**迭代评估报告**

　　　　　　　　　　　　　　　　　　评估日期：2021-3-22

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 第十五组 | 项目名称 | 交我唱APP |
| 迭代名称 | 界面原型迭代 | 实际起止日期 | 2021-2-26~2021-3-23 |
| 任务达成情况： 1静态前端代码1.1完成的任务简述  |  | | --- | | 1.手机注册界面。 | | 2.动态页面。 | | 3.发现页面。 | | 4.K歌页面。 | | 5.K歌结算界面。 | | 6.我的界面 | | 7.我的歌曲界面。 | | 8.我的信息界面。 |  5音频编码学习及其对应教程5.1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 | | 探究人生增强算法，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。 |  5.2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。 | | | |
| 问题、变更和返工：  声音对齐  回声消除 1前端的问题、变更和返工 **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。 | | | |