软件(结构)设计说明(SDD)

说明：

1.《软件(结构)设计说明》(SDD)描述了计算机软件配置项(CSCI 的设计。它描述了 CSCI 级设计决策、CSCI 体系结构设计(概要设计)和实现该软件所需的详细设计。SDD 可用接口设计说明 IDD 和数据库(顶层)设计说明 DBDD 加以补充。

2.SDD 连同相关的 IDD 和 DBDD 是实现该软件的基础。向需方提供了设计的可视性，为软件支持提供了所需要的信息。

3.IDD 和 DBDD 是否单独成册抑或与 SDD 合为一份资料视情况繁简而定。

小组成员：陈静雯（组长），聂方正，官腾飞，姚思彤，孙思娴

小组分工：

1-2：聂方正 fangzhengNie

3-4.1：孙思娴 bbbbvfhjv

4.2-4.3：陈静雯 Quiet-chen

4.4-4.5：官腾飞 guan041109

5-7：姚思彤 songyin33

Github 项目地址：https://github.com/Quiet-chen/SoftwareEngineering

目录

1 引言

1.1 标识

本文档适用于“校园二手商品交易平台”（Campus C2C Trading Platform，简称 CCTP），版本号 1.0.0，初始发行版本为开发测试版。系统由学生开发团队自主设计并维护，采用 Web 技术栈实现，主要面向高校学生群体，支持个人对个人（C2C）的二手商品发布、搜索、交易及社区互动功能。当前系统无商业投资方，由团队独立运营，后续版本迭代计划以开源形式发布。

1.2 系统概述

本系统旨在解决高校内二手商品流转效率低、交易信任度不足的问题，提供轻量化的在线交易服务。系统核心功能包括：商品图文发布、地理位置标记、担保交易流程、双向信用评价及兴趣社区互动。开发团队由在校学生组成，基于课余时间采用敏捷开发模式推进，初期运行现场定位为山东大学内部局域网，未来计划扩展至多校区互联场景。

1.3 文档概述

本设计说明书旨在详细描述校园二手交易平台系统的结构设计、模块划分、接口设计、数据结构、运行逻辑与实现细节。它是系统开发与维护的重要技术文档，也是团队成员进行编码、测试、部署的重要依据。
本说明书主要面向以下对象：
开发人员：用于指导代码实现与模块集成；
测试人员：用于理解系统结构与功能，制定测试策略；
运维人员：了解系统部署结构与执行流程；
项目审查人员：用于评估系统设计的完整性与合理性。
文档不涉及敏感数据，但要求使用者遵守团队内部保密协议，禁止向第三方泄露未公开的技术实现细节。

1.4 基线

本《软件结构设计说明书》的编写工作是在以下两项主要文档的基础上进行的，这些文档构成了本系统设计的基线依据：
《校园二手交易平台软件需求可行性分析报告》
（1）对系统的背景、目标用户群、技术可行性、经济可行性和风险因素进行了全面分析；
（2）明确了项目的实施必要性与开发可行性，为后续系统开发奠定了方向性基础。
《校园二手交易平台软件需求规格说明书》V1.0
（1）对系统的功能需求、性能需求、用户角色、业务流程、系统接口、安全性和数据存储等内容做了详细规范；

（2）明确了系统的功能模块划分，是本设计说明中模块设计、接口定义、数据结构设计和流程设计的核心依据。
上述文档的评审与确认标志着系统需求的正式冻结，作为设计阶段的输入资料，其内容对本说明书的编写起到了指导和约束作用。

此外，在需求文档基础上，还参考了如下初步资料与辅助文档：
系统功能原型图（UI 草图与页面流程图）；
数据库逻辑结构草案；
技术选型初步方案（Java + Spring Boot + Vue.js + MySQL）；
小组内部开发流程和编码规范文档。
这些基线内容为本设计说明书中各模块的结构设计、接口通信、执行流程和数据组织等提供了依据与标准。

2 引用文件

1.《校园二手交易需求调研报告》
2.《Web 应用安全开发规范》
3.《支付宝开放平台 Web 接入指南》
4.《Vue.js 官方文档》
版本：3.3.4
维护方：Vue.js Core Team
链接：https://vuejs.org
5.《Spring Boot 官方指南》
版本：3.2.0
发布方：VMware
链接：https://spring.io/projects/spring-boot
6.《MySQL 8.0 参考手册》
版本：8.0.36
发布方：Oracle Corporation
链接：https://dev.mysql.com/doc/refman/8.0/en/
7.《山东大学校园网络使用管理办法》
发布单位：山东大学信息化办公室

3 CSCI 级设计决策

本章根据需要分条给出 CSCI 级设计决策，即 CSCI 行为的设计决策
以下是针对二手交易市场线上平台的 CSCI 级设计决策分析：

a. 接口设计决策

采用 RESTful API 作为主要接口规范，支持 JSON 数据格式
用户端提供 Web/App 双平台统一接口
支付系统接口符合 PCI-DSS 安全标准
第三方登录接口支持 OAuth 2.0 协议

b. 行为响应决策

商品搜索响应时间<500ms（95%分位）
交易流程采用状态机模式，包含 6 个标准状态
采用基于信誉分的交易风险评估算法
异常输入处理：自动过滤 SQL/XSS 注入字符

c. 数据呈现决策

商品列表采用分页加载（每页 20 条）
用户数据采用分级显示策略（公开/隐私分离）
交易历史支持 CSV 导出功能
图片存储使用 CDN 加速访问

d. 安全保密决策

敏感数据采用 AES-256 加密存储
双重认证机制（短信+邮箱）用于关键操作
隐私数据遵循 GDPR"默认保护"原则
每日自动备份+异地容灾存储

e. 系统级决策

微服务架构保证模块可独立扩展
采用蓝绿部署实现无缝更新
监控系统覆盖所有关键性能指标
日志保留策略：业务日志 30 天，审计日志 1 年

设计约定：

交易状态依赖支付系统回调
信誉评分依赖历史交易数据
移动端适配遵循 Material Design 规范
所有时间戳采用 ISO 8601 标准

4 CSCI 体系结构设计

采用模块化思想，各层通过 REST API 交互。业务逻辑层的状态机实现需严格遵循 3.1 约定的转换规则，数据层设计需满足 ACID 与 BASE 的混合使用场景。

4.1 体系结构

4.1.1 程序(模块)划分

系统分层架构

采用经典三层架构（表现层/业务逻辑层/数据层），依赖关系为单向调用：
表现层：React/Vue 前端框架（依赖浏览器兼容性）
业务层：Spring Boot 微服务（依赖 JDK 11+环境）
数据层：MySQL 集群+Redis 缓存（依赖数据库主从同步状态）

4.1.2 程序(模块)层次结构关系

单向依赖：表现层仅调用业务逻辑层，业务逻辑层仅调用数据访问层，确保依赖路径清晰且单向。
解耦目标：业务逻辑层作为核心枢纽，隔离前后端技术细节，如前端框架（React/Vue）与后端数据库（MySQL）的差异。

各层交互机制

层级	职责描述	交互规范与依赖条件
表现层	用户界面渲染与交互事件处理	通过 REST API 调用业务逻辑层 依赖前端框架兼容性
业务逻辑层	执行商品发布，交易规划校验等	调用数据层接口获取持久化数据，依赖数据库连接池状态
数据层	数据存储与缓存管理	以来主从数据库同步状态及 Redis 缓存可用性

4.2 全局数据结构说明

本章说明本程序系统中使用的全局数据常量、变量和数据结构。

4.2.1 常量

数据文件路径及名称：main/java/com/rabbiter/fm/common/

ErrorMsg：错误信息枚举

```
1       ACCOUNT_EXIT("用户已存在"),
2       ACCOUNT_Ban("账号已被封禁"),
3       ACCOUNT_NOT_EXIT("用户不存在"),
4       PASSWORD_IS_NOT_SAME("密码不一致"),
5       PASSWORD_RESET_ERROR("修改密码失败"),
6       EMAIL_SEND_ERROR("邮件发送失败  请重试"),
7       PARAM_ERROR("参数错误"),
8       SYSTEM_ERROR("系统错误"),
9       REGISTER_ERROR("注册失败"),
10      FILE_TYPE_ERROR("文件类型错误  请选择.jpg 或.png"),
11      FILE_UPLOAD_ERROR("文件上传失败"),
12      FILE_NOT_EXIT("文件不存在"),
13      FILE_DOWNLOAD_ERROR("文件下载异常"),
14      FILE_SIZE_ERROR("文件过大"),
15      OPERAT_FREQUENCY("操作频繁  稍后重试"),
16      MISSING_PARAMETER("缺少参数"),
17      COOKIE_ERROR("请重新登录"),
18      EMAIL_LOGIN_ERROR("登录失败  账号或密码错误"),
19      JSON_READ_ERROR("json 参数解析错误"),
20      FORM_NUMBER_ERROR("表单 id 错误"),
21      REPEAT_COMMIT_ERROR("请勿重复提交"),
22      COMMIT_FAIL_ERROR("提交失败"),
23      FAVORITE_EXIT("收藏已存在"),
24      IDLE_ITEM_LABEL_EXIST("该分类下存在闲置商品"),
25      TYPE_HAS_EXIST("分类名称已存在");
26      private String msg;
```

PathUtils：路径相关字符串

```
string path ;
```

OrderTask：商品相关

```
private long time;
private OrderModel orderModel;
```

4.2.2 变量

数据文件路径及名称：main/java/com/rabbiter/fm/
系统中未定义统一的全局变量，而是通过局部变量管理。各 Model 类中变量用于数据对象传输，如。

```
private Long id; // 自增主键
private String consigneeName; // 收货人姓名
private String consigneePhone; // 收货人手机号
private String provinceName; // 省份名
private String cityName; // 市名
private String regionName; // 区名
private String detailAddress; // 详细地址
private Boolean defaultFlag; // 是否默认地址
private Long userId; // 所属用户 ID
```

4.2.3 数据结构

数据文件路径及名称：main/java/com/rabbiter/fm/model/

4.2.3.1 AdminModel

AdminModel 是用于管理后台管理员信息的数据模型类，封装了管理员的账号信息、密码、姓名等属性，并重写了常用的 equals、hashCode 和 toString 方法，便于在集合中操作与日志调试。
定义：

```
public class AdminModel implements Serializable {
    private Long id; // 自增主键
    private String accountNumber; // 管理员账号
    private String adminPassword; // 管理员密码
    private String adminName; // 管理员名称
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	唯一标识主键	自增主键
accountNumber	String	管理员账号	登录名
adminPassword	String	管理员密码	需加密存储
adminName	String	管理员姓名	显示用途

4.2.3.2 AddressModel

AddressModel 是用于管理用户收货地址的数据模型类，封装了姓名、手机号、省市区、详细地址、是否为默认地址等信息。此类主要用于订单系统中的地址管理功能，包括地址的添加、查询、更新和删除等。

定义：

```
public class AddressModel implements Serializable {
    private Long id;
    private String consigneeName;
    private String consigneePhone;
    private String provinceName;
    private String cityName;
    private String regionName;
    private String detailAddress;
    private Boolean defaultFlag;
    private Long userId;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	地址记录唯一标识	自增主键
consigneeName	String	收货人姓名	-
consigneePhone	String	收货人手机号	建议使用手机号格式校验
provinceName	String	所在省	-
cityName	String	所在市	-
regionName	String	所在区	-
detailAddress	String	详细地址	包括街道、楼号等信息
defaultFlag	Boolean	是否为默认收货地址	true 为默认地址
userId	Long	归属用户主键 ID	外键，关联 UserModel

4.2.3.3 FavoriteModel

FavoriteModel 是用户收藏功能的数据模型类，用于表示用户收藏的闲置物品。包含收藏时间、关联用户和关联商品信息，并支持嵌套查询具体的闲置信息。

定义：

```
public class FavoriteModel implements Serializable {
    private Long id;
    private Date createTime;
    private Long userId;
    private Long idleId;
    private IdleItemModel idleItem;
    private static final long serialVersionUID = 1L;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	收藏记录唯一标识	自增主键
createTime	Date	收藏时间	创建收藏时的时间戳
userId	Long	用户主键 ID	外键，关联 UserModel
idleId	Long	闲置物品主键 ID	外键，关联 IdleItemModel
idleItem	IdleItemModel	闲置物品对象	作为嵌套对象，用于冗余查询展示信息

4.2.3.4 IdleItemModel

IdleItemModel 表示平台中用户发布的闲置物品，是系统的核心商品实体。用于管理闲置商

品的信息展示、搜索过滤与发布状态维护。
定义：

```
public class IdleItemModel implements Serializable {
    private Long id;
    private String idleName;
    private String idleDetails;
    private String pictureList;
    private BigDecimal idlePrice;
    private String idlePlace;
    private Integer idleLabel;
    private Date releaseTime;
    private Byte idleStatus;
    private Long userId;
    private UserModel user;
    private static final long serialVersionUID = 1L;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	闲置物品唯一标识	自增主键
idleName	String	闲置物品名称	简要商品标题
idleDetails	String	闲置物品详情	商品描述内容，支持富文本或换行
pictureList	String	图片地址列表	图片 URL 以分隔符拼接存储（如 JSON/逗号）
idlePrice	BigDecimal	闲置物品价格	精确到小数的价格
idlePlace	String	发货地区	文本形式的地区名（如“北京-海淀区”）
idleLabel	Integer	分类标签	可映射为标签枚举，例如：1-电子 2-书籍等
releaseTime	Date	发布时间	商品首次发布的时间戳
idleStatus	Byte	状态	状态值枚举：1=发布中；2=下架；0=已删除
userId	Long	发布用户主键 ID	外键，关联 UserModel

user	UserModel	用户详细信息	嵌套对象，仅用于展示使用，不入库

4.2.3.5 MessageModel

MessageModel 表示用户在平台上对闲置商品的留言信息。支持一级留言与二级回复功能，允许留言关联特定用户与留言记录，实现用户间的问答交流。
定义：

```
public class MessageModel implements Serializable {
    private Long id;
    private Long userId;
    private UserModel fromU;
    private Long idleId;
    private IdleItemModel idle;
    private String content;
    private Date createTime;
    private Long toUser;
    private UserModel toU;
    private Long toMessage;
    private MessageModel toM;
    private static final long serialVersionUID = 1L;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	留言主键	自增主键
userId	Long	留言用户的主键 ID	发起留言的用户 ID
fromU	UserModel	留言用户完整信息	展示用，不入库
idleId	Long	所留言的闲置商品主键 ID	外键，关联 IdleItemModel
idle	IdleItemModel	所留言的闲置商品对象	展示用，不入库
content	String	留言内容	限制字数，一般不超过 300 字符
createTime	Date	留言时间	留言创建时间

toUser	Long	回复目标用户的主键 ID	若非回复，则为 null
toU	UserModel	回复目标用户对象	展示用，不入库
toMessage	Long	回复的留言 ID	若非二级回复，则为 null
toM	MessageModel	回复的留言对象	展示用，不入库

4.2.3.6 OrderAddressModel

OrderAddressModel 用于表示订单的收货地址信息，记录订单关联的收件人姓名、手机号以及详细地址。

定义：

```
public class OrderAddressModel implements Serializable {
    private Long id;
    private Long orderId;
    private String consigneeName;
    private String consigneePhone;
    private String detailAddress;
    private static final long serialVersionUID = 1L;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	主键 ID	自增主键
orderId	Long	关联的订单主键 ID	外键，关联订单模型
consigneeName	String	收货人姓名	不应超过 20 个字符
consigneePhone	String	收货人手机号	采用字符串便于兼容格式
detailAddress	String	详细地址	建议限制长度不超 255 字

4.2.3.7 OrderModel

OrderModel 表示平台中的订单信息，包含订单编号、关联用户与商品、价格、支付信息、订单状态等。

定义：

```
public class OrderModel implements Serializable {
        private Long id;
        private String orderNumber;
        private Long userId;
        private UserModel user;
        private Long idleId;
        private IdleItemModel idleItem;
        private BigDecimal orderPrice;
        private Byte paymentStatus;
        private String paymentWay;
        private Date createTime;
        private Date paymentTime;
        private Byte orderStatus;
        private Byte isDeleted;
        private static final long serialVersionUID = 1L;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	订单主键 ID	自增主键
orderNumber	String	订单编号	应全局唯一
userId	Long	用户主键 ID	外键，关联用户
user	UserModel	用户对象	业务中用于嵌套用户详情
idleId	Long	闲置物品主键 ID	外键，关联闲置商品
idleItem	IdleItemModel	闲置物品对象	业务中用于嵌套商品详情
orderPrice	BigDecimal	订单价格	建议保留两位小数
paymentStatus	Byte	支付状态	0=未支付，1=已支付
paymentWay	String	支付方式	如：支付宝、微信、余额等
createTime	Date	创建时间	下单时间

paymentTime	Date	支付时间	可为空，表示尚未支付
orderStatus	Byte	订单状态	如：0=待支付，1=已完成，2=取消等
isDeleted	Byte	是否逻辑删除	0=否，1=是

4.2.3.8 TypeModel

TypeModel 表示平台中闲置物品的分类信息，用于标识商品类型、类别等。
定义：

```
public class TypeModel implements Serializable {
    private Long id;
    private String name;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	分类主键 ID	自增主键
name	String	分类名称	如：“数码产品”、“书籍”

4.2.3.9 UserModel

UserModel 表示平台用户的基本信息，包括账号、密码、昵称、状态等，是用户登录和个人资料管理的核心数据结构。
定义：

```
public class UserModel implements Serializable {
    private Long id;
    private String accountNumber;
    private String userPassword;
    private String nickname;
    private String avatar;
    private Date signInTime;
    private Byte userStatus;
}
```

取值等说明：

字段名	类型	说明	备注
id	Long	用户主键 ID	自增主键
accountNumber	String	登录账号（手机号）	登录名，全平台唯一
userPassword	String	登录密码	建议加密存储
nickname	String	用户昵称	用户可自定义
avatar	String	头像 URL	图片链接
signInTime	Date	注册时间	用户首次注册时间
userStatus	Byte	用户状态	如 0=正常，1=禁用，2=注销等

4.3 CSCI 部件

4.3.1 软件配置项标识

配置项名称	类型	唯一标识符	描述
UserModel	类（Model）	com.rabbiter.fm.model.UserModel	用户数据实体模型类
OrderModel	类（Model）	com.rabbiter.fm.model.OrderModel	订单数据实体模型类
OrderAddressModel	类（Model）	com.rabbiter.fm.model.OrderAddressModel	订单地址数据实体模型类
TypeModel	类（Model）	com.rabbiter.fm.model.TypeModel	商品类型数据实体模型类
common 配置文件	资源文件	com.rabbiter.fm.common	系统公共常量及配置

4.3.2 软件配置项静态关系

OrderModel 通过 userId 与 UserModel 关联；通过 idleId 与 IdleItemModel 关联。
OrderAddressModel 通过 orderId 关联对应的 OrderModel。
UserModel 与其他模块存在一对多关系，代表一个用户可以有多个订单。
TypeModel 为分类数据，可能与闲置物品 IdleItemModel 关联。

4.3.3 软件配置项用途及需求分配

UserModel：用于管理和存储用户信息，支持用户登录、注册和账户管理功能，满足 CSCI

需求编号 REQ-USER-001。
OrderModel：管理订单业务数据，满足 CSCI 需求编号 REQ-ORDER-001。
OrderAddressModel：用于订单收货地址管理，满足 REQ-ORDER-ADDRESS-001。
TypeModel：用于商品分类管理，支持浏览和筛选功能，满足 REQ-TYPE-001。
common 配置文件：集中管理系统参数和常量，满足系统整体配置需求 REQ-COMMON-001。

4.3.4 软件配置项开发状态/类型

所有上述 Model 均为项目新开发的代码，版本号 1.0。
设计符合 Java Bean 规范，支持序列化。
尚未重用第三方库的模型代码，所有代码均由团队自主开发。

4.3.5 计算机硬件资源使用

运行于校园二手交易平台服务器，典型环境为 windows。
资源需求如下：
1. 处理器能力：支持多线程，典型负载下占用 CPU 不超过 20%。
2. 内存容量：每实例预计占用内存约 100MB。
3. 存储容量：数据库存储所有模型数据，模型本身为轻量级 Java 类。
4. 网络设备：依赖于数据库和前端通信，网络带宽需求中等。
以上资源均符合系统资源分配计划，无特别硬件限制。

4.3.6 软件配置项存放库

所有 Model 类存放于版本管理库 git 的目录 src/main/java/com/rabbiter/fm/model/中。
公共常量及配置存放于 src/main/java/com/rabbiter/fm/common/目录。
版本控制系统采用 Git，主分支为 main，版本标识使用语义化版本号。

4.4 执行概念

4.4.1 执行控制流

本系统采用基于 Spring Boot 和 Vue 的前后端分离架构，执行控制流主要体现在前后端的交互以及后端服务之间的调用。前端通过 Vue 框架构建用户界面，用户操作触发前端事件，前端通过 HTTP 请求与后端进行通信。后端采用 Spring Boot 框架，接收前端请求后，通过 Controller 层调用 Service 层的业务逻辑处理，Service 层进一步调用 Dao 层与数据库进行交互，最终将处理结果返回给前端。

4.4.2 数据流

数据流主要涉及用户数据、商品数据、订单数据等的传输与处理。用户在前端页面进行操作（如注册、登录、发布商品、购买商品等），前端将相关数据封装为 JSON 格式，通过 HTTP 请求发送到后端。后端接收到请求后，解析 JSON 数据，进行相应的业务处理，并将结果以 JSON 格式返回给前端。例如，用户注册时，前端将用户信息（用户名、密码等）发送到后端，后端验证信息合法性后存储到数据库，并返回注册成功或失败的响应。

4.4.3 动态控制序列

系统的动态控制序列主要体现在用户操作的响应流程中。以用户登录为例，用户在登录页面输入账号和密码后点击登录按钮，前端触发登录事件，将账号和密码发送到后端。后端接收到登录请求后，验证账号和密码的正确性，如果验证通过，生成用户会话信息并返回登录成功的响应，前端接收到响应后跳转到主页面；如果验证失败，返回登录失败的响应，前端提示用户登录失败。

4.4.4 状态转换图

系统的状态转换主要体现在用户状态和商品状态的转换。用户状态包括未登录、已登录、被封禁等，用户通过登录操作从未登录状态转换到已登录状态，通过注销操作从已登录状态转换到未登录状态，管理员可以通过操作将用户状态设置为被封禁。商品状态包括上架、下架、已售出等，用户可以通过发布操作将商品状态设置为上架，通过下架操作将商品状态设置为下架，当商品被购买后，商品状态转换为已售出。

4.4.5 并发执行

系统支持多用户同时在线操作，后端服务采用多线程处理机制，能够同时处理多个前端请求。当多个用户同时访问系统时，后端通过线程池分配线程来处理每个用户的请求，保证系统的响应速度和稳定性。例如，多个用户同时发布商品时，后端可以同时处理多个商品发布请求，将商品信息存储到数据库中。

4.4.6 异常处理

系统在执行过程中会进行异常处理，以保证系统的稳定运行。后端在处理请求时，会捕获可能出现的异常，如数据库操作异常、网络异常等，并返回相应的错误信息给前端。前端接收到错误信息后，会提示用户操作失败的原因。例如，当数据库连接失败时，后端会捕获异常并返回数据库连接失败的错误信息，前端提示用户系统繁忙，请稍后再试。

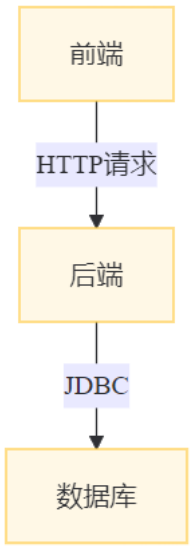
4.5 接口设计

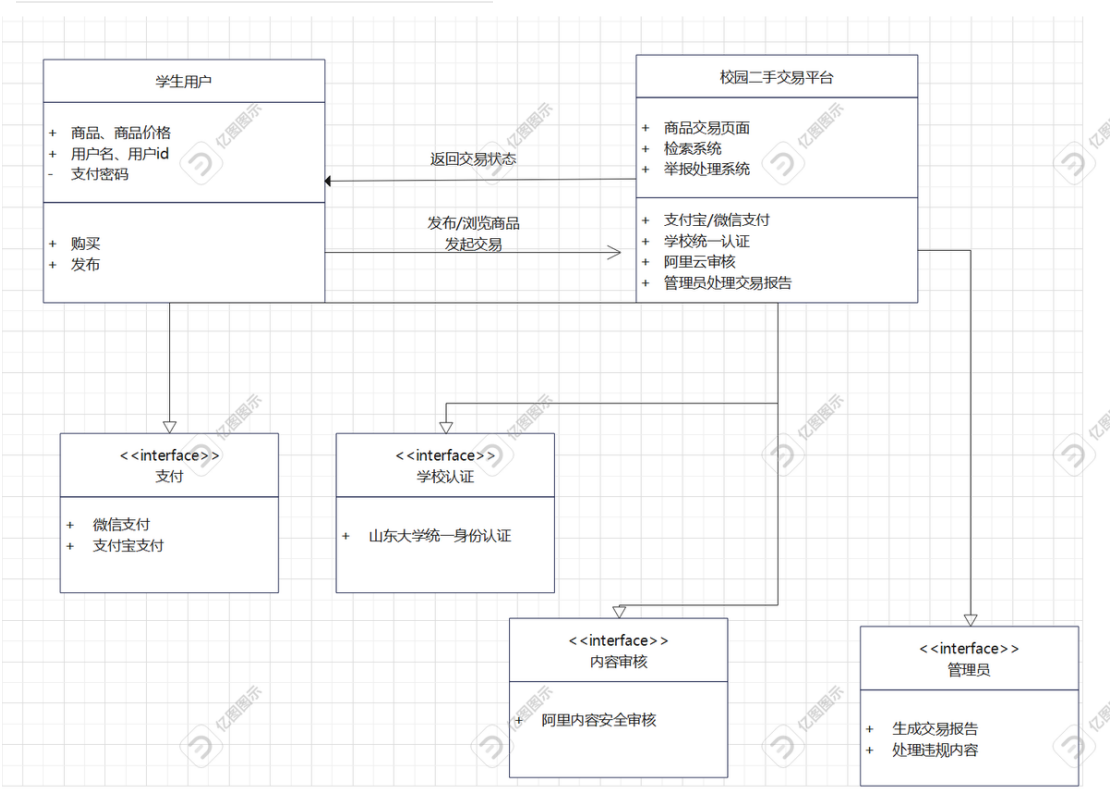
4.5.1 接口标识与接口图

本系统的主要接口包括前端与后端的接口、后端与数据库的接口。前端与后端的接口通过 HTTP 协议进行通信，后端与数据库的接口通过 JDBC 进行通信。接口标识如下表所示：

接口名称	接口编号	版本	文档引用
前端与后端接口	API-001	1.0	本设计说明第4.5.2节
后端与数据库接口	API-002	1.0	本设计说明第4.5.3节

接口图如下所示：





4.5.2 前端与后端接口

前端与后端接口采用 RESTful API 风格，通过 HTTP 请求进行通信。接口特性如下：

- 优先级：高
- 接口类型：实时数据传输
- 数据元素特性：
  - 用户名：
    - 项目唯一标识符：API-001-001
    - 非技术名称：用户名
    - 数据类型：字符串
    - 大小与格式：长度不超过 20 个字符
    - 来源：前端 ■ 接收者：后端
  - 密码：
    - 项目唯一标识符：API-001-002
    - 非技术名称：密码
    - 数据类型：字符串
    - 大小与格式：长度不超过 32 个字符
    - 来源：前端
    - 接收者：后端
- 数据元素集合体特性：
  - 用户信息：
    - 项目唯一标识符：API-001-003
    - 非技术名称：用户信息
    - 数据元素及其结构：包含用户名、密码、昵称等
    - 来源：前端
    - 接收者：后端
- 通信方法特性：
  - 项目唯一标识符：API-001-004
  - 通信链路：HTTP
  - 消息格式化：JSON
  - 流控制：无
  - 安全性：使用 HTTPS 加密
- 协议特性：
  - 项目唯一标识符：API-001-005
  - 协议优先级：高
  - 合法性检查：验证 HTTP 状态码
  - 错误控制：返回错误信息
  - 同步：无连接保持

4.5.3 后端与数据库接口

后端与数据库接口采用 JDBC 进行通信。

接口特性如下：

- 优先级：高
- 接口类型：数据的存储与检索
- 数据元素特性：
  - 用户 ID：
    - 项目唯一标识符：API-002-001
    - 非技术名称：用户 ID
    - 数据类型：整数
    - 大小与格式：无限制
    - 来源：后端
    - 接收者：数据库
  - 商品名称：
    - 项目唯一标识符：API-002-002
    - 非技术名称：商品名称
    - 数据类型：字符串
    - 大小与格式：长度不超过 50 个字符
    - 来源：后端
    - 接收者：数据库
- 数据元素集合体特性：
  - 商品信息：
    - 项目唯一标识符：API-002-003
    - 非技术名称：商品信息
    - 数据元素及其结构：包含商品名称、价格、描述等
    - 来源：后端
    - 接收者：数据库
- 通信方法特性：
  - 项目唯一标识符：API-002-004
  - 通信链路：JDBC
  - 消息格式化：SQL 语句
  - 流控制：无
  - 安全性：无
- 协议特性：
  - 项目唯一标识符：API-002-005
  - 协议优先级：高
  - 合法性检查：验证 SQL 语句合法性
  - 错误控制：返回错误信息
  - 同步：无连接保持

5 CSCI 详细设计

5.1 支付接口 (IF-PAY-001)

a. 设计决策

采用指数退避算法实现 3 次重试机制（间隔时间：1s, 3s, 9s）

交易状态轮询策略：TRADE_SUCCESS 前每 5 秒查询一次（最长持续 300 秒）

b. **约束条件**

必须使用 TLS 1.2+加密传输

金额精度必须保留两位小数

交易号必须保证全局唯一性

f. **逻辑说明**

4) 操作序列：

- 启动条件：收到用户支付指令
- 控制流：

A[生成trade_no] --> B[HTTPS请求支付宝] --> C{响应≤2s?} -->|是| D[解析trade_status] -->|否| E[重试计数器+1] --> F[记录事务日志]

5) 异常处理：

- HTTP 503 错误：触发重试机制
- 加密失败：立即中止交易并通知管理员

5.2 学校认证接口 (IF-AUTH-002)

a. **设计决策**

采用 HMAC-SHA256 签名验证 IP 白名单

校内 VPN 通道使用 L2TP/IPsec 协议

b. **约束条件**

学号必须符合^[0-9]{10}$正则表达式

仅允许从 192.168.1.0/24 网段访问

f. **逻辑说明**

3) 响应处理：

- HTTP 403 时触发人工复核流程
- 响应时间>1s 时记录性能日志

5.3 内容审核接口 (IF-MOD-003)

a. **设计决策**

采用异步回调机制处理审核结果

使用环形缓冲区实现频率控制（100 请求/分钟）

b. **约束条件**

必须包含 dataId 作为唯一追踪标识

文本长度≤5000 字符

f. **逻辑说明**

5) 错误处理：

- suggestion=block 时触发内容隔离流程
- 超时未响应时重新提交审核队列

5.4 通用特性

d. **过程式命令**

支付日志清理命令：find /var/log/payment -mtime +30 -exec rm {} \;

e. **时序控制**

优先级仲裁规则：

1.IF-PAY-001 独占带宽≥500Kbps

2.IF-AUTH-002 最大延迟≤5s

3. 3.IF-MOD-003 允许延迟≤60s

f. **错误恢复**

支付接口事务日志格式：[YYYY-MM-DD HH:MM:SS] | trade_no | status | retry_count | operator_id

自动通知条件：连续 3 次重试失败或 TLS 握手异常

6 需求的可追踪性

a. 从本 SDD 中标识的每个软件配置项到分配给它的 CSCI 需求的可追踪性

1. **表现层组件： React/Vue 前端框架**：这些组件负责用户界面的展示和交互，它们直接对应于用户需求中关于界面设计、用户体验和交互流程的部分。例如，用户登录界面、数据展示界面等需求会被分配到这些前端框架组件中。
2. **业务层组件： Spring Boot 微服务**：这些微服务处理业务逻辑，包括数据处理、业务规则验证等。它们直接对应于 CSCI 需求中关于业务功能、数据处理逻辑和业务规则的部分。例如，用户注册验证、订单处理等需求会被分配到相应的 Spring Boot 微服务中。
3. **数据层组件： MySQL 集群**：用于数据的持久化存储，对应于 CSCI 需求中关于数据存储、数据检索和数据一致性的部分。
    a. **Redis 缓存**：用于提高数据访问速度，减轻数据库负载，对应于 CSCI 需求中关于性能优化、缓存策略和数据一致性的部分（在缓存与数据库同步的上下文中）。

b. 从每个 CSCI 需求到它被分配给的软件配置项的可追踪性

为了确保每个 CSCI 需求都被正确实现，并且可以在系统中找到对应的实现部分，我们需要建立从 CSCI 需求到软件配置项的反向追踪。

1. **用户界面需求：** 例如，“系统应提供一个直观、易用的用户登录界面”这一需求，会被分配到 React/Vue 前端框架中的相关组件。在 SDD 中，应明确记录这一需求与前端组件之间的对应关系。
2. **业务功能需求：** 例如，“系统应支持用户注册功能，并验证注册信息的有效性”这一需求，会被分配到 Spring Boot 微服务中的用户注册微服务。同样，在 SDD 中应有明确的记录，说明这一需求是如何被微服务组件实现的。
3. **数据存储与检索需求：** 例如，“系统应确保用户数据的持久化存储，并提供高效的数据检索功能”这一需求，会被分配到 MySQL 集群和相关的数据访问层组件。SDD 中应详细记录这一需求与数据层组件之间的对应关系，包括数据表的设计、索引策略等。
4. **性能优化需求：** 例如，“系统应优化数据访问速度，减少用户等待时间”这一需求，可能会被分配到 Redis 缓存组件，以及相关的缓存策略设计。在 SDD 中，应明确记录这一需求与缓存组件之间的对应关系，以及缓存策略的具体实现方式。

7 注解

1.数据库设计补充说明

a. 支付事务相关表结构及索引策略详见《支付事务数据库设计说明》(DBDD-PAY-001) 第 3 章
b. 商品分类数据字典参见附录 B《商品标签编码规范》

2.第三方服务依赖

c. 支付宝接口 SDK 版本：alipay-sdk-java-4.38.10.ALL
d. 山东大学认证系统 VPN 配置模板见团队内部 Wiki《VPN 接入指南》

3.性能测试数据

e. 支付接口压力测试报告（2024-05-15）：
    i. 平均响应时间：1.2s（95%分位）
    ii. 最大并发量：128 请求/秒

4.兼容性说明

f. 前端框架最低兼容要求：
    i. Vue.js 3.2+ / React 18+
    ii. 浏览器支持：Chrome ≥85, Edge ≥90, Firefox ≥78

5.安全审计记录

g. 加密算法通过山东大学信息安全实验室 FIPS 140-2 验证
h. 漏洞扫描报告编号：SEC-AUDIT-2024-036（2024 年第二季度）

6.部署注意事项

i. JDK 必须使用 Azul Zulu 11.0.22+版本（其他版本存在 TLS 1.3 兼容性问题）
j. Redis 需配置持久化策略：AOF 模式+每秒同步

7.扩展引用

k. 微服务监控方案参见《Prometheus 监控配置手册》
l. 日志分析规则库地址：git@internal:log-rules/cctp.git

附录

附录可用来提供那些为便于文档维护而单独出版的信息(例如图表、分类数据)。为便于处理，附录可单独装订成册。附录应按字母顺序(A, B 等)编排。