**Maintenance Request System Design Critique**

**The Simplifiers**

**Daniel Beamer**

**C10755282**

**April 9, 2020**

This document is a summary of my thoughts on another team’s wireframes, and their relation to Nielsen’s heuristics. Contained in this document are nine Usability Aspect Reports (UARs) which detail these thoughts on a (mostly) page-by-page basis. The following is an outline of the contents of this critique document:

1. **Executive Summary**
2. **Introduction**
3. **Usability Aspect Reports (UARs)**
4. **Conclusion**
5. **Reflection**
6. **Executive Summary**

This summary is a brief condensation of the report that follows detailing my thoughts on Team 9’s wireframe user interface. The report contains nine Usability Aspect Reports (UARs), with each UAR denoting a different violation of (or adherence to) one of Nielsen’s heuristics. The idea behind these UARs was to highlight the main issues with the wireframe – if there were any – and to allow the author team to make adjustments to their design as necessary.

In general, issues that were found with the system were minimal, and those that were sound were often small in scale (save for a couple of them). These issues ranged from color coordination to overall clutter, but none were major enough to warrant massive redesigns. For specific issues, refer to each UAR’s title to determine where the violation occurs. This report also includes a brief conclusion to summarize the problems found in a more succinct format for ease of reading. Additionally, the report includes a brief reflection detailing the specifics of the creation of this report.

1. **Introduction**

The system I was presented with appeared to be a maintenance company / company communication channel. Its main purpose is to serve as an avenue of communication between a resident of an apartment etc. and the company in charge of performing maintenance on the units of said apartment. It features a login screen, a set of screens to create an account (for either a resident or a company contractor), a notification system which connects to a direct messaging (DM) system, a maintenance request form for residents, and a work orders listing for contractors. There existed no internal jargon associated with the system, as it was a straightforward concept that many interact with on a daily basis in the form of texting.

1. **Usability Aspect Reports**

|  |  |
| --- | --- |
| No. Daniel Beamer - 1\_\_\_-D\_\_\_ | Problem/Good Aspect Good Aspect |
| **Name:**  Sign In | |
| **Evidence:**  **Interface aspect:** Entire Sign-In page  **Heuristic:** Aesthetics and minimalist design | |
| **Explanation:**  The heuristic is adhered to because nothing is present in the UI that does not need to be. The UI clearly indicates to the user what is expected of them. Users are typically familiar with sign in screens (unless they are completely new to technology, which is our first assumption – that the user is at least technologically literate), and so a simple login screen is easiest for a user to interact with. | |
| **Severity or Benefit:**  **Rating:**  NA – good aspect  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This will likely be something that a user interacts with every time they use the app, as they must sign in to continue (unless some sort of auto sign in is enabled), so it is important for it to be clean and painless.  *Impact:* Medium impact. It is important for a user to be able to sign in, as otherwise they could not access the remainder of the application!  *Persistence:* As noted previously, signing in might be something the user does every time they open the app, or it could be a one-time thing if the app keeps the user signed in. This determines the persistance. Thankfully, as this is a good aspect, this is not a worry.  *How I weighted the factors:* N/A | |
| **Possible solution and/or trade-offs:**  Such a design does not have many trade-offs. A simple UI is almost always a good thing. The only thing of note might be that some users find the design *too* simple, or something to that nature, and not aesthetically pleasing enough. Almost a non-factor. | |
| **Relationships:**  N/A | |

# 

|  |  |
| --- | --- |
| No. Daniel Beamer - 2\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Create Account | |
| **Evidence:**  **Interface aspect:** Textbox for selecting a username on the Create Account screen  **Heuristic:** Visibility of system status / Error prevention | |
| **Explanation:**  These heuristics are violated under the assumption that users must have unique usernames (which one would conclude is a valid assumption to make). This system does not provide for the situation that a user attempts to use a username that is already claimed. | |
| **Severity or Benefit:**  **Rating:**  3 – major problem (a problem that should definitely be solved with high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This issue has the potential to occur with any given registration after the first. While it is admittedly a low *probability* of occurring, if it does occur this could be a highly difficult issue to realize as a user.  *Impact:* This likely has a high impact, but could be easily overcome. If the user does not realize a username is taken, it could be a long time before they realize the problem / decide to try a different one. But a simple text display saying it is taken could easily fix the issue.  *Persistence:* The persistence of this problem is likely quite low. Since this is a registration, if the user caves and uses a different name, they will only encounter this issue once (assuming they realize the issue). If they do not realize the issue however, it could take some time before they attempt this.  *How I weighted the factors:* While a low frequency issue (as it potentially only occurs once per user if their second name choice is not taken), the inability to know whether or not a username is taken can be a massive pain, as you are now left guessing as to whether or not you can proceed with the registration process. | |
| **Possible solution and/or trade-offs:**  A display at near the textbox displaying whether the username is taken would solve this. While it would not include any heuristic tradeoffs, it might include some server-side tradeoffs, requiring the system to check if the name is taken in order to display this text. | |
| **Relationships:**  N/A | |

# 

|  |  |
| --- | --- |
| No. Daniel Beamer - 3\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Create Account | |
| **Evidence:**  **Interface aspect:** Textboxes for selecting a password on the Create Account screen  **Heuristic:** Visibility of system status / Error prevention | |
| **Explanation:**  These heuristics are violated under the assumption that users must have matching “Password” and “Confirm Password” fields (which one would conclude is a valid assumption to make). This does not provide for the situation that a user attempts to use two different strings in these fields. | |
| **Severity or Benefit:**  **Rating:**  3 – major problem (a problem that should definitely be solved with high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This issue has the potential to occur with any given registration. It can occur quite easily via a finger slipping to the wrong key, and if it does occur this could be a highly difficult issue to realize as a user (especially if the fields are censored using \*’s as many password fields do).  *Impact:* This likely has a high impact, but could be easily overcome. If the user does not realize the passwords are different, it could be a long time before they realize the problem / decide to try a different one. But a simple text display saying it is taken could easily fix the issue.  *Persistence:* The persistence of this problem is likely very low. If the user is competent in their typing, they likely may not mistype their password and will not encounter this problem at all. If they do mistype one of the fields – while it may take some time for them to realize the problem – it should only occur once, as they are unlikely to make the mistake again.  *How I weighted the factors:* While a low frequency issue, the inability to know whether or not a your password fields match can be a massive pain, as you are now left guessing as to whether or not you can proceed with the registration process (or are perhaps left frustrated as to why you cannot proceed). | |
| **Possible solution and/or trade-offs:**  A display at near the textbox displaying whether the password fields match would solve this. This likely would not come with any tradeoffs, as it creates a simple text string that alerts the user of their error. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 4\_\_\_-D\_\_\_ | Problem/Good Aspect Good Aspect |
| **Name:**  Home (Resident) In-Prog RQs | |
| **Evidence:**  **Interface aspect:** In-Progress Status Indicator  **Heuristic:** Visibility of System Staus | |
| **Explanation:**  The heuristic is adhered to the page shows the current status of any in-progress requests the user has. This allows the user to know that their request has been recognized, and is currently being looked at. Or, perhaps, if it is already finished if the system supports that. | |
| **Severity or Benefit:**  **Rating:**  NA – good aspect  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This is likely a high-frequency item. The user is likely using the app with the intent of making a new request or checking up on a current one, so having this visible will be useful.  *Impact:* High impact. It is important for a user to be able to know the status of their current requests, as it would allow them to keep up with the situation without having to driectly get in contact with someone else involved in the process.  *Persistence:* This system likely will take action almost every time the user opens the app, so the fact that it exists is extremely helpful to the user.  *How I weighted the factors:* N/A | |
| **Possible solution and/or trade-offs:**  Such a design does not have many trade-offs. A useful portion of the user interface is almost always a good thing. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 5\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Messages Home (Resident) / Messages Home (Maintenance) | |
| **Evidence:**  **Interface aspect:** Color indicator on message for status  **Heuristic:** Consistency and Standards | |
| **Explanation:**  This heuristic is violated because red typically indicates that there is a problem with the system. I’ll emphasize that yellow and green are fine for their intended purposes here, but red does not fit its intended purpose. | |
| **Severity or Benefit:**  **Rating:**  2 – minor problem (would be nice to solve, but not a high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This occurs any time the user checks their messages and has a completed request. Which would happen forever in the future post-request.  *Impact:* This likely has a medium impact, as it would lead to some confusion the first few times before the user realizes that red != bad in this scenario.  *Persistence:* The persistence of this problem could be quite high. The user will see this every time they check their messages (whether it be for the request in question or for another, newer request), so if they do not reconcile the different meaning of red in this situation the issue could occur many times.  *How I weighted the factors:* While this issue can cause confusion for the user at first, there are other factors that allow the user to recognize the intention of the symbol. That said, it is an issue that the user will encounter many times, leading me to weigh the frequency and impact much more highly in the issue. | |
| **Possible solution and/or trade-offs:**  A possible solution could be to remove color from the completed messages entirely, and perhaps replace it with a check mark or similar symbol. This would prevent confusion over color, and allow a recognizable symbol for use instead of colors (this also has the advantage of not affecting colorblind individuals [like myself]). While I find this unlikely, a possible tradeoff to this could be the user misinterpreting the check mark to mean something different – perhaps that the company determined there were no issues. But, again, I find this unlikely. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 6\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Messages Home (Resident) / Messages Home (Maintenance) | |
| **Evidence:**  **Interface aspect:** Cog for selecting/deleting messages  **Heuristic:** Consistency and Standards | |
| **Explanation:**  This heuristic is violated because cogs would typically be used in a “Settings” situation. Its use here does not reflect its standard use. | |
| **Severity or Benefit:**  **Rating:**  2 – minor problem (would be nice to solve, but not a high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This occurs any time the user decides to delete a message. It could also occur any time the user is attempting to access account settings or something similar.  *Impact:* This likely has a medium impact, as it would lead to some confusion the first few times before the user realizes that the cog is not a “Settings” button, but rather a method for the user to select and delete messages.  *Persistence:* The persistence of this problem could be quite high. The user will see this every time they check their messages, so if they do not reconcile the different meaning of the cog in this situation the issue could occur many times.  *How I weighted the factors:* This could be an issue that the user does not encounter for quite some time, as they will not encounter it until they either attempt to delete messages or perhaps attempt to find settings (which I realize do not exist in this wireframe, but is a valid thing for a user to search for). This confusion may not be easily solved, so I weighted the impact more highly in this case. | |
| **Possible solution and/or trade-offs:**  A possible solution could be to remove the cog entirely, and instead have a simple button that says “Select”. This is similar in style to how many phones allow you to select multiple messages or photos, and would remove any confusion the user may have about the funciton of the button. From that point, the user can then click on another button to decide what to do with the selected messages. One tradeoff to this might be a loss of a minimalist design due to the new menus, but I feel that the benefits would outweigh that cost. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 7\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Maintenance Form – When? | |
| **Evidence:**  **Interface aspect:** Title of form page (“When?”)  **Heuristic:** Help and Documentation (?) | |
| **Explanation:**  I was admittedly having a hard time determining which heuristic to file this under. It is more of a general confusion of usage comapred to a fundamental problem with the system. This heuristic is violated because, to the uninformed user, the form could be interpreted as asking “When did the issue occur?” rather than “When will you be available?”. I understand this is a nit-picky thing, but it was initially confusing. | |
| **Severity or Benefit:**  **Rating:**  1 – cosmetic problem (does not matter too much)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This occurs any time the user decides to schedule a maintenance request. It hopefully would only occur the first time a user attempts it, and afterwords they would realize the meaning.  *Impact:* This likely has a small impact, as it would lead to some confusion at first in its initial meaning, but could otherwise be understood after some thought.  *Persistence:* This problem’s persistance is likely low, as once the user realizes their misinterpretation they should not encounter it again.  *How I weighted the factors:* This is likely a very small issue overall. It is more of an initial confusion of usage rather than a fundamental problem with the system. The impact and persistance are what most factored into the rating, as it is a temporary, easily solvable issue. | |
| **Possible solution and/or trade-offs:**  A possible, simple solution would be to simply change “When?” to “When are you available?”. This small change would prevent this confusion from occurring in the first place. The tradeoff to this could be losing a minimalistic design, but three more words does not strike me as much of a cluttered screen. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 8\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Maintenance Form – Summary | |
| **Evidence:**  **Interface aspect:** Contact Method Selection  **Heuristic:** Flexibility and Efficiency of Use | |
| **Explanation:**  (This is under the assumption that the user is meant to be able to edit the information from this summary page. This was not made clear from the wireframe as to whether this was just meant to show the user, or also allow them to change the information. The written annotation suggests the first option, but the wireframe itself suggests the second.)  This feels like more of an oversight than a misdesign, but “Contact Method” does not allow you to change the method of contact (switching between email, text, or phone). | |
| **Severity or Benefit:**  **Rating:**  3 – major problem (a problem that should definitely be solved with high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This occurs any time the user decides to schedule a maintenance request if they decide upon a different method of communication. How often that may occur varies on the user, however.  *Impact:* This could lead to a moderate impact, as if the user cannot change their contact information from this summary, they must return to a previous page to rectify the information instead, which becomes a hassle.  *Persistence:* This problem’s persistance varies on the user, but on average I would say is low. A user is not likely to change their mind on a contact method when the Contact information screen is just prior to this one, but this issue will occur every time the user encounters this situation.  *How I weighted the factors:* I weighed impact more heavily in this evaluation. While it may not be something that occurs for every user, or even every time, the times that it does happen will likely inconvenience the user in their progress towards their goal – especially when everything else on the summary screen appears to be able to be edited. | |
| **Possible solution and/or trade-offs:**  The simple solution is to include the three options listed on the previous screen (“Maintenance Form – Contact”). The only tradeoff to this would be another menu item, leading to more clutter. | |
| **Relationships:**  N/A | |

|  |  |
| --- | --- |
| No. Daniel Beamer - 9\_\_\_-D\_\_\_ | Problem/Good Aspect Problem |
| **Name:**  Accepted Jobs | |
| **Evidence:**  **Interface aspect:** Entire Screen  **Heuristic:** Aesthetics and Minimalist Design | |
| **Explanation:**  This heuristic is violated as the page is almost entirely filled with text. While individual orders may be separated by a line, it is not enough to differentiate one from another. | |
| **Severity or Benefit:**  **Rating:**  3 – major problem (a problem that should definitely be solved with high priority)  **Justification (Frequency, Impact, Persistence):**  *Frequency:* This occurs any time the user checks their accepted jobs. Assuming they have more than a couple jobs this could become quite cluttered.  *Impact:* This could lead to a moderate impact, as it could prevent a user from differentiating between multiple jobs, perhaps missing one in the list entirely.  *Persistence:* The persistence of this problem is moderate, as it varies based on how many accepted job the user has. However, based on the wireframe, it seems as any more than four will yield undesirable results, and that is not too many to assume a contractor may have at any given point.  *How I weighted the factors:* I weighed all three evenly here. The overall clutter of the page causes many different problems, and I feel should be addressed. | |
| **Possible solution and/or trade-offs:**  While I understand the idea of overlaying the cards of jobs on the screen, it causes more clutter than it is worth. I would recommend leaving gaps between each job, which would ease in the readability of each job individually. While this would eat into more screen real estate, scrolling through the jobs would be much less of a hassle to find the one you want. | |
| **Relationships:**  N/A | |

1. **Conclusion**

As the UARs denoted, there were many small issues that can be easily addressed, some more severe than others. Some important issues were a matter of standards regarding the use of colors, prevention of errors on the registration page, standards regarding cogs, slightly ambiguous text, missing functionality in the maintenance form summary, and general cluttering of the screen in the accepted jobs screen.

All in all, these issues are easily fixed with just a few changes, and overall the system appears to be very solid. There was clearly a lot of thought put into the various forms that were being submitted, and the navigation of the app makes good sense. Generally speaking if I were to encounter this app while living in an apartment complex, I would be well able to understand and make good use of it, and I feel that that is a good indicator for its quality.

1. **Reflection**

In truth, I had some difficulty finding the full nine problems for this report. The system was well put together enough to the point that many aspects of the wireframes were airtight with little room for improvement (in my eyes). So while I was able to find the full nine problems, some are a bit nit-pickier than others might be.

I am not going to give an absolute and say that I found all problems in this system, as there may very well be some that I did not perceive. That being said, I reported all the problems that I did in fact perceive.

Overall, I spent about eight hours in total of real work (not counting time spent away from the laptop) creating the UARs from beginning to end. I made sure to be thorough in my work, as I did not want to miss anything.

Generally speaking, Nielsen’s heuristics feel like a good measure to evaluate usability problems with. While there are occasionally issues that I feel do not fit into any of the heuristic categories (See UAR #7 for an example), the majority of situations are covered by at least one of the heuristics.

UARs appear to be an effective, if overly lengthy, method of describing usability problems. They effectively highlight the issues that one can expect to find in regards to a system, but those that do not read them often may not know what the different categories entail.

In general, I would not change anything about this project. It was straightforward and to the point, and was truthfully quite easy to complete while still being comprehensive.