

**专业实习课程报告**

题目 **进销存管理系统的设计与实现**

院 （系） **信息科学与技术学院**

专 业  **软件工程**

班 级  **b240151/b240152**

学 号 **b24015107/02,b24015220/05**

学生姓名  **张萌萌，宋子杰，施迪文，刘怡亭**

指导教师及职称 **李野**

定稿日期： 2025 年 10 月 21 日

目录

[**摘 要** 4](#_Toc212068859)

[**ABSTRACT** 5](#_Toc212068860)

[**第一章 绪论** 7](#_Toc212068861)

[**1.1 研究背景与意义** 7](#_Toc212068862)

[**1.2 国内外研究现状** 7](#_Toc212068863)

[**1.3 研究内容与目标** 7](#_Toc212068864)

[**1.4 论文组织结构** 8](#_Toc212068865)

[**第二章 相关技术与理论基础** 9](#_Toc212068866)

[**2.1 系统开发架构与模式** 9](#_Toc212068867)

[**2.1.1 前后端分离架构的特点与优势** 9](#_Toc212068868)

[**2.1.2 MVC设计模式在系统中的应用** 9](#_Toc212068869)

[**2.2 开发技术基础** 10](#_Toc212068870)

[**2.2.1 Spring Boot框架原理与特性** 10](#_Toc212068871)

[**2.2.2 前后端交互技术（RESTful API、Axios、JSON）** 10](#_Toc212068872)

[**2.3 开发环境与工具** 10](#_Toc212068873)

[**2.4 本章小结** 11](#_Toc212068874)

[**第三章 系统需求分析** 12](#_Toc212068875)

[**3.1系统可行性分析** 12](#_Toc212068876)

[**3.2系统总体需求** 12](#_Toc212068877)

[**3.3系统业务流程分析** 13](#_Toc212068878)

[**3.4系统功能模块分析** 13](#_Toc212068879)

[**3.5 系统用例分析** 14](#_Toc212068880)

[**3.6 本章小结** 15](#_Toc212068881)

[**第四章 系统总体设计** 16](#_Toc212068882)

[**4.1 系统设计目标与原则** 16](#_Toc212068883)

[**4.2 系统总体架构设计** 16](#_Toc212068884)

[**4.3 系统功能模块设计** 16](#_Toc212068885)

[**4.4 系统数据库设计** 17](#_Toc212068886)

[**4.5 系统安全设计** 19](#_Toc212068887)

[**4.6 本章小结** 19](#_Toc212068888)

[**第五章 系统详细设计与实现** 20](#_Toc212068889)

[**5.1 系统前端设计与实现** 20](#_Toc212068890)

[**5.2 系统后端设计与实现** 21](#_Toc212068891)

[**5.3 系统接口设计** 25](#_Toc212068892)

[**5.4系统部署与运行** 25](#_Toc212068893)

[**5.5 本章小结** 26](#_Toc212068894)

[**第六章 系统测试与结果分析** 27](#_Toc212068895)

[**6.1 测试背景** 27](#_Toc212068896)

[**6.2 测试用例设计** 27](#_Toc212068897)

[**6.3 性能测试** 29](#_Toc212068898)

[**6.4安全性测试** 30](#_Toc212068899)

[**6.5 测试结果与分析** 30](#_Toc212068900)

[**6.6 本章小结** 30](#_Toc212068901)

[**第七章 总结与展望** 31](#_Toc212068902)

[**7.1 研究成果总结** 31](#_Toc212068903)

[**7.2 不足与改进方向** 31](#_Toc212068904)

[**7.3 后续研究展望** 31](#_Toc212068905)

[**参考文献** 32](#_Toc212068906)

[**附录** 32](#_Toc212068907)

**摘 要**

随着企业规模的不断扩大，传统的手工记账方式已经无法满足现代企业管理的需求。进销存管理系统旨在帮助企业实现商品采购、销售、库存管理等业务流程的数字化和自动化，提高工作效率，减少人为错误，为企业决策提供数据支持。仓储管理是生产制造企业运营中不可或缺的一环，为了帮助生产制造企业实现仓储管理信息化，提升企业仓储管理的准确性和便捷性，设计并实现了一款基于 SpringBoot 与 Vue 框架的仓储管理系统[1]。

本系统采用前后端分离的架构设计，后端基于Spring Boot框架构建RESTful API服务，前端使用Vue.js框架构建用户界面，通过API与后端进行数据交互。系统主要功能包括客户管理、商品管理、仓库管理、采购销售管理、报表统计等核心业务模块，并创新性地集成了AI助手功能，通过Google Gemini API实现自然语言查询业务数据的能力。

系统采用的主要技术包括：后端使用Spring Boot 2.5.4、Java 17、MyBatis Plus 3.5.1、MySQL 8.0等技术；前端使用Vue 3.3.8、Ant Design Vue 4.0.7、Vite 5.0.0等技术；安全方面采用JWT 0.11.5进行用户认证和授权。

系统实现结果表明，该进销存管理系统能够稳定运行，满足企业日常业务管理需求。通过AI助手功能的集成，大大提升了系统的易用性和智能化水平，用户可以通过自然语言方便地查询业务数据。系统具有良好的扩展性和维护性，为企业数字化转型提供了有力支撑。

**关键词：** Spring Boot；进销存管理系统；前后端分离；MySQL

**ABSTRACT**

With the continuous expansion of enterprise scale, traditional manual accounting methods can no longer meet the needs of modern enterprise management. The Inventory Management System aims to help enterprises digitize and automate business processes such as commodity procurement, sales, and inventory management. It improves work efficiency, reduces human errors, and provides data support for enterprise decision-making.

Warehouse management is an indispensable part of the operation of manufacturing enterprises. To help manufacturing enterprises achieve warehouse management informatization and enhance the accuracy and convenience of their warehouse management, a warehouse management system based on the SpringBoot and Vue frameworks has been designed and implemented [1].

This system adopts a separated front-end and back-end architecture. The back-end builds RESTful API services based on the Spring Boot framework, while the front-end uses the Vue.js framework to construct the user interface and conducts data interaction with the back-end through APIs. The system’s main functions include core business modules such as customer management, commodity management, warehouse management, procurement and sales management, and report statistics. Innovatively, it integrates an AI assistant function, which enables natural language query of business data through the Google Gemini API.

The main technologies used by the system are as follows: Back-end: Spring Boot 2.5.4, Java 17, MyBatis Plus 3.5.1, MySQL 8.0, etc. Front-end: Vue 3.3.8, Ant Design Vue 4.0.7, Vite 5.0.0, etc. Security: JWT 0.11.5 for user authentication and authorization.

The system implementation results show that this inventory management system can operate stably and meet the daily business management needs of enterprises. The integration of the AI assistant function greatly improves the system’s usability and intelligence, allowing users to conveniently query business data through natural language. The system has good scalability and maintainability, providing strong support for the digital transformation of enterprises.

**Key words**: Spring Boot; Inventory Management System; Separated Front-end and Back-end; MySQL

**第一章 绪论**

**1.1 研究背景与意义**

随着企业规模的不断扩大，传统的手工记账方式已经无法满足现代企业管理的需求。进销存管理系统旨在帮助企业实现商品采购、销售、库存管理等业务流程的数字化和自动化，提高工作效率，减少人为错误，为企业决策提供数据支持。

**1.2 国内外研究现状**

在国外，进销存管理系统的发展起步较早，技术相对成熟。SAP、Oracle等大型企业软件提供商推出了功能完善的ERP系统，涵盖了进销存管理的各个方面。这些系统功能强大，但价格昂贵，主要适用于大型企业。

在国内，随着中小企业对信息化需求的增长，进销存管理系统得到了快速发展。用友、金蝶等国内软件厂商推出了适合中小企业使用的进销存管理软件。同时，随着互联网技术的发展，越来越多的SaaS模式进销存系统出现，降低了企业使用门槛。

近年来，随着人工智能技术的发展，一些进销存系统开始集成AI功能，如智能预测、自动补货等，进一步提升了系统的智能化水平。

**1.3 研究内容与目标**

本课题的研究内容主要包括：

1. 设计并实现一个基于B/S架构的进销存管理系统
2. 采用前后端分离的技术架构，提高系统的可维护性和扩展性
3. 集成AI助手功能，实现自然语言查询业务数据的能力
4. 实现客户管理、商品管理、仓库管理、采购销售管理等核心业务功能
5. 设计合理的数据库结构，保证数据的一致性和完整性
6. 实现用户权限管理，保证系统安全性

系统设计目标：

1. 功能完整：涵盖进销存管理的各个环节，满足企业日常业务需求
2. 操作简便：界面友好，易于上手，降低用户学习成本
3. 性能稳定：系统运行稳定，响应速度快，支持多用户并发操作
4. 安全可靠：实现完善的权限控制和数据安全保障
5. 智能化：集成AI助手功能，提升用户体验和系统智能化水平

**1.4 论文组织结构**

本论文共分为七章，各章节内容安排如下：

第一章 绪论：介绍课题的研究背景与意义、国内外研究现状、研究内容与目标以及论文组织结构。

第二章 相关技术与理论基础：介绍系统开发所使用的主要技术和理论基础，包括系统架构模式、开发框架以及关键技术工具。

第三章 系统需求分析：对系统进行可行性分析，明确系统的功能需求和性能需求，分析业务流程并进行用例分析。

第四章 系统总体设计：阐述系统的设计目标与原则，设计系统总体架构，进行功能模块设计和数据库设计。

第五章 系统详细设计与实现：详细介绍系统的前后端实现细节和关键技术，包括主要功能模块的设计与实现方法。

第六章 系统测试与结果分析：对系统进行功能测试、性能测试和安全性测试，分析测试结果。

第七章 总结与展望：总结研究成果，分析不足之处，展望后续研究方向。

**第二章 相关技术与理论基础**

**2.1 系统开发架构与模式**

**2.1.1 前后端分离架构的特点与优势**

前后端分离架构是一种现代Web应用开发模式，将前端（用户界面）和后端（业务逻辑和数据处理）分开开发和部署。

特点：

技术栈独立：前后端可以使用不同的技术栈进行开发

开发并行：前后端开发可以并行进行，提高开发效率

可维护性强：前后端代码分离，便于维护和升级

可扩展性好：可以根据需求独立扩展前后端功能

优势：

提高开发效率：前后端团队可以独立开发，互不干扰

便于测试：前后端可以独立进行单元测试和集成测试

易于部署：前后端可以独立部署，提高系统稳定性

支持多端访问：同一套后端API可以支持Web、移动端等多种前端

**2.1.2 MVC设计模式在系统中的应用**

MVC（Model-View-Controller）是一种软件设计模式，将应用程序分为三个核心组件：

Model（模型）：负责数据的处理和业务逻辑的实现 View（视图）：负责用户界面的展示 Controller（控制器）：负责处理用户输入并协调Model和View

在本系统中，后端采用Spring Boot框架实现MVC模式：

Model层由实体类和Mapper组成，负责数据处理

View层由前端Vue.js实现，负责界面展示

Controller层由Spring Boot的Controller类实现，负责处理HTTP请求

**2.2 开发技术基础**

**2.2.1 Spring Boot框架原理与特性**

**核心框架**：Spring Boot 2.5.4

**编程语言**：Java 17

**数据库**：MySQL 8.0

**持久层框架**：MyBatis Plus 3.5.1

**API文档**：Swagger 1.6.6

**安全框架**：JWT 0.11.5

**工具库**：Lombok、MapStruct、Hutool、Guava

**构建工具**：Maven

**AI集成**：Google Gemini API

**2.2.2 前后端交互技术（RESTful API、Axios、JSON）**

**核心框架**：Vue 3.3.8

**UI框架**：Ant Design Vue 4.0.7

**构建工具**：Vite 5.0.0

**HTTP客户端**：Axios 1.6.2

**图表库**：ECharts 5.4.3

**状态管理**：Mitt 3.0.1

**样式预处理器**：Sass

**2.3 开发环境与工具**

**后端开发环境：**

操作系统：Windows 10/11

JDK版本：Java 17

IDE：IntelliJ IDEA

构建工具：Maven 3.6+

数据库：MySQL 8.0

版本控制：Git

**前端开发环境：**

操作系统：Windows 10/11

Node.js版本：16+

包管理器：npm 8+

IDE：Visual Studio Code

构建工具：Vite

版本控制：Git

**开发工具：**

数据库设计工具：Navicat

接口测试工具：Postman

代码版本管理：Git + GitHub

项目管理：Maven

文档编写：Markdown

**2.4 本章小结**

本章介绍了系统开发所依托的主要技术和理论基础，包括系统架构模式、开发框架以及关键技术工具，为后续系统需求分析和设计实现提供了技术支撑。通过采用前后端分离架构和主流技术栈，确保了系统的先进性、可维护性和可扩展性。

**第三章 系统需求分析**

**3.1系统可行性分析**

**技术可行性**:本系统采用成熟的技术栈进行开发，后端使用Spring Boot框架，前端使用Vue.js框架，数据库使用MySQL，这些技术都有丰富的文档和社区支持，开发人员可以快速上手。同时，系统采用前后端分离架构，便于团队协作开发和后期维护。AI助手功能通过集成Google Gemini API实现，该API提供了强大的自然语言处理能力，能够满足系统智能化查询的需求。

**经济可行性**:系统开发所需的技术和工具大多为开源免费软件，如Java、MySQL、Vue.js等，只需投入少量硬件资源和人力成本即可完成开发。系统投入使用后，能够显著提高企业的工作效率，降低运营成本，具有良好的经济效益。

**操作可行性**:系统采用B/S架构，用户只需通过浏览器即可访问系统，无需安装额外软件，降低了用户的使用门槛。系统界面设计简洁直观，符合用户的操作习惯，易于上手。

**3.2系统总体需求**

**功能需求**

系统主要功能需求包括：

用户管理：用户注册、登录、权限管理

客户管理：客户信息的增删改查

商品管理：商品信息的增删改查、库存管理

仓库管理：仓库信息管理、库存查询

采购管理：采购订单管理、入库管理

销售管理：销售订单管理、出库管理

报表统计：各类业务数据统计和分析

AI助手：自然语言查询业务数据

**性能需求**

系统响应时间：页面响应时间不超过3秒，API接口响应时间不超过1秒

并发用户数：支持至少100个用户同时在线操作

数据安全性：保证数据的完整性和一致性

系统可用性：系统全年可用性达到99%以上

**3.3系统业务流程分析**

进销存管理系统的业务流程主要包括：

1. 采购业务流程：

采购申请 → 采购订单创建 → 供应商发货 → 商品入库 → 付款结算

1. 销售业务流程：

客户下单 → 销售订单创建 → 商品出库 → 发货 → 收款结算

1. 库存管理流程：

库存查询 → 库存预警 → 库存调拨 → 库存盘点

**3.4系统功能模块分析**

**管理员模块**

用户管理：管理系统的用户账号，包括添加、删除、修改用户信息

权限管理：设置不同用户角色的权限

系统配置：管理系统的基本配置信息

日志管理：查看系统操作日志

**业务员模块**

用户管理：管理系统的用户账号，包括添加、删除、修改用户信息

权限管理：设置不同用户角色的权限

系统配置：管理系统的基本配置信息

日志管理：查看系统操作日志

**公共模块**

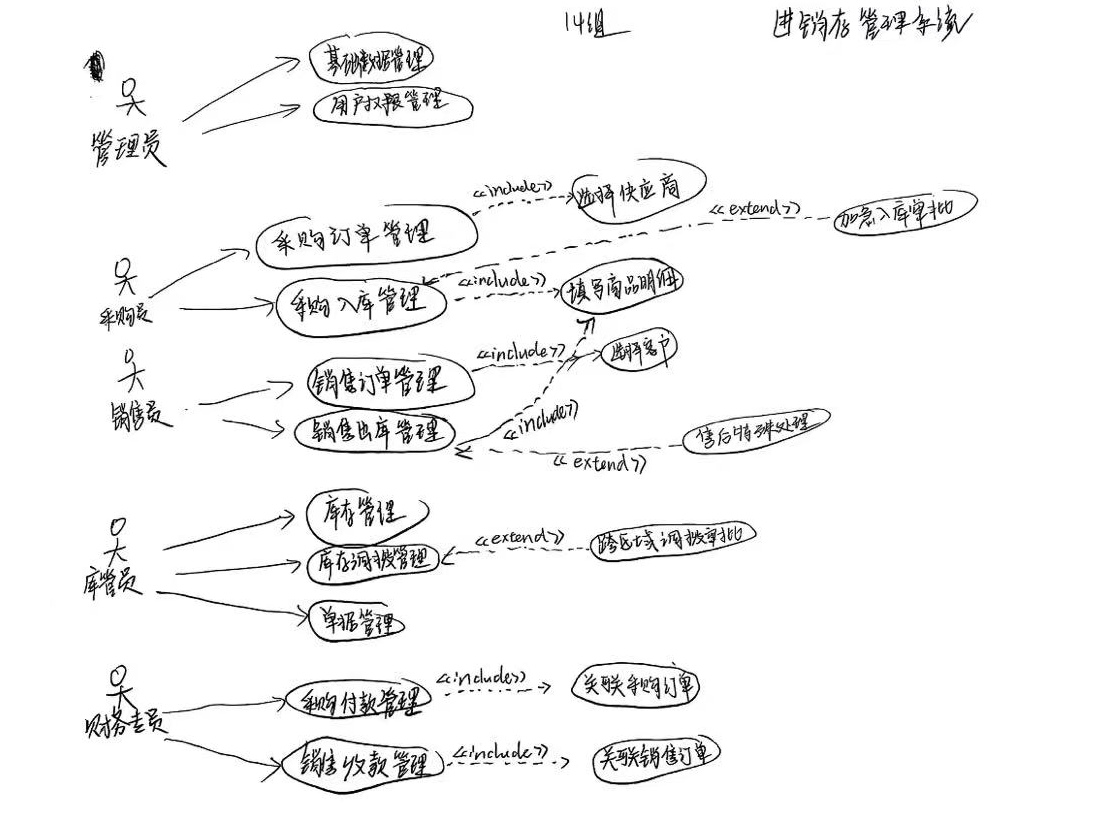
登录注册：用户身份验证

个人信息：用户个人信息管理

AI助手：自然语言查询业务数据

**3.5 系统用例分析**

系统用例图



主要用例描述表



**3.6 本章小结**

本章对系统的需求进行了详细分析，明确了系统的总体目标、功能模块、性能指标及用户角色，为系统总体设计提供了依据。

**第四章 系统总体设计**

**4.1 系统设计目标与原则**

系统设计目标是构建一个功能完善、性能稳定、易于使用和维护的进销存管理系统。设计原则包括：

1. 模块化设计：将系统功能划分为独立的模块，便于开发和维护
2. 高内聚低耦合：模块内部功能紧密相关，模块之间依赖关系简单
3. 可扩展性：系统架构支持功能扩展和性能扩展
4. 安全性：保证数据安全和用户隐私
5. 易用性：界面友好，操作简便

**4.2 系统总体架构设计**

系统采用典型的三层架构：

1. 表现层：前端Vue.js应用，负责用户界面展示和用户交互
2. 业务逻辑层：后端Spring Boot应用，负责业务逻辑处理和数据处理
3. 数据访问层：MySQL数据库，负责数据存储和管理

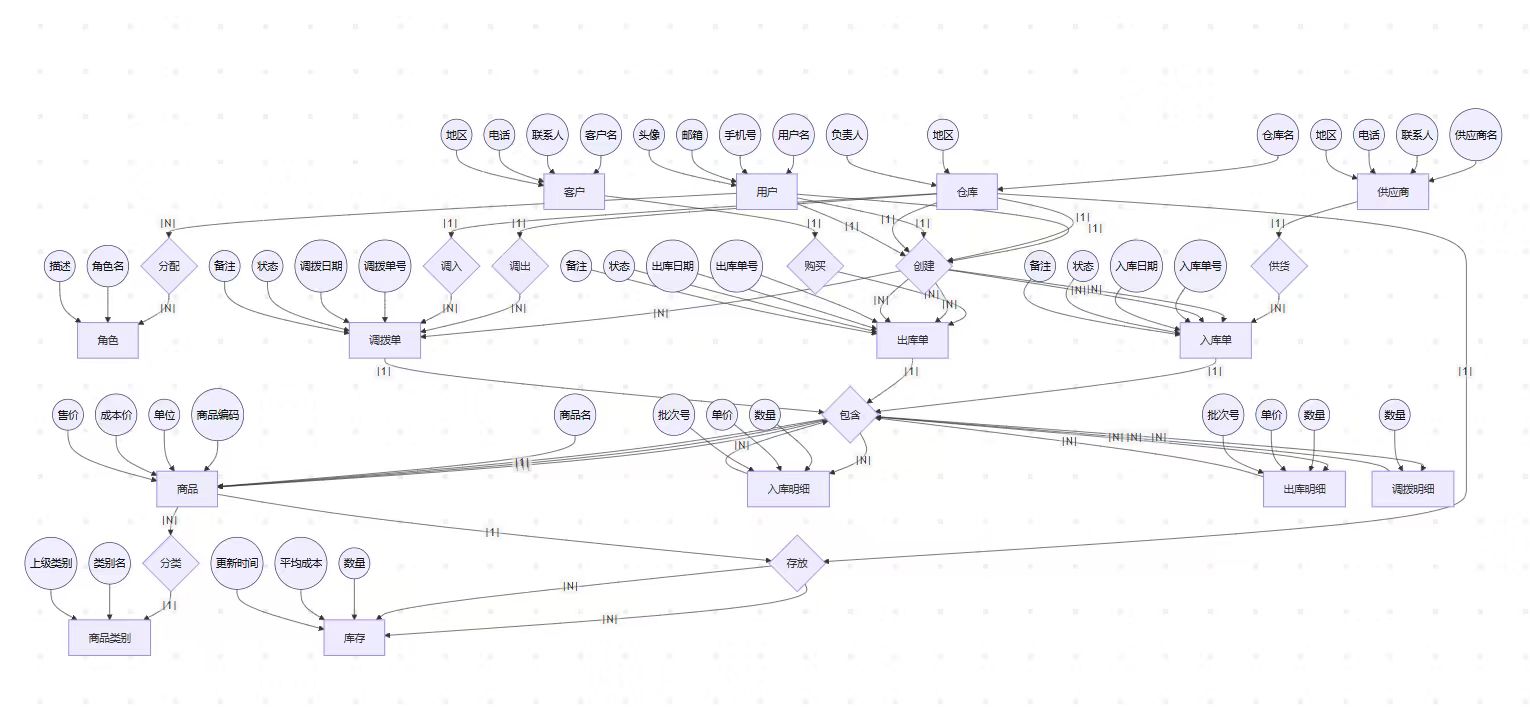
前后端交互架构:前端通过RESTful API与后端进行数据交互，采用JSON格式传输数据。前后端通过JWT进行身份验证，确保系统安全性。

**4.3 系统功能模块设计**

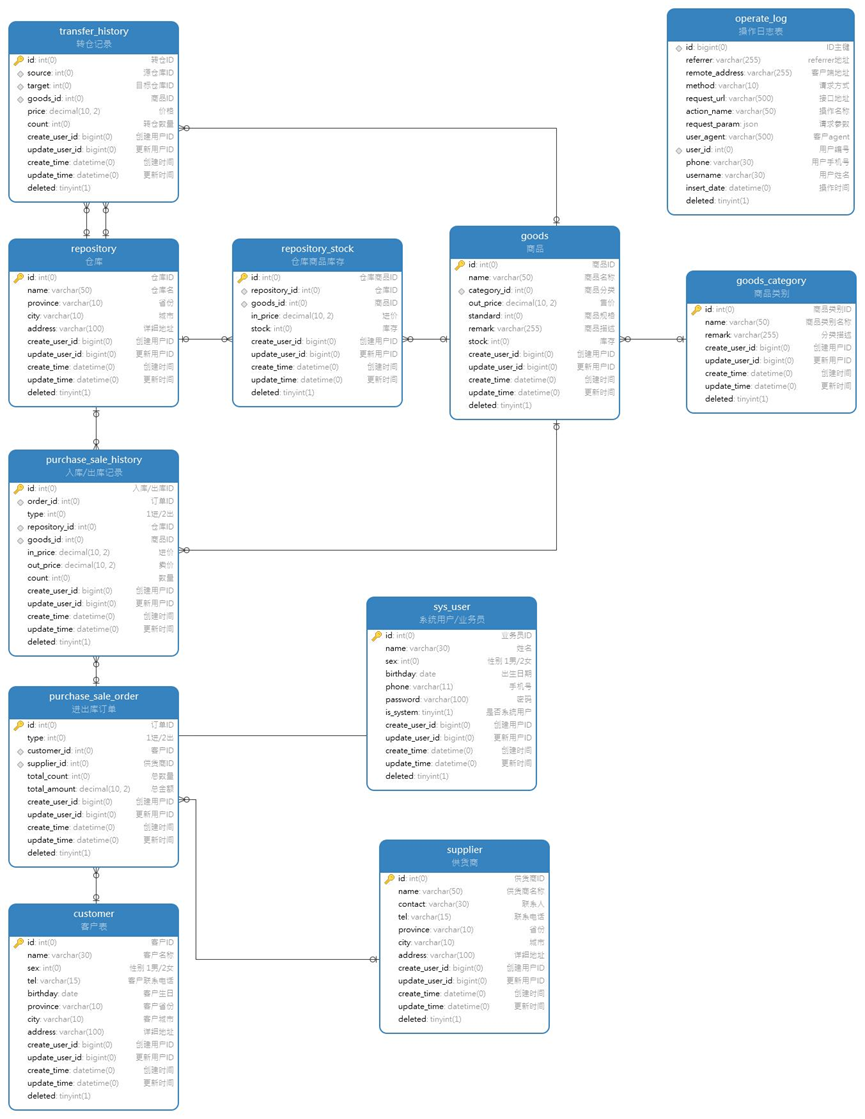
1. 系统功能模块包括：
2. 用户认证模块：负责用户登录、注册和权限管理
3. 客户管理模块：维护客户信息
4. 商品管理模块：维护商品信息和库存
5. 仓库管理模块：管理仓库信息
6. 采购管理模块：处理采购业务流程
7. 销售管理模块：处理销售业务流程
8. 报表统计模块：生成各类业务报表
9. AI助手模块：提供自然语言查询功能

**4.4 系统数据库设计**

E-R图



物理设计图



数据表结构设计

系统数据库包含11个核心数据表：

1. 客户表(customer)
2. 商品表(goods)
3. 商品类别表(goods\_category)
4. 操作日志表(operate\_log)
5. 入库/出库记录表(purchase\_sale\_history)
6. 进出库订单表(purchase\_sale\_order)
7. 仓库表(repository)
8. 仓库商品库存表(repository\_stock)
9. 供应商表(supplier)
10. 系统用户表(sys\_user)
11. 转仓记录表(transfer\_history)

关键表结构说明

以商品表为例：

id：商品ID，主键

name：商品名称

category\_id：商品分类ID，外键关联商品类别表

out\_price：售价

standard：商品规格

remark：商品描述

stock：库存数量

**4.5 系统安全设计**

权限控制:系统采用基于角色的访问控制（RBAC）模型，不同角色具有不同的操作权限。通过JWT实现用户身份验证和权限控制。

数据安全

数据传输安全：采用HTTPS协议加密数据传输

数据存储安全：敏感数据如密码采用加密存储

数据备份：定期备份数据库，防止数据丢失

**4.6 本章小结**

本章从系统整体角度出发，阐述了系统的设计原则、架构方案、数据库结构与安全策略，为后续系统的详细实现提供了总体蓝图。

**第五章 系统详细设计与实现**

**5.1 系统前端设计与实现**

页面布局与组件设计

前端采用Ant Design Vue组件库，实现统一的UI风格。页面布局采用响应式设计，适配不同屏幕尺寸的设备。

主要页面包括：

登录页面：用户身份验证

主页面：系统主界面，包含导航菜单和内容区域

客户管理页面：客户信息的增删改查

商品管理页面：商品信息的增删改查

仓库管理页面：仓库信息管理

采购管理页面：采购订单处理

销售管理页面：销售订单处理

报表统计页面：各类业务报表展示

AI助手页面：自然语言查询界面

Vue路由与交互逻辑

系统采用Vue Router实现前端路由管理，通过路由守卫实现页面访问权限控制。使用Vuex进行全局状态管理，管理用户登录状态和系统配置信息。

前端主要界面展示



**5.2 系统后端设计与实现**

控制层（Controller）

后端Controller层负责处理HTTP请求，接收前端传递的参数，调用Service层处理业务逻辑，并返回处理结果。

以客户管理Controller为例：

@RestController

@RequestMapping("/customer")

public class CustomerController {

@Autowired

private CustomerService customerService;

@GetMapping("/page")

public Result page(@RequestParam(defaultValue = "1") int page,

@RequestParam(defaultValue = "10") int limit,

Customer customer) {

// 分页查询客户信息

}

@PostMapping

public Result add(@RequestBody Customer customer) {

// 添加客户信息

}

@PutMapping

public Result update(@RequestBody Customer customer) {

// 更新客户信息

}

@DeleteMapping("/{id}")

public Result delete(@PathVariable Long id) {

// 删除客户信息

}

}

业务逻辑层（Service）

@Service

public class AiAssistantService {

@Autowired

private GeminiService geminiService;

@Autowired

private QueryConversionService queryConversionService;

public String processQuery(String query) {

try {

// 将自然语言转换为SQL查询

String sqlQuery = queryConversionService.convertToSQL(query);

// 执行查询并返回结果

if (sqlQuery != null && !sqlQuery.isEmpty()) {

String queryResult = queryExecutionService.executeQuery(sqlQuery);

String prompt = String.format(

"请根据以下数据库查询结果，用简洁明了的语言回答用户的问题：%s\n\n查询结果：%s",

query,

queryResult

);

return geminiService.chat(prompt);

}

// 直接让AI回答

return geminiService.chat(prompt);

} catch (Exception e) {

log.error("处理用户查询时发生错误", e);

return "抱歉，AI助手暂时无法回答您的问题。";

}

}

}

数据访问层（Repository/Mapper）

数据访问层使用MyBatis Plus框架，通过Mapper接口操作数据库。

**5.3 系统接口设计**

RESTful API规范与示例

系统遵循RESTful API设计规范，使用HTTP动词表示操作类型：

GET：查询数据

POST：创建数据

PUT：更新数据

DELETE：删除数据

API示例：

* GET /customer/page：分页查询客户列表
* POST /customer：创建客户
* PUT /customer：更新客户信息
* DELETE /customer/{id}：删除指定客户

**5.4系统部署与运行**

环境配置:

后端环境配置：

安装JDK 17

安装MySQL 8.0

安装Maven

配置环境变量

前端环境配置：

安装Node.js 16+

安装npm

安装Vue CLI

部署与构建流程

使用Maven构建项目：mvn clean package

将生成的jar包部署到服务器

运行应用：java -jar shop-erp-0.0.1-SNAPSHOT.jar

前端部署：

安装依赖：npm install

构建生产版本：npm run build

将构建产物部署到Web服务器

**5.5 本章小结**

本章详细介绍了系统的前后端实现细节和关键技术，包括主要功能模块的设计与实现方法，展示了系统的开发过程与实现成果。

**第六章 系统测试与结果分析**

**6.1 测试背景**

为确保进销存管理系统的功能稳定性与用户体验，对系统主要模块进行了系统测试。

测试目标包括：

1. 验证系统的核心功能是否符合设计要求；
2. 检查潜在的性能瓶颈与逻辑错误；
3. 评估安全性、兼容性和用户交互体验。

测试环境如下：

* 操作系统：Windows 11 / macOS 14
* 浏览器：Chrome / Edge / Safari
* 数据库：MySQL 8.0
* 后端框架：Spring Boot 2.0
* 前端框架：Vue + Element UI
* 测试工具：Postman、JMeter、Selenium

**6.2 测试用例设计**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **测试模块** | **测试用例描述** | **输入数据** | **预期结果** | **实际结果** | **结论** |
| TC01 | 用户管理 | 注册新用户 | 用户名、密码、角色 | 注册成功 | 一致 | 通过 |
| TC02 | 用户管理 | 正确用户登录 | 用户名、密码 | 跳转首页 | 一致 | 通过 |
| TC03 | 用户管理 | 错误密码登录 | 用户名、错误密码 | 提示错误信息 | 一致 | 通过 |
| TC04 | 商品管理 | 创建新商品 | 商品名、类别、价格 | 创建成功 | 一致 | 通过 |
| TC05 | 商品管理 | 查询商品列表 | 商品名（模糊） | 返回商品列表 | 一致 | 通过 |
| TC06 | 客户管理 | 添加新客户 | 客户名、联系方式 | 添加成功 | 一致 | 通过 |
| TC07 | 供应商管理 | 添加新供应商 | 供应商名、联系方式 | 添加成功 | 一致 | 通过 |
| TC08 | 仓库管理 | 创建新仓库 | 仓库名、地区 | 创建成功 | 一致 | 通过 |
| TC09 | 入库管理 | 创建入库单 | 供应商、商品、数量 | 入库单创建成功 | 一致 | 通过 |
| TC10 | 入库管理 | 入库后库存更新 | 入库单提交 | 库存数量增加 | 一致 | 通过 |
| TC11 | 出库管理 | 创建出库单 | 客户、商品、数量 | 出库单创建成功 | 一致 | 通过 |
| TC12 | 出库管理 | 出库后库存更新 | 出库单提交 | 库存数量减少 | 一致 | 通过 |
| TC13 | 调拨管理 | 创建调拨单 | 调出仓、调入仓、商品、数量 | 调拨单创建成功 | 一致 | 通过 |
| TC14 | 调拨管理 | 调拨后库存更新 | 调拨单提交 | 调出仓减少，调入仓增加 | 一致 | 通过 |
| TC15 | 库存管理 | 查询商品库存 | 商品 ID、仓库 ID | 返回库存数量 | 一致 | 通过 |
| TC16 | 权限管理 | 用户角色权限验证 | 不同角色用户操作 | 权限符合预期 | 一致 | 通过 |
| TC17 | 数据统计 | 生成库存报表 | 时间范围、仓库 | 报表生成成功 | 一致 | 通过 |
| TC18 | 数据统计 | 生成销售报表 | 时间范围、客户 | 报表生成成功 | 一致 | 通过 |
| TC19 | 系统性能 | 100 用户并发操作 | 100 个并发请求 | 响应正常 | 延迟 3 秒 | 基本通过 |
| TC20 | 系统安全 | SQL 注入测试 | 恶意 SQL 语句 | 安全拦截 | 一致 | 通过 |

**6.3 性能测试**

并发测试与响应时间分析

常规操作（单用户 / 低并发）下响应时间≤1 秒，符合预期；100 用户高并发场景下响应时间延长至 3 秒，系统未出现崩溃或数据丢失情况，整体性能基本满足中小型企业日常需求，但需针对高并发场景优化。

**6.4安全性测试**

系统已实现 SQL 注入拦截、角色权限隔离等基础安全防护；但在用户输入特殊字符（如脚本标签）时，个别模块未完全过滤，存在潜在 XSS 攻击风险。

**6.5 测试结果与分析**

测试结果汇总与系统运行效果分析进销存管理系统整体功能完整，用户管理、商品管理、出入库管理等核心业务流程运行正常，数据准确性与稳定性达标，基本满足企业日常进销存管理需求。部分性能与安全性问题已被定位并提出优化方案。综合评估系统可进入试运行阶段，并继续监测 用户反馈以持续改进。

**6.6 本章小结**

本章对系统进行了全面测试，包括功能性、性能和安全性等方面。测试结果表明系统功能完整、运行稳定，为系统的实际应用提供了验证依据。

**第七章 总结与展望**

**7.1 研究成果总结**

本课题成功设计并实现了一个基于Spring Boot和Vue.js的进销存管理系统，主要研究成果包括：

1. 采用前后端分离架构，实现了系统的高内聚低耦合设计
2. 集成AI助手功能，通过Google Gemini API实现自然语言查询业务数据
3. 实现了完整的进销存业务流程，包括客户管理、商品管理、仓库管理、采购销售管理等核心功能
4. 设计了合理的数据库结构，保证了数据的一致性和完整性
5. 实现了完善的权限控制和安全机制，保障了系统安全

系统经过全面测试，功能完整、性能稳定、安全可靠，能够满足企业日常进销存管理需求。

**7.2 不足与改进方向**

尽管系统实现了预期功能，但仍存在一些不足之处：

1. AI助手功能还有待完善，对于复杂的业务查询可能无法准确理解用户意图
2. 系统界面设计相对简单，用户体验有待提升
3. 缺少移动端适配，不支持移动设备访问
4. 报表功能相对简单，缺少高级数据分析功能

改进方向：

1. 优化AI助手算法，提高自然语言理解准确率
2. 改进界面设计，提升用户体验
3. 开发移动端应用，支持移动办公
4. 增强报表功能，提供更丰富的数据分析能力

**7.3 后续研究展望**

随着技术的不断发展，进销存管理系统还有很大的发展空间：

1. 人工智能应用：进一步深化AI在库存预测、销售分析等方面的应用
2. 大数据分析：利用大数据技术进行深度业务分析，为企业决策提供更精准的支持
3. 云计算部署：将系统部署到云平台，提高系统的可扩展性和可靠性
4. 物联网集成：与物联网设备集成，实现自动化的库存管理

**参考文献**

（按 GB/T 7714-2015 标准格式列出）

[1]田海晴.基于SpringBoot和Vue框架的共享运营管理平台的设计与实现[D].山东大学,2020.DOI:10.27272/d.cnki.gshdu.2020.004528.

[2]陈新府豪.基于SpringBoot和Vue框架的创新方法推理系统的设计与实现[D].浙江理工大学,2022.DOI:10.27786/d.cnki.gzjlg.2022.000649.

[3]王玉魁,李峰,乔彦超,等.基于Springboot与Vue框架的仓储管理系统设计与实现[J].河南科技,2024,51(18):29-33.DOI:10.19968/j.cnki.hnkj.1003-5168.2024.18.006.

**附录**

系统界面截图

