**习惯**

**APP**

**说明文档**

队伍名称： 陈永煊组

组长信息： 陈永煊 华南师范大学 计算机学院

组员信息： 卢凯锋 华南师范大学 计算机学院

赖卓宇 华南师范大学 计算机学院

黄毅敏 华南师范大学 计算机学院

目录

[一、 产品设计方案 3](#_Toc534465272)

[可行性报告 3](#_Toc534465273)

[产品定位及目标 4](#_Toc534465274)

[产品内容策划 4](#_Toc534465275)

[开发日程表 4](#_Toc534465276)

[技术解决方案 5](#_Toc534465277)

[推广方案 5](#_Toc534465278)

[运营规划书 5](#_Toc534465279)

[二、产品实现方案 6](#_Toc534465280)

[系统的主要功能 6](#_Toc534465281)

[UI界面设计 8](#_Toc534465282)

[关键技术和技术难点 8](#_Toc534465283)

[用户体验记录和分析 11](#_Toc534465284)

[已完成的改进和存在的问题 12](#_Toc534465285)

[三、 测试大纲和测试报告 12](#_Toc534465286)

[产品安装及使用说明 13](#_Toc534465287)

# 产品设计方案

## 可行性报告

习惯的养成决定一个人的成败，所以在当今的社会，养成好的习惯,改掉坏的习惯非常必要。但如今，随着智能手机的普遍，娱乐方式的多样化，养成好习惯似乎越来越难,同时改掉坏习惯也越来越难，刷抖音，打王者，轻易的把我们的注意力转移走。想坚持好习惯,改掉一个坏习惯，你需要一个习惯养成app。

1. **行业市场分析：**

当今社会，坚持变得越来越难,放弃变得越来越简单，养成好习惯和改掉坏习惯的重要性越来越高，但在这个诸多诱惑的时代，单靠个人意志力去养成习惯，对大部分年轻人来说是很困难的事。但是通过每个年轻人都离不开的手机,就提高了实现的可能性. 本产品是基于app的习惯养成应用，提供打卡功能，通过提醒功能鼓励使用者制定各种计划，坚持每天实现计划并打卡，持续进行以实现养成良好习惯的效果。

1. **同类产品分析：**

我们分析了两个Google Play上的同类app：“小日常”、“LOOP循环”。这两个软件都同样具有设置目标和日常打卡等基本的功能，其中“小日常”拥有非常良好的图形界面,使用简单，用户更愿意去使用

[](https://github.com/android-app-development-course/2018_4_Super/blob/master/picture/11.png)

而“LOOP循环”则是用户界面平凡,但却有非常好的数据统计功能,能够根据打卡情况生成表格,直观感受坚持习惯的总体情况。

1. **竞争优势：**

我们的app相较于同类型的其他app，联合了两个同类产品优点,又具有良好的图形界面,又能够兼具良好的数据统计功能.同时,同类的app主要目标都是养成好习惯,我们可以添加改掉坏习惯的功能,使我们的app功能更加强大,受众面更广

## 产品定位及目标

**目标用户群**

由于本产品目的是为了让用户改变并尽快养成习惯，因此本产品的目标用户应具备以下特点:

1. 较为年轻，善于尝试新鲜事物；

2. 需要改变习惯；

基于对日常生活的经验，在校大学生以及年轻的上班族是非常符合条件的用户群体，主要原因是大学生对自身原本的生活习惯不满意而自求改变或者是年轻的上班族所处的环境容易变化，当环境变化后需要改变之前的生活习惯等。

**应用场景**

1. 大学生小明，男，21岁，作为一名大三的学生，课程没有大二大一的多了，时间也充裕了起来，但他没有好好的利用好这些时间来提升自我，每天都是吃了睡睡了吃，他深感不能再这样颓废下去，因此他寻找一款能提醒与安排自身每日习惯的app。

2. 上班族小刚，男，26岁，作为一名出来社会不久的年轻人，几乎每年都在寻找更有利，更有前途的职位，因此他每年都在不一样的环境下工作，不同的工作时间经常打乱他的日常习惯，因此他下载了一个能打卡，能快速养成习惯的app来适应不同环境下的工作习惯。

## 产品内容策划

该app的主要功能模块包括：

\* 设置目标：用户可以自行设置要养成习惯的目标，以及持续的时间。

\* 日常打卡：用户可以在完成设定的目标后，可进行打卡，记录用户自己的心情、评论以及上传照片。

\* 分享功能：向他人分享自己达成的成就或自己的打卡记录。

\* 统计功能：统计完成的目标、养成的习惯、坚持的天数等，用户可以得到一个详细的统计信息。

\* 提醒功能：用户可以设置提醒时间，让app在规定时间提醒用户要完成的目标。

## 开发日程表

- 第一阶段：

- 设计app的各个UI界面

- 实现UI

- 第二阶段

- 逐步实现app的功能

- 根据实际情况，对app的功能进行调整

- 测试功能的实现效果

- 第三阶段

- 测试app运行情况

- 查找潜在的bug并修改

## 技术解决方案

本产品是一款打卡类型的应用,基于ANDROID 开发，主要的开发环境是：JDK + Android STUDIO+ANDROID SDK,主要开发语言是:JAVA+XML+SQL.

主要的难点是数据存储,可能涉及到数据库设计以及服务器搭建,服务器资金等问题,目前采取的解决方案是:暂不搭建服务器,将数据存储在手机存储空间里。

## 推广方案

为了充分吸引用户，本产品采用以下推广方案：

1. 平台推广:通过在各大安卓市场(例如:Google Play,应用宝,华为,小米,百度手机助手,豌豆荚等)同时上线,发布产品,提高用户量

2. 分享推广:在用户分享界面提供软件的二维码,使已知用户来发展未知用户

3. 社区推广:通过在微信,QQ,百度贴吧,微博等拥有大量用户的热门社区推广

4. 合作推广:与某些APP(例如其他同学的安卓产品)互相推广,来交换各自的用户。

## 运营规划书

本产品对于所有普通用户提供免费服务，商业运营模式如下：

1. 广告告服务:通过在某些空白位置镶入广告,以向广告发行方收取一定的广告费用,费用跟广告被点击的次数成正比.因此在用户数量比较多的时候才能获得可观的收入,在用户数量比较少的时候尽量不投入广告以此更加吸引用户来使用.

2. 赞助:通过每日打卡能够获取一定的积分,积分能够换取一些由赞助方提供的奖励,例如产品打折,实物奖励等等,将会向赞助方收取一定赞助费

3. 虚拟物品:通过出售软件内部的虚拟物品,例如补签卡,改名卡等等,获取一定收入.

4. 打赏功能:通过打赏功能,让好评用户提供资金来继续完善我们的APP,使用户得到更好的体验。

# 二、产品实现方案

## 系统的主要功能

（1） 设置习惯

输入：习惯名、提醒语、提醒时间、重复规律

输出：主界面显示新建的习惯

实现方法：通过弹窗输入对应的信息将信息记录到数据库并且MainActivity中保存该新建习惯的必要信息，并将其显示出来。

（2） 设置闹钟

输入：提醒时间

输出：到时间弹窗显示提醒内容

实现方法：通过遍历数据库，查找已设置的闹钟与现在时间最为相近的一个开启闹钟服务，通过软件接受服务开启弹窗提醒。

（3） 重设闹钟

输入：点击重设闹钟按钮

输出：无

实现方法：通过遍历数据库，查找在此时间之前的闹钟设置，选择与现在时间最接近的一个闹钟，并更改其闹钟设定。

（4） 打卡

输入：点击打卡按钮

输出：首页进度条增加

实现方法：通过监听对应按钮点击事件，在数据库中记录对应习惯已打卡的相应日期，并增加进度条进度。

（5） 修改习惯

输入：习惯名、提醒语、提醒时间、重复规律

输出：主界面显示修改后的的习惯

实现方法：点击习惯跳转到具体习惯页面后，点击右上角菜单栏，再点击修改，修改完保存即可。

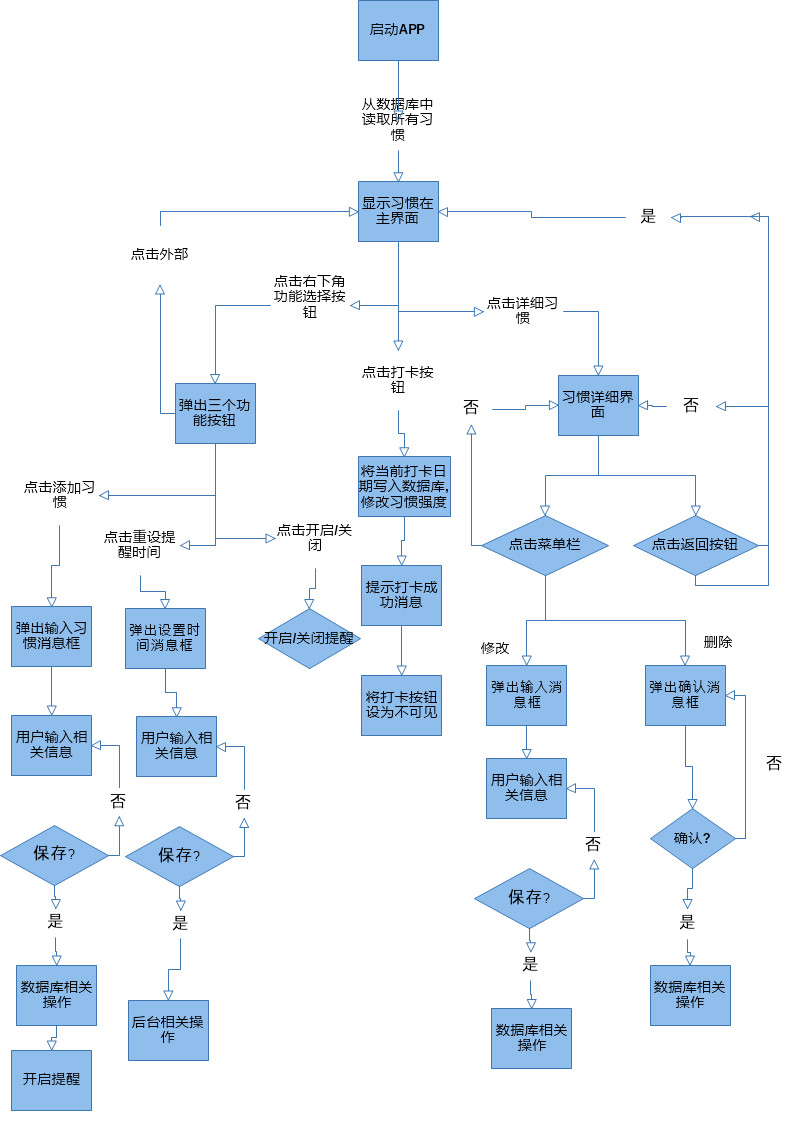
（6） 删除习惯

输入：点击删除菜单按钮

输出：无

实现方法：点击习惯跳转到具体习惯页面后，点击右上角菜单栏，再点击删除，删除数据库与该习惯相关的信息。

总流程图:



## UI界面设计

参考目前的主流app的设计，决定以简洁的UI作为软件设计的总体方向，并设计#009777的深绿色作为软件的主题色，辅以橙黄色的渐变色作为点缀。

软件的主界面设计，为了实现高度自定义的扇形展开式按钮，重写了AppCompatActivity，在MainActivity继承，方便在不同界面重用。主界面采用RecyclerView 和对应adapter的设计来展示不同的习惯以及用com.xx.roundprogressbar.RoundProgressBar来展示目前进度，当点开app或者进度发生变化时会有进度条前进的动画，RecyclerView的notifyItemMoved 方法实现长按adapter来进行拖动调换位置。

子页面运用同样的颜色搭配来展现用户的习惯打卡记录。从上至下依次为已打卡日历表，最长连续签到日期进度条以及习惯强度折线图。

## 关键技术和技术难点

**重复定时发送通知提醒**

定时使用到的是SQLite类、service类、AlarmManager类

通知使用到的是BroadcastReceiver类、NotificationManager类

设置时间调用TimePickerDialog

**public void** showTimePickerDialog(Activity activity, Calendar calendar, **final** Habit habit) {  
 **new** TimePickerDialog(activity,  
 *// 绑定监听器* **new** TimePickerDialog.OnTimeSetListener() {  
 @Override  
 **public void** onTimeSet(TimePicker view, **int** hourOfDay, **int** minute) {  
 habit.**remind\_hour** = hourOfDay;  
 habit.**remind\_minute** = minute;  
 habit.**remind**=**true**;  
 Calendar cal = Calendar.*getInstance*();  
 cal.set(cal.get(Calendar.***YEAR***), cal.get(Calendar.***MONTH***), cal.get(Calendar.***DATE***), 0, 0, 0);  
 cal.set(Calendar.***MILLISECOND***, 0);  
 }  
 }  
 *// 设置初始时间* , calendar.get(Calendar.***HOUR\_OF\_DAY***)  
 , calendar.get(Calendar.***MINUTE***)  
 *// true表示采用24小时制* , **true**).show();  
}

将该时间写入数据库中，并启动服务。服务的工作是获取当前的时间并遍历数据库，在数据库中寻找的距离当前时间最近的习惯提醒，设置定时发送广播

AlarmManager类创建实例的方法如下：



参考博客，了解到AlarmManager的用法



我们的提醒功能在发送完通知后，启动服务再次寻找距离最近的时间设置提醒，所以我们只用设置一次性闹钟，选择第一种方法。

type选择RTC\_WAKEUP，trggerAtMillis通过Calendar类将时间转换成毫秒

代码为

AlarmManager am = (AlarmManager) getSystemService(***ALARM\_SERVICE***); *//这里是系统闹钟的对象*PendingIntent pi = PendingIntent.*getBroadcast*(**this**, 0, startNotification, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
Calendar instance = Calendar.*getInstance*();  
instance.set(Calendar.***HOUR\_OF\_DAY***, **hour**);*//小时*instance.set(Calendar.***MINUTE***,**minute**);*//分钟*instance.set(Calendar.***SECOND***, 0);*//秒//设置事件*am.set(AlarmManager.***RTC\_WAKEUP***, instance.getTimeInMillis(), pi);*//提交事件，发送给 广播接收器*

BroadcastReceiver类收到广播后，获取广播中的信息

调用NotificationManager类发送通知

通过查阅资料，对NotificationManager的使用有了了解





代码为

Intent intent = **new** Intent(context, MainActivity.**class**);*//这里是点击Notification 跳转的界面，可以自己选择*PendingIntent pi = PendingIntent.*getActivity*(context, 0, intent, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
**manager** = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
Notification notification = **new** NotificationCompat.Builder(context)  
 .setSmallIcon(R.mipmap.***assist\_9***) *//设置通知图标。* .setTicker(**content**) *//通知时在状态栏显示的通知内容* .setContentInfo(**"习惯提醒"**) *//内容信息* .setContentTitle(**title**) *//设置通知标题。* .setContentText(**content**) *//设置通知内容。* .setAutoCancel(**true**) *//点击通知后通知消失* .setDefaults(Notification.***DEFAULT\_ALL***) *//设置系统默认的通知音乐、振动、LED等。* .setPriority(NotificationCompat.***PRIORITY\_MAX***)  
 .setContentIntent(pi)  
 .build();  
**manager**.notify(***NOTIFICATION\_ID\_1***, notification);

通过该种方法实现了定时重复通知的功能。

**长按adapter拖动实现位置交换**

这个功能主要还是参照RecycleView原本的特性实现的，参考了网上recycleview实现瀑布流的方法：

<https://blog.csdn.net/lmj623565791/article/details/45059587>

代码中对recycleview使用方法：rv\_falls.setLayoutManager(new StaggeredGridLayoutManager(2, StaggeredGridLayoutManager.VERTICAL)); 实现了两列的瀑布流布局，再对adapter使用方法Adapter.notifyItemMoved(startPos, endPos);实现长按拖拽交换位置。

## 用户体验记录和分析

用户体验反映如下：

**优点**：

（1）功能实用，能起到培养习惯的效果；

（2）使用简单，没有复杂的操作；

（3）统计数据做的很全面，用户可以分析自己的习惯

**缺点**：

（1）具体按钮的功能不够清晰，不清楚按钮用途；

（2）在习惯的统计页面，各个具体图表内容不清楚；

（3）统计页面的折线图关键点缺少，数据没有显示；

（4）点击删除会直接删除习惯，不能防止误触；

（5）只有绿色的主题色不符合其气质。

**基本分析**：

根据用户反馈，我们的app优胜之处在于功能齐全实在，数据统计全面，能满足大部分用户的需求，而不足之处在于部分需要关注细节的地方还有所欠缺，给予用户的提醒做得还不够。因此我们根据用户反映进行了后期的改进，已经修改了绝大部分令用户不满意的地方。

## 已完成的改进和存在的问题

**已改进：**

1. 添加按钮功能注释；
2. 添加图表内容注释；
3. 增加点击折线图对应部分显示详细数据功能；
4. 删除前弹窗提示询问是否删除，多重确认。

**存在的问题：**

1. 程序不在后台时，手机系统可能会中止通知服务。
2. 应该提供给用户更改主题色的功能。

# 测试大纲和测试报告

**内容 测试结果**

新建习惯速度 不到一秒

定时通知功能 正常

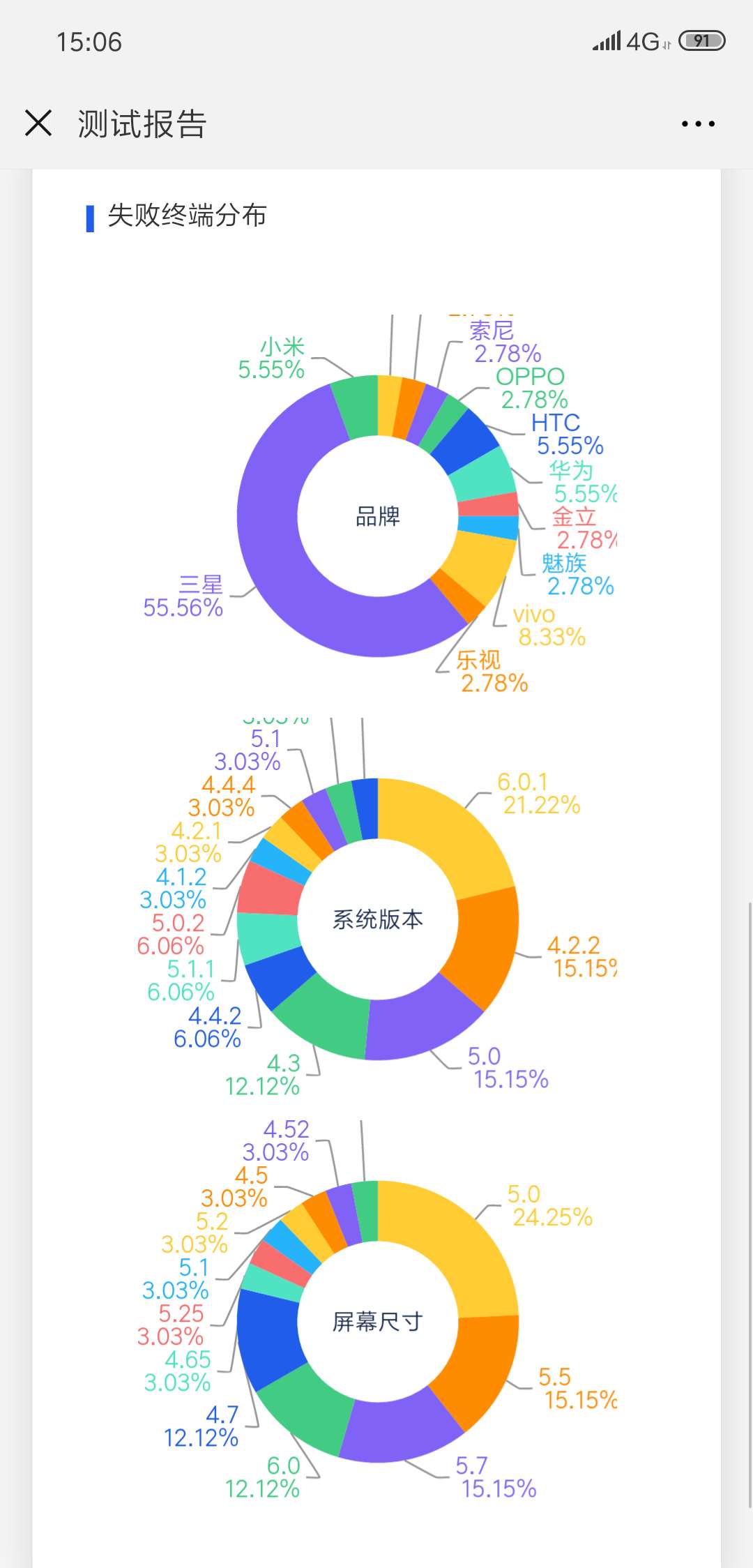
修改通知功能 正常

关闭/开启通知功能 正常

修改习惯功能 正常

删除习惯功能 正常

软件安装 安卓系统版本在7.0及以上



## 产品安装及使用说明

**安装方法：**

点击软件apk包即可安装。

**使用说明：**

1. 安装完成后点击主界面“+”号再点击新建习惯按钮即可新建需要养成的习惯。



1. 根据弹窗提示输入需要添加的内容即可
2. 主界面会显示目前已有的习惯以及进度，每次打卡进度将会增加。
3. 点击对应习惯能查看目前已打卡的日期、最长连续打卡天数以及已有的习惯强度。
4. 在对应习惯的详情界面菜单栏的按钮中能对习惯进行修改和删除。
5. 当设置时间到后会通知提醒用户打卡，点击通知即可跳转app。