Heuristic Evaluation (Individual)

Due: At the start of your studio (Thurs/Fri 11/9-11/10)

Overview

You have been hired as a consultant to another group in the class. They are building a new user interface for their course project, but they would like some outside assistance in finding any problems with their prototype interface. Your TA will send you links to your assigned team's medium-fidelity presentation slides and medium-fidelity prototype.

Evaluation

You will perform a heuristic evaluation (individually) of your assignment team's user interface using only the materials they turned in for their last project assignment ("Medium-fi Prototype" slides, prototype README & working demo). Using their tasks, task flows, interface design, screen shots, and medium-fi prototype, you will apply Nielsen's heuristics to the user interface. You should be able to get all of this information from their last assignment. Read their slides first and then run their prototype. Your evaluation will use both the information in the presentation and the prototype.

Please use the heuristics and numbering scheme from our lecture slides on heuristic evaluation (also described in Nielsen's chapter). You will produce a report showing the problems in the interface.

Report

Your report (Google Doc) will **list each of the problems found** in the following format:

problem#. **heuristic violated**
description of problem, rationale for why you think it violates the heuristic & suggestion to fix

For example:

1. **H4 Consistency & Standards**
The interface used the string "Save" on the first screen for saving the user's information, but used the string "Store" on the second screen. Users may be confused by this different terminology for the same function.
Fix: Use "Save" on all screens.

2. **H3 User Control & Freedom**
The interface brings the user into a set of preference screens when they select "New User", but doesn't allow the user out of the dialog until they fill out all four screens. There is no way to cancel from any of the screens if a user came into the first screen by accident.
Fix: Add a "Cancel" function to each screen in the sequence.

Your report will also **summarize the number of violations found in each of the ten heuristic categories** (make a table – see below) and give the total number of violations in the entire interface. Finally, your report should close with some **overall recommendations** you have for improving the user interface given what you read in their presentation slides and what you experienced in testing their prototype (1 paragraph).

Example Table for Summary of Violations

Category	**# Violations**
H1: Visibility of System Status	
H2: Match b/w System & World	
H3: User Control & Freedom	
H4: Consistency & Standards	
H5: Error Prevention	
H6: Recognition not Recall	
H7: Flexibility & Efficiency of Use	
H8: Aesthetic & Minimalist Design	
H9: Help Users with Errors	
H10: Help & Documentation	
Total Violations	

Deliverables

You will email/share your individual write-up to/with your TA (Google Doc) by the due date (start of studio). **Make sure your email subject is formatted as follows: "CS147 Individual HE [Project Name You Evaluated] - [Your Name]".** Make sure to bring a copy of the source file on laptop/Google Doc as you will be using this again with a group in this week's studio. Please give your file a name that identifies you (e.g., JohnDoe-ProjectYouEvaluated-HE). **Your write-up should follow this outline with separate sections for the top-level items:**

1. Prototype (one sentence description of the UI you are evaluating)
2. Violations Found (i.e., the list)
3. Summary of Violations (table)
4. Recommendations (1 paragraph)

Here is an example report from 2015:
https://drive.google.com/a/stanford.edu/file/d/0B7BFy2Z_-KPxMkdjMzdwNkRuQzQ/view?usp=sharing

Grading Criteria

You will be graded on how complete your HE report is in terms of coverage of the issues present in the user interface design, clarity of your violation descriptions, and quality of your recommendations. You should concentrate on the interface the group has *designed*, not only on what has been *implemented*. Reports that continually focus on features that are missing, but will clearly be added will be marked down (e.g., "there should be help on this screen... and this screen..." – if it is a globally missing feature like "help", you can report it once). Please focus on evaluating what they have designed so far.

Prototype Description (10 points)

- Did you accurately and succinctly describe the prototype you are evaluating?

Violations (50 pts)

- Are you finding violations across all three tasks?
- Are you finding different kinds of violations, not just similar violations in many places? Make a note if something is frequent, but don't worry about citing every example.
- Do you have some less obvious violations (if they exist) in addition to the more obvious ones? Don't worry if it's hard to pick a category for a violation. It's more important that you spotted a difficult part of the interface. Many violations, even if they are small, will be helpful to the team.
- Are you focusing too much on missing features rather than giving helpful feedback on what you see?
- Is your feedback oriented around the tasks the team designed for?

Summary (15 pts)

- Have you summarized your results in a table? Make sure this is not just a laundry list of every violation, but a helpful and easy to read summary that gives the violations by category, as well as overall.

Recommendations (25 pts)

- Is there any feedback you have that doesn't fit neatly into the violations?
- What are your general impressions when using the prototype? Do you have any additional feedback that you think would be helpful to the other team?
- Is there a larger trend or way of thinking that is spread across many of the violations you found?

Ten Usability Heuristics by Jakob Nielsen (2nd version)

These are ten general principles for user interface design. They are called “heuristics” because they are more in the nature of rules of thumb than specific usability guidelines.

H1. Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

H2. Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

H3. User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

H4. Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

H5. Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

H6. Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable when appropriate.

H7. Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

H8. Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

H9. Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

H10. Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.