# Nighty Bird

# Your Stay-up-late Assistant

Yuefeng Zhou
Xuping Lei

# Chapter 1

# Overview

NightyBird is an application for people to adjust their sleeping habits. It will remind user to go to bed when the user is staying up. It generates reports to give user advices on sleeping, sports and diets.

Users can check in when he is going to sleep or when he wakes up. The application will record that time for the users. Users can also input or edit his sleeping data manually. NightyBird provides users several interfaces to manipulate his sleep data.

When the user is staying up very late, NightyBird will send notifications to the user to urge him to sleep. They "stay up late" is defined by the user. The user can also set up the period of notifications when staying up.

Users can view a daily report or a weekly report. The weekly report contains a range chart that makes it look more intuitive. In addition to the range chart, some suggestions are also included like sports advices or diet advices. These suggestions are fetched by a web service. The NightyBird application send user's sleep data to our server, the server responses with these suggestions.

class diagram for different packages

# Chapter 2

# User Interfaces

We have four main activities: main page, report, history and settings and a navigation drawer fragment that user can use to navigate between the four activities. A preference fragment is attached to setting activity and a history editor activity is triggered by the history activity.

Chapter 3

# Main Page and Reminder

# Chapter 4

# Report

On the report page, the NightyBird application presents user a weekly report as long as the user has data of at least 7 days. The report contains a range chart and a report paragraph.

## 4.1 Weekly Range Chart

The range chart includes 7 sleeping range of the past week. We use an external library called AChartEngine to draw the chart for us. The chart will be drawn on creation or resume of the report activity.

a figure

## 4.2 Report Web Service

There's a text view below the range chart. We use this text view to present our advices for this user. The text of this text view is dynamically loaded by our report web service. When the report activity is created or resumed every time, the report service client called ReportServiceClient will be invoked to fetch a paragraph of advices for this user.

## 4.3 The Client

The client is implemented by ReportServiceClient class which extends android.os.AsyncTask. Therefore, the client will executes its task in the background task other than the activity thread.

To use this client, we pass an TextView object into it and starting executing asynchronously. In it task, it send a http request to the the server and server will response with a JSON string. The http request will contain the user's sleep data, either daily data or weekly data. The JSON string will parsed into a JSON object, a field called "report" will be fetched and the preset text view will be set the text to be the content of the report.

The server ip can be configured by users in the setting activity.

In case of any failure like network failure, the text view will display the problem and there will be a dialogue telling user about it.

## 4.4 The Server

# Chapter 5

# Database and Sleep Data History

## 5.1 Schema

We a content provide to maintain user's sleep data. We have structured sleep data class called SleepData that contains a start time, an end time and an id as well. This id will be filled by the inner database and it is the primary key of the sleep data table.

We define our database table as following:

```
CREATE TABLE SleepDataTable
(SDID INTEGER PRIMARY KEY AUTOINCREMENT,
 StartTime LONG NOT NULL,
 EndTime LONG NOT NULL)
```

The id is automatically created by the database and we fill the start time and end time by unix time. The reason we use unix time is because its a uniform time and we will not have any ambiguity across platforms and people. It is also convenient for us to compare them.

In addition to that, we also have a singleton class called SleepManager which exposes interfaces to insertion, update or deletion of the database. Besides, some utilities like converting Date class to date string are also provided.

## 5.2 History List

In the history activity, we use a list view to present user's sleep data. It also provides interfaces for users to add, edit or delete them.

The list view uses an adapter to get the sleep data item from database, i.e. our SleepDataManager.

Users can click Add New button to add an item to the list or he can long-click any item of the list, then a dialogue will be pop up to prompt users to chose edit, delete or cancel.

When users chose to add or edit an item, a history editor activity will be started where users call edit his sleep data. After the operation is done, users can click OK button that will return to history activity with a result containing the sleep data. The history activity will invoke the SleepDataManager to update the sleep data table accordingly.

The history editor contains one date picker and two time picker. It prompts users to input his sleep time and wake up time for some date. When data is loaded to our SleepDataManger, if its sleep time is after its wake up time, the SleepDataManage will adjust its date to make it valid.

The update by the history editor will be display immediately after users edit it. Users can click back button to discard any update.

# Chapter 6

# User Preference

The last not least activity is the setting activity. When it is created, it presents the UserPreferenceManager which is a fragment extending android.preference.PreferenceFragment. It simply places content according to the preference.xml. So we can configure the preference entry in the preference.xml.

Our provided user preferences are: user's name, gender, user's stay-up-late threshold, period for reminder and server ip for report service as well.

We have a singleton PreferenceManager which provides interfaces to other activities to read and update user's preference programatically.

The PreferenceManager invokes android.content.SharedPreferences to read and edit users' settings.

# Chapter 7

# Other Features

## 7.1 Multiple Resolution Support

## 7.2 Start View

## 7.3 Debugger

We have a debugger class that makes toast anywhere we want. We use this to debug our application.