EXPERIMENT 03: CLI,GUI AND VUI

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:

1. CLI:

### Step 1: Set Up the Python Environment

* Ensure Python is installed on your system (python --version).
* Create a new Python script file, e.g., todo.py.

Step 2: Define Core Functions

* add\_task(task) → Appends a task to a list or file.
* view\_tasks() → Displays all tasks with numbering.
* remove\_task(task\_number) → Deletes a selected task.
* save\_tasks() → Stores tasks in a file (e.g., tasks.txt).
* load\_tasks() → Retrieves tasks from the file at startup.

Step 3: Implement User Interaction via CLI

* Use input() to get user commands (e.g., "add", "view", "remove", "exit").
* Implement a loop to continuously accept commands.
* Use if-elif conditions to call respective functions.

Step 4: Handle File Storage (Optional but Recommended)

* Store tasks in a text file (tasks.txt) for persistence.
* Read and write tasks using open() in append/read mode.
* Ensure error handling for file operations.

Step 5: Test and Run the Program

* Run the script: python todo.py.
* Add, view, and remove tasks to test functionality.
* Improve with enhancements like colored output (colorama) or JSON storage.

### **2.GUI:**

### **Step 1: Set Up the Python Environment**

* Ensure Python is installed (python --version).
* Install **Tkinter** (built-in) or other GUI frameworks like PyQt (pip install PyQt6).

### **Step 2: Create the GUI Layout**

* Use Tkinter to create a main window.
* Add widgets:
  + **Entry Box** for adding tasks.
  + **Listbox** for displaying tasks.
  + **Buttons** for **Add, Remove, and Clear** actions.

### **Step 3: Implement Task Management Functions**

* add\_task() → Adds a task from the entry box to the listbox.
* remove\_task() → Deletes the selected task.
* clear\_tasks() → Clears all tasks.
* save\_tasks() → Saves tasks in a file (tasks.txt).
* load\_tasks() → Loads tasks on startup.

### **Step 4: Handle Events & User Input**

* Bind buttons to respective functions (command=add\_task).
* Use **Double-click events** to remove tasks.

### **Step 5: Test & Run the GUI Application**

* Run python gui\_todo.py.
* Ensure all buttons and actions function correctly.
* Enhance UI with ttk styling or migrate to PyQt for advanced design.

3.VUI:

### **Step 1: Install Dependencies**

* Install speechrecognition, pyttsx3, and pyaudio:

bash

CopyEdit

pip install speechrecognition pyttsx3 pyaudio

### **Step 2: Set Up Speech Recognition & Synthesis**

* Use speech\_recognition to convert voice to text.
* Use pyttsx3 for text-to-speech responses.

### **Step 3: Implement Voice-Controlled Functions**

* listen\_command() → Captures user voice and converts it to text.
* add\_task(task) → Adds a task from spoken input.
* remove\_task(task\_number) → Removes a spoken task index.
* speak(text) → Provides audio feedback.

### **Step 4: Implement Command Processing Logic**

* Recognize voice commands like **"Add task: Buy groceries"** or **"Remove task 2"**.
* Use if-elif to process recognized commands and call respective functions.

### **Step 5: Test & Improve Recognition**

* Run python vui\_todo.py.
* Test various commands and fine-tune recognition accuracy.
* Add NLP improvements using **Google Speech API** or **Whisper AI** for better accuracy.

PROGRAM:

CLI- Command line interface

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 yoour 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()



GUI – Graphical User Interface

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)

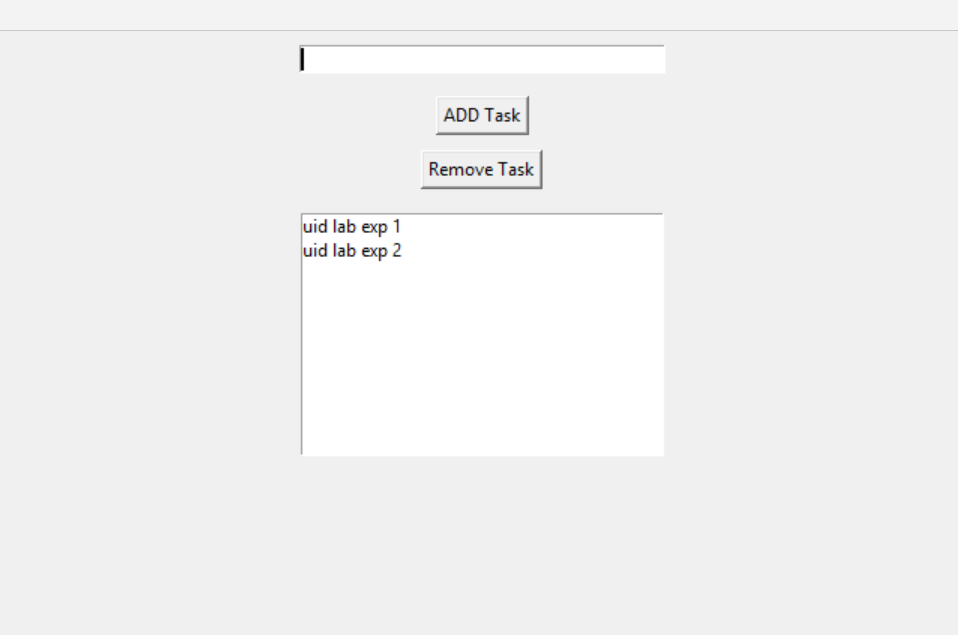
remove\_button = tk.Button(app, text="Remove Task", command=remove\_task)

remove\_button.pack(pady=5)

task\_list = tk.Listbox(app, width=40, height=10)

task\_list.pack(pady=10)

app.mainloop()



VUI – Voice User Interface

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

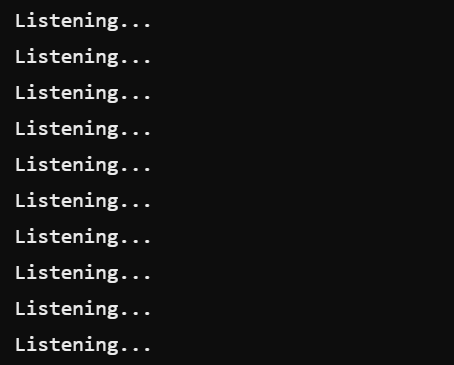
        else:

            engine.say(";Invalid option. Please try again.")

            engine.runAndWait()

if \_\_name\_\_ == "\_\_main\_\_":

    main()



RESULT:

User satisfaction varies based on familiarity—CLI is fast for experienced users, GUI is intuitive for general users, and VUI offers hands-free convenience but may have recognition limitations.