Beacon-Reflection Statement

Neema Boutorabi	***18694142***	***Luigi Sacco***	***21139143***
Andre Tertzakian	***17682148***	***Emre Pekel***	***48488134***
Alexander Ford	***18361148***	***Omar Dzick***	***36579143***
Konrad Izykowski	***26506148***		

Learning Experiences

Among the many things that our team has learnt over the course of these 4 months, this project has provided us all with a solid introduction to Android platform development, software management, and quality control. Our team members feel that undertaking a large software project such as Beacon has made us more confident in designing, maintaining and revising user-centred applications.

We have familiarized ourselves with the SCRUM development model and its key features, these include:

- A product backlog that outlines what needs to be done, in what order it should be done, and how long it will take to do it.
- Frequent inspections that help determine how close the development team is to achieving a sprint milestone/goal.
- Organized sprints that help delegate responsibilities and set appropriate milestones.

We found using the SCRUM software design methodology to be an efficient way of defining and delegating tasks to our group members. SCRUM's flexibility also allowed us to continually re-assess our requirements and use cases as we better understood the demands of our key demographics.

We also discovered the importance of adhering to standard software design patterns. We learnt that disregarding design principles in favor of quick solutions often lead to smelly, hard to read code that was very difficult to maintain and add new features to.

Certainly in the future we will more rigorously apply good design principles in the early stages of development as it often pays off in the long term, by creating easier to manage code that requires far less refactoring.

Although all of us understood the importance of testing code going into this project, we often looked at it as having less priority than it should have. We wrote unit tests for our data utility methods and used Espresso tests for some of our UI implementations. These tests, however, were not always run after implementations were added to the source code (we did not always use regression testing methods). Had we tested on every change, we may have been able to catch more bugs before they proliferated. In the future, we will give testing as much, if not more, priority in a project of this size; we would even consider assigning someone's role to be primarily for testing.

Challenges

Our team agrees that despite the flexibility and versatility that SCRUM is able to provide, the most time-consuming part of our project was adapting our implementation to match changing requirements. This process led to the introduction of a lot of unnecessary bugs and a general sense that despite a growing code base, our project was not reaching a state in which it felt presentable. In fact, at a certain point, we had two different versions of the project developed independently by two different team members. The key issue here was that a subtle modification to our project requirements was not communicated effectively to all our team members and hence led to an uncoordinated effort.

Furthermore, it has been a challenge for us to integrate the work of each team member at certain times during the project. As the work was divided, and everyone was focused on implementing the features they were assigned, we failed to continuously integrating our code. This often resulted in tiresome conflict resolutions, and the introduction of new bugs. Even though we eventually handled these situations successfully, it would have been less painful and less time-consuming if we merged code more often, and put greater emphasis on creating integration tests for every new feature.