软件代码走查说明

城院生态圈

——基于微信小程序的城院动植物交流论坛



项目名称： 城院生态圈

专业班级： 软件工程1903

小组成员：黄依豪，梁晓勇，李东泽

指导教师： 杨枨老师

软件代码走查说明

说明：

《软件代码走查说明》 描述了软件在编程开发过程中根据代码规范以及功能检测对所涉及的代码格式以及性能等一系列检查。该文档可能还要用《软件测试报告》和《软件代码清单说明》以补充。

**版本记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 01 | 2021/12/12 | 0.1 | 黄依豪 | 报告的初始版本制作 |
| 02 | 2021/12/14 | 0.2 | 黄依豪、李东泽、梁晓勇 | 报告内容的填充 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[软件代码走查说明 2](#_Toc60075366)

[1引言 4](#_Toc60075367)

[1.1标识 4](#_Toc60075368)

[1.2系统概述 4](#_Toc60075369)

[1.3文档概述 5](#_Toc60075370)

[1.4基线 5](#_Toc60075371)

[2引用文件 5](#_Toc60075372)

[3代码走查规范报告 6](#_Toc60075373)

[4注解 17](#_Toc60075374)

[附录 18](#_Toc60075375)

**根据《GB8567－06计算机软件产品开发文件编制指南》中项目测试阶段结合实际情况调整后的《软件代码走查说明》内容如下：**

# 1引言

## 1.1标识

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： | SE2021-G005-软件代码走查说明 |
| 当前版本： | 0.2 |
| 作者： | 黄依豪，梁晓勇，李东泽 |
| 完成日期： | 2021-12-14 |

## 1.2系统概述

本文档用于说明对于“城院生态圈”的软件测试计划。

本项目是通过小程序实现用户的交流论坛——“城院生态圈”，以爱护动植物、科普相关知识为目的，作为一个论坛平台供校内学生们交流动植物。力争做到界面清新、充实功能、无广告、无收费，操作简单易上手，吸引用户加入。本项目实现的小程序最终会在微信小程序的搜索栏下找到并运行。

开发历史：

2021/9/15-2021/9/23 课题选择

2021/9/23-2021/10/10 项目计划制定

2021/10/10-2021/10/17 可行性分析报告制定

2021/10/18-2021/10/24 需求分析报告制定

2021/11/3-2021/11/7 系统设计说明报告制定

2021/11/8-2021/11/13 软件测试计划制定

2021/11/8-2021/11/13 用户手册制定

2021/11/8-2021/11/14 详细计划制定

相关文档：

SE2021-G005-项目想法1.0.docx

SE2021-G005-项目计划表0.1.docx

SE2021-G005-可行性分析报告0.1.docx

SE2021-G005-需求分析报告1.0.docx

SE2021-G005-系统设计说明报告1.0.docx

SE2021-G005-用户手册1.0.docx

投资方、开发方：

G005小组

支持机构：

浙大城市学院

## 1.3文档概述

软件项目的测试计划是描述测试目的、范围、[方法](https://baike.baidu.com/item/%E6%96%B9%E6%B3%95/2444" \t "_blank)和[软件测试](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95/327953" \t "_blank)的重点等的文档。对于验证[软件产品](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E4%BA%A7%E5%93%81/6800028" \t "_blank)的可接受程度编写测试计划文档是一种有用的方式。

本系统设计说明用于指导开发“城院生态圈”小程序项目顺利进行并最终通过评审的项目产品。本系统设计说明面向项目组全体成员。

## 1.4与其他计划的关系

是系统设计下的用于计划软件设计过程。

## 1.5基线

基线是软件生存期各个开发阶段的工作成果。

重要的检查点是里程碑，重要的需要客户确认的里程碑是基线。

主要基线：

SE2021-G005-项目想法1.0.docx

SE2021-G005-项目计划表0.1.docx

SE2021-G005-可行性分析报告1.1.docx

SE2021-G005-软件需求说明(SRS)1.1.docx

SE2021-G005-系统设计说明1.0.docx

# 2引用文件

本章应列出本文档引用的所有文档的编号、标题、修订版本和发行日期，也应标识不能通过正常的供货渠道获得的所有文档的来源。

[1]GB+T-8567-2006计算机软件文档编制规范 09 - 系统(子系统)设计(结构设计)说明(SSDD)

[2]SE2021-G005-项目想法1.0.docx

[3]SE2021-G005-项目计划表0.1.docx

[4]SE2021-G005-可行性分析报告0.1.docx

[5]SE2021-G005-需求分析报告1.0.docx

[6]SE2021-G005-系统设计说明报告1.0.docx

[7]SE2021-G005-用户手册1.0.docx

[8]张海藩，牟永敏.软件工程导论(第6版)[M]北京：清华大学出版社

[9]梁勇. JAVA语言程序设计(第10版)[M]北京：机械工业出版社

[10]王珊，萨师煊. 数据库系统概论(第5版)[M]北京：高等教育出版社

[11]覃征，李旭，王卫红. 软件体系结构(第4版)[M]北京：清华大学出版社

# 3代码走查规范报告

**参考评审表一：**

|  |  |  |
| --- | --- | --- |
| 评审对象 | 论坛模块 | 评审意见 |
| 评审人 | 李东泽 |
| 被评人 | 梁晓勇、黄依豪、李东泽 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 基本通过 |
| 4 | 结构设计能够满足机能变更 | 是“机能变更”还是“功能变更” |
| 5 | 整个代码体系结构组合合理 | 通过 |
| 6 | 所有主要的数据构造描述清楚，合理 | 通过 |
| 7 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 8 | 所有的接口模块化，因此修改时不影响其他代码模块 | 通过 |
| 9 | 代码体系构架对空间和速度都已经进行考虑 | 基本通过 |
| 10 | 提供了处理数据的策略 | 通过 |
| 11 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 通过 |
| 12 | 一个业务如果进行多次数据库更新、添加、删除是否正确添加事务。 | 采用No-SQL，无防止误操作回滚等设置 |
| 13 | 进行逻辑与、逻辑或判断时是否使用短路与、短路或。 | 通过 |
| 14 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 15 | 对象是否使用工厂获取。 | 对象定义的数据库模型、不需要使用工厂模式 |
| 16 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 通过 |
| 17 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 18 | 检查代码和详细设计是否一致。 | 基本通过 |
| 19 | 检查类中是否有无效的代码或者是无用的代码。 | 有少数说明例子代码段，无用代码存在，但不超过5% |
| 20 | 不要使用System.out.print()输出，需要进行日志处理 | 有部分使用 |
| 21 | 类的生成是否都有工厂模式而不是直接调用。 | 通过 |
| 22 | 所有的文件名符合文件命名规范，见名知意 | 通过、小组成员看的清楚、了解 |
| 23 | 文件和模块分组清晰 | 通过 |
| 24 | 所有的代码行在80字符以内 | 通过 |
| 25 | 每个程序文件都小于2000行 | 通过 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 不通过，查询语句基本都使用 |
| 2 | 对需要处理的字符串定义为StringBuffer ，常量定义成静态的。 | 通过 |
| 3 | 所有的变量名都小于32字符 | 通过，小部分不符合 |
| 4 | 每行代码都小于80字符 | 通过 |
| 5 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 6 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 7 | 代码排版是否规范 | 通过 |
| 8 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 9 | 复杂的表达式具备可读性 | 通过 |
| 10 | 续行缩进 | 通过 |
| 11 | 括号在合适的位置 | 通过 |
| 12 | 每个顺序的小块用空行隔开 | 通过 |
| 13 | 注释和代码对齐或接续在代码之后 | 通过 |
| 14 | 显示的文本无拼写和语法错误 | 通过 |
| 15 | 代码与操作符的优先级无关 | 通过 |
| 16 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 通过 |
| 6 | 函数和其他代码松耦合 | 基本通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法说明清楚 | 通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 基本通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 递归定义了出口 | 通过 |
| 21 | 递归局限于一个函数 | 通过 |
| 22 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 通过 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 每一个变量都初始化了 | 通过 |
| 5 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 通过 |
| 6 | 所有的变量都有最小的活动范围 | 通过 |
| 7 | 所有的全局变量都描述清楚 | 通过 |
| 8 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 有些冗余的判断、设计的时候未设计好 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 通过 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 未删除注释、用于组内的维护 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 基本通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

**参考评审表二：**

|  |  |  |
| --- | --- | --- |
| 评审对象 | 动物城友会模块 | 评审意见 |
| 评审人 | 梁晓勇 |
| 被评人 | 黄依豪 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 基本通过 |
| 4 | 结构设计能够满足机能变更 | 是“机能变更”还是“功能变更” |
| 5 | 整个代码体系结构组合合理 | 通过 |
| 6 | 所有主要的数据构造描述清楚，合理 | 基本通过 |
| 7 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 8 | 所有的接口模块化，因此修改时不影响其他代码模块 | 通过 |
| 9 | 代码体系构架对空间和速度都已经进行考虑 | 基本通过 |
| 10 | 提供了处理数据的策略 | 通过 |
| 11 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 有些未达标，为能在所有的对数据库的操作中写上关闭异常 |
| 12 | 一个业务如果进行多次数据库更新、添加、删除是否正确添加事务。 | 采用No-SQL，无防止误操作回滚等设置 |
| 13 | 进行逻辑与、逻辑或判断时是否使用短路与、短路或。 | 通过 |
| 14 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 15 | 对象是否使用工厂获取。 | 对象定义的数据库模型、不需要使用工厂模式 |
| 16 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 通过 |
| 17 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 18 | 检查代码和详细设计是否一致。 | 基本通过，但是还有些许缺陷 |
| 19 | 检查类中是否有无效的代码或者是无用的代码。 | 有少数说明例子代码段 |
| 20 | 不要使用System.out.print()输出，需要进行日志处理 | 也不能使用System.err |
| 21 | 类的生成是否都有工厂模式而不是直接调用。 | 通过 |
| 22 | 所有的文件名符合文件命名规范，见名知意 | 通过、小组成员看的清楚、了解 |
| 23 | 文件和模块分组清晰 | 通过 |
| 24 | 所有的代码行在80字符以内 | 通过 |
| 25 | 每个程序文件都小于2000行 | 由于项目的框架原因，有部分文件超出 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 没有用SQL |
| 2 | 对需要处理的字符串定义为StringBuffer ，常量定义成静态的。 | 通过 |
| 3 | 所有的变量名都小于32字符 | 通过，小部分不符合 |
| 4 | 每行代码都小于80字符 | 通过 |
| 5 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 6 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 7 | 代码排版是否规范。 | 基本符合规范 |
| 8 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 9 | 复杂的表达式具备可读性 | 通过 |
| 10 | 续行缩进 | 通过 |
| 11 | 括号在合适的位置 | 通过 |
| 12 | 每个顺序的小块用空行隔开 | 通过 |
| 13 | 注释和代码对齐或接续在代码之后 | 通过 |
| 14 | 显示的文本无拼写和语法错误 | 通过 |
| 15 | 代码与操作符的优先级无关 | 通过 |
| 16 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 通过 |
| 6 | 函数和其他代码松耦合 | 基本通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 基本通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法说明清楚 | 基本通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 基本通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 递归定义了出口 | 通过 |
| 21 | 递归局限于一个函数 | 通过 |
| 22 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 基本符合要求 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 每一个变量都初始化了 | 通过 |
| 5 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 通过 |
| 6 | 所有的变量都有最小的活动范围 | 通过 |
| 7 | 所有的全局变量都描述清楚 | 通过 |
| 8 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 有些冗余的判断、设计的时候未设计好 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 少数部分考虑不周 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 未删除注释、用于组内的维护 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 基本通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

**参考评审表三：**

|  |  |  |
| --- | --- | --- |
| 评审对象 | 动植物科普界面 | 评审意见 |
| 评审人 | 黄依豪 |
| 被评人 | 李东泽 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 基本通过 |
| 4 | 结构设计能够满足机能变更 | 是“机能变更”还是“功能变更” |
| 5 | 整个代码体系结构组合合理 | 通过 |
| 6 | 所有主要的数据构造描述清楚，合理 | 基本通过 |
| 7 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 8 | 所有的接口模块化，因此修改时不影响其他代码模块 | 通过 |
| 9 | 代码体系构架对空间和速度都已经进行考虑 | 基本通过 |
| 10 | 提供了处理数据的策略 | 通过 |
| 11 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 有些未达标，为能在所有的对数据库的操作中写上关闭异常 |
| 12 | 一个业务如果进行多次数据库更新、添加、删除是否正确添加事务。 | 采用No-SQL，无防止误操作回滚等设置 |
| 13 | 进行逻辑与、逻辑或判断时是否使用短路与、短路或。 | 通过 |
| 14 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 15 | 对象是否使用工厂获取。 | 对象定义的数据库模型、不需要使用工厂模式 |
| 16 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 基本通过 |
| 17 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 18 | 检查代码和详细设计是否一致。 | 基本通过，但是还有部分缺陷 |
| 19 | 检查类中是否有无效的代码或者是无用的代码。 | 有以前写过的代码 |
| 20 | 不要使用System.out.print()输出，需要进行日志处理 | 也不能使用System.err |
| 21 | 类的生成是否都有工厂模式而不是直接调用。 | 通过 |
| 22 | 所有的文件名符合文件命名规范，见名知意 | 通过、小组成员看的清楚、了解 |
| 23 | 文件和模块分组清晰 | 通过 |
| 24 | 所有的代码行在80字符以内 | 通过 |
| 25 | 每个程序文件都小于2000行 | 由于项目的框架原因，有部分文件超出 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 没有用SQL |
| 2 | 对需要处理的字符串定义为StringBuffer ，常量定义成静态的。 | 通过 |
| 3 | 所有的变量名都小于32字符 | 通过，小部分不符合 |
| 4 | 每行代码都小于80字符 | 通过 |
| 5 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 6 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 7 | 代码排版是否规范。 | 基本符合规范 |
| 8 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 9 | 复杂的表达式具备可读性 | 通过 |
| 10 | 续行缩进 | 通过 |
| 11 | 括号在合适的位置 | 通过 |
| 12 | 每个顺序的小块用空行隔开 | 通过 |
| 13 | 注释和代码对齐或接续在代码之后 | 通过 |
| 14 | 显示的文本无拼写和语法错误 | 通过 |
| 15 | 代码与操作符的优先级无关 | 通过 |
| 16 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 基本通过 |
| 6 | 函数和其他代码松耦合 | 基本通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 基本通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法说明清楚 | 基本通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 基本通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 递归定义了出口 | 通过 |
| 21 | 递归局限于一个函数 | 通过 |
| 22 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 通过 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 每一个变量都初始化了 | 通过 |
| 5 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 基本通过 |
| 6 | 所有的变量都有最小的活动范围 | 通过 |
| 7 | 所有的全局变量都描述清楚 | 基本通过 |
| 8 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 有些冗余的判断、设计的时候未设计好 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 有部分还需修改 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 未删除注释、用于组内的维护 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 基本通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

# 4注解

Flutter： Google 开源的 UI 工具包，帮助开发者通过一套代码库高效构建多平台精美应用，支持移动、Web、桌面和嵌入式平台。

Django：一个由 Python 编写的一个开放源代码的 Web 应用框架。使用 Django，只要很少的代码，Python 的程序开发人员就可以轻松地完成一个正式网站所需要的大部分内容，并进一步开发出全功能的 Web 服务 Django 本身基于 MVC 模型，即 Model（模型）+ View（视图）+ Controller（控制器）设计模式，MVC 模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。

# 附录

**参考资料：**

[1] Flutter [EB/OL].https://flutterchina.club/ -2020/11/04

[2] Django 简介 [EB/OL]. https://www.runoob.com/django/django-intro -2020/12/05

[3]代码走查规范表[EB/OL]. https://blog.csdn.net/honsty2010/article/details/10987515 -2020/12/25