各位老师、同学，大家上午好~！我是吴桐，我答辩的论文题目是基于Flask框架的物资管理系统的设计与实现，我的导师是任桐炜老师。

今天我将按照这样的提纲来演讲。

首先介绍一下项目背景。物资是物料资源的简称，一般情况下公司会把工作中所消耗物品的各种生产材料作为物资来管理，常用Excel表格来记录，这是一种传统的管理方式，所有的跟踪、记录和管理是围绕Excel进行，这样的方式存在着一些弊端：比如物资状态的更新不能及时反馈给其他用户，不支持对物资进行批量操作，操作权限局限在文件级别，还有文件容易丢失，数据容易出错等等。

在我实习的上海EMC公司UFE部门内部存在着类似的问题，这是一个Firmware测试团队，他们需要管理的物资是指为了满足Firmware测试而购买或者申请使用EMC自己生产的各种存储服务器和各种零件等等，部门现有的管理方式是采用Excel表格管理，由管理员负责登记信息，发送最新的表格给其他用户。在实际工作中，发现了如前面陈述的各种问题，于是他们希望能够有一个系统来管理物资。因此所要设计与实现的是一个应用于公司内部的管理系统；是能够改善传统的Excel管理的信息系统；是能够对物资设备进行操作，能够导入导出Excel表格，进行在线消息通知和邮件通知，让物资的状态变化及时通知到相关人员的管理信息系统。

这个项目由我负责并且是我独立完成的设计与实现。

经过分析发现，系统需求主要有物资基础管理，用户管理，物资使用管理，包括了对物资的请求、归还、转让和报废等操作，导入导出Excel表格，消息通知和邮件通知。

此外，还提出了一些非功能需求，尤其是易用性，用户界面应该简洁、尽量在设计上遵循所见即所得的原则。

下面，先介绍一下项目的总体设计。按功能模块划分为：物资管理模块、用户管理模块、其他辅助管理模块，下面又细分了这些子模块。

系统采用了三层结构：表现层、逻辑层和数据层。表现层是项目的前端部分，利用了这些前端框架和技术构建了用户界面，表现层向下层的逻辑层发起调用，在项目中是通过表单请求或者Ajax请求来做到的。逻辑层负责响应表现层发送的请求，进行业务逻辑处理，并返回响应结果。主要包含了两块，路由控制逻辑和业务处理逻辑。数据层负责数据操作和维护。一方面提供给逻辑层数据访问的接口，一方面构建了数据模型，进一步映射到数据库。系统的数据库采用了Flask内置的Sqlite数据库，因为系统供内部使用，每天访问量不会很大，也没有高并发的要求，所以对数据库的要求不是很高。从数据模型映射到数据库，是通过Flask-SqlAlchemy插件提供的对象关系映射机制做到的。

从开发的角度来看，前端通过Ajax请求或者表单的方式向路由控制组件发送资源请求，控制组件将调用服务组件提供的业务逻辑接口，另一方面也会调用数据相关的组件提供的数据访问接口，做业务处理，最后将结果响应给前端。服务器相关组件向控制组件提供Flask框架的服务比如核心的路由服务。在Flask框架中，路由服务又是通过叫Werkzeug的开源库实现的，Werkzeug是一个WSGI（Web Service Gateway Interface）工具箱，除了路由服务，还提供[HTTP](http://baike.baidu.com/view/9472.htm)头解析和封装、request和[response对象](http://baike.baidu.com/subview/1145873/1145873.htm)、WSGI 1.0 规范100%兼容、支持[Unicode](http://baike.baidu.com/view/40801.htm)、URI、[会话管理](http://baike.baidu.com/subview/2879217/2879217.htm)等等。

这是一个服务器端做出响应的序列。用户界面触发事件，向服务器请求资源，服务器通过Flask框架提供的路由系统来匹配URL，做业务处理，再通过模板引擎来渲染出结果页面，返回给前端显示。

下面是一些数据库表的设计，有物资表，用户表和历史记录表。在用户表中，维护了一个方便查询该用户负责物资的特殊字段，是由SqlAlchemy提供的relationship类型，这个字段由SqlAlchemy自动维护，就是当物资表中增加了一条某Owner负责的物资记录时，用户表中的这个字段会自动增加对该物资的记录，因此不需要开发人员去维护。

接下来介绍详细模块的设计与实现。这一部分，我将按照后端，前端，以及三个关键功能的设计与实现来介绍。

首先是基于Flask框架的后端设计与实现。配置模块是配置了路径、上传和下载Excel表格文件的存放路径、会话类型以及数据库资源定位等等。

启动模块导入Flask实例，从gevent导入WSGIServer，启动服务器。设置为多线程的模式。为什么用gevent在后边的消息通知介绍中会讲到。

DAO模块设计了各个DAO类，提供了访问数据的接口，这些接口会被不同模块经常用到，因此设计实现为python的静态方法，省去了与它耦合的其他模块对各个DAO类实例的维护。

数据模型模块建立起与数据库表的一个映射。比如一个Inventory类映射了数据库中的物资表。

路由模块设计实现为一个app\_route文件，实现了所有的跳转控制。同时也是前台与后台交互的一个主要模块。这是app\_route文件的一个片段，处理了添加物资的业务请求，做出响应。在前端通过ajax请求来触发这个业务。

前端的设计实现采用了Jinja2模板引擎，利用了模板的继承特性提高代码的重用率

而且采用了Bootstrap框架，一方面节省了开发成本和时间，一方面也让系统呈现出了像这样的比较简洁、风格统一的界面

这是界面上的一个交互细节，遵循了所见即所得的原则。物资的列表，勾选某一物资，左上角的操作按钮才出现，勾选多个物资，按钮有可用或不可用状态变化。

下面简单介绍三个关键功能的设计和实现。首先是消息通知，实现机制类似于发布/订阅机制，由服务器发布消息给前端浏览器，由前端向服务器订阅资源，这个资源在服务器端是一个不断查询消息队列的处理方法，而一旦检测到有消息产生，就把消息按照协议打包，返回给前端，再由前端进行筛选，如果当前用户是这个消息的通知对象，就显示消息通知，否则不会显示。在服务端的发布方法中，虽然系统需求中对高并发并没有要求，但在设计时还是考虑到了并发，采用gevent提供的协程机制，每产生一条消息通知，就创建一个greenlet协程，当这个协程完成后，将会回到原来的主协程。

再介绍一下导入导出Excel，一个导入Excel的流程比较简单，主要的处理是通过服务模块的服务来逐行地解析Excel文件，每读完一行数据，创建一个物资实例，全部读完再批量添加到数据库。导出Excel是反向的过程，在这里就不讲了。

最后介绍一下邮件通知的设计和实现。主要是利用Python的标准库smtplib来完成，smtplib提供一个SMTP类，可以创建实例，通过实例的sendmail方法来实现发送邮件。这个功能点暂时实现了一个本地运行的版本，安装了一个叫做dsmtpd的简单SMTP邮件服务器来进行本地测试。

最后，总结和展望一下工作。物资管理系统主要对改善传统的Excel管理做了一个解决方案，提高了物资管理效率和时效性。在技术架构方面，展现了如何实践Flask框架+前端的方式来开发一个Web应用，同时实践了模板引擎和前端框架，起到了提高代码重用率和开发效率的效果，完成了简单的导入导出Excel表格、消息通知和邮件通知的工作。今后的拓展是加入统计和基于角色的控制访问功能。

以上，就是我今天的答辩内容，谢谢大家（各位老师还有同学们）！

问：这个项目做多久？

用了三个月的时间，不过在这个期间还同时做着一个自动化测试软件的重构工作，所以有效的时间应该不到3个月

问：为什么选用Flask框架？特色：

嗯，首先，因为部门的同事比较熟悉python脚本语言，所以框架就局限在python web框架上面了，同类有Django, tornado等等，最终选择Flask框架是因为它是轻型框架，也就是核心简单，拓展的功能通过插件来实现，没有Django框架那么多的需要配置的东西，这样比较灵活，对于像物资管理系统这样的并不是很大很复杂的系统来说，已经足够了，而且学起来也简单容易，Flask框架的文档很齐全，社区很活跃。正好同事也熟悉，方便他们以后维护。所以各种原因导致最后选择了Flask框架。

问：为什么看到功能模块划分这里有的模块没有设计与实现？其他，看起来不是真实应用的截图？

首先是这样的，初期项目设计时，是考虑到了今后的拓展在里面，为了完整性，所以在划分中体现出来，在后来的具体实现阶段，因为时间的关系，我就只能做到现在做的这么多，留下那些是拓展的模块交给同事去完善了。

嗯，一开始是本地运行的版本，因为时间关系，后边的部署工作就留给同事了。

问：你觉得项目有哪些难点？

对于我来说，学习成本还是比较高的，这是一点。包括Flask框架和它的一些插件，Bootstrap框架和前端技术，尤其刚开始并不太熟悉前端开发，自己用JavaScript去实现一些细节，很耗时间，花了不少功夫去学习，后来学习使用Bootstrap框架后就节省了不少前端开发的时间。再就是那些关键的功能，可能以前没有接触过，导入导出Excel，消息通知，邮件通知这几个，思考探索设计方案还是花了一些功夫。

问：物资记录达到多大规模？  
现有的差不多几十个Excel文件，文件里记录的物资大概有几百个这样的