# 目录

[目录 1](#_Toc52300804)

[1. 简介 1](#_Toc52300805)

[1.1测试目的 2](#_Toc52300806)

[1.2背景 2](#_Toc52300807)

[1.3范围 3](#_Toc52300808)

[2. 测试需求 4](#_Toc52300809)

[3.测试策略 4](#_Toc52300810)

[3.1测试类型 4](#_Toc52300811)

[3.1.1 数据和数据库测试 4](#_Toc52300812)

[3.1.2 功能测试 5](#_Toc52300813)

[3.1.3 用户界面测试 6](#_Toc52300814)

[3.1.4 性能测试 6](#_Toc52300815)

[3.1.5 容量测试 7](#_Toc52300816)

[3.1.6 访问和安全性测试 7](#_Toc52300817)

# 简介

## 1.1测试目的

1. 确保该开发的软件产品完成了它所包含或承诺的全部功能，并且所
2. 有的用户可以访问到的功能都有明确的书面说明。
3. 确保用户使用界面简洁美观大方。
4. 确保该软件满足性能和效率的要求，满足用户实际需求。
5. 确保该软件访问的安全性，避免数据信息泄露。

### 1.2背景

移动应用时代的到来，使得高校信息化这一话题又有了新的生命。在数据中心不断增强的服务性能之外最显著的特征就是在用户移动端的精彩表现。单纯用PC的时代将一去不复返。以手机、平板电脑介质为代表的移动终端应用将为高校信息化带来巨大变革。移动应用不只是在手机上运行软件那么简单，它涉及到高校信息化应用场景的完善、扩展，让数据无所不在，通过广泛的产业联合作为用户提供低成本整体解决方案。  
  
 在高校及科研院所，每天都会产生大量的业务数据、一卡通日常消费流水、日常网络行为日志等大流量数据，这些数据虽然对于其他业务意义不大，也没有共享的必要，但针对这些数据建立专门的数据仓库和大规模数据处理架构，进行分布式数据采集、分布式计算、虚拟化、内存计算等方式，建立一整套从采集、存储、分析、计算、展现的架构将有利于智能部门掌握数据走势，并在此基础上提供决策支持。因此，针对日常数据建立数据仓库，并在此基础上对数据进行展示与分析是当前高校信息化的工作重点之一。  
  
 同时，在高校及科研院所，一些公共设施经常需要定期维护和及时维修，不及时的检修和一些突发的事件可能会造成重大财产损害，甚至危害学生和职工的生命。如何防治和及时应对，成为大家普遍关心的热点问题。根据调查，比如在高校校园内，经常有公共设施:房屋、电力设备等的损坏，经常会出现教学用品：电脑、投影仪等的故障，这些情况大大影响了正常的教学工作活动，及时地维修和定期的维护是必要的。如果仅仅等待维修人员的定期检修和维护，一些重大突发的事故隐患可能无法及时发现。作为学校的学生和教工群体遇到这类事情，往往会发现不知如何处理，不知道向那个部门汇报的情况，传统的电话报修系统也不便于管理和维护，无法支持先进的数据分析和智能故障预测。并且，对于目前高校，如果大量增加维修保养人员无疑会增加额外的支出。

在这样的现实需求中，学生和教工普遍使用的智能手机中开发手机应用程序，构造一个智慧校园应用系统将可以很好地解决这一问题。

## 1.3范围

代表性单元测试代码：  
    @Test  
    public void countByProvinceAndStu\_grade() {  
        List<List> sb = studentBasicRepos.countByProvinceAndStu\_grade();  
        //  int length = sb.size();  
        for (List a : sb) {  
            System.out.println(a.get(1).getClass());  
  
            System.out.println(a.get(0) + "  / " + a.get(1) + "   /  " + a.get(2));  
            //.out.println();  
        }  
    }

    @Test  
    public void countByCityAndStu\_grade() {  
        List<List> sb = studentBasicRepos.countByCityAndStu\_grade();  
        //  int length = sb.size();  
        for (List a : sb) {  
            System.out.println(a.get(0) + "  / " + a.get(1) + "   /  " + a.get(2));  
            //.out.println();  
        }  
    }

 @Test  
    public void countByCity() {  
        List<List> sb = studentBasicRepos.countByCity();  
        //  int length = sb.size();  
        for (List a : sb) {  
            System.out.println(a.get(0) + "  / " + a.get(1));  
            //.out.println();  
  
        }  
    }

# 测试需求

1 . 性能/容量：需要支持的用户的数量（尤其是峰值的并发用户数量），用户能够接受的响应时间，数据规模（TPC-C/TPC-D以及新一代的业界性能测试标准中还有千万级数据规模的测试项目）

2 可靠性/可用性/可复原性： 可靠性之与在规定的一段时间和条件下软件维持其功能服务以及性能水平的能力有关的一组属性（可用性是另外一种说法）。

1. 可维护性/可管理性：包括系统在无人工干预条件下的稳定性，自排错能力，可测试性都属于这个范畴。故障的可排查能力，系统的修正，升级，备份，恢复机制以及方便与否，都属于这个范畴。
2. 安全：传输加密，存储加密，可破解性，以及各种未被授权的用户行为如何防范和控制。
3. 易用性
4. 数据一致性：包括数据的编码和语言，冗余数据的一致性要求

# 3.测试策略

## 3.1测试类型

### 3.1.1 数据和数据库测试

|  |  |
| --- | --- |
| 测试目标： | 确保数据库访问方法和进程正常运行，数据不会遭到损坏。 |
| 方法： | 1.调用各个数据库访问方法和进程，并在其中填充有效的和无效的数据或对数据的请求。  2．检查数据库，确保数据已按预期的方式填充，并且所有数据库  3．事件都按正常方式出现:或者检查所返回的数据，确保为正当的理由  检索到了正确的数据。 |
| 完成标准： | 所有的数据库访问方法和进程都按照设计的方式运行，数据没有遭到  损坏。 |

### 3.1.2 功能测试

|  |  |
| --- | --- |
| 测试目标： | 确保测试对象的功能正常，其中包括导航、数据输入、处理和检索等。 |
| 方法： | 利用有效的和无效的数据来执行各个用例、用例流或功能，以核实以  下内容:  1.在使用有效数据时得到预期的结果.  2．在使用无效数据时显示相应的错误消息或警告消息.  3．各业务规则都得到了正确的应用。 |
| 完成标准： | 所计划的测试已全部执行。  所发现的缺陷已全部解决。 |

### 3.1.3 用户界面测试

|  |  |
| --- | --- |
| 测试目标： | 核实以下内容:  1．通过浏览测试对象可正确反映业务的功能和需求，这种浏览包括窗口与窗口之间、字段与字段之间的浏览，以及各种访问方法.  2．（Tab健、鼠标移动和快捷键)的使用  3．窗口的对象和特征(例如: 菜单、大小、位置、状态和 中心)都符合标准。 |
| 方法： | 为每个窗口创建或修改测试，以核实各个应用程序窗口和对象都可正确地进行浏览，并处于正常的对象状态。 |
| 完成标准： | 1.证实各个窗口都与基准版本保持- -致，或符合可接受标准  2.需考虑的特殊事项: .  并不是所有定制或第三方对象的特征都可访问。 |

### 3.1.4 性能测试

|  |  |
| --- | --- |
| 测试目标： | 1.核实所指定的事务或业务功能在以下情况下的性能行为:  正常的预期工作量  预期的最繁重工作量 |
| 方法： | 1使用为功能或业务周期测试制定的测试过程。  通过修改数据文件来增加事务数量，或通过修改脚本来增加每项  事务的迭代次数。  脚本应该在- -台计算机上运行(最好是以单个用户、单个事务为  基准)，并在多台客户机。 |
| 完成标准： | 单个事务或单个用户:在每个事务所预期或要求的时间范围内成功地完成测试脚本，没有发生任何故障。  多个事务或多个用户:在可接受的时间范围内成功地完成测试脚本，没有发生任何故障。 |

### 3.1.5 容量测试

|  |  |
| --- | --- |
| 测试目标： | 核实测试对象在以下大容量条件下能否正常运行:  连接(或模拟了)最大(实际或实际可承受)数量的客户机，所有客户机在长时间内执行相同的、且情况(性能)最差的业务功能。  已达到最大的数据库大小(实际的或按比例缩放的)，而且同时执行了多个查询或报表事务。 |
| 方法： | 使用为性 能评价或负载测试制定的测试。  应该使用多台客户机来运行相同的测试或互补的测试，以便在长时间内产生最繁重的事务量或最差的事务组合(创建最大的数据库大小(实际的、按比例缩放的、或输入了代表性数据的数据库)，并使用多台客户机在长时间内同时运行查询和报表事务 |
| 完成标准： | 所计划的测试已全部执行，而且在达到或超出指定的系统限制时没有出现任何软件故障。 |

### 3.1.6 访问和安全性测试

|  |  |
| --- | --- |
| 测试目标： | 应用程序级别的安全性: [核实主角只能访问其所属用户类型已被授权使用的那些功能或数据。  系统级别的安全性:核实只有具备系统和应用程序访问权限的主角才能访问系统和应用程序 |
| 方法： | 应用程序级别的安全性: 确定并列出各用户类型及其被授权使用的功能或数据。  为各用户类型创建测试，并通过创建各用户类型所特有的事务来核实其权限。  修改用户类型并为相同的用户重新运行测试。对于每种用户类型，确保正确地提供或拒绝了这些附加的功能或数据。 |
| 完成标准： | 各种已知的主角类型都可访问相应的功能或数据，而且所有事务都按照预期的方式运行，并在先前的应用程序功能测试中运行了所有的事务。 |