

**A REPORT ON**

**“ANDROID APPLICATION DEVELOPMENT AND TESTING”**

BY

SRUJAN JHA 2012B1A7744H

Prepared in partial fulfilment of the

“Practice School-I”

Course No. BITS C221/BITS C231/BITS C241

AT

**“FACT VENTURES PVT LTD”**

A Practice School‑I station of

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**JUNE-JULY, 2014**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE**

**PILANI (RAJASTHAN)**

**Practice School Division**

**Station**: FACT VENTURES PVT LTD **Centre:** MUMBAI

**Duration: From:**  23-05-2014 **To:** 17-07-2014

**Date of Submission:** 16-07-2014

**Title of the Project**:

**“ ANDROID APPLICATION DEVELOPMENT AND TESTING ”**

|  |  |  |
| --- | --- | --- |
| **ID No.** | **Name of student** | **Discipline** |
| 2012B1A7744H | SRUJAN JHA | M.Sc. (Hons.) Biological Science & B.E. (Hons.) Computer Science |

**Name of expert Designation**

Vishal Gondal Chief Executive Officer (CEO,GOQii)

Sachin Janghel Chief Technical Officer (CTO,GOQii)

Nikhil Lotke Android Developer (GOQii)

Paresh Patil Application Tester (GOQii)

**Name of the PS-1 Faculty:** Mr Swagat Kishore Mishra

**Key Words:** Android App Development, Graphs, Social Integration

**Project Areas:**

Android SDK, Eclipse, Highcharts API, Facebook SDK

**ABSTRACT:**

Our project is based on Android App Development, where we developed few sample projects to improve the user-app interaction in the present GOQii app.

**Problem Statement:**

1. To enhance the user-friendliness of the app.
2. To implement new features in the app.

**Works Accomplished:**

1. A sample application which focused on enhancing the Logging experience of the user in the present GOQii app.
2. Enhanced the User-Interface using many small widgets:
   1. Range-Seek-Bar for entering Start and End time of an activity.
   2. Deletion of ToDo activities.
   3. Number picker for entering Frequency.
   4. GIF renderer
3. Implemented Alarm & Notifications for reminder and logging purposes.
4. Implemented an optimized database using SQLite.
5. Implemented and enhanced the present graphs with better animated graphs using HighCharts API.
6. Implemented a navigation drawer for the app which works by gesture detection for better display of important menu items.
7. Integrated the GOQii app with social media to some extent.

**Results and Conclusion:**

We were able to make some of the basic and advanced UI elements that could help the organisation to improve their app features so as to enhance the user-experience.

So overall, we learnt a lot of things and successfully implemented them.

|  |  |  |
| --- | --- | --- |
| Signature(s) of Student(s) |  | Signature of PS faculty: |
| Date: |  | Date: |

# Contents

# About GOQii

# GOQii Ecosystems

## Band

## Coach

## GOQii App

## Karma points

# Project In Detail

# Log Activity

## Main Page

## Logging Page

## ToDo Page

## Alarm & Notifications

## Range Seek bar

## SQLite

# Graphs

## Problem

## Solution

# Sliding Menu

## Horizontal scroll view

## Custom List

## Simple Gesture Filter

## Horizontal Scroll View With List Menu

# Social Integration

## Facebook

# ABOUt GOQii

GOQii is headquartered in California with offices in Mumbai and Shenzhen. It is dedicated to enabling a permanent shift to a healthier lifestyle with the combination of advanced wearable technology, the world’s leading experts, coaches and karma. GOQii focuses on sustained engagement & goal reinforcement and offers a complete ecosystem as a single integrated solution, which is a powerful combination of technology and human connection. GOQii is a collaboration of some of the world’s leading experts and an amazing management team who are driven by a passion for healthy sustainable living, and who believe that all of us contain boundless potential. GOQii’s goal is to enable millions of people across the world to “be the force” by helping them unleash their untapped potential.

GOQii is backed by leading angel investors: Amit Singhal (Google), Anil Godhwani (Habitera), Bharat Vasan (MyBasis), Dr. Shriram Nene, Esther Dyson, Goutam Godhwani (Simply Hired), Kanwaljit Bombra (Dell), Madhuri Dixit-Nene (Bollywood Actress), Mahesh Samat (Epic), Mike McNamara (Flextronics), Prashant Gulati(TiE), Sanjay Parthasarthy (Indix), Sanjay Vaswani, Steve Luczo (Segate), Taher Khorakiwala and Vijay Vashee (Microsoft).

# GOQii Ecosystems

GOQii provides a service, which is a combination these five components:

**Band:** Measures the activities and sleep of the user.

**App:** Tracks nutrition, lifestyle and karma of the user.

**Coach:** A personal guide to healthier living and goal reinforcement of the user.

**Experts:** Passionate minds who design the GOQii Life.

**Karma:** Live healthy, earn karma points, and donate.

## The Band

### Features

#### BLUETOOTH LOW ENERGY

The band sync with the Web/PC application or the mobile version of the Application using the technology of Bluetooth.

#### VIBRA HUMM

Custom designed humm vibration to wake up the user at their desired time and to remind them to be active. A sort of “*silent*” alarm, which is also a great feature of the band.

#### SWEAT PROOF AND WATER RESISTANT

The band is water and sweat resistant, however, the organisation feels it is not suitable for swimming.

#### 360° MOTION SENSOR

The band measures all user’s activity and sleep quality

#### POWER PACK

The band is believed to work for 4-5 days on a single charge

#### TOUCH SCREEN OLED DISPLAY

Clear, responsive & intuitive display

#### Available in three colors:

Red-Black Black Red

****

### Functionalities

**Steps:** Measures the steps, while walking or jogging.

**Distance:** View distance travelled while active, indoor or outdoor.

**Karma points:** Through the steps, earn Karma points to donate. The user can check his Karma points to stay motivated.

**Time:** A display of time along with date.

**Calories:** The user can view how many calories he burnt during the day.

**Active Time:** Measures how long the user stays active.

**Sleep:** Monitors the sleeping pattern, see how long the user rest.

**Remote Hi5:** The coach send a remote Hi5, when the user do exceedingly well.

## Coach

Health, fitness and habit experts help the coaches to provide the user sustained engagement and goal reinforcement.

### Habits

The coach helps the user identify what factors cue one’s habits. Working with experts, the coach suggests scientifically proven techniques to help the user shift their habits for the better.

### Goals

The coach helps the user set smarter targets. They partner with the user and give them the tools and motivation to make the small changes in the user’s life that lead to a big impact.

### Connect

Use the GOQii App to interact with the coach. Whether the user is looking for information or a patient ear, the coach is always by the user’s side.

### Data

Quantified self meets online coaching. The coach analyses all the information the user collect with the user’s device and app. They help the user find the patterns and suggest strategies to help them achieve the user’s goals every day.

### Hi5

The coach send a remote Hi5, when the user do exceedingly well.

## GOQii app

The app transforms the user’s raw collected data into meaningful analytics for the coach and the user.

## KARMA

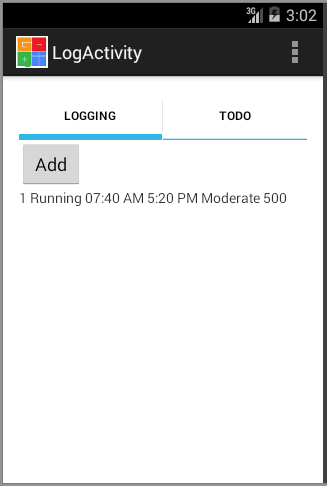
There is great happiness in doing good. As a GOQii Player the user gets a great new way of spreading smiles. The user meet his daily targets and earn Karma Points. The user can then donate his Karma Points to the favourite causes. The GOQii Donor Partners (OxFam) would donate money to karma partners the user choose to donate his/her karma points to. The user gains one Karma Point on completing 390 steps.

# PROJECT: Log Activity

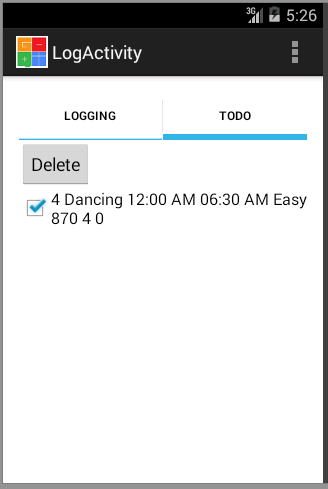
GOQii app has a functionality to log one’s activities like running, walking, dancing, skipping, one’s food, one’s sleep, and one’s water intake. We worked on one of these functionality which was logging activities. We were to add one more functionality which is “Adding to ToDo”. This would make things work faster and enhance the usability of logging functionality. Normally, user doesn’t want to log each and every repeating tasks. So, we shifted the responsibility of the user to the app. Basically, the user can add an item to his ToDo list, and he need not complete the logging action. The app will send a notification, waiting for the user to just say “Yes” or “No”, and will log the repeating tasks to the logging database accordingly. We divided this project into 3 parts: Main Screen, Logging and ToDo.

## MainScreen

This page of the app was just for the development part and will not be integrated into the main app. As we weren’t provided the old version of the main GOQii app, we had to make our own gateways to Logging and Todo pages. The page consisted of two tabs, one listing the contents of the Logging Database and the other listing the contents of the ToDo Database.

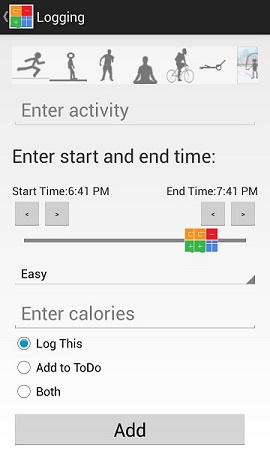
Logging Tab:

It contains a list view showing the contents of the database of Logging.

ToDo Tab:

On the TO DO tab we have a delete button which when clicked the list view gets modified with the each row having a checkbox. If we select the checkbox and click the delete button again the activity is deleted from the To Do list and the database.

## Logging:



* Horizontal Scroll view with Webviews to render GIFs of the activities.
* An editText box which is editable only when “Other” in activity box is selected.
* Range Seek Bar and its contents.
* Spinner with Intensity views
* Calories Text Box
* Radio Group

### Activities listed at the top with their respectiveImage:-

In these activities listed at the top we used images which are in GIF format. Android doesn’t have inbuilt widgets to render the GIF images. So, we had to use the webview because normal images are not animated. So on using webviews image animation doesn’t get disturbed .For each webview we setup on touch listener which on touching displays the name of that specific activity.

### Start time and end time of the respective activity:-

After clicking that specific activity we can set the start time and end time of the activity. For this we set up the slider in such a way that we can set the slider at the desired positions so that it displays the start time and end time of the activity.

As the slider is touch sensitive we set up four buttons two for each start-time and end- time. Out of those the two buttons at the start time one is for decreasing the time by 10min and the other is for increasing the time by 10min.Similarly the two buttons near the end button performs the same function.

### Intensity of the activity:-

Intensity of a particular activity is divided into three types.

They are: - 1.Easy 2.Moderate 3.Intense

We can select only one among these three. In order to implement this we implemented the spinner. The function of a spinner is, it allows us to select only among all the options provided.

### Number of calories consumed:-

After entering the data of a particular activity calories consumed will be displayed.

### Radio group buttons:-

There are three radio buttons in a radio group. We used the radio group because it allows us to select one among those all available buttons. The names of these three buttons are:-

1. Log this

2. Add to ToDo list

3. Both

On clicking “Log this” the activity adds to logging list.

On clicking “Add to ToDo” list the activity adds to ToDo list.

On clicking “Both” the activity adds to both logging list and ToDo list.

### Add button

On clicking the add button, navigation takes place in three different ways.

1. Return to Main Page after adding the contents of the page to Logging Database.
2. Go to ToDo Activity page.
3. Go to ToDo Activity page after adding the contents of the page to Logging Database.

## TO DO ACTIVITY

The GOQii app that we were making consisted of two important activities, the **LOG activity** and the **TO DO activity** namely. Both of which were integrated together with all sort of user friendly options i.e. the buttons and checkboxes etc.

**LOG Activity** consists of the information about any activity that the user just performed or did it in past and he/she wants to log that activity in the app so that he may receive proper guidance and tips from his/her personal coach.

Log Activity includes:

* The Activity Name (Along with animations).
* The start and end times of the activity being logged.
* Intensity of the Activity.
* Calories Burnt.
* Checkboxes to add the activity to Log, To Do or both.

So if the user just checks the Log Activity, the data is stored within a database which we retrieved on the home page under the Log Task Tab using SQL adapters and fetching it the Query.

But if the To Do or Both is selected and the button is clicked the **ToDoActivity.java** is opened which takes from user:

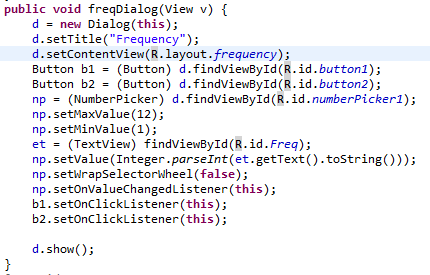
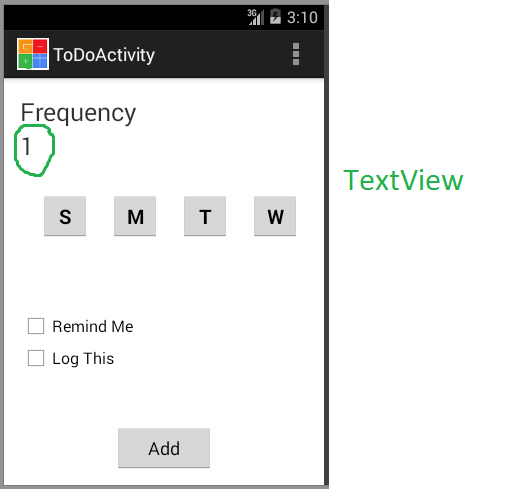
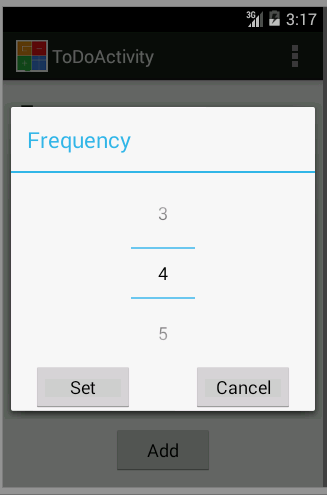
* Frequency of the activity added to ToDo.
* The days of the week on which the user wants to perform the activity.
* The Remind Checkbox
* The Log This Checkbox
* The Add Button.

Firstly we **get intent** and flags from the other classes and check whether the task number i.e. the primary key in the database table is not already existing. If it does then the button shows update instead of add and we can change the frequency and days of the activity and update the database. But if the task number is not already existing then the button is for adding the activity to ToDo database.

### FREQUENCY

In the frequency part we have set up a **number picker** in a layout, which pops up on clicking the TextView which is made clickable.

The TextView shows the frequency as 1 by default. On clicking it, a **Dialog box** pops out showing a number picker to choose the frequency and two buttons to Set and Cancel the number set as frequency in the TextView as shown below.



On clicking the Frequency Picker.

### SETTING DAYS

This is the portion that took much of our time to think of optimal ways to store the days on which the alarm needs to be set.

The obvious solution was that we store 7 different Boolean variables for seven buttons each and give them the values according to the button being clicked.

That is, giving them value 1 initially and toggling between 0 and 1 on each further click. Then we stored the values of each variable separately in separate columns in the database. But if we see the length of the code that might have been used to make this simple thing to happen, then it would seem a worthless job to work on.

So what we did is that we made each button given a value of 1 initially and on each further click the value toggles between 0 and 1. And we converted the Sequence of Sunday to Saturday Button into its Binary equivalent.

e.g. If below is the button pattern in the app with bold as Selected else otherwise then its binary equivalent is calculated as:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S | **M** | T | **W** | **T** | F | **S** |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |

Binary Equivalent =

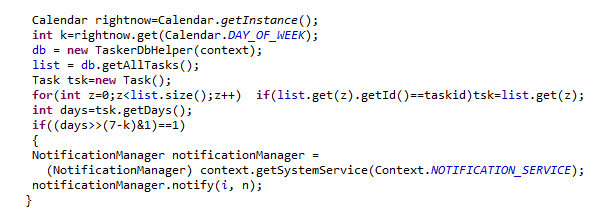
0\*2^6 + 1\*2^5 + 0\*2^4 + 1\*2^3 + 1\*2^2 + 0\*2^1 + 1\*2^0

Which is equal to 32 + 8 + 4 + 1 = 45.

Now the storing problem is solved, but how do we get the alarm set up.

We retrieve the value stored i.e. 45 and convert it back to its binary equivalent.

Now in our Alarm setting class we have given the condition that if the present day number say for Wednesday (it is 4) then we right shift the binary number 7-4 i.e. 3 times and we operate the result and 1 with AND operator. If that is 1 then alarm is to be set for today (Wednesday). Here’s the code snippet for the same.



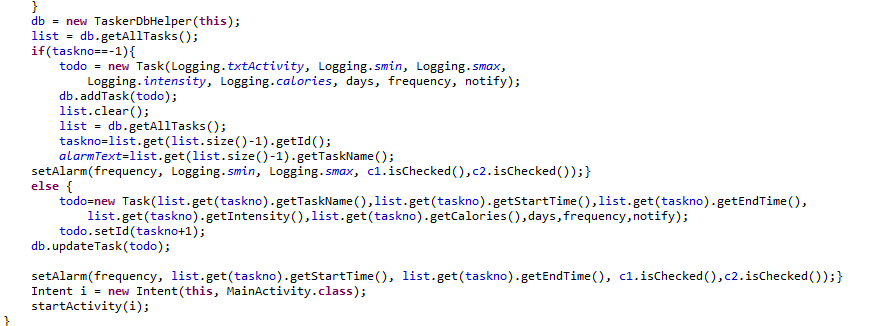
### REMIND ME CHECKBOX

This is the checkbox that would set an alarm (notification) at the end time of the activity set up by calling the Alarm function. This would help user to finally enter his routine activities in the app according to his/her convenience without forgetting.

Here the alarm function i.e. setAlarm has parameters as:

* Frequency
* Minimum time
* Maximum time
* Checbox1.isChecked()
* Checbox2.isChecked()

Here’s the code snippet for the alarm integration of the activity.



### LOG THIS CHECKBOX

This checkbox is for adding the activity in the Log at the same time as entering in the To Do list because sometimes user might have done the activity the same time as he/she is entering it in To Do. So to avoid repetition the checkbox just does the same thing as what happens on adding to To Do. The only difference is that the To Do database also stores the frequency and integer value of notify acc. To the checkboxes checked apart from the basic details of the activity logged.

C:\Users\intel\Pictures\Screenshots\Database to do.png

Above is the screenshot of the entry of the database which shows that:

* The activity is Dancing whose id is 4.
* The activity is to be done from 12:00 AM to 6:30 AM.
* The activity is of Easy intensity.
* The activity burns 870 calories.
* The activity needs to be repeated 4 times the day.
* 0 is the way of saving the days on which the activity needs to be done (As was explained in previous section).

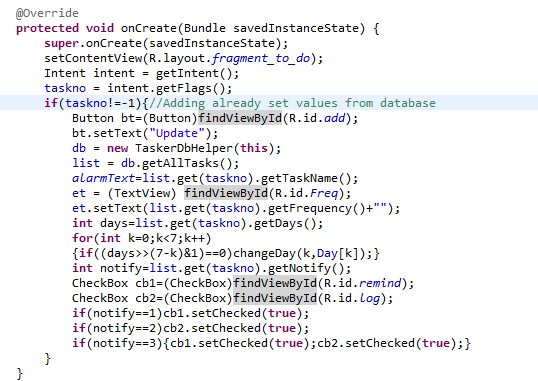
### ADD BUTTON

On pressing the add button the same thing occurs as what took place in Log Activity. We execute a SQL query to store the data entered in a database table.

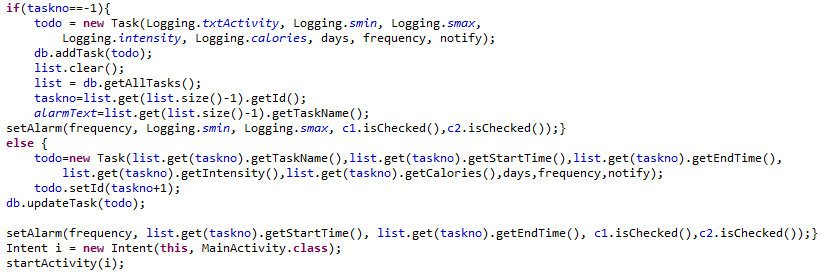
After the Add Button is clicked the intent is sent to MainActivity.java where in To Do tab we can see that our activity has been added.

### UPDATE

In the listView, on clicking any list row we can edit the frequency and the integer value of notify for the activity. This is taken care of by the code:



And the code



## Alarm and Notification

For the to-do activity of the application had to be generated notifications before the start time of the activities in the to-do list as reminders and also at the end time to log the activity. We made a function *setAlarm ()* for its which has been explained below.

### setAlarm ()

For this function we took start time, end time, days, frequency and the types of notification (remind, logthis) as input. We used the start time set by the user and generated alarms at the required times at equal interval depending on the value of the frequency. We ensured that all the alarms were between 6 am to 10 pm. We set the first alarm at start time if it was between 6 am and 10 pm otherwise we set the first alarm at 6 am of that day. Then we generated the remaining alarms using the value of frequency. We divide the 16 hours between 6 am and 10 pm into equal intervals by dividing it by frequency.

Using the value of the interval we added it to start time to generate new alarms. If the alarm crossed 10 pm we took the difference between 10 pm and the alarm and added it to 6 am of the next day and set the alarm for that time. This was done to ensure that all alarms were in 6am to 10pm range. We now used the value of the two variables remind and logthis to decide if the notification was to be generated before the start time or after the end time or both times. For the case of remind we had already generated the times of the alarms using the start time as written above while for logthis we generated the alarms by adding the value of duration of the activity to the times of the remind alarms. We also set the alarms repeating every 24 hrs.

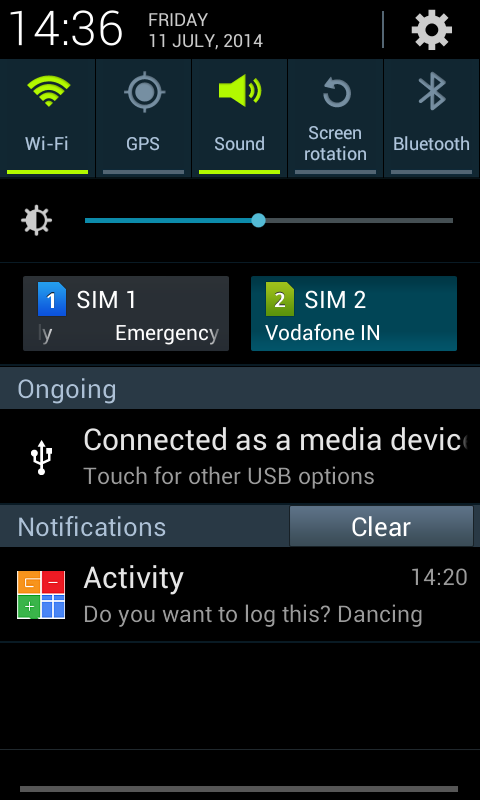
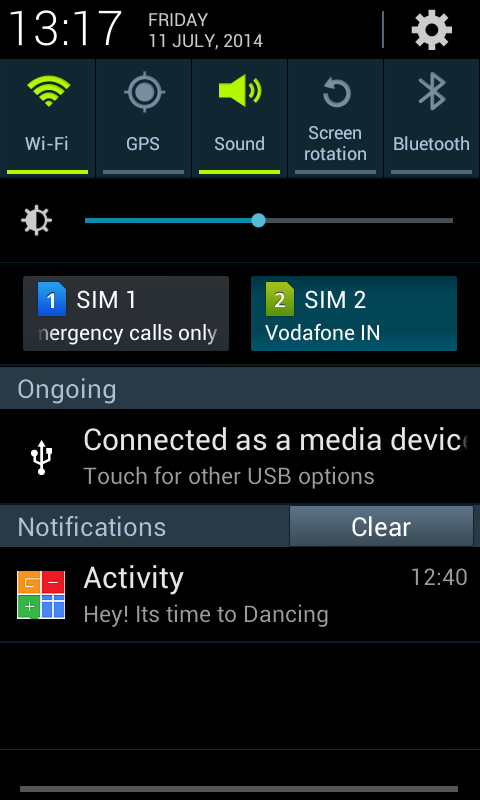
Each alarm was given an id .This id was decided using the id of the activity in the table for which the alarms was being set. We had also set the text for the notification depending on whether it was remind or logthis type. We sent the intent to the alarm receiver class in both cases to generate notifications when the alarm rings

### AlarmReciever

This class is called whenever an alarm rings. In this class we set whatever we want to do at the time of the alarm. We generated the notification in this class.

We used the alarm id to get back the id of the activity for which this alarm was set. We got the name of the activity and used the text we got from *setAlarm ()* function to generate the content of the notification. Now to ensure that the notification was only displayed on the selected days we checked if the current day was selected by using the days variable. If it was selected the notification was generated otherwise nothing would happen.

Thus this is how we generated the alarm and the notification for the application.



Left: The screenshot showing the Remind Me Notification, which appears just before the Start Time of the Activity.

Right: The screenshot showing the Log Notification, which appears just after the End Time of the Activity. On clicking the notification, the user will be navigated to the Logging Page of the app, with all the information filled in using the database. The user thus saves his time by just clicking on “Add” to log an activity.

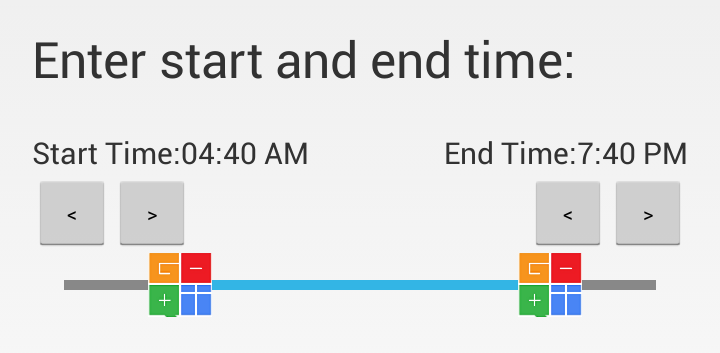
## Range Seek Bar

The [Android widget class library](http://developer.android.com/reference/android/widget/package-summary.html) contains only one "slider" like widget: the [SeekBar](http://developer.android.com/reference/android/widget/SeekBar.html). SeekBar only supports one thumb control and thus handles only one selectable value. If we need a widget allowing to handle two values (like a minimum and maximum value or a range), this RangeSeekBar widget might suit our needs. It supports controlling minimum and maximum values on a numerical range of the following types: Long, Double, Integer, Float, Short, Byte and BigDecimal. We modified an ongoing project of Range-Seek-Bar.

### Additions:

On tapping the seek-bar, the nearest thumb will come to the tapped position. We also added four buttons for updating the thumbs positions. This would increase the accuracy of the range-seek-bar.  
Two text views were added for representing the Start and End time, which updates correspondingly with the thumbs of the range-seek-bar.

A point on the range seek bar represents a value in “Long” value which when multiplied by 1000\*5 gives the value in current date. By default the start time is the current time and end time is an hour in addition to the current time.



## Sqlite

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication. SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Concerning to our app we used two SQLite database classes. One for logging list and the other todolist.

### Important contents in a SQLite database

For each and every database the following are necessary

1. DATABASE\_NAME

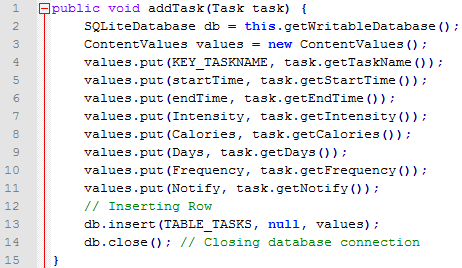
2. DATABASE\_TABLE

3. DATABASE\_VERSION

### Storing information to database

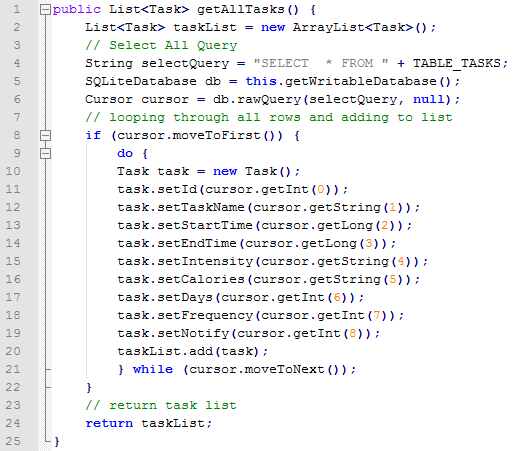
When we enter click on the add button both in logging and to do list, the data gets stored both into the SQLite database. For doing this we have written a function named addtask.

**Code:**



### Displaying the stored information

In order to display the stored information we created a list view in which it displays all the activity information stored in the database when as soon as we open the application.



# PROJECT: Graphs

GOQii app uses graphs to display data related to steps count, active and deep sleep hours, water intake and calories burnt. These raw data is again divided and presented as a function of time of a particular day.

## Problem:

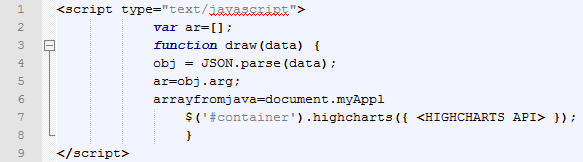


The problem with the existing graph was unreadability. The user couldn’t understand what the graphs suggested. There was no animation applied to the graph and it had no functionality defined when the user taps the graph.

## Solution:

AndroidPlot is an API for creating dynamic and static charts within the Android application. However, it did not provide animation to the graphs. Hence, we used APIs provided by Highcharts. Highcharts provides APIs for Web projects and it comes at a price. Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, area range, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types.

So, a method of web scraping to get the JavaScript files that the APIs use to construct the animated graphs was finally used. The close look on the HTML source files of the website [www.highcharts.com](http://www.highcharts.com) revealed that the graph was constructed in real time using the files like highcharts.js, jquery.js, and data.js. We wrote a HTML code which would make use of the same JavaScript files to construct graphs with our data.



The script was a JavaScript function which gets data from android while runtime and assigns the data accordingly to the variables used in <HIGHCHARTS API>. The “#container” is the html tag for a virtual container where the graph drawn will be placed. We then use a WebView in Android Layout to display the graph generated as html. There was a problem that android couldn’t change the data in the Html files and it can only contact with the JavaScript file using a string variable. Hence all the data which comprises of the graph data, the type of the graph, whether it is 3D or not and the width of the graph needed to be sent as a string data only.  
Thus, JSON was used.

### JSON or JavaScript Object Notation:

It is an [open standard](http://en.wikipedia.org/wiki/Open_standard) format that uses [human-readable](http://en.wikipedia.org/wiki/Human-readable) text to transmit data objects consisting of [attribute–value pairs](http://en.wikipedia.org/wiki/Attribute%E2%80%93value_pair). It is used primarily to transmit data between a server and web application, as an alternative to [XML](http://en.wikipedia.org/wiki/XML).

{

"args": [10,20,30,40,50,60,70,80,90,100,110,120],

"width": 400.0,

"chart": "column",

"thd": **true**

}

This is a format of JSON which was used in the project. Using the previous JavaScript snippet, we could get the graph data as an array of number using obj.args, the width as obj.width, the type of graph as obj.chart and whether the graph should be 3D or not as a Boolean data using obj.thd.



Line no.: 2-5: Defines the type of chart and its properties, whether the graphs should be in 3D or not.

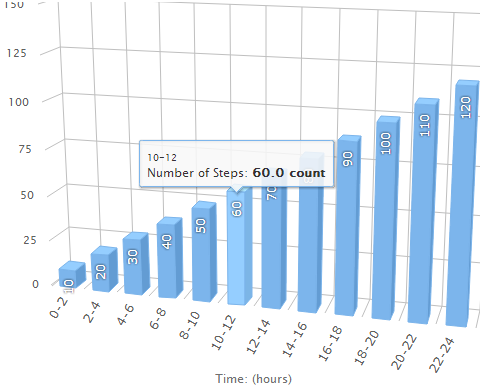
Line no.: 6-13: Defines the properties of X-axis, the fonts, font size etc.

Line no.:14: Defines the properties of Y-axis, the fonts, font size etc.

Line no.:15, 16: Defines the properties of the context menu that appears when graph is tapped.

Line no.:17-26: Encapsulates the data on which the graph is drawn.

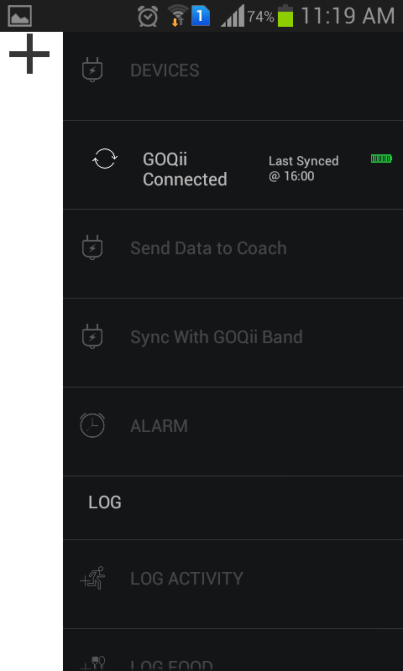
The above JSON would produce a graph which looks like this:



The above graph is a sample graph for Number of steps taken at a particular range of time intervals. A day is divided into 12 columns with 2 hours each. The graph is just for an illustration and doesn’t represent an actual data.

# PROJECT: SLIDING MENU

Sliding Menu is a special type of sliding layout which opens only till a specific length. Unlike *Navigation Drawer* functionality of Android, it does not overlap the main menu but pushes the main menu and then slides. This allows the user to see a part of main menu like Facebook and Google application.

For achieving this package named *grimbo.android.demo.slidingmenu was created*. Under this four classes were included*- HorzScrollViewWithListMenu, MyHorizontalScrollView, CustomList and SimpleGestureFilter*. Also for adding layouts we included different layouts required for the app.

## MyHorizontalScrollView

This class defines the horizontal scroll which we apply on the three layouts. This class also describes till where the layout slides. For this we took the width of the button present on the main layout and scrolled our layout till there so that the button will be visible when we scroll. Scrolling can be done with the help of a button or by swiping.

## CustomList

In this class we defined our list which appears on the left and the right layouts. In the list two ImageViews, two TextViews and two Views are added. This class also contains certain conditions to describe what happens when a particular row is pressed.

## SimpleGestureFilter

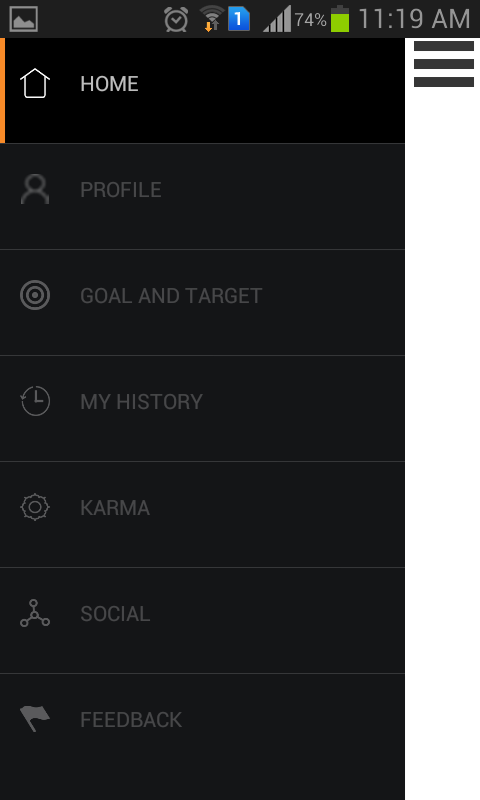
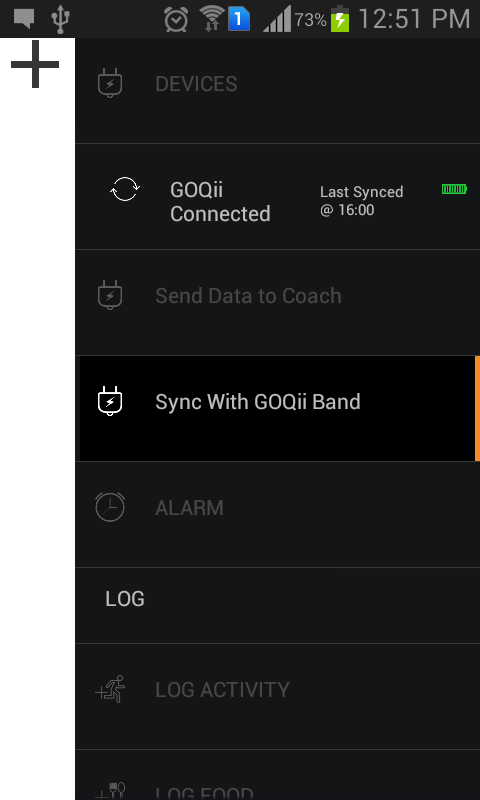
This class describes the swiping gesture. It calculates various distances in order to decide whether the user has swiped to right or left.

## HorzScrollViewWithListMenu

This class is the main activity class of the app. The images and the test written in every row has been describes here. All the actions performed are described in this class. This class interacts with the app and tells what it to do when something is clicked. It has methods like *onSwipe* and *onClick* which defines what does click and swipe do.

These are the main activities:

* On starting the app, the first highlighted row will be the HOME row.
* On clicking any row the text and image is highlighted and the background color changes to black.
* There appears an orange bar on right side or left side of the row depending on the layout opened.
* The color of all the unclicked items is changed to grey and the bar disappears.

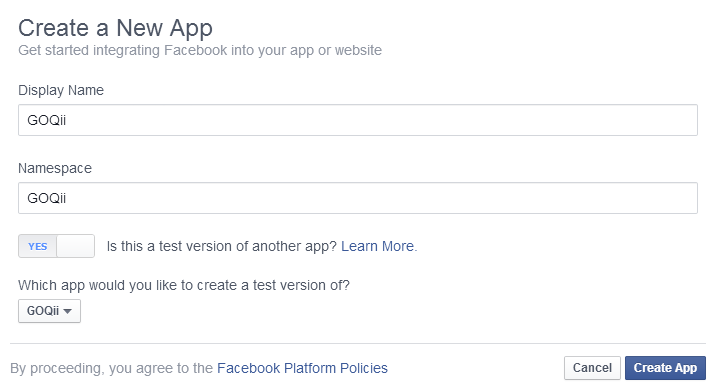
**LEFT SIDE MENU RIGHT SIDE MENU**

# PROJECT: SOCIAL INTEGRATION

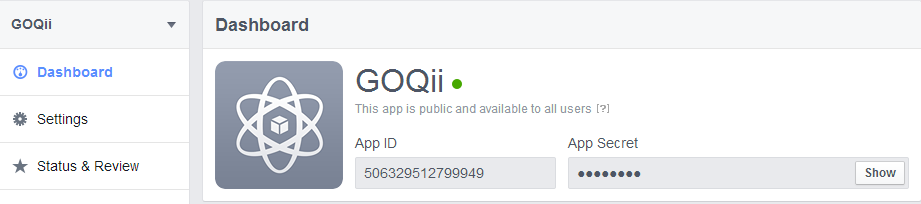
GOQii app needed to be integrated with social giants like facebook and twitter to enhance and personalise the user experience. The user would be able to share his activities which includes all the logging activities that the user performs using the app or band.

## Facebook:

The Facebook SDK for Android is the easiest way to integrate an Android app with Facebook's platform. The SDK provides support for Login with Facebook authentication, reading and writing to Facebook APIs and support for UI elements such as pickers and dialogs.

We had to import the Facebook SDK to Eclipse, which was our platform for android app development. In order to use the SDK, we had to register the app on <https://developers.facebook.com/>. 

We made a note of the App ID, which would enable us to use the Facebook SDK as a personalised user.



The samples provided with the SDK helped us to write code for the facebook integration. We also used the FQL (Facebook Query Language) to help us perform graph search to get the list of friends the user has. It also helps us in noting the current number of likes and shares to a particular post which was shared using the GOQii app. We customised the sharing in such a way that it links to the GOQii site ([www.goqii.com](http://www.goqii.com)).

### THE END