2装备管理系统方案

整个管理系统我们在设计时，使用了MVC的设计模式以及层级封装的设计思想，将整个系统分为多个功能模块，在开发时以测试先行，分块实现，在各个模块进行了单元测试满足最终功能要求时再进一步合并。同时，在数据传输和维持部分，我们使用了SQLite数据库，以及使用静态内存共享类进行数据交互传输。

2.1装备管理系统系统方案

整个项目我们分为如下几个模块：数据库部分，包括数据库连接模块，数据库管理模块，数据维持内存共享模块和SD卡中的图片库。工具部分，主要包括UI界面组件Adapter的重写，以及核心QR码读写分析调用。UI部分，包括增加界面，删除界面，修改界面，查找界面，分类总览界面以及QR码扫描界面。

各模块具体功能分析如下：

数据库连接模块，实现了和数据库交互的底层操作，包括创建新数据库，更新数据库版本。

数据库管理模块，实现了对数据库不同表项增删改查基本操作，同时对查询出的结果进行数据封装，生成UI界面和工具模块可直接处理的数据形式。

数据维持内存共享模块，因为整个系统包含多级的数据交互，考虑到Java语音和Android系统的内存管理机制，我们是通过一个DataHolder类中的Static成员进行有效数据保持，从而实现了多个UI界面共享处理数据，以及多级数据传递的功能。

SD卡中的图片库，程序中我们提供单项的图片属性添加，由此我们需要提供一个本地的图片保持机制。然而，若直接将图片文件以Image形式添加进数据库中，由于图片文件的大信息量的特点，势必会给数据库增加压力，随着使用过程的不断深入，图片文件逐渐增多，数据库的查询和搜索效率将会受到影响。考虑到以上缺陷，我们在保存图片的时候，使用SD空间，创建本地的文件文件夹，专门用于对应项的图片保存，同时，程序中会对SD卡的图片库进行严格的读写控制，维持图片库的总体规模。

UI界面组件工具，主要是为分类总览界面重新封装数据Adapter工具类，因为程序的特异性，我们需要对我们特殊的数据格式进行处理，所以我们无法直接使用系统自带提供的Adapter。

QR码工具部分，主要实现了对ZXing包的调用和获取数据的封装和处理。

增加界面，增加界面提供向数据库内新增项目，在该界面后台中，我们通过使用数据库工具，实现了防止同名项目的创建等容错功能。

删除界面，删除某项数据

修改界面，该界面中提供了对某项数据的修改，同时可以给该项数据添加图片注释内容，图片是通过调用设备自带的摄像头来获取的。

查找界面，该界面实现了对数据的具体信息查询，可以查看对应的内容和该数据项目的图片注释内容

分类总览界面，使用了ExpandableListView组件，实现了分类，多级显示条目。同时，点击条目会自动跳转到删除，修改和查询该条目的界面去。

QR码扫描界面，调用摄像头捕捉和QR码分析工具，实现对QR码的采集，并直接对该扫描数据进行查询，转入查询界面

图片采集界面，调用设备摄像头，拍摄图片，保存进图片库，并关联图片和该数据项键值。

2.2.1 SQLite数据库管理系统

目前SQLite以其出色的性能和轻量的规模，受到了很多移动开发平台的青睐，当然我们所使用的Android平台作为最流行的移动系统，更是对SQLite完美支持。系统的开发环境之间就提供了SQLite的环境，开发者不需要额外配置。SQLite的使用方式和普通的大型数据库，如Mysql，MSsql，基本雷同。Android开发中的SQLite数据库使用支持，原始Sql语句的操作以及Android环境特有的 ContentProvider工具类的操作。当然，更为强大和详细的操作介绍，由于本系统使用不到，所以在此不作赘述。

本系统中，主要是在DataHelper类中调用Android系统的SQLiteHelper类，实现了对数据库创建和更新的底层操作，可以理解为是和本系统中数据库的根本联系模块。而在DataManager数据库操作类中进行了对数据库操作的封装，实现了具体的增删改查相关功能和数据封装。SQLite数据库为本程序提供了强大的后台支持，使得程序的性能得到了保证。

2.2.2 ZXing条码扫描工具

ZXing是由谷歌公司支持开发的开源QR码扫描项目，鉴于谷歌公司杰出的世界级团队引领，我们有理由相信，站在这样的巨人的肩膀上一定会看到的更远。经过对ZXing项目的详细分析和取用，我们参考ZXing的源码，自己重新写了ZXing工具包，实现了QR码的扫描，经过实际测试，我们发现该功能模块性能稳定，而且高效。