开场白

大家好，下面由我来代表我们组的四位组员”xxx“进行汇报。

汇报分为三个部分”xxx“。

微服务架构设计方案

p4

微服务架构设计是一种将应用程序作为一组小型服务的集合来开发的架构风格，每个服务实现特定的业务功能，并且可以独立部署和扩展。这些服务是松散耦合的，并通过轻量级的通信机制进行交互。

选择该设计方案是因为它具有PPT中的这些优势，其中比较重要的有可拓展性，比如“放心购”系统中，如果团购服务的使用量激增，可以只扩展团购服务，而不需要扩展整个应用程序。还有技术多样性，例如，“放心购”系统中的商品服务可能使用适合处理大量读写操作的数据库，而订单服务可能使用另一种优化事务处理的数据库。

p5

经小组讨论，我们设计了11种服务，下面依次进行简单的介绍：

用户服务，负责处理用户注册、登录、信息维护等功能。

供应商服务，管理供应商信息，如注册、商品发布等，同时允许供应商通过API导入商品信息。

商品服务，管理商品的详细信息，提供商品的搜索和过滤功能。

团购服务，负责团购的创建、管理和生命周期跟踪。

订单服务，管理订单的创建、状态跟踪和支付处理，提供订单查询和订单历史记录。

评价服务，管理并展示用户的评价。

投诉服务，处理并跟踪用户的投诉和反馈。

团长服务，管理团长的注册、认证和信息维护，提供团长开团和团购管理的功能。

监管服务，提供数据统计和投诉管理功能，该服务可以访问和分析平台的核心业务数据。

运维服务，负责平台的运维管理，提供系统后台管理功能包括用户权限管理，角色管理等。

通知服务，负责发送系统通知，如订单状态更新、团购开始等。

p6-p7

这张图展示了我们设计的各个微服务之间的交互逻辑。可以看出，各个服务之间松散耦合，通过通信机制进行交互。

从团购管理服务开始，该服务上接团长管理服务下接订单管理服务，在面向用户的方面，通过用户管理服务实现用户的注册和认证，在面向商品的方面，通过供应商服务和商品服务对接订单管理服务。

红圈内展示了我们的系统的核心功能，此外系统还外接了诸如xxxx等功能。

p8-p10

下面介绍我们在具体实现中使用的设计模式。

在用户服务中，可以使用工厂模式来创建不同类型的用户对象，比如普通用户、团长以及供应商。

在订单服务中，可以使用策略模式来定义不同的支付策略。

在通知服务中，可以使用观察者模式来实现事件驱动的通知系统。

面向对象架构设计方案

p12

在开始设计具体内容之前，有必要先明确领域和对象的关系。”读PPT内容“。

实体对象：具有生命周期和唯一标识，通过唯一标识判断相等性，能够进行增删改查操作，能够记录状态。

值对象：用于描述，没有唯一标识，只能通过属性判断是否相等，通常是不可变对象，通过实体对象进行操作。

p13-p16

这张图展示了我们各个类之间的交互。首先来看上半部分

供应商：负责发布和维护商品清单。可以增加、更新和删除商品。

商品：商品的详细信息，包括名称、分类、描述、图片、批发规格、价格和商品编码。

团长：可以一键开团，管理团购信息，并查询历史团购信息。

团购：团购的详细信息，包括团购名称、供应商、联系二维码、商品信息、服务小区、团购时间和状态。

团购状态：团购的状态，包括开团中、已截团和已完成。

而下半部分，

商品信息：团购中的商品信息，包含商品、团购价格和推荐理由。

团员：可以参团并查看自己的团购信息。

团员团购信息：团员参与的团购信息，包含团购信息、订单项和订单状态。

订单项：订单中的商品及其数量和价格。

订单状态：订单的状态，包括待处理、已支付和已收货。

方案对比

最后，经小组讨论，我们对比了上述的两种方案，简单总结如下。

微服务架构方案

优点：

可扩展性：独立的服务可以根据需要单独扩展，适合处理不同服务间的负载变化。

技术多样性：每个服务可以采用最适合其功能的技术栈，灵活应对不同技术需求。

独立部署：服务可以独立更新和部署，加快开发速度，降低整体风险。

数据隔离：每个服务可以拥有自己的数据库，避免数据操作上的耦合。

缺点：

系统复杂度：管理多个服务增加了系统的复杂性，需要更多的协调和监控工具。

性能开销：服务间通信可能引入延迟，对性能有一定影响。

数据一致性：多个服务操作同一业务数据可能引发一致性问题。

面向对象模型架构方案

优点：

业务对齐：模型直接反映业务逻辑，便于理解和维护业务规则。

复用性：领域模型促进代码复用，特别是在业务逻辑层面。

可维护性：清晰的业务边界和规则使得维护更为方便。

模型驱动：有利于保持代码和业务的一致性。

缺点：

模型复杂性：良好的领域模型需要深入理解业务，初期设计成本高。

灵活性有限：当业务规模扩大或业务过于复杂时，扩展成本较高。