HCI 440 Homework Assignment 3

Conceptual design, Intermediate design, Detailed design, Structured inspection, Evaluation test plan

Team 3

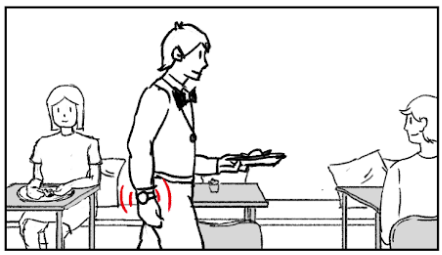
Ryne Minor
Andrew Ortolano
Shiwen Qin
Goran Somic
QiWen "Quinn" Zhang

24 February 2020
v1

Conceptual design

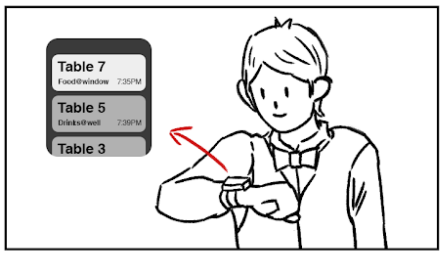
STORYBOARD

1



When Alex is walking through the restaurant, he receives an alert from his apple watch.

2

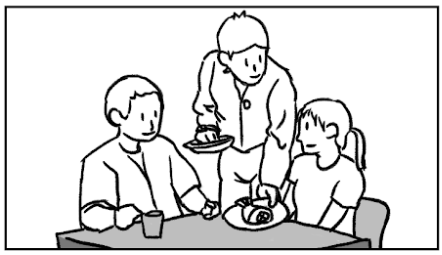


Alex looks down at his apple watch. The kitchen notifys him to pick up the food of table 7.

3

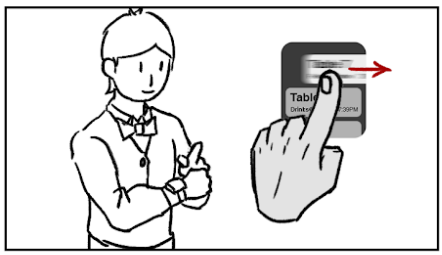
Alex goes to the kitchen to pick up food.

4



Alex delivers food to table 7 and puts them on the table.

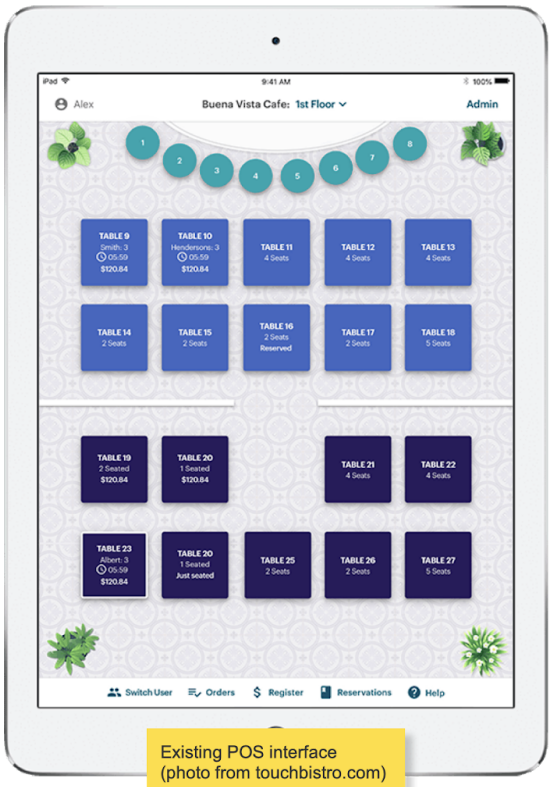
5



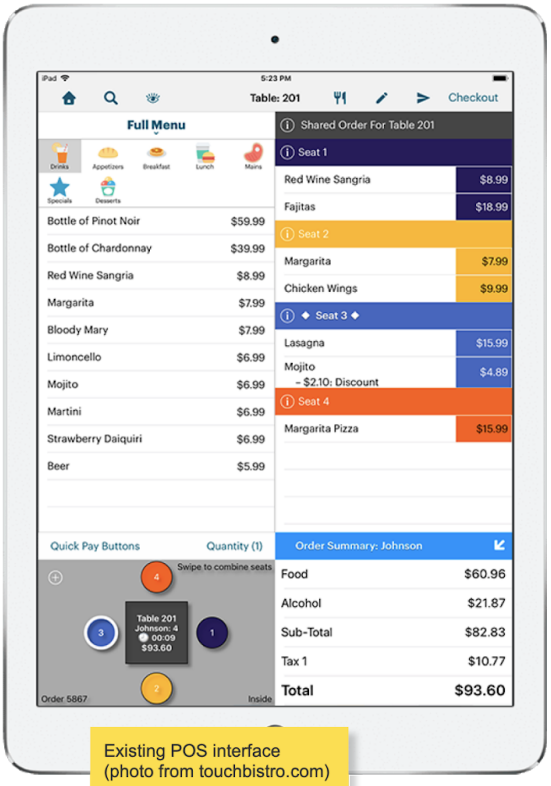
After finishing this task, Alex swipes right to dismiss it.

Intermediate design and context

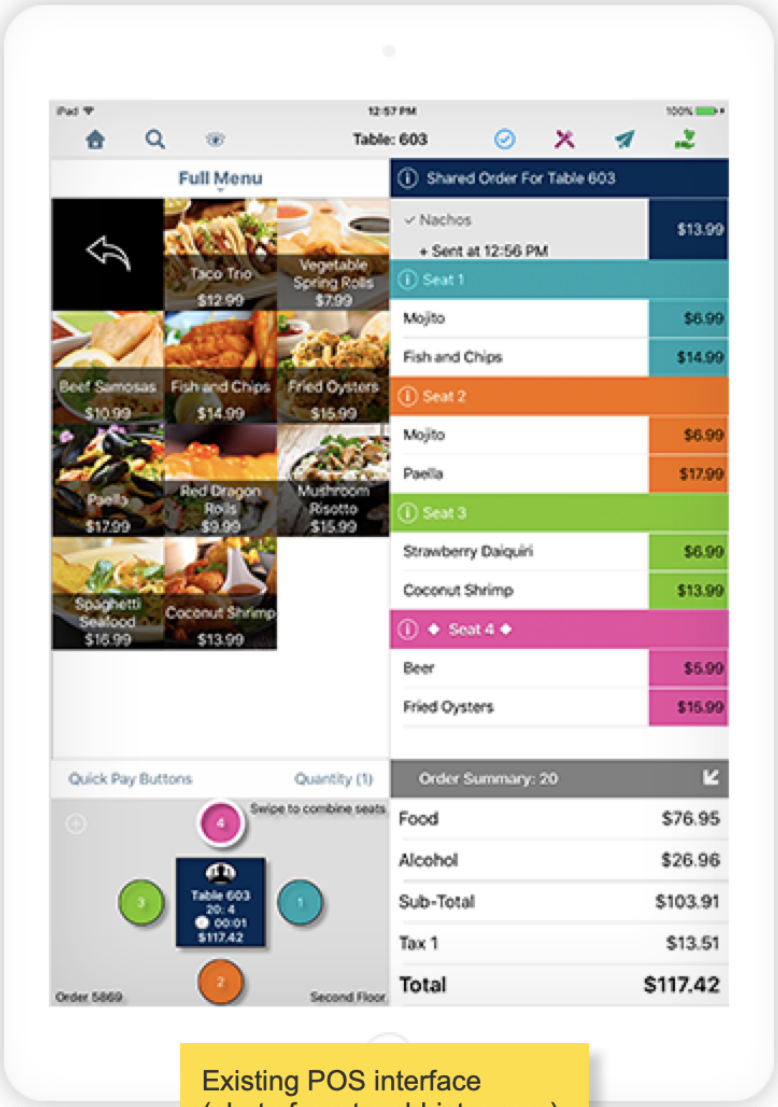
The smartwatch will work best if integrated into existing POS workflows. The following screens represent a typical POS system — Touch Bistro.



Existing POS interface
(photo from touchbistro.com)



Existing POS interface
(photo from touchbistro.com)



This screen demonstrates that the Touch Bistro POS can track the items a table has ordered, as well as a timestamp of when the order was sent in.

Existing POS interface (photo from touchbistro.com)

Table: 123

Full Menu

Shared order for table 123

Nachos	$12.23
Sent at 12:56 PM	

Seat 1

Mojito	$xx.xx
Fish and chips	$xx.xx

Seat 2

Mojito	$xx.xx
Paella	$xx.xx

Seat 3

Strawberry daiquiri	$xx.xx
Coconut shrimp	$xx.xx

Seat 4

Beer	$xx.xx
Fried oysters	$xx.xx

Table 123
20:4
00:01
$117.42

Order summary: 20

Food	$76.95
Alcohol	$26.96
Sub-total	$103.91
Tax 1	$13.51
Total	**$117.42**

Abstracted view

Heading 1
Heading 2
Heading 3
Headline / Label
Body Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.
Caption 1 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Caption 2 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Footnote

Typography, content-safe region, layout.

Taken from Apple's Human Interface Guidelines:

Layout:
https://developer.apple.com/design/human-interface-guidelines/watchos/visual-design/layout/

Typography:
https://developer.apple.com/design/human-interface-guidelines/watchos/visual-design/typography/

The Apple Watch visual guidelines provide font size and line-height recommendations for:

- Heading 1
- Heading 2
- Heading 3
- Body text
- Caption 1
- Caption 2
- Footnote

{table identifier}

{Notification} @ {location} {time}

Notification "bubble"

Table identifier, shown in Heading 1 style

Notification details shown in Headline/Label style

Table 77

Food @ window 12:15pm

TMS App 7:11pm

Table 77

Food @window 7:10pm

Table 74

Drinks @well 7:09pm

Table 78

Screen demonstrating notifications.

Swipe to dismiss.

Detailed design

Table-management smartwatch

Wireframes
v1
16 February 2020

Below, we demonstrate wireframes of notifications that a server may receive on their smartwatch, as well as the source.

Some notifications are the direct result of another user taking an action. For example, when the host assigns a table to a server in their POS, the server should receive a notification without the host needing to do anything else.

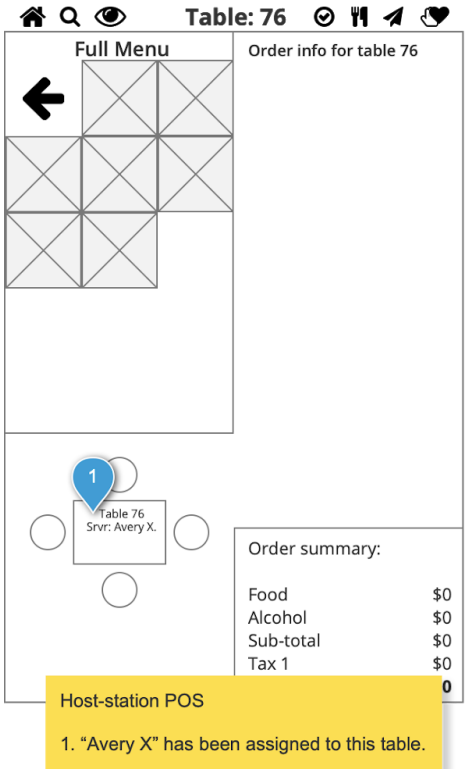
Other notifications are automated and would require configuration. For example, in a restaurant where food is typically delivered to tables by runners, the server would not receive a notification of when the food is ready, but would receive a notification to perform the 3-bite check *x* minutes after the runner picked up a table's food.

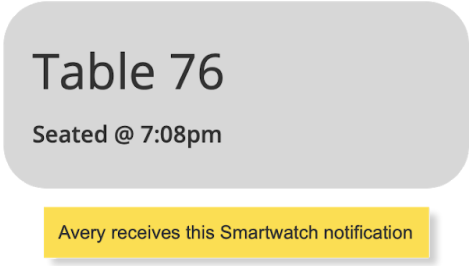
There are multiple sources for inputs:

- Host station
- Bar
- Kitchen
- System (automated inputs)

Notification 1 — Host assigns a table to a server

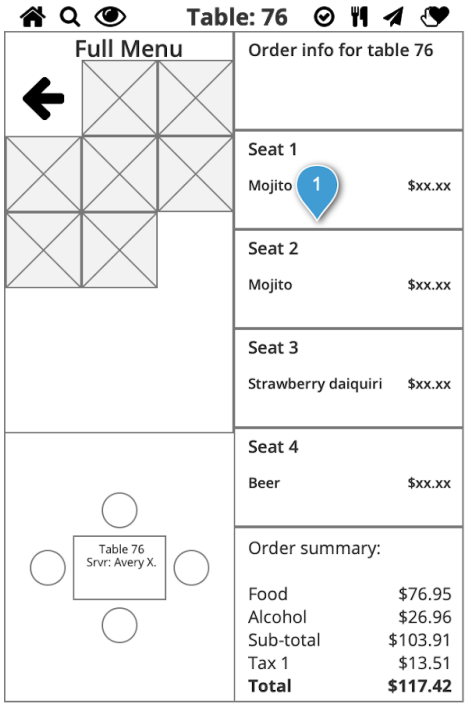
When the host assigns a table to the server in the POS, the server receives a notification. No additional action on the part of the host is required.





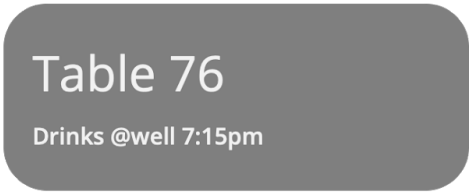
Notification 2 — Bar notifies server

After the server has taken a table's bar orders and entered them into the restaurant POS, the bar's POS interface will notify the bartenders of the order to be completed. When the bartender acknowledges the order as complete, the POS sends a notification to the server's smartwatch.



Bar-station POS

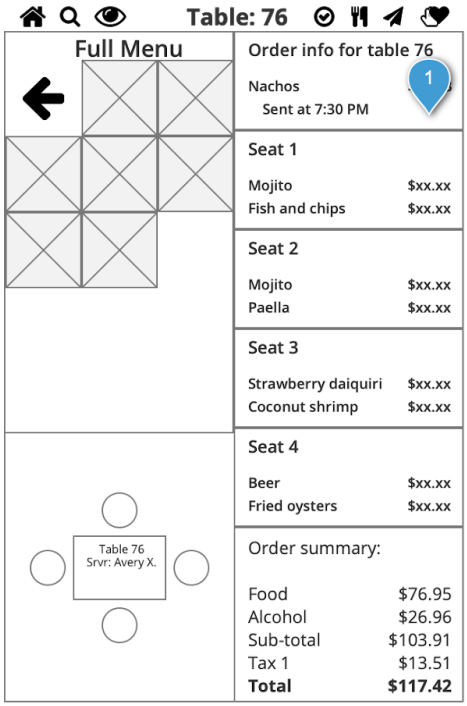
1. Bartender has fulfilled bar order.



Avery receives this Smartwatch notification

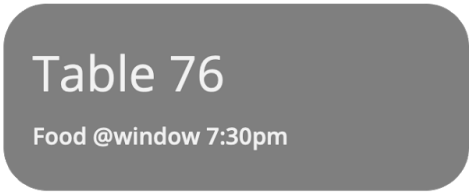
Notification 3a — Kitchen notifies server for food pickup

The server then takes a table's food orders and enters those orders into the POS system. According to our interviews, the orders are sent to stations within the kitchen, and sorted by course (



Kitchen-station POS

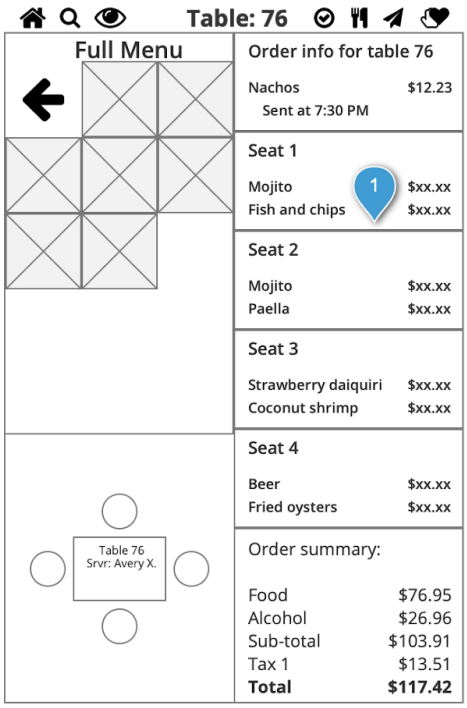
Kitchen has fulfilled appetizer-course order.



Avery receives this Smartwatch notification

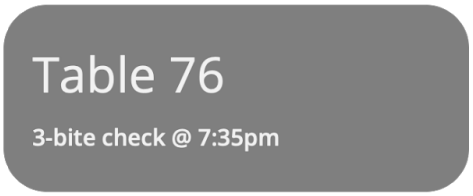
Notification 3b — Kitchen notifies server for 2- to 3-bite check

In a work environment where food is being expedited to tables by runners, the server will not receive a notification to pick up food at the window, but instead to check on the table a few minutes after a runner has delivered it.



Kitchen-station POS

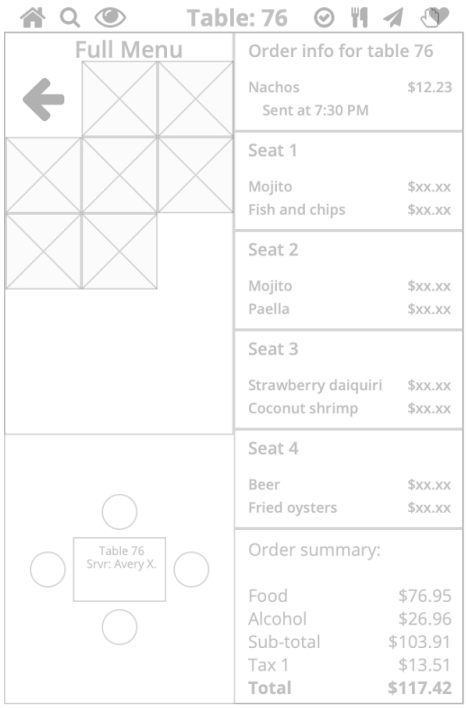
Kitchen has fulfilled entrée-course order.



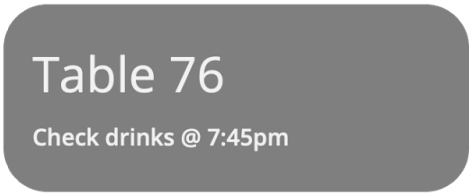
Avery receives this Smartwatch notification

Notification 4 — Drink recheck notification

At an interval after drinks have initially been delivered to a table, determined by the restaurant manager, the server will receive a notification to check on drinks.



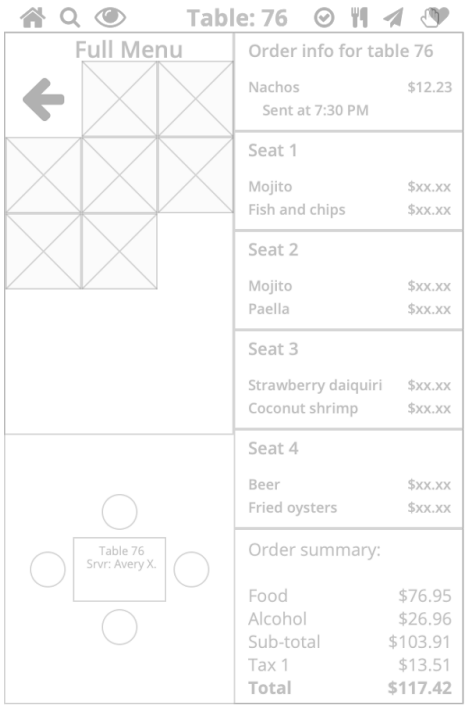
POS independent; automatic notification.



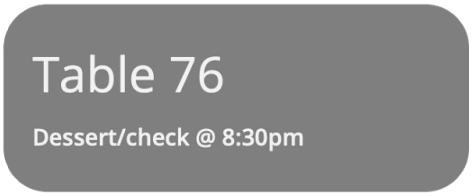
Avery receives this Smartwatch notification

Notification 5 — Time for the check or dessert?

At an interval after drinks have initially been delivered to a table, determined by the restaurant manager, the server will receive a notification to check on a table to see if they're ready for a dessert, or for the check.



POS independent; automatic notification.



Avery receives this Smartwatch notification

Notification styles

Restaurant service is stressful and high-pressure; the goal of this system integration is to reduce stress on the server.

Notifications will:

- Show up on the server's smartwatch
- Be color-coded to display urgency (see below)
- Be dismissable via swipe
- Be dismissable as a group via admin interface

Notifications will *not*:

- Vibrate the device
- Chime or make an audible alert sound
- Use "pressure" colors — primary red and yellow, for instance

Screen demonstrating notifications.

Swipe to dismiss.

Notes:

Notifications are stacked chronologically, from oldest at the top to newest at the bottom.

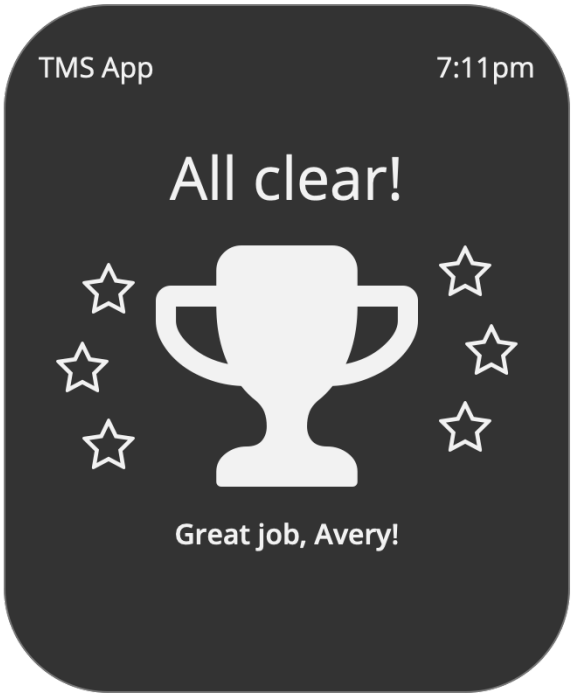
However, some actions are so time-sensitive that they will bubble to the top regardless of timestamp:

- Tables should be greeted within 30 seconds of seating

On this screen, a table has been seated, and the alert has not been dismissed. It will persist at the top of the screen until it is dismissed.

Positive reinforcement

Additionally, the system will provide positive reinforcement *only*. When the server has cleared their device, a graphic will be displayed alongside an affirming message.



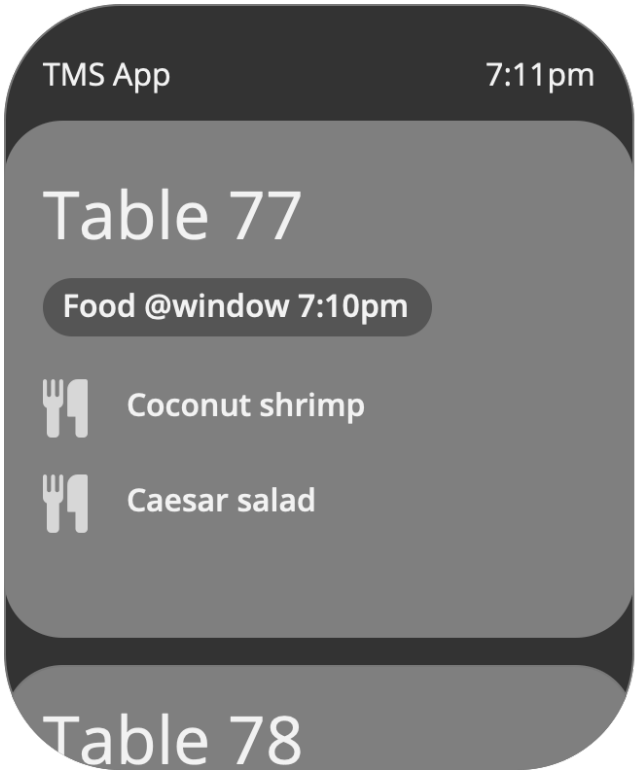
The success message will be positive. It should not order the server to go stay busy — “Time to do your sidework!” is not an appropriate message for this screen.

It could also, depending on POS system integration, show the server the dollar amount of tips they have earned on their shift so far.

User actions

The primary users will be able to perform a limited number of actions on the smartwatch:

- Swipe to dismiss alerts
- Tap to unfold alerts



Unfolded view of a “food in window” alert

Unfolded view of a “drinks at the well” alert

Structured inspection

Quick overview:

In order to maximize insight into usability of our product, we decided to follow two methodologies. The first inspection will be based off of Shneiderman's "eight golden rules of design" and the second will entail heuristics from Nielson's methodology. Although many of the heuristics from the two methodologies overlap, we feel that using a variation of both will give us better insight into our products usability.

Inspection 1

Methodology:

Heuristics based off Ben Shneiderman's Eight Golden Rules of Design.

Overview of Heuristics used:

The heuristics used in this evaluation are derived from Ben Shneiderman's eight golden rules of design. These eight design principles are used to create interfaces that maximize productivity, train users via visual cues and color hierarchies, and minimize frustration due to poor task design implementation. The eight areas highlighted are as follows:

1.) Consistency

2.) Shortcuts

3.) Feedback

4.) Closure Dialogue

5.) Error Handling

6.) Action Reversal

7.) Support Internal Control

8.) Reduce Memory Overload

Consistency:

--Are familiar icons, colors, menu hierarchy used with maximum consistency? Y

--Is information conveyed in a standardized way so users are able to apply current knowledge of the system and there exists no need to learn new methods of use for the same action? Y

--Are calls-to-action consistent across all tasks? Y

Shortcuts:

--Shortcuts available for repetitive tasks for frequent users? N

--Shortcuts simplify frequent tasks and save user time on repetitive actions and or tasks? N

--Shortcuts designed to maximize ease of use for frequent users? N

Informative Feedback:

--System offers informative feedback to users? Y

--Uses color, highlighted field, and or other variation to optimize feedback visibility? Y

--System designed to offer error messages to users? N

--Error messages designed to provide maximum detail needed to address error with task and or action? N

*We need to implement error messages(see more detail in error handling section)

Closure Dialogue:

--Dialogue box displayed for users after each task is completed? N

--Dialogue designed to provide closure post task completion? N

--Dialogue box displayed after completion of all tasks? Y

--User knows when interaction with system complete? Y

*We need to implement completion notification for each task.

Error Handling:

--System provides consistent error messages with instructions to handle error? N

--Instructions are simple and provided in a step by step fashion? N

--Missing input handled with appropriate message/visual cue? N

*We need to consider adding error messages when appropriate and it needs to be designed with simplicity for users to handle.

Action Reversal:

--Designed to allow users to edit input? Y

--completion action reversal done in simple manner/possible? N

--System designed to allow action reversal with user time in mind(I.e edit does not take too much time)? N

*We need to consider allowing users to reverse task completion in case of mistakenly closing task or entering wrong input potentially or even with possibly order editing while possible.

Support Internal Control:

--System designed to allow users to initiate actions? Y

--System designed to give users a sense of maximum control over events occurring in the system? Y

--System behaves as expected upon task initiation? Y

Reduce memory overload:

--System designed with a simple interface with proper info hierarchy? Y

--Interface designed to simplify tasks by use of recognition rather than recall? Y

--System designed to train users via cues and or notifications to recognize tasks needed to be completed? Y

Inspection 2

Evaluation Goals:

In our evaluation, we will test the overall usability of the application. We will list the associated issues with each task and highlight the for developers to pay more attention in the development iteration. The results will be used for guidance and suggestions of what needs to be resolved in the upcoming versions of the application.

Methodology:

Our evaluation was based on the usability heuristics defined by Jakob Neilson. This method will help us explore different aspects of the application, evaluate the problems, and highlight them for possible improvements in the design. The ten usability heuristics are defined below:

1. Visibility
2. Real world and system match
3. User control and freedom
4. Consistency and Standards
5. Error prevention
6. Recognition versus recall
7. Flexibility and ease of use
8. Aesthetically pleasing and minimal design
9. User ability to recognize, diagnose, and recover from design
10. Help and documentation

Visibility of System (Keeping the users informed)

1. When an operator performs an action/task, is there some sort of feedback? **Y**
2. Is there a pop up field for error messages? Can the user see the error/error details? **N**
3. When data entry is required, are pages/fields labeled clearly and show visible relationships? **Y**
4. If the user makes a mistake, can they go back. **Y**
5. If the users need more information when performing a task, is that available? **N/A**

The visibility of the TMS system is proficient. We may be missing an important aspect which includes error messages. There is no screen that shows what happens if an error occurs. If

presented with an error, the user should be able to view anything associated with it. The user is sent other messages and notifications that let them know the actions were successful.

Real world and system Match

1. Are there visual clues? (i.e. colors, metaphors) **Y**
2. Are the icons tangible and familiar? **Y**
3. Are any shapes used? If so, do they match cultural norms? **Y**
4. Do colors match expectations of common color codes? (i.e. red = stop) **Y**
5. Is the navigational structure balanced and in depth? **N/A**
6. Are menu choices logical and clearly defined? **Y**
7. When navigating the app, are the menus in order to reduce memory overload? **Y**
8. Are related fields shown on the same screen? **Y**
9. Is the language clear? **Y**
10. Does the language match that of the user? **Y**

User control and freedom

1. Can users easily go forward to the next task or go back if needed? **Y**
2. Does the system have multiple pages related to entering data of the same type? If so, can they easily go forward and back amongst the pages to enter the required info? **Y, Y**
3. Can users skip pages/fields when no information is applicable? **N/A**
4. Are buttons to move forward and back or exiting, marked appropriately? **Y**
5. Is there any information available if the user makes a mistake? **N**
6. Can users personalize what they see? (i.e. default views) **N**
7. Can users personalize how they receive notifications? (soft tone, vibrate) **N**
8. Can users change settings? **N**
9. When a task is complete, does the system automatically clear it? Does this require action from the user? **N, User must swipe, that clears the error**
10. Are users prompted to confirm task assignment/completion? **Y**
11. Can users reverse/cancel any task or operation? **N**

This could be for future iterations later down the road, but currently the app is not customizable. Each user should be allowed some sort of customizability. For example a vibrating feature might be desired. How do the users know when they receive a notification if they are not looking at the watch? Also default views may be helpful. Users should be allowed to customize the layout or prioritize a table or task that wasn't done by the system or administrator.

Consistency and Standards

1. Is navigation consistent across all screens? **Y**
2. Is task completion consistent on all screens? **Y**
3. Is assigning tasks consistent on all screens? **Y**
4. Is default color coding consistent? **Y**
5. Are the appropriate colors used and all screens? **N**
6. How many colors are used (no more than 4 should be used) **3**
7. Is regular feedback provided when completing/assigning a task? **Y**
8. Is feedback straightforward and to the point when user error occurs? **Y**
9. Are the menu items consistent across all screens and platforms? **Y**
10. Consistent stylistic design? (icons and buttons) **Y**

In our first version, we need to focus on the color coding of the application. Color codings can be used for prioritizing the tasks the users receive. They should be consistent with the mental model and matching on all screens. Using standard colors as so will ensure the vocabulary and operation of the application is consistent.

Error Prevention

1. Are potential human errors avoidable? If they are about to make a big mistake are they clearly warned/notified? **Y, N/A**
2. Are touch screen buttons big enough to avoid fat finger mistakes? **Y**
3. Does the touch screen work if it gets wet or users fingers are wet? **Y, Degraded**
4. Do data fields have pre populated default values? **Y**
5. Are the touch screen area items spread out sufficiently and not crowded? **Y**
6. Is data input case sensitive? **Y**
7. If a user accidentally closes a task, can they restore or go back? **N**
8. Is there invisible touch screen space where a user taps and nothing will happen? **Y**

To prevent errors, we should think about implementing confirmation dialogs, especially when a user closes out a task. One they close a task, a dialog could pop up stating “Are you sure you want to mark this complete?” There should also be an undo option. Similar to undoing a sent email, the user could undo a swipe for up to so many seconds.

Recognition versus Recall

1. Does the user have to navigate between multiple screens? If so, are the menu maps or any navigational aids? **Y , Y, Home Button**

2. When entering data, are there any optional fields and are they marked? **N/A**
3. Does the user know how to mark a task complete? Are there any cues? **Y, Y**
4. Are controls grouped together by action? Do they show the sequence in action for task completion? **Y**
5. With areas of text, is the space proportionate and adequate among them? **Y**
6. After a user completes a task, is there feedback that it was completed and is it clearly annotated to move to the next action? **Y**
7. Are prompts formatted correctly and visually appealing? **N, Font size may be too small?**
8. Can the user tell the state of the system by looking at the screen? **Y**
9. Is color contrast (bright and dull) reflected appropriately across the system? **N**

This is set up well. The user receives notifications when they need to do something. All that is required to remember is to swipe to close. Messages provide the status of their tasks. The app operates on smartwatches similar to the native apps, providing maximum recognition.

Flexibility and ease of use

1. Is there a search box or button? **Y, Button**
2. Is the search area easily recognizable? **Y**
3. When searching is there an auto completion feature or suggestions? **N/A**
4. Are buttons and fields easily understandable? **Y**
5. Is there more than one way to complete a task? (i.e. swiping to complete) **N**

Overall the system seems flexible and easy to use. However, we may be missing some functionality with the watch app. It looks as if all they can do is swipe. What are other things that can be done from the watch besides swiping to complete tasks, taping to open them, and receive notifications? What if the server would like to report a problem? They should be able to do that from the device. Perhaps a command for holding down a notification or table button?

Aesthetically pleasing and minimal design

1. When making a decision, is all required information portrayed on the screen? **N**
2. Is the layout clear and annotated appropriately? **Y**
3. Are there images on screen? Can they be viewed on one screen without scrolling? **N/A**
4. Do captions aid in understanding images, buttons, fields? **Y**
5. Are icons/buttons recognizable and not cluttered? **Y**
6. Are on screen animations clear and recognizable? **Y**

7. Do icons and buttons stand out from the background? **Y - implement color coding?**
8. Are icons and buttons grouped based on relationships? **Y**

Currently the background is dark grey and buttons are lighter grey. It may be difficult for some users to tell the difference between the fields and background. Implementing color scheming will help with this.

User ability to recognize, diagnose, and recover from design

1. When there is an error is it clearly notified? **N**
2. Is whatever needs to be changed/undone marked? **N**
3. Can the user unswipe a completed task? **N**

Mistakes are common and happen because all of the users are human. In our application, it should be easy to recognize and undo any action performed by the user. When the user swipes a task having the option to go back can be crucial. We could have standard error codes in place. For example, if they accidentally closed something that was out of order, the system could prompted with a message stating you must bring the appetizer out before the main course or what not.

Help and Documentation

1. Are there instructions when performing user tasks? **Y**
2. When a user selects or closes a task is there more information provided, explaining what has been done or what is selected? **Y**
3. Is there a visible help button/screen? **N**
4. Can the user easily go back from the help screen if needed? **Y**
5. Is all information or open tasks easy to find? **Y**
6. Can the users report a problem for follow up? **N**
7. If a user access help, can they go back to work where they left off? **N/A**
8. Does task marking automatically save? If the user were to lose connection, when they get it back does it go to where they left off? **Y**

Evaluation Test Plan For TMS

Introduction:

The Table-Management Smartwatch (TMS) will simplify the work of restaurant servers by alerting them of the next task or customer interaction they should address. The intention of the TMS is to take a series of complex, parallel tasks — managing several tables at once, all at different points in their dining experiences — and to make that job feel like an easy-to-manage series of sequential tasks

Objectives:

This test is designed to find primary users' impressions and quick reactions towards TMS, and actual infeld usability of TMS, including time saving, message delivering, and interface usability.

Tasks:

- Time management and time saving
- Information delivery and organization
- "Learning curve"

Time management and time saving

General:

For this part, we will test the efficiency of the interface and as well as comparing our task/order management function (Is the display user friendly? Are the timer and reminders helpful? etc.).

Method:

We will use *design walkthroughs* to display possible scenarios to figure out the potential result of our test.

Detail:

- This test will take about 20 minutes for each participant to finish, and overall about 1 hour and 40 minutes to finish the whole section.
- This test would require 3-5 servers or managers as our participants.
- We will provide illustrations of our products and explanations of our products to all of the participants then go through the scenarios and questions with the participants to collect answers as our data.

Key guiding thoughts and scenarios:

Looking at our designed product interface, are the letters legible and the compositions reasonable for users to skim through and find the correct bottom in a timely manner?
When changing a shift while there are ongoing serving on a certain table, can the server log in in less than 30 seconds and catch up with the left tasks? After the server finishes table one, and need to go back to the tasks on the last table, can they use a shortcut (such as scrolling to the opposite direction) instead of taking a long time to scroll all the way down?

Is it easy and clear for users to find the tasks that they need to complete, such as a shortcut? Can users differentiate the urgency of tasks clearly? Are the time calculations reasonable? (Important)

When a server comes back from break and does not know what is going on, can they catch up in 1 minutes? When it is rush hour, can the server find out what's the most urgent task? And quickly after finishing most of the tasks, the server was reminded to serve beverage to a table, is the calculation trustworthy (80% accuracy) that the server does not need to reflect and only need to take the action?

Is the reminder strong enough for the user to sense it but not so strong that it became a distraction or noise?

When walking through a table of kissing couples while the remainder came up, will the couple notice the reminder and become annoyed or they could not hear it?

Information delivery and organization

General:

For this part, we will test on the successful delivery of the messages and information between servers and managers to make sure it is efficient.

Method:

We will use design walkthroughs to display possible scenarios to figure out the potential result of our test.

Detail:

- This test will take about 15 minutes for each participant to finish, and overall about 1 hour and 20 minutes to finish the whole section.
- This test would require at least 2 servers and 1 manager as our participants.
- Assuming the participants already know about our product, go through the scenarios and questions with the participants to collect answers as our data.

Key guiding thoughts and scenarios:

- Can the server receive the messages in time with least delay? When several messages come at once, is the display mechanism strong enough to guid the server through the messages in the shortest time?
- When a server was preparing orders, the task reminder came up, and then the message quickly came up too, can the server tell the difference? When he walked out and wanted to serve a table, the manager wanted to ask him to stop, can he receive the message before walking to the table or after arriving at the table? Can the manager set different notification bizz for specific messages so it is faster for the server to receive the information, without looking at the screen?

“Learning curve”

General:

This part is aimed to find out how user friendly TMS is in general.

Method:

We will use design walkthroughs to display possible scenarios to figure out the potential result of our test.

Detail:

- This test will take about 5 minutes for each participant to finish, and overall about 30 minutes to finish the whole section.

- This test would require 3-5 servers or managers (preferably same ones from previous tasks) as our participants.
- Assuming the participants already know about our product and have an understanding on how easy it is to learn the functions, it is okay to directly go through the scenarios and questions with the participants to collect answers as our data.

Key guiding thoughts and scenarios:

- Is it self-intuitive enough for users to learn the basic functions in 10 minutes and become adapted after 1 day of using?
- When a restaurant ordered TMS for the servers, will they be able to start to use the TMS when the day it arrived and will they be able to fully embrace the watching in a period of time that will not affect their performance heavily? Can they make up the money spent on TMS in a short time?