USER INTERFACE AND DESIGN

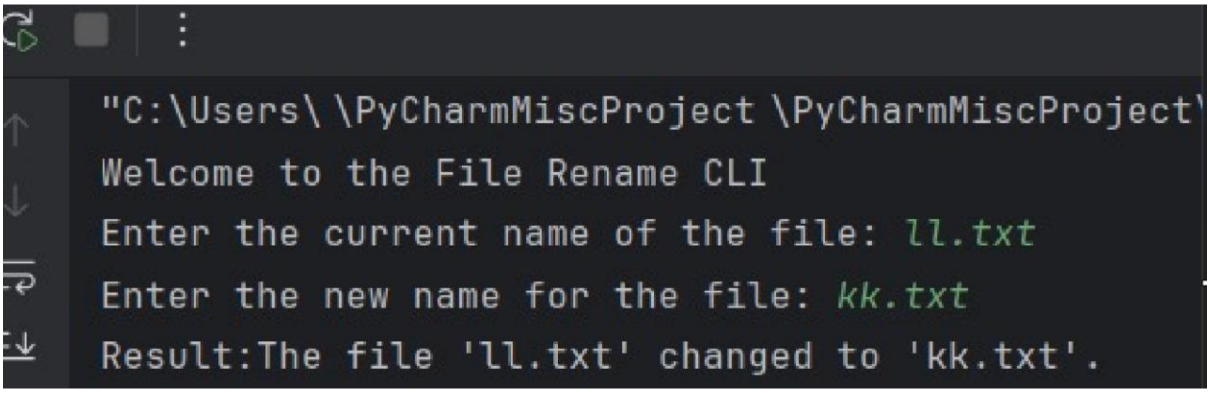
EXP. NO. : 3 Name: RAGUL A (230701252)

CLI(Command Line Interface):

Python code:

```
import os
def rename_file(old_name, new_name):
    try:
        os.rename(old_name, new_name)
        print(f"File renamed from '{old_name}' to '{new_name}' successfully.")
    except FileNotFoundError:
        print(f"Result:The file '{old_name}' changed to '{new_name}'.")
    except FileExistsError:
        print(f"Result: A file '{new_name}' is changed.")
    except Exception as e:
        print(f"An error occurred: {e}")
def main():
    print("Welcome to the File Rename CLI")
    old_name = input("Enter the current name of the file: ")
    new_name = input("Enter the new name for the file: ")
    rename_file(old_name, new_name)
if name == "main":
    main()
```

OUTPUT:



```
"C:\Users\ \PyCharmMiscProject \PyCharmMiscProject\
Welcome to the File Rename CLI
Enter the current name of the file: ll.txt
Enter the new name for the file: kk.txt
Result:The file 'll.txt' changed to 'kk.txt'.
```

USER INTERFACE AND DESIGN

GUI(Graphical User Interface):

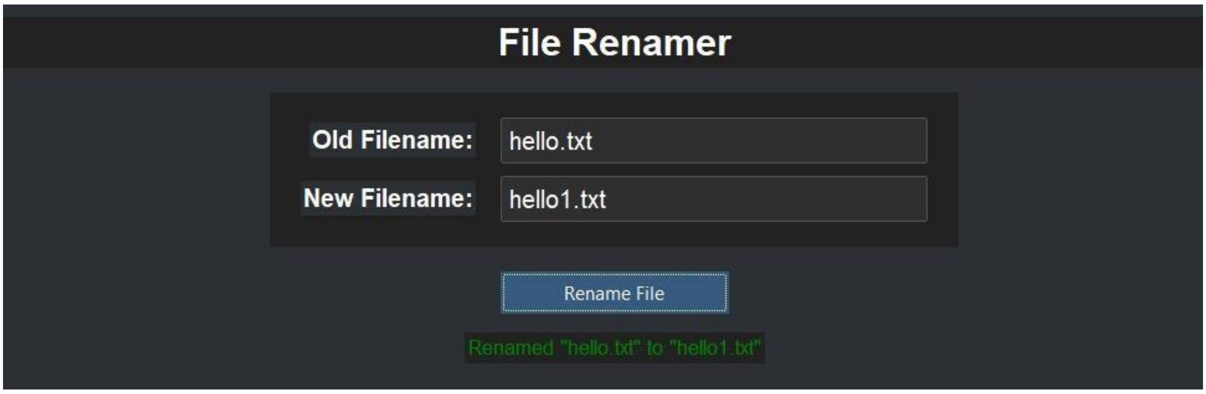
Python code:

```
import tkinter as tk
from tkinter import ttk
import ttkbootstrap as tb  # Modern UI theme
from PIL import Image, ImageTk  # For icons


def rename_file():
    old_name = old_file_entry.get()
    new_name = new_file_entry.get()
    if old_name and new_name:
        status_label.config(text=f'Renamed "{old_name}" to "{new_name}"', foreground="green")
    else:
        status_label.config(text="Please enter both filenames", foreground="red")


root = tb.Window(themename="darkly")  # Dark modern theme
root.title("File Renamer")
root.geometry("500x300")
root.resizable(False, False)
root.configure(bg="#2C2F33")  # Dark background


# Title Label
title_label = tk.Label(root, text="File Renamer", font=("Arial", 18, "bold"), fg="white",
bg="#23272A")
title_label.pack(fill="x",  pady=10)


# Input Frame
frame = tk.Frame(root, bg="#2C2F33", padx=15, pady=15)
frame.pack(pady=10)


old_file_label = ttk.Label(frame, text="Old Filename:", font=("Arial", 12, "bold"),
background="#2C2F33", foreground="white")
old_file_label.grid(row=0, column=0, padx=10, pady=5, sticky="e")
```

USER INTERFACE AND DESIGN

```
old_file_entry = ttk.Entry(frame, width=30, font=("Arial", 12))

old_file_entry.grid(row=0, column=1, padx=10, pady=5)


new_file_label = ttk.Label(frame, text="New Filename:", font=("Arial", 12, "bold"), background="#2C2F33", foreground="white")

new_file_label.grid(row=1, column=0, padx=10, pady=5, sticky="e")


new_file_entry = ttk.Entry(frame, width=30, font=("Arial", 12))

new_file_entry.grid(row=1, column=1, padx=10, pady=5)


# Rename Button

rename_button = tb.Button(root, text="Rename File", bootstyle="primary", command=rename_file, width=20)

rename_button.pack(pady=10)


# Status Label

status_label = tk.Label(root, text="", font=("Arial", 10), bg="#2C2F33", fg="white")

status_label.pack(pady=5)


root.mainloop()
```

OUTPUT:



File Renamer

Old Filename: hello.txt

New Filename: hello1.txt

Renamed "hello.txt" to "hello1.txt"

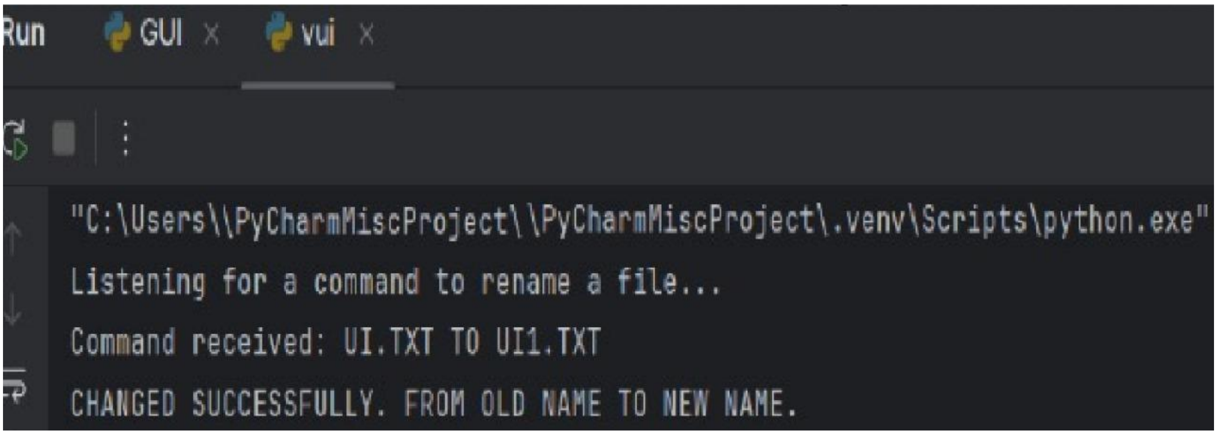
USER INTERFACE AND DESIGN

VUI(Voice User Interface):
Python code:

```
import speech_recognition as sr

import os


def rename_file(old_name, new_name):


    try:

        if not os.path.exists(old_name):

            print(f"Error: The file '{old_name}' does not exist.")

            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 renaming file: {e}")



def parse_voice_command(command):

    try:

        command = command.lower()

        if "rename" in command and "to" in command:

            words = command.split()

            old_name_index = words.index("rename") + 1

            new_name_index = words.index("to") + 1

            old_name = words[old_name_index]

            new_name = words[new_name_index]

            return old_name, new_name

        else:

            print("Invalid command format. Please use: 'Rename <old_name> to <new_name>'.")

            return None, None

    except Exception as e:

        print(f"Error parsing command: {e}")
        return None, None



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

    print("Listening for a command to rename a file...")
    with mic as source:
        recognizer.adjust_for_ambient_noise(source)
        try:
            audio = recognizer.listen(source, timeout=20)
        except sr.WaitTimeoutError:
            print("Listening timed out. No command detected.")
            return

    try:
        command = recognizer.recognize_google(audio)
        print(f"Command received: {command}")
        old_name, new_name = parse_voice_command(command)
        if old_name and new_name:
            rename_file(old_name, new_name)
    except sr.UnknownValueError:
        print("Sorry, I couldn't understand the command. Please try again.")
    except sr.RequestError as e:
        print(f"Could not connect to Google Speech Recognition service; {e}")



if _name_ == "_main_":
    listen_for_comand()
```

USER INTERFACE AND DESIGN

OUTPUT:



```
"C:\Users\\PyCharmMiscProject\\PyCharmMiscProject\.venv\Scripts\python.exe"
Listening for a command to rename a file...
Command received: UI.TXT TO UI1.TXT
CHANGED SUCCESSFULLY. FROM OLD NAME TO NEW NAME.
```