EXP NO 3

CLI (Command Line Interface)

For the **CLI**, you can provide a Python script that renames a file via command-line arguments. Here's how you could implement it:

Python Code

```
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])
```

GUI (Graphical User Interface)

In the **GUI**, we will use **Tkinter** to create a simple interface where the user can type the old and new filenames.

```
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")
```

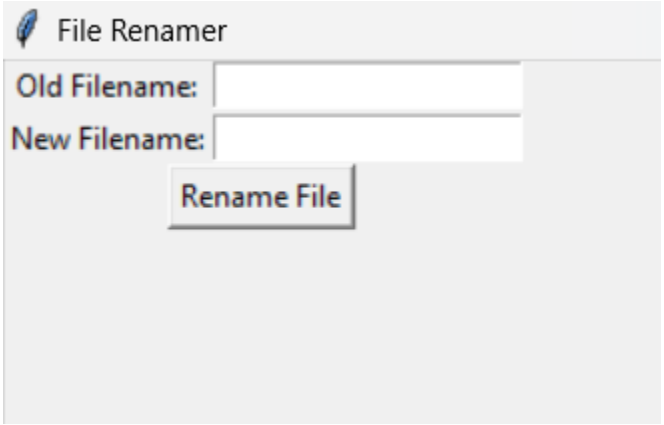
```
# Create and place labels, entries, and buttons
tk.Label(root, text="Old Filename:").grid(row=0, column=0)
tk.Label(root, text="New Filename:").grid(row=1, column=0)

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()
```



VUI (Voice User Interface)

For the **VUI**, we can use the **SpeechRecognition** library to listen for voice commands. The user will say something like, "Rename oldfilename.txt to newfilename.txt."

```
import speech_recognition as sr
import os


def rename_file_from_voice_command(command):
    # Extracting old and new filename from the command
    try:
        words = command.split(" ")
        old_name = words[1]
        new_name = words[3]


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


def listen_for_command():
    recognizer = sr.Recognizer()
    mic = sr.Microphone()


    print("Listening for command to rename a file...")
    with mic as source:
```

```
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        command = recognizer.recognize_google(audio)
        print(f"Command received: {command}")
        rename_file_from_voice_command(command)
    except sr.UnknownValueError:
        print("Sorry, I couldn't understand the command.")
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech
Recognition service; {e}")

if __name__ == "__main__":
    listen_for_command()
```