面向对象程序课程设计报告3

一、前端和后端详细设计

（1）模块划分及模块间关系

1.1 后端模块架构

服务端采用分层架构设计：

网络通信层（Network Layer）

├── Java Netty服务器 (EmshopNettyServer.java)

│ ├── 端口：8080

│ ├── 协议：TCP/IP + 文本协议

│ ├── 消息分隔：换行符(\n)

│ └── 连接管理：长连接复用

├── 会话管理 (UserSession)

│ ├── 用户状态跟踪

│ ├── 权限验证

│ └── 并发控制

└── 协议处理器 (MessageHandler)

├── 命令解析

├── 参数验证

└── 响应封装

业务逻辑层（Business Layer）

├── JNI接口层 (EmshopNativeInterface.java)

│ ├── 系统管理接口

│ ├── 用户管理接口

│ ├── 商品管理接口

│ ├── 购物车管理接口

│ └── 订单管理接口

└── C++实现层 (emshop\_native\_impl\_oop.cpp)

├── EmshopServiceManager (服务管理器)

├── UserService (用户服务)

├── ProductService (商品服务)

├── CartService (购物车服务)

├── OrderService (订单服务)

├── AddressService (地址服务)

├── CouponService (优惠券服务)

└── ReviewService (评论服务)

数据访问层（Data Access Layer）

├── 数据库连接池 (DatabaseConnectionPool)

│ ├── 连接管理：5个连接

│ ├── 线程安全：mutex保护

│ ├── 自动重连：MYSQL\_OPT\_RECONNECT

│ └── 资源管理：RAII模式

├── 数据库配置 (DatabaseConfig)

│ ├── 连接参数管理

│ ├── 配置文件解析

│ └── 环境变量支持

└── SQL执行器

├── 预编译语句

├── 事务支持

└── 异常处理

1.2 前端模块架构

Qt客户端采用MVC架构：

表现层（Presentation Layer）

├── 主窗口 (MainWindow)

│ ├── 选项卡管理

│ ├── 状态栏显示

│ ├── 菜单栏功能

│ └── 工具栏操作

├── 产品列表界面 (ProductListWidget)

│ ├── 商品展示网格

│ ├── 分类筛选

│ ├── 搜索功能

│ └── 商品详情弹窗

├── 购物车界面 (CartWidget)

│ ├── 购物车商品列表

│ ├── 数量修改操作

│ ├── 价格计算显示

│ └── 结算流程管理

├── 订单界面 (OrderWidget)

│ ├── 订单列表展示

│ ├── 订单状态筛选

│ ├── 订单详情查看

│ └── 订单操作按钮

└── 登录对话框 (LoginDialog)

├── 服务器连接配置

├── 用户凭据输入

├── 连接状态显示

└── 错误消息提示

网络通信层（Network Layer）

├── TCP客户端 (EmshopTcpClient)

│ ├── QTcpSocket连接管理

│ ├── 心跳机制：30秒间隔

│ ├── 消息收发处理

│ └── 连接状态管理

├── 客户端适配器 (ClientAdapter)

│ ├── 信号转发机制

│ ├── 接口统一封装

│ ├── 类型转换处理

│ └── 兼容性保障

└── 协议处理

├── 文本命令发送

├── JSON响应解析

├── 错误处理机制

└── 消息队列管理

1.3 模块间关系

前端Qt客户端 ←→ 网络TCP连接 ←→ Java Netty服务器 ←→ JNI调用 ←→ C++业务层 ←→ 数据库MySQL

关系特点：

- 松耦合：模块间通过接口通信，便于独立开发和测试

- 分层清晰：每层负责特定职责，便于维护和扩展

- 异步处理：网络通信采用异步模式，提高并发性能

（2）UI流程设计

2.1 用户登录流程

启动客户端 → 显示登录对话框 → 输入服务器地址(localhost:8080) → 点击连接 →

TCP连接建立 → 输入用户名密码 → 发送LOGIN命令 → 服务器验证 → 返回JSON响应 →

登录成功：创建主窗口 → 显示欢迎消息 → 进入主界面

登录失败：显示错误提示 → 重新输入

2.2 商品浏览购买流程

主界面加载 → 发送GET\_PRODUCTS命令 → 服务器返回商品列表 → 商品网格显示 →

用户浏览商品 → 点击商品详情 → 弹出详情对话框 → 设置购买数量 →

点击"加入购物车" → 发送ADD\_TO\_CART命令 → 服务器处理 → 返回操作结果 →

UI更新购物车状态 → 显示成功消息

2.3 订单管理流程

切换到订单页面 → 发送GET\_ORDERS命令 → 服务器返回订单列表 → 订单树形显示 →

用户选择状态筛选 → 更新显示列表 → 双击查看订单详情 → 弹出详情窗口 →

可选操作：支付/取消/确认收货 → 发送相应命令 → 服务器处理 → 状态更新 → UI刷新

2.4 UI设计原则

- 直观易用：界面布局清晰，操作流程符合用户习惯

- 响应及时：网络操作显示进度条，状态变化实时反馈

- 错误友好：详细的错误提示信息，指导用户正确操作

- 一致性：整体风格统一，按钮样式和颜色主题保持一致

（3）类的设计及说明

3.1 服务端核心类设计

EmshopNettyServer类：

- 继承：无

- 职责：TCP服务器管理，客户端连接处理，消息分发

- 主要方法：

\* start(): 启动服务器

\* shutdown(): 关闭服务器

\* handleMessage(): 处理客户端消息

\* manageSession(): 管理用户会话

- 设计模式：单例模式

- 线程安全：是

EmshopServiceManager类：

- 继承：无

- 职责：统一管理各种业务服务，提供服务工厂功能

- 主要方法：

\* getInstance(): 获取单例实例

\* initializeServices(): 初始化所有服务

\* getUserService(): 获取用户服务实例

\* getProductService(): 获取商品服务实例

- 设计模式：单例模式 + 工厂模式

- 线程安全：是

UserService类：

- 继承：BaseService

- 职责：用户管理业务逻辑，包括注册、登录、信息管理

- 主要方法：

\* authenticateUser(): 用户认证

\* registerUser(): 用户注册

\* updateProfile(): 更新用户信息

\* getUserById(): 根据ID获取用户信息

- 设计模式：模板方法模式

- 线程安全：是

ProductService类：

- 继承：BaseService

- 职责：商品管理业务逻辑，包括商品CRUD操作

- 主要方法：

\* getProducts(): 获取商品列表

\* searchProducts(): 搜索商品

\* addProduct(): 添加商品

\* updateStock(): 更新库存

- 设计模式：模板方法模式

- 线程安全：是

DatabaseConnectionPool类：

- 继承：无

- 职责：数据库连接池管理，提供高效的数据库连接

- 主要方法：

\* getInstance(): 获取连接池实例

\* getConnection(): 获取数据库连接

\* releaseConnection(): 释放连接

\* initialize(): 初始化连接池

- 设计模式：单例模式 + 对象池模式

- 线程安全：是，使用std::mutex

3.2 客户端核心类设计

MainWindow类：

- 继承：QMainWindow

- 职责：主界面管理，选项卡控制，状态显示

- 主要方法：

\* setEmshopClient(): 设置客户端适配器

\* setupUI(): 初始化界面布局

\* onConnectionStateChanged(): 处理连接状态变化

\* onAuthenticated(): 处理认证成功事件

- 设计模式：观察者模式

- 线程安全：否（主线程UI）

EmshopTcpClient类：

- 继承：QObject

- 职责：TCP网络通信，协议处理，连接管理

- 主要方法：

\* connectToServer(): 连接服务器

\* authenticate(): 用户认证

\* sendTextCommand(): 发送文本命令

\* handleTextResponse(): 处理文本响应

- 设计模式：观察者模式 + 状态模式

- 线程安全：是（Qt信号槽机制）

ClientAdapter类：

- 继承：QObject

- 职责：适配器模式实现，桥接TCP客户端和UI组件

- 主要方法：

\* getProducts(): 获取商品列表

\* addToCart(): 添加购物车

\* createOrder(): 创建订单

\* disconnectFromServer(): 断开连接

- 设计模式：适配器模式 + 代理模式

- 线程安全：是

ProductListWidget类：

- 继承：QWidget

- 职责：商品列表展示，商品搜索，商品操作

- 主要方法：

\* setEmshopClient(): 设置客户端

\* loadProducts(): 加载商品数据

\* setupUI(): 初始化界面

\* onProductsReceived(): 处理商品数据接收

- 设计模式：观察者模式 + MVC模式

- 线程安全：否（主线程UI）

（4）数据库详细设计及说明

4.1 核心表结构设计

用户表(users)：

字段设计：

- user\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (用户ID，主键)

- username: VARCHAR(50) UNIQUE NOT NULL (用户名，唯一)

- password: VARCHAR(255) NOT NULL (密码，加密存储)

- phone: VARCHAR(20) (手机号)

- email: VARCHAR(100) (邮箱)

- role: ENUM('admin', 'vip', 'user') DEFAULT 'user' (用户角色)

- status: ENUM('active', 'banned', 'inactive') DEFAULT 'active' (用户状态)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

- updated\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP (更新时间)

索引设计：

- PRIMARY KEY (user\_id)

- UNIQUE KEY uk\_username (username)

- INDEX idx\_email (email)

- INDEX idx\_phone (phone)

商品表(products)：

字段设计：

- product\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (商品ID，主键)

- name: VARCHAR(200) NOT NULL (商品名称)

- description: TEXT (商品描述)

- price: DECIMAL(10,2) NOT NULL (商品价格)

- stock\_quantity: INT DEFAULT 0 (库存数量)

- category: VARCHAR(100) (商品分类)

- brand: VARCHAR(100) (商品品牌)

- image\_url: VARCHAR(500) (商品图片URL)

- status: ENUM('active', 'inactive', 'deleted') DEFAULT 'active' (商品状态)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

- updated\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP (更新时间)

索引设计：

- PRIMARY KEY (product\_id)

- INDEX idx\_category (category)

- INDEX idx\_name (name)

- INDEX idx\_price (price)

- INDEX idx\_status (status)

订单表(orders)：

字段设计：

- order\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (订单ID，主键)

- user\_id: BIGINT NOT NULL (用户ID，外键)

- total\_amount: DECIMAL(12,2) NOT NULL (订单总金额)

- status: ENUM('pending', 'paid', 'shipped', 'delivered', 'cancelled', 'refunded') DEFAULT 'pending' (订单状态)

- shipping\_address\_id: BIGINT (收货地址ID)

- coupon\_code: VARCHAR(50) (优惠券代码)

- discount\_amount: DECIMAL(10,2) DEFAULT 0.00 (优惠金额)

- payment\_method: VARCHAR(50) (支付方式)

- remark: TEXT (订单备注)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

- updated\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP (更新时间)

外键约束：

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

- FOREIGN KEY (shipping\_address\_id) REFERENCES user\_addresses(address\_id)

订单明细表(order\_items)：

字段设计：

- item\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (明细ID，主键)

- order\_id: BIGINT NOT NULL (订单ID，外键)

- product\_id: BIGINT NOT NULL (商品ID，外键)

- quantity: INT NOT NULL (购买数量)

- unit\_price: DECIMAL(10,2) NOT NULL (单价)

- total\_price: DECIMAL(10,2) NOT NULL (小计金额)

外键约束：

- FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE CASCADE

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

购物车表(shopping\_cart)：

字段设计：

- cart\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (购物车ID，主键)

- user\_id: BIGINT NOT NULL (用户ID，外键)

- product\_id: BIGINT NOT NULL (商品ID，外键)

- quantity: INT NOT NULL (商品数量)

- added\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (添加时间)

唯一约束：

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

4.2 扩展表结构

用户地址表(user\_addresses)：

字段设计：

- address\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (地址ID)

- user\_id: BIGINT NOT NULL (用户ID)

- receiver\_name: VARCHAR(100) NOT NULL (收货人姓名)

- receiver\_phone: VARCHAR(20) NOT NULL (收货人电话)

- province: VARCHAR(50) NOT NULL (省份)

- city: VARCHAR(50) NOT NULL (城市)

- district: VARCHAR(50) NOT NULL (区县)

- detail\_address: VARCHAR(500) NOT NULL (详细地址)

- postal\_code: VARCHAR(10) (邮政编码)

- is\_default: BOOLEAN DEFAULT FALSE (是否默认地址)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

优惠券表(coupons)：

字段设计：

- coupon\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (优惠券ID)

- code: VARCHAR(50) UNIQUE NOT NULL (优惠券代码)

- name: VARCHAR(200) NOT NULL (优惠券名称)

- type: ENUM('fixed', 'percentage') NOT NULL (优惠类型)

- value: DECIMAL(10,2) NOT NULL (优惠值)

- min\_amount: DECIMAL(10,2) DEFAULT 0.00 (最小使用金额)

- max\_discount: DECIMAL(10,2) (最大优惠金额)

- total\_quantity: INT NOT NULL (总数量)

- used\_quantity: INT DEFAULT 0 (已使用数量)

- start\_time: TIMESTAMP NOT NULL (开始时间)

- end\_time: TIMESTAMP NOT NULL (结束时间)

- status: ENUM('active', 'inactive', 'expired') DEFAULT 'active' (状态)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

商品评价表(product\_reviews)：

字段设计：

- review\_id: BIGINT PRIMARY KEY AUTO\_INCREMENT (评价ID)

- user\_id: BIGINT NOT NULL (用户ID)

- product\_id: BIGINT NOT NULL (商品ID)

- order\_id: BIGINT NOT NULL (订单ID)

- rating: INT NOT NULL CHECK (rating >= 1 AND rating <= 5) (评分)

- content: TEXT (评价内容)

- is\_anonymous: BOOLEAN DEFAULT FALSE (是否匿名)

- status: ENUM('pending', 'approved', 'rejected') DEFAULT 'pending' (审核状态)

- created\_at: TIMESTAMP DEFAULT CURRENT\_TIMESTAMP (创建时间)

4.3 数据库设计原则

规范化原则：

- 遵循第三范式(3NF)，消除数据冗余

- 合理使用外键约束，保证引用完整性

- 避免过度规范化，平衡性能和存储

性能优化：

- 主键和外键自动创建索引

- 频繁查询字段建立复合索引

- 分页查询优化，使用LIMIT和OFFSET

- 避免全表扫描，合理使用WHERE条件

数据完整性：

- 主键约束：保证记录唯一性

- 外键约束：维护表间关系完整性

- 检查约束：限制字段值范围

- 非空约束：保证关键字段不为空

（5）通信协议详细设计

5.1 协议层次架构

物理层：TCP/IP协议

- 传输协议：TCP

- 端口：8080（可配置）

- 连接类型：长连接

- 字符编码：UTF-8

传输层：Netty框架

应用层：文本+JSON协议

- 请求格式：纯文本命令

- 响应格式：JSON结构化数据

- 消息类型：命令字+参数

- 错误处理：统一错误码体系

5.2 消息格式定义

请求消息格式：

COMMAND PARAM1 PARAM2 PARAM3 ...\n

示例：

LOGIN admin 123456\n

GET\_PRODUCTS electronics 1 10\n

ADD\_TO\_CART 1001 2\n

响应消息格式：

{

"success": boolean, // 操作是否成功

"message": "string", // 操作结果描述

"data": { // 具体业务数据

// 根据不同操作返回不同结构

},

"error\_code": integer, // 错误代码(成功为0)

"timestamp": long // 时间戳

}

登录成功响应示例：

{

"success": true,

"message": "登录成功",

"data": {

"user\_info": {

"user\_id": 1,

"username": "admin",

"role": "admin"

}

},

"error\_code": 0,

"timestamp": 1695020430000

}

商品列表响应示例：

{

"success": true,

"message": "获取商品列表成功",

"data": {

"products": [

{

"product\_id": 1001,

"name": "iPhone 15",

"price": 5999.00,

"stock\_quantity": 50,

"category": "electronics",

"description": "最新款iPhone"

}

],

"total\_count": 100,

"current\_page": 1,

"page\_size": 10

},

"error\_code": 0,

"timestamp": 1695020430000

}

5.3 命令类型定义

用户认证类命令：

- LOGIN username password : 用户登录

- REGISTER username password phone email : 用户注册

- LOGOUT : 用户登出

- CHANGE\_PASSWORD old\_password new\_password : 修改密码

商品管理类命令：

- GET\_PRODUCTS [category] [page] [pageSize] : 获取商品列表

- SEARCH\_PRODUCTS keyword [page] [pageSize] : 搜索商品

- GET\_PRODUCT\_DETAIL product\_id : 获取商品详情

- ADD\_PRODUCT name price stock category description : 添加商品（管理员）

- UPDATE\_PRODUCT product\_id name price stock : 更新商品（管理员）

- DELETE\_PRODUCT product\_id : 删除商品（管理员）

购物车管理类命令：

- GET\_CART : 获取购物车

- ADD\_TO\_CART product\_id quantity : 添加到购物车

- UPDATE\_CART\_QUANTITY product\_id quantity : 更新购物车数量

- REMOVE\_FROM\_CART product\_id : 从购物车移除

- CLEAR\_CART : 清空购物车

订单管理类命令：

- CREATE\_ORDER address\_id [coupon\_code] [remark] : 创建订单

- GET\_ORDERS [status] [page] [pageSize] : 获取订单列表

- GET\_ORDER\_DETAIL order\_id : 获取订单详情

- CANCEL\_ORDER order\_id reason : 取消订单

- CONFIRM\_ORDER order\_id : 确认收货

系统管理类命令：

- PING : 心跳检测

- GET\_SYSTEM\_STATUS : 获取系统状态（管理员）

- GET\_USER\_LIST [page] [pageSize] : 获取用户列表（管理员）

5.4 协议安全性设计

会话管理：

- 基于Channel ID的会话跟踪

- 用户登录后创建UserSession对象

- 会话超时自动清理机制

- 并发会话控制

权限验证：

- 每个命令执行前进行权限检查

- 基于角色的访问控制(RBAC)

- 管理员权限和普通用户权限分离

- 敏感操作需要二次验证

参数验证：

- 命令参数数量验证

- 参数格式验证(数字、字符串长度等)

- 业务规则验证(库存充足性、用户状态等)

- SQL注入防护

错误处理：

- 统一的错误码体系

- 详细的错误信息描述

- 异常情况日志记录

- 客户端友好的错误提示

二、对于技术的迭代和架构的迭代的说明

2.1 技术栈演进

初始设计（第一周）：

- 服务端：纯C++实现

- 客户端：控制台程序

- 通信：简单TCP套接字

- 数据库：直接SQL调用

第二周优化：

- 服务端：Java Netty + C++ JNI混合架构

- 客户端：增加Java Netty客户端

- 通信：基于Netty的高性能通信

- 数据库：连接池+事务支持

第三周完善（当前）：

- 服务端：保持Java Netty + C++ JNI架构

- 客户端：Qt5 + TCP客户端替代Java客户端

- 通信：文本协议替代JSON协议，提高兼容性

- 数据库：完整的面向对象设计，支持复杂业务

2.2 架构迭代说明

网络通信架构变更：

原设计：WebSocket协议 + SSL加密

现实施：TCP协议 + 文本命令

变更原因：

- WebSocket需要HTTP升级握手，增加复杂性

- SSL证书配置在开发环境中遇到困难

- 文本协议更直观，便于调试和扩展

- TCP协议性能更高，延迟更低

客户端技术选型调整：

原计划：Java Netty客户端

现实施：Qt5 C++客户端

调整原因：

- Qt提供更专业的桌面UI体验

- C++性能优于Java客户端

- Qt信号槽机制更适合事件驱动编程

- 跨平台支持更完善

协议设计优化：

原方案：纯JSON协议

现方案：文本命令+JSON响应

优化考虑：

- 请求命令文本化，减少解析开销

- 响应JSON化，保持结构化数据优势

- 混合协议兼顾性能和可读性

- 便于不同语言客户端实现

2.3 未推翻的核心设计

业务架构：

- 分层架构设计保持不变

- JNI接口设计保持稳定

- 数据库表结构基本不变

- 面向对象设计原则贯穿始终

技术原则：

- 高性能要求保持不变

- 可扩展性考虑持续体现

- 安全性设计不断加强

- 用户体验优化持续进行

三、计划及计划实现的偏差情况说明

3.1 原始计划（基于第一周设计）

第一阶段（第1-2周）：

- 完成服务端基础框架 已完成

- 实现基本的TCP通信 已完成

- 完成数据库设计和初始化 已完成

- 实现用户登录注册功能 已完成

第二阶段（第3-4周）：

- 完成商品管理系统 已完成

- 实现购物车功能 已完成

- 开发基本的客户端界面 进行中（Qt客户端）

- 完成订单管理功能 已完成

第三阶段（第5-6周）：

- 客户端UI完善 当前阶段

- 系统集成测试 待开始

- 性能优化 待开始

- 文档完善 进行中

3.2 实际进度偏差分析

超前完成的部分：

- C++业务层架构：提前完成了完整的面向对象重构

- JNI接口实现：超额完成，实现了完整的企业级架构

- 数据库连接池：实现了生产级别的连接池管理

- 服务端稳定性：已达到生产环境要求

延后的部分：

- 客户端开发：从Java客户端改为Qt客户端，需要重新学习Qt框架

- 协议调试：文本协议替换JSON协议，需要重新调试

- UI界面开发：Qt学习曲线较陡，界面开发进度略慢

- 集成测试：由于技术栈变更，集成测试延后

偏差原因分析：

技术选型调整：

- WebSocket协议遇到SSL配置困难

- Java客户端UI体验不如专业Qt界面

- 协议兼容性考虑导致设计调整

学习成本：

- Qt框架学习需要额外时间

- Netty框架深入理解需要时间

- C++17现代特性应用需要学习

3.3 当前计划调整

剩余任务优先级：

高优先级：

- Qt客户端UI逻辑完善

- 客户端与服务端集成调试

- 基本功能端到端测试

- 核心业务流程验证

中优先级：

- 界面美化和用户体验优化

- 错误处理和异常情况处理

- 系统性能测试和优化

- 详细的技术文档编写

低优先级：

- Web管理后台开发

- B/S架构

- 高级功能实现（推荐系统等）

四、现在完全掌握的部分和困难的部分

4.1 完全掌握的技术部分

C++后端开发：

面向对象设计原理和实践

设计模式应用

内存管理和RAII资源管理

线程安全和并发编程

STL容器和算法使用

异常处理和错误管理

MySQL C++ Connector使用

JSON数据处理（nlohmann/json）

Java网络编程：

Netty框架核心概念和使用

TCP/IP协议编程

JNI本地接口编程

内存管理和垃圾回收理解

数据库技术：

MySQL数据库设计原理

SQL语句编写和优化

数据库连接池实现

数据库安全性设计

系统架构设计：

分层架构设计原则

模块化开发和接口设计

系统扩展性和维护性考虑

代码质量和测试策略

4.2 正在学习提升的部分

Qt GUI开发：

Qt框架核心概念理解

信号槽机制深入应用

Qt设计器使用技巧

QML和QtQuick学习

跨平台开发经验积累

Qt性能优化技术

网络协议设计：

协议版本兼容性处理

大数据量传输优化

网络异常处理策略

协议安全性加强

负载均衡和高可用设计

项目管理：

版本控制高级用法

持续集成和自动化构建

代码审查和质量控制

项目文档标准化

4.3 仍然困难的技术部分

Qt高级特性：

自定义控件开发复杂度高

Qt多线程编程模式理解不深

Qt样式表(QSS)掌握不够

Qt插件系统和动态加载

系统优化：

大规模并发处理经验不足

内存泄漏检测和性能分析

系统监控和运维自动化

分布式系统设计经验缺乏

前端技术：

现代前端框架应用不熟练

用户体验设计能力有限

4.4 技术提升计划

短期提升目标

1. 深入学习Qt信号槽机制和事件处理

2. 掌握Qt网络编程和异步处理

3. 熟练使用Qt设计器创建复杂界面

4. 学习Qt样式表实现界面美化

总结：

本项目C++业务层和Java网络层的结合展现了良好的架构设计能力。虽然在Qt客户端开发方面遇到了学习曲线挑战，但通过技术栈调整和持续学习，项目整体进度符合预期。未来将重点关注用户体验优化、系统性能提升和技术文档完善，力争在课程结束时交付一个功能完整、技术先进的电商系统。

55240425屈熙宸

2025年9月18日