基于 mall 电商系统的会员管理与积分体系开发实践

（总课时：36 课时，理论课 18 课时 + 实操课 18 课时）

一、实训概述

1. 项目背景

mall 电商系统是基于 **SpringBoot+MyBatis** 的企业级电商解决方案，涵盖前台商城（商品浏览、购物车、订单）与后台管理（商品、订单、权限）核心模块。本次实训聚焦电商核心的“会员体系”，在原有基础上扩展两大核心功能：

- **管理端会员管理模块**：实现会员信息管控、等级配置、积分查询、状态管理，满足运营对会员生命周期的精细化管理需求；
- **前台会员每日签到领积分功能**：构建积分获取场景，支持“连续签到阶梯奖励”机制，提升用户活跃度与留存率。

通过该实践，学生可掌握企业级“用户体系设计 + 积分流转逻辑 + 前后端协同开发”全流程，贴合电商平台真实运营需求。

2. 实训目标

目标类型	具体内容
技术目标	1. 掌握 SpringBoot 分层开发（Controller/Service/Mapper）、MyBatis 关联查询（会员 - 等级 - 积分）；2. 学会 Redis 缓存应用（防重复签到、缓存连续签到天数）；3. 熟练 Vue 组件化开发（管理端表格筛选、前台签到日历）；4. 理解数据关联逻辑（签到→积分→会员等级自动升级）。
业务目标	1. 理解电商会员体系核心逻辑（等级规则、积分获取 / 消耗、签到激励）；2. 掌握“管理端数据管控 + 前台用户交互”的业务闭环（运营配置→用户使用→数据反馈）。
工程目标	1. 具备“需求→设计→开发→测试→部署”的完整工程能力；2. 养成规范化文档编写习惯（数据库设计文档、接口文档、测试报告）。

3. 适用对象与课时分配

- **适用对象**：计算机相关专业大三学生（具备 Java 基础、SQL 语法、Vue 组件基础，了解简单业务逻辑设计）；
- **课时总览**：36 课时（理论 16 课时 + 实操 20 课时），分 7 个阶段实施，具体分配如下：

阶段	阶段名称	课时	理论 / 实操占比	核心产出
一	项目部署与业务梳理	4	2/2	可运行的本地系统、业务流程图
二	数据库设计与优化	4	2/2	表结构脚本、ER 图、索引优化报告
三	后端核心接口开发	8	4/4	后端代码、Swagger 文档、单元测试报告
四	前端模块开发	8	3/5	前端代码、页面演示视频
五	系统集成与测试	6	2/4	集成测试报告、并发测试报告
六	部署与文档整理	3	1/2	项目部署、完整实训文档
七	项目答辩	3	2/1	答辩 PPT、演示视频

二、实训准备

1. 环境与工具

类别	具体工具 / 版本	用途说明	实操重点
开发环境	JDK 1.8、Maven 3.6+	后端项目构建与依赖管理	配置 Maven 镜像（阿里云）
数据库	MySQL 8.0、Redis 6.2+	存储会员 / 签到 / 积分数据、缓存签到状态	启动 Redis 服务、测试连接
开发工具	IDEA 2022+、VS Code	后端代码开发、前端 Vue 项目编写	IDEA 导入 mall 项目、配置插件

测试工具	Postman、JMeter 5.5	接口测试、签到并发场景压力测试	Postman 保存接口集合
版本控制	Git、Gitee	代码托管与版本管理（分支开发）	克隆 mall 源码、创建功能分支
设计工具	DrawIO、Navicat	绘制 ER 图、数据库表结构设计	Navicat 查看原有表结构

2. 资料准备

- **核心资源**:
  a. mall 项目源码（含原有表 ums_member（会员表）、ums_integration_history（积分记录表））；
  b. 数据库初始化脚本（mall.sql）、Swagger 接口文档（启动后访问 http://localhost:8080/swagger-ui.html）；
- **业务资料**:
  a. 电商会员等级规则参考（如 “0 积分 = 普通会员，1000 积分 = 白银会员，3000 积分 = 黄金会员”）；
  b. 签到积分体系案例（连续 1-6 天 10 积分 / 天，连续 7 天 20 积分，连续 30 天 50 积分）；
- **技术文档**:
  a. SpringBoot 数据校验（@Valid）、MyBatis 关联查询（association）；
  b. Vue 表格组件（Element UI el-table）、Axios 请求拦截（携带 Token）。

3. 预习内容

- **后端**：SpringBoot RESTful 接口设计、MyBatis 条件构造器（QueryWrapper）、Redis String/Hash 结构;
- **前端：**Vue 路由跳转（vue-router）、Element UI 表单校验、Axios 异常处理;
- **业务：**会员等级与积分的关联逻辑（积分达标自动升级）、签到防重复提交思路。

三、实训任务与实施步骤（36 课时：理论 16 + 实操 20）

阶段一：项目部署与业务流程梳理（4 课时：理论 2 + 实操 2）

任务 1：本地环境搭建（2 课时：实操为主）

课时目标：学生能独立搭建 mall 本地前后端环境，完成双角色登录。

实操步骤：

1. 克隆 mall 源码（git clone https://gitee.com/macrozheng/mall.git），导入 IDEA；
2. 执行 mall.sql 初始化数据库，重点查看 ums_member（会员表）、ums_integration_change_history（积分变化历史记录表）字段；
3. 配置 application.yml（数据库连接 url/username/password、Redis 地址 localhost:6379）；
4. 启动后端服务（运行 MallApplication.java），访问 Swagger 文档验证接口可用性；
5. 运行前端项目：管理端（cd mall-admin-web → npm install → npm run dev）、
6. 会员前台商城后端服务：（运行 MalPortalApplication.java），访问 Swagger 文档验证接口可用性；
7. 会员前台商城后端服务：（使用了 uni-app 专用开发工具 HBuilder X（App 开发版）开发 mall-app-web）；
8. 完成登录：管理员（账号 admin 密码 macro123）、普通会员（注册测试账号 test123/123456）。

交付物：可正常运行的本地 mall 系统（前后端联调成功，截图留存）。

任务 2：核心业务流程分析（2 课时：理论为主）

课时目标：学生能梳理现有流程，设计新增功能的业务闭环。

理论 + 实操步骤：

1. 梳理现有会员流程：会员注册→完善资料→购物获积分→积分抵扣订单；
2. 设计新增流程：
   - 管理端：运营配置会员等级（积分下限 + 权益）→ 查询会员列表→ 编辑会员信息 / 调整等级→ 查看积分明细；
   - 前台：会员点击签到→ 系统验证是否已签到→ 发放积分（连续签到额外奖励）→ 更新记录；
3. 绘制业务流程图（用 DrawIO）：管理端会员管理流程、前台签到积分流程；

4. 编写简易需求说明书（含用户故事，如 “运营可按等级筛选会员”“会员签到后查看连续天数”）。

交付物：业务流程图（DrawIO 文件）、需求说明书（Word 文档）。

阶段二：数据库设计与优化（4 课时：理论 2 + 实操 2）

任务 1：现有表结构分析与扩展（1 课时：理论 + 实操）

课时目标：学生能识别现有表的不足，完成字段扩展。

实操步骤：

1. 分析原有核心表：
    - ums_member（会员表）：现有字段 id/username/phone/integration（当前积分），需新增 member_level_id（关联等级表）、status（0 - 正常 / 1 - 禁用）；
    - ums_integration_history（积分表）：现有字段 member_id/integration/type（1 - 购物获积分），需新增 related_id（关联签到记录 ID，便于追溯）；
2. 编写 ALTER 脚本，在 Navicat 中执行并验证字段新增结果。

交付物：表结构扩展 SQL 脚本（alter_table.sql）。

任务 2：新增表设计（2 课时：理论 + 实操）

课时目标：学生能根据业务需求设计表结构，明确字段约束。

理论 + 实操步骤：

1. 分析 ums_member_level（会员等级表）：存储等级规则，字段如下：
2. 绘制 ER 图（用 DrawIO），标注表间关联（ums_member→ums_member_level 多对一，ums_member→ums_member_checkin 一对多）。

交付物：新增表 SQL 脚本（create_table.sql）、ER 图（DrawIO 文件）。

任务 3：索引优化（1 课时：理论 + 实操）

课时目标：学生能根据查询场景设计索引，验证优化效果。

理论 + 实操步骤：

1. 设计索引（基于高频查询场景）：
    - ums_member：新增索引 idx_member_level_id（按等级筛选会员）、idx_phone（手机号快速查询）；

- ums_member_checkin：新增联合唯一索引 uk_member_id_checkin_date（防止单日重复签到）；
- ums_integration_history：新增索引 idx_member_id_create_time（查询会员近期积分变动）；

2. 编写 CREATE INDEX 脚本并执行；
3. 测试优化效果：用 EXPLAIN 分析优化前后的查询耗时（如 “按等级筛选会员” 从 500ms 降至 80ms），记录对比结果。

交付物：索引脚本（create_index.sql）、索引优化报告（含 EXPLAIN 截图）。

阶段三：后端核心接口开发（8 课时：理论 4 + 实操 4）

任务 1：管理端会员管理接口开发（4 课时：理论 2 + 实操 2）

课时目标：学生能开发 RESTful 接口，实现会员 / 等级管理逻辑。

1.1 接口设计（1 课时：理论）

接口路径	方法	功能描述	入参（示例）	出参（示例）
/api/admin/member/list	GET	会员列表（分页 + 筛选）	pageNum=1&pageSize=10&levelId=2	{total:100,list:[{id:1,...}]}
/api/admin/member/{id}	GET	会员详情	路径参数 id=1	{id:1,username:"test123",...}
/api/admin/member	POST	新增会员	{username:"newuser",phone:"138..."}	{success:true,id:101}
/api/admin/member/{id}	PUT	编辑会员（状态 / 等级）	路径参数 id=1，Body {status:0,levelId:2}	{success:true}
/api/admin/member/level/list	GET	会员等级列表	-	[{id:1,name:"普通会员",...}]
/api/admin/member/level	POST	新增会员等级	{name:"白银会员",minIntegration:1000}	{success:true,id:2}

1.2 核心逻辑实现（2 课时：实操）

实操步骤：

1. 基础代码：创建表的 Entity、Mapper 接口；
2. 编写 Service 层（以 MemberAdminService 为例）：
   - 会员列表查询：用 MyBatis Plus 分页插件（IPage），关联 ums_member_level 查询等级名称；
   - 会员等级调整：校验等级状态（仅启用等级可分配），更新 ums_member.member_level_id；
   - 新增会员：密码用 BCrypt 加密（调用 mall 原有 PasswordEncoder），初始积分 0，默认分配最低等级；
3. 编写 Controller 层（MemberAdminController）：
   - 加 @RestController 注解，定义接口路径；
   - 用 @RequestParam/@RequestBody 接收参数，用 @Api 注解生成 Swagger 文档；
4. 接口鉴权：复用 mall 原有 SpringSecurity 框架，配置 /api/admin/member/** 仅 “管理员” 角色可访问。

1.3 接口测试与单元测试（1 课时：实操）

1. 用 Swagger 测试接口（访问 http://localhost:8080/swagger-ui.html），验证功能（如新增会员、调整等级）；
2. 编写单元测试（用 JUnit5）：
   - 测试场景：新增会员时密码加密、编辑会员时等级校验；
   - 要求覆盖率≥85%，生成测试报告。

交付物：MemberAdminController.java、MemberAdminService.java、Swagger 截图、单元测试报告。

任务 2：会员签到与积分接口开发（4 课时：理论 2 + 实操 2）

课时目标：学生能开发签到接口，实现防重复、连续天数计算逻辑。

2.1 接口设计（1 课时：理论）

接口路径	方法	功能描述	入参（示例）	出参（示例）
/api/member/checkin	POST	会员签到	{memberId:1}（从 Token 解	{success:true,integration:10,continuousDays:3}

			析）	
/api/member/checkin/history	GET	签到记录（分页）	pageNum=1&pageSize=7	{total:30,list:[{checkinDate:"2024-05-01",...}]}
/api/member/checkin/continuous	GET	查询当前连续签到天数	-（从 Token 解析 memberId）	{continuousDays:3}

2.2 核心逻辑实现（2 课时：实操）

实操步骤：

1. 编写 Service 层（MemberCheckinService）：
   - **防重复签到**：先查 Redis 缓存（键 member:checkin:{memberId}:{today}，值 1），存在则返回 “今日已签到”；无缓存则查 ums_member_checkin 表确认；
   - **连续天数计算**：查询会员昨日签到记录，存在则 continuousDays=昨日天数+1，否则重置为 1；
   - **积分奖励规则**：连续 1-6 天 10 积分，连续 7 天 20 积分，连续 30 天 50 积分；调用 IntegrationService 新增积分记录（类型 2-签到获积分），关联签到记录 ID；
   - **缓存更新**：签到成功后，Redis 存入当日状态（过期时间 23:59:59），缓存连续天数（键 member:checkin:continuous:{memberId}，过期 1 天）；
2. 编写 Controller 层（MemberCheckinController）：
   - 从 Token 解析当前登录会员 ID（避免手动传参）；
   - 处理异常（如 “会员禁用” 返回 403，“重复签到” 返回 400）。

2.3 接口测试与单元测试（1 课时：实操）

1. 用 Postman 测试接口：验证 “重复签到拦截”“连续 7 天积分奖励”“会员禁用后无法签到”；
2. 编写单元测试：测试 checkin() 方法的核心逻辑（如连续天数计算、积分发放），覆盖率≥85%。

交付物：MemberCheckinController.java、MemberCheckinService.java、Swagger 截图、单元测试报告。

阶段四：前端模块开发（8 课时：理论 3 + 实操 5）

任务 1：管理端会员管理页面开发（5 课时：理论 2 + 实操 3）

课时目标：学生能开发 Vue 页面，实现管理端数据展示与交互。

1.1 会员列表页（member-list.vue）（2 课时：实操）

实操步骤：

1. 页面结构（基于 Element UI）：
   - 筛选区：等级下拉（加载 /api/admin/member/level/list）、状态选择（正常 / 禁用）、搜索框（手机号 / 用户名）、“新增会员” 按钮；
   - 表格区：展示 id/username/phone/integration/levelName/status，操作列（查看详情、编辑、禁用）；
   - 分页控件：用 el-pagination，绑定 pageNum/pageSize，切换时刷新表格；
2. 数据交互：
   - 用 Axios 封装请求（src/utils/request.js），携带管理员 Token；
   - 筛选条件变化时，调用 /api/admin/member/list 刷新表格；
3. 样式适配：用 Element UI 布局，确保 PC 端显示正常。

1.2 会员详情 / 编辑页（member-detail.vue）（1.5 课时：实操）

实操步骤：

1. 详情区：展示会员基础信息（用户名、手机号）、积分信息（当前积分、积分明细，调用 / api/admin/member/{id}）；
2. 编辑区：
   - 状态切换（el-select，选项 “正常 / 禁用”）；
   - 等级调整（el-select，加载启用的等级列表）；
   - “保存” 按钮：调用 /api/admin/member/{id}，成功后提示并返回列表页；
3. 表单校验：禁用状态下不可调整等级。

1.3 会员等级管理页（member-level.vue）（1.5 课时：理论 + 实操）

实操步骤：

1. 表格区：展示等级 id/name/minIntegration/description/status；
2. 新增弹窗：
   - 表单字段：等级名称、最低积分、权益描述；
   - 校验规则：最低积分非空且大于现有最高等级（调用接口校验）；
   - “提交” 按钮：调用 /api/admin/member/level，成功后刷新表格；

3. 理论讲解：Vue 组件通信（父子组件传值）、表单校验逻辑。

交付物：管理端 3 个 Vue 页面（member-list.vue/member-detail.vue/member-level.vue）、页面截图。

任务 2：前台会员签到页面开发（3 课时：理论 1 + 实操 2）

课时目标：学生能开发前台页面，实现签到交互与数据展示。

2.1 签到入口页（checkin.vue）（1.5 课时：实操）

实操步骤：

1. 页面结构：
    - 状态区：显示“今日已签到”（绿色图标）或“立即签到”（橙色按钮），下方展示连续签到天数（调用 /api/member/checkin/continuous）；
    - 奖励区：文本展示签到规则（“连续 1-6 天 10 积分，7 天 20 积分”）；
    - 记录入口：“查看签到历史”按钮，跳转至签到记录页；
2. 交互逻辑：
    - 点击“立即签到”：调用 /api/member/checkin，成功后更新状态区（图标 + 文字），弹出 el-message 提示“获得 10 积分”；
    - 未登录拦截：页面加载时检查 Vuex 中的会员 ID，无则跳转登录页；

2.2 签到记录页（checkin-history.vue）（1.5 课时：实操）

实操步骤：

1. 页面结构：
    - 筛选区：el-date-picker（选择时间范围）、“查询”按钮；
    - 列表区：展示 checkinDate/integration/continuousDays（调用 /api/member/checkin/history）；
    - 分页控件：用 el-pagination，支持分页切换；
2. 数据交互：选择时间范围后，调用接口筛选记录；
3. 理论讲解：Vuex 状态管理（缓存会员 ID）、路由守卫（未登录拦截）。

交付物：前台 2 个 Vue 页面（checkin.vue/checkin-history.vue）、页面演示视频（录屏 1 分钟）。

阶段五：系统集成与测试（6 课时：理论 2 + 实操 4）

任务 1：全流程联调（2 课时：实操）

课时目标：学生能验证前后端接口联调，确保业务闭环。

实操步骤：

1. 管理端流程验证：
    - 新增等级（“白银会员”，最低积分 1000）→ 新增会员（分配 “普通会员”）→ 编辑会员等级为 “白银会员”→ 查看详情（等级更新成功）；
2. 前台流程验证：
    - 会员登录→ 点击 “立即签到”→ 查看签到记录（新增当日记录）→ 查看积分（增加 10 积分，积分表新增 “签到” 类型记录）；
3. 记录联调问题（如 “编辑会员等级后积分未同步”），协同修复。

交付物：联调问题清单（含修复方案）。

任务 2：功能测试（2 课时：理论 1 + 实操 1）

课时目标：学生能设计测试用例，验证功能正确性。

理论 + 实操步骤：

1. 设计测试用例（覆盖核心场景）：

测试场景	预期结果	实际结果	测试状态
会员单日重复签到	返回 “今日已签到”，无重复记录		
连续签到 7 天	积分奖励 20，连续天数 7		
管理端禁用会员等级	该等级不可分配给会员		
会员禁用后签到	返回 “账号已禁用，无法签到”		

2. 执行测试用例，记录 BUG（如 “连续 30 天后天数未重置”），提交开发修复；
3. 回归测试：修复后重新验证，确保 BUG 关闭。

交付物：功能测试报告（含测试用例、BUG 清单）。

任务 3：并发测试（2 课时：理论 1 + 实操 1）

课时目标：学生能使用 JMeter 测试接口并发性能，验证数据一致性。

理论 + 实操步骤：

1. 理论讲解：并发测试原理、Redis 防重复签到的作用；
2. 实操步骤：
    - 用 JMeter 创建测试计划：模拟 100 个会员同时签到（调用 /api/member/checkin）；
    - 配置参数：线程数 100，循环次数 1， ramp-up 时间 1 秒；
    - 执行测试：监控响应时间（要求≤300ms）、检查数据库（无重复签到记录，积分正确发放）；
3. 生成测试报告：记录平均响应时间、错误率。

交付物：JMeter 测试计划（.jmx 文件）、并发测试报告（含响应时间曲线图）。

阶段六：部署与文档整理（3 课时：理论 1 + 实操 2）

任务 1：实训文档整理（理论 1 + 实操 2）

课时目标：学生能整合各阶段交付物，形成完整文档。

实操步骤：

1. 整合交付物：
    - 需求文档：需求说明书、业务流程图；
    - 设计文档：数据库表结构、ER 图、索引报告；
    - 开发文档：前后端代码（标注核心逻辑注释）、接口文档（Swagger 截图）；
    - 测试文档：功能测试报告、并发测试报告；
    - 部署文档：部署步骤、访问地址；
1. 撰写实训总结报告：
    - 技术收获（如 Redis 防重复签到、MyBatis 关联查询）；
    - 问题与解决方案（如 “连续天数计算错误→用 LocalDate 比对昨日日期”）；
    - 改进建议（如 “添加积分过期规则、签到补签功能”）。

交付物：完整实训文档（PDF 格式，含所有交付物）、实训总结报告（Word 文档）。

阶段七：项目答辩（3 课时：理论 2 + 实操 1）

任务 1：答辩准备（2 课时：实操）

课时目标：学生能整理答辩材料，准备功能演示。

实操步骤：

1. 制作答辩 PPT（结构：项目背景→需求分析→技术栈→功能实现（分模块演示）→测试结果→总结与展望）；
2. 准备材料：实训文档、代码仓库地址（Gitee/GitHub，含提交记录）、3 分钟功能演示视频（精简版）。

交付物：答辩 PPT、演示视频（3 分钟）、代码仓库链接。

任务 2：答辩与点评（1 课时：理论 + 演示）

流程：

1. 分组汇报：每人 5 分钟（3 分钟讲解 + 2 分钟演示）；
2. 评分与点评：根据功能完整性、技术深度、答辩表现打分，指出改进方向。

交付物：答辩评分表、评委点评记录。

四、实训考核标准

考核维度	占比	核心指标
功能完整性	30%	管理端会员 CRUD / 等级管理正常（10 分）、前台签到防重复 / 积分奖励正确（10 分）、数据关联一致（10 分）
技术实现质量	25%	后端接口规范（8 分）、Redis 缓存合理（7 分）、前端交互流畅（10 分）
测试与部署	20%	测试用例覆盖全面（8 分）、并发测试达标（7 分）、部署成功可访问（5 分）
文档与答辩	25%	文档规范完整（10 分）、PPT 逻辑清晰（5 分）、问题回答准确（10 分）

五、实训注意事项

1. **代码规范**：遵循阿里巴巴 Java 开发手册（POJO 类名用名词、方法名用动词）、Vue 组件命名规范（PascalCase 格式）；
2. **数据安全**：会员手机号前端脱敏展示（如 “138****1234”），密码用 BCrypt 加密存储，接口需 Token 鉴权；
3. **并发处理**：签到接口必须用 Redis 防重复，避免高并发下数据库压力过大；

4. **版本管理**：使用 Git 分支管理（dev 分支开发、master 分支发布），提交信息规范（如 feat: 新增秒杀订单查询接口）；
5. **问题排查**：优先查阅官方文档，避免重复开发；利用 IDE 调试工具、日志打印定位问题。