PayAbility

by Nicole Crawford, Lynne Sneed, Jorge Cueto, and Musila Munuve

“Empowering people with math learning disabilities.”

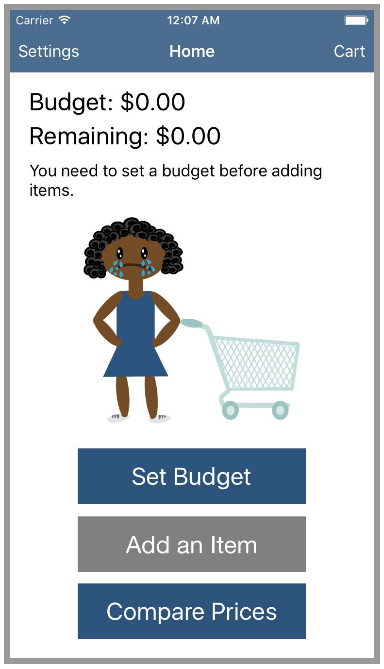
Problem and Solution Overview

Problem

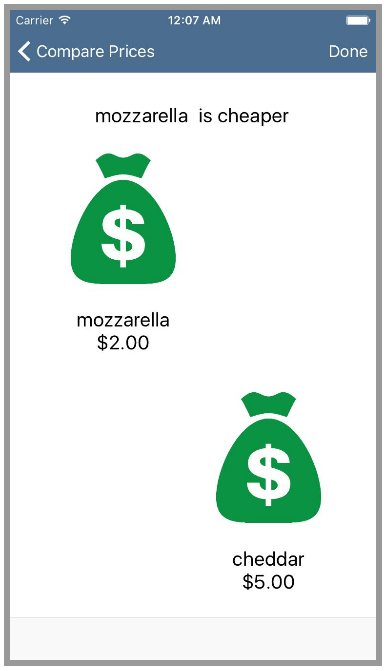
Over 5% of people have disabilities such as nonverbal learning disorder and dyscalculia, rendering them unable to do basic math calculations, such as determining whether a $5 bill is greater than a $10 bill. In daily life, this makes it difficult to compare prices and pay for things while shopping.

Solution

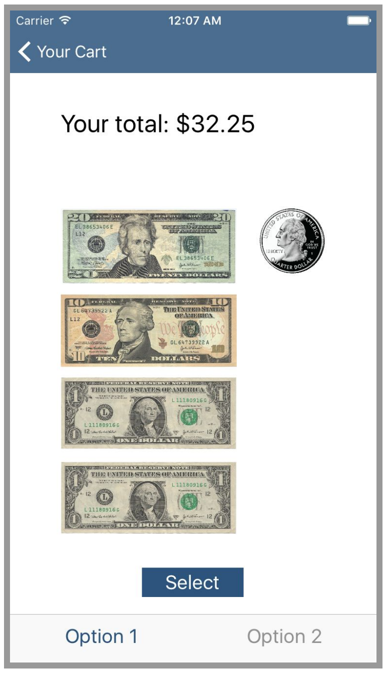
To make paying easier and less stressful, the PayAbility app helps users stay within budget, compare prices, and pay with exact change by using pictures to depict quantities in an emotionally engaging way, as you can see below.



Budget: $0.00
Remaining: $0.00
You need to set a budget before adding items.



mozzarella is cheaper



Your total: $32.25

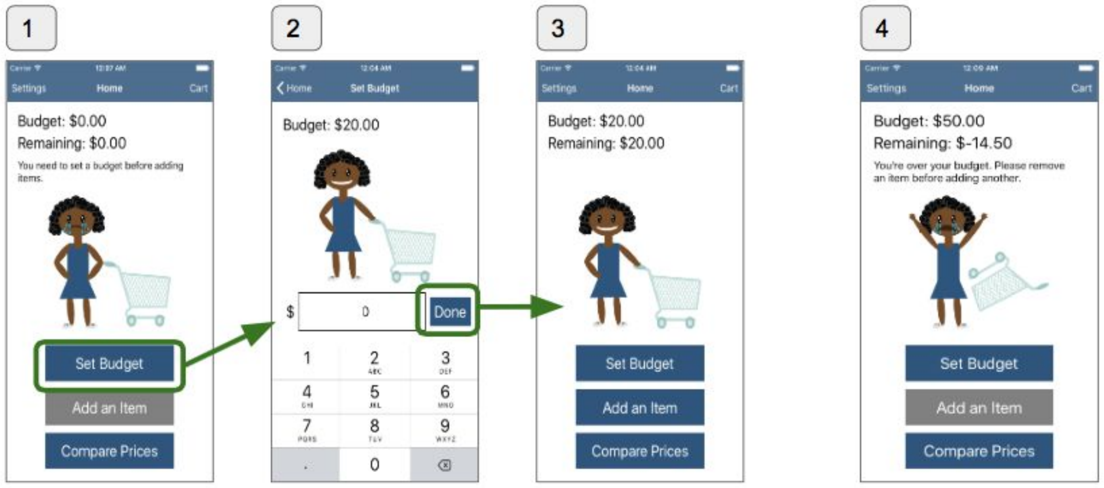
Tasks & Final Interface Scenarios

Our three main tasks were “Stay Within Budget”, “Compare Prices”, and “Pay With Exact Change.” We ultimately decided to focus on these three tasks because they directly address fundamental needs that we discovered during the needfinding interviews we conducted with individuals who have math learning disabilities.

Simple Task: Stay Within Budget

PayAbility helps users to stay within a budget by prompting users to input a budget amount before they can start shopping and showing users their current budget status in an engaging and easy-to-understand way with pictures of a girl pushing a shopping cart. We decided to address this task because needfinding interview participants shared that they have difficulty keeping track of how much they spend at the store, since their math learning disability makes it difficult to understand the magnitude of numbers and quantities.

Stay Within Budget Storyboard


Budget: $0.00
Remaining: $0.00
You need to set a budget before adding items.

Budget: $20.00
$ 0

Budget: $20.00
Remaining: $20.00

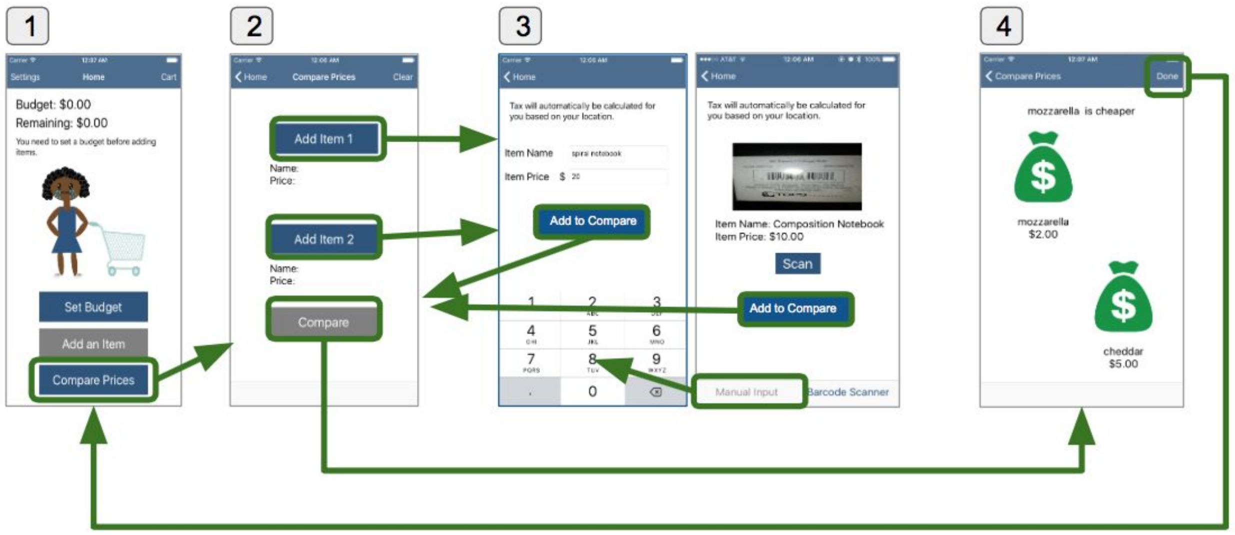
Budget: $50.00
Remaining: $-14.50
You're over your budget. Please remove an item before adding another.

1. Upon opening the app, the user sees the Home screen. If the budget is 0, the user will be prompted to input a budget amount before they can add items to their cart (the Add an Item button is gray because it is disabled until the user inputs a budget amount).
2. The user can use the number pad to input their budget amount and select Done to return to the Home screen.
3. Back on the Home screen, the facial expression on the picture of the girl reflects the current state of the budget in an easy-to-understand way. If a large portion of the budget remains, the girl is happy, and if the user is close to going over the budget limit, the girl is nervous.
4. If users ever go over their budget when adding items to the cart, they are prompted to remove an item from their cart to stay within budget before they can add another item to their cart or checkout.

Medium Task: Compare Prices

PayAbility helps users to compare prices by allowing users to input the price of two items and then showing which one is cheaper using an easy-to-understand picture of money bags. We decided to address this task because needfinding interview participants shared that they have difficulty comparing the prices of items at the store, making the process of choosing which product to buy more stressful.

Compare Prices Storyboard

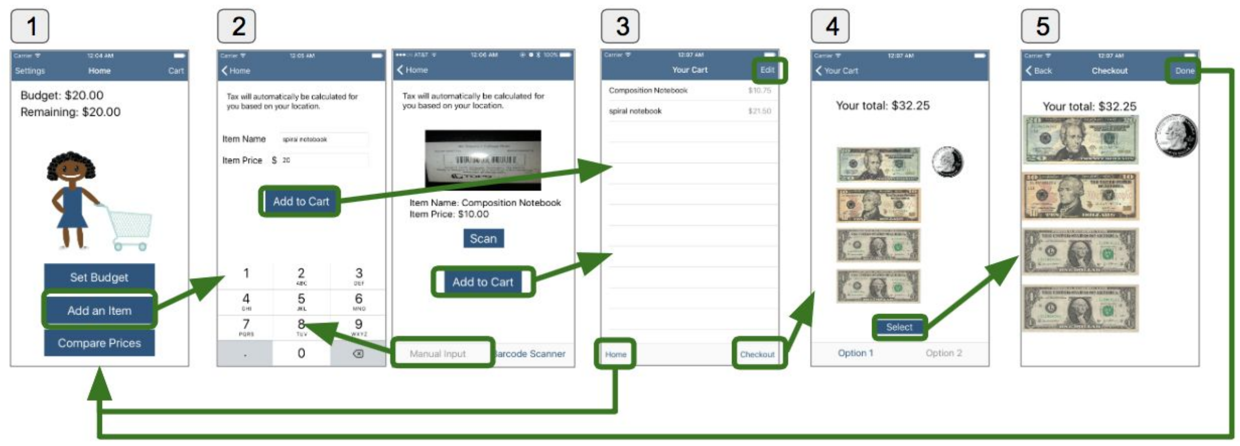


1. From the Home screen, the user can select the Compare Prices button, which takes them to a screen that allows them to add two items to compare.
2. Clicking on the Add Item 1 button or the Add Item 2 button takes the user to the item input screens in Step 3. The Compare button is disabled until the user adds both items to compare. When both items are added to compare, the user can select the Compare button to see the comparison results in Step 4.
3. There are two ways to input items to compare: manually inputting the item's name and price using the number pad and keyboard, as well as using the iPhone's camera to scan a barcode and automatically extract the item's information. The user can toggle between the two input methods using the Manual Input and Barcode Scanner buttons located along the bottom of the screen. The user can select the Add to Compare button to add the item to their comparison and return to Step 2.
4. The comparison results visually illustrate which item is cheaper in an engaging and easy to understand way using money bag pictures. The cheaper item is shown in a "lighter" money bag floating at the top of the screen, while the more expensive item is shown in a "heavier" money bag that has sunk to the bottom of the screen. Moreover, the screen indicates which item is cheaper with a message at the top of the screen. The user can select the Done button to return to the Home screen.

Complex Task: Pay With Exact Change

PayAbility helps users to pay with exact change by allowing users to add items to their cart, automatically calculating tax for users based on their location, and then displaying pictures of the bills and coins that users can use to pay for their purchase when they are ready to checkout. We decided to address this task because needfinding interview participants shared that they have difficulty figuring out which bills they need to hand the cashier to pay for their purchase. One interviewee mentioned feeling anxious while taking out money from his wallet at the cash register, as he worried that he was taking too long to pay, and he could sense people were impatiently waiting in line behind him. Another interviewee mentioned that one time, she was tricked into paying more money than her actual total.

Pay With Exact Change Storyboard



1. To start shopping and adding items to the cart, users can select the Add an Item button from the Home screen, which takes them to the item input screens in Step 2.
2. There are two ways to input items to add to the cart: manually inputting the item's name and price using the number pad and keyboard, as well as using the iPhone's camera to scan a barcode and automatically extract the item's information. The user can toggle between the two input methods using the Manual Input and Barcode Scanner buttons located along the bottom of the screen. The user can select the Add to Cart button to add the item to their cart, at which point the user is taken to the Cart screen in Step 3.
3. In the Cart screen, users can see a list of all of the items they have in their cart. The tax is automatically calculated for each item based on the user's current location. From the Cart screen, users have several options. Users can select the Edit button in the top right corner of the screen to select an item to delete from the cart (which can be especially helpful if users have gone over their budget), users can select the Home button in the bottom left corner of the screen to return to the Home screen if they want to add another item to their cart, check their remaining budget, or compare prices, and users can select the Checkout button when they are ready to see how they can pay for their purchase.
4. When users select the Checkout button from the Cart, they are brought to the Payment Options screen in Step 4. This screen shows the total amount at the top of the screen, in addition to displaying a small thumbnail image of the bills and coins that can be used to pay for the total amount. The user can view different payment options that display different types of bills and coins by selecting the Option 1 and Option 2 buttons along the bottom of the screen. The user can tap the Select button to pick a Payment Option and proceed to Step 5.
5. Users see a magnified version of the selected payment option, allowing users to put their phone down on a table or counter and easily glance at the payment option displayed on the screen as they are taking the money out of their wallet. When they are done paying,

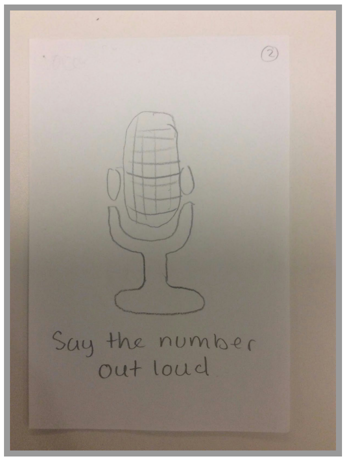
users can select the Done button on the top right corner of the screen to return to the Home screen.

Design Evolution

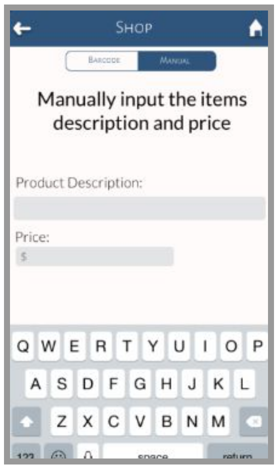
In the journey from our initial needfinding, to our tens of sketches and storyboards, narrowing down to one design for our Low-Fi prototype, improving upon this design for our Medium-Fi prototype and finally incorporating our Heuristic Evaluation to build our High-Fi prototype there have been a lot of changes in our design. In this section there will be a discussion of what these changes were and why they were made followed in the next section by a discussion of the changes made as a result of the Heuristic Evaluation

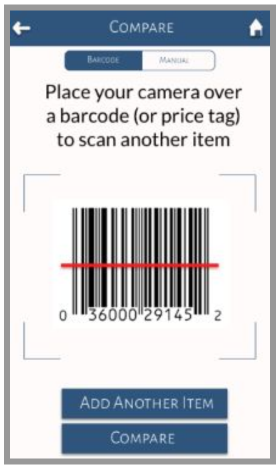
The first change we made was switching from a microphone as way of the user inputting their current product and it's details to add it to the cart. The reason we had this in the prototype was to make input as easy as possible in an already stressful situation. Another reason was to avoid the use of a numeric keyboard which we thought may have brought back the very problem with numbers that we were trying to solve. In our round of testing our participants pointed out that the microphone may be impractical because of how often they would have to speak into their phone and how conspicuous it would be. To fix this we changed our interface to include a manual input with a qwerty and numeric keyboard and a barcode scanner. Allowing our user ease of input as well as freedom to choose between two different methods.

Low-Fi Prototype Item Input Screen
Voice Input



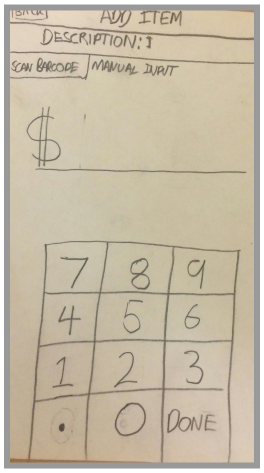
Medium-Fi Item Input Screens
Manual Input Barcode Input

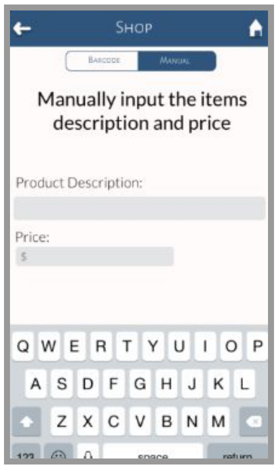



Compare
Barcode
Manual
Place your camera over a barcode (or price tag) to scan another item
Add Another Item
Compare

In our next round of testing we received feedback on the manual and barcode input when we watched our users interact with this feature. All three of our participants struggled when instructed to switch between the manual and barcode mode and this was an immediate red flag. We initially had the options laid out in a similar format to the tabs of an internet browser. To fix the problem we changed it to a segmented control bar which is familiar, as it's used in other mobile apps already. You can see the updated design in the right-hand side screenshot below.

Low-Fi Prototype Item Input Screen Medium-Fi Prototype Item Input Screen

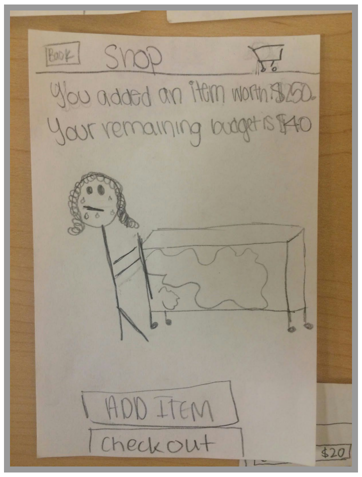


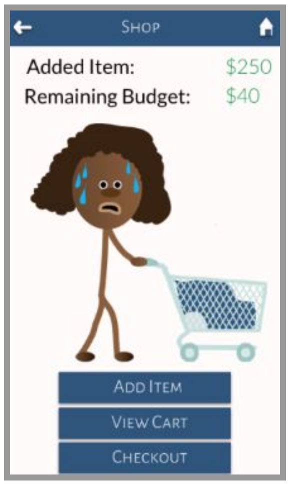


One change that came internally as our prototype became more sophisticated was to add animations into the app. The animations are present throughout the storyboard and mimic the user's emotions in a lighthearted way that engages them emotionally and makes the process less daunting. This idea came after observing our user interact with the interface and seeing the lack of emotional reaction during this process. As the fidelity of our prototype increased so did the quality of our animations and the users' engagement with them.

Low-Fi Prototype Home Screen

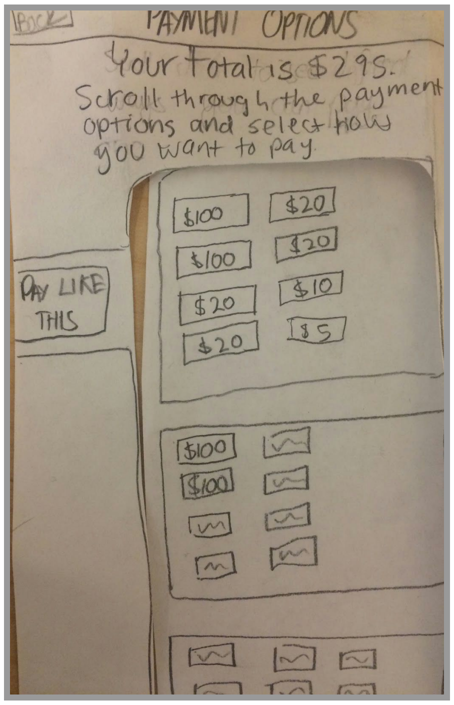
Medium-Fi Prototype Home Screen



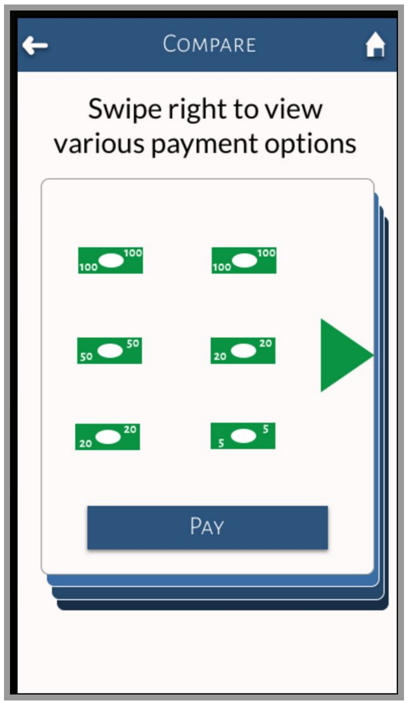


The last and possibly most important change we made was the payment options screen. The objective of this screen was to show the user in a visual form exactly how much they had to pay for the items in their cart and the different currency combinations they could use to do this. This is the aspect that we found our target audience had the biggest challenges with and therefore we dedicated a lot of time thinking about and iterating through different designs. Initially we had the user scrolling down a list of different options before selecting the one that matched the currency they had in their wallets at the time. We sketched a few more ideas before settling on a format that would have the user look through different options, only one being on screen at any given time, before selecting one. This seemed to solve the confusion that users had with the first representation by using a simple minimal representation.

Low-Fi Prototype Payment Options



Medium-Fi Prototype Payment Options



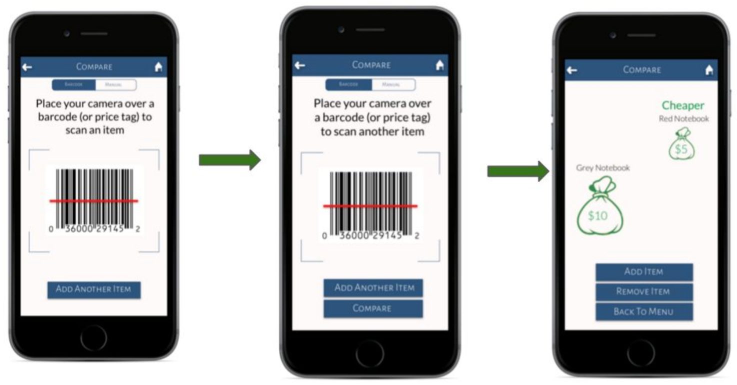
Major Usability Problems Addressed

1) [H2-7 Flexibility and Efficiency of Use] [Severity 3]

Evaluators commented, "Ended up going in a loop trying to figure out how to compare the two items in the prototype. I would click the manual tab and enter the pencil, and then press continue shopping and add the laptop but after I added the laptop, the compare prices screen didn't come up and it was back to having no items in the cart. The comparison only came up if I clicked 'add another item' under the barcode, which isn't on the manual page and I overlooked it several times. Make this button more prominent and on the manual input page."

This problem was due to implementation feasibility in Marvel. In Marvel, allowing users to click on the same button multiple times was not possible. Therefore, we had to include additional buttons that would not be featured in the actual design so users could complete the task. We gave specific instructions in the readme describing how to complete the task since it was not intuitive based on the design because of this. For the final prototype, we explicitly state that the user can only compare two items and remove all unnecessary buttons.

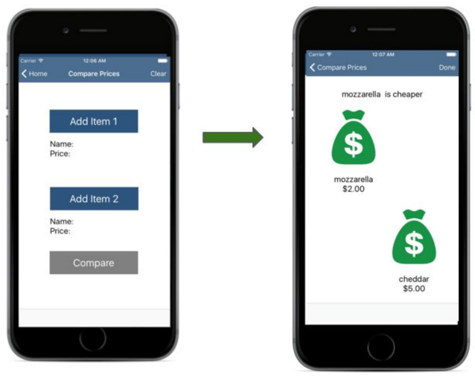
Before:


Compare
Place your camera over a barcode (or price tag) to scan an item
Add Another Item
Compare
Place your camera over a barcode (or price tag) to scan another item
Add Another Item
Compare
Compare
Cheaper
Red Notebook
$5
Grey Notebook
$10
Add Item
Remove Item
Back To Menu

1) Click the Add Another Item to scan the barcode for the red notebook

2) Click compare to scan the barcode for the grey notebook (the compare button essentially acts like the person bringing their phone to the item)

After:



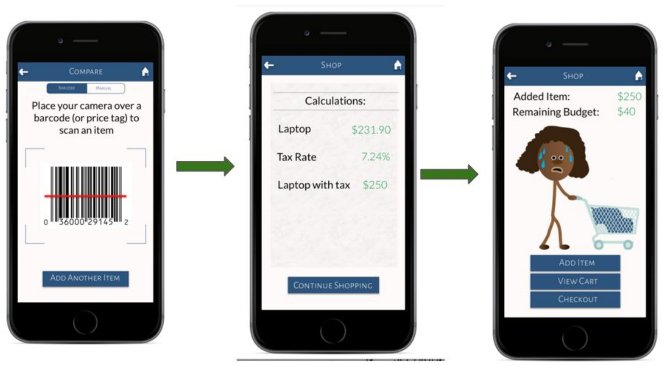
All user input steps are on the same screen.

2) [H 2-4 User Control and Freedom] [Severity 3]

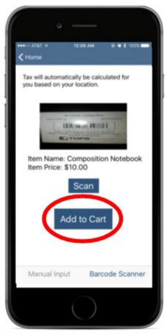
Evaluators commented, “Users may not always want to add an item immediately after scanning it. There is no way to ‘undo’ an added item or to check before adding it.□”

We fixed this issue by bringing users to a confirmation screen after they scan the item. Before, the item would be automatically placed in the cart.

Before:



After:


Home
Tax will automatically be calculated for you based on your location.
Item Name: Composition Notebook
Item Price: $10.00
Scan
Add to Cart
Manual Input
Barcode Scanner

After scanning an item and seeing the calculations, it was automatically added to the cart and removed from the budget.

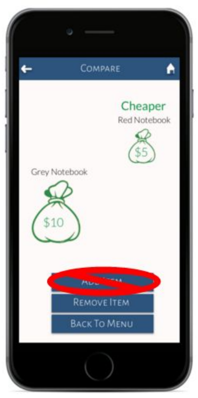
As shown after scanning, the user can decide whether they would like to add the item to the cart.

3) [H 2-5 Error Prevention] [Severity 3]

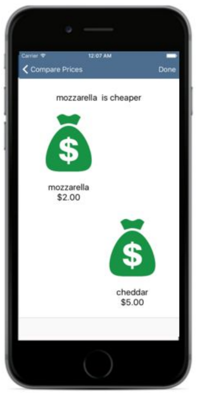
Evaluators commented, "On the compare screen where the comparison between items is shown, it is unclear what the 'add item' button will do: is it adding one of these two items to the cart or is it adding another item to the comparison?"

We agreed that there was no need for an add item button on the comparison screen. We fixed this by completely removing the add item button and allowing users to click on the moneybag to add it to the cart for the final prototype.

Before:



After:



4) [H2-7 Flexibility and Efficiency of Use][Severity 3]

Evaluators commented, "The compare items page doesn't have a clear way to add items from the comparison to your cart, which forces the users into unnecessary additional actions."

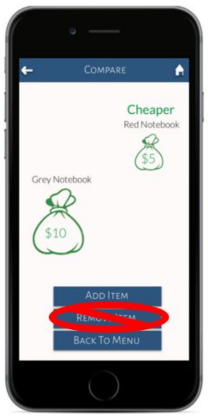
See above for explanation of design changes.

5) [H2-7 Flexibility and Efficiency of Use][Severity 3]

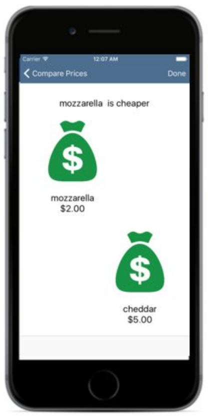
Evaluators commented, "The compare items page 'remove item' button seems to force more clicks than necessary. Removing items could be as simple as pressing an 'x' next to the item."

We decided that the remove item button did not make sense while comparing prices because you are not removing items from anything. Therefore, we completely got rid of this button for the final prototype.

Before:



After:



6) [H2-7 Flexibility and Efficiency of Use][Severity 3]

Evaluators commented, "Not having a login, hinders the user in a number of ways: If they accidentally close the app, they may lose all of their information, and the Shop button can't 'learn' what dollars and coins they use most often."

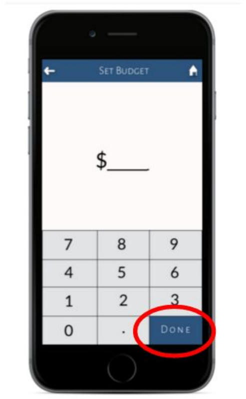
We did not implement a login because the user is not interacting with other users in the app. The information for a given user can be stored on their specific phone.

7) [H2-3 User Control and Freedom] [Severity 4]

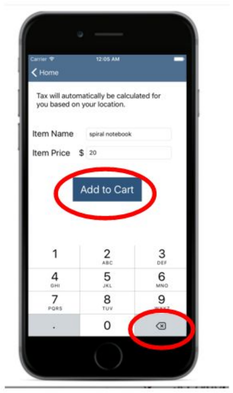
Evaluators commented, "When I'm inputting a price manually, I can't erase any of the numbers I've inputted if I make a mistake; you need a backspace button."

We redesigned the calculator to include a backspace button so users can correct errors easily, and we moved the add to cart button next to the text area. Before, there was only a done button that was in the place of the backspace button.

Before:



After:


Tax will automatically be calculated for you based on your location.

Item Name spiral notebook

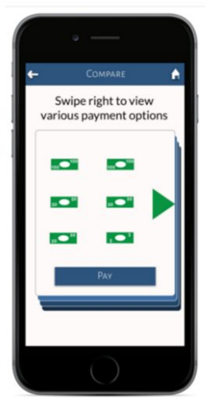
Item Price $ 20

8) [H2-3 User Control and Freedom] [Severity 4]

Evaluators commented, "The screen where the user can swipe to select a payment option can be overwhelming as the possibilities of how to pay for something increases to extreme amounts as the total price goes up (so many different UI screens). To simplify this, the UI should ask the user what bills they have and how many of each they have, and then display the possibilities from there in order to prune the seemingly infinite combinations of payment options."

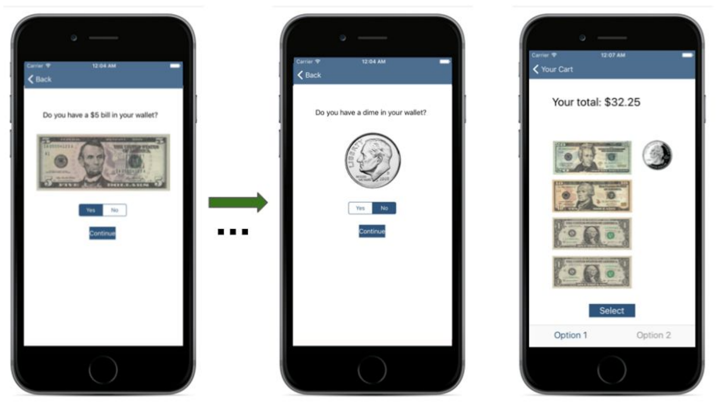
To address this problem, users can now go to the Settings screen from the Home screen in order to input exactly what types of bills and coins they have to limit the number of payment options shown. Also, through Wizard of Oz, a machine learning model organizes the order of the payment options based on bills that the user typically uses. Before, users would not enter what type of bills they had and would just be brought directly to the payment options screen.

Before:



Users just see their payment options after checking out.

After:

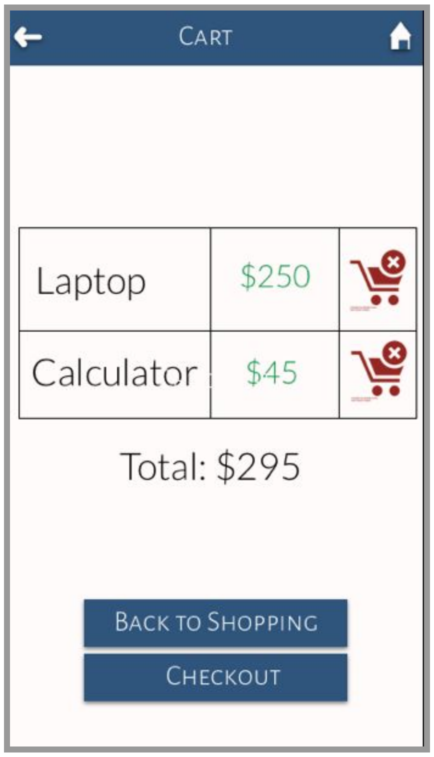


Users are looped through every type of bill and coin to indicate which types of bills and coins they currently have.

Payment options are based on the money they currently have in addition to the machine learning algorithm.

Furthermore, we made some other modifications to our design to make the app easier to use and also in response to significant implementation hurdles that we encountered while coding our app in XCode. For instance, we modified the Cart screen from the Medium-Fi prototype, making it more in line with iOS's standard UITableView model. We removed the red cart buttons and simply utilized the built-in "swipe to delete" feature from the UITableView. We could not figure out how to add a button icon to the UITableView, so we decided to use the built-in feature. This change may ultimately be for the best, as most users are familiar with the "swipe to delete" feature since it is incorporated into many high-profile iOS apps, such as Apple's iOS Mail app. From the screenshots below, you can also see that we decided to replace the special font and cream-colored background from the Medium-Fi prototype with the more conventional System font and white background used in iOS, as you can see below. These aesthetic changes, which are present across all of the screens in our app, make the app easier to read on a small iPhone screen and also allow users who are familiar with these conventional iOS visual characteristics to feel right at home in our app.

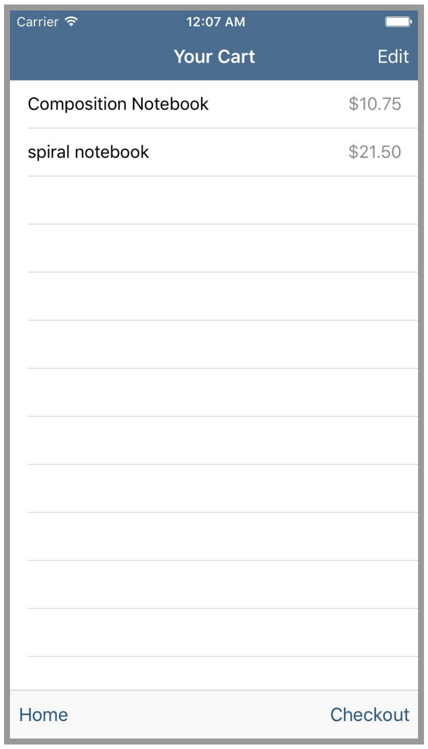
Medium-Fi Cart



Laptop	$250	
Calculator	$45	

Total: $295

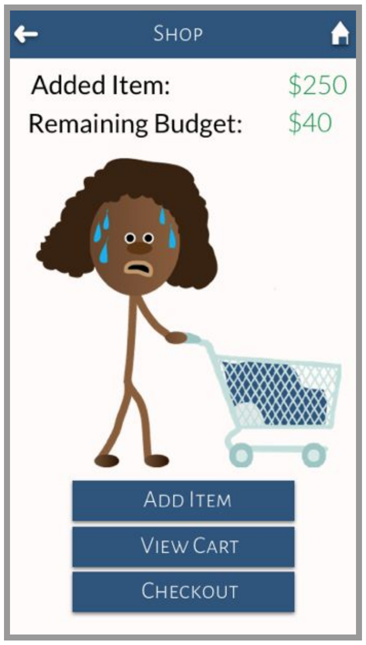
Hi-Fi Cart



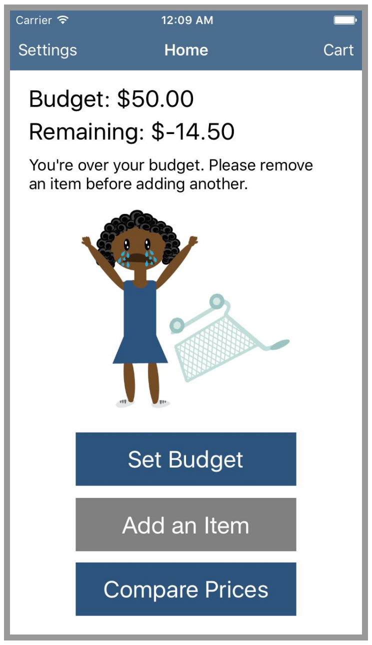
Composition Notebook	$10.75
spiral notebook	$21.50

Finally, while we were coding the app in XCode, we found out that it was difficult to dynamically modify images that were placed in a UIImageView. As a result of this implementation obstacle, we removed the ability to see different colored blobs appearing inside the cart in the picture that is on the Home screen. Instead, we decided to focus on more clearly conveying different emotions through our Home screen picture, making the illustration of the girl more detailed and realistic, as you can see below.

Medium-Fi Home Screen



Hi-Fi Home Screen



Prototype Implementation

The Hi-Fi prototype is an iOS app that was built in XCode using Objective C. XCode was helpful because the storyboard interface made it easy to visualize how all of the views in our app could come together and also create transitions, or segues between screens. Moreover, dragging and dropping various visual elements, such as buttons and labels onto the storyboard interface made laying out each view an effortless experience. Objective C was helpful because it was relatively easy to learn, considering that we had never used it before. The included frameworks available for making iOS applications also provided some useful functionality, such as the ability to store variables in NSUserDefaults, which allowed us to save certain values that we wanted to access from all parts of the app, including the user's total budget and their total purchase amount.

However, in some ways XCode and Objective C were not very helpful to us as we were building our Hi-Fi prototype. XCode is only available on Mac, and collaboration on XCode is difficult because putting together code from different sources is not a straightforward process. In some instances, files would disappear and have to be uploaded into the XCode project again. In addition, some of the visual elements provided by XCode to build iOS apps in Objective C were not very customizable. For instance, we found it difficult to alter the default appearance of the UITableView as well as the toolbar along the bottom of the screen.

A couple of features in our app required the use of Wizard of Oz techniques. We used Wizard of Oz to simulate the feature of calculating users' tax rate based on their location. Since calculating tax automatically would require access to the phone's GPS, in conjunction with a database containing detailed information about different locations and the tax rate that exists in each location, we set the tax rate of each item to be 7.5%, the California state sales tax, whenever a user adds an item to the cart. Although our prototype actually does not use GPS data, this Wizard of Oz technique allows us to pretend that we can access the user's current location and that we have a database that allows us to determine the tax rate based on their location.

We also used Wizard of Oz to determine which combination of bills and coins the user needs to pay for their total purchase amount. Dynamically determining which bills and coins to show for each total would require creating a machine learning model and being able to generate images of all possible combinations of all possible totals, which would be difficult to implement in a limited time span. Instead, we pretend that we are somehow able to figure out what bills and coins to show the user. However, the payment options are limited to any combination of inputs of $10, $20, and $30 items, and there are only two payment options per total purchase amount. In addition, we do not use the bills and coins that the user sets up in the Settings screen, since this would require creating a very broad range of bill and coin combinations.
In building our app, we also used hardcoding to streamline the development process. For example, since we would need to create and manage an extremely large database to support most barcodes and provide accurate item information across various stores, we ultimately decided to just hardcode the result of the barcode scan so that it always return the same item information - Composition Notebook, $10.00. Moreover, as mentioned in the paragraphs above, we have a limited set of payment option images that have been hardcoded for totals that result from any combination of a $10, $20, and $30 item. For example, for a total of $10.75 ($10 with the hardcoded 7.5% tax rate), we always return a picture of a $10 bill and three quarters as Payment Option 1, and two $5 bills and three quarters as Payment Option 2, regardless of which bills and coins the user selected in the Settings screen.

There are a couple of features that are currently missing but could be added in the future. We could use the barcode API's provided by Amazon and Walmart to return valid item name and price information for products that are sold in those to stores, such that we would no longer have to return the same hardcoded result each time the barcode scanner is used. We could also incorporate an optical character recognition API in order to collect item information based on the text that is on item price labels, to provide a third input alternative to the existing manual input and barcode input methods.

Summary

Overall, we believe that we have ultimately succeeded in designing an app that can empower people with math learning disabilities to go shopping with confidence. After conducting needfinding interviews, we discovered that common tasks such as staying within a budget, comparing prices, and paying with exact change proved to be quite stressful and daunting for people with nonverbal learning disorder and dyscalculia. We realized that we could have a significant and real impact on the lives of these individuals by creating an app that specifically focused on making these tasks easier and less stressful buy using fun and engaging pictures to depict different money amounts. Throughout the process of designing, prototyping, and evaluating, we constantly modified our design, adding new features and removing others. After conducting user testing using our Low-Fi prototype, for instance, we realized that a manual input option was easier than a voice input option for users. Similarly, after conducting a heuristic evaluation, we learned that users would like to input the types of bills and coins they have in their wallet, so we added this feature in the Settings. After a series of design iterations, we finally built our Hi-Fi prototype as an iOS app using Objective C. Though we rely on Wizard of Oz and hardcoding techniques for some features, our Hi-Fi prototype provides users with all of the key functions that we set out to build, and we are proud of what we have accomplished this quarter. There is a lot of potential to build upon the design foundation we have created and create a positive impact on the lives of people with math learning disabilities.

Bibliography

For icons and images used in our app and website.

- Money Bag: Created by Igor Yanovsky from the Noun Project
- Animated Black Girl: Created by Lynne Sneed
- Shopping cart w/ Animated Black Girl: Created by Jorge Cueto
- Blobs in Shopping Cart: Created by Lynne Sneed and Jorge Cueto
- Shopping Cart Icon: Created by Franc from the Noun Project
- Logo Design: Jorge Cueto
- Aisle Sign Design: Lynne Sneed
- Header image on website: http://www.sablog.kpmg.co.za/wp-content/uploads/2014/05/Happy-black-shopping-woman-smiling-at-the-mall-holding-bags.jpg
- Person Icon on Poster: by Tahsin Tahil From the Noun Project
- Shopping Aisle Image on Pitch Slide: http://opt-food.com/wp-content/uploads/2014/11/img_2746.jpg
- Tutorial followed by Lynne for Animated Black Girl: http://design.tutsplus.com/tutorials/how-to-illustrate-a-cute-basketball-player-with-simple-shapes--vector-5937
- Arrow Icon on Poster: by Pham Thi Dieu Linh from the Noun Project