Brad Reyes, Clay Jones, Crystal Tjoa, Dana Murphy
Professor Landay
CS194H
18 March 2017

SkillSwap Final Report

Problem Description

Our focus is on getting help to students who are struggling to learn a new skill. These students may lack the resources for formal teaching, require teaching immediately or at inopportune times, or have trouble without the guidance of face-to-face teaching. It's worth noting that many of these students have their own strengths and skills, which they could potentially teach other struggling students. However, often these students don't see themselves as potential teachers, or implicitly consider being a learner and a teacher somewhat mutually exclusive. Thus, the potential of students teaching each other remains untapped, and many student who are trying to learn are left without help.

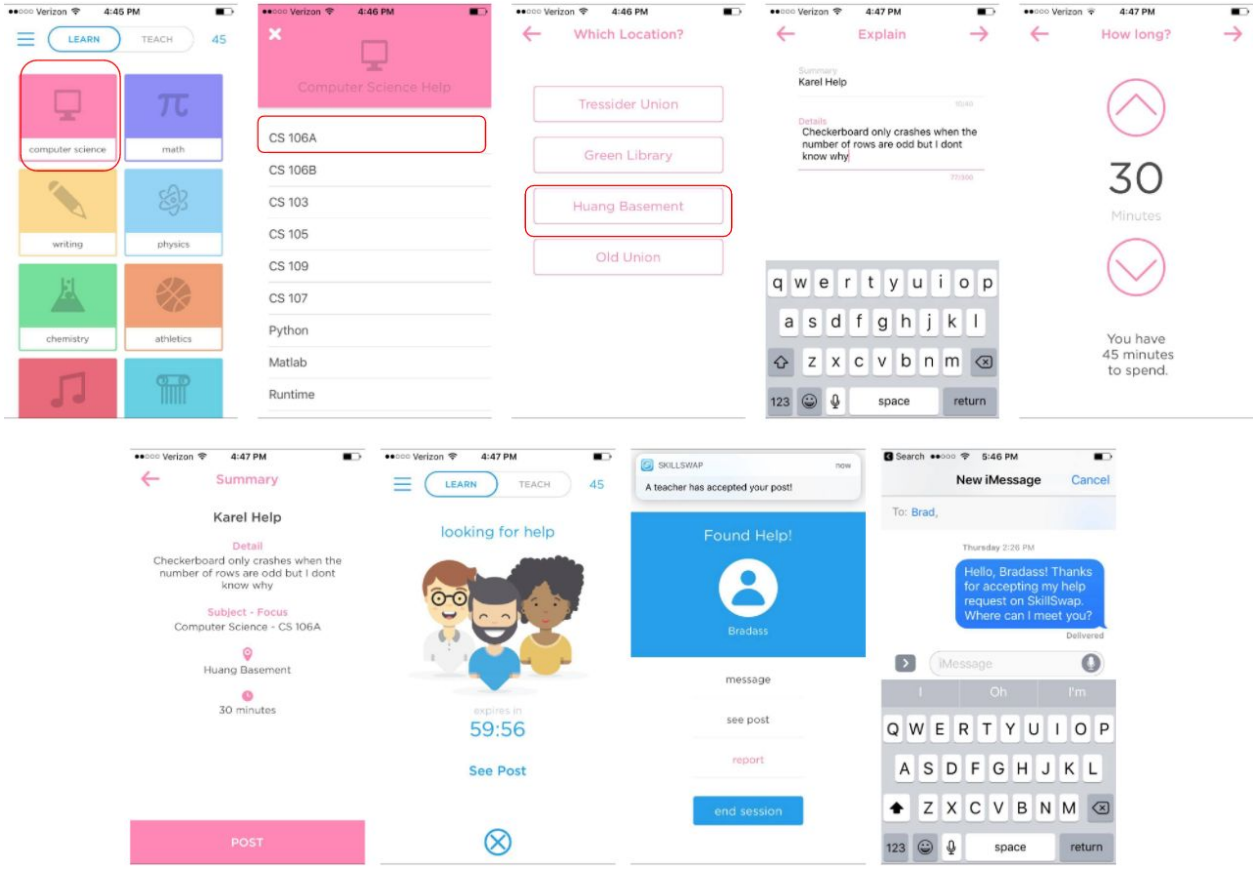
Our Solution

The help/waiting page of our final app

Our solution is SkillSwap, an app designed to bring together students to learn

from and teach each other. Users can post topics they want to learn and get help from other users on the app. SkillSwap uses a bartering system among students; by helping teach others on the app, users earn the ability to get help on topics they want to learn. In this way, all of the users can get instant, face-to-face tutoring on subjects of their choice, while their own knowledge is utilized to help other students on the app. In this way, the resource of other students is being fully tapped, and everyone is able to get help with the skills and topics they want to learn.

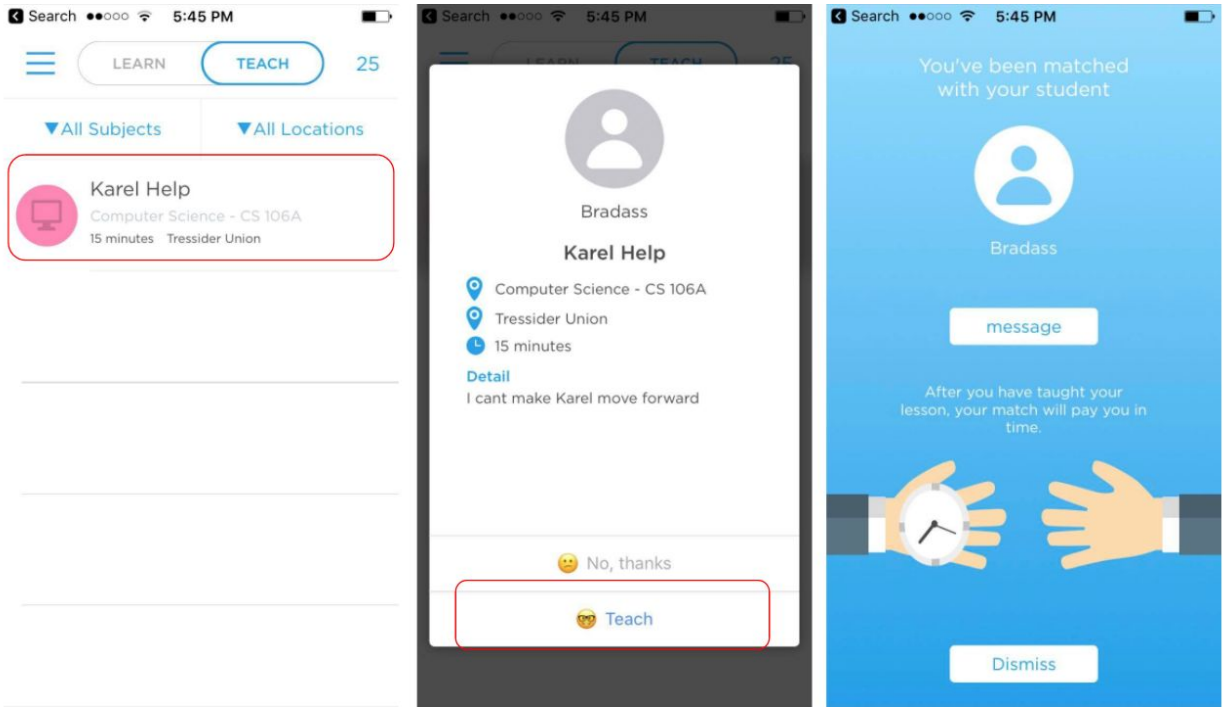
Tasks and Task Flows



Simple Task: Learning a Skill

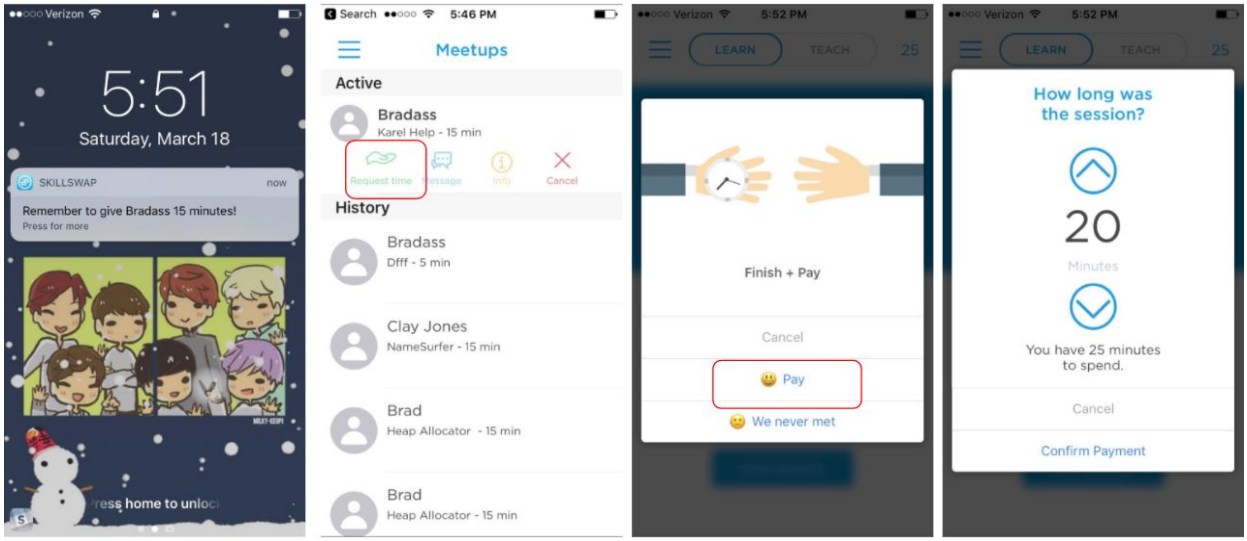
Our simple task is to find a skill and post a teaching request. From the home screen, users select the subject they want help in, and a specific topic under that subject. In each subject, we have an “Other” topic, so users have the ability to customize their request if they don’t find their class or topic already listed. We tried to strike a balance between offering a wide variety of skills while still restricting it to topics that would be popular enough to easily find teachers. After selecting a topic, users fill out various other information, including which public location they want to meet, a

description of their problem, and how long they think the problem will take. They can preview everything before posting it; once it is posted, they will be redirected to a waiting screen until they are matched with a teacher or time expires. If they match with a teacher, they can message the teacher using SMS. We chose to make this our simple task because we think users will use it the most. We tried to make it as simple as possible for users to complete this task, since it will be the main selling point of our app. In fact, one of the reasons why we have so many screens for this task is because we want to walk our user through each step as easily as possible.



Medium Task: Teaching a Skill

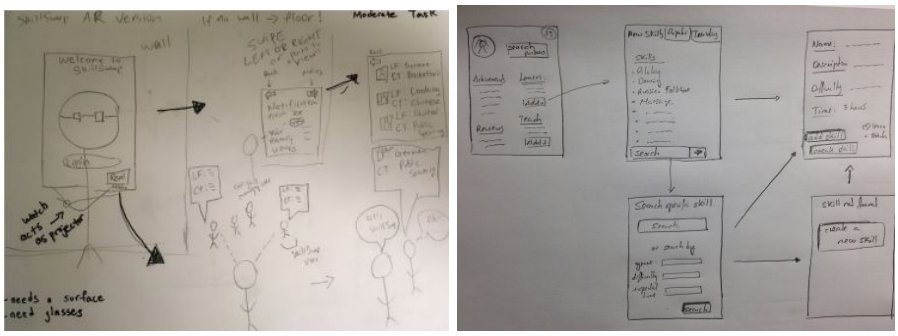
Our medium task is to match with someone and teach a skill. In the Teach page, users can see all the teaching requests. In the case where there are a large amount of requests, users can filter requests by subject and location. After clicking a request, they can view the entire post and decide whether or not to accept the request. After they accept a request, they can message their student. We chose this as our medium task not because it is more difficult to complete from a UI perspective, but because it is harder to teach a skill than to learn a skill. In general, we found that most users are willing to teach if they think they can be helpful, especially if they get a reward from it.



Complex Task: Managing Payments

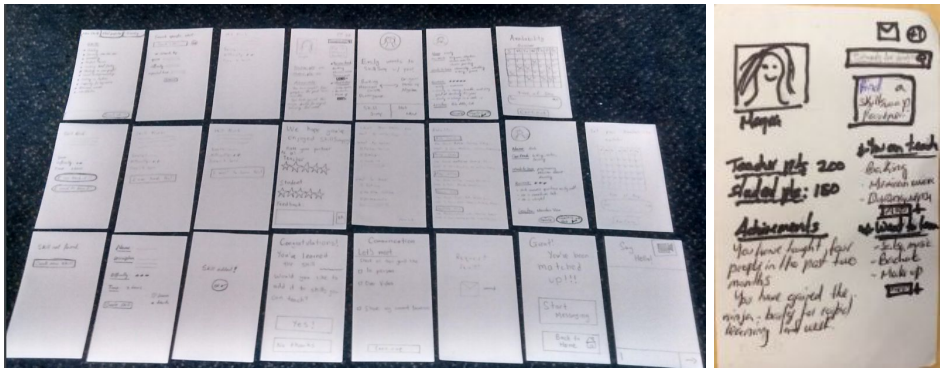
Our complex task is to manage payments. From the Meetups page (accessible from the side menu), users can look at their active meetups- meetups that the user has agreed to teach but have not been marked as completed. If the student has forgotten to pay the user, the “Request time” button will send a notification to the student reminding them to pay the teacher. From the Learn page, students can click “End session,” “Pay,” and then specify the amount of time the student-teacher pair actually spent together. For example, if a student requested 10 minutes for help with a problem, but the meetup actually lasts 20 minutes, the student can pay the teacher overtime. As a design choice, we decided not to let the student pay the teacher less time than originally estimated. This is to prevent the student from gaming the system and underpaying the teacher. We chose this as our complex task because we don’t expect users to use the reminder function very often. The time economy is unique to SkillSwap, and we could not think of a viable solution to automate managing payments. Since we got a lot of feedback both in the Lab Study and the Field Study that users wanted to make sure they got paid fairly, we created the complex task in order to create a semi-flexible payment system that holds students accountable.

Design Evolution



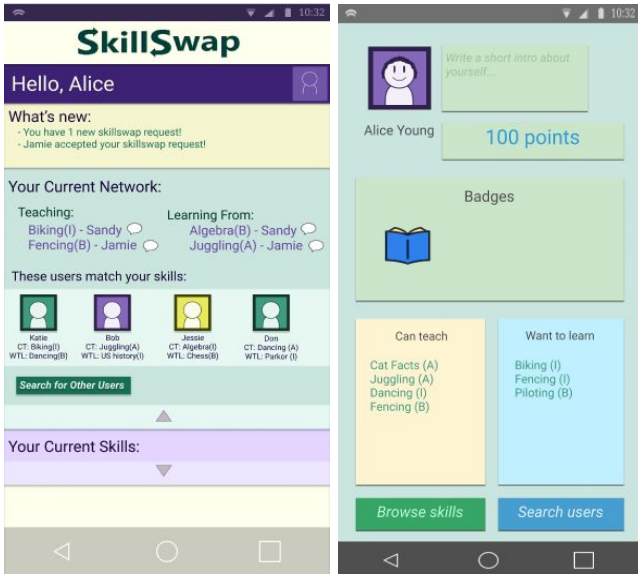
Left: sketches of our AR prototype. Right: pages of our android prototype

Our initial sketches of the app focused on the basic concept of people being able to swap skills with one another. While there were several ideas thrown around, from apple watches to virtual cities of learners, we ultimately narrowed it down to two main concepts. The first was an augmented reality program where the user could look around and see markers above people indicating what skills they could potentially learn and teach. The other was an android app where users could register and search for other users who wanted to swap skills. We ultimately decided on the android app, because our concept requires a large number of users and a phone app is accessible to a larger number of people.



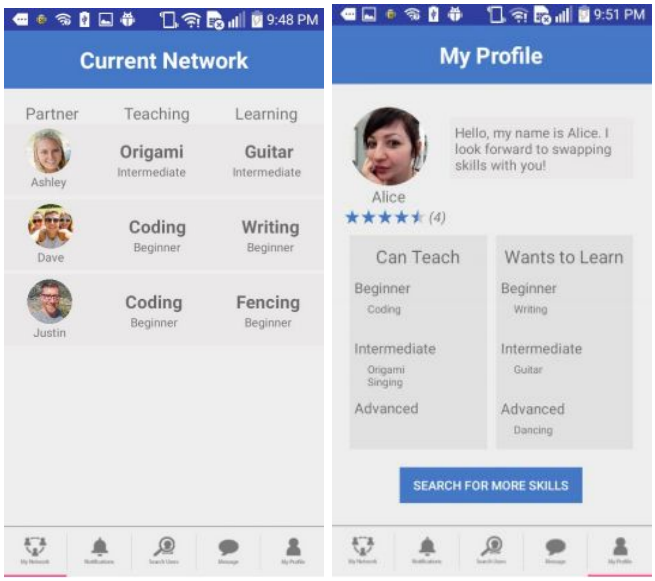
Left: every page of our paper prototype. Right: the profile page of our paper prototype

Our paper prototype helped us narrow down some key details and flow of our app. We ultimately decided on five main functionalities of our app: searching and adding skills to one's profile, sending a SkillSwap request to another user, accepting a SkillSwap request from another user, and arranging to meet with SkillSwap partners.



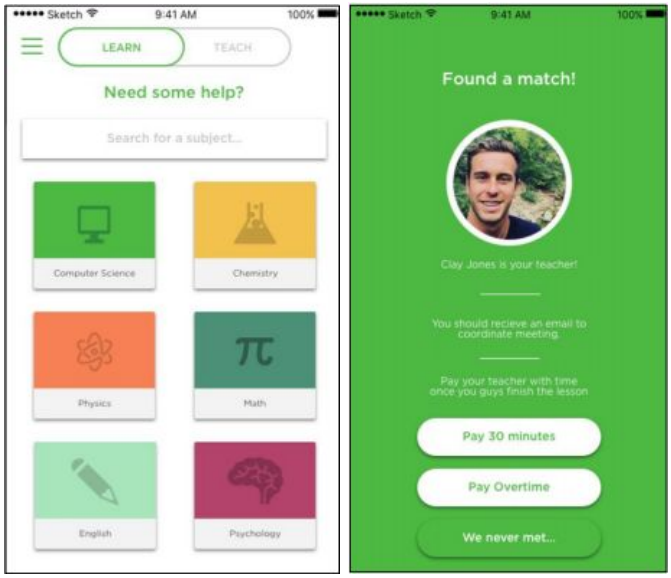
Left: the main page of our med-fi prototype. Right: the profile page of our med-fi prototype

In our medium fidelity prototype, we switched from paper to a simple digital representation of the app. We designed the app's pages in Figma and linked the pages together in Marvel. We modified the flow of the app to include a "home screen", which provided a central location which could link to each of the main functionalities. This let the user complete each given task using far fewer taps. We also had our first shot at colored pages and experimented with different ways to visually display information without resorting to walls of texts. Finally, we tested different forms of personalization on the app, such as app-generated suggestions of SkillSwap partners and skills to add based on previous activity.



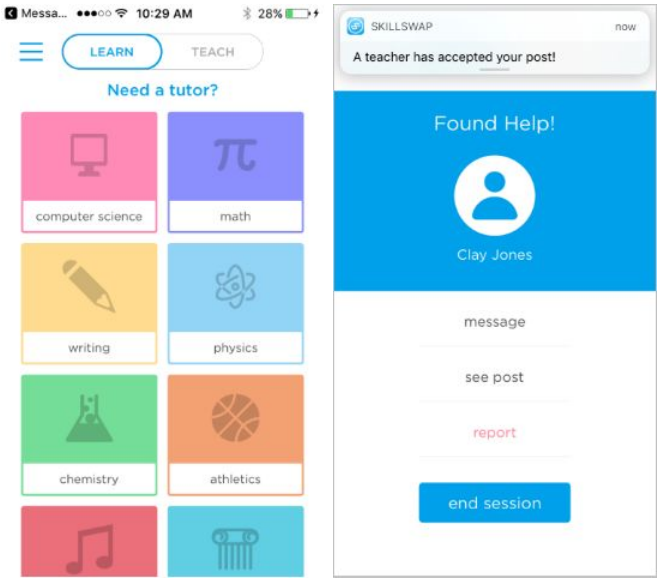
Left: the main page of our hi-fi prototype. Right: the profile page of our hi-fi prototype

Our first hi-fi prototype involved switching from Marvel to Android Studio, and resulted in major visual and structural changes in the app. We completely redid the color scheme to a single color and modified the structure of several pages in tables to make the information easier to parse and require less text to convey ideas. We knew from our feedback that our mainscreen felt like it was trying to cover too much at once. Because of this, we traded the main screen out for a navigation bar at the bottom of each screen, and expanded the first page the users sees to be the network's page (the most important aspect of the former main screen). We also trimmed away several extra features, such as app-generated suggestions and calendar tabs, to focus on the core functionality of the app.



Left: the main page of our hi-fi prototype 2. Right: the match page of our hi-fi prototype 2

Our second hi-fi prototype involved both switching to an iOS platform and pivoting to a new model of giving and receiving help. Rather than swapping with specific users, this version of the app switched to an Uber-like model of sending out a general help request for a topic, which could then be immediately accepted by another user. We also switched to an in-app monetary system for skills. Rather than having to find a single user who matches both what the user wants to learn and what they want to teach, the user can earn credits by teaching and then spend those credits to learn a different skill or topic. For the sake of standardisation among different skills, credits were based on the amount of time a user spends learning/teaching a skill. Finally, we narrowed the focus of our app in terms of our demographic and the skills being exchanged. Rather than allowing any skills for any length of time, we focused on one-off help sessions in a handful of primarily academic categories.



Left: the main page of our hi-fi prototype 3. Right: the match page of our hi-fi prototype 3

Our third high fidelity prototype refined several aspects of the app based on user testing and feedback. We changed the payment system so that users can pay a teacher overtime if they stayed longer than the expected amount of time. We focused on users meeting in public places to avoid security concerns that were brought up in user testing. We added phone number verification for when a user signs up with an account. We added push notifications to alert users to changes and remind them of sessions and payments. Aesthetically we made minor changes to the overall color scheme and format based on feedback from professional UI designers. Finally, we fully implemented several of the front-end and back-end interactions required to make the app functional beyond a lab testing environment.

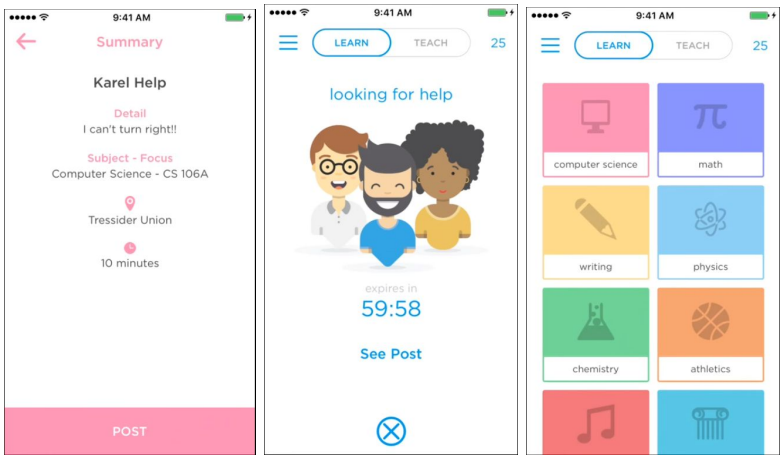
Our most useful evaluation technique, especially during the second quarter of the project, was lab testing and feedback. Each of us came to the project with very different mental models on how the app should be structured and how we should expect users to interact with the app. We would often spend hours debating approaches, both of which sounded plausible and logical yet were in direct conflict with one another. We ultimately resolved the dispute by testing the original hi-fi prototype with a mock-up of an uber-model design. The results demonstrated that users found the revised version of the app more intuitive and useful than the former structure. We continued to use lab testing and user feedback throughout our redesigns of the app. For example, repeated responses over the accuracy of the credit requests led to the inclusion of overtime payments. User concerns surrounding the security of meeting someone at a dorm or private place led to the modification of meeting at public locations for skill swapping. Negative responses around manual credit requests resulted in our automated request

system. Thus, lab testing and feedback helped ground our guesses about the app in actual data and provided a practical guidance for our later revisions of the app.

Final Interface

This section covers the final UI design, features, unimplemented features, tools used, and directions on how to download SkillSwap.

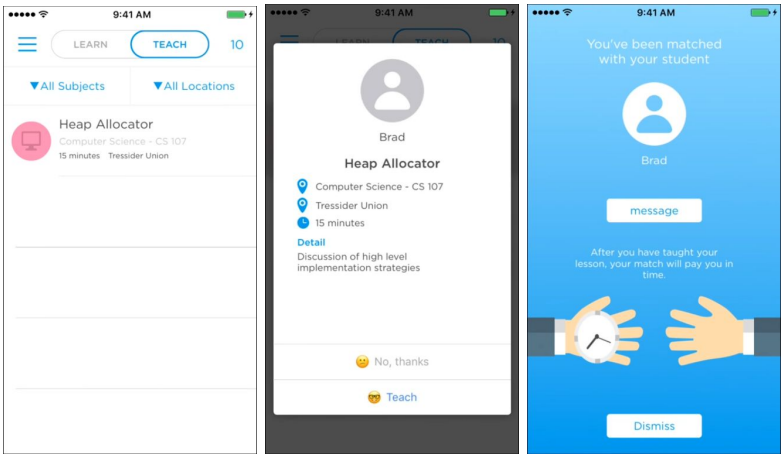
Final UI Design



Figures 1, 2, and 3

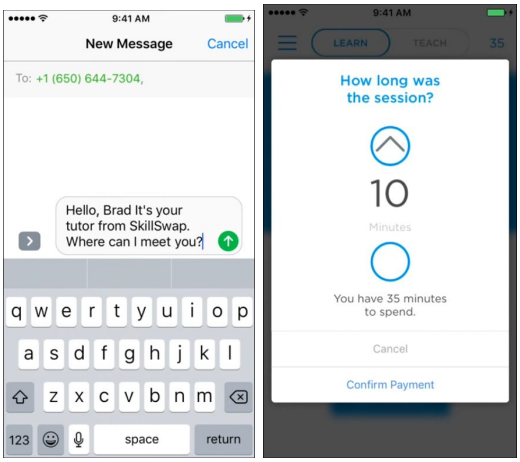
In this section, we'll be talking about the functions and features our app has, and at the same time, be talking about the user interface design that makes this functionality discoverable, useful, and effective.

The final UI design is greatly different than the design and functionality of our initial design at the beginning of the quarter. The main function of the app is to match students/learners with teachers. A learner can post a meetup by adding location details, a summary, details about the skill the user wants to learn, and the minutes the user wants to spend on the session (fig 1). The person who posted the session gets taken to a "Looking for help" page, at which point the user can review the post he or she made, cancel the post, or wait until a teacher accepts the post (fig 2). In terms of the user interface, the home screen has two columns of subjects visualized by colored boxes which the user can choose (fig 3). The user picks a topic from that subject and enters all relevant information right after, which is streamlined through a pipeline of screens that let the user enter all relevant information via text boxes and buttons to get to the last waiting screen.



Figures 4, 5, and 6

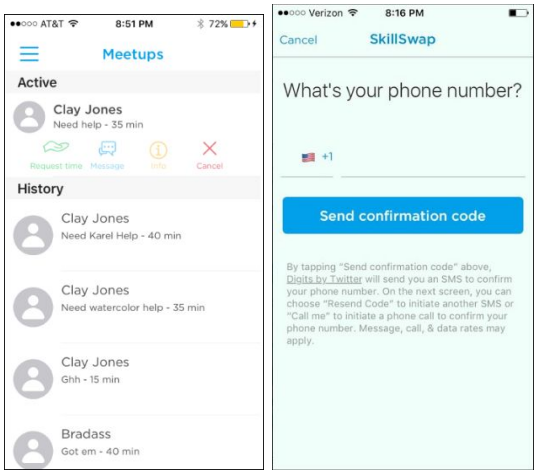
On the other side, a teacher can accept a post by looking at all the meetups that are open. A user can switch easily between the teacher and learner tabs by clicking on the top tabs of the home screen (fig 4). The teacher tab also has an option to filter by location and subject, making it simple to find a meetup that is either near you or a meetup that fits your skillset. A teacher can see all the information of the post set by the learner before accepting it (fig 5), at which point, the teacher is brought to a screen showing that he or she has accepted the post (fig 6), and the student is given a notification when the teacher accepts the post.



Figures 7 and 8

At this point, our app has a feature that allows the learner and teacher to connect via text with the phone numbers associated with their account that they verified. This is where the two parties can connect and sort out any logistical issues with meeting in person (fig 7). Once the session is done, our app allows the learner to pay the teacher. The cost of the session (in minutes) will always at least be the cost that the learner put down, since that was like a promise to the teacher that the session would be around

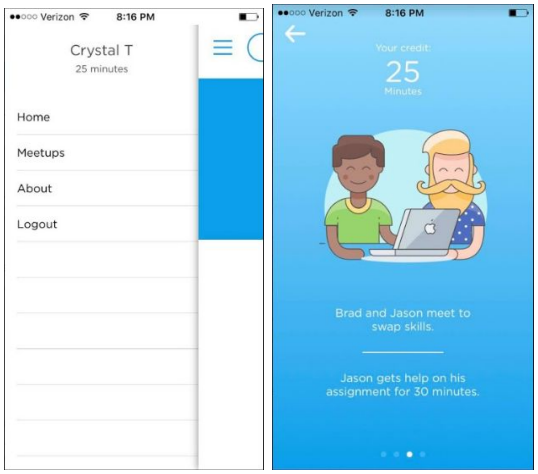
that long. If the learner had an especially good session, or the meetup went overtime, the learned can pay more that the time he or she originally stated (but never under) (fig 8). At this point, the teacher will get a notification that the learned has paid him or her.



Figures 9 and 10

The last main task we wanted to implement was having some sort of reporting system. We did two things to incentivize people to pay their minutes. The first is that we don't allow people to post another meetup before paying their previous meetup. The second thing is that we implemented a "Request Time" button, which a teacher can use to send a notification to the student to remind them to pay for their session (fig 9). The teacher (and learner) can also message each other via iMessage and the phone numbers they provided with the push of a button in order to sort out any discrepancies about payment.

Our three main tasks of learning (posting a meetup), teaching (accepting a meetup), and ensuring payment (sending a reminder) are all covered at this point. In terms of other functionality, we also have a secure login system. With the press of a button, a user can login and register by giving their phone number. Our app sends a verification code to the phone number they entered to ensure that the number is valid (fig 10). This is especially important since our app relies so heavily on push notifications and a bank account of minutes. Without verification, a user can make many accounts and get free minutes. The user does not have to login every time he or she opens the app, only when the user logs out.



Figures 11 and 12

The logout can be found in the sidebar menu by clicking on the hamburger menu. This can be used to logout, switch between the home screen and the meetups screen, and see our about page. The meetup screen is used to see active meetups you have (meetups where you still need to get paid) and see your history of past meetups. It is here that you can also request time from the learner and also message him or her through iMessage.

The last major feature of our app is a tutorial. By clicking on the minutes you have in your account on the home screen, a simple tutorial pops up. The tutorial explains the minute system we have set up in our app, the process of posting a meetup and accepting one, as well as the benefit of gaining minutes so that you can spend them on skills you want to learn.

Features Left Unimplemented:

The following is a list of features that were left unimplemented for various reasons. Most were due to time constraints and the fact that we prioritized other features due to feedback from user tests and class feedback.

- *No way to buy credits/minutes*: Part of our business model was allowing user to use real money to purchase minutes if they were unable or unwilling to earn those minutes through teaching. This would help address the inevitable imbalance of students to teachers as well as provide a form of monetization for the app. Unfortunately doing so would have required processing user's credit card information, which creates a slew of additional security issues and would have hurt our chances of being accepted into the app store. This is obviously something we would eventually want to implement, but we decided to get our app

rolling with the non-monetized version for now.

- *Messaging system is not in-app*: A complaint in our very first user test was that users (specifically women) would not feel comfortable meeting with a complete stranger. To alleviate this, we made a feature that encourages people to meet in public locations. However, sharing phone numbers with strangers does not keep this spirit. We felt that in-app messaging was not a major focus of our app, however, and that this would be at the bottom of our priority list since the more important features to fix and implement had to do with swapping skills and user flow for doing so. In-app messaging would also take a not inconsequential amount of time, and the pay-off for having it completed was not enough to warrant making it a priority.

- *No Rating/Karma System*: This is also something that we wanted to implement but didn't have time to, but this was next on our priority list. It would basically be a rating system and hopefully incentivize users to be better teachers and learners (like Yelp super-users) and create a sense of reputation within the SkillSwap community. Only reason we didn't do this is because we didn't have time, we instead prioritized a tutorial, app design cleanup, tightening up the server/backend, and fixing noticeable usability bugs so that the features we had then all worked well

- *User has to enable push notifications*: This is needed for a lot of the core functionality of the app. The app becomes much less effective if the user does not enable push notifications

Along with not making these features, there are also some small bugs that we did not get to as they were low on our priority list. Every bug we've encountered can be fixed by closing the app and re-entering. These are all relatively small bugs since they don't totally break the app. Here is a list of most (if not all) of them:

- *No checking for going under your minute count*: You can have negative minutes. This doesn't happen too often, since we have checks in the front end and the app won't allow you to pay with minutes you don't have. However, the server doesn't have any checks for this. You can cause this bug by paying a teacher overtime when you do not have that many minutes in your bank.

- *You can teach yourself*: This was useful for debugging and testing, and it doesn't break anything since you will have the same number of minutes afterwards (you pay yourself in the same transaction in the database), so this wasn't a priority

- *Sending a reminder gives wrong error message*: The functionality completely works as it actually sends the notification, but the app says it did not go through (even though it did)

- *Will sometimes duplicate meetups when posted:* We're not too sure why, but the bug was discovered after posting to the app store, so we didn't have time to fix it

- *Time expiration doesn't mean anything*: We say that the session expires, but nothing actually happens when time ends. The meetup stays there. This was not a top priority since it doesn't break anything.

- *Accepted screen sometimes doesn't appear*: This bug happens rarely. On rare occasions, the learner's screen wont update when a teacher accepts a post. You just have to exit the app and re-enter to get on the screen you should be on.

For our last iteration in the app store, we did not have to use any wizard of oz techniques. The server independently handles all push notifications and has a durable database that keeps all information about users and meetups, and also handles all requests by itself. We did fake interactions in earlier iterations, and implemented push notifications at the very end, but in terms of implementation and functionality, we were able to not use wizard of oz techniques to have a functioning app.

Tools We Used

For the frontend of our project we primarily used Swift. We also used Digits by Twitter Authentication to verify user's phone numbers, Apple push notifications to send push notifications to the user, and iMessage integration to allow users to send text messages to each other. For the backend of the project, we used NodeJS/Express hosted through Heroku, and made the server a RESTful API. We used a Postgres Database that came free with the server we hosted on Heroku. We shared code through Github and messaged each other directly through Groupme. For designing the app, we primarily used Swift and Figma, as well as occasionally Photoshop, Gimp, and Paint Shop Pro for additional visual elements. We also got a lot of our images and icons from freebiesupply.com (credit to Vincent Le Moign and Netguru). Our concept video was made in iMovie. For user testing, we used lookback and other screen recorders to

analyze user test data. We generally found all of our tools helpful and all helped us create our final product, but Sketch, icons at freebiesupply, Swift, and Heroku were all the most helpful for creating SkillSwap.

Download Information

The iOS app is available on the App Store. You can download it by clicking the link below or by searching “SkillSwap Learn and Teach” on the App Store.
https://itunes.apple.com/us/app/skillswap-learn-teach/id1214665162?mt=8

Making It Real

Our team consists of three Stanford undergraduate seniors and one Stanford graduate student. We’ve all experienced difficulties and frustrations getting help with academic subjects in our crucial time of need. We also noticed that it is often difficult to expand your network of support as the years go on. There was no easy way to reach out to peers for help and collaboration. We are located in a university that has both a challenging academic environment and a community of students that are more than willing to help their peers when presented with the opportunity. Our team can continue to test our concept and refine our design before trying to expand it to other universities.

In order to turn this app into a business, we would have to find ways to increase the numbers. We would post flyers on Stanford campus, especially in places with a high volume of people waiting for TAs, like in the LAIR or in office hours. We would also promote the app through e-mails, social media, and tabling. Finally, we could host SkillSwap Meets where people would gather in a large common area like Tressider to exchange skills. We want to ensure that users have a great first-time experience with the app: in order to do this, we want to ensure that users successfully match with a teacher and have a great in-person experience. In order to do this, we could initially pay exceptional tutors to be SkillSwap teachers. Once the user base becomes self-sustainable, then we could stop paying these teachers.

With the app in it’s current state, the subjects are tailored specifically to Stanford students, so our current market size is around 16,000 users. However, if we are successful on Stanford campus (hitting 20% of students on campus), we would consider scaling up to all universities, increasing our market size to 20 million in the USA alone. The reason why we are focusing on university students for this app is because our app relies on people being able to meet each other easily and instantly, which is most reasonably achievable in high density areas, such as college campuses.

We have two different strategies for making money. The first is allowing users to purchase additional minutes when they do not have the time or energy to earn minutes by teaching. Secondly, there are certain screens, like the waiting page, that would be great real estate for advertisements. We would use targeted ads based on the subject the user is trying to learn (for example, selling iClickers for science topics).

Long term, this app could have a huge impact on paid tutors and teaching assistants. If users could access help for free, than they might feel less of a need to go to office hours or pay for a tutor. The amount of impact we have could depend on how many minutes we allow in the economy. For example, if we expect users to have an average of 30 minutes in their time bank at any given time, than we could expect the app to be used for shorter meetings, perhaps only focusing on one problem. However, if we expect users to have about 2 hours in their time bank on average, then we might turn into a full-fledged crowdsourced tutoring service. Like other crowdsourcing apps like Uber and Lyft, we are leveraging connections within the community to provide a cheaper and more convenient alternative to tutoring.

Summary

What separates our app from other learning platforms is challenging the false dichotomy of learners and teachers. We utilize the strengths and teaching potential of each person on our app and help create a network of students providing immediate and face-to-face guidance with each other. Right now we are focused on creating a community within Stanford University- however, as our app gains traction, we could potentially expand it to campuses across the country. Our ultimate goal is to help as many students as possible, and foster nationwide community of learners and teachers.