|  |  |
| --- | --- |
| Version: 1.1 | 2014.09.28 |

**Mobile Audio Conference 外包项目要求**

**F\_001 项目交付内容**

服务器端：

1. 完整的程序，包括 Audio Streaming源代码， 数据库的SQL文件以及其他configuration 的文件。

2. 文档：API的详细内容，每个函数的用途，客户端使用时需要提供的参数，返回数据的格式，发生错误时的错误码等。关键的参数给予解释，请给出你建议的数值大小。

3. 服务器端的架设：对服务器主机的要求，具体的架设步骤，比如端口，防火墙设置等。

客户端：

1. iOS跟Android平台上的与服务器通讯的API源代码。 我方自己的软件通常不会（少数例外）直接调用Streaming Server端的API，而是通过你编写的API来达到功能。提供iOS跟Android平台上能够支持你设计的的数据结构的麦克风，音频播放器的API的源代码。

2. 注释：API每个函数的用途，使用时需要提供的参数，返回数据的格式，发生错误时的错误码等。

3. 简单的程序，用于演示如何在iOS及Android平台上使用API。 在服务器架设好之后，该程序应该能够达到发言于收听的功能。

4. 用户界面：不要求外观，一个页面就可以。简单的“开始演讲”按钮，“开始收听”按钮等就好。

5. 代码里面要有足够的注释，以便我方后期进行整合。

6. 客户端的代码应该模块化，用户界面与实际实现Streaming/演讲/播放的class应该分开。

**F\_002 支持平台**

服务器端：

使用java语言，以C/S架构的形式来完成。现在我方的程序使用的 Xampp。 服务器API使用的 servlet。推送功能使用的php。应避免冲突常用的端口(8080, 80等)。或者端口由一个configuration文件来设置以便修改。

客户端：

能够容易的支持Android, iOS的程序。iOS > 6.0， 如果6.0缺少一些重要的API，也可以只支持7.0以上。 Android > 2.2。

**F\_003 第三方资源使用协议**

如果你的程序里使用到第三方的软件，代码，API等，应该使用Apache, MIT, BSD, GNU等可以免费用于商业用途的。所使用的第三方资源的协议应该是不限制我们对代码进行修改的。

**F\_004 程序的音频质量及效率**

服务器端：

我们暂定的是一个服务器500（此处可能会受限于服务的的好坏，你可以给出你估算出来的数据）个房间， 每个房间10个主要用户， 可以演讲及收听，同一时候只能一位用户说话。次要用户只能收听， 人数不限（理论上不限，不过如果服务器有什么限制，请给我们一个大概的上限值）。请你根据你的经验给出一台普通服务器能够支持的人数，房间数等。具体参数可以讨论，以便我们购置服务器。

客户端：

以常见的QQ, 微信， Skype为参照，在网速稳定的情况下，延迟不应该超过2秒。音频质量应该在普通说话的水平，不需要音乐质量。在保证其通话效果的情况下，尽量减少流量。你可以提供几个不同音质的文件以及其长度1分钟音频所占的流量给我们进行选择。

客户端API应该具有有效率的内存管理，不应出现memory leak, memory corruption, memory overflow等会导致程序crash的错误。

**F\_005 音频房间的创建**

我们对Streaming专用名词不熟悉，以下有些错的请给指出。暂且我们把每一个房间对应的数据流称作一个频道Channel。

简写：

SS：你所建立的Streaming服务器

AS：我们的Application 服务器

大概的流程如下：

1. 用户request到AS创建一个房间，得到一个房间ID

2. AS用房间ID request到SS创建一个频道并得到一个对应Channel ID

3. 通过这个Channel ID， 用户可以调用你在客户端提供的API向该频道发言或者收听该频道的信息。

(这里有一点改动：请参见F\_012)

**F\_006 服务器端功能**

为我们的服务器提供的API：（我们的服务器直接调用）

1. 你的Streaming 服务器需要提供创建频道的 API，参数为房间ID和boolean 值“是否保存音频”，返回Channel ID. (Channel ID 在数据库为BIGINT 数据， 设置为Auto Increment)

2. 通过Channel ID, 我们可以关闭/删除某个频道，可以改变“是否保存音频”

3. Streaming的功能基本上由你的客户端API内部调用，请根据你的客户端API来对应着设计。

**F\_007 客户端功能**

为我们客户端提供的API:(将由我们的Android/iOS程序调用，你的这些API内部需要跟你的服务器通讯)

演讲者：

1. 通过 Channel ID, 建立一个上传的stream， 返回一个UpStream的对象。

2. UpStream 对象调用start/init 函数，麦克风开启， 并开始传输音频。

3. Upstream对象调用stop函数， 关闭麦克风， 停止传输音频。

收听者：

1. 通过 Channel ID, 建立一个下载的stream， 返回一个DownStream的对象。

2. DownStream 对象调用start/init 函数，播放器开启，开始播放接收到的音频。

3. Downstream对象调用stop函数， 关闭播放器， 停止传输音频。

5. 用户可以中途加入收听。

麦克风跟音频播放器将嵌入在你的UpStream跟DownStream内。他们的Lifecycle将由UpStream跟DownStream去控制。

**F\_008 音频录制并保存**

用一个数据库表格来保存ChannelID以及对应的音频文件路径和录制时间。演讲将被保存在服务器。

**F\_009 客户端设置**

音量调控

DownStream应提供音量控制的API. 按百分比调节。不需要UI，只需要setVolumn(int volumn)的函数即可。

音质调控

如果codec支持的话，音质可分3档，音质越好，数据流量越大。音质在创建UpStream/DownStream的时候设定。收听者设置的音质差于演讲者的音质的时候就设定为演讲者的音质。

**F\_010 错误/异常的处理**

1. 每一个API功能如果涉及的有出错或者跑出异常的可能，都需要返回相应的Error Code.

2. 如果Channel已经被关闭，而用户还在上传或收听， 返回相应的Status Code。

3. 如果中途断网，Channel不必关闭，用户仍然可以重新创建相应的UpStream/DownStream继续发言或收听。

**F\_011 Configuration的管理**

服务器端：

你所使用的数据库url：端口， 数据库用户名， 密码放在统一的一处地方进行设置。

**F\_012 UpStream/DownStream 通道的管理**

**在建立一个通道的时候，除了ChannelId, 是否可以再同是返回跟这个Id对应的一个随机字符串,主要是为了避免他人通过其他不正当的方式连接到这个通道。用户连接到这个通道的时候需要提供ChannelId以及这个字符串。**

**你的服务器在储存通道资料的时候，也将对应的ChannelId跟随机字符串储存。用户需要这个字符串的时候必须是从我们的服务器读取（登陆以后，并且有一定的权限，这部分Validation将由我们这边的服务器提供，你的服务器只需要保证只有提交了正确的ChannelId以及随机字符串的用户可以连接）。**

**或者你根据你的服务器的设计提供一个其他的方案。**