怎样设计一份完整的App的测试用例

一·测试用例存在的意义/目的；

作为测试人员，测试用例设计这个环节存在的必要性和意义；通过对需求点的测试设计从而避免测试点的遗漏，测试用例设计的存在对质量和效率起到最基本的促进作用。

从效率角度，在开发人员提测前就做好的高质量测试设计，在测试执行阶段，执行的效率更高，而且不用再去费心构造设计，按计划执行完测试用例后，那么这个需求的测试就基本完成了；（测试过程中发现新的问题/漏掉的测试点/或需求调整时，需要及时补充）；

二·什么是好的测试用例？

每个测试人员的用例设计风格各有不同，但好的测试用例设计有以下共性：

1.测试设计结构组织合理。从测试用例的组织是开展测试的起点，良好的组织能够帮助我们快速定位到我们想关注的部分，这个部分的好坏关系到测试工作的持续性发展。  
2.测试用例设计覆盖全面且不冗余，用精简的语言描述清楚一条测试用例，任何人员按照用例步骤也可以完成测试；用较少的测试用例描述清楚需求测试点的覆盖。  
3.测试用例设计具有可执行，可判定，可再现的特点，即在测试前提符合的前提下，按照测试步骤每一个测试用例都可以顺利执行，同时呈现相应的预期结果，而且测试用例在被多次执行的结果都应该是相同的。

三·分析需求并设计框架：

1.分析需求重要性不言而喻，根据需求设计层层剥析，对开发的设计逻辑深入理解，最后再去构造/设计测试用例，这样测试点挖掘的往往更深，场景更全，发生漏测的几率也更低。

2.对于App，还需要考虑移动端应用不可忽略的场景与隐形需求；如（UI交互，性能，压力，兼容，中断测试等异常情况）；

3.建议由提纲挈领到逐步细化，先写功能点，颗粒度适中，细节方面可以放在用例中在进行覆盖；

4.通过思维导图等方式列出所有的测试点，合理的设计测试用例框架（如分类模块，摘出公共模块、专项测试等），能更全面地覆盖测试点；

四．App测试用例设计：

移动端App存在于不同领域及行业，针对不同业务需求会用不同的设计方式；

但一般都会包含以下这些方面：

1．功能性测试：

（1）根据不同产品需求，对应不同的功能需求

（2）根据经验/对比竞品，分析出隐形需求；

2．兼容性适配性测试：

（1）Android、iOS版本的兼容性

（2）手机分辨率兼容性

（3）网络的兼容性：2G/3G/4G/Wifi，弱网下、断网时

（4）APP跨版本的兼容性

（5）不同厂商手机兼容性

（6）手机屏幕适配（刘海屏/曲面屏）

3.安装、卸载测试

（1）生成apk文件在真机上可以安装及卸载；

（2）Android手机端通过使用安装工具。如：豌豆荚

（3）App Store安装与卸载

4.在线升级测试：

（1）验证数字签名

（2）升级后可以正常使用

（3）在线跨版本升级。

5．网络测试：

（1）外网测试主要现实模拟客户使用网络环境，检验客户单程序在实际网络环境中使用情况进行业务操作。

（2）外网测试主要覆盖到WiFi/2G/3G/4G...net/wap、电信/移动/联通、所有可能的组合进行测试。

原则：

（1）尽可能全面覆盖用户的使用场景，测试用例中需要包含不同网络排列组合的各种可能。

（2）还有模拟信号屏蔽时候。客户端的影响等。还有做外包场景测试，在高山、丘陵、火车上等特殊环境下进行全面测试、

6．接口性测试：

（1）client端和service端的交互

（2）client端的数据更新和service端的数据是否一致

（3）Client端更新时断开了

（4）client端更新时和service端挂了

7．业务逻辑测试：

（1）业务逻辑测试：主要测试客户端业务能否正常完成。

（2）功能点测试：主要测试客户端功能点是否正常使用

（3）关联性测试：主要测试客户端与PC端的交互，客户端处理完后，PC端与客户端数据一致。

8．异常测试：

（1）交互异常性测试：客户端作为手机特性测试，包括被打扰的情况：如来电、来短信、低电量测试等，还要注意手机端硬件上，如：待机，插拔数据线、耳机的各操作不会影响客户端。

（2）异常性测试：主要包含了断网、断电、服务器异常等情况下，客户端能都正常处理，保证数据正确性。

**9. 安全测试：**

性能测试可以作为专项测试：

# 系统性能测试：（负载测试，压力测试，极限测试，容量测试）

<1>性能分析：

**1. 系统业务特点和用户行为分析；**

(1) 系统使用高峰时段分析

(2) 系统高峰期业务应用分析

**2. 系统性能指标分析**

(1) 并发用户数量设计

a）极限法

b）用户趋势分析

c）经验评估法

(2) 事务平均响应时间

**3. 性能测试执行策略分析**

(1) 单业务测试

(2) 混合业务测试

(3) 疲劳强度测试

<2>一般关注点：

（1）压力测试：

（2）电量流量测试

（3）CPU、内存消耗

（4）APP启动时长

（5）Crash率

（6）内存泄露

（7）客户端侧性能测试：

1.基准性能测试：主要通过压服务器端接口测试及客户端在不同网络环境下响应速度。

2.大数量的测试：主要在特定环境下，客户端一次性更新大量的数据及人员列表时，客户端能否正常处理，分为三种情况：

    a.客户端第一次使用，第一次就更新大量数据及人员列表。

   b.客户端在平时更新中，更新大量的数据

    c.客户端已经在手机本地下载跟多数据后，再次更新大量

五·设计测试用例方法：

<1> 流程分析法：

---主要是针对业务流程来做测试； 用户使用场景：如安装流程测试；

1.三个概念

基本流：核心的、最常用的流程，而且是成功的流程； ---等级：高

备选流：会出现，不是特别常用的流程，也是成功的流程； --等级：中

异常流：流程不能走通的、不成功的流程； ---等级：低

2.注意：一条测试路径可以对应一个用例，也可以对应多个用例；

流程分析法重点关注的是流程是否能走通；至于每个中间节点的非法情况，可以单独再设计用例来覆盖测试；

<2> 等价类：

可以在任何地方使用等价类来设计用例；

可以与 边界值/输入域/输出域 等方法结合使用！

1.思路：

将所有可能出现的取值一一列举出来，然后将这些取值进行分类，分为类之后，一个类中的所有取值都是等价的，那么从每一类中挑一个进行测试即可，如果这个测试完之后没问题，那么从这一类中挑选其他的取值来测试也是没有问题的。因为它们之间是等价的。

2.分类原则：

如果有一些取值它的输出是一样的，并且处理规则也是一样的，那么就可以将这些取值放到一个类中； --这个类就是等价类。

3.两个概念：

有效等价类：满足条件的，即：是合法的

无效等价类：不满足条件的； 即：是非法的。

<3> 正交试验法：

1. 应用场景：主要用在多条件组合查询测试。
2. 优缺点：

等价类： 优点：高效 缺点：不组合

正交试验法： 折中；考虑组合，但是工作量不会特别大 ---推荐的

----考虑组合，但是考虑的不是所有输入之间的全排列组合；考虑的是每两个输入之间的全排列组合。

所有输入之间全排列组合：优点：测试全面； 缺点：工作量偏大

3.思想：如果每两个输入做了全排列组合测试，没有问题，那么就认为三三组合在一起不会有问题，四四组合也不会有问题，更多的组合在一起都不会有问题。

4.两个概念：

因子：指的是输入 ； 例如：查询条件

状态：指的是输入的取值；

<4>边界值分析法：

是一种补充方法；针对等价类这个方法来进行补充；补充如果有边界要采用边界的值进行测试；

1.思想：边界是问题多发区，一定要覆盖边界的测试。

2.三个概念：

上点：边界上的点；

离点：离边界最近的点

内点：范围内的点

3.测试原则：优先测试上点和离点；内点的测试优先级低，有时间就测试，没时间也可以不测试。

4.原因：要求上点和离点要分别覆盖有效和无效的情况；先确定上点，再去离点；如果上点在有效，那么离点从无效中取；如果上点在无效，那么离点就从有效中取。

5.应用场景：凡是有边界的地方都能用

<5> 状态迁移法：

1.应用场景：用来测试状态之间的切换的测试

如：播放模块，语言切换功能等； ---针对不同状态之间的切换进行测试

<6> 输入/输出域：

补充方法；

只要有输入与输出，就可以使用；

<7> 判定表：

1.应用场景 - 主要用来测试游戏规则、负责的处理规则

优点：覆盖所有输入之间的全排列组合，测试最全的； 同时考虑输入和输出的。

缺点：工作量比较大，费时间；设计的过程比较麻烦一些。

2.四个概念：

条件桩：指的是输入

条件项：指的是输入的取值；

动作桩：指的是输出

动作项：指的是输出的取值

条件对应输入； 动作对应输出；

<8> 因果图法：在判定表上的基础上做了改进；

<9> 错误猜测法：

----只是一个补充方法；在用其他方法设计完用例之后，可以选择用这个方法来补充一下。

1. 注意：不能出现一个功能模块的用例全部都是用错误猜测法来设计的。
2. 根据自身的测试经验来猜测开发容易遗漏、犯错的地方；

关键点： (1).测试经验；

(2).对开发的了解；（了解开发的技术能力、业务能力等等）

<10> 异常分析法：

----主要是针对异常测试来设计用例的;就是针对被测对象来分析哪些异常可能会出现，是需要测试的。

1.可靠性测试：---逆向思维的测试 ---测试的优先级比较低

a.随机测试：毫无目的在页面进行随机测试； --在发布软件之前；

b.异常测试：--破坏性测试； 手动来构造一些异常来测试被测软件的应对能力；

2.常见的异常：

硬件方面的异常：磁盘空间满了、突然掉电、磁盘损害 等等

网络方面的异常：网络断连、网络闪断等等

数据库方面的异常：表空间满了、表丢失了等等

**六.用例规范；**

1. 一般的测试用例包含：
2. 用例编号：按照项目需求来编号，目的是为了将编号对应不同的模块或类别，方便查找；
3. 一级/二级模块：所属的模块；
4. 用例描述/用例标题：尽量描述简介清晰，能表明用例的重点条件或步骤；
5. 优先级/重要级别：按照该用例所覆盖的需求排列优先级，一般优先级高的写在前面；
6. 前置条件：说明用例所需要的必要条件；当一些条件不易切换时，为了方便测试，可以按照前置条件来设计用例；
7. 用例步骤：描述要清晰，明确；颗粒度把握好；
8. 预期结果：按照步骤操作时一定会出现的结果；
9. 实际结果/测试结果：测试时的实际结果；

有的用例是直接标注Ture/Fales；

1. 备注/描述： 添加备注，或对实际测试结果进行描述，方便定位；
2. 测试人员
3. 开发人员

也有可能会包含：再次测试结果，是否通过等；