COMMAND LINE INTERFACE:

A text-based interface where users interact with a system by typing commands. It is efficient but requires knowledge of specific commands . A Command Line Interface (CLI) is a text-based interface used to interact with software and operating systems by typing commands into a terminal or console window. Unlike graphical user interfaces (GUIs), which rely on visual elements like buttons and icons, a CLI requires users to input commands manually to perform tasks, such as file management, system configuration, and running programs.

```
import os import sys def rename_file(TECHATHON, new_name):       try:
        os.rename(TECHATHON, new_name)          print(f"File renamed from {TECHATHON}
to {new_name}")      except FileNotFoundError:
        print(f"Error: {TECHATHON} not found.")      except Exception as e:
                print(f"An error occurred: {e}")

   if __name__ == "__main__":       if
len(sys.argv) != 3:
                        print("Usage: python rename_file_cli.py <old_filename>
<new_filename>")     else:
                rename_file(sys.argv[1], sys.argv[2])
```

OUTPUT

```
Usage: python rename_file_cli.py <old_filename> <new_filename>

[Done] exited with code=0 in 0.031 seconds
```

GRAPHICAL USER INTERFACE:

A visual interface that allows users to interact with a system using graphical elements like buttons, icons, and windows, making it more user-friendly. GUIs make software more accessible and user-friendly by enabling users to control and navigate the system through graphical elements, rather than requiring them to know specific commands. For example, when you interact with your computer or phone, you use a GUI to click on icons, drag files, or open applications.

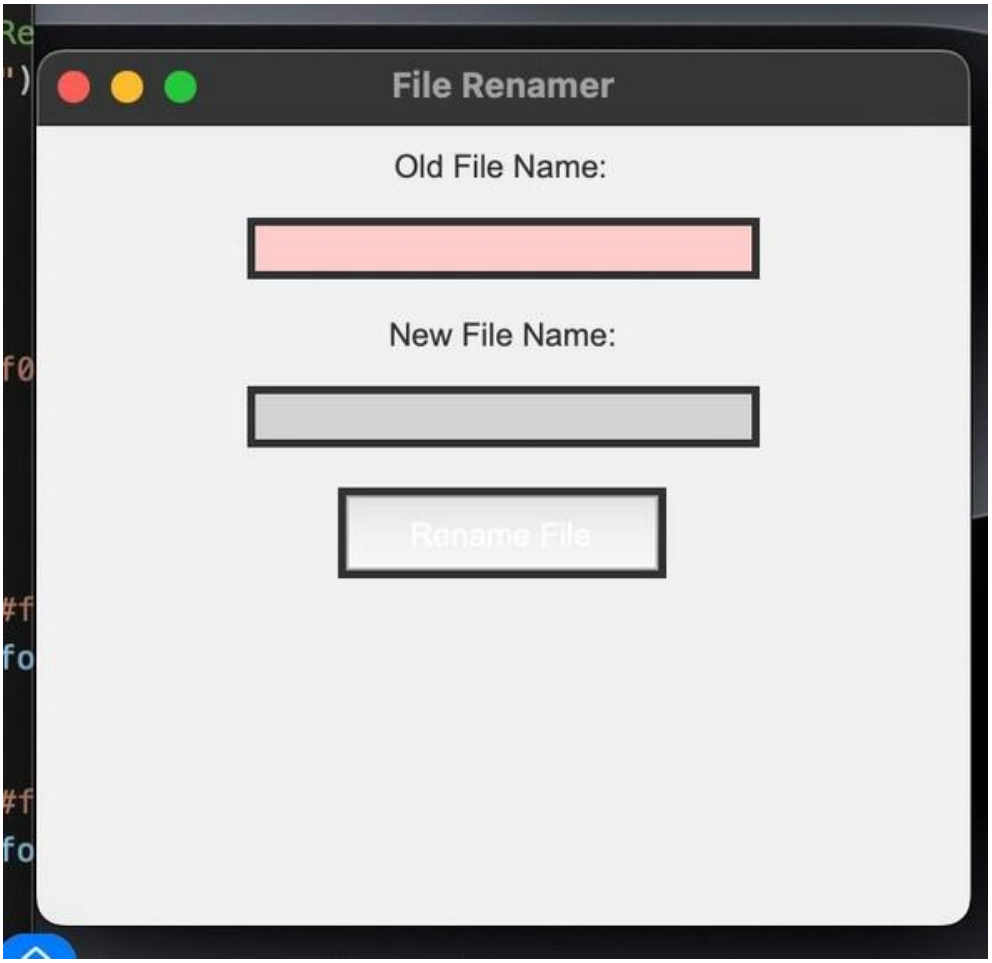
Key components of a GUI include:

- Icons: Small pictures or symbols that represent files, applications, or functions.
- Buttons: Elements that the user can click to execute an action.
- Windows: Rectangular areas of the screen where content (text, images, etc.) is displayed.
- Menus: Lists of options or commands that can be selected. Text Fields: Areas where
- users can input text.

```
import os import tkinter as tk from tkinter import messagebox,
PhotoImage def rename_file():
        old_name = old_name_entry.get()    new_name     =
       new_name_entry.get() if not old_name or not
       new_name:
```

```
messagebox.showerror("Error", "Please enter both old and new file names.")
                return

        if not os.path.exists(old_name):
                        messagebox.showerror("Error", "The specified file does not exist.")
                return
try:
         os.rename(old_name, new_name)    messagebox.showinfo("Success", f"File renamed to
{new_name}")          except Exception as e:          messagebox.showerror("Error", f"Failed to rename
file: {e}")

# Create the main window root = tk×Tk()
root.title("File Renamer")
root.geometry("350x300")
root×configure(bg="#f0f0f0")

# Load images (Ensure the image files exist in the same directory) try:

        icon = PhotoImage(file="icon.png")     # Replace with your image file tk.Label(root, image=icon,
        bg="#f0f0f0")×pack(pady=5)
except Exception as e:          print(f"Image not
loaded: {e}")

try:
         banner = PhotoImage(file="banner.png")      # Replace with your image file          tk.Label(root,
image=banner, bg="#f0f0f0")×pack(pady=5) except Exception as e:
                print(f"Banner image not loaded: {e}")

# Labels and entry fields
tk.Label(root, text="Old File Name:", bg="#f0f0f0", fg="#333", font=("Arial", 12))×pack(pady=5)
old_name_entry = tk.Entry(root, width=30, font=("Arial", 10), bg="#ffcccc") old_name_entry×pack(pady=5)

tk.Label(root, text="New File Name:", bg="#f0f0f0", fg="#333", font=("Arial", 12))×pack(pady=5)
new_name_entry = tk.Entry(root, width=30, font=("Arial", 10), bg="#d3d3d3") new_name_entry×pack(pady=5)

# Rename button
rename_button = tk×Button(root, text="Rename File", command=rename_file, bg="#007bff", fg="white",
font=("Arial", 12), padx=10, pady=5) rename_button×pack(pady=10)

# Run the Tkinter event loop
root.mainloop()
```

OUTPUT



VOICE USER INTERFACE:

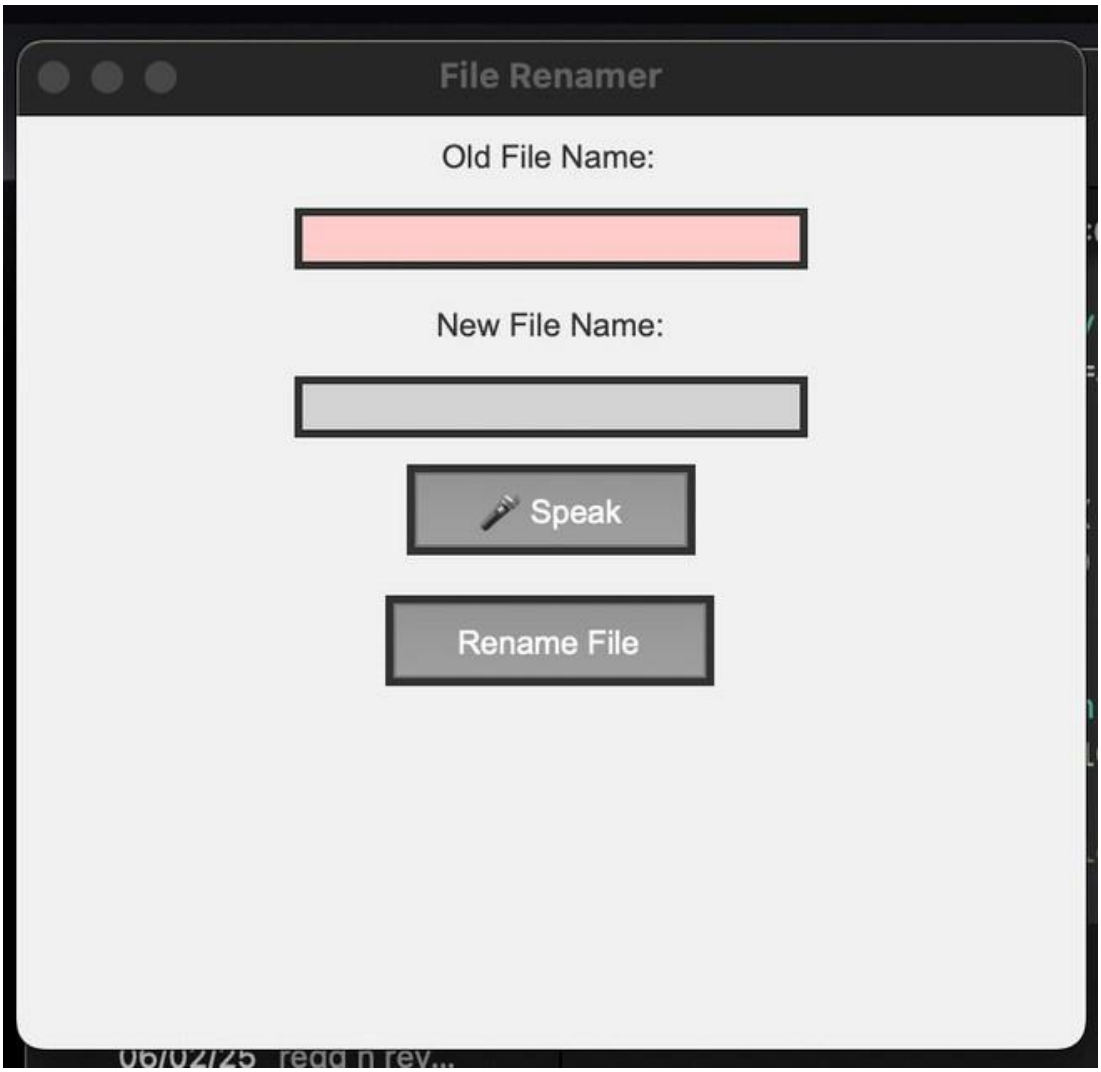
A system that allows users to interact through voice commands, using speech recognition technology, commonly found in virtual assistants like Siri and Alexa.A VUI (Voice User Interface) is a type of user interface that allows users to interact with a system or device through voice commands rather than using a graphical interface or physical input devices like a keyboard or mouse. With a VUI, users can control and navigate a system using spoken language.

```
import os  import tkinter as tk from
tkinter  import messagebox, PhotoImage i
mport re
try:
        import speech_recognition as sr except
ImportError:
        messagebox.showerror("Error",  "SpeechRecognition  module  is  not  installed.  Run  'pip  install
SpeechRecognition'.")
        exit()

def recognize_speech():
        recognizer = sr×Recognizer()  with
sr.Microphone() as source:
        messagebox.showinfo("Voice Input", "Listening...")          try:
                        audio = recognizer.listen(source)
                          text = recognizer.recognize_google(audio)

                match = re×search(r"rename (.+) to (.+)", text, re.IGNORECASE)          if match:
        old_name      =      match.group(1).strip()      new_name =
match.group(2).strip()      old_name_entry.delete(0,    tk.END)
```

```
        old_name_entry.insert(0, old_name)          new_name_entry.delete(0,
        tk.END)         new_name_entry.insert(0,    else:new_name)

        messagebox.showerror("Error", "Could not understand the rename command.")  except
sr.UnknownValueError:
        messagebox.showerror("Error", "Could not understand audio")     except sr.RequestError:
        messagebox.showerror("Error", "Could not request results, check your internet connection")

def rename_file():
        old_name = old_name_entry.get()    new_name     =
       new_name_entry.get() if not old_name or not
       new_name:
 messagebox.showerror("Error", "Please enter both old and new file names.")
                return

       if not os.path.exists(old_name):
                        messagebox.showerror("Error", "The specified file does not exist.")
                return
 try:
        os.rename(old_name, new_name)    messagebox.showinfo("Success", f"File renamed to
{new_name}")         except Exception as e:          messagebox.showerror("Error", f"Failed to rename
file: {e}")

# Create the main window root = tk×Tk()
root.title("File Renamer")
root.geometry("400x350") root×configure(bg="#f0f0f0")

# Load images (Ensure the image files exist in the same directory) try:
        icon = PhotoImage(file="icon.png")   # Replace with your image file          tk.Label(root,
image=icon, bg="#f0f0f0")×pack(pady=5) except Exception as e:    print(f"Image not loaded: {e}")

try:
        banner = PhotoImage(file="banner.png")      # Replace with your image file          tk.Label(root,
image=banner, bg="#f0f0f0")×pack(pady=5) except Exception as e:          print(f"Banner image not loaded:
{e}")

# Labels and entry fields
tk.Label(root, text="Old File Name:", bg="#f0f0f0", fg="#333", font=("Arial", 12))×pack(pady=5)
old_name_entry = tk.Entry(root, width=30, font=("Arial", 10), bg="#ffcccc") old_name_entry×pack(pady=5)

tk.Label(root, text="New File Name:", bg="#f0f0f0", fg="#333", font=("Arial", 12))×pack(pady=5)
new_name_entry = tk.Entry(root, width=30, font=("Arial", 10), bg="#d3d3d3") new_name_entry×pack(pady=5)

# Voice input button voice_button = tk×Button(root, text="🎤 Speak", command=recognize_speech,
bg="#28a745", fg="white", font=("Arial", 12), padx=10, pady=5) voice_button×pack(pady=5)
```

```
# Rename button
rename_button = tk×Button(root, text="Rename File", command=rename_file, bg="#007bff", fg="white",
font=("Arial", 12), padx=10, pady=5) rename_button×pack(pady=10)

# Run the Tkinter event loop
root.mainloop()
```

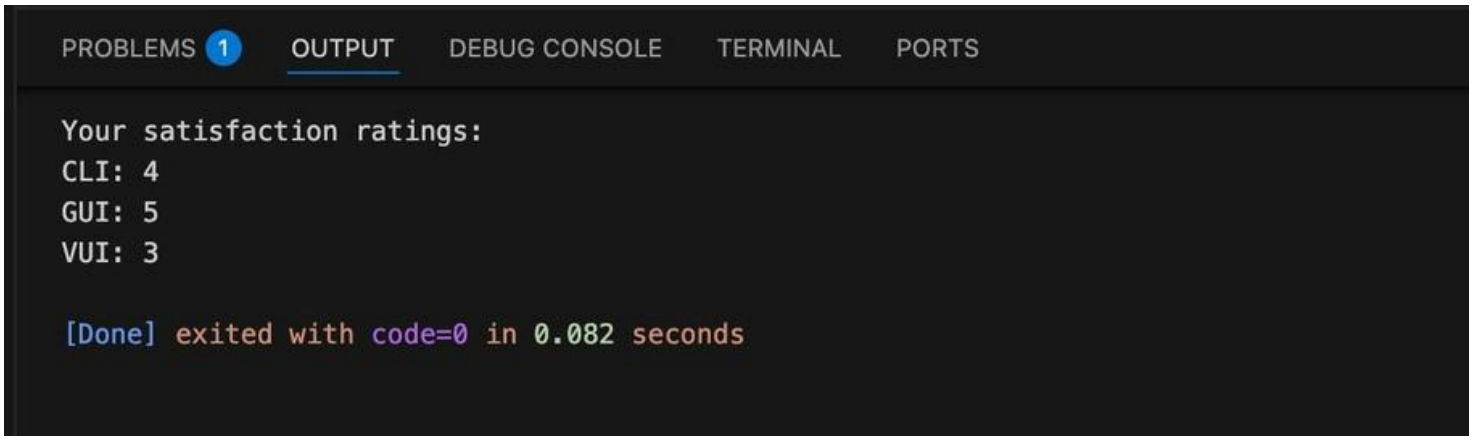
OUTPUT



COMPARE

```
def survey():
 cli_satisfaction = 4  gui_satisfaction = 5
            vui_satisfaction = 3

        print("\nYour    satisfaction    ratings:")        print(f"CLI:
{cli_satisfaction}")  print(f"GUI: {gui_satisfaction}")
          print(f"VUI: {vui_satisfaction}") survey()
```

OUTPUT


PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Your satisfaction ratings:
CLI: 4
GUI: 5
VUI: 3
[Done] exited with code=0 in 0.082 seconds