



# Good Morning

## Wake Up Assistance Application

SENG 371     Luke McLaren   Rahat Mahbub  
Assignment 3a   Adam Kroon   Graeme Bates

## Table of Contents

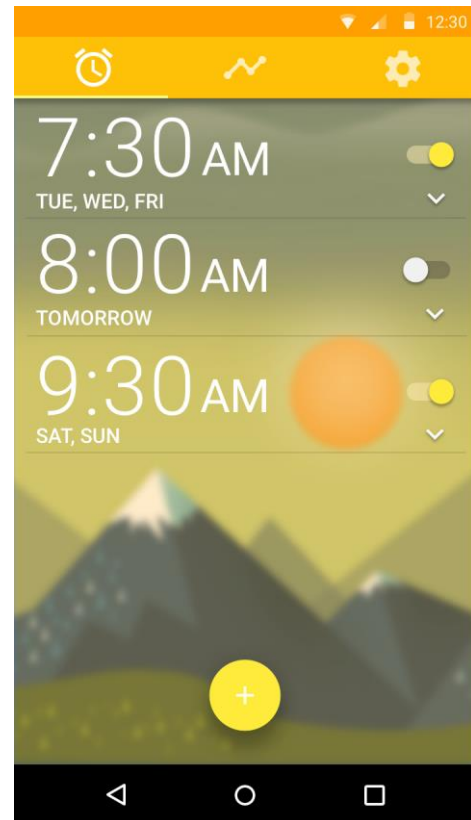
Executive Summary.....	3
Project Summary .....	4
Core Alarm .....	5
Speech Synthesis.....	5
LIFX Smart Lights .....	5
Google Calendar API .....	6
SQLite Database Infrastructure .....	6
Application Testing Schedule and Framework.....	6
User Interaction.....	7
Application Installation .....	7
Alarm Creation.....	8
Alarm Deletion.....	11
Alarm Snooze/Dismissal .....	13
Alarm Notifications.....	15
Alarm Settings.....	17
Activities.....	21
Fragments .....	21
Activity Views & Composition.....	23
Main Activity View & Composition .....	24
Alarm Configuration Activity View & Composition .....	24
Sleep Statistics Activity View & Composition.....	25
Settings Activity View & Composition .....	26
Alarm Activity View & Composition .....	26
Fragments.....	27
Navigation fragment.....	28
Information Display fragment .....	28
Configuration fragment.....	29
Confirmation fragment .....	29
Alarm Interaction fragment .....	30
Virtual Assistant fragment .....	31

Management Plan.....	32
Core Alarm .....	32
Google Calendar.....	33
Speech Synthesis.....	33
Expanded System .....	34
Relationship Between Features.....	35
Technical Specifications - SQLite Introduction .....	37
Database Operations.....	37
Database Creation .....	37
Table Creation.....	37
Fetching Data.....	38
Database Helper Class .....	38
Async Helper Class .....	38
Entity Framework .....	39
Database Schema .....	39
Content Providers and URIs.....	41
Content Provider Insert, Update and Delete .....	43
Alarm Service .....	47
Testing.....	59
Testing Schedule .....	59
Unit Testing Schedule .....	59
Functional Testing Schedule .....	60
Integration Testing Schedule .....	60
Performance Testing Schedule.....	60
Acceptability Testing Schedule .....	60
JUnit Tests.....	61
Espresso.....	62
Monkey.....	63
Walkthrough Functional Test Cases .....	63
Team Planning.....	72
References.....	73

## Executive Summary

LARGE Software developers have been given the task of developing a mobile application that intends to assist users in waking up efficiently and gracefully in the morning. This application is called Good Morning. The SoftStart development group has provided LARGE Software with a Detailed Request for Proposal for this application. In this document, SoftStart outlines objectives, constraints and known interactions that they are looking to see in a successful prototype demonstration before the end of March 2016.

This Detailed Request for Proposal has served LARGE developers as a guideline to begin development of the Good Morning application. Given the timeline, and known objectives as well as constraints provided for the application, LARGE developers have been able to identify and research topics that are relevant to developing a minimum application system before the end of March. These topics would include a core alarm, smart assistant, machine learning, and smart technology integration. From this research, developers have been able to begin mock-ups on user interface design and interaction. LARGE developers have also created a Management Plan for the Good Morning application. In this plan, a finalized minimal system has been identified, which will meet SoftStart's specific needs. This plan discusses all known relationships between the application and APIs that will be used during development. Finally this Management Plan outlines which developers will be completing specific tasks in order to deploy a working application before the deadline at the end of March 2016.



Technical specifications have been completed to give the clients an in-depth knowledge for both the UI and database for the Good Morning application. This knowledge demonstrates how key, agreed upon requirements for this application will be developed for the final demonstration in late March. Finally, an outline of the comprehensive testing schedule for this application has been created. Application testing frameworks such as Espresso and Monkey will be used to ensure that the user interface seamlessly interacts with the application database to provide users with a robust experience overall.

## Project Summary

The Good Morning application project was proposed by the SoftStart project team. The main functionality of this application is assisting users (a targeted group of young adults or professionals between the ages of 18 and 32) in the process of waking up in the Morning. To do so, this application aims to utilize the following technologies (as outlined by SoftStart in their Detailed RFP):

- **Core Alarm** - a system through which the user's mobile device will reliably and gracefully assist the user in waking up.
- **Smart Assistant** - a summarization of additional information that the user utilizes on their mobile device. Such information would be comprised of missed or unseen instant messages, emails and phone calls as well as notifications on upcoming daily calendar events or weather information for the upcoming day.
- **Machine Learning** – the application should be able to learn from user habits, such as sleep patterns, to be able to adapt or better assist the user throughout the process of waking up through the morning.
- **Smart Technology Integration** – ideally the Good Morning application will allow users to setup and control smart devices such as the *LIFX* wakeup light to assist the process of gracefully waking up in the morning. Integration of smart watches in the application will allow for use of accelerometers on additional devices to detect waking time during REM sleep cycles.

Along with the aforementioned utilized technologies, the Good Morning application will meet the following development constraints in order to meet the prototype demonstration at the end of March:

- The application will be available for users to test for mobile devices that use Android operating systems
- Prototype will be made available in English as a primary language with capability to expand to multiple languages
- The ability to interact with various types of smart technology (Prototype will specifically interact with the *LIFX* wakeup light)

In order to accomplish these tasks the Good Morning application will have to accurately, efficiently and securely utilize APIs of existing applications or technologies. *LARGE* Software developers have completed background research in these fields to provide a framework for the development of the Good Morning application.

## Core Alarm

- Market currently saturated with alarm applications available publicly on Github
- Most applications not compatible with Android V4.4 and above
- Researching code base of current highly used applications will allow LARGE developers to focus on refining the UI experience of the alarm
- Developers are looking to potentially integrate the SensorManager API to allow devices to anticipate when to implement an alarm at an appropriate point during a REM sleep cycle

## Speech Synthesis

- Voice Notify will be a primary example of how to construct an application that properly synthesizes speech
- Main developer goal is to create an application that requires little to no interaction from the user at first, utilizing speech synthesis to read off displayed information
- Speech synthesis will be synchronized with alarms to ensure that information is given in a slow, steady stream. This will allow for maximum comprehension on the user's end.

## LIFX Smart Lights

- LIFX application sets limits to the user by limiting the control of a light's color or brightness, as well as whether or not it is on.
- Good Morning aims to tap into this technology to allow the user to control when and how the light is turned on
- *SoftStart* has stated that a graceful transition to waking up is preferred. This would mean not abruptly turning on lights, but a smooth and slow transition.
- LIFX API is made public, which will allow LARGE developers to apply the aforementioned effects to smart lights used by clients



## Google Calendar API

- Google is the standard application base for all Android operating system phones
- LARGE developers main goal is to be able to hook into Google Calendar API to summarize upcoming events in the user's day
- Codebase for Google Calendar is widely used through many applications whose code is made available publicly online

With the provided RFP for the Good Morning application from *SoftStart*, the Software Developers at LARGE Software Inc. have created a plan to effectively engaged the user throughout their morning, delivering relevant information at a suitable pace. This engagement will be done through a well-planned and thoroughly testing User Interface, ensuring it meets the client's needs through product demonstrations. This document will also thoroughly outline the management process that LARGE Software developers plan to follow. This will ensure that the client understands what the developers understand to be a minimal product for demonstration by the end of term, and a complete breakdown of all major features that the developers intend to have working by the time.

## SQLite Database Infrastructure

SQLite will be the local relational database engine used to house all information generated by the Good Morning application. A knowledgeable walkthrough of this database engine has been documented for both the client and developers to ensure that working knowledge concerned with the back-end of this application is documented. This walkthrough covers basic database operations such as creating a locally hosted database, creating tables, and fetching data. It also gives a brief overview of this application will utilize Androids Async helper class to ensure that the overall user experience goes un-interrupted, as well as how the Good Morning application might possibly interact with other applications through the use of Content Providers and URIs.

## Application Testing Schedule and Framework

The final section of this document will walk the reader through the comprehensive testing schedule that LARGE software developers have created. This schedule outlines what unit tests will be complete, when and how the application user interface will be tested, along with functional testing, performance testing and finally user acceptability testing. This thorough testing schedule will ensure that the Good Morning application is released to public in excellent working condition, meeting all of the initial application requirements given to LARGE Software developers.

## User Interaction

### Application Installation

In order to use the Good Morning application it must first be installed on the user's device. Since the application is solely an application for Android devices users must download it from the Google Play Store. Once installed the user can navigate to the application through their application menu and begin using it.

### Installation Interaction

The user must access the Google Play Store from their phone, this requires that their phone use the Android Os. From the Google Play Store the user can search for "Good Morning" to find the application developed by LARGE Software Inc. Once the application has been located, the user can select it to be brought to it's application page. From the application page select "INSTALL", this will begin the installation process which may take a few moments to download and install. Once the user has been notified that the application has been installed on their phone they may access it from their application menu.

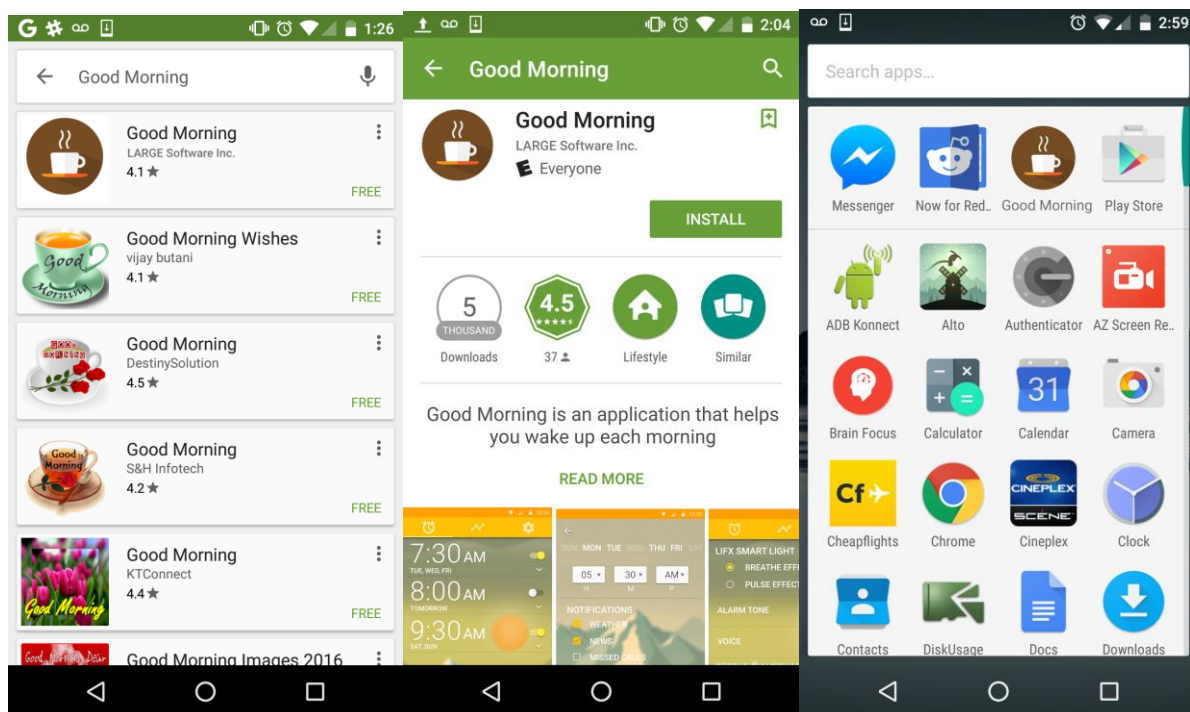


Figure 1: Installation flow, Google Play Store search, application Page, application available in the user's application menu



## Installation Interaction

Description	User interaction which culminates in the creation of a new alarm
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Device</li><li>• Google Play Store</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• Device must be using the Android operating system</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Access the device</li><li>2. Navigate to the Google Play Store</li><li>3. Search for “Good Morning”</li><li>4. Select the “Good Morning” application developed by LARGE Software Inc.</li><li>5. From the application page press “INSTALL”</li><li>6. Wait a moment until download and installation has completed</li><li>7. Access the Good Morning application from application menu</li></ol>
Rules	<ul style="list-style-type: none"><li>• User must have a Google account in order to install Play Store applications</li></ul>

## Alarm Creation

Create alarms using the Good Morning application to wake you up in the morning. The created alarm will wake the user up at their specified time or within a window of time if smart integrations are used.

At the time the alarm has begun it will present the user with two options, either “snooze” the application for a specified time (usually 10 minutes) or dismiss it. The snoozing input will delay the alarm for a specified time, while the dismiss input will close the alarm. After dismissal of the alarm, notifications which have been specified will be vocally read using the built-in virtual assistant. **Once the virtual assistant has finished it will ask the user to verbally confirm that they have received the notifications. If the user gives a negative response the virtual assistant will repeat them, otherwise interactions with the application will be complete.**

## Creation Interaction

Open the Good Morning application. The home screen which shows the user's current alarms and provides access to the configuration screen will be displayed. Press the addition symbol to begin creating the new alarm. The configuration screen allows characteristics of the alarm to be specified such as, time, notifications, and vocals used, and smart integrations will be displayed. Specify the time for the alarm to wake the user up, leaving the default configurations in the additional fields. After confirming that the alarm is configured as the user wants, press the "OK" button at the bottom of the configuration screen finalizing the creation process. You will be presented with an Android Toast which informs the user how long till the created alarm will go off, as well as being returned to the home screen which will now display the user's newly created alarm.

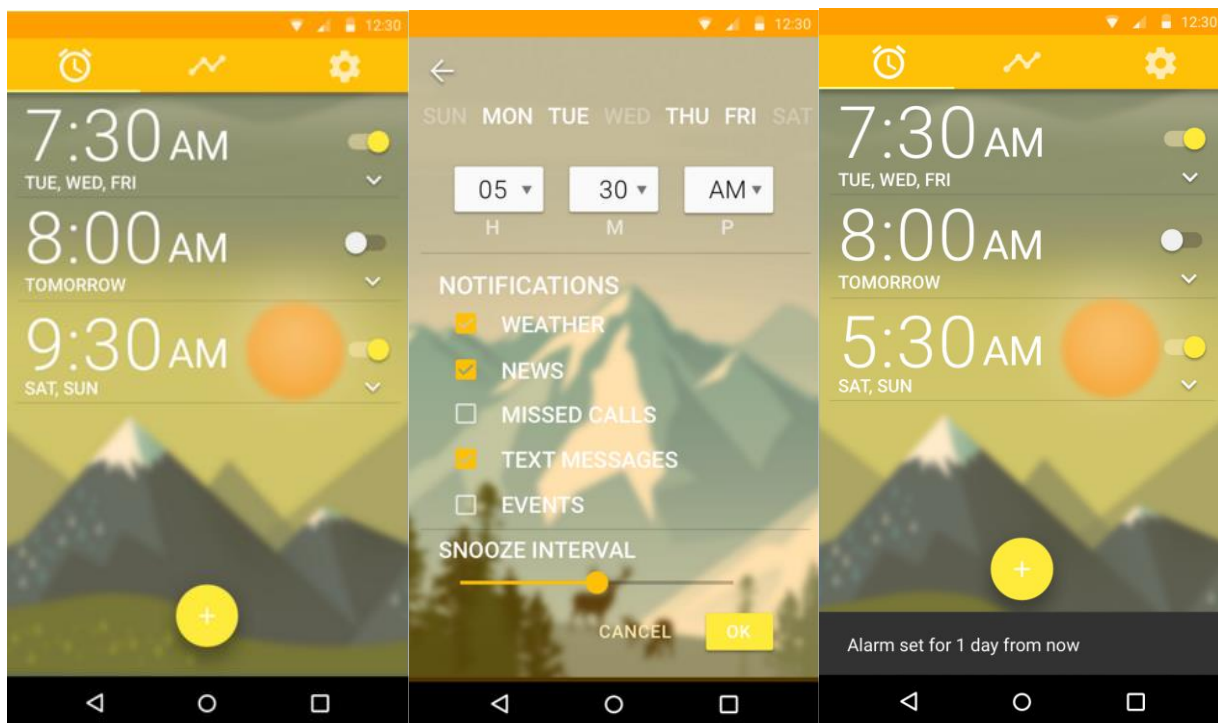


Figure 2: Creation flow, displaying the home page pre-alarm-creation, alarm creation page, and the home page post-alarm-creation.

## Creation Use Case Description

Description	User interaction which culminates in the creation of a new alarm
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• The application must have the permissions to access all notification streams</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Press the addition symbol</li><li>3. Configure the alarm's settings</li><li>4. Assign a time to the alarm</li><li>5. Confirm alarm configuration</li><li>6. Select "OK" to finalize alarm creation</li></ol>
Rules	<ul style="list-style-type: none"><li>• If no notifications are selected the application produces a common alarm.</li><li>• The alarm will default to being enabled on creation, this can be changed from the home screen.</li></ul>

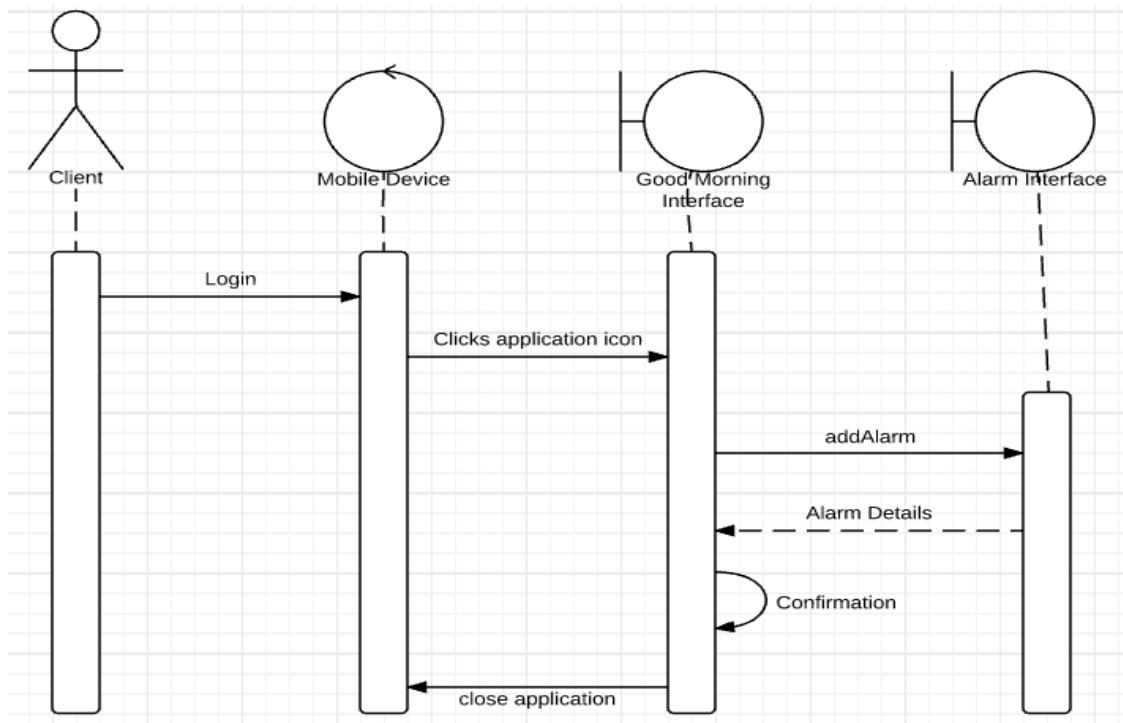


Figure 3: Alarm Creation Sequence Diagram

## Alarm Deletion

The user may want to delete their configured alarms once they are no longer needed or for any other reason. Please keep in mind that alarms can be reconfigured which may be more appropriate compared to deleting an alarm which is an irreversible action.

### Deletion Interaction

Open the Good Morning application. The home screen will be displayed along with all of the user's current alarms. Select the alarm to be deleted. The Alarm will expand to reveal additional operations, "Configure" and "Delete". Selecting Configure will allow the user to reconfigure the alarms current configuration. Selecting "Delete" will permanently and irreversibly remove the alarm. Select the "Delete" option, and a confirmation dialog will appear giving the user two options "Cancel" and "Confirm", selecting the former will return the user to the home screen with the alarm intact while selecting the later will finalize the deletion of the alarm. Select the "Delete" option. The home screen will be displayed and the alarm will be removed from the user's current alarms. The home screen will then display an option to undo the deletion, reverting the previous deletion and displaying the alarm again.

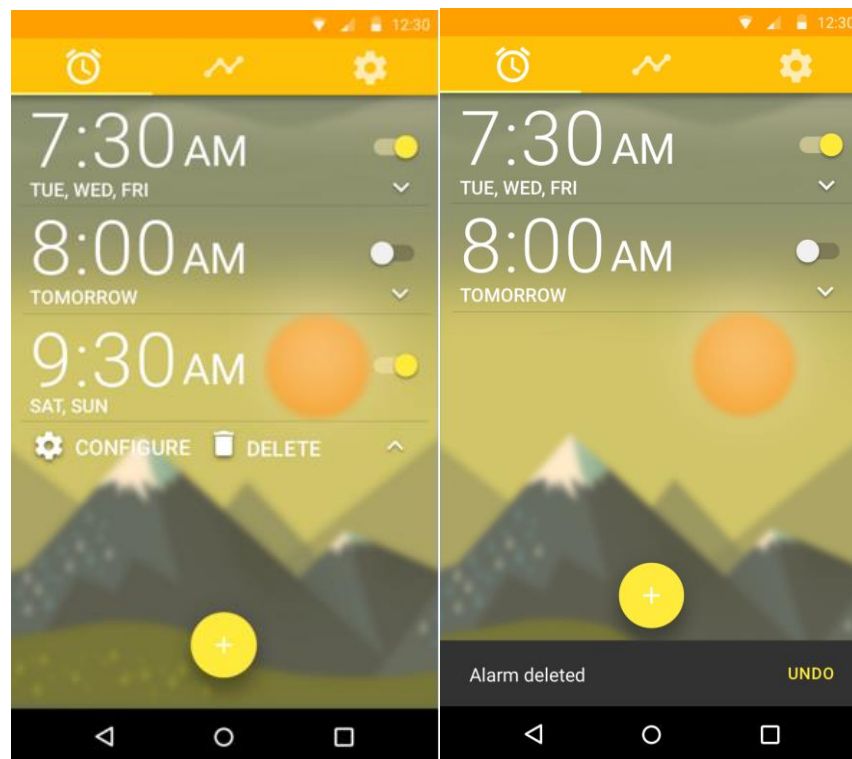


Figure 4: Deletion Flow, displaying the home page pre-alarm-deletion, and the home page post-alarm-deletion

## Deletion Use Case Description

Description	User interaction to delete the alarm from their list of preconfigured alarms displayed on the home page.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• The user must have at least one alarm to delete</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. From the home screen select the alarm to be deleted</li><li>3. Press the displayed “Delete” option</li><li>4. (Optional) Select undo to revert alarm deletion</li></ol>
Rules	<ul style="list-style-type: none"><li>• Once an alarm has been deleted and the undo option has disappeared the alarm is permanently deleted</li></ul>

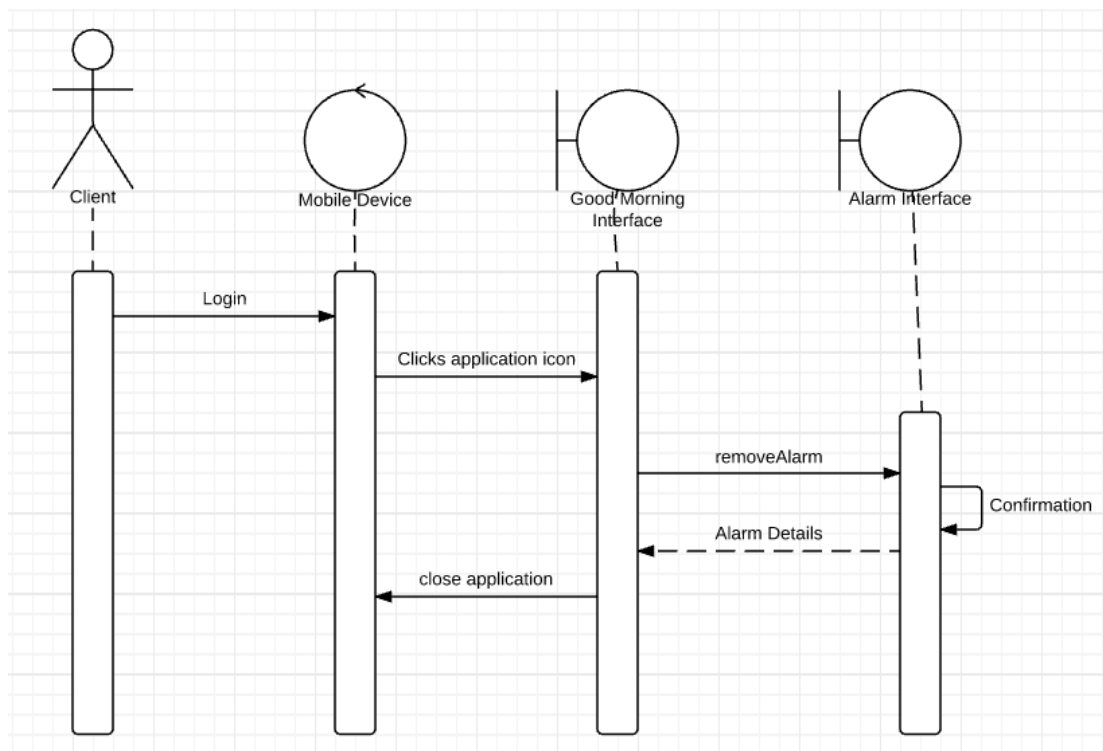


Figure 5: Alarm Deletion Sequence Diagram

## Alarm Snooze/Dismissal

When the user's alarm goes off they will be presented with two choices, either to “Snooze” or “Dismiss”.

Snoozing the alarm will silence the alarm and delay it for a configured period of time until it goes off again, this is set within the configure alarm screen. Snoozing the alarm is good for getting an extra bit of sleep before having to wake up.

Dismissing an alarm will silence it and begin the virtual assistant which vocally reads the user their configured notifications. Dismissal of the alarm is the trigger which begins the key component of the Good Morning application.

### Snoozing Interaction

Open the Good Morning application, and the home screen will be displayed along with all of the user's current alarms. To set the snooze interval, either create a new alarm or configure an old one. Once on the configuration screen of said alarm, select the “Snooze Interval” field and input an integer value ranging between 1 and 59, this value will represent the snoozing interval in minutes. Confirm the configuration of the alarm and select “Done”. Now when the alarm begins selecting the “Snooze” option will delay the alarm for the specified interval.

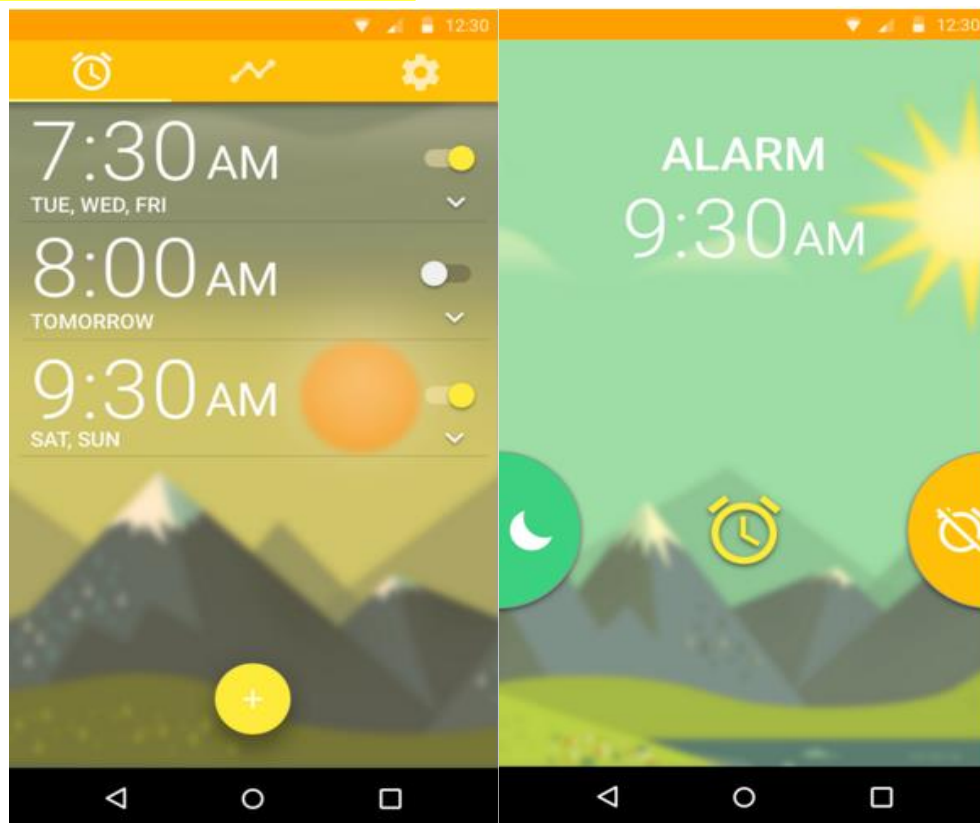


Figure 6: Alarm Snoozing/Dismissal flow; showing the set alarm, and the alarm once it has begun



### Snoozing Use Case Description

Description	User interaction to delay the current alarm by a pre-set time amount. This action also silences the alarm.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• An alarm must be configured</li><li>• An alarm must be currently sounding</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. The alarm sounds</li><li>2. Swipe right on the snooze button</li></ol>
Rules	<ul style="list-style-type: none"><li>• The alarm will snooze for the pre-set time configured</li></ul>

### Dismissing Interaction

Once the Alarm begins select the dismiss option. The alarm will silence and the vocal notifications will begin. The vocal notifications will alert the user with the configured information.

### Dismissing Use Case Description

Description	User interaction to dismiss the current alarm.. This action also silences the alarm.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• An alarm must be configured</li><li>• An alarm must be currently sounding</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. The alarm sounds</li><li>2. Swipe left on the dismiss button</li></ol>
Alternate Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Open the alarm that is going to sound</li><li>3. Turn off the alarm</li></ol>
Rules	<ul style="list-style-type: none"><li>• Alarms may only be dismissed once</li></ul>
Post conditions	<ol style="list-style-type: none"><li>1. The alarm is dismissed until the next time it sounds</li><li>2. The user's notifications will be read aloud if applicable</li></ol>

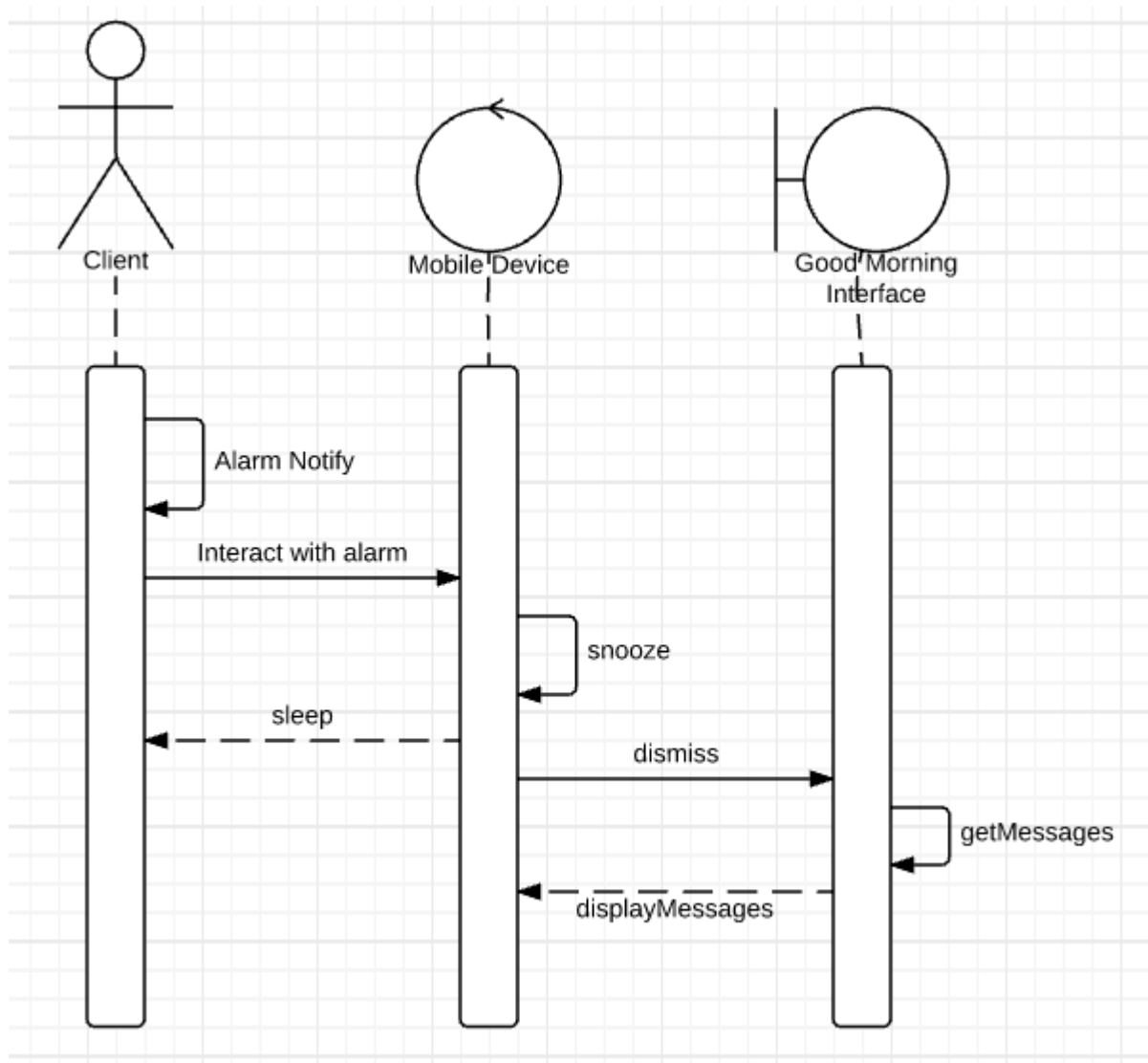


Figure 7: Alarm Snooze and Dismissal Sequence Diagram

## Alarm Notifications

Once the alarm has been dismissed it will vocally read the notifications which the user has configured using Google's built in speech capabilities. This will allow the user to receive a concise summary of valuable information of the user's day.

### Notification Configuration Interaction

Open the Good Morning application. The home screen will be displayed along with all of the user's current alarms. To set the notifications, either create a new alarm or configure an old one. Once on the configuration screen of said alarm, under the "Notifications" header select the notifications which the user would like to receive (calendar events, missed notifications and calls, weather information, news, etc.). Confirm the configuration of the alarm and select "Done".

### Notification Configuration Interaction

Description	User interaction when receiving the notifications associated with an alarm.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• User must have an alarm configured with notifications</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Dismiss an Alarm</li><li>2. Configured notifications will be read aloud</li></ol>
Rules	<ul style="list-style-type: none"><li>• If the alarm does not have any associated notifications nothing will be read</li></ul>

## Alarm Settings

The Alarm Settings screen will allow the user to configure various core alarm features and smart Internet of Things integrations. It can be accessed by tapping the “Gear” icon or by swiping to the right.

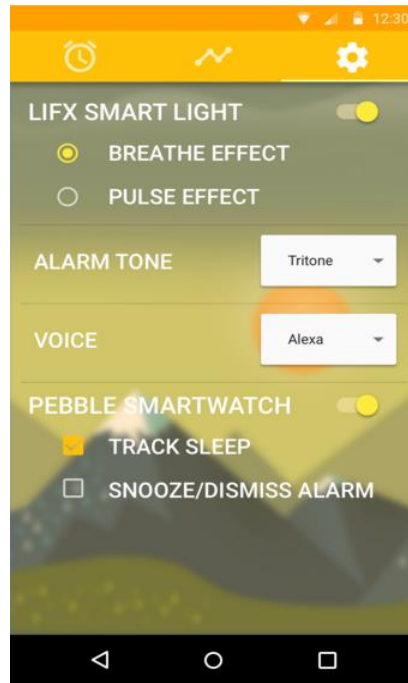


Figure 8: Alarm settings menu with integration, tone, and voice configurations

## LIFX Light Bulb

The LIFX Light Bulb is a smart lightbulb that can be configured to gently wake up the user with a breathe effect where lights are turned on slowly or with a pulse effect for the heavy sleepers. Turning on the LIFX Light Bulb by pushing the slider automatically discovers and connects to any light bulb that are connected in the same network.

### LIFX Configuration Interaction

Description	User interaction when enabling LIFX smart light capability.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li><li>• LIFX Smart Light</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• User must have a LIFX smart light installed and configured to their home wireless network</li><li>• User must be connected to their home wireless network on their phone</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Select the setting tab</li><li>3. Configure the light options</li><li>4. Press or swipe on the toggle switch next to the LIFX name</li></ol>
Post conditions	The light will now perform its configured operation upon an alarm sounding

### Alarm Clock Sound

Selecting the drop-down menu beside Alarm Tone allows the user to play and select one of the available thirty different alarm clock sounds to wake the user. The tones include nature sounds to gently wake up light sleepers and loud tones for heavy sleepers.

### Alarm Tone Configuration Interaction

Description	User interaction to configure the tone used when the alarm sounds
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• The user may use custom tones which they must download themselves</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Select the setting tab</li><li>3. Press the “Alarm Tone” drop down</li><li>4. Select an alarm tone</li></ol>
Rules	<ul style="list-style-type: none"><li>• This alarm tone is universal for all alarms</li></ul>

## Configure Voice

The user can select any one of five available voices to read them notifications when they wake up by pressing the drop-down menu beside Voice.

## Voice Configuration Interaction

Description	User interaction to configure the voice used to read notifications post alarm dismissal
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• The user may use custom voices which they must download themselves</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Select the setting tab</li><li>3. Press the “Voice” drop down</li><li>4. Select an alarm tone</li></ol>
Rules	<ul style="list-style-type: none"><li>• The user must have at least one voice file downloaded</li></ul>



## Pebble Smartwatch

The Pebble Smartwatch is a smartwatch that can connect via Bluetooth to a smart phone, track motion during sleep and perform simple actions against notifications. Tapping the switch beside Pebble Smartwatch, opens up a Bluetooth pair window that allows the Good Morning app to pair up with the user's Pebble Smartwatch. Enabling the "Track Sleep" option gathers accelerometer data from the Pebble Smartwatch while the user is sleeping and stores it in the Good Morning app so that Sleep Motion Graphs can be generated. Enabling the "Snooze/Dismiss Alarm" option allows the user to dismiss or snooze an alarm from their smartwatch.

Description	User interaction to configure Pebble smartwatch to track sleep and snooze/dismiss alarm from the watch.
Actors	<ul style="list-style-type: none"><li>• User</li><li>• Application</li><li>• Pebble Smartwatch</li></ul>
Preconditions	<ul style="list-style-type: none"><li>• User must have a Pebble smartwatch.</li><li>• User must have the Pebble Android app.</li><li>• User must be connected to their smartwatch via Bluetooth.</li></ul>
Basic Steps	<ol style="list-style-type: none"><li>1. Open the application</li><li>2. Select the setting tab</li><li>3. Tap the "Pebble Smartwatch" integration button</li><li>5. Select "Track Sleep" and/or</li></ol>
Post conditions	

## Activities

Various actions and views will be used in the creation of the Good Morning application. In order to represent these through our Android application LARGE software will make use of many “Activities” and “Fragments”. Activities are defined as “application component[s] that provides a screen with which users can interact”. [2] Fragments are defined as used to represent portions of behavior or user interface in an Activity, and are considered subsets of the Activity. The following activities have been identified for use in representing the functionality of our system:

- Main Activity
  - Displays the user’s configured alarms and allows navigation
- Alarm Configuration Activity
  - Allows users to adjusted pre-existing/newly-created alarms
- Sleep Statistics Activity
  - Allows users to view data collected on their sleeping habits
- Settings Activity
  - Users are able to configure connected devices and sounds settings
- Alarm Activity
  - Users will be able to dismiss/snooze their configured alarm

Latent knowledge also implies that all of these activities should allow navigation to all other activities; the user should never feel stuck within an activity.

## Fragments

Based on the identified Activities it’s possible to deduce some required fragments to deliver the required functionality of each activity. One positive use of the modular design of fragments is that identified functionality which is shared among activities can be represented as a single fragment cloned across multiple activities. This modularity of fragments will allow the system to be produced faster, with greater extendibility, and a lower overall complexity.

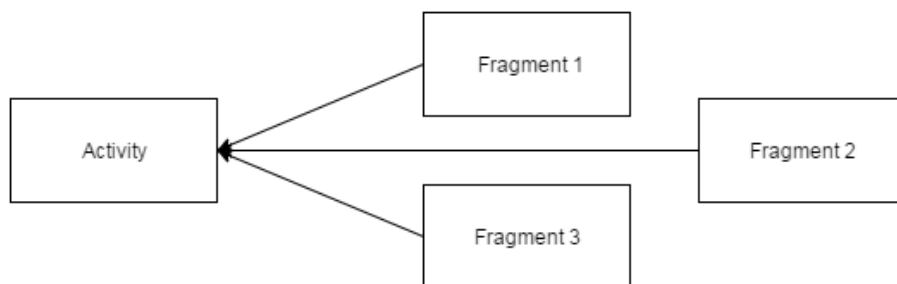


Figure 9: Example of an Activity’s composition of fragments

### Main Activity Fragments

This activity should display all the alarms which the user currently has configured, and clearly display the user's navigation options. These two functionality requirements implies the use of two fragments, one to allow the navigation to other activities, and another to display the alarms/information to the user.

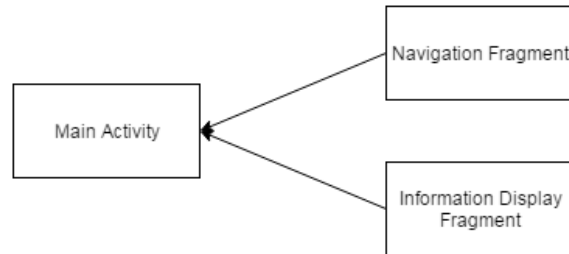


Figure 10: Main activities' composition of navigation and information display fragments.

### Alarm Configuration Activity Fragments

This activity will allow the user to configure an alarm, confirm and save configurations, and navigate between activities. These three functionality requirements implies the use of three fragments, a configuration, a confirmation, and a navigation fragment.

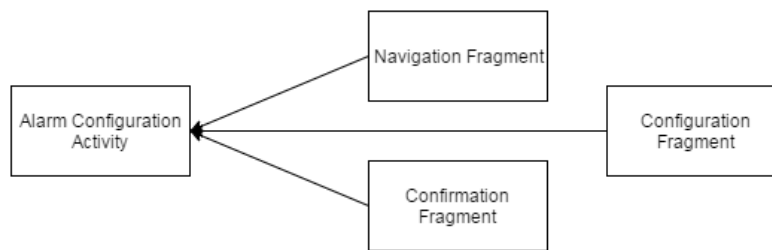


Figure 11: Alarm Configuration activities' composition of navigation, conformation and configuration fragments

### Sleep Statistics Activity Fragments

This activity will allow the user to view data collected on the user's sleep, and navigate between activities. These two functionality requirements implies the use of two fragments, a fragment to view sleep statistics/information, and a navigation fragment.

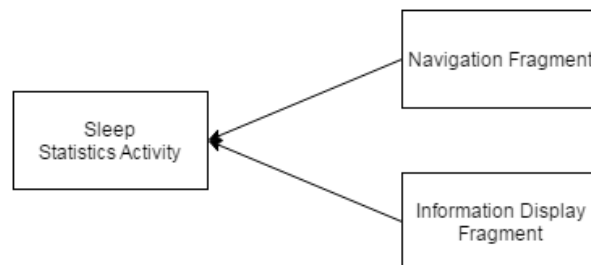


Figure 12: Sleep Statistics activities' composition of navigation and information display fragments.

### Settings Activity Fragments

This activity will allow the user to configure their settings, and navigate between activities. These two functionality requirements implies the use of two fragments, a configuration, a confirmation, and a navigation fragment.

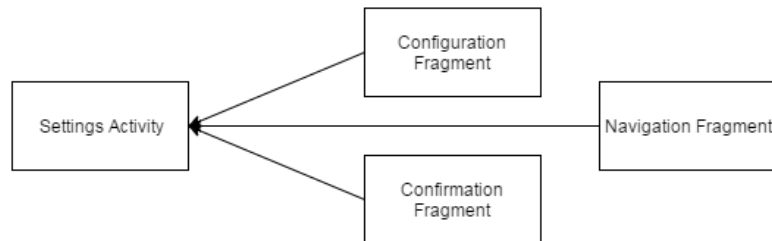


Figure 13: Sleep Statistics activities' composition of navigation and information display fragments.

### Alarm Activity Fragments

This activity will allow the user to manipulate their alarms. This functionality only requires the creation of an alarm interaction fragment.



Figure 14: Alarm activities' composition of the alarm interaction fragment.

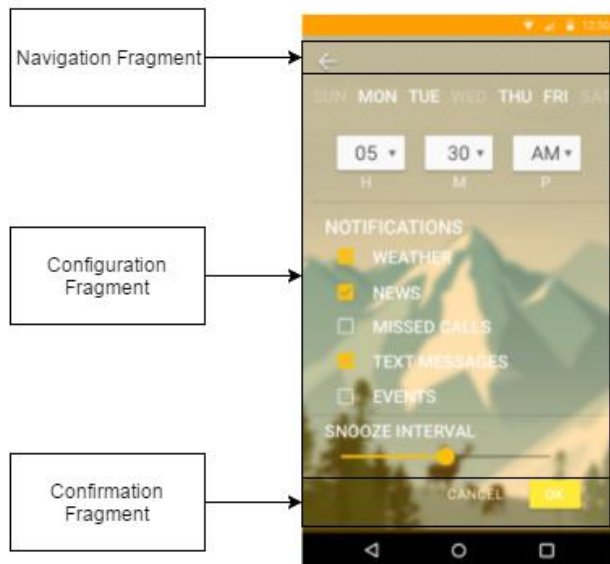
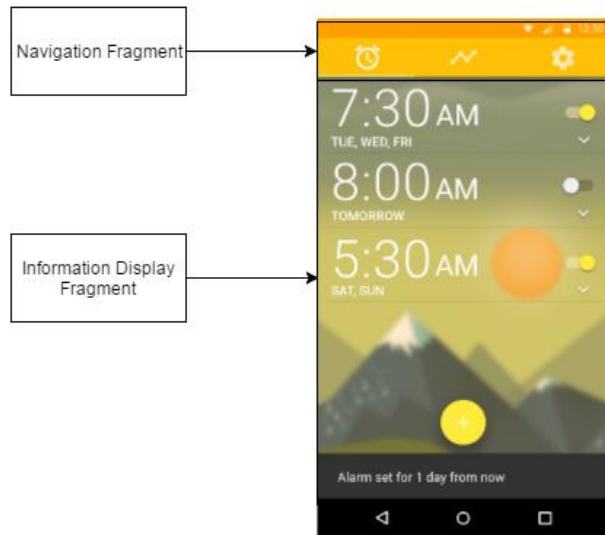
## Activity Views & Composition

From previous user interface designs we are able to begin the construction of activities and their respective composition of fragments. From these composition diagrams it becomes apparent areas in which fragments are duplicated. Once these cloned fragments are identified it is a matter of creation and then composing the activities of the modular fragments based on their functionality.

## Main Activity View & Composition

The Main activity is where users are able to view their alarms and their status, and navigate to any other activity within the Good Morning application. This activity will use the following fragments:

- Navigation fragment
  - Navigate to all other Activities, except Alarm interaction
- Information Display fragment
  - Displays configured alarms



## Alarm Configuration Activity View & Composition

The Alarm configuration activity is where users are able to change the configuration for a given alarm, confirm their configuration, and navigate back to the Main activity. This activity will use the following fragments:

- Navigation fragment
  - Navigate to the Main activity
- Configuration fragment
  - Allows the configuration of the alarm settings
- Confirmation fragment
  - Locally save the configuration fragment

## Sleep Statistics Activity View & Composition

The Sleep Statistics activity is where users are able to view information on their sleeping habits. This activity will use the following fragments:

- Navigation fragment
  - Navigate to the Main, and Settings activity
- Information Display fragment
  - Displays the sleep statistics

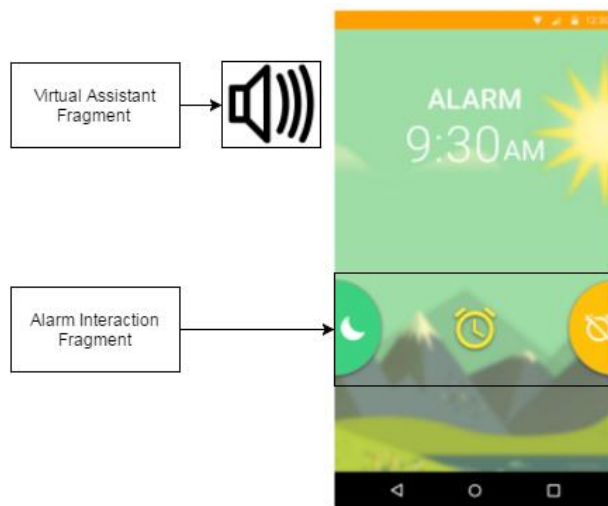
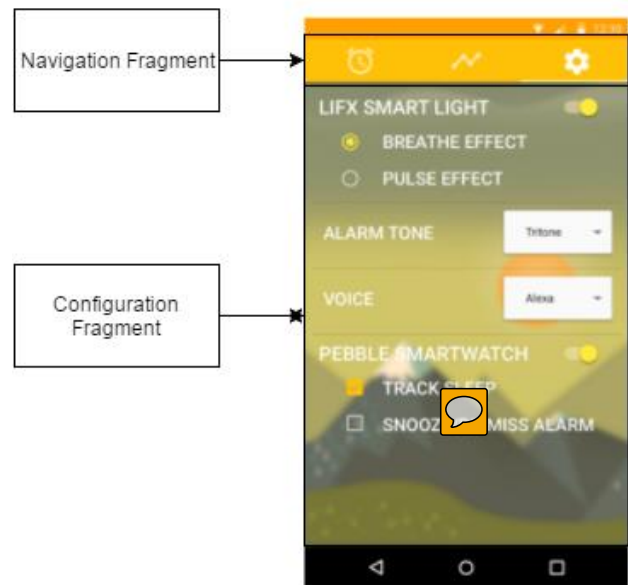




## Settings Activity View & Composition

The Settings activity is where users are able to configure their connected devices, and navigate to other activities. This activity will use the following fragments:

- Navigation fragment
  - Navigate to the Main, and Sleep statistics activities
- Configuration fragment
  - Configure the integrated devices, configure which sounds are used for the alarm and the virtual assistant's voice.



## Alarm Activity View & Composition

The Alarm activity is where users are able to interact with their alarm, and have their virtual assistant read relevant notifications. This activity will use the following fragments:

- Virtual Assistant fragment
  - Reads off configured alarm notifications to the user
- Alarm Interaction fragment
  - Allows the user to dismiss or snooze the alarm

## Fragments

From these composition representations we are able to conclude the fragments which will be needed to deliver the functionality of Good Morning. The fragments which will be needed are as follows:

- Navigation fragment
  - Based on location of user configures the display to allow navigation to other activities
- Information Display fragment
  - Based on the data viewed by the user, configures the screen to display the information
- Configuration fragment
  - Based on the data to be configured, allows the configuration of the data.
- Confirmation fragment
  - Depending on the configuration fragment associated with the activity, locally stores the configuration fragment's data.
- Alarm interaction fragment
  - Allows the user to dismiss or snooze the alarm.
- Virtual Assistant fragment
  - Reads aloud the notifications configured by the user for the alarm

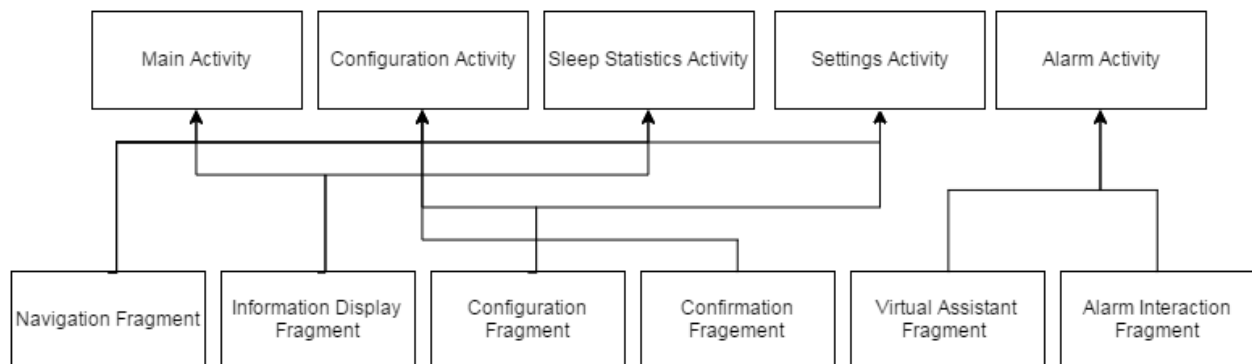


Figure 15: Sleep Statistics activities' composition of navigation and information display fragments.

## Navigation fragment

While five of the six activities use the navigation fragment, not all of them allow navigation to the same activities, for example the alarm configuration activity only allows navigation back to the main page via the return button. This implies that the navigation fragment must be able to handle displaying various links to activities dependent on the current location of the user.

## Implementation

Using android's standard ActionBar we are able to create a navigation fragment as so:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    final ActionBar actionBar = getActionBar();

    // Specify that tabs should be displayed in the action bar.
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

    // Create a tab listener that is called when the user changes tabs.
    ActionBar.TabListener tabListener = new ActionBar.TabListener() {
        public void onTabSelected(ActionBar.Tab tab, FragmentTransaction ft) {
            // show the given tab
        }

        public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction ft) {
            // hide the given tab
        }

        public void onTabReselected(ActionBar.Tab tab, FragmentTransaction ft) {
            // probably ignore this event
        }
    };

    // Add 3 tabs, specifying the tab's text and TabListener
    for (int i = 0; i < 3; i++) {
        actionBar.addTab(
            actionBar.newTab()
                .setText("Tab " + (i + 1))
                .setTabListener(tabListener));
    }
}
```

Figure 16: Standard Android ActionBar Example Code

## Information Display fragment

The Information Display fragment must be able to handle various forms of data, whether it be alarm data or statistical data. This fragment will be able to display the given data on the screen in a user friendly way. Its ability to handle various forms of data will allow this fragment to be used in future development as the system is expanded.

## Implementation

Using android's standard Fragment we are able to create an information display fragment as so:

```
public class InformationDisplayFragment extends Fragment {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

Figure 17: Android Standard Fragment Example Code

## Configuration fragment

The Configuration fragment will handle interactions with the user about their configurable data. These interactions can range from switching a flag on/off, selecting an option from a dropdown, or integrating certain supported devices.

## Implementation

Using android's PreferenceActivity we are able to create a configuration fragment as so:

```
public class ConfigurationFragment extends PreferenceActivity  
    implements Preference.OnPreferenceChangeListener {  
    private void bindPreferenceSummaryToValue(Preference preference) {  
        // Set the listener to watch for value changes.  
        preference.setOnPreferenceChangeListener(this);  
  
        // Trigger the listener immediately with the preference's  
        // current value.  
        onPreferenceChange(preference,  
            PreferenceManager  
                .getDefaultSharedPreferences(preference.getContext())  
                .getString(preference.getKey(), ""));  
    }  
}
```

Figure 18: Android PreferenceActivity Example Code

## Confirmation fragment

The Confirmation fragment will handle locally storing the data which is configured within the Configuration fragment. Although the fragment is always associated with a Configuration fragment the Configuration fragment does not always associate with the Confirmation fragment, for example in the Settings activity there is no confirmation dialog.

## Implementation

Using android's PreferenceActivity we are able to create a confirmation fragment as so:

```
public class ConfirmationFragment extends PreferenceActivity
    implements Preference.OnPreferenceChangeListener {

    public boolean onPreferenceChange(Preference preference, Object value) {
        String stringValue = value.toString();

        preference.setSummary(stringValue);

        return true;
    }
}
```

Figure 19: Android PreferenceActivity Example Code

## Alarm Interaction fragment

The alarm interaction fragment will allow users to snooze or dismiss an alarm. The snoozing action will delay the alarm by the value stored as snooze value in the local alarm data. If the alarm is dismissed the fragment will initialize the Virtual Assistant fragment to begin the notification interaction.

## Implementation

Using android's WakefulBroadcastReceiver we are able to create an Alarm Interaction fragment as so:

```
import android.support.v4.content.WakefulBroadcastReceiver;
public class AlarmInteractionFragment extends WakefulBroadcastReceiver {
    @Override
    public void onReceive(final Context context, Intent intent) {

        if(context.dismiss){
            // Begin Virtual Assistant fragment
        }else{
            // Snooze for preset value of time
        }

    }
}
```

Figure 20: Android WakefulBroadcastReceiver

## Virtual Assistant fragment

Using the alarm data, configured notifications specifically, the Virtual assistant fragment will use `contentValues()` to encapsulate alarm data notification. Once retrieved, using the speech feature, these notifications will be read aloud.

## Implementation

Using android's TextToSpeech library we are able to create a Virtual Assistant fragment as so:

```
public class VirtualAssistantFragment implements OnInitListener {  
    private TextToSpeech tts;  
  
    public VirtualAssistantFragment(Context context){  
        tts = new TextToSpeech(context, this);  
    }  
  
    public void VirtualAssistantFragment(String text){  
        HashMap<String, String> hash = new HashMap<String,String>();  
  
        hash.put(TextToSpeech.Engine.KEY_PARAM_STREAM,  
                String.valueOf(AudioManager.STREAM_NOTIFICATION));  
        tts.speak(text, TextToSpeech.QUEUE_ADD, hash);  
    }  
}
```

Figure 21: Androids TextToSpeech Example Code



## Management Plan

Following the provided schedule, a final product demonstration is planned for the end of March 2016. Currently, developers are working on finding the best combination of implementations for the core features to ensure the best experience for all users. For this final demonstration, LARGE developers plan to have the following features available as a minimal system of Good Morning:

- Core Alarm
- Google Calendar
- Core Interactions/Settings Page
- Speech Synthesis

This section of the Management Plan document will discuss in detail, said implementations of core application features. Expanded system features that developers hope to implement on top of the planned minimal Good Morning system given appropriate time and resources are also described.

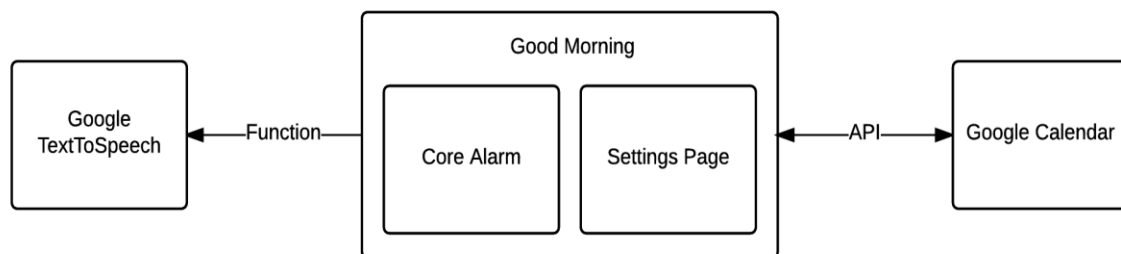


Figure 22: Minimal System Relationship Diagram

## Core Alarm

The core alarm may be implemented in a few ways. The first implementation involves creating a proprietary alarm application. This version will integrate the settings page and alarm application together in a single application.

### Proprietary Core Alarm

- Utilizes an aesthetically-pleasing user-interface
- When the alarm goes off, the user may choose to dismiss or snooze the alarm
- Displays user-editable clocks for set alarm times

The second implementation uses Android APIs to create, modify, and delete alarms.

## Core Alarm using Google Clock

- Settings are separated from the clock functionality
- Alarms are set in settings, but use Google Clock as the UI

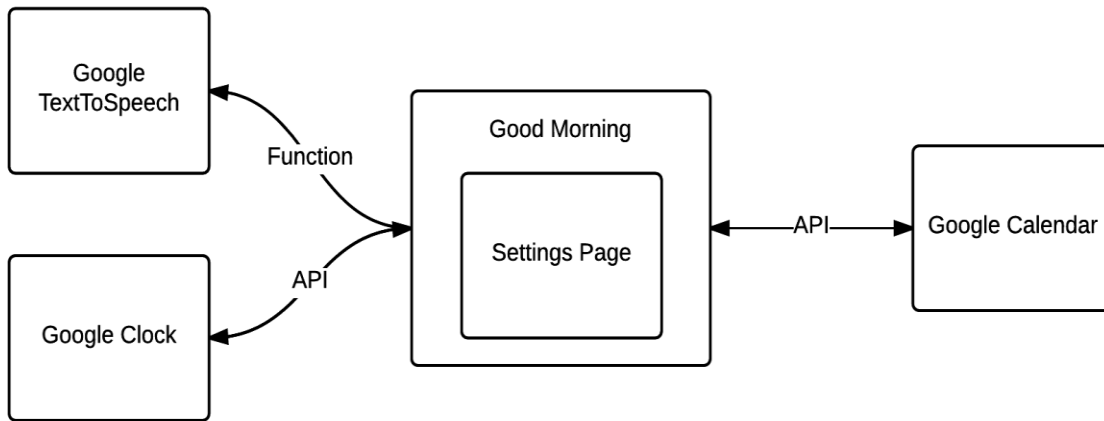


Figure 23: Minimal System Relationship Diagram Utilizing Google Clock

## Google Calendar

The Good Morning application is planned to be initially released on the Android platform. The application will access the Google Calendar API, ensuring that users are notified of any events throughout the day. The Good Morning application will do this in the following ways:

- Access upcoming events from Google Calendar and display via Good Morning User Interface
- Ensure that daily events (holidays, birthdays etc.) are interpreted correctly and separately from appointments or scheduled events
- Summarize events in chronological order
- Ensure to notify user of importance of impending events approaching (ex. User wakes up at 7:30am, and has a scheduled event happening within an hour of wake up)
- If implemented with expanded feature speech synthesis, Good Morning should be able to accurately and efficiently convey upcoming events to users without having to be read

## Speech Synthesis

The Good Morning application will synthesize the notifications specified by the user to speech output. This output will be read to the user once their alarm has been dismissed, updating them with information useful to their day.

LARGE Software Inc. will leverage the following to complete the speech synthesis requirement:

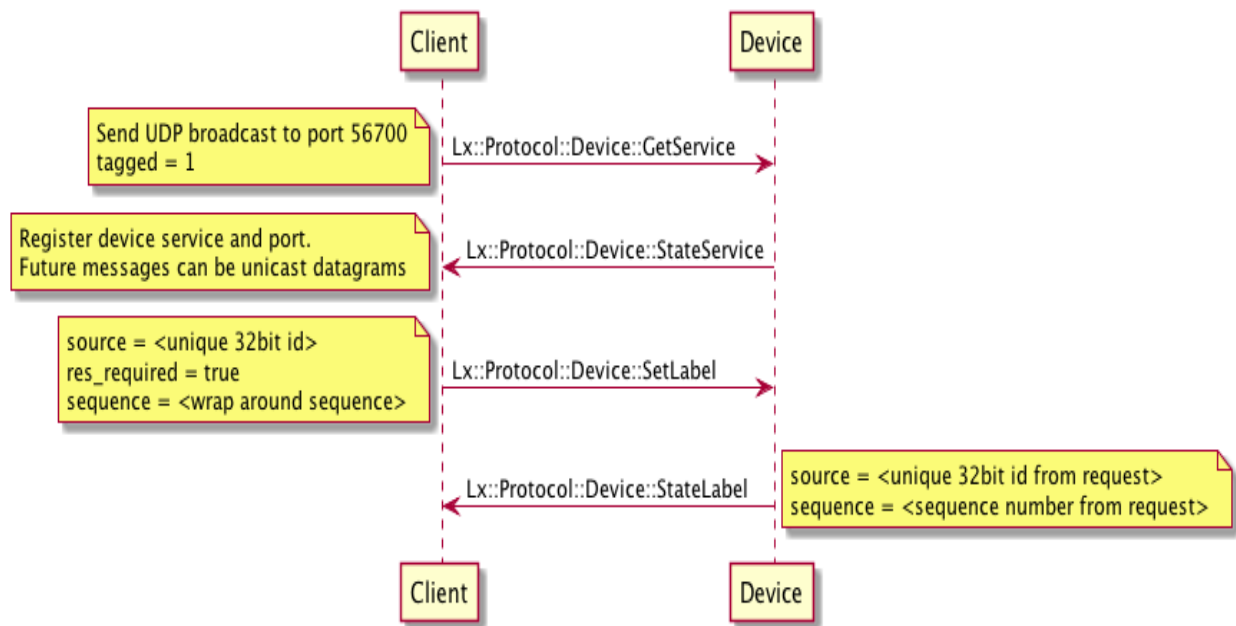
- Use Google TextToSpeech functionality
- Use Core Alarm and Settings to know when to activate
- Read configured notifications and information using phone speakers or connected device

## Expanded System

The expanded system includes features that enhance the user experience. These features may potentially be included, but are not guaranteed to be finished for the release product.

### LIFX Smart Lights

- Uses LIFX Developer API
- Uses Core Alarm and Settings to know when to activate



- Sends commands to the lights to dim, brighten, change color, or turn on and off

Figure 24: Interaction of connection between client and LIFX device [1]

### Music Wakeup

- Extension of Core Alarm app
- Uses phone speakers or connected device to play sound
- User may specify what music will wake them up instead of a typical alarm
- User may choose to have the music gradually increase in volume

### Machine Learning

- Uses custom algorithms
- Learns when the user wakes up, and their typical daily schedule

- Uses knowledge of the user to set dynamic alarms
- Reminds them of things that they may have forgotten that they
- Use of additional devices such as smart watches (accelerometers specifically) to track wake ups during REM sleep patterns

## Relationship between Features

The diagram below demonstrates how the extended functionality would be incorporated with the minimal functionality. In this case, Good Morning may be considered as either the proprietary core alarm, or the core alarm using Google Clock.

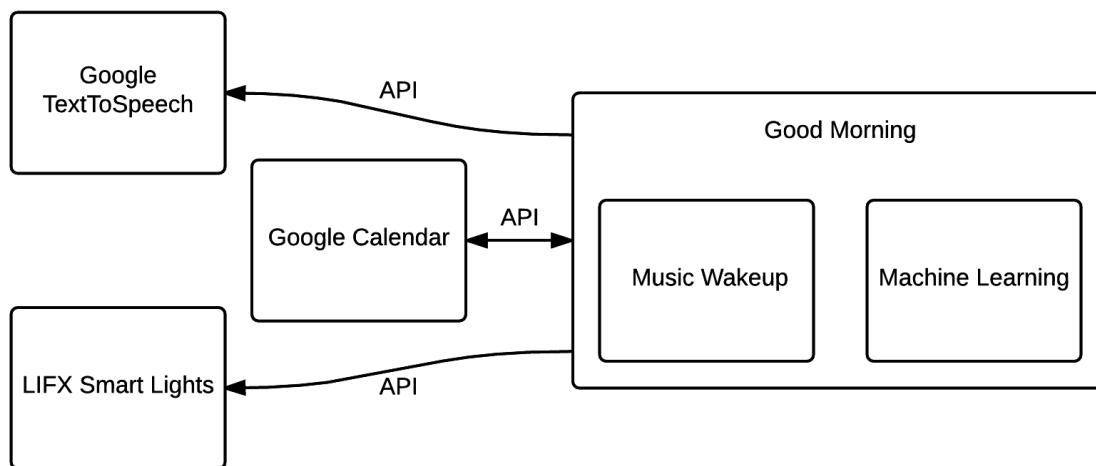


Figure 25: Expanded System Relationship with Core Alarm

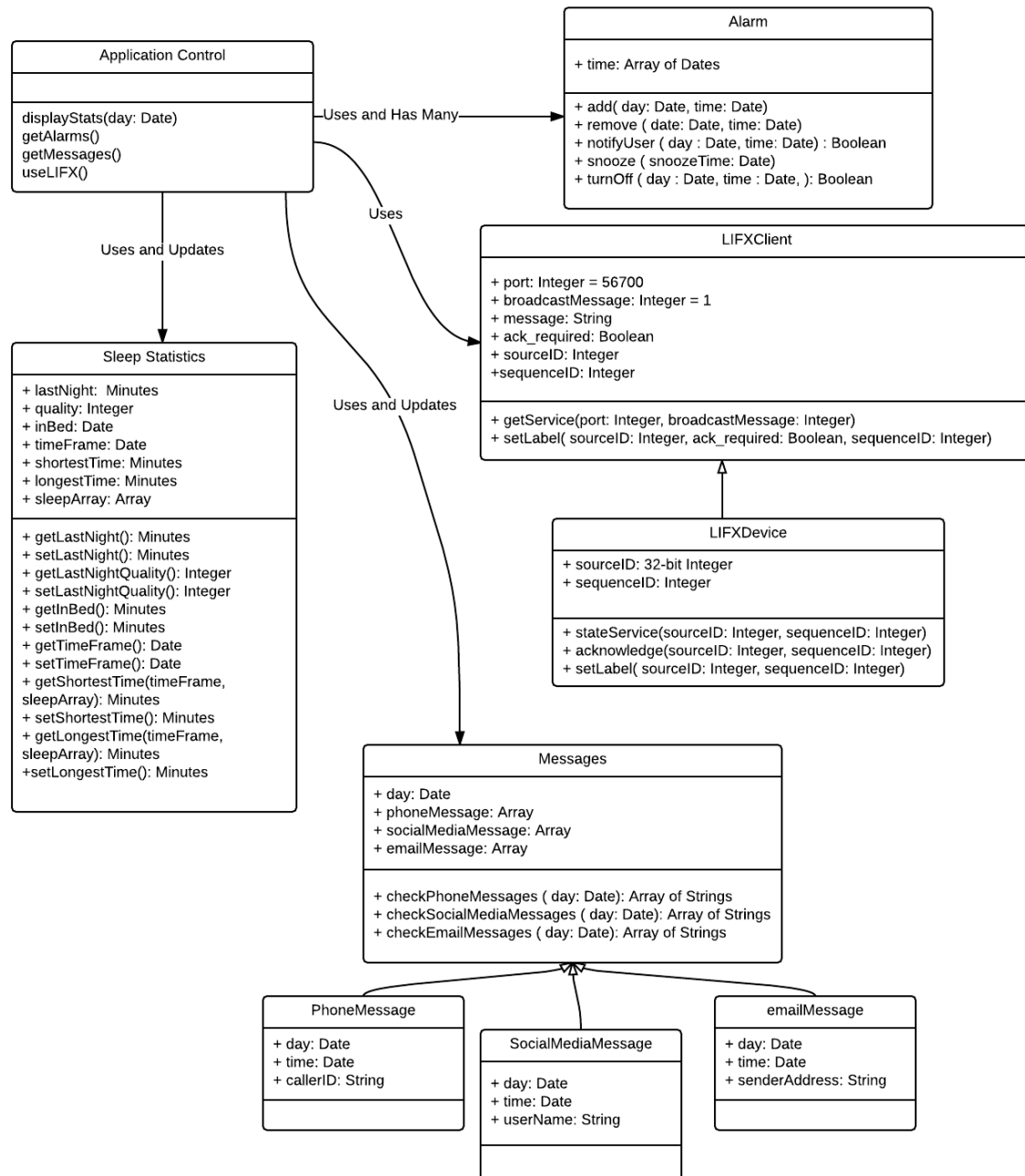


Figure 26: Class Diagram of the Good Morning application

## Technical Specifications - SQLite Introduction

To meet the requirements of storing data, the Good Morning application will require an operational relational database. To accomplish this, Good Morning will implement SQLite, an open source SQL relational database. Unlike typical client-server databases, SQLite stores data locally on devices which allows applications to quickly store and access information.[3] This ensures that users have a positive and efficient interaction with any application that implements and uses SQLite.

Because the Good Morning application will be initially deployed on Android devices, it makes choosing SQLite easy due to all Android devices having a SQLite database implementation already built in. [4] For scalability reasons, implementing SQLite for the initial Android deployment makes developing this application for Apple devices easier, due to all iOS devices having SQLite implemented on them as well.

## Database Operations

### Database Creation

In order to create a database for the Good Morning application, developers will call the `openOrCreateDatabase` method from the SQLite package and pass the name of the Good Morning database as a parameter to the method (Figure 27).

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);
```

Figure 27: Database Creation

### Table Creation

After the database has been created, tables will have to be then created to store the data required for the Good Morning application to run correctly. In this case, both an alarms and notifications table will be made, as seen in Figure 28.

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS Alarms(Day VARCHAR,Time INTEGER);");  
  
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS Notifications( ServiceName VARCHAR, Message  
VARCHAR, SentBy VARCHAR, DateSent DATE);");
```

Figure 28: Database Table Creation

## Fetching Data

When the application requires information to be retrieved from the database, an object of the Cursor class will be made, and the `rawQuery` method will be called to return a resultset with the Cursor pointing to the table from which the query was executed on. Figure 29 shows an example of how the application will retrieve alarm information from the Alarm database table.

```
Cursor resultSet = mydatabase.rawQuery("Select * from ALARM where Day = 'INSERT TODAYS NAME  
HERE'",null);  
  
resultSet.moveToFirst();  
  
Integer time = resultSet.getInt(1);
```

Figure 29: Data Fetch

## Database Helper Class

A helper class will be written in order to manage all relational operations on the Good Morning database. The helper class will automatically manage creation and updates of new and current tables for the application. Figure 30 is an example of what the Good Morning helper class might look like.

```
public class DBHelper extends SQLiteOpenHelper {  
    public DBHelper(){  
        super(context,DATABASE_NAME,null,1);  
    }  
    public void onCreate(SQLiteDatabase db) {}  
    public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}  
}
```

Figure 30: Database Helper Class

## Async Helper Class

The Async helper class ensure that there is a seamless transition of data from the database to the application user interface. The class allows background operations to publish results to the UI thread without having to manipulate threads or headers. [5] [6] The Good Morning application will utilize the Async helper class in all transactions between the database and UI to ensure that users will have an un-interrupted application experience.

## Entity Framework

Entity Framework is an Object/Relational Mapping (ORM) framework that enables developers to work with relational data as domain specific objects, eliminating the need for most of the data access plumbing code that developers are required to write. [5] Utilizing the Entity framework will allow developers to focus on domain classes, and then work with those classes to create information within the database. In this case, Good Morning will have two main entities as seen in Figure 31 and 32.

```
public class MyAlarm{  
    Public integer appID { get; set;}  
    Public integer myAlarmTime { get; set;}  
    Public string myAlarmDays { get; set;}  
    Public string mySnooze { get; set;}  
}
```

Figure 31: Alarm Entity

```
public class Notification{  
    Public integer appID { get; set;}  
    Public string myAppName { get; set;}  
    Public string myNotification { get; set;}  
}
```

Figure 32: Notification Entity

## Database Schema

The term schema refers to the organization of data as a blueprint of how a database is constructed.[5] In the case of the Good Morning application the database will be constructed around two base tables (Alarm and Notifications). For Android application development, a contract class will be written, which will explicitly specify the layout of the schema in a systematic and self-documenting fashion.[7] The contract class itself acts as a container for constraints that define names for URI (uniform resource identifier), tables and columns in those tables. Figure 33 and 34 will outline the basic structure for both the Alarm and Notification contracts. Figure 35 shows a basic Entity-Relationship Diagram of the Good Morning Database.



```

public final class alarmContract {
    public alarmContract() {}

    public static abstract class Alarm implements BaseColumns {

        public static final String TABLE_NAME = "Alarm";

        public static final String COLUMN_NAME_ENTRY_ID = "appID";

        public static final String COLUMN_NAME_TITLE = "alarmTime";

        public static final String COLUMN_NAME_TITLE = "alarmDays";

        public static final String COLUMN_NAME_TITLE = "alarmSnooze";

    }
}

```

Figure 33: Alarm Contract Class

```

public final class notifcationsContract {

    public notificationContract() {}

    public static abstract class Alarm implements BaseColumns {

        public static final String TABLE_NAME = "Notifications";

        public static final String COLUMN_NAME_ENTRY_ID = "appID";

        public static final String COLUMN_NAME_TITLE = "appName";

        public static final String COLUMN_NAME_TITLE = "notificationMessage";

    }

}

```

Figure 34: Notification Contract Class

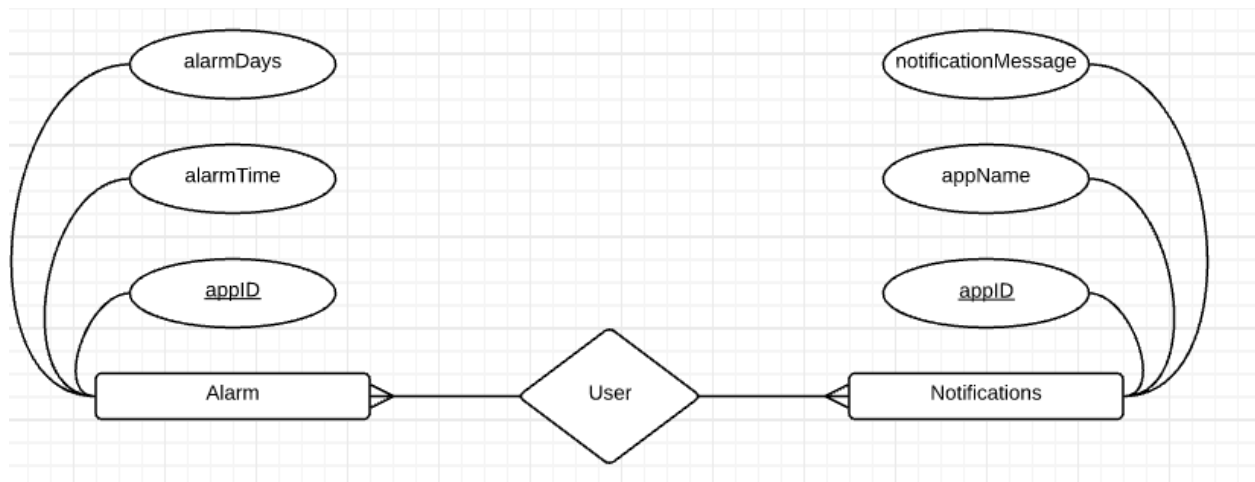


Figure 35: Entity-Relationship Diagram

## Content Providers and URIs

A content provider acts as a manager for how an application accesses a central data repository. The provider is directly related to an Android application, in this case the Good Morning application, and works directly with the application UI (user interface).[8]

A defined requirement of the Good Morning application was that it could interact with other applications on a specific user's mobile device. In order to meet this requirement, a content provider will be required. The content provider will ensure that consistent and standard interface to any data generated by the Good Morning application, and it will also handle all inter-process communication between applications, while ensuring that all inter-process communication is done in a secure fashion.[8]

In order for applications to access data from the Good Morning content provider, they will do so using the ContentResolver client object. The ContentResolver methods provide the basic “CRUD” (Create, Read, Update, Delete) functions to any data that they require. An example of how an application would access Good Morning alarm data is seen in Figure 36.

```
mCursor = getContentResolver().query(  
    UserAlarms.Times.CONTENT_URI, // The content URI of the alarm table  
    mProjection, // The columns to return for each row  
    mSelectionClause // Selection criteria  
    mSelectionArgs, // Selection criteria  
    mSortOrder); // The sort order for the returned rows
```

Figure 36: Sample Content Resolver [8]

In order to for other applications to retrieve data from the Content Provider, then need to do the following in order:

- Request read access permissions for the provider
- Define the code that sends a query to the provider

Requesting read access permissions from the Good Morning application. This request cannot be done at run-time while applications interact with each other, instead this permission is asked for once in the manifest created using the <user-permission> element followed by the permission required from the provider. In this case, alarm data would be requested by using the following code:

```
android.permission.READ_USER_ALARM
```

Next, the query will be constructed to access the Good Morning data required. Figure 37 will show an example of how to access this data.

```
// A "projection" defines the columns that will be returned for each row

String[] mProjection =

{

    UserAlarm.Alarm.appID, // Contract class constant for the _ID column name

    UserAlarm.Alarm.alarmTime, // Contract class constant for the time column name

    UserAlarm.Alarm.alarmDays // Contract class constant for the days column name

};

// Defines a string to contain the selection clause

String mSelectionClause = null;

// Initializes an array to contain selection arguments

String[] mSelectionArgs = {""};
```

Figure 37: Content Provider Query Example [8]

## Content Provider Insert, Update and Delete

CRUD actions can be performed through a Content Provider. The following example code in Figure 38 shows how alarm information can be handed from an outside application to the Good Morning application if required by calling the `ContentResolver().insert()` method. Please note that all examples within this section can be replicated for use in the Notifications table as well with minor modifications to the example code listed below in Figures 38, 39 and 40.

```

// Defines a new Uri object that receives the result of the insertion

Uri mNewUri;

// Defines an object to contain the new values to insert

ContentValues mNewValues = new ContentValues();

/*
 * Sets the values of each column and inserts the word. The arguments to the "put"
 * method are "column name" and "value"
 */

mNewValues.put(UserAlarm.Alarms.appID, "1");
mNewValues.put(UserAlarm.Alarms.alarmTimes, "0645");
mNewValues.put(UserAlarm.Alarms.alarmDays, "M,W,F");

mNewUri = getResolver().insert(

    UserAlarms.Alarms.CONTENT_URI, // the user dictionary content URI

    mNewValues // the values to insert

);

```

Figure 38: Content Provider Insert Example [8]

Similarly, applications can also request to update information through Content Provider. Figure 39 illustrates an example of how an outside application would attempt to update alarm information in the Good Morning application database through the use of the `ContentResolver().update()` method.

```
// Defines a new Uri object that receives the result of the insertion

Uri mNewUri;

// Defines an object to contain the new values to insert

ContentValues mNewValues = new ContentValues();

/*
 * Sets the values of each column and inserts the word. The arguments to the "put"
 * method are "column name" and "value"
 */

mNewValues.put(UserAlarm.Alarms.appID, "2");
mNewValues.put(UserAlarm.Alarms.alarmTimes, "0645");
mNewValues.put(UserAlarm.Alarms.alarmDays, "M,W,F");

mNewUri = getContentResolver().insert(

    UserAlarm.Alarms.CONTENT_URI, // the user dictionary content URI

    mNewValues // the values to insert

);
```

Figure 39: Content Provider Update Example [8]

Finally, applications can delete data if so required through the Content Provider as well. Figure 40 illustrates an example of how an outside application would attempt to delete alarm information in the Good Morning application database through the `ContentResolver().delete()`

```
// Defines selection criteria for the rows you want to delete

String mSelectionClause = UserAlarms.Alarm.appID "2";

String[] mUserAlarmTime = {"0645"};

// Defines a variable to contain the number of rows deleted

int mRowsDeleted = 0;

// Deletes the alarms that match the selection criteria

mRowsDeleted = getContentResolver().delete(

    UserDictionary.Words.CONTENT_URI, // the user dictionary content URI

    mSelectionClause // the column to select on

    mUserAlarmTime // the value to compare to

);
```

Figure 40: Content Provider Delete Example [8]

## Alarm Service

After the user presses “OK” button in the create alarm activity, the provided information about notifications, days and snooze interval is stored in the SQLite database. This data is only used to display the alarms from the Main Alarm Activity. An intricate mechanism of Intents, Broadcasts and Services provided by the Android OS is used to start the alarm at the specified time.

Following the Android specification ensures that the alarm will start regardless of the Good Morning application being run on the foreground. This will also have little to no effect with battery life since no running daemon is used to check the time but the Android OS decides to wake the Good Morning Alarm Activity at the correct time. The complete mechanism is explained in detail below.

The CreateAlarm Activity contains a reference to the AlarmManager object instantiated by the Android OS. The AlarmManager class “provides access to the system alarm services. These allow you to schedule your application to be run at some point in the future. When an alarm goes off, the [Intent](#) that had been registered for it is broadcast by the system, automatically starting the target application if it is not already running.”[11] The AlarmManager class cannot be instantiated directly but the AlarmManager object instantiated by the OS can be retrieved using:

```
Context.getSystemService(Context.ALARM_SERVICE)
```

Before the AlarmManager can be scheduled to be delivered, a pending intent and a broadcast receiver needs to be setup. First, a new AlarmReceiver class is created so that when the AlarmManager it sends a broadcast at the scheduled time, it can receive an Intent to start the desired activity. The AlarmReceiver class is inherited from the BroadcastReceiver class. To ensure that the BroadcastReceiver only receives broadcasts intended for the Good Morning application, the following line is added to the AndroidManifest.xml.

```
<receiver android:name=".AlarmReceiver"/>
```



This will automatically execute the `onReceive()` method inside the `AlarmReceiver` class when the `AlarmManager` broadcasts an intent for Good Morning. “A `BroadcastReceiver` object is only valid for the duration of the call to [onReceive\(Context, Intent\)](#). Once your code returns from this function, the system considers the object to be finished and no longer active.”[12]

Furthermore, the Alarm Manager mentions that “The Alarm Manager holds a CPU wake lock as long as the alarm receiver's `onReceive()` method is executing. This guarantees that the phone will not sleep until you have finished handling the broadcast. Once `onReceive()` returns, the Alarm Manager releases this wake lock. This means that the phone will in some cases sleep as soon as your `onReceive()` method completes.”[11]

This severely limits the functions that can be done from this class. Therefore, another class is created called `RunningAlarm.java` that extends from the `Activity` is created to actually sound the alarm and provide the user to snooze/dismiss. More about the `RunningAlarm` class will be discussed later. To start the `RunningAlarm` activity from the `onReceive()` method inside `BroadcastReceiver` the following code and intent is used:

```
Intent runningAlarm = new Intent();
runningAlarm.setClassName("com.LargeSoftware",
    "com.LargeSoftware.RunningAlarmActivity");
runningAlarm.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(runningAlarm);
```

Setting `FLAG_ACTIVITY_NEW_TASK` makes it seem to the user that the app has opened up with the `Running Alarm Activity` as the first activity.

Intents are typically used to launch Activities as shown above. However, `AlarmManager` requires a class similar to `Intent` called `PendingIntent`. The activity or service inside this `PendingIntent` is what will be initiated during broadcast by the `AlarmManager` “By giving a `PendingIntent` to another application, you are granting it the right to perform the operation you have specified as if the other application was yourself (with the same permissions and identity).”[13] To create a `PendingIntent`, we first create an `Intent` and then pass it to `PendingIntent`’s `getBroadcast` method as shown below:

```
Intent alarmReceiver = new Intent(this, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, alarmReceiver,
    FLAG_UPDATE_CURRENT);
```

This pendingIntent is then set via the AlarmManager. However, the AlarmManager requires the time to be set in milliseconds. To get the alarm time in milliseconds, the calendar class can be used. “Calendar is an abstract base class for converting between a Date object and a set of integer fields such as YEAR, MONTH, DAY, HOUR, and so on.”[14] In addition to easy manipulation of date and time, the calendar class automatically conforms to locale-sensitive information according to what is default for the phone. The following code shows, how we can get the alarm time in milliseconds retrieved from the user’s input from the create alarm activity:

```
if (pm == true) hour += 12;
Calendar calendar = Calendar.getInstance();
calendar.setTimeInMillis(System.currentTimeMillis());
calendar.set(Calendar.HOUR_OF_DAY, hour);
calendar.set(Calendar.MINUTE, minute);
```

Next, the getTimeInMillis() method can be used to retrieve the time in milliseconds as set by the above code. Finally, the setExact() method of Alarm Manager can be used to “Schedule an alarm to be delivered precisely at the stated time.”[11] with the following code:

```
alarm_manager.setExact(AlarmManger.RTC_WAKEUP, calendar.getTimeInMillis(),
pending_intent);
```

It should be noted that there is a distinction between the AlarmManager’s set() method and the setExact() method. Unlike the set() method, the setExact() method is not battery friendly if repeatedly used. The set() method allows the Android OS to decide the best time to wake up the device based on multiple factors that includes if any other apps have also requested to be woken up during the vicinity of the time set by this app. This allows Android to wake up once and perform multiple operations together. With setExact(), however, the system ignores any other applications and broadcasts an intent precisely at the set time. Therefore, multiple wake ups may be required in a small time window making it damaging to the battery. Since Good Morning is an alarm application, it is one of the rare cases where using setExact() is justified.

The pending Intent created above is passed in the Alarm Manager so that the AlarmReceiver class and its onReceive() Method is executed when the Alarm is delivered. As mentioned above, the AlarmReceiver class will start the RunningAlarm Activity. The Android Developer Reference describes “An activity is a single, focused thing that the user can do.”[15] By following the spec, RunningAlarm Activity consists only of two buttons named Snooze and Dismiss. As soon as the RunningAlarm Activity starts up, it starts playing the alarm tone set by the user and retrieved

from the Settings' PreferenceManager. The following sample code shows how the tone is played by the Good Morning app:

```
mMediaPlayer = new MediaPlayer();  
mMediaPlayer = MediaPlayer.create(this, R.raw.sound1);  
mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);  
mMediaPlayer.setLooping(true);  
mMediaPlayer.start();
```

The two possible interactions are Snooze and Dismiss. If the user presses the Snooze button, the Content Provider is queried via the following code to retrieve the Snooze time set by the user:

```
mCursor = getContentResolver().query(  
    UserAlarms.Times.CONTENT_URI,          // The content URI of the alarm table  
    mProjection,                            // The columns to return for each row  
    mSelectionClause                        // Selection criteria  
    mSelectionArgs,                        // Selection criteria  
    mSortOrder);
```

Once it retrieves the data from the cursor, the alarm manager object is called once again. However, this time the `setRepeating()` method is used to schedule a repeating alarm with the previous pending intent and the snooze time as parameters. This allows the `RunningAlarm` Activity to be started again (and the alarm tone to be run) once the snooze period has been finished. A `CursorLoader` must be used “to perform the cursor query on a background thread so that it does not block the application's UI.” [17] Blocking the application UI means that any button presses during the query would not be registered by the application and result in a frustrating user experience.

The app dismisses itself immediately after the user presses one of the two buttons by calling:

```
Activity.finish();
```

So that the user or the Android System can go back to its initial state and thereby releasing any resources. “When calling `finish()` on an activity, the method `onDestroy()` is executed this method can do things like:

1. Dismiss any dialogs the activity was managing.

2. Close any cursors the activity was managing.
3. Close any open search dialog.”[16]

This is vital since having a cursor pointing to the Content Provider means that the Activity will consume unnecessary resource. Not doing this would have been fine in the case where the application is simply placed in the background but in this case, the `setRepeating()` method would create a new instance of the `RunningAlarm` activity after the snooze time has passed regardless of it sleeping in the background.

According to the specified requirements, the Good Morning app must inform the user of any missed notifications by speaking aloud. As can be seen from the Android Activity Lifecycle, once an Activity has been Destroyed there is no way to perform such actions. The developers at Large Software also understand that having an activity with an UI to only speak of notifications would be a subpar user experience. A better alternative would be an app which can speak aloud notifications in the background while the user can perform other tasks with their smartphone.

Therefore, on pressing the Dismiss button, a `NotificationPlayerService` is started in the background. This `NotificationPlayerService` extends from the `Service` class and implements the `TextToSpeech` interface to speak aloud notifications. This service must first be registered to the Android System via the `AndroidManifest.xml` file by adding the following lines:

```
<service android:name=".NotificationPlayerService  
        android:enabled="true"></service>
```



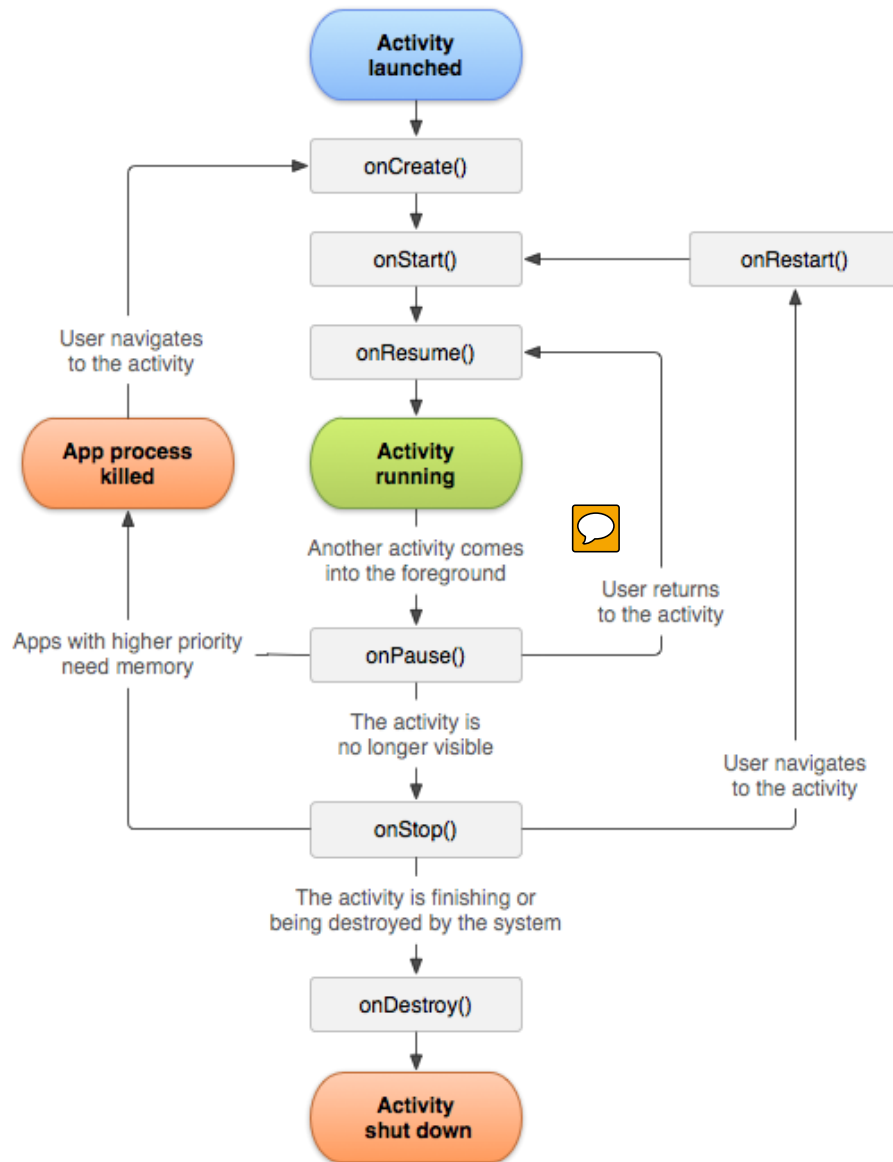


Figure 41: Android Activity Lifecycle[18]

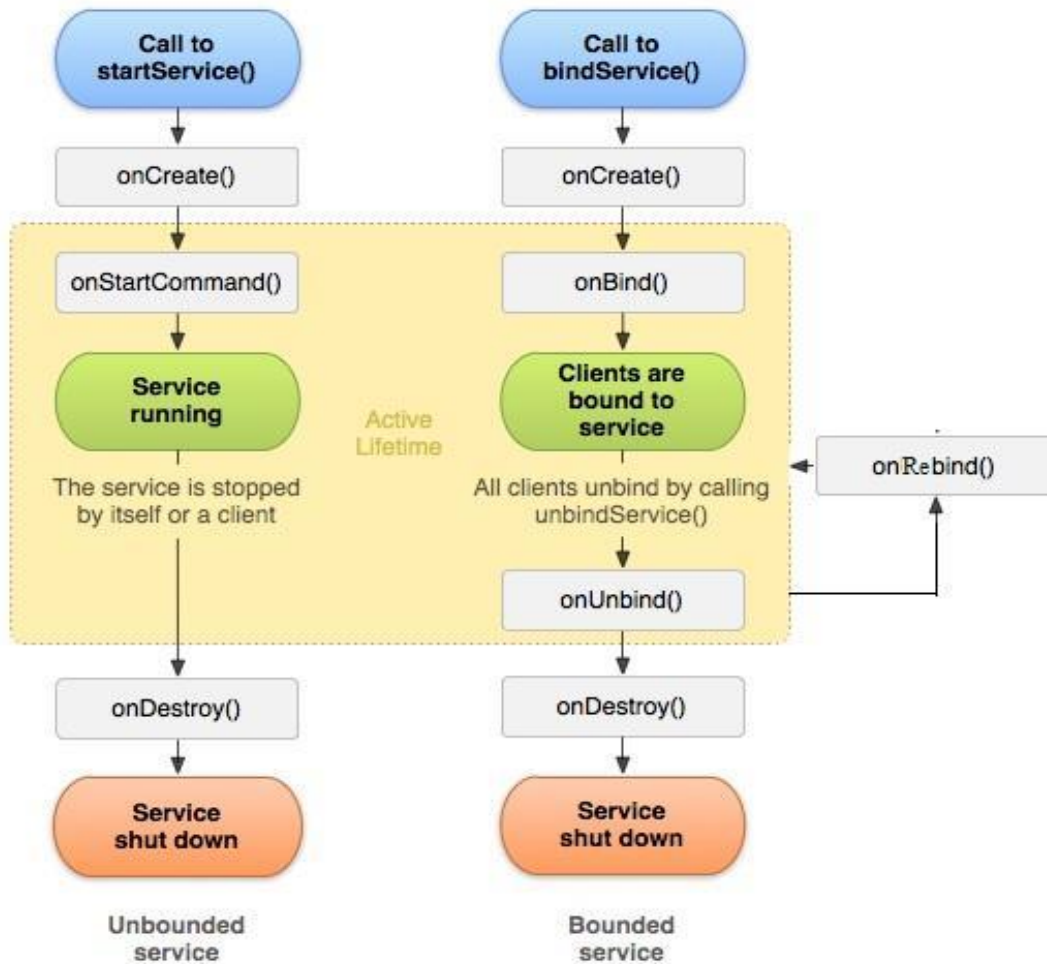


Figure 42: Android Service Lifecycle[19]

The NotificationPlayerService would be an unbounded service and will be started by the RunningAlarm Activity and it will destroy itself once all the unread notifications have been spoken. As the above Service Lifecycle figure suggests, having a lifecycle untied to an UI activity allows the app to achieve the functional requirement goal. The Notifications for the NotificationPlayerService will also be read via a CursorLoader from the Notifications content provider. The mechanism for storing notifications in this content provider as they are received by the Android system will be discussed in another section of this document.

The NotificationPlayerService leverages Android’s built-in TextToSpeech interface. This interface “Synthesizes speech from text for immediate playback or to create a sound file.”[20] Any service requires the `onCreate()`, `onDestroy()` and `onBind()` abstract methods to be implemented. During the `onCreate()` method, the content provider is queried for all the Notification information. This information is then appended to a String to be used by the `onInit()` method. The `onInit()` method is required for the TextToSpeech interface to be

implemented. In this method, the language is set to English and the actual speech synthesis is started. The following code shows how it is done in the Good Morning application:

```
mTts.setLanguage(Locale.US);  
mTts.speak(notificationText, TextToSpeech.QUEUE_FLUSH, null);
```

An `onUtteranceCompleted()` method is also implemented with a call to the `stopSelf()` function. This method is automatically called by the Android system when the speaking the text has been completed. This ensures that the service is destroyed as soon as playing back the speech has been completed.

The `onCreate()` function also calls another method called the `showNotification()`. The `showNotification()` method uses the Notification Builder to create a notification for the Good morning app itself so that the user can cancel the `NotificationPlayerService`, if they choose to do so. The following code shows how the `showNotification()` is implemented [21]:

```
// Set the info for the views that show in the notification panel.  
Notification notification = new Notification.Builder(this)  
    .setSmallIcon(R.drawable.good_morning) // the status icon  
    .setTicker(text) // the status text  
    .setContentTitle(getText(R.string.local_service_label)) // the label of the entry  
    .setContentText(text) // the contents of the entry  
    .setContentIntent(cancelNotificationPlayer) // The intent to send when the entry is  
clicked  
    .build();  
  
// Send the notification.  
mNM.notify(NOTIFICATION, notification);
```

Since only an Intent can be passed to a notification, a temporary Activity class is created to call `stopService()` where the `NotificationPlayerService` is passed in as the parameter.

## LIFX smart light

The official LIFX LAN protocol provides a very low-level API that is cumbersome to use from a high-level perspective of the Good Morning application. Although, the HTTP API is developer friendly, the requirements documents specified that the LIFX smart light bulbs must work only inside a local wireless network of the user's home. Large Software's research showed that there existed an official Android SDK that is currently unsupported by the LIFX team. However, there is a community supported java sdk for LIFX that is widely used with sufficient documentation. It is called the lifx-sdk-java and can be found at: <https://github.com/besherman/lifx-sdk-java>

Research by Large Software developers suggested that LIFX smart light bulbs contain clocks inside them and supports the setting of alarms. Although, using the built-in alarm feature was the initial approach, it failed to meet the requirement specifications. The built-in LIFX alarm can only have up to two alarms per light [22]. The built-in alarm can also take in a single color as the parameter so the desired breathing or pulse effect as per the requirements would not be possible.

As a result, a different approach was taken via the RunningAlarm Activity. Firstly, the lifx-sdk-java was added to the project as a dependency via Gradle. As mentioned earlier, the RunningAlarm Activity is started by the Alarm Manager at precisely the specified time and it goes on to playing the pre-set Alarm Tone. Similar to checking for the Alarm Tone, RunningAlarm Activity first checks the Settings PreferenceManager, if the LIFX Light bulb option is enabled or not and if so, what lighting effect to send to the light bulbs.

The first step is to discover the light bulbs that are connected to the current wireless network. To do so, an UDP broadcast is sent out to port 56700 and any LIFX light bulb connected to the network should respond with a Device::StateService message.



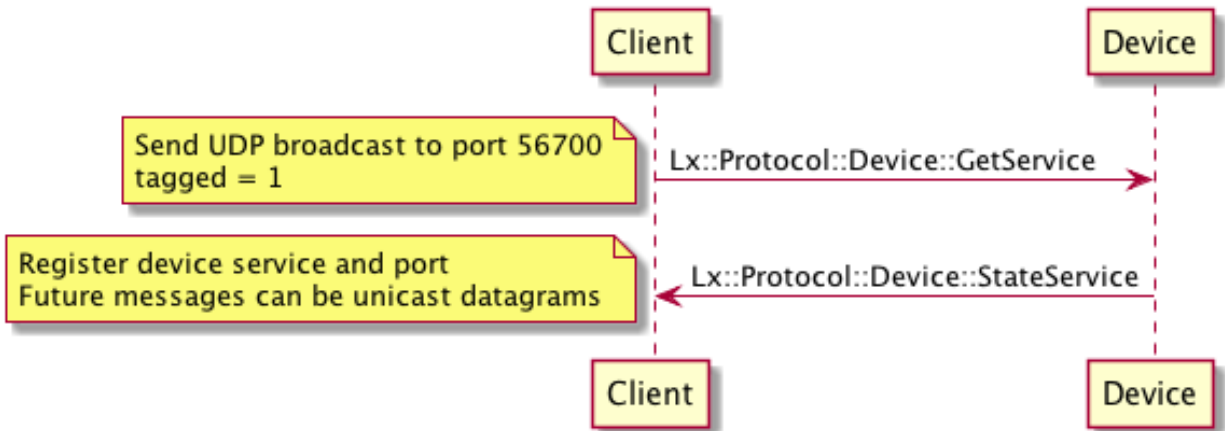


Figure 43: LIFX Light bulb Discovery[14]

Any network activity such as sending out broadcasts and waiting for a reply is a UI blocking operation. To ensure a smooth experience to the user, a new LIFX class is created that inherits from the `AsyncTask` class. “`AsyncTask` enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.”[23] The `AsyncTask` consists of a `doInBackground()` method that must be overridden. Inside the `doInBackground()` method, the following code is executed:

```
LFXClient client = new LFXClient();
```

The `getLights()` method of the client object consists of all the currently connected LIFX Light bulbs. After connections have been established, colors or temperatures for each light can be easily set by following the request/response protocol as shown below:

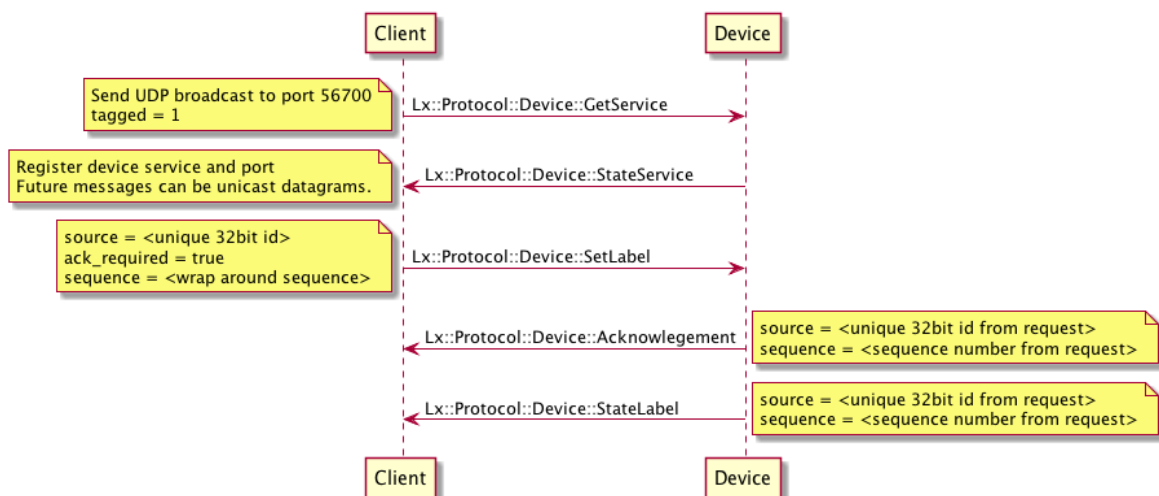


Figure 44: Request with acknowledgement and response[14]

The sample code for a pulse effect is given below which randomly changes colors of all lights[25]:

```
private final List<Color> colors = Arrays.asList(
    Color.BLUE, Color.CYAN, Color.GREEN, Color.MAGENTA, Color.ORANGE,
    Color.PINK, Color.RED, Color.YELLOW);

Color color = null;
do {
    color = colors.get(random.nextInt(colors.size()));
} while(color == last);

for(LFXLight light: lights) {
    if(!light.isPower()) {
        light.setPower(true);
    }
    light.setColor(color);
}
last = color;
```

The sample code for a breathe effect where the lights are turned on and go through the different color temperatures are as follows[16]:

```
temp += 500;

if(temp > MAX_TEMP) {
    temp = MIN_TEMP;
}
System.out.println(temp);
LFXHSBKColor color = new LFXHSBKColor(Color.WHITE, temp);
for(LFXLight light: lights) {
    if(!light.isPower()) {
        light.setPower(true);
    }
    light.setColor(color);
}
```

The AsyncTask class containing the above methods is bound to the RunningAlarm Activity. Therefore, as soon as the Dismiss or Snooze button is pressed any light activity will be immediately stopped. It will only run again if the RunningAlarm Activity is started again as the result of a snooze or a new alarm intent from the Alarm Manager.

## Pebble Smart Watch

As specified in the Requirements Specification, the Pebble smart watch will not be used for sleep tracking at this early stage of the product and is an extended feature of the product. At this stage, the Good Morning app will only send a notification to the user's Pebble smartwatch whereby the user can snooze or dismiss the alarm from the smart watch. Large Software hopes that the nudge from the Pebble smart watch would help heavy sleepers wake up.

For the actionable notification to work, the user must have Pebble Time app connected to their Pebble smart watch and enable third party notifications. Connecting to Pebble is out of scope for this document and can be found in the manual for the user's smartwatch. The user must also ensure that Pebble Smart Watch Integration is enabled from the Good Morning App.

The actionable notification is created from the RunningAlarm Activity via the NotificationBuilder as follows [27]:

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(context);  
builder.setSmallIcon(R.drawable.ic_launcher);  
builder.setContentTitle("Wearable Pomodoro");  
builder.setContentText("Starting the timer ...");
```

Next, two pending intents are created, one to perform the snooze action and another to start the NotificationPlayerService on Dismiss. Both the intents can be added as actions. However, the following wearable API available from the latest Android SDK must be used so that the button pressed by a user on the Pebble smart watch can be received by the Good Morning app.

```
builder.extend(new NotificationCompat.WearableExtender().addAction(Snooze));  
builder.extend(new NotificationCompat.WearableExtender().addAction(Dismiss));
```

Finally, the NotificationManager can be used to produce the notification using the notify() method from the RunningAlarm Activity. Now, the user can perform the Snooze/Dismiss action from their Pebble Smartwatch.

# Testing

In order to ensure that all requirements are met by the Good Morning application thorough testing and validation must occur.

## Testing Schedule

To verify that the Good Morning application performs as specified various testing must be performed. This testing must adhere to a strict schedule to ensure that the application is delivered by the specified date with all intended functional and non-functional requirements being met. LARGE software had derived the following testing schedule from suggestions from the client as well as our fragment functionality:


	March												
Functionality	19	20	21	22	23	24	25	26	27	28	29	30	31
Navigation	Unit Testing		Functional Testing				Integration Testing						
Data Management	Unit Testing		Functional Testing				Integration Testing						
Alarm			Unit Testing		Functional Testing		Integration Testing				Performance Testing		Acceptance Testing (Demo)
Virtual Assistant					Unit Testing		Functional Testing		Integration Testing				
Integrations						Unit Testing		Functional Testing		Integration Testing			

Figure 45: Testing Schedule

## Unit Testing Schedule

Initially LARGE software will unit test the Navigation and Data Management functionality in parallel for two days, involving Luke McLaren as well as Adam Kroon due to their expertise on the subjects. From there the Alarm will be unit tested for two days by Rahat Mahbub who has the responsibility over development. Finally the unit testing of the Virtual assistant functionality will begin followed the day after by the unit testing for integration functionality, this will be done by the entire team as the majority of development shall be completed at this time. Note that the integrations are given shorter testing periods due to the lack of necessity in the final design.

## Functional Testing Schedule

Once the unit tests have been completed, the functionality of modules can be tested for. LARGE software will test the Navigation and Data Management functionality in parallel for two days, again involving Luke McLaren as well as Adam Kroon. The following two days are used for testing the functionality of the alarm, again by Rahat Mahbub. Finally the virtual assistant and integration functionality will be tested in practical parallel by the team.

## Integration Testing Schedule

Once the testing of modular function has been completed, the modules must be tested to ensure that they can integrate with each other. LARGE software will test that the navigation, data management and alarm modules are integral in parallel for two days, this testing will involve Adam Kroon, Rahat Mahbub and Luke McLaren. These tests have been scheduled in parallel because of the inherent relation between them, since the alarm cannot function without data management, and navigation; and vice versa. Finally the virtual assistant and the integrations will be tested for a day and a half in parallel since they are both minor in scale.

## Performance Testing Schedule

After all modules have been tested to ensure that they work as a complete application LARGE software will be able to begin to test the performance of the overall system. Considering that there is not many performance dependent functionality within the application only a day and a half has been allocated for this period. The entire team is to take part in performance testing all aspects of the application since a majority of development will be complete by this time.

## Acceptability Testing Schedule

Once performance of the application has been tested and corrected, the final testing of acceptability is scheduled. This testing involves presenting the Good Morning application to the clients and performing a walkthrough with them. The walkthrough will ensure that all client requirements specified have been met. Once the client has confirmed the acceptability of the application (less than %5 change in system required between sessions), LARGE Software can complete the development of the Good Morning application.

## JUnit Tests

JUnit is “the most popular and widely-used testing” [28] framework for Java. JUnit tests will be created for each method in the Good Morning application. JUnit test will be written in a modular fashion, so that future tests may be integrated as part of the testing suite as new features are implemented.

Android Studio is the IDE of choice for developing the Good Morning application. It supports executing tests on the local machine using the JUnit test runner, in addition to, executing instrumented tests on an Android device. Running local tests are ideal for tests that do not depend on the Android Framework. As a result, they can be tested quickly on the JVM. Some methods that will be tested locally are listed below:

getTimeInMillis() and  
playMedia()

Due to strong dependency on the Android framework, most tests will be done after Instrumentation. Having Instrumentation information allows unit tests to be run under the correct context of the application. Instrumented unit tests are also the only way to run unit tests on the UI or database of the app. The test framework below shows how Instrumentation allows the ability to run tests dependent on the Android framework.

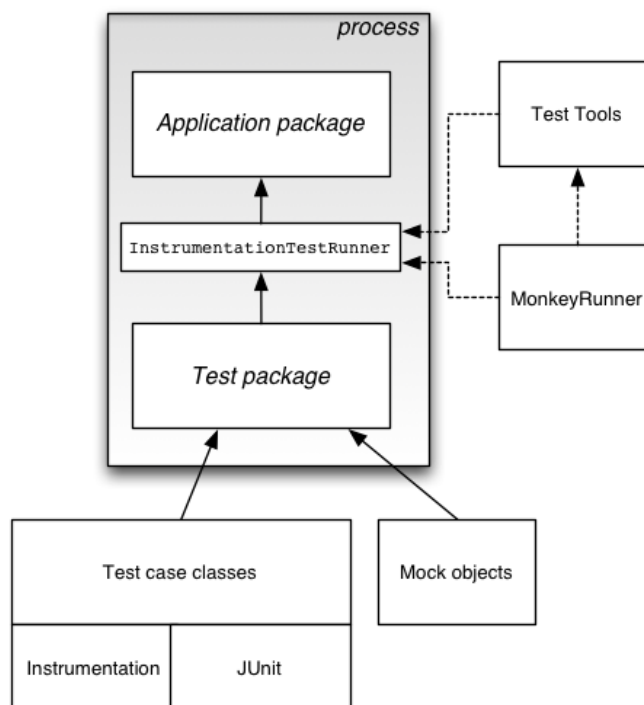


Figure 46: Android Test Framework

It is vital that any methods related to the content provider or database are correct and that any future change to the table or query methods doesn't corrupt the data since that could result waking up a user at the middle or worse, not waking up a user at all. To start off, a TestDb class is created that inherits from AndroidTestCase. The first method that needs to be overridden is the setUp() method. Inside the setUp() method, the database file is completely deleted to ensure that the test device doesn't contain any existing data.

Next, a testCreateDb() method is created where a database containing the appropriate tables are created as specified under the Database section of this document. The following code retrieves a cursor that can be iterated to retrieve all the tables in the database.

```
Cursor c = db.rawQuery("SELECT name FROM sqlite_master WHERE type='table'", null);
assertTrue("Error: This means that the database has not been created correctly",
    c.moveToFirst());
```

Next, a HashSet is created which retrieves table names from the Content Provider as defined by the Database Contract. The created table and the contract is matched as follows:

```
do {
    tableNameHashSet.remove(c.getString(0));
} while( c.moveToNext() );

assertTrue("Error: Your database was created without both the Notification entry and Alarm entry tables",
    tableNameHashSet.isEmpty());
```

Similar code is used to ensure that the correct columns have been created.

## Espresso

Espresso will be used to test the UI of the Java application. These are to test user flows within the application. With Espresso, reliability and efficiency can be tested, as UI elements may be manipulated accurately and consistently during testing.

The database tests are continued with Espresso by programmatically creating an alarm with the following code:

```
onView(R.id.create_alarm).perform(click());
onView(withText("MON")).perform(click());
onAdapterView(R.id.hour)
    .atPosition(1)
    .perform(click());
onView(R.id.button).perform(click());
```

The above code would press the Floating action bar on the home page and create an alarm with Monday as the date and 1 AM as the time for an alarm. At this point, the alarm table should only have one entry containing Monday as the date and 1 AM for the alarm time. The data can be retrieved as a content value using a cursor and an assert function would check if both the values match. The test will fail otherwise.

The above is only a sample test for the database. Espresso is used throughout the application to perform complete user work flow tests as well. Espresso is also used to test the UI thread synchronization to improve reliability, and view matching to ensure that the view is not hiding elements and assert usability.

## Monkey

Monkey is a tool that acts as a pseudo-random UI tester. When ran, it will randomly make keystrokes and screen presses. These are pseudo-random as the tests can be repeated by using the same random seed. Monkey will be used for fuzz testing and for reliability testing, and to ensure the application will not unexpectedly close if erroneous inputs are recorded.

## Walkthrough Functional Test Cases

The following test cases are essential requirements from SoftStart. These are tests that can be completed as walkthroughs, and shown to the client to be verified. Walkthrough tests may also be confirmed and handled by the client as well. Espresso may be used to automate the UI tests to be shown to the client. Large Software believes that applying Behaviour Driven Development principles ensures high quality software. As a result, all features of Good Morning application have been documented with Gherkin tests to ensure a very high test coverage.

## Alarm Interface

The alarm interface will be the first part of the application to be tested since it is the most important and integral feature. Testing will take place for 3 days, with “Create” being the first feature tested. Unit tests for the source code will be first, then UI and walkthroughs, and finally functional tests to ensure the application is as robust as possible.



## Create Alarm

Inputs	User creates alarm from home screen
Outputs	Alarm is created, and enabled
Exceptions	If alarm is already created on this day, error is displayed.

### Gherkin:

Feature: Create a new alarm for the alarms list

Scenario: User wants to create a new alarm

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the alarms page

When:

1. User presses *Floating Action Button* for adding a new alarm
2. User selects "Monday" and "Thursday" dates using buttons at top of screen
3. User selects "9" using the hour drop down menu
4. User selects "30" using the minute drop down menu
5. User selects "AM" using the AM/PM drop down menu
6. User selects "Missed Calls", "Missed Texts", and "Events" checkboxes in the *Notifications* section
7. User selects "10 Minute" snooze interval using the slider in the *Snooze Interval* section
8. User presses "OK"

Then:

1. The alarm rings at the next 9:30AM Monday and Thursday
2. The user snoozes the phone and waits for 10 minutes
3. The alarm rings again at 9:40AM
4. The alarm is dismissed by the user
5. The user's missed calls and texts, and events for the day are read aloud by the phone

## Delete Alarm

Inputs	User opens dropdown on an alarm User selects delete
Outputs	Prompt is displayed that gives users the option of "Confirm" or "Cancel" User selects an option
Exceptions	None

## Gherkin:

Feature: Delete a configured alarm in the alarms list

Scenario: User wants to delete an alarm

### Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the alarms page
4. User has 9:30AM Monday/Thursday alarm in their existing alarms

### When:

1. User selects the down arrow on the alarm for 9:30AM Monday/Thursday
2. User selects the delete button
3. User ignores the undo deletion *Snackbar*

### Then:

1. The alarm is deleted from the alarm page
2. The alarm does not sound at 9:30AM Monday/Thursday

### Undo Deletion of Alarm

Inputs	User follows path for test 3 User selects undo after deleting the alarm
Outputs	Alarm returns back to original state, before being deleted
Exceptions	None

### Gherkin:

Feature: Undo a deleted alarm in the alarms list

Scenario: User wants to undo a deleted alarm

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the alarms page
4. User has just finished deleting the 9:30AM Monday/Thursday alarm

When:

1. User selects "Undo" from the undo deletion *Snackbar*

Then:

1. The alarm is on the alarm page again
2. The alarm sounds at 9:30AM Monday/Thursday

### Settings Page

Inputs	User opens dropdown on an alarm User selects configuration for an alarm User changes settings
Outputs	New settings should be active immediately If the time for an alarm was changed
Exceptions	If no notifications are selected, then the alarm should simply be the sound and nothing else (similar to the stock android alarm clock, with the added snooze functionality).

*Gherkin:*

Feature: Change an alarm on the alarms list

Scenario: User wants to alter an alarm

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the alarms page
4. User has 9:30AM Monday/Thursday alarm in their existing alarms

When:

1. User selects the down arrow on the alarm for 9:30AM Monday/Thursday
2. User selects the "Configure" button
3. User selects "Tuesday"
4. User selects "10" using the hour drop down menu
5. User selects "45" using the minute drop down menu
6. User selects "PM" using the AM/PM drop down menu
7. User deselects "Missed Calls", "Missed Texts", and "Events" checkboxes in the *Notifications* section
8. User selects "Weather" checkbox in the *Notifications* section
9. User selects "30 Minute" snooze interval using the slider in the *Snooze Interval* section
10. User presses "OK"

Then:

1. The alarm rings at the next 10:45PM Tuesday
2. The user snoozes the phone and waits for 30 minutes
3. The alarm rings again at 11:15PM
4. The alarm is dismissed by the user
5. The user's weather for the day are read aloud by the phone

## Snooze

Inputs	Alarm is going off User selects snooze
Outputs	Alarm is snoozed for specified time for given alarm
Exceptions	None

## Gherkin:

Feature: Snooze a sounding alarm

Scenario: User wants to snooze a sounding alarm

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User has created a 9:30AM Monday/Thursday 10 Minute Snooze alarm

When:

1. The alarm sounds at 9:30AM Monday or Thursday
2. The user snoozes the phone and waits for 10 minutes

Then:

1. The alarm sounds again at 9:40AM

## LIFX Smart Light Integration

### LIFX Light Configuration

Inputs	LIFX Smart Light is not Configured Phone and Smart Light are on the same LAN Phone is connected to internet
Outputs	Phone is connected to smart light Smart light is configured
Exceptions	None

### Gherkin:

Feature: LIFX Smart Light Configuration

Scenario: User wants to connect their LIFX Smart Light to the Good Morning application

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the settings page

When:

1. User presses the toggle button in the *LIFX Smart Light* section
2. User is prompted to configure their LIFX Smart Light in the LIFX Application
3. User completes configuring their LIFX Smart Light
4. User returns to the Good Morning application

Then:

1. The toggle button in the *LIFX Smart Light* section on the settings page is in the “On” position
2. User selects the “Breathe” effect
3. LIFX Smart Light is configured

### LIFX Is Triggered When User Alarm Sounds

Inputs	Phone is connected to same LAN as LIFX Smart Light Smart light is configured Alarm is configured
Outputs	Alarm should trigger light
Exceptions	Should not trigger if phone is not on the same LAN during alarm time

### Gherkin:

Feature: LIFX Light Activation

Scenario: User wants to use their LIFX Smart Light with the Good Morning application

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User has configured their LIFX Smart Light with “Breathe” effect
4. User has created a 9:30AM Monday/Thursday 10 Minute Snooze alarm

When:

1. The alarm sounds at 9:30AM Monday or Thursday

Then:

1. The LIFX Light brightens with a “Breathe” effect

### Pebble Smart Watch Configuration

Inputs	Pebble Smart Watch is not Configured
Outputs	Phone is connected to Pebble using Bluetooth Smart watch is configured
Exceptions	None

### Gherkin:

Feature: Pebble Smart Watch Configuration

Scenario: User wants to connect their LIFX Smart Light to the Good Morning application

Given:

1. User has Good Morning application installed
2. User has Good Morning application opened
3. User is on the settings page

When:

1. User presses the toggle button in the *Pebble Smart Watch* section
2. User is prompted to connect to their Pebble Smart Watch with Bluetooth
3. User types the code given on phone screen on the watch
4. User types the code given on the watch screen on the phone
5. User returns to the Good Morning application

Then:

1. The toggle button in the *LIFX Smart Watch* section on the settings page is in the “On” position

2. User selects the "Snooze/Dismiss Alarm" button
3. Pebble is configured

#### Pebble Smart Watch Dismiss

Inputs	Pebble Smart Watch is Configured Phone and Pebble are connected using Bluetooth Alarm sounds User presses "Dismiss" on their Pebble Smart Watch
Outputs	Alarm is dismissed
Exceptions	None

#### Gherkin:

Feature: Dismiss sounding alarm using Pebble Smart Watch

Scenario: User wants to dismiss their Good Morning alarm using their Pebble Smart Watch

##### Given:

1. User has Good Morning application installed
2. User has configured their Pebble Smart Watch with "Dismiss/Snooze Alarm"
3. User is connected to their Pebble Smart Watch
4. User has created a 9:30AM Monday/Thursday

##### When:

1. Alarm sounds at 9:30AM Monday or Thursday

##### Then:

1. User presses "Dismiss" on their Pebble Smart Watch
2. Alarm is dismissed



## Team Planning

Team Member	Team Role	Development Responsibilities
Luke McLaren	Web Master	System Architect
Adam Kroon	Project Lead	Integrations
Rahat Mahbub	Toolsmith	User Interface
Graeme Bates	Documentation Expert	Integrations

## References

- [1] "LIFX/lifx-protocol-docs", *GitHub*, 2015. [Online]. Available: <https://github.com/LIFX/lifx-protocol-docs/tree/master/workflows>. [Accessed: 04- Mar- 2016].
- [2] "Activities | Android Developers", *Developer.android.com*, 2016. [Online]. Available: <http://developer.android.com/guide/components/activities.html>. [Accessed: 18- Mar- 2016].
- [3] "Most widely deployed SQL database engine,". [Online]. Available: <https://www.sqlite.org/mostdeployed.html>. Accessed: Mar. 17, 2016.
- [4] "Android SQLite database Tutorial," *www.tutorialspoint.com*, 2016. [Online]. Available: [http://www.tutorialspoint.com/android/android\\_sqlite\\_database.htm](http://www.tutorialspoint.com/android/android_sqlite_database.htm). Accessed: Mar. 17, 2016.
- [5] "AsyncTask,". [Online]. Available: <http://developer.android.com/reference/android/os/AsyncTask.html>. Accessed: Mar. 18, 2016.
- [6] "Processes and threads,". [Online]. Available: <http://developer.android.com/guide/components/processes-and-threads.html>. Accessed: Mar. 18, 2016.
- [7] EntityFrameworkTutorial, "What is entity framework?," 2015. [Online]. Available: <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>. Accessed: Mar. 17, 2016.
- [8] "Database schema," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: [https://en.wikipedia.org/wiki/Database\\_schema](https://en.wikipedia.org/wiki/Database_schema). Accessed: Mar. 17, 2016.
- [9] "Saving data in SQL databases,". [Online]. Available: <http://developer.android.com/training/basics/data-storage/databases.html>. Accessed: Mar. 17, 2016.
- [10] "Content provider basics,". [Online]. Available: <http://developer.android.com/guide/topics/providers/content-provider-basics.html>. Accessed: Mar. 17, 2016.
- [11] "AlarmManager,". [Online]. Available: <http://developer.android.com/reference/android/app/AlarmManager.html>. Accessed: Mar. 19, 2016.
- [12] "BroadcastReceiver,". [Online]. Available: <http://developer.android.com/reference/android/content/BroadcastReceiver.html>. Accessed: Mar. 19, 2016.

- [13] "PendingIntent,". [Online]. Available: <http://developer.android.com/reference/android/app/PendingIntent.html>. Accessed: Mar. 19, 2016.
- [14] "Calendar,". [Online]. Available: <http://developer.android.com/reference/java/util/Calendar.html>. Accessed: Mar. 19, 2016.
- [15] [Online]. Available: <https://developer.android.com/reference/android/app/Activity.html>. Accessed: Mar. 19, 2016.
- [16] "What exactly activity.finish() method is doing?," 2016. [Online]. Available: <http://stackoverflow.com/questions/10847526/what-exactly-activity-finish-method-is-doing>. Accessed: Mar. 19, 2016.
- [17] "CursorLoader,". [Online]. Available: <http://developer.android.com/reference/android/content/CursorLoader.html>. Accessed: Mar. 19, 2016.
- [18] [Online]. Available: [https://developer.android.com/images/activity\\_lifecycle.png](https://developer.android.com/images/activity_lifecycle.png). Accessed: Mar. 19, 2016.
- [19] [Online]. Available: [http://www.klebermota.eti.br/wp-content/service\\_lifecycle1.png](http://www.klebermota.eti.br/wp-content/service_lifecycle1.png). Accessed: Mar. 19, 2016.
- [20] "TextToSpeech,". [Online]. Available: <http://developer.android.com/reference/android/speech/tts/TextToSpeech.html>. Accessed: Mar. 19, 2016.
- [21] "Service,". [Online]. Available: <http://developer.android.com/reference/android/app/Service.html>. Accessed: Mar. 19, 2016.
- [22] besherman, "Besherman/lifx-sdk-java," GitHub, 2016. [Online]. Available: <https://github.com/besherman/lifx-sdk-java>. Accessed: Mar. 19, 2016.
- [23] "AsyncTask,". [Online]. Available: <http://developer.android.com/reference/android/os/AsyncTask.html>. Accessed: Mar. 19, 2016.
- [24] "Workflow diagrams · LIFX LAN Protocol," LIFX LAN Protocol. [Online]. Available: <http://lan.developer.lifx.com/docs/workflow-diagrams>. Accessed: Mar. 19, 2016.

[25] besherman, "Besherman/lifx-sdk-java," GitHub, 2015. [Online]. Available: <https://github.com/besherman/lifx-sdk-java/blob/master/src/main/java/com/github/besherman/lifx/examples/lights/LightEx10RandomlyBlinkLights.java>. Accessed: Mar. 19, 2016.

[26] besherman, "Besherman/lifx-sdk-java," GitHub, 2015. [Online]. Available: <https://github.com/besherman/lifx-sdk-java/blob/master/src/main/java/com/github/besherman/lifx/examples/lights/LightEx12ChangeColorTemp.java>. Accessed: Mar. 19, 2016.

[27] "Send a smile with Android actionable notifications,". [Online]. Available: <https://developer.pebble.com/blog/2014/12/19/Leverage-Android-Actionable-Notifications/>. Accessed: Mar. 19, 2016.

[28] "Testing concepts,". [Online]. Available: [http://developer.android.com/tools/testing/testing\\_android.html](http://developer.android.com/tools/testing/testing_android.html). Accessed: Mar. 19, 2016.

# C3 Evaluation of the Technical Design Document

## Google Calendar Integration

1. Create valid API key.

Inputs	User signs up with google account
Outputs	User receives API key Calendar functionality is unlocked
Exceptions	If password or username are incorrect they do not receive a key

2. Get events from current day.

Inputs	You have connected with the google API
Outputs	Events are updated within the app, and the app now reads you your upcoming events.
Exceptions	User does not have any events on current day

3. Chronologically order the events.

Inputs	Application requests events
Outputs	Events are ordered in app and read to the user in chronological order
Exceptions	User does not have any events on current day

4. Validate how many times we can request data from the calendar API.

Inputs	Many events exist for current day Application requests events
Outputs	Events are added to applications event database
Exceptions	Too many requests are made and API stops giving responses

## Alarm Interface

1. Create an alarm, the alarm should now be enabled.

Inputs	User creates alarm from home screen
Outputs	Alarm is created, and enabled
Exceptions	If alarm is already created on this day, error is displayed.

2. Select different notification settings and times.

Inputs	User opens dropdown on an alarm User selects configuration for an alarm User changes settings
Outputs	New settings should be active immediately
Exceptions	If no notifications are selected, then the alarm should simply be the sound and nothing else (similar to the stock android alarm clock, with the added snooze functionality).

3. Deleting an Alarm

Inputs	User opens dropdown on an alarm User selects delete
Outputs	Prompt is displayed that gives users the option of "Confirm" or "Cancel"
Exceptions	None

4. Undoing Deletion

Inputs	User follows path for test 3 User selects undo after deleting the alarm
Outputs	Alarm returns back to original state, before being deleted
Exceptions	None

## 5. Snooze

Inputs	Alarm is going off User selects snooze
Outputs	Alarm is snoozed for specified time for given alarm
Exceptions	None

## Integration with LIFX Smart Light

### 1. Configure LIFX light with app.

Inputs	LIFX Smart Light is not Configured Phone and Smart Light are on the same LAN Phone is connected to internet
Outputs	Phone is connected to smart light Smart light is configured
Exceptions	Invalid username or password does not allow configuration

### 2. LIFX light should not be triggered if user is not connected to local network (ie. at friends house).

Inputs	Phone is not connected to same LAN as Smart Light Smart light is configured Phone is connected to internet
Outputs	Alarm should not trigger light
Exceptions	None

### 3. Toggle integration on/off

Inputs	LIFX Smart Light is Configured Phone is on same LAN and Smart Light Phone is connected to internet User Toggles smart light integration
Outputs	Integration is toggled correctly Smart light is still configured but not connected

Exceptions	Invalid username or password will not allow configuration
------------	---

4. Select breathe/pulse effects.

Inputs	LIFX Smart Light is Configured Phone is on same LAN and Smart Light Phone is connected to internet User Toggles Breathe or Pulse
Outputs	Breathe or Pulse is correctly selected
Exceptions	None

## Implementation Specific Tests

### Alarm Integration Tests (Page 31)

Inputs	Android Alarm Library WakefulBroadcastReceiver Alarm Interaction Fragment
Outputs	Alarm Dismissed
Exceptions	None

Alarms should be able to be dismissed or snoozed based on interaction with the default Android alarm library. Mock the library and write tests.

### Text to speech (Page 32)

Inputs	Android Text To Speech Android Virtual Assistant Text to be read
Outputs	Text is read out and is understandable Volume is appropriate
Exceptions	On invalid text the text to speech does nothing

Text to speech should not be unit tested, it should be tested using quality assurance personnel. Make sure that text to speech integration is not overlooked.



## Database Tests (Page 62)

### Test for Injections

Inputs	SQLite database connection Query String
Outputs	None
Exceptions	If user attempts a SQL injection the database should error

SQLite should not allow any code to be executed from user queries. Make sure to test that all user input is sanitized before a query is sent to the database.

## Removal of Tables (Page 63)

Inputs	SQLite database connection Query String
Outputs	Removed Table
Exceptions	If a user tries to remove a table that is not initialized it should error

Removing of tables should be handled gracefully. The removal of tables should be a solid and scalable interaction between the application and the database. Should error on removal of a non existent table