ARCHITECTURAL DESIGN DOCUMENT
Software Engineering Project Course
Sleep Cycle Alarm Clock
University of Helsinki
http://www.cs.helsinki.fi/en/courses/581260

Design and programming by
Juho Gävert
Tero Huomo
Joosa Kurvinen
Paul Saikko
Lauri Väre

Customer
Beddit Oy

When the text references the customer, it stands for Beddit Oy as a company for which this application was developed. When the text references a user, it means anyone using the application which this documentation regards.

1. Introduction: Architectural Overview

Beddit alarm clock for Android is a custom alarm clock application that uses the Beddit sleep monitoring device to determine the most convenient time to wake up. The user can set the wake up time as well as a time interval during which the alarm may ring. During that interval the Beddit Sleep Cycle Alarm Clock application will query Beddit's server for info about the user's current sleep stage and determine whether or not the user should be woken up. If the current sleep stage is not optimal for waking up, the application will try again every few minutes. The alarm will be launched when the correct sleep phase is detected or the end of the time interval is reached.

The user can choose to have the application display simple info screen about the last night, showing time spent in light and deep sleep after the alarm. From the info screen the user can also indicate his or her perceived sleep quality with the "Feeling good" and "I'm tired" buttons. Clicking these buttons will further redirect the user to the Beddit sleep diary page where more detailed information can be entered. The user can also view the last night's sleep info view by pressing the menu button and selecting the appropriate choice.

The application consists of multiple views. The first view encountered by a user is the login screen which prompts the user log in to his or her beddit account and authorize the application to access the user's data. The login and authorization is done via the familiar Beddit website in a controlled manner. The main view consist of a simple analog clock interface for picking the time and interval for the alarm. Separate buttons for setting and cancelling the alarm are also shown on the main view. The menu button shows an Android-style menu bar with options for opening the application settings, a help screen, and the sleep info view. Alarm entries are saved in Android's native alarm manager. At the set time, the alarm manager will start a background process which will fetch new data from the Beddit API start the alarm or postpone it based on the sleep stage. If the alarm is started, a dialog is displayed on the screen allowing the user to dismiss or snooze the alarm. The alarm tone and phone vibration start when the dialog is shown. The phone is woken up and the keyguard is disabled for the duration of the alarm.

2. System Purpose

2.1 Functional Requirements

- The user must be able to set alarm on and off.
- The user must be able to choose the alarm time and an interval of 0-45 minutes.
- The user must be able to update the alarm time and interval by adjusting the position of the clock hands.
- The user must be able to connect the application to his or her Beddit account.
- The application must wake the user up within the given interval.
- The application must connect to the Beddit servers to get sleep data.
- The application must use up to date sleep data to determine if the user should be woken up.
- The user must be able to cancel the alarm before or after the alarm is triggered.
- The user must be able to set the alarm on snooze.
- The user must be able to set the snooze length.
- The application must set a constant notification for the set alarm.
- The application must be easily localizable for several languages.
- The user must be able to choose the preferred sleep stage to wake from.
- The user must be able to get the last night's sleep data directly to the phone.
- The user must be able to get to the Beddit sleep diary directly from the application.

2.2 Non-functional Requirements

- The Beddit alarm application will be implemented as an Android application.
- The application must support Android 2.2 (Froyo, SDK version 8) and newer.
- The application must take in account as many special situations as possible, e.g. the user speaking on the phone when the alarm goes off
- The user interface must be simple and intuitive to use.
- The user interface should follow the color scheme defined in the Beddit brand book.
- The application must have both dark and light color styles.

3. Structure

For more detailed information on the application code structure see the javadoc at https://github.com/bedditor/ohtu-Alarmclock/tree/master/beddit_alarm/docs.

3.1 Activities

The program consists of the following seven android activities.

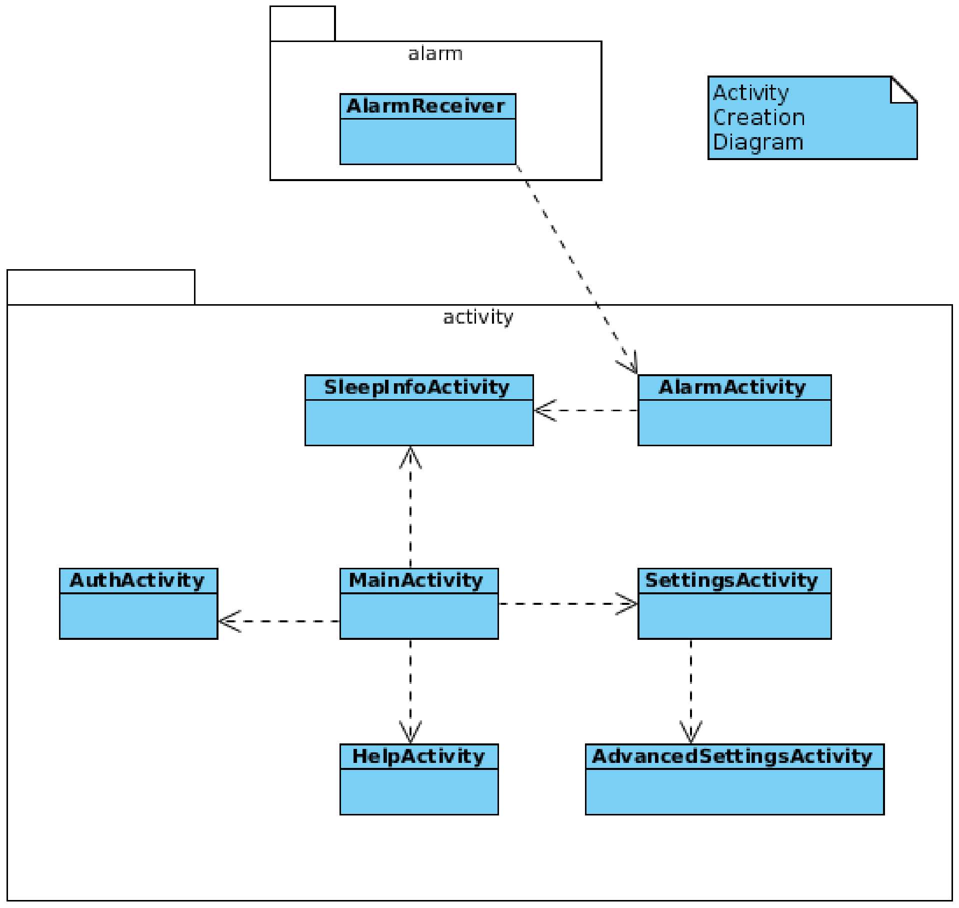


Image 1: Creation of Activities

MainActivity is the launcher activity of the application. If the user has not logged in, it starts AuthActivity for logging in and authorizing. The user can also open HelpActivity, SettingsActivity and SleepInfoActivity from here.

AuthActivity shows a webview for connecting to beddit account and handles the authorization.

HelpActivity shows the user some instructions on how to use the application. No interaction with the user.

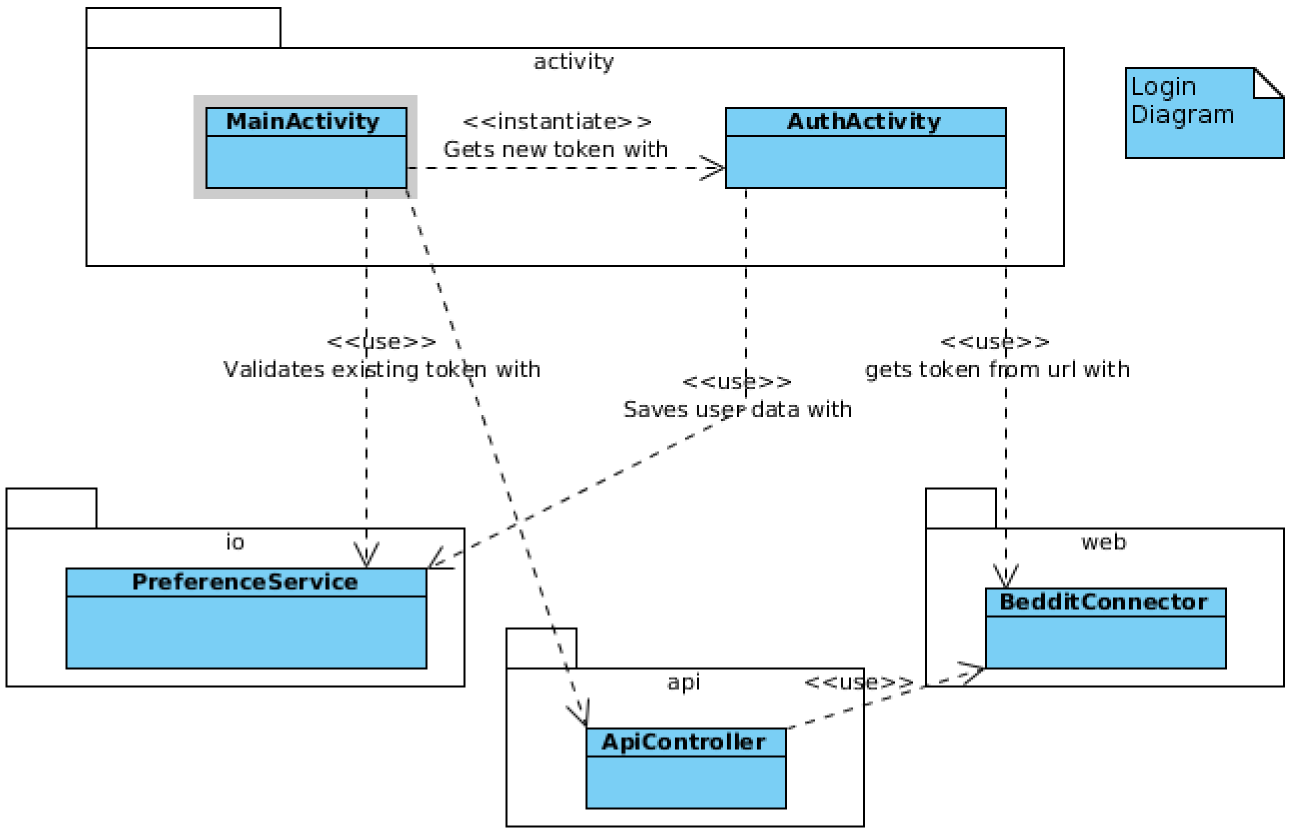
SettingsActivity opens a preferences view where user can disconnect from beddit, change settings or start the AdvancedSettings activity.

AdvancedSettingsActivity activity shows the user some less often used settings.

SleepInfoActivity shows the user info on how he or she slept last night.

AlarmActivity is launched by the AlarmReceiver. It plays an alarm tone, vibrates the phone and lets the user dismiss or snooze the alarm. When dismissed it shows SleepInfoActivity if the setting for it is enabled.

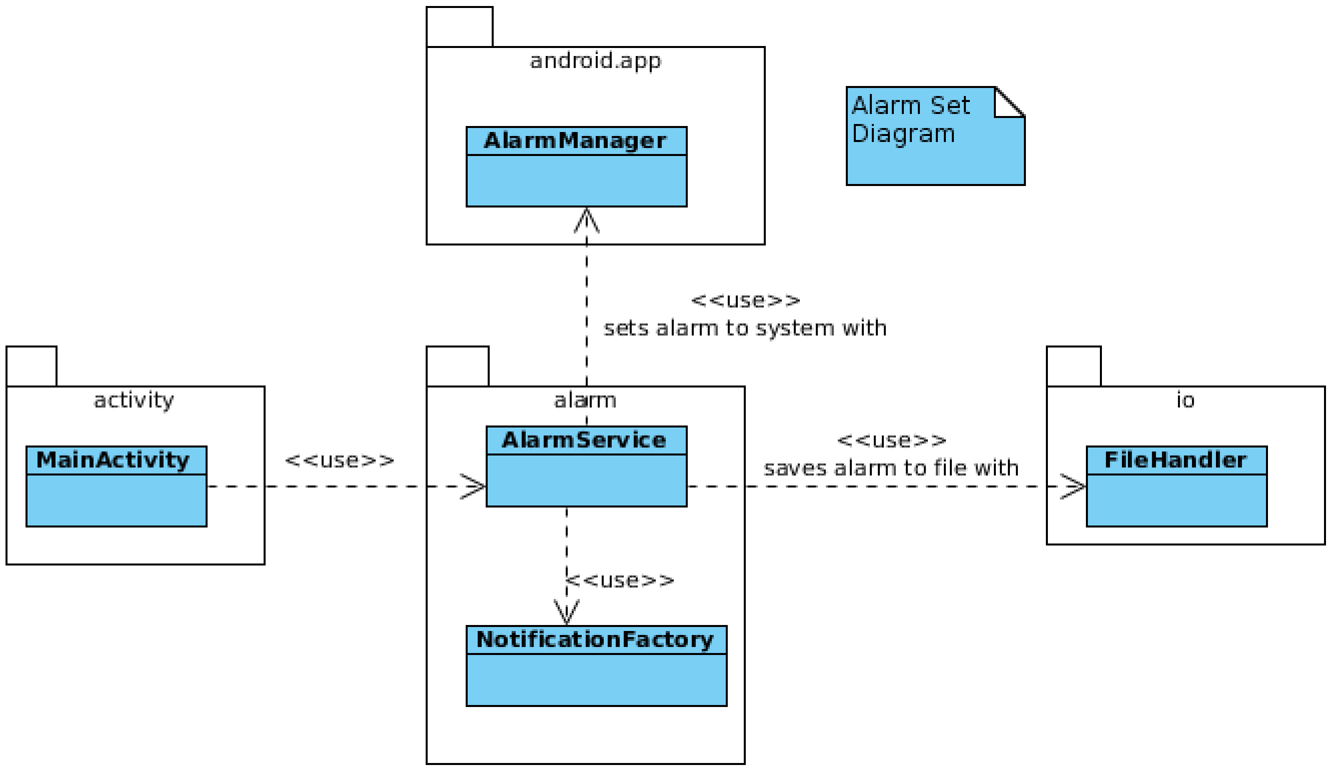
3.2 Connecting to beddit account



When user launches the program, the MainActivity is launched. In its onResume method, which is automatically called before showing the activity, it checks if there already exists a saved token. If not, it starts the AuthActivity for logging in and retrieving new token. Otherwise it lets user into the program and starts a TokenChecker which asynchronously checks if the token is still valid. If the TokenChecker finds out token is not valid, it starts the AuthActivity.

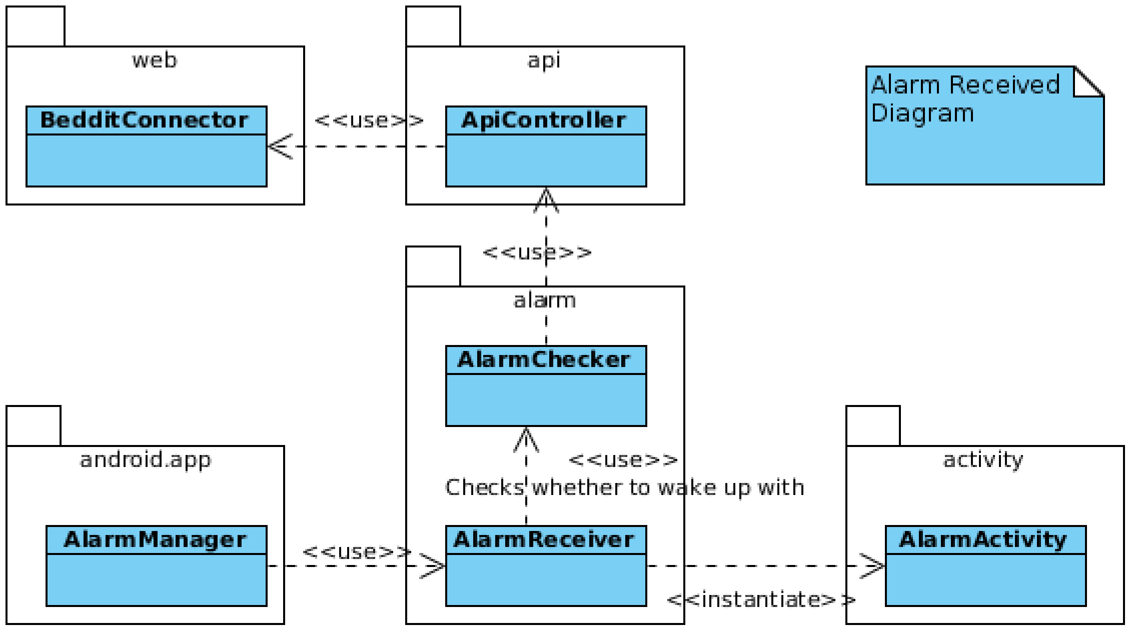
AuthActivity uses BedditConnector through ApiController (the above diagram is slightly outdated) to retrieve token from server. It then retrieves user data and saves both to preferences with PreferenceService.

3.3 Setting alarm



When the user sets a new alarm, AlarmService calculates the next wake up attempt time and sets it through Android’s AlarmManager. It also uses the NotificationFactory to set a notification and FileHandler to save the alarm to file.

3.4 Handling wake up



AlarmReceiver receives Intents broadcasted by the AlarmManager. It checks if the user should be woken up and if so, an AlarmActivity is launched. If not another Alarm is scheduled with the AlarmService a short time later. This repeats until a good time to wake the user is found or the wake up interval ends. The checking for wakeup is done by making API calls to the Beddit API via AlarmChecker using the ApiController and BedditConnector classes to form and execute the calls respectively. AlarmChecker compares the data from the API to make decide whether or not to launch an alarm. BedditConnector forms and executes the API queries requested by ApiController, which also saves and parses them using BedditJsonParser.

3.5 Quick summary of other important classes and packages

- *alarm*
  - Alarm: a class that contains information on single alarm.
  - AlarmTimePicker: an interface for using a time picker.
  - BootReceiver: handles setting the alarm back to manager on device boot.
  - NotificationFactory: handles setting notifications.
  - WakeUpLock: handles unlocking phone on alarm etc.
- *api*
  - *jsonparser*
    - BedditJsonParser: interface for parsing beddit json.
    - *classimpl*
      - BedditJsonParserImpl: implementation using Gson library
- *music*
  - MusicHandler: handles playing music.
  - ShowStopper: prevents application from playing music and vibrating forever. Runs as a separate thread that sleeps until it is time stops the music and

vibration.

- *utils:* general utility and helper classes
- *views:* package for custom Android GUI-components
- *web*
    - BedditLoginClient: handles the web side of Auth2 authorization process. Prevents user from wandering off during the authorization process.

3.6 Other important resources

res/raw/secret - a small json-format file that contains the client ID and client secret for the application. Client ID and client secret are not distributed through version control. The client ID and client secret are used when contacting Beddit server for the first time.
When creating the file locally, the file's contents should look like the following:

```
{
client_id:  "REPLACE_WITH_CLIENT_ID",
client_secret: "REPLACE_WITH_CLIENT_SECRET"
}
```

res/raw/alarm.m4a - for now, the application supports only a single alarm tone, which is distributed with the application. It is in the m4a format, but android supports many other audio formats. To change the default alarm tone for the application just replace this file

3.7 Localization

In Android localization is quite straightforward: Android searches for a localization of a string (xml element in a certain .xml file) from the localization folder for the language of the operating system. If there's no localization for it, the default string is taken instead. The default localization folder is **res/values** in which for this project the strings are in english.

To include additional localizations, create a new directory in **res/** with the name **values-xx,** where **xx** is replaced with the desired country code. For example **res/values-fi** for Finnish or **res/values-sv** for Swedish localizations. Copy the xml files with translated strings into the localization folder. Currently, only the **res/values/strings.xml** and **res/values/arrays.xml** files have locale-specific contents so it is enough to provide translations for these. If a translated string is not present in the localization folder for the operating system's preferred language, Android falls back to using the default string value for that string.

4. Dynamic Behavior: Scenarios

4.1 Use cases

Scenario: The time picker component minute hand is dragged.
Description: The user drags the minute hand on the touchscreen.
Component Interactions: The digital time display is updated with each tick of the hand. The position of the hour hand is also updated on the screen to reflect the selected time.

Scenario: The time picker component hour hand is dragged
Description: The user drags the hour hand on the touchscreen.
Component Interactions: The digital time display is updated with each tick of the hand.

Scenario: The time picker component slider is dragged
Description: The user drags the slider on the touchscreen.
Component Interactions: The clock face interval arc is updated as the slider is moved.

Scenario: The time picker component minute hand is clicked
Description: The user touches the touch screen somewhere within the range of motion of the minute hand’s grab point.
Component Interactions: A thread is started to move the minute hand to the targeted position.

Scenario: The time picker component hour hand is clicked
Description: The user touches the touch screen somewhere within the range of motion of the hour hand’s grab point.
Component Interactions: A thread is started to move the hour hand to the targeted position.

Scenario: The time picker component slider is clicked
Description: The user touches the touch screen somewhere within the bounding rectangle of the interval slider.
Component Interactions: A thread is started to move the slider to the targeted position.

Scenario: The alarm is set.
Description: The user presses the set alarm button
Component Interactions: The set alarm button is disabled and the cancel alarm button is enabled. A popup message is shown. An android notification bar element is created. The alarm manager is used to create an alarm in the android system.

Scenario: The alarm is cancelled.
Description: The user presses the cancel alarm button
Component Interactions: The set alarm button is enabled and the cancel alarm button is disabled. A popup message is shown. The android notification bar element is removed. The alarm manager is used to remove the android system alarm.

Scenario: The user updates the alarm with the time picker.
Description: The user manipulates the time picker component while an alarm is set.
Component Interactions: When the user action is finished, the main activity is notified of the new alarm time and interval and updates the alarm notification and alarm manager entry. A popup message is shown to indicate that the alarm was updated.

Scenario: The app is opened from the notification bar entry.
Description: An alarm is set and the user clicks on the alarm notification.
Component Interactions: The app is opened like in a normal startup situation.

Scenario: The user presses the android home button
Description: In every activity, the home button will cause the android home screen to be shown.
Component Interactions: The current activity is paused and put in the background.

Scenario: The user presses the android back button

Description: The current activity is closed and the previous activity is opened unless in a dialog.
Component Interactions: The android system destroys the current activity and resumes the activity on top of the activity stack.

Scenario: The user presses the android menu button
Description: If in the main activity, the menu bar will be displayed.
Component Interactions: A menu object is created and each menu button is given an intent that will be started on click.

Scenario: The user opens the settings screen.
Description: The user clicks the settings menu entry.
Component Interactions: A new activity is started to show the preferences.

Scenario: The user opens the help screen.
Description: The user clicks the help menu entry.
Component Interactions: A new activity is shown to display help text.

Scenario: The user opens the sleep info screen.
Description: The user clicks the sleep info menu entry.
Component Interactions: The sleep info activity is started with a setting that indicates the buttons should not be shown. A loading dialog is displayed while the sleep info is loaded from the beddit servers.

Scenario: The user disconnects from beddit.
Description: From the settings menu, the user clicks on the disconnect from beddit entry.
Component Interactions: The user settings and access token are purged from the preference service and the authentication activity is started. If an alarm was set, it is removed after giving a confirmation dialog to the user.

Scenario: The user enables or disables the sleep summary and beddit diary feature after an alarm.
Description: In the settings menu, the user clicks on the show summary of sleep data text to toggle the checkbox.
Component Interactions: The user's choice is saved to the preference service.

Scenario: The user sets the snooze time length.
Description: In the settings menu, the user clicks on the snooze time element and selects a time value from the dialog.
Component Interactions: A dialog box to select the a time from is displayed and the user's selection is stored into the preference service.

Scenario: The user sets the wake up sleep stage.
Description: In the settings menu, the user clicks on the sleep stage element and selects a time value from the dialog.
Component Interactions: A dialog box to select the a sleep stage from is displayed and the user's selection is stored into the preference service.

Scenario: The user wants to edit advanced settings.
Description: The user clicks on the advanced settings element in the settings menu.
Component Interactions: A new activity is started to display the advanced preferences.

Scenario: The user wants to change the application color theme.
Description: In the advanced settings activity, the user selects alarm color theme and chooses the dark or light theme from the shown dialog.
Component Interactions: A dialog box to select the a color theme from is displayed and the user's selection is stored into the preference service.

Scenario: The user wants to change the alarm ring time.
Description: In the advanced settings activity, the user selects alarm length and makes a choice from the dialog.
Component Interactions: A dialog box to select the a sleep stage from is displayed and the user's selection is stored into the preference service.

Scenario: The user signs into the beddit service with correct login information
Description: When the authentication activity is started, a loading dialog is shown while a customized web browser loads the login screen. The user enters their beddit credentials into the login form and submits it. A loading dialog is again shown while the beddit website processes the login.
Component Interactions: None, this happens internally within the web browser.

Scenario: The user signs into the beddit service with incorrect login information
Description: The user enters an invalid password or username into the login form. This is handled by the beddit website.
Component Interactions: None, this happens internally within the web browser.

Scenario: The user allows the beddit alarm application to access his/her sleep information
Description: After successful logging in to the beddit website, a page is shown where the user can allow the beddit alarm application to access sleep data. On pressing yes, the user will be taken to the main activity.
Component Interactions: At the end of this process an access token is acquired from the beddit api and stored into the preference service so the application can be used without logging in every time.

Scenario: The user does not allow the beddit alarm application to access his/her sleep information
Description: After successful logging in to the beddit website, a page is shown where the user can allow the beddit alarm application to access sleep data. On pressing no, the user will be taken to the main activity and a dialog will be shown explaining that the application cannot be used without logging in to the beddit service. On closing the dialog, the application exits.
Component Interactions: An android dialog is shown, the main activity is closed.

Scenario: The alarm is triggered
Description: The alarm rings, the phone starts vibrating, and the screen is unlocked. A dialog is showed where the user can
Component Interactions: The alarm activity is started. Android system functions are used to trigger vibration and alarm tone.

Scenario: The user snoozes the alarm
Description: The user presses the snooze button on the alarm activity.
Component Interactions: The alarm activity is closed and the alarm manager is used to create a new alarm.

Scenario: The user dismisses the alarm
Description: The user presses the dismiss button on the alarm activity.
Component Interactions: The alarm activity is closed. If the sleep info setting is on in the preference service, the sleep info activity is started.

Scenario: The sleep info screen is showed after an alarm.
Description: The sleep info activity is started. A loading screen appears while the nights sleep info is retrieved from the beddit servers. On the sleep info screen the user can indicate how he/she feels after waking up. This opens the browser to the beddit diary where the user can make more detailed observations.
Component Interactions: The default browser is started

4.2 Background behaviors

Scenario: First time startup
Description: When the application is first started it should direct the user to a login screen and afterwards show a default time on the time picker component.
Component Interactions: The main application activity asks for the alarm time from the file handler component and gets a default time when none is found. The main application activity then checks the existence of an access token and when it finds none, starts the user authentication activity.

Scenario: Normal startup
Description: On normal startup, the current or last alarm time is shown on the time picker and the access token is verified.
Component Interactions: The main application activity gets the last alarm time from the file manager and sets the time picker to that time. The main application starts an asynchronous task to validate the access token. When the task finishes, it will show a dialog if the access token is invalid allowing the user to exit or re-authenticate.

Scenario: The correct color theme is applied to the main activity.
Description: The shown color theme should be correct at all times.
Component Interactions: When the main activity resumes, it reads the current color theme from the preference service and applies the correct background and time picker colors.

Scenario: If the alarm is set, the application will start checking for a proper wake up time at the beginning of the set interval.
Description: When the alarm is set, Android’s alarm manager is set start the alarm receiver at the proper time.
Component Interactions: Android’s alarm manager is called with an intent that instructs it to wake the alarm receiver process.

Scenario: If the current time is within the wakeup interval and the user is in the proper sleep stage, an alarm will be initiated.
Description: After the alarm receiver is called, it uses the alarm checker to determine if the user is in the correct sleep stage. If the sleep stage is correct and the data is up to date an alarm is initiated. Otherwise the alarm requests the server to update the information and sets the alarm receiver to be called in a short time.
Component Interactions: The alarm checker makes api calls through the api controller object. Android’s alarm manager is used to schedule the next check.

Scenario: If no alarm has been started before the end of the interval time, an alarm is initiated.
Description: The alarm receiver checks if there is enough time to get more up to date results from the server before the final alarm time. If not, it will start the alarm activity.
Component Interactions: The alarm activity is stated.

Scenario: The user interacts with the time picker component.
Description: The user manipulates one of the time picker's parts in some way.
Component Interactions: When the action is finished, the time picker reports the new time to all of its listeners.

5. Deployment

- The application can be compiled into a .apk Android package file with an IDE that supports the Android SDK such as IntelliJ Idea or Eclipse.
- The application can be installed to an Android phone running version 2.2 (Froyo) or later.
- To install the application, enable installation of non-Market applications under Android application settings then open the .apk file.
- Logging in to a Beddit account is required to use the sleep cycle alarm clock application or any of its features.
- If the application is to be distributed via Android market, the customer will be responsible for digitally signing the application and Android market deployment.

6. See Also

For further information, see the JavaDocs, the testing documentation report, and the license file. These can all be found in the git repository at https://github.com/bedditor/ohtu-Alarmclock.