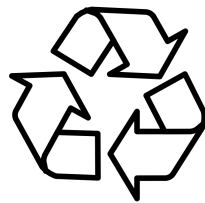


Track-My-Waste

Design Critique of Team 3



Jeffrey Black

CPSC 6140: Human-Computer Interaction

Document 4

9 April 2020

Document Description	3
Executive Summary	4
Introduction	5
UARS	6
Final Conclusion	29
Reflection	30

Document Description

This document outlines the process of critiquing some peers' design for an app. In it, I take a novice's perspective and give a description of the app at a first glance. I describe what I understand to be the purpose, the main features, and the flow of the app. Afterwards, I make a more in-depth critique of the app, providing Usability Aspect Reports (UARs) for the provided interface wireframes. That is, using Nielsen's usability heuristics, I describe the things that the app does well and does bad on. The UARs describe a specific aspect of the app, provide evidence and explanations for various good and bad aspects of the interface, provide severity/benefit ratings, offer a possible solution/tradeoffs for the current solution, and descriptions of how it relates to other UARs/features of the app. After the section of UARs, I provide a quick summary of the problems I found and recommendations I have. Lastly, I provide a reflection on the process--talking about the number of problems I found, the time spent on the process, thoughts on Nielsen's heuristics, thoughts on the use of UARs, and improvements on the process of critiquing designs.

Executive Summary

The outcome of the document is a succinct description of a food delivery app that supports group ordering, 12 Usability Aspect Reports (9 problems, 3 good aspects)--wherein I describe issues and good aspects of the interface and provide suggestions to make them better--, a synopsis of the design critique results--wherein i give an extremely high-level overview of the trends in the problems I found--, and a reflection on the design critique process overall--where I describe the process I went through and provide feedback on the use of Nielsen's 10 Usability Heuristics and the assignment as a whole.

Introduction

The interface I was assigned appears to be for a food delivery service. It seems like an UberEats/Postmates/DoorDash-type app. It allows the user to choose delivery or takeout, let's the user choose a restaurant to order from, choose items off the menu to order (and options to customize the meal), then view the status of the order.

Interestingly, the interface has support for group orders. So, one can invite friends to an order, consolidate all of the different orders into one order (even if they're from different restaurants), then split the bill in a variety of ways. So, think of one of the apps I listed above, but with support for group orders.

UARS


No. JB-01	Problem
Name: No sign-up/log-in screen	
<p>Evidence</p> <p>Interface Aspects: There should be a page which allows a user to sign in to or sign up for an account.</p> <p>Heuristic: Consistency and standards</p>	
<p>Explanation</p> <p>It's pretty standard for most apps to have some sort of log-in/sign-up feature. I'm assuming that the application doesn't use some other metric other than an account to keep up with users—maybe I'm making an incorrect assumption and say, phone number and name (provided by the phone) are used to identify a user. But, assuming that's not the case, there should probably be a page where the user can create an account or log into their account, especially given the fact that there's a host of profile aspects (such as meal history, favorites, friends, upcoming orders, etc).</p>	
<p>Severity</p> <p>Rating: 4</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> The problem is extremely common. Any person that uses the app will come across this issue. The problem gets a high frequency rating.</p> <p><i>Impact:</i> The problem is pretty low-impact in that it's fairly easy to fix. One just has to make a screen where the user can sign in or sign up for the app. However, it's high impact in the sense that having some sort of account system (and as a result, a log-in/sign-up system) is an integral feature of the app. The high impact aspects trump the low-impact aspects, so it gets a high impact rating.</p> <p><i>Persistence:</i> This problem is persistent. It applies to the entirety of the app. If someone doesn't have the ability to have an account, then I'm not entirely sure how the rest of the app functionality is achieved. It gets a high persistence rating.</p> <p><i>Weighting:</i> I gave the severity rating a 4 because of the combination of it being a high-frequency, high-impact, and high-persistence problem. Luckily, it's extremely easy to fix! The ease of fixing it brings it to even higher of a priority.</p>	

Possible Solution (and Tradeoffs)

A possible solution, and I think one of the only solutions, would be to provide a sign-up/log-in page when the user opens the app without having logged in before. Some downsides to this solution would be that the user wouldn't be able to explore the app before committing to an account, but the need for an account trumps the need to explore an app. An alternative to my solution would be to offer a sign-up/log-in page if the user hasn't logged in before, but also provide a button that allows the user to explore the app before committing to a full account.

Relationships

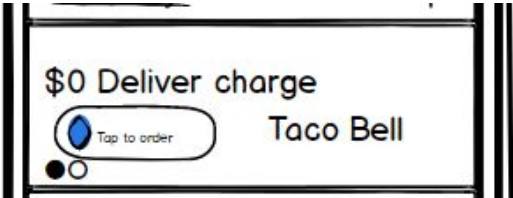
I think that this UAR is related to pretty much all of the other UARs considering the extent to which having an account is necessary for the functionality of most of the other features.

No. JB-02	Problem
Name: Rating system: Stars vs. Numbers	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Aesthetic and minimalist design</p>	
<p>Explanation</p> <p>The heuristic is violated because you include both a star (which is half-filled) and a rating number. I think it is confusing to have a half-filled star next to a rating number that doesn't map to it.</p>	
<p>Severity</p> <p>Rating: 2</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> The problem has high frequency, considering all of the restaurants in the app use the same rating system. Any time someone wants to view a rating for a restaurant, they encounter the redundant half-star.</p> <p><i>Impact:</i> Luckily, the problem is easy to solve and has a relatively low impact. One can deduce the meaning behind the half-star and the numbers beside it in its current state.</p> <p><i>Persistence:</i> Unfortunately, this is a high-persistence problem—it's present during the main function of the app, always. If one wants to make an order, they come across the problem. Even if it's their second, third, fourth, etc. order.</p> <p><i>Weighting:</i> Even though the problem is high-frequency and high-persistence, the impact is so low that it almost doesn't matter. Hence the severity rating 2. It's not an app-breaking problem, but it would be nice to have it touched up.</p>	
<p>Possible Solution (and Tradeoffs)</p> <p>The solution that comes to mind is using strictly stars and ditching the number</p>	

system. So, you could have five stars which fill up with color based on the rating. This would provide the same decimal-based rating system that's used, but it would be in a more visual, less redundant format. As it stands, the star is used to indicate that the number beside it is a rating--you circumvent the need to indicate that it's a rating system by simply using only stars (as well as the number of ratings it's based off of). The big problem with my solution is that you don't have the raw numbers to be able to make more in-depth comparisons between restaurant ratings, but I'm not sure how big an issue that is--I can't recall a time where I chose a restaurant based off of a difference of a tenth or twentieth in their rating. Granted, this may not be an issue if you provide different levels of being filled up for each star. So the stars for a restaurant with a 4.5-star rating would be slightly less filled up than those for a 4.7-star rating.

Relationships

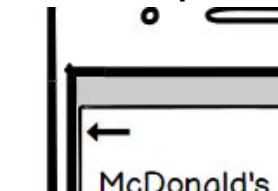
This aspect is not explicitly related to any of the other UARs, but it might be something that affects the user's experience and the pages they visit in the app.

No. JB-03	Problem
Name: The purpose of this section is unclear	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Visibility of system status</p>	
<p>Explanation</p> <p>This aspect breaks the heuristic because it's unclear what this section of the screen is for. It has a delivery charge, a restaurant name, an order button, and two dots on the left (which appear to be an indicator for the page of the section you're on), so one might assume it's a "Today's Deals" section, but I'm not 100% sure.</p>	
<p>Severity</p> <p>Rating: 3</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This problem seems to have medium-high frequency. It's only an issue if someone wants to get a deal and doesn't know that there's a deal section, but for everyone else it appears as an ambiguous section that they come across every time they open the home page of the app.</p> <p><i>Impact:</i> I would say there's a medium impact with this aspect. One can deduce the meaning of the section pretty easily if they've used meal-delivery apps before. However, to the novice user, the section could be quite confusing. Also, for users that are specifically looking for deals on the meals they're ordering, they might not understand that there's a deals section on the home page.</p> <p><i>Persistence:</i> The aspect has high persistence. Every time a user opens up the home page, they come across the issue.</p> <p><i>Weighting:</i> I went with a severity rating of 3 because of the high persistence and frequency as well as the medium impact. I think it should be a priority to fix the aspect so that new users of the app are thrown off by an unnamed section.</p>	
Possible Solution (and Tradeoffs)	

A possible solution (and I think the only real solution if you want to keep the section) is to add a title. A title with “Today’s Deals” or whatever the intended function of the section is would help to clarify the purpose behind the section. It’s hard to come up with any downsides to using a title, outside of the possibility that it might not be as visually appealing as just an icon indicating that the section is for deals.

Relationships

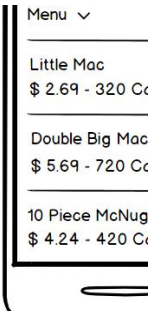
The UAR is not really related to any of the other UARs I wrote. I assume that when you press “Tap to Order,” it brings you to the order page. The order page is fine, so it doesn’t have a UAR.

No. JB-04	Good Aspect
Name: Use of back buttons	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: User control and freedom, error prevention</p>	
<p>Explanation</p> <p>The heuristics are adhered to because you provide the user a way to change their mind after moving to a screen. I think it's extremely important to include back buttons in an app where food is involved, especially because the user may change their mind multiple times or may desire to browse for a while before making an order.</p>	
<p>Benefit</p> <p>Rating: NA</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> Every type of user--aside from that which already knows what they want to order and doesn't make a single mistake when ordering--can use the back buttons. Thus, the back buttons have a high-frequency benefit.</p> <p><i>Impact:</i> For this app in particular, the use of back buttons has a high impact. When dealing with food, people are prone to changing their minds several times before coming to a final selection for the meal they want. Having a back button allows for users to quickly browse menus and find the food they want.</p> <p><i>Persistence:</i> The need for back buttons is a high-persistence thing. Just about every page in the app has a need for reverse navigation so that the user doesn't have to go through all of the menus again.</p> <p><i>Weighting:</i> The use of back buttons crucial for the app. Most users will frequently have a need to go back a page or two in the app (particularly when browsing restaurant menus). Not having back buttons would be a level-4 severity issue.</p>	
Possible Solution (and Tradeoffs)	

I don't think there are any negative tradeoffs to using a back button. There's the possibility that the back button is taking up space that could be used for other features, and one could just swipe right to navigate backwards in pages, but then there would be a visibility of system status/recognition rather than recall issue where the user might not know they can navigate backwards.

Relationships

Because the aspect has particular use in the menu, it's somewhat related to UAR JB-05 and JB-07. JB-05 is related because it pertains to browsing menu items, just as this UAR does. It's related to JB-07 because I suggest the use of breadcrumbs when choosing entrees, sides, and drinks off the menu--a back button is important for sequences with breadcrumbs.

No. JB-05	Problem
Name: No pictures of menu items	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Recognition rather than recall, match between system and real world</p>	
<p>Explanation</p> <p>Recognition rather than recall is violated because in the app's current state, the user has to have a preconception of what exactly the menu item is--there's no indicator of what the menu item is outside of the name. Match between system and real world is broken because typically, when you're at a restaurant or fast food place, you have a menu with visuals or the food in front of you to choose from--you know what the food looks like as you make a menu item selection.</p>	
<p>Severity</p> <p>Rating: 3</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> All people who use the app are prone to encounter this problem. The only person not prone to the problem is the user that orders fast food so often that they already know what the menu items look like. Thus, it has high frequency.</p> <p><i>Impact:</i> The problem can be overcome by googling the menu item's name, but that's such a massive inconvenience. When a user is making a meal selection, having the visual of the meal is extremely important. Thus, it has a medium-high impact.</p> <p><i>Persistence:</i> This problem has high persistence. Considering the main function of the app is to order food, any user who uses the app will encounter the problem as they're using it. This problem exists and affects users until it is resolved.</p> <p><i>Weighting:</i> Given the high frequency and persistence, as well as the relatively high</p>	

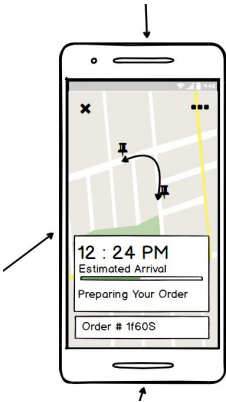
impact of the problem, I gave the aspect a severity rating of **3**. It's not app-breaking, but it's a major inconvenience for someone ordering food to not be able to see what's offered on the menu.

Possible Solution (and Tradeoffs)

The most straight-forward solution is to provide pictures of the meals on the menu. The downside to this solution is that the people maintaining the app have to provide pictures of each and every menu item, which I imagine is a time-consuming process. Alternatively, restaurants could add the pictures of the menu items themselves. That would require a restaurant dashboard, though.

Relationships

This aspect is related to JB-04 and JB-07. It's related to JB-04 in that both aspects are integral to the menu-browsing experience (the ability to quickly recognize what the menu items are as well as navigate between menu pages via back buttons). It's related to JB-07 in that JB-07 addresses the process of selecting menu items for different parts of a meal (entree, sides, drink). Having pictures for those menu items would certainly improve the experience of the user as it relates to JB-07.

No. JB-06	Problem
Name: Can't cancel order	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: User control and freedom,</p>	
<p>Explanation</p> <p>The heuristic is violated because the user is unable to cancel the order after placing it. I assume that some users might make an order by mistake and only catch that mistake once the order is placed. There should be some way to handle such an instance.</p>	
<p>Severity</p> <p>Rating: 4</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This is a problem that can be experienced by any type of user--so long as they make a mistake and catch it after they have already placed the order, the aspect is present. Therefore, it's a high frequency issue.</p> <p><i>Impact:</i> This is a high impact issue--the user should be able to cancel their order shortly after making it. Else they may make an order by mistake and not be able to correct it. Such a feature is pretty typical to see in food delivery apps.</p> <p><i>Persistence:</i> This problem is a medium persistence issue. It has the likelihood of only occurring once while the user is using the app to make a purchase. However, it is an issue present for every order the user makes.</p> <p><i>Weighting:</i> I gave the issue a severity rating of 4 mostly because of the impact and</p>	


frequency. It's pretty important to allow the user to correct a mistake in the instance that they make one with their order, else the user may end up wasting money on something they didn't really want.

Possible Solution (and Tradeoffs)

A possible solution would be adding a "cancel order" button that cancels the order request. The tradeoffs to this are obvious—you have to figure out which order statuses are acceptable to cancel on. I think that the benefits of adding a cancel button outweigh the trouble you have to go through to implement it though.

Relationships

This UAR is particularly related to JB-11, where I comment on how informative the order status screen is.

No. JB-07	Problem
Name: Breadcrumbs vs tabs	
<p>Evidence:</p> <p>Interface Aspects:</p>  <p>Heuristic: Match between system and the real world, visibility of system status, error prevention</p>	
<p>Explanation</p> <p>Match between system and the real world is violated because a lot of the time, you pick your meal items in a sequence, not all at the same time (as indicated by the use of tabs). Visibility of system status is broken because you aren't able to tell if you've already selected an entree, a side, or a drink at any given moment in the meal selection process. And error prevention is broken because the user may neglect the "Sides" and "Drinks" tabs as they aren't required parts of the sequence involved in ordering a meal.</p>	
<p>Severity</p> <p>Rating: 3</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This problem seems like a fairly common one--usually when ordering a meal you get an entree, a side, and a drink. Any user going with that standard will encounter the problem, therefore it gets a medium-high frequency.</p> <p><i>Impact:</i> The problem is easy to overcome as long as you understand that you have to select from all of the tabs in order for you to get the full meal. However, for those new to the app, it might not be obvious and they may neglect sides and drinks entirely. Therefore, I'm giving it a medium impact.</p> <p><i>Persistence:</i> The problem is recurring in that it happens any time someone makes an order. There is potential for users to be bothered by it repeatedly, therefore it gets high persistence</p> <p><i>Weighting:</i> I gave the severity rating a 3 because it's not app-breaking, but it may cause confusion amongst users. I'm sure most users would be able to figure out that they need to navigate between the tabs to choose a side and a drink for their meal, but</p>	

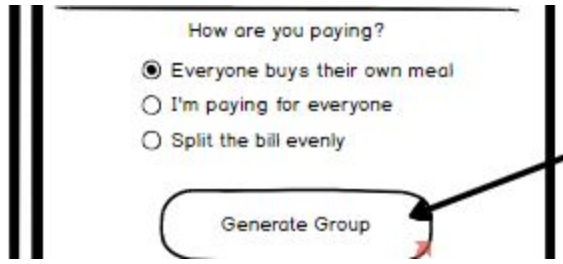
it's still something that may end up being problematic.

Possible Solution (and Tradeoffs)

I think the best solution would be to turn the tab navigation into a sequence of events with breadcrumbs as the user selects their entree, sides, and drink. This would help the aspect adhere to match between system and the real world because it turns the meal ordering process into the sequence you typically follow when ordering a meal in the real world. It would help the app adhere to visibility of system status because the breadcrumbs would inform the user of the stage they're at in the meal ordering process and what steps are too come (entree, sides, drinks). And it would help with error prevention, as it mitigates the potential issues of users forgetting to order sides and a drink.

Relationships

This is related to UAR JB-06 in that this UAR addresses an issue that may lead someone to cancelling their order (addressed by JB-06). It's also related to JB-04 and JB-05, in that both of those UARs address the user experience of browsing the menu and ordering a meal.

No. JB-08	Problem
Name: Preemptively splitting the bill	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Match between system and the real world</p>	
<p>Explanation</p> <p>The heuristic is broken because a user may not know how they want to split their bill amongst their group members until after everyone knows what's being bought. This is how it typically goes amongst friends buying meals together anyways--you decide how to divvy up the tab once everyone knows what they're getting.</p>	
<p>Severity</p> <p>Rating: 3</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This is a medium-frequency issue. Not every user is going to purchase a meal as a group, and some groups may already know how they want to divvy up the bill before they order the meal. However, oftentimes a group of people ordering a meal together may not know how they want to split the bill until after everyone knows what they're getting, and as a many groups may stumble into this issue.</p> <p><i>Impact:</i> This is a pretty high-impact issue. When ordering as a group, the group may not know how to divide the bill until after everyone knows what they're getting. So maybe the group isn't satisfied with the way that the bill is split by the time everyone has selected meals, and they want to change it. If they want to change it, they'd have to redo the order.</p> <p><i>Persistence:</i></p> <p>Given that the issue shows up any time the user wants to make a group order and isn't satisfied by the way they chose to split the bill, it's high-persistence. There is potential for the user to be repeatedly bothered by it.</p>	

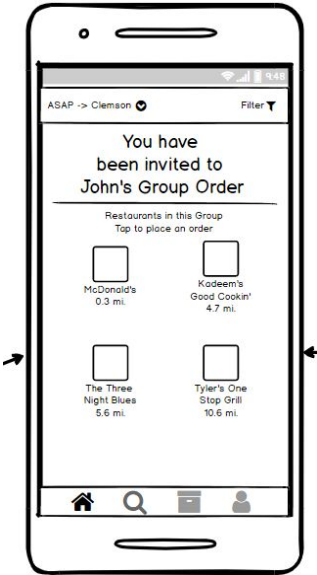
Weighting: I gave this issue a rating of **3** because it's a relatively low-risk issue. If the group of users isn't happy with how they split the bill, there are other means outside of the app to accommodate the changes to the way they purchased the meals. It's a frustrating issue nonetheless, so it's one that should be addressed at a semi-high priority.

Possible Solution (and Tradeoffs)

A possible solution would be to move the tab splitting process to the end-page where you actually confirm your order. That would allow users to better understand how the bill is being split and make adjustments for various circumstances that the paying group might find themselves in.

Relationships

This UAR is particularly related to UAR JB-10, where I provide a suggestion to add more ways to split the bill. It's related to JB-09 and JB-12 in that they all relate to the group meal ordering process.

No. JB-9	Good Aspect
Name: Great group ordering system	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Match between system and the real world</p>	
<p>Explanation</p> <p>From what it seems like, the app allows everyone in the group order to open up the app on their phone and make their own orders in parallel for the group order--that's a great idea. It matches the real world possibility that multiple people can make orders at different restaurants (or even the same restaurant) at the same time, and takes advantage of the fact that in the real world, just about everyone has a phone (and as a result, can make orders independently of one another).</p>	
<p>Benefit</p> <p>Rating: NA</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This is a medium-high-frequency benefit. Not all users are going to be making group orders frequently, but to the extent that people do, people are able to take advantage of the aspect.</p> <p><i>Impact:</i> This is a high-impact aspect. It saves a bunch of time in the ordering process if all of the people ordering can do so in parallel. It also prevents people from having</p>	

to share their phone. I think that it's a great addition overall.

Persistence: The benefit is also **highly** persistent. Every time a user wishes to make a group order, they can take advantage of the system in place.

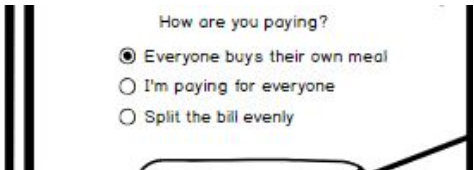
Weighting: The big benefit I see being added to the user with this aspect is convenience. You just choose the group members in your order and let them make the orders themselves.

Possible Solution (and Tradeoffs)

The major tradeoff to the heuristic comes when you consider the possibility that someone you're ordering for doesn't have an account with the app. There should be an aspect of the app where you can create an order for someone without an account (which would require using the person making the order's phone).

Relationships

This aspect is loosely related to JB-08, JB-10, and JB-12 in that they're all related to group ordering.

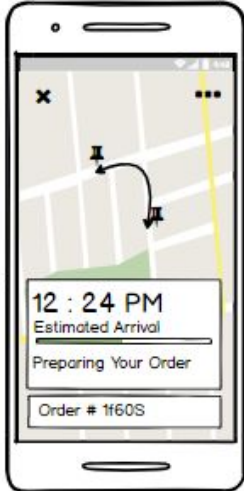
No. JB-10	Problem
Name: Consider more ways to split the tab	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Match between the system and the real world</p>	
<p>Explanation</p> <p>This heuristic is broken because the user is limited in the options they have to split a bill. They only have an even split, each user buys their own meal, and one person pays for all. This neglects the situation where multiple users split the bill in more unique ways. For instance, for a group of five, someone may cover the majority of the bill while the rest pay a smaller portion of it. In the real world, such is possible.</p>	
<p>Severity</p> <p>Rating: 2</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> This is a medium frequency problem. It's fairly common because the group payment system is used any time a group wants to order a meal, but the percentage of users using the group order feature is probably relatively low.</p> <p><i>Impact:</i> The problem is medium impact. It definitely prevents the user from splitting a bill in more-customized ways, but the user can reconcile the issue by paying friends to account for customized bill-splitting in the real world. It's not a difficult problem to overcome.</p> <p><i>Persistence:</i> The problem is highly persistent. Any time the user uses the group order system, they come across the problem. And overcoming the problem once doesn't overcome the problem for future orders.</p> <p><i>Weighting:</i> I gave the problem a severity rating of 2 because of the medium frequency and impact. It's not an issue every user is going to encounter, and it can be overcome through real-life money sharing methods. It's not an app-breaking issue, it's just a minor inconvenience for those using the group order system.</p>	

Possible Solution (and Tradeoffs)

A possible solution would be to provide a percentage-based bill splitting system. It could default at an even split, then the users could change the bill proportions as they see fit. The good part of this would be that it would support any type of bill splitting that the group of friends desires. The bad part is that it's more work to figure out ample percentages than it is to select options from a radio menu. Perhaps there could be presets the user can choose, then they can customize them with percentages?

Relationships

This UAR is particularly related to UAR JB-08, where I address preemptively splitting the group's bill. It's related to JB-09, and JB-12 in that they all relate to the group meal ordering process.

No. JB-11	Good Aspect
Name: Very informative order status screen	
<p>Evidence</p> <p>Interface Aspects:</p>  <p>Heuristic: Visibility of system status</p>	
<p>Explanation</p> <p>The heuristic is adhered to because it employs multiple measures to be transparent about the current status of the order. From a full-scale map, to the estimated arrival, to the current status of the order, to the order number. There's no question about what's going on with the order.</p>	
<p>Benefit</p> <p>Rating: NA</p> <p>Justification (Frequency, Impact, Persistence):</p> <p><i>Frequency:</i> Every user that makes an order is affected by the aspect. It's a screen that shows once an order is made, always. So, it's a high frequency aspect.</p> <p><i>Impact:</i> The impact of the aspect is also high. If the user weren't able to see the status of the order or the estimated arrival, they may not be able to plan for the meal arrival. That would be terrible!</p> <p><i>Persistence:</i> The aspect is also highly persistent. Every time a user makes an order, the aspect shows up. Even after the first order.</p>	


Weighting: There are multiple benefits to having the order status screen--you clear up concerns about when an order will be delivered, where the order is currently at, and what's going on with the order at any given moment. It's an extremely useful aspect.

Possible Solution (and Tradeoffs)

While the aspect in general is great, I'm not sure why you have an order number. I don't see that info being relevant until there's a need to contact support about an order. Perhaps keep that hidden until it's absolutely necessary!

Relationships

This UAR is related to JB-06, where I talk about ways you can change the delivery status screen to allow for more user freedom.

No. JB-12	Problem
Name: What about group pick-up orders?	
Evidence Interface Aspects:  Heuristic: Match between system and real world	
Explanation This heuristic is broken because the app only supports picking up an order when ordering as an individual. In the real world, a group of friends can get together and go to various restaurants to get food.	
Severity Rating: 3 Justification (Frequency, Impact, Persistence): <i>Frequency:</i> I imagine this is a low -frequency issue. The only types of users to be affected by this issue are ones that go out as a group of friends and pick up food as a group--a cohort that I imagine is rather small. <i>Impact:</i> The impact is extremely high . There's a total lack of support for the action. That is, it can not be overcome. <i>Persistence:</i> This problem occurs any time a user wants to make a take-out group order, and is thus highly persistent. <i>Weighting:</i> I gave it a severity rating of 3 because while it's certainly inconvenient (as noted by the high impact and persistence), it's got an extremely low frequency. I can't imagine a significant amount of users needing to make group orders, but there should be an option for the user to make a group takeout order regardless.	
Possible Solution (and Tradeoffs) The obvious solution is to implement the feature. I think this would be well-achieved by providing a takeout option for the group ordering page, which restricts the set of restaurants the group can order from to those which support takeout. There's not really any negative tradeoffs I can foresee--maybe the fact that implementing it is an annoying process, but I don't think it will be too difficult to implement.	
Relationships This UAR is related to UARs 8, 9, and 10. They all relate to group orders.	

Final Conclusion

The problems I found generally fall into three categories: not allowing the user to perform actions they might want to do, confusing the user, and functionality issues. For instance, providing a method to further customize the bill splitting method for group orders falls under “not allowing the user to perform actions they might want to do”; the lack of a section header for the sales section falls under “confusing the user”; and not including a sign-up screen falls under “functionality issues.”

The recommendations for redesign are luckily pretty simple. They’re mostly minor formatting changes, with a few order-of-events changes included. For example, one of the formatting changes I included was “add a section header to the sales section”; one of the order-of-events changes I included was “split the bill at the end of the group ordering process instead of at the beginning.”

The UARs go through all types of problems and both types of redesigns in-depth.

Reflection

I found nine problems and three good aspects. That's not to say I found all of the problems through. There were some that I found but didn't feel like writing up a full UAR for, and some that I found that I didn't think were all too important in the grand scheme. Originally understanding the interface and finding the problems I wanted to address took about half an hour to 45 minutes, and finalizing the UARs took about 15 minutes per UAR.

I think that Nielson's Heuristics are a very powerful tool to improve a design. Generally speaking, so long as you follow them you'll have a usable app. However, I didn't feel like Nielson's ten heuristics were enough to cover my concerns with the app. Take for instance the lack of a sign-in/sign-up page on the app. It doesn't seamlessly fit into one of the 10 heuristics Nielsen provides--perhaps a heuristic for "includes all of the basic functionality needed to make the app work" should have been included, even though that's not really a traditional usability concern as much as it is a functionality concern. I still found it important to point out, nonetheless. I think that there are some improvements to be made on Nielsen's original 10 heuristics.

I think that UARs are generally a good way to describe usability problems. Even though they're time-consuming to complete, they really help get to the crux of usability issues and organize design critiques in an easy-to-understand manner. I don't think there's anything I'd add to the UAR template, but I might remove the *weighting* section from the severity/benefit section. It seems kind of redundant--you can get more specialized critiques and understand the critic's emphasis on sources of severity by reading through the *frequency*, *impact*, and *persistence* sections.

If I had to do the assignment again, I personally would have made myself spend more time on the assignment so I could be more comprehensive in my critiques. In an effort to improve the assignment, I'd provide a section for full-fledged UARs and a section for UARs that aren't quite finalized--just so that the interface creators have more hints as to where to look for smaller problems/problems that the critic didn't feel like writing up an extensive UAR for. I might also get some of the undergrads to provide their own solution to some of the problems in the UARs--I think it was a fun exercise and does a lot to show the student how Nielsen's heuristics can be used to brainstorm interesting interface ideas.