README for PayAbility Medium-Fi Prototype

Contents:

Tools

Tools used:

In designing the user interface for our medium-fi prototype, we used Sketch to create the layout and appearance of each screen in the application, and we used Marvel to stitch together the transitions between screens.

How the tools helped:

Both Sketch and Marvel had a low learning curve, which made it very easy to start building our medium-fi prototype with these tools even though we had never used them before.

Sketch helped us give our prototype a more realistic look by providing an iOS template that we could use to create buttons that resemble those found in commercial iOS applications.

Marvel helped us give our prototype a more realistic feel and flow by allowing us to add iOS transitions in between screens, as well as simulate button clicks.

Marvel also had the added benefit of allowing for multiple team members to collaborate on a single project online.

How the tools did not help:

Both Sketch and Marvel did not help us to develop any backend infrastructure to allow our prototype to perform calculations such as comparing prices and calculating tax, so we had to rely on Wizard of Oz techniques and hardcoding to demo some of the features in our application.

Both Sketch and Marvel did not help us prototype quickly, as it took a long time to use these tools to build our prototype.

These tools also did not help us simulate some of the more advanced features of our application, including using the camera to scan bar codes and price tags, collecting the user's location information, and creating dynamic cartoon animations.

Moreover, Sketch made it hard to work together on designing the prototype because it does not have any online collaboration features. Sketch also made it difficult to make changes to our prototype on the fly, as the program would try to force us to make edits across our entire template even if we just wanted to modify a single button.

Similarly, Marvel made it difficult to add multiple click hotspots per screen, which prevented us from from including the tip calculation screen.

Limitations

Limitations/tradeoffs:

Because the current prototype does not have access to the full set of technological features that are available on a phone, it cannot simulate using the camera to scan bar codes and price tags (since there is no outward facing camera on a computer), collecting the user's current geographic location using GPS data (since there is no GPS chip on a computer), and displaying dynamic cartoon animations of the girl pushing the shopping cart (since Sketch only allows us to create static visual elements).

Since the current prototype does not have the backend to support dynamically changing the interface based on what the user has input, the prototype is also limited in the sense that it can only work with a specific set of items in a specific order, which we describe in the Instructions section of this document.

Moreover, due to the limitations of Marvel, some of the screens and features we designed are not included in our prototype's task flow, even though they have been drawn out. For example, the tip calculation screen is not included in the prototype's flow because we could not simulate the functionality of being able to select 'yes' and 'no' multiple times on the same screen.

Wizard of Oz techniques:

We used Wizard of Oz techniques in determining the best combination of bills and coins to display to the user in the "Payment Options" screen when they are ready to pay. In reality, we would need to develop an algorithm that makes several mathematical calculations on the fly to translate a final price (say, $295) into a reasonable set of bills and coins that can be used to pay for that price, in addition to creating a machine learning model to present the most useful sets of bills and coins first. For this medium-fi prototype, however, we simply rely on Wizard of Oz techniques to pretend that we can successfully translate a final price into a set of bills and coins and also present the most useful set of bills and coins first, even though we have not yet built the backend to support this functionality.

We also used Wizard of Oz techniques in determining what item the app recommends the user should remove from their cart if they exceed their budget. In reality, we would need to develop an algorithm that determines what item can be removed in order to bring the total purchase amount below the budget limit, while implementing a machine learning model that takes into account the user's preferences and past purchase history to make sure that the item that is recommended to be removed is one that the user does not care too deeply about. For this medium-fi prototype, however, since we have not yet developed the algorithm or machine learning model to make such recommendation possible, we rely on Wizard of Oz techniques to pretend that we can successfully take the price information and the user's purchase history and preferences into account to make a good recommendation for what item the user should remove if they go over their budget. We hardcoded the pencil as the recommended item to remove in our demo.

Hardcoding

We hardcoded the total budget amount and the price of each item that needs to be compared or purchased in our demo (both before and after tax), since the prototype does not yet have the backend built in to dynamically perform the math calculations to determine which of two prices is larger or how much tax needs to be added to an item.

We hardcoded the price that would result from using the barcode scanner, since the prototype does not allow us to use the camera to actually scan any barcodes but we still wanted to show that the barcode scanning functionality is available in the app.

Moreover, since the cartoon animations in our interface need to be updated based on what items have been added and how much of the budget remains but our prototype does not yet support dynamically updating animations on the fly, we also had to hard code the specific items and the order of the items that can be purchased in the demo.

The hardcoded amounts and prices are:

$0 for the starting budget
$300 for the updated budget

$5 for the price of the red notebook
$10 for the price of the gray notebook

$231.90 for the pre-tax price of the laptop
$250 for the post-tax price of the laptop

$9.28 for the pre-tax price of the pencil
$10 for the post-tax price of the pencil

$41.74 for the pre-tax price of the laptop
$45 for the post-tax price of the laptop

$295 for the final purchase amount

The hardcoded order of the items to purchase is:

1. laptop
2. pencil*
3. calculator

*The pencil is also hardcoded as the recommended item to remove when the user goes over budget, since we have not yet built the algorithm to dynamically determine which item to recommend the user to remove.

Finally, we hardcoded the user's current location at an office supply store. In reality, we would need to use the phone's GPS chip, in addition to data from a map service such as Google Maps, to approximate the user's current location. For example, if we determine that the user is currently at an office supply store, we can offer the hint that office supplies are taxed and don't require a tip. Similarly, if we determine that the user is currently at a grocery store, we can offer the hint that raw fruits and vegetables are not taxed and also don't require a tip. However, since we do not have access to GPS or map data in our demo, we hardcoded that the user is currently at an office supply store so that we can offer an appropriate hint to help the user figure out whether they need to add tax or a tip to their purchase.

Instructions for individual tasks

Please open the app prototype on a **phone's** web browser (such as Safari on iPhone or Chrome on Android) by going to the following link: https://marvelapp.com/78g661

Task 1 (Medium): Compare prices

1. Click compare items
2. Compare a $5 red notebook and a $10 grey notebook
3. Click the Add Another Item to scan the barcode for the red notebook and then click compare to scan the barcode for the grey notebook (the compare button essentially acts like the person bringing their phone to the item)

Task 2 (Simple): Set A Budget

1. Click set budget
2. Input a budget of $300 using the on screen calculator

Task 3 (Complex): Pay with exact change

1. Buy three items, a pencil, a laptop, and a calculator in that order
2. for the pencil, manually input the price 9.28 and include tax and no tip
3. for the laptop, scan the barcode for the laptop by clicking the barcode in the top left
4. for the calculator, scan the barcode for the calculator by clicking the bottom right corner of the barcode frame