Milestone 3

Digital and Interpersonal Communication

CPSC 4140 - section 001
2/28/18

Austin Youngblood

Noah Axelrod

Walter Thompson

Carson Sallis

Alexander Stone

https://github.com/StoneHub/4140

Usability Specifications and Evaluation Plan:

A quantitative benchmark tasks would be:

- How many errors does the user run into while performing a task?
- How long does it take to perform a given task? (Sending message, receiving message, sending image, placing a marker, taking a picture.)
- How often would a given feature be used?
- What UI elements get interacted with the most.

A few tasks that users will perform in the upcoming test for qualitative evaluation of the interface is:

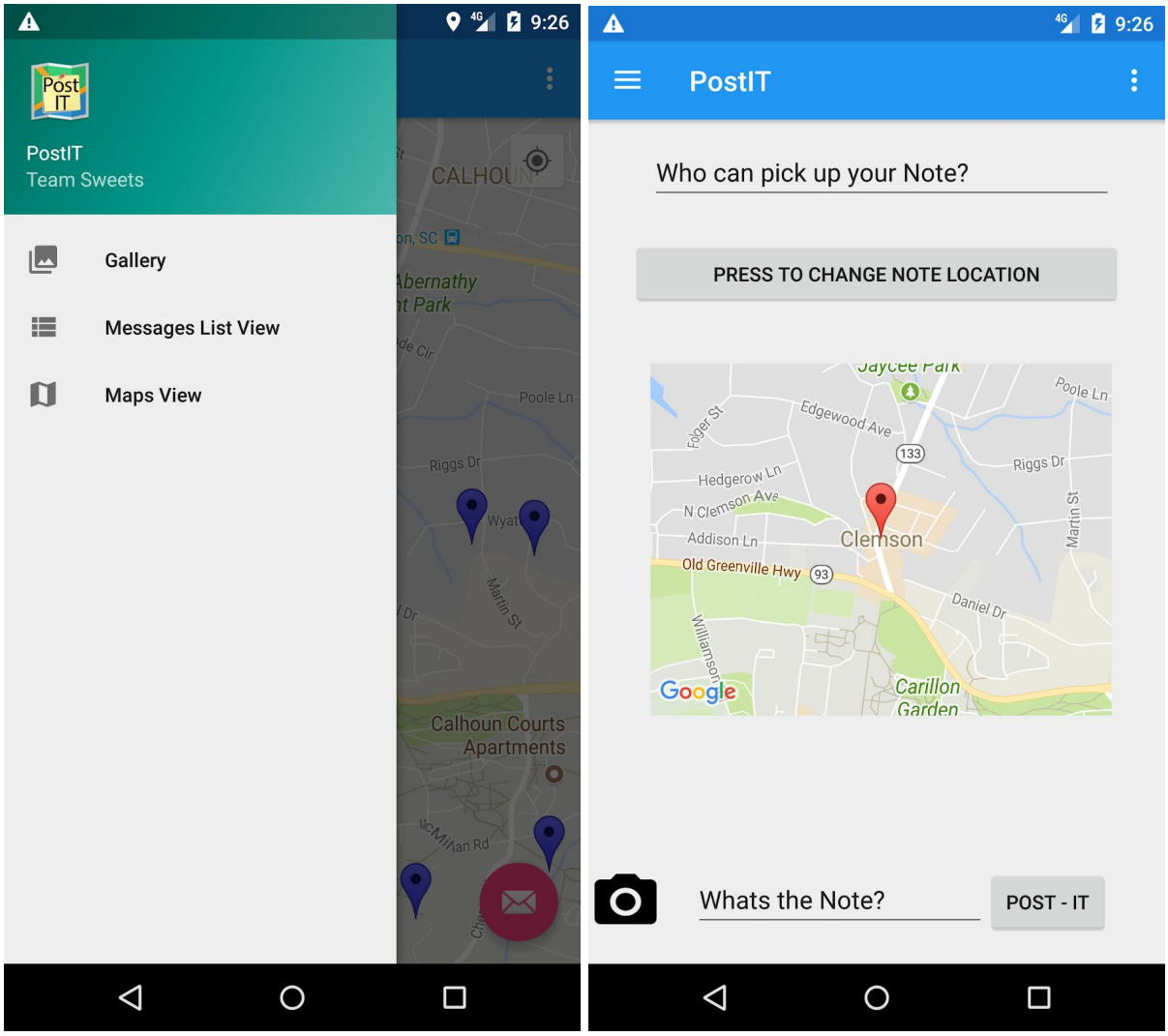
- If user will send a message/image or both
- How a user will choose to view received messages (will the user tap on a nearby blue marker or using the left panel menu)
- How a user will place a marker (will they typically place markers on their current location or nearby)
- How a user will take a picture/ import from gallery
- Where or when will users use this service E.G. interesting locations or daily routes.

Giving the test users an android device with the PostIT software loaded. We then observe them interacting with the app to see what peoples attention is drawn too. After initial observations we put the PostIT software into logging mode for tracking quantitative statistics of the users interaction. During the logging observation session, we capture user input to track how long it takes a user to perform a specific function. This data will be used and analyzed to graph any abnormalities in the software and to improve user experience.

Prototype Description:

Sending a Message

Our prototype's home page uses the Google Maps API to show the user's current location. The user can drop a red marker onto the map within a small radius of their current location. This marker indicates a message that the user can drop at that location. After a marker is dropped, the user can tap on the mail icon at the bottom-right to compose a message. The user can then select the contact who they wish to send the message to and compose the message. The page also depicts a map of the location where the user chose to drop the message marker. In addition to a simple text message, users can also attach pictures and videos to their texts just like an ordinary SMS message. This can be done by opening the side panel on the home page and taping on the camera or gallery options. When the user has reviewed their message, they can press "Post-It" to post a message at that location for the desired user(s) to see.

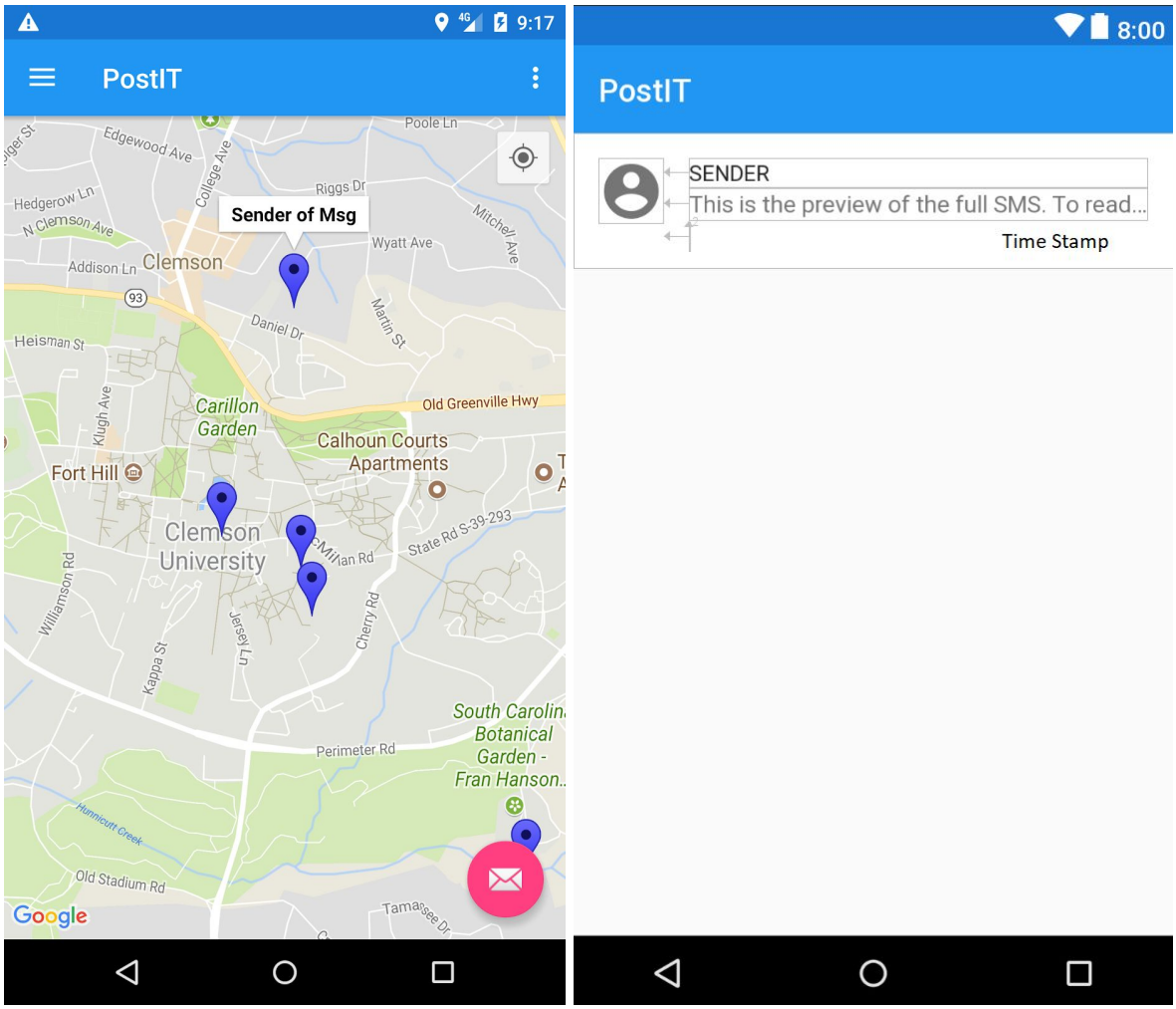


Our current implementation relies on SMS to send and receive messages. This allows users to easily send and receive messages from multiple people. Though it has not yet been implemented, users will receive standard SMS notifications when a message has been left for them. However, the notification will only display the name of the user that sent the message. To view the contents of the message, the receiver must go to the location where the sender dropped the message.

Receiving a Message

As previously mentioned, a user must travel near another user's marker on the map in order to read the message. In addition to the user's current location and markers, the home screen shows markers/messages that have been left by other people. These markers appear as blue markers on the map. If the user taps on one of these markers, they can view who dropped the message. Once the user is in range of the message, they can view it by either tapping on the marker or by opening the left panel menu and tapping on "Messages List View". The message list is

essentially the same as any text message application. It has the user's photograph from your contact list, their name/number from your contact list, and a preview of the text message.



Implementation Challenges:

One of the biggest initial implementation challenges we had with our design was making sure our version control was taken care of between the five of us. Mastering the use of git commands and the git terminal was tricky and caused us to lose some time in figuring it out. Even the build in git client inside Android Studio was difficult to use at times. Sometimes lines of code would just disappear from each other's computers so we needed to be sure we had backups at all times.

Another implementation challenge that we faced in our design was making sure the Maps implementation worked to our desired specifications. Luckily there were plenty of guides online that helped us with the use of the API and how to use the various customization options that we had to use. The main problem we ran into was making sure we could keep the unread messages on the screen when a user would tap the screen to lay a message at a desired location. When the screen would refresh it would display a completely blank map with only a pin for the user's

message while the new unread messages would just disappear. Once we got around that problem, most of the implementation became easier to manage.

An aspect we weren't able to get to for this part of the project that we wanted to was getting camera and gallery implementation for laying down messages. To do this we would have needed to design more fragments of the app as well as making sure that this implementation was as easy to use as possible. We would have also needed to make sure we had the permissions coded in properly so that the app would be allowed access to the camera and personal storage of the phone. We also felt that for the purposes of the prototype and the demonstration we would be giving for studio 3, having these features was not the highest priority on our to do list. There were other, more important aspects of the application that we wanted to get to first.

Another part of the application that we did not get to implement was a database to store our messages in. We were planning to use Google Firebase to be our main database for the application. There were two reasons that the Firebase did not get implemented as of yet or at all. The first reason was that it was extremely difficult to get it to work with the time that we had to get other things done for the demonstration. Also since this was part of the backend, so it was not currently necessary for the system to work at for the studio presentation. The second reason was that we found another way to use our app as a messaging service, instead of a backend database we can use the SMS messaging platform that every phone has in its place. This way anything that we need to be done can be done through that without implementing a database so as of this moment the database is not currently needed or it might not be at all. We want to explore both routes further, as the best solution has not been identified.

Another idea that may be implemented would be the public and private view of the map that shows messages in a different way. Users can have an individual map with their friends icons as well as a public map with icons that show public messages in the area. This is an option that we would love to add to the app. Though that may be, it might be added on in a later update of the app in case the important options that need to work do not get done early on in the development of the final product. The main problem that we were aiming to solve with this application was improving interpersonal communication between people, so making sure that we had everything involving messages between two people was the first priority.

Design Justification: "How does this design fit in with our original problem?"

In our design, we sought to provide some solution to mitigate the problem of digital communication lacking the depth and subtleties of interpersonal communication. In other words, we want to provide a solution that closed the gap between digital communication and interpersonal communication. Our design was partially inspired by current social media trends. Applications such as Snapchat have entered this space by making messages and media disappear after a short amount of time. This lessens the gap between digital and personal communication because it makes the digital messages behave more like a face-to-face conversation. When people speak in person, there is no expectation that the information being exchanged will be easily accessible at a later time.

We chose the location based messenger because we felt that it was accomplished the most of our three design mockups. In addition to offering the benefits of giving users the feeling

that they are leaving a personal touch to the message, the location based messenger also offers many of the benefits of the "gamify" model that we proposed. Like our gamify model, the location based messenger app has the capability to create a game-like experience. For instance, users could have a scavenger hunt or challenges to visit certain places etc.

Our studio 2 also influenced our decision. After presenting our three mockups to our peers, we noticed that the gamify idea and location-based messenger received the most interest on piazza as well as in terms of questions. One of the main things that we learned from our studio 2 was that we need to consider the safety of the user when they are using a location based messenger app. We initially planned to allow users to leave public messages as well as friend messages. We decided to hold off on implementing public chat due to the fact that we would need further research on location safety on mobile devices.

In our applications design, we took inspirations from apps such as Snapchat and Pokemon Go in hopes to simulate the feeling of leaving a physical Post-It note for a friend or family member. This is the thought process that we used in order to create a more personalized messaging platform that can help provide more emotions to digital communication.

Response to Studio 3:

As a whole, we received some very positive remarks about our demonstration for studio 3. We also have been given a ton of great ideas about improvements for the interface, and were given good questions for us to think about in further design for this application. We were asked about placing messages in certain private areas such as over the ocean or just restricted public areas. We will be working on implementing a warning like in Pokemon Go to remind people that they should not be wandering into restricted areas or anything of that nature. We would like modify the marker so that a user can only lay a message within a certain radius of their current location. It would be annoying leaving messages for your friends in a location that would be extremely difficult for them to reach. One thing that was brought up to us that we really did not take into account was color blindness. A lot of the app has the same color scheme and the differentiations between different colored pins can be easily missed if a user is colorblind. Therefore, we are going to look into revamping the color scheme and making sure that we are keeping in mind the colors that we are using.

Another idea given to the group by the feedback is on first time use a person will be prompted with information explaining the different icons and how to use the system. This way confusion on how to use the app will be eliminated and the person can jump right into the experience it was designed for. There were also a few comments about potential cluttering of the system if there are a lot of messages in a really small area. This can definitely become a problem so we will look into ways to filter out messages or some way to just switch to the messages you want to see. Also a great idea given to us was that we should look into having the map snap to major points of interest to allow placing of messages at specific locations or buildings without having to rely on the imprecise nature of touchscreen commands. Looking into expiration timers like Snapchat could also help reduce cluttering of a map and help improve the gamification aspect of the design. We were thinking of having a maximum time that messages stay active on the map, but still allowing users to adjust that time to anything smaller than that maximum time.

Technology Justification:

For the development of our prototype, we decided that the best technology we could use for it was to program the prototype as an android application. In general, most of the popular messaging applications in use are primarily based in the mobile platform. This way our prototype fits into that medium and we can get better feedback in the future because we are already developed for a mobile device. As far as coding in Android versus IOS, we decided upon using Android because all of our team members owned Android devices. Google has plenty of tutorials, guides, and free API's to help us add features to our help and help us when things go wrong in the programming process. It didn't take long at all to find the Google Maps API and how to implement it into our application. Android itself is also based off of the Java platform which all of us are very comfortable with so that helped us out immensely. The Android Studio IDE that we used is based off of the IntelliJ platform which we used a lot in our Software Development class. This just helped our flow when coding since we already knew a lot of the keyboard shortcuts necessary to work efficiently. Developing in a familiar environment allowed us as learning developers to understand and focus on the parts of creating a top to bottom application that are new to us. Android Studio also has a built in Git connection that allowed us to keep in sync with what another person was doing with the code. This way we could work on our own different segments of the prototype then bring it all together at the end to make one cohesive product. While tricky to figure out at first, we were able to get it under control and keep our version control on point.