Purdue University

Sprint Retrospective

MonsterChat Application

Team #14:
Rayten Arnold Rex, Cole Baughn, Vishal Gill, Zeyu Pan, Andrew Sytsma, Michael Young

1. What Went Well?

In Sprint 1, we were able to complete most of our user stories mentioned in the Sprint 1 planning document successfully. Things were going a little slow during the first two weeks because there was a steep learning curve for learning iOS development and implementing MCF with Swift 2.0 and for learning the P2P networking framework for Android using sockets. The GUIs were fairly simple for both iOS and Android because Xcode and Android Studio allow for an easy drag and drop way to make them with a little bit of tweaking.

We implemented the following tasks successfully:

- **As a user, I want to be able to use the app on Android phones because I want to use it on my Android phone.**
    - We were able to get the app running on Android devices. The GUI was made, the background was designed, and the P2PManager was implemented for Android devices.
- **As a user, I want to be able to use the app on iOS phones because I want to use it on my iOS phone.**
    - We were able to get the app running on iOS devices. The GUI was made, the background was designed, and the P2PManager was implemented for iOS devices. There were issues in the process of writing the code for iOS devices, so the parts are not linked correctly, but the application works.
- **As a user, I want to be able to view messages sent to me because it will allow me to take part in the conversation.**
    - We were able to display received messages on the GUI. The tools for creating the GUI made it easy to display the messages.
- **As a user, I want to be able to see messages that I have sent because it will allow me to better keep track of the on-going conversation.**
    - We were able to display sent messages on the GUI. The tools for creating the GUI made it easy to display the messages.
- **As a user, I literally need emoticons because I want to express emotions in a picture format.**
    - We were able to use and display emoticons. We didn't realize that emoticons were already handled by the OS, so this was already finished for us.
- **As a user, I want to be able to send and receive messages with other iOS devices because I want to communicate with others.**
    - We were able to send and receive messages between multiple iOS devices. We had to convert the code from previous iOS versions to iOS 9 because there was no tutorial or practical info on how to use the MCF. Once the code could compile it was hard to test because the iPhone could not interact with the simulator so we used two simulators instead.

- **As a user, I want to be able to send and receive messages with other Android devices because I want to communicate with others.**
    - We were able to send and receive messages between multiple Android devices. It was a little difficult getting everything setup because the networking was complicated to code, and some things were not working for weird reasons.

2. What Did Not Go Well?

Although we were able to implement most of our user stories, still there were a few tasks that were unsuccessful. Originally iOS and Android were supposed to have the backround implemented in C++ then wrapped with Swift and Java respectively. Once we looked into wrapping we found out it would exclude some functionality and take a longer time to run because it would have to convert instructions back and forth from the native languages. So we abandoned it all together and decided it would be much easier and a more efficient use of our time if we implemented the background in the native languages. Because it took a long time to figure out communicating with MCF we did not have enough time to combine the background written in Swift with the MCF and GUI.

We implemented all user stories successfully except this one:

- **As a developer, I want to use background code to connect the GUI and the network without rewriting code because rewriting code is a bad practice and takes more time, and the network and the GUI should not have to interact.**
    - We were doing some research after we started this sprint, and we found that using C++ as a background was impractical because it would involve a lot of conversions leading to double coding, and it would actually make our application run slower due to having to make function calls from a non-native languages.

3. What Should Be Improved?

During Sprint 2, there are ways we can work to improve ourselves as a group and as individuals. We need to have an easier method for letting other group members know what is currently finished in the project. We also need to work on finishing tasks faster. Also, we need to working on using github better.

- To fix the issue of not knowing when tasks are finished, we can use a trello board to mark off what is completed and commit to github more often.
- To fix the issue of not finishing tasks on time, we can work on tasks at an earlier time to allow extra time for unexpected events.
- When using github, we need to do more branching to help avoid possible issues with multiple commits.