“珞樱”软件系统设计说明书

书

1. **引言**
   1. **编写目的**

本设计说明书旨在为“珞樱”校园论坛微信小程序的设计提供基础，以指导程序系统的组成和体系结构、用例设计、构件设计、数据设计和部署设计等方面的工作。预期读者包括项目团队成员、开发人员以及项目相关利益相关者。

* 1. **软件系统概述**

一．项目名称：“珞樱”校园论坛微信小程序

二．大致需求描述：“珞樱”论坛旨在为武大师生提供一个在线交流的平台，包括但不限于发布帖子、评论、私信等功能，具有良好的性能和用户体验。

* 1. **定义**

本文件中用到的专门术语、容易引起歧义的概念、关键词缩写及其他需要解释的内容将在相应部分进行解释说明。

* 1. **参考资料**

一．《小程序框架参考文档》

二．《小程序组件参考文档》

三．《小程序 API 参考文档》

四．《小程序服务端 API 参考文档》

五．《微信开发者工具参考文档》

六．《微信云托管参考文档》

七．《微信云开发参考文档》

**2软件设计约束和原则**

1. **设计约束**

一．环境约束： 指定了系统运行的环境要求，包括硬件平台、操作系统、网络环境等。我们的目标是开发一个微信小程序校园论坛，那么系统需要在微信小程序的开发环境下进行设计和开发。

二．开发工具和语言： 我们需要使用微信提供的小程序框架进行软件开发，后端采用团队熟悉的语言和框架，以提高开发效率。

三．标准规范： 我们制定了系统设计和实现需要遵循的标准和规范，包括编码规范、安全标准、数据格式规范等。这些规范有助于提高系统的可维护性和可扩展性，并确保系统的质量。

四．容量和性能要求： 我们将尽可能优化数据存储容量、并发用户数、响应时间等。考虑到团队力量和时间有限，我们将尽可能提高用户的体验，同时兼顾项目的完成进度持续推进。

五．灵活性和配置要求： 我们开发的小程序需要支持不同的用户配置选项或者动态扩展功能。

1. **设计原则**

我们将遵循以下设计原则：

一．实现用户需求： 系统的设计和实现应当紧密地围绕用户需求展开，确保系统能够有效地满足用户的实际需求，并提供良好的用户体验。

二．可维护性： 系统的设计应当注重可维护性，即系统应当易于理解、修改和扩展，以便在后续的演化过程中能够快速响应新的需求和变化。

三．可扩展性： 系统的设计应当具备良好的可扩展性，即系统应当能够方便地进行功能扩展和性能提升，以适应不断变化的业务需求和用户规模。

四．安全性： 系统的设计应当考虑安全性，包括数据安全、用户身份验证、访问控制等方面，以防止恶意攻击和数据泄露。

五．模块化设计： 系统的设计应当采用模块化的设计思想，将系统分解为多个相互独立的模块，每个模块具有清晰的责任和接口，便于管理和维护。

六．分层设计： 系统的设计应当采用分层的设计思想，将系统划分为多个层次，每个层次负责不同的功能和职责，以降低系统的耦合度和复杂度。

七．简单性原则： 系统的设计应当尽量保持简单，避免过度设计和复杂性，以提高系统的可理解性和可维护性。

1. **软件设计方案** 
   1. **体系结构设计**

### 体系结构设计思想

**模块化**：为了提高代码的可维护性和可扩展性，系统应该被划分为独立的模块，每个模块负责一组特定的功能。

**解耦合**：各个模块之间的依赖关系应尽可能地减少，以便在一个模块发生变化时，不会对其他模块产生广泛的连锁反应。

**可扩展性**：设计应允许系统在未来容易地添加新功能或扩展现有功能，以适应不断变化的用户需求和技术发展。

**安全性**：保护用户数据和系统安全是设计的关键考虑因素，应采用加密、身份验证和授权机制来防止未授权访问和数据泄露。

**性能**：系统应设计为能够处理高并发请求，并确保快速响应用户操作。

**可用性**：系统应具有高可用性，确保用户可以随时访问服务。

### 采用的体系结构风格

对于珞樱论坛软件，可以采用**三层架构风格**，它包括以下层次：

**表示层（Presentation Layer）**：负责与用户交互，提供用户界面，并将用户的输入转发给后端逻辑处理。

**业务逻辑层（Business Logic Layer）**：处理应用程序的核心功能，如用户认证、帖子管理、评论处理等。

**数据访问层（Data Access Layer）**：与数据库交互，执行数据的CRUD操作，并提供数据给业务逻辑层。

### 逻辑模型（包图）

在包图中，我们可以定义以下几个主要的包：

**表示层包**：

包含视图和控制器类，如LoginPage、PostView、CommentView等。

**业务逻辑层包**：

包含处理应用程序逻辑的类，如UserManager、PostManager、CommentManager等。

**数据访问层包**：

包含与数据库交互的类，如UserRepository、PostRepository、CommentRepository等。

**实体包**：

包含代表数据库中表的实体类，如User、Post、Comment等。

**服务包**：

包含提供特定服务的类，如EmailService、NotificationService等。

**安全包**：

包含处理安全相关的类，如AuthenticationManager、AuthorizationManager等。

### 部署设计（部署图）

在部署图中，我们可以展示以下组件和它们的部署环境：

**Web服务器**：

托管表示层，处理用户的HTTP请求，提供静态资源和动态页面。

**应用服务器**：

运行业务逻辑层和数据访问层的代码，处理来自Web服务器的请求。

**数据库服务器**：

存储用户数据、帖子、评论等信息，由数据访问层进行操作。

**缓存服务器**：

存储频繁访问的数据，以减轻数据库服务器的压力，提高响应速度。

**文件存储**：

存储用户上传的文件，如图片、文档等。

**负载均衡器**：

分配请求到多个应用服务器，提高系统的可用性和性能。

**安全设备**：

如防火墙、入侵检测系统等，保护系统免受外部攻击。

* 1. **用户界面设计**

3.2.1

原型设计：基于用户期望的简洁明了、操作便捷的原则，设计了“珞樱”论坛软件的用户界面原型。首页展示论坛热门帖子和最新动态，提供搜索栏方便用户快速找到感兴趣的内容；个人中心页面包含用户个人信息、我的帖子、我的收藏等模块；课程评价页面提供了课程搜索和教师评价功能：

一.首页：展示论坛的热门帖子和最新动态，包含一个顶部导航栏（包含登录/注册按钮、个人中心入口、主页链接）、搜索栏（允许用户搜索帖子和课程评价）、帖子列表（展示帖子标题、简介、作者、发布时间）。

二.个人中心：展示用户个人信息（如头像、昵称、个人简介）、我的帖子（用户发布的帖子列表）、我的收藏（用户收藏的帖子列表）等选项。

三.课程评价页面：提供课程搜索框，展示课程列表和对应的教师评价。

下面是用户界面草图：





3.2.2

设计类图与跳转关系：定义了用户界面的关键组件，如帖子列表组件、评论组件、用户信息组件等，并明确了这些组件之间的跳转关系。例如，从首页点击帖子标题可以跳转到帖子详情页面，从个人中心可以跳转到编辑个人信息页面等，包括：

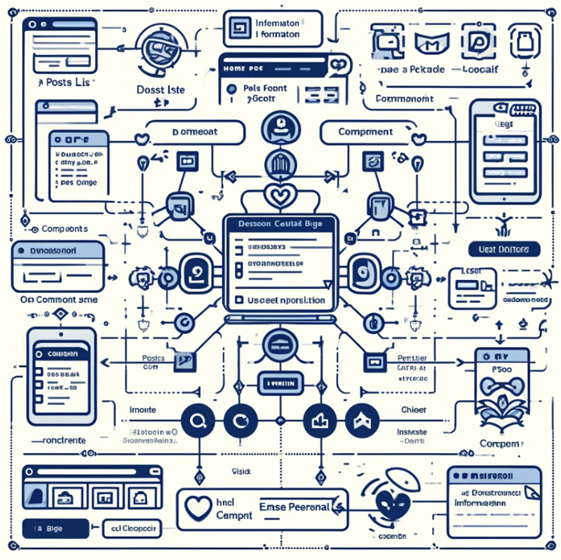
一．用户(User) 类：包含用户的基本信息如用户名、密码、邮箱等属性，以及注册、登录等方法。

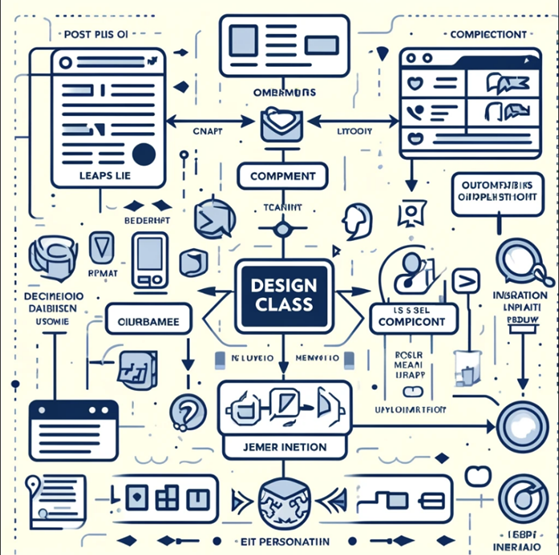
二．帖子(Post) 类：包含帖子的标题、内容、发布时间等属性，以及创建、编辑、删除帖子等方法。

三．评论(Comment) 类：包含评论的内容、发布时间等属性，以及创建、删除评论等方法。

四．课程(Course) 类：包含课程的名称、描述、教师等属性，以及添加、更新课程信息等方法。

下面是一个设计草图：





* 1. **用例设计**

用例的设计模型，包括描述用例实现的顺序图、用例实现的设计类图等，并提供必要的文字补充说明

下面分为几个情境进行用例图分析

**用户注册**：

用户提交注册信息（用户名、密码、邮箱）。

客户端（Client）发送注册请求到服务器（Server）。

服务器验证注册信息的有效性。

用户管理类（UserManager）处理注册逻辑。

数据访问类（DataAccess）将新用户信息存储到数据库（Database）。

服务器返回注册成功或失败的响应给客户端。

**发帖**：

用户填写并提交新帖子信息。

客户端发送发帖请求到服务器。

服务器验证用户身份和帖子内容。

帖子管理类（PostManager）创建帖子对象。

数据访问类更新数据库中的帖子信息。

服务器返回发帖成功或失败的响应给客户端。

**评论帖子**：

用户提交评论内容。

客户端发送评论请求到服务器。

服务器验证用户身份和评论内容。

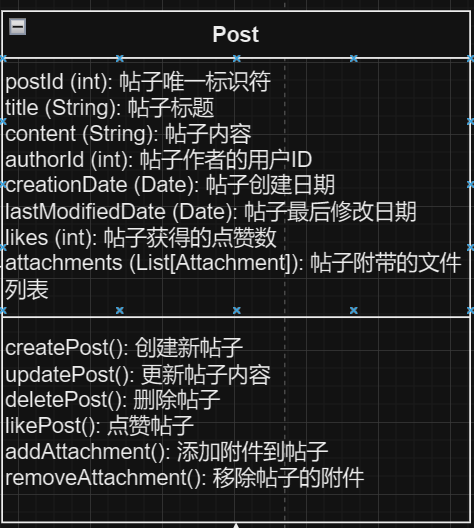
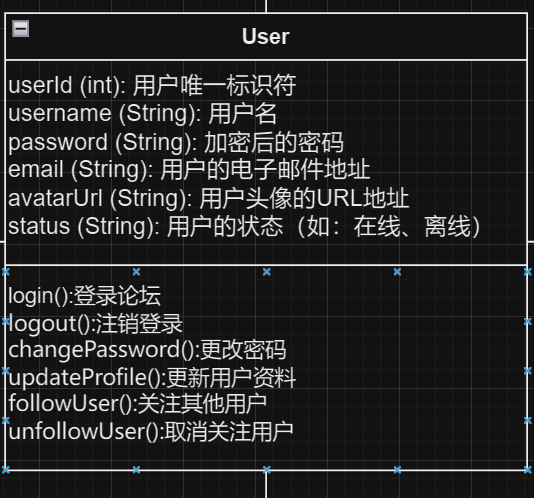
评论管理类（CommentManager）创建评论对象。

数据访问类更新数据库中的评论信息。

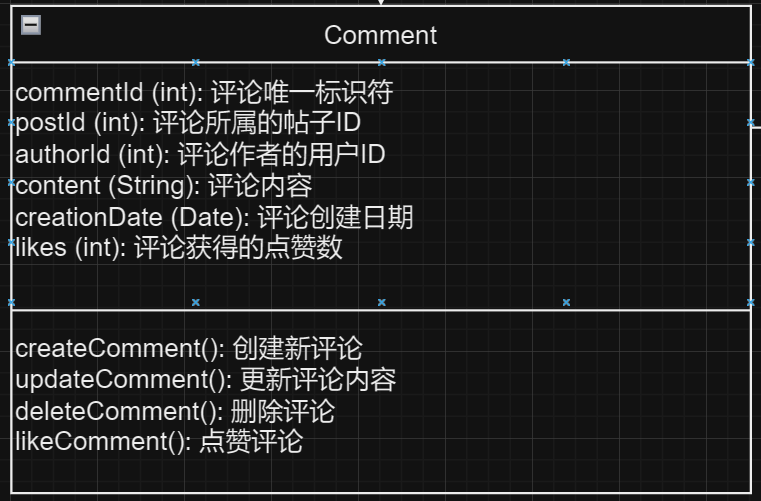
服务器返回评论成功或失败的响应给客户端。

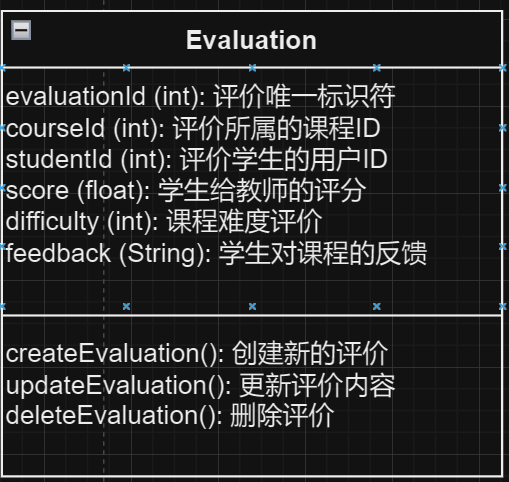
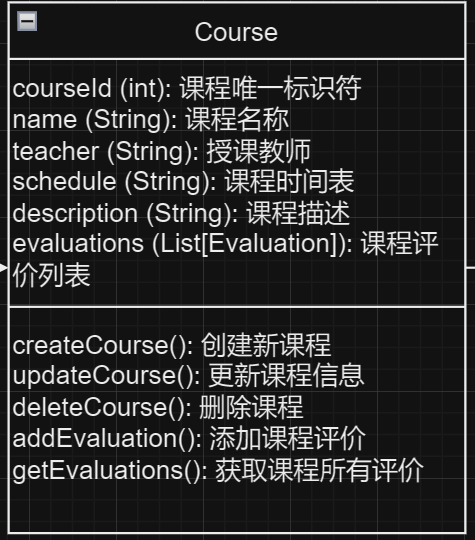
* 1. **类设计**

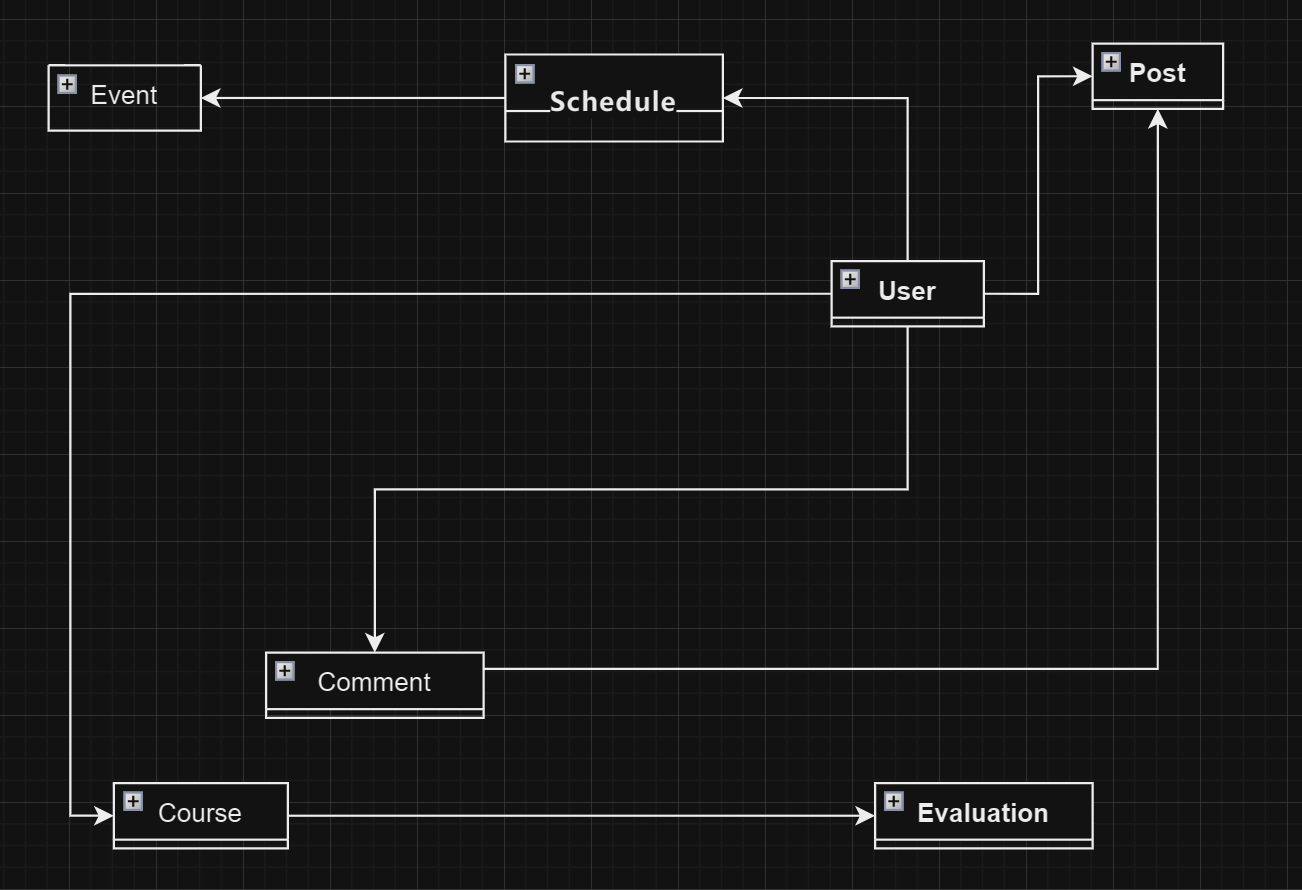
给出各个类的实现模型，包括详细描述各个类的可见范围、类的属性和方法，给出精化后的类图，描述类方法的活动图，类对象的状态图等，并提供必要的文字补充说明











以下为一些控制类

**用户管理类（UserManager）**：

负责处理用户的注册、登录、信息更新等逻辑。

与数据访问类交互，执行数据库操作。

**帖子管理类（PostManager）**：

负责处理帖子的创建、更新、删除和获取列表等逻辑。

与数据访问类交互，执行数据库操作。

**评论管理类（CommentManager）**：

负责处理评论的创建、更新、删除和获取列表等逻辑。

与数据访问类交互，执行数据库操作。

**数据访问类（DataAccess）**：

负责与数据库交互，执行CRUD（创建、读取、更新、删除）操作。

可以进一步细分为用户数据访问（UserDataAccess）、帖子数据访问（PostDataAccess）等。

* 1. **数据设计**

给出软件系统中永久数据的设计模型，包括描述数据库及表的设计类图，描述数据操作的活动图、必须提供必要的文字补充说明

**用户表（Users）**：

UserID (PK): 用户唯一标识符。

Username: 用户名。

PasswordHash: 加密后的密码。

Email: 用户的电子邮件地址。

AvatarUrl: 用户头像的URL地址。

Status: 用户的状态（如：在线、离线）。

**帖子表（Posts）**：

PostID (PK): 帖子唯一标识符。

UserID (FK): 外键，指向用户表的用户ID。

Title: 帖子标题。

Content: 帖子内容。

CreationDate: 帖子创建日期。

LastModifiedDate: 帖子最后修改日期。

LikeCount: 帖子获得的点赞数。

**评论表（Comments）**：

CommentID (PK): 评论唯一标识符。

PostID (FK): 外键，指向帖子表的帖子ID。

UserID (FK): 外键，指向用户表的用户ID。

Content: 评论内容。

CreationDate: 评论创建日期。

LikeCount: 评论获得的点赞数。

**课程表（Courses）**：

CourseID (PK): 课程唯一标识符。

Name: 课程名称。

Teacher: 授课教师。

Schedule: 课程时间表。

Description: 课程描述。

**评价表（Evaluations）**：

EvaluationID (PK): 评价唯一标识符。

CourseID (FK): 外键，指向课程表的课程ID。

StudentID (FK): 外键，指向用户表的学生ID。

Score: 学生给教师的评分。

Difficulty: 课程难度评价。

Feedback: 学生对课程的反馈。

**日程安排表（Schedules）**：

ScheduleID (PK): 日程安排唯一标识符。

UserID (FK): 外键，指向用户表的用户ID。

StartDate: 日程安排开始日期。

EndDate: 日程安排结束日期。

**事件表（Events）**：

EventID (PK): 事件唯一标识符。

ScheduleID (FK): 外键，指向日程安排表的日程安排ID。

Title: 事件标题。

StartTime: 事件开始时间。

EndTime: 事件结束时间。

Location: 事件地点。

Description: 事件描述。