SpeakEasy

Language learning on the go with contextual suggestions and real-time feedback

Gabriela Groth (Manager, Design), Eric Chew (Development), Tommy Truong (Documentation), Carlos Araujo (User testing)

Problem and Solution Overview

Language students have a hard time learning a new language. They are often removed from real world situations while learning, and once they are on-the-ground trying to communicate, they lose the support they had in the classroom. SpeakEasy strives to overcome these limitations by creating a context-sensitive, on-the-go language learning environment. With instant contextual phrases suggestions, dynamic speaking support, and helpful growth tracking, SpeakEasy allows users to learn and master a language even while performing other activities.

Figure 1. Overview of SpeakEasy features.

Tasks

In order to enable users to learn a new language easily and intuitively while having contextual support, we redefined our 3 representative tasks to more accurately guide our UI design:

1) Receive and use phrase suggestions in real-world conversations (complex)
2) Understand and reproduce the correct pronunciation of words (medium)
3) Integrate language learning into daily life (simple)

Our three tasks cover all important aspects of learning to speak a new language. They cover pronouncing words correctly, understanding what was said to enable the correct response, providing suggestion to enable smooth conversation and giving feedback so that a student can learn and practice a new language without disrupting his or her daily routine.

Task 1: Receive and use phrase suggestions in real-world conversations (complex)

We chose this task because one of the major problems students of a new language have is figuring out how to express themselves in a new language. By giving context sensitive suggestions and providing translations of these phrases, SpeakEasy can help users converse in a more natural way.

Users who are newer to a language can look through specific phrase suggestions in their native language and get the translation of the phrase they would like to use. Users who are more familiar with the language can quickly scan through suggested phrases in the new language if they have forgotten exactly how to phrase a response. Further, if a user is not sure how to pronounce the suggested phrase, SpeakEasy can give them the correct pronunciation.

<u>Storyboard:</u>

The user is approached by a waiter and lets the application sit idle for three seconds so it goes into translation mode.

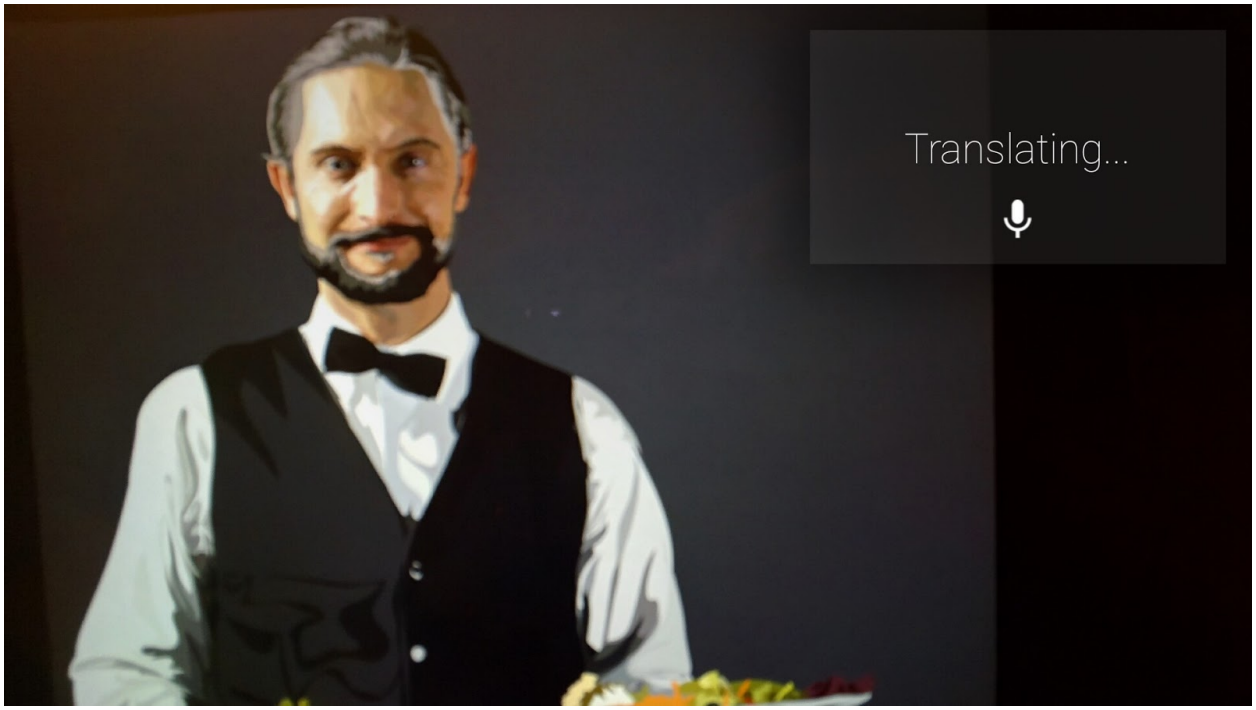


Figure 2. Translation Mode

The waiter then asks what he or she would like to eat, saying “Que quieres comer?” which is translated by the application.

Figure 3. Translation Mode (With input translated)

The user needs help answering back, so they tap the device for phrase suggestions.

Figure 4. Tapping google glass.

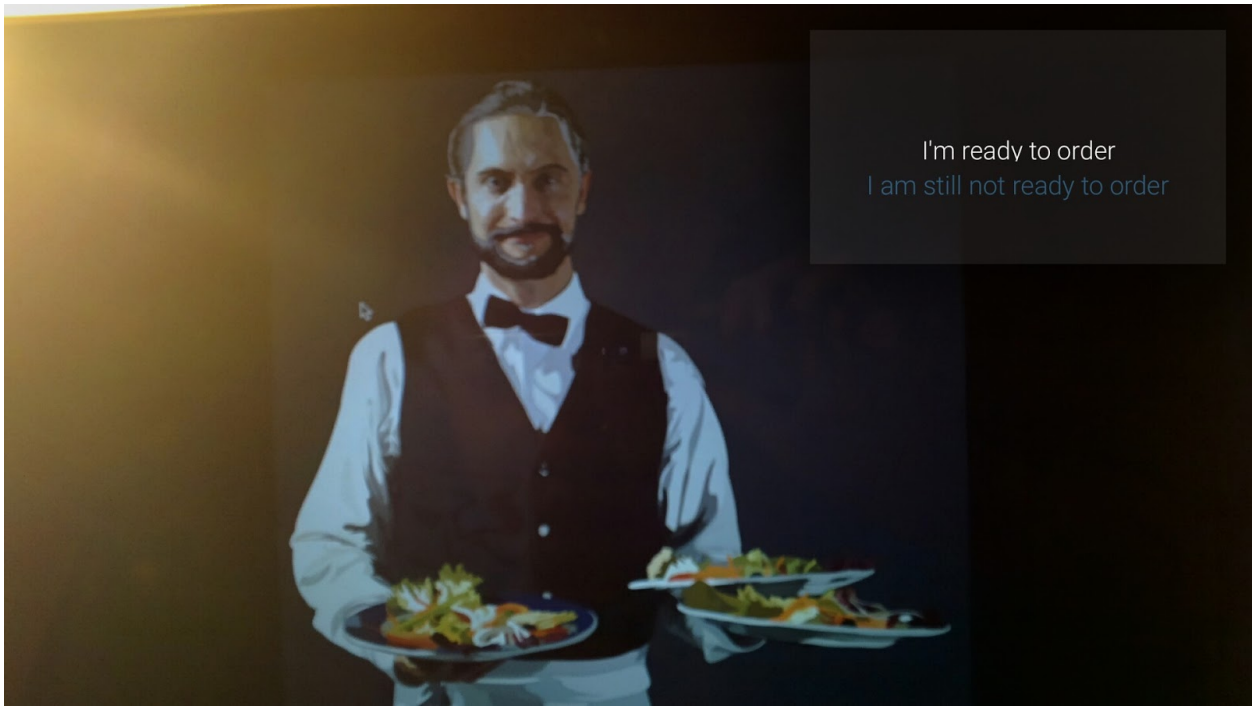


Figure 5. Phrases Mode.

They then swipe forward and backward to cycle through the phrases until they get to what they want.

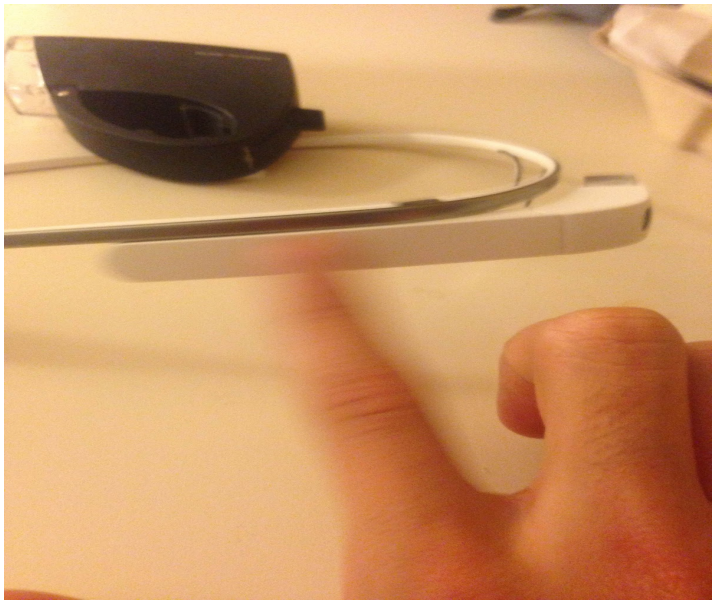


Figure 6. Swipe gesture on google glass.

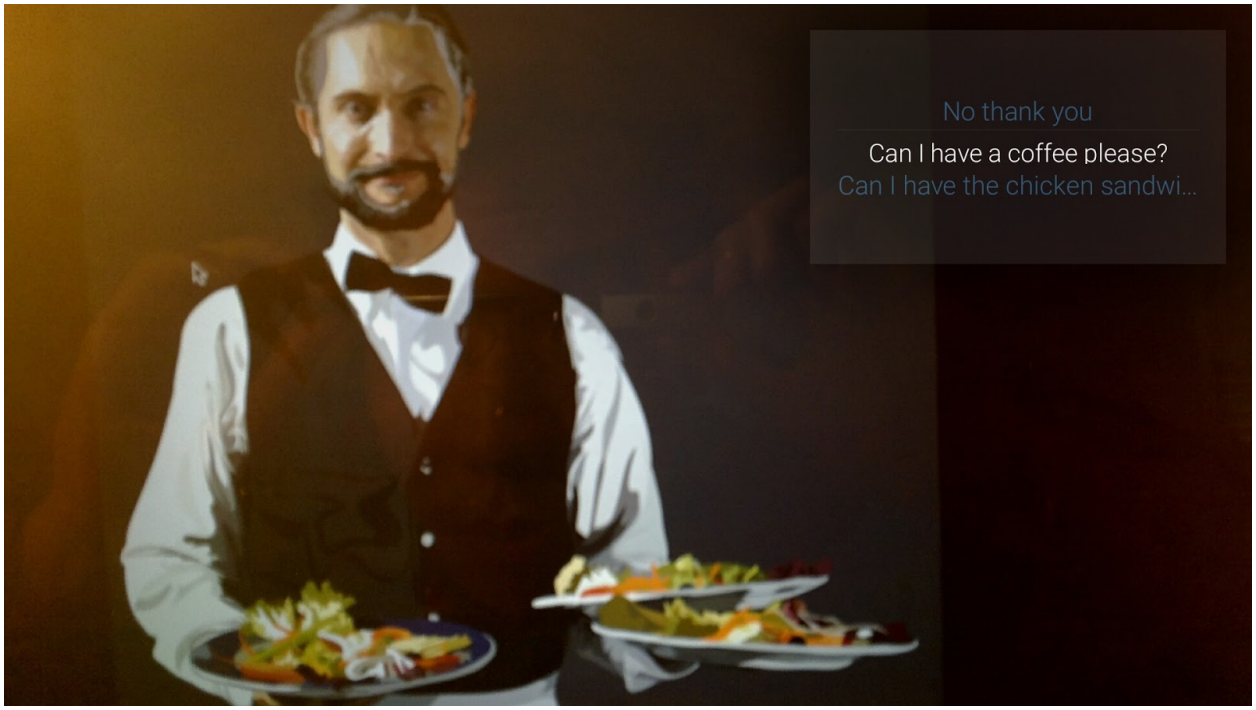


Figure 7. Phrases Mode


Figure 8. Tap Gesture

They can tap anytime to see the phrases suggestion in the language they are attempting to speak.

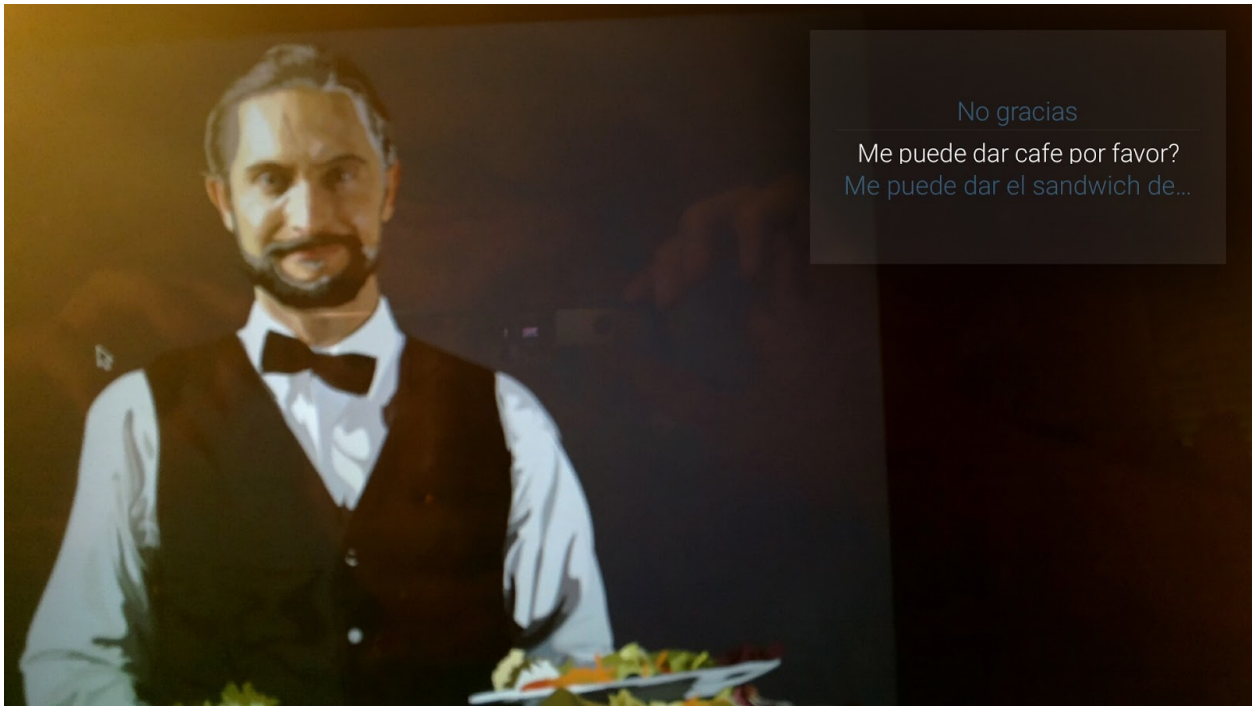


Figure 9. Phrases Mode

Task 2: Understand and reproduce the correct pronunciation of words (medium)

We chose this task because many intermediate language learners may be able to recognize and recall words in other languages but have trouble correctly pronouncing words. In classroom settings, users can get immediate help pronouncing words. With SpeakEasy, we provide this functionality by allowing users to hear the correct pronunciation of suggested phrases. So if a user is unclear about how to pronounce one of the phrases our glass application suggests, they can tap and hear how a native speaker would pronounce the word.

Storyboard

The user has chosen a phrase, but does not know what it means or how to pronounce it.

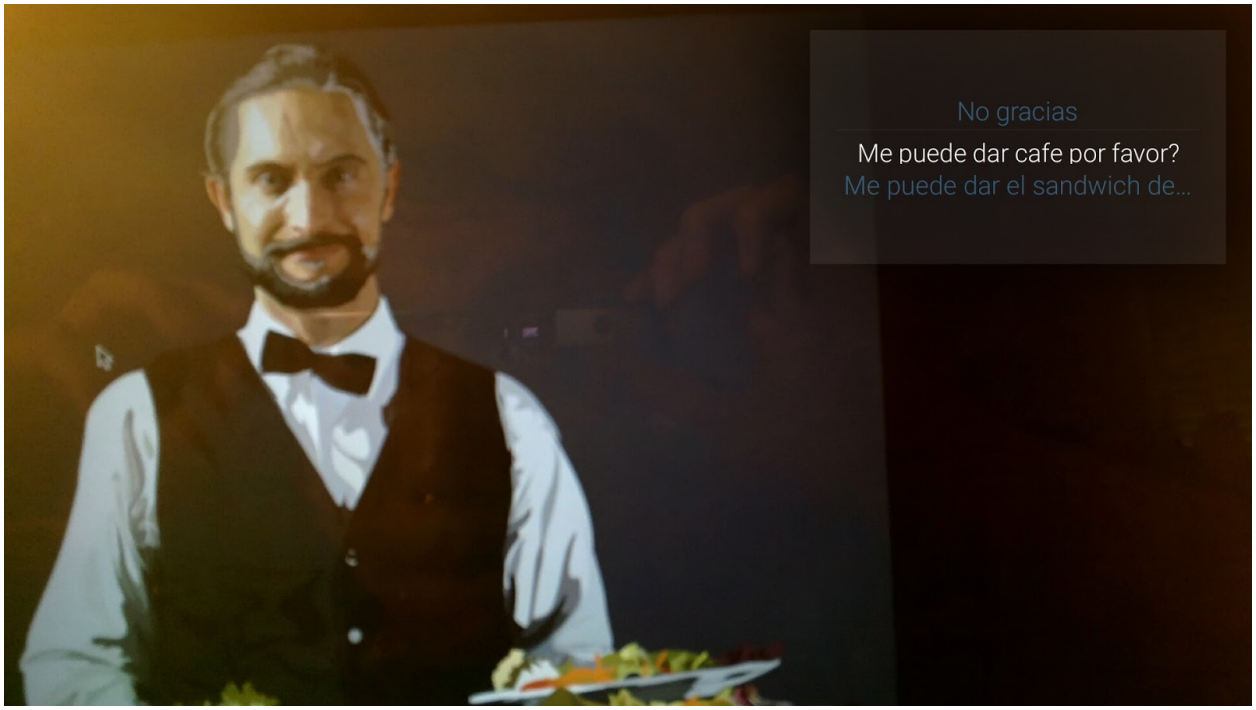


Figure 10. Phrases Mode

The user can tap to see the translation of the phrase suggestions.


Figure 11. Tap Gesture

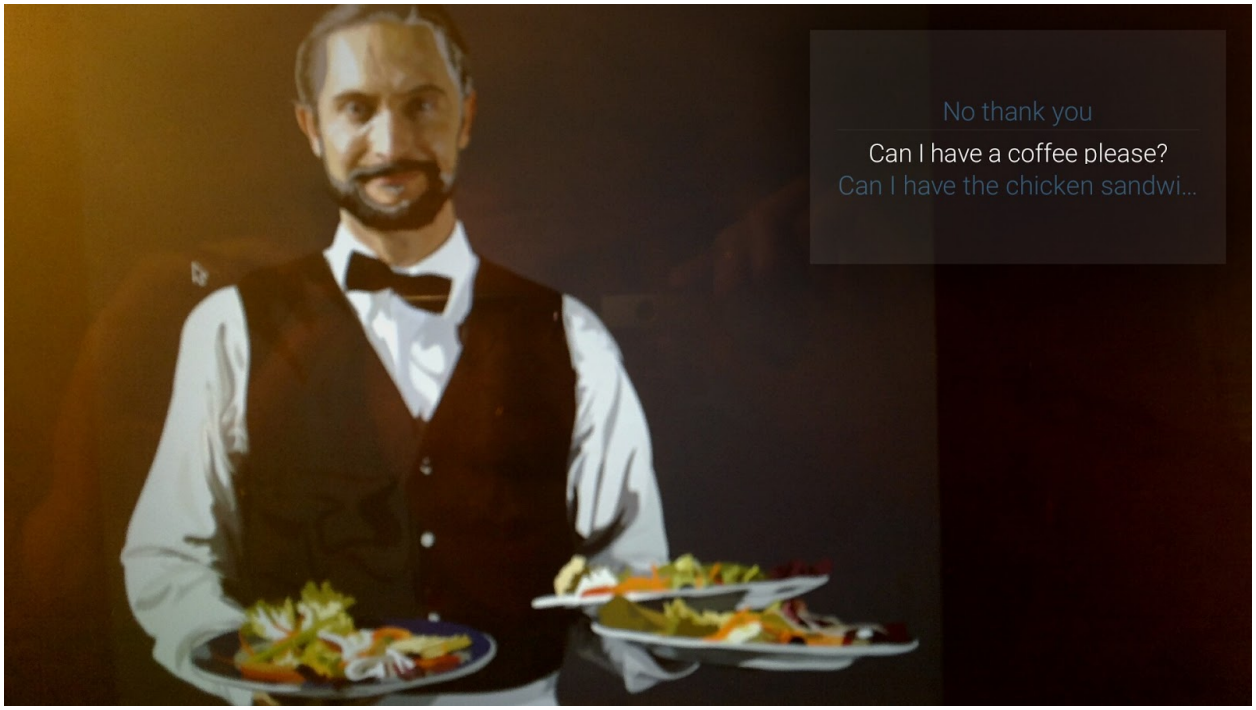


Figure 12. Phrases Mode

The user can also two finger tap the device to hear the pronunciation of the phrase.


Figure 13. Two Finger Tap

Audio of “Me puede dar cafe por favor” plays from speaker

Figure 14. Google Glass Speaker

Task 3: Integrate language learning into daily life (simple)

We chose this task because students of a new language often do most of their learning in classroom or isolated settings. This makes learning a new language time consuming and hard to fit into a schedule. By integrating learning into everyday life, a student is more likely to improve their language skills. SpeakEasy gives instant feedback to users about their pronunciation and speaking skills throughout the day.

Storyboard:

The user goes to the mobile app, which starts off on this home page. At the home page, they will see some summary statistics of how their language skills have been changing over time.

Figure 15. Progress screen

If the user wants to see what words they mispronounced or did not know throughout the day, they can go to the “Missed Words” tab. They can click on the blue speaker icon to hear the correct pronunciation of the word. Below the missed word is listed the translation of that word.

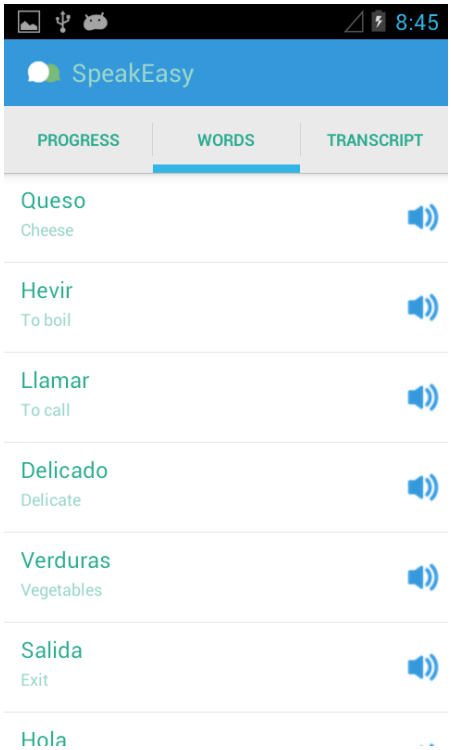


Figure 16. Missed Words screen

Users can go to the “Transcripts” tab which will show them a list of conversations they recorded throughout the day. They can then select one and see a transcript of that conversation.

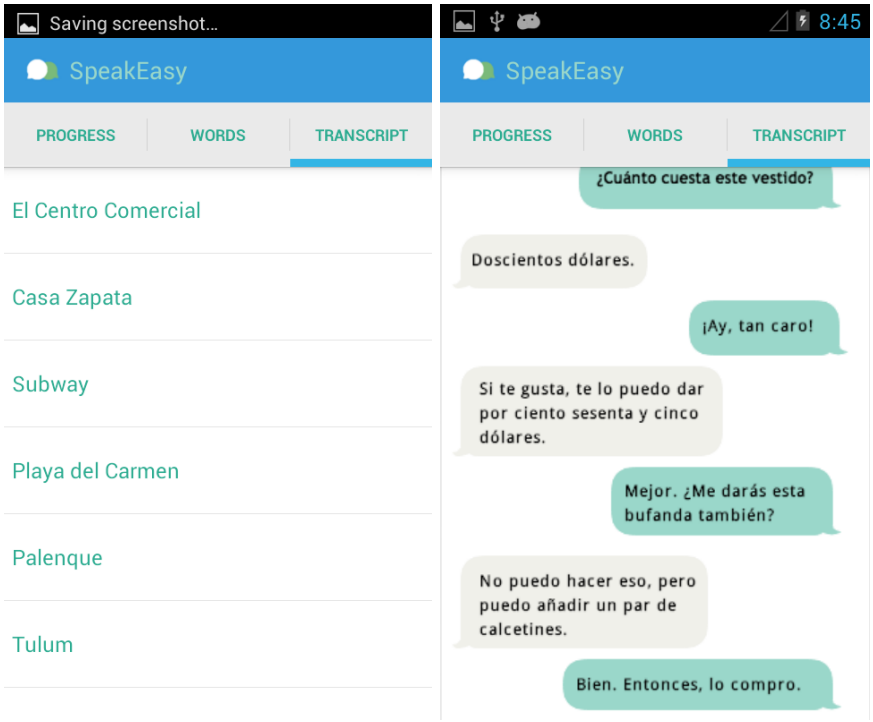


Figure 17. Transcripts screens

Major Usability Problems Addressed

1. [H2-3. User control and freedom] [Severity 4]
Violation: In the medium-fi prototype for the Glass portion, users can't easily navigate back to previous screens and could potentially end up in a loop that they can't break out of.

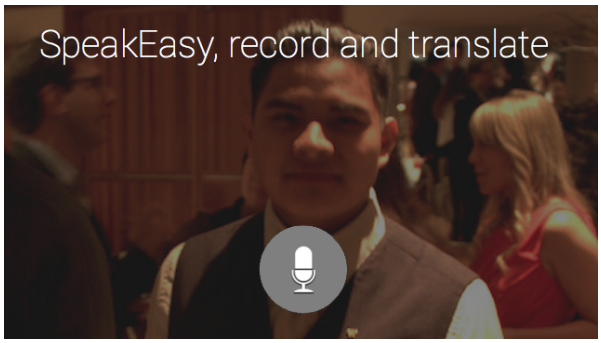
Fix: In our hi-fi prototype, we simplify the number of screens they can navigate through to two, a Phrases Mode and a Translation Mode. Translation Mode automatically starts when the user has been idle after a certain amount of time, and users can easily go back to Phases Mode by using any gesture on the touchpad (indicating to Glass that they are no longer idle).

2. [H2-4. Consistency and standards][Severity 3]
Violation: In the medium-fi prototype for the Glass portion, there was a microphone icon on the screen that the user might try to press.

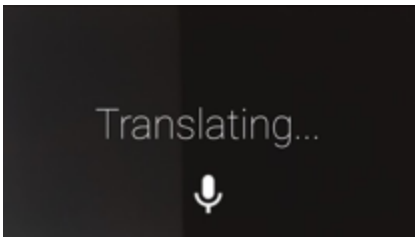
Fix: We changed the microphone icon that appears in the user's view on the Glass screen to the one natively used by Google Glass during voice recognition so users will not reasonably think that they can press it and will understand that the glass is expecting some voice input. We also made the Glass screens (called "cards") match the platform standards such as having a black background so that they would be familiar to users who have experience with

Glass, and made it more clear that the application was doing something by adding "..." to the end of translating to show that it was in the process of translation.

Before:



After:



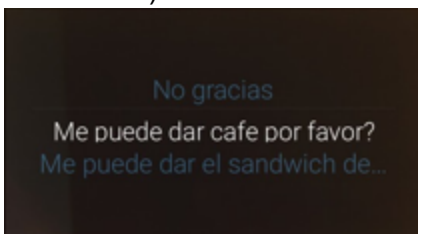
Figures 18 & 19.

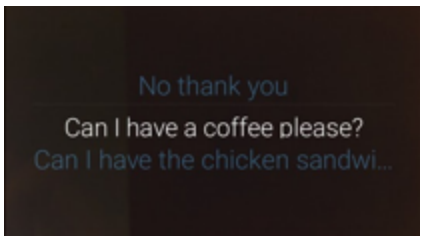
3. [H2-7. Flexibility and efficiency of use][Severity 3]
Violation: Scrolling through phrases in the Glass medium-fi prototype could be clunky because only one phrase appears on screen at once.

Fix: In the hi-fi prototype, several phrases are displayed at once, but one phrase is highlighted. This allows the user to scan and navigate multiple phrases easily but still focus on one phrase.

Before:

After (more than one phrase at a time, tap to translate):





Figures 20, 21, & 22.

4. [H2-2 Match between system and the real world][Severity 3]

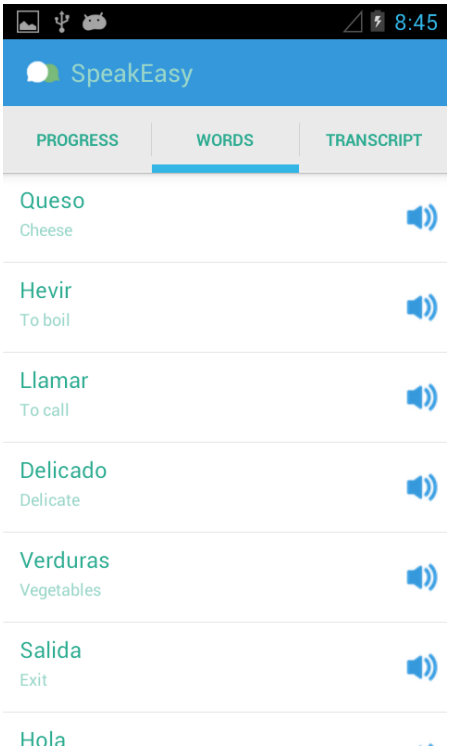
Violation: The speaker icons in the mobile medium-fi prototype did not look clickable and were not “3-D”. The evaluator provided an example of what the icon might look like.

Figure 23.

Fix: We added speaker icons colored with an “action” color to make it clear that they are clickable. We did not make the speaker icon 3-D because it does not match the overall look of our app, and furthermore, the example the evaluator provided was clearly not 3-D either.

Before:

After:



Figures 24 & 25.

5. [H2-6 Recognition rather than recall][Severity 3]
Violation: In the Glass portion, the user must remember when it is appropriate to tap. Though the tutorial walks them through it, it might be too hard to remember.

Fix: One of Glass’s main limitations is space - we cannot attempt to cram too much information into one card. The evaluator thought that we should add instructions on what

gestures to use on each screen, but this would make each card too cluttered. Instead, we attempted to add visual cues. For example, rather than showing just a single phrase, we made our phrases look like list. Since lists are often swept through in mobile applications, we thought that the same intuition would carry over and that users would use the analogous swipe gesture for glass. For features that we could not assign visual cues to, we tried to make the most evident gestures map to features that the user would want to use the most. For example, tapping, which is probably the most used gesture on Glass, was mapped to translating the phrases between native/foreign languages, as users had heavily used this translation feature in previous tests.

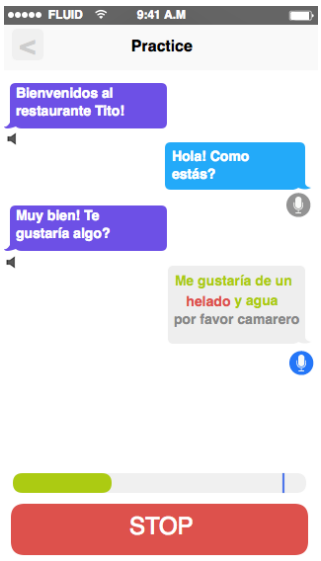
6. [H2-3 User Control and Freedom][Severity 3]
Violation: In the Glass portion, the user might accidentally tap to pronounce, and it might not be clear on how to cancel pronunciation of words and phrases which would make conversations awkward if the Glass keeps pronouncing things out loud.

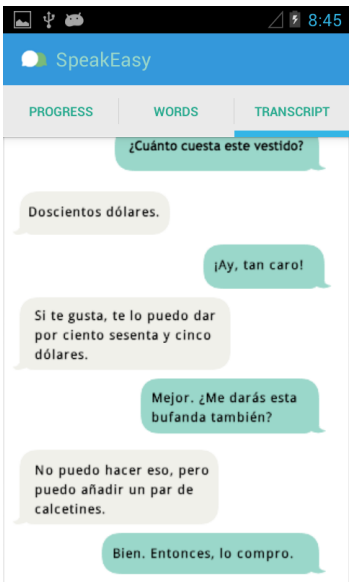
Fix: We did not think this was too much of an issue in the hi-fi prototype on Glass because Glass's speaker uses "Bone Conduction Transduction" that allows only the wearer to hear what Glass says. The phrases provided are also relatively brief and concise, so it will not interfere with the conversation. We did change the gesture to start pronunciation to be a two-finger tap, which is less often used and therefore harder to accidentally do.

7. [H2-3 User Control and Freedom][Severity 3]
Violation: In the medium-fi prototype, words were colored differently and it wasn't clear what these words meant.

Fix: We originally wanted to use colors to signify to users how well they pronounced a word in order to emphasize what words they should focus on improving. We realized that the colors might be confusing to users, so we decided to scrap that feature and just display the words consistently to avoid confusion.

Before: After:



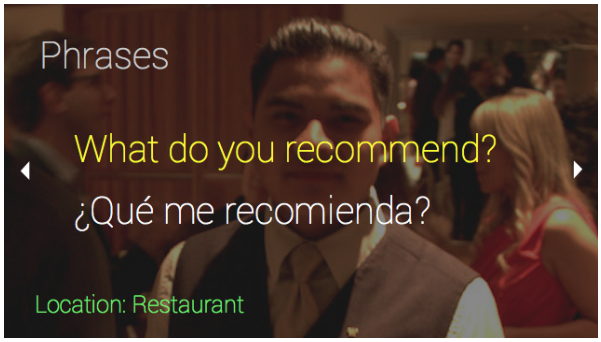


Figures 26 & 27.

8. [H2-4 Consistency and Standards[Severity 3]
Violation: The two portions of the app did not look similar to one another and seem like they are from different apps.

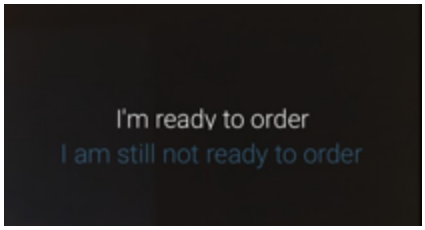
Fix: We developed consistent app icons to use across both platforms to both emphasize our brand as well as make it clearer that the two portions are part of the same app. We also developed a style guide/color scheme to color our website and prototypes. The mobile and glass portions, by necessity, will still look somewhat different because they are different platforms suited for different uses. The Glass app must have colors that can stand out when overlaid transparently over the user's surroundings and must be stripped down and relatively bare to avoid cluttering the user's vision. The mobile app has no such limitations.

Before: (inconsistent look of glass and mobile aspects)
Glass:



After: (use of style guide, color scheme, more consistent icons)

Glass:



Mobile:

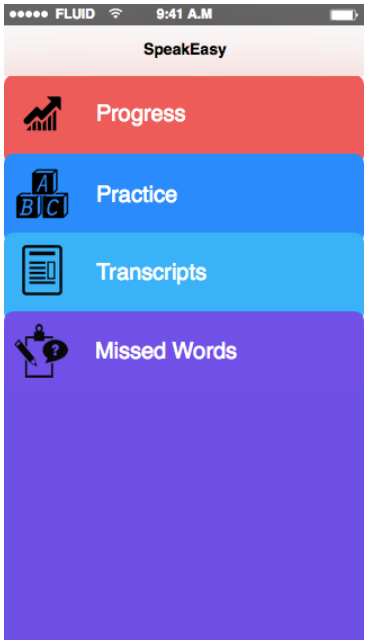


Figure 28.

Mobile:

Figure 29.

Icons:

Figures 30 & 31.

Color guide:

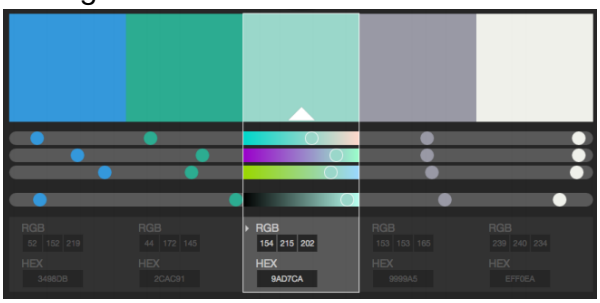


Figure 32.

9. [H2-1 Visibility of System Status][Severity 3]
Violation: There should be an indication that the mobile and Glass apps are linked together.

Fix: We did not yet link the two portions of the app in our hi-fi prototype. If we were to work on this app further and link the two, there would be clear indications that the two are linked and allow the users to pull information from the Glass app into the mobile app and vice versa. In general though, Google Glass requires linking to a mobile device to function and will indicate to the user automatically if their mobile devices are not connected.

Design Evolution

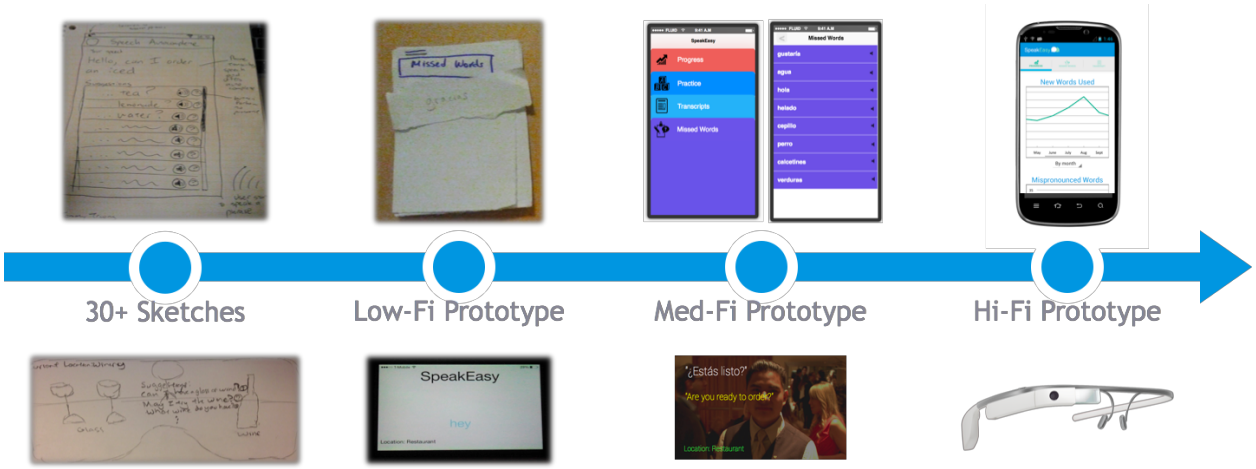


Figure 33. Overview of design evolution.

Sketches:

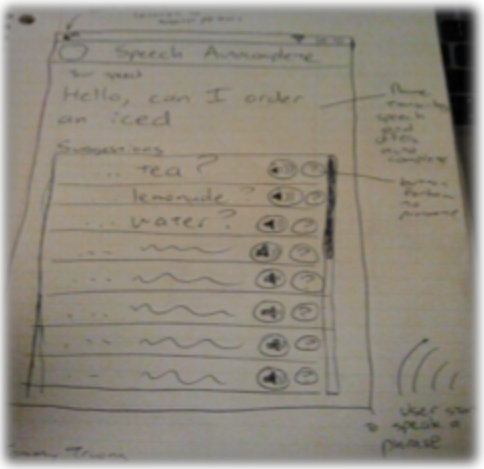


Figure 34. Mobile application sketches.

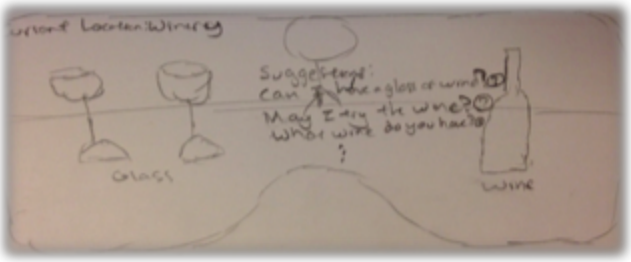


Figure 35. Glass application sketches.

We started the process of designing SpeakEasy by creating sketches of potential solutions. Originally, we created sketches of a mobile application (figure) that encompassed features that ended up in our glass application like real-time suggestions. Some of our sketches had features that did not end up in our final design. One idea was a virtual reality application that

would annotate items in a user's field of vision. We did not end up using this solution because we felt that not only was it not feasible to implement, but there were other learning features that would be more helpful to a user.

Low-fi Prototype and User Testing:

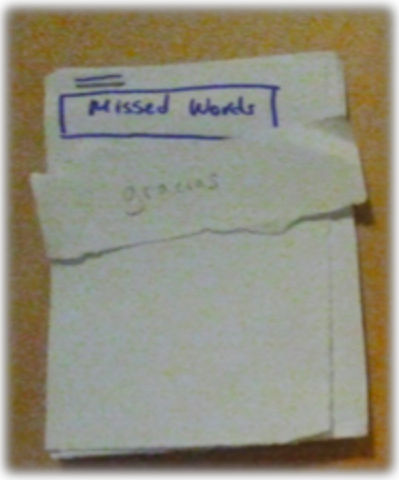


Figure 36. Mobile paper prototype.

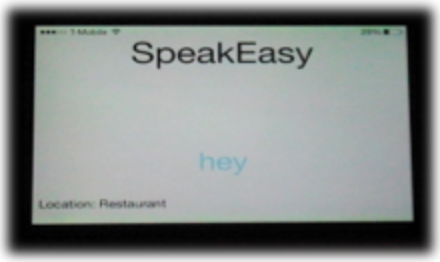


Figure 37. Glass low-fi prototype.

The next step was creating a paper prototype of our application. We used user testing to evaluate this prototype. The paper prototype of our mobile application ended up being pieces of paper on which we could write down the words that users pronounced. Our glass prototype was a mobile application which would display phrase suggestions that a "wizard-of-oz" would type via a computer interface. Users were asked to interact with someone playing a waiter, who would speak to our user testers in spanish.

After the users interacted with the glass prototype, we gave them the mobile prototype to interact with. Users were then able to "click" on the words that they miss pronounced to hear a correct pronunciation of the word.

After completing user testing, we decided to add a translation feature to our glass application. Many of the user testers had difficulty understanding what the “waiter” was saying to them. Based on this feedback, we added the translation feature to our solution. Users also commented that it would be helpful to have transcripts of their conversations to review at a later time. We also added this to our next design iteration.

Medium-fi Prototype and Heuristic Evaluation:

Figures 38 & 39. Medium-fi prototype of mobile application.

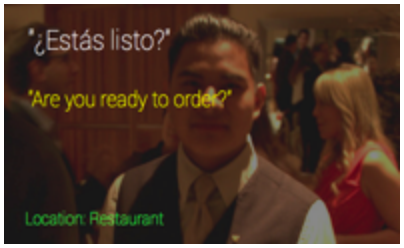


Figure 40. Medium-fi prototype of glass application.

For our medium-fi prototype, we used prototype tools Justinmind Prototyper and FluidUI. We were able to create an example that was much closer to what our final designs would look like than the low-fi prototypes. After creating these prototypes, we were able to have our design evaluated by “experts” to find flaws in the designs. These “expert evaluators” found many places we could improve our application, which we documented in “Major Usability Problems Addressed” above.

Hi-fi Prototype:

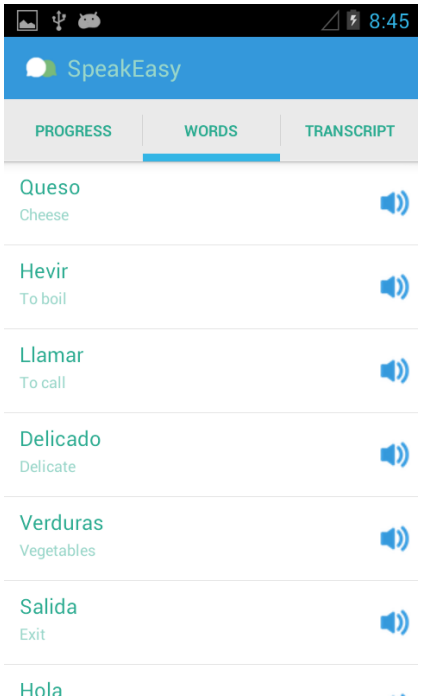
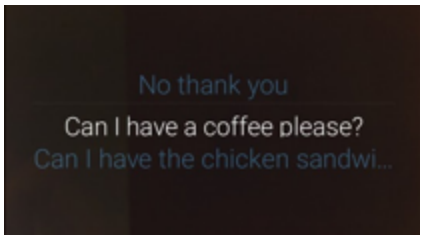
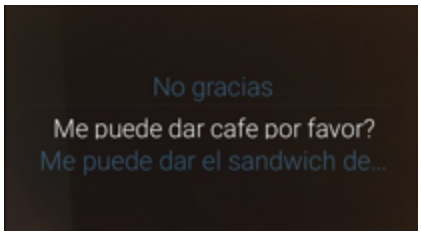
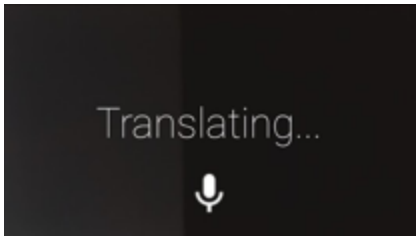


Figure 41. Native android application.







Figures 42, 43 & 44. Native glass application.

Our final prototype consisted of a native android application and a native glass application. We made major changes from our medium-fi to our hi-fi application to create a more usable solution. These changes included creating a unified look to both our android application and our glass application. We created a style guide of colors and fonts to use throughout the application. We also simplified and streamlined the look of both applications by using fewer colors. We also changed various icons throughout both applications to more closely resemble icons that users are used to from the real-world.

Prototype Implementation

The prototypes we built were native Android and Google Glass applications. We were able to build them both on the same development platform, Google's Android Studio, which was very convenient since we did not need to constantly switch development environments. Even so, Android Studio was still in beta and therefore quite limited. While we could use virtual emulators for the mobile application, there was no support for virtual glass emulation. This slowed down our Glass development immensely, as we could not start until we actual had a physical Google Glass.

Within the mobile application, we utilized the Android SDK to do our development. The SDK was very well documented on the Google Developer website and also had strong community of Android developers to aid our development. This starkly contrasted the Android GDK which we used for our Glass development. Since Google Glass is still in beta, the GDK's documentation is poor and is updated monthly. This resulted in many online tutorials and StackOverflow posts referencing deprecated technologies that just were not supported. All the while, the community for Google Glass was very limited, and only had members in North America and the UK. As a result, there was a lack of information on how to do things in languages other than English, such as voice recognition and translation in Spanish. The GDK was also very limited in the number of native features that were available for Glass. While SDK libraries could be used in conjunction with GDK libraries, many of the SDK libraries such ones for displaying scrollable lists did not have support for Google Glass. Therefore, many of these sorts of functionality had to be hacked together for the Google Glass application. For Google Glass, we also utilized the Google Voice XE 22 libraries to build a custom voice recognition system for live translation. While the libraries were powerful and offered much freedom for what we could do with voice recognition, there was no documentation on how to use them, as they appeared to be advanced internal libraries that were not meant to be used by the general public.

The applications we built relied on no wizard of oz techniques, but heavily used hard coded data. It was too difficult for us to set up a communication framework between the mobile application and the Google Glass application, so we relied on visual and audio data was hard coded in the mobile application. This includes the graphs on the progress page, the missed words, their translations, and their audio pronunciation recordings on the missed word page and the transcripts on the transcript page. Much of the data on the Google Glass application was hard coded as well such as the phrases that were displayed to the user, their translations, and recordings of the pronunciations. While the live translation was a bit more dynamic, as we were able to get the voice recognition system to use NLP to return strings that it heard, we still had to hard code a map of potential input strings to their translations that the system could then use to display various translations.

While we were able to simulate all of our intended functionality, the majority of it was hard coded. In the end, we were still missing a lot of components that would allow us to transition from this static system to a dynamic one. One of the major missing components was an

entire backend server and user profile system. This is a component that we would definitely implement next, as having user profiles and a login system could be used to sync up the google glass with the mobile application. This would allow for real time data to be sent to the mobile application, which we could leverage to show actual progress statistics and provide targeted support. The backend server could also be used to do intensive calculations such as producing context sensitive phrase suggestions and processing live translations which would vastly improve battery life on the Google Glass as well as make the experience more real and functional.

Beyond functionality, there were also many UI and UX flaws that we did not have time to perfect, especially on Google Glass. With only four gestures (swipe forward, swipe back, tap, and two finger tap), it was extremely difficult to figure out what functionality we should include on glass. When we had finally narrowed down the features we would include on glass, it was difficult figuring out how to match them up with intuitive gestures, since we had to split the four gestures up into gestures for internal usage within the feature itself, and gestures for external navigation through the applications various other features. With more time, it would have been great to do more usability testing for our Google Glass application to ensure that our chosen gestures were simple and intuitive.