大家好：

不好意思，由于之前表述不当，让大家以为最后一个test project很简单，一两个人就可以弄完。实际恰恰相反，作为给我们时间最长的一个project，这个却恰恰是最繁杂的一个，需要我们全组通力合作才能完成，内容如下：

1. 单元测试（邓，符，谭）

需使用simple test 框架，像Junit一样生成测试函数接口，然后我们人工手动填内部代码。填完之后，再使用Xdebug之类的代码覆盖率工具检测达到40%，黑盒白盒结合。

*代碼覆蓋率:*

*两种评判方式：line coverage/statement coverage------------*[*Xdebug*](http://xdebug.org)

*path coverage*

然后根据测试结果，写出对应文档（包括测试内容，测试代码，预期结果，实际结果，发现的BUG，截图等）

*另外本项目还需要代码覆盖验证脚本，代码测试脚本。*

*同时考虑到第一个***Integration Testing（**包括3个集成测试文件，使用2个stub，一个mock，运行脚本，运行说明**）**:是基于单元测试的，工作量不大，所以这部分放在一起顺便完成。

1. 系统测试

我看了下，挑出了最适合我们的三个情况：

一定要记得到wiki看一下自己那部分的详细要求

1. **安全测试- Security Testing:（两部分）（赖）**

A．人工review，提出问题，列出脆弱点清单

B.软件框架测试- automated security checking tool

由于涉及到数据库注入，稳定性等专业问题，所以主要使用专业的测试框架比较好。我综合情况，粗略觉得推荐Rats，Wapiti（黑盒测试），Nikto，Acunetix Web Vulnerability Scanner，四种，测试者在其中挑选自己认为好的，进行测试，写出审核的方面，测试的攻击类型，攻击方法。

同样需要文档，参照一.，另外还有自己根据特定测试类型要求增添的条目。

这里的要求可能不完整，一定要记得到wiki看一下自己那部分的详细要求

1. **界面功能测试- Functional/UI Testing（同样两部分）（何）**
2. 人工点击按钮，操作···，提出问题，不人性化的方面
3. 再使用Selenium之类的框架-**automated UI testing**

参照课件，Lecture 33UI Automation Testing。

把某个功能通跑一遍，需要三份不同情况，文档同上，另外包括录制的脚本，另外还有自己根据特定测试类型要求增添的条目。

这里的要求可能不完整，一定要记得到wiki看一下对应自己那部分的详细要求

1. **性能测试-Performance Testing:（廖）**

学习使用Xdebug 之类的性能分析框架或软件

需提交application运行之后的性能分析log记录，

并找出发现性能瓶颈及验证方法，找出可能的解决方案。

这里的要求可能不完整，一定要记得到wiki看一下对应自己那部分的详细要求

1. Bug追踪

根据未在测试中覆盖到的bug，（*“未”-个人理解应该是开发过程中提出，解决的问题，未实现的功能，应该是github上的issues之类的或者是其他类型的除了我们做的3三种系统测试之外另外方面的Bug，如****Reliability Testing***）

在**bug tracker**中列出发现的弱点，未实现的功能等问题

**BUG文档须包括**：描述，再现方式（即什么情况下出现bug，严重性，当前解决状态）

及处理状态："Fixed" or "Won't Fix"（写明原因）

*同时不能一次性全部提交（分批），而且发现bug要均衡（每个人都要用）*

四.最后提交形式：

提交在Github，包括相关测试脚本，

相应部分测试文档，测试工具，

系统/集成测试描述**brief (at least 1-2 page)，包括**选择的系统测试类型等

测试文档清单**Test\_Practices\_List.txt**

注意排版布局。