**迭代评估报告**

　　　　　　　　　　　　　　　　　　评估日期：2021-5-4

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 第十五组 | 项目名称 | 交我唱 |
| 迭代名称 | 技术原型迭代 | 实际起止日期 | 2021-3-29~2021-5-5 |
| 任务达成情况：（完成的任务、实现的功能、进度、质量等） 1前端实现账户登录用例、K歌用例和完整界面1.1完成的任务简述  1. 手机注册/登录页面与功能 2. 曲目搜索及推荐页面与功能 3. 搜索结果选择页面与功能 4. K歌页面 5. K歌结算界面  1.2实现的功能  |  |  |  | | --- | --- | --- | | 功能 | 进度 | 结果质量 | | 手机号登录 | 提供用户登录功能，从后端的用户列表中获取确认信息。在前端页面中展示登陆状态。提供选项切换至注册页面。 | 较好，简易登陆状态显示完成。 | | 手机号注册 | 提供用户注册功能，将注册请求发送至后端。提供选项切换至登陆页面。 | 较好，能够避免非法的请求与重复的注册。 | | 音乐搜索栏 | 提供搜索功能，将用户的搜索目标发送至后端并获取搜索结果，渲染结果列表以供选择。 | 较好，提供搜索前十曲目。 | | 音乐推荐栏 | 提供推荐目录供用户直接选择，无需搜索。 | 较好，且反应速度快。 | | 音乐搜索结果列表 | 将用户搜索的结果渲染为可供选择的列表，并导向k歌界面。 | 较好，界面简单直观。 | | 伴奏/原唱切换 | 在K歌界面提供伴奏与原唱的切换。 | 从后端下载经过处理的伴奏，存在一定延迟。切换过程无延迟，但暂不支持在歌曲中间切换，切换后需要从头开始。 | | UI风格 | 使用底部导航栏进行规划分页，使用了flexbox进行整体布局。美术风格为使用蓝色系纯色作为主题的简洁风格，图标等则使用统一的彩色底白色。 | 较好，页面整体简洁美观，易于操作。 | | 代码风格 | 使用了组件式开发，利于代码重复利用，降低冗余。 | 大体上采取统一的代码风格。React-Native没有定义统一的编程规范，这里我们采用前人经验总结出的约定规范，并在实际过程中得到初步实施。包括：   1. 路径、组件、变量的命名规范。 2. 使用flexbox统一布局、渲染和状态变更分离的页面布局规范。 3. 约定注释格式的注释规范。 |  2后端实现用例效果，接入数据库，并给出教程2.1完成的任务简述 1. 文件上传下载  2. 网易云API接入  3. 跟flask进行通信  4. 数据库连接和通信  5. 日志记录和备份 2.2实现的功能  |  |  |  | | --- | --- | --- | | 功能 | 进度 | 结果质量 | | 上传文件和下载文件 | 可以通过http请求从服务器下载文件到本地和上传文件到服务器 | 较好，但受限于服务器带宽，传输速度一般 | | 搜索歌曲 | 可以用网易云API通过关键字搜索来获取相关的歌曲列表 | 较好 | | 获取歌曲 | 可以用网易云API或者歌曲的详细信息，包括音频文件，图片，歌手，歌词 | 较好 | | 后端通信 | 可以通过http请求，将客户发送的部分请求转发给另一flask架构后端，并且繁忙时返回等待提示 | 质量尚可，但还需要更完善的异常处理 | | 用户登录注册 | 可以通过对服务器上mysql数据库的访问，实现用户的登录和注册，并返回相应成功或出错信息 | 较好 | | 日志记录和备份 | Spring后端除调试时间外，一直保持在服务器后台运行，并进行了日志记录，并且对每6个小时内的日志进行单独备份 | 较好 |  2.2复现教程 各个接口的访问：给出了spring各个可供外部访问接口的介绍，具体用途，传输方法和请求参数，以及返回的数据类型和内容。  服务器上脚本：写了脚本能让spring重启，还有进行日志的输出后清空，使得所有组员都能快捷地进行重启spring和整理查看日志。  服务器上文件结构：给出了服务器上各个文件夹其中内容的用途以及文件命名标准，以及各个请求对应操作的文件夹。 3音频编码学习及其对应教程3.1完成的任务和进度  |  |  | | --- | --- | | 任务 | 进度 | | 进阶音频知识的学习，包括编码解码的原理、混音器、均衡器、效果器的原理等 | 已完成。 | | 学习探究ffmpeg组件的使用，针对我们的项目进行api封装 | 已完成 | | 探究人声增强算法，实现并测试 | 已完成，采用RNNoise神经网络实现。 | | 探究回声消除算法，实现并测试 | 基本完成，但是因为一些缺陷搁置。 | | 混响器、效果器的实现、封装与测试， | 已完成，采用Sox库实现。 | | 录音器的实现。 | 已完成，其中Java层负责获取数据流，react native层控制业务逻辑和缓存，实现了录音按句子划分、重录和重录上一句等功能。 | | 将原始音频数据写为wav格式文件 | 已完成 | | 将人声增强算法集成到录音的数据流中。 | 已完成 | | 实现一个简单的加载动画 | 已完成，加载动画位于一个透明图层覆盖在当前图层上，有效避免了加载时用户乱点出错的问题。 | | 唱歌界面的改进优化。 | 已完成，在唱歌界面进行了对录音器的控制和维护，梳理了业务逻辑，使得界面组件的选取更合理 | | 整合ffmpeg组件和sox组件，跑通从获取伴奏到获得成品整个流程。 | 已完成。已经修复了大量的bug，性能已经尽可能做好但仍有提升空间。 | | 结算界面的实现。 | 本来计划给用户自定义调整混音与效果器的界面，但是尚未完成，现在只是播放默认效果的成品。 |  3.2质量评价 对于音频算法的探究质量基本符合预期，只有一个算法存在缺陷且没有什么好的解决办法。  录音器组件工作正常，几乎没有遇到bug。  采用了异步的方式提升了并行型，提高了性能，并且用promise特性保证了顺序执行。更多的效果器预设或可自定义效果器暂时还没有做，有待以后完成。 4原唱消声算法和歌曲打分算法4.1完成的任务简述  1. 实现原唱消除算法的适配器。 2. 实现歌曲打分算法的适配器。 3. 使用Flask框架将两个算法封装为服务。 4. Flask和Spring实现无缝衔接，Spring能够正确请求，Flask能够正确响应。  4.2实现的功能  |  |  |  | | --- | --- | --- | | 任务 | 进度 | 结果质量 | | 实现原唱消除算法的适配器 | 现在可以通过请求来获取到原唱对应的伴奏和人声部分，效果较好，仅受到服务器性能影响。针对这个问题我们采取了服务器缓存功能，进行优化。 | 完成，性能目前较低，只支持单进程。 | | 实现歌曲打分算法的适配器 | 现在可以通过请求获取到对应的分数，仅受到服务器性能影响。 | 完成，性能目前较低，只支持2并发。 | | 使用Flask框架将两个算法封装为服务 | 已完成。 | 效果较好。 | | Flask和Spring实现无缝衔接，Spring能够正确请求，Flask能够正确响应 | 已完成。 | 在快速请求/多用户请求过程中，依旧会有少量问题。 |  5测试5.1完成的任务简述  1. 了解android sdk工作原理，了解与安卓交互的原理，学会用adb操作设备。 2. 安卓真机自动化测试。使用appium-selenium框架来撰写脚本。 3. 安卓真机单元测试。使用pytest-unittest框架来撰写单元测试脚本。 4. 测试报告生成。学会用allure-html生成测试报告。  5.2实现的功能  |  |  | | --- | --- | | 任务 | 进度 | | 了解android sdk工作原理，了解与安卓交互的原理，学会用adb操作设备。 | 成功用adb与安卓真机或模拟器获得链接并操作。 | | 安卓真机自动化测试。使用appium-selenium框架来撰写脚本。 | 成功使用appium自动化打开kgeapp并打开各个UI界面。 | | 安卓真机单元测试。使用pytest-unittest框架来撰写单元测试脚本。 | 看完了pytest-unittest网课，学会了基本的单元测试方法，准备将其与appium结合来测试各个组件。 | | 测试报告生成。学会用allure-html生成测试报告。 | 学会了allure-html，但还未付诸实践，待单元脚本写完便可实现。 | | | | |
| 评审/测试的结果：（执行了哪些评审和测试？评审和测试的结果如何？） 1评审1.1内部评审 在软件开发的过程中，我们持续在项目组内定期召开内部评审会，对前一段时期的软件本身及其衍生文档进行评审。 1.1.1技术方案评审 在每周的例行组会中，我们会召开技术方案评审。针对本迭代的最终目标和当前的问题，评审当前的方案。主要针对方案主要能够解决的问题、可落地性、实现代价、是否要引入新组件、扩展性等方面进行内部评审，最终对提议的方案给出一个反馈意见。 1.1.2需求评审 在软件前景文档和软件需求规约每一个版本完成后，本项目组在微信群中立即召开非正式需求评审，要求项目组每一个成员检查文档中的每一个需求，每一行文字，每一张图标，并且从正确性、清晰性、无二义性、一致性、必要性、完整性、可实现性、可验证性、可测性等指标形成反馈意见。 1.1.3代码走查 代码走查可以提高软件的质量，以及可维护性。这样就可以减少查找错误的时间，提高解决bug的效率，提高开发效率的同时降低后期的维护成本。  本APP在本轮迭代中分为前端组、后端组、算法组进行分别开发。组内成员共同进行一个或多个功能或模块的开发。在合作开发的过程中，每个项目组成员自动充当代码走查的评审员，项目组成员需要保证其他人写的代码是可读易懂的。如果出现了难以理解的混乱的代码，需要及时给出评审意见，将意见反馈给那位写出晦涩代码的成员。 1.2同行评审1.2.1正规检视 我们充分利空课后时间，邀请沈备军老师对我们交我唱APP的迭代计划、Vision文档、用例模型等进行同行评议，给予我们一些改进的建议，同样地，助教老师对我们每周提交的软件相关文档进行正规检视后，会对软件的文档给出一个同行评审建议。基于此，我们再召开内部小组会对反馈内容进行消化和吸收，应用到之后的工程中去。 2技术原型测试2.1测试目的 测试前端技术原型的典型用例是否正常运转、布局和UI是否合理美观、测试界面的用户友好程度、测试已实现的前端交互产生的结果是否符合业务逻辑等。 2.2测试结果  1. 搜索功能测试。   对于冷门歌曲的搜索准确率不高。  对于符号或字母的搜索误差较大，有时会出现只有一首歌关联，其余完全不相关的可能性。  对于单个空格或emoji，不支持搜索。对于空格在单词之间，也不支持。此时需将空格切换成下划线。例如，stay with me，搜索失败；stay\_with\_me， 搜索成功。  搜索到的歌曲较少，只有一面，没有翻页的功能，大大减少了准确性与多样性。  搜索到的歌，点击k歌后，会发给后端进行伴奏请求，此时响应时间较长，且不能进行其他操作，只能强退重新开始，并等待后端响应，相对不方便。推荐加个等待按钮，在等待时能做其他事，例如浏览界面。   1. 主要K歌功能各项测试：   （1）：上一句按钮。  可以完美做到定位到上一句并重新录音，并将打分重置。  但只能连续点十次，也就是最多重定位十次，多了便不再有反应。分析原因是重定位需要算力，服务器压力过大。  （2）：重唱按钮。  重唱按钮完好，且功能完善。  （3）：原唱伴唱按钮。  响应时间短，可以精准在原唱和伴奏之间定位并切换。效果很好，且重复点击不会出现问题。  （4）：完成按钮以及后续操作：  点击完成按钮后立刻进入结算界面。结算界面有平均得分，生成作品，保存等按钮。目前平均得分有bug，有时会显示NAN。生成作品与保存按钮目前无用。还未设置本地缓存。  （5）：自动打分算法。  实现了分句，并按句打分。每句话的得分显示在UI界面。但有时相邻的两句话切换时，打分并不会更新，出现问题。打分结果大致准确。经过多次测试与人工评判，分数与人为评分单调性一致。但分数方差太大，大则30亿，少则几百万。数字过多导致界面拥挤，观察不便。  （6）：录音功能与噪声消除。  当伴奏较响时，录音功能会大打折扣。音质会变差，并且音节可能会流失。原因极为可能是噪声消除算法将人声也一并消除。由此可见，噪声消除算法需要进一步改良，避免这种情况。并且，结算界面展示的录音时间与伴奏时间出现一些错位。录音时间慢伴奏时间一拍，约0.5s，需要作细微的调整使其对齐。  以上的噪声音质问题均为扬声器麦克风录制时出现的。若使用耳机，则效果会非常好。音质好，音节流失会很少。但录音与伴奏相差时间依旧存在。需要加入偏移量。  （7）：伴奏消声算法。  效果极佳，基本能做到消除95%的人声及痕迹，很令人满意。但唯一的问题在于，服务器的内存只有2G，算力不足，因此无法使用参数最多的高配伴奏消声，只能将参数减半，导致目前可以在服务器运行的伴奏没有那么清晰，一些人声依然存在。  （8）：效果器与混音器。  在使用效果器与混音器之前，声音非常单薄无力。使用效果器混音器后，音色混响明显有所好转。并且可以自己调参数来改变效果器的效果，实现各种奇妙的音色与曲风。  （9）：暂停键。  可以正常使用，效果较好。 | | | |
| 问题、变更和返工：（遇到的问题、发生的变更、是否需要返工等） 1.前端 **问题1：**异步通讯带来的加载完成时间不确定问题，可能会导致应用的崩溃。  **变更：**在使用异步返回的变量之前，对其内容进行判断，若其值没有被更新，则保持等待而不是直接使用。  **返工：**少量返工，并额外增加的警告与提醒措施进一步防止异常发生。  **问题2：**使用父子组件作为主要传递参数的方法大大增加了组件代码的复杂度与编程的困难程度。  **变更：**引入全局变量global的使用，代替父子组件的复杂嵌套，使得在多数情况下能够简单传参同时保证数据的安全性。  **返工：**部分返工，修改了数据的保存方法以及引用方法。 Spring后端 **问题1：**后端实际场景下，要处理较多的并发请求，原本写了demo代码的php语言对于并发的支持没有java好，性能也差一些。而且考虑到，其他课程用的是spring架构，改用spring技术栈能复用得多一些，组员要新入手参与后端开发也更容易一些。  **变更：**改为用spring进行后端的开发，参照原来demo框架的思路，先把上一个迭代实现了的功能都转由spring来实现，在此基础上，再进行后续的开发。  **返工：**需要返工，本地部署了spring开发环境，服务器上部署了java运行环境，对之前demo里的功能重新进行了实现。  **问题2：**一开始通过文件流进行文件的上传下载一直出错没能解决，就考虑通过base64编解码的方法，通过传字符串来实现文件传输。  **变更：**后端实现了base64的编解码，但是实际使用必然会占据一定的计算资源，而且会徒增用户的等待时间，并非长久之计，后来还是解决了文件直接传输的问题，base64的编解码就暂时用不上，后续如果需要加密，可以考虑使用。  **返工：**需要返工，编码成字符串只是一个临时替代的方案，后来还是改为直接传输。 3.音频组 **问题1**：录音器的开启和关闭有延迟，会导致录音与伴奏错位的现象。  **变更**：大致估测了延迟量，并且在业务逻辑中将延迟量考虑在内。改变了原本录音器的开关逻辑，现在录音器在整个唱歌界面中保持开启，屏幕中的开始暂停按钮只是控制是否获取数据流，以此来降低延时。  **问题2**：早期由于不了解录音的流程，封装的api并不符合实际问题的需求。并且由于采用的是c语言到java语言再到js语言三层的封装，几乎不知道底层发生了什么事情。  **返工**：重新设计api，加入了大量的异步并行操作以提高性能，加入了异常检测和处理机制来确保组件正常工作，通过函数返回值的方式将c层的信息传递到java层，并通过Promise的方式将java层的信息传递到react native层。  **问题3：**将回声消除组件实现了后才发现现有的算法都是针对语音通话的，音频采样率限制的很低，难以满足k歌用户的需求。  **变更**：本来打算自己训练一个算法，但是这个算法难度较高，且对与越大的采样率越难以训练。所以暂时只能使用佩戴耳机录音的方式，再寻找更好的解决方式。 | | | |
| 经验和教训： 前端 1.数据使用时的安全性和可靠性在前端需要确切的保障，因此在开展编程之前应对数据的储存和使用方式开展完整的规划。  2.前后端通讯接口关键在于两边的一致性，在选取方法后前后端积极沟通能够有效地提升端口方法的开发效率。 2.Spring后端 1. 在调通信时，前后端同学要保持良好的沟通，协调好传输方式、传输格式、因调试临时下线等信息，避免因为信息滞后出现错误，然后浪费大量排查错误的时间。  2. 封装一些脚本能有利于自己快速执行一套指令，也能为组员提供一个方便的接口，不需要搞懂流程也能快速实现功能。  3. 异常处理非常重要，有利于整个软件稳定长久的运行，出现错误后也能更快地找到问题原因进行修正。  4. 对于已经调试通过的代码，要妥善备份保存，使得后续更改后出现难以解决的错误，也能版本回退，不至于很长一段时间用不了或者推倒重写，跟别人对接的时候，也只需要进行一次，后续都能进行成功地复用。 3.Flask算法后端 1.Flask虽然提供了默认的多进程功能，但是由于服务器的算力上限较低，限制了高算力需求的神经网络的前向传播算法，导致最终效果只能实现较低的并发量。我们尚未开发出前端算力的潜力，希望在第三次迭代中继续将算力前移来降低服务器的负载。 | | | |