EXP NO 02:
TITLE: 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:

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.")
```

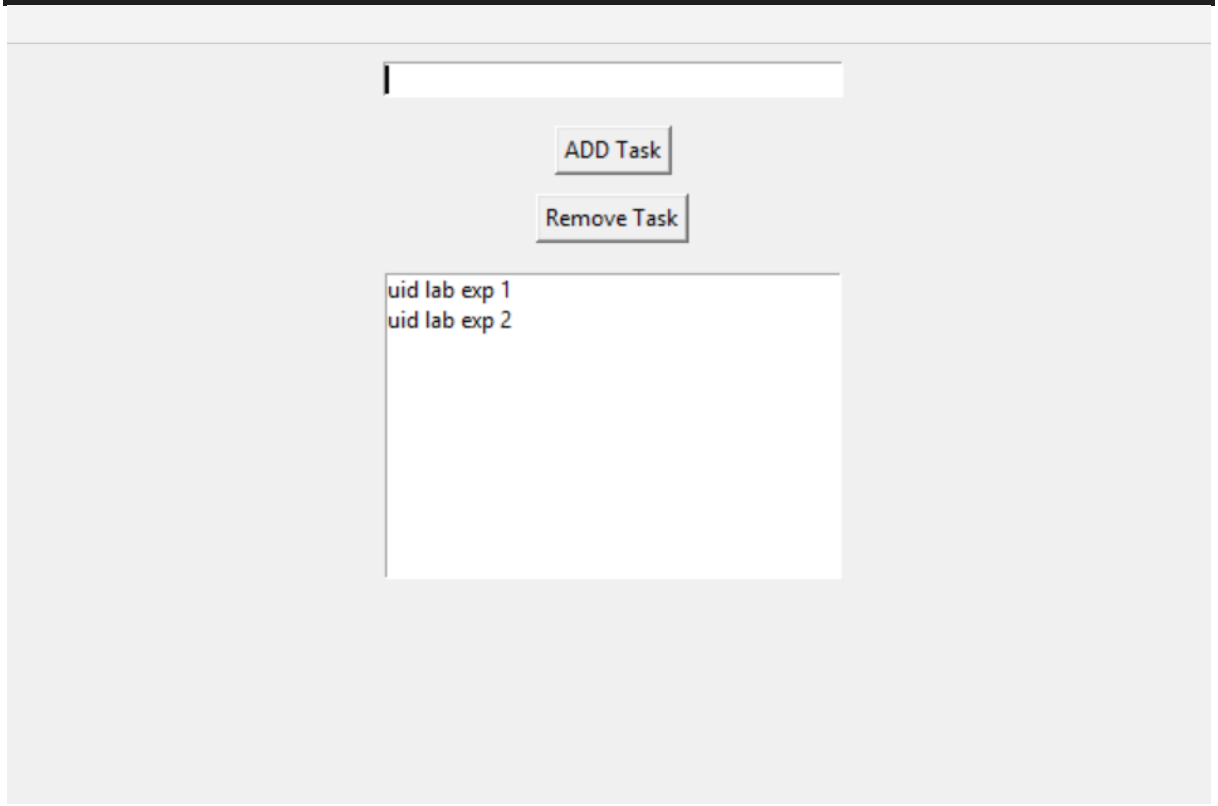
```
def view_tasks():
    if tasks:
        print("Your tasks:")
        for idx,task in enumerate(tasks,1):
            print(f"{idx}.{task}")

    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()
```

```
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/HDC0422041/Documents/CLI 312.py ==========

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:2.
No tasks to show.

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:1.
Enter task: we
Task 'we'added.

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:1.
Enter task: the
Task 'the'added.

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:1.
Enter task: mee
Task 'mee'added.

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:2.
Your tasks:
1.we
2.the
3.mee

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:3.
Enter task number to remove: 1
Task'we'removed.

Options: 1. Add Task 2.View Tasks 3.Remove Task 4.Exit
enter yoour choice:4.
Exiting..
>>> S
```

GUI – Graphical User Interface

```python
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()
```

```
Listening...
(Recognized: "add task")
Listening...
(Recognized: "Buy groceries")
Task Buy groceries added

Listening...
(Recognized: "add task")
Listening...
(Recognized: "Finish project report")
Task Finish project report added

Listening...
(Recognized: "view tasks")
Your tasks are
Buy groceries
Finish project report
```

```
Listening...
(Recognized: "remove task")
Listening...
(Recognized: "1")
Task Buy groceries removed

Listening...
(Recognized: "view tasks")
Your tasks are
Finish project report

Listening...
(Recognized: "exit")
Exiting...
```

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.