质量属性需求

1. 可重用性

项目中许多功能需要被重复使用，譬如通过网络交换数据信息，读取和修改配置文件信息，控制音乐播放的各种操作。将这些功能集成封装在服务层的不同的类中，需要这些功能时引入某服务的依赖即可。在前端方面，设计出可重用的独立组件来展示歌曲信息，重用相同的组件和功能，但可以展示不同的歌曲信息。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 开发人员 |
| 刺激 | 使用系统中的歌曲播放模块 |
| 制品 | 代码 |
| 环境 | 设计时 |
| 响应 | 通过调用封装好的音乐播放服务实现对音乐播放的控制 |
| 响应度量 | 无需修改在系统各部分都可直接控制音乐的播放 |

1. 可维护性

项目开发维护过程中可能会遇到需要修改原有设计的情况，可维护性要求系统易于修改。在项目设计时使用开闭原则等设计方法来减少耦合，尽量避免代码修改。比如音乐播放器在设计网络服务类时先设计接口，然后用不同的实现类来拓展不同的功能。再比如音乐播放器的界面设计时可能会经常改动，但从界面中抽象出的数据和对数据的操作不变。于是就可以使用MVVM（Model-View-ViewModel）模式，将数据和对数据的操作写在ViewModel层中，界面写在View层中。这样使得界面的数据行为和具体样式解耦，对界面的修改不会导致业务逻辑代码的改变。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 开发人员 |
| 刺激 | 音乐播放器界面设计改变 |
| 制品 | 代码 |
| 环境 | 开发时 |
| 响应 | 界面的更改不影响逻辑控制代码 |
| 响应度量 | 开发人员修改的代码行数 |

1. 功能正确性

最终开发出的软件必须符合需求中要求的功能，在音乐播放器中就是使得音乐的播放控制、播放列表创建和管理、网络下载音乐和登录注册等功能被有效的实现。这是最重要和基础的质量属性。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 开发人员 |
| 刺激 | 开发音乐播放模块 |
| 制品 | 代码 |
| 环境 | 开发 |
| 响应 | 参照文档要求实现对应播放控制功能 |
| 响应度量 | 功能需求实现的比例 |

1. 概念完整性

在软件正常运行时，如果开发人员将该软件进行跨平台重构，或者更换不同歌曲源后需要按原方式播放时，如果和原设计冲突将导致难以重构的情况。所以歌曲数据的保存和处理应当遵循相同的数据结构，用统一的设计来进行数据库设计、前端界面设计和业务逻辑设计。当不同的开发人员设计相同的功能时就有了统一的参照，也便于沟通协作。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 开发人员 |
| 刺激 | 新成员加入 |
| 制品 | 代码 |
| 环境 | 在开发时 |
| 响应 | 新成员遵循统一的歌曲数据结构和编码规范进行开发 |
| 响应度量 | 新成员完成开发任务的时间 |

1. 易用性

易用性包含易理解性、易学习性和易操作性。当今市场上，存在许多音乐播放器，但大部分界面冗杂。本系统具有更加便捷的使用方式，让用户拥有更加简洁的界面，提高了易用性。当出现网络错误时有提示信息，文件异常时有提示信息并且把有问题的歌曲文件从播放列表中移除。在各个界面也增加了相应的文字说明，具有详细完善的平台用户使用说明手册，都提高了产品的易用性。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 用户 |
| 刺激 | 播放被删除的歌曲 |
| 制品 | 系统 |
| 环境 | 运行时 |
| 响应 | 将被删除的歌曲从播放列表中删除并及时给用户弹出提示信息 |
| 响应度量 | 提示信息在1秒钟内弹出 |

1. 可定制性

用户使用音乐播放器时，界面可以按照用户的喜好进行个性化修改。如更改背景颜色，背景图片，切换日间和夜间模式等。

|  |  |
| --- | --- |
| 场景元素 | 示例 |
| 刺激源 | 用户 |
| 刺激 | 按照喜好修改界面样式 |
| 制品 | 系统 |
| 环境 | 运行时 |
| 响应 | 界面样式按照用户选择的选项改变 |
| 响应度量 | 用户可修改样式的数量 |