

**TEXT TO VOICE CONVERTOR**

**A PROJECT REPORT**

*Submitted by*

**ARUN SANJEEV M S (927623BCS011)**

***in partial fulfillment for the completion of the course***

**CGB1121- PYTHON PROGRAMMING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

**MAY 2024**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

**MAY 2024**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“Text to Voice Convertor”** is the bonafide work of **ARUN SANJEEV M S (927623BCS011)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a course was conferred on an earlier occasion on this or any other candidate.

|  |  |
| --- | --- |
| Signature | Signature |
| **Dr.M.MURUGESAN, M.E., Ph.D.,** | **Dr. K.CHITIRAKALA, M.Sc., M.Phil.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science | Department of Freshman Engineering, |
| and Engineering, |  |
| M. Kumarasamy College of Engineering, | M. Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |



Submitted for the viva-voce examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARATION**

I declare that the project report on **“Text to Voice Convertor”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1121- PYTHON PROGRAMMING**.

|  |
| --- |
| **Signature** |
| SAA |
| ARUN SANJEEV M S |

Place: Thalavapalayam

Date:

**ACKNOWLEDGEMENT**

It is with great pride that I express our gratitude and in-debt to our institution “**M.Kumarasamy College of Engineering(Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

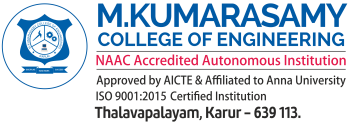
I would like to thank **Dr.B.S.MURUGAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks **Dr. M. MURUGESAN, M.E., Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project guide **Dr. M. MURUGESAN, M.E., Ph.D.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** Assistant Professor, Department of Computer Science and Engineering,for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**VISION OF THE INSTITUTION**

To emerge as a leader among the top institutions in the field of technical education

# MISSION OF THE INSTITUTION

* Produce smart technocrats with empirical knowledge who can surmount the global challenges.
* Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
* Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

**VISION OF THE DEPARTMENT**

To achieve education and research excellence in Computer Science and Engineering.

# MISSION OF THE DEPARTMENT

* To excel in academic through effective teaching learning techniques
* To promote research in the area of computer science and engineering with the focus on innovation
* To transform students into technically competent professionals with societal and ethical

Responsibilities.

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO 1:** Graduates will have successful career in software industries and R&D divisions through continuous learning.

**PEO 2:** Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

**PEO 3:** Graduates will excel in their profession by being ethically and socially responsible.

r

**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in

diverse teams, and in multidisciplinary settings.

1. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations.
2. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

1. **PSO1: Professional Skills:** Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.
2. **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

**ABSTRACT**

This project presents a Python-based text-to-voice converter utilizing the Google Text-to-Speech (gTTS) library for high-quality speech synthesis and Tkinter for an intuitive graphical user interface (GUI). The application allows users to input text, select language and accent options, and convert the text into audible speech with a simple click. The Tkinter interface enhances user interaction by providing a clean, user-friendly design with text input fields, customization options for speech parameters, and playback controls. The generated speech can be saved as an audio file for future use. This project demonstrates the seamless integration of natural language processing and GUI development, providing a versatile tool for a variety of applications including accessibility for the visually impaired, language learning aids, multimedia content creation, and automated reading systems. By leveraging the powerful capabilities of gTTS and the simplicity of Tkinter, this text-to-voice converter offers an effective and user-friendly solution for converting written text into spoken words.The application not only highlights the technical integration of gTTS and Tkinter but also emphasizes user experience through its responsive and interactive design. Users can easily customize the speech output by selecting different languages and accents, making the tool adaptable to diverse linguistic needs. The project also includes features such as real-time text-to-speech conversion and the ability to save audio files in various formats, enhancing its practicality. This project serves as an educational example for developers interested in natural language processing and GUI development, showcasing how Python libraries can be combined to create accessible and functional software. Moreover, it addresses real-world needs by providing a solution that can assist individuals with reading difficulties, support language learning, and enable the creation of audio content for various purposes.

# LIST OF ABBREVIATIONS

**ABBREVIATIONS**

* **gTTS:** Google Text-to-Speech
* **Tkinter:** Standard GUI (Graphical User Interface) toolkit for Python
* **Entry:** A widget in Tkinter used to accept user input
* **Button:** A widget in Tkinter used to add buttons to the GUI
* **Label:** A widget in Tkinter used to display text or images
* **Message:** A widget in Tkinter used to display multiline text
* **AudioSegment:** A module from the pydub library used for audio processing

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | viii |
|  | **LIST OF FIGURES** | ix |
|  | **LIST OF ABBREVIATIONS** | x |
| **1** | **INTRODUCTION** |  |
|  | * 1. Introduction to Python | 1 |
|  | 1.1.1. Overview | 1 |
|  | 1.1.2. Programming Paradigms | 1 |
|  | 1.1.3. Standard Library | 1 |
|  | 1.1.4. Third-Party Libraries and Frameworks | 1 |
|  | 1.1.5. Versions of Python | 1 |
|  | 1.1.6. Python Tools | 2 |
|  | 1.1.7. Versatility and Adoption | 2 |
| **2** | **PROJECT DESCRIPTION** |  |
|  | 2.1. Project Introduction | 3 |
|  | 2.1. Project Objective | 3 |
|  | 2.3. Problem Statement | 3 |
|  | 2.4. Libraries Used | 3 |
| **3** | **SYSTEM ANALYSIS** |  |
|  | 3.1. Existing System | 5 |
|  | 3.1.1. Disadvantages | 5 |
|  | 3.2 Proposed System | 6 |
|  | 3.2.1. Advantage | 6 |
| **4** | **SYSTEM DESIGN & MODULES** |  |
|  | 4.1. Block Diagram | 7 |
|  | 4.2. Module Description | 7 |
|  | 4.2.1. Determining Question Index | 7 |
|  | 4.2.2. Scoring Mechanism | 7 |
|  |
|  |  |  |
| **5** | **CONCLUSION & FUTURE ENHANCEMENT** |  |
|  | 5.1. Conclusion | 9 |
|  | 5.2. Future Enhancement | 9 |
| **6** | **APPENDICES** |  |
|  | Appendix A-Source code | 11 |
|  | Appendix B -Screen shots | 12 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO.** |
| 4.1 | BLOCK DIAGRAM | 18 |
|  |  |  |

**CHAPTER 1**

# INTRODUCTION

**1.1 INTRODUCTION TO PYTHON**

**1.1.1. OVERVIEW**

Python is a widely-used, high-level programming language renowned for its readability and simplicity, making it an ideal choice for both novice and seasoned programmers. Created by Guido van Rossum and released in 1991, Python's core philosophy emphasizes code readability and straightforward syntax, allowing developers to write clear and concise code more efficiently compared to other languages like C++ or Java.

**1.1.2. PROGRAMMING PARADIGMS**

Python supports various programming paradigms, including procedural, object-oriented, and functional programming. This flexibility, combined with a dynamic type system and automatic memory management, facilitates the development of a wide range of applications, from simple scripts to complex software systems.

**1.1.3. STANDARD LIBRARY**

The language's comprehensive standard library, often referred to as "batteries-included," provides built-in modules and functions for handling many programming tasks, such as file I/O, system calls, and even web services. This extensive library helps streamline the development process by offering ready-to-use solutions for common programming challenges.

**1.1.4. THIRD-PARTY LIBRARIES AND FRAMEWORKS**

One of Python's significant strengths is its extensive ecosystem of third-party libraries and frameworks. Popular libraries such as NumPy and Pandas enable efficient data manipulation and analysis, while frameworks like Django and Flask streamline web development.

**1.1.5. VERSIONS OF PYTHON**

Python has undergone significant evolution since its inception, with two major versions in use today:

Python 2: Released in 2000, Python 2.x series was a major milestone and widely used for many years. However, it reached its end of life on January 1, 2020, and is no longer maintained.

Python 3: Introduced in 2008, Python 3.x series brought substantial improvements and changes to the language, such as better Unicode support, a more consistent syntax, and enhanced standard libraries. Python 3 is the recommended version for all new projects.

**1.1.6. PYTHON TOOLS**

Python's ecosystem includes numerous tools that enhance productivity and development experience:

* **IDEs and Code Editors:** Popular options include PyCharm, VS Code, and Jupyter Notebook, which offer features like syntax highlighting, code completion, and debugging.
* **Package Management:** Tools like pip and conda facilitate the installation and management of Python libraries and dependencies.
* **Virtual Environments:** virtualenv and venv allow developers to create isolated environments for different projects, ensuring dependency conflicts are avoided.
* **Testing Frameworks:** unittest, pytest, and nose are commonly used for writing and running tests to ensure code reliability and correctness.
* **Build Tools:** setuptools and wheel help in packaging Python projects, making them easy to distribute and install.
* **Documentation Generators:** Tools like Sphinx are used to create comprehensive documentation for Python projects.
* **Linters and Formatters:** pylint, flake8, and black help maintain code quality and consistency by enforcing coding standards and formatting.

**1.1.7. VERSATILITY AND ADOPTION**

Python's simplicity and versatility have led to its widespread adoption in various fields, including web development, data science, artificial intelligence, automation, and scientific computing. Its active community continually contributes to a rich repository of resources, tutorials, and documentation, making it easier for developers to learn and apply Python effectively.

**CHAPTER 2**

## PROJECT DESCRIPTION

## 2.1. PROJECT INTRODUCTION

## This project is a Python-based text-to-voice converter utilizing the Google Text-to-Speech (gTTS) library for speech synthesis and Tkinter for the graphical user interface. The application allows users to input text, which is then converted into audible speech. The intuitive Tkinter GUI facilitates easy interaction, providing text input fields, voice customization options, and playback controls. The generated speech can be saved as an audio file for future use.

## 2.2. PROJECT OBJECTIVE

## The objective of this project is to provide a versatile and user-friendly tool for converting written text into spoken words. It aims to enhance accessibility for the visually impaired, support language learning, and enable the creation of audio content for various applications.

## 2.3. PROBLEM STATEMENT

## The problem addressed by this project is the need for an efficient and straightforward way to convert text into speech. This involves designing a graphical interface where users can input text, customize speech parameters, and listen to the generated audio. The program should handle text-to-speech conversion accurately, provide playback options, and allow users to save the audio output.

## 2.4. LIBRARIES USED

1. **gTTS:** Google Text-to-Speech library for converting text into speech.
2. **Tkinter:** Python's standard GUI toolkit used for creating desktop applications. It provides tools for building interactive UIs with buttons, labels, entry fields, etc.
3. **Pytxt3:** is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.

**CHAPTER 3**

## SYSTEM ANALYSIS

## 3.1. EXISTING SYSTEM

## Existing systems for text-to-speech conversion often rely on complex setups or paid services. These systems may not offer a straightforward user interface or the flexibility to customize speech parameters easily. Additionally, they might not integrate well with other applications or provide options to save the generated speech for later use.

## 3.1.1. DISADVANTAGES

1. **Complex Setup:** Existing systems may require complex configurations or installations.
2. **Limited Customization:** Some systems offer limited options for customizing speech output, such as selecting different voices or adjusting speed.
3. **Cost:** Many advanced text-to-speech services are paid, which can be a barrier for individual users or small projects.

## ****Integration Issues:**** Existing solutions may not integrate seamlessly with other applications or Platforms.

## PROPOSED SYSTEM

## The proposed system aims to address these issues by providing a simple, user-friendly text-to-speech converter using Python. It leverages the gTTS library for high-quality speech synthesis and Tkinter for an intuitive graphical interface. This system allows users to input text, customize speech parameters, listen to the generated speech, and save the output as an audio file.

## 3.2.1. ADVANTAGES

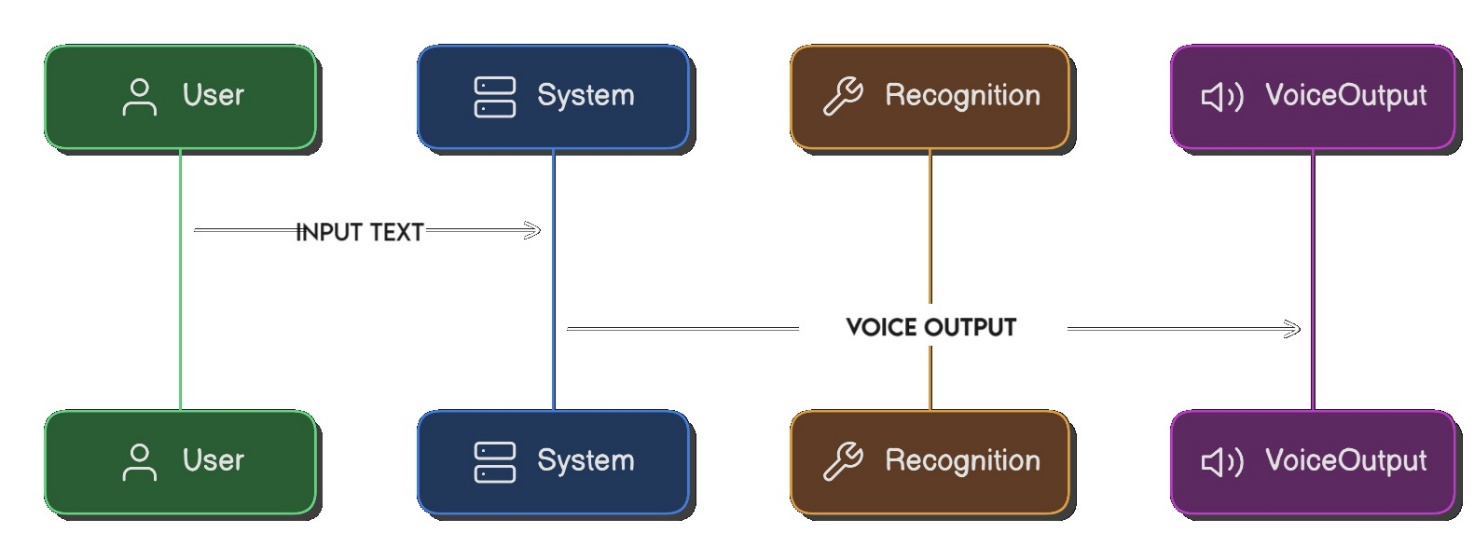
1. **Simplicity:** The application is easy to set up and use, making it accessible for all users.
2. **Customization:** Users can choose different languages and accents, adjust speech speed, and customize other parameters.
3. **Cost-Effective:** The application uses free, open-source libraries, reducing costs.
4. **Integration:** The generated audio files can be easily integrated with other applications or platforms.

**CHAPTER 4**

## SYSTEM DESIGN & MODULES

## 4.1. BLOCK DIAGRAM

## 



## FIG:4.1

## 4.2. MODULE DESCRIPTION

##### 4.2.1 Text Input Module

This module handles user input through a Tkinter Entry widget. Users can type the text they wish to convert into speech.

##### 4.2.2 Speech Synthesis Module

This module uses the gTTS library to convert the input text into speech. It allows users to select the language, adjust the speech rate, and customize other parameters.

##### 4.2.3 Playback Module

This module utilizes generated speech. It provides buttons for playing, pausing, stopping the audio and saving the audio.

##### 4.2.5 User Interface Module

This module, built with Tkinter, integrates all other modules into a cohesive and interactive graphical user interface. It includes buttons, labels, and other widgets for user interaction.

## 

**CHAPTER 5**

## CONCLUSION & FUTURE ENHANCEMENT

## 5.1. CONCLUSION

## The developed text-to-voice converter successfully integrates the gTTS library for speech synthesis and Tkinter for a user-friendly interface. The application provides a practical tool for converting text into speech, supporting accessibility, language learning, and multimedia content creation. It demonstrates the effective combination of natural language processing and GUI development.

## 5.2. FUTURE ENHANCEMENT

 I**mproved UI Design:** Enhance the visual appeal and usability of the application by incorporating modern design elements and themes.

 **Additional Languages and Voices:** Expand the range of supported languages and add more voice options for greater flexibility.

 **Real-Time Text-to-Speech:** Implement real-time text-to-speech conversion for instant feedback.

 **Integration with Other Services:** Enable integration with other services, such as cloud storage or translation APIs.

 **Mobile Application:** Develop a mobile version of the application

 **Error Handling:** Implement robust error handling mechanisms to gracefully handle unexpected situations, such as file not found errors or invalid user inputs.

 **Accessibility Features:** Ensure the application is accessible to users with disabilities by incorporating features such as keyboard navigation, screen reader compatibility, and high contrast modes.

## APPENDICES

## APPENDIX A-SOURCE CODE

## from gtts import gTTS

## import os

## import tkinter as tk

## from tkinter import filedialog, messagebox

## def text\_to\_speech():

## mytext = text\_entry.get("1.0", "end-1c")

## language = languages[language\_var.get()]

## slow = speed\_var.get()

## 

## if not mytext.strip():

## messagebox.showwarning("Input Error", "Please enter some text.")

## return

## try:

## myobj = gTTS(text=mytext, lang=language, slow=slow)

## file\_path = filedialog.asksaveasfilename(defaultextension=".mp3", filetypes=[("MP3 files", "\*.mp3")])

## if file\_path:

## myobj.save(file\_path)

## os.system(f"start {file\_path}")

## messagebox.showinfo("Success", "Audio file saved and played successfully!")

## except Exception as e:

## messagebox.showerror("Error", f"An error occurred: {str(e)}")

## def play\_audio():

## mytext = text\_entry.get("1.0", "end-1c")

## language = languages[language\_var.get()]

## slow = speed\_var.get()

## 

## if not mytext.strip():

## messagebox.showwarning("Input Error", "Please enter some text.")

## return

## try:

## myobj = gTTS(text=mytext, lang=language, slow=slow)

## file\_path = "temp\_audio.mp3"

## myobj.save(file\_path)

## os.system(f"start {file\_path}")

## except Exception as e:

## messagebox.showerror("Error", f"An error occurred: {str(e)}")

## def set\_defaults():

## text\_entry.delete("1.0", "end")

## text\_entry.insert("end", "Enter text here...")

## language\_var.set("English")

## speed\_var.set(False)

## # Mapping full language names to gTTS language codes

## languages = {

## "English": "en",

## "Spanish": "es",

## "French": "fr",

## "German": "de",

## "Italian": "it"

## }

## # Setting up the main window

## root = tk.Tk()

## root.title("Text to Voice Converter")

## root.geometry("1526x850")

## root.configure(bg="#2c3e50")

## # Font settings

## font\_name = "Lemon Milk"

## font\_bold = "Lemon Milk Bold"

## times = "Montserrat"

## # Top Labels

## top\_label1 = tk.Label(root, text="M KUMARASAMY COLLEGE OF ENGINEERING - KARUR", font=(font\_bold, 18), bg="#2c3e50", fg="white")

## top\_label1.pack(pady=5)

## top\_label2 = tk.Label(root, text="PYTHON END SEMESTER PROJECT", font=(font\_bold, 17), bg="#2c3e50", fg="white")

## top\_label2.pack(pady=5)

## top\_label3 = tk.Label(root, text="M S ARUN SANJEEV", font=(font\_name, 16), bg="#2c3e50", fg="white")

## top\_label3.pack(pady=5)

## # Registration Number Label

## reg\_label = tk.Label(root, text="Reg No: 927623BCS011", font=(font\_name, 16), bg="#2c3e50", fg="white")

## reg\_label.pack(pady=5)

## # Title Label

## title\_label = tk.Label(root, text="Text to Voice Converter", font=(font\_bold, 20), bg="#2980b9", fg="white")

## title\_label.pack(pady=10, fill=tk.X)

## # Text Entry Widget

## text\_entry = tk.Text(root, wrap="word", font=(times, 12), height=5, width=150, fg="grey")

## text\_entry.pack(pady=10, padx=10)

## text\_entry.insert("end", "Enter text here...")

## # Language Dropdown

## language\_var = tk.StringVar(root)

## language\_var.set("English")

## language\_label = tk.Label(root, text="Select Language:", font=(font\_name, 12), bg="#2c3e50", fg="white")

## language\_label.pack(pady=5)

## language\_menu = tk.OptionMenu(root, language\_var, \*languages.keys())

## language\_menu.pack(pady=5)

## # Speed Checkbox

## speed\_var = tk.BooleanVar()

## speed\_check = tk.Checkbutton(root, text="Slow Speed", variable=speed\_var, font=(font\_name, 12), bg="#2c3e50", fg="white")

## speed\_check.pack(pady=5)

## # Play Button

## play\_button = tk.Button(root, text="Play", command=play\_audio, font=(font\_bold, 12), bg="#3498db", fg="white", width=10)

## play\_button.pack(pady=5)

## # Start Button

## convert\_button = tk.Button(root, text="SAVE", command=text\_to\_speech, font=(font\_bold, 14), bg="#27ae60", fg="white", width=15, height=2)

## convert\_button.pack(pady=20)

## # Reset Button

## reset\_button = tk.Button(root, text="Reset", command=set\_defaults, font=(font\_bold, 12), bg="#e74c3c", fg="white")

## reset\_button.pack(pady=5)

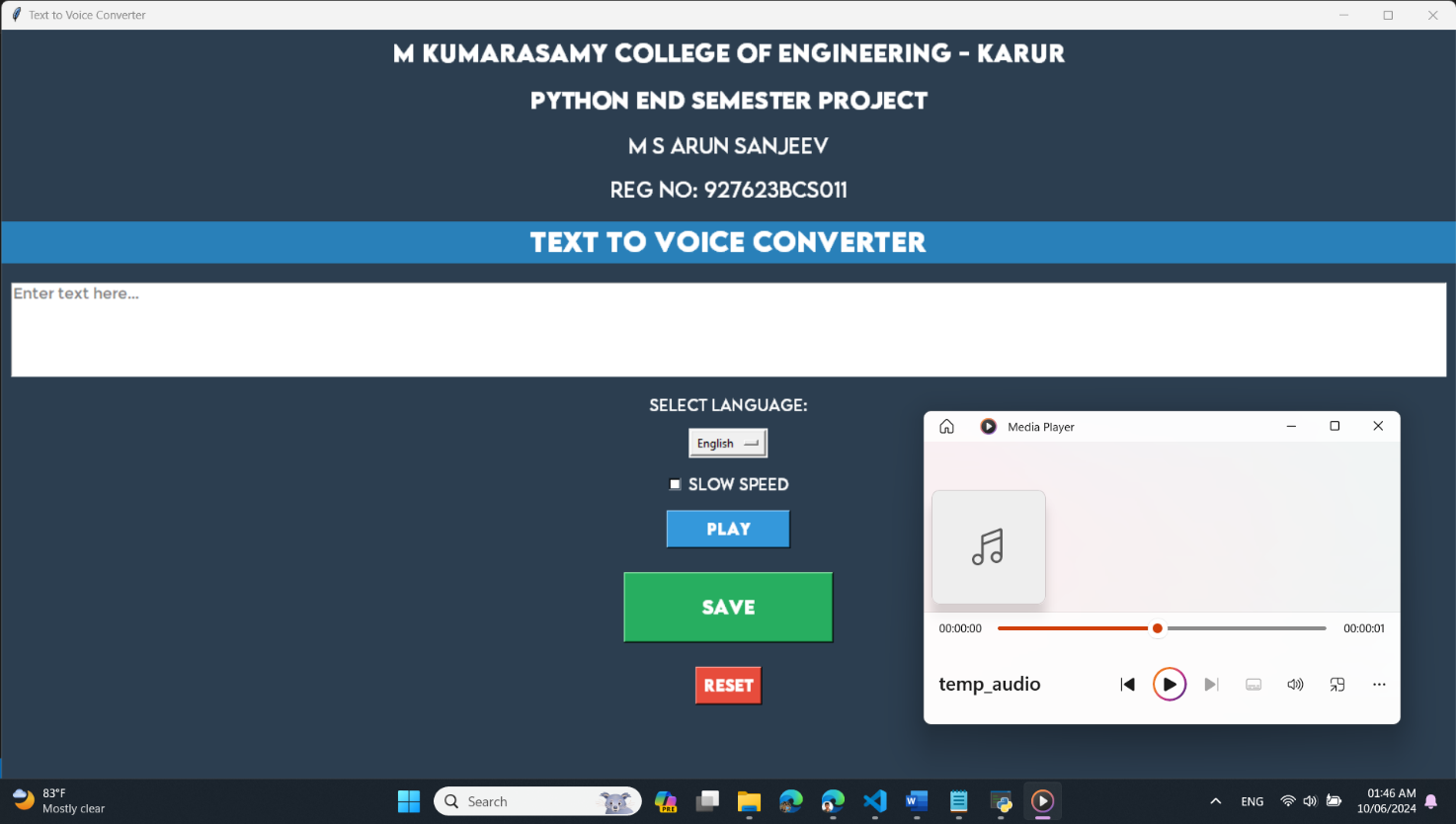
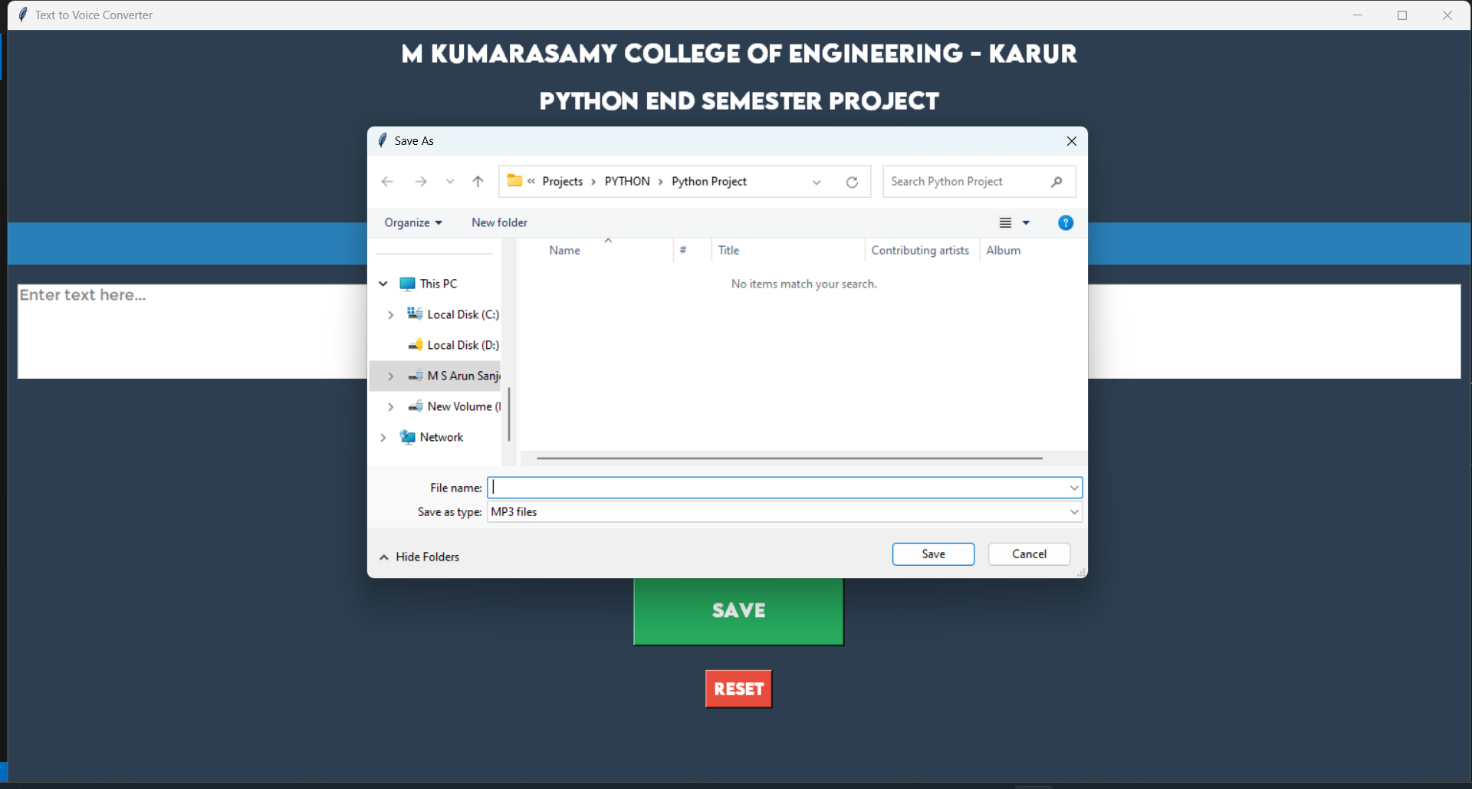
## # Run the application

## root.mainloop()

## APPENDIX B

## SCREEN SHOTS

## USER INTERFACE:

1. **SAVING MODULE:**
2. **PLAYING MODULE:**