EXPERIMENT 03: CLI, GUI AND VUI

Aim:

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 (Command-Line Interface)**

- **Set Up**:
  - Install Python and create a new script (e.g., todo.py).
- **Core Functions**:
  - add_task(task): Adds a task.
  - view_tasks(): Displays all tasks with numbering.
  - remove_task(task_number): Deletes a task by index.
  - save_tasks(): Saves tasks to a file.
  - load_tasks(): Loads tasks from a file at startup.
- **User Interaction**:
  - Use input() to capture user commands (add, view, remove, exit).
  - Implement a loop to continuously handle commands.
- **File Storage**:
  - Tasks are saved in tasks.txt.
  - Use file handling functions (open()) for reading and writing tasks.

- **Test & Run**:
    - Test the functions by running the script and checking task management functionality.

2. GUI (Graphical User Interface)

- **Set Up**:
    - Install Python, Tkinter (or PyQt6).
- **Create GUI Layout**:
    - Main window, entry box for task input, listbox for task display, buttons for add, remove, and clear actions.
- **Task Management Functions**:
    - add_task(): Adds a task from the input box to the list.
    - remove_task(): Deletes a selected task.
    - clear_tasks(): Clears all tasks.
    - save_tasks(): Saves tasks in a file.
    - load_tasks(): Loads tasks at startup.
- **Event Handling**:
    - Bind buttons to respective functions and implement task removal on double-click.
- **Test & Run**:
    - Ensure all actions (add, remove, clear) work correctly in the GUI and run the script.

3. VUI (Voice User Interface)

- **Install Dependencies**:
    - Install necessary libraries: speechrecognition, pyttsx3, pyaudio.
- **Set Up Speech Recognition & Synthesis**:

- Use speech_recognition to convert speech to text.
- Use pyttsx3 for text-to-speech responses.

- **Voice-Controlled Functions**:
  - listen_command(): Captures voice input.
  - add_task(task): Adds a task based on spoken input.
  - remove_task(task_number): Removes a task using voice input.
  - speak(text): Provides audio feedback.
- **Command Processing**:
  - Recognize voice commands (e.g., "Add task: Buy groceries" or "Remove task 2") and map them to functions.
- **Test & Improve**:
  - Test the script with different commands and refine speech recognition accuracy.

Program & Output:

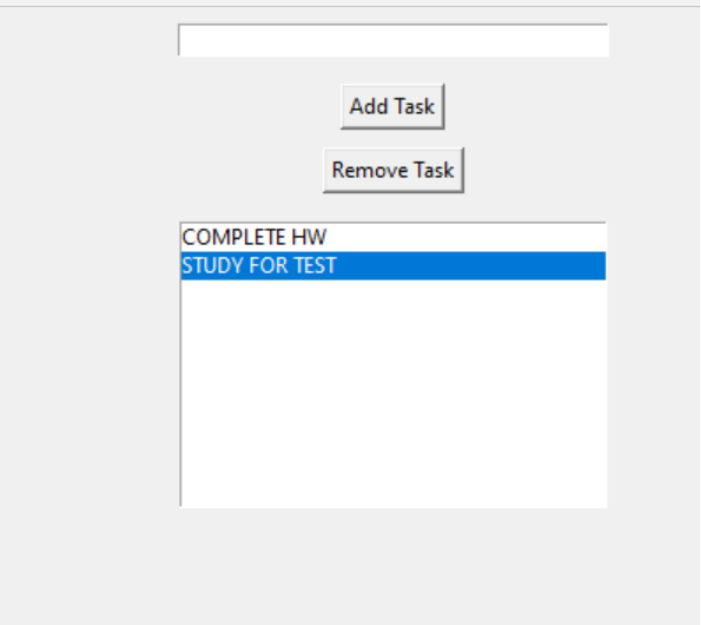
CLI:

```
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 task 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.try again.")
if __name__=="__main__":
    main()
```

```
Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice
invalid choice.try again.

Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice1
enter task:do hw
Task 'do hw'added.

Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice2
Your tasks:
1.do hw

Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice3
enter task number to remove:1
Task'do hw' removed

Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice2
no tasks to show

Options: 1.add task 2.view task 3.remove task 4.exit
enter your choice
```

GUI:

```
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:

```
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
```

```python
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...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
```

Result:

The experiment successfully developed a To-Do application with CLI, GUI, and VUI interfaces, each allowing efficient task management and seamless user interaction.