Reflection Statement

Fridge Manager

Authors: Xuewei (Vivy) Wang, Kazuki Fukushima, Jeffrey Chen, Raphael Strebel, Jason Wang, Sangeetha Kamath

---

The moment our group decided to work on Fridge Manager, we were aware of the fact that time was going to be one of our biggest challenges. However, when it came to determining the size of our scope, we lacked experience in developing a mobile application so we weren't able to accurately make a prediction on the different tasks that the project could be divided up into. Based on each group member's interest and strength, we divided our groups into three major sub teams: one deals with algorithm, one deals with database, and the other deals with UI design. Nevertheless, if we had done more research and consulted with more expertises in the field, we would know better in terms of the development cycle of mobile application, and could have done a better job in dividing the project into different tasks.

Our first planning document which included backlogs and sprints was ineffective because the time we thought it would take to complete each task was largely underestimated. In retrospect, although our prediction in the end was still an underestimation (even after iterations of improving our prediction), the project itself helped us understand what really goes into developing a mobile application and the time it takes. Furthermore, following a SCRUM setup in our planning document also helped kept our group on track by having frequent meetings and we were able to prioritize the features that were essential to our application.

Another flaw in our planning is that we should have set up more time for refactoring, and budgeted more time for it. In our original plans we did not set aside any time at all for testing or refactoring, only setting time for developing features. As a result, most of our time was spent on developing features and not enough was spent on refactoring the code and implementing what we learned about design patterns. The code would have been a lot cleaner and there would be a lot less redundant code if we had spent more time on implementing design patterns such as the Strategy pattern.

In terms of testing and validation, we did not have any automated tests in the beginning of the development phases. After we added in a new features, we manually tested it by loading the application onto an Android device and see if the application respond in a way that we expect it to be. However, we later realized that this is too time consuming and decided to add in more automated tests. We implemented many unit tests for important methods inside each class. Moreover, we added instrumented UI tests as integration tests to further reduce the time spent on manual testings.

And while we had non-automated testings and minimal automated testings after developing features, we basically completed each feature without a complete automated testing before moving on to a different feature. We realized towards the end that especially for larger

projects, testings and refactoring should be done immediately as soon as feature has been completed. It is more simple to create automated tests earlier in the project than later in the project.

All in all, our group learned that for future software projects, strictly following software development model discussed in the lecture will provide a smoother and a more transparent progress on the project. Additionally, we will spend more time to do research on the different aspects of the project, and we can now apply our experience to better predict the time required for different aspects of the project. We also understand the importance of having a separate task for refactoring and testing, which will make more for an accurate plan.