VOCALVOCAB: AUDIO DICTIONARY & SYNONYMS

“SUMMER INTERNSHIP” – A PROJECT REPORT

Submitted by

PAVITHRA.S (22127036)

BACHELOR OF COMPUTER SCIENCE

WITH DATA ANALYTICS

SRI RAMAKRISHNA COLLEGE OF ARTS & SCIENCE

NAVA INDIA, COIMBATORE – 641 006

MAY – 2024

SRI RAMAKRISHNA COLLEGE OF ARTS & SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report **"VOCALVOCAB: AUDIO DICTIONARY & SYNONYMS"** is the bonafide work of **"PAVITHRA.S (22127036)"** who carried out the project work under my supervision.

SIGNATURE OF GUIDE

Dr. V. VIJAYAKUMAR MCA., M.Phil., Ph.D
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE WITH DATA ANALYTICS
SRI RAMAKRISHNA COLLEGE OF ARTS & SCIENCE
NAVA INDIA, COIMBATORE – 641 006

SIGNATURE OF HOD

Dr. V. VIJAYAKUMAR MCA., M.Phil., Ph.D
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE WITH DATA ANALYTICS
SRI RAMAKRISHNA COLLEGE OF ARTS & SCIENCE
NAVA INDIA, COIMBATORE – 641 006

TABLE OF CONTENTS

CHAPTER NO	TITLE	PG.NO
I	ABSTRACT	4
II	INTRODUCTION	5
	1.1 Natural Language Processing (NLP)	
	1.2 Components of NLP Used	
	1.3 User Interface (UI) Development	
	1.4 Components of UI Development Used	
III	OVERVIEW OF THE PROBLEM	8
	2.1 Problem Study	
	2.2 Challenges of the study	
	2.3 Hardware and Software Requirements	
IV	METHODOLOGY	10
V	RESULTS AND DISCUSSION	12
VI	CONCLUSION	16
VII	REFERENCES	17
VIII	APPENDIX	18

CHAPTER 1

ABSTRACT

This project implements a text-to-speech word meaning retrieval system using Python and several libraries including NLTK, gTTS, and IPython widgets. The system allows users to input words and retrieve their meanings from WordNet, while also providing audio pronunciations of the meanings. The code begins with the installation of necessary packages and the downloading of required NLTK data. It defines a function to remove HTML tags from text and a function to generate and queue audio using gTTS. An audio queue is managed to ensure sequential playback of the generated audio files.

The get_meaning function retrieves word meanings from WordNet, handles proper nouns, and displays the meanings using IPython display functions. If no meanings are found, an appropriate error message is displayed and spoken. Interactive widgets, including a text input field and a button, are created to facilitate user input. The widgets are connected to the get_meaning function, allowing users to submit words and receive immediate feedback. The user interface is built with IPython widgets, providing a clean and interactive experience.

The system's functionality is demonstrated through screenshots from Jupyter Lab, showing the code execution, word meaning retrieval, and audio playback. This project highlights the capabilities of Python in natural language processing and audio synthesis, making it a useful tool for educational purposes and accessibility enhancement. Future improvements could include support for multiple languages and additional features to enrich user interaction.

CHAPTER 2

INTRODUCTION

This chapter deals with the concepts of Natural Language Processing (NLP) components like NLTK (Natural Language ToolKit), WordNet, gTTS (Google Text-to-Speech), User Interface (UI) Development components like IPython Display, Widgets and some core concepts of python.

1.1 Natural Language Processing (NLP)

Natural language processing (NLP) is a subfield of computer science and Artificial Intelligence (AI). It is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language.
NLP is the core technology behind virtual assistants, such as the Oracle Digital Assistant (ODA), Siri, Cortana, or Alexa. [1,2]

NLP Benefits:

- NLP can be used to automate the gathering, processing and organization of information with less manual effort.
- Improving customer satisfaction and experience by identifying insights using sentiment analysis.
- The ability to analyze both structured and unstructured data, such as speech, text messages, and social media posts. [3]

Applications of NLP:

- Online chatbots
- Text Prediction and Autocorrect
- Sentiment Analysis
- Voice Assistants
- Language Translator [4]

1.2 Components of NLP Used

There are several NLP components that are mainly used in this project. They are:

a) **NLTK (Natural Language Toolkit):** The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education. [5]

b) **WordNet:** WordNet is a database of words in the English language. Unlike a dictionary that's organized alphabetically, WordNet is organized by concept and meaning. The database contains 155,327 words organized in 175,979 synsets for a total of 207,016 word-sense pairs; in compressed form, it is about 12 megabytes in size. It includes the lexical categories nouns, verbs, adjectives and adverbs but ignores prepositions, determiners and other function words. [6,7]

c) **gTTS (Google Text-to-Speech):** Google Text-to-Speech, often called gTTS, is a special tool created by Google. It turns written words into speech, making it easier for us to hear what's written on a screen. It is a Python library and CLI tool to interface with Google Translate's text-to-speech API. **[**8,9]

1.3 User Interface (UI) Development

User Interface Development is the development of websites, web applications, mobile applications and software development. User Interface plays a key role in the "software development life cycle". The goal of the user interface is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals. UI Development System can be considered as the middle groundwork by combining both design sensibilities and technicalities together. UI developers are skilled at making something look good and function correctly in a browser/device at the same time. UI development process includes writing code for images, animations, sliders, text fields, buttons, etc. [10,11]

1.4 Components of UI Development Used

Several UI Development components that are mainly used in this project are:

- **IPython Display**

  IPython means interactive Python. It is an interactive command-line terminal for Python. It will provide an IPython terminal and web-based (Notebook) platform for Python computing. It has more advanced features than the Python standard interpreter and will quickly execute a single line of Python code. Fernando Perez created it in the year 2001. [12]

- **IPython Widgets**

  Widget is a broad term that can refer to either any GUI (graphical user interface) element or a tiny application that can display information and/or interact with the user. A widget can be as rudimentary as a button, scroll bar, label, dialog box or check box; or it can be something slightly more sophisticated like a search box, tiny map, clock, visitor counter or unit converter. [13]

CHAPTER 3

OVERVIEW OF THE PROBLEM

This chapter tells about the study of the problem, the challenges that were faced during the study and the hardware and software requirements that are required to solve the study effectively.

2.1 Problem Study

The goal of this project is to create an interactive application that allows users to input a word and receive its meanings, both displayed on the screen and spoken aloud. This combines Natural Language Processing (NLP) techniques with user-friendly interface elements to provide an engaging and educational tool.

Understanding the meanings of words is fundamental to language learning and comprehension. Traditional methods of looking up words in a dictionary can be time-consuming and lack interactive elements that enhance learning. This project leverages modern technology to make this process more interactive and engaging, especially for users who benefit from auditory learning.

2.2 Challenges of the study

Several challenges were faced during the development of this script.

CHALLENGES:

- Estimating the duration of the generated audio to synchronize with text display.
- Managing the playback of multiple audio files in sequence required an effective queuing system. Ensuring that the next audio played only after the previous one finished, and handling the removal of played audio files, added complexity.
- Updating the output dynamically while maintaining the user interface's responsiveness was challenging. The code had to ensure that the output was cleared and updated efficiently without causing delays.

- Robust error handling was essential to manage cases where definitions could not be found or other issues occurred. Providing meaningful error messages and continuing to function smoothly in such cases required careful implementation.
- Handling different types of user input, including empty input or invalid words, required validation checks to provide appropriate feedback to the user and avoid processing errors.

2.3 Hardware and Software Requirements

HARDWARE:

The system used in this project is Windows 11. It has 11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz processor with 8.00GB RAM and 64-bit operating system, x64-based processor system type.

SOFTWARE:

The software used in this summer internship project is Jupyter Notebook. It is software that provides an interactive environment to develop and present computational content, combining code, visualizations, and narrative text in a single, shareable document. In this project it is used as the development environment for writing and testing code. The language used to code is Python.

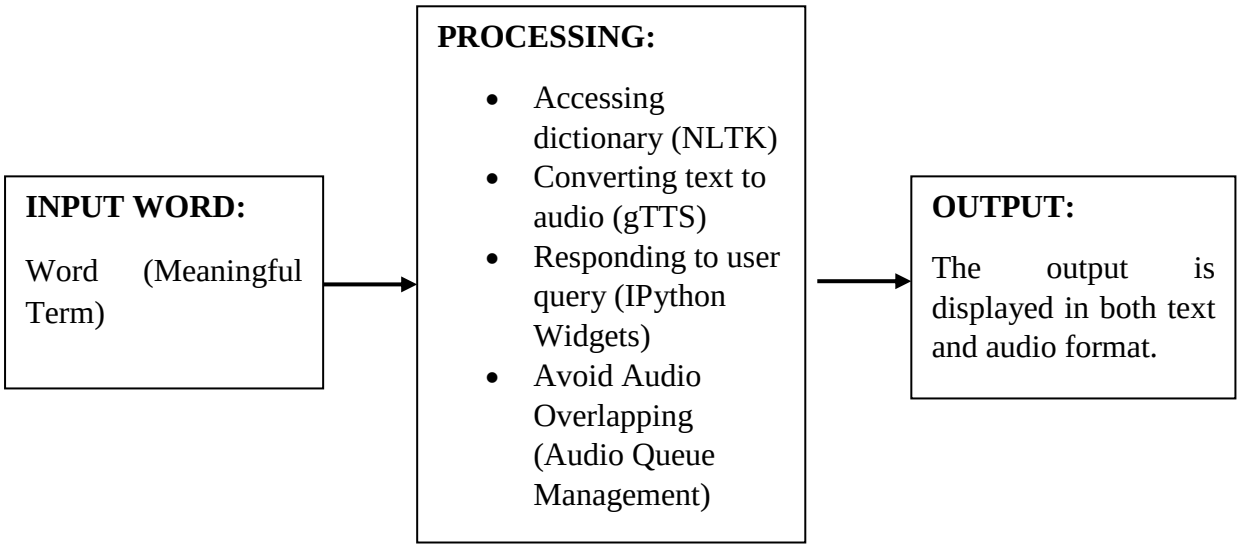
Along with this several libraries are used such as NLTK (Natural Language Toolkit) for word definitions retrieval, WordNet, gTTS (Google Text-to-Speech) for audio generation, IPython Display, IPython Widgets (ipywidgets) for interactive interface elements. Several modules such as Python os Module, Python time Module, Regular Expressions (re Module), Tkinter (tk Module), PyDictionary for synonym retrieval, pyttsx3 for managing text-to-speech functionalities are also used.

CHAPTER 4

METHODOLOGY

This chapter comprehensively outlines the methodologies integral in this project. It details the systematic approaches utilized to leverage advanced technologies and frameworks, ensuring robustness and reliability. Each methodology is meticulously designed to facilitate efficient data processing, user interaction, and seamless integration of audio-visual components.

GENERIC FRAMEWORK:



(Fig 4.1)

STEP 01: The input must be a meaningful english word as this project only accepts the language english, so input must be given according to it.

STEP 02: The processing happens with the mentioned libraries and algorithms for accurate output. The mechanisms used in processing are:

1. NLTK:

- **Methodology:** Utilizes NLTK (Natural Language Toolkit) along with WordNet to access lexical databases for semantic analysis.

- **Purpose:** Retrieves word meanings (synonyms, definitions, parts of speech) to provide structured data about the entered words.

2. Text-to-Speech Conversion (gTTS - Google Text-to-Speech):

- **Methodology:** Employs gTTS library to convert text-based word definitions into audio files.
- **Purpose:** Facilitates auditory presentation of word meanings, enhancing accessibility and user interaction.

3. Interactive User Interface (IPython Widgets):

- **Methodology:** Utilizes IPython widgets (Text for input and Button for interaction) to create an interactive interface.
- **Purpose:** Enables users to input words and trigger the retrieval and display of their meanings, enhancing user experience and engagement.

4. Audio Queue Management:

- **Methodology:** Implements a queue (audio_queue) to manage the sequence of audio playback.
- **Purpose:** Ensures that audio files are played in the correct order corresponding to the displayed word meanings, maintaining coherence between visual and auditory information.

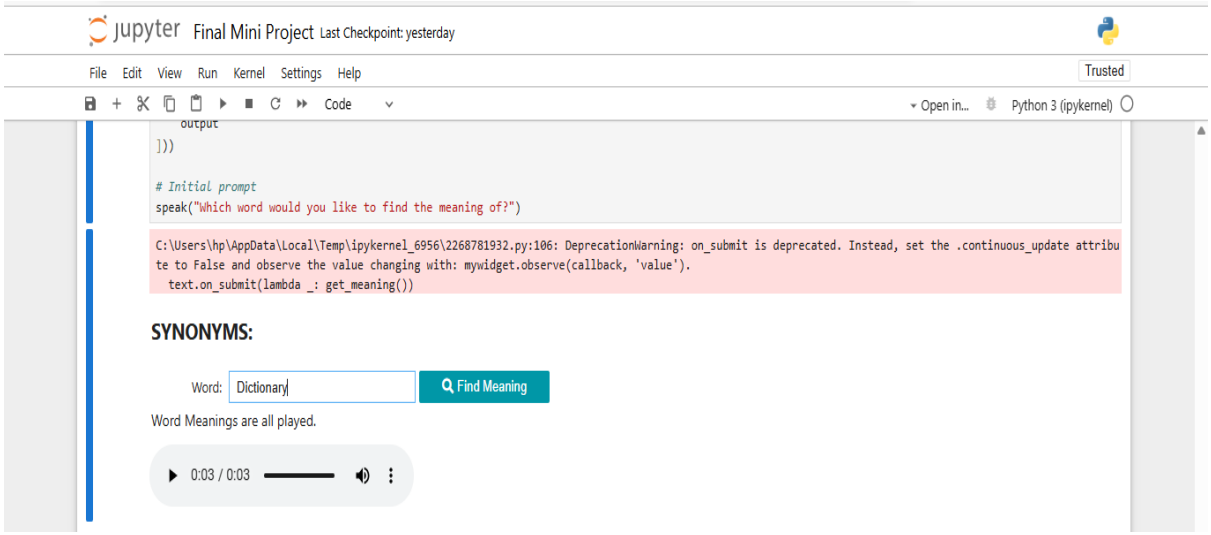
STEP 03: The output is displayed in both text and audio format.

CHAPTER 5

RESULTS AND DISCUSSION

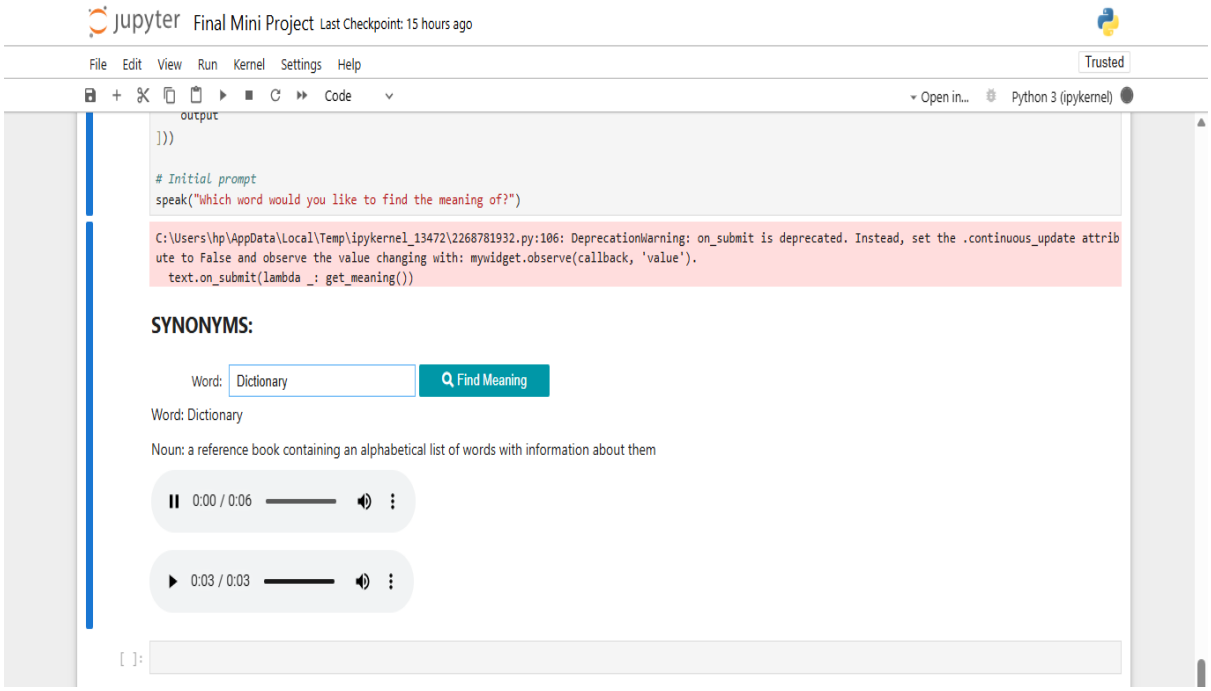
CASE 1:

INPUT:



(Fig 5.1)

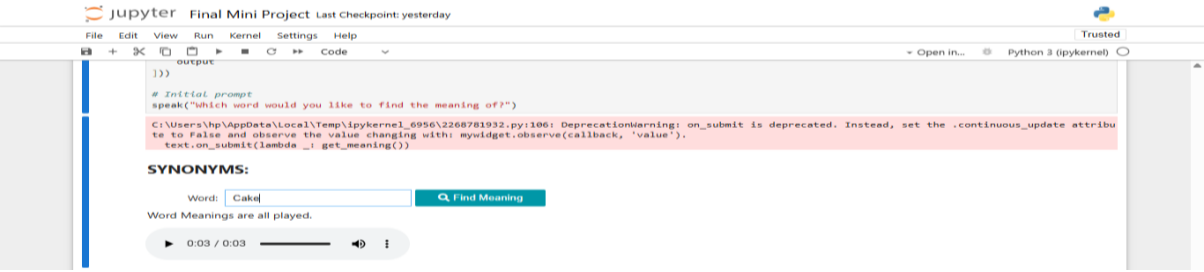
OUTPUT:



(Fig 5.2)

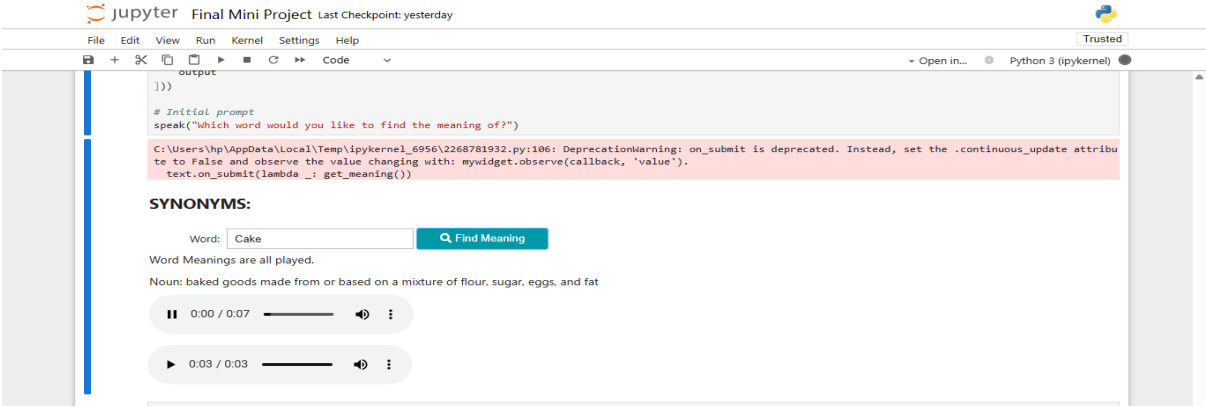
CASE 2:

INPUT:



(Fig 5.3)

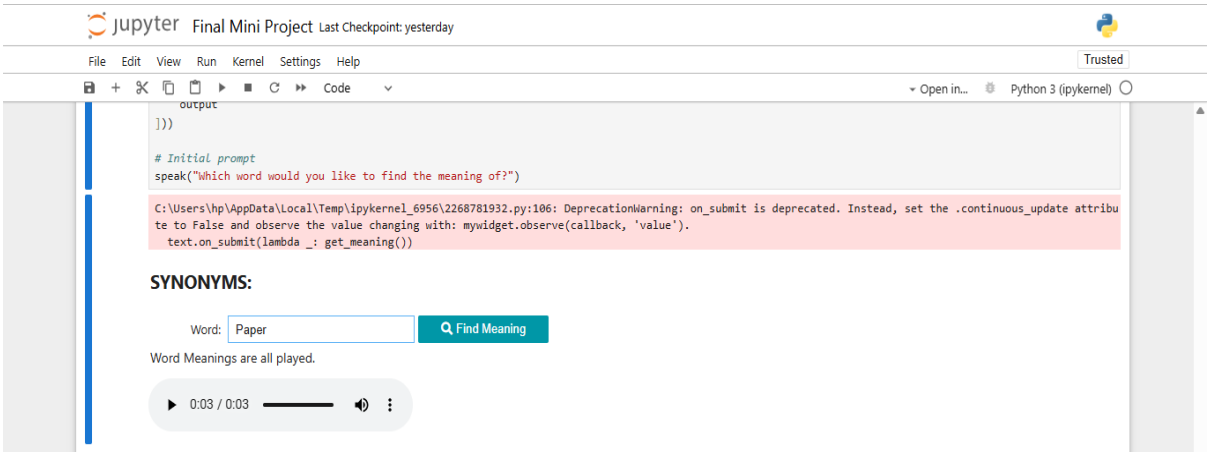
OUTPUT:



(Fig 5.4)

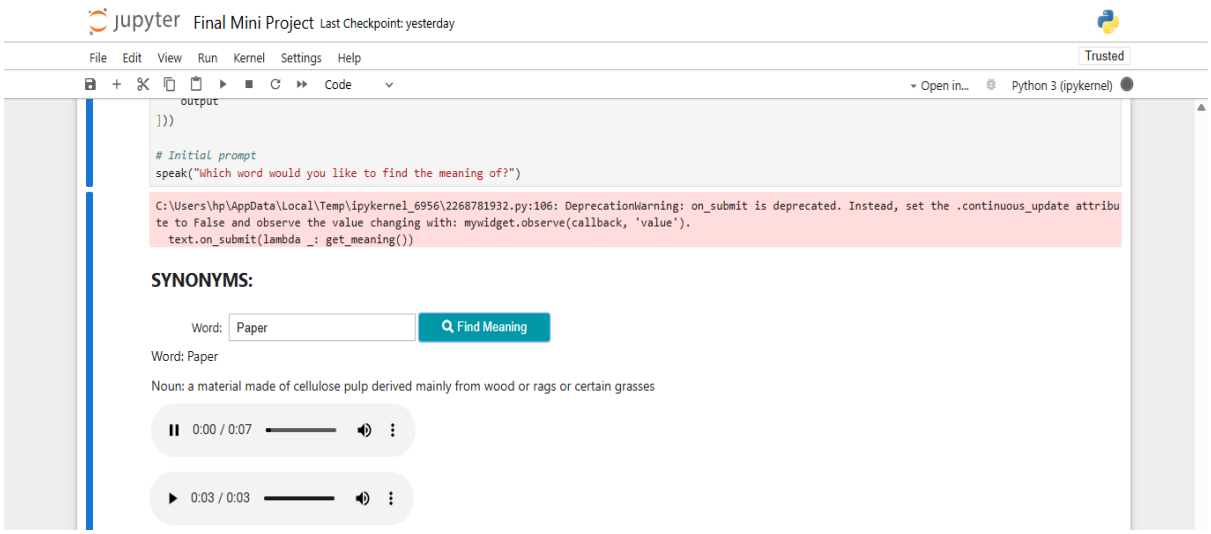
CASE 3:

INPUT:



(Fig 5.5)

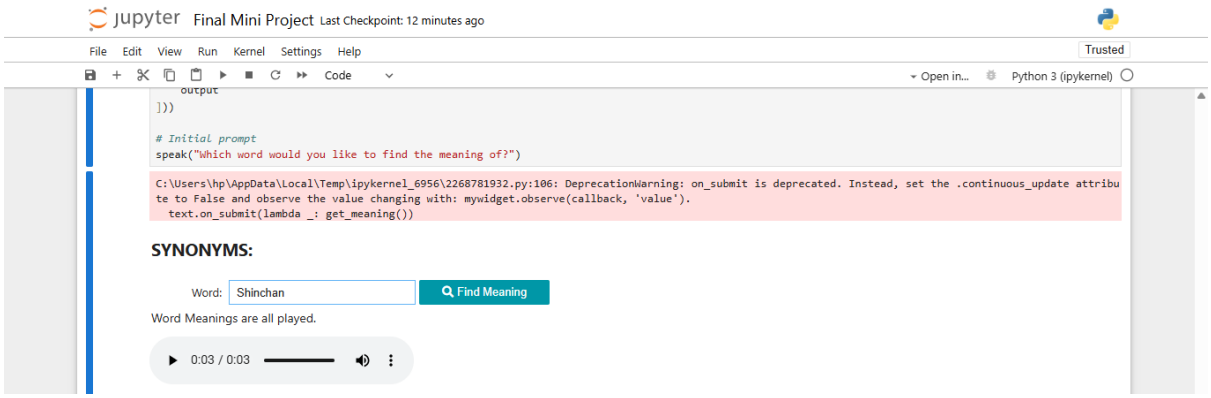
OUTPUT:



(Fig 5.6)

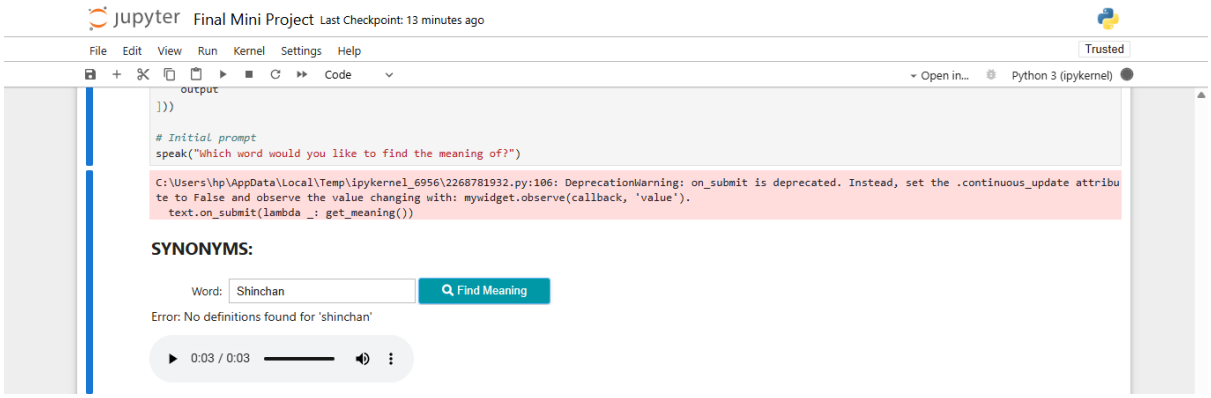
CASE 4:

INPUT:



(Fig 5.7)

OUTPUT:



(Fig 5.8)

NOTE: For the words that are not included, an error message will be displayed.

Each case runs based on the following steps:

User Interface and Input:

The code provides a user interface using IPython widgets, allowing users to input a word to find its meaning.

Text Processing and Meaning Retrieval:

Upon entering a word, the system processes it using NLTK to strip HTML tags from any input and convert it to lowercase.

It then retrieves the word meanings from WordNet, a lexical database for English, handling cases where the exact word or its title form (if it's a proper noun) may not have definitions.

Output Display:

The word and its meanings (if found) are displayed using Markdown formatting in the output area. Each meaning includes the part of speech and its definition.

Audio Playback:

Simultaneously, the system generates audio using gTTS (Google Text-to-Speech) for each meaning found and plays it back to the user.

Audio files are managed in a queue, ensuring sequential playback and cleanup after each word meaning is played.

DISCUSSION:

From the four cases in the result section it is proved that three of the audio and text meanings are perfectly correct except one.

Reasons for error:

- Word is not found in the dictionary.

CHAPTER 6

CONCLUSION

In conclusion, the implemented text-to-speech word meaning retrieval system effectively utilizes Python and various libraries such as NLTK, gTTS, and IPython widgets. This system allows users to input words and obtain their definitions through WordNet's extensive lexical database. The integration of gTTS facilitates the generation of audio pronunciations for each word definition, enhancing accessibility and user engagement.

Key features include the management of audio playback using a queue mechanism, ensuring seamless and synchronized delivery of audio feedback. The user interface, powered by IPython widgets, provides a user-friendly experience with interactive input and clear, formatted output of word meanings.

This project demonstrates Python's capabilities in natural language processing and audio synthesis, making it suitable for educational and accessibility-focused applications.

CHAPTER 7

REFERENCES

This chapter has details about the references that had been used along with the links.

TEXT BOOK:

1.https://s3.amazonaws.com/ebooks.syncfusion.com/downloads/Natural_Language_Processing_Succinctly/Natural_Language_Processing_Succinctly.pdf?AWSAccessKeyId=AKIAWH6GYCX36ZTC52O4&Expires=1719762178&Signature=zm0TmiJWiDPALs3EJBD9JHJlM0U%3D

2.https://karczmarczuk.users.greyc.fr/TEACH/TAL/Doc/Handbook%20Of%20Natural%20Language%20Processing,%20Second%20Edition%20Chapman%20&%20Hall%20Crc%20Machine%20Learning%20&%20Pattern%20Recognition%202010.pdf

WEBSITES:

3.https://www.coursera.org/articles/natural-language-processing

4.https://aws.amazon.com/whatis/nlp/#:~:text=Natural%20language%20processing%20(NLP)%20is,manipulate%2C%20and%20comprehend%20human%20language

5. https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk

6. https://devopedia.org/wordnet

7.https://en.wikipedia.org/wiki/WordNet#:~:text=WordNet%20is%20a%20lexical%20database,of%20a%20dictionary%20and%20thesaurus

8. https://speechify.com/blog/gtts/

9. https://gtts.readthedocs.io/en/latest/

10. https://www.charterglobal.com/user-interface-development/

11. https://www.uxpin.com/studio/blog/ui-design-vs-ui-development/

12.https://www.javatpoint.com/ipythondisplay#:~:text=IPython%20means%20interactive%20Python.,single%20line%20of%20Python%20code

13. https://www.techopedia.com/definition/3452/widget

CHAPTER 8

APPENDIX:

This chapter consists of the coding used in this project with its explanation.

CODING WITH EXPLANATION:

```
!pip install tk
!pip install PyDictionary
!pip install pyttsx3
!pip install ipywidgets
!pip install gtts
```

- Installing Required Packages Installs necessary Python packages (tk, PyDictionary, pyttsx3, ipywidgets, gtts) required for GUI development, text-to-speech conversion, and word definition retrieval.

```
import nltk

nltk.download('wordnet')

nltk.download('omw-1.4')
```

- Downloading NLTK Data: Downloads the WordNet dataset from NLTK, which is essential for word meaning retrieval.

```
import nltk
import os
import time
import re
from nltk.corpus import wordnet
from gtts import gTTS
from IPython.display import display, Audio, Markdown
import ipywidgets as widgets
# Download required NLTK data
```

```
nltk.download('wordnet', quiet=True)
nltk.download('omw-1.4', quiet=True)
# Queue to manage audio playback
audio_queue = []
def remove_html_tags(text):
    """Remove HTML tags from text."""
    clean = re.compile('<.*?>')
    return re.sub(clean, '', text)
def speak(text, remove_audio=True):
    """Generate and queue audio for text-to-speech."""
    plain_text = remove_html_tags(text)
    tts = gTTS(text=plain_text, lang='en')
    audio_file = "output.mp3"
    tts.save(audio_file)
    # Estimate audio duration (words per second)
    audio_duration = len(plain_text.split()) / 2.5
    audio_queue.append((audio_file, audio_duration))
    if len(audio_queue) == 1:
        play_next_audio(remove_audio)
```

- Importing Libraries: Imports essential libraries (nltk, os, time, re, wordnet from NLTK, gTTS from gtts, display, Audio, Markdown from IPython.display, widgets from ipywidgets) for natural language processing, audio handling, display functionalities, and user interface components.

- Queue Initialization for Audio Playback: Initializes an empty list (audio_queue) to manage the sequence of audio files for text-to-speech output.

- Function to Remove HTML Tags: Defines a function (remove_html_tags) that removes HTML tags from text, ensuring clean text input for processing.

- Function for Text-to-Speech Conversion: Defines a function (speak) that converts text to speech using gTTS, saves the output as an audio file (output.mp3), estimates the audio duration based on the text length, and manages the audio playback queue.

```
def play_next_audio(remove_audio=True):
    """Play the next audio in the queue."""
    if audio_queue:
        audio_file, audio_duration = audio_queue[0]
        display(Audio(audio_file, autoplay=True))
        time.sleep(audio_duration + 2)  # Reduced buffer time
        if remove_audio:
            try:
                os.remove(audio_file)
            except OSError:
                pass  # Ignore if file doesn't exist or can't be removed
        audio_queue.pop(0)  # Remove the played audio file from the queue
        if audio_queue:
            play_next_audio(remove_audio)  # Play the next audio if queue is not empty
def get_meaning(change=None):
    """Get and display the meaning of a word."""
    word = text.value.strip().lower()  # Convert input to lowercase
    if not word:
        return
    try:
        synsets = wordnet.synsets(word)
        if not synsets:
            raise ValueError(f"No definitions found for '{word}'")
        # Clear previous output
        output.clear_output(wait=True)
        with output:
            display(Markdown(f"*Word:* {text.value.strip()}"))
            for synset in synsets:
                pos_name = {'n': 'Noun', 'v': 'Verb', 'a': 'Adjective', 'r':
'Adverb'}.get(synset.pos(), synset.pos().capitalize())
                meaning_text = f"{pos_name}: {synset.definition().strip()}"
                display(Markdown(meaning_text))
                speak(f"{pos_name}. {synset.definition().strip()}", remove_audio=False)
```

```
        text.value = ""  # Clear input after processing
    except ValueError as ve:
        output.clear_output(wait=True)
        with output:
            display(Markdown(f"*Error:* {str(ve)}"))
        speak(f"No definitions found for '{word}'")
    except Exception as e:
        output.clear_output(wait=True)
        with output:
            display(Markdown(f"*Error:* {str(e)}"))
        speak(f"An error occurred: {str(e)}")
```

- Function to Play Next Audio: Defines a function (play_next_audio) that plays the next audio in the queue (audio_queue), manages autoplay, removes audio files after playback, and clears the output display after all audio files have been played.

- Function to Retrieve Word Meaning: Defines a function (get_meaning) that retrieves and displays the meaning of a word using WordNet, handles exceptions, clears output display, and triggers text-to-speech conversion for each definition.

```
# Create widgets
text = widgets.Text(description='Word:', placeholder='Type a word')
button = widgets.Button(description='Find Meaning', button_style='info',
icon='search')
output = widgets.Output()
# Connect widgets
text.on_submit(lambda _: get_meaning())
button.on_click(get_meaning)
# Display widgets
display(widgets.VBox([
    widgets.HTML('<h2>SYNONYMS:</h2>'),
    widgets.HBox([text, button]),
    output
]))
```

 # Initial prompt

speak("Which word would you like to find the meaning of?")

- Creating User Interface Widgets: Creates widgets using ipywidgets for building an interactive user interface.

- Connecting Widgets and Displaying Interface: Connects the text input widget to trigger get_meaning function on submit and button widget to trigger the same function on click, facilitating user interaction.

- Displaying Interface Elements: Displays a structured user interface using ipywidgets to enhance user experience.

- Initial Prompt for User Interaction: Initiates a text-to-speech prompt asking the user for input, enhancing user interaction with spoken instructions.