大家好，我们的项目是一个课堂教学辅助系统。

我们项目起源于五律协同观这门课课上所使用的那套系统。在上课的时候我们可以通过微信完成各项课堂功能，最让我印象深刻是匿名地对问题进行讨论。所以，以这套系统为原型，我们的系统希望能够做到课堂签到、举手、讨论、共享文件、参与考试和信息推送的功能。

之后，我们对可能系统相关的质量属性和场景进行了分析。由于时间限制，今天我只会覆盖其中最重要的三点质量属性及其场景，他们是性能、可用性和安全性。性能，即系统需要快速响应用户操作，并且能够支持住短时间大量用户的操作，例如签到、举手、考试提交等。可用性方面，我们需要系统能够在出错时能够减少数据的损失，并在网络连接出现问题时能够自动尝试恢复。安全性方面，我们要求我们的数据不得被泄露或者非法修改。

我们使用了P2P和C/S两种架构进行了设计。在P2P的初步设计中，我们将系统分成了两种节点，服务器端和客户端。一般的P2P网络中每个节点都是同等的，但是如果有一台固定地址的服务器，用户想要加入连接时，只需要连接上服务器即可，不需要通过其他设备就能接入网络。而且，服务器程序由于功能比较简单，所以可以放在讲台上的电脑、甚至路由器上。在客户端，我们把每个功能都独立划分为一个模块，并且加入了网络、权限和数据持久化三个公用的模块。对于C/S，类似的，六个功能需求服务器端和客户端都有对应的模块进行处理，并且两端均有网络模块负责网络通信。新增了一个登录模块用于登录功能；在服务器端，还新增了数据库和权限模块。

整个系统的设计我们了12和11次迭代，我相信你们也不想详细听所有模块的全过程。所以，下面将介绍对之前提到的三个关键的质量属性，即性能、安全性和可用性，通过举例的方式，来说明我们是怎样进行设计的。

针对性能，我们拿P2P和C/S中都有的签到模块举例子。在P2P架构中，我们通过场景1这个架构驱动因素，发现了**密集网络请求**这个设计关注点。针对这个关注点，我们列出了三种可能的模式。其中，同步请求，其实就是没有特殊处理；集群和负载均衡明显不适用于P2P网络。最终，我们选择了消息队列模式，即使每个签到请求进行排队，模块依次处理队列中的请求。这样可以保证先到的请求先处理，保证响应时间，也能降低同时处理多个请求可能带来的锁的性能问题。而在C/S架构方面，设计关注点和P2P架构类似，但是由于当请求量太大的时候，服务器端性能的问题很多是在数据库的读写上，所以我们采用了对数据进行缓存的模式，即是将签到请求首先写入内存，等积累到一定量的时候，再批量写入数据库。最终，C/S架构下的签到模块如图所示。

对于可用性，最重要的模块，是负责网络通信的网络模块。在C/S架构中，我们发现了**对客户端透明**和**健康检测**两个子关注点。对客户端透明，即是说错误处理逻辑不应暴露给客户端。对于这个关注点，我们选择了代理处理故障的模式，因为网络模块本身就可以看作一个代理，实现起来比较简单。使用代理进行网络请求也方便以后可能的分布式改造：服务器的架构改变不会对客户端造成太大的影响。对于健康检测，在Ping-Echo和heartbeat中，我们选择了Ping-Echo，是因为心跳需要服务器往客户端发消息，这在很多协议（例如常见的HTTP和RPC）里是不容易实现的。同时，当客户端过多时，服务器端给很多客户端发消息，容易带来性能问题。最终，我们网络模块就设计好了。客户端以一定频率向服务器端发送ping，以查看网络情况。若在ping过程中、或者网络请求过程中出现故障，网络模块作为代理，会尝试重新执行，若多次重试失败，才通知用户网络已经断开。

在文件和数据库模块中，我们体现了对系统的安全性的重视。在C/S和P2P的文件模块中，我们都采用了加密文件的策略以保护数据安全。而在数据库模块，我们在确保数据安全的同时，将其他属性（例如性能、可用性等）进行综合考虑，设计出了这样的架构。分布式数据库增加吞吐量，同时也能在数据损坏时进行恢复；检查点和日志，加入外部的备份系统，实现了数据备份和恢复的功能；而独立的安全管理模块进一步确保数据安全性。

最终，我们设计出了如下的P2P架构，服务器端节点只负责客户端的注册和监控客户端的网络情况；而注册后的客户端之间通过网络模块直接交流，每个模块根据自己的特点有对应的架构设计。我们也实现了C/S架构。C/S架构的客户端除了文件模块需要与文件系统交互以外，其他的模块通过网络模块与服务器端进行通信。服务器端的每个模块也根据自己的功能和质量属性特点，进行了相应的设计。

比较P2P和CS架构可以发现，P2P架构由于中心服务器功能简单，所以服务器硬件要求低，维护成本比较低。P2P网络也有高可用性的优势，没有单点失效的问题。而C/S架构简单，业界开发和维护经验丰富，且对客户端资源消耗低，用户体验会高于P2P架构。同时，C/S架构也更加容易实现一些功能。看上去二者各有各的好处，但是我们仔细一想会发现，C/S架构中的硬件成本已经微不足道：几百块钱一年就能有一个很强的、足够支持数千人的服务器；高可用性也可以在需要的时候，使用分布式、微服务等架构将服务器端进行改造。而P2P有其无法解决的问题，例如对于全时可用的文件共享功能，没有第三方服务的P2P网络只能采用每个节点都保存所有文件的解决方法，这个解决方法严重浪费网络和硬件资源。所以根据项目的特点，我们选择了C/S架构。

最后，我想为大家播放我们系统的demo视频。我们严格按照CS的C&C模块进行了接口和模块设计，使用了Kotlin和JavaFX实现了Demo应用，并且实现了之前提到的Ping-Echo机制，这样进一步确认了我们的架构并不是水中花镜中月，而是真实的、可用的架构。谢谢!