**WithU**

**----------------------------------**

1. **What were the main difficulties so far?**

First and foremost the main difficulty for everyone has been having to adjust to new languages/API’s that we have never worked with before. Yes the tutorials did help but they can only do so much and the rest has been a lot of just learning from mistakes. As for the backend the main difficulty so far has been dealing with Android’s Asynchronous Loading. We wanted to design our model to consist of service classes where the activities can call methods that will read/write data to Firebase. The problem with this was for many of our read calls to the database we needed to return the objects to the classes calling these service class methods which is difficult with asynchronous loading. That being said we have managed to overcome this after spending a lot of time looking into it and have decided to implement interfaces to do callbacks to manipulate the data from reads the way we want. For the front end of the application learning more about the google maps API was a challenge. We wanted to integrate directions into our application, but as we started to look deeper into the documentation you are only able to do that through the web services. Another problem we encountered was with android studio emulator showing the map in our application. Most of the time the map didn’t show up when we were testing it. Another difficulty the group encountered was working with github. Even though we all did the git homework assignment, setting up and working with our actual project was difficult. Some team members encountered problems, but as time went on the team got more comfortable working with the version control software.

1. **Were there any features you did not implement as planned, and why? Are you pushing some features to later iterations, and if so, why?**

Originally when first designing the iteration plan we had hoped to try to get the full request walk and actual walk progress up until they reach the destination all in iteration 1 done. However, due to the bigger than expected learning curves of android development/Google Maps API/ Firebase we had to tame those high expectations and instead focused on getting the login/register setup with full Firebase Authentication, as well as doing everything up until the point of actually creating the walk. We also ran into some difficulty when trying to pair two walkers. We have decided to push pairing back until iteration 2 as well, since this is the backbone to walkers even being able to get requests. This way in iteration 2 we can focus solely on the main walk process complete with all of its accessory features. We see that main walk progress as being the most complicated given the constant location retrieval/updating in the backend, on top of having to work closely with Google Maps API and notification features so it would be best to push that off until iteration 2 when we can focus all of our attention on that.

1. **What tests did you prepare for this iteration, and what are they covering? Did the tests you wrote deviate from your plan? What features are you not testing yet? Did you use any test frameworks, such as JUnit, the Android Monkey, Selenium, etc.?**Our test suite follows the Testing Pyramid strategy recommended by Android and includes three layers of small, medium and large tests.  
     
     
     
     
      
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   
   1. **Unit Tests**  
      These are low fidelity tests that run quickly on the local JVM, test small logical blocks of code in isolation and form the bulk of the test-suite. These local unit tests are used for testing the logic in Services and Presenters.  
      These tests have no dependencies on the Android Framework and we use Mockito and Powermock to mock dependencies on third party libraries such as Firebase. These are Junit tests running on junit4.PowerMockRunner.
   2. **Integration Tests**  
      These tests sit in between small and large tests and test how multiple app components interact, but they don't test the full app. These are mostly hermetic UI tests on Activities and Views that simulate the behavior of external dependencies but use real-world-like data to drives the app UI.  
      These are also Junit tests running on Roboelectric, which provides testing-friendly, Java-based logic stubs that emulate the Android framework. These tests provide the fidelity of running tests on an Android device while still executing more quickly than tests run on actual fully-instrumented emulators or devices.
   3. **UI Tests**  
      These are large cross-cutting tests that run on emulators and real devices and test common workflows through the UI across all layers of the application and evaluate the application as a whole.   
      These tests will be included in the next iteration of the application. We will be using Espresso testing framework and AndroidJUnitRunner test runner for these.

All the current tests can be run using the following commands

|  |
| --- |
| $ ./gradlew **clean** **test** |

The testing strategy is closely aligned with our proposed design plan. We made testing and sensible code coverage a high priority in this iteration and by running and maintaining these tests consistently, we ensure the correctness, functional behavior, and usability of our application at all times. The rapid feedback on failure allows for early detection of issues and ensures peace of mind during refactoring.

1. **Optionally give a URL and instructions for using your application in the current stage. This makes sense for purely web-based projects, but it may be impractical for projects that must be installed on a client device.**

See ***README.md*** in GitHub repository. We have sent email invitation to give two TAs access to our GitHub repository ***spatterson4/WithU***(unfortunately, we can not find Professor Tracy Lewis-Williams). If you have problem accessing the repository, please contact Sam([spatterson4@wisc.edu](mailto:spatterson4@wisc.edu)).