Sprint 1 Retrospective Document

Alarm^2

Scott Walters, Ashwin Chidanand, Kalpan Jasani, John Redmon

What went well during the last Sprint?

- User Story #1: As a user, I would like to set an alarm based on time, so that I can be woken like how traditional alarms wake us up

We clearly understood the architecture of the API available to android developers by going over the documentation. This initial learning curve will hopefully help in the project in the future and even further on in software engineering. For example, team members clearly understood the concept of using XML files in your code, and using manifests and intents.

One thing that went well was making the API between the application and the service. First of all there is no service, but system calls instead. We made a class that could be a wrapper class for many system calls. This sprint, we established the time alarms, and hence such alarm scheduling is implemented in our application. We felt that making such an API was worth the small inexpensive time invested, providing clean code and easier management of it.

1. We can easily add more code to the class that provides the API. It is called MyAlarmManager.
2. Debugging and changing class names is easy. For example, if we develop a new scheme for alarm objects, we can rename the type of object we are taking as parameters in some of the functions in the MyAlarmManager class.

- User Story #2: As a developer, I would like to verify the user's number using an automatic verification code so that people don't misuse the application by adding wrong numbers

This user story was completed. This user story was not incredibly difficult to implement but instead was challenging in the testing phase. We used a Hint Picker to allow the user to select their phone number and ran a verification method to see if the number was registered. This was difficult to test because the test phone did not have a phone number so we had to try other phones from other people.

- User Story #5: As a user, I want to set default text and voice messages for new alarms so that I do not have to create a custom message/recorded message for every new alarm

This user story was essentially completed. The default text and voice messages are created at the start of the activity. They can be deleted and/or edited by the user. They are stored in seperate arraylists and displayed using a listView. They can be chosen, and the logic is in place for the values to be sent to the receiving activity, but we did not actually connect the recipient activity in this sprint.

- User Story #7: As a user, I would like to be able to set custom text messages so that others can get notified with this message if I fail to manually stop my alarm.

This user story was completed and extended. Text messages can be dynamically added, deleted and edited. They are stored in an arraylist of strings displayed using a listView. Custom voice messages were also added. They are stored in an arraylist of custom objects that hold a string of the display name and the audio file. These are also displayed using a listView.

- User Story #9: As a developer, I would like to set up a server that handles communication

This user story was completed. A server has been set up to handle REST calls from the client, create, update, delete, and get. Although there was a big learning curve, as we did not have experience with servers in the past, the Flask framework turned out to be a good choice. As the app is in its early stages, constant updates to server requests will be necessary. With the basic requests defined and working, it will be simple to add and modify the server code to better handle our client's needs as the app becomes more developed.

What didn't go well during the last Sprint?

The biggest issue we had was having to drop a user story completely. Much of our time was spent learning the API of Android Studio, more so than we had taken into account. There were portions of user stories that suffered because of how much time was spent learning the API. We had noted this would be a potential issue in the sprint planning document, but even though we kept it in mind it still overcame us.

We also definitely need to spend more time on testing. Since there was a time crunch, and lack of experience with testing, we did not have testing.

- User Story #1: As a user, I would like to set an alarm based on time, so that I can be woken like how traditional alarms wake us up

    Task: Pull in default ringtones from OS

There are a few parts, for which the method to do them was figured, but they were not yet implemented. These are:

- Displaying a screen to select a contact
- Displaying a screen for selecting a phone's ringtone (external ringtones not a part of this application)

A good point to note here is that many of these are interrelated, and knowing one makes it easier to implement the others. Hence, it is vital that if one team-member comes to know about it, he takes the responsibility to advise other teammates.

- User Story #4: As a user, I would like to set a custom ringtone

One team member found it tough to understand the variety of GUI "widgets" and "containers" that could be used. On retrospect, we feel aligning similar tasks, such as GUI handling, to one or two people, would help as they would know about it more. Or at the least, we can teach each other quickly if we need to do that part.

- User Story #6: As a user, I would like to be able to record my own voice message so that others can be notified with a call if I fail to manually stop my alarm.

This user story was written without us being familiar with the workings of Android Studio. Upon building a voice recorder interface, we decided it would be more beneficial for the user to record their voice directly and have the file automatically selected rather than jumping through multiple screens to select a pre-existing file. Additionally, we chose not to trim a file to 45 seconds because it seemed unlikely a recorded message would last that long and if it did, there was probably significance to the message. As such, this user story was completed but it does not match the descriptions and acceptance criteria given. We consider this story complete but think the transmutation of its original implementation should be noted.

- User Story #8: As a user, I would like to be able to set a contact who would be called to wake me up so that in case I do not wake up or sleep past my alarm I will have a safety net

This user story is incomplete. This is a functionality that will moved to the sprint 2 tasks, but due to underestimation of time to complete tasks, we did not reach this task.

How should the team improve for the next Sprint?

One of our issues was with our initial planning. We split individual user stories amongst our whole group as well as underestimated the time required to complete several of our tasks. The hours were split evenly among the members, but the tasks were not divided efficiently. Throughout the sprint we often ended up trading tasks to better fit what we we're already working on. In a few instances this resulted in a loss of time.

- A good point to remember is to have cross communication between team-members, where one person's knowledge might make it easier for him to do or contribute to other member's task.
- Having more reliability of the importance of certain user stories. In some cases, our user stories become invalid if the circumstances are different. This happens when we understood that a better option than a background service is to use the phone's scheduling services. Also, instead of trimming a file to 45 seconds, we now adopt a different approach.
- We can definitely look into formal methodology of testing. This is done in the same sprint with small increments in running code. This can guarantee robustness and quick check-up until it is late.