****

***Aptech Shahrah-e-Faisal Center***

***TEAM MEMBERS:***

|  |  |  |
| --- | --- | --- |
| ***NAME:*** | ***STUDENT’S ID:*** | ***PROJECT RESPONSIBILITIE’S:*** |
| ***Muslim*** | ***1334422*** | ***Backend*** |
| ***Hira*** | ***1545872*** | ***Backend*** |
| ***Aalock*** | ***1448863*** | ***Frontend*** |
| ***Farhan*** | ***1444343*** | ***Frontend*** |
| ***Umer*** | ***1448863*** | ***Documentation*** |

***PROJECT GENERATIVE AI: FACULTY:***

***TalktoText MISS AIMAN***

***Executive Summary***

This document outlines the complete project plan for **TalkToText Pro**, an advanced, AI-driven application designed to streamline the process of meeting documentation. By leveraging a modern tech stack centered on Django for the backend and React for the frontend, TalkToText Pro will provide a seamless workflow that transforms raw audio into clean, structured, and insightful notes. The system automates transcription, translation, and intelligent summarization, delivering actionable information such as executive summaries, key discussion points, This solution aims to significantly reduce the time and effort spent on manual note-taking, improving productivity and information retention for individuals and teams.

# 

***Problem Definition:***

In today's fast-paced professional environment, meetings are an essential component of collaboration and decision-making. However, the process of documenting these meetings is often inefficient and prone to errors. Key challenges include:

* **Information Loss:** It is difficult for a human to simultaneously participate in a discussion and accurately capture every detail, leading to missed points and decisions.
* **Time Consumption:** The time spent manually transcribing and summarizing meeting recordings after the fact is significant and detracts from core responsibilities.
* **Lack of Structure:** Manual notes are often inconsistent, unstructured, and difficult to search or share, making it challenging to retrieve specific information later.
* **Language Barriers:** Teams working across different geographical locations may face challenges when meeting in a language that not all members are proficient in.

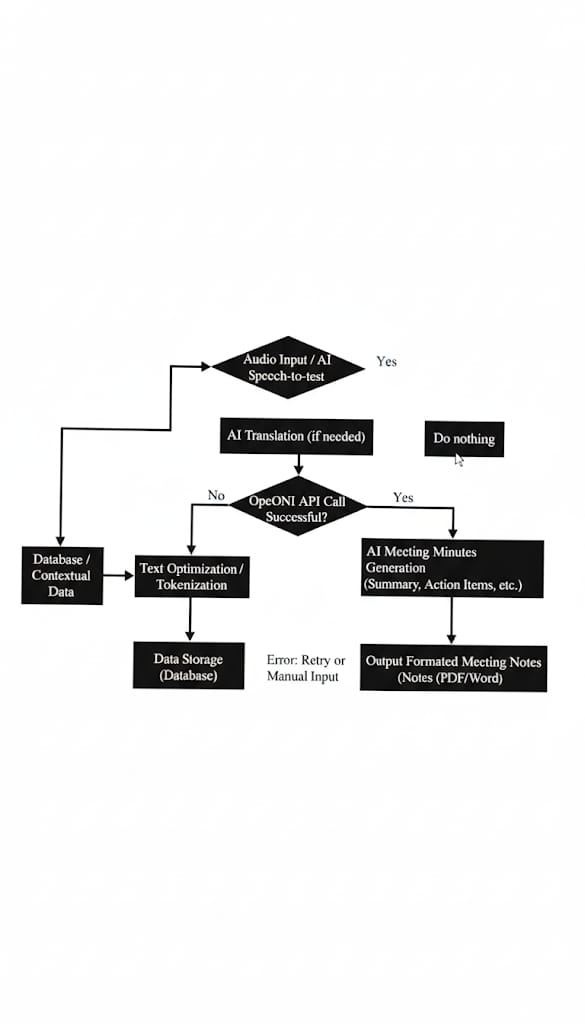
These problems collectively hinder organizational efficiency, slow down project execution, and lead to communication gaps within teams.

***Proposed Solution - TalkToText Pro***

TalkToText Pro addresses these challenges by offering an end-to-end automated solution. The application acts as an intelligent assistant, taking meeting audio and processing it through a series of sophisticated AI models to produce a comprehensive set of final notes. The key components of our solution are:

* **Automated Transcription:** Utilizing state-of-the-art AI models like OpenAI’s Whisper, the system accurately converts speech into text.
* **Seamless Translation:** For multi-lingual environments, the system automatically detects the language and translates the transcript into English, providing a standardized output.
* **Intelligent Summarization:** Using powerful Large Language Models (LLMs) from OpenAI, the application distills long transcripts into a concise executive summary, critical discussion points, clear action items, and documented decisions.
* **Structured Output:** The final notes are presented in a clean, structured format, making them easy to read, understand, and share.
* **Secure and Accessible History:** All processed notes and transcripts are securely stored in a database, allowing users to access and search their meeting history at any time.

By automating this entire pipeline, TalkToText Pro saves valuable time, improves the accuracy of meeting records, and ensures that key information is never lost.

***Diagram:***

***TalkToText Pro: Installation and Execution Guide***

This document provides a comprehensive guide to setting up and running the TalkToText Pro project on your local machine. Follow these instructions carefully to get the application up and running.

***1.Prerequisites***

Before you begin, ensure the following software is installed on your system:

* **Python 3.13 :** For the Django backend.
* **Node.js and npm:** For the React frontend.
* **MongoDB:** A local installation or a cloud-based service like MongoDB Atlas.
* **Git:** For cloning the project repository.
* **OpenAI API Key:** API key from OpenAI for the transcription, translation, and summarization features.

***2. Project Setup***

Start by cloning the project repository from GitHub.

https://github.com/farhanshaikh00/Talk-To-Text

The project directory should contain two main folders: backend and frontend.

***3. Backend Setup (Django)***

The backend is responsible for all the core logic, including user authentication, file processing, and AI model integration.

1. ***Navigate to the backend directory:***

cd backend

1. ***Create a Python virtual environment:***

This isolates the project's dependencies from your system-wide Python environment

python -m venv .venv

1. ***Activate the virtual environment:***

On Windows:

venv\Scripts\activate

On macOS/Linux:

source venv/bin/activate

1. *** Install backend dependencies:***

The required packages are listed in the requirements.txt file.

pip install -r requirements.txt

1.  ***Configure environment variables:***

Create a .env file in the backend directory and add your API keys and database connection string. Replace the placeholder values with your actual keys and URLs.

.env file

DJANGO\_SECRET\_KEY=your\_django\_secret\_key

OPENAI\_API\_KEY=your\_openai\_api\_key

MONGODB\_URI=mongodb://localhost:27017/talk\_to\_text\_pro

1. *** Run database migrations:***

This sets up the initial database schema.

python manage.py migrate

1. ***Start the Django development server:***

python manage.py runserver

***4. Frontend Setup (React)***

The frontend is the user interface of the application.

***1. Navigate to the frontend directory in a new terminal window:***

cd frontend

***2. Install frontend dependencies:***

npm install

***3. Configure API endpoint:***

Create a .env file in the frontend directory. This tells the React app where to find the Django backend.

.env file

REACT\_APP\_API\_URL=http://localhost:8000/api

***4. Start the React development server:***

npm run dev

***Functional Requirements***

This section details the core functionalities of the TalkToText Pro application.

***1. Audio Input***

* The system must accept meeting audio files in common formats, including .mp3, and .mp4.
* Users must be able to provide a link to an online meeting recording from platforms such as Zoom, Microsoft Teams, or Google Meet.
* The system should support uploading audio files from a user's local device.

***2. Speech-to-Text Transcription***

* The application will utilize an AI model, such as OpenAI's Whisper or Google Speech-to-Text, to convert spoken words into a raw text transcript.
* The transcription process must be accurate and capable of handling various accents and speech patterns.
* The system should provide a progress indicator during the transcription phase.

***3. Multi-Language Translation***

* The system must automatically detect the language of the uploaded audio.
* If the language is not English, the system must automatically translate the full transcript into English.
* The translated text should be stored separately from the original transcript.

***4. Text Cleaning and Optimization for LLMs***

* The system must perform a pre-processing step on the transcribed text.
* This includes removing filler words (e.g., "um," "uh"), repeated phrases, and long pauses.
* The processed text must be optimized to fit within the token limits of the LLM API (e.g., OpenAI's GPT models).
* The optimization should be intelligent, preserving the core meaning and context of the conversation.

***5. AI-based Structured Notes Generation***

* The application will send the optimized text to the OpenAI GPT API.
* The LLM will generate a structured output containing the following key sections:
  + **Short Executive Summary:** A concise, high-level overview of the entire meeting.
  + **Key Discussion Points:** A bulleted list of the main topics and ideas discussed.

***6. Secure User Authentication***

* The system must include a secure login system.
* Users can register with an email and password.
* All user data, including passwords, must be securely hashed and salted.
* The login system ensures that users can only access their own meeting history and data.

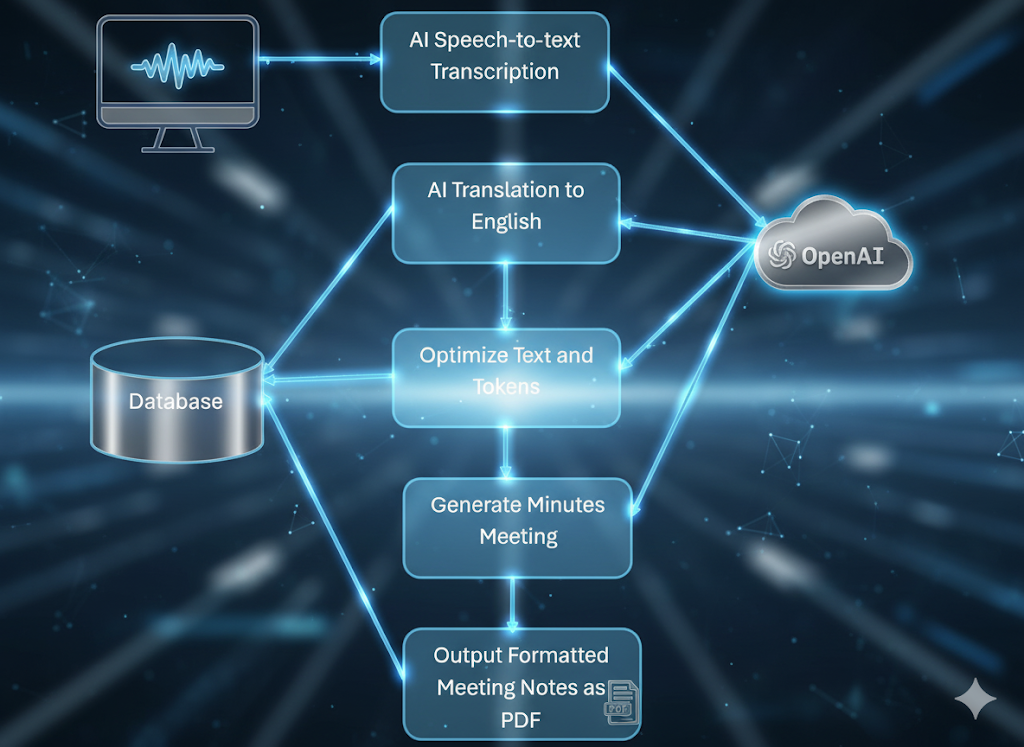
***7.Database History Retrieval***

* All stages of the processing (raw transcript, translated text, optimized text, and final notes) must be stored in a MongoDB database.
* Users must have a dashboard view where they can see a history of all their processed meetings.
* The system must provide a search and filter function to easily find past meetings.

1. ***Export and Sharing***

* Users must be able to export their final notes.
* Supported export formats include PDF documents.

***TalktoText Structure:***

***This structure we are following:***

***Non-Functional Requirements***

***1. Performance and Speed***

* **Goal:** A 30-minute audio file should be processed and converted into notes within 1-2 minutes.
* This will require a highly efficient and asynchronous backend architecture to handle the computationally intensive tasks of transcription and summarization.

***2. Accuracy***

* **Goal:** The transcription accuracy should be between 85-90% under good audio quality conditions.
* This accuracy is dependent on the underlying AI models (Whisper) and can be improved with additional fine-tuning.

***3. Security***

* All user data, including uploaded audio files, transcripts, and final notes, must be stored securely.
* Data will be encrypted both at rest (in the database) and in transit (via SSL/TLS).
* User passwords will be stored as one-way hashes to prevent unauthorized access.

***4. Scalability***

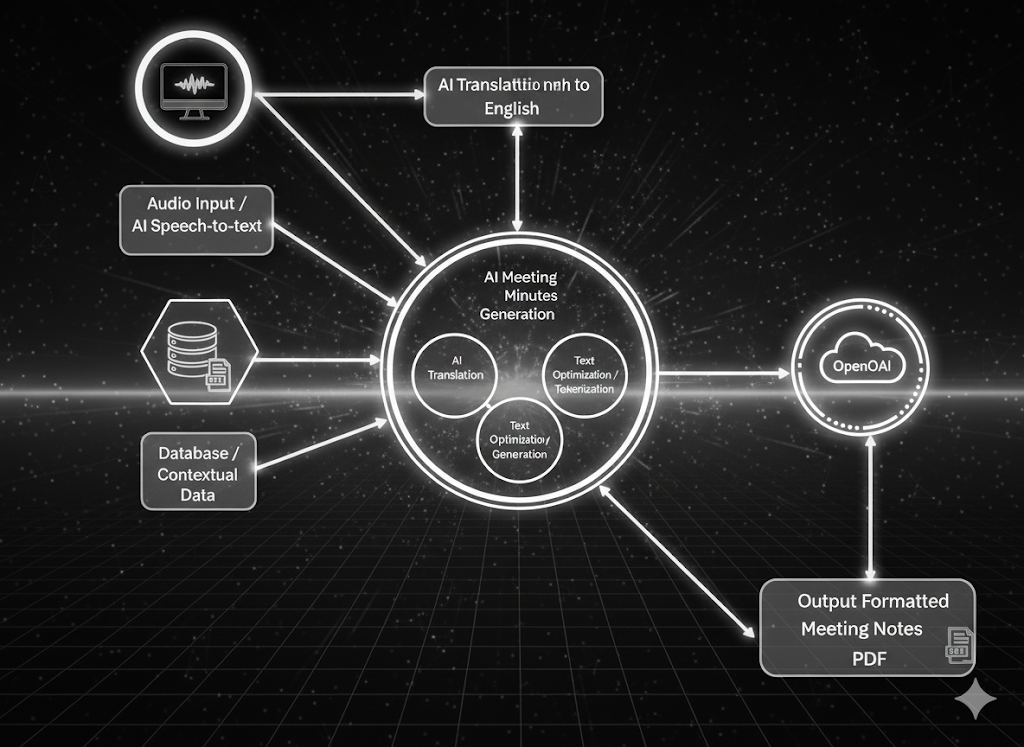
* The architecture must be scalable to handle multiple simultaneous uploads from different users.
* The backend will use an asynchronous task queue to manage parallel processing, preventing the system from becoming unresponsive under heavy load.

***5. Usability***

* The user interface must be simple, intuitive, and responsive.
* The UI should be optimized for different devices (desktops, tablets, and mobile phones).
* The workflow from upload to notes retrieval should be clear and require minimal user interaction.

***Architecture:***

***The Architecture we are following:***

******

***SOFTWARE:***

***Technologies can be used:***

* **Frontend:** HTML5 or any other frontend programming languages.
* **Backend:** Flask/Django.
* **Data store:** PDF and MongoDB.
* **Programming/IDE:** Python, Jupyter, Notebook, Anaconda,

Google Colab.

* **Libraries:** tensorflow, NLTK, Keras, OpenAI API, Python libraries and Transformers.

**THE END**

***THANKYOU***