面向对象程序课程设计报告2

一、项目需求的进一步理解和描述

1.1 项目背景与目标

基于第一次报告的基础需求，经过深入的技术调研和实际开发，我对项目需求有了更加清晰和深入的理解。本项目旨在构建一个完整的电商交易平台，集成多层架构设计、高性能网络通信、面向对象设计模式等现代软件工程技术。

1.2 功能需求细化

1.2.1 核心业务功能

- 用户管理系统: 包含用户注册、登录、权限管理、会话管理等功能

- 商品管理系统: 商品操作、分类管理、库存控制、搜索功能

- 购物车系统: 购物车增删改查、批量操作、结算功能

- 订单管理系统: 订单创建、状态跟踪、支付集成、售后服务

- 权限管理系统: 基于角色的访问控制、多级权限验证

- 数据分析系统: 销售统计、用户行为分析、系统监控

1.2.2 非功能性需求

- 性能要求: 支持高并发访问，系统响应时间<200ms

- 可扩展性: 模块化设计，支持功能动态扩展

- 安全性: 数据加密传输，SQL注入防护，权限验证

- 可维护性: 清晰的代码结构，完整的文档体系

- 可用性: 友好的用户界面，完善的错误处理

1.3 技术需求

1.3.1 架构需求

- 多层架构: 表现层、业务层、数据层分离

- 跨语言集成: Java、C++、SQL的无缝集成

- 网络通信: 基于Netty的高性能TCP/IP通信

- 数据持久化: MySQL数据库的高效访问

1.3.2 质量需求

- 代码质量: 遵循面向对象设计原则，应用设计模式

- 测试覆盖: 单元测试、集成测试、性能测试

二、相关技术讨论及解决方案

2.1 C/S结构与B/S结构

2.1.1 C/S结构实现

本项目采用C/S架构作为主要架构模式：

技术选择:

- 客户端: Java Netty Client，提供原生性能和丰富的UI交互

- 服务端: Java Netty Server + C++ JNI，兼顾开发效率和执行性能

优势:

- 高性能：原生代码执行，减少网络延迟

- 功能丰富：可实现复杂的客户端逻辑

- 安全性：减少数据在网络中的暴露

实现方案:

采用Netty客户端架构，通过ChannelPipeline处理网络通信。

2.1.2 B/S结构扩展

项目具备向B/S架构扩展的能力：

扩展方案:

- 基于现有JNI接口构建RESTful API

- 使用Spring Boot框架快速搭建Web服务

- 前端可使用Vue.js或React构建管理后台

2.2 MVC框架

2.2.1 MVC模式应用

项目严格遵循MVC设计模式：

Model层 (数据模型):

实现用户、商品、订单等核心业务实体类。

View层 (表现层):

- 控制台交互界面 (AdminConsole, UserConsole)

- 网络协议消息格式

- 未来的客户端界面

Controller层 (控制层):

- EmshopNettyServer: 网络请求控制

- PermissionService: 权限控制

- 各种Service类: 业务逻辑控制

2.2.2 MVC优势体现

- 职责分离: 业务逻辑与表现逻辑分离

- 代码复用: Model层可被多个View共享

- 易于维护: 修改UI不影响业务逻辑

- 测试友好: 各层可独立进行单元测试

2.3 持久层技术

2.3.1 数据持久化方案

采用原生SQL + 连接池的持久化方案：

技术栈:

- 数据库: MySQL 8.0+

- 连接技术: MySQL Connector/C++

- 连接池: 自实现的DatabaseConnectionPool

- 数据格式: JSON数据交换

持久层架构:

采用C++实现的DatabaseConnectionPool连接池管理，支持事务处理和SQL优化。

2.3.2 持久层优势

- 高性能: 原生C++实现，无ORM开销

- 连接复用: 连接池减少连接创建开销

- 事务支持: 完整的ACID事务特性

- SQL优化: 预编译语句，防止SQL注入

2.4 设计模式

2.4.1 单例模式

应用场景: 全局唯一资源管理

实现数据库连接池的单例模式，确保全局唯一实例。

2.4.2 工厂模式

应用场景: 服务对象创建管理

通过EmshopServiceManager统一管理各种Service对象的创建。

2.4.3 模板方法模式

应用场景: 统一的业务处理流程

定义BaseService抽象类，统一业务处理流程。

2.4.4 策略模式

应用场景: 权限验证策略

PermissionService采用策略模式实现灵活的权限控制。

2.5.1 Web服务架构

基于现有JNI接口构建RESTful API，使用Spring Boot框架提供Web服务支持。

2.6 并发和互斥

2.6.1 并发控制机制

Java层并发控制:

- Netty的事件循环机制处理I/O并发

- ConcurrentHashMap管理用户会话

- 线程安全的消息队列

C++层并发控制:

使用mutex和condition\_variable实现线程安全的数据库连接池管理。

2.6.2 互斥场景处理

库存互斥:

使用atomic原子操作实现商品库存的并发安全更新。

2.7 JavaScript技术

2.7.1 JavaScript前端交互技术

JavaScript在用户体验优化中的核心作用：

前端架构设计思路:

- 模块化JavaScript开发，职责清晰分离

- 事件驱动的用户交互处理

- 异步数据加载，提升页面响应速度

- 客户端数据验证，减少服务器压力

JavaScript应用场景:

- 动态表单验证: 实时检查输入合法性

- 异步数据更新: 无刷新页面内容更新

- 购物车实时计算: 动态计算总价和优惠

- 搜索自动补全: 提升用户搜索体验

- 实时通知系统: WebSocket消息推送

Vue.js单页面应用架构:

- 渐进式框架特性，可与现有Java后端无缝集成

- 组件化开发模式，将电商功能模块化设计

- 商品展示组件: ProductList.vue、ProductCard.vue、ProductDetail.vue

- 购物车组件: ShoppingCart.vue、CartItem.vue、CartSummary.vue

- 用户管理组件: UserProfile.vue、LoginForm.vue、RegisterForm.vue

- 订单管理组件: OrderList.vue、OrderDetail.vue、OrderStatus.vue

- Vuex状态管理，集中管理用户状态、购物车状态、商品状态

- Vue Router路由管理，实现单页面应用的页面切换

- Axios HTTP客户端，与Java Netty服务器进行RESTful API通信

响应式UI框架整合:

- Element Plus (Vue) / Ant Design (React) 企业级组件

- 响应式商品网格布局，自适应不同屏幕尺寸

- PWA渐进式Web应用，支持推送通知

2.7.2Web技术栈整合方案

完整Web解决方案的技术思路：

技术栈协调:

- JavaScript实现客户端交互逻辑

- 与现有Netty服务器协同工作

界面设计理念:

- 响应式设计适配多终端

- 渐进式Web应用提升体验

- 组件化开发提高复用性

- 主题切换满足个性化需求

集成优势:

- 统一的技术栈便于维护

- 丰富的UI交互提升用户体验

- 标准化开发流程

- 良好的可扩展性

2.8 同步和异步通信

2.8.1 Netty异步通信

采用Netty的异步I/O模型，使用CompletableFuture实现异步请求处理。

2.8.2 JNI同步调用

JNI接口采用同步调用模式：

- 保证数据一致性

- 简化错误处理

- 便于事务管理

混合模式优势:

- 网络I/O异步处理，提高并发能力

- 数据库操作同步处理，保证事务完整性

三、前端和后端概要设计

3.1 模块划分

3.1.1 后端模块架构

服务端架构 (EmshopNettyServer)

├── 网络层

│ ├── Netty服务器

│ ├── 连接管理

│ └── 协议处理

├── 业务层

│ ├── 用户服务

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 商品服务

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 订单服务

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ └── 权限服务

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

├── 数据层

│ ├── JNI接口

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ └── 数据库连接池

└── 工具层

├── 日志管理

├── 配置管理

└── 缓存管理

3.1.2 前端模块架构

客户端架构

├── 界面层

│ ├── 管理员控制台

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 用户购物界面

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ └── 网络客户端

├── 业务层 (Business Layer)

│ ├── 用户管理

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 商品管理

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 购物车管理

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ └── 订单管理

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

├── 网络层 (Network Layer)

│ ├── 网络管理器

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ ├── 消息处理器

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

│ └── 连接管理器

│ ├── C++实现 (emshop\_native\_impl\_oop.cpp)

└── 数据层

├── 本地缓存

├── 会话管理

└── 数据模型

3.2 UI设计（大体功能一览，待完善）

3.2.1 管理员控制台界面设计

=== Emshop管理员控制台 ===（极少数功能实现待完善）

[1] 商品管理

[1.1] 查看所有商品

[1.2] 添加新商品

[1.3] 搜索商品

[1.4] 编辑商品

[1.5] 删除商品

[1.6] 库存管理

[2] 用户管理

[2.1] 查看所有用户

[2.2] 搜索用户

[2.3] 角色管理

[2.4] 封禁/解封用户

[3] 订单管理

[3.1] 查看所有订单

[3.2] 按状态筛选

[3.3] 订单详情

[3.4] 更新订单状态

[4] 系统管理

[4.1] 系统统计

[4.2] 系统状态

[4.3] 日志查看

[4.4] 缓存管理

3.2.2 用户购物界面设计

=== Emshop用户购物平台 ===

[1] 商品浏览

[1.1] 查看所有商品

[1.2] 按分类浏览

[1.3] 搜索商品

[1.4] 商品详情

[1.5] 查看评价

[2] 购物车管理

[2.1] 查看购物车

[2.2] 添加商品

[2.3] 修改数量

[2.4] 移除商品

[2.5] 结算购买

[3] 订单中心

[3.1] 我的订单

[3.2] 订单详情

[3.3] 取消订单

[3.4] 确认收货

[4] 个人中心

[4.1] 个人信息

[4.2] 地址管理

[4.3] 优惠券

[4.4] 我的评价

3.2.3 UI设计原则

- 功能完整: 覆盖所有业务场景

- 错误处理: 详细的错误提示和操作指导

- 响应及时: 操作反馈及时，状态显示清晰

3.3 主要类的划分

3.3.1 服务端核心类

主要包括EmshopNettyServer服务器类、UserSession会话类、PermissionService权限类等。

3.3.2 客户端核心类

主要包括EmshopNettyClient客户端类、AdminConsole管理员控制台类、UserConsole用户控制台类等。

3.3.3 JNI接口类

EmshopNativeInterface类提供系统管理、用户管理、商品管理、购物车管理、订单管理等JNI接口方法。

3.4 数据库设计

3.4.1 核心表结构

-- 用户表

CREATE TABLE users (

user\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

phone VARCHAR(20),

email VARCHAR(100),

role ENUM('admin', 'vip', 'user') DEFAULT 'user',

status ENUM('active', 'banned', 'inactive') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- 商品表

CREATE TABLE products (

product\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(200) NOT NULL,

description TEXT,

price DECIMAL(10,2) NOT NULL,

stock\_quantity INT DEFAULT 0,

category VARCHAR(100),

brand VARCHAR(100),

image\_url VARCHAR(500),

status ENUM('active', 'inactive', 'deleted') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- 订单表

CREATE TABLE orders (

order\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id BIGINT NOT NULL,

total\_amount DECIMAL(12,2) NOT NULL,

status ENUM('pending', 'paid', 'shipped', 'delivered', 'cancelled', 'refunded') DEFAULT 'pending',

shipping\_address\_id BIGINT,

coupon\_code VARCHAR(50),

discount\_amount DECIMAL(10,2) DEFAULT 0.00,

payment\_method VARCHAR(50),

remark TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

-- 订单明细表

CREATE TABLE order\_items (

item\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

order\_id BIGINT NOT NULL,

product\_id BIGINT NOT NULL,

quantity INT NOT NULL,

unit\_price DECIMAL(10,2) NOT NULL,

total\_price DECIMAL(10,2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

-- 购物车表

CREATE TABLE shopping\_cart (

cart\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id BIGINT NOT NULL,

product\_id BIGINT NOT NULL,

quantity INT NOT NULL,

added\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id),

UNIQUE KEY uk\_user\_product (user\_id, product\_id)

);

-- 用户地址表

CREATE TABLE user\_addresses (

address\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id BIGINT NOT NULL,

receiver\_name VARCHAR(100) NOT NULL,

receiver\_phone VARCHAR(20) NOT NULL,

province VARCHAR(50) NOT NULL,

city VARCHAR(50) NOT NULL,

district VARCHAR(50) NOT NULL,

detail\_address VARCHAR(500) NOT NULL,

postal\_code VARCHAR(10),

is\_default BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

3.4.2 扩展表结构

-- 商品评价表

CREATE TABLE product\_reviews (

review\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id BIGINT NOT NULL,

product\_id BIGINT NOT NULL,

order\_id BIGINT NOT NULL,

rating INT NOT NULL CHECK (rating >= 1 AND rating <= 5),

content TEXT,

is\_anonymous BOOLEAN DEFAULT FALSE,

status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

-- 优惠券表

CREATE TABLE coupons (

coupon\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,

code VARCHAR(50) UNIQUE NOT NULL,

name VARCHAR(200) NOT NULL,

type ENUM('fixed', 'percentage') NOT NULL,

value DECIMAL(10,2) NOT NULL,

min\_amount DECIMAL(10,2) DEFAULT 0.00,

max\_discount DECIMAL(10,2),

total\_quantity INT NOT NULL,

used\_quantity INT DEFAULT 0,

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

status ENUM('active', 'inactive', 'expired') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 用户优惠券关联表

CREATE TABLE user\_coupons (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id BIGINT NOT NULL,

coupon\_id BIGINT NOT NULL,

status ENUM('unused', 'used', 'expired') DEFAULT 'unused',

obtained\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

used\_at TIMESTAMP NULL,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (coupon\_id) REFERENCES coupons(coupon\_id)

);

3.4.3 数据库设计原则

- 规范化: 遵循第三范式，避免数据冗余

- 索引优化: 主键、外键、查询字段建立索引

- 约束完整: 外键约束、检查约束、唯一约束

- 扩展性: 预留扩展字段，支持业务增长

3.5 通信协议设计

3.5.1 协议层次结构

应用层: JSON格式业务数据

├── 请求类型: LOGIN, GET\_PRODUCTS, ADD\_TO\_CART, etc.

├── 参数数据: 业务相关参数

└── 响应格式: 统一的JSON响应结构

传输层: Netty TCP/IP

├── 消息分隔: 换行符分隔 (\n)

├── 字符编码: UTF-8编码

└── 连接管理: 长连接复用

物理层: 网络套接字

├── 协议: TCP/IP

├── 端口: 8080 (可配置)

└── 连接: 客户端-服务器长连接

3.5.2 消息格式定义

请求消息格式:

COMMAND PARAM1 PARAM2 PARAM3 ...

响应消息格式:

{

"success": true,

"message": "操作成功",

"data": {

// 具体的响应数据

},

"error\_code": 0,

"timestamp": 1684567890123

}

3.5.3 消息类型定义

定义了完整的请求消息类型枚举，包括用户认证(LOGIN、REGISTER等)、商品管理(GET\_PRODUCTS、ADD\_PRODUCT等)、购物车管理(ADD\_TO\_CART、GET\_CART等)、订单管理(CREATE\_ORDER、GET\_ORDERS等)、系统管理(PING、INIT等)各类操作。

3.5.4 协议安全性设计

- 会话管理: 基于Channel的用户会话跟踪

- 权限验证: 每个请求进行权限检查

- 参数验证: 输入参数格式和范围验证

- 错误处理: 统一的错误响应格式

3.6 主要逻辑流程图

3.6.1 用户登录流程图

客户端发起登录请求 -> 服务器接收请求 -> 解析用户名和密码 -> 调用JNI登录接口 ->

C++验证用户凭据 -> 验证成功? -> 是: 创建用户会话 -> 返回登录成功信息 -> 客户端更新登录状态

-> 否: 返回登录失败 -> 客户端显示错误信息

3.6.2 商品购买流程图

浏览商品列表 -> 选择商品详情 -> 添加到购物车 -> 查看购物车 -> 选择收货地址 ->

选择优惠券 -> 确认订单信息 -> 创建订单 -> 库存充足? -> 是: 扣减库存 -> 生成订单 ->

清空购物车 -> 返回订单信息

-> 否: 返回库存不足 -> 提示用户调整数量

3.6.3 权限验证流程图

接收客户端请求 -> 解析请求类型 -> 需要权限验证? -> 否: 直接处理请求 -> 调用JNI业务接口

-> 是: 获取用户会话 -> 用户已登录? ->

否: 返回未登录错误

是: 检查操作权限 -> 权限充足? -> 否: 返回权限不足错误

-> 是: 调用JNI业务接口 -> C++处理业务逻辑 -> 返回处理结果

3.6.4 订单状态流转图

pending(待支付) -> paid(已支付) -> shipped(已发货) -> delivered(已送达) -> completed(已完成)

pending(待支付) -> cancelled(已取消)

paid(已支付) -> refunded(已退款)

shipped(已发货) -> refunded(已退款)

delivered(已送达) -> after\_sale(售后) -> refunded(已退款)

delivered(已送达) -> after\_sale(售后) -> completed(已完成)

四、其他：

4.1 项目目前的技术亮点

4.1.1 多语言混合架构

在保证程序的主要功能由cpp实现的前提下，本项目成功实现了Java与C++的深度集成，通过JNI技术将Java的跨平台特性与C++的高性能特性完美结合。这种架构在保证开发效率的同时，确保了系统的执行性能。

4.1.2 面向对象设计的深度应用

项目严格遵循面向对象设计原则，广泛应用了多种设计模式：

- 单例模式: 全局资源管理，确保系统资源的统一性

- 工厂模式: 对象创建管理，提供灵活的对象生成策略

- 模板方法模式: 统一业务流程，提高代码复用性

- 策略模式: 权限验证策略，支持灵活的权限控制

4.1.3 高性能网络通信

基于Netty框架实现的网络通信层，具有以下优势：

- 高并发支持: 事件驱动的异步I/O模型

- 低延迟: 零拷贝技术和高效的内存管理

- 可扩展性: 支持大量并发连接

- 稳定性: 完善的异常处理和连接管理

4.2 未来扩展方向

4.2.1 技术架构优化

- 微服务架构: 将单体应用拆分为微服务

- 容器化部署: 使用Docker进行应用容器化

- 云原生改造: 适配Kubernetes等云原生平台

- 分布式缓存: 引入Redis等分布式缓存系统

4.2.2 功能特性增强

- 移动端支持: 开发Android/iOS移动应用

- 实时通信: 集成WebSocket实现实时消息推送

- 大数据分析: 集成数据分析和商业智能

- 人工智能: 引入推荐系统和智能客服

4.2.3 性能和安全优化

- 性能监控: 集成APM性能监控系统

- 安全加固: 实现HTTPS、OAuth2.0等安全机制

- 负载均衡: 实现高可用和负载均衡

- 数据备份: 完善的数据备份和灾难恢复

具体进度可了解：https://github.com/AshIgnis/JLU\_Emshop\_System

55240425屈熙宸

2025年9月10日