Software Requirements and Design Document

For

Group PuzzAlarm

Version 3.0

Authors:
Devin Moure
Zach Gutierrez
Ryan Gutierrez

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

Our system consists of an Android application created using the native Android SDK with Java. The application's intention is to provide users with a way to wake up in an interactive way that will in turn, make them more productive. The interface is made up of two pages: the home page and the alarms page. The home page greets the user upon logging into the application, allowing the user to choose between two possible puzzles when waking up. From the home page the user can go to the alarms page to set an alarm. When the alarm rings, the user can only turn it off by solving the game or the puzzle presented to them.
Our target audience is anyone with an Android phone that currently has trouble waking up in the mornings. Some of our target audience may even be experiencing turning their alarms off or snoozing them while still half-asleep and they end up oversleeping. This is the perfect market for this application as this solves the problem by forcing the user to be cognitive enough to solve a puzzle before the alarm turns off.

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

High: 1-4
Medium: 5
Low: 6

1. Allow users to store Alarms that will wake and ring the device at chosen time.
2. Open game or puzzle on alarm device wake, while the device is ringing.
3. Turn off alarm if user solves puzzle, if user does not solve puzzle give them a new puzzle or give them more chances, depending on what puzzle the user is solving.
4. Allow users to delete alarms.
5. Home screen gives puzzle options to select from.
6. Home screen greets users.

3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

Persistence - Alarms need to ring even if the app is closed or screen is locked.

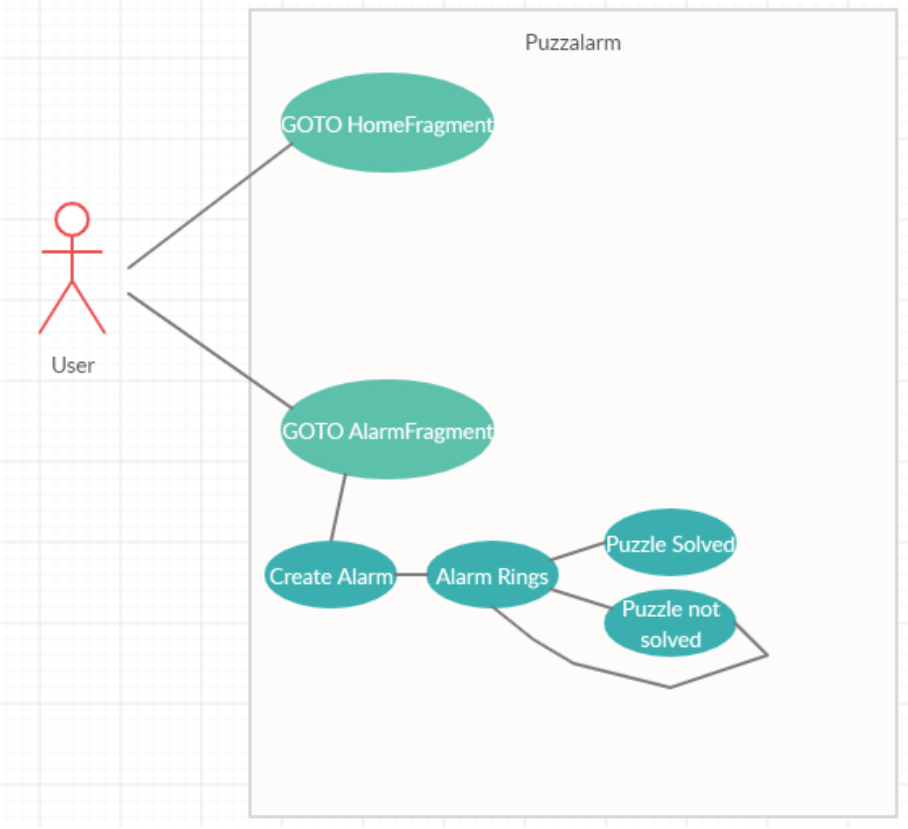
Reliability - Alarm needs to ring at the user's set time.

Software Quality - App needs to function with minimal bugs

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



5. Class Diagram and/or Sequence Diagrams (15 points)

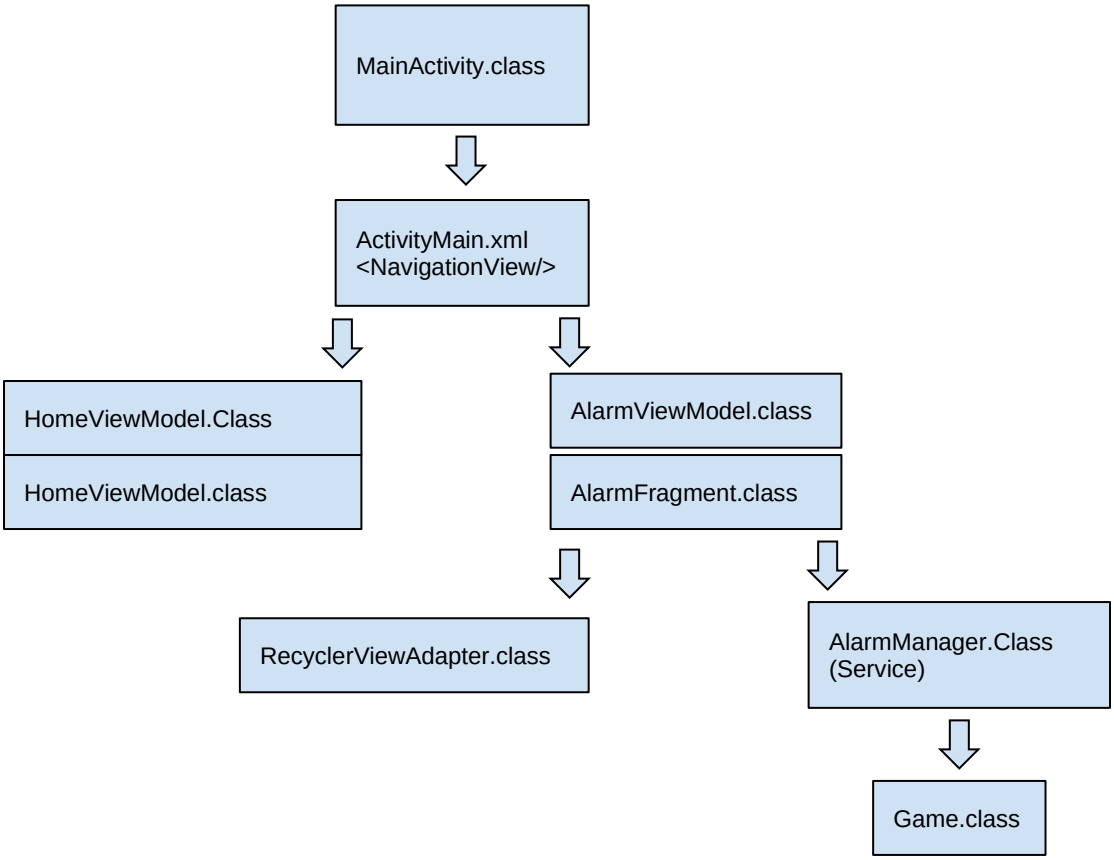
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*



6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

The application will run on any Android Device running at least Android 8.0 API level 26. This app will run on unreleased Android 11. The app consists of two artifacts, the home screen and the alarm screen that holds a list of the alarms. The home screen is the landing page when the user opens the application and shows a greeting along with the game options. The two screens communicate with each other to ensure persistence of the chosen game, the NavigationView XML component in ActivityMain.xml handles the navigation logic.

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

Our project is fully dependent on the Android platform. Most components in the application are from the Android API or from a Java utility package. The only artifact of PuzzAlarm that would have relied on a third party service was the call to a weather API which we decided to forego.