EXPERIMENT-3

NAME: JAI AAKASH S ROLL NO:240701198

1. COMMAND LINE INTERFACE:

PYTHON CODE:

```
FILE_NAME = "notes.txt"
notes = []

# Load notes from file
def load_notes():
    try:
        with open(FILE_NAME, "r") as file:
            for line in file:
                line = line.strip()
                if "|" in line:
                    title, content = line.split("|", 1)
                    notes.append({"title": title,
"content": content})
    except FileNotFoundError:
        pass


# Save notes to file
def save_notes():
    with open(FILE_NAME, "w") as file:
        for note in notes:
            file.write(note["title"] + "|" +
note["content"] + "\n")


def add_note():
    title = input("Enter note title: ")
    content = input("Enter note content: ")
    notes.append({"title": title, "content":
content})
    save_notes()
    print("Note added successfully!")


def view_notes():
    if not notes:
        print("No notes found.")
        return

    print("\nYour Notes:")
    for i, note in enumerate(notes, start=1):
```

```
            print(f"{i}. {note['title']} -
{note['content']}")


    def search_notes():
        keyword = input("Enter keyword to
search: ").lower()
        found = False

        for note in notes:
            if keyword in note["title"].lower() or
keyword in note["content"].lower():
                print(f"Found: {note['title']} -
{note['content']}")
                found = True

        if not found:
            print("No matching notes found.")


    def delete_note():
        view_notes()
        try:
            num = int(input("Enter note number to
delete: "))
            if 1 <= num <= len(notes):
                deleted = notes.pop(num - 1)
                save_notes()
                print(f"Deleted note:
{deleted['title']}")
            else:
                print("Invalid note number.")
        except ValueError:
            print("Please enter a valid number.")


    def menu():
        print("\n--- Advanced CLI Note Manager
---")
        print("1. Add Note")
        print("2. View Notes")
        print("3. Search Notes")
        print("4. Delete Note")
        print("5. Exit")


    load_notes()

    while True:
        menu()
```

```
    choice = input("Enter your choice: ")

    if choice == "1":
        add_note()
    elif choice == "2":
        view_notes()
    elif choice == "3":
        search_notes()
    elif choice == "4":
        delete_note()
    elif choice == "5":
        print("Exiting application. Goodbye!")
        break
    else:
        print("Invalid choice. Try again.")
```

OUTPUT:

```
C:\Users\karth\OneDrive\Desktop>py cliui.py

--- Advanced CLI Note Manager ---
1. Add Note
2. View Notes
3. Search Notes
4. Delete Note
5. Exit
Enter your choice: 1
Enter note title: Maths
Enter note content: complete probability and statistics
Note added successfully!
```

2. GRAPHICAL USER INTERFACE:

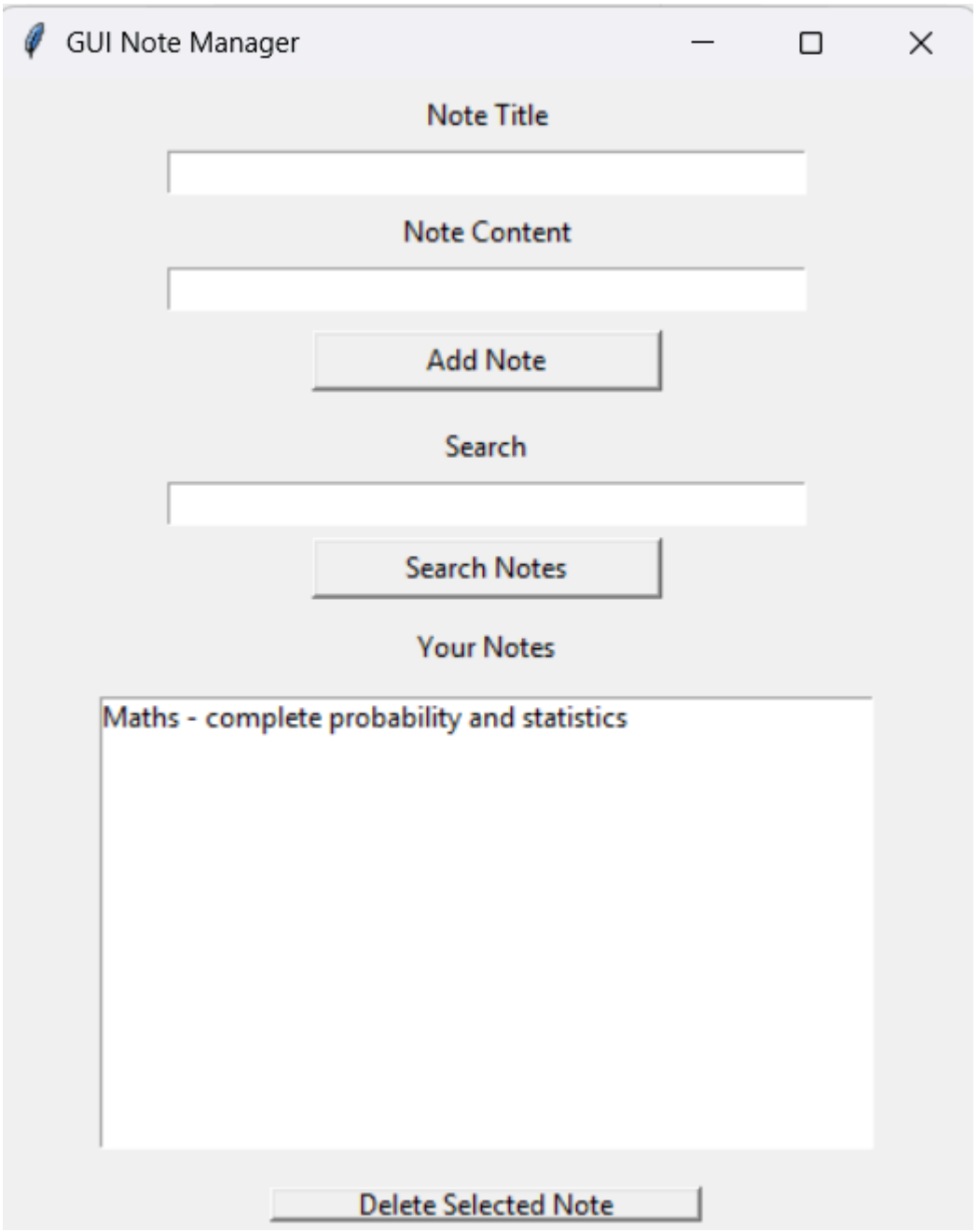
PYTHON CODE:

```
import tkinter as tk
from tkinter import messagebox

FILE_NAME = "notes.txt"
notes = []

# ---------- File Handling ----------
def load_notes():
    try:
        with open(FILE_NAME, "r") as f:
            for line in f:
                line = line.strip()
                if "|" in line:
                    title, content = line.split("|", 1)
                    notes.append({"title": title, "content": content})
    except FileNotFoundError:
        pass

def save_notes():
    with open(FILE_NAME, "w") as f:
        for n in notes:
            f.write(f"{n['title']}|{n['content']}\n")

# ---------- Actions ----------
def add_note():
    title = title_entry.get().strip()
    content = content_entry.get().strip()

    if not title or not content:
        messagebox.showwarning("Input Error", "Title and content are required.")
        return

    notes.append({"title": title, "content": content})
    save_notes()
    refresh_list()
    title_entry.delete(0, tk.END)
    content_entry.delete(0, tk.END)
    messagebox.showinfo("Success", "Note added successfully!")

def delete_note():
    selected = notes_list.curselection()
    if not selected:
        messagebox.showwarning("Selection Error", "Select a note to delete.")
        return

    index = selected[0]
    deleted = notes.pop(index)
    save_notes()
    refresh_list()
    messagebox.showinfo("Deleted", f"Deleted note: {deleted['title']}")
```

```
def search_notes():
    keyword = search_entry.get().lower().strip()
    notes_list.delete(0, tk.END)

    for n in notes:
        if keyword in n["title"].lower() or keyword in n["content"].lower():
            notes_list.insert(tk.END, f"{n['title']} - {n['content']}")

def refresh_list():
    notes_list.delete(0, tk.END)
    for n in notes:
        notes_list.insert(tk.END, f"{n['title']} - {n['content']}")

# ---------- UI ----------
root = tk.Tk()
root.title("GUI Note Manager")
root.geometry("420x500")

tk.Label(root, text="Note Title").pack(pady=5)
title_entry = tk.Entry(root, width=45)
title_entry.pack()

tk.Label(root, text="Note Content").pack(pady=5)
content_entry = tk.Entry(root, width=45)
content_entry.pack()

tk.Button(root, text="Add Note", width=20, command=add_note).pack(pady=8)

tk.Label(root, text="Search").pack(pady=5)
search_entry = tk.Entry(root, width=45)
search_entry.pack()
tk.Button(root, text="Search Notes", width=20, command=search_notes).pack(pady=5)

tk.Label(root, text="Your Notes").pack(pady=5)
notes_list = tk.Listbox(root, width=55, height=12)
notes_list.pack(pady=5)

tk.Button(root, text="Delete Selected Note", width=25, command=delete_note).pack(pady=10)

# ---------- Start ----------
load_notes()
refresh_list()
root.mainloop()
```

OUTPUT:

```
C:\Users\karth\OneDrive\Desktop>py gui.py
```



3. VOICE USER INTERFACE:

PYTHON CODE:

```
import speech_recognition
  as sr
import pyttsx3
import os


FILE_NAME = "vui_tasks.txt"
tasks = []
```

```
engine = pyttsx3.init()

def speak(text):
    engine.say(text)
    engine.runAndWait()

def listen():
    r = sr.Recognizer()
    with sr.Microphone() as
    source:
        speak("Listening")
        try:
            audio =
    r.listen(source, timeout=5)
            command =
    r.recognize_google(audio)
            print("You said:",
    command)
            return
    command.lower()
        except:
            speak("Sorry, I did
    not understand")
            return ""

def load_tasks():
    tasks.clear()
    if
    os.path.exists(FILE_NAME
    ):
        with open(FILE_NAME,
    "r") as f:
            for line in f:
```

```
    tasks.append(line.strip())

def save_tasks():
    with open(FILE_NAME,
    "w") as f:
        for task in tasks:
            f.write(task + "\n")

def add_task(command):
    task =
    command.replace("add
    task", "").strip()
    if task:
        tasks.append(f"{task} |
    Pending")
        save_tasks()
        speak("Task added
    successfully")
    else:
        speak("Please say the
    task name")

def
    remove_task(command):
    task =
    command.replace("remov
    e task", "").strip()
    for t in tasks:
        if task in t:
            tasks.remove(t)
            save_tasks()
            speak("Task
    removed")
```

```
            return
    speak("Task not found")

def list_tasks():
    if not tasks:
        speak("You have no
    tasks")
    else:
        speak("Your tasks are")
        for t in tasks:
            speak(t)

def
    complete_task(command)
    :
    task =
    command.replace("compl
    ete task", "").strip()
    for i, t in
    enumerate(tasks):
        if task in t:
            tasks[i] =
    t.replace("Pending",
    "Completed")
            save_tasks()
            speak("Task marked
    as completed")
            return
    speak("Task not found")

def main():
    load_tasks()
    speak("Voice task
    manager started")
```

```
    while True:
        command = listen()

        if "add task" in
    command:
            add_task(command)

        elif "remove task" in
    command:

    remove_task(command)

        elif "list tasks" in
    command:
            list_tasks()

        elif "complete task" in
    command:

    complete_task(command)
        elif "exit" in command
    or "stop" in command:
            speak("Goodbye")
            break

        else:
            speak("Please say a
    valid command")

main()
```

```
C:\Users\TCS\Desktop>python vui.py
Traceback (most recent call last):
  File "C:\Users\TCS\Desktop\vui.py", line 1, in <module>
    import speech_recognition as sr
ModuleNotFoundError: No module named 'speech_recognition'
```

4. USER SATISFACTION COMPARISON:

PYTHON CODE:

```
def survey():
    print("Rate your satisfaction with the following interfaces (1-5):")

    # Get user input for each interface
    try:
        cli_satisfaction = int(input("CLI (1-5): "))
        gui_satisfaction = int(input("GUI (1-5): "))
        vui_satisfaction = int(input("VUI (1-5): "))

        # Ensure valid ratings
        if not (1 <= cli_satisfaction <= 5 and 1 <= gui_satisfaction <= 5 and 1 <= vui_satisfaction <= 5):
            print("Please enter ratings between 1 and 5 only.")
            return

        # Display the ratings
        print("\nYour satisfaction ratings:")
        print(f"CLI: {cli_satisfaction}")
        print(f"GUI: {gui_satisfaction}")
```

```
        print(f"VUI: {vui_satisfaction}")

        # Calculate the average satisfaction
        avg_satisfaction = (cli_satisfaction + gui_satisfaction + vui_satisfaction) / 3
        print(f"\nAverage Satisfaction Score: {avg_satisfaction:.2f}")

    except ValueError:
        print("Invalid input! Please enter numbers between 1 and 5.")

# Run the survey function
if __name__ == "__main__":
    survey()
```

```
C:\Users\TCS\Desktop>python uii.py
Rate your satisfaction with the following interfaces (1-5):
CLI (1-5): 3
GUI (1-5): 5
VUI (1-5): 4

Your satisfaction ratings:
CLI: 3
GUI: 5
VUI: 4

Average Satisfaction Score: 4.00
```