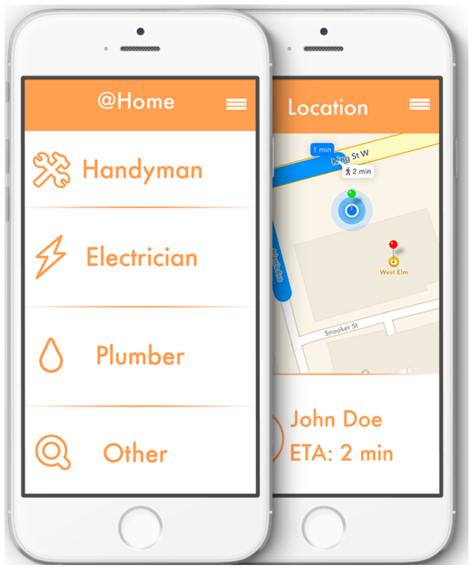
@Home - High-Fidelity Prototype Report

Adam Khorakiwala — Team Manager
Laila Chima — Development
Gavin Nelson — Design
Documentation and User Testing Shared

Problem and Solution Overview

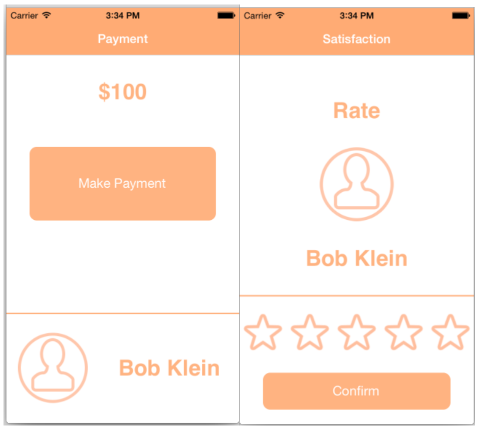
The garbage disposal is clogged. You've just chipped a bathroom tile. Your apartment is a mess. The leg on your chair snapped. These are just some of the many problems that we face in the home. While we'd all love to take out the time to make sure our homes are up to spec, the pace of life today and perhaps general laziness holds us back. For many of these problems, we need the expertise of a plumber or electrician and the efficiency of a handyman or cleaner. However, finding someone nearby with the skill, time, and willingness to do the job is time intensive enough to postpone the task to take care of more immediate concerns. @Home aims to solve these sort of home problems in the fastest and easiest way possible. Our solution is a platform that helps people access home service providers while helping service providers establish a market for their product. A customer can open the app, select a service, add some relevant details, and be matched with service providers near them. A service provider is notified when they are being matched and can choose to accept the request. When the customer requests the service, the provider knows where to go and proceeds to the location in order to provide the service. People need a faster, easier, and more reliable way to solve problems at home. People need @Home.



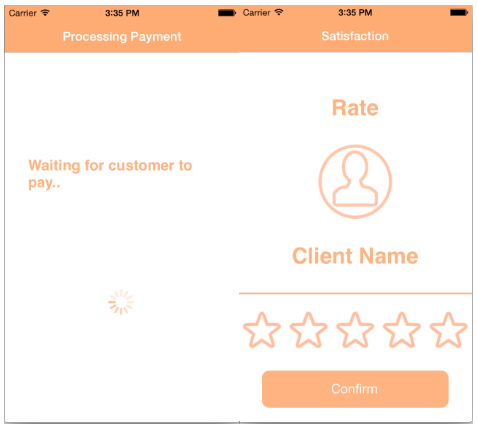
Tasks and Interface Scenarios

1. Processing a Payment (low complexity)

As soon as the service provider taps "finish" on his/her app, the user will be prompted to confirm the payment on their app. Please note that this prompting and ordering does not exist in the prototype because we have not implemented any server side functionality. Both parties will be required to rate one another after the payment process. We chose this task because we wanted our solution to be truly end-to-end, from trying to find a service provider or a customer all the way down to the final transaction. Only by pursuing an end-to-end solution could we truly solve the problem of poor usability and inefficiency.



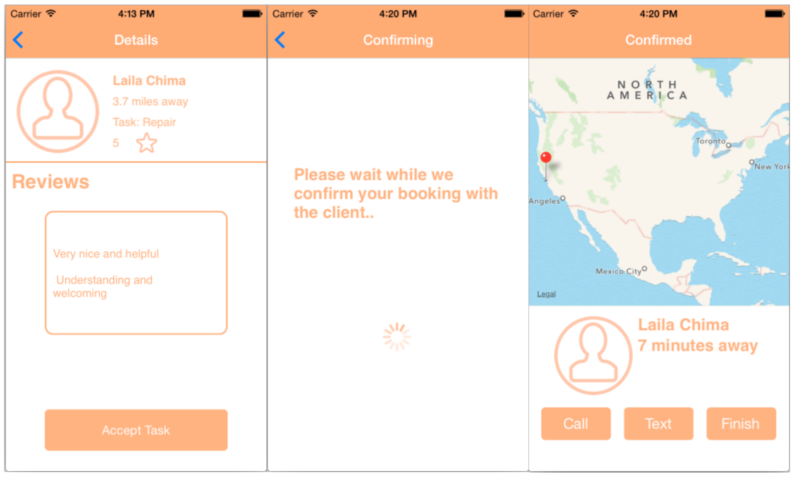
User Workflow



Service Provider Workflow

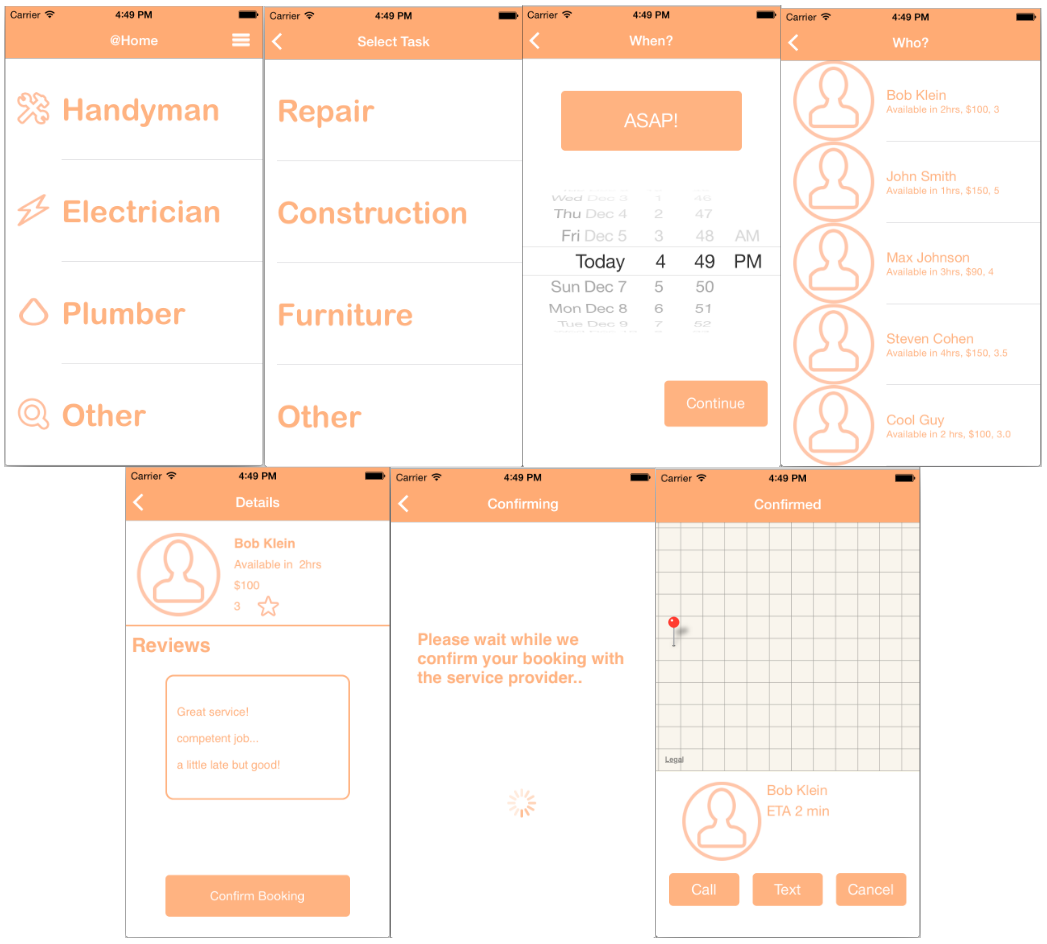
2. Responding to a request (medium complexity)

A service provider can respond to a request in two ways. If @Home matches a user search with a particular service provider, the service provider will be notified and can accept the match. Note that the notification functionality has not been implemented because of server side constraints. A service provider can also browse through a list of live requests that they are qualified and available for, from which the provider can select and accept a task. If the user selects the service provider as well, the service provider will be taken to a confirmation screen that will display all the relevant information regarding the task and the client. We chose this task because service providers told us that they needed an easier way to access potential clients. They also mentioned that they had no way of being able to fill free time in their schedule efficiently. After clicking the notification/selecting from list:



3. Finding and booking a service provider (high complexity)

A user requests a service by defining the category of problem and then specifying the particular task within that category. Then, the user defines a time within the next 3 days when they would like the service provider to arrive. @Home compiles a list of service provider nearby who are available to solve the problem. The user can go through the reviews, prices, and ratings of the various options and book one of his or her choosing. Then, the user is taken to the confirmation screen after booking the service provider. This is really the core task of @Home. In order to solve the problem of fast and easy home problem solving, we needed to make the search and booking experience seamless. Choosing this task was essential to solving the problem.



Great service!
competent job...
a little late but good!

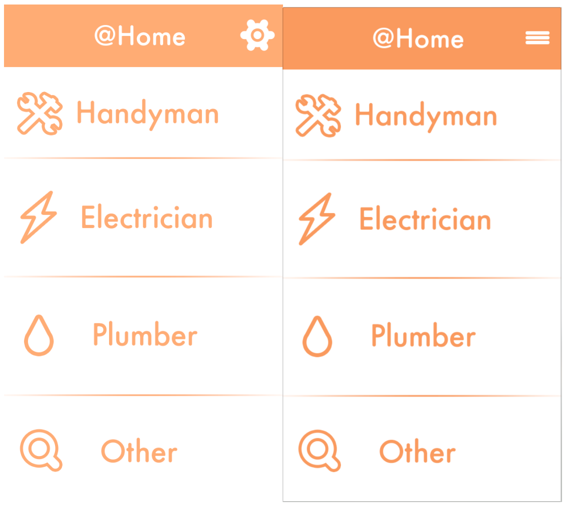
Please wait while we confirm your booking with the service provider..

Major Usability Problems Addressed

2. [Consistency and Standards (H2-4)] [Severity 3] [Found by: A,B]

The menu has an icon which appears to be a cog. Convention would state that you intend this option to be 'Settings'. When clicked, a page appears where you can select ''profile'' or ''settings'' which makes use of the same icon of a cog. How are these two ''settings'' buttons different? Moreover, accessing a profile from a settings list does not make intuitive sense. Fix: Instead of having the cog in the menu bar, there could be a icon with the three horizontal lines which is the convention used on most apps today as a ''main menu''.

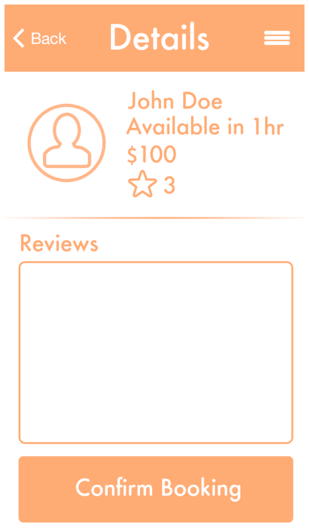
This was a severity 3 issue that was found by the group evaluating @Home. We decided to follow their advice and change the cog icon in the top right of the screen shown below to the three bars as the evaluation suggested shown below. After reading over these suggested changes, our group agreed with what they were saying and decided to make the changes.



8. [Error prevention (H2-5)] [Severity 4] [Found by: A,B]

When a service provider decides on a customer, it seems like tapping on the customer dispatches the provider automatically. This leaves room for mistakes if the provider is being quick with their fingers, in which case, the customer may be faced with needing a service and watching providers cancel their calls if they've made a mistake. Fix: There should be a clearly labeled button to confirm their reservation or some kind of confirmation screen.

This was a severity 4 issue found by the group evaluating @Home. We felt this was a very important issue to fix after it was brought to our attention because it could have lead to a lot of user errors in the app. We fixed this issue by adding a confirm screen for the customer and service provider where they can read review and go over the information one more time before clicking the clearly marked confirm button as pictured below.



17. [H2-5 Error prevention] [Severity 4] [Found by: B]

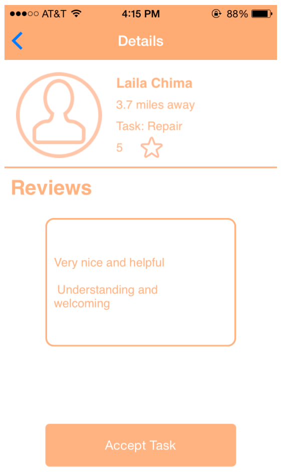
Unfortunately, there will be occasions where the customer feels it necessary to contend/dispute the charge amount. ‘Make payment’ does not allow for variation and may fool individuals by not allowing them to follow up the payment. Fix: Dissect the ‘make payment’ screen into the possible options of ‘dispute’ and ‘tip’. The service provider should also be able to edit the payment amount (for example, when expensive materials were needed over the time commitment).

This was a severity 4 issue that we made some initial changes from but did not totally stick to their suggestions. We felt that adding “dispute” and “tip” options in the app wasn’t the best idea because service providers provide an approximate quote and then complete their task. The customer is expected to pay the quoted amount and potentially more if the service provider needs to charge them for materials, but this will all be acknowledged while the service provider is working. Adding a dispute option on the payment screen wouldn’t lead to anything productive after the job is done. If a service provider needs to make the customer aware of an increase in price, he will do so before he buys the parts and continues working. If that is not ok with the customer, than the service provider will not finish the job and will not be left without being compensated after having finished.

19. [H2-1 Visibility of system status] [Severity 3] [Found by: B]

When choosing a potential customer, there may exist several factors that could influence your decision. The list of customers for the providers is rather devoid of information and makes me wonder if I should know about more these customers? The service provider doesn’t know whether he is capable of performing the task, or what tools he will need. Fix: Add a small tag to each individual on this screen explaining the job / critiques of working for this individual.

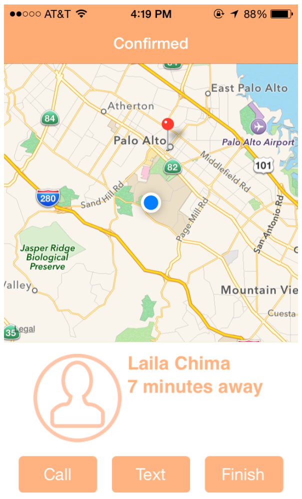
This is another severity 3 issue that was found by the group evaluating @Home. We felt this was a good suggestion and decided to make the appropriate changes. Now, the customer has the ability to add a more specific task through use of the other category, and the service provider can see what this task is when they go to confirm the request from a customer. In the final high-fi prototype, this data is hardcoded as “repair”, but in the below screen you can see where the information would be displayed.



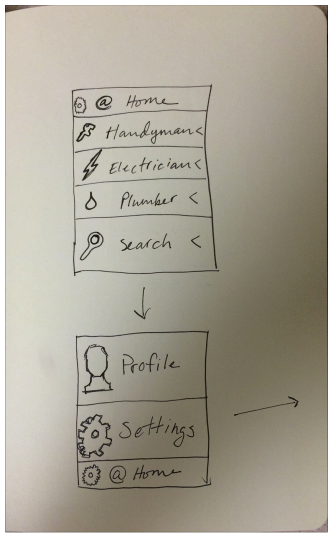
21. [H2-3 User control and freedom] [Severity 3] [Found by: B]

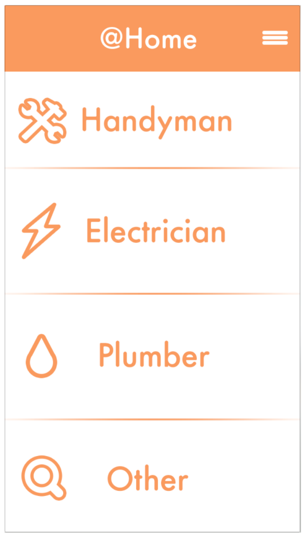
Once a job is accepted, the provider has no option to cancel the job if they arrive and the task is too difficult. This may also manifest as the user creating a poor description of the task at hand; thereby, reaching out for the wrong type of service. Fix: A simple ‘cancel job’ option following confirmation.

This is the final severity 3 issue that was found with @Home. This was another problem that we agreed needed fixing and we did so by adding another screen to both the service provider and customer prototypes after a booking has been confirmed. While the service provider is on his way, both the service provider and customer have the ability to call each other, text each other, or cancel the booking as you can see in the screenshot below.

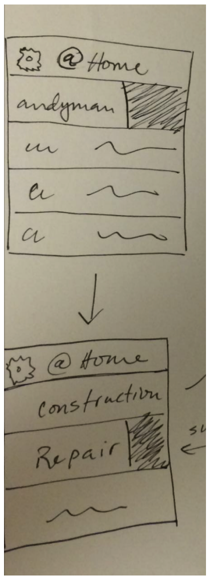


Design Evolution (Gavin)
The overall design theme of @Home didn’t change too much from our initial sketches. Below you can see the first sketch drawn of the UI and workflow of @Home and the final document we used for the high-fi prototype.

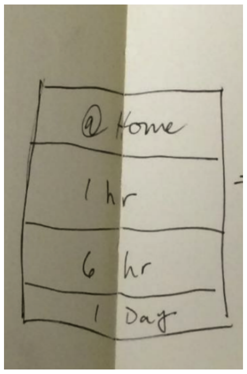




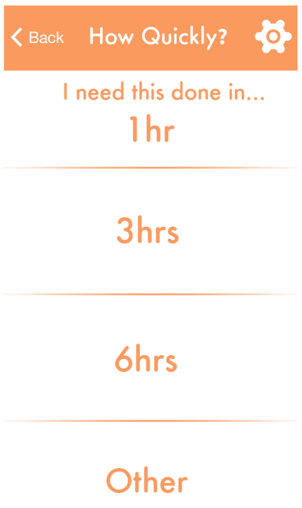
Since our UX was well received by the people we interviewed and showed our low-fi and medium-fi prototypes too mainly only additions and minor changes were needed throughout the process. Initially we wanted to have a swiping function to select the categories off the main screen but ended up scrapping that idea for the more intuitive and easier to use tapping to select an option. You can see a rough sketch of how the swiping would animate below.



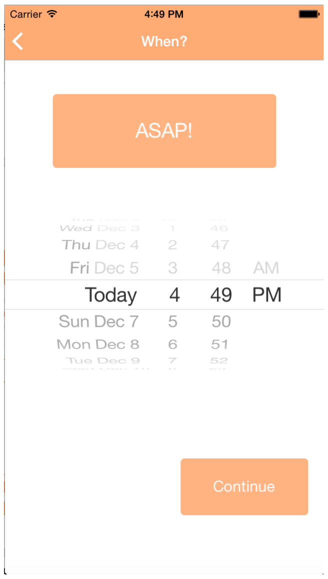
One of the more major design changes was the time screen. We struggled with the design of this screen significantly as we watched users not interact with it properly, or not understand its functionality. The initial time screen can be seen in the sketch below.



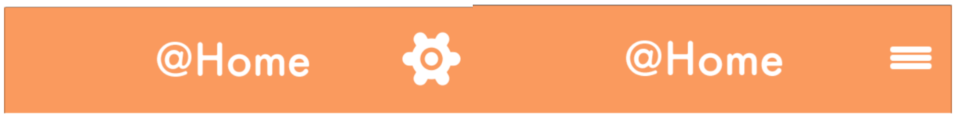
We then decided to add text to attempt to give the users a better understanding of what they were accomplishing on the page as seen below.



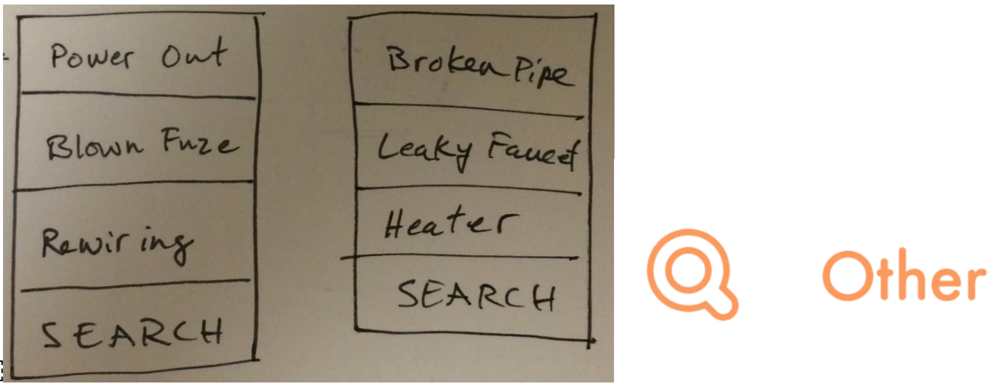
We finally settled on creating this scrolling time wheel and adding an "ASAP" button to attempt to streamline the process and give users complete control over the time. The final time screen can be seen below.



We also made other minor changes such as changing the cog in the top bar that when tapped brought up a profile and settings menu to three bars representing a menu as seen below.

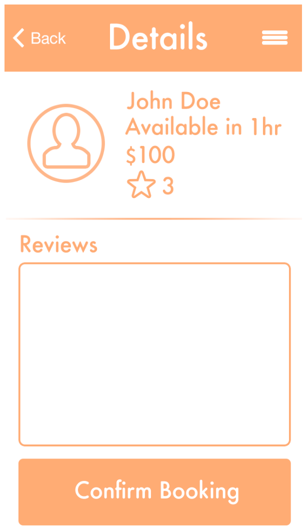


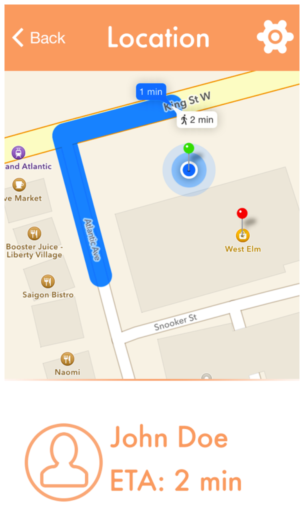
The cog icon we initially had presented some confusion because when it was selected, there was a larger cog icon that then represented a settings menu. The heuristic evaluation that was conducted on @Home suggested we make this change. Another minor change along the way was the scrapping of “search” for an icon and “other.”



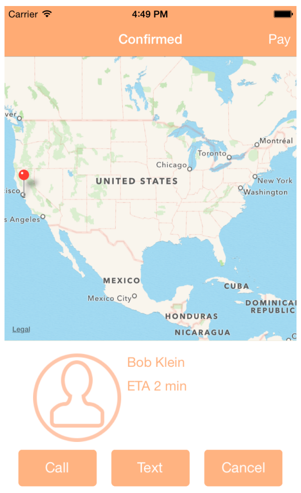
This was a consistency issue that was brought to our attention from interviews and the heuristic evaluation. Not every screen in our low-fi prototype said “search” at the bottom, and search was a tad ambiguous for some screens. We decided to change to “other” (as seen above) for the medium-fi and high-fi prototypes.

The other major design changes occurred after the heuristic evaluation of our medium-fi prototype and were all additions to our design to make it more intuitive to use and prevent errors. We added a detail screen to allow users and service providers to review information before confirming the booking and added functionality to cancel a request from the map screen. Both of these additions can be seen below.





--->



(medium-fi) (high-fi)

Again, most of the major design changes were additions to the design or little tweaks here and there because we were happy with the response we got from initial user testing of the design. The user testing interviews helped greatly with refining the design because we found that we knew how the app was supposed to function and could use it seamlessly because we created it, but the average user was stumbling over some design problems such as the time screen. This caused us to take a step back and examine the little things in the design and try and make it as user friendly and intuitive as possible for the final product.

Prototype Implementation (Laila)

We used Xcode (Version 6.1) to build the prototype, developing exclusively in Objective-C without any use of SWIFT, on a Macbook Pro with OS X Version 10.9.5. It helped since our entire project was an iOS app, which requires the use of Xcode and can be done exclusively using Macs. There was no downside of using these tools, since there is no other option for developing iOS apps.

Since our prototype consists of two apps and they are required to have communication between them, we would have required a server running in order to fully implement our app. For this reason, we had to use Wizard of Oz techniques in quite a few places to make it seem like there was communication between our two apps. If you look at the service providers' app, in the place where you click "Accept task," we take you through a "waiting for customer to accept" screen which waits 5 seconds before moving to the "confirmed" screen. This is a classic Wizard of Oz technique; we don't yet have a server that could communicate to the client that this service provider wants to accept your task, but this waiting screen gives the impression that the client actually has to accept the task before the job is confirmed. Similarly, when a service provider hits "Finish" indicating that the job has been done, he/she is taken through another waiting screen, this one saying "waiting for the customer to pay," which implies that the customer has to pay before rating takes place. Of course, in practice, this is a feature we'd actually like to implement; we'd like for the service provider to wait while the customer pays, as this would happen while the service provider is still at the customer's home. However, we do not yet have this functionality running, so we simply wait a hard-coded 5 seconds before moving on to the next screen.

In the client's app, all the options for tasks (handyman, plumber, etc) and then the work they can do (sink, bathroom, etc) are hardcoded and predetermined by the developer. To make up for this, we added option "Other" that allows the user to customize his/her task. Furthermore, all the details about Service Providers (their names, distances, ratings, etc) are hard-coded as we don't have any service providers who have signed up yet. Similarly, on the service provider's app, all the information about requests is hard-coded since we don't have any requests coming in yet. Furthermore, the details about the task requester (the page you are taken to when you click on a request, which includes information about the person making the request), is all hard-coded data. Lastly, there are some places where there is a hard-coded number of seconds after which the next screen appears. For instance, in the waiting screens we discussed earlier, there is a 5-second delay after the next screen appears. In practice, we would wait until the server told us that the information we were waiting for had arrived and then we would move on. Similarly, we would usually wait for the service provider to hit "finish" and then move on to the payment screen in the users' app. However, we currently have hard-coded that 10 seconds after the user confirms the service provider, he/she is taken to the payment page. In practice, this would be a lot more, since no task would take 10 seconds.

All the backend functionality (information transmitting from user to service provider, data on service providers and clients) is missing right now. Along with this, we'll need to add sign up pages for clients and service providers. We also want to add filters for both apps, so that users clients can specify the price range and distance they are looking for and only get suggested service providers who fit those criteria, and similarly, service providers can specify the distance, time availability and rating of the users they are looking for. This way, both have to look through less data before finding a good fit!