Project 1 - Automating Usability Testing

Start Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a website url
Available May 12 at 12am - May 25 at 11:59pm

Introduction

Usability testing evaluates whether a product is usable by its intended user population to accomplish tasks effectively. This assignment takes a hands-on approach where students will **develop a Streamlit app** to automate usability testing tasks. The app will handle participant interaction, data collection (e.g., questionnaires, task completion time), and result aggregation. The goal is to create a tool that facilitates structured usability testing and generates a final usability report.

This is a great project to use your knowledge in human-computer interaction, applying the usability goals and design principles to come up with an automated usability testing. Usability testing itself is an important part of HCI, but believe it or not, it is still performed very rudimentarily. After some years teaching this course, I had the idea of having the students automate the usability testing project.

The most interesting part of this is that in this first project, you create the tool for usability testing, then in the next project, you will be asked to create a cool data-driven app, and then, in the third project, you will use your own usability testing tool developed in Project 1 to test Project 2. How cool is that?

Objective

This assignment focuses on creating a **Streamlit app** designed to automate usability testing tasks. The app will serve as a tool to guide users through a usability test, collect relevant data, and generate an aggregated report. This assignment does not involve conducting usability tests but focuses on the **development** of the tool.

A usability testing tool typically supports these steps:

- Presents consent and demographic forms.
- Guides participants through tasks.
- Collects data on task performance and feedback.
- Aggregates and visualizes results.

Steps

Step 1: Research Usability Testing Review the key aspects of usability testing, including test design, task creation, and feedback collection, from https://www.nngroup.com/articles/usability-testing-101

Step 2: Design the Usability Testing Workflow Plan the workflow for the app, which should include:

1. **Consent Form**: A form where users acknowledge and accept the terms of the usability test.
2. **Demographic Questionnaire:** Collect basic demographic information, such as age, occupation, and familiarity with similar tools.
3. **Task Guidance**: Present a set of tasks for participants to complete, and record data such as time taken and success/failure.
4. **Exit Questionnaire**: Gather participant feedback about their experience with the tested product.
5. **Report Generation**: Aggregate and summarize data from the above steps.

Step 3: Build the Streamlit App Use Python's **Streamlit** framework to implement the following features:

1. **Home Page**:
   - Introduce the usability test.
   - Include navigation to other sections.
2. **Consent Form Page or Tab**:
   - Display a customizable consent form.
   - Include a checkbox for digital acceptance.
3. **Demographic Questionnaire Page**:
   - Use input forms (e.g., text boxes, dropdowns, radio buttons).
   - Save responses to a file (e.g., CSV or JSON).
4. **Task Page**:
   - Allow the user to select a task and track task-related inputs (e.g., completion time, success/failure).
   - Include a text box for observers to add notes.
5. **Exit Questionnaire Page**:
   - Provide a mix of quantitative (e.g., Likert scale) and qualitative (e.g., open-ended) questions.
   - Save responses to a file.
6. **Report Page**:
   - Summarize all collected data.
   - Include visualizations like bar charts or tables.

Please download the files with examples of each one of the necessary usability testing steps. These are just samples, you do not need to automate each one exactly. I am posting them here so you can see how usability testing is done when not automated. Form can be downloaded here. ↓

Step 4: Test Your App Run your app and ensure that:

- All forms and features function as expected.
- Data is correctly saved and can be retrieved for report generation.

Additional Requirement: Create a Video Presentation

1. **Create a 3-Minute Video**
   - Record a **short video (3 minutes max)** explaining and demonstrating your project.
   - The video must include:
     - A brief overview of your code structure (just the main components, not detailed explanations).
     - A demonstration of running the automated usability testing tool.
     - At least one example of interacting with the tool (no real data, just some mockup data).
   - You **do not need to show your face**
   - in the video, but doing so is encouraged for a personal touch.
2. **Upload to YouTube**
   - Upload your video to YouTube.
   - You may set the video as **Unlisted** so that only people with the link can view it.
   - Submit the YouTube link along with your project.

Deliverables

1. A GitHub link containing:
   - Your Streamlit app code (`.py` file).
   - Any additional resources (e.g., sample CSV files, README for setup instructions).
   - Explanation of the tool's features and workflow (Optional)
2. The YouTube link to your project demonstration video.

Example Template

Code template:

```
import streamlit as st
import pandas as pd
import time
import os

# Create a folder called data in the main project folder
DATA_FOLDER = "data"
if not os.path.exists(DATA_FOLDER):
    os.makedirs(DATA_FOLDER)

# Define CSV file paths for each part of the usability testing
CONSENT_CSV = os.path.join(DATA_FOLDER, "consent_data.csv")
DEMOGRAPHIC_CSV = os.path.join(DATA_FOLDER, "demographic_data.csv")
TASK_CSV = os.path.join(DATA_FOLDER, "task_data.csv")
EXIT_CSV = os.path.join(DATA_FOLDER, "exit_data.csv")


def save_to_csv(data_dict, csv_file):
    # Convert dict to DataFrame with a single row
    df_new = pd.DataFrame([data_dict])
    if not os.path.isfile(csv_file):
        # File doesn't exist, write with headers
        df_new.to_csv(csv_file, mode='w', header=True, index=False)
    else:
        # File, we need to append without writing the header!
        df_new.to_csv(csv_file, mode='a', header=False, index=False)


def load_from_csv(csv_file):
    if os.path.isfile(csv_file):
        return pd.read_csv(csv_file)
    else:
        return pd.DataFrame()


def main():
    st.title("Usability Testing Tool")

    home, consent, demographics, tasks, exit, report = st.tabs(["Home", "Consent", "Demographics", "Task", "Exit Questionnaire", "Report"])

    with home:
        st.header("Introduction")
        st.write("""
        Welcome to the Usability Testing Tool for HCI.

        In this app, you will:
        1. Provide consent for data collection.
        2. Fill out a short demographic questionnaire.
        3. Perform a specific task (or tasks).
        4. Answer an exit questionnaire about your experience.
        5. View a summary report (for demonstration purposes).
        """)


    with consent:
        st.header("Consent Form")

        # TODO: Create your consent form and a variable called consent_given

        if st.button("Submit Consent"):
            if not consent_given:
                st.warning("You must agree to the consent terms before proceeding.")
            else:
                # Save the consent acceptance time
                data_dict = {
                    "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
                    "consent_given": consent_given
                }
                save_to_csv(data_dict, CONSENT_CSV)

    with demographics:
        st.header("Demographic Questionnaire")

        with st.form("demographic_form"):
            #TODO: Create the demographic form
            submitted = st.form_submit_button("Submit Demographics")
            if submitted:
                data_dict = {
                    "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
                    "name": name,
                    "age": age,
                    "occupation": occupation,
                    "familiarity": familiarity
                }
                save_to_csv(data_dict, DEMOGRAPHIC_CSV)

    with tasks:
        st.header("Task Page")

        st.write("Please select a task and record your experience completing it.")

        # For this template, we assume there's only one task, in project 3, we will have to include the actual tasks
        selected_task = st.selectbox("Select Task", ["Task 1: Example Task"])
        st.write("Task Description: Perform the example task in our system...")

        # Track success, completion time, etc.
        start_button = st.button("Start Task Timer")
        if start_button:
            st.session_state["start_time"] = time.time()

        stop_button = st.button("Stop Task Timer")
        if stop_button and "start_time" in st.session_state:
            duration = time.time() - st.session_state["start_time"]
            st.session_state["task_duration"] = duration

        success = st.radio("Was the task completed successfully?", ["Yes", "No", "Partial"])
        notes = st.text_area("Observer Notes")

        if st.button("Save Task Results"):
            duration_val = st.session_state.get("task_duration", None)

            data_dict = {
                "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
                "task_name": selected_task,
                "success": success,
                "duration_seconds": duration_val if duration_val else "",
                "notes": notes
            }
            save_to_csv(data_dict, TASK_CSV)

            # Reset any stored time in session_state if you'd like
            if "start_time" in st.session_state:
                del st.session_state["start_time"]
            if "task_duration" in st.session_state:
                del st.session_state["task_duration"]


    with exit:
        st.header("Exit Questionnaire")

        with st.form("exit_form"):
            # TODO: likert scale or other way to have an exit questionnaire

            submitted_exit = st.form_submit_button("Submit Exit Questionnaire")
            if submitted_exit:
                data_dict = {
                    "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
                    "satisfaction": satisfaction,
                    "difficulty": difficulty,
                    "open_feedback": open_feedback
                }
                save_to_csv(data_dict, EXIT_CSV)
                st.success("Exit questionnaire data saved.")

    with report:
        st.header("Usability Report - Aggregated Results")

        st.write("""Consent Data""")
        consent_df = load_from_csv(CONSENT_CSV)
        if not consent_df.empty:
            st.dataframe(consent_df)
        else:
            st.info("No consent data available yet.")

        st.write("""Demographic Data""")
        demographic_df = load_from_csv(DEMOGRAPHIC_CSV)
        if not demographic_df.empty:
            st.dataframe(demographic_df)
        else:
            st.info("No demographic data available yet.")

        st.write("""Task Performance Data""")
        task_df = load_from_csv(TASK_CSV)
        if not task_df.empty:
            st.dataframe(task_df)
        else:
            st.info("No task data available yet.")

        st.write("""Exit Questionnaire Data""")
        exit_df = load_from_csv(EXIT_CSV)
        if not exit_df.empty:
            st.dataframe(exit_df)
        else:
            st.info("No exit questionnaire data available yet.")

        # Example of aggregated stats (for demonstration only)
        if not exit_df.empty:
            st.subheader("Exit Questionnaire Averages")
            avg_satisfaction = exit_df["satisfaction"].mean()
            avg_difficulty = exit_df["difficulty"].mean()
            st.write(f"""Average Satisfaction: {avg_satisfaction:.2f}""")
            st.write(f"""Average Difficulty: {avg_difficulty:.2f}""")


if __name__ == "__main__":
    main()
```

- Customize the consent text, demographic questions, tasks, and exit questions to match your exact idea.
- This example saves data in CSV files. If you wish, you might switch to a database or JSON format.
- The "Report" tab just displays a simple data table and a couple of example stats. You can expand this with charts (e.g., `st.bar_chart`, `matplotlib`, `altair`) to visualize results in more interesting ways (data visualization is a passion of mine!).
- **The tasks will be defined in Project 3, for now, only placeholders.**

Next ▸

Submission

✕ **Not Submitted!**
Submission Details

Grade: 0% (100 pts possible)
Graded Anonymously: no

Comments:
No Comments