**EXPERIMENT 3 : DEVELOP AND COMPARE CLI,GUI AND VUI FOR THE SAME TASK AND ACCESS USER SATISFACTION USING PYTHON(Tkinter FOR GUI,SPEECH RECOGNITION FOR VUI),TERMINAL**

**Aim :**

To develop and compare Command Line Interface (CLI), Graphical User Interface (GUI), and Voice User Interface (VUI) for a task management system using Python and evaluate their usability.

**Procedure :**

1. COMMAND LINE INTERFACE (CLI) :

A CLI is a text-based interface where users interact with the system by typing commands in a 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 :**

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

1. GRAPHICAL USER INTERFACE (GUI):

A GUI is a visual-based interface where users interact with the system using buttons, input fields, and other graphical elements.

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:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. VOICE USER INTERFACE (VUI) :

A VUI allows users to interact with a system using voice commands, making it a hands-free alternative.

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:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Result :**

CLI is efficient and fast but requires users to remember commands. GUI is user-friendly and visually appealing but consumes more resources. VUI enables hands-free operation but may misinterpret speech in noisy environments. Each interface has its strengths, and the choice depends on user preference and application requirements.