TABLE OF CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 About The Project

The SPEAKO project is an innovative web-based Text-to-Speech (TTS) application developed using the Django framework to transform written text into natural and expressive speech. It supports multiple input formats such as PDF, DOC, and plain text, allowing users to either upload documents or type text directly into the web interface. One of its key features is automatic text summarization, which uses Natural Language Processing (NLP) techniques to shorten lengthy documents and make them easier to listen to. The project integrates powerful tools like Google Text-to-Speech (gTTS) and Amazon Polly to deliver realistic and high-quality audio output, with options to adjust voice type, pitch, speed, and language for a personalized experience. With its **clean, secure, and responsive interface**, SPEAKO ensures smooth functionality across both computers and mobile devices, catering to a wide range of users including **students, educators, professionals, and individuals with reading difficulties**. The system focuses on enhancing **accessibility, productivity, and user convenience** by converting complex written material into easily understandable spoken form. Overall, SPEAKO represents a modern and intelligent use of **AI-driven speech synthesis and text summarization**, promoting digital inclusivity and making information consumption more efficient and engaging.

1.1.1 Objective of The Project

The primary **objective of the SPEAKO project** is to develop an intelligent and user-friendly **Text-to-Speech (TTS) web application** that converts written content into clear and natural-sounding speech. The project aims to make reading and information access easier by allowing users to **upload text, PDF, or DOC files**, which can then be automatically **summarized and converted into audio**. By integrating **Natural Language Processing (NLP)** for summarization and advanced **speech synthesis technologies** such as Google TTS and Amazon Polly, SPEAKO enhances accessibility and saves users' time. It is designed to assist a wide range of users including **students, professionals, and individuals with reading difficulties** by providing customizable options like **voice type, language, pitch, and speed**. Ultimately, the objective of SPEAKO is to bridge the gap between written and spoken content, promote inclusivity, and improve the way people interact with digital text through innovativeAI-drivenfeatures.

1.1.2 Scope of The Project

The scope of the SPEAKO project includes the development of a powerful and user-friendly web-based Text-to-Speech (TTS) system that converts written content and uploaded documents into clear and natural-sounding speech. The application supports multiple file formats such as PDF, DOC, and plain text, allowing users to either upload files or type text directly into the system. It also features automatic text summarization using Natural Language Processing (NLP) to condense lengthy content before reading it aloud. With integrated tools like Google TTS and Amazon Polly, the system offers options to customize voice type, language, speed, and pitch, providing a personalized audio experience for every user.

The platform is designed to cater to **students, professionals, educators, and individuals with reading or visual difficulties**, making information more accessible and convenient. Its functionality spans across **text extraction, summarization, audio generation, playback, and download**, ensuring a complete and smooth workflow from document to speech.The system design emphasizes **ease of use, efficiency, and accessibility**, offering a clean and secure web interface.

- Requires minimal user training.
- The web application is **highly user-friendly and interactive**.
- Promotes **paperless and accessible learning**.
- Ensures **fast and accurate text-to-speech conversion**.
- Provides **efficient processing and easy traceability**.
- Avoids **data duplication and redundant uploads**.
- A **menu-driven interface** enhances user navigation.
- **Previously uploaded data and summaries** can be accessed for future use.

INITIAL INVESTIGATION AND FEASIBILITY STUDY

2.1 INITIAL INVESTIGATION

The initial investigation for the SPEAKO project highlights the increasing need for accessible and intelligent text-to-speech applications that make digital content easier to consume. In today's fast-paced world, users often struggle with reading lengthy documents, research papers, or study materials due to time constraints or accessibility challenges. This is especially true for students, professionals, and individuals with visual or reading difficulties who require efficient ways to access information without relying solely on visual reading. Existing text-to-speech tools are often limited, either by paid subscriptions, restricted file format support, or lack of summarization capabilities.

A **market analysis** reveals that while several TTS applications like **Amazon Polly**, **IBM Watson**, and **Natural Reader** provide text-to-speech services, most focus solely on speech generation and do not include features like **automatic summarization** or **multi-format document handling**. SPEAKO aims to fill this gap by offering an **integrated platform** that supports **PDF, DOC, and text files**, summarizes the content using **Natural Language Processing (NLP)**, and then converts it into **natural-sounding speech**. This combination of summarization and speech synthesis makes SPEAKO stand out as both a productivity and accessibility tool.

The **technical investigation** indicates that the project will be developed using the **Django framework (Python)** for the backend, providing a stable, scalable, and secure environment. It will integrate APIs like **Google Text-to-Speech (gTTS)** and **Amazon Polly** for high-quality audio output. The system will feature modules for **file upload, text extraction, summarization, voice customization, playback, and download**, all designed within a **user-friendly and responsive web interface**.

With its focus on usability, efficiency, and inclusivity, the SPEAKO project aims to deliver a **reliable and intelligent text-to-speech solution** that enhances learning, accessibility, and information consumption. It aligns with current technological trends in **AI-driven speech applications** and **digital accessibility**, catering to the growing demand for smarter, more user-centric communication tools.

2.2 FEASIBILITY STUDY

System feasibility is a test or evaluation of the complete system plan. Such an evaluation is necessary to define the application area along with the extent and capability to provide the scope of computerization together with suggested output and input format and potential benefits. Feasibility study is a proposal according to the work ability, impact on the organization, ability to meet user's needs, and efficient use of resources. The feasibility study for the project SPEAKO—a Text-to-Speech Web Application—has been conducted to determine whether the proposed system is practical, cost-effective, and implementable within existing technological and organizational constraints.

The feasibility analysis evaluates the candidate system and determines whether it meets performance requirements, user needs, and organizational goals. The purpose of this study is to investigate the current methods of text reading, evaluate the possible application of computer-based approaches, estimate costs, and assess the overall benefits of implementing the system. The study also examines the effect of the system on existing users and the need for additional personnel or training. Since all projects are feasible given infinite resources, it is essential to determine the most practical approach for successful development under real-world limitations.

The key combinations are involved in the feasibility study:

- Economic Feasibility.
- Technical Feasibility.
- Behavioral Feasibility.
- Operational Feasibility

2.2.1 Economic Feasibility

The *SPEAKO* application demonstrates strong economic feasibility with low initial investment for design, development, and deployment. It leverages open-source technologies such as Python, Django, spaCy, and gTTS, which significantly reduce software licensing and maintenance costs. Since the system operates on standard computer hardware with basic internet connectivity, infrastructure expenses remain minimal.From a functional perspective, *SPEAKO* improves productivity and accessibility by automating text summarization and speech generation, helping users save time and effort. These benefits provide substantial value for educational and professional users. In the long term, the system's efficient automation and scalability contribute to cost savings, making *SPEAKO* a financially viable and sustainable project.

2.2.2 Technical Feasibility

FFrom a technical perspective, *SPEAKO* is highly feasible, built using **Python** and the **Django framework** for a secure and scalable backend. It integrates **spaCy** for text summarization and **gTTS** for text-to-speech conversion, ensuring efficient and reliable performance. The frontend uses **HTML**, **CSS**, and **JavaScript**, providing a responsive and user-friendly interface. With minimal hardware needs and options for cloud deployment, *SPEAKO* is technically sound, easy to maintain, and adaptable for future enhancements.

2.2.3 Behavioral Feasibility

SPEAKO is expected to gain high user acceptance due to its simplicity and convenience. The application features a **user-friendly interface** that allows users to upload text or documents and convert them into speech with customizable options for voice, pitch, and speed. Designed for **students, professionals, and users with reading difficulties**, it requires little to no training. By improving accessibility and saving time, *SPEAKO* ensures smooth user adaptation and positive engagement.

2.2.4 Operational Feasibility

The operational feasibility of *SPEAKO* is reinforced by its efficient architecture and streamlined functionality. The system enables users to easily upload text, PDF, or DOC files and generate clear, natural-sounding speech outputs with minimal effort. Built on the Django framework, the platform ensures smooth operation, secure data handling, and reliable performance across devices. Designed for scalability, *SPEAKO* can accommodate an increasing number of users and files as demand grows. Its modular design—including text extraction, summarization, and text-to-speech components—ensures efficient resource management and easy maintenance. The system's strong focus on accessibility and user convenience makes it highly practical for educational, professional, and personal use. Overall, *SPEAKO* can operate reliably and efficiently to meet both present and future user needs.

SYSTEM ANALYSIS

3.1 System Analysis

Analysis is a structured method for identifying and solving problems. Analysis implies breaking something into its parts so that the whole may be understood. The definition of system analysis not only process analysis but also that of synthesis, which implies the process of putting parts together to form a new whole. All the activities relating to the life cycle phase must be performed managed and document. To design a system, we need requirements of the system and the specification document are prepared in this phase. The purpose of this document is to specify the functional requirement of the software that is to build. The specifications are intended to guide the activities, relationships and all other objectives. The main thing is to find what is to be done to solve the problems with the current system. In the phase the problems or drawbacks of the current system is identified and the necessary actions to solve these problems are recommended

3.2 Existing System

The existing systems for converting text to speech are limited in functionality and accessibility. Most available tools, such as Amazon Polly, IBM Watson, and Microsoft Azure Speech Services, primarily focus on text-to-speech conversion but lack integrated features like document upload, summarization, and customization. These services are often paid and provide limited free access, making them less suitable for regular users or students.

Other available applications, including **Natural Reader** and **Speechify**, allow users to listen to text but typically support only a few file formats and require premium subscriptions for advanced features. Older offline tools like **eSpeak**, **Festival**, and **MaryTTS** deliver low-quality, robotic voices and lack user-friendly interfaces, especially on mobile platforms.

Overall, the existing systems are fragmented and fail to provide a unified, convenient, and cost-effective solution. Users face challenges such as limited file compatibility, lack of summarization, restricted customization options, and poor accessibility. This highlights the need for an intelligent, web-based platform like *SPEAKO* that can seamlessly combine document support, smart summarization, and natural-sounding speech output within a single, easy-to-use interface.

3.3 Proposed System

The proposed system, *SPEAKO*, is a web-based Text-to-Speech (TTS) application designed to overcome the limitations of existing systems by offering an integrated and user-friendly solution. Built using the Django framework, *SPEAKO* allows users to upload text, PDF, or DOC files, automatically summarize lengthy content, and convert it into natural-sounding speech.The system uses advanced **Natural Language Processing (NLP)** tools such as **spaCy** for summarization and **gTTS (Google Text-to-Speech)** for high-quality audio generation. Users can customize the **voice type, pitch, speed, and language**, providing a personalized listening experience. With a clean and responsive interface, the platform ensures accessibility across both desktop and mobile devices.In addition to its core functionality, *SPEAKO* prioritizes **security and scalability**, supporting encrypted file handling and HTTPS-based communication. Its modular architecture—comprising text extraction, summarization, speech synthesis, and playback modules—ensures efficient performance and easy maintenance. By combining document processing, smart summarization, and customizable speech output in one platform, *SPEAKO* offers an innovative and accessible solution for students, professionals, and users with reading difficulties.

3.4 Software Requirement Specification (SRS)

The Software Requirement Specification (SRS) defines the complete description of the *SPEAKO* application, outlining its purpose, functionality, performance, and design constraints. It serves as a foundation for system development by describing what the system will do and how it will perform from both the user and developer perspectives.

A. Purpose

The purpose of *SPEAKO* is to provide an efficient, web-based **Text-to-Speech (TTS)** system that allows users to upload text or document files (PDF, DOC) and convert them into clear, natural-sounding speech. The system also summarizes lengthy documents using AI-based tools, helping users save time and enhance accessibility.

B. Scope

SPEAKO is designed to assist students, professionals, and individuals with reading difficulties by simplifying content consumption. It supports file uploads, automatic summarization, speech customization, and audio playback or download. The system runs on web browsers and is compatible with both desktop and mobile devices.
Key objectives include:

- Converting text and documents into natural speech.
- Providing automatic text summarization.
- Allowing customization of voice, pitch, speed, and language.
- Ensuring data security and user-friendly operation.

C. Overall Description

❖ **Product Perspective**

The *SPEAKO* application is a web-based Text-to-Speech (TTS) system designed to convert text and document files into clear, natural-sounding speech. It also provides automatic text summarization, enhancing accessibility and convenience for users who wish to consume information efficiently. This section gives an overview of the system's key functions, user characteristics, constraints, and assumptions.

❖ **Product Functions**

• **Admin Functions:** Manage user accounts, monitor system activity, and maintain database and security settings.

• **User Functions:** Register, log in, upload text/PDF/DOC files, and generate or download speech output.

• **Text Processing Functions:** Extract text from files and summarize content using **spaCy**.

• **Text-to-Speech Functions:** Convert text or summaries into speech using **gTTS** with adjustable voice, pitch, speed, and language.

• **Playback Functions:** Play, pause, and download generated audio.

• **Security Functions:** Ensure data privacy and safe file handling through **HTTPS** encryption.

❖ **User Characteristics**

- **Admin:** Skilled in managing web applications, capable of overseeing user activities, maintaining security, and managing system operations.
- **User:** Students, professionals, and visually impaired individuals with basic computer and internet knowledge, using the system to convert text or documents into speech.

❖ Constraints

• Requires an **active internet connection** for text summarization and speech generation through online APIs.

• Dependent **on third-party services** like gTTS and spaCy, which may have limitations or usage caps.

• File **size and format restrictions** may apply for upload and processing.

• The system currently supports **browser-based access only**, limiting offline functionality.

• The **performance** may vary depending on network speed and server load.

❖ Assumptions and Dependencies

- Users have stable internet and modern web browsers.
- APIs like gTTS and spaCy remain available and functional.
- The Django environment supports all required configurations.
- Uploaded files are valid and secure.
- Future scalability possible through cloud deployment or API updates.

❖ Functional Requirements

• **User Interaction:** Allows users to upload text, PDF, or DOC files or enter text directly for processing.

• **Text Extraction:** Extracts readable text from uploaded document files.

• **Summarization:** Applies Natural Language Processing (NLP) techniques using **spaCy** to summarize lengthy content into concise, meaningful information.

• **Text-to-Speech Conversion:** Converts text or summarized content into natural, human-like audio using **gTTS (Google Text-to-Speech)**.

• **Customization Options:** Enables users to choose voice type, pitch, speed, and language to personalize the output.

• **Audio Playback and Download:** Provides options to listen to the generated audio or download it for offline use.

• **Security and Accessibility:** Uses secure (HTTPS) connections to protect data and ensures compatibility across devices and browsers.

❖ **Role-Based Access Control**

•**Admin:** Has full access to manage users, monitor system activities, maintain database records, and ensure application security.

• **User:** Can register, log in, upload text or documents, generate and customize speech, play or download audio, and manage personal content.

D. Non-Functional Requirements

❖ **Performance**

- The system should process and convert text to speech quickly with minimal delay.

❖ **Usability**

- The interface must be simple, intuitive, and accessible to all users with little or no training.

❖ **Security**

- All data transfers and file uploads must be secured through HTTPS and safe encryption methods.

❖ **Scalability**

- The application should handle multiple users and large files efficiently as usage grows.

❖ **Portability**

- The application should work smoothly on various devices and web browsers.

❖ **Maintainability**

- The system's modular design should allow easy updates and integration of new features.

❖ **Availability**

- The application should be accessible online at all times with minimal downtime.

E. User Interface Requirements

• The interface should be simple, responsive, and user-friendly for easy navigation.

• Users should be able to upload text/files, customize voice, and play or download audio easily.

F. System Architecture

❖ **System Design**

The The *SPEAKO* system is designed using a client-server architecture, where the client interacts with the web interface, and the server processes user requests through the Django framework. The server handles text extraction, summarization, and text-to-speech conversion using Python-based libraries, while the database stores user and processing information using SQLite.

- **Front-end:** HTML, CSS, JavaScript (Django Templates).
- **Back-end:** Python, Django Framework.
- **Database:** SQLite. **APIs/Libraries:** gTTS (Google Text-to-Speech), spaCy (NLP for Summarization).

❖ **Data Flow**

1. **User Interaction**: Users interact with the web application via a browser.
2. **Request Handling**: The Django server processes the requests, interacting with the database.
3. **Response Generation**: The server sends back the processed data to the client for display.

G. Conclusion

The *SPEAKO* project successfully delivers a smart and accessible **Text-to-Speech web application** that converts text, PDF, and DOC files into clear, natural-sounding speech. By combining **text summarization**, **multi-language support**, and **voice customization**, the system enhances user convenience and accessibility. Built using **Python** and **Django**, *SPEAKO* provides a secure, scalable, and user-friendly solution that benefits students, professionals, and individuals with reading challenges. Overall, it promotes efficient content consumption and demonstrates the effective integration of AI-driven tools in everyday applications.

MODULES

4.1 MODULES

In a software project, a module is a collection of source files and build settings that divides a project into distinct units of functionality. Modules can be used to: Make software easier to use, define program boundaries, Implement separate modules for different areas of a program, and partition the system design or code.
Modules can be designed to be reusable and can be called by the program as needed. They can also be imported from others, which may be called a library

4.1.1 Admin Module

The **Administrator** is responsible for managing the overall functioning of the *SPEAKO* application. The admin oversees user activities, maintains the database, ensures security, and manages reports related to system usage and performance.

a) The admin has full control over the system and its operations.
b) The admin logs in using a valid email ID and password to access the admin dashboard.
c) The admin can view all registered users and their activities.
d) The admin is responsible for monitoring file uploads, text-to-speech conversions, and summarization reports.
e) The admin can manage and maintain the database, including user details and processed files.
f) The admin is responsible for maintaining and updating the entire system to ensure smooth operation.
g) The admin has the provision to manage APIs, handle configuration updates, and ensure data security.
h) The admin can monitor system logs and performance reports for reliability and usage statistics.
i) The admin can view user feedback and take necessary actions for improvement.
j) The admin can respond to user complaints or technical issues related to file processing or speech generation.
k) The system provides a secure **logout option** for the admin to end the session safely.

4.1.2 User Module

The **User Module** allows registered users to interact with the *SPEAKO* application. Users can upload text or document files, generate speech, and customize audio output according to their preferences.

a) The system should have a provision for users to **register and log in** using a valid email ID and password.

b) After logging in, users should have the permission to **view and edit their profile information**.

c) Users should be able to **upload text, PDF, or DOC files** or manually enter text for speech conversion.

d) The system should allow users to **customize voice type, pitch, speed, and language** before generating audio.

e) Users should have the provision to **listen to or download** the generated audio output.

f) Users should be able to **view previous conversions** or access their audio history if available.

g) Users should have the option to **submit feedback or report issues** regarding the system's performance.

h) The system should include a secure **logout option** to safely end the session.

4.1.3 Summarization Collector

The **Summarization Module** in *SPEAKO* is responsible for condensing lengthy text or document content into concise and meaningful summaries. This helps users save time and focus on key information before converting it into speech.

a) The system should automatically **extract text** from uploaded files such as PDF and DOC.

b) The module should use **Natural Language Processing (NLP)** through the **spaCy library** to identify key sentences and generate summaries The collector should have the permission to edit the password that is already given by the admin.

c) Users should have the provision to **view the summarized text** before proceeding to speech conversion.

d) The module should maintain the **accuracy and coherence** of summarized content.

e) The summarized text should be **stored temporarily** for text-to-speech processing.

4.1.4 TTS Module

The **Text-to-Speech (TTS) Module** in *SPEAKO* is responsible for converting text or summarized content into natural-sounding audio output.

a) The system should use the **gTTS (Google Text-to-Speech)** API for generating realistic and high-quality speech.

b) Users should have the option to **customize voice settings**, including language, pitch, and speed.

c) The module should efficiently process the input text and generate speech with minimal delay.

d) The generated audio should be available for **instant playback or download.**

4.1.5 Playback/Download Module

The Playback and Download Module in *SPEAKO* allows users to listen to the generated speech output and download it for offline use.

a) The system should provide **audio playback controls** such as play, pause, and stop.

b) Users should have the option to **download the generated audio file** in a standard format.

c) The module should ensure **smooth and uninterrupted playback** within the browser.

d) The system should display **status messages or alerts** during audio generation and download processes**.**

4.1.6 Security Module

The **Security Module** in *SPEAKO* ensures the protection of user data, files, and system operations from unauthorized access or misuse.

a) The system should use **HTTPS encryption** to secure all communications between the client and serve.

b) Uploaded files should be **validated and scanned** to prevent malicious content.

c) The module should include **secure authentication** for user and admin login.

d) The system should implement **session management and logout features** to ensure data privacy**.**

FLOW DIAGRAMS

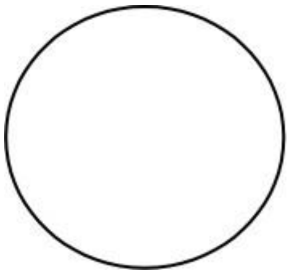
5.1 Data Flow Diagram(DFD)

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

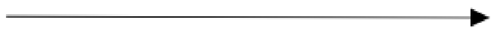
Data Flow Diagram Symbols

External entity: An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

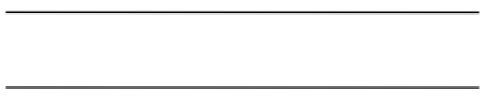
Process: Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence.



Data Flow: Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represent material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi- directional.



Data Store: Also known as warehouse. The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.



Levels of DFD

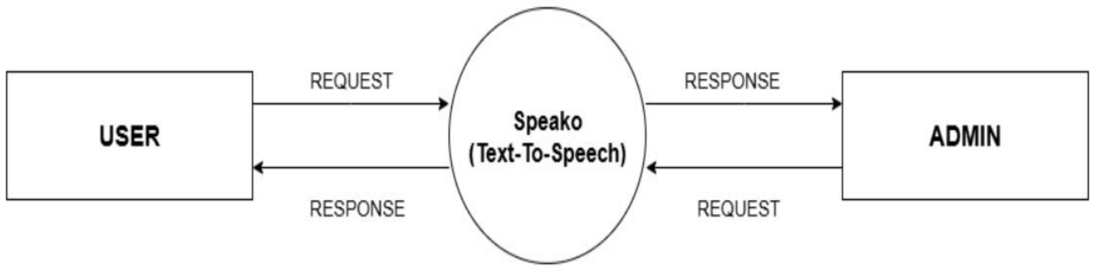
DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

- ■ 0-levelDFD
- ■ 1-levelDFD
- ■ 2-levelDFD

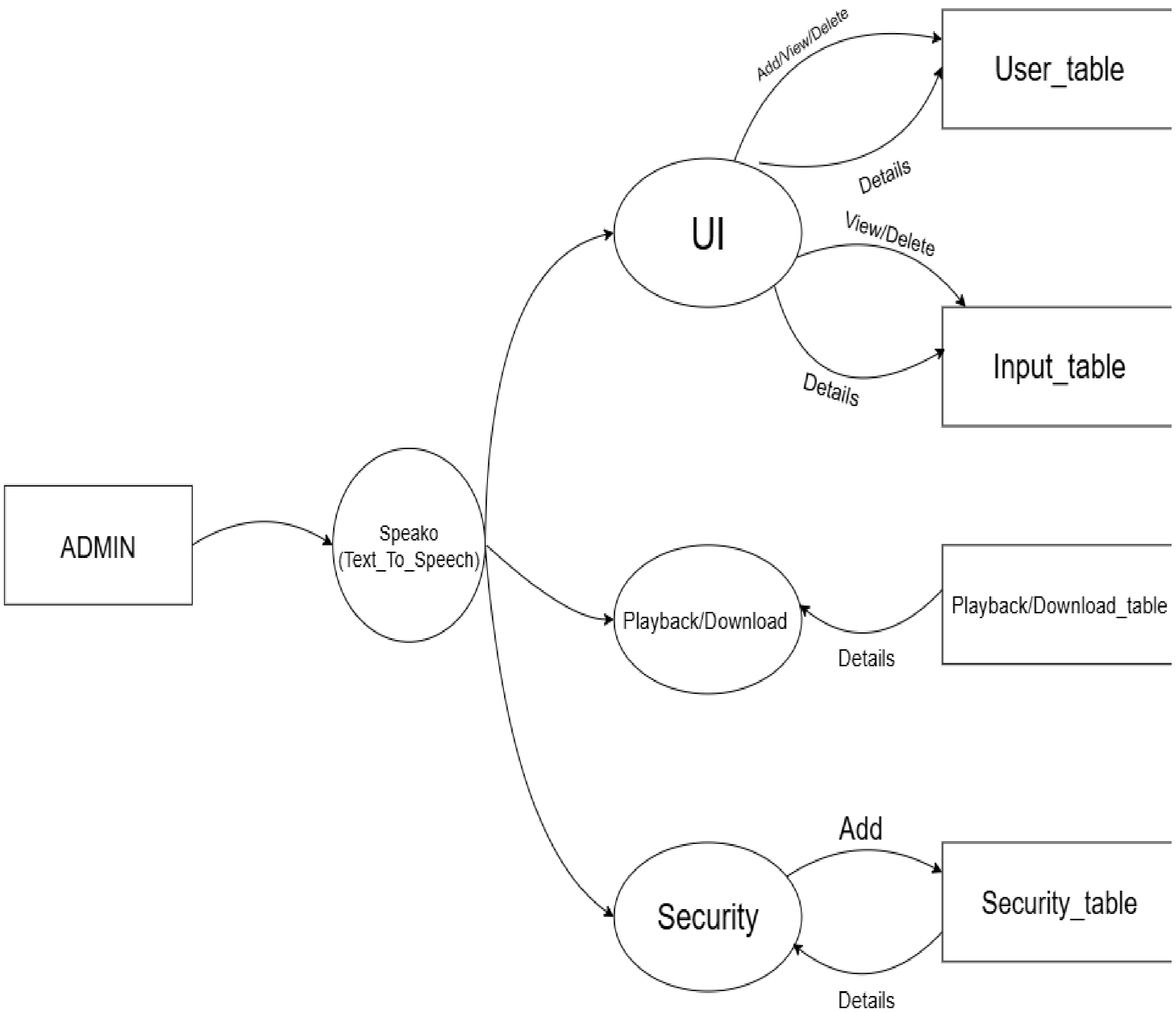
Rules for creating DFD

- ❖ The name of the entity should be easy and understandable without any extra assistance
- ❖ The processes should be numbered or put in ordered list to be referred easily.
- ❖ The DFD should maintain consistency across all the DFD levels.
- ❖ A single DFD can have maximum processes upto 9 and minimum 3 processes.

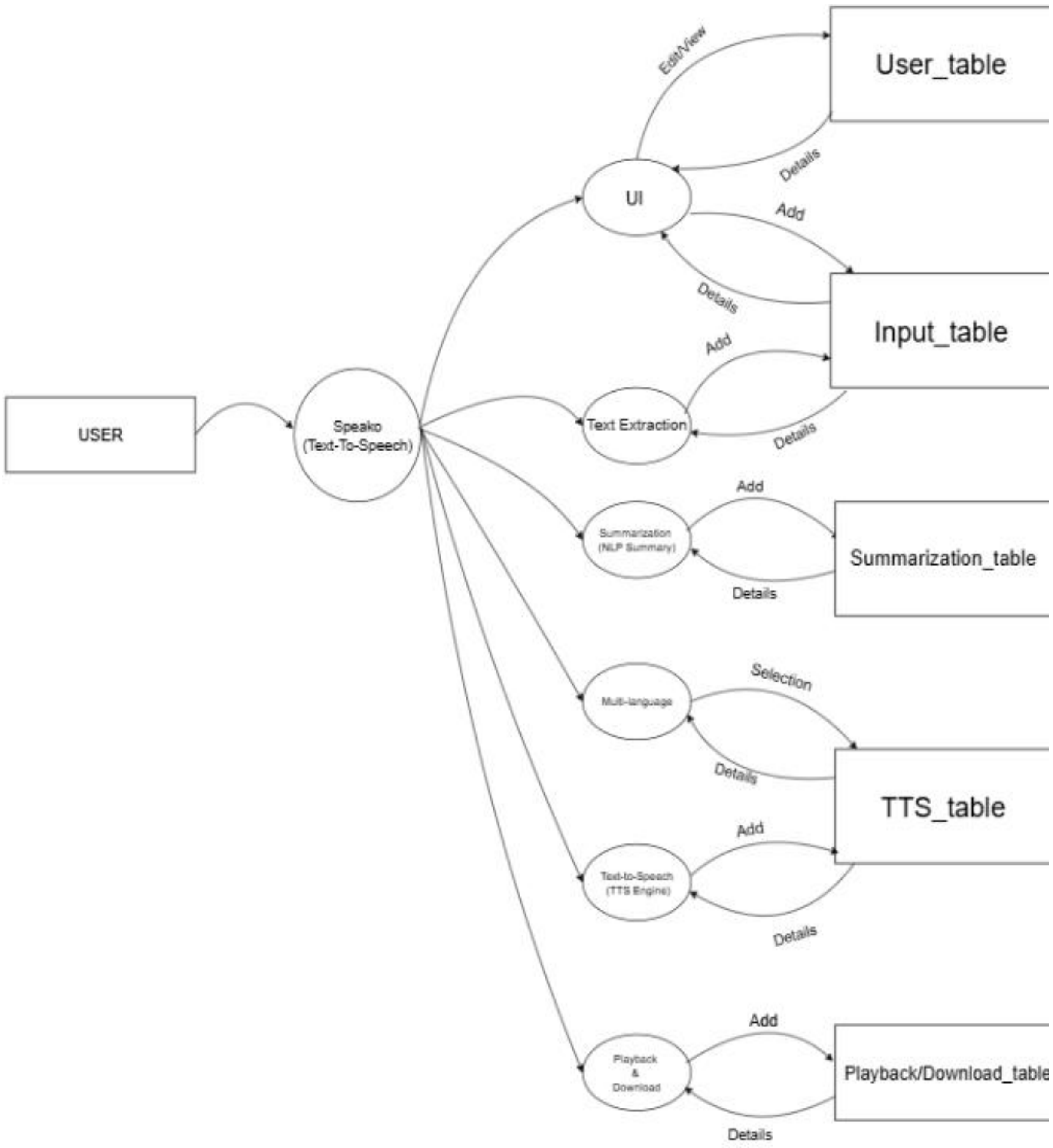
CONTEXT LEVEL-LEVEL 0 DFD



LEVEL 1-ADMIN



LEVEL 1-USER


Edit/View
User_table
Details
UI
Add
Details
Input_table
Add
Details
USER
Speako (Text-To-Speech)
Text Extraction
Add
Summarization (NLP Summary)
Summarization_table
Details
Selection
Multi-language
Details
TTS_table
Add
Text-to-Speech (TTS Engine)
Details
Add
Playback & Download
Playback/Download_table
Details

SYSTEM DESIGN

6.1 SYSTEM DESIGN

System designing is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. it is a solution to a "how to" approach compared to system analysis which is a "what is" orientation. it translates the system requirements into ways of making them operational. The design phase focuses on the detailed implementation of the system recommended in the feasibility study. The system which is in making is developed by working on two different modules and combining them to work as a single unit. That single unit is the one which is known as the new software. We go through the different design strategies to design the system we are talking about. In the input design we decide which type of input screens are going to be used for the system in making. In the output design we decide the output screens and the reports that will be used to give the output and in the database design we decide what all tables will be required and what all fields will be there in those tables. Each of them discussed briefly below.

6.2 Input Design

Input design converts user-oriented inputs to computer-based formats, which requires careful attention. The collection of input data is the most expensive part of the system in terms of the equipment used and the number of people involved. In input design, data is accepted for computer processing and input to the system is done through mapping via a map support or links. Inaccurate input data is the most common cause of errors in data processing. The input screens need to be designed more carefully and logically. A set of menus is provided which help for better application navigation. While entering data in the input forms, proper validation checks are done and messages will be generated by the system if incorrect data has been entered. The objective of input design is to create an input layout that is easy to follow and prevent operator errors. It covers all phases of input from creation of initial data into actual entry of the data to the system for processing. The input design is the link that ties the system into world of its users. The user interface design is very important for any application. The interface design defines how the software communication within itself, to system that interpreted with it and with human who use it. The input design requirements such as user friendliness, consistent format and interaction dialogue for giving the right message and help for the user at right time are also considered for the development of the project.

6.3 Output Design

Outputs are the most important and useful information to the user and to the department. Intelligent output designs will improve systems relationships with the user and help much in decision-making. Outputs are also used to provide a permanent hard copy of the results for later use. The forms used in the system are shown in the appendix. The outputs also vary in terms of their contents, frequency, timing and format. The users of the output, its purpose and

sequence of details to be printed are all considered. The output forms a system in the justification for its existence. If the outputs are inadequate in any way, the system itself is inadequate. The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose. Hence it is necessary to design output so that the objectives of the system are met in the best possible manner.

6.4 Table Design

The efficiency of an application using SQLITE-3 Server is mainly dependent upon the database tables, the fields in each table and joined using the fields contained in them to retrieve the necessary information. A table is a set of data elements that is organized using a model of vertical columns and horizontal rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index. The primary objective of a database design is fast response time to inquiries, more information at low cost, control of redundancy, clarity and ease of use, accuracy and integrity of the system fast recovery and availability of powerful end-user language.

There are mainly 9 tables in the project. They are,

1. User
2. Input
3. Summarization

4. TTS
5. Playback/Download Category
6. Security

1. Table: User

Description: Allows users to upload files, enter text, and customize voice settings.

Sl.no	**Field Name**	**Data type**	**Size**	**Constraints**
1	User_id	IntegerField		Primary key
2	Name	Varchar	100	Not Null
3	Email	Varchar	150	Not Null
4	Password	Varchar	50	Not Null
5	Role	Varchar	20	Default'User'
6	Created_at	Datetime		Current_timestamp
7	Last_login	Datetime		Null

2. Table: Input

Description: Extracts text from PDF and DOC files.

Sl.no	**Field Name**	**Data type**	**Size**	**Constraints**
1	Input_id	IntegerField		Primary key
2	User_id	IntegerField		Foreign key
3	Input_text	Text		Null
4	File_name	Varchar	500	Null
5	File_type	Varchar	100	Null
6	Extracted_text	LongText		Null
7	Status	Varchar	20	Default 'pending'
8	Created_at	DateTime		Current_Timestamp

3. Table: Summarization

Description: Summarizes long text using **spaCy** NLP tools.

Sl.no	Field Name	Data type	Size	Constraints
1	Summary_id	IntegerField		Primary key
2	Input_id	IntegerField		Foreign key
3	summary_text	LongText		Not Null
4	method	Varchar	50	Null
5	created_at	DateTime		Current_Timestamp

4. Table: TTS

Description: Converts text or summaries into speech using **gTTS**.

Sl.no	Field Name	Data type	Size	Constraints
1	TTS_id	IntegerField		Primary key
2	summary_id	IntegerField		Foreign key
3	language_code	Varchar	50	Not Null
4	voice_type	Varchar	50	Null
5	pitch	Varchar	20	Null
6	speed	Varchar	20	Null
7	audio_file	Varchar	255	Not Null
8	format	Varchar	50	Default'mp3'
9	created_at	DateTime	50	Current_Timestamp

5. Table: Playback/Download

Description: Lets users play or download generated audio.

Sl.no	Field Name	Data type	Size	Constraints
1	Playback_id	IntegerField		Primary key
2	TTS_id	IntegerField	50	Foreign key
3	Action_Type	Varchar	50	Not Null
4	Timestamp	DateTime	50	Current_Timestamp

6. Table: Security

Description: Ensures safe file handling and data protection.

Sl.no	Field Name	Data type	Size	Constraints
1	Sec_id	IntegerField		Primary key
2	User_id	IntegerField		Foreign key
3	Encryption_type	Varchar	50	Not Null
4	Token	Varchar	255	Null
5	Created_at	DateTime		Current_Timestamp

.

TOOLS AND PLATFORM

7. TOOLS AND PLATFORM

7.1 **Front End:** HTML, CSS, JavaScript, Django Templates

The The front end of *SPEAKO* is developed using **HTML**, **CSS**, **JavaScript**, and **Django Templates** to provide an interactive and responsive user experience. **HTML** structures the web pages and interface elements, while **CSS** enhances the design with proper layout, colors, and styles. **JavaScript** adds interactivity, allowing real-time responses like file uploads and audio playback controls. **Django Templates** integrate these components with backend data, enabling dynamic content rendering. Together, these technologies make *SPEAKO* visually appealing, easy to navigate, and compatible across devices.

7.2 Back End: Python

The back end of a Django application is developed using Python, a versatile and easy-to-read programming language. In Django, Python handles all server-side logic, including managing requests, processing data, and interacting with the database. Python's readability and extensive libraries make it ideal for creating robust and scalable web applications, enabling Django to provide a smooth, efficient backend experience.

7.3 FrameWork: Django

Django is a high-level web framework for building backend applications using Python. It simplifies web development by providing built-in tools for handling database management, URL routing, authentication, and more. Django follows the Model-View-Template (MVT) architecture, which promotes organized and maintainable code, making it easy to build robust and secure applications quickly.

7.4 Platform: Visual Studio Code

Visual Studio Code (VS Code) is a lightweight and powerful code editor used as the primary platform for developing and managing the project. With support for multiple programming languages, extensions, and features like debugging, syntax highlighting, and Git integration, VS Code makes coding, testing, and deploying applications convenient and efficient. It’s especially helpful for Django development, as it provides tools to streamline coding in Python and managing web files in one workspace.

7.5 Database: SQLite

SQLite is a lightweight, self-contained database engine that Django uses by default. It stores data in a single file on disk, making it simple to set up and ideal for development and small projects. SQLite requires minimal configuration and provides essential features for managing data, like creating tables, querying data, and handling relationships. Although it's not typically used for high-traffic applications, SQLite is highly effective for testing, prototyping, and small-scale applications due to its simplicity and portability.

TESTING

8. TESTING

8.1 Unit Testing

In *SPEAKO*, each module was tested individually to verify its functionality before integration. Unit testing focused on validating the smallest units of the software design, such as the **Text Extraction**, **Summarization**, **Text-to-Speech**, and **Playback** modules. Each module was executed separately to ensure it produced the expected outputs based on the design specifications.

During unit testing, different validation checks were applied for file formats, input text, and audio output. Errors detected within the modules were corrected immediately to ensure accurate and efficient performance. Unit testing helped verify that all modules independently worked as intended before combining them into the full system

8.2 Integration Testing

Data can be lost across an interface, one module can have an adverse effect on the other sub functions, when combined they may not perform the desired functions. Integrated testing is the systematic testing to uncover the errors within the interface. This testing is done with simple data and the developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance. After splitting the programs into units, the units were tested together to see the defects between each module and function. It is testing to one or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as part of unit or functional testing, integration testing can involve putting together of groups of modules and functions with the goal of completing and verifying meets the system requirements.

8.3 System Testing

System testing evaluated the complete *SPEAKO* application as a whole. It ensured that all modules and components, when integrated, performed effectively under different conditions. This phase confirmed that the entire system functioned reliably and met the overall objectives of performance, usability, and accessibility

The perquisites for System Testing are: -

• All the components should have been successfully Unit Tested.

• All the components should have been successfully integrated.

• Testing should be completed in an environment closely resembling the production environment. When necessary, iterations of System Testing are done in multiple environments.

8.3.1 Functional Testing

Functional testing was performed to ensure the *SPEAKO* system met all requirements specified in the SRS. It verified features such as **user login**, **file upload**, **summarization**, and **speech playback**. Each functionality was tested against expected outcomes to confirm proper system behavior and compliance with user needs.

8.3.2 Usability Testing

Usability testing ensured that the *SPEAKO* interface was simple, responsive, and easy to navigate. It tested the accessibility of controls such as file uploads, voice customization, and audio playback. The test results showed that users could operate the system effortlessly without prior training, confirming a positive user experience.

8.3.3 Performance Testing

Performance testing measured how efficiently the system handled large documents and multiple requests. It evaluated processing speed during **text extraction**, **summarization**, and **audio generation**. The system responded quickly, producing speech output with minimal delay, ensuring stability even under heavy use.

8.3.4 Security Testing

Security testing was carried out to ensure the safety of user data and file uploads. The system was tested for vulnerabilities and verified to operate under secure **HTTPS connections**. Checks were implemented to prevent malicious file uploads and unauthorized access, confirming that the system meets essential security standards.

8.3.5 Compatibility Testing

Compatibility testing ensured that the *SPEAKO* application worked smoothly across different **browsers (Chrome, Firefox, Edge)** and devices such as **desktops, laptops, and smartphones**. The tests confirmed consistent performance and display quality across all platforms.

8.4 Sample Test Cases

TC No.	Test Steps	Expected Result	Actual Result	Status
1	Run application and navigate to login screen	Login screen is displayed. A filed for entering username, a field for entering password and a button to submit should be present	Login screen has been displayed, fields for entering email address and password together with a log in button is available.	Pass
2	Enter an invalid username and invalid password and press the button	A message should be displayed stating that user name and password are invalid	A message has been displayed stating that user name and password are invalid	Pass
3	Enter a valid username and password and press the button	User must successfully login to the webpages.	A message has been displayed stating that the login successful and navigate into home page	Pass
4	Enter a valid username and leave password and press the button	A message should be displayed stating that please enter the user name and password	A message has been displayed stating that please enter the username and password	Pass
5	Leave username and password and press the button	A message should be displayed stating that please enter the user name and password	A message has been displayed stating that please enter the username and password	Pass
6	Leave username and enter a valid password and press the button	A message should be displayed stating that please enter the user name and password	A message has been displayed stating that please enter the username and password	Pass

SYSTEM IMPLEMENTATION

9.SYSTEM IMPLEMENTATION

9.1 Implementation

The implementation of the *SPEAKO* system involved setting up the development environment, integrating all modules, and deploying the application for use. The application was developed using **Python** and the **Django framework**, which provides a robust structure for handling both frontend and backend operations efficiently.

The **frontend** was created using **HTML, CSS, and JavaScript**, ensuring a responsive and interactive user interface. The **backend** handled user requests, text processing, and speech generation using Django, with **SQLite** serving as the database for storing user and file details. All modules—**Text Extraction**, **Summarization**, **Text-to-Speech**, **Playback**, and **Security**—were integrated and tested to ensure proper functionality. After successful testing, the system was deployed on a **Django development server**, allowing users to access it through a web browser.

Finally, the implemented system was evaluated for performance and usability. The *SPEAKO* application proved to be reliable, secure, and user-friendly, successfully meeting all functional and technical requirements.

9.2 Problem Statement

In today’s digital era, individuals frequently encounter large volumes of written content such as articles, research papers, reports, and study materials. Reading through lengthy documents can be time-consuming and challenging, especially for users with visual impairments or reading difficulties. Existing text-to-speech applications often have limitations—they may not support multiple file formats like PDF or DOC, lack summarization features, or provide robotic and unnatural voice outputs. Many such tools also require paid subscriptions for full functionality.

Additionally, Hence, there is a need for a **web-based application** that can efficiently convert text and documents into **natural-sounding speech**, while also offering **automatic summarization** and **customization options** for voice, pitch, speed, and language. The *SPEAKO* project aims to address these limitations by developing an intelligent, user-friendly, and accessible solution for seamless content consumption.

9.3 Problem Definition

The main objective of the *SPEAKO* project is to develop a **web-based Text-to-Speech (TTS) application** that allows users to convert text, PDF, and DOC files into **natural-sounding speech**. The system aims to overcome the shortcomings of existing TTS tools by integrating **text summarization**, **multi-language support**, and **customizable voice features** within a single platform.

The application should allow users to upload or input text, summarize lengthy content using **Natural Language Processing (NLP)**, and convert the summarized text into audio using **Google Text-to-Speech (gTTS)**. It must provide options to adjust **voice type, pitch, speed, and language**, ensuring a personalized listening experience.

Additionally, the system should feature a **user-friendly interface**, **secure data handling**, and **cross-platform compatibility**. The goal is to make content consumption faster, more accessible, and convenient for students, professionals, and users with reading challenges

.

CONCLUSION

10 . CONCLUSION

The *SPEAKO* project successfully achieves its goal of creating an intelligent and user-friendly **Text-to-Speech (TTS) web application**. By integrating **text extraction**, **automatic summarization**, and **speech generation**, the system provides a complete and accessible solution for converting text and documents into natural-sounding audio.

Developed using **Python**, **Django**, **gTTS**, and **spaCy**, *SPEAKO* ensures efficiency, security, and scalability. Its customizable features for voice, pitch, speed, and language make it highly adaptable to user preferences. The application not only enhances accessibility for users with reading difficulties but also improves productivity for students and professionals.

Overall, *SPEAKO* demonstrates the effective use of modern web technologies and AI tools to simplify content consumption, promote inclusivity, and provide a seamless digital experience.

FUTURE ENHANCEMENTS

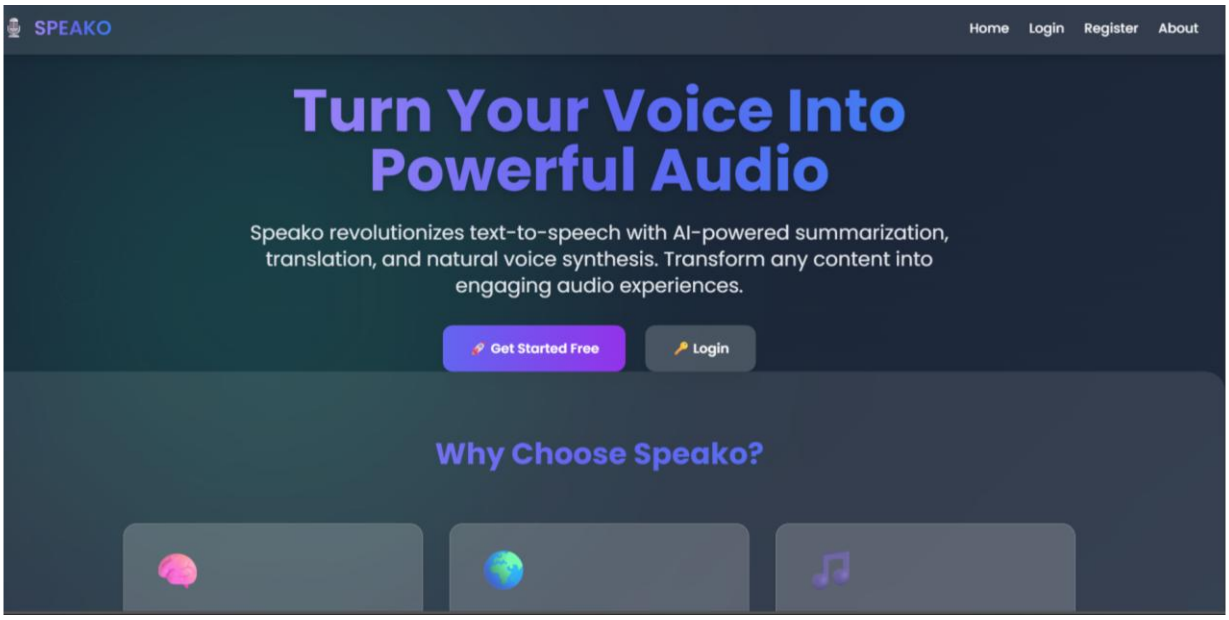
In the future, *SPEAKO* can be enhanced with several advanced features to improve its performance, accessibility, and user experience. One of the major enhancements could include the integration of advanced voice synthesis technologies to provide more natural and expressive speech, including emotion control and tone variation. An **offline mode** can also be introduced to allow users to convert text into speech without the need for an internet connection. Additionally, **cloud storage integration** could enable users to save and retrieve their converted audio files directly from cloud platforms such as Google Drive or Dropbox. The summarization module can be upgraded with more advanced **AI and machine learning models** for better accuracy and contextual understanding. Furthermore, developing a **mobile application** version of *SPEAKO* for Android and iOS devices would enhance accessibility and allow users to enjoy its features on the go. These enhancements would make *SPEAKO* a more versatile, efficient, and user-friendly platform for a broader range of users.

APPENDICES

11. APPENDICES

APPENDICES A:Sample Screens

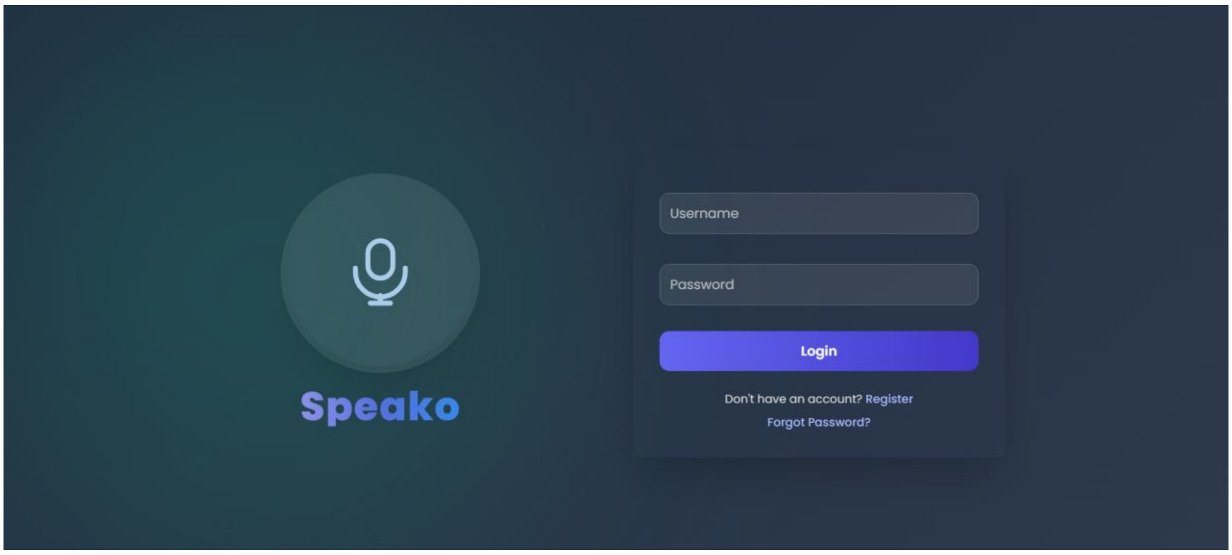
Home Page



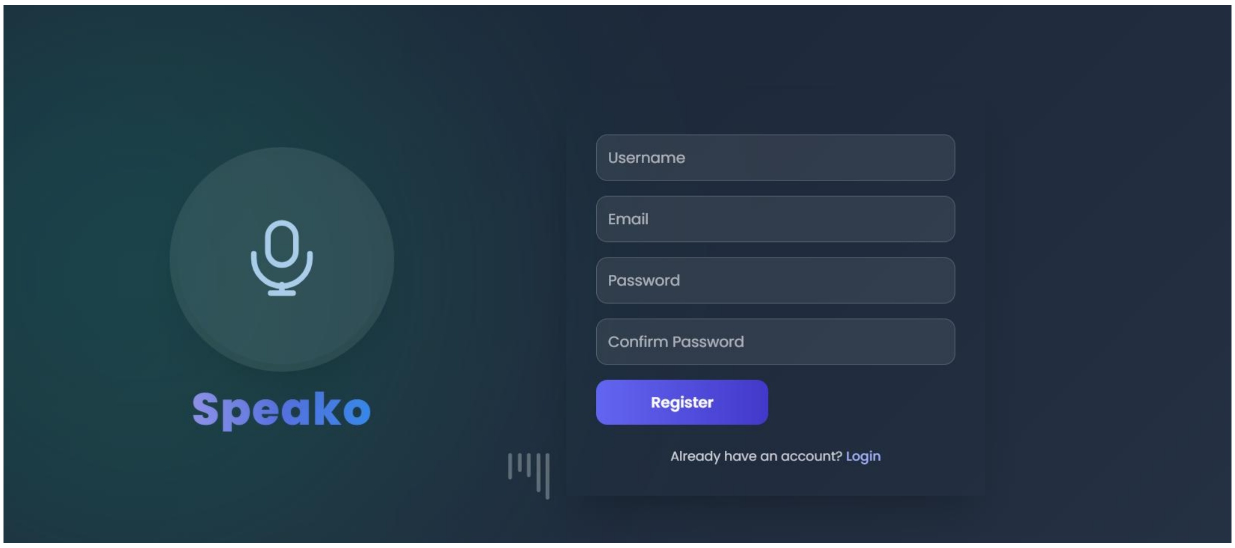
Turn Your Voice Into Powerful Audio

Speako revolutionizes text-to-speech with AI-powered summarization, translation, and natural voice synthesis. Transform any content into engaging audio experiences.

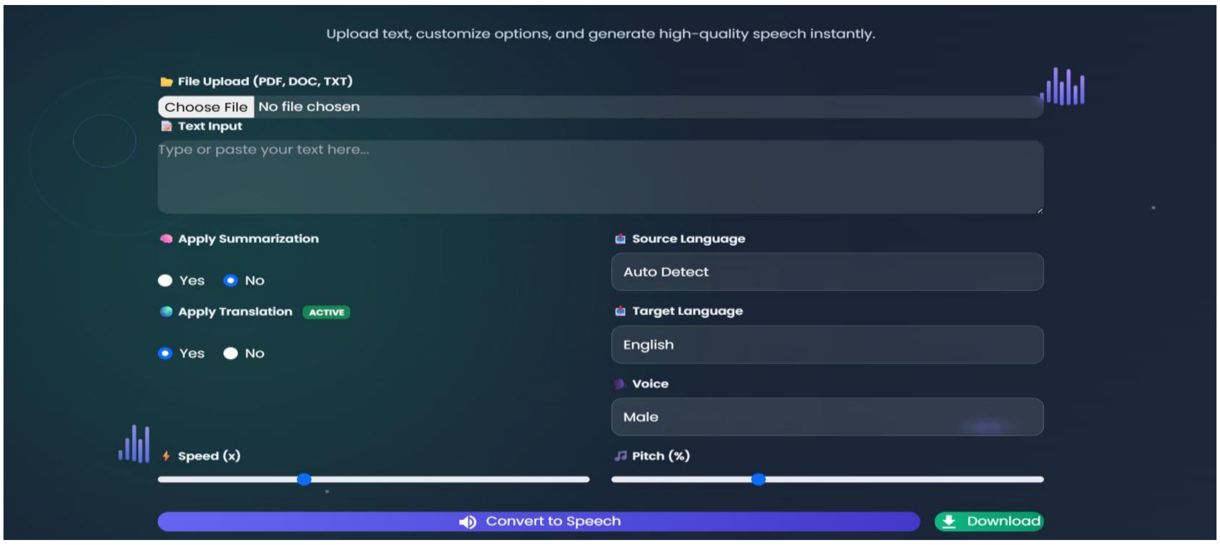
Login Page



User Registration



User Homepage


Upload text, customize options, and generate high-quality speech instantly.

File Upload (PDF, DOC, TXT)

Choose File No file chosen

Text Input

Type or paste your text here...

Apply Summarization: Yes [✓ No]

Apply Translation ACTIVE: [✓ Yes] No

Source Language: Auto Detect

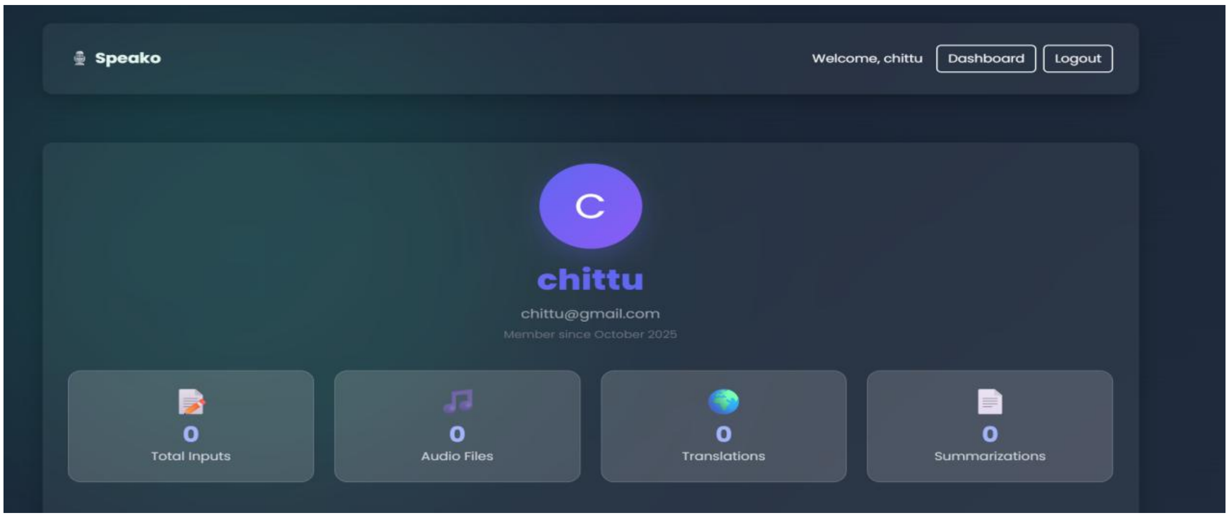
Target Language: English

Voice: Male

Speed (x)

Pitch (%)

User Profile



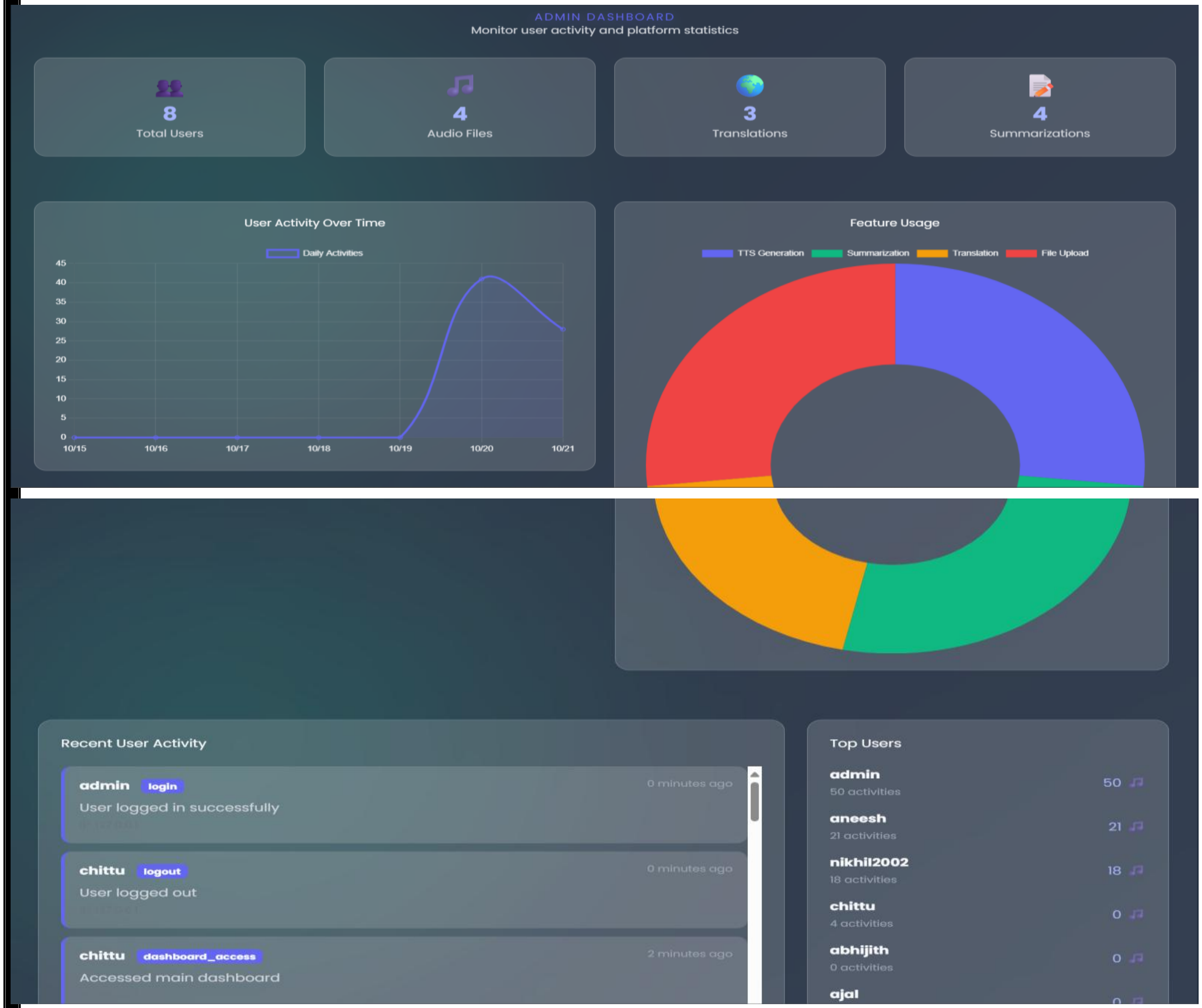
chittu

chittu@gmail.com

Member since October 2025

0	0	0	0
Total Inputs	Audio Files	Translations	Summarizations

Admin Dashboard



Recent User Activity

admin login — 0 minutes ago
User logged in successfully

chittu logout — 0 minutes ago
User logged out

chittu dashboard_access — 2 minutes ago
Accessed main dashboard

Top Users

admin 50 activities	50
aneesh 21 activities	21
nikhil2002 18 activities	18
chittu 4 activities	0
abhijith 0 activities	0
ajal	0

APPENDICES B:Sample Code

```
from import os
import io
import secrets
import hashlib
from datetime import datetime, timedelta
# Using 'from django.conf import settings' is
cleaner than 'import os' for path joining in
Django
from django.conf import settings
from django.shortcuts import render, redirect
from django.http import JsonResponse
from django.contrib.auth import authenticate,
login, logout
from django.contrib.auth.models import User
from django.views.decorators.http import
require_POST
from django.contrib.auth.decorators import
login_required
from django.core.mail import send_mail
from django.template.loader import
render_to_string
from django.utils import timezone

# --- CORE LIBRARIES ---
# Use a try-except block here to prevent the
entire server from crashing if libraries are
missing
try:
    import spacy
    from gtts import gTTS
    import PyPDF2
    from docx import Document as
DocxDocument
    from googletrans import Translator
    from transformers import pipeline

    # Load the NLP model once globally for
efficiency
    nlp = spacy.load("en_core_web_sm"
SPACY_LOADED = True

    # Initialize translator
    translator = Translator()
    TRANSLATION_LOADED = True

    # Initialize summarization pipeline
    try:
        summarizer = pipeline("summarization", model="facebook/bart-large-cnn", device=-
1)  # CPU
        ADVANCED_SUMMARIZATION_LOADED = True
    except:
        ADVANCED_SUMMARIZATION_LOADED = False

except (ImportError, OSError) as e:
    # This block executes if libraries are missing or model download failed
    print(f"WARNING: Some features may be unavailable. Error: {e}")
    SPACY_LOADED = False
    TRANSLATION_LOADED = False
    ADVANCED_SUMMARIZATION_LOADED = False

    # Still require other non-spacy libraries for TTS and file parsing
    try:
        from gtts import gTTS
        import PyPDF2
        from docx import Document as DocxDocument
    except ImportError as tts_e:
        # Critical error if TTS libraries are also missing
        print(f"CRITICAL ERROR: Core TTS/File library missing: {tts_e}")
        raise # Re-raise the error if essential libraries are missing

# Import your local models
from .models import Input, Summarization, Translation, TTS, UserActivity, PlatformStats,
PasswordResetToken

# --------------------------------------------------
# A. AUTHENTICATION AND NAVIGATION VIEWS
# --------------------------------------------------

# Home (Landing Page)
def home(request):
    """Renders the main home/landing page."""

 # Check if the user is already logged in, and redirect them to the appropriate dashboard
    # Only redirect if they're not explicitly navigating to home (e.g., from login/register
pages)
    if request.user.is_authenticated and not request.GET.get('stay', False):
        if request.user.is_staff:
            return redirect('admin_dashboard')
 else:
            return
redirect('dashboard')
    # Assuming home.html is
in core/templates/home.html
or
speako_project/templates/ho
me.html
    return render(request,
'home.html')

# Register view
def register(request):
    """Handles user
registration."""
    if request.method ==
'POST':
        username =
request.POST.get('username
')
        email =
request.POST.get('email')
        password1 =
request.POST.get('password
1')
        password2 =
request.POST.get('password
2')

        # Basic validation
        if
User.objects.filter(username
=username).exists():
```

User registration

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Register | Speako</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- Tailwind CSS CDN -->
  <script src="https://cdn.tailwindcss.com"></script>

  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700;800&
display=swap" rel="stylesheet">
  <link rel="icon" href="{% static 'favicon.svg' %}" type="image/svg+xml">
  <link rel="stylesheet" href="{% static 'css/style.css' %}">

  <style>
    body {
      font-family: 'Poppins', sans-serif;
      margin: 0;
      min-height: 100vh;
      color: #f9fafb;
      /* Dynamic Gradient Background - matching dashboard */
      background: linear-gradient(135deg, #0f172a, #1e3a8a, #4338ca);
      background-size: 300% 300%;
      animation: gradientShift 12s ease infinite;
      position: relative;
      padding: 40px;
    }

 /* Entry animations */
    .fade-in-up {
      opacity: 0;
      transform: translateY(16px);
      animation: fadeUp 700ms ease-out forwards;
    }
    .fade-in-left {
      opacity: 0;
      transform: translateX(16px);
    animation: fadeLeft 700ms ease-out forwards;
      }
      .anim-delay-100 { animation-delay: 100ms; }
      .anim-delay-200 { animation-delay: 200ms; }

      @keyframes fadeUp {
        to { opacity: 1; transform: translateY(0); }
      }
      @keyframes fadeLeft {
        to { opacity: 1; transform: translateX(0); }
      }
      @keyframes gradientShift {
        0% { background-position: 0% 50%; }
        50% { background-position: 100% 50%; }
        100% { background-position: 0% 50%; }
      }

      /* Blurred glowing circles for aesthetic effect */
      body::before, body::after {
        content: '';
        position: absolute;

border-radius: 50%;
      filter: blur(160px);
      opacity: 0.35;
      z-index: 0;
    }
    body::before {
      width: 600px;
      height: 600px;
      background: #6366f1; /* Indigo */
      top: -200px;
      left: -150px;
    }
    body::after {
      width: 700px;
      height: 700px;
      background: #10b981; /* Emerald */
      bottom: -250px;
      right: -200px;
    }
```

```
register-container {
      position: relative;
      z-index: 1;
    .form-input {
      width: 100%;
      padding: 12px;
      border-radius: 12px;
      border: none;
      margin-bottom: 20px;
      background: rgba(255, 255, 255, 0.1);
      color: #f9fafb;
      backdrop-filter: blur(6px);
    }
    .form-input::placeholder {
      color: #e5e7eb;
      opacity: 0.7;
    }
    .btn-gradient {
      background: linear-gradient(90deg, #6366f1, #4338ca);
      color: #fff;
      padding: 12px 18px;
      border-radius: 12px;
      border: none;
      font-weight: 600;
      cursor: pointer;
      transition: all 0.3s ease;
      width: 100%;
    }
 .btn-gradient:hover {
      background: linear-gradient(90deg, #4338ca, #312e81);
      transform: translateY(-2px);
      box-shadow: 0 5px 15px rgba(99, 102, 241, 0.4);
    }
.error-message {
      background: rgba(239, 68, 68, 0.2);
      border: 1px solid rgba(239, 68, 68, 0.3);
      color: #fca5a5;
      padding: 12px;
      border-radius: 8px;
      margin-bottom: 20px;
      text-align: center;}
```

APPENDICES C: Hardware and Software Requirements

- Hardware Requirements

Processor	:	Intel Core i3
Random Access Memory	:	4GB or above
Hard Disk/SSD	:	240GB
Monitor	:	Color Monitor
Keyboard	:	Standard
Video	:	800X600 256 colors

- Software Requirements

Operating System	:	Windows 11 or Higher
Front End	:	HTML,CSS,Javascript
Environment	:	Django
Database	:	SQLite
Documentation Tool	:	MS Word

APPENDICES D: Bibliography

Book References:

- Taming Python By Programming by Dr. Jeeva Jose

Publications:

1. **Dauzon, S., Bendoraitis, A., & Ravindran, A.** (2016). *Django: web development with Python*. Packt Publishing Ltd . Publisher Site | Google Scholar
2. **Shaw, B., Badhwar, S., Bird, A., KS, B. C., & Guest, C.** (2021). *Web Development with Django: Learn to build modern web applications with a Python-based framework*. Packt Publishing Ltd. Publisher Site | Google Scholar

Website References:

- Retrieved from www.stackoverflow.com
- Retrieved from www.w3schools.com
- Retrieved from www.github.com
- Retrieved from www.python.org
- Retrieved from www.tutorialspoint.com
- Retrieved from www.javatpoint.com