



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)

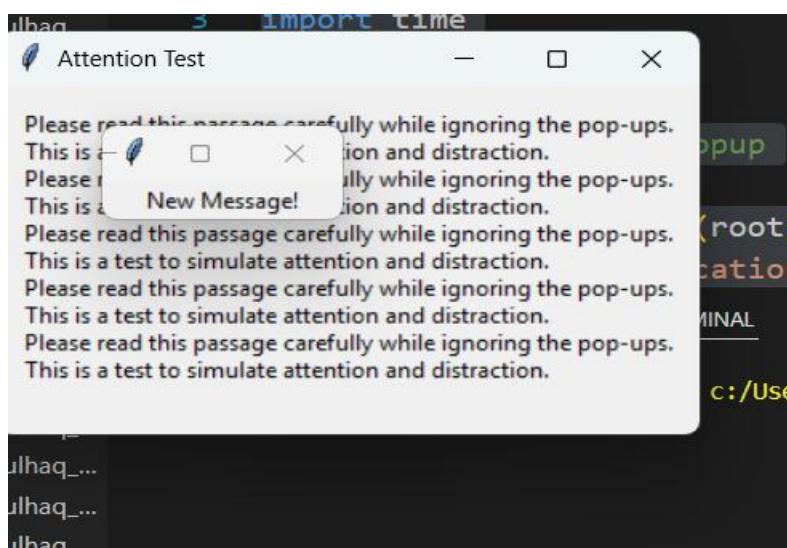
# Start popup thread

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.

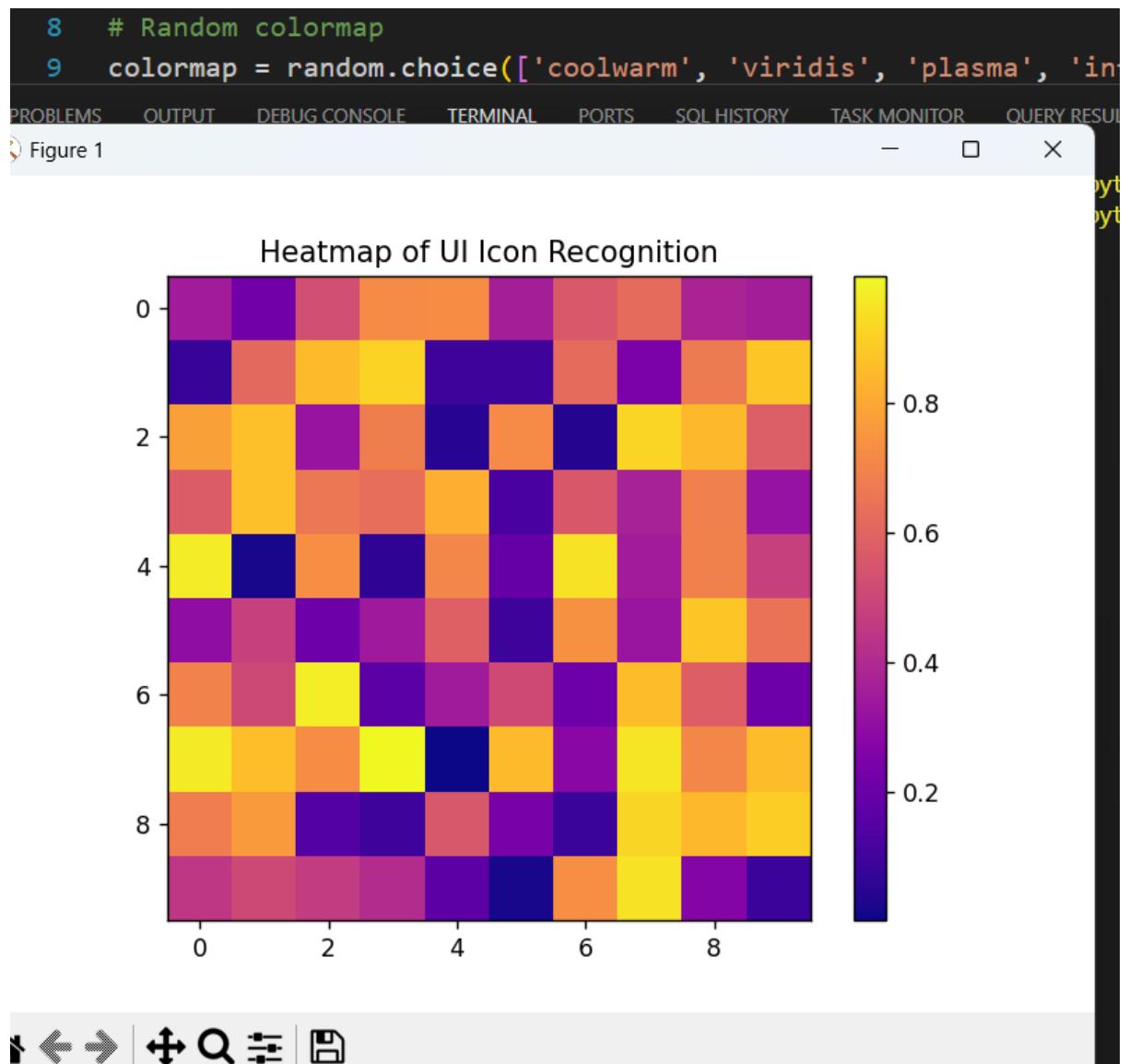
Tools:

- 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:

The screenshot shows a terminal window with several tabs open. The current tab is 'tTinker.py' which contains Python code for creating a Tkinter login form. The code defines a 'submit' function that prints the submitted username and password. It then creates a Tkinter window titled 'Login Form'. On the left side of the terminal, there is a preview of the Tkinter window showing fields for 'Username' and 'Password' with a 'Login' button.

```
File Edit Selection View Go Run Terminal Help
Username: 55853
Password: msa
Login
55853_Farhanulhaq...
Attendance.html
attendance.txt
DataMainuplation.py
Selenium.py tTinker.py Syntax.docx
tTinker.py > ...
1 import tkinter as tk
2
3 def submit():
4     username = entry_user.get()
5     password = entry_pass.get()
6     print(f"Submitted:\nUsername: {username}, Password: {password}")
7
8     root = tk.Tk()
9     root.title("Login Form")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
PS C:\Users\farha\OneDrive\HCI--Lab> & c:/Users/farha/OneDrive/HCI--Lab/.venv\Scripts\python.exe tTinker.py
Submitted:
Username: 145, Password: msa
Submitted:
Username: 55853, Password: msa
```

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:

The screenshot shows a terminal window with the following output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR QUERY RESULTS (PREVIEW)

PS C:\Users\farha\OneDrive\HCI--Lab> & c:/Users/farha/OneDrive/HCI--Lab/.venv/scripts/python.exe c:/Users/farha/OneDrive/HCI--Lab/task_completion.py
Perform 5 simple tasks (e.g., clicking, drawing, filling forms)...
Task 1: Press Enter when done...
Start performing the task... Press Enter when finished.
Task 2: Press Enter when done...
Start performing the task... Press Enter when finished.
Task 3: Press Enter when done...
Start performing the task... Press Enter when finished.
Task 4: Press Enter when done...
Start performing the task... Press Enter when finished.
Task 5: Press Enter when done...
Start performing the task... Press Enter when finished.

Task Completion Times:
 Task  Time Taken (s)
0      1      2.56
1      2      1.27
2      3      0.30
3      4      0.18
4      5      0.19

```

PS C:\Users\farha\OneDrive\HCI--Lab>

Summary Table of Lab 4

<u>Activity</u>	<u>Focus</u>	<u>Tools Used</u>	<u>Output</u>
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