请适当穿插相应交付制品中的内容。只提交一份心得体会，除了整个组的体会，最好有个人体会。

# 总体体会

兴高采烈 – 斗志昂扬 – 跌宕起伏 – 险象环生- 死不瞑目

# 需求分析兼测试人员 - 干悦

在跟客户的沟通中，我有很多的体会。

首先，客户刚开始提的要求都是不清不楚、不明不白的，比如“收学邮的新邮件”，这个时候我们需要耐心地提醒客户把需求说的更加具体，让他充分表达自己的意愿。否则在确定需求以后用户再来说自己之前没有说清楚、说完全就麻烦了。这个时候我们可以使用“您能具体说一下您的需求吗”、“什么什么您打算要什么样的呢”之类的语句来引导客户。

第二，当我们提醒客户要说的更加具体以后，客户可能又会开始东拉西扯。这样的话又会没有条理，很容易有重复或者遗漏，那么就会增加沟通的时间和次数，浪费很多不必要的时间。此时就需要我们来理出一条主线，让客户沿着这条主线往下讲自己的需求。如果是功能模块比较明显的就可以按照功能分块让客户讲。如果是功能不多但是流程清晰的，可以让客户按照用户会怎么样一步步使用这个软件来讲。比如在我们这个chrome学邮插件里，基本的流程就是登陆、查看新邮件列表、查看某一封邮件并处理。我们是从查看新邮件列表开始引导客户的，询问用户这个界面上需要显示什么、有什么功能。然后说到查看某一封邮件的界面。最后讨论用户没有提到的登陆，给出一个方案问用户是否可以。

第三，正确、科学地引导客户。引导客户是需要特别的技巧的，如果主动问客户是否需要什么功能，那么客户一般倾向于说是，这样就给自己增加了困难；如果特意避免提到某一项功能，又有可能被客户在以后提出，那么还是给自己增加了麻烦。但是被动地听客户提需求也是不对的，我们在之前提到了，客户是需要被引导的，否则是有很大可能不能很好地表达自己到底需要什么的。在实际的沟通中，我们向客户提到了是否需要声音提醒，幸好客户确实不需要，但是客户如果有这个需要也是很正常的，所以我们选择了提出这个问题。另外对于登陆这个问题，我们主动提出了一种比较简单的解决方案，也没有提出在登陆界面上可以有其他可选参数。用户接受了这个建议，所以还是比较成功的。

第四，这是我们当时没有做好的一点，后来想起，在和客户沟通完以后，我们实际上应该自己整理一下思路，当场和客户复述一遍，让客户听一下我们的理解是否一致。而我们的做法是沟通之后自己整理好需求给客户发邮件，这样会增加理解不一致后重新开会讨论的可能性，导致拖慢进度。

在测试过程中的体会如下。（我tm还没测试呢！）

# 设计兼集成人员-周予维

由于我们的项目是前台和后台两个完全独立的模块，不存在代码整合集成，所以只存在设计过程。

首先，在收到需求之后，我们对于Chrome插件进行了一定了解。根据Google给出的开发者教程，进行了简单的学习，对于一个插件所需要的开发平台、开发流程、开发模块都有了大致的了解。在了解过后并着手写了一个hello world级别的程序了解在开发过程中可能出现的最基本

的配置、编码等问题并解决。

其次，由于我们推出的时chrome浏览器学邮插件，而在chrome应用商店中，已经存在了google提供的Gmail插件和网易公司提供的网易邮箱插件。所以我们在安装了网易的邮箱插件后对这款插件进行学习，并且希望借鉴其设计、架构。而网易邮箱插件属于使用cookie调用网易邮箱的接口进行邮件操作，而我们的学号邮箱却没有提供相关的接口，这种想法直接搁浅。在这之后我们尝试通过后台直接解析学号邮箱页面，然而学号邮箱的页面结构并不利于做解析，并且每一份邮件都是由超链接链接到一个新的页面，因此如果真的能够解析该页面也会有很大的时间延迟，用户体验也肯定会很差，因此这种方法也被抛弃了。在进行直接学习、解析页面失败之后，我们尝试使用C/S结构。服务器端使用IMAP协议收取邮件并转码为插件端能够直接解析的json数据包；用SMTP发送邮件；使用cache维护缓存链接队列，使得在随后的通信中并不需要再次通过登陆操作，加快服务器的应答速度，改善用户的体验。在每次大致的设计过后，都应该进行一个十分仔细的必要性分析，可能一个完备的设计最后却因为一个技术难题不得不使得整个项目重构，而每次重构不仅仅要推翻之前大部分代码，开发团队一段时间内的思路也必须做出及时的改变，更不用说之前已经投入的精力与成本。所以可行性分析是必不可少的。

第三，我们有两个程序员，很自然地划分了前后台职能，由夏旭华负责前台、张文戎负责后台，中间通过Ajax进行通讯。所以我们定义了登陆、获取未读邮件数、获取未读邮件头、获取未读文件内容、标记已读、删除、发送邮件（未实现）接口。事实证明，最后的接口对接十分完备。

# 模块开发人员-夏旭华

我负责的是整个chrome插件部分开发。

在此之前，我从未接触过chrome插件的编写，通过了解，大概掌握了插件大概自己原先写过的静态页加js。由于采用C/S架构模式，插件端与服务器端完全分离，因而设计文档一出，我便很兴奋地开始实现我负责的那一部分。从界面部分html到css再到js控制页面生成，一切都似乎是很顺利的，直到编写完ajax通信部分，开始对ajax进行测试时感觉世界好像不是那么美好了。

对于chrome，ajax跨域请求是被默认拒绝的。然而我已经在插件许可列表中加入了被测试网站，应当可以获得跨域的数据，所以瞬间无所适从。大概花了一整天查阅了各种关于chrome插件开发的资料，才终于知道原来ajax跨域请求时必须由插件的后台页面来发送，而不是由前台的弹出页面。知道这些之后，我的重点就开始转移到了前后台页面之间如何通信方面。写了一段测试代码，发现前台页面成功地调用了后台方法，反悔了服务器端的测试数据。与服务器之间的通信问题终于得以解决。

随后的工作就是加上用户需求中的各项功能和为用户体验所做的优化部分。然而因需求中的回复邮件和转发邮件在实现中比较复杂，需要涉及到包括界面、编码等各种问题，在这两周的开发周期内感觉时间不够，于是将其暂时改成了直接打开邮箱首页，准备以后再进行二次开发。

在本次lab中主要的收获是自己的代码审阅和模块化很重要。因为很难写出没有bug的代码，所以在自己代码的单元测试之后肯定会有debug的阶段。而如果自己的代码模块化程度很高的话，那么每一次单元测试后如果出现bug，就只需对少量的代码进行阅读，debug的效率也会高许多，可以在这短短的开发周期中实现更多的功能。

# 模块开发人员-张文戎

本次开发我负责服务器端功能的开发

一开始我们以为可以通过js代码直接进行邮箱操作的，结果在研究之后发现必须要有服务器端才能实现，考虑到松耦合性的设计，我们采取了经典的C/S构架模式。也是因为C/S构架的缘故，我们组很容易的就完成了模块接口的定义，而且很成功的，接口到成品出现都没有进行过修改。

开发过程中服务器配置在学院机房里，而不是我的电脑上，这样在开发阶段、尤其是后期调试阶段，能够迅速服务器反馈代码的迭代结果，事实上在debug阶段也确实如此。

在服务端，一开始采用pop3去收取邮件，但是随后就发现了许多问题，操作邮箱的功能不足，流处理极为复杂，几乎无法避免各种bug，后来转用imap协议（php有完善的imap处理库）后迅速解决了各种问题，随后SMTP协议也用一个现成的开源项目解决了。第三方解决方案万岁！~

在服务器完成之后，配合客户端进行接口配对之后，我开始进行模块测试和debug。因为是C/S构架并且接口定义明确，使得测试过程变得方便，因而也能在有限的时间内进行更全面的测试。在模块测试阶段，我先代码审查检查完整性，发现2个问题，之后用等价类划分法进行黑盒测试，发现7个问题，轻松解决。至于性能测试，因为网络问题较大，完成度不高。

总体感想：在整个开发过程中，消耗了最多周期的原因在于设计方案的反复推翻，很大程度上原因归咎于技术可以行调研不足造成，也可说是技术风险规避不足。最后还是靠第三方解决方案搞定这些问题，从这点来看，很多情况下，一个项目很可能将技术难点归于第三方解决才是最好的方案，并不是所有功能都要自主开发才行，这应该也是如今行业分工的原因所在。

接口的正确定义的确能有效降低代码直接的耦合度，分割工作进而增加效率、方便开发测试，这次在C/S架构下更进一步规范了接口定义，才深切体会到接口规范带来的实惠。

这个过程让我进一步意识到事前对项目设计方案研究的重要性，降低返工率、减少技术风险、进一步规范接口、增加效率，方便测试保证正确性……所以也有说法，项目花一半时间在设计上，也不是胡说啊……