



AI-POWERED MEETING NOTES REWRITER

Project Name: TalkToText Pro

Category: Generative AI Odyssey



TABLE OF CONTENTS

PROBLEM DEFINITION.....	3
BACKGROUND.....	3
SCOPE OF PROJECT	4
FUNCTIONAL REQUIRMENTS	5
NON-FUNCTIONAL REQUIRMENTS	7
PROPOSED SOLUTION.....	8
DESIGN SPECIFICATION.....	10
DETAIL STEPS TO EXECUTE THE PROJECT	14
<u>PROJECT INSTALLATION GUIDE</u>	15
TEST DATA USED IN THE PROJECT	20
DIAGRAMS	23
GITHUB LINK.....	26
PUBLISHED BLOG:	27
TASK ALLOCATION	28
SCREENSHOTS	29
PROJECT DELIVERABLES.....	35
HARDWARE REQUIREMENTS	38
Hardware:	38
SOFTWARE REQUIREMENTS	38
Software:.....	38



PROBLEM DEFINITION

In today's remote-first world, virtual meetings are crucial for collaboration.

However, manually taking notes during discussions is tedious and error-prone.

Users often miss important points while trying to write everything down. There is no automated, real-time system that accurately transcribes, summarizes, and structures meeting conversations for future reference.

TalkToText Pro aims to solve this by providing an AI-powered system that converts meeting speech into clean, actionable, and structured notes.

BACKGROUND

With the rise of remote work, online learning, and virtual collaboration, meetings have become increasingly common. Platforms such as Microsoft Teams, Google Meet, and Zoom now play an integral role in professional and academic environments. Despite this shift, the process of documenting meeting discussions, action points, and decisions remains largely manual and error prone.



Participants often struggle to take accurate notes while actively engaging in the conversation, leading to missed information, inconsistencies, and a lack of reliable documentation. As a result, organizations and teams are increasingly looking for automated solutions that can transcribe and summarize meeting content efficiently and accurately.

To address this requirement, a solution must be created that can convert speech into meeting notes. With the emergence of AI tools and technologies, utilizing them to build this solution is appropriate.

SCOPE OF PROJECT

TalkToText Pro will:

- Accept meeting audio from various sources (Teams, Zoom, Meet)
- Support multiple languages and audio formats (.mp3, .wav, .mp4)
- Convert speech to text
- Translate non-English content to English
- Clean and optimize transcripts
- Summarize into clear, structured meeting notes
- Store history of meetings for later retrieval
- Export notes as PDF/Word and allow sharing via email

This tool will help professionals, students, and teams maintain organized meeting records without manual effort.



FUNCTIONAL REQUIRMENTS

The project aims to build an AI-powered meeting notes re writer using Generative AI, LLMs, and OpenAI technologies. It will convert raw meeting audio into clean, structured notes highlighting key decisions, action items, and takeaways.

KEY FUNCTIONAL REQUIREMENTS:

Audio Input UI – Upload meeting audio files (.mp3, .wav, .mp4) or provide links from Microsoft Teams or Google Meet.

Language Selection – Choose the original language of the audio.

Speech-to-Text – Transcribe audio using AI-based speech recognition (e.g. Whisper).

Language Translation – Automatically translate non-English transcripts to English.



Text Optimization – Clean and optimize text (remove fillers, duplicates, adjust tokens).

Meeting Notes Generation – Use OpenAI API to create structured notes (summary, key points, decisions, actions, sentiment).

Display Results – Show the generated notes in a clear, readable format on the UI.

Download & Export – Export notes as PDF or Word and share via email.

Data Storage – Store all transcript versions and final notes in a database.

View History – Let users view and retrieve past meeting notes.

Progress Tracker – Show real-time status of each processing stage.

User Login – Secure login system to manage meeting records and history.



NON-FUNCTIONAL REQUIRMENTS

There are several non-functional requirements that should be fulfilled by the application. These are listed as follows:

- 1. Performance:** The application must process a 30-minute meeting recording and generate notes within one to two minutes.
- 2. Secure:** Meeting data and summaries must be securely stored with access restrictions.
- 3. Accuracy:** Transcription and summarization must achieve at least 85–90% accuracy under good audio conditions.
- 4. Scalable:** The application should be scalable to handle multiple meeting uploads simultaneously.
- 5. Usable:** The UI must be simple, intuitive, and mobile-responsive for both technical and non-technical users.



PROPOSED SOLUTION

TalkToText Pro will deliver an end-to-end AI-powered system that transforms raw meeting audio into clean, structured, and actionable notes. The solution integrates several advanced AI components to ensure accurate transcription, multilingual support, and intelligent summarization.

WORKFLOW OVERVIEW:

Audio Input:

Users can upload recorded meeting audio files (e.g., .mp3, .wav, .mp4) or provide links to online meetings from platforms like Microsoft Teams, Google Meet, or Zoom. This ensures flexibility in handling different sources and formats.

AI-Powered Transcription:

The audio input will be processed through speech recognition tools such as Whisper or Google Speech-to-Text to convert spoken content into accurate text transcripts, even in multi-speaker or noisy environments.



Language Translation:

If the audio is not in English, the transcript will be automatically translated using Google Translate API or OpenAI ChatGPT translation models, enabling multilingual collaboration.

Text Cleaning & Token Optimization:

The raw or translated text will be cleaned to remove filler words, repetitions, and noise. Token optimization ensures that the content fits within OpenAI API limits for efficient summarization processing.

AI Summarization & Notes Generation:

The optimized text will be processed using OpenAI ChatGPT to generate structured meeting notes, including an executive summary, key discussion points, decisions taken, action items, and optional sentiment analysis.

Secure Data Storage:

All versions (raw transcripts, translated text, cleaned text, and final notes) will be securely stored in a MongoDB database for easy retrieval and future reference.



Exportable Output:

The final structured notes will be displayed in a clean UI and can be downloaded in PDF or Word format. Users will also be able to share notes directly via email or integrate them into productivity tools.

DESIGN SPECIFICATION

Frontend

Framework & Tools:

- Built using Next.js (App Router, v14+) with TypeScript.
- Tailwind CSS for styling with a custom neon dark theme (primary neon yellow #FFD34D, base black #0b0b0b).
- Framer Motion for smooth animations, transitions, and micro-interactions.
- HeadlessUI / shadcn for accessible dialogs, dropdowns, and components.
- React Query (or SWR) for API requests and client-side caching



UI & UX:

- Dark-first design with toggleable light mode.
- Subtle animated background (gradient/noise/particles) with option to toggle animations.
- Responsive and mobile-friendly layout with collapsible sidebar navigation.
- Components include: Upload Form (drag & drop), Progress Tracker, Notes Viewer, History List, Auth Forms, Theme Toggle.

Accessibility:

- Screen reader support, ARIA labels, aria-live for status updates.
- Full keyboard navigation support

Backend:

Framework & Language:

- Developed using Flask (Python) with Celery for background task management.

Core Integrations:

- Google Generative AI (Gemini) for language understanding and question answering.



- Google Gen-AI Embeddings (embedding-001) for document embeddings.
- FAISS for similarity search and vector database operations.
- LangChain to handle:
 - . Splitting large documents into chunks.
 - . Managing embeddings.
 - . Chaining LLM queries for efficient and context-aware responses.
- API Endpoints (consumed by Next.js frontend):
 - . /auth/register, /auth/login for user authentication.
 - . /api/upload for uploading PDF/meeting links.
 - . /api/status/{upload_id} for real-time progress tracking.
 - . /api/notes/{note_id} to fetch processed meeting notes.
 - . /api/download/pdf/{note_id}, /api/download/docx/{note_id} for exports.
 - . /api/history to retrieve user's past queries and results.

Model:

- Primary LLM: Groq latest.
- Embeddings: Google Gen-AI embedding-001.



- Flow: Uploaded PDFs → Chunking via LangChain → Embeddings stored in FAISS → Groq QA powered by relevant chunks.

Database:

- FAISS used as the primary vector store (local disk-based or persisted snapshot).
- Stores embeddings for semantic similarity search.
- Metadata & user history stored in persistent DB (Postgres or equivalent, depending on deployment).

Key Features:

- PDF(s) file upload with drag & drop support.
- Background processing of large files with real-time polling.
- Automatic chunking of large documents for scalability.
- Conversational AI: Users ask questions → System retrieves context from embeddings → Gemini answers.
- Download options: Export processed notes in PDF/DOCX.
- Responsive UI with futuristic neon black/yellow theme.



QnA History Saving:

- Persistent Storage: Stores all user queries, document notes, and AI responses.
- Easy Retrieval: Dedicated History page with filters (date, language, status).
- User-Friendly Interface: Organized browsing with paginated note cards, search, and bulk export options.

DETAIL STEPS TO EXECUTE THE PROJECT

Backend Requirements:

- Install Python 3.10+
- Install Flask 3.0+
- Install pymongo (MongoDB connector).
- Install langchain (for chunking + processing).
- Install faiss-cpu (for vector similarity search, optional if hybrid search needed).
- Install Gork SDK / API client (whichever library Gork provides).
- Install additional libraries listed in requirements.txt.



Frontend Requirements:

- Install Node.js 18+ and pnpm / npm / yarn.
- Framework: Next.js (14+) with App Router.
- Styling: Tailwind CSS.
- Animations: Framer Motion.
- UI Components: shadcn/headlessui.
- Data fetching: React Query or SWR.

PROJECT INSTALLATION INSTRUCTIONS

BACKEND SETUP

1. Clone backend repository:

```
git clone <your-backend-repo-url>
```

```
cd backend.
```



2. Create and activate virtual environment:

```
python -m venv venv
```

```
source venv/bin/activate    # Linux/Mac
```

```
venv\Scripts\activate      # Windows
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Create a .env file:

```
GORK_API_KEY=<your_gork_api_key>
```

```
MONGO_URI=mongodb://localhost:27017
```

```
DB_NAME=your_database_name
```

5. Run backend server:

```
python wsgi.py
```

FRONTEND SETUP

1. Clone frontend repository:

```
git clone <your-frontend-repo-url>
```

```
cd frontend
```




2. Install dependencies:

npm install

or

pnpm install

3. Create .env.local:

NEXT_PUBLIC_API_BASE_URL=http://localhost:5000

NEXT_PUBLIC_APP_NAME=TalkToText Pro

4. Run Next.js server:

npm run dev

5. Open frontend at:

<http://localhost:3000>

WORKFLOW

- Open the app in browser.
- Upload PDF file.
- Backend will:
 1. Extract text → split into chunks (LangChain).



2. Store documents + embeddings in MongoDB.
3. Send query + context to Gork model for QnA.
4. Save QnA history in MongoDB collection.

- Frontend will:

1. Show real-time loader.
2. Display answers + allow revisiting past history.
3. Support download in PDF/DOCX.

DEPLOYMENT:

- Frontend (Next.js)
- Deploy to Vercel or Netlify.
- Add env variables:

NEXT_PUBLIC_API_BASE_URL=<your_backend_url>

NEXT_PUBLIC_APP_NAME=TalkToText Pro

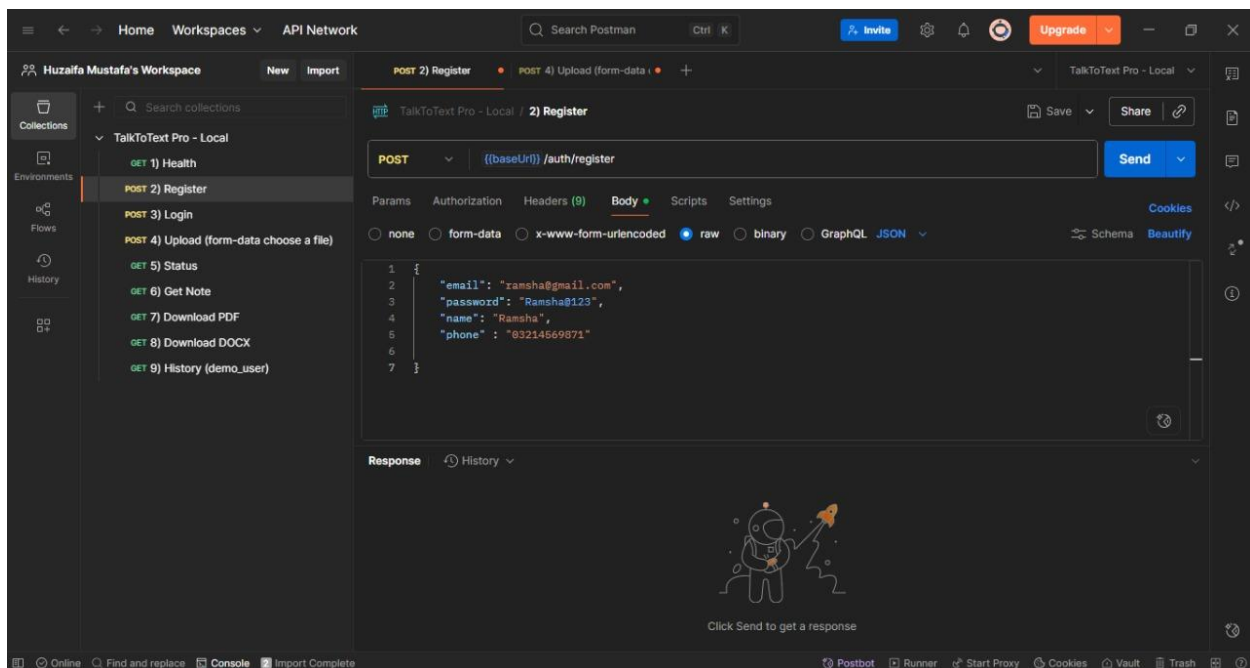


- Backend (Flask + MongoDB)
 - Deploy to Render / AWS / GCP / Railway / Heroku.
 - Use MongoDB Atlas (cloud database).
-
- Update .env with:
GORK_API_KEY=<production_key>
MONGO_URI=<Atlas_connection_string>
DB_NAME=prod_db
-
- Use Railway / uWSGI + Nginx for production server.



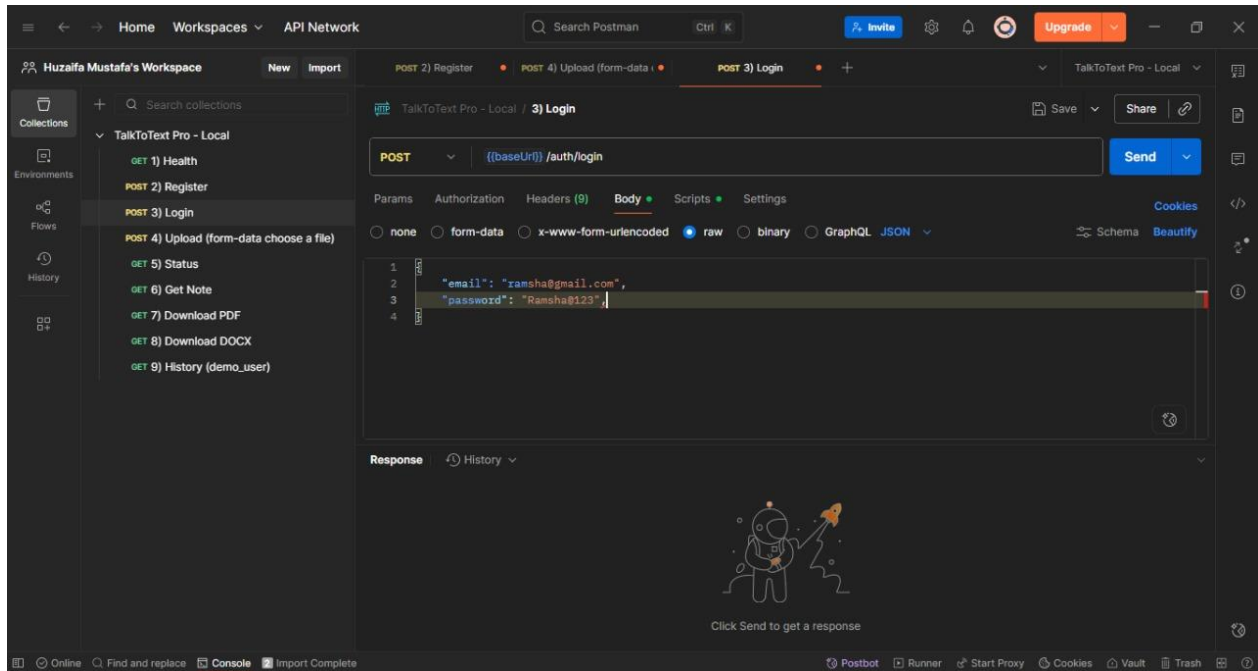
TEST DATA USED IN THE PROJECT

Backend Register Testing:



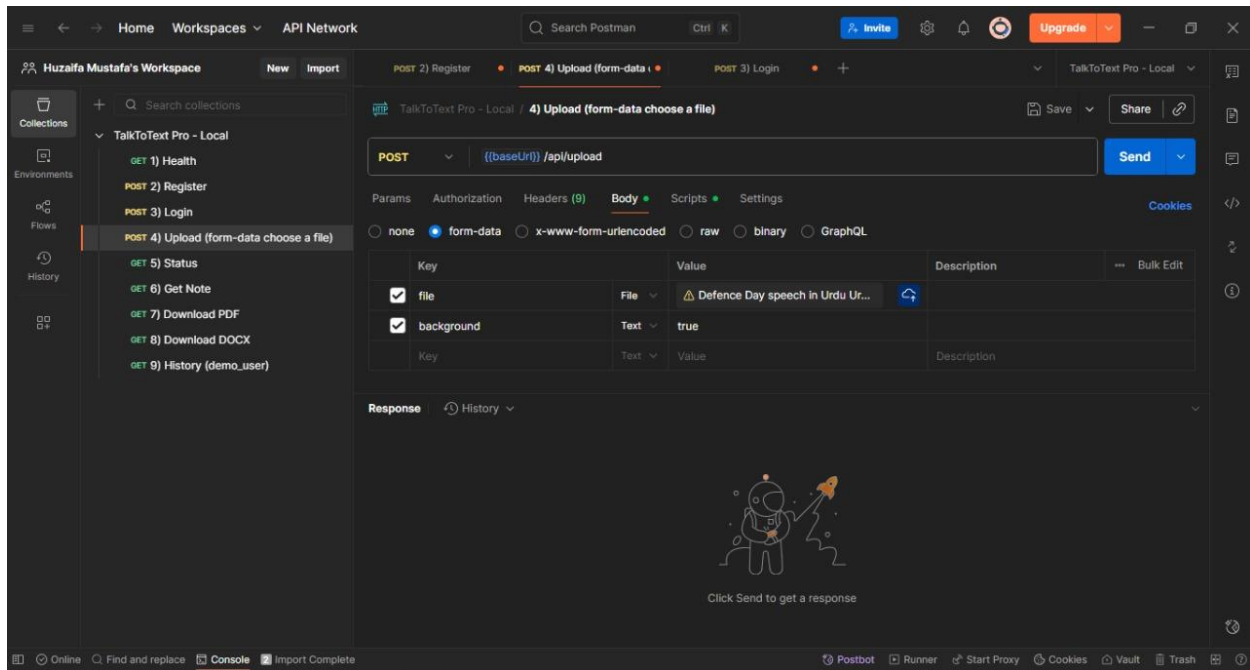


Login Testing:





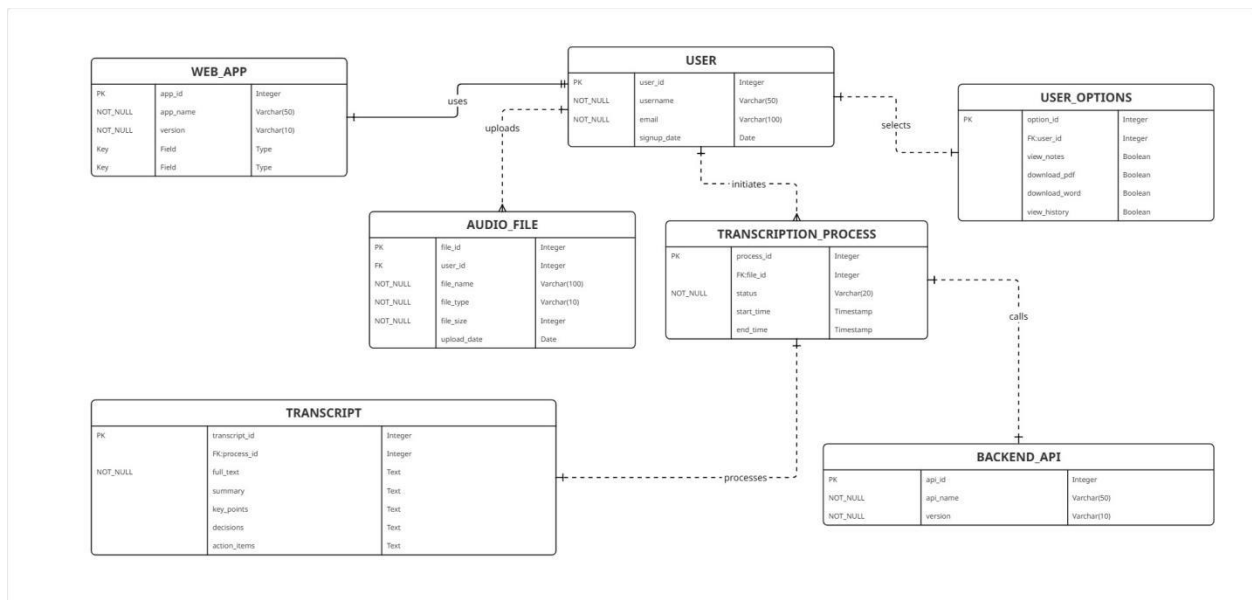
Upload Testing :





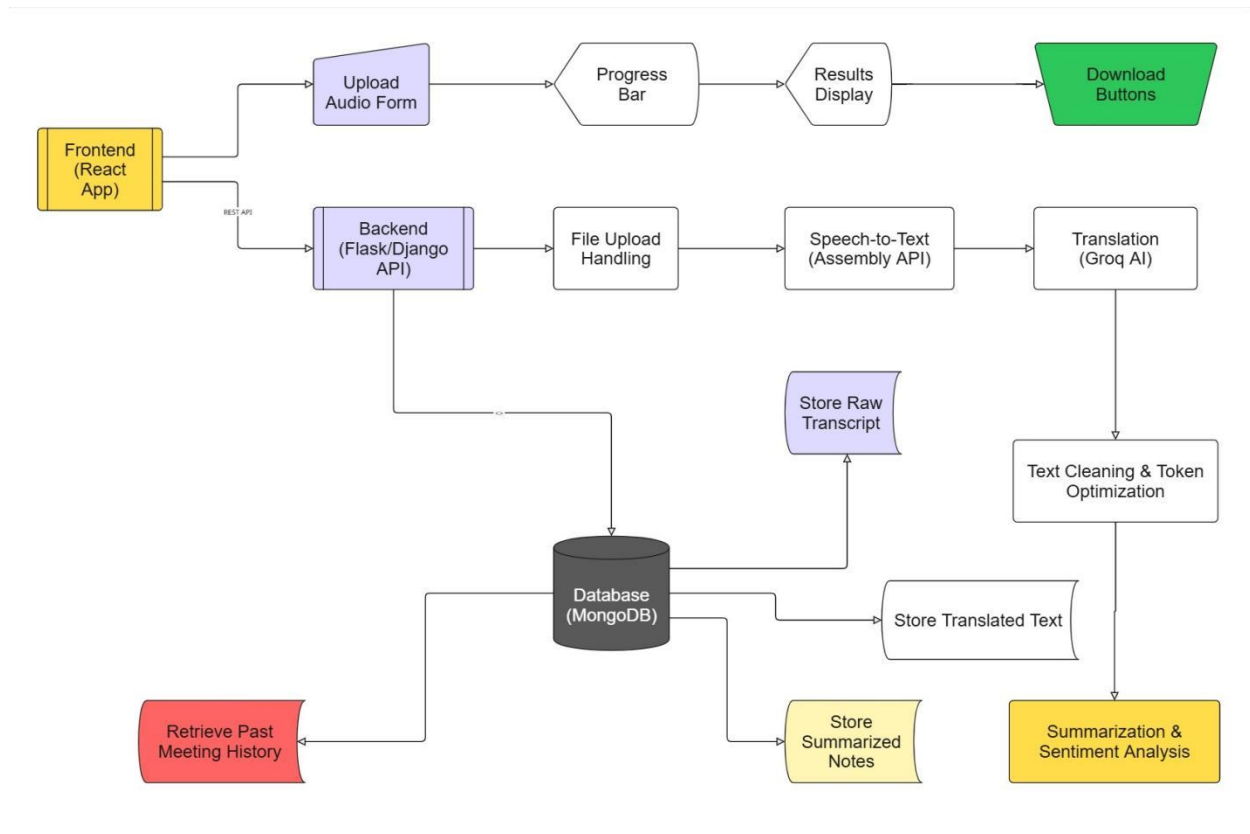
DIAGRAMS

User flow diagram for the project:



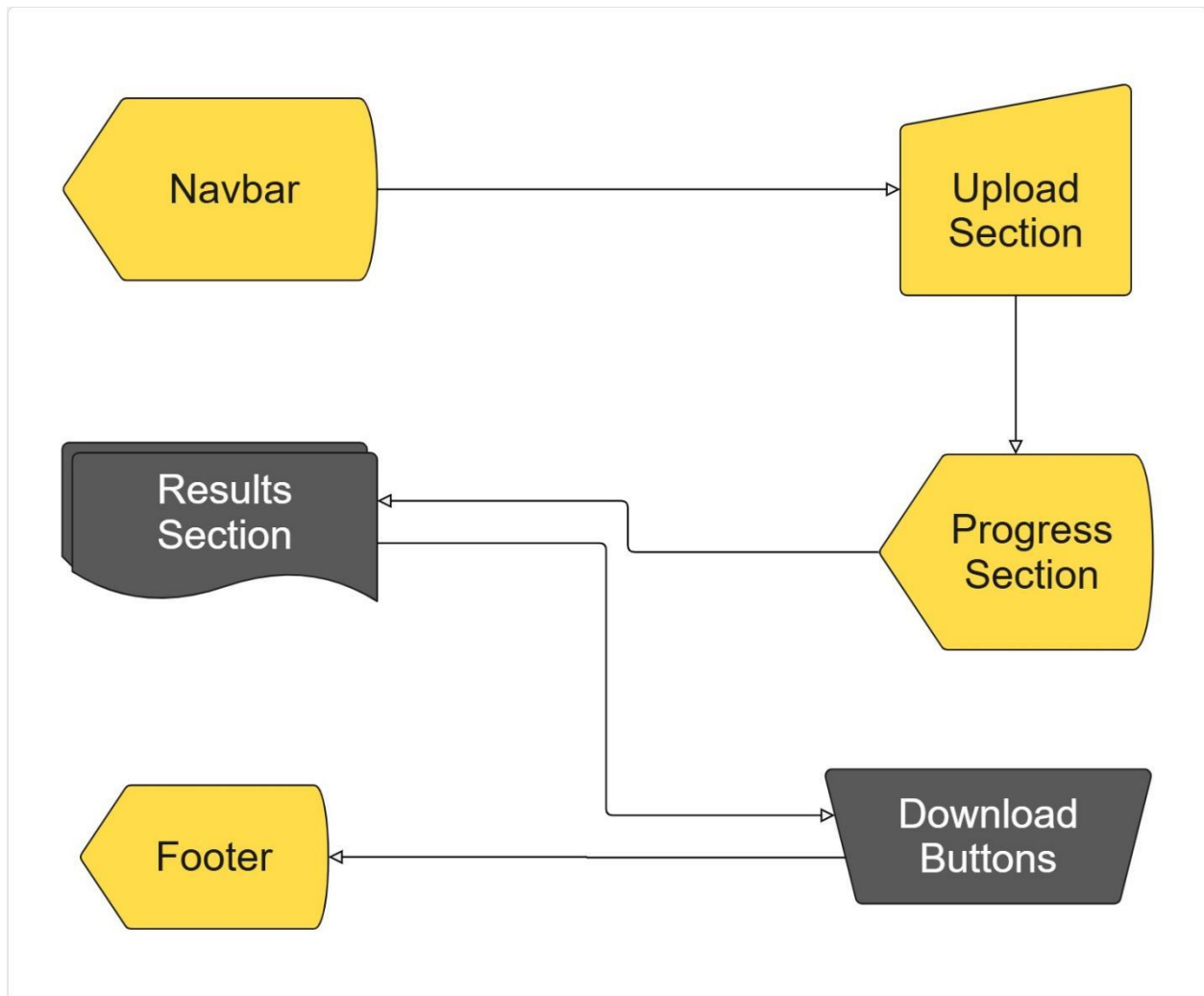


System Architect Diagram:





UI Wireframe:





User Journey Map:

- User opens the chatbot interface.
- Uploads PDF Files.
- Receives a confirmation and a progress indicator while the file is being processed.
- Asks a question related to the uploaded content.
- TalktoTextPro responds with a relevant answer based on document context.

GITHUB LINK:

You can access the complete source code on the GitHub repository:

<https://github.com/huzaifaair/talktotext-frontend>

<https://github.com/huzaifaair/talktotext-backend>



LINK TO PUBLISHED BLOG:

The project is also documented in a detailed blog post on a free blogging platform. The blog covers the problem statement, LLMs, Gen-AI, Design choices and challenges faced during development.

<https://talktotextpro.wordpress.com>

LINK TO DEPLOYED APPLICATION:

This is the link of deployed application.

<https://talktotext-frontend-u1x6.vercel.app/>



TASK ALLOCATION

Project Title: GenMaster



Huzaifa Mustafa

Project Task

BACKEND ON PYTHON



Amaan

Project Task

INTEGRATION



Ramsha Anwar

Project Task

FRONTEND &
DOCUMENTATION




SCREENSHOTS

Step 1: Open the application and upload PDF File(s).

Upload Meeting Recording
Transform your meetings into actionable insights with AI-powered analysis

Upload Meeting Recording
Upload an audio/video file or provide a meeting URL for AI-powered transcription and analysis

Upload File **Meeting URL**



Drop your file here
Supports audio and video files (MP3, MP4, WAV, etc.)

Choose File

Language **Extract Duration (seconds)**
Auto-detect 120

☐ **Process in background (recommended for large files)**

Start Processing



Step 2: Once file uploaded you can ask for meeting notes.

Upload Meeting Recording


Transform your meetings into actionable insights with AI-powered analysis

Upload Meeting Recording

Upload an audio/video file or provide a meeting URL for AI-powered transcription and analysis

Upload File

Meeting URL

 Summarizelt — AI Document Assistant.mp4
7.19 MB

Language
Auto-detect ▼

Extract Duration (seconds)
120

☒ Process in background (recommended for large files)

Start Processing



Step 3: If the answer is available in context, it will give you this response.

Processing Your Meeting

Please wait while we transcribe and analyze your recording

Processing Your Meeting

Upload ID: 4f245d08-8251-4fa5-ab87-422fd671034c summarizing

Summarizing 85%

✓ Uploaded

🕒 Extracting Audio

🕒 Transcribing

🕒 Translating

🕒 Optimizing

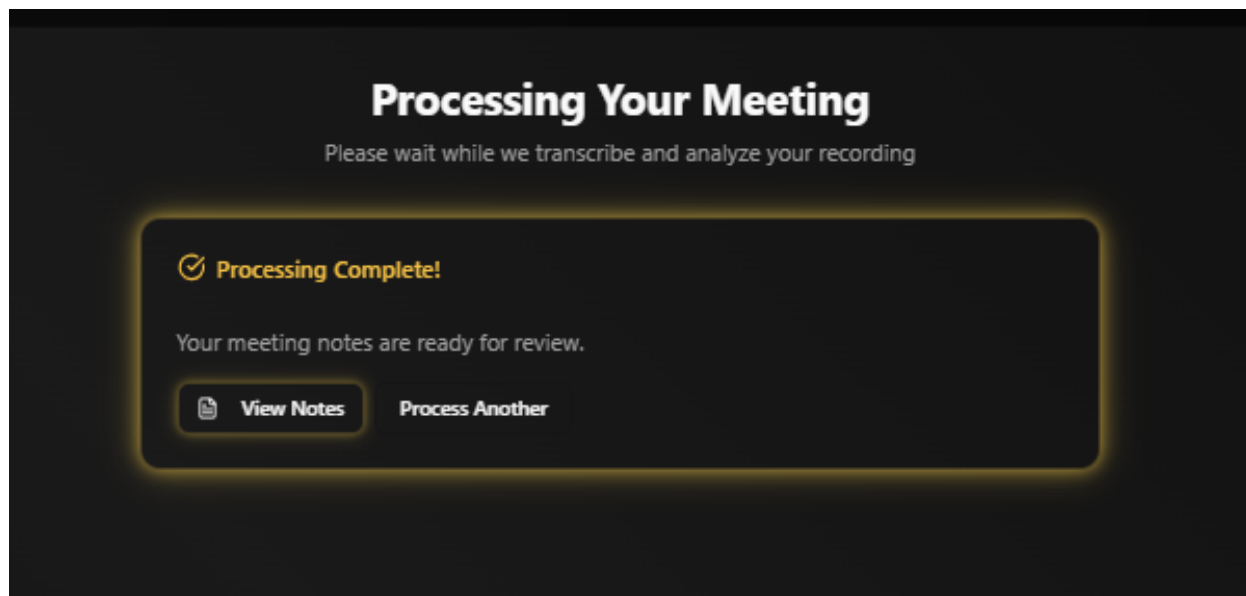
🕒 Summarizing

✓ Complete

🕒 Estimated time remaining: 2-5 minutes



Step 4: If your answer is not available in context, here is your response.





Meeting Notes

9/14/2025

Download PDF Download DOCX Copy Text Share

Final Notes

Abstract Summary

Summary: Meeting discussion centered around addressing bathroom renovation and plumbing issues.

Objective: To explore reliable options for bathroom remodeling and maintenance.

Key takeaway: Shineview Technical Services offered a solution to these problems.

Key Points

Addressing bathroom renovation and plumbing issues

Leaking toilets and old fittings can be replaced and renovated efficiently

No mess, no delays, and quality work that lasts is guaranteed

Action Items

Client - Research Shineview Technical Services for bathroom renovation options

By: Next meeting

Shineview Technical Services - Provide a quote for bathroom renovation services

By: End of week

Sentiment

The meeting tone was professional and solution-oriented, focusing on addressing immediate concerns and offering reliable solutions for bathroom renovation and maintenance.

Transcript

Cleaned Transcript

Is your bathroom giving you trouble? Leaking toilet, old fittings, mess everywhere. Don't worry. We fix plumbing problems, replace old WCs, and renovate bathrooms the right way. Our team is fast, clean and professional. No mess, no delays, just quality work that lasts. Our goal? To give you a bathroom that not only works better, but feels better every single day. So what are you waiting for? Call Shineview Technical Services today and get the job done right now.




Step 5: You can view/manage chat history.

Meeting History

Note – 9/14/2025

📅 9/14/2025, 7:57:46 PM

Abstract Summary - Summary: Meeting discussion centered around addressing bathroom renovation and plumbing issues. - ...

 [View](#)



PROJECT DELIVERABLES

We are required to design, develop, and document the project. The final submission must include both the working source code and a comprehensive project report.

Project Report Must Include:

Problem Definition

Clearly state the problem that the project solves and why it is important.

Design Specifications

Mention details of the frontend (Next.js + Tailwind), backend (Flask + Gork), and database (MongoDB).

Diagrams

Include diagrams such as:

User Flow Diagram

System Architecture Diagram

User Journey Map



Detailed Execution Steps

Explain step-by-step how to set up the project environment and run it locally.

Test Data

Provide details of the sample PDF files and queries used to test the project.

Project Installation Instructions

Write installation instructions for both frontend and backend with environment variables and deployment notes.

GitHub Repository Link

Provide the public GitHub repository URL where the project code is uploaded.

The repository must include all source code files.

Add .ipynb files for Jupyter Notebook/Colab if you demonstrate the backend pipeline (e.g., embeddings, Gork model test).

Grant public access so reviewers can test.



Published Blog Link

Write and publish a blog post (minimum 2000 words) on a free blogging platform (Blogger, Ghost, Medium, Tumblr, etc.)

The blog should cover:

Project Idea and Motivation

Technology Stack (Next.js, Flask, Gork, MongoDB, FAISS, LangChain)

Step-by-Step Implementation Guide

Screenshots / Diagrams

Challenges and Solutions

Final Outcomes and Future Scope

Submit the blog link along with project documentation.

ReadMe File

Add a README.md file in the GitHub repository that includes:

Overview of the project

Tech stack used

Setup instructions

Usage instructions

Assumptions (if any)



HARDWARE REQUIREMENTS

Hardware:

Intel Core i5/i7 Processor or higher

8 GB RAM or higher

Color SVGA

500 GB Hard Disk space

Keyboard and Mouse

SOFTWARE REQUIREMENTS

Software:

Technologies to be used:

1. **Frontend:** Next.js and Tailwind
2. **Backend:** Flask
3. **Data Store:** Word / PDF
4. **Programming/IDE:** VS Code
5. **Libraries:** AI API, Python libraries, and pre-trained transformers