ExplainUX

Jerome Uriah Ng
45461844

Code Overview

Code Title :ExplainUX
Student Name: Jerome Uriah Ng Jia Jun
Date:26/10/2025
Repository Link https://github.com/JeromeUriah/ExplainUX.git

ExplainUX Overview

ExplainUX is a Streamlit-based interactive evaluation chatbot design to assist students and novice designers to conduct structured heuristic evaluations of digital interfaces. This program aims to simplify traditionally expert-driven UX evaluation process by transforming it into a guided, conversational workflow powered by Large Language Models (LLMs). Within the interface, users will input a brief description of a target interface or module, select relevant usability heuristics to evaluate and the system will automatically generate scores, justification and ethical reflections. These outputs are created in real time and presented with a downloadable CSV results table, supports reflection, comparison and documentation for reports or assignments.

ExplainUX currently is integrated with a set of Nielsens Usability Heuristics which covers principles such as visibility of system status, user control, error prevention and minimalism and embeds them within a multi-agent prompt architecture. This architecture includes three coordinated LLM-driven agents:

1. The Clarifier Agent: Interprets user description and asks brief follow-up questions to ensure context accuracy.
2. The Scorer Agent: Generates heuristics compliance rating (1–10) with text rationales and actionable recommendations.
3. The Ethics Reviewer Agent: Reflects on potential fairness, accessibility or ethical concerns. Prompting designers to consider diverse users perspectives and responsible AI principles.

With this three-agent design, ExplainUX encourages reflexive thinking about social and ethical implications in usability evaluations. This embodies Human-Centered AI values by balancing both automation and human judgment by providing transparent, explainable reasoning while leaving final interpretation to users. It acts as an educational tool and a reflective evaluation companion which helps students internalize usability heuristics, recognize bias and document their evaluative reasoning in a structured and replicable manner.

Main Features and Structure

Project Layout:

|_ app.py
|_ agents.py
|_ heuristics.py
|_ requirements.txt
|_ ,env

UI and Data Handling:

- Inputs (Interface description, heuristic multi-selection, run evaluation button)
- Interactive results grid with columns for Heuristic, Score, Why, Improvements and Ethical reflections
- CSV export
- Debug section that lists available models from SDK

Framework and Runtime

- Python + Streamlit allows a reactive and lightweight UI

Configurations

- Requirements.txt imports all dependencies

Model Integration

- Google GenAI SDK (Gemini 2.5)

Multi-Agent Prompts

- CLARIFIER_AGENT: Verifies context sufficiency
- SCORER_AGENT: Output numeric score, “Why” and actionable improvements
- ETHICS_AGENT: Fairness, inclusion and accessibility reflections with mitigations

Heuristic Catalog

- Heuristic.py contains all 10 Nielsen's heuristics with id, name, description, diagnostic questions and a fairness consideration.
- Separates domain knowledge from main chunk of code allowing for addition of other heuristic principles easier.

Data Flow:

1. User inputs summary of interface and select heuristics
2. App.py builds a structured data package to send to LLM
3. Each agent will return a JSON that is the put into rows
4. Rows are then aggregated into a dataframe and displayed through st.data_editor and can be downloaded into a CSV

Core Logic

1. Collect Inputs (UI)
   - Streamlit renders a text area for the interface descriptions and multiselect
   - Users click "Run Evaluation"
2. Build data package per heuristic
   - For each selected heuristic, the app will create a small and structured data package made up of the interface summary, heuristic name and guiding questions
   - Data package is then passed through each agent in agent.py
     i. Clarifier – ask minimal follow up questions for context sufficiency
     ii. Scorer – returns score, why, improvements
     iii. Ethics Reviewer – returns risk, affected users and mitigations
3. Call the LLM with strict JSON
   - App.py uses Google GenAI SDK
   - Prompts are structured with response_mime_type parameter set to "application/json" ensuring that outputs are a consistent, machine-readable JSON format
4. Aggregate Results
   - Each heuristic produces one row and is combined into a pandas dataframe
5. Display and Export
   - Steamlit shows the table and provides CSV download
   - Debug section can be expanded to see all available modesl to verify SDK setup
6. Configuration and Data
   - .env will hold GOOGLE_API_KEY (loaded thru dotenv)
   - Requirements.txt pins imports google-generativeai >= 0.8.0, steamlit, pandas and python-dotenv
   - Heuristics.py supplies the full Nielsen sets

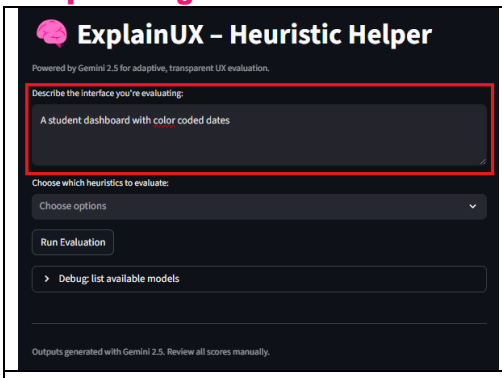
Challenges and Solutions

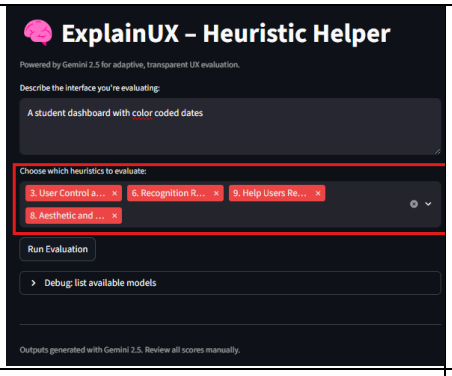
1. Unstructured or Inconsistent AI outputs
   Problem: Early testing showed that model sometimes returned long text strings instead of key-value pairs, this made it difficult to extract "score", "why" and "improvement" reliably. Without the structure streamlit had issues rendering results in a table form.
   Solution: Using [response_mime_type="application/json"] within the model configuration responses were constrained to machine-readable JSON.
2. SDK Import Conflicts
   Problem: During the setup imports like google import genai caused import error due to mismatched versions of Google Generative AI SDK. Preventing any content generation calls from running. Additionally, having it configured in app.py initially made debugging more difficult.
   Solution: The configuration was externalized into requirements.txt to improve readability. The API key was moved from hard-coded variables in app.py into a separate .env file, loaded securely using the python-dotenv package. This change simplified setup across development environments and enabled stable, repeatable initialization of the Gemini model client without exposing credentials in code.
3. Handling Missing or Null responses
   Problem: There were instances where the AI failed to return a full JSON object causing KeyError or TypeError execptions when building the results table
   Solution: Error handling was added in the [_chat_json()] helper to check for None or unexpected keys. If a response failed validation, the system logs the error in Streamlit and inserts a placeholder row ("N/A", "Response unavailable").

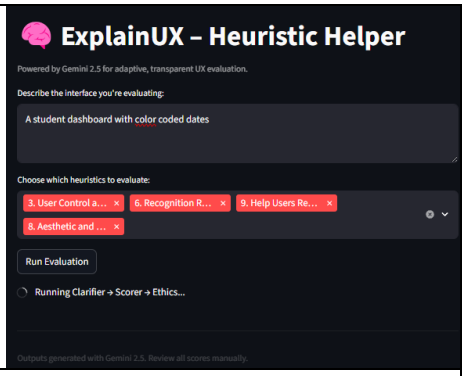
Learnings and Improvements

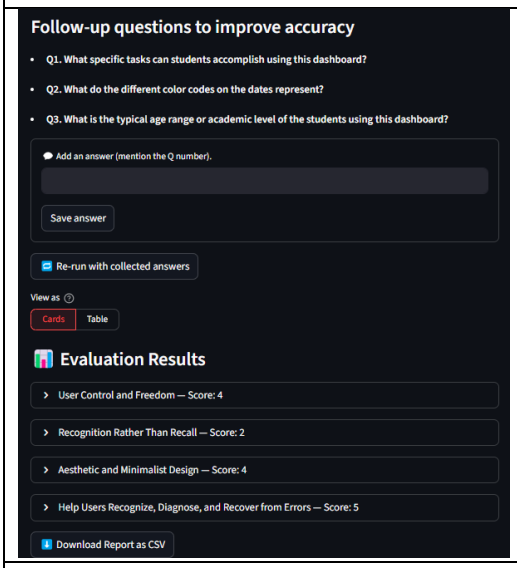
1. Importance of Modular Architecture for Maintainability
   Learning: Developing this application showed me how modular code design improves flexibility and troubleshooting significantly. Separating the application's functionality across distinct files made it easier to isolate bugs and update individual components without breaking others.
   Improvements: Future iterations could introduce a configuration layer or controller class that dynamically loads heuristics and agent roles from external JSON files rather than Python scripts. This can allow users to add or modify heuristics through a web interface without editing the core code.
2. Balancing Automation with Human in the Loop
   Learning: Integrating LLMs into UX evaluation taught me that automation can streamline heuristic analysis but must be carefully balanced with human judgment. This experience showed that ExplainUX should not replace human reasoning but rather guide reflective evaluation through structured feedback and ethical awareness.
   Improvements: ExplainUX could incorporate explainability features, such as showing confidence levels, rationale tracebacks, or prompt previews for each score. These transparency mechanisms would help users understand how and why a score was generated which encourages critical engagement.

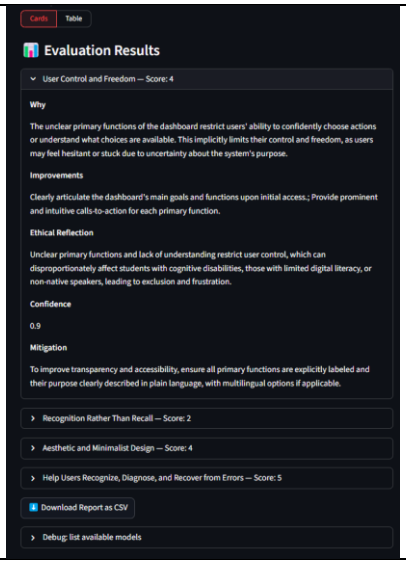
Sample Usage

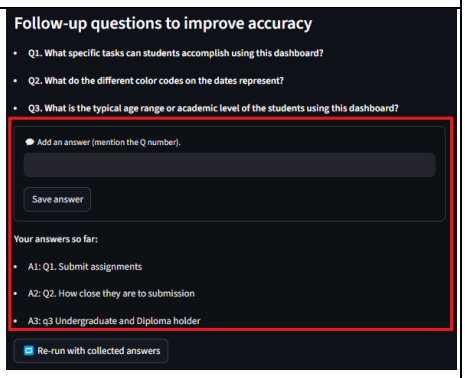


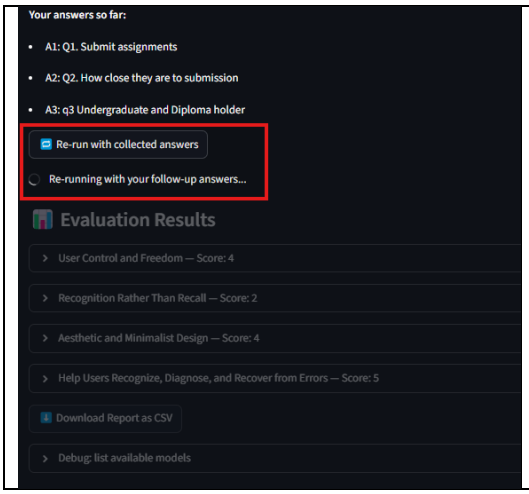












User Evaluation

Two methodologies were used to evaluate ExplainUX:

1. Think-Aloud Usability Testing: Participants performed three tasks (below) while verbalizing their thoughts and reactions while adding prompts about AI ethics post-task
    - Running evaluations with multiple heuristics
    - Re-running after answering clarifying questions
    - Downloading and reviewing results
2. System Usability Scale: After each task participants rated ten usability statements on a 1-5 scale to provide a quantitative usability benchmark

Participants were chosen based on their experience level in heuristic frameworks

Results Summary

Think-Aloud Usability Testing Results

Participants	Experience Level	Task Completion	Key Observations	Ethical Perceptions
1	New	All task completed	Was unclear about heuristics and re-run step. Improved after interaction.	Did not fully grasp how each stage transformed their output. Hesitated to trust if AI fairness comments reflected real bias. Trust was reactive.
2	Intermediate	All task completed	Confident Navigation Reflected on fairness and collaboration. Used Clarifier effectively	Recognized transparency as structural clarity. described the reflections as "balanced and educational," Trust was collaborative
3	Expert	All task completed	Had some issues with depth of explainability.	They asked for visibility of prompts or model justifications Noted that AI's ethical insights must be supported by clear, empirically traceable justification rather than abstract claims. Trust was calibrated and conditional

System Usability Scale Results

Participant	Experience Level	SUS Score (/100)	Notes

1	New	75.0	Clear layout but onboarding confusion lowered confidence.
2	Intermediate	85.0	Transparent, efficient, and aligned with UX workflow.
3	Expert	95.0	Seamless use, only requested deeper explainability.

Appendix A

I acknowledge the use of AI and generative AI tools in completing this assignment. Details of which tools were used and how they were used are provided in the table below, along with appropriate in-text and full references. I take responsibility for critically evaluating and integrating the AI-generated content, and ensuring it adheres to academic integrity standards

AI Model Used and Date	Language Translation	Gammer/ Style	Planning/ Drafting	Research / Background Information	Content Creation text	Content Creation Visual	Content Creation Code	Feedback	Others (Provide Details)
ChatGPT 5		x		x			x	x	