Heuristic Evaluation

Dave Schreifels

Team 1 – Grad 6

Michigan Technological University

The Undergraduate Design

Team Platypus is designing a system to document the rich history of the Keweenaw by offering locals the ability to upload pictures and stories related to various artifacts they may have in their possession – old photographs, a trophy grandpa might have had – that sort of thing. There's a great deal of focus on the history of this area; the primary focus is to create an app that is easy to use for folks who are often not so technologically savvy but still want to have fun and share their collective history with each other.

Description of UI Domain

The domain of this particular UI is, in a sense, a distributed survey to the end users; there are semi-rigid fields to fill out for any contribution made to the app, and these form the basis for the content on the app. It's worth noting that the app *isn't* just a survey, though, as users are not only completely voluntarily supplying information but are also never polled for it in the first place. Further, use of the app is not limited to the survey side of it, and even the survey side of it calls into question the idea that it can be constrained to this domain. Regardless of these points, however, it may be useful to look at this app through that lens.

The app is looking for information from users; even if not every user contributes, user contribution will necessarily form the basis of the content on the app. In this sense, it is a survey application. However, user interaction is planned as well, and this would suggest that the app also fits into the domain of social media. There is no forum in development for the app – no specific area where interaction will be relegated. Rather, the current idea is to have interaction on a piece-by-piece basis on the specific content submissions; this might foreseeably be better accomplished in some sort of localized hub once more content has been collected, but I digress.

The app's end goal is to both survey for historical artifacts and provide an interactive environment to discuss the submitted content. Therefore, it may be best to think of the app as having one foot in each of two domains – distributed survey, and social media.

Heuristic Principles & Problem Uncovered Thereby

Jakob Nielsen's 10 heuristics for user-interface design would seem to fit this application quite well. The heuristics are broadly suited for user interfaces, which make them ideal for an app that spans more than one domain, as one wouldn't want to shoehorn the app into heuristics of the specifics of either domain. This broader approach will allow the app to "explore the space," in a manner of speaking.

Heuristic Principle & Descriptions	Usability Comments & Concerns Important points are bolded for your convenience
Visibility of System Status The user should, at all times, understand what the application is doing through appropriate and timely communication from the app.	The only improvement I'd suggest in the system's current iteration is that, **when the map is attempting to load, there should be some indication that it's doing so**. Users might think the app is broken. The system is otherwise very good at conveying the limited status information it needs to.
Match Between System and the Real World This (ironically unintuitively named) heuristic refers to the interface using layman's terms instead of technical jargon. Any text should be short and intuitive.	The system has very little text at the moment, however, what's currently there does a pretty good job. The text on the sidebar is short and intuitive; descriptions of each field in the submission pop-up are concise and informative. A textbook example of good design.
User Control & Freedom The user shouldn't feel like they're powerless to influence the app. The primary focus is on the ability to leave and return to areas of the app that were unintentionally accessed, in particular, by the common functions of "undo" and "redo."	There are two small improvements to be made with regard to user control. The first is **a scroll bar for the zoom function**, which will allow users to do larger zooms with ease. The **buttons should be retained**. The second is that **the graphic for the docking button is unintuitive;** however, there may be no good way to improve this. The concise mouseover text is a good workaround; a button simply saying "dock" might be better, perhaps within the shape currently serving as the button.
Consistency & Standards Simply, the user should not have to think about whether or not two separate functions will do the same thing. Apple's computer keyboards, which have very little differentiation between the two delete keys, is an example of a consistency failure.	There is good consistency in the app. No two functions are alike. Each time the X symbol is used, it is used to close a dialogue or window. Other buttons do not implicitly suggest that they might be for some other purpose. **On the previously mentioned docking button:** it's not a matter of the button indicating the wrong thing, but rather of wondering what it does before you click it or move your mouse over it.

Heuristic Principle & Descriptions	Usability Comments & Concerns Important points are bolded for your convenience
Error Prevention This is fairly self-explanatory. In addition to providing usable error messages, the system should be designed to *prevent user error.* This is a common flaw spanning several industries.	The only problem found here is that **the submission form must allow for incomplete dates**, *at least* when the estimation box is checked. When making my test point in Torch Lake, I attempted to give an estimated date of "2017," but I wasn't allowed to submit without a full date. This will likely frustrate users who either don't know the full date or don't want to supply it. A line or two of code bypassing the month and day requirement when the estimate box is checked would be ideal, requiring only a year for date estimates. Depending on the client's desires, it could also be acceptable to replace the estimate button with an "I don't know the date" box, but this would probably not be a good idea for this app.
Recognition rather than Recall The system should not make users have to remember very much; rather, it should make options and actions clearly visible and available. In short, the user shouldn't have to remember much from one dialogue to the next, and instructions should be easily accessible when appropriate.	The only information to be retained at present is the form data while you're filling it out, which is easily accessed by scrolling up. However, this brings up an interesting idea. A desirable feature in this app would probably be a **favourites section**, where users can **save and organize points** they want to come back to later.
Flexibility & Efficiency of Use The interface should have options which don't impede or encumber the novice user but which an advanced user can use to speed up their use of the app. Frequent actions should be accelerable.	**On the previously mentioned docking button:** This is where the docking button gets some praise. The novice user wouldn't necessarily worry about this button, and it's relatively unobtrusive. However, once the user gets a feel for the app, it can be a nice way to get the dialogue out of the way or to expand it for ease of entry. Perhaps the same box but with an arrow pointing to the upper right of the screen would be a good redesign for this button.

Heuristic Principle & Descriptions	Usability Comments & Concerns Important points are bolded for your convenience
Aesthetic & Minimalist Design Dialogues should contain only necessary information. Unnecessary or irrelevant information competes with necessary information for attention.	The only point I have to make here is on the submission form: **the name field is redundant**. This is not a critical error, but it is a minor annoyance to the user. Instead of having multiple clarifying points, simply say “What’s your name? (Optional)” and remove the text below the text entry box. The text within the box should say something like “providing your name can help other users find your submissions.”
Help Users Recognize, Diagnose, and Recover from Errors A subset of the Matching & Error Prevention heuristics, the idea here is that users should be able to understand error messages. Ideally these won’t need to be shown very often, but when they are they need to be in plain English.	Error messages are clear and concise, and as an added bonus bring you straight to where you need to go to fix the error. Excellent work, however… **I shouldn’t be able to submit a blank point**. I did this on accident, and there doesn’t appear to be a way to delete it on this end; **there probably should be a way to edit and delete your own points**. I realize this might be difficult in consideration of the desire to avoid making user accounts.
Help & Documentation As with the previous heuristic, it’s ideal if this isn’t necessary; however, any documentation or help information that may be necessary to include should be easy to search and should stay focused on the task at hand. Tasks listed should be concrete and small in scale.	This app is simple enough that I don’t think a lot of documentation is necessary. However, **an About page might be a good idea.**

Critical Concerns & Illustration

Obviously not all of the above concerns are especially problematic; listed here are the primary concerns that developers should focus on finding solutions to.

1. **Zoom Functionality**

   The zoom buttons are excellent and useful, but when the app zooms all the way in when a historical point is clicked (a good feature, I note), it's difficult to zoom back out afterward. The user must either repeatedly scroll the scroll wheel or repeatedly click the button. A small, imprecise scroll bar for the zoom feature would circumvent a lot of frustration.

   Example: John, and elderly gentleman, suffers from tendinitis. It's gotten better over the years, but he still finds the repeated clicking to zoom back out hurts after a short while. He'd contribute more, but he just can't manage that much repeated motion in one sitting.

2. **Allow Submission of Incomplete Dates**

   This may seem a minor point, but I suspect it is actually quite critical. If the user isn't allowed to input partial dates, they may become frustrated; many of these historical artifacts will not have specific dates attached, but an estimate of the year could be reasonably made in most cases. I suggest the following approach:

   - If the user doesn't check the estimate box and the date is incomplete, throw an error asking the user if the date is just an estimate and indicating the estimate checkbox. The wording is crucial, but the team has so far been quite good about user-friendly wording.
   - If the user does check the box, allow them to leave the day and month blank. If they leave the year blank, prompt them to give their best guess at the year or decade.

   Example: Suppose your elderly grandmother is trying to submit a picture of her late husband when he was young and spry. She doesn't know the exact day, but she knows it was June of 1928; that's what the label on the photo says. She enters the information she has, checks the estimate box, and… is denied, because she hasn't input a specific day. She becomes frustrated, and closes the app without submitting. She later badmouths the app at her dinner party for all its complexity.

   Probably the biggest thing to keep in mind with this app is that people who can barely turn on a PC without frustration are commonplace in our target audience. These are the people we need to cater to. I think the team is doing an overall excellent job of this so far.