管理文档

一. 项目计划及完成情况

时间	任务	人员	完成情况
3月16日~3月19日	设计模块，搭建服务端与客户端框架	罗毅	已完成
3月20日~3月22日	编写客户端界面	罗毅	已完成
3月23日~3月25日	重构服务端与客户端代码，提取出可复用构件	罗毅	已完成
3月27日~3月29日	写管理文档、复用文档	罗毅	已完成
3月28日~3月29日	撰写Client, Server的Javadoc	罗毅	已完成
3月20日~3月21日	修改消息范式，改用JSON发送消息	冯夏令	已完成
3月23日~3月26日	编写Utiils单元测试	冯夏令	已完成
3月27日~3月29日	编写系统功能性测试	冯夏令	进行中
3月19日~3月20日	编写数据库操作功能模块	廖山河	已完成
3月21日~3月25日	实现对消息的处理、客户端状态转换、计数等后台逻辑并处理bug	廖山河	已完成
3月28日	撰写DatabaseUtils，LogUtils, StringUtils的Javadoc	廖山河	已完成
3月19日	构建Client,Server端测试目录	彭程	已完成

3月20～3月24日	编写Client端单元测试	彭程	已完成
3月25日～3月27日	编写Server端单元测试	彭程	已完成
3月27日～3月29日	编写Client端和Server端功能测试	彭程	进行中
3月29号	完成管理文档	彭程	已完成
3月26日~3月27日	Cilent端消息队列缓存	范亮	未完成
3月28日	功能测试用例及文档	范亮	完成

二. 总结

这两周的开发过程中，我们小组在有限的时间内，通过组内成员的分工和协调，完全实现了应用程序的各项要求，总体说来进度和效率都保持得不错，最后项目也成功落地，算是比较出色的完成了第一次上机实践，但也存在以下两点问题：
1.组内分工并不是很明确；有时候组内成员之间不知道对方在写什么，这就导致了部分功能的重复编写，造成了时间上的浪费。而且由于预先没有把功能点细分开来，任务没有落实到每个人身上，导致这次开发过程稍显混乱，组内成员有时临时“接活”。组内分工不够明确，部分工作没有落地；
2.在使用git的过程中，由于没有确定统一的开发规范，导致分支稍显混乱,由于很多工作的展开需要一些前置工作，而每个人都在各自dev分支上进行独立开发，合并代码没有统一约定的规范，这就导致某些人的工作无法开展，导致了效率低下的问题。

解决方案：
1.Teambition的引用
Teambition作为一款提高团队工作效率，明确成员具体分工的应用，将被我们引入到接下来的开发工作中来，其灵活性和确定性将解决我们在这次开发过程中面临的问题。在我们接下来的作业中，我们会创建项目的工作看板，将任务分为Planning,Working on,Testing和Completed四个阶段：
在Planning看板中，我们会将该前期小组讨论的任务细分放在这里，这是一个明确的任务集合，即我们完成项目需要完成哪些具体的task，并且在线下的讨论中将这些task分配到每一位成员，明确每个人需要做的事情；
一旦组内成员开始开发工作，他可以将Planning中自己的任务移动到Working on看板中，告知其他成员自己正在发开，如果开发过程中遇到困难，开发者可以手动@其他人表示需要帮助，以及时通知自己的组员，及时获取帮助。同时任务一旦移入Working on看板，开发者最好设定deadline，以保证开发的进度；
当一个任务完成之后，组员将任务移动到Testing看板中，表示任务已经完成，等待别人测试，（在我们团队中，罗毅的代码能力是相对较强的，testing部分工作目前由他完成，他一旦确定该任务没有问题，就将该任务移动到Completed看板）,至此，一个任务的生命周期才算结束。

2.GitHub开发规范
我们决定在接下来的开发过程中规范我们github的分支操作，具体流程为：
1.在master分支下开一个dev分支，这是我们的开发分支，所有代码合并在此，形成一个相对稳定版本（即一个开发的流程告一段落）后才将dev分支merge到master分支；
2.在dev分支下，我们分别以dev_xxx作为命名规范，xxx表示功能点（比如开发客户端，可创建远程分支dev_Client），在这个分支上，每个人再来明确自己的task（此处宜与teambition上任务一一对应），基于此来创建自己的分支，比如teambition上指定罗毅开发Client的图形界面，他可以创建dev_Client_ui_luoyi（dev_x1_x2_x3）(x1:大功能点，x2:小功能点－－对应teambition上任务，x3:开发者)；一旦完成开发任务，就在github上提出一个pull request，告知测试者（目前是罗毅）来merge代码，一旦功能无误，代码合并，此时对应teambition上任务完成，我们删除对应task的分支（也可以在阶段任务完成后再删除，以防需要对代码进行修改），进行接下来的开发；一旦一个大的功能点完成（如Client开发完成），我们将其合并到dev分支上，表示这部分开发告一段落；

总结：接下来的开发我们将结合teambition和github的规范流程来进行开发，保证所有明确的任务都有人做，所有人都有事情做，达到提高开发效率，规范开发流程的目的。