Design Document

**Classes**

* **AlarmClock:**  Main class that listens to user UI event changes to and updates the alarm database and creates the correct alarm instance (Alarm, FailSafeAlarm, ChallengeAlarm)
* **SetAlarm:** Listens to AlarmClock for user’s input and inserts alarm settings into database. Also remove and fetches alarm records. Superclass of AlarmEdit.
* **AlarmEdit:**  Modifies the settings of any existing alarm. Extends the SetAlarm class
* **Alarm:**  Each instance indicates an alarm schedule with no special mode attached to it. Also becomes the superclass for FailSafeAlarm and ChallengeAlarm.
* **FailSafeAlarm:**  An instance of the alarm with fail safe mode enabled. Extends the Alarm class and includes a

snooze counter. When fail safe is active, intialize ActiveAlarm with the ring duration user has

* **ChallengeAlarm:** Extends the Alarm class and calls WakeUpChallenge to generate the UI to the algorithm

problem. Simultaneously, ActiveAlarm is called to generate the alarm sound and will listen to the WakeUpAlarm if user answer’s question correctly.

* **ActiveAlarm:** Turns on the alarm ring or vibrate. Called by Alarm, FailSafeAlarm and WakeUpAlarm.
* **WakeUpChallenge:** Randomly generate a math problem with respect to the difficulty level passed through with parameter values.

The following diagram depicts the relationship between each class and some of the operations specific to the classes’ functionality.

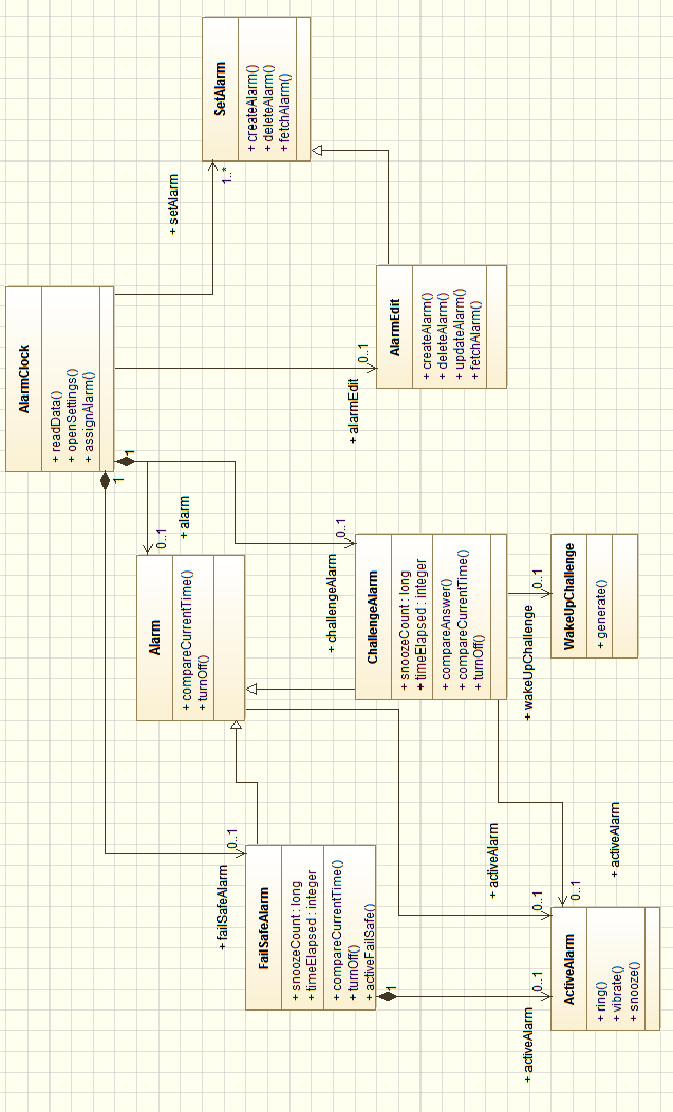
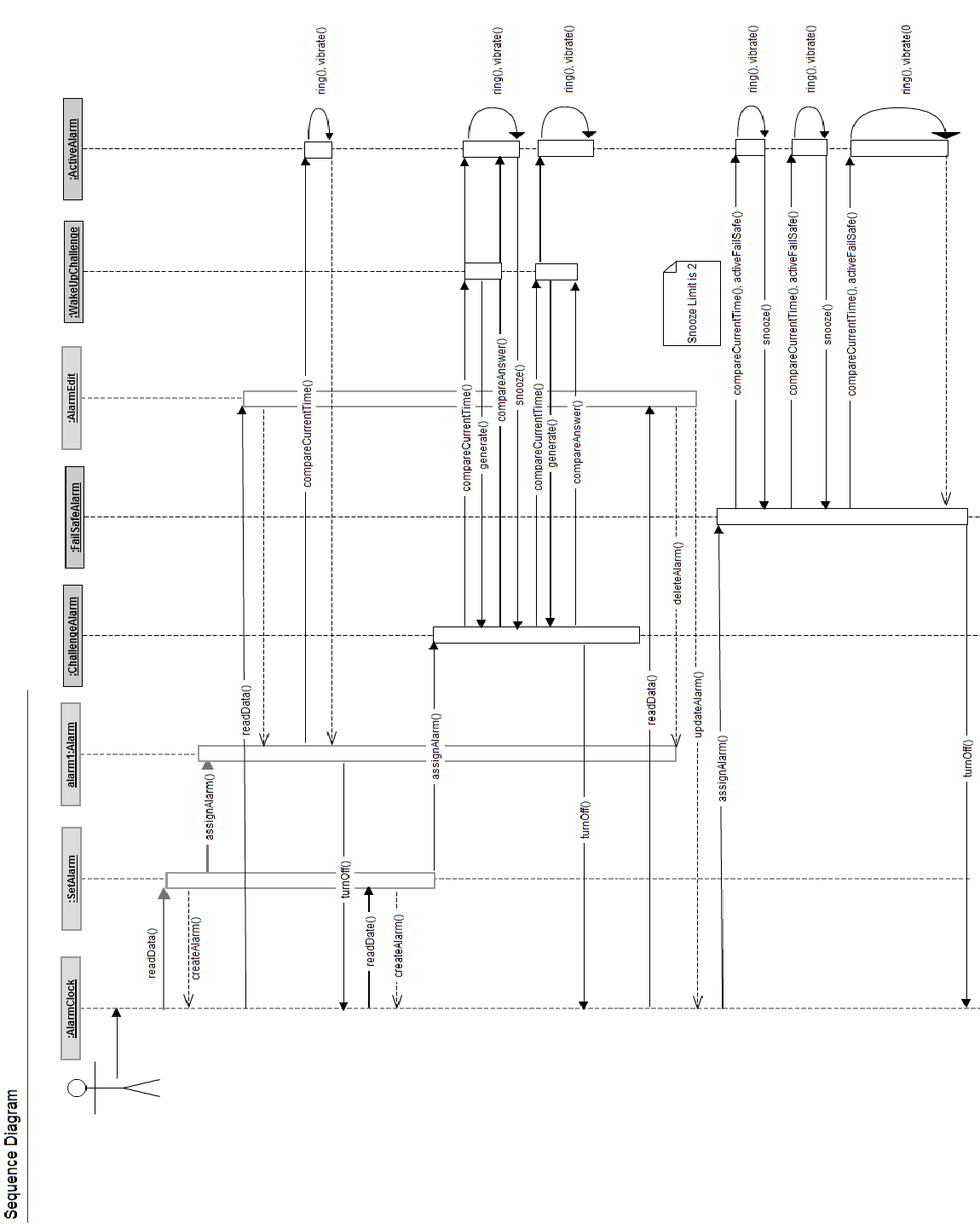


Diagram 1: Class diagram of the alarm clock app



**Sytem Callbacks/Events**

* Trigger the alarm as scheduled in the database. Event system; checking a bulk of scheduled alarms.
* Notification of any overlapping alarms. (Event)
* Listen for alarm dismissal or snooze for a native alarm setting (System)
* Listen for snooze in failsafe mode; trigger FailSafe to be active when snooze count hits 0; listen for FailSafe to expire
* Trigger Wake-Up mode and generate challenge UI; routinely if hit snooze after each challenge solved. Also listen for time expiration mark to close Wake-Up mode.

**Data Management**

Will utilize the Android library to store alarm settings in SQLite database. It will include:

* **import android.database.Cursor;**
* This interface provides random read-write access to the result set returned by a database query.
* **import android.database.SQLException;**
* An exception that indicates there was an error with SQL parsing or execution.

* **import android.database.sqlite.SQLiteDatabase;**
* Exposes methods to manage a SQLite database
* **import android.database.sqlite.SQLiteOpenHelper;**
* A helper class to help manage database creation and version management

Storage will not be heavy as each data row will consist of these settings for each alarm schedule

* name
* time
* date
* repeat day
* snooze limit
* challenge difficulty level
* alarm duration
* vibrate?
* wake-up challenge on?
* Failsafe on?
* Alarm on?

**User Interface**

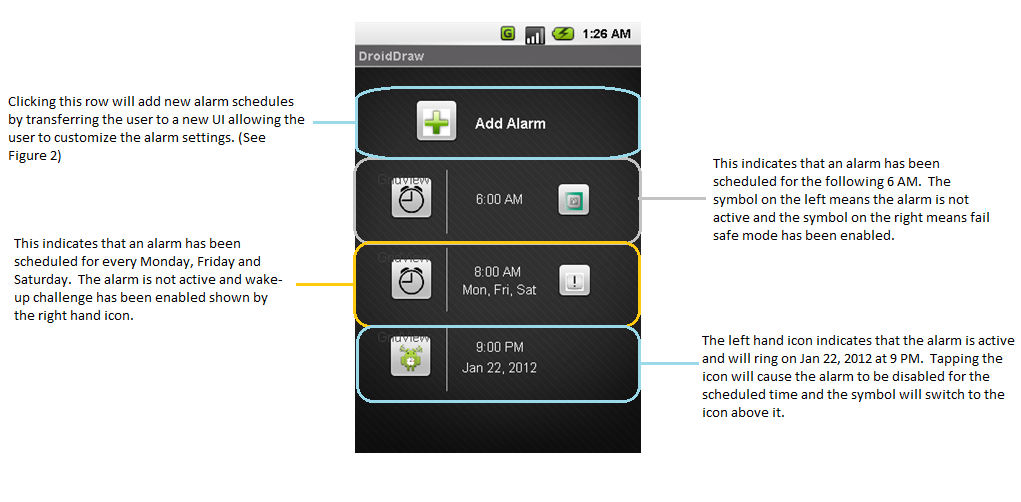
****

Figure 1: Main Menu of alarm app

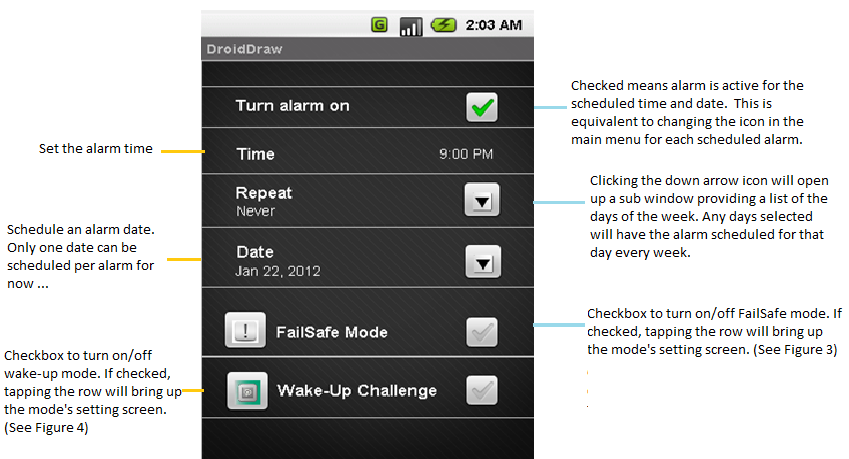


Figure 2: Alarm settings. User can access this screen either by adding an alarm

or modifying an existing alarm.



Figure 3: FailSafe mode settings.

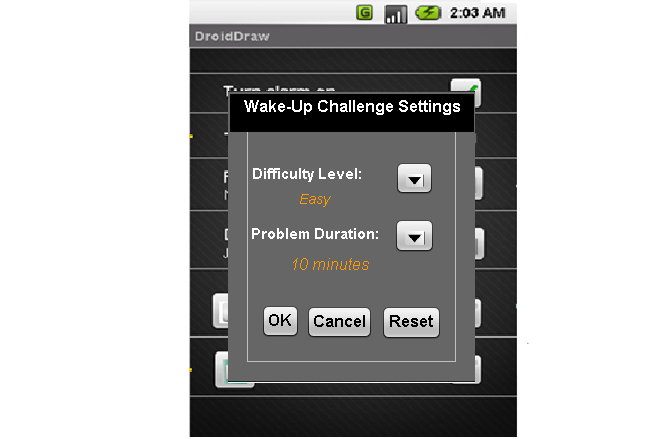


Figure 4: Wake-Up Challenge Settings

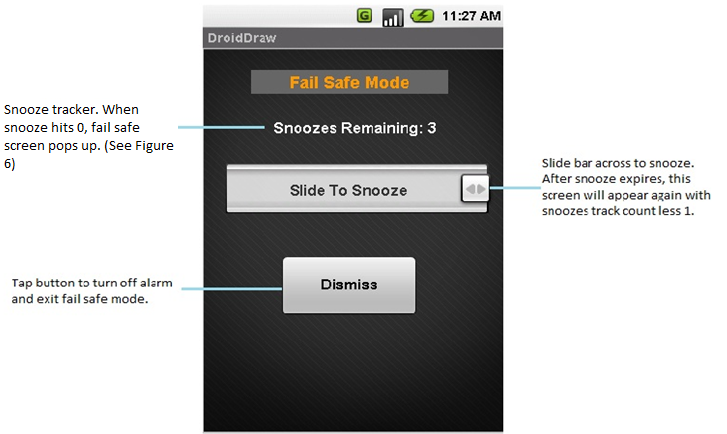


Figure 5: Alarm with Fail Safe Mode on.

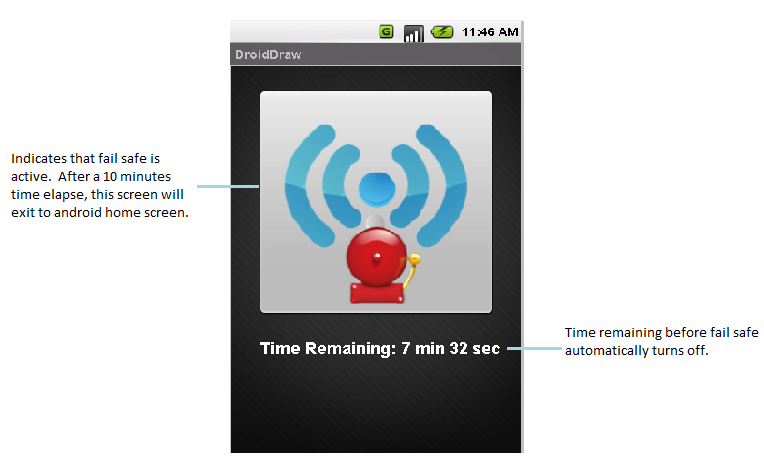


Figure 6: Fail Safe Active.

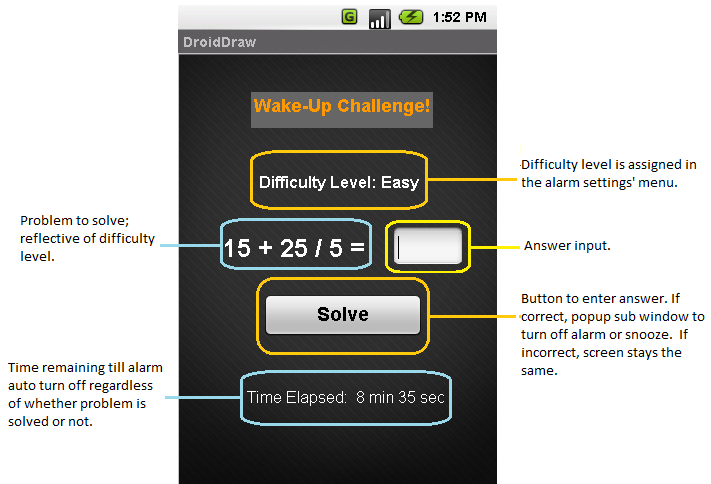


Figure 7: Wake-Up Challenge Active

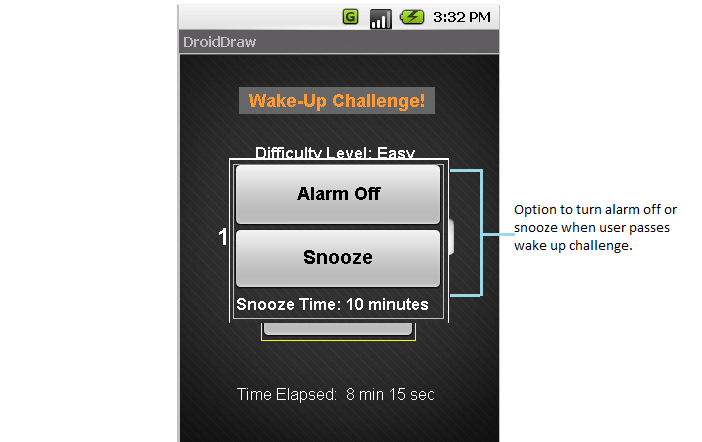


Figure 8: Wake-Up Challenge solved