

Exercise 3:

Develop and compare CLI, GUI, and Voice User Interfaces (VUI) for the same task and assess user satisfaction using Python (Tkinter for GUI, Speech Recognition for VUI), Terminal

AIM:

The aim is to develop and compare Command Line Interface (CLI), Graphical User Interface (GUI), and Voice User Interface (VUI) for the same task, and assess user satisfaction using Python (with Tkinter for GUI and Speech Recognition for VUI) and Terminal.

PROCEDURE:

i) CLI (Command Line Interface)

CLI implementation where users can add, view, and remove tasks using the terminal.

```
tasks = []
def add_task(task):
    tasks.append(task)
    print(f"Task '{task}' added.")

def view_tasks():
    if tasks:
        print("Your tasks:")
        for idx, task in enumerate(tasks, 1):
            print(f"{idx}. {task}")
    else:
        print("No tasks to show.")
```

```

def remove_task(task_number):
    if 0 < task_number <= len(tasks):
        removed_task = tasks.pop(task_number - 1)
        print(f"Task '{removed_task}' removed.")
    else:
        print("Invalid task number.")

def main():
    while True:
        print("\nOptions: 1.Add Task 2.View Tasks 3.Remove
Task 4.Exit")
        choice = input("Enter your choice: ")

        if choice == '1.':
            task = input("Enter task: ")
            add_task(task)
        elif choice == '2.':
            view_tasks()
        elif choice == '3':
            task_number = int(input("Enter task number to
remove: "))
            remove_task(task_number)
        elif choice == '4':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

if name_== " main ":
    main()

```

OUTPUT:

The screenshot shows the Spyder Python IDE with a file named `temp.py` open. The script defines a list `tasks` and three functions: `add_task`, `view_tasks`, and `remove_task`. The `main` function uses a `while True` loop to present a menu of options (1. Add Task, 2. View Tasks, 3. Remove Task, 4. Exit) and processes user input. The console on the right shows the execution of the script, displaying the menu and the results of user choices: adding 'Create Button', viewing the tasks, and removing the task at index 1.

```

1 tasks = []
2 def add_task(task):
3     tasks.append(task)
4     print(f"Task '{task}' added.")
5 def view_tasks():
6     if tasks:
7         print("Your tasks:")
8         for idx, task in enumerate(tasks, 1):
9             print(f"{idx}. {task}")
10    else:
11        print("No tasks to show.")
12 def remove_task(task_number):
13     if 0 < task_number <= len(tasks):
14         removed_task = tasks.pop(task_number - 1)
15         print(f"Task '{removed_task}' removed.")
16     else:
17         print("Invalid task number.")
18 def main():
19     while True:
20         print("\nOptions: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit")
21         choice = input("Enter your choice: ")
22         if choice == '1':
23             task = input("Enter task: ")
24             add_task(task)

```

```

In [1]: runfile('C:/Users/mdars/.spyder-py3/temp.py', wdir='C:/Users/mdars/.spyder-py3')

Options: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit
Enter your choice: 1
Enter task: Create Button
Task 'Create Button' added.

Options: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit
Enter your choice: 2
Your tasks:
1. Create Button

Options: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit
Enter your choice: 3
Enter task number to remove: 1
Task 'Create Button' removed.

Options: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit
Enter your choice: 4
Exiting...

```

ii) GUI (Graphical User Interface)

Tkinter to create a simple GUI for our To-Do List application.

```

import tkinter as tk
from tkinter import messagebox

```

```
tasks = []
```

```

def add_task():
    task = task_entry.get()
    if task:
        tasks.append(task)
        task_entry.delete(0, tk.END)
        update_task_list()
    else:
        messagebox.showwarning("Warning", "Task cannot be empty")

```

```

def update_task_list():
    task_list.delete(0, tk.END)
    for task in tasks:
        task_list.insert(tk.END, task)

```

```
def remove_task():
    selected_task_index = task_list.curselection()
    if selected_task_index:
        task_list.delete(selected_task_index)
        tasks.pop(selected_task_index[0])

app = tk.Tk()
app.title("To-Do List")

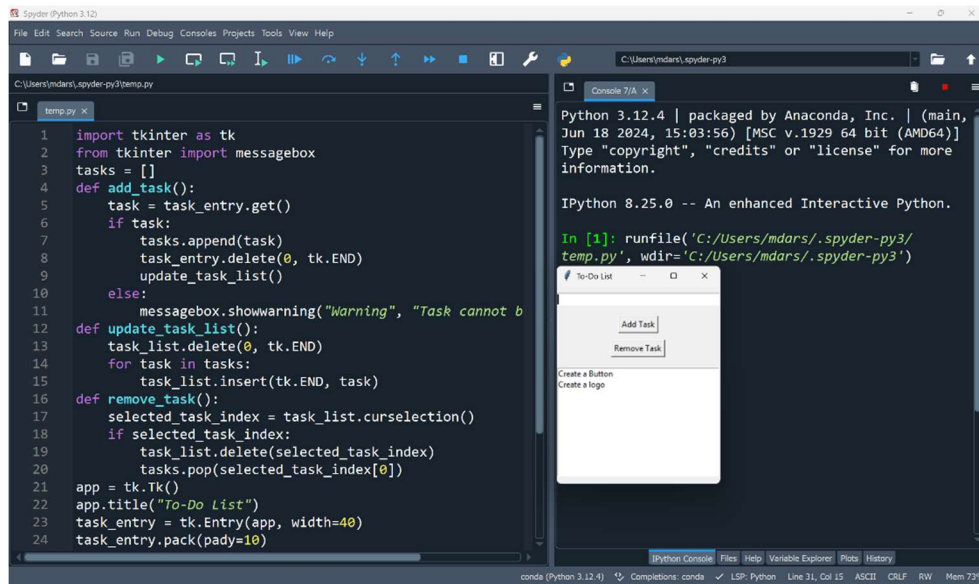
task_entry = tk.Entry(app, width=40)
task_entry.pack(pady=10)
add_button = tk.Button(app, text="Add Task",
command=add_task)
add_button.pack(pady=5)

remove_button = tk.Button(app, text="Remove Task",
command=remove_task)
remove_button.pack(pady=5)

task_list = tk.Listbox(app, width=40, height=10)
task_list.pack(pady=10)

app.mainloop()
```

OUTPUT:



iii) VUI (Voice User Interface)

speech_recognition library for voice input and the pyttsx3 library for text-to-speech output. Make sure you have these libraries installed (pip install SpeechRecognition pyttsx3).

```
import speech_recognition as sr
import pyttsx3
```

```
tasks = []
recognizer = sr.Recognizer()
engine = pyttsx3.init()
```

```
def add_task(task):
    tasks.append(task)
    engine.say(f"Task {task} added")
    engine.runAndWait()
```

```
def view_tasks():
    if tasks:
        engine.say("Your tasks are")
        for task in tasks:
            engine.say(task)
```

```

else:
    engine.say("No tasks to show")
    engine.runAndWait()

def remove_task(task_number):
    if 0 < task_number <= len(tasks):
        removed_task = tasks.pop(task_number - 1)
        engine.say(f"Task {removed_task} removed")
    else:
        engine.say("Invalid task number")
    engine.runAndWait()

def recognize_speech():
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)
        try:
            command = recognizer.recognize_google(audio)
            return command
        except sr.UnknownValueError:
            engine.say("Sorry, I did not understand that")
            engine.runAndWait()
            return None

def main():
    while True:
        engine.say("Options: add task, view tasks, remove
task, or exit")
        engine.runAndWait()

        command = recognize_speech()
        if not command:
            continue

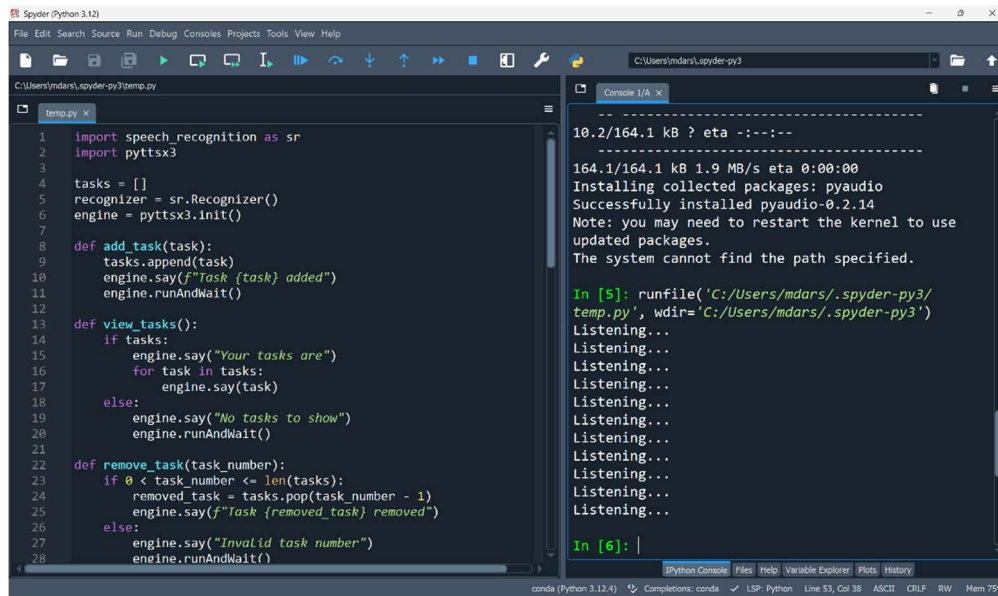
        if "add task" in command:
            engine.say("What is the task?")
            engine.runAndWait()
            task = recognize_speech()
            if task:

```

```
        add_task(task)
elif "view tasks" in command:
    view_tasks()
elif "remove task" in command:
    engine.say("Which task number to remove?")
    engine.runAndWait()
    task_number = recognize_speech()
    if task_number:
        remove_task(int(task_number))
elif "exit" in command:
    engine.say("Exiting...")
    engine.runAndWait()
    break
else:
    engine.say("Invalid option. Please try again.")
    engine.runAndWait()

if name_ == " main ":
    main()
```

OUTPUT:



The image shows the Spyder Python IDE interface. The left pane displays a Python script named `temp.py` with the following code:

```
1 import speech_recognition as sr
2 import pytsx3
3
4 tasks = []
5 recognizer = sr.Recognizer()
6 engine = pytsx3.init()
7
8 def add_task(task):
9     tasks.append(task)
10    engine.say(f"Task {task} added")
11    engine.runAndWait()
12
13 def view_tasks():
14     if tasks:
15         engine.say("Your tasks are")
16         for task in tasks:
17             engine.say(task)
18     else:
19         engine.say("No tasks to show")
20     engine.runAndWait()
21
22 def remove_task(task_number):
23     if 0 < task_number <= len(tasks):
24         removed_task = tasks.pop(task_number - 1)
25         engine.say(f"Task {removed_task} removed")
26     else:
27         engine.say("Invalid task number")
28     engine.runAndWait()
```

The right pane shows the IPython console output:

```
-----
10.2/164.1 kB ? eta --:--
-----
164.1/164.1 kB 1.9 MB/s eta 0:00:00
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.14
Note: you may need to restart the kernel to use
updated packages.
The system cannot find the path specified.

In [5]: runfile('C:/Users/mdars/.spyder-py3/
temp.py', wdir='C:/Users/mdars/.spyder-py3')
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...

In [6]:
```

The status bar at the bottom indicates the environment is `conda (Python 3.12.4)` with various tool icons.

RESULT:

The output was verified successfully.