RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#01

Lab Report: Introduction to HCI and Integration with Computer Graphics

Purpose of the Lab

The purpose of this lab was to introduce the fundamental concepts of Human-Computer Interaction (HCI) and explore how HCI integrates with Computer Graphics (CG) in modern technologies. The lab emphasized understanding user-centered design principles, evaluating design examples, discovering wireframing tools, and discussing the role of interaction in CG applications like virtual reality and video games.

Task 1: Introduction to HCI

1. Discussion

What is HCI and Its Importance?

Human-Computer Interaction (HCI) is the study and design of the interaction between people (users) and computers. It focuses on creating systems that are efficient, intuitive, and user-friendly. HCI is important

because it ensures that technology serves the needs of users, improves productivity, reduces errors, and enhances user satisfaction.

Key Components of HCI

- **User:** The person who interacts with the computer system.
- **Computer**: The system or device (hardware and software) with which the user interacts.
- **Interaction:** The communication between the user and the computer, involving inputs (e.g., clicking, typing) and outputs (e.g., displays, sounds).

Examples of Good and Bad HCI Designs

- **Good HCI Design:**
  - **Google Search:** Simple, fast, and intuitive interface.
  - **Spotify App:** Easy navigation, personalized playlists, clean design.
- **Bad HCI Design:**
  - **Old Banking Websites:** Complicated forms, confusing navigation.
  - **Overloaded Mobile Apps:** Apps with too many options and small buttons making them hard to use.

2. List of Wireframing Tools

- **Figma:** Cloud-based, collaborative design tool.
- **Adobe XD:** Popular for creating interactive wireframes and prototypes.
- **Sketch:** Mac-only tool known for vector UI design.
- **Balsamiq:** Low fidelity wireframing tool for rapid sketching.

- **InVision:** Focuses on prototyping and collaboration.

Task 2: Integration of HCI and CG

Discussion

How HCI and CG Work Together

In applications like virtual reality (VR), augmented reality (AR), and video games, HCI ensures that user interactions are natural and intuitive, while CG creates visually appealing and realistic environments. Good HCI design enables users to easily control and navigate CG worlds, making the experience more immersive and engaging.

Role of User Interaction in CG Applications:

User interaction is crucial in CG-based applications. It determines how users:

- Move within virtual spaces (e.g., VR headsets, motion controllers).
- Interact with digital objects (e.g., touchscreens, gestures).
- Receive feedback (e.g., visual, audio, or haptic).

Post-Lab Questions

1. Key Differences Between HCI and CG

- HCI focuses on how users interact with technology.
- CG focuses on how visual content is generated and displayed.
- **Complement:** HCI ensures users can effectively interact with CG environments, while CG provides the visual feedback users need.

2. Why is Usability Important in HCI?

Usability ensures that systems are easy to learn, efficient to use, and satisfying.

Examples:

- **Good Usability: Google Maps:** intuitive navigation, clear directions.
- **Bad Usability:** Overcomplicated e-commerce checkout pages leading to abandoned purchases.

3. Challenges of Creating Realistic Computer Graphics

- **Challenges:**
    - High computational cost.
    - Realistic lighting, textures, and physics simulations.
    - Achieving smooth, real-time rendering.
- **Impact:** Poor graphics realism can break immersion and frustrate users, negatively affecting user experience.

4. Reflection on Tools Used

- **Easy:** Using Figma's drag-and-drop interface to create wireframes.
- **Difficult:** Learning advanced features like interactive prototyping or component libraries in Figma and Adobe XD.

Conclusion

This lab highlighted the importance of designing user-centered systems and the crucial role of integrating HCI with CG in modern digital experiences. The practice with wireframing tools and discussions about real-world applications provided a strong foundation for understanding how to create effective, engaging user interfaces.

THE END

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#02

Lab Report: HCI Design Observations, Analysis, and Redesign

Purpose of the Lab

The purpose of this lab was to observe real-world examples of good and bad designs, critically evaluate website usability based on HCI principles, practice redesigning poor interfaces, and present findings to improve our understanding of user-centered design.

Activity 1: Observing Real-World Designs

Table Comparing Good and Bad Designs

Aspect	Good Design Example	Bad Design Example
Device/Interface	iPhone Home Screen	Overly Complex TV Remote
Positive Features	Simple icons, consistent layout, intuitive gestures	Too many buttons, unclear labeling, hard to operate
Device/Interface	Google Search Website	Confusing Elevator Button Panel

Positive Features	Minimalist interface, clear search focus, quick feedback	No clear labeling, confusing floor numbers
Device/Interface	ATM at Modern Bank	ATM at Old Train Station
Positive Features	Touchscreen with step-by-step guidance	Poorly labeled buttons, slow response times

Activity 2: Analyzing Website Usability

Selected Websites

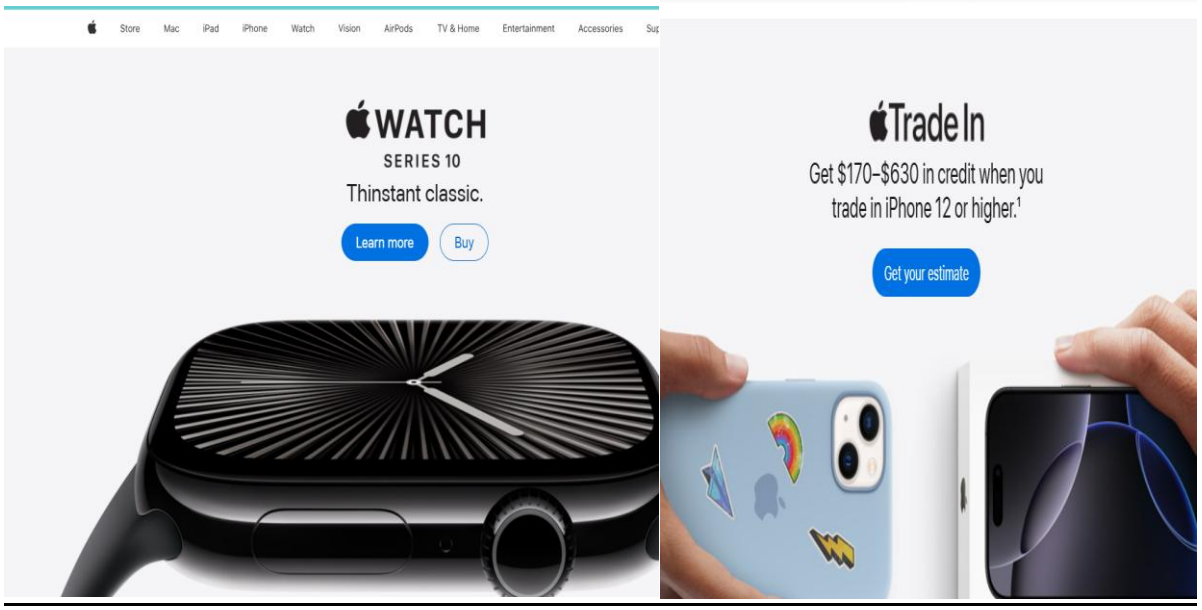
- **Good Website: Apple**
- **Bad Website: Arngren.net**

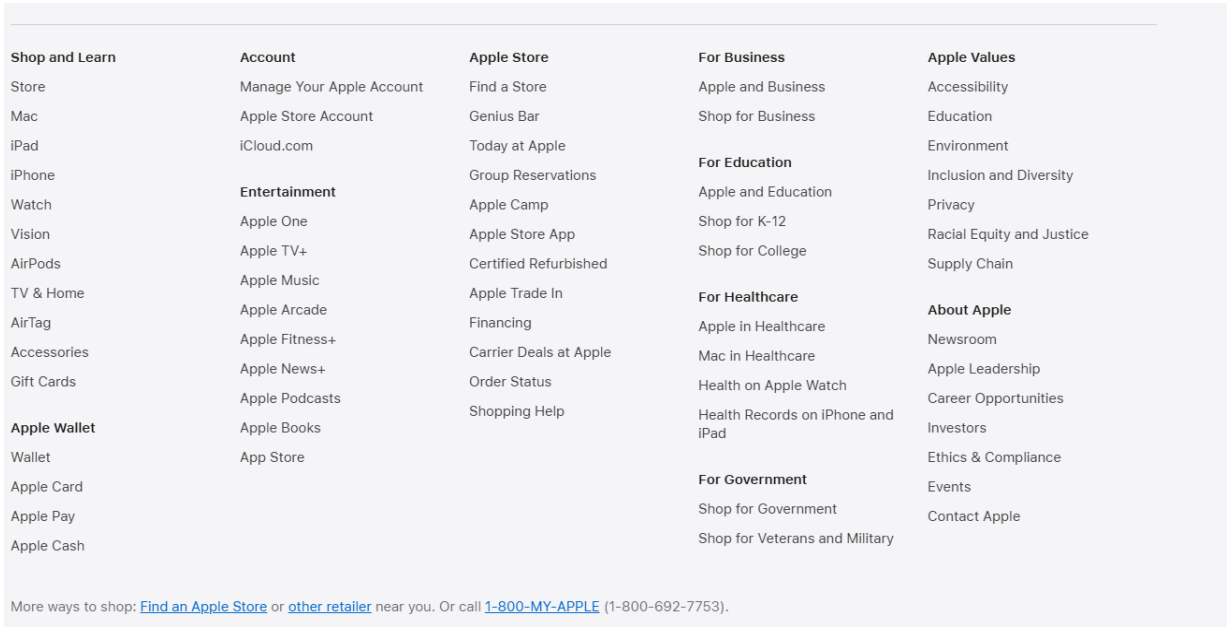
Analysis Report

Apple Website (Good Design):

- **Navigation: Clean and easy to navigate, menus are simple and logical.**
- **Visual Design:** Minimalist aesthetic, strong use of white space.
- **Responsiveness:** Optimized for both mobile and desktop.
- **Feedback:** Quick responses to user actions like clicking a product.
- **Accessibility:** Good use of alt text, color contrast, and readability.

Screenshot Example:





Shop and Learn
Store
Mac
iPad
iPhone
Watch
Vision
AirPods
TV & Home
AirTag
Accessories
Gift Cards

Apple Wallet
Wallet
Apple Card
Apple Pay
Apple Cash

Account
Manage Your Apple Account
Apple Store Account
iCloud.com

Entertainment
Apple One
Apple TV+
Apple Music
Apple Arcade
Apple Fitness+
Apple News+
Apple Podcasts
Apple Books
App Store

Apple Store
Find a Store
Genius Bar
Today at Apple
Group Reservations
Apple Camp
Apple Store App
Certified Refurbished
Apple Trade In
Financing
Carrier Deals at Apple
Order Status
Shopping Help

For Business
Apple and Business
Shop for Business

For Education
Apple and Education
Shop for K-12
Shop for College

For Healthcare
Apple in Healthcare
Mac in Healthcare
Health on Apple Watch
Health Records on iPhone and iPad

For Government
Shop for Government
Shop for Veterans and Military

Apple Values
Accessibility
Education
Environment
Inclusion and Diversity
Privacy
Racial Equity and Justice
Supply Chain

About Apple
Newsroom
Apple Leadership
Career Opportunities
Investors
Ethics & Compliance
Events
Contact Apple

More ways to shop: Find an Apple Store or other retailer near you. Or call 1-800-MY-APPLE (1-800-692-7753).

Arngren.net (Bad Design)

- **Navigation:** Extremely cluttered; too many links without organization.
- **Visual Design:** Chaotic layout, no clear visual hierarchy.
- **Responsiveness:** Not optimized for modern devices, difficult to read on mobile.
- **Feedback:** Clicking links often results in confusing behavior.

- **Accessibility:** Poor text contrast, small font size, no structure.

Screenshot Example:

Summary

Apple's website follows HCI principles by making interaction clear, simple, and satisfying, whereas Arngren.net overwhelms the user, leading to frustration and confusion.

Activity 3: Redesign Exercise

Selected Interface

- **Problematic Interface:** Confusing Microwave Control Panel

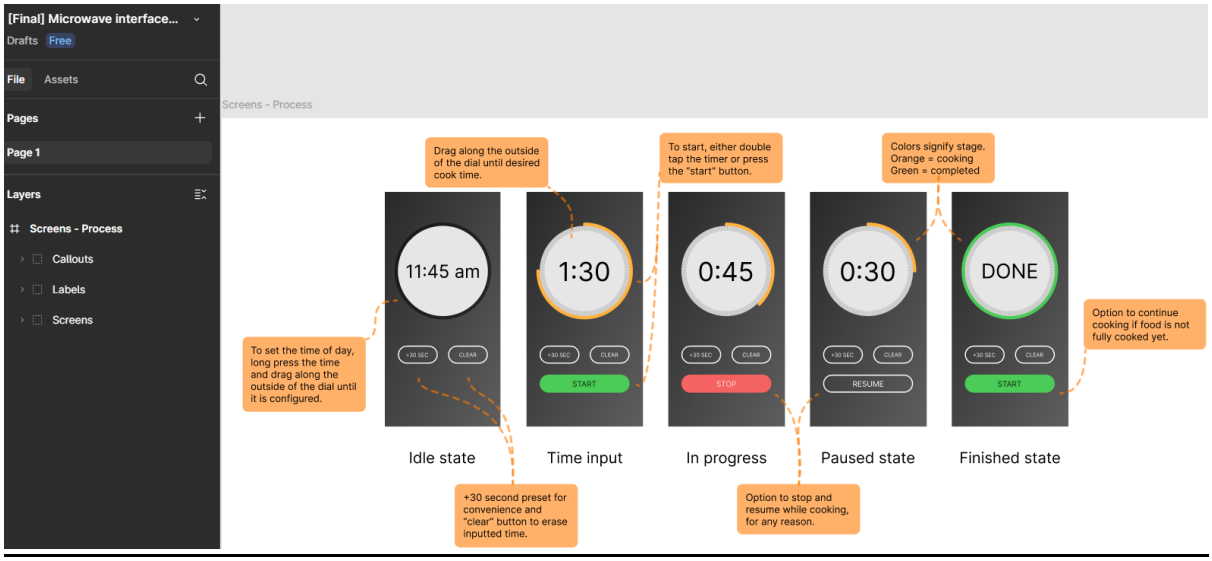
Original Design Photo:

Issues Identified

- Buttons are poorly labeled (e.g., "Auto Cook 1" without explanation).
- Overwhelming number of small, similar buttons.
- Important features (e.g., Start/Stop) are not highlighted.

Redesign Proposal

Sketch/Redesigned Interface:



Redesign Improvements

- Clear labeling ("Start", "Stop", etc.).
- Group related functions (cooking presets together, timer functions separately).
- Use larger buttons for frequent actions.
- Highlight critical buttons with color or size.

HCI Principles Applied:

- **Visibility:** Important functions are easy to find.
- **Affordance:** Buttons are labeled clearly so their purpose is obvious.
- **Feedback:** Display gives immediate confirmation of input (e.g., "Cooking Started").

Activity 4: Group Discussion and Presentation

Discussion Points

- **Common Mistakes in Bad Designs:**
    - Lack of clarity.

    - Overloaded interfaces.
    - No clear feedback or instructions.
- **How Good Design Improves Experience:**
    - Increases user satisfaction.
    - Reduces errors and frustration.
    - Enhances speed and efficiency.
- **Trade-offs Between Simple and Feature-Rich Interfaces:**
    - **Too simple:** Might limit advanced users.
    - **Too complex:** Might overwhelm beginners.
    - **Ideal:** Balance by offering basic and advanced modes (e.g., "simple view" and "advanced settings").

References

- Don Norman, *The Design of Everyday Things*.
- Steve Krug, *Don't Make Me Think*.
- Tools used: Figma, Adobe XD.

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#03

1. WIMP Interface: Task Manager (Tkinter)

Code:

```
import tkinter as tk
from tkinter import simpledialog, messagebox, filedialog
import json

tasks = []

def add_task():
    title = simpledialog.askstring("Add Task", "Enter task title:")
    deadline = simpledialog.askstring("Add Task", "Enter deadline (e.g., 2025-05-10):")
    if title and deadline:
        tasks.append({"title": title, "deadline": deadline})
        refresh_tasks()

def edit_task():
    selected = task_listbox.curselection()
```

```
    if selected:
        index = selected[0]
        task = tasks[index]
        new_title = simpledialog.askstring("Edit Task", "Edit title:",
initialvalue=task["title"])
        new_deadline = simpledialog.askstring("Edit Task", "Edit deadline:",
initialvalue=task["deadline"])
        if new_title and new_deadline:
            tasks[index] = {"title": new_title, "deadline": new_deadline}
            refresh_tasks()

def delete_task():
    selected = task_listbox.curselection()
    if selected:
        index = selected[0]
        tasks.pop(index)
        refresh_tasks()

def refresh_tasks():
    task_listbox.delete(0, tk.END)
    for task in tasks:
        task_listbox.insert(tk.END, f"{task['title']}- Due: {task['deadline']}")

def save_tasks():
    file_path = filedialog.asksaveasfilename(defaultextension=".json")
    if file_path:
        with open(file_path, "w") as f:
            json.dump(tasks, f)
```

```
        messagebox.showinfo("Saved", "Tasks saved successfully.")

def load_tasks():
    file_path = filedialog.askopenfilename(filetypes=[("JSON files", "*.json")])
    if file_path:
        with open(file_path, "r") as f:
            loaded_tasks = json.load(f)
            tasks.clear()
            tasks.extend(loaded_tasks)
            refresh_tasks()

# Setup window
root = tk.Tk()
root.title("Task Manager (WIMP Interface)")
root.geometry("400x400")

# Menu bar
menu_bar = tk.Menu(root)
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Save", command=save_tasks)
file_menu.add_command(label="Load", command=load_tasks)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)
menu_bar.add_cascade(label="File", menu=file_menu)
root.config(menu=menu_bar)

# Task list
task_listbox = tk.Listbox(root, font=("Arial", 12))
```

task_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

Buttons

btn_frame = tk.Frame(root)

btn_frame.pack(pady=10)

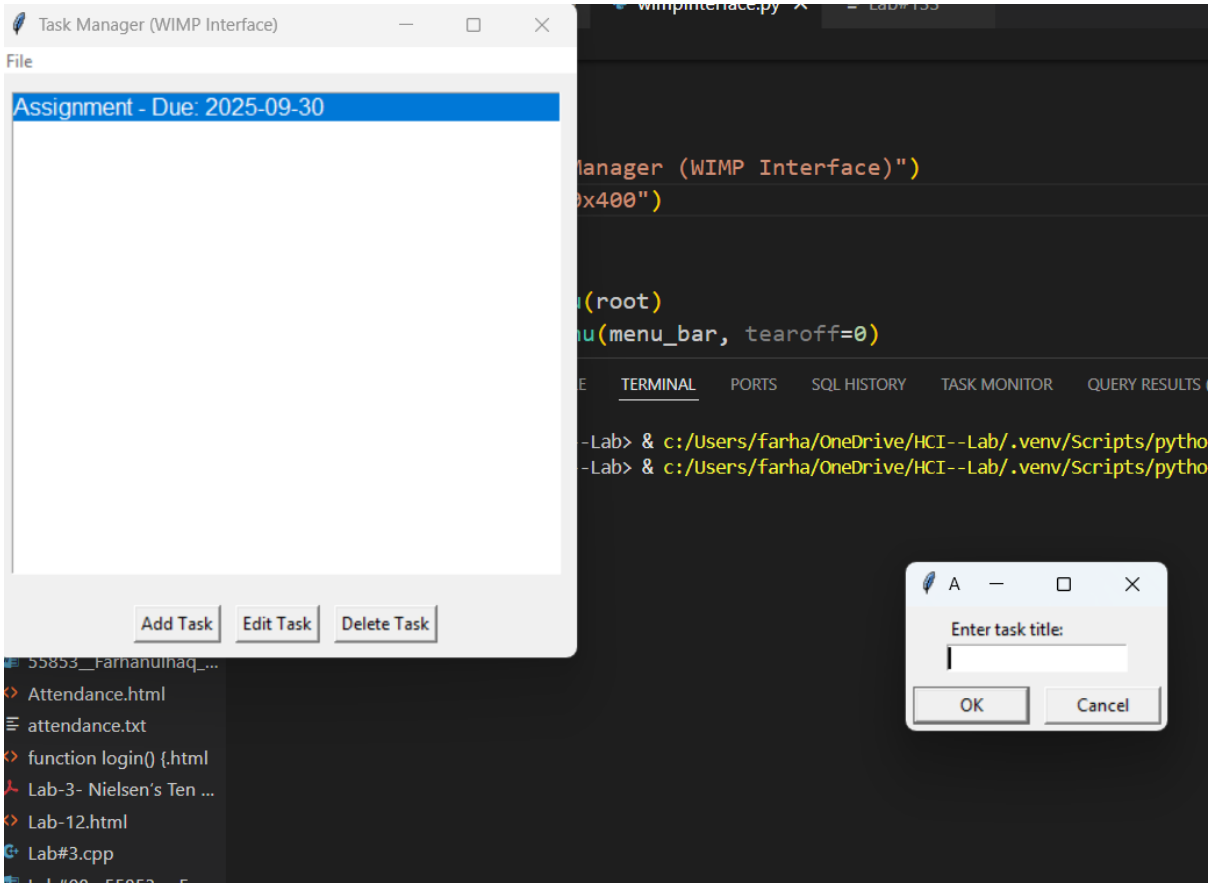
tk.Button(btn_frame, text="Add Task", command=add_task).grid(row=0, column=0, padx=5)

tk.Button(btn_frame, text="Edit Task", command=edit_task).grid(row=0, column=1, padx=5)

tk.Button(btn_frame, text="Delete Task", command=delete_task).grid(row=0, column=2, padx=5)

root.mainloop()

Interface:



2. Direct Manipulation Enhancements:

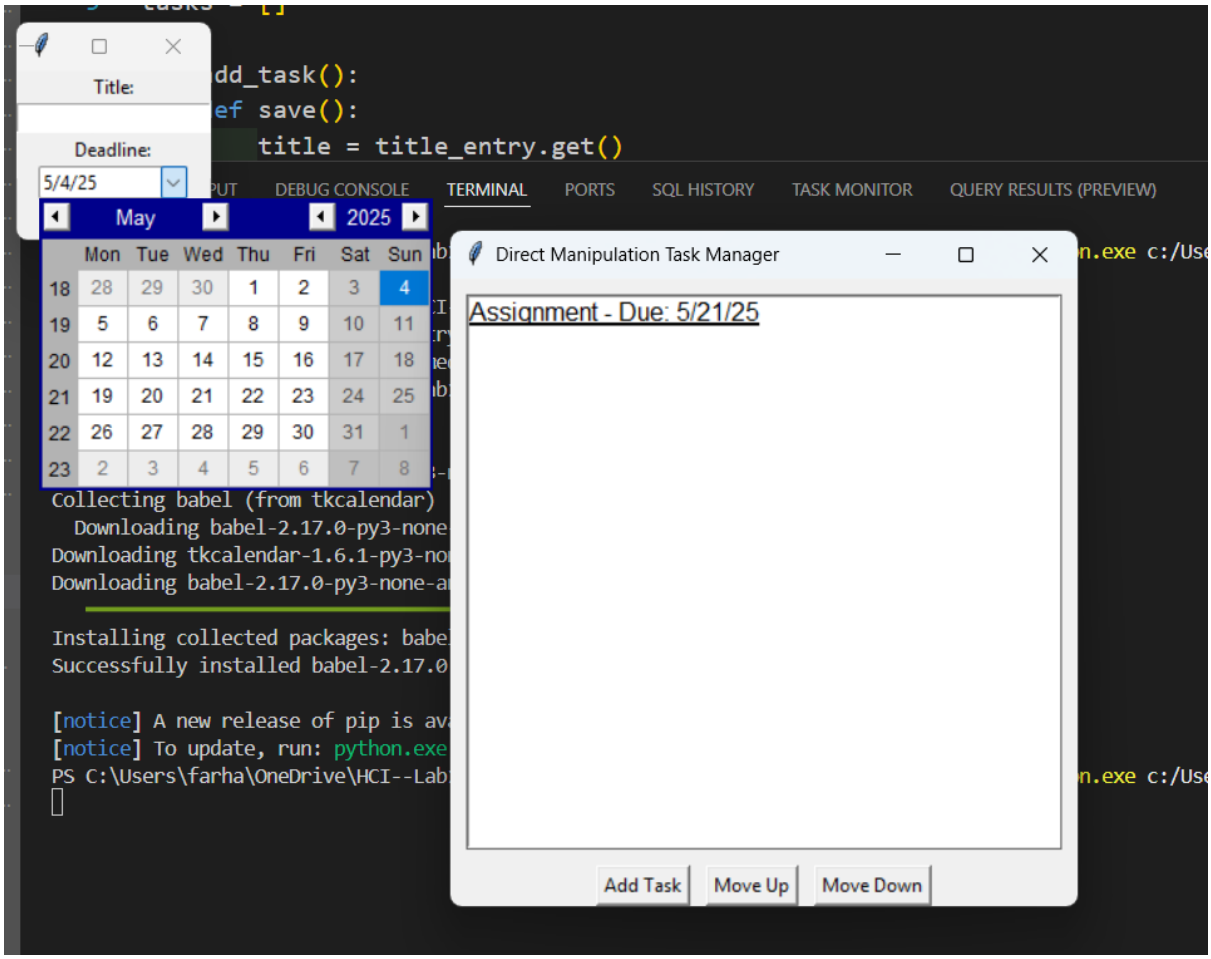
Code:

```
import tkinter as tk
from tkinter import simpledialog, messagebox
from tkcalendar import DateEntry

tasks = []

def add_task():
    def save():
        title = title_entry.get()
        deadline = date_entry.get()
        if title:
            tasks.append({'title': title, 'deadline': deadline})
            refresh_tasks()
            popup.destroy()

    popup = tk.Toplevel(root)
    popup.title("Add Task")
    tk.Label(popup, text="Title:").pack()
    title_entry = tk.Entry(popup)
    title_entry.pack()
    tk.Label(popup, text="Deadline:").pack()
    date_entry = DateEntry(popup, width=12, background='darkblue',
                foreground='white', borderwidth=2)
    date_entry.pack()
    tk.Button(popup, text="Save", command=save).pack()
```

```
def move_up():
    index = task_listbox.curselection()
    if index and index[0] > 0:
        tasks[index[0] - 1], tasks[index[0]] = tasks[index[0]], tasks[index[0] - 1]
        refresh_tasks()
        task_listbox.selection_set(index[0] - 1)

def move_down():
    index = task_listbox.curselection()
    if index and index[0] < len(tasks) - 1:
        tasks[index[0] + 1], tasks[index[0]] = tasks[index[0]], tasks[index[0] + 1]
        refresh_tasks()
        task_listbox.selection_set(index[0] + 1)

def refresh_tasks():
    task_listbox.delete(0, tk.END)
    for task in tasks:
        task_listbox.insert(tk.END, f"{task['title']} - Due: {task['deadline']}")

root = tk.Tk()
root.title("Direct Manipulation Task Manager")
root.geometry("400x400")

task_listbox = tk.Listbox(root, font=("Arial", 12))
task_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

btn_frame = tk.Frame(root)
```

```
btn_frame.pack()


tk.Button(btn_frame, text="Add Task", command=add_task).grid(row=0, column=0,
padx=5)
tk.Button(btn_frame, text="Move Up", command=move_up).grid(row=0, column=1,
padx=5)
tk.Button(btn_frame, text="Move Down", command=move_down).grid(row=0,
column=2, padx=5)


root.mainloop()
```

Output:



3.CLI Calculator — Add More Operations

Here we add exponentiation and modulus operations to the CLI calculator:

Code:

```
def cli_calculator():
    print("Welcome to the Extended CLI Calculator!")
    while True:
        print("\nAvailable Commands: add, subtract, multiply, divide, exponent, modulus, quit")
        command = input("Enter a command: ").strip().lower()
        if command == "quit":
            print("Exiting the calculator. Goodbye!")
            break
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
            if command == "add":
                print(f"Result: {num1 + num2}")
            elif command == "subtract":
                print(f"Result: {num1 - num2}")
            elif command == "multiply":
                print(f"Result: {num1 * num2}")
            elif command == "divide":
                print(f"Result: {num1 / num2}")
            elif command == "exponent":
                print(f"Result: {num1 ** num2}")
            elif command == "modulus":
                print(f"Result: {num1 % num2}")
```

```
            else:
                print("Invalid command. Try again.")
        except ValueError:
            print("Invalid input. Please enter numbers.")
        except ZeroDivisionError:
            print("Error: Division by zero.")
cli_calculator()
```

Output:

```
Welcome to the Extended CLI Calculator!

Available Commands: add, subtract, multiply, divide, exponent, modulus, quit
Enter a command: add
Enter first number: 45
Enter second number: 4
Result: 49.0

Available Commands: add, subtract, multiply, divide, exponent, modulus, quit
Enter a command:
```

---

4. Menu-Driven To-Do List — Mark Tasks as Completed

Here we add an option to mark tasks as completed:

Code:

```
def menu_driven_todo():
    tasks = []  # List to store tasks with completed status
    while True:
        print("\n--- To-Do List Menu ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Mark Task as Completed")
```

```
print("4. Delete Task")
print("5. Quit")

choice = input("Enter your choice: ").strip()

if choice == "1":
    task = input("Enter the task: ")
    tasks.append({"title": task, "completed": False})
    print(f"Task '{task}' added.")

elif choice == "2":
    if tasks:
        print("\nTasks:")
        for i, task in enumerate(tasks, 1):
            status = "✓" if task["completed"] else "✗"
            print(f"{i}. [{status}] {task['title']}")
    else:
        print("No tasks found.")

elif choice == "3":
    if tasks:
        task_num = int(input("Enter the task number to mark as completed: "))
        if 1 <= task_num <= len(tasks):
            tasks[task_num - 1]["completed"] = True
            print(f"Task '{tasks[task_num - 1]['title']}' marked as completed.")
        else:
            print("Invalid task number.")
    else:
```

```
            print("No tasks to mark.")

        elif choice == "4":
            if tasks:
                task_num = int(input("Enter the task number to delete: "))
                if 1 <= task_num <= len(tasks):
                    removed_task = tasks.pop(task_num - 1)
                    print(f"Task '{removed_task['title']}' deleted.")
                else:
                    print("Invalid task number.")
            else:
                print("No tasks to delete.")

        elif choice == "5":
            print("Exiting the to-do list. Goodbye!")
            break

        else:
            print("Invalid choice. Try again.")

menu_driven_todo()
```

Output:

```
Output

--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 1
Enter the task: Assignment
Task 'Assignment' added.

--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 2

Tasks:
1. [X] Assignment

--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 5
Exiting the to-do list. Goodbye!
```

5.Form-Based Interface — Complete Registration Form

Here we extend the user registration form with more fields and validation:

Code:

```
def form_based_registration():
    print("--- User Registration Form ---")
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    age = input("Enter your age: ")
    password = input("Create a password: ")

    # Basic Validation
    if not name or not email or not age or not password:
        print("Error: All fields are required.")
        return

    if "@" not in email or "." not in email:
        print("Error: Invalid email format.")
        return

    try:
        age = int(age)
        if age <= 0:
            print("Error: Age must be positive.")
            return
    except ValueError:
        print("Error: Age must be a number.")
        return

    print("\nRegistration Successful!")
```

```
    print(f"Name: {name}")
    print(f"Email: {email}")
    print(f"Age: {age}")
    # Note: For security reasons, we should not print password details.

form_based_registration()
```

Output:

```
Output
--- User Registration Form ---
Enter your name: Farhan-Ul-Haq
Enter your email: F.ulhaq@gmail.com
Enter your age: 19
Create a password: Farhan123

Registration Successful!
Name: Farhan-Ul-Haq
Email: F.ulhaq@gmail.com
Age: 19

=== Code Execution Successful ===
```

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#04

Lab-4: Cognitive Processes in Human-Computer Interaction (HCI)

Activity 1: Attention and Distraction in User Interfaces

Objective:

Show how random pop-ups affect reading attention.

Tools:

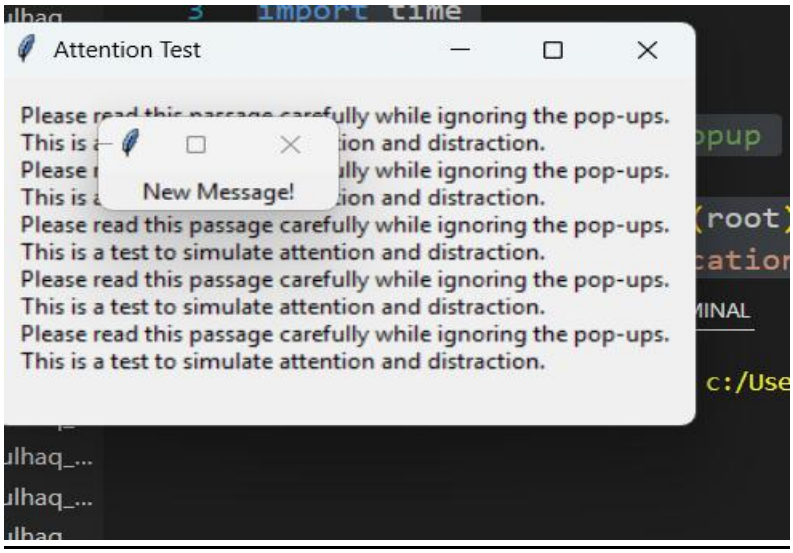
- Python (tkinter, threading, time, random)

Code:

```
import tkinter as tk
import threading
import time
import random


# Function to display popup
def show_popup():
    popup = tk.Toplevel(root)
```

```
    popup.title("Notification")
    tk.Label(popup, text="New Message!").pack()
    popup.after(2000, popup.destroy)  # auto close
# Background thread for random popups
def random_popups():
    while True:
        time.sleep(random.randint(3, 8))
        root.after(0, show_popup)
# Main Window
root = tk.Tk()
root.title("Attention Test")
tk.Label(root, text="Please read this passage carefully while ignoring the pop-ups.\n"
                    "This is a test to simulate attention and distraction.\n" * 5,
         wraplength=400, justify='left').pack(padx=10, pady=10)
# Start popup thread
threading.Thread(target=random_popups, daemon=True).start()


root.mainloop()
```

Output:



Expected Outcome:

- Students understand how random pop-ups cause distractions.

Activity 2: Perception and Recognition in UI Design

Objective:

Understand how color and icon design affect recognition.

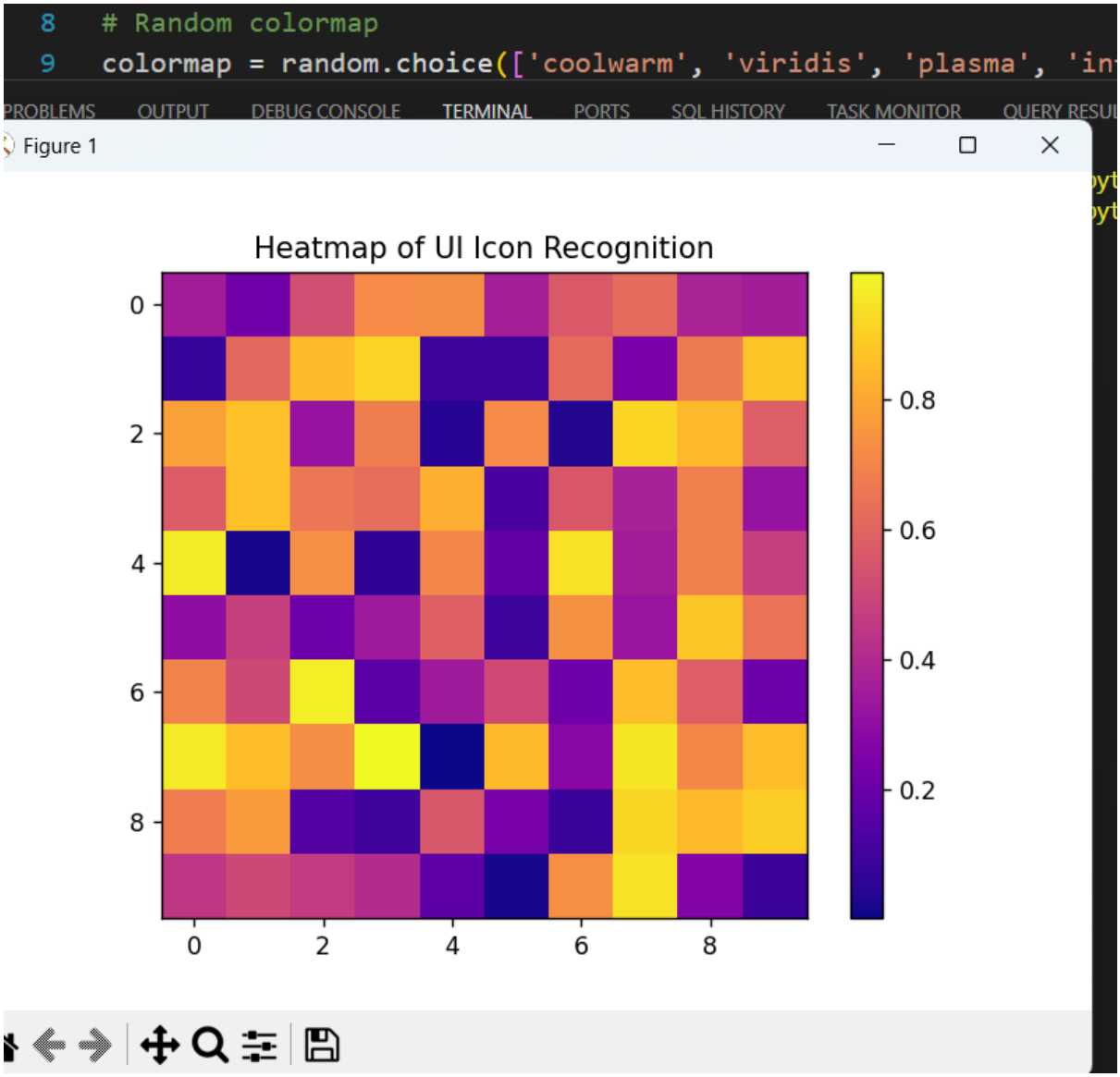
Tools:

- Python (matplotlib, numpy, random)

Code Implementation:

```
import matplotlib.pyplot as plt
import numpy as np
import random

# Simulated recognition pattern data
data = np.random.rand(10, 10)

# Random colormap
colormap = random.choice(['coolwarm', 'viridis', 'plasma', 'inferno'])

plt.imshow(data, cmap=colormap)
plt.colorbar()
plt.title("Heatmap of UI Icon Recognition")
plt.show()
```

Output:



Expected Outcome:

- Understand how proper color schemes and icon design improve recognition speed.

Activity 3: Memory and Automation in User Interfaces

Objective:

Compare manual vs. automated form filling.

Tools:

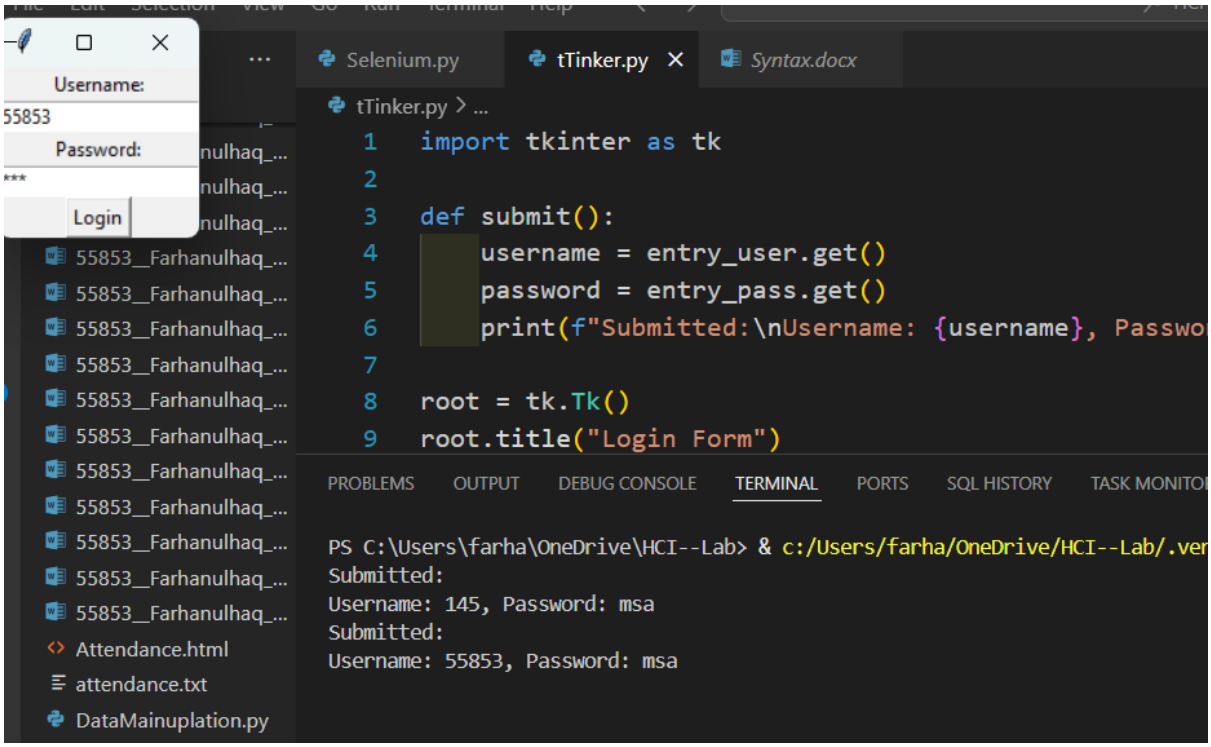
- Python (selenium, time)
- Tkinter (for manual form filling simulation)

Part A – Manual Tkinter UI import time:

```
import tkinter as tk


def submit():
    username = entry_user.get()
    password = entry_pass.get()
    print(f"Submitted:\nUsername: {username}, Password: {password}")


root = tk.Tk()
root.title("Login Form")


tk.Label(root, text="Username:").pack()
entry_user = tk.Entry(root)
entry_user.pack()


tk.Label(root, text="Password:").pack()
entry_pass = tk.Entry(root, show='*')
entry_pass.pack()


tk.Button(root, text="Login", command=submit).pack()
```

root.mainloop()

Output:



Part B – Automated Form Filling using Selenium:

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import time

Initialize the Chrome WebDriver

driver = webdriver.Chrome()

Open a login page (example site)

driver.get("https://example.com/login")

time.sleep(2) # Wait for page to load

Locate username and password fields

```
username = driver.find_element("name", "username")
password = driver.find_element("name", "password")

# Enter login credentials
username.send_keys("testuser")
password.send_keys("testpassword")
password.send_keys(Keys.RETURN)  # Submit the form

time.sleep(2)  # Wait after submission

driver.quit()  # Close the browser
```

```
log_data = []  # List to store task completion times

for i in range(5):
    start_time = time.time()
    input(f"Task {i+1}: Perform the action and press Enter...")
    end_time = time.time()
    log_data.append({'Task': i+1, 'Time Taken (s)': end_time - start_time})

# Create DataFrame and display results
df = pd.DataFrame(log_data)
print(df)
```

Expected Outcome:

- Students realize how auto-filling forms reduces cognitive load and saves time.
- Students suggest UI features like remember me checkboxes, form autofill, and credential managers.

Activity 4: Learning and Skill Acquisition in Software Use

Objective:

Track how users improve over tasks using time logs.

Tools:

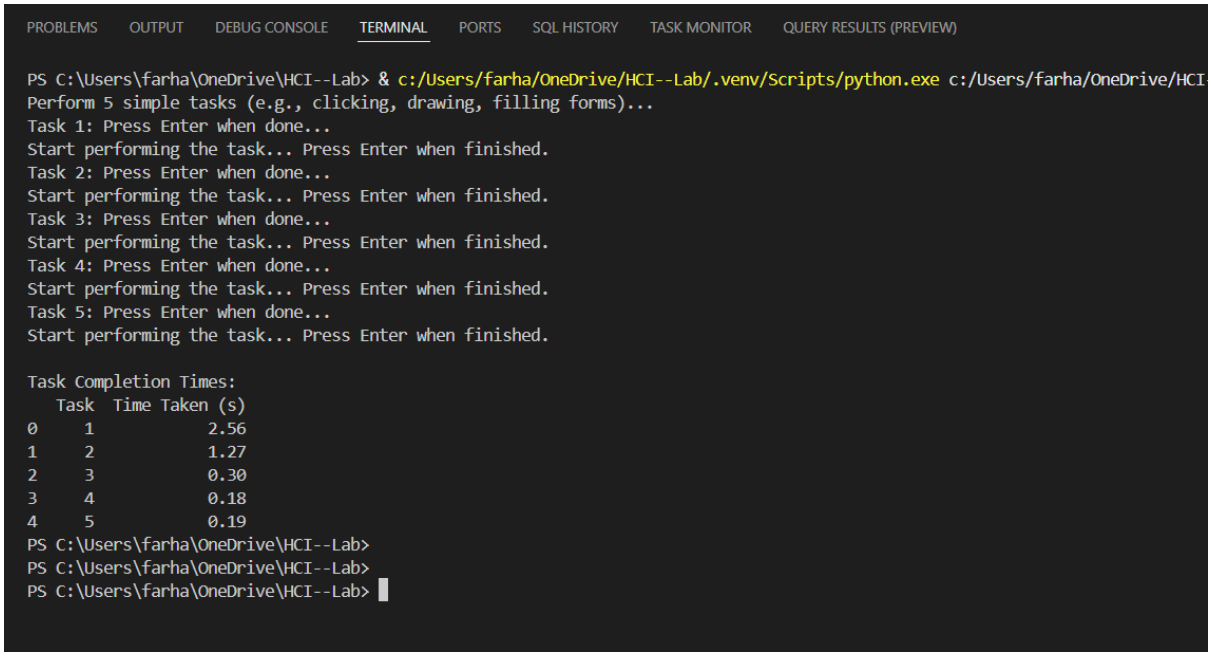
- Python (pandas, time)

Code Implementation:

```
import pandas as pd
import time

log_data = []

print("Perform 5 simple tasks (e.g., clicking, drawing, filling forms)...")

for i in range(5):
    input(f"Task {i+1}: Press Enter when done...")
    start = time.time()
    input("Start performing the task... Press Enter when finished.")
    end = time.time()
    log_data.append({'Task': i+1, 'Time Taken (s)': round(end - start, 2)})

df = pd.DataFrame(log_data)
```

```
print("\nTask Completion Times:")
print(df)
```

Output:



Summary Table of Lab 4

Activity	Focus	Tools Used	Output
Activity 1	Attention & Distraction	Tkinter, Threading	UI distraction simulation
Activity 2	Perception & Recognition	Matplotlib	Heatmaps and UI redesign
Activity 3	Memory & Automation	Selenium	Automated form filling
Activity 4	Learning & Skill Acquisition	Pandas	Learning curve data

📚 Helping Material / References

- "Human-Computer Interaction" by Alan Dix et al.

- "Computer Graphics: Principles and Practice" by John F. Hughes et al.
- Figma / Adobe XD tutorials online.

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#05

Norman's Model

Case Study: Online Banking System – Transfer Money

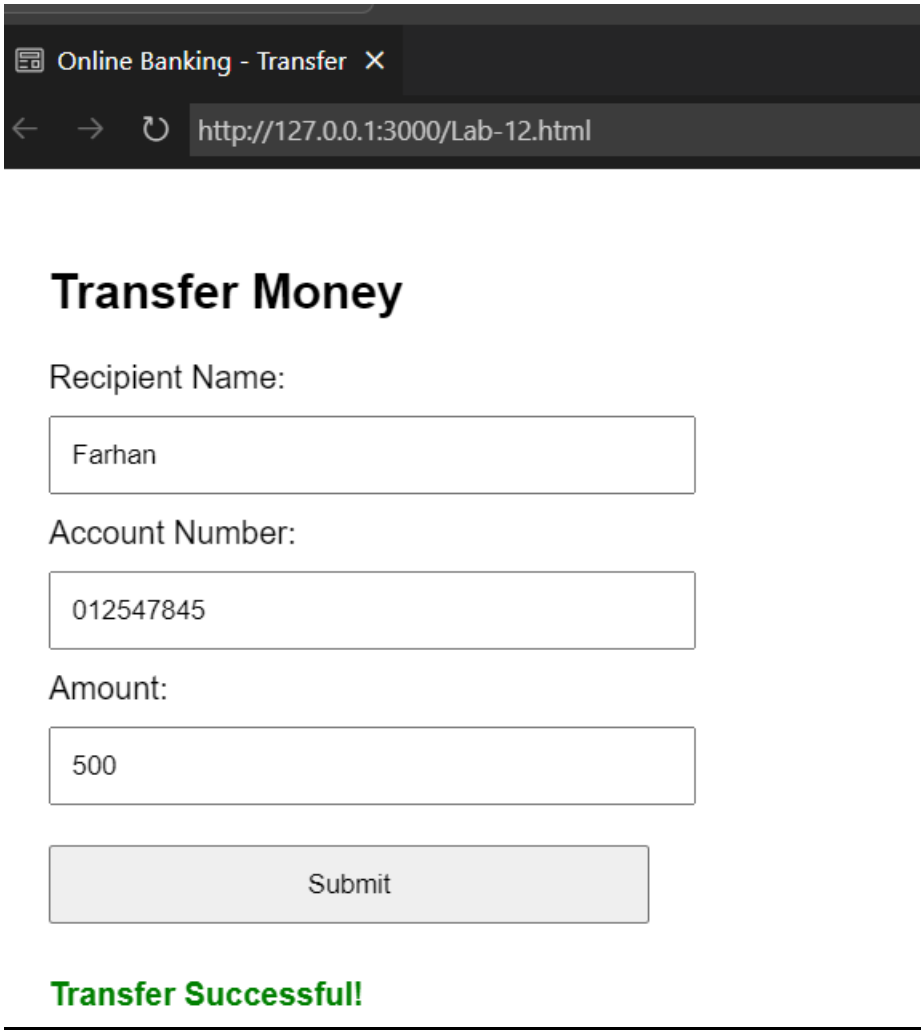
Activity#1

HTML Code: Simple Transfer Form Mockup:

```
<!DOCTYPE html>
<html>
<head>
  <title>Online Banking - Transfer</title>
  <style>
    body { font-family: Arial; padding: 20px; }
    input, button { padding: 10px; margin: 10px 0; width: 300px; }
    #message { color: green; font-weight: bold; }
  </style>
</head>
<body>
```

```
  <h2>Transfer Money</h2>
  <label>Recipient Name:</label><br>
  <input type="text" id="name"><br>
  <label>Account Number:</label><br>
  <input type="text" id="account"><br>
  <label>Amount:</label><br>
  <input type="number" id="amount"><br>
  <button onclick="submitTransfer()">Submit</button>
  <p id="message"></p>

  <script>
    function submitTransfer() {
      const name = document.getElementById('name').value;
      const account = document.getElementById('account').value;
      const amount = document.getElementById('amount').value;
      if (name && account && amount) {
        document.getElementById('message').innerText = "Transfer Successful!";
      } else {
        document.getElementById('message').innerText = "Please fill all fields.";
        document.getElementById('message').style.color = "red";
      }
    }
  </script>
</body>
</html>
```

Output:

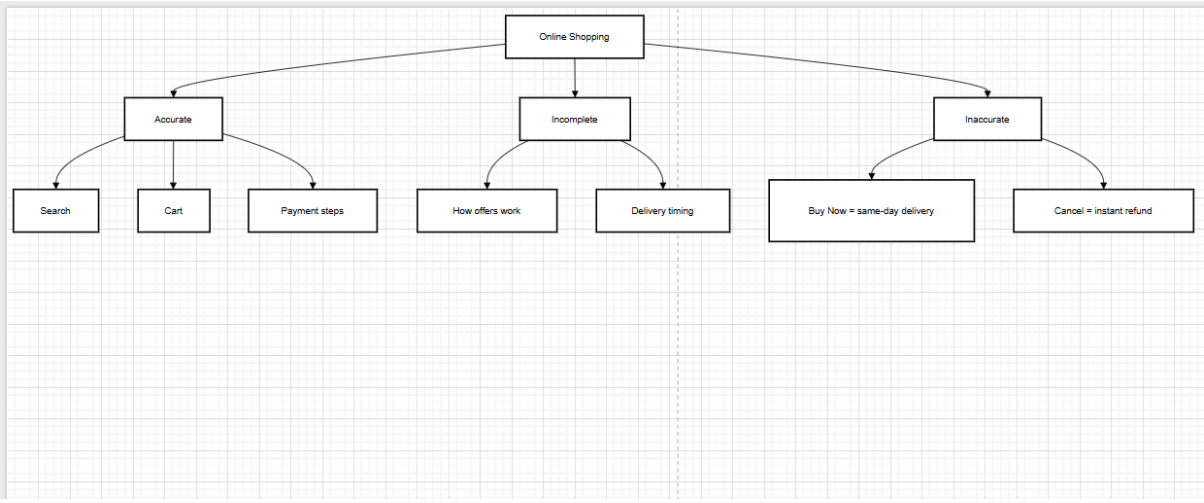

Online Banking - Transfer
http://127.0.0.1:3000/Lab-12.html
Transfer Money
Recipient Name:
Farhan
Account Number:
012547845
Amount:
500
Submit
Transfer Successful!

Activity 2: Mental Model Mapping

Objective:

Create a mental model map of a familiar system showing accurate, incomplete, and inaccurate understandings.

Mind Map:



Activity 3: Norman's Model Role-Play

Objective:

Act out the Norman's model stages with a chosen system scenario.

Example Scenario: Booking a Flight Online

Norman's Model Breakdown

Stage	Role-Play Example
Goal	"I need to fly from Delhi to Mumbai next week"
Intention	"I want to book a ticket using MakeMyTrip"
Action	Opens site → Searches flights → Selects one → Enters details → Pays
Execution	Clicks confirm and waits for loading screen

Feedback	Gets ticket confirmation + email
Interpretation	"The booking is successful and I should get an e-ticket soon"
Evaluation	Confirms booking status from "My Bookings" page

Role-Play Setup:

- **User:** Interacts with booking interface
- **System:** Other students act as responses (e.g., confirmation, errors)
- **Observer:** Takes notes on execution and evaluation stages

Activity 4: User Interface Redesign

Objective:

Redesign a bad UI using Norman's principles + Mental Model corrections.

Example: Poor UI – ATM Interface

Problems:

- No progress feedback
- Confusing button layout
- No error message on failed PIN

Redesigned ATM UI – HTML Prototype:

```
<!DOCTYPE html>
<html>
<head>
  <title>ATM Simulator</title>
  <style>
```

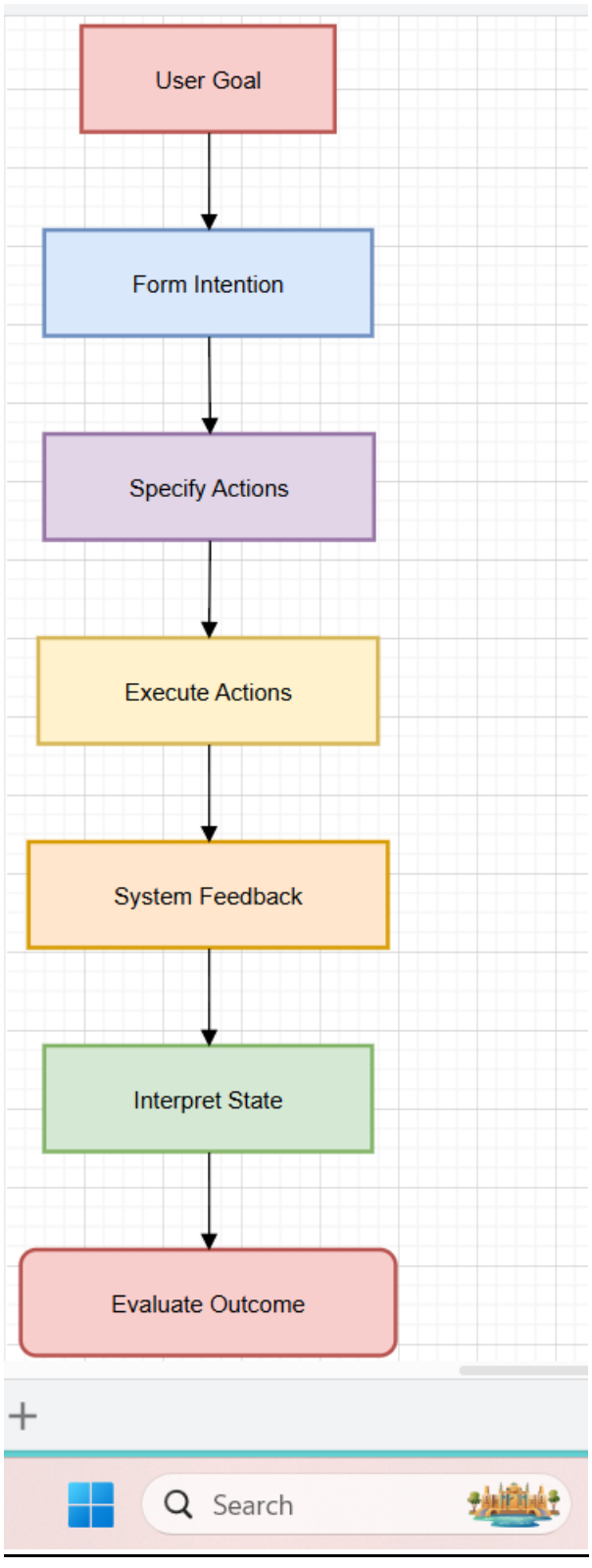
```
    body { font-family: Arial; padding: 20px; background: #e0f7fa; }
    input, button { margin: 10px 0; padding: 10px; width: 250px; }
    #status { font-weight: bold; margin-top: 20px; color: green; }
  </style>
</head>
<body>
  <h2>ATM Withdrawal</h2>
  <label>Enter PIN:</label><br>
  <input type="password" id="pin"><br>

  <label>Amount to Withdraw:</label><br>
  <input type="number" id="amount"><br>

  <button onclick="process()">Withdraw</button>

  <div id="status"></div>

  <script>
    function process() {
      const pin = document.getElementById("pin").value;
      const amount = parseInt(document.getElementById("amount").value);
      const status = document.getElementById("status");

      if (pin !== "1234") {
        status.innerText = "❌ Incorrect PIN";
```

```
      status.style.color = "red";
    } else if (amount > 10000) {
      status.innerText = " ⚠️ Daily limit exceeded.";
      status.style.color = "orange";
    } else {
      status.innerText = ` Please collect ${amount}`;
      status.style.color = "green";
    }
  }
 </script>
</body>
</html>
```

Output:

Deliverables

1. Diagram of Norman's Model



2. Mental Model Reflection

At first, I assumed banking apps worked instantly and always clearly. After doing the lab and observing UI gaps, I understood how incomplete and inaccurate models affect user trust. My updated model includes understanding feedback, error states, and limitations."

3. Redesigned UI (HTML Above already done)

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#06

Prototyping

Activity 1: Issues-Evaluators Matrix Redesign

Objective:

Redesign the confusing graph (Issues-Evaluators Matrix) using an effective prototyping tool or method.

Redesign Idea:

A clean matrix-style table layout where:

- Rows = Evaluators (E1, E2, E3...)
- Columns = Identified Usability Issues (Issue 1, Issue 2…)
- Checkmarks (✓) indicate which evaluator found which issue.

Html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Issues-Evaluators Matrix</title>
```

```
  <style>
    table {
      width: 60%;
      border-collapse: collapse;
      margin: 30px auto;
    }
    th, td {
      border: 1px solid #999;
      padding: 10px;
      text-align: center;
    }
    th {
      background-color: #4CAF50;
      color: white;
    }
    td.check {
      background-color: #c8e6c9;
    }
  </style>
</head>
<body>
  <h2 style="text-align:center;">Issues-Evaluators Matrix</h2>
  <table>
    <tr>
      <th>Evaluator</th>
      <th>Issue 1</th>
      <th>Issue 2</th>
      <th>Issue 3</th>
```

```
    <th>Issue 4</th>
   </tr>
   <tr>
    <td>Evaluator 1</td>
    <td class="check">✓</td>
    <td></td>
    <td class="check">✓</td>
    <td></td>
   </tr>
   <tr>
    <td>Evaluator 2</td>
    <td></td>
    <td class="check">✓</td>
    <td class="check">✓</td>
    <td class="check">✓</td>
   </tr>
   <tr>
    <td>Evaluator 3</td>
    <td class="check">✓</td>
    <td></td>
    <td></td>
    <td></td>
   </tr>
  </table>
</body>
</html>
```

Inteface:

Issues-Evaluators Matrix

Evaluator	Issue 1	Issue 2	Issue 3	Issue 4
Evaluator 1	✓		✓	
Evaluator 2		✓	✓	✓
Evaluator 3	✓			

Activity 2: Redesigned Chat Window

Step 1: Existing Chat Window Issues

- Outdated UI style.
- No dark mode.
- No emoji reactions.
- No media/file sharing.
- No voice input.
- No read receipts or typing indicators.

Step 2: Redesign Goals

- Improve usability, accessibility, and aesthetics.
- Add:

- Dark/Light Mode toggle
- Emoji support
- Read Receipts
- Voice-to-text input
- File sharing
- Typing status
- Custom themes

Html Code:

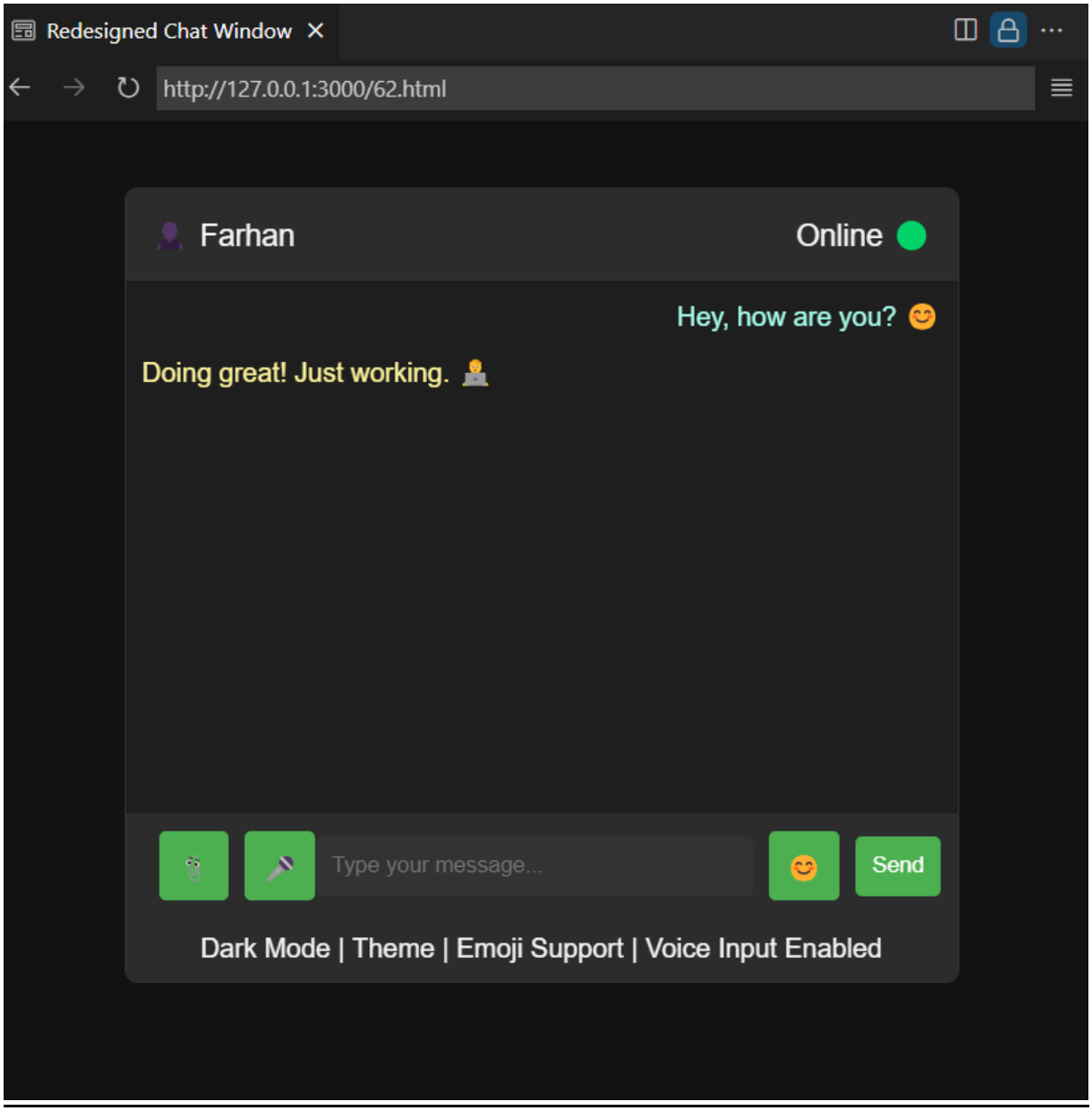
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Redesigned Chat Window</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #121212;
      color: #f5f5f5;
      margin: 0;
      padding: 0;
    }
    .chat-container {
      max-width: 500px;
      margin: 40px auto;
      border: 1px solid #333;
      border-radius: 8px;
      background-color: #1e1e1e;
```

```
  overflow: hidden;
}
.chat-header {
  background-color: #2c2c2c;
  padding: 15px;
  font-size: 18px;
  display: flex;
  justify-content: space-between;
}
.chat-area {
  height: 300px;
  overflow-y: auto;
  padding: 10px;
  background-color: #1e1e1e;
  border-top: 1px solid #333;
  border-bottom: 1px solid #333;
}
.message {
  margin-bottom: 12px;
}
.message.user {
  text-align: right;
  color: #a7ffeb;
}
.message.friend {
  text-align: left;
  color: #fff59d;
}
```

```
.chat-input {
  display: flex;
  align-items: center;
  padding: 10px;
  background-color: #2c2c2c;
}
.chat-input input {
  flex: 1;
  padding: 10px;
  border-radius: 4px;
  border: none;
  background-color: #333;
  color: #fff;
}
.chat-input button, .chat-input span {
  margin-left: 10px;
  background-color: #4caf50;
  border: none;
  padding: 10px;
  border-radius: 4px;
  cursor: pointer;
  color: #fff;
}
.chat-footer {
  padding: 10px;
  text-align: center;
  background-color: #2c2c2c;
}
```

```
  </style>
</head>
<body>
  <div class="chat-container">
    <div class="chat-header">
      <div> 👤 Friend</div>
      <div>Online 🟢 </div>
    </div>
    <div class="chat-area">
      <div class="message user">Hey, how are you? 😊 </div>
      <div class="message friend">Doing great! Just working. 👨‍💻 </div>
    </div>
    <div class="chat-input">
      <span> 📎 </span>
      <span> 🎤 </span>
      <input type="text" placeholder="Type your message...">
      <span> 😊 </span>
      <button>Send</button>
    </div>
    <div class="chat-footer">
      Dark Mode | Theme | Emoji Support | Voice Input Enabled
    </div>
  </div>
</body>
</html>
```

Interface:



THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab#07

Neilson's Heuristic

Activity 1: Heuristic Evaluation Report

Selected Interface: Facebook
Heuristics Used: Nielsen's Ten Heuristics

Example 1: Clicking the Like Button with a Slow Internet Connection

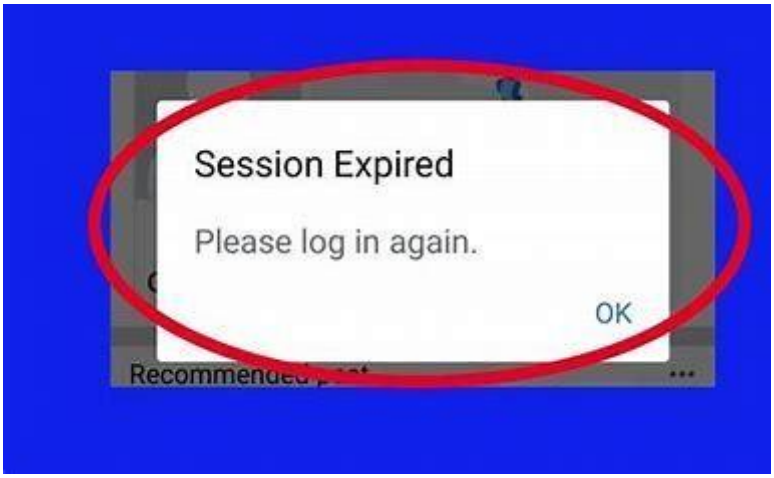
- When a user clicks the **Like** button on a post while experiencing a slow or unstable internet connection, there is **no immediate visual feedback**.
- The user is left wondering whether the action was registered or not.

Screenshot:

Example 2: Delayed Login Response

- If a user enters incorrect login credentials, Facebook sometimes takes time to respond.
- No immediate error message appears, leaving the user uncertain.

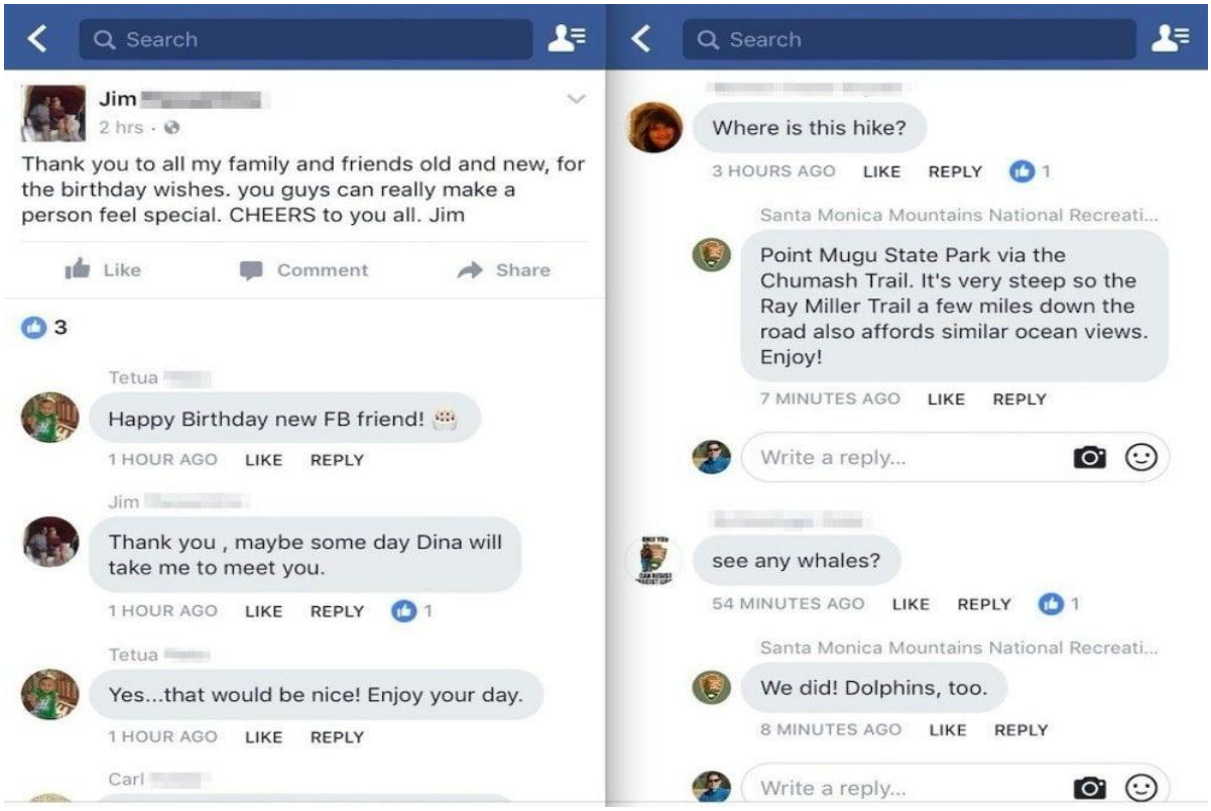
Screenshot:



Example 3: Long-Loading Comment Submission

- Sometimes when posting a comment, there is **no loading animation**, causing the user to click multiple times.
- This leads to **duplicate comments** being posted.

Screenshot:



Example 4: No Clear Error Message for Failed Post Upload

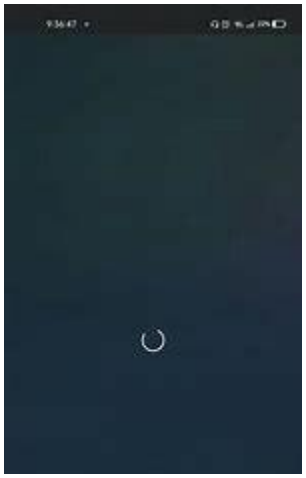
- If a post upload fails due to a network issue, Facebook does not always display a **clear retry option**.
- Users are left confused about whether they should try again.

Screenshot:

Example 5: Stuck on Loading Screen

- Occasionally, when switching between tabs or reloading, Facebook gets stuck on a **blank or infinite loading screen** without any message.
- Users are left uncertain if the app is working.

Screenshot:



Why These Violate the Heuristic:

- No immediate indication that an action is in progress.
- Users are left in uncertainty, reducing trust in the system.

4. Suggested Improvements

To improve system visibility, Facebook should:

- Display a **temporary loading indicator** (e.g., a gray placeholder Like animation) until the server confirms the action.
- Show a **message** like “Processing Like…” when the action is delayed.
- Implement a **retry mechanism** in case of connection issues.
- Ensure immediate error feedback for login failures.
- Add real-time status indicators for comment submission and post uploads.

Activity 2: UI Component Redesign

Component Chosen: Old Chat Window

Problems in Original Chat Design:

- No dark/light mode support
- No clear separation between user and others' messages
- No emoji or voice message option

Nielsen's Heuristics Applied:

- Consistency and Standards: Improved iconography, follows messaging app conventions.
- Aesthetic and Minimalist Design: Removed clutter, used minimal design with subtle colors.
- Flexibility and Efficiency of Use: Added emoji support, dark mode toggle.

Redesigned HTML Chat UI (Prototype):

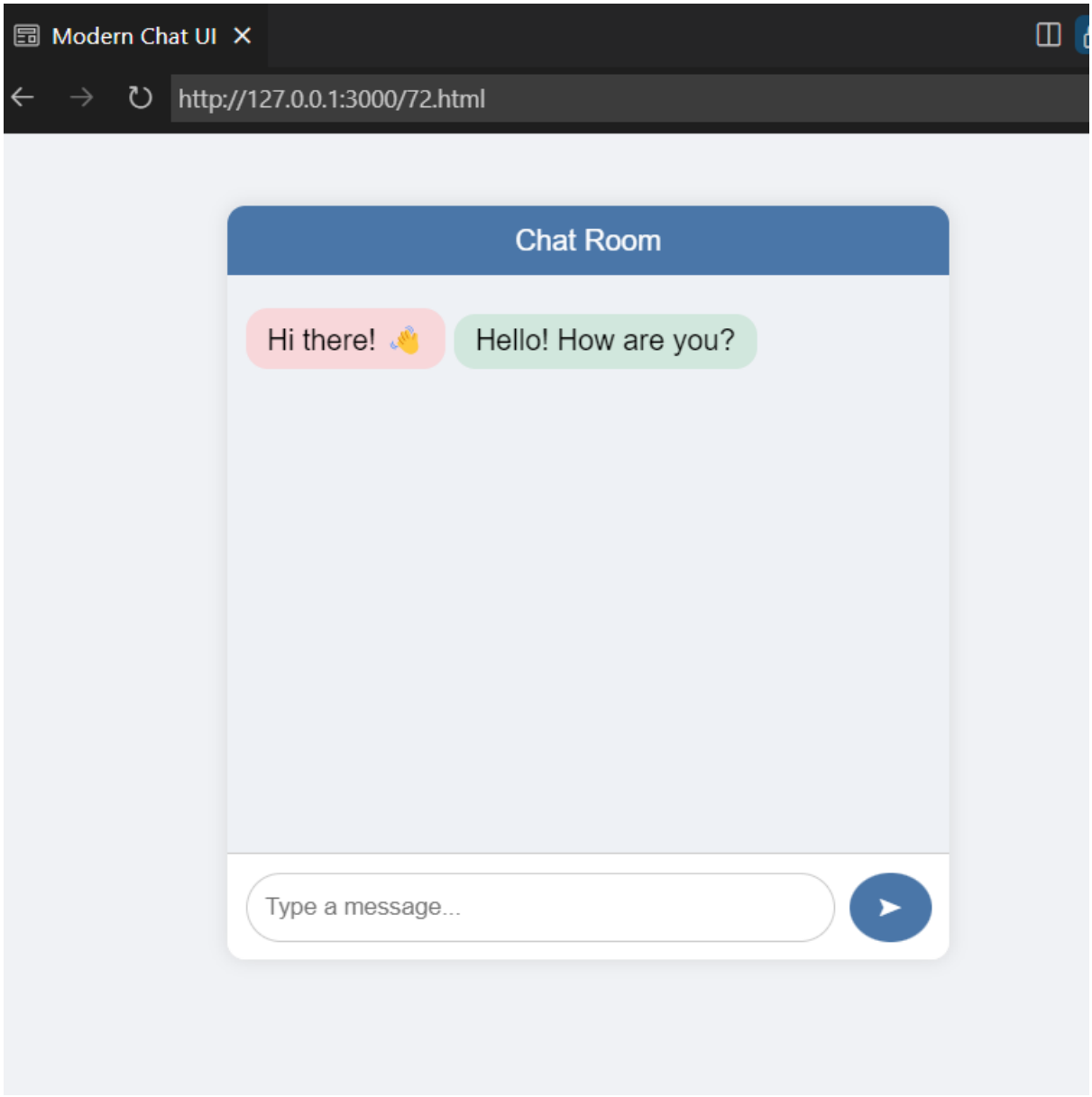
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Modern Chat UI</title>
 <style>
  body {
   margin: 0;
   font-family: Arial, sans-serif;
   background-color: #f0f2f5;
  }
  .chat-container {
```

```
  max-width: 400px;
  margin: 40px auto;
  background: #fff;
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
.chat-header {
  background: #4a76a8;
  color: white;
  padding: 10px;
  text-align: center;
}
.chat-body {
  height: 300px;
  overflow-y: auto;
  padding: 10px;
  background: #eef1f5;
}
.message {
  margin: 8px 0;
}
.from-user {
  text-align: right;
  background: #d1e7dd;
  padding: 6px 12px;
```

```
  border-radius: 12px;
  display: inline-block;
}
.from-others {
  text-align: left;
  background: #f8d7da;
  padding: 6px 12px;
  border-radius: 12px;
  display: inline-block;
}
.chat-input {
  display: flex;
  border-top: 1px solid #ccc;
  padding: 10px;
  background: #fff;
}
.chat-input input {
  flex: 1;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 20px;
}
.chat-input button {
  background: #4a76a8;
  color: white;
  border: none;
```

```
      padding: 10px 16px;
      margin-left: 8px;
      border-radius: 50%;
      cursor: pointer;
    }
  </style>
</head>
<body>

<div class="chat-container">
  <div class="chat-header">Chat Room</div>
  <div class="chat-body">
    <div class="message from-others">Hi there! 👋</div>
    <div class="message from-user">Hello! How are you?</div>
  </div>
  <div class="chat-input">
    <input type="text" placeholder="Type a message..." />
    <button>➤</button>
  </div>
</div>

</body>
</html>
```

Interface:


Modern Chat UI
http://127.0.0.1:3000/72.html
Chat Room
Hi there!
Hello! How are you?
Type a message...

Activity 3: Group Discussion Summary

Q1: Most Important Heuristic?

- *“Recognition Rather Than Recall”* because it reduces user memory load and improves usability drastically.

Q2: Real-World Violation Example:

- *Old government forms (PDFs)* often violate error prevention and consistency leading to confusion and delays in task completion.

Q3: Heuristics in AR/VR:

- “Visibility of System Status” is critical in AR/VR to prevent user confusion.
- “User Control and Freedom” ensures users can easily exit or undo interactions in immersive environments.

THE END

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab-08

Usability Engineering in Human-Computer Interaction

Task:

Designing a Usable E-commerce Website Task Description: Design an e-commerce website for a fictional online store selling outdoor gear. The website should allow users to browse products, add items to cart, and checkout.

Html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Outdoor Gear Store</title>
    <style>
        /* General Styles */
        body {
```

```
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: white;
    padding: 10px 0;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

header .logo {
    font-size: 24px;
    font-weight: bold;
    padding-left: 20px;
}

header nav ul {
    list-style-type: none;
    margin: 0;
    padding-right: 20px;
}

header nav ul li {
```

```
    display: inline;
    margin-left: 20px;
}

header nav ul li a {
    text-decoration: none;
    color: white;
    font-size: 18px;
}

/* Main Section */
main {
    padding: 20px;
}

.product-list {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: space-around;
}

.product-item {
    background-color: white;
    padding: 10px;
    width: 200px;
    text-align: center;
    border: 1px solid #ddd;
```

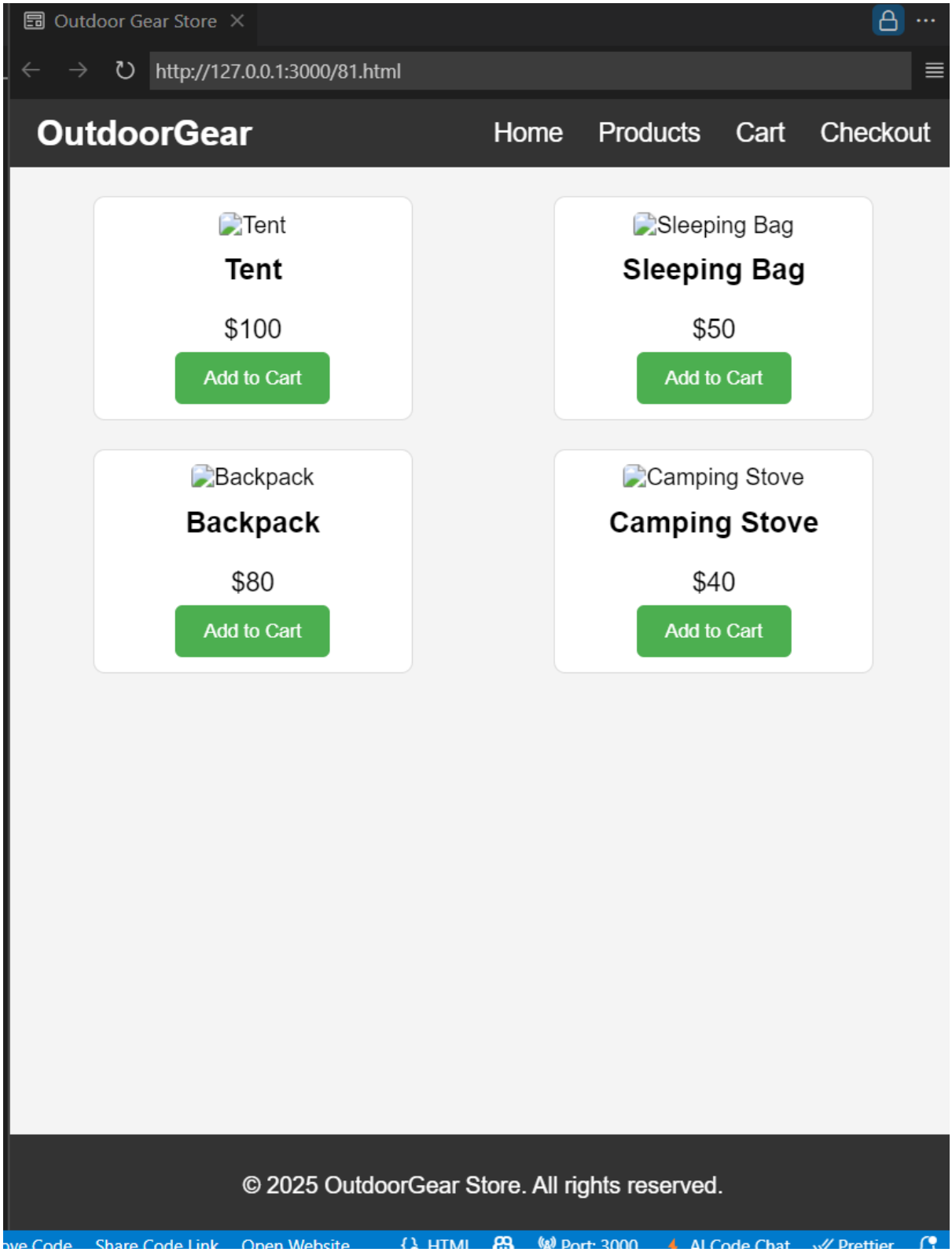
```
    border-radius: 8px;
}

.product-item img {
    width: 100%;
    height: auto;
    border-radius: 5px;
}

.product-item h3 {
    font-size: 20px;
    margin-top: 10px;
}

.product-item p {
    font-size: 18px;
    margin: 5px 0;
}

.product-item button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
```

```
        .product-item button:hover {
            background-color: #45a049;
        }

        /* Footer */
        footer {
            background-color: #333;
            color: white;
            text-align: center;
            padding: 10px 0;
            position: absolute;
            width: 100%;
            bottom: 0;
        }
    </style>
</head>
<body>

    <header>
        <div class="logo">OutdoorGear</div>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Products</a></li>
                <li><a href="#">Cart</a></li>
                <li><a href="#">Checkout</a></li>
            </ul>
        </nav>
```

```
</header>

<main>
  <section class="product-list">
    <div class="product-item">
      <img src="https://via.placeholder.com/150" alt="Tent">
      <h3>Tent</h3>
      <p>$100</p>
      <button>Add to Cart</button>
    </div>
    <div class="product-item">
      <img src="https://via.placeholder.com/150" alt="Sleeping Bag">
      <h3>Sleeping Bag</h3>
      <p>$50</p>
      <button>Add to Cart</button>
    </div>
    <div class="product-item">
      <img src="https://via.placeholder.com/150" alt="Backpack">
      <h3>Backpack</h3>
      <p>$80</p>
      <button>Add to Cart</button>
    </div>
    <div class="product-item">
      <img src="https://via.placeholder.com/150" alt="Camping Stove">
      <h3>Camping Stove</h3>
      <p>$40</p>
      <button>Add to Cart</button>
    </div>
```

```
    </section>
  </main>

  <footer>
    <p>&copy; 2025 OutdoorGear Store. All rights reserved.</p>
  </footer>

</body>
</html>
```

Interface:


Outdoor Gear Store
http://127.0.0.1:3000/81.html
OutdoorGear
Home
Products
Cart
Checkout
Tent
Tent
$100
Add to Cart
Sleeping Bag
Sleeping Bag
$50
Add to Cart
Backpack
Backpack
$80
Add to Cart
Camping Stove
Camping Stove
$40
Add to Cart
© 2025 OutdoorGear Store. All rights reserved.

Activity#02 User Interview

Objective: Understand the needs and goals of a potential user for an e-commerce outdoor gear website.

Interview Summary:

- **User Profile**: Farhan, 28, adventure enthusiast, frequently shops online for camping gear.
- **Interview Method:** Semi-structured interview (in person)
- **Questions & Responses:**

    1. **Q: How often do you shop online for outdoor gear?**
       A: About once a month.
    2. **Q: What do you usually look for in an outdoor gear website?**
       A: Fast loading, clear product categories, real user reviews, and secure checkout.
    3. **Q: What features do you find most useful?**
       A: Product comparisons and filters by brand/price.
    4. **Q: What problems have you faced on other e-commerce websites?**
       A: Confusing navigation and lack of mobile responsiveness.

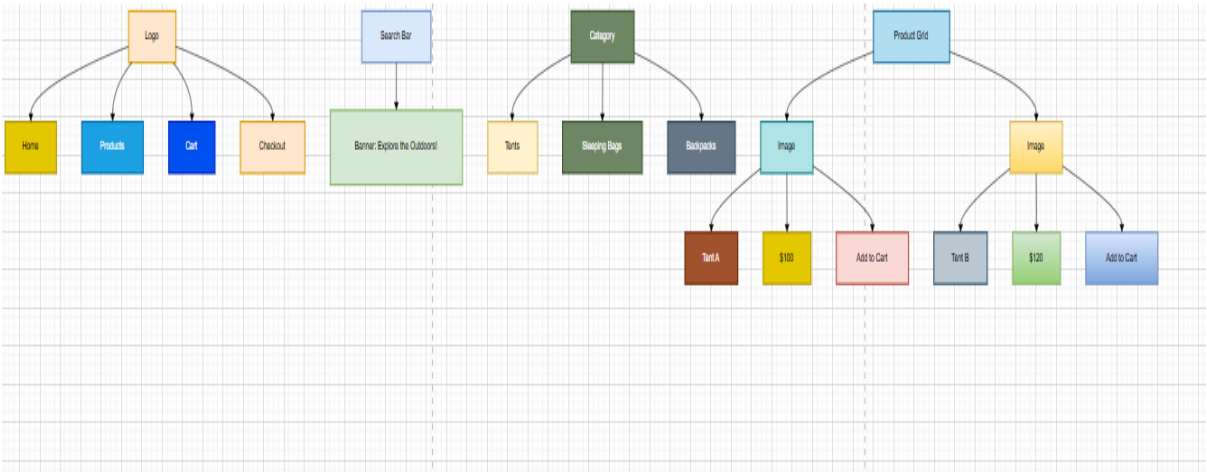
Insights:

- Users prioritize ease of navigation and trust indicators (reviews, secure checkout).
- Filtering options and responsive design are key usability factors.

Activity#03

Wireframing

Result:



Activity#04

Usability Testing Plan

Application: Outdoor Gear Store Website
Goal: Test how easily users can browse products, add to cart, and checkout.

Plan Outline:

- **Participants:** 5 target users (outdoor enthusiasts, age 18–40)
- **Test Scenarios:**
    1. Find and view details of a sleeping bag.
    2. Add a backpack to the cart.
    3. Navigate to the checkout page and simulate a purchase.
- **Metrics to Measure:**
    - Task completion time
    - Error rate
    - User satisfaction (via post-test survey)

- **Method:**
  - In-person testing or remote screen-sharing
  - Think-aloud protocol (user narrates their thoughts)
- **Data Collection:**
  - Observation notes
  - Screen recordings
  - User feedback form
- **Success Criteria:**
  - 80% task completion rate
  - Positive satisfaction score (4 out of 5 average)

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

LAB-9

SYSTEM'S EVALUATION BASED ON NEILSEN'S HEURISTICS

Part 1: Heuristic Evaluation Table (Facebook Interface)

Feature	Heuristic Violated	Issue Description	Severity	Suggested Improvement
Add Friend	User Control & Freedom	No option to cancel request immediately after sending	Major	Add an "Undo" or "Cancel Request" option right after clicking
Change Password	Error Prevention	No clear confirmation before saving new password	Major	Add a confirmation dialog with option to view new password

Upload Cover Photo	Visibility of System Status	No progress indicator or success confirmation after upload	Minor	Add loading bar + “Upload Successful” message
Search a Person	Recognition Rather Than Recall	Search suggestions don’t show recent profiles clearly	Minor	Show recent searches and match highlights
Update Timeline Post	Help Users Recognize, Diagnose, and Recover from Errors	Errors (like post failure) show generic messages	Major	Show specific reasons (e.g., "Connection Lost")

Part 2: Filled Questionnaire Results (Sample Responses)

For 3 participants (example only):

Question	P1	P2	P3
1. Does FB provide clear feedback after actions?	Yes, always	Sometimes	Sometimes
2. Are the terms clear (e.g., Timeline)?	Very clear	Somewhat	Very clear
3. Can you undo friend requests/posts?	Somewhat	No, difficult	Somewhat
4. Do buttons behave consistently?	Yes	Sometimes	Yes

5. Does Facebook warn you before irreversible actions?	Sometimes	Sometimes	No
6. Do recent profiles help in search?	Very helpful	Somewhat	Not helpful
7. Are shortcuts/quick actions available?	Somewhat	No	No
8. Is interface cluttered?	Minimal	Somewhat	Somewhat
9. Are error messages helpful?	Somewhat	No	Somewhat
10. Is help easy to find?	Very easy	Somewhat	No

Part 3: Analysis and Insights

- **Common Violations Identified:**
    - Poor user control and freedom (e.g., canceling friend requests).
    - Lack of error prevention and specific feedback.
    - Inconsistent visibility of system status (e.g., uploading images).
- **Heuristics Most Violated:**
    - User Control & Freedom
    - Error Prevention
    - Help Users Recognize, Diagnose, and Recover from Errors

Part 4: Redesign Suggestions

Feature	Problem Identified	Redesign Suggestion
Add Friend	No cancel option	Add a temporary “Undo” banner or icon
Upload Cover Photo	No confirmation of success	Display toast message + image preview after upload
Change Password	Easy to save by mistake	Add “Show password” toggle and confirmation step
Update Post	Errors are vague	Provide error code or description (e.g., “You’re offline”)
Search	Poor recognition	Add “Recent Searches” list and “Suggestions” box below the bar

Part 5: Reflection (Theory vs Practice)

Aspect	Theory (Nielsen’s Heuristics)	Practice (Facebook)
Error prevention	Should warn users before they make irreversible actions	Often skips warnings (e.g., no prompt before saving new password)
User control	Users should be able to easily undo mistakes	No undo after sending a friend request

Recognition over recall	System should help users remember (e.g., search history)	Doesn’t provide search history or smart suggestions
Feedback	System should inform users of progress or success	Uploads and actions sometimes lack feedback
Recovery from errors	Users should get helpful error messages	Facebook’s error messages are generic or uninformative

THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

Lab-10

Design Decisions, Prototyping & Usability Evaluation

Activity 1: Food Delivery App

1. Design Decisions (with Justification)

Decision 1: Text-based Search vs. Voice Search

- **Chosen:** Text-based search
- **Justification:** Most users are accustomed to typing when using search features. Text input ensures better accuracy, is more universally supported, and avoids issues in noisy environments.

Decision 2: Add-to-Cart Button Placement

- **Chosen: Place "Add to Cart" beside each food item**
- **Justification:** Minimizes the number of taps, improves flow, and supports user efficiency. It aligns with Fitts's Law (ease of access to frequently used controls).

2. Paper Prototype (3+ Screens)

You should create the following paper sketches:

- Screen 1: Login / Signup
- Screen 2: Home (restaurant list with search bar)
- Screen 3: Menu screen with food items and "Add to Cart"
- Screen 4: Checkout screen

You can draw on paper or create digitally using Figma/Balsamiq. Include:

- Buttons
- Navigation icons
- Search bar
- Cart icon

Html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Food Delivery App</title>
 <style>
  body {
   font-family: Arial;
   background: #f2f2f2;
   margin: 0;
  }
  .container {
   max-width: 400px;
   margin: auto;
```

```
  padding: 20px;
  background: white;
  border-radius: 10px;
  margin-top: 30px;
}
h2 {
  text-align: center;
}
input, button {
  width: 100%;
  padding: 10px;
  margin-top: 10px;
  margin-bottom: 10px;
  font-size: 16px;
  border-radius: 5px;
  border: 1px solid #ccc;
}
button {
  background: #28a745;
  color: white;
  border: none;
}
button:hover {
  background: #218838;
}
.item {
  display: flex;
  justify-content: space-between;
```

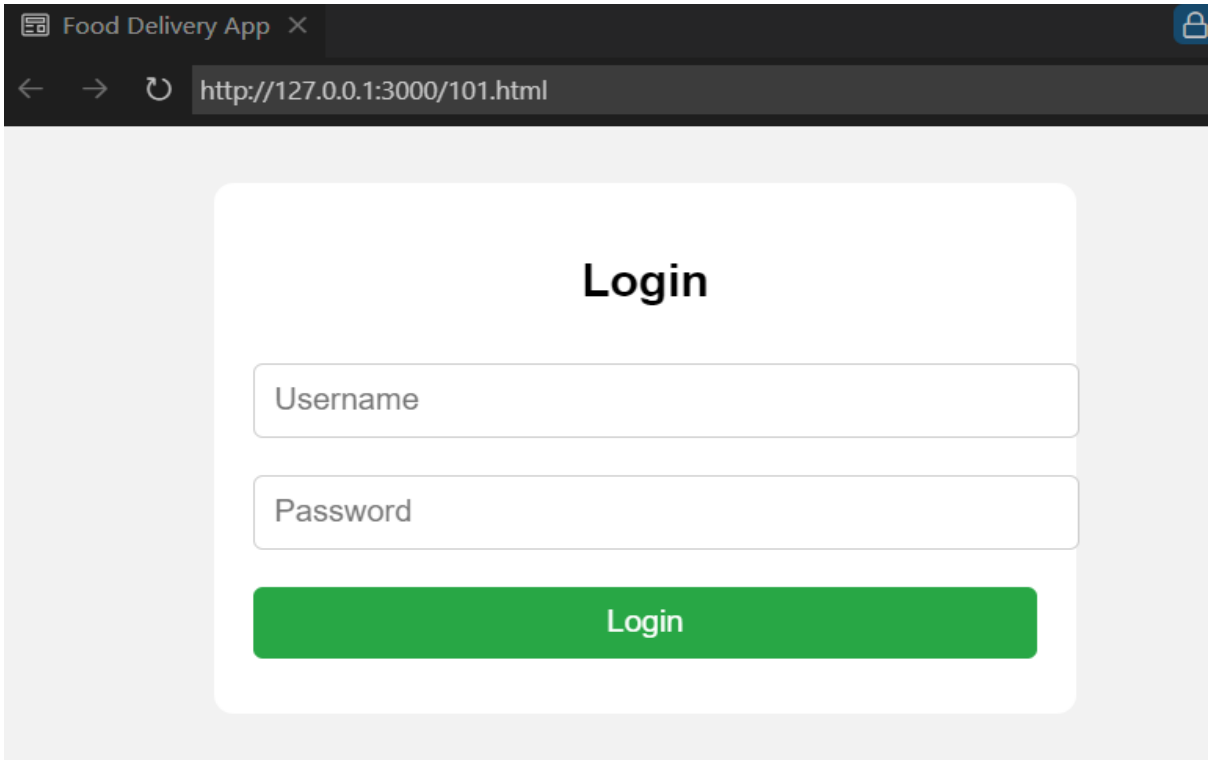
```
    margin: 10px 0;
  }
  .hidden {
    display: none;
  }
 </style>
</head>
<body>

 <!-- Login Section -->
 <div class="container" id="login">
  <h2>Login</h2>
  <input type="text" placeholder="Username" id="user">
  <input type="password" placeholder="Password" id="pass">
  <button onclick="login()">Login</button>
 </div>

 <!-- Home Section -->
 <div class="container hidden" id="home">
  <h2>Restaurants</h2>
  <div class="item"><span>Pizza Point</span><button
onclick="add('Pizza')">Add</button></div>
  <div class="item"><span>Burger Town</span><button
onclick="add('Burger')">Add</button></div>
  <div class="item"><span>Biryani House</span><button
onclick="add('Biryani')">Add</button></div>
  <button onclick="goToCart()">Go to Cart</button>
 </div>
```

```
<!-- Cart Section -->
<div class="container hidden" id="cart">
  <h2>Cart</h2>
  <div id="cartList"></div>
  <button onclick="checkout()">Checkout</button>
</div>

<!-- Confirmation Section -->
<div class="container hidden" id="confirmation">
  <h2>Order Confirmed!</h2>
  <p>Thank you for your order.</p>
</div>

<script>
  let cart = [];

  // Login function
  function login() {
    const u = document.getElementById("user").value;
    const p = document.getElementById("pass").value;
    if (u && p) {
      show("home");
    } else {
      alert("Enter credentials");
    }
  }

  // Add item to cart
```

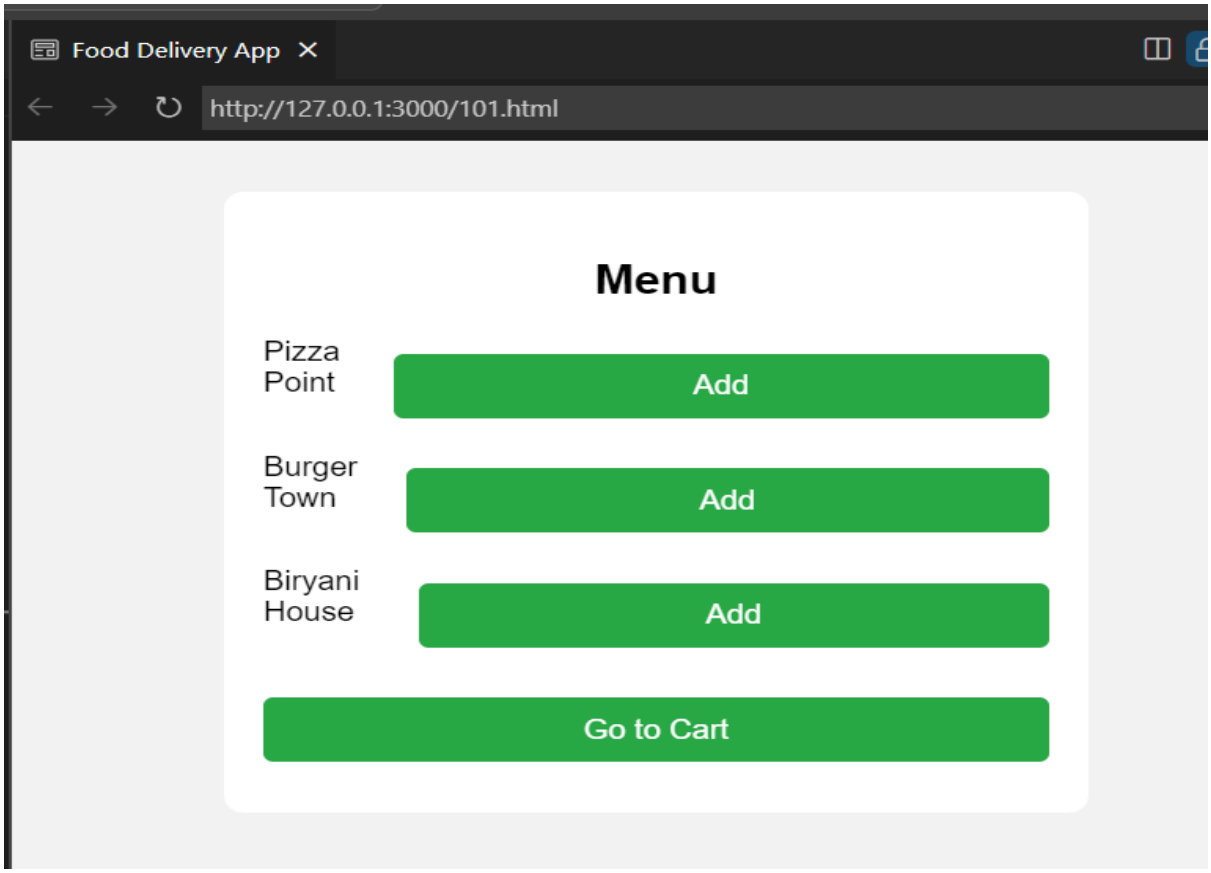
```
  function add(item) {
    cart.push(item);
    alert(item + " added to cart!");
  }

  // Go to Cart page
  function goToCart() {
    const list = document.getElementById("cartList");
    list.innerHTML = "";
    cart.forEach((item, index) => {
      list.innerHTML += `<div class="item"><span>${item}</span><button
onclick="remove(${index})">Remove</button></div>`;
    });
    show("cart");
  }

  // Remove item from cart
  function remove(index) {
    cart.splice(index, 1);
    goToCart();
  }

  // Checkout function
  function checkout() {
    if (cart.length === 0) {
      alert("Cart is empty!");
      return;
    }
```

```
    cart = [];
    show("confirmation");
  }

  // Show the selected page
  function show(id) {
    ["login", "home", "cart", "confirmation"].forEach(sec => {
      document.getElementById(sec).classList.add("hidden");
    });
    document.getElementById(id).classList.remove("hidden");
  }
 </script>
</body>
</html>
```

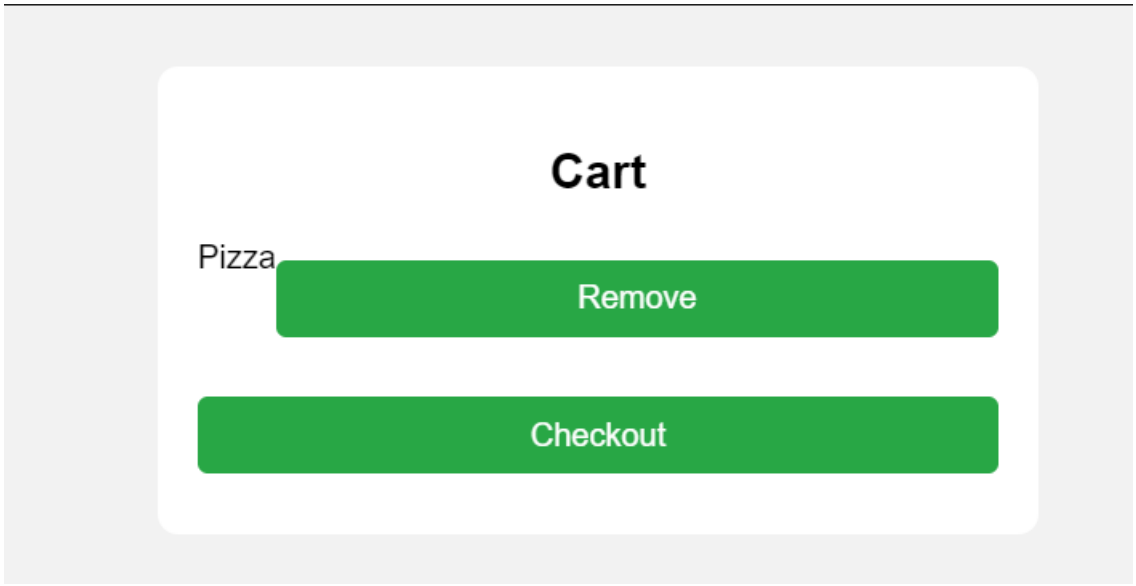
Interface 1:



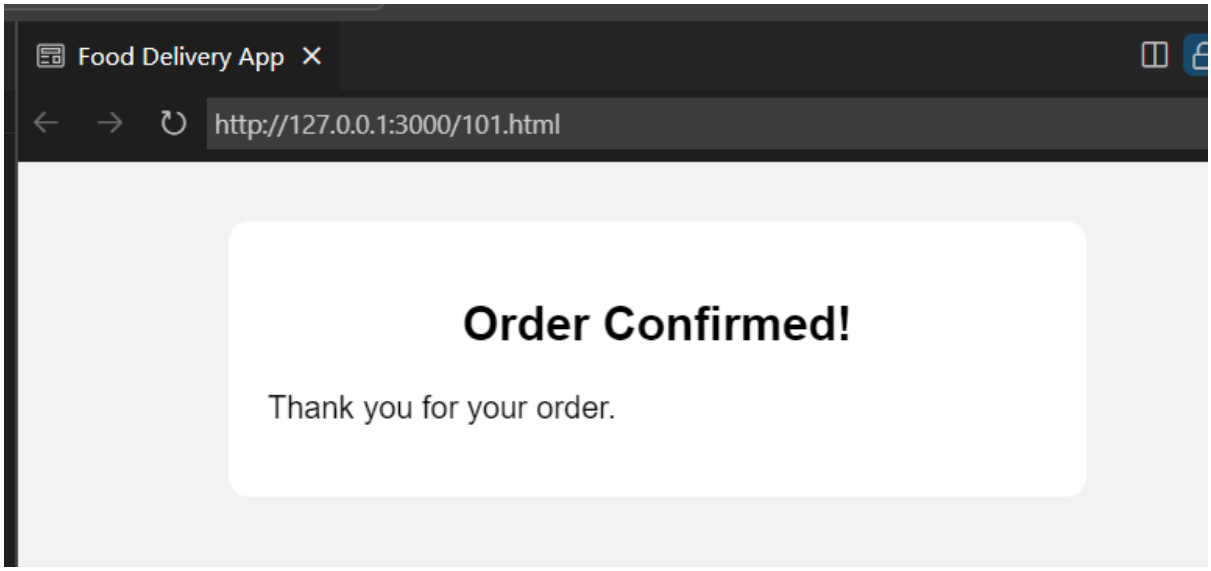
Interface 2:



Interface 3:



Interface 4:



Activity 2: Student Attendance App

1. Design Decisions

Decision 1: Checkbox vs. Dropdown for Marking Attendance

- **Chosen: Checkbox**
- **Justification:** It's faster, requires fewer clicks, and visually confirms the presence status instantly.

Decision 2: Show Date and Time Automatically

- **Chosen:** Yes, display date/time on attendance screen.
- **Justification:** Helps teachers ensure the correct session is being marked. Reduces cognitive effort and errors.

2. HTML Code:

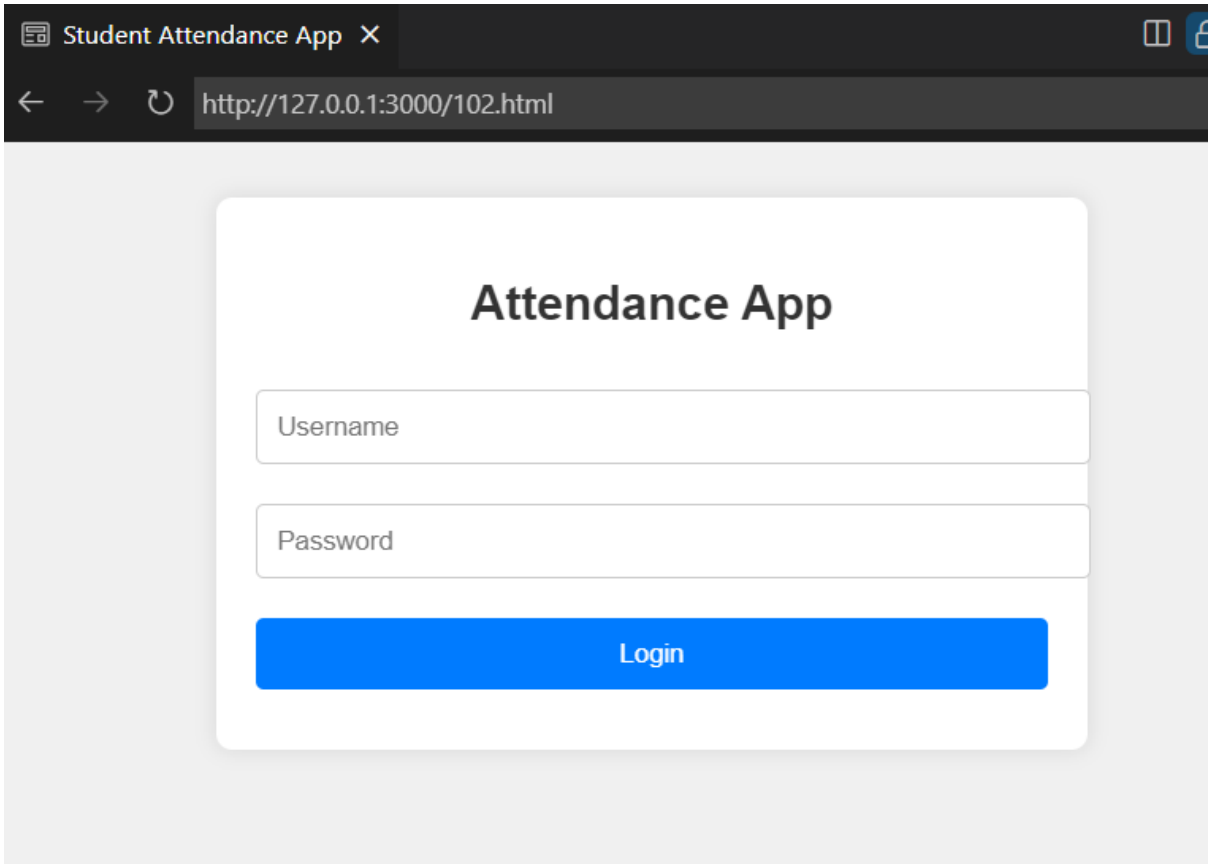
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Student Attendance App</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   background: #f0f0f0;
   padding: 20px;
  }
  .container {
   max-width: 400px;
   margin: auto;
   background: white;
   border-radius: 8px;
   padding: 20px;
   box-shadow: 0 0 10px rgba(0,0,0,0.1);
  }
  h2, h3 {
   text-align: center;
   color: #333;
  }
  input, button {
   width: 100%;
```

```
  padding: 10px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
}
button {
  background: #007bff;
  color: white;
  cursor: pointer;
  border: none;
}
button:hover {
  background: #0056b3;
}
.student {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
}
.hidden {
  display: none;
}
.feedback {
  color: green;
  text-align: center;
  margin-top: 10px;
}
.info {
```

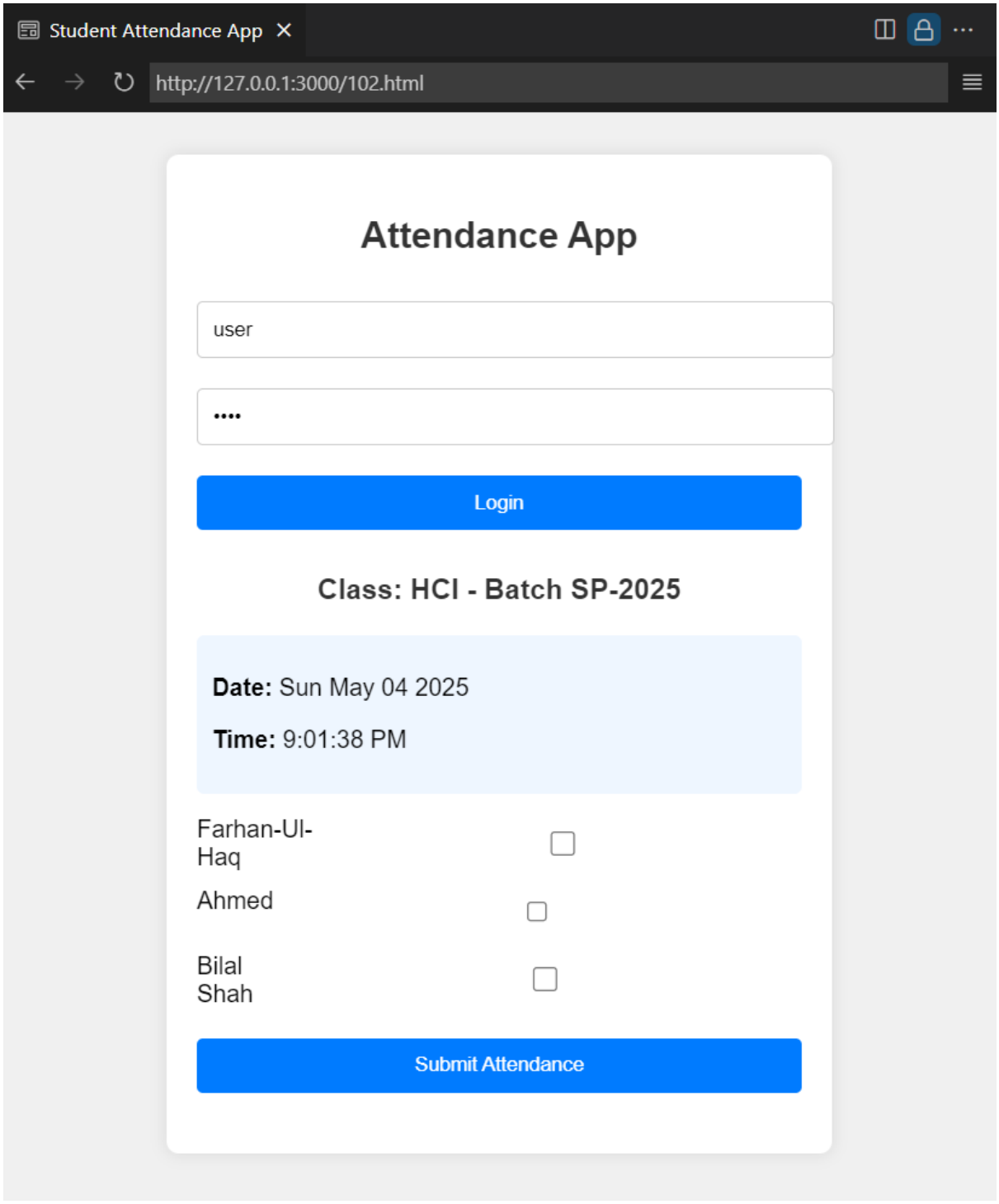
```
    background: #eef6ff;
    padding: 10px;
    margin-bottom: 15px;
    border-radius: 5px;
   }
  </style>
</head>
<body>
  <div class="container">
   <h2>Attendance App</h2>
   <input type="text" id="username" placeholder="Username" />
   <input type="password" id="password" placeholder="Password" />
   <button onclick="login()">Login</button>

   <div id="attendance" class="hidden">
    <h3>Class: HCI - Batch SP-2025</h3>
    <div class="info">
     <p><strong>Date:</strong> <span id="date"></span></p>
     <p><strong>Time:</strong> <span id="time"></span></p>
    </div>
    <div class="student"><span>Farhan-Ul-Haq</span><input
type="checkbox"/></div>
    <div class="student"><span> Ahmed</span><input type="checkbox"/></div>
    <div class="student"><span>Bilal Shah</span><input type="checkbox"/></div>
    <button onclick="submit()">Submit Attendance</button>
    <div class="feedback" id="msg"></div>
   </div>
  </div>
```

```
  <script>
    function login() {
      const user = document.getElementById("username").value;
      const pass = document.getElementById("password").value;
      if (user && pass) {
        document.getElementById("attendance").classList.remove("hidden");
        setDateTime();
      } else {
        alert("Please enter both username and password.");
      }
    }

    function setDateTime() {
      const now = new Date();
      document.getElementById("date").innerText = now.toDateString();
      document.getElementById("time").innerText = now.toLocaleTimeString();
    }

    function submit() {
      document.getElementById("msg").innerText = "Attendance submitted
successfully!";
    }
  </script>
</body>
</html>
```

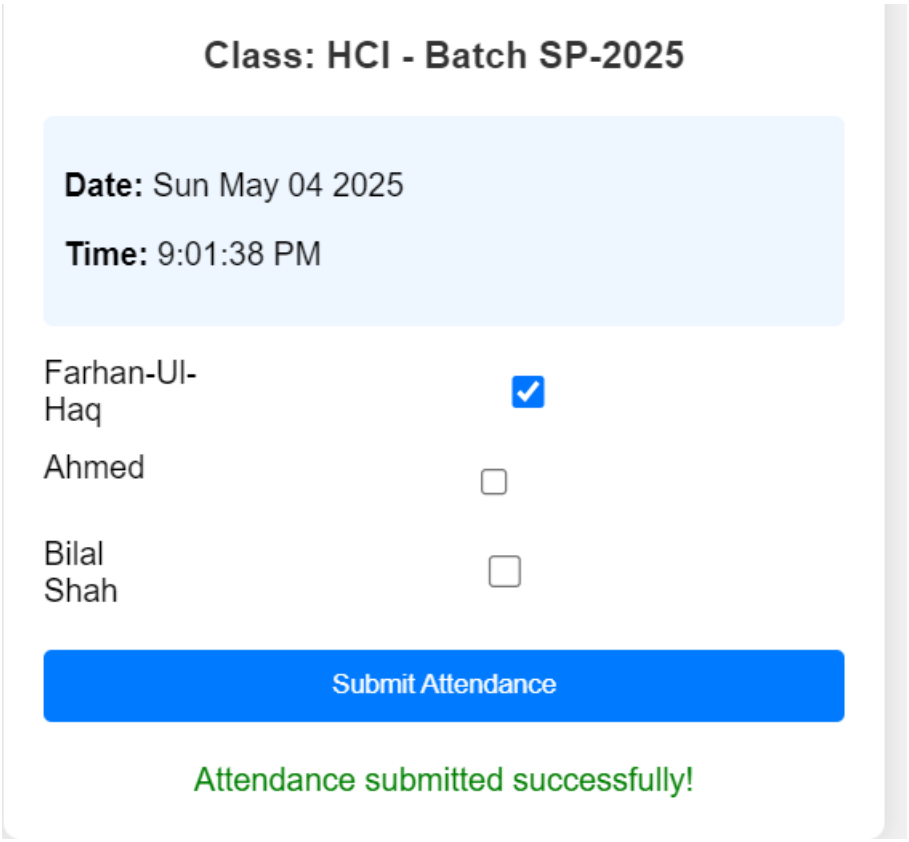
Interface 1:


Student Attendance App
http://127.0.0.1:3000/102.html
Attendance App
Username
Password
Login

Interface 2:


Student Attendance App
http://127.0.0.1:3000/102.html
Attendance App
user
••••
Login
Class: HCI - Batch SP-2025
Date: Sun May 04 2025
Time: 9:01:38 PM
Farhan-Ul-Haq
Ahmed
Bilal Shah
Submit Attendance

Interface 3:



THE END

RIPHAH INTERNATIONAL UNIVERSITY

Name	**Farhan-Ul-Haq**
Sap id	**55853**
Submitted to	**Sir, Usman Sharif**
Subject	**Human Computer Interaction**

LAB-11

CREATING PERSONAS

Task 1: Persona Creation (Individual Work)

Scenario: Design a food delivery app catering to diverse user types.

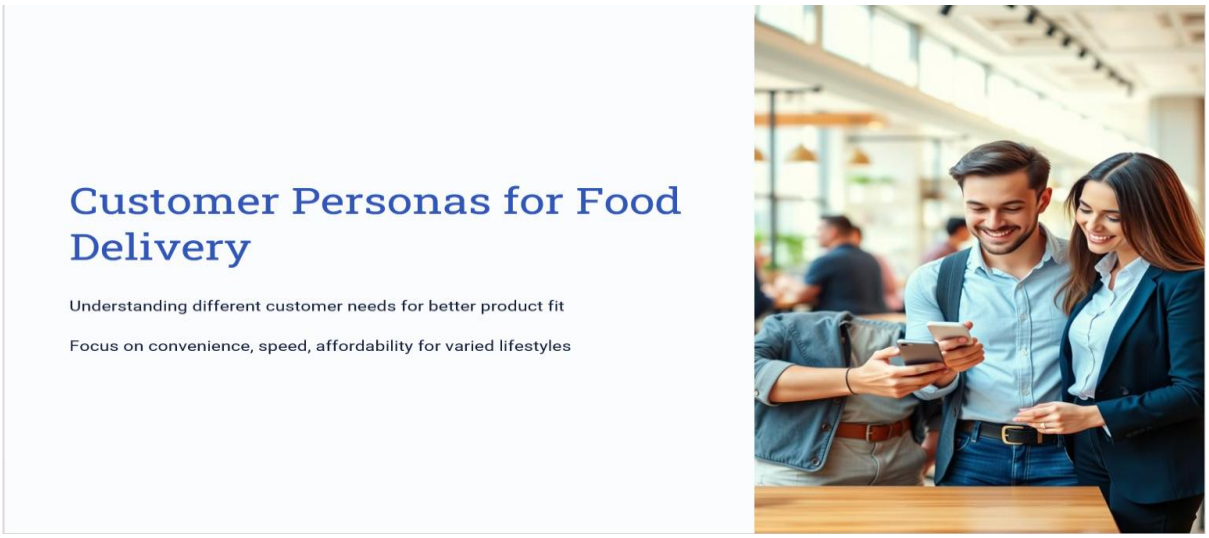
- **Step 1: Conduct Research**
  Identify user types:
  - **Students:** Want budget-friendly, quick options.
  - **Working Professionals:** Need efficiency and healthy meals.
  - **Elderly Users**: Prefer simple navigation with accessibility features.
    Gather behavioral data based on these groups.
- **Step 2: Define Persona Attributes**
  For each persona, include:
  - **Name, Age, Job, Background**
  - **Goals & Needs**
  - **Frustrations & Pain Points**

- **Tech Proficiency**

Example Persona:

- **Farhan (The Busy Professional)**
  - **Age: 25**
  - **Occupation:** Marketing Manager
  - **Goals:** Quick, healthy meals, minimal effort
  - **Frustrations:** Time constraints, dislikes long wait times
  - **Tech Usage:** Smartphone, one-click ordering
- **Hassam (The College Student)**
  - **Age:** 19
  - **Occupation:** Full-time College Student
  - **Goals:** Affordable, filling meals, quick access
  - **Frustrations:** Extra costs (delivery fees), long wait times
  - **Tech Usage:** Frequent smartphone user, prefers app notifications
- **Step 3: Sketch the Persona in Powerpoint**

Slide 1:



For Personas:

Detailed Persona Insights

Farhan (The Busy Professional)

- Age 25, Marketing Manager
- Seeks quick, healthy meals with minimal effort
- Dislikes long wait times, values smartphone one-click ordering

Hassam (The College Student)

- Age 19, Full-time student
- Prioritizes affordable, filling meals
- Frustrated by delivery fees and slow service
- Prefers app notifications, frequent smartphone use

Task 2: Applying Personas in Design (Group Work)

Scenario: Use the personas from Task 1 to brainstorm design improvements for the food delivery app.

- **Step 1:** Discuss Interaction with the App
    - **Farhan:** Needs quick, easy access to healthy meals with a one-click order button.
    - **Hassam:** Wants student discounts and fast, budget-friendly meal options.
- **Step 2: Identify Key Features**
  For Farhan:
    - Favorites for quick reordering
    - Healthy meal filters
    - One-click ordering
    - Real-time delivery tracking

For Hassam:

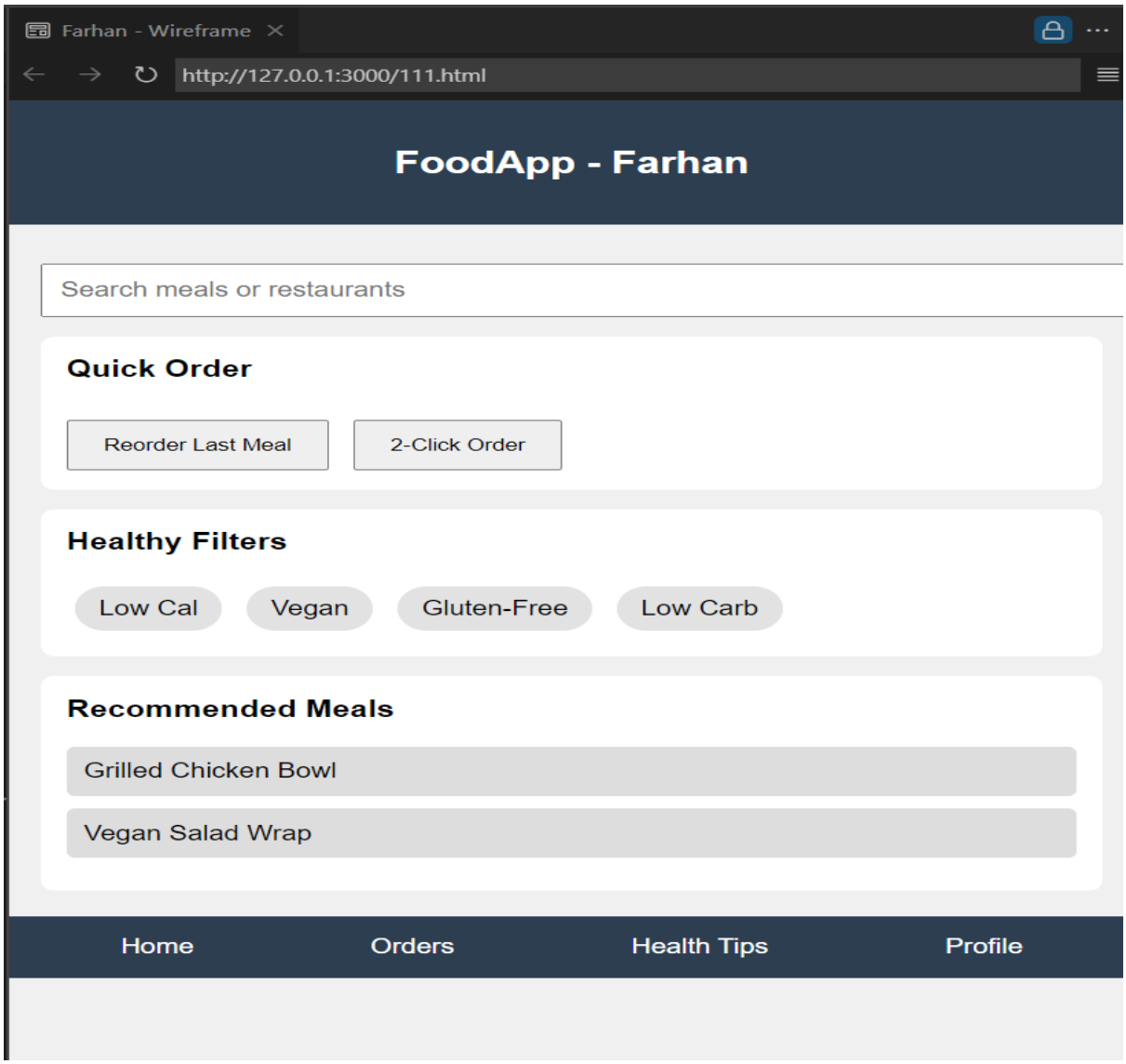
- Discounts & meal bundles
- Quick access to large meals
- Late-night ordering options

- **Step 3: Sketch a Wireframe**
  Create wireframe sketches integrating the features required for each persona.

  For example:

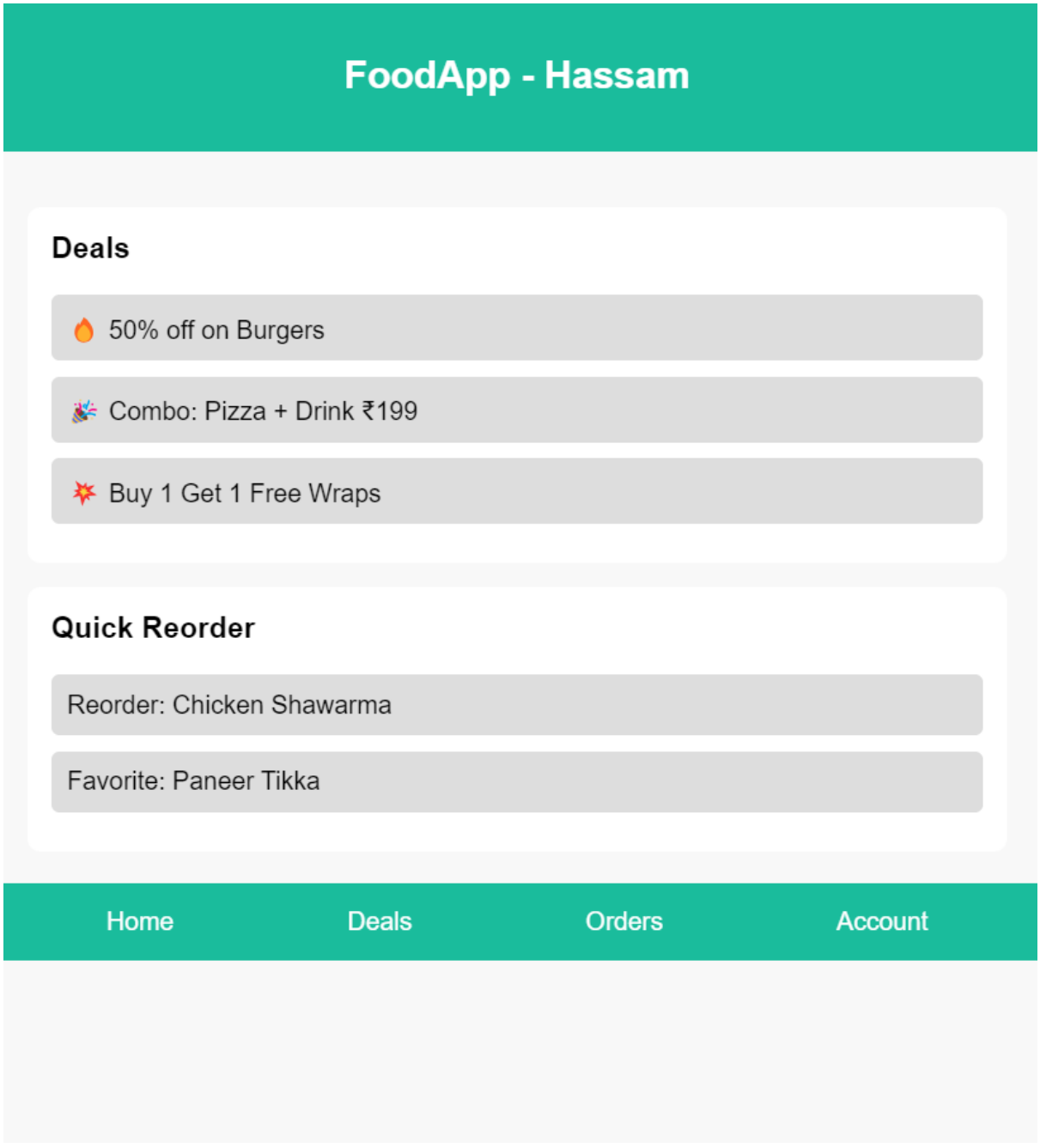
  - **Farhan:** The home screen features a “Quick Order” section and healthy filters.

For Farhan Quick Order + Healthy Filters:



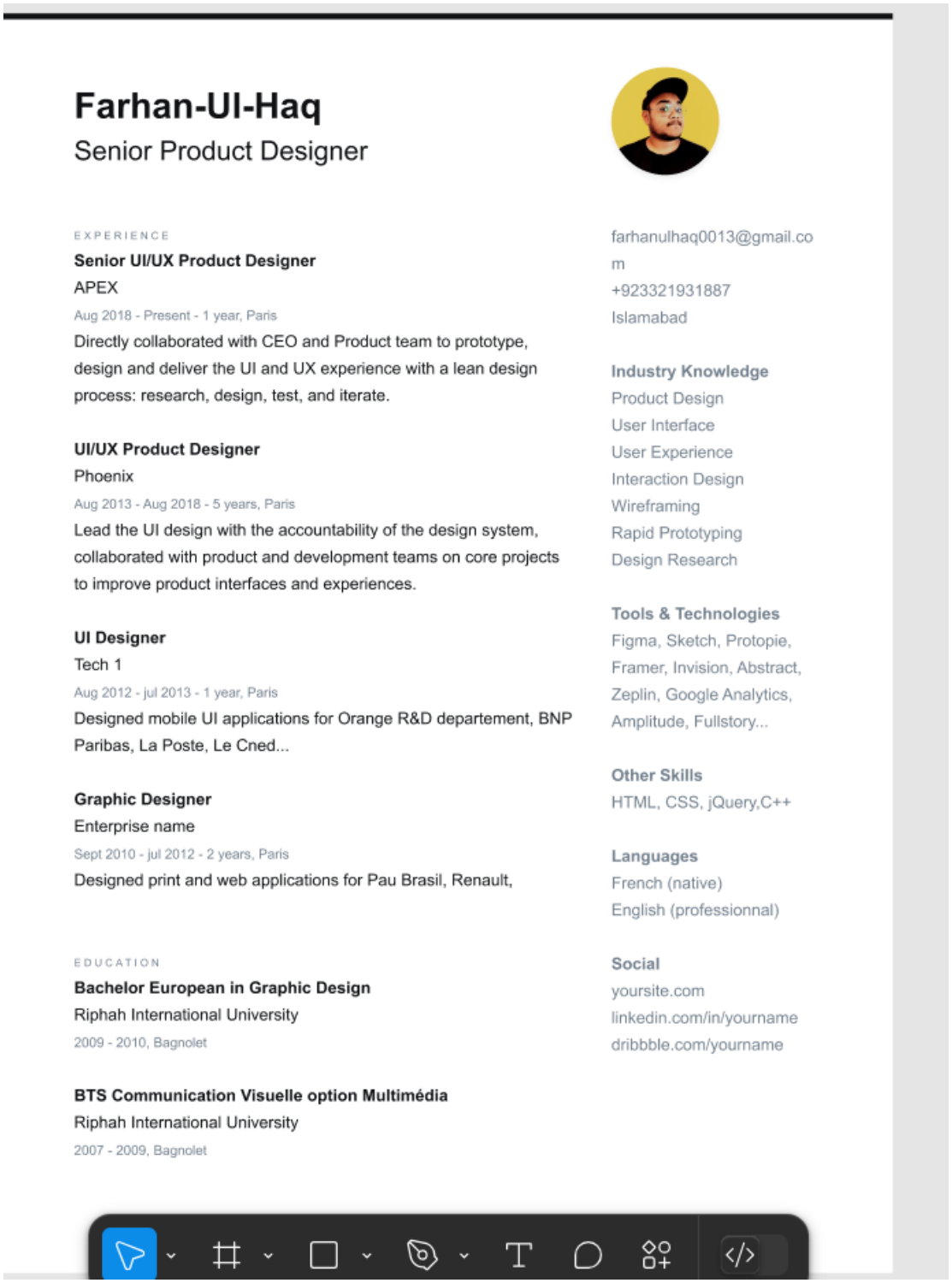
- **Hassam:** The home screen has a "Deals" section and access to quick reorders.

For Hassam Deals + Quick Reorder:



CV for an technical and Non Technical User:

For Technical user:



Farhan-Ul-Haq

Senior Product Designer

EXPERIENCE

Senior UI/UX Product Designer
APEX
Aug 2018 - Present - 1 year, Paris
Directly collaborated with CEO and Product team to prototype, design and deliver the UI and UX experience with a lean design process: research, design, test, and iterate.

UI/UX Product Designer
Phoenix
Aug 2013 - Aug 2018 - 5 years, Paris
Lead the UI design with the accountability of the design system, collaborated with product and development teams on core projects to improve product interfaces and experiences.

UI Designer
Tech 1
Aug 2012 - jul 2013 - 1 year, Paris
Designed mobile UI applications for Orange R&D departement, BNP Paribas, La Poste, Le Cned...

Graphic Designer
Enterprise name
Sept 2010 - jul 2012 - 2 years, Paris
Designed print and web applications for Pau Brasil, Renault,

EDUCATION

Bachelor European in Graphic Design
Riphah International University
2009 - 2010, Bagnolet

BTS Communication Visuelle option Multimédia
Riphah International University
2007 - 2009, Bagnolet

farhanulhaq0013@gmail.com
+923321931887
Islamabad

Industry Knowledge
Product Design
User Interface
User Experience
Interaction Design
Wireframing
Rapid Prototyping
Design Research

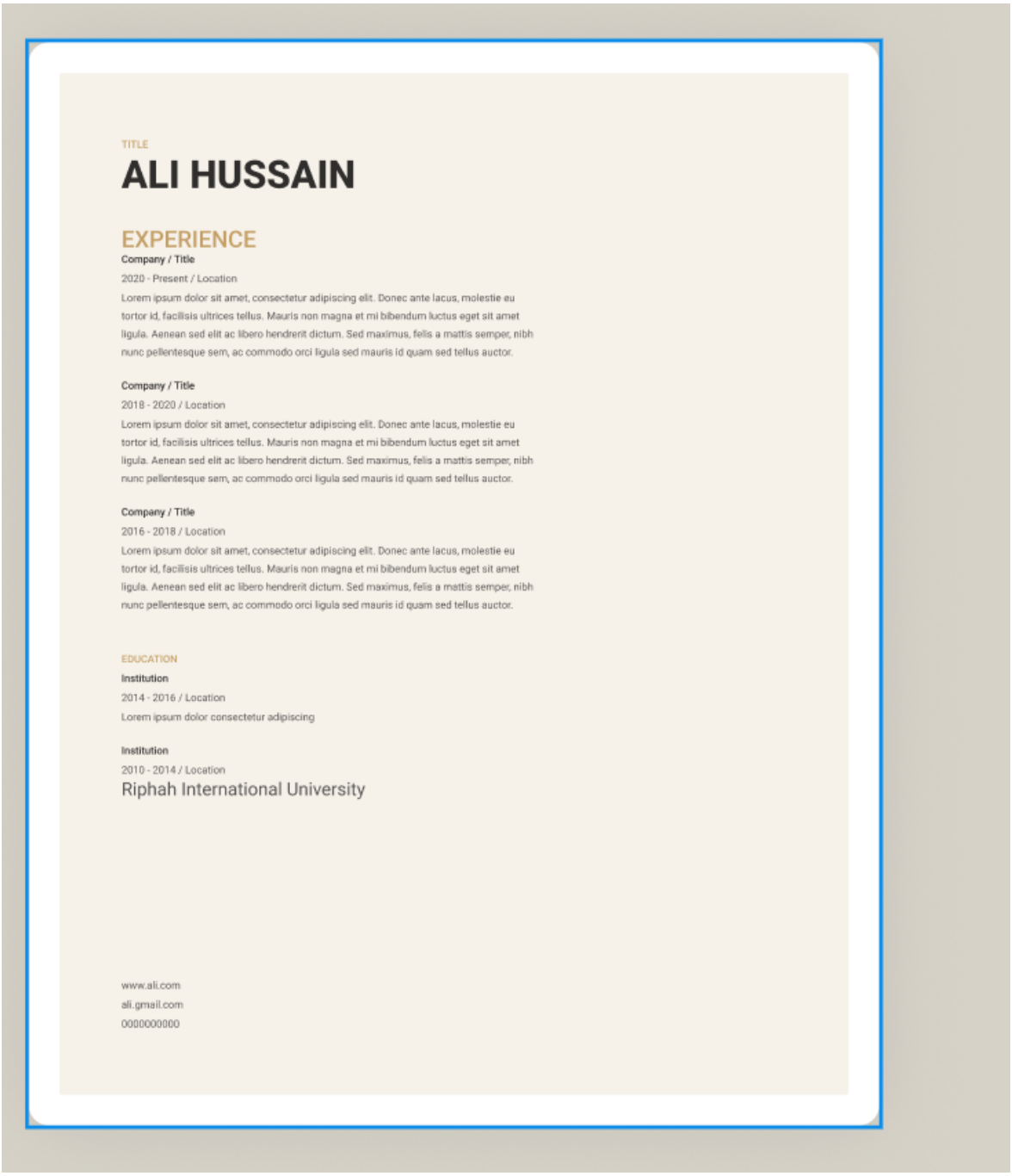
Tools & Technologies
Figma, Sketch, Protopie, Framer, Invision, Abstract, Zeplin, Google Analytics, Amplitude, Fullstory...

Other Skills
HTML, CSS, jQuery,C++

Languages
French (native)
English (professionnal)

Social
yoursite.com
linkedin.com/in/yourname
dribbble.com/yourname

For NON-Technical user:



TITLE

ALI HUSSAIN

EXPERIENCE

Company / Title

2020 - Present / Location

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ante lacus, molestie eu tortor id, facilisis ultrices tellus. Mauris non magna et mi bibendum luctus eget sit amet ligula. Aenean sed elit ac libero hendrerit dictum. Sed maximus, felis a mattis semper, nibh nunc pellentesque sem, ac commodo orci ligula sed mauris id quam sed tellus auctor.

Company / Title

2018 - 2020 / Location

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ante lacus, molestie eu tortor id, facilisis ultrices tellus. Mauris non magna et mi bibendum luctus eget sit amet ligula. Aenean sed elit ac libero hendrerit dictum. Sed maximus, felis a mattis semper, nibh nunc pellentesque sem, ac commodo orci ligula sed mauris id quam sed tellus auctor.

Company / Title

2016 - 2018 / Location

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ante lacus, molestie eu tortor id, facilisis ultrices tellus. Mauris non magna et mi bibendum luctus eget sit amet ligula. Aenean sed elit ac libero hendrerit dictum. Sed maximus, felis a mattis semper, nibh nunc pellentesque sem, ac commodo orci ligula sed mauris id quam sed tellus auctor.

EDUCATION

Institution

2014 - 2016 / Location

Lorem ipsum dolor consectetur adipiscing

Institution

2010 - 2014 / Location

Riphah International University

www.ali.com

ali.gmail.com

0000000000