UID EX 3

Rama Thulasi G
230701259

Command Line Interface (CLI)

- A **Command Line Interface (CLI)** is a text-based interface where users interact with a computer by typing commands.
- It is commonly used by developers, system administrators, and advanced users to perform tasks efficiently.
- Very fast for experienced users who know commands.

Advantages:

- Faster for experienced users
- Uses fewer system resources

Disadvantages:

- Requires memorizing commands
- No visual elements

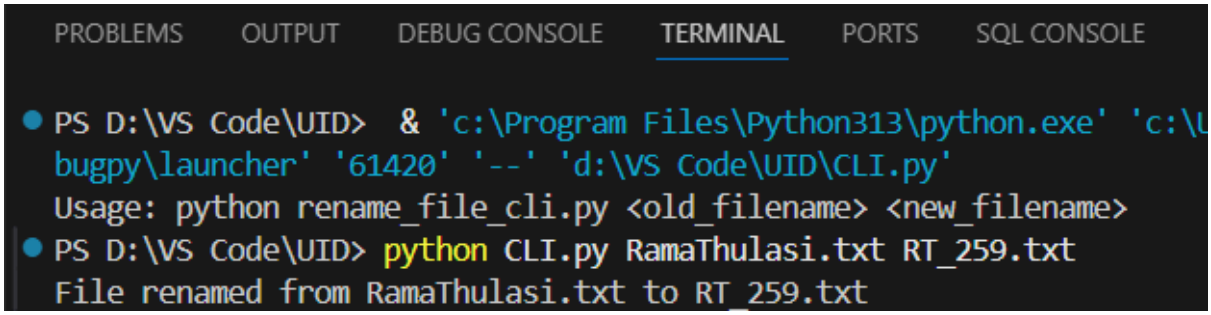
Implementation:

GUI.py | VUI.py | CLI.py | RamaThulasi.txt

CLI.py > rename_file

```python
import os
import sys

def  rename_file(old_name, new_name):
    try:
        os.rename(old_name, new_name)
        print(f"File renamed from {old_name} to {new_name}")
    except FileNotFoundError:
        print(f"Error: {old_name} 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:



Graphical User Interface (GUI)

- A **Graphical User Interface (GUI)** allows users to interact with a computer using visual elements such as windows, buttons, icons, and menus.
- It is designed for ease of use, making computers accessible to non-technical users.

Advantages:

- Easy to use, beginner-friendly
- Does not require memorizing commands
- More interactive with visuals

Disadvantages:

- Slower than CLI for advanced users
- Requires more system resources
- Limited automation capabilities

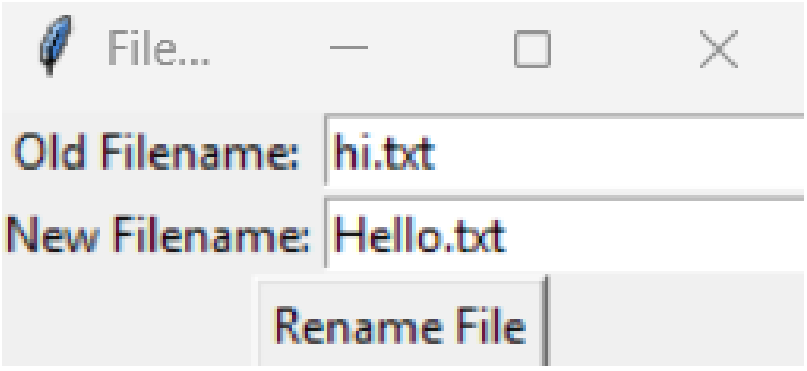
Implementation:

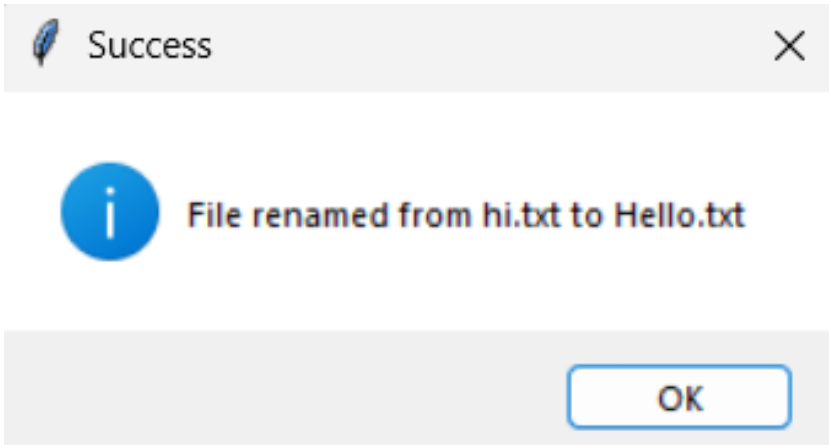
GUI.py | VUI.py | CLI.py | RamaThulasi.txt

GUI.py > rename_file

```python
import tkinter as tk
from tkinter import messagebox
import os

def rename_file():
    old_name = old_filename_entry.get()
    new_name = new_filename_entry.get()

    try:
        os.rename(old_name, new_name)
        messagebox.showinfo("Success", f"File renamed from {old_name} to {new_name}")
    except FileNotFoundError:
        messagebox.showerror("Error", f"File {old_name} not found.")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {e}")

# Set up the main window
root = tk.Tk()
root.title("File Renamer")

# Create and place labels, entries, and buttons
tk.Label(root, text="Old Filename:").grid(row=0, column=0)
tk.Label(root, text="New Filename:").grid(row=1, column=0)

old_filename_entry = tk.Entry(root)
old_filename_entry.grid(row=0, column=1)

new_filename_entry = tk.Entry(root)
new_filename_entry.grid(row=1, column=1)

rename_button = tk.Button(root, text="Rename File", command=rename_file)
rename_button.grid(row=2, columnspan=2)

# Start the Tkinter event loop
root.mainloop()

```

Output:





Voice User Interface (VUI)

- A **Voice User Interface (VUI)** allows users to interact with a computer or device using spoken commands.
- It is commonly used in smart assistants, voice-controlled applications, and accessibility features.

Advantages:

- Hands-free and convenient
- Easy to use for all ages
- Useful for people with disabilities

Disadvantages:

- Accuracy depends on speech recognition
- Struggles with different accents and background noise
- Limited to predefined commands

Implementation:

GUI.py | VUI.py | VU.py | CLI.py | RamaThulasi.txt

VU.py > ...

```python
import speech_recognition as sr
import os

def rename_file_from_voice_command(old_name, new_name):
    """Renames a file using the provided old and new names."""
    try:
        old_name += ".txt"
        new_name += ".txt"

        if not os.path.exists(old_name):
            print(f"❌ Error: '{old_name}' not found.")
            return

        os.rename(old_name, new_name)
        print(f"✅ File successfully renamed from '{old_name}' to '{new_name}'")
    except Exception as e:
        print(f"❌ Error: {e}")

def listen_for_filename(prompt):
    """Listens for a single filename input via voice command."""
    recognizer = sr.Recognizer()
    mic = sr.Microphone()

    with mic as source:
        recognizer.adjust_for_ambient_noise(source, duration=3)  # Increase noise adaptation
        print(f"🎤 {prompt}")
        try:
            audio = recognizer.listen(source, timeout=10, phrase_time_limit=5)  # Increased timeout
            command = recognizer.recognize_google(audio, language="en-US")
            print(f"📝 You said: {command}")
            return command.strip().replace(" ", "_")  # Replace spaces with underscores
        except sr.UnknownValueError:
            print("❌ Could not understand. Please try again.")
            return None
        except sr.WaitTimeoutError:
            print("⏳ Timeout: No speech detected. Try speaking louder and clearly.")
            return None
```

Output:

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
🎤 Say the new name for the file (without .txt)
📝 You said: welcome
✅ File successfully renamed from 'hello.txt' to 'welcome.txt'
PS D:\VS Code\UID>  d:; cd 'd:\VS Code\UID'; & 'c:\Program Files\Python313\python.
win32-x64\bundled\libs\debugpy\launcher' '54536' '--' 'd:\VS Code\UID\VU.py'
🔊 Welcome to the Voice-Controlled File Renamer!
🎤 Say the name of the file you want to rename (without .txt)
📝 You said: welcome
🎤 Say the new name for the file (without .txt)
📝 You said: hello
✅ File successfully renamed from 'welcome.txt' to 'hello.txt'
PS D:\VS Code\UID>
```