An iOS App made for CSCI 379 – HCI

By Devon Wasson, Li Li, Kyle Raudensky, and Chris Shadek

Bucknell University, Professor Evan Peck

Link to overview Medium post

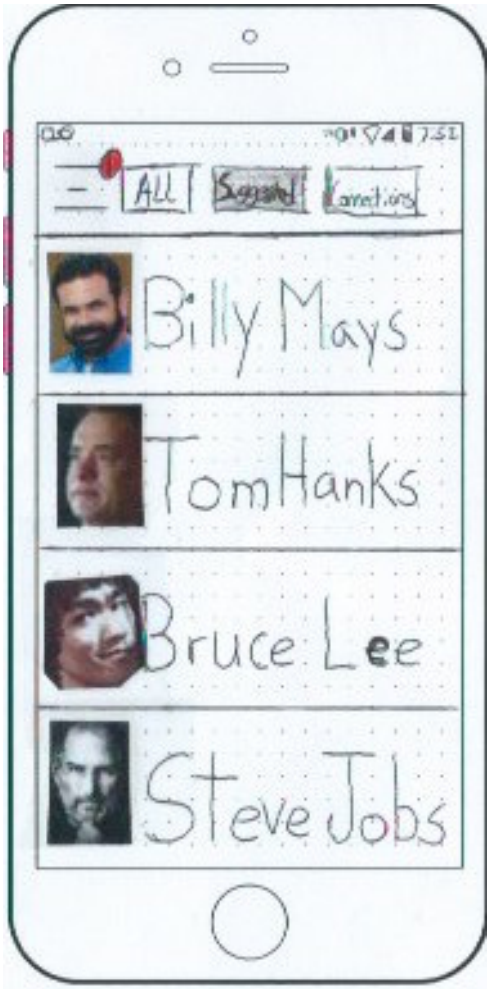
Design Evolution

Low-Fi Prototype

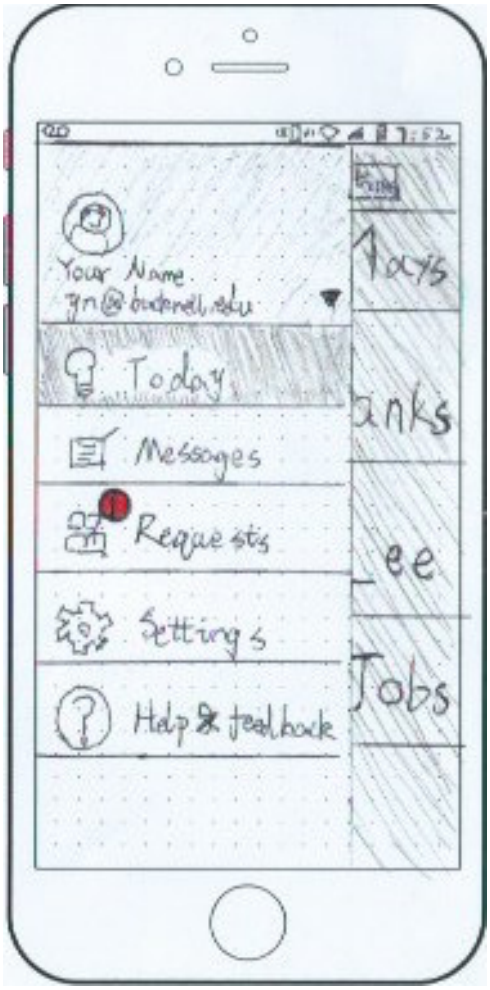
Our initial design focused on an easy-to-navigate layout. We aimed to have as few 'pages' as possible while maintaining expected functionality. The main screen was designed to be a natural place for users to want to be, but also act as a springboard for intuitively navigating to other important pages. Given the nature of our app, we decided it made the most sense to have the main page be a list of profiles belonging to the people with whom the user crossed paths today. This would encourage use of the main functionality, which is looking for people you may know and connecting with them. From there, we also included various short links on the main page. At the top, there is the ability to change filters. These filters alter the order in which users appear on the main page, and hopefully allow the user more freedom in finding people with whom they want to connect. Finally, we included a button to open up a side-menu. This menu lists other features and pages, in addition to displaying a red notification if the user has any message requests or notifications.

We used paper templates found on the web to model our application. We tried to make the application look realistic so when our subjects would test the prototype, they would not be distracted by any abnormalities. To make the prototype look clean and seamless, we printed out tiny pictures of our 'users' and used invisible tape to secure them to the prototype. Having color and high quality pictures helped convince our subjects this had potential to be a real, without making it seem like we spent an enormous amount of time on the application. This was crucial as subjects are less likely to give honest feedback if they think the designers spent a lot of time on the prototype. Our design was made with just enough detail to look realistic while still maintaining a prototype look and feel.

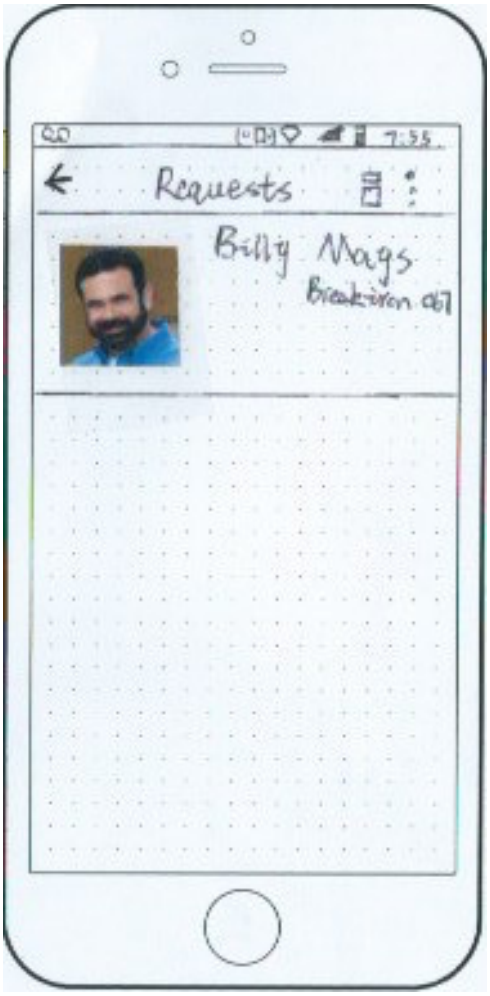
The default setting on the main page is to view by ‘suggested.’ At the time, we were not sure exactly how suggested would work, but the idea was to have the app suggest other users first if the app thought those users were people the current user was most likely interested in talking to. We imagined the logic would be something like giving users a score based on how frequently they crossed paths, where they crossed paths, and how long they were in the same location. The ‘All’ filter would allow the user to see everyone they crossed paths with today, while the ‘Connections’ filter would allow the user to see a list of all people with whom they were already connected.



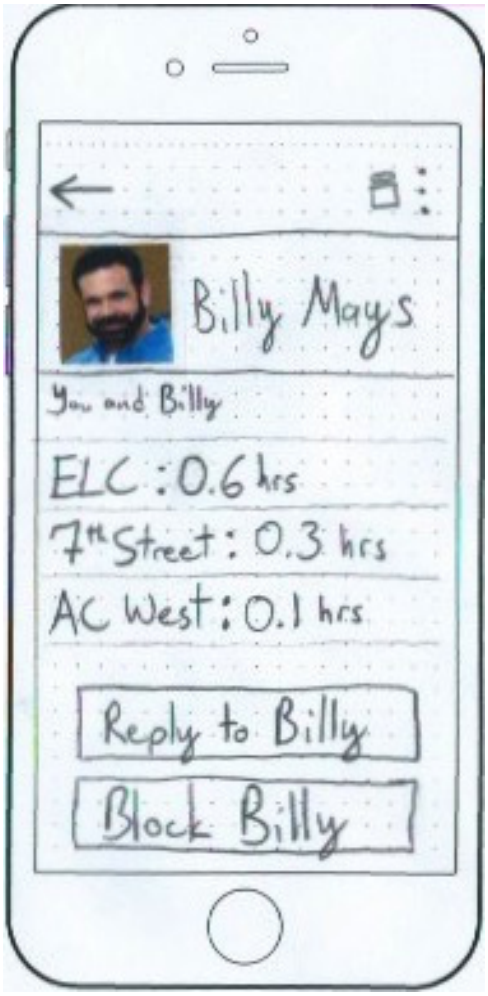
Tapping the top left corner would open up a sidebar menu. This menu held links to all other pages the user may have wanted to view. For example, the user's profile would be displayed at the top, and the next option would be to go back to the home page. Besides settings and help, there were two other sections: messages and requests. They acted as the main pages for the second aspect of the app.



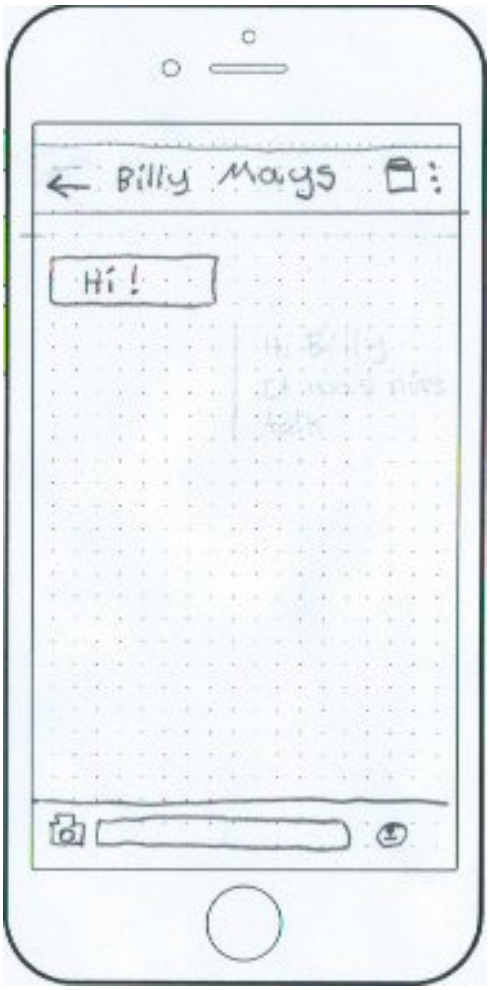
When a user wants to connect with another user, they have to send a message. However, messages are not automatically accepted by receiving users. Instead, the receiving user must accept a message ‘request’ by clicking on the request tab. Once the user does so, a new page is opened, displaying all pending message requests from other users.



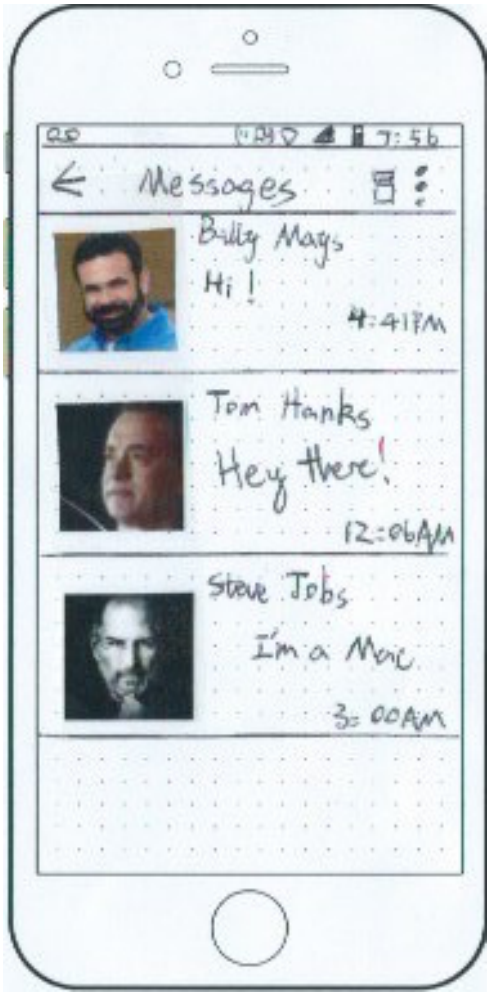
In this example, Billy Mays wants to send a message to the user. From here, the user can decide if they want to message him back or if they want to dismiss him. Swiping his profile card to the left would bring up a dismiss button, while tapping on the profile card would open up Billy May's profile.



From the profile page, the user can see information about Billy Mays, such as where they crossed paths, and for how long. The user can then either reply to Billy's message, or block him entirely. Blocking would open up a dialogue box confirming the block, and notifying the user they could never interact with Billy Mays again on the app once blocked. However, if the user deiced to accept his request, they would tap 'Reply to Billy' and be presented with a typical message view.



This is a fairly intuitive process as the style for the messaging system perfectly reflects that of any standard messaging system in an iOS app. The user can then click the back buttons at the top left corner to take them back pages throughout the app. Now, the user can go back to the side menu, and click on the 'messages' option. This will bring up a new page, which displays all active messages.



Users can now see a thumbnail image, the name, the most recent message, and the last time they crossed paths with other users with whom they are conversing. For consistency, clicking on any of the items here would open up that user's profile, and as before, you can reply to their message or find out more information about them from their profile page.

Low-Fi Feedback

Our group interviewed four people and had them test out our paper prototype. We had our users perform a variety of tasks. The first task was to browse through the main feed of the app. We wanted to test if it was intuitive that the first screen displayed was the most important screen. During our testing, all users could identify it and interacted with the app in a natural way, such as clicking on profiles and switching filters. We then asked users to send a message to someone. Most users found the easiest way to do this was to click on a profile, tap 'send a message' and then write out a message. However, some users opened up the side menu and tried to click on the 'messages' tab. This tab was designed to only hold active messages. This was the first obvious point in which our prototype needed reworking.

The next task asked users to check their message requests and send a message back to any unread messages they may have. Users could instantly identify the red dot on screen as a notification and could navigate to the appropriate screen to find said notification. While it was easy because of these cues, it is here where it became apparent that having messages and requests split into two pages might not be the best way to go.

The fourth task was to change the filter to display all people. Everyone could easily accomplish this task, but there was confusion as to why they were doing it. This was the first time the idea that our filters were nonsensical was brought to our attention.

The fifth task was to block Steve Jobs. All users easily figured out the best way to interact with another user was to find their profile and go from there. Interestingly enough, some users found him on the main page, while some found him under their active message list. Either way, users could then block Steve once they were on his profile.

The final task was to send a message to Bruce Lee. While this was similar to the last task, we wanted to see if it was just as intuitive to go to their profile first, or to go to the messages tab first. The results were split between both options. About half of the users looked for him in their messages, while the other half found him on the main page and sent a message through his profile. While either would work, it was interesting to note that our users did not prefer one method over another.

Low-Fi Changes

After all of our testing, we gathered the feedback and errors in usage by annotating our paper prototype. We realized a few things while doing this:

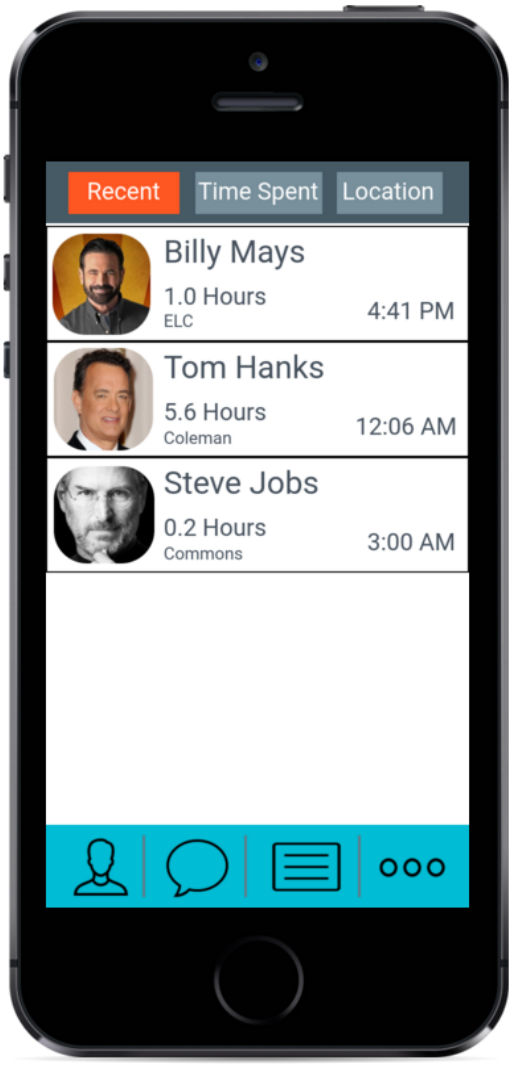
- The filters at the top should be changed to read Recently Seen, Most Time Together, and Browse by Location.
- Each smaller profile on the main page should have information at a glance, like where they were most recently seen by you.
- The 'Requests' tab needs to be implemented as a filter on the Message tab to make things less cluttered and more intuitive.
- There should be a tab to view profile, as the way we had it set up did not offer enough functionality, such as changing your picture or managing your blocked contacts.
- The 'Messages' tab needs a filter at the top to switch between Message Requests and Current Conversations.
- Each conversation should have a small button in the top right to allow users to quickly view that person's profile.
- Tapping on a name in the messages section should bring you right to the message, rather than to the profile.
- The side menu bar should be replaced by tabs on the bottom of the screen. The tabs are the main menu, messages, and your profile, in addition to a 'more options' tab to house any other necessary pages.

These changes were then implemented in the medium fidelity prototype.

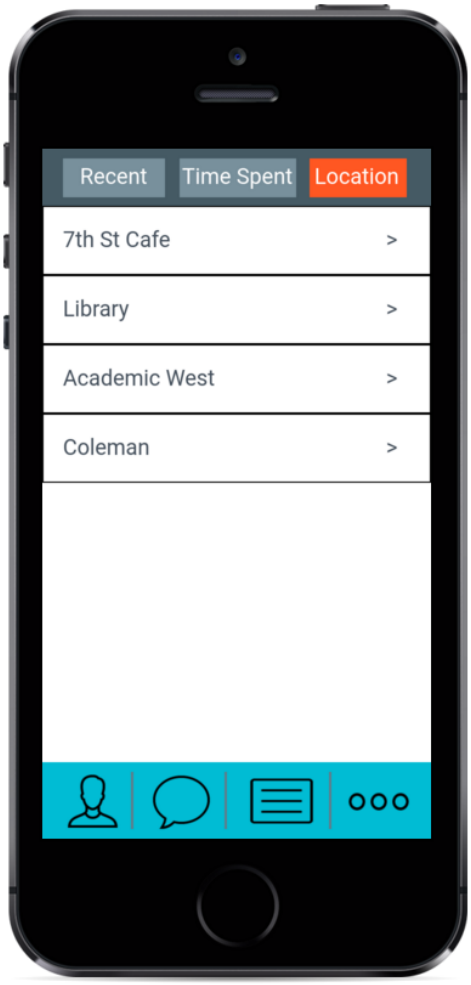
Med-Fi Prototype

With these changes in mind, we set out to make our medium fidelity prototype. We used a tool on marvelapp.com to build our prototype. We started off with the main page.

Besides adding in color, this page saw quite a few changes from our paper prototype. As discussed, we renamed the filters at the top to make more sense. *Recently* helps the user browse by people they might have recently seen, *Time Spent* helps users find people they spend the most time with, and *Location* helps users browse by different building locations, so they can find someone they were near in a specific building. Clicking on *Time Spent* or *Recent* just shifts the order of the main page, but clicking on *Location* makes the main feed look like the picture below.

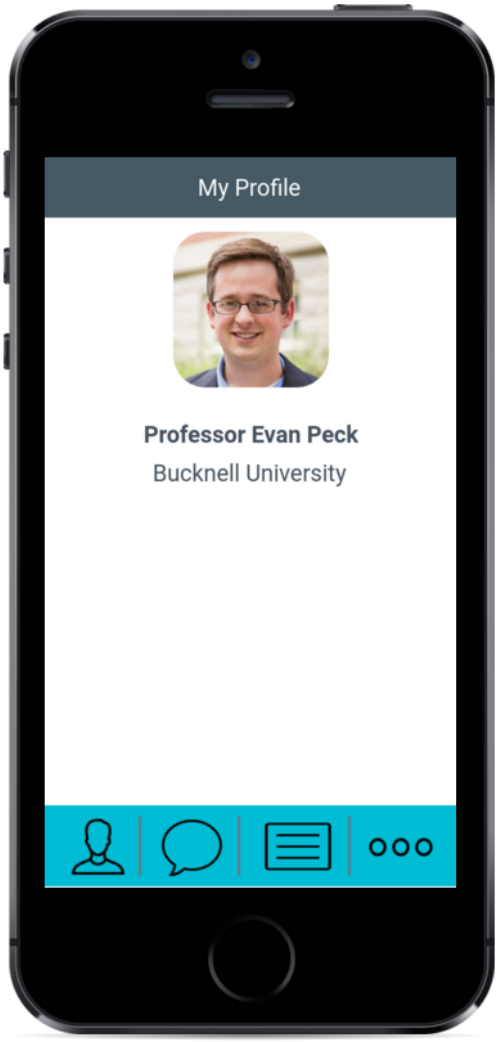


From here, a user can click on any of the buildings they have been in, and browse other users with that building as a filter.



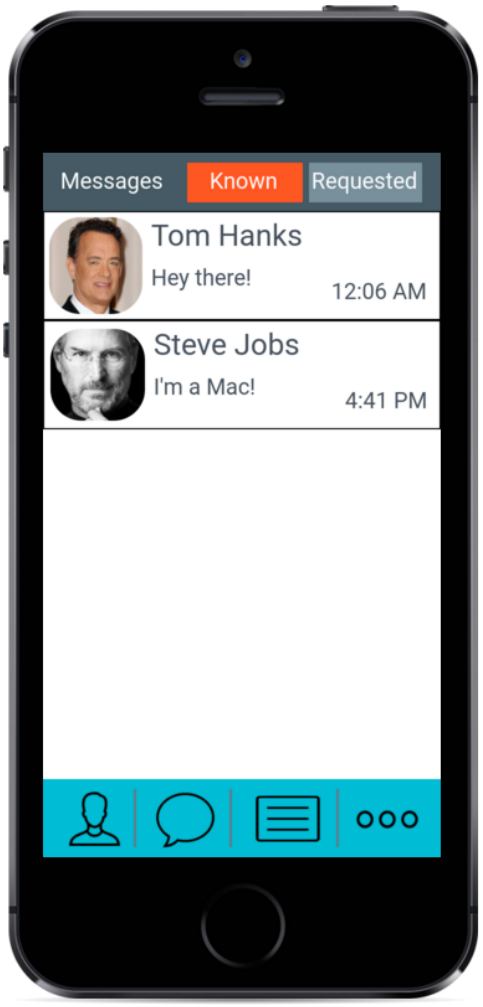
Additionally, we implemented buttons in the bottom rather than as a side bar. The first button represents the user's profile, the second is the user's messages, and the third it the user's main feed. The fourth button would lead to more options, but considering we had nothing to put in there there was no specific functionality.

Tapping on the ‘three lines’ button brings you back to the main screen, but the rest have other functions. For example, if you were to click on the first button, or the ‘outlined person’ button, you would be taken to your user profile.

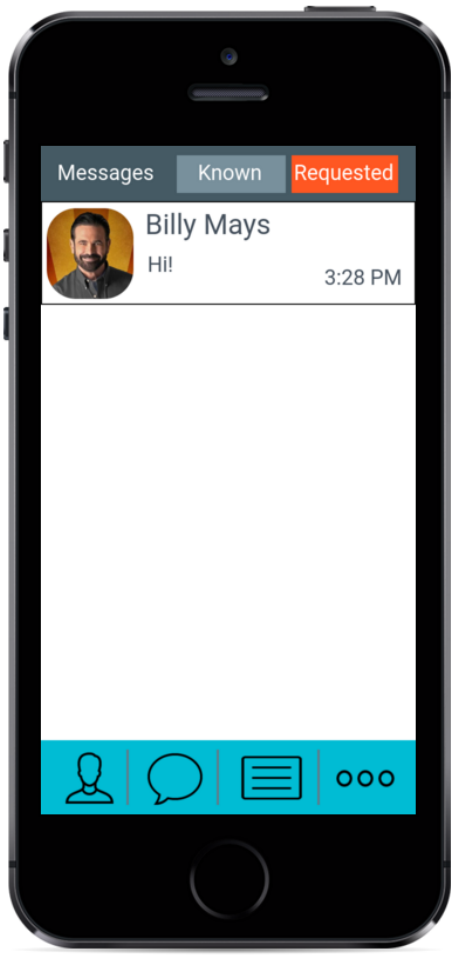


Here, the user can see their profile picture, their name, and the university they attend. A possible feature would be to add editing options here, but given the time constraints we did not add this functionality.

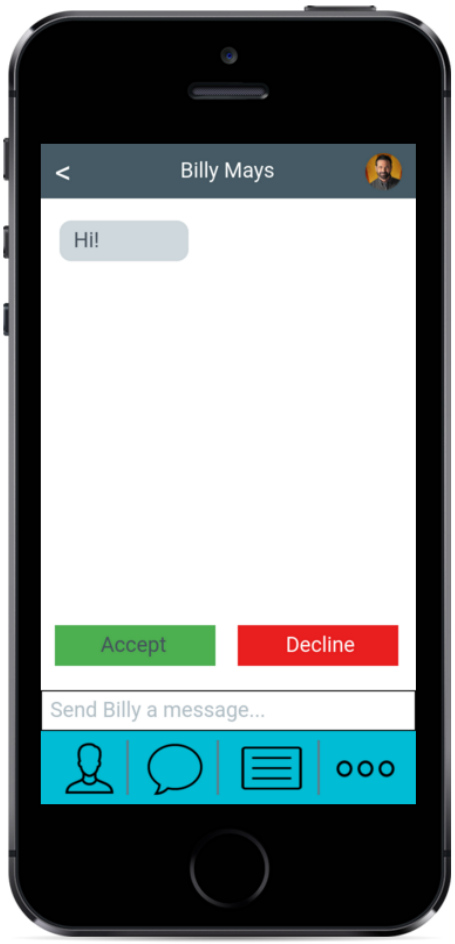
Next, clicking on the messages button takes the user to a page displaying their current messages or requests. As mentioned, we consolidated messages to people already known and pending messages. Having *Known* highlighted will only show active messages.



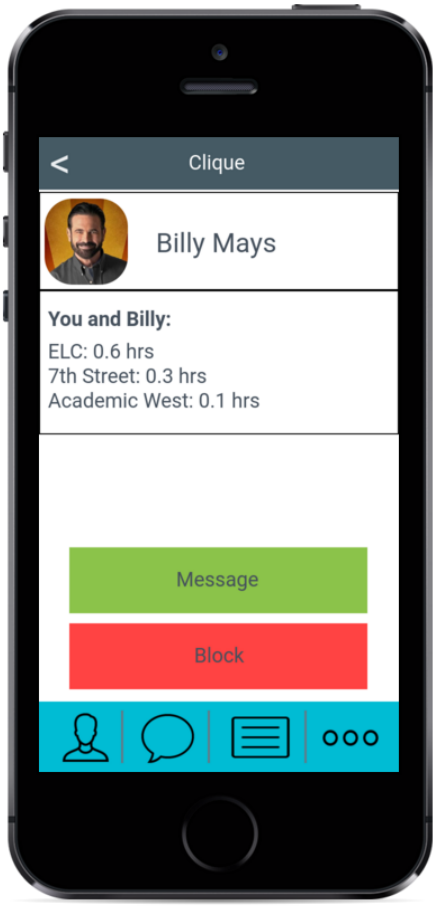
Switching over to *Requested* will bring up a list of people who have requested to message the user. This consolidation allows the user to experience a more intuitive process in responding to different kinds of messages within the app.



From here, a user can click on the “requesting user’s” message to bring up another screen. This is also a change from the original design. Instead of bringing up the profile of the user, the message itself is displayed. However, the user must physically press *Accept* or *Decline* to continue or delete the message. As shown below, this allows the user more freedom in deciding if they want to converse with the other user or not before making any decisions. Pressing *Decline* will exit the page and delete the request, while pressing accept will open up the ability to send a message back on the same page.



Tapping the picture in the top corner will take the user to that user's profile, so they can get more information about the user with whom they will interact. From the other user's profile, the user can see a picture, name, and list of times/locations they shared with that user. Additionally, there is the functionality for messaging or blocking the user within their profile. From here, users can still navigate with the bottom bar to quickly get back to more useful screens, or they can use the back button to return to the previous page.



Med-Fi Changes

At this point, we let other people in our class use the prototype. From this, we could gather a lot more feedback. Unfortunately, some of the feedback was asking for features which would be hard to implement given time constrains, and some of it was requests that had to do with the limitations of the website we used to build the prototype. The rest, however, was very constructive. Below is a list of important feedback we received from our medium fidelity testers.

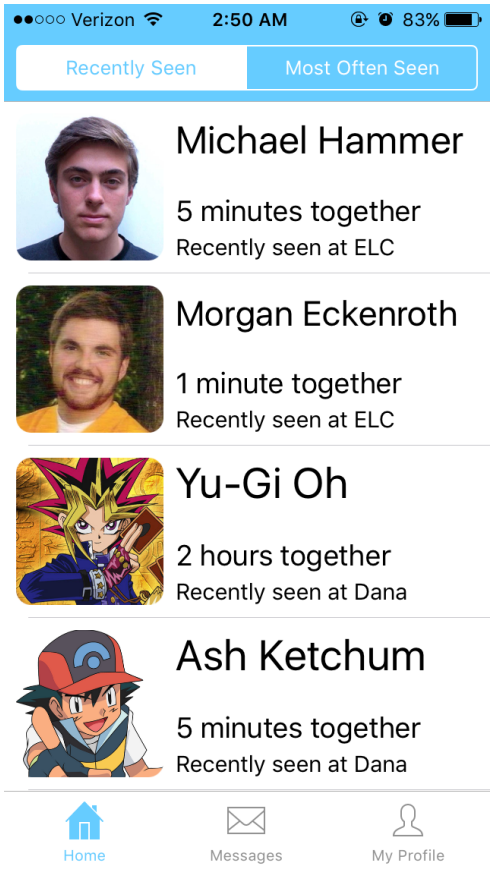
- Implement highlighting to show current status of buttons at bottom of screen.
- Reorder the icons at the bottom to be more intuitive.
- Change box with lines icon to be a home icon.
- Add functionality to the three dots button or remove it.
- Reword or remove *Last Seen* and *Total Time Together* bits of information on the quick view of a profile.
- Remove floating point times, either convert to minutes or round down to hours.
- Enable users to edit their own profiles.
- Add a search bar.
- Add a block list in the more options tab, and allow for unblocking.
- Need to be more clear on the difference between blocking and declining.
- On another user's profile, have only three most recent time pairs, rather than total time.
- Prompt the user if another user has disconnected from them in messages.
- Put white text on the red and green buttons.
- Change blue colors to be more consistent.
- Reword the filter options.
- Make consistent transitions.
- Be more specific with location names (i.e. Bertrand Library instead of just Library).

Aside from these changes we got from our feedback, we implemented a few more changes to make things more usable. We added a message button to the top right of each user profile rather than at the bottom. We also added the bottom tab bar to every page for easier navigation. Finally, we removed the sort by building feature, as we thought it detracted from the human-centric design of the application.

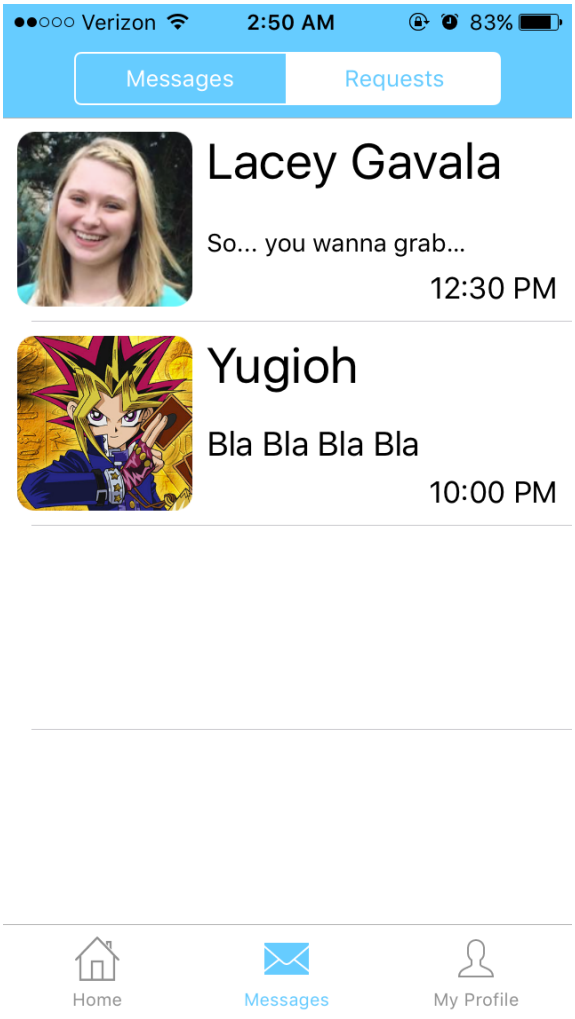
Final Product

With these changes in mind, we finally set to work on our final product. The natural place to begin after launching the app is at the main page. As seen, we added in highlighting on the bottom tab bar so users would know their current location within the app's navigation. We also removed browse by building, and we cleaned up the color scheme. Transitions between pages are now consistent, and times now appear as either whole minutes or hours. Additionally, we reordered, remodeled, and removed some icons on the bottom of the page.

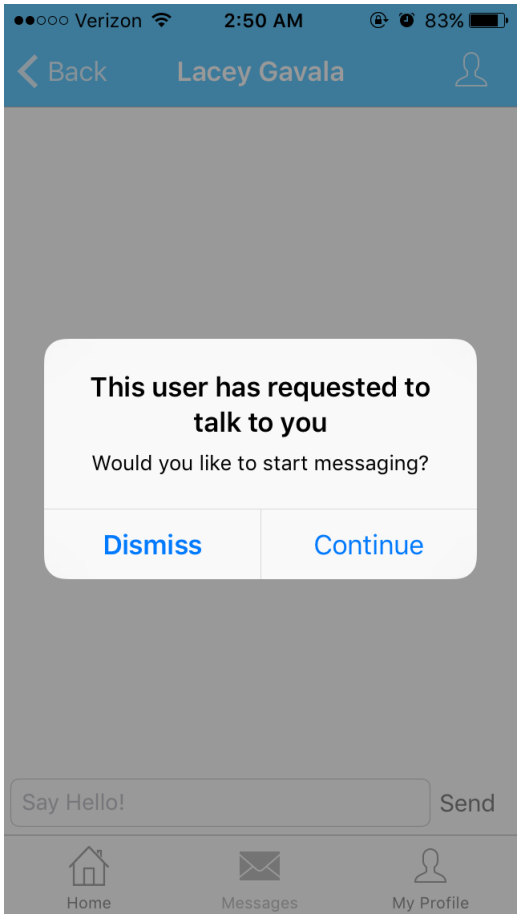
As seen, other small stylistic changes have been made to be more visually appealing. Consistency with the blue and grey colors, in addition to accent highlighting, really helps make the page look and feel clean.



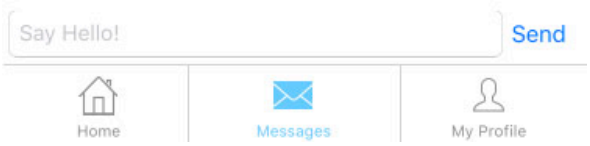
From here, the user can easily navigate to their messages. One change we implemented was to have the requested messages filter be the default, if there is currently a requested message in the inbox.



Here, one can see the list of people who have requested to message our user. Clicking on the message will take the user to the message screen, with the option to accept or delete.

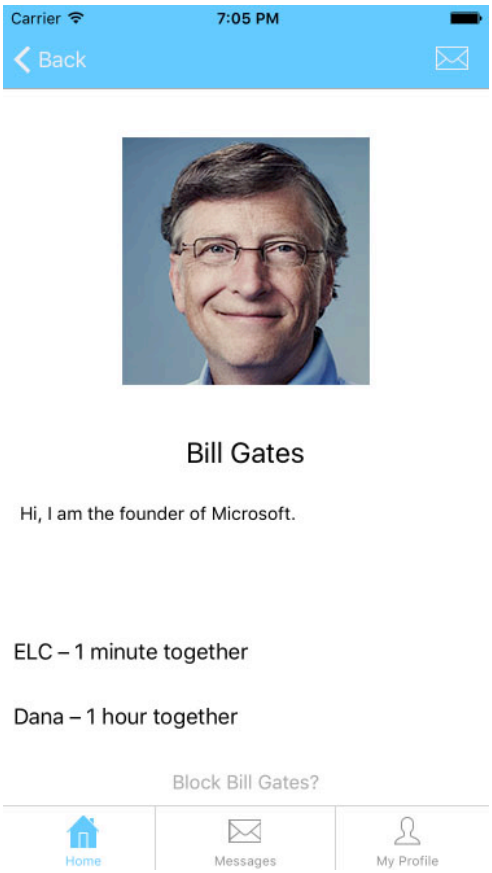


If a user accepts a message, or if they switch to the *Messages* tab and click on one of their active messages, they will be greeted with this screen.



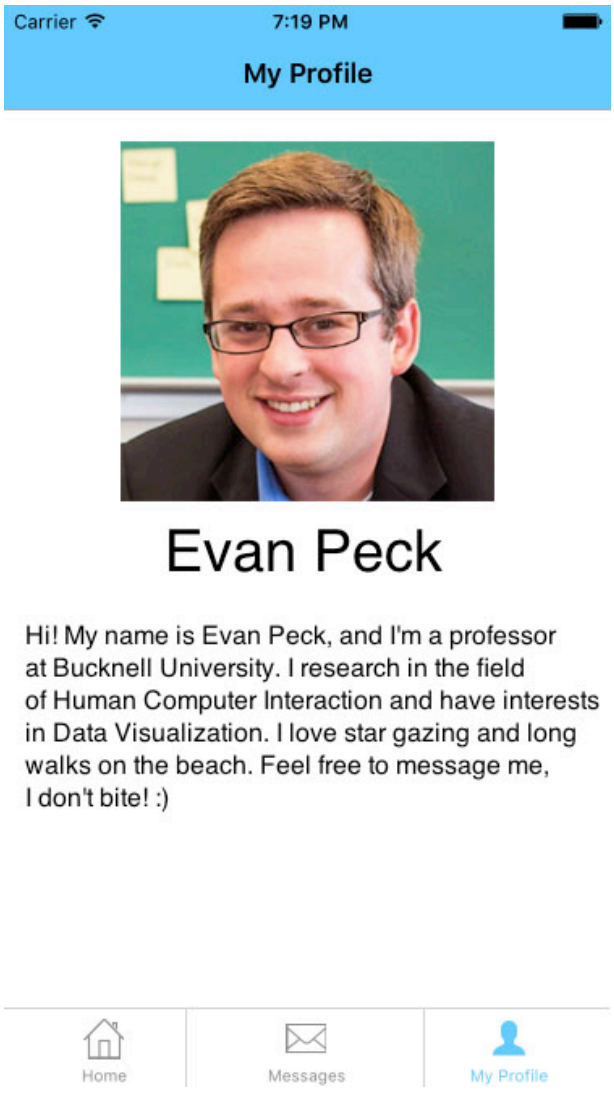
Here, Bill Gates has sent a message to our user. This function works exactly as expected. The text boxes scale to fit just the message. The text box at the bottom brings up the keyboard so users can type and send messages to other users. If the keyboard is brought up and there are more messages, the messages will shift up, rather than remaining hidden behind the keyboard. From here, the user can either go back to their messages, or click on Bill's profile picture to bring up his profile.

If the user decides to view another user's profile, they will be greeted with a page like the one below. We changed some things up in the user profile. Rather than having large and out of place *Message* and *Block* buttons, we made them subtler yet still intuitive.



Putting a message icon in the top right corner was a more intuitive location, rather than forcing the option on the user. We had the block option be less prevalent as well. We do not want to encourage our users to block everyone they come across, but we still want the option to be there for hopefully rare instances when users might not want to be bothered by certain individuals. By placing the *Block* button at the bottom, it is sometimes even not visible right away. If there are multiple places the users share together, the block option can even be pushed off screen, and a user must scroll to find it. This is acceptable as it is an intuitive place for the user to find a block button (much like an unsubscribe button in an email) and is also not always right in out front. This way, users can search for the *Block* button if needed, but because it is a less utilized feature it can be placed in a less obvious location.

Finally, we made changes to the current user profile. That is, the profile of the person using the application, not other users' profiles. The only real change we made was to remove the user's school and replace it with their biography. In the future, this biography would be editable from here, but this was something we did not have time to implement.



We wanted this profile page to have a similar feel to the other users' profile pages but with slightly modified functionality and design. By not having the lists of matching locations or a message button, and by highlighting the *My Profile* tab, the user is sufficiently alerted to which page they are on. However, repetition in style helps the user feel more comfortable when navigating the application and standardizes scheme and design.

Prototype Implementation

Technologies We Used

Rather than limiting ourselves to only a front end on this application, we utilized our wide variety of specialties to achieve real implementation and core features. We started by using Azure to host a virtual machine. The server is running our Ubutnu virtual machine 24/7 so the application can access the servers at any time, without any problems or complications. The backend database is MongoDB. We opted for this NoSQL-type database so we could keep information in JSON format rather than spend time developing protocols for switching between SQL and JSON formatting. The code for making calls between the database and the actual application uses Parse Server code, written in JavaScript. However, because Parse is closing its doors in the coming months, we opted to host an open source version of Parse on our Azure server. Additionally, we are utilizing the Parse Dashboard to help visualize out data on the back end. The Parse Dashboard allows us to view, add, and modify user data on the web through a simple interface. This made testing the server connection with dummy data easy, which served as a crucial feature because we needed the database to match the application while we were developing. It was extremely useful to be able to reformat the database on the fly.

Of course, the most important part of this application is the front end. Because we designed an iOS application, we used Xcode as our development environment. In addition, we wrote our application in Swift 2.0 instead of Objective-C in an effort to stay on the cutting edge of technology. We also utilized a tool called CocoaPods to manage all dependencies we used in making the app. To get user location information, we used Geofences. Rather than taking location data frequently, Geofences only send information to the servers when a user enters or leaves a predefined Geofence. This saves on data consumption and battery life, in addition to being less invasive of the user's private information. Also, setting up Geofences around buildings in a college campus makes sense for what our app is attempting to accomplish. We want to know when people are in the same general area, so using Geofence technology makes sense. While using more accurate user data might increase resolution and get more specific recommendations about other users, the pros of Geofences outweighed the cons.

Aside from the more direct technologies we used, we also used web-based tools to help us in the development process. We hosted our own Jira server on Azure to keep track of our sprints.

We designed a mood board to help shape how we wanted the app to feel. From this we developed a style guide for reference when making our app. Finally, we referenced an iOS style guide when building our application to adhere to Apple's design standards.

What Works

One aspect of our application we are very proud of is how much actually functions. Instead of just making a fancy version of a wireframe, we also implemented a lot of back end technology. As mentioned above in the technologies we used, we utilized a MongoDB to store and call our data. The application is not currently sending data to the backend, but once it does, we will be able to easily access it and populate the app correctly with user information. All we would have to do to get this working is to decide which locations are the Geofences, and place that information into the application. The retrieve calls grab user data, manipulate it, and display it in the application. So even though our methods for making and gathering user data may not be implemented, we still have a working backend. The application populates forms with the hard coded data we wrote into the backend using the Parse web client. We can populate different users that the current user has seen, where they have been seen, and for how long. We fill in their relevant information in different labels, such as headings for the user profile. Additionally, other users' profiles list all common places with the current user, those users' profile pictures, their bios, and their full names. The user's profile also works in the sense that it displays information for the current user, including their picture, name, and bio.

The real-time messaging system also works. Users can send and receive messages instantly with other users. We hard coded which users are talking to whom, but the actual back end system works. Two users are connected through a PubNub channel, to which they are both *subscribers*. When one user sends a message, they are *publishing* to that channel. Because the other user is subscribed to that channel as well, they are listening for messages and will automatically be notified when a message is published so they can pull it from the server for viewing. The app will instantly populate the messaging field and update the view when a message is published/received. The keyboard functionality is complete. The user can tap the text field to bring the keyboard up onto the view, pushing the previous messages up the page so they are not hidden.

The sorting functions also work correctly. On the main page, we allow the user to sort between people they have seen recently, and people they see most often. Choosing one of these options will

change the order in which users appear on the main feed. Allowing users the freedom to change how they find other users is crucial to the apps success. We wanted to make it easy to find that one person you were looking for, and this function makes it possible to do just that. Additionally, we have switching functions on the messages page, allowing the user to switch between ongoing messages and initial message requests. That way, the user can quickly see who has sent them a message recently, and can accept or decline the message from there. On the other tab, the user can access the conversations already in progress.

Much of the messaging functionality works. Messaging multiple users, one of the core features of our application, is implemented. We have the messaging system set up such that users can hold multiple conversations with other users. The requests feature works exactly as intended. Every time a user receives a brand new message, it will go to the *Requests* filter instead of *Messages*. When a user taps on a requested message, they will be prompted to accept or dismiss. If they dismiss, they will be taken back out to requests if there are more quests, or to their messages if there are no more requests. If they accept the request, they may continue messaging the user. When they back out of the conversation, that user's messaging block will be moved to the active messages filter and removed from the requests filter. Again, the display one sees when backing out of a message works as before, going to requests if there are more requests, or right to messages if there are no requests.

Finally, the basic functionality of the app works. This might be something one could easily look over, but it is important to mention. All basic aspects of the application work as one might expect. The user can scroll to see more users on the main page, tap a user to view their full profile, and message that user. They can also navigate through and between all of these pages with ease. The user can view their own profile, go check out their messages, respond to new messages, and check out that person's profile all with just a few taps on the screen. We designed the app to be a fluid and easy to use as possible, and the core functionality of our app shows just that.

What's Faked

Even though we have a lot of functionality in our application, there are still a few things we just had to fake. These are features we would like to implement in the future, but are currently just shown on screen through manual hard coding. We needed these features for demonstrational purposes but did not have the time to implement them correctly for authentic use.

Messaging is only slightly faked in the application. We have messaging for multiple channels implemented, but currently those channels are hard coded. Because we aren't gathering user data live, it did not make sense to waste time making messages respond to live data. Instead, we opted to hard code some default users into messages and requests, and made it possible to interact with them.

Geofence technology also is half implemented, half unfinished. The code for Geofences is in our application, and it records when a user enters or exits the bounds of a defined Geofence. However, we did not have time to figure out the exact locations for fences we wanted on campus. Because we would have to find suitable center points for each building, and potentially have multiple points for some buildings, we decided this was not needed for our demonstration. We have some user data faked, including where each user has been. If we implemented Geofences and actual user profiles, then we would have real time data from real users.

We also have not implemented actual users. We have hard coded user IDs into the database, and information to go with them. As we downloaded the application to our phones, we changed the hard coded ID to be unique for each of us. That way, when we show it off, it will appear as if we each have user accounts. Even though we are not generating unique IDs for users, we are still allowing users to have information in the database, which is crucial to the correct functionality of our app. Once actual users are implemented, the functionality of most of the application will immediately fall into place.

What's Missing and Thoughts for the Future

Even though we have a lot of implementation and usability, a few things were left out from our design, mostly due to time. We would have liked to have included a search bar to allow searching of specific users. When considering usability, a search bar makes a lot of sense. A user might already know the name of the person they want to talk to, so allowing them to just search instead of having to scroll through potentially all users would be a nice feature. The search bar would help in the messages section too. If a user has several active messages, it might be difficult to find the one they want easily. Implementing a search bar would make it easier overall for the user to complete desired tasks.

We would have liked to implement a more robust log in system. Currently the application treats each unique device as a new user, but it would be nice to have some sort of cross-platform log in system, so users with multiple devices can sync their information. This would allow someone to log in with their iPad but still see the information gathered from their iPhone. This would also open up the potential for having a web client. While a web client would be designed for desktop and thus not really effect the Geofence aspect of the application, it would still allow for users to communicate with each other in a platform in which they might be more comfortable.

Two smaller problems we came across when designing the app had to do with timing. When do we reset data for users, and what if the user is traveling across time zones? What we originally planned was to keep data historically, but only show the most recent data. However, we decided it would be best to only have the most recent data, as users will most likely not be looking for someone they passed by yesterday. But the question then arose of do we reset all data at midnight? At 3:00 AM? Do we push off data once it is 24 hours old? We did not come up with a solution for this, and it would have to be something we implement going forward. What if a user crosses into a different time zone? How do we handle data then? Again, these are questions for us to consider in the future, and were not entirely relevant to our prototype.

Finally, we also would have liked to implement more editable features for users. Specifically, it would have been good to allow users the option to change their profile picture and description. Currently, these options are hard coded directly and obviously that is not a sustainable model. In the future, we would have these options be editable and storable on both the device and the server. That way, users can update and send their changes to the servers, ready to be pulled by other users. This would make the app more of a realistic end product.

Final Thoughts

Even though the application is not implemented fully, we believe our application already shows a lot of promise. We followed good style and usability guidelines. Because of this, we believe our app is already close to something people would be excited to use if it was on the market. This is a huge step in development and we could not be happier with the outcome. There are still features to add and minor tweaks that can be done, but overall the application is something of which we are proud.

We took an ambitious step in choosing to address college hookup culture as the theme of our project. We needed to develop specifically for users to interact with one another, rather than just a tool to complete a task. Finding a way to appeal to users both in usability and in features was a daunting task. Because our application is trying to solve a human-to-human problem, we had to develop in a way that encouraged natural usage. Rather than designing our application as a tool, we had to make sure it felt like a realistic way to really connect with other people. If we did not make it feel natural, our users could easily reject and replace it with other methods.

Considering all of this, the most valuable part of the human-centered design process was understanding the context of use. Because we had to develop an application that would facilitate human-to-human interaction, we needed to clearly understand and define how each user would ideally use this product to meet others. We needed to build it in such a way it could not be easily abused, but was also intuitive. It needed to help people find others and reach out in a simple way. Understanding it was to be used not as a dating app, but to make connections was crucial to our success. Once we understood clearly how people would want to use our app, it became much easier to design the application itself.

If we could repeat this process, there is not much we would change. We could figure out what exactly our potential users wanted and needed during our interviews, and the feedback we received from our prototypes helped us tremendously. At no point in the development do we think we went in the wrong direction. We started big and thought through lots of different ideas and avenues. Our interviews helped us refine our ideas. Prototype implementation allowed us to see what worked and what did not. By the end, we designed and created a product in which we take pride.