Software Implementation and Testing Document

For

Group PuzzAlarm

Version 3.0

Authors:
Devin Moure
Ryan Gutierrez
Zach Gutierrez

1. Programming Languages (5 points)

List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

The PuzzAlarm project is an Android app written in Java using the Android SDK. We chose to use Java over Kotlin because everyone in our group is familiar with Java given Kotlins level of popularity compared to Java, Kotlin lacks a great amount of community support.

2. Platforms, APIs, Databases, and other technologies used (5 points)

List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

Our project uses many components that come from Android Jetpack, which focuses on providing components that follow best practices, free from writing boilerplate code, and simplifying tasks.

3. Execution-based Functional Testing (10 points)

*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

Functional Requirement 1:

- We tested the first functional requirement by actually adding an alarm to the app and waiting. Once the time of the alarm and the time of day matched up we waited for a ring.

FR 2:

- We ensured that the testing game started on the alarm wake and that the alarm noise continued until the game was completed.

FR 3:

- Only half done: We have yet to implement the correct functionality if the puzzle is not solved. To test puzzle solution we just did the relatively simple math required to solve it.

FR4:

- Tested through creating alarms, deleting them and ensuring they were removed from internal storage.

FR5:

- Tested by selecting a puzzle and ensuring the puzzle selection persisted through configuration changes. Then ensuring that the selected puzzle was presented when an alarm rang.

FR6:

- Just ensured that the Text View persisted through configuration changes.

4. Execution-based Non-Functional Testing (10 points)

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

Non-Functional testing was performed using brute-force, rigorous testing through what would be regular use of the app by a common user. In other words, we set up various alarms to ensure their accuracy. Persistence was tested up until it could have been, it functioned properly with the app in the background but not with the app closed. Apparently google considers apps that run things while closed, malware.

5. Non-Execution-based Testing (10 points)

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

When running into issues, we would take a break and come back to walk through the code. For example, when we first began we implemented the RecyclerView inside the MainActivity class without having a good understanding of it, and it was giving a NullPtr Error. After we revisited the

issue, it was very clear what the issue was and we fixed it by moving the RecyclerView into the alarm fragment class. We also often give each other walkthroughs of what we have recently implemented whenever we have a chance to meet. We tested the internal storage implementation non-executional-y by looking at the file that gets alarm data stored in it after execution.