automated system that:

* Accurately transcribes meeting audio into text using state-of-the-art speech recognition.
* Generates concise summaries that capture key discussion points, decisions, and action items.
* Provides a user-friendly interface for uploading, processing, and downloading meeting notes.
* Saves time, reduces manual effort in documentation, and improves organizational productivity.
* Ensures accessibility by supporting multiple languages and lightweight deployment.

INTRODUCTION

Meetings are an integral part of decision-making and collaboration within organizations. However, participants often face challenges such as difficulty recalling discussion details, missing critical action items, and spending significant time on manual documentation. Traditional note-taking methods are prone to human error, inconsistent formatting, and loss of important information.

The **AI Meeting Summarizer** addresses these challenges by combining **speech-to-text transcription** with **automatic summarization** and **action item extraction**. Using state-of-the-art AI models, the system converts audio recordings into structured text, highlights essential points, and generates downloadable reports in formats such as Word or PDF.

This solution is built with scalability and user-friendliness in mind, utilizing frameworks like **Gradio** for interactive interfaces, **Whisper** for accurate multi-language transcription, and **Transformers** for lightweight summarization models. Designed as a productivity tool for businesses, educational institutions, and remote teams, the AI Meeting Summarizer ensures that meeting knowledge is retained, shared, and actionable.

By automating the entire pipeline—from audio capture to report generation—the system empowers users to focus more on collaboration and decision-making rather than administrative tasks. Ultimately, this project demonstrates how AI technologies can streamline communication workflows, improve efficiency, and bridge the gap between spoken and written information.

Problem Area of the Project

Organizations and teams face several challenges in handling meetings effectively:

* **Time-consuming manual note-taking**: Participants must focus on writing instead of actively engaging.
* **Loss of critical information**: Key points and decisions may be forgotten or misinterpreted.
* **Inefficient knowledge sharing**: Meeting outcomes are often not properly documented or circulated.
* **Information overload**: Lengthy discussions make it difficult to recall or extract actionable insights.
* **Inconsistency**: Notes vary depending on the participant’s attentiveness or writing style.

These issues lead to **reduced productivity, miscommunication, and missed opportunities for action**.

Purpose of the Project

The purpose of the **AI Meeting Summarizer** is to **bridge the gap between spoken discussions and actionable documentation** by:

* Automating transcription and summarization, thereby minimizing human error.
* Providing **structured, reliable meeting notes** that can be easily stored and shared.
* Freeing participants to focus on meaningful contributions rather than administrative tasks.
* Supporting remote and hybrid teams by ensuring no participant misses out on critical information.
* Enhancing productivity and decision-making by making key outcomes instantly available.

In essence, the project aims to **transform unstructured audio data into structured knowledge**, empowering teams and organizations to work smarter and more efficiently.

Overview of the Project

The **AI Meeting Summarizer** is designed as a productivity tool that automates the process of meeting documentation. The system allows users to upload an audio file of a meeting, after which it:

1. **Transcribes** the speech into accurate text using the Whisper model.
2. **Processes** the transcript with NLP models to summarize lengthy content.
3. **Identifies action items and decisions** from the discussion.
4. **Exports** the summarized results into professional formats (PDF/Word).

The project leverages **open-source AI frameworks** such as Whisper (for speech-to-text) and Hugging Face Transformers (for summarization). The user interacts with the system through a lightweight web interface built on **Streamlit/Gradio**, making it easy to deploy and use in real-world scenarios.

System Analysis

**a. Existing System**

* Meetings are usually recorded (audio/video) or documented manually by note-takers.
* Common practices:
  + Writing **manual minutes of meeting (MoM)**.
  + Relying on **human transcription services** (time-consuming, expensive).
  + Using basic note-taking apps without **AI-powered summarization**.
* **Problems with existing system:**
  + Manual transcription is slow and prone to errors.
  + Reviewing long recordings is inefficient.
  + Key decisions and action items often get missed.
  + Hiring human transcribers incurs cost.
  + No real-time or automated summarization support.

**b. Proposed (Prompted) System: AI Meeting Summarizer**

* Upload meeting audio → system transcribes using **Whisper** (ASR).
* AI-powered summarizer (e.g., **transformer models like BART, T5**) generates concise summaries.
* Provides:
  + **Full transcript** for reference.
  + **Key points** extracted.
  + **Action items** highlighted.
* Features:
  + Supports multiple languages.
  + User-friendly web app via **Gradio**.
  + Runs lightweight models for faster processing.
* **Advantages:**
  + Saves time (minutes instead of hours).
  + Cost-effective (uses open-source models).
  + Improves productivity & decision-making.

Feasibility Report

**a. Technical Feasibility**

* **Technology stack:**
  + Python, Hugging Face Transformers, OpenAI Whisper, Gradio.
  + Pre-trained models (no need for large custom datasets).
  + Can run on standard CPU, though GPU improves performance.
* **Integration feasibility:**
  + Easy integration with meeting platforms (via upload feature).
  + Cloud deployment possible (Heroku, AWS, GCP).
* **Scalability:**
  + Start small (local/offline), expand to cloud APIs.
  + Possible real-time transcription in future.

**Conclusion:** Technically feasible with current open-source tools.

**b. Operational Feasibility**

* **Ease of use:**
  + Simple UI for uploading audio → automatic summary.
  + Minimal training required for users.
* **Adoption potential:**
  + Teams, corporates, educational institutions can benefit.
  + Reduces burden on employees/students to take detailed notes.
* **Support & maintenance:**
  + Uses open-source, so community support is available.

**Conclusion:** High operational feasibility since it aligns with user needs.

**c. Economic Feasibility**

* **Costs:**
  + Development cost: minimal (open-source).
  + Hosting: low-cost cloud VM ($5–10/month).
  + No licensing fee (models are open-source).
* **Savings:**
  + Cuts cost of professional transcription services (~$1–2 per minute of audio).
  + Saves employees’ time → improves productivity.
* **ROI:**
  + High ROI, as it reduces operational inefficiency at very low investment.

**Conclusion:** Economically highly feasible.

## System Requirement Specification Document

## a) Module Description

1. **Auth & Workspace (optional/MVP-lite)**
   * Users (or anonymous in MVP), workspaces, project selection.
2. **Upload & Ingestion**
   * Accept MP3/WAV, validate size/format, store temp file, metadata capture.
3. **Transcription**
   * Run Whisper (tiny/base) → full text + timestamps (optional).
4. **Summarization**
   * Chunk transcript → summarize with T5-small/DistilBART → merge.
5. **Action Item Extraction**
   * Regex rules (baseline) or small LLM prompt → tasks with owners/dates (if found).
6. **Report Generation**
   * Compose Summary + Action Items (+ optional transcript) → DOCX/PDF export.
7. **Storage & Retrieval**
   * Persist meeting record, transcript, summary, action items, artifacts.
8. **UI/Delivery**
   * Gradio app: upload, progress, outputs, download; simple history view.

## b) Process Flow

1. User uploads audio → 2) Validate → 3) Transcribe → 4) Summarize → 5) Extract actions → 6) Persist → 7) Render & allow download.
2. **Exceptions**

* Invalid file → error toast.
* Transcription fails → retry once, show fallback.
* Summarization timeout → show transcript only.
* No action items → show “none detected”.

## c) Software Requirements

* **Runtime**: Python 3.10+
* **Framework/UI**: Gradio (or Streamlit)
* **AI/ML**: openai-whisper, transformers, torch
* **NLP helpers**: sentencepiece, (optional) sacremoses
* **Docs/Export**: python-docx (and/or reportlab for PDF)
* **Ops**: uvicorn/gunicorn (if converting to API), Docker (optional)
* **OS**: Windows/Linux/macOS; **FFmpeg** required for Whisper

## d) Hardware Requirements

* **Dev/Laptop**: 8 GB RAM (min), 4 CPU cores recommended
* **Server (CPU only)**: 2 vCPU, 4–8 GB RAM (for small models)
* **Disk**: ~2–5 GB for models + cache
* **GPU (optional)**: any CUDA GPU (4–8 GB VRAM) greatly speeds up STT & summarization

## DFD (Data Flow Diagrams)

**Level 0 (Context)**

Upload audio

AI Meeting Summarizer

USER

Report

(Action Items)

Summary

Transcript

**Level 1**

Audio file

Temp-storage

USER

Upload & Validate

Validate file path

Transcription Engine (Whisper)

Transcript

Summarization Engine (T5-small)

Summarization

Action Extractor (Regex)

Action items

Results

USER

UI render (Gradio)

# **Technology Description**

The **AI Meeting Summarizer** integrates several modern technologies to provide transcription, summarization, and reporting in an efficient and scalable manner:

### 1. ****Programming Language & Environment****

* **Python 3.10+**
  + Primary development language for AI/ML integration, rapid prototyping, and deployment.
  + Rich ecosystem of libraries for NLP, speech recognition, and reporting.

### 2. ****AI & NLP Frameworks****

* **OpenAI Whisper**
  + State-of-the-art speech-to-text (STT) system.
  + Handles multiple languages with high accuracy.
* **Hugging Face Transformers**
  + Provides lightweight summarization models like T5-small or DistilBART.
  + Enables text-to-text processing (summarization, action item extraction).
* **SentencePiece / Sacremoses**
  + Tokenizers required by certain MT/NLP models.

### 3. ****User Interface****

* **Gradio**
  + Python-native UI frameworks for building fast, interactive web apps.
  + Provides file upload, progress indicators, and report download options.

### 4. ****Reporting & Export****

* **python-docx**
  + Generates Word reports combining transcripts, summaries, and action items.
* **ReportLab (optional)**
  + Enables PDF report generation for distribution.

### 5. ****Supporting Tools****

* **FFmpeg**
  + Required for Whisper to preprocess/normalize audio files.
* **Docker (optional)**
  + Containerizes the application for consistent deployment across environments.
* **Uvicorn/Gunicorn (optional)**
  + For production API serving if scaling beyond MVP.

# **Testing & Debugging Techniques**

To ensure reliability and correctness, a mix of **unit tests, integration tests, and user acceptance tests** are applied:

### 1. ****Unit Testing****

* **Transcription Module**
  + Test audio file ingestion (valid/invalid formats).
  + Mock Whisper to verify pipeline correctness.
* **Summarization Module**
  + Validate output length reduction and coherence.
  + Check edge cases (very short or very long transcripts).
* **Action Item Extraction**
  + Test regex/LLM prompt outputs for known patterns.
  + Ensure empty or noisy transcripts don’t crash.

### 2. ****Integration Testing****

* Upload → Transcribe → Summarize → Extract → Report workflow tested as a whole.
* Database integration: verify persistence and retrieval of transcripts/summaries.
* UI integration tests for Gradio: simulate file uploads, button clicks, and downloads.

### 3. ****System & Performance Testing****

* Test long audio files (30–60 min) to measure transcription latency and memory usage.
* Benchmark summarization throughput using different model sizes (T5-small vs bart-large).
* Stress test: multiple users uploading files simultaneously (for multi-user deployment).

### 4. ****Debugging Techniques****

* **Verbose Logging**
  + Enable logging at each pipeline stage (upload, STT, summarization, report).
  + Store logs for each request in a separate session folder.
* **Model Dry Runs**
  + Run summarizer with test strings before deployment to confirm tokenizer/model compatibility.
* **Graceful Error Handling**
  + Catch and display user-friendly errors for missing dependencies (e.g., FFmpeg, sentencepiece).
* **Incremental Testing**
  + Run pipeline step-by-step with small audio (1–2 mins) before scaling.
* **Profiling Tools**
  + Use cProfile or line-profiler to detect slow spots in transcription/summarization.
* **Replay Testing**
  + Keep a library of test meeting audios (short/medium/long, multiple accents) to repeatedly test after updates.

### 5. ****User Acceptance Testing (UAT)****

* End-users test with their own meeting recordings.
* Validate readability of summaries and correctness of extracted action items.
* Feedback loop to fine-tune summarization length and action item formatting.

**Bibliography:**

1. Gradio Documentation:
   1. Gradio – A Python library for building web apps.
   2. <https://www.gradio.app/docs>
2. **Whisper Documentation:**
   1. A Speech-to-text model from OpenAI.
   2. <https://platform.openai.com/docs/models>
   3. <https://huggingface.co/docs/transformers/en/model_doc/whisper>
3. **Hugging Face Transformers:**
   1. Pre-trained models and libraries used for text summarization and NLP tasks.
   2. <https://huggingface.co/docs/transformers/main/en/index>
4. **FFMPEG Documentation:**
   1. FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created
   2. <https://ffmpeg.org/documentation.html>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_THE END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**