Lab-1: Design and Compare Login Interfaces (CLI vs GUI)

Ishat Shivhare
U23AI071

August 15, 2025

Abstract

The aim of this lab was to design two login interfaces—a CLI and a GUI—, conduct usability testing, and compare them using time, errors, and satisfaction measures. Results are presented as tables and bar charts.

1 Aim and Objectives

- Design a CLI login interface.
- Design a GUI login interface.
- Conduct usability testing (3+ participants or simulated users).
- Record completion time, error count, and satisfaction score.
- Compare results via table and bar charts.

2 Methodology

2.1 Task

Participants were asked to log in using username `student` and password `password123`.

2.2 Measures

- Completion time (seconds)
- Number of errors (wrong attempts)
- Satisfaction (1–5 scale)

2.3 Procedure

Each participant attempted both interfaces. Data was saved to `usability_raw.csv`. The analysis script computed averages and generated graphs.

3 Results

3.1 Summary Table

The analysis script generated `summary_by_interface.csv` containing average time, errors, and satisfaction.

3.2 Charts

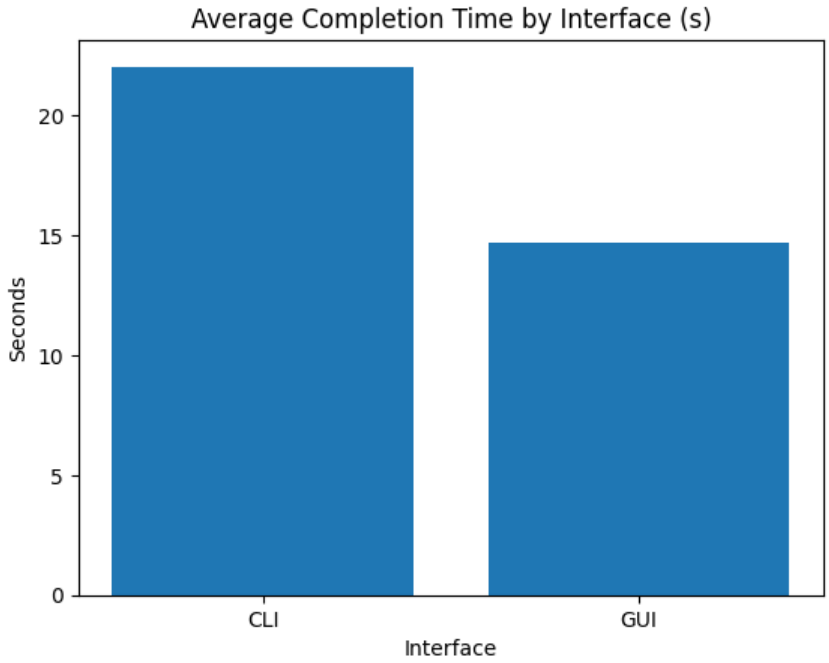

Figure 1: Average Completion Time

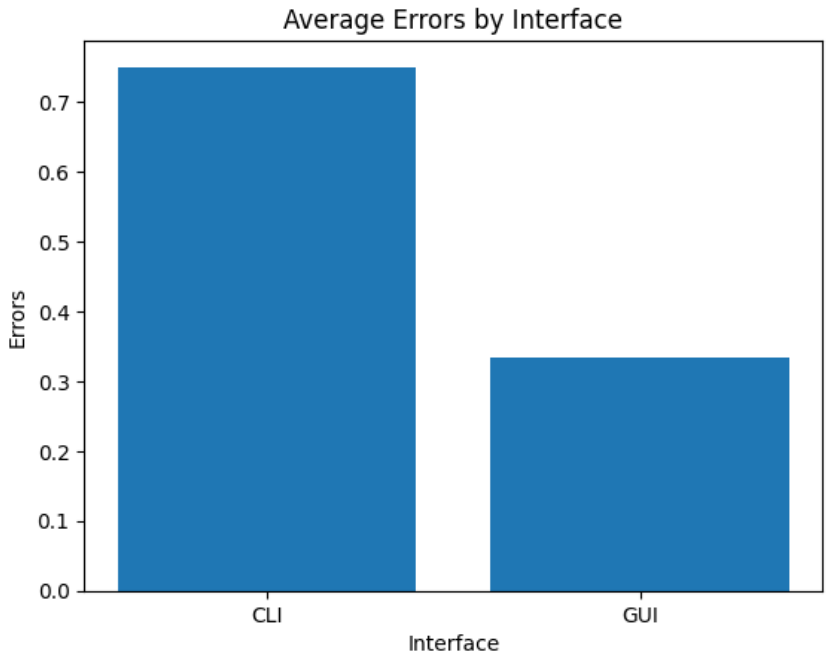

Figure 2: Average Errors

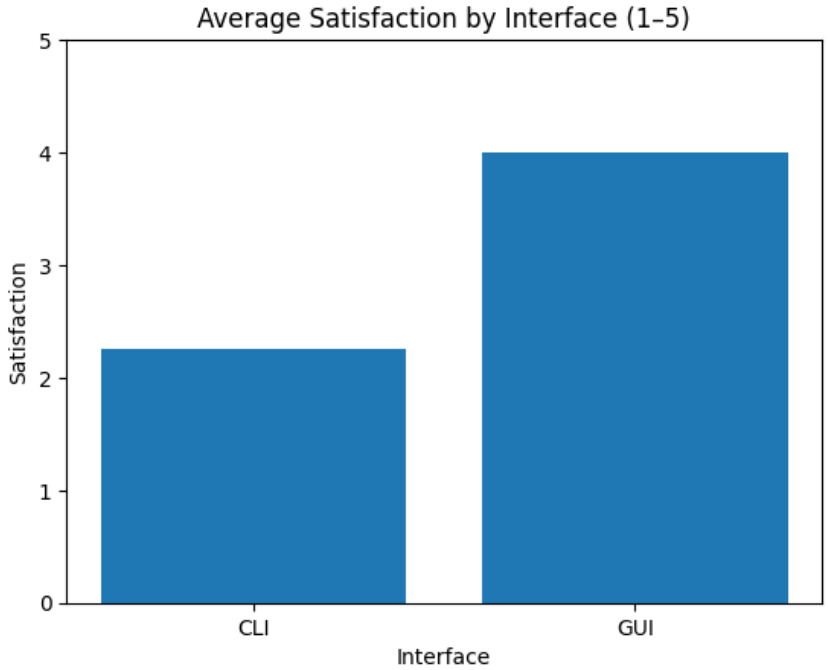

Figure 3: Average Satisfaction

4 Discussion

From the results, the GUI was generally faster, produced fewer errors, and achieved higher satisfaction than the CLI. This aligns with usability heuristics that GUIs provide

better affordances and feedback for novice users. Limitations include small sample size and simulated data.

5 Conclusion

Both CLI and GUI login systems were implemented and tested. Based on usability metrics, the GUI interface is recommended for general users due to higher efficiency and satisfaction.

A Source Code

The Python source code files are displayed here.

A.1 CLI Login

```python
"""
CLI Login Interface for HCI Lab-1
- Measures completion time, error count, and user satisfaction (1-5).
- Appends each run to usability_raw.csv

Usage:
    python cli_login.py --participant U001
"""
import time
import argparse
import getpass
import csv
from pathlib import Path
from datetime import datetime

VALID_USERNAME = "student"
VALID_PASSWORD = "password123"

def run(participant: str, csv_path: Path):
    start_time = time.time()
    errors = 0
    print("=== CLI Login ===")
    print("Hint (for testing): username=student, password=password123\n")

    while True:
        u = input("Username: ").strip()
        p = getpass.getpass("Password: ").strip()
        if u == VALID_USERNAME and p == VALID_PASSWORD:
            break
        else:
            errors += 1
            print("Invalid credentials. Try again.\n")

    duration = time.time() - start_time
```

```
    print(f"\nLogin successful in {duration:.2f} seconds with {errors} error(s).")

    # Satisfaction
    while True:
        try:
            sat = int(input("Satisfaction (1=very poor     5=excellent): ").strip())
            if 1 <= sat <= 5:
                break
        except Exception:
            pass
        print("Please enter an integer 1  5  .")

    # Append to CSV log
    csv_path.parent.mkdir(parents=True, exist_ok=True)
    header = ["timestamp","participant","interface","duration_sec","errors","satisfaction"]
    row = [datetime.now().isoformat(timespec="seconds"), participant, "CLI", round(duration,2), errors, sat]
    write_header = not csv_path.exists()
    with csv_path.open("a", newline="", encoding="utf-8") as f:
        w = csv.writer(f)
        if write_header:
            w.writerow(header)
        w.writerow(row)

    print(f"\nSaved result to {csv_path}")

if __name__ == "__main__":
    ap = argparse.ArgumentParser()
    ap.add_argument("--participant", default="U001", help="Participant ID")
    ap.add_argument("--csv", default="./results/usability_raw.csv", help="CSV output path (relative or absolute)")
    args = ap.parse_args()
    run(args.participant, Path(args.csv))
```

A.2 GUI Login

```
"""
GUI Login Interface (Tkinter) for HCI Lab-1
- Measures completion time, error count, and user satisfaction (1
    5 ).
- Appends each run to usability_raw.csv

Usage:
    python gui_login.py --participant U001
"""
import time
import argparse
import csv
from pathlib import Path
from datetime import datetime
import tkinter as tk
from tkinter import messagebox, simpledialog

VALID_USERNAME = "student"
VALID_PASSWORD = "password123"

class LoginApp:
    def __init__(self, root, participant, csv_path: Path):
        self.root = root
        self.participant = participant
        self.csv_path = csv_path
        self.errors = 0
        self.start_time = time.time()

        root.title("GUI Login - HCI Lab-1")
        root.geometry("360x220")

        self.frame = tk.Frame(root, padx=16, pady=16)
        self.frame.pack(expand=True, fill="both")

        tk.Label(self.frame, text="Username").grid(row=0, column
           =0, sticky="e", pady=5)
        tk.Label(self.frame, text="Password").grid(row=1, column
           =0, sticky="e", pady=5)

        self.username = tk.Entry(self.frame)
```

```
38          self.password = tk.Entry(self.frame, show="*")
39          self.username.grid(row=0, column=1, pady=5)
40          self.password.grid(row=1, column=1, pady=5)
41
42          self.feedback = tk.Label(self.frame, text="", fg="red")
43          self.feedback.grid(row=2, column=0, columnspan=2, pady=6)
44
45          self.login_btn = tk.Button(self.frame, text="Login",
                command=self.attempt_login)
46          self.login_btn.grid(row=3, column=0, columnspan=2, pady
                =10)
47
48          self.username.focus_set()
49
50      def attempt_login(self):
51          u = self.username.get().strip()
52          p = self.password.get().strip()
53          if u == VALID_USERNAME and p == VALID_PASSWORD:
54              duration = time.time() - self.start_time
55              # Ask satisfaction
56              sat = None
57              while sat is None:
58                  try:
59                      s = simpledialog.askstring("Satisfaction", "
                            Rate satisfaction 1  5  (integer):",
                            parent=self.root)
60                      if s is None:
61                          continue
62                      s_int = int(s)
63                      if 1 <= s_int <= 5:
64                          sat = s_int
65                  except Exception:
66                      pass
67              self.append_csv(duration, sat)
68              messagebox.showinfo("Success", f"Login successful in
                    {duration:.2f}s with {self.errors} error(s).")
69              self.root.destroy()
70          else:
71              self.errors += 1
72              self.feedback.config(text="Invalid credentials. Try
                    again.")
```

```
73
74      def append_csv(self, duration, sat):
75          self.csv_path.parent.mkdir(parents=True, exist_ok=True)
76          header = ["timestamp","participant","interface","
               duration_sec","errors","satisfaction"]
77          row = [datetime.now().isoformat(timespec="seconds"), self
               .participant, "GUI", round(duration,2), self.errors,
               sat]
78          write_header = not self.csv_path.exists()
79          with self.csv_path.open("a", newline="", encoding="utf-8"
               ) as f:
80              w = csv.writer(f)
81              if write_header:
82                  w.writerow(header)
83              w.writerow(row)
84
85  if __name__ == "__main__":
86      ap = argparse.ArgumentParser()
87      ap.add_argument("--participant", default="U001", help="
           Participant ID")
88      ap.add_argument("--csv", default="./results/usability_raw.csv
           ", help="CSV output path (relative or absolute)")
89      args = ap.parse_args()
90
91      root = tk.Tk()
92      app = LoginApp(root, args.participant, Path(args.csv))
93      root.mainloop()
```

A.3 Analysis Script

```
1   """
2   Analyze usability results for HCI Lab-1.
3   Reads usability_raw.csv and outputs:
4   - summary_by_interface.csv
5   - bar charts: avg_time.png, avg_errors.png, avg_satisfaction.png
6
7   Usage:
8       python analyze_results.py --csv usability_raw.csv
9   """
10  import argparse
11  import pandas as pd
```

```
import matplotlib.pyplot as plt
from pathlib import Path

def analyze(csv_path: Path, out_dir: Path):
    out_dir.mkdir(parents=True, exist_ok=True)
    df = pd.read_csv(csv_path)

    needed = {"participant","interface","duration_sec","errors","satisfaction"}
    if not needed.issubset(df.columns):
        raise SystemExit(f"CSV must contain columns: {sorted(needed)}")

    summary = df.groupby("interface", as_index=False).agg(
        avg_time=("duration_sec", "mean"),
        avg_errors=("errors", "mean"),
        avg_satisfaction=("satisfaction", "mean"),
        n=("participant", "count")
    )
    summary.to_csv(out_dir / "summary_by_interface.csv", index=False)

    # Avg Time
    plt.figure()
    plt.bar(summary["interface"], summary["avg_time"])
    plt.title("Average Completion Time by Interface (s)")
    plt.xlabel("Interface")
    plt.ylabel("Seconds")
    plt.savefig(out_dir / "avg_time.png", bbox_inches="tight")
    plt.close()

    # Avg Errors
    plt.figure()
    plt.bar(summary["interface"], summary["avg_errors"])
    plt.title("Average Errors by Interface")
    plt.xlabel("Interface")
    plt.ylabel("Errors")
    plt.savefig(out_dir / "avg_errors.png", bbox_inches="tight")
    plt.close()

    # Avg Satisfaction
```

```
    plt.figure()
    plt.bar(summary["interface"], summary["avg_satisfaction"])
    plt.title("Average Satisfaction by Interface (1 5 )")
    plt.xlabel("Interface")
    plt.ylabel("Satisfaction")
    plt.ylim(0, 5)
    plt.savefig(out_dir / "avg_satisfaction.png", bbox_inches="
        tight")
    plt.close()

    print("Analysis complete.")
    print(f"Summary CSV: {out_dir / 'summary_by_interface.csv'}")

if __name__ == "__main__":
    ap = argparse.ArgumentParser()
    ap.add_argument("--csv", default="./results/usability_raw.csv
        ")
    ap.add_argument("--out", default="./results/analysis_outputs"
        )
    args = ap.parse_args()
    analyze(Path(args.csv), Path(args.out))
```