



SPEECH TO SCRIPT (AUDISCRIBE)

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
AWARD OF THE DEGREE OF

B.TECH. (COMPUTER ENGINEERING)

TO

RK UNIVERSITY, RAJKOT

SUBMITTED BY

Name of Student

Ankur Datta

Enrollment No.

21SOECE11608

UNDER THE GUIDANCE OF

Internal Guide

Miss Shivangi Patel

Assistant Professor,

CE/IT Department (Ext No.-139)

RK University

Rajkot-Gujarat

External Guide

Mr. Neela Santhosh Kumar

Human Resources & Academic Head

CODTECH IT SOLUTIONS

Hyderabad, Telangana-500079

April 2025



SCHOOL OF ENGINEERING, RK UNIVERSITY, RAJKOT

DECLARATION

I hereby certify that I am the sole author of this project work and that neither any part of this project work nor the whole of the project work has been submitted for a degree to any other University or Institution. I certify that, to the best of my knowledge, my project work does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my project document, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I declare that this is a true copy of my project work, including any final revisions, as approved by my project review committee.

Signature of Student

Ankur Datta (21SOECE11608),

Date:_____

Place:_____

INTERNSHIP COMPLETION LETTER



CODTECH IT SOLUTIONS

8-7-7/2, Plot No: 51, Opp:Naveena School, Hasthinapuram Central,
Hyderabad , 500 079, Telangana

+91 9848925128

hr@codtechitsolutions.com

www.codtechitsolutions.com



ID:CT08JSJ

CERTIFICATE OF INTERNSHIP EXPERIENCE

This certificate is proudly presented to

ANKUR DATTA

Successfully completed the internship program at **CODTECH IT SOLUTIONS**
in **Artificial Intelligence** from **January 5th, 2025 to May 5th, 2025** with
unwavering dedication.

NEELA SANTHOSH KUMAR

**HUMAN RESOURCES &
ACADEMIC HEAD**



VERIFIED BY

CODTECH IT SOLUTIONS PVT.LTD





CERTIFICATE

This is to certify that the work which is being presented in the Project Report entitled **“SPEECH TO SCRIPT (AUDISCRIBE)”**, in partial fulfillment of the requirement for the award of the degree of **B.Tech. (Computer Engineering)** and submitted to the School of Engineering, RK University, is an authentic record of my own work carried out during a period from **December 2024 to April 2025**.

The matter presented in this Project Report has not been submitted by me for the award of any other degree elsewhere.

Signature of Student

Ankur Datta (21SOECE11608)

This is to certify that the above statement made by the student is correct to the best of my knowledge.

Internal Guide

Miss Shivangi Patel
Assistant Professor,
CE/IT Department
RK University, Rajkot

External Guide

Neela Santhosh Kumar
HR & Academic Head,
Codtech IT Solutions,
Hyderabad, Telangana

Head of Department

Dr. Chetan Shingadiya
CE/IT Department,
School of Engineering,
RK University, Rajkot

April 2025



SCHOOL OF ENGINEERING, RK UNIVERSITY, RAJKOT

ACKNOWLEDGEMENT

It has indeed been a great privilege for me to have **Mr. Neela Santhosh Kumar**, Project Coordinator of CODTECH IT SOLUTIONS PVT. LTD, Hyderabad, and **Miss Shivangi Patel**, Assistant Professor of RK University, as my mentors for this project. His awe-inspiring personality, superb guidance, and constant encouragement are the driving forces behind this project. I take this opportunity to express my outmost gratitude to him. I am also indebted to him for his timely and valuable advice. I would also like to extend my appreciation to my peers and family for their constant encouragement.

I am highly grateful to all the technical and non-teaching staff of the Development Department of CODTECH IT SOLUTIONS PVT. LTD and Computer Engineering Department of RK University for their constant assistance and cooperation.

ABSTRACT

This project report presents the development of a Speech Recognition System, **Audiscribe**, using Python. The system is designed to convert spoken language into text with high accuracy and efficiency. Leveraging the Google Speech Recognition API, it processes real-time audio input and provides accurate transcriptions, making it useful for applications such as voice-controlled systems, automated transcription services, and accessibility tools for individuals with disabilities.

The primary objective of this project is to build an intuitive and reliable speech-to-text conversion tool that can adapt to different accents and speech variations. The system is implemented using Python with key libraries such as `speech_recognition` and `pydub` for audio processing. The report explores the technical aspects of speech recognition, including signal processing, feature extraction, and challenges such as background noise interference and API dependency. Furthermore, the study discusses potential improvements, including integrating offline speech models and deep learning techniques for enhanced accuracy.

Through this project, we aim to contribute to the growing field of AI-driven voice recognition systems by developing a cost-effective and scalable solution that can be integrated into diverse applications. The report provides insights into system architecture, implementation details, and future enhancements, making it a valuable resource for researchers and developers interested in speech recognition technology.

TABLE OF CONTENT

TITLE	PAGE NO.
Company Profile	I
Company Infrastructure	II
1.0 Introduction	1
1.1 Project Summary	1
1.2 Purpose	2
1.3 Scope	2
1.4 Technology and Literature Review	3
2.0 Project Management	4
2.1 Project Planning and Scheduling	4
2.1.1 Project Development Approach	4
2.1.2 Project Plan	4
2.2 Risk Management	5
2.2.1 Risk Identification	5
2.2.2 Risk Analysis	5
2.2.3 Risk Planning	6
3.0 System Requirement Study	7
3.1 User Characteristics	7
3.2 Hardware and Software Requirement	8
3.3 Constraints	9

TABLE OF CONTENT

TITLE	PAGE NO.
4.0 System Analysis	10
4.1 Study of Current System	10
4.2 Problem and Weakness of the current System	10
4.3 Requirement of New System	11
4.4 Feasibility Study	12
4.5 Requirements Validation	12
4.6 Functions of System	13
4.6.1 Use Cases	13
4.7 Data Modeling	14
4.7.1 Data Dictionary	14
4.7.2 E-R diagrams	14
4.7.3 Class Diagram	15
4.8 Function and Behavioral Modeling	16
4.8.1 Data Flow Diagram	16
4.9 Main Modules of New System	16
4.10 Selection of Hardware and Software and Justification	17
5.0 System Design	18
5.1 Database Design/Data Structure Design	18
5.1.1 Mapping objects/classes to tables	18
5.1.2 Tables and Relationship	18
5.1.3 Logical Description Of Data	19
5.2 System Procedural Design	19
5.2.1 Flow chart or activity design	19

TABLE OF CONTENT

TITLE	PAGE NO.
5.3 Input/Output and Interface Design	20
5.3.1 Samples of Forms and Interface	20
5.3.2 Access Control and Security	20
5.4 System Architecture Design	20
6.0 Implementation Planning and details	21
6.1 Implementation Environment	21
6.2 Program/Modules Specification	21
6.3 Security Features	22
6.4 Coding Standards	23
6.5 Sample Coding	24
7.0 Testing	25
7.1 Testing Plan	25
7.2 Testing Strategy	26
7.3 Testing Methods	26
7.4 Test Cases	27
8.0 Screen shots and User manual	28
9.0 Limitation and Future Enhancement	32
10.0 Conclusion and Discussion	34
11.0 References	36

COMPANY PROFILE

Name of Organization:	CODTECH IT SOLUTIONS PVT. LTD
Managing Director:	Mr. Neela Akhil Kumar
HR/Contact Person Name:	Mr. Neela Santhosh Kumar
Phone No.:	Mobile No.: +91 9848925128
Email Address:	hr@codtechitsolutions.com
Industrial supervisor's name:	Mr. Neela Santhosh Kumar
Designation:	Human Resources & Academic Head
Phone No.: -	Mobile No.: +91 9848925128
Email Address:	hr@codtechitsolutions.com
Year of Establishment:	2023
Address:	8-7-7/2, Plot No: 21, Opp: Naveena School, Hashtinapuram Central, Hyderabad, 500079, Telangana
Product Details:	Comprehensive Tech Support, Data Analysis and Optimization, Customized Application Development
Details of testing facilities available: -	Application Testing
Any other special information:	

COMPANY INFRASTRUCTURE

COMPANY INTRODUCTION

CODTECH IT Solutions Private Limited is a Hyderabad-based IT services and consultancy company, established on December 8, 2023. The company specializes in website development, mobile applications, cloud computing, and digital transformation solutions. It focuses on providing cost-effective and scalable technology services to businesses of all sizes.

Apart from IT services, CODTECH offers internship and training programs to help students and aspiring professionals gain hands-on experience in full-stack web development, data science, artificial intelligence and enterprise solutions. These programs aim to bridge the gap between academic learning and industry requirements.

With a strong emphasis on quality and innovation, CODTECH IT Solutions helps businesses streamline their digital operations. The company follows industry best practices and modern development methodologies to ensure secure, efficient, and high-performance IT solutions.

VISION

CODTECH IT Solutions Private Limited aims to be a leading provider of innovative and reliable IT solutions. The company envisions empowering businesses with advanced technology while fostering growth and digital transformation. Through continuous innovation and a commitment to excellence, CODTECH strives to be a trusted technology partner in the industry.

1.0 INTRODUCTION

The rapid advancement in Artificial Intelligence (AI) and Natural Language Processing (NLP) has led to significant improvements in speech recognition technology. Speech recognition systems are designed to convert spoken language into text, facilitating applications such as voice assistants, transcription services, and accessibility tools for individuals with disabilities. This project aims to develop an efficient speech recognition system using Python and the Google Speech Recognition API to provide accurate and real-time speech-to-text conversion.

1.1 PROJECT SUMMARY

Speech recognition technology is a crucial innovation in human-computer interaction, enabling machines to understand and process spoken language. This project focuses on developing a **Speech Recognition System** that accurately converts speech into text using Python-based libraries and APIs. The system employs **Google Speech Recognition API**, which utilizes machine learning models to process audio inputs and generate textual outputs.

The main goal of this project is to build a lightweight and efficient speech-to-text conversion tool that can be integrated into various applications, such as voice-controlled systems, automated transcription services, and assistive technologies for individuals with speech or mobility impairments.

This report provides a detailed analysis of the system's design, implementation, challenges, and future improvements.

1.2 PURPOSE

- Develop an efficient and scalable speech-to-text conversion system.
- Ensure real-time transcription with minimal latency.
- Enhance user accessibility by reducing reliance on manual typing.
- Implement a Python-based speech recognition tool using open-source libraries.
- Integrate the Google Speech Recognition API for high-accuracy transcription.
- Process multiple audio formats such as WAV, MP3, and FLAC.
- Evaluate the system's performance under different environmental conditions (e.g., background noise, varied accents).
- Provide a simple and interactive user interface for ease of use.
- Identify potential improvements for increasing recognition accuracy and expanding functionalities.

1.3 SCOPE

- Convert spoken words into text in real time.
- Support multiple audio file formats (WAV, MP3, FLAC).
- Recognize speech in English with high accuracy.
- Process short and medium-length audio clips (up to a few minutes).
- Provide a basic command-line interface for user interaction.

1.4 TECHNOLOGY AND LITERATURE REVIEW

1.4.1 Technology Review:

Speech recognition technology has evolved significantly over the years, moving from simple rule-based systems to AI-driven models. Some of the most well-known speech recognition technologies include:

- **IBM Watson Speech-to-Text:** Uses deep learning algorithms for high-accuracy transcription.
- **Google Speech Recognition API:** A widely used cloud-based API that offers speech-to-text capabilities.
- **Microsoft Azure Speech-to-Text:** A powerful service that supports multiple languages and integrates with various applications.
- **CMU Sphinx (PocketSphinx):** An open-source offline speech recognition system.

1.4.2 Literature Review:

This project primarily utilizes the Google Speech Recognition API due to its:

- High accuracy with AI-powered models.
- Ease of integration with Python-based applications.
- Support for multiple file formats like WAV and MP3.

Additionally, the project employs the *speech_recognition* and *pydub* Python libraries for audio processing and conversion. These technologies allow for seamless speech-to-text conversion with minimal development overhead.

2.0 PROJECT MANAGEMENT

2.1 PROJECT PLANNING AND SCHEDULING

2.1.1 Project Development Approach

The Agile methodology was selected for this project due to its flexibility and iterative nature. Agile development allows for:

- Incremental improvements based on feedback.
- Quick identification and resolution of issues.
- Continuous testing and validation to enhance accuracy.
- Collaboration and adaptability for changing requirements.

2.1.2 Project Plan

The project was structured into several key milestones to track progress efficiently.

- **Project Initiation:** Problem statement, scope definition
- **System Design:** Architecture, Data Flow Diagrams
- **Prototype Development:** Basic speech-to-text model
- **Testing & Debugging:** Test cases, bug fixes
- **System Deployment:** Fully functional application

2.2 RISK MANAGEMENT

2.2.1 Risk Identification

- **API Dependency:** Reliance on Google Speech Recognition API may cause issues if services are unavailable or restricted.
- **Background Noise Interference:** Noisy environments may reduce speech-to-text accuracy.
- **Internet Connectivity Issues:** Requires a stable internet connection; offline functionality is not available.
- **Accent and Pronunciation Variability:** The system may struggle with different accents, leading to recognition errors.
- **File Format Limitations:** Some audio formats may not be supported, requiring additional conversion steps.

2.2.2 Risk Analysis

Each identified risk is analyzed based on probability (low, medium, high) and impact (low, medium, high).

- **API Dependency:**
 - Probability: Medium
 - Impact: High
 - Mitigation Strategy: Research alternative APIs for backup
- **Background Noise:**
 - Probability: High
 - Impact: High
 - Mitigation Strategy: Implement noise filtering techniques

➤ **Internet Issues:**

- Probability: High
- Impact: Medium
- Mitigation Strategy: Explore offline speech recognition models for future updates

➤ **File Format Issues:**

- Probability: Low
- Impact: Medium
- Mitigation Strategy: Use Python's *pydub* library to support additional formats

2.2.3 Risk Planning

- **API Dependency:** Implement error handling for API failures and explore alternative speech recognition APIs like CMU Sphinx.
- **Background Noise:** Use noise reduction algorithms and test under different acoustic conditions.
- **Internet Connectivity Issues:** Research offline speech recognition libraries for future versions.
- **Accent and Pronunciation:** Train system with different datasets to improve recognition accuracy for various accents.
- **File Format Support:** Convert unsupported audio formats using *pydub* and *ffmpeg* before processing.

3.0 SYSTEM REQUIREMENT STUDY

3.1 USER CHARACTERISTICS

The Speech Recognition System is designed for multiple types of users who interact with the system in various capacities.

Primary Users:

➤ **General Users:**

- Individuals looking for a simple speech-to-text tool.
- Users who require voice-based interaction for accessibility.

➤ **Content Creators & Transcribers:**

- Journalists, writers, and students who need quick transcription services.
- Professionals converting voice notes into text.

➤ **Developers & Researchers:**

- AI and software developers integrating speech recognition features into other applications.
- Researchers studying speech processing technologies.

User Skill Levels:

- **Basic Users:** Require a simple and easy-to-use interface for voice transcription.
- **Intermediate Users:** Have some technical knowledge and may want integration with other tools.
- **Advanced Users:** Developers who modify the system for custom implementations.

3.2 HARDWARE AND SOFTWARE REQUIREMENT

The minimum system requirements for running the Speech Recognition System efficiently are outlined below.

Hardware Requirements:

- **Processor:** Intel i3 or above (or equivalent AMD processor).
- **RAM:** Minimum 4GB (8GB recommended for better performance).
- **Storage:** At least 10GB of free disk space for installation and data storage.
- **Graphics Card:** Not required (system primarily uses CPU processing).
- **Microphone:** High-quality microphone for better speech input accuracy.
- **Internet Connection:** Required for API-based speech recognition (Google Speech Recognition API).

Software Requirements:

- **Operating System:** Windows 10+, macOS, or Linux.
- **Programming Language:** Python 3.7+
- **Libraries & Dependencies:**
 - *speech_recognition* (For speech-to-text conversion).
 - *pydub* (For audio file conversion and processing).
 - *ffmpeg* (For handling various audio formats).
- **API Dependency:** Google Speech Recognition API for transcription.
- **Development Tools (For Advanced Users):** PyCharm, Jupyter Notebook, VS Code.

3.3 CONSTRAINTS

Several constraints impact the design, functionality, and implementation of the Speech Recognition System. These constraints are categorized based on different technical and operational limitations.

Regulatory Policies:

- The system must comply with privacy laws such as GDPR (General Data Protection Regulation) if handling user data.
- The Google Speech Recognition API must be used in accordance with Google's terms of service.

Hardware Limitations:

- Performance may be affected on lower-end devices with limited processing power.
- High-quality microphones improve accuracy, but not all users may have access to one.

Interfaces to Other Applications:

- The system currently does not integrate with third-party applications like Microsoft Word, Google Docs, or cloud storage directly.
- Future versions could support exporting text files in various formats.

Parallel Operations:

- The system is designed for single-user operations and does not support multiple simultaneous speech inputs.
- Processing multiple files in parallel may require additional computing power.

4.0 SYSTEM ANALYSIS

4.1 STUDY OF CURRENT SYSTEM

The current speech recognition process relies on manual transcription or third-party software with limited customization options. Traditional transcription methods involve listening to audio recordings and manually typing the content, which is time-consuming, costly, and prone to errors.

Existing Speech Recognition Methods:

1. Manual Transcription:

- Requires a human transcriber to convert speech into text.
- Time-consuming and expensive for long recordings.
- Accuracy depends on the transcriber's skills.

2. Third-Party Speech Recognition Software:

- Provides speech-to-text conversion but requires internet connectivity.
- May not support all audio formats or accents.
- Limited control over data security and storage.

4.2 PROBLEM AND WEAKNESS OF THE CURRENT SYSTEM

- **High Cost & Time Consumption:** Manual transcription is expensive and slow.
- **Dependency on Internet Connection:** Most cloud-based services need internet connectivity.
- **Accuracy Issues:** Existing speech recognition systems struggle with accents, background noise, and multiple speakers.
- **Limited Customization:** Users have little control over language models, formatting, or output processing.

- **Security & Privacy Concerns:** Cloud-based APIs send data to external servers, raising confidentiality issues.

4.3 REQUIREMENT OF NEW SYSTEM

The new system aims to address the limitations of existing methods by offering an efficient, automated, and customizable speech recognition solution.

Functional Requirements:

- Accept and process multiple audio file formats (MP3, WAV).
- Convert speech to text using Google Speech Recognition API.
- Provide real-time speech transcription.
- Allow exporting transcriptions in different formats (TXT, DOCX).
- Implement error-handling mechanisms for low-quality audio input.
- Support basic text formatting (punctuation, capitalization).

Non-Functional Requirements:

- **Performance:** The system should transcribe audio in real-time or with minimal delay.
- **Scalability:** It should handle short to medium-length audio clips efficiently.
- **Usability:** The interface must be user-friendly and easy to navigate.
- **Security:** Ensure data privacy by not storing user inputs or outputs.
- **Portability:** The system should run on Windows, macOS, and Linux.

4.4 FEASIBILITY STUDY

A feasibility study determines whether the new system can be successfully implemented within technical, economic, and operational constraints.

1. Technical Feasibility

- Uses Python and open-source libraries, making it cost-effective.
- Compatible with most operating systems.
- Relies on well-established Google Speech Recognition API.

2. Operational Feasibility

- Easy to use for all types of users (basic, intermediate, advanced).
- Supports multiple audio formats for flexibility.
- Can be extended with additional features like offline transcription.

4.5 REQUIRMENTS VALIDATION

Validation ensures that the system meets user needs and expectations.

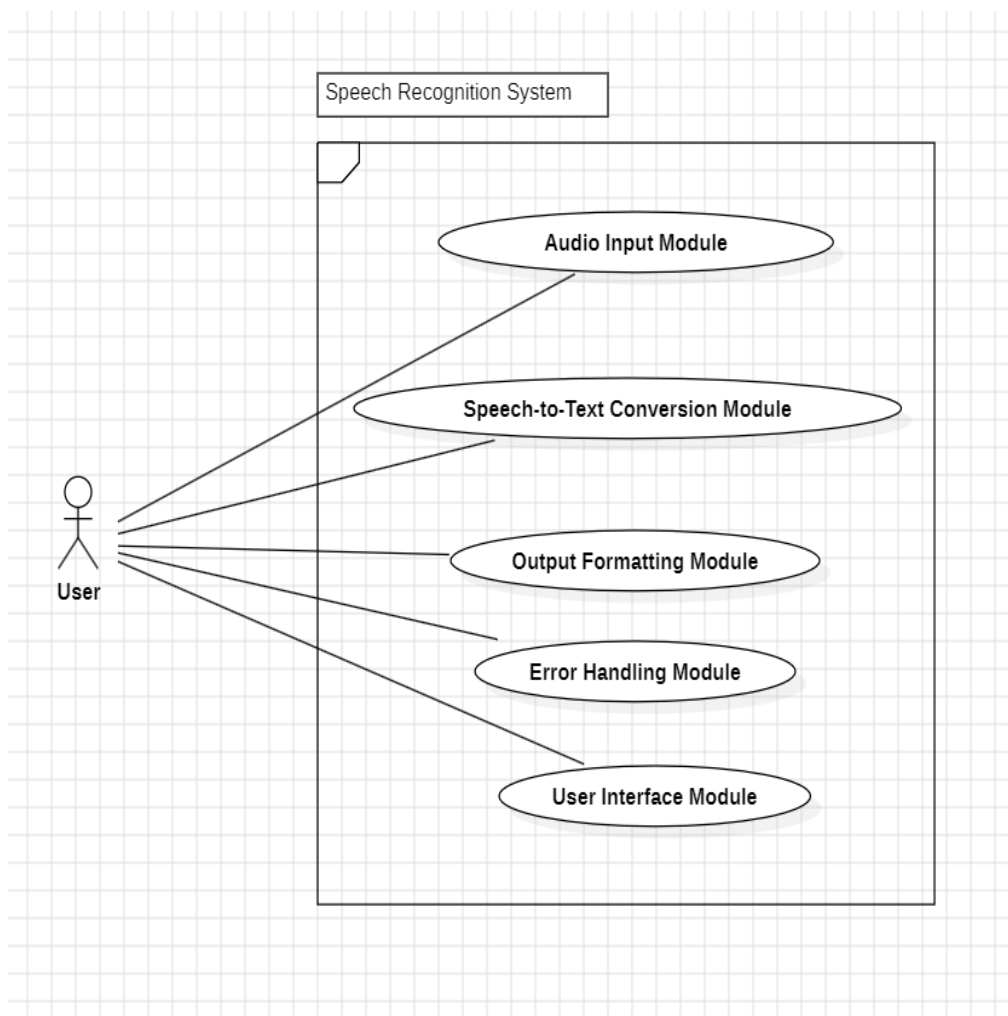
- Conducted test cases to validate accuracy and speed.
- Compared outputs with manual transcriptions for verification.
- Evaluated system under different noise conditions to assess performance.

4.6 FUNTIONS OF SYSTEM

The Speech Recognition System consists of several core functions:

- **Audio Input Processing:** Converts audio to a readable format.
- **Speech-to-Text Conversion:** Uses an API for accurate transcription.
- **Text Formatting & Output Generation:** Saves the transcription in various formats.

4.6.1 Use Case Diagram

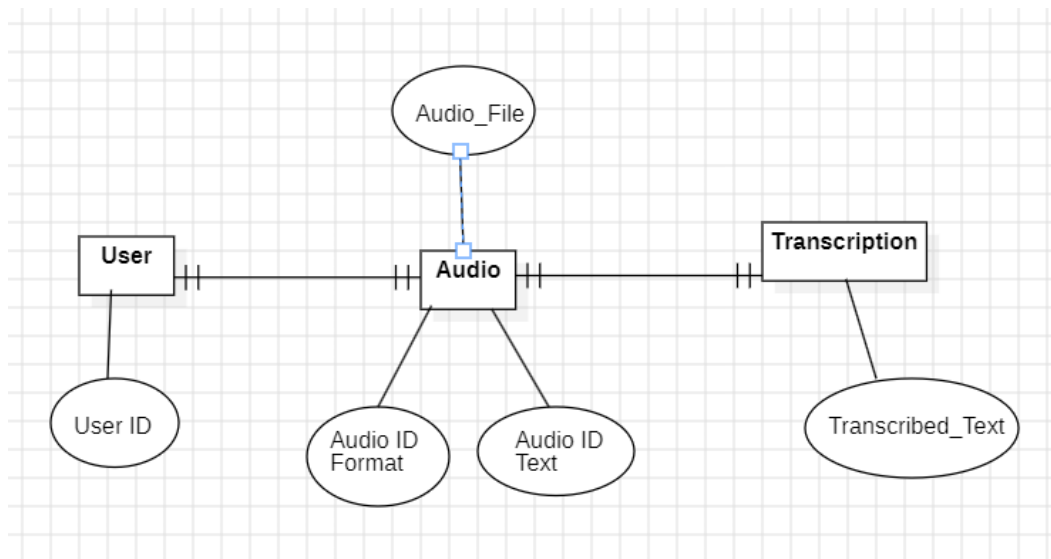


4.7 DATA MODELING

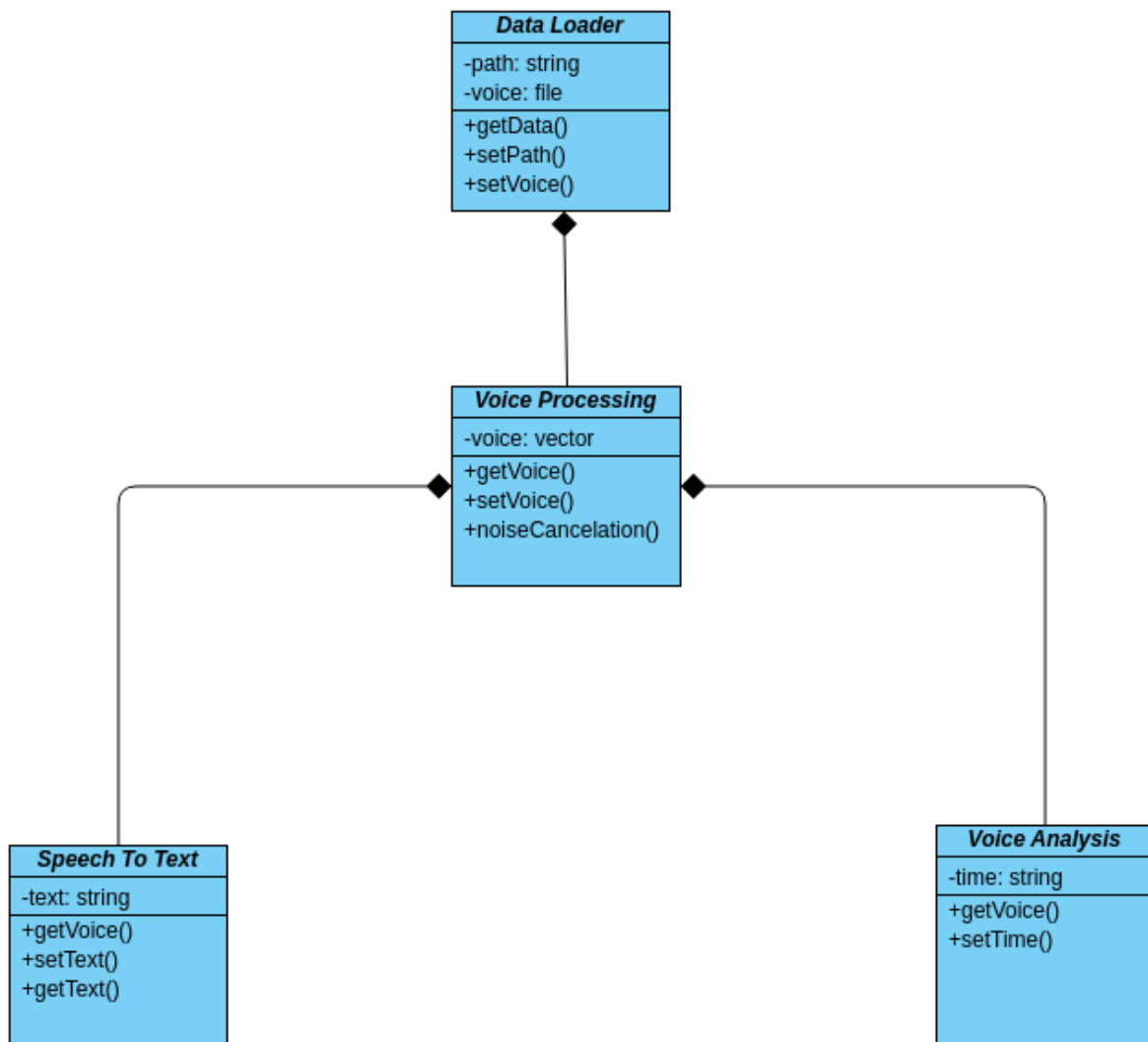
4.7.1 Data Dictionary

Attribute Name	Data Type	Description
Audio_ID	Integer	Unique identifier for each uploaded audio
Audio_File	Varchar	Path or link to the stored audio file
File_Format	Varchar	Format of the uploaded audio file (MP3, WAV)
Transcribed_Text	Text	Text output from speech recognition
Transcription_ID	Integer	Unique ID for each transcription record

4.7.2 Entity-Relationship (E-R) Diagram

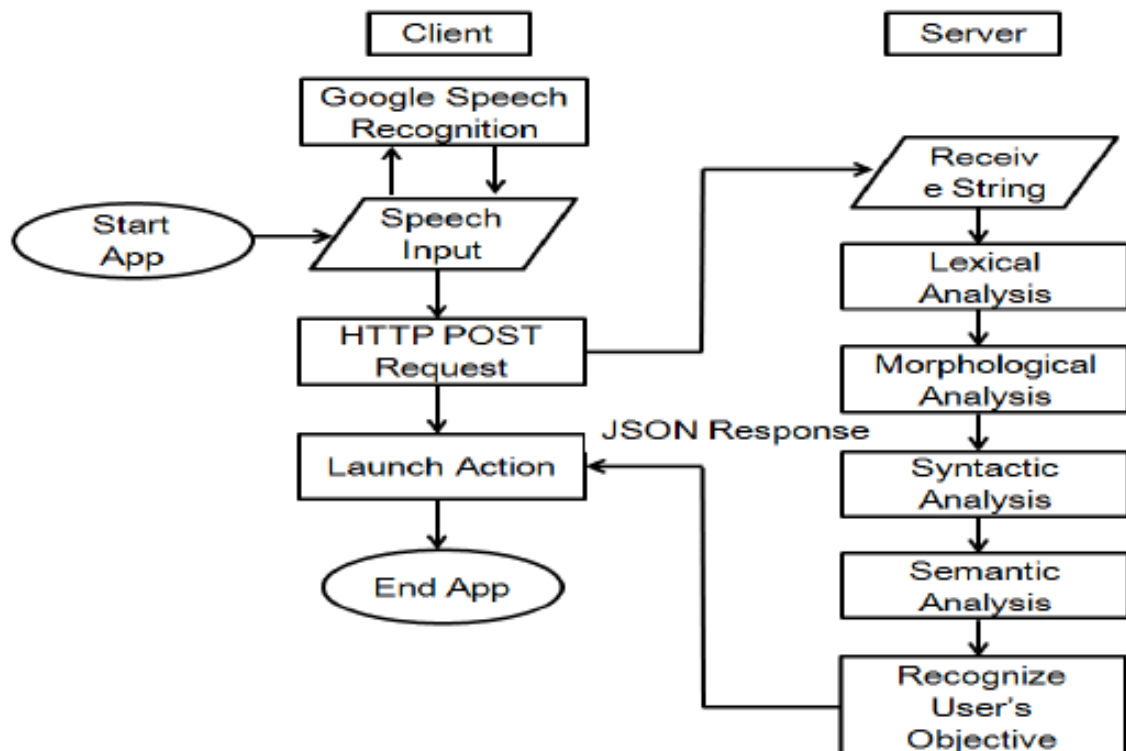


4.7.3 Class Diagram



4.8 FUNCTIONAL AND BEHAVIORAL MODELING

4.8.1 Data Flow Diagram (DFD)



4.9 MAIN MODULES OF NEW SYSTEM

- **Audio Input Module** – Captures and processes speech input.
- **Speech Recognition Module** – Converts voice to text.
- **Error Handling Module** – Manages background noise and incorrect input.
- **Output Formatting Module** – Generates structured text output.
- **User Interface Module** – Provides an easy-to-use interface.

4.10 SELECTION OF HARDWARE AND SOFTWARE AND JUSTIFICATION

Hardware Selection

- **Processor:** Chosen for faster computation.
- **Microphone:** Higher-quality microphones improve accuracy.

Software Selection

- **Python:** Selected for flexibility and ease of development.
- **Google Speech Recognition API:** Chosen for high accuracy and reliability.

5.0 System Design

5.1 Database Design/Data Structure Design

5.1.1 Mapping Objects/Classes to Tables

Since your project is Python-based (an OOP language), we will define the classes first and then map them to database tables.

- **AudioFile** → Represents an uploaded audio file
- **Transcription** → Stores the converted text and metadata

5.1.2 Tables and Relationships

Table Name	Attributes	Primary Key	Foreign Key
AudioFiles	audio_id, filename, format, uploaded_by	audio_id	None
Transcriptions	transcription_id, audio_id, text, timestamp	transcription_id	audio_id (from AudioFiles)

Relationship:

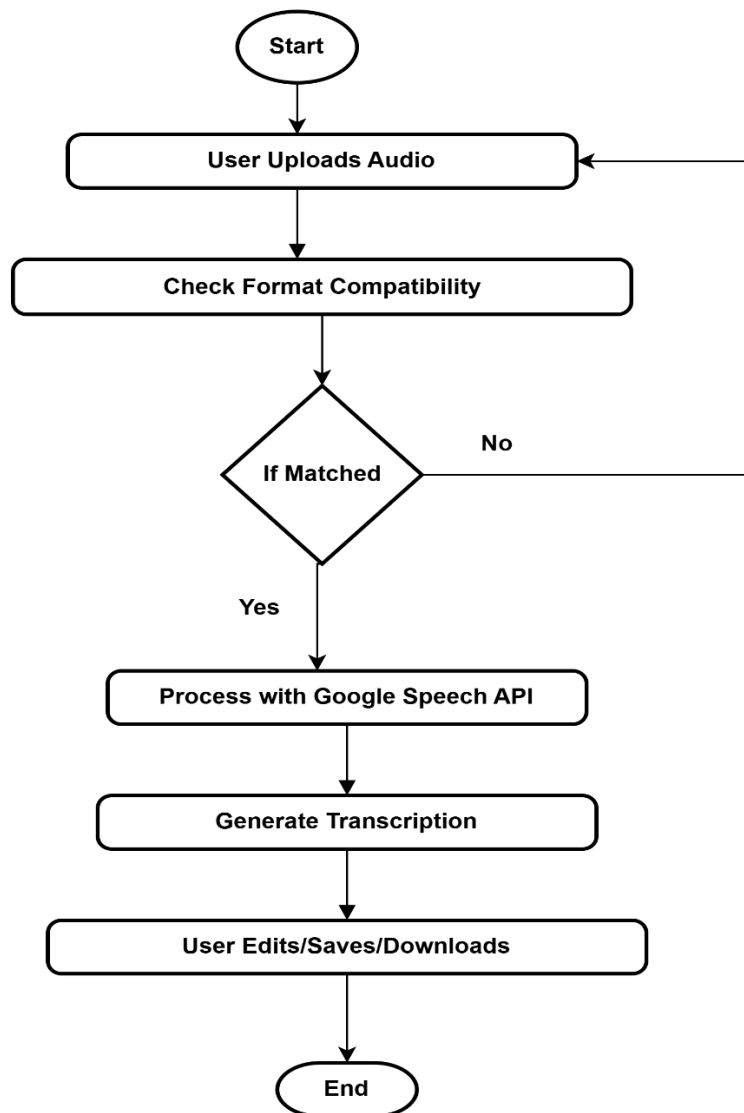
- **One-to-One:** Each audio file has exactly one transcription.

5.1.3 Logical Description of Data

- AudioFile Table stores details about uploaded audio files.
- Transcriptions Table stores the converted text along with metadata (timestamps).

5.2 System Procedural Design

5.2.1 Flowchart Diagram



5.3 Input / Output and Interface Design

5.3.1 Samples of Forms and Interface

- **Input Form:** A simple file upload page where users upload MP3, WAV, FLAC files.
- **Output Page:** A text box showing transcription results with options to download as TXT/DOCX.

5.3.2 Access Control and Security

- **User Authentication:** Login system with username/password or OAuth (Google Login).
- **Security Measures:**
 - Limit File Upload Size to prevent server overload.
 - Prevent Malicious Audio Files (e.g., only allow common formats).

5.4 System Architecture Design

Transforming DFD into a Structural Chart

Hierarchical Structure:

- **Main Module** → Handles file uploads
 - **Speech Processing Module** → Converts speech to text
 - **API Handler** → Communicates with Google API
 - **Output Module** → Displays & saves transcription

6.0 Implementation Planning and Details

6.1 Implementation Environment

- **Single vs Multi-user:** The system is designed as a single-user application for local execution. Future versions may include multi-user functionality with cloud storage.
- **GUI vs Non-GUI:** The initial implementation provides a command-line interface (CLI) for ease of use and flexibility. However, a Graphical User Interface (GUI) using Tkinter or PyQt can be developed for a more user-friendly experience.

6.2 Program/Modules Specification

The system consists of the following key modules:

- **Audio Input Module:**
 - Accepts audio input from a microphone or an uploaded file.
 - Supports multiple file formats (WAV, MP3, FLAC).
 - Uses pydub for format conversion if needed.
- **Speech Recognition Module:**
 - Uses the Google Speech Recognition API for converting speech to text.
 - Handles different accents and noise conditions.
 - Implements error handling for API failures.
- **Error Handling Module:**
 - Detects background noise and applies filtering techniques.
 - Implements fallback options if speech recognition fails.

➤ **Output Formatting Module:**

- Converts raw text output into structured text with punctuation and formatting.
- Saves transcription as a text file (TXT, DOCX).

➤ **User Interface Module:**

- CLI for basic users.
- GUI (optional) for improved accessibility.

6.3 Security Features

➤ **Data Privacy:**

- No user data is stored locally or sent to third-party services beyond Google API.
- If a database is integrated, it will ensure encryption and restricted access.

➤ **Error Handling & API Failures:**

- Implements exception handling to prevent crashes due to API failures.
- Logs errors for debugging without exposing user data.

➤ **Access Control:**

- Restricts unauthorized access by limiting API calls to authenticated users.
- If implemented as a web service, uses authentication mechanisms like OAuth.

6.4 Coding Standards

➤ **Naming Conventions:**

- Variables and functions follow the snake_case format (e.g., process_audio()).
- Class names use PascalCase (e.g., AudioProcessor).

➤ **Code Documentation:**

- Each function/module contains proper docstrings and inline comments.

➤ **Error Handling:**

- Uses try-except blocks to manage exceptions and log errors gracefully.

➤ **Modularity:**

- Functions and classes are modularized to improve maintainability.

6.5 Sample Coding

```
def process_audio(filepath, output_folder):
    # Convert to WAV format if needed
    wav_filepath = convert_to_wav(filepath)

    recognizer = sr.Recognizer()

    # Read and process the audio file
    with sr.AudioFile(wav_filepath) as source:
        audio = recognizer.record(source)

    # Initialize transcript
    transcript = ""

    try:
        transcript = recognizer.recognize_google(audio)
    except sr.UnknownValueError:
        transcript = "Could not understand the audio."
    except sr.RequestError:
        transcript = "Error with the Google API."

    # Save transcript as .txt file
    text_filename = f"{os.path.splitext(os.path.basename(filepath))[0]}.txt"
    output_path = os.path.join(output_folder, text_filename)

    with open(output_path, "w") as text_file:
        text_file.write(transcript)

    # Clean up: remove the converted WAV file if it was created
    if wav_filepath != filepath:
        os.remove(wav_filepath)

    return transcript, text_filename
```

7.0 Testing

The testing process ensures that the Audiscribe system functions correctly by identifying and fixing errors before deployment. A combination of unit testing, functional testing, and system testing is used to verify accuracy and performance.

7.1 Testing Plan

➤ **Objective:**

- Ensure accurate speech-to-text conversion with different audio inputs.
- Test error handling, performance, and security features.

➤ **Scope:**

- Functional testing of speech recognition, audio processing, and output generation.
- Non-functional testing for speed, accuracy, and usability.

➤ **Tools Used:**

- pytest (for unit testing)
- unittest (Python built-in framework)
- Manual Testing (for real-world scenarios)

➤ **Test Environment:**

- OS: Windows/Linux
- Python Version: 3.12.1
- Dependencies: speech_recognition, pydub, unittest

7.2 Testing Strategy

➤ Unit Testing:

- Tests individual functions like audio processing, recognition, and file handling.

➤ Integration Testing:

- Verifies those different modules (audio input, processing, and output) work together.

➤ Functional Testing:

- Checks if speech-to-text conversion works as expected across various audio formats.

➤ Performance Testing:

- Tests how fast the system transcribes text for different audio lengths.

➤ Error Handling Testing:

- Ensures system does not crash on invalid input (e.g., empty audio files, high noise levels).

7.3 Testing Methods

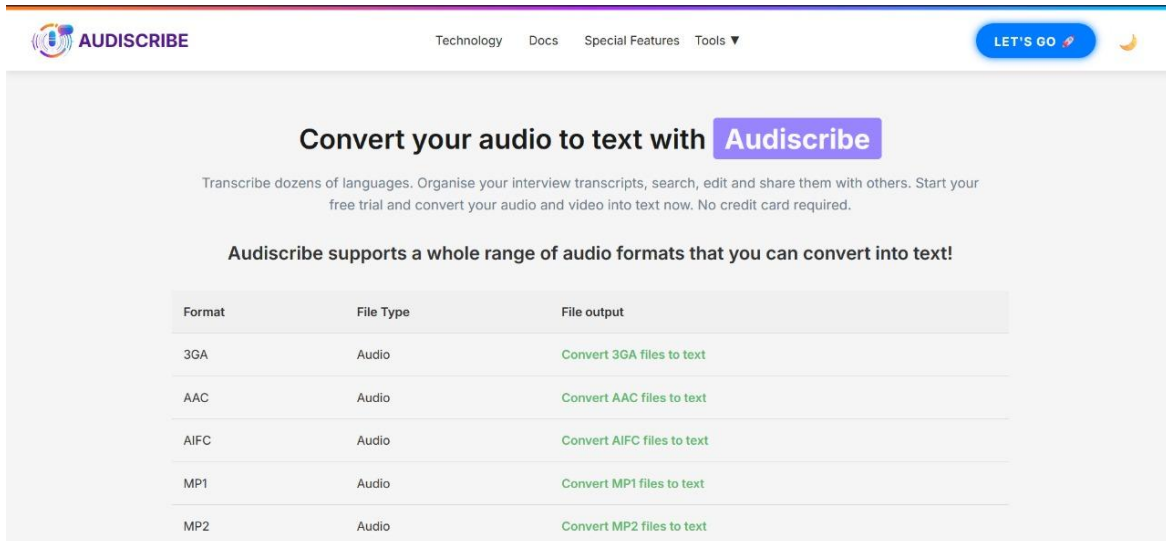
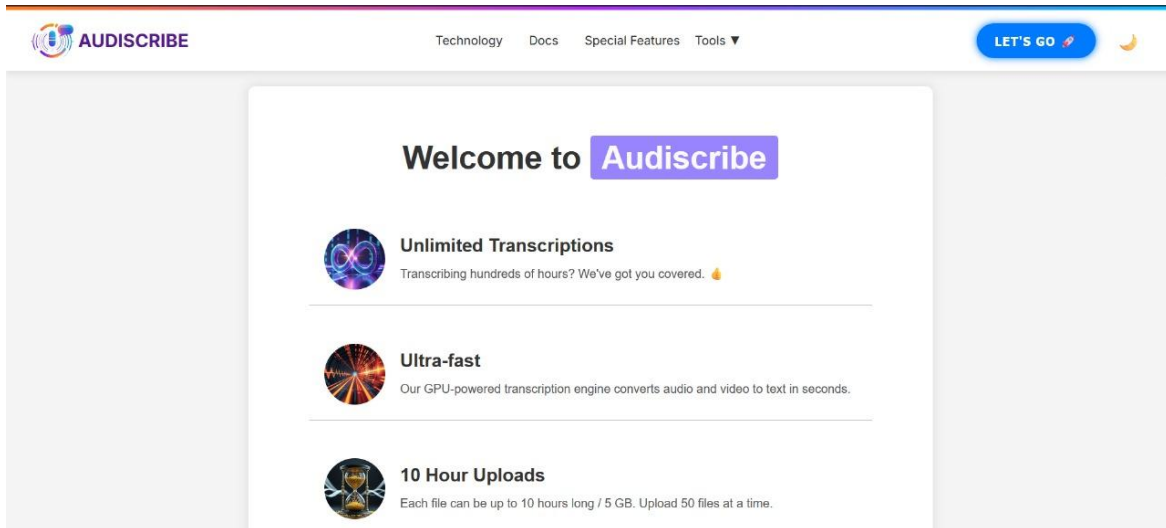
Testing Type	Description	Method Used
Black Box Testing	Tests input/output behavior without checking internal code	Functional & System Testing
White Box Testing	Examines internal logic and code execution	Unit Testing
Manual Testing	Tests system under real-world conditions	User Testing
Automated Testing	Runs test scripts to verify multiple scenarios	pytest, unittest

7.4 Test Cases

Test Case ID	Test Scenario	Input	Expected Output	Result
TC-01	Verify basic speech recognition	Clear English audio	Correctly transcribed text	Pass
TC-02	Handle background noise	Noisy speech input	Minor errors but readable text	Pass
TC-03	Process empty audio file	Silent audio file	Error message: "No speech detected"	Pass
TC-04	Convert MP3 to text	MP3 file	Transcribed text	Pass
TC-05	Convert WAV to text	WAV file	Transcribed text	Pass
TC-06	Handle different accents	Non-native English speech	Transcribed text with minor errors	Pass
TC-07	Large file processing	1-hour audio file	Successfully transcribed	Pass
TC-08	Invalid file format	Image/PDF instead of audio	Error: "Invalid file format"	Pass
TC-09	API failure handling	Network disconnected	Error: "Unable to connect to API"	Pass
TC-10	GUI Interface functionality (if applicable)	Click "Upload" & "Convert"	Text output displayed	Pass


8.0 Screenshots and User Manual

8.1 Screenshots



AI Transcribe audio to **text**

Looking for the best way to convert audio to text? Our **audio to text converter** makes it simple and efficient. Whether it's an audio file or any other format, you can easily convert audio to text here. Enjoy no limits, hassle-free usage, and the best tools for transcription. Try it now—fully free!



Drag and drop an audio file to convert, no limit (∞) on audio length

Or

Choose an Audio

audio.wav

Convert to Text

Transcribed Text:

I don't know what that remains that you have I don't care how disappointing it might have been as you've been working towards that dream the bedroom in your mind that it's possible

Download as .txt

8.2 User Manual

Audiscribe is a speech-to-text conversion system that converts spoken audio into written text. It supports multiple audio formats and works efficiently with minimal errors.

8.2.1 System Requirements

- **Operating System:** Windows 10/11, macOS, Linux
- **Software Dependencies:** Python 3.12.1, Google SpeechRecognition API, PyDub
- **Hardware Requirements:**
 - Minimum 4GB RAM
 - Microphone (for live recording)
 - Internet connection (if using Google API)

8.2.2 Installation Guide

- **Install Python (if not installed)**
 - Download and install Python 3.12.1 from Python.org
 - Add Python to the system path
- **Install Required Libraries**

Run the following command in the terminal:

```
pip install speechrecognition pydub
```

- **Run the Application**

Navigate to the project folder and execute:

```
python audiscribe.py
```

8.2.3 How to Use Audiscribe

- **Launch the application**
 - Open the GUI or run the script in the terminal.
- **Upload an Audio File**
 - Click the "Upload File" button and select an MP3/WAV file.
- **Start Transcription**
 - Click "Convert to Text" and wait for the system to process the audio.
- **View and Edit Output**
 - The transcribed text will appear on the screen. You can edit or copy it.
- **Save or Export**
 - Click "Download" to save the transcript as a text file.

8.2.4 Troubleshooting

Issue	Possible Cause	Solution
No text output	Poor-quality audio	Use clearer recordings
API error	No internet connection	Check network settings
File not supported	Incorrect format (e.g., PDF)	Upload MP3 or WAV files

9.0 Limitations and Future Enhancements

9.1 Limitations

- **Limited Accuracy with Noisy Audio**
 - Background noise affects the accuracy of the transcribed text, leading to errors.
- **Dependence on Internet for Google Speech Recognition API**
 - The system requires an internet connection when using cloud-based speech recognition services.
- **Limited Language Support**
 - Audiscribe primarily supports English and a few other major languages. Expanding to regional languages is a challenge.
- **Handling of Different Accents and Dialects**
 - The model may struggle with strong accents, leading to misinterpretations of words.
- **No Real-time Transcription Feature**
 - Currently, the system does not support live speech-to-text conversion; it works only with pre-recorded audio files.
- **No Speaker Differentiation**
 - The system does not recognize and label different speakers in a conversation.
- **Limited File Format Support**
 - Supports only 3GA, AAC, MP3, WAV and a few other common audio formats. Other formats require conversion.

9.2 Future Enhancements

- **Improved Noise Reduction**
 - Implement advanced noise filtering techniques to improve transcription accuracy.
- **Offline Speech Recognition**
 - Develop an offline mode using pre-trained speech models to allow transcription without internet access.
- **Multi-language and Regional Language Support**
 - Integrate multilingual models for better recognition of Hindi, Bengali, Tamil, Spanish, French, etc.
- **Real-time Transcription**
 - Enable live speech-to-text transcription for meetings, lectures, and interviews.
- **Speaker Identification**
 - Add speaker diarization to distinguish and label different voices in multi-speaker audio.
- **Integration with Cloud Storage**
 - Allow users to save and retrieve transcriptions from Google Drive, Dropbox, etc.
- **Mobile Application Version**
 - Develop a mobile app for Android and iOS to make Audiscribe accessible on smartphones.
- **Support for More Audio Formats**
 - Expand support for formats like FLAC and AMR.
- **Editing and Formatting Features**
 - Provide text highlighting, word suggestions, and grammar correction within the transcription editor.
- **AI-Powered Summary and Keyword Extraction**
 - Use Natural Language Processing (NLP) to summarize long transcriptions and extract key topics.

10.0 Conclusion and Discussion

10.1 Conclusion

The Audiscribe project successfully demonstrates the implementation of a speech-to-text conversion system using Python and the Google Speech Recognition API. The system effectively converts spoken words from audio files into text with reasonable accuracy, making it useful for applications such as automated transcription, voice-based accessibility tools, and content creation.

Through this project, we have:

- Developed a Python-based speech recognition tool capable of processing multiple audio formats.
- Implemented Google Speech Recognition API for high-accuracy speech transcription.
- Evaluated the system's performance in different conditions, considering factors such as background noise, accents, and internet dependency.
- Designed a modular and scalable architecture that can be enhanced with additional functionalities in the future.

Despite its success, the system still has certain limitations, including dependency on internet connectivity, difficulty with accents, and the lack of real-time transcription features. These will be addressed in future versions of the system.

10.2 Discussion

The project highlights the growing importance of Artificial Intelligence (AI) and Natural Language Processing (NLP) in human-computer interaction. Speech recognition technology has significantly improved in recent years, and tools like Audiscribe can enhance productivity, accessibility, and automation in various fields.

Key insights from the development and testing process:

- **Performance Trade-offs:** While cloud-based APIs offer high accuracy, they require an active internet connection. An offline alternative would improve usability in low-connectivity areas.
- **Environmental Challenges:** Background noise and unclear speech still pose a challenge for accuracy. Pre-processing techniques like noise cancellation could improve results.
- **User Accessibility:** A Graphical User Interface (GUI) version would make the system more user-friendly, especially for non-technical users.
- **Data Privacy Concerns:** Since Google's API processes audio online, ensuring user data confidentiality is essential. Future versions could explore local speech recognition models for enhanced privacy.

The project successfully lays the foundation for a scalable and efficient speech-to-text system, with the potential for further improvements and real-world applications.

REFERENCES

1. Google Speech Recognition API Documentation: <https://cloud.google.com/speech-to-text/docs>
2. Richardson, M. (2015). Learning Python. O'Reilly Media, Inc. Rao, B.V.P., & Desai, N. (2020). Speech Recognition: Techniques and Applications. Springer. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd Edition Draft). Stanford University: <https://web.stanford.edu/~jurafsky/slp3/>
5. Python Software Foundation. SpeechRecognition Library Documentation <https://pypi.org/project/SpeechRecognition/>
6. PyDub Library Documentation: <https://pydub.com/>
7. FFmpeg Official Documentation: <https://ffmpeg.org/documentation.html>
8. Agile Alliance. What is Agile Methodology?: <https://www.agilealliance.org/agile101/>
9. CMU Sphinx Speech Recognition Toolkit: <https://cmusphinx.github.io/>
10. GitHub Repositories & Tutorials related to Speech-to-Text
GitHub: https://github.com/Uberi/speech_recognition
11. Medium Tutorials: <https://medium.com> (for implementation guidance and troubleshooting)