Introduction:

My problem domain stemmed from an issue I had experienced myself. I and those I know have routinely found it difficult to easily split bills, whether it's a bar tab, a grocery bill, or housing, it's always more difficult than it should be. Thus, I set out to design an application that would leverage the technology in my phones to make this process faster and simpler. There are certainly numerous options to solve this problem already on the market. Venmo, is a major player that doesn't do receipts, but is widely used as an actual payment processor for bill splitting purposes. Splitwise, is a full featured application that is superb for splitting monthly expenses with housemates and keeping track of small bill splits occurring between friends. However, Splitwise does not offer any sort of receipt scanning to quickly split bills. Finally, Tab is a new player and the most like my implementation. Tab allows scanning a receipt and turning it into a simple list that users can split their bills with. They're accomplishing many of the things I want to, but only support PayPal, which is frustrating to users.

Starting my design, I had a few major goals in mind. I wanted to use the camera to take a picture of a receipt to split and then use text-recognition software to grab the important details of the receipt and display it in the app itself. This was my primary goal to make the splitting process easier. However, while the design is relatively simple, the implementation of this technology would be a major task, which would be rife with technical challenges. On top of receipt scanning, I also wanted to allow splitting to occur without a receipt if one wasn't available or needed. Once the items from the receipt were recognized and added to the application, I wanted to allow users to pick the items they were solely responsible for or were going to share with other users. This required the entire bill to be shared with other users of the application and allow editing/selecting to occur simultaneously. The design for this sharing process was one I really had to think about to make it as seamless and quick as possible. Once the selection of items has occurred, I wanted to make sure tax/tip was evenly distributed with the ability to make small additions or reductions if needed. Finally, users could request or send money to the other users with a variety of payment methods, which would be routed through my application and then sent via the preferred payment method. This would allow a Venmo user to pay a PayPal user seamlessly. This one feature left little to design because it would be a major issue to work out engineering and business wise, but from a user standpoint it would be an invisible step.

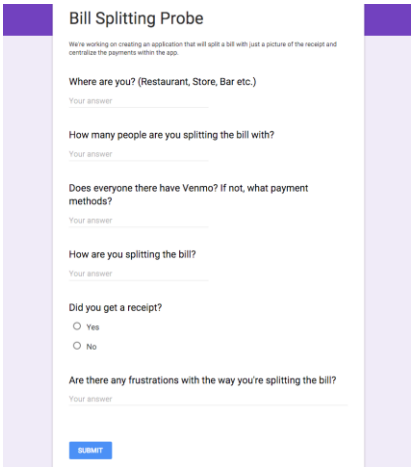
Understanding:

In my previous project I conducted contextual inquiries in a sit-down interview format, working with an existing product to find breakdowns and areas for improvement that my users encountered. However, all I had for this project was a perceived problem area without a product to test. Thus, I had to find out if my target users felt my perceived problem and use their feedback to create the solution. My user population was comprised of students who consistently have a reason to split bills and receipts (eating out etc.), all use iPhones, and agreed bill splitting was too difficult of a process.

I used a cultural probe, specifically diaries to gather information that I could use to build a solution to my problem. In order to make the process as simple as possible, I created a simple Google form that my participants could fill out after each task. The form asked a few basic

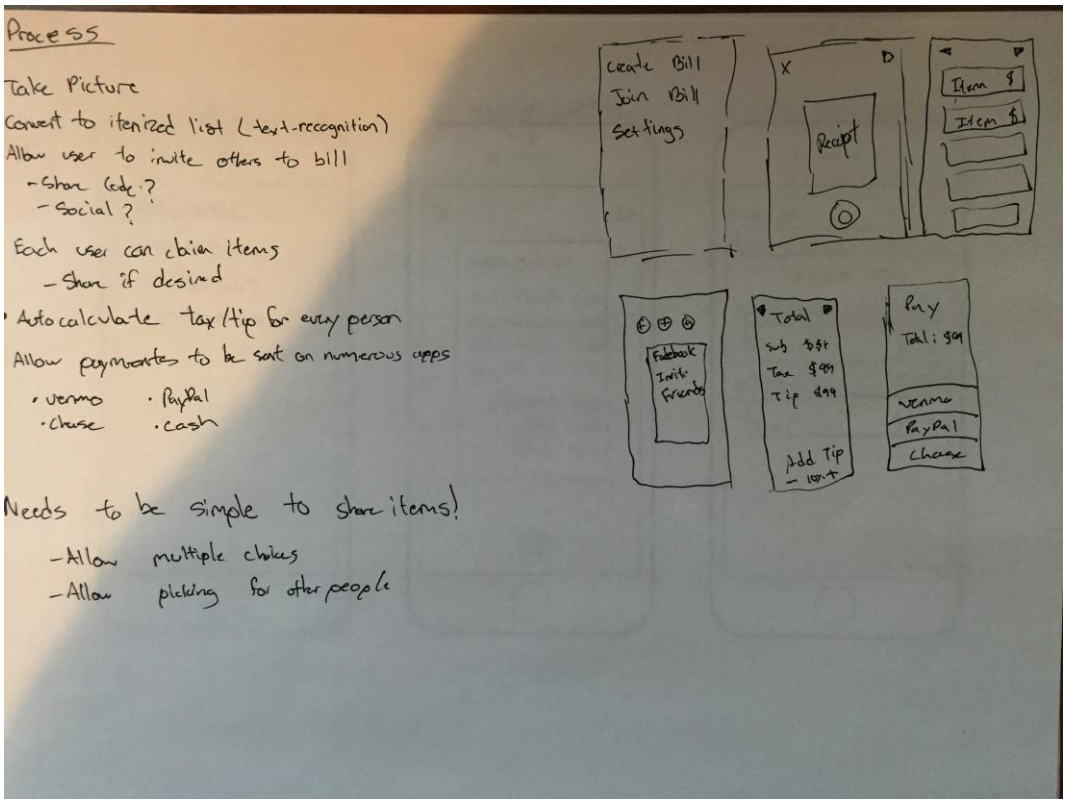
questions such as the location/business where the bill split occurred, how many people were involved, the method involved, if a receipt was taken, and most importantly the frustrations that arose with the bill split. After gathering 10-15 responses from each of my participants I compiled all my data into an Excel spreadsheet to try and find repeated information that might shed light on problems, breakdowns, and opportunities. I immediately saw frustrations that arose due to different preferred payment methods, sales tax issues, remembering cash repayments, and counting individual items from a receipt. These major issues represent the breakdowns that occur in my participants' current splitting methods, but also the opportunities to design solutions to fix them. Having all my data in an excel spreadsheet was useful, but to pare down all that info into what I thought was most relevant required an affinity diagram. I categorized my affinity diagram by "issues" regarding receipts, payment methods, people/location, as well as flow ideas for my own solution. This allowed us to better structure my user data and add my own comments so I could easily build relevant models.

My most insightful model came through the flow model where I was able to see how users communicate with each other, discussing the best method of splitting bills. While analyzing the data from my diary studies I found this was where users were most frustrated. When large groups of friends are forced to pay with one receipt, things can get very complicated if not everyone has the same form of payment service. All my users found themselves in this situation multiple times so trying to pay people back with cash resulted in a long and difficult process. Some people had to go to ATMs to get cash where there were fees involved, others only had large bills which required breaking down, and in all cases, no one was able to pay the exact amount, someone always ended up overpaying. On the other hand, when my users went out with people who all had Venmo the process and communication of splitting the bill was painless because everyone can pay exactly what they only down to the cent in a quick and easy way.



Ideation:

My contextual inquiry made it apparent that my participants had serious problems with their current process of bill splitting. Breakdowns such as difficulty matching preferred payment

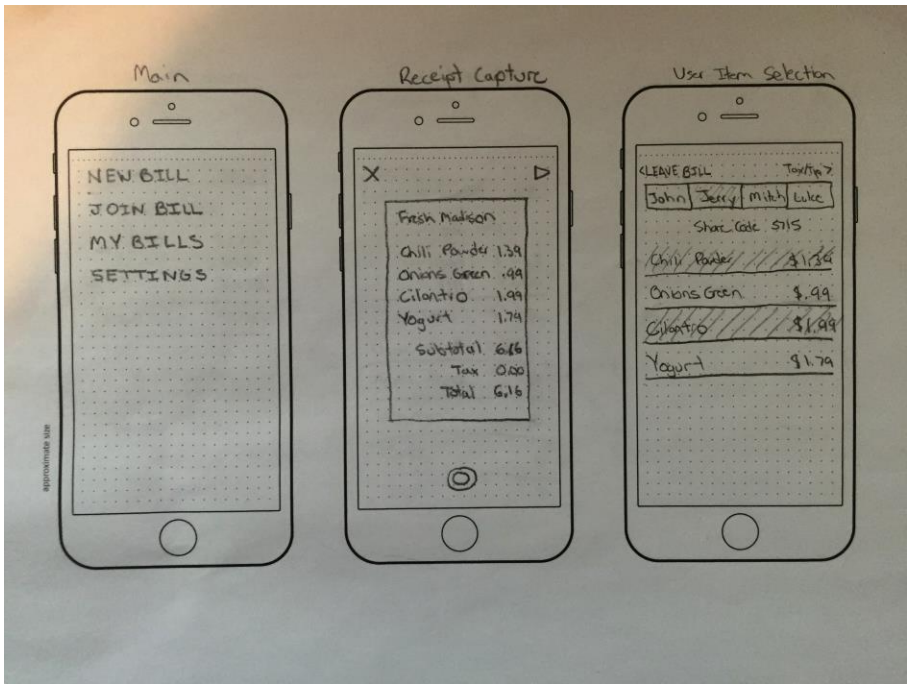


methods, tracking cash payments, counting individual items, and dealing with sales tax and tips were common among all my participants. Most importantly, knowing these breakdowns allowed us to recognize opportunities for my application to solve them. The most major breakdown that occurred and is the primary function of my design was the need to count and add up every item on a receipt by hand. This took my users a lot of time, was often inaccurate, and left the responsibility on one person. With today's text-recognition software this doesn't have to be a problem anymore. By leveraging this software, I can design an application that takes a photo of the receipt and deciphers receipt items, sales tax, tip etc. and format all of this into an easy-to-use interface. Then allowing users to join the bill, picking their own items, and paying each other accordingly will greatly reduce the strain of counting, tallying, and keeping track of a receipt. One problem I had anticipated in the first place and was noted by my participants was not being able to pay each other back unless everyone had the same payment application. This was a major problem I wanted to tackle and would be more of an engineering issue than a design one. I wanted to make it possible to pay someone with Venmo even if the other person only had PayPal. So, from a design standpoint, I wanted to allow numerous payment options for users at the final repayment screen while keeping the dirty work of transferring etc. in the application itself. Tracking cash payments was also a difficulty for my participants, but luckily this again could be solved with a small design feature. Just as I would allow various virtual payment methods, I would allow cash payments that on the bill would be recorded and easily checked who paid whom with cash or other methods. A small breakdown that frequently occurred was the issue of sales tax and tip splitting. With cash this splitting was nearly impossible, as these

amounts rarely were neatly divisible. Even with payment applications that allowed perfect splitting, the sales tax and tip had to be calculated for each person by hand or in a calculator. Typically, this was just too much for most people, so someone ended up just taking most of the obligation, which was a major frustration. By designing my application to recognize the sales tax/tip amount and divide it evenly in proportion to each user's bill I would solve this problem entirely.

Prototyping:

To begin my prototyping, I wanted to take my basic ideas and sketches and bring more precision, clarity, and direction to them. To do this, I found iPhone screen templates that I could mimic an app design on paper. The paper prototypes were very similar to my final design, much in part due to the extensive ideas I had on the design before prototyping even began. I started with the simplest main screen I could think of. I played around with more options, but ultimately, I wanted users to get into the app and be able to dive in without a cluttered interface. Starting a new bill, I wanted the option to take a picture of a receipt using the built-in camera, once again keeping as little in the way as possible. Finally, I reached the first real design opportunity, the item choice screen. Here I wanted to display items in a table fashion with the name and price easily viewable. When selecting the items, I needed to deal with the case where multiple users wanted to split an item. I first were creating duplicate items to allow this, but as this got to be too cluttered, I had one item that could be claimed by multiple people and the price would be split for each user's total. I also needed to set a specific area to check multiple user "tabs", so users could both see which people had joined the bill and check on individual tabs. I decided to allow users to add users to the bill without even using the share code or social invite, simply in the case where someone didn't have the app and wouldn't be able to participate. In this way the other users could account for another person's obligations without requiring an app download immediately. I toyed with the idea of allowing others to pay on that person's behalf but dropped it as it might create too much confusion, so simply being aware of the obligations was enough. Once all items were selected, I wanted to show the total bill amount for everyone with sales tax and tip included. This design choice was simply to let everyone know the entire bill size and allow small changes such as an included tip, discounts, or changing the tip size across the whole bill. Once again, I strove for keeping it uncluttered and as clean as possible. Finally, my last screen in the process was the user "tab" screen with payment options. This screen displayed each user's tab, which could be seen by everyone and signified each person's calculated portion of the bill. As stated in my original goal, I wanted to allow multiple payment methods. I wanted to make this a one-screen process, so I integrated these payment options on the tab page itself. Ultimately, the user only needs to click on their preferred payment method, the user to pay and the transaction can be completed.



Evaluation:

In this project I primarily used the method of user performance testing to evaluate my design. Using CogTool I was able to simulate user performance in a quick and easily testable method.

One of the biggest design challenges I faced was deciding how to best bring everyone who wanted to split a bill into the process. I bounced from having one person in charge of the splitting and then simply making a payment request, to social media integration, to a code that code be shared to join a bill, and for a location-based signal that could be sent. The problem really was a matter of ease and speed; thus, user performance testing was a perfect evaluation method. I decided the best options were social media integration via bill invites and a share code that users could tell or text their friends. Thus, I designed two solutions that I tested with CogTool.

One added point is the importance of freedom or flexibility. This was something I thought of only at the end of my designing. I realized that while the Facebook or Twitter invites could potentially be quicker and would allow additional ways to notify users (say a Facebook alert or Twitter DM), this method was restrictive on who the bill could be split with. What if a user wanted to split a bill with someone, they weren't friends with on Facebook? Would they have to add the person on Facebook out of the app or could I allow that in the app? That might take a considerable amount of time and both users may not be interested in connecting via social media. In this case, a four-digit share code that could be given across the table or through a quick text message would likely be less intrusive, easier to use, and quicker.

Once I finished the high-fidelity outlines for both my designs, I were ready to implement them into CogTool. For both designs I had to take into consideration the time taken for the main user of the app to communicate with their friends to get them to join the app, the other users to join the app, and a short confirmation on how the bill is being split. Starting with the "Share Code"

design, the main user is given the share code right away and they can start selecting the items they bought, but in order to correctly split with friends they had to wait around 15 seconds for the other users to open the app and type in the code to join the bill. Thus, the main user didn't have to spend much time on the actual invites, but the wait for others was a bottleneck. One improvement here could be allowing the main user to not only choose their own items but add people even if they haven't accepted the invite and then pick items for that person. Then upon joining the bill the other user could select the entire profile of the previously added user and take over all the previously added items.

The "Facebook Invite" design on the other hand allowed the other users to simply get a push notification that would automatically open to the bill, which made the wait for users to join much quicker. However, with the Facebook invite, as it is quick for other users, it turns out to be very slow for the main user of the app. Instead of telling the friends sitting around him the 4-digit code, which takes about 6 seconds, they had to scroll through all their Facebook friends to find the ones they are sitting right next to, then make sure they have everyone selected and send out the invites. This process on average took the user over 30 seconds to complete for only 2 users and will get exponentially longer the more users they are splitting a bill with. At this point I realized this step could be drastically improved by adding a search functionality to the friends list search and possibly even adding a "frequently split" tab that lists frequent friends. Once all users were added into the app selecting what everyone ordered had similar times for both designs averaging at about 25-30 seconds, however it can only be completed once every user finish so this time mainly depends on the response time of other users or how quickly they can get into the app once they have a code or invite. With all these times taken into consideration and computed through the CogTool the "Share Code" design finished in 53.5 seconds and the "Facebook Invite" design finished in 83.7 seconds, which was the opposite of my original guess, primarily due to the long searching times to find other users in the Facebook invite screen.

Final Solution: When starting to create my high-fidelity prototype, it was time to take all the information I have gathered about my app, good and bad, to combine it into one working

mobile app. My main goal was to create a clean, organized and easy to use app that people

would enjoy using. Ease of use and reducing the amount of time in app were the two most import design features I planned for while beginning my designs in SketchApp.

Selecting Purchased Items

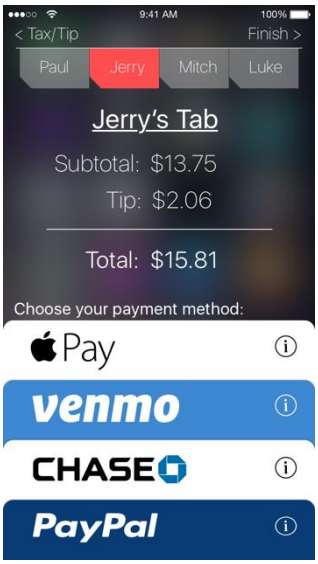
With a large group of people and only one receipt to pass around it can take a long time to pass around the receipt and have everyone select what they bought and remember the price. I not only gave the receipt to everyone but had the app read the receipt and divide the bill into selectable items. This significantly reduces the time it takes to split the bill because now everyone can see the full receipt and individually select what items they bought. On the screen you will be able to see live what other people are selecting and to simply split an item with someone you both select the same item, and the cost of the item will get divided accordingly.

Adding Friends to Bill

This step of the design process the most troubling part I had and there was a lot of back and forth about it. I came down to two main designs for trying to add friends to the bill as simple as possible. The first one is with a **Share Code** this is a simple 4-digit code that users would verbally tell the friends around them so they could open my app and quickly join the bill by using that code. It is very simple and quick which is exactly what I need and since the users are always right next to each other when splitting bills, it is easy to share the code with the people around you. My next idea of inviting people to the bill was through **Social Networks** this, just like any other app that uses this feature, would have you search through your friends list to find the people around you that you would like to invite. Although this function works and has its benefits it is significantly slower than just sharing a code, and since the people you are sharing with will always be right next to you there is no need to find them on a social network.

Payment Methods

As I learned from my probe study and from first-hand experience, splitting bills can become a long and frustrating process the more people you have and the more different types of payments that come with each user. I found in my study that even if all the users had some



form of mobile payments, they still had to resort to splitting the bill with cash because no one

had over lapping payment methods. In my design I changed that, enabling users to choose whatever mobile payment they prefer, having the application handle the transactions so the user doesn’t have to deal with any of the frustrations.