

实验三：软件设计与实现

题目背景

实验三 软件设计

一、实验目的

二、实验仪器及设备

三、实验步骤与要求

四、实验内容

1.内容：

2. 总体设计

2.1 设计原理概述

2.2 系统实现方案和技术路线

2.3 开发工具与环境

3. 系统结构设计

3.1 软件功能结构设计（软件结构图）

3.2 功能结构说明

4. 数据结构/数据库设计

4.1 数据库表设计

5. 用户界面设计

5.1 系统主操作流程！！！（系统主操作流程，用流程图表示）！！！！

5.2 用户主界面设计

6. 详细设计

6.1 模块说明书

7. 软件实现情况

7.1 前端实现

7.2 后端实现

7.3 数据库实现

7.4 测试情况

五、思考题

1. 在软件工程活动中如何进行项目管理和质量控制？

2. UML建模有什么优势?
3. 结构化设计的主要原理是什么?
4. 详细设计的图形工具有哪些?

图形

1. 软件功能结构图 (Software Function Structure Diagram)
2. 数据库 ER 图 (Entity-Relationship Diagram)
3. 系统主操作流程圖 (Main Operation Flow Diagram)
4. 模块说明书 - 用户注册与登录模块流程图

实验三：软件设计与实现

题目背景

第二组：题目图书馆流通管理

办证、书库、借书、还书、查询等功能

注意：这些小题目，设定甲方已有企业或校园网络、局域网并互联，具有相应的计算机设备和服务器等软硬件环境。开发周期可按6个月规划，经费按1万元计划。

1 实验的目的、意义和内容

2 总体设计

3.1 设计原理概述

3.2 系统实现方案和技术路线

3.3 开发工具与环境

3 系统结构设计

3.1软件功能结构设计

(软件结构图)

3.2 功能结构说明

(对每一个模块承担的功能进行简要说明)

4 数据结构/数据库设计

5 用户界面设计

5.1 系统主操作流程.

(系统主操作流程, 用流程图表示)

5.2 用户主界面设计

6 详细设计

(模块说明书: 对主要模块进行设计, 可用程序流程图等描述)

7 软件实现情况

.....

实验三 软件设计

一、实验目的

1. 通过本实验掌握综合运用软件工程原理和技术进行项目开发与管理的知识。
2. 加深对所学软件设计原理和理论的理解。
3. 掌握结构设计的具体过程与步骤。
4. 能够运用软件设计的理论对小型项目进行软件结构设计。
5. 能够掌握设计规范和撰写设计说明书。

二、实验仪器及设备

1. 硬件环境: PC 机
2. 软件环境: Windows环境, C语言系统或其他语言工具 (如Java、Python等)

三、实验步骤与要求

1. 需求分析与规格说明书的完善: 结合互联网资源和相关书籍, 整理和完善前期的需求分析结果, 确保所有需求都被准确理解和记录。

2. **系统开发方案和技术路线**：基于项目的特性选择合适的技术栈，例如前端使用Vue.js，后端使用Spring Boot，数据库选择MySQL等。
3. **总体软件结构设计**：确定系统的高层架构，包括客户端与服务器之间的交互模式、主要功能模块及其关系等。
4. **数据结构设计**：设计数据库表结构，确保数据的有效存储和快速检索。
5. **接口设计**：定义各个模块之间的通信接口，确保不同模块之间能够顺畅地交换信息。
6. **模块算法和内部结构设计**：对关键模块进行深入设计，包括算法的选择、变量的定义等。
7. **系统设计说明书编写**：根据上述设计内容，撰写详细的设计说明书，包括但不限于系统架构图、数据流程图、数据库ER图等。

四、实验内容

1.内容：

- 对“图书馆流通管理”项目进行需求分析。
- 完成系统的总体设计和详细设计。
- 编写详细的设计说明书。

2. 总体设计

2.1 设计原理概述

- **模块化设计**：将系统划分为多个独立的模块，每个模块负责一个特定的功能。
- **结构化设计**：采用自顶向下、逐步求精的方法，从整体到局部进行设计。
- **面向对象设计**：以对象为中心，封装数据和行为，提高代码的复用性和可维护性。

2.2 系统实现方案和技术路线

- **前端**：使用Vue.js框架，实现响应式用户界面。
- **后端**：使用Spring Boot框架，处理业务逻辑。
- **数据库**：使用MySQL数据库，存储系统数据。
- **安全**：采用HTTPS协议保障数据传输安全，使用Spring Security进行权限管理。
- **开发工具**：IDEA、Visual Studio Code、Postman等。
- **版本控制**：使用Git进行版本控制。

2.3 开发工具与环境

- 操作系统：Windows 11
- 开发工具：IntelliJ IDEA、Visual Studio Code
- 数据库管理工具：MySQL Workbench
- 版本控制工具：Git
- API测试工具：Postman

3. 系统结构设计

3.1 软件功能结构设计 (软件结构图)

1	+-----+
2	系统层
3	+-----+
4	用户管理
5	图书管理
6	借阅管理
7	查询服务
8	通知提醒
9	+-----+
10	模块层
11	+-----+
12	用户注册与登录
13	用户信息维护
14	图书录入
15	图书分类
16	借书操作
17	还书操作
18	续借功能
19	图书查询
20	借阅记录查询
21	到期提醒
22	新书通知
23	+-----+
24	数据层
25	+-----+
26	用户表
27	图书表
28	借阅记录表
29	分类表
30	日志表
31	+-----+

3.2 功能结构说明

- **用户管理：**
 - **用户注册与登录：** 读者注册成为会员，通过用户名和密码登录系统。
 - **用户信息维护：** 读者可以修改个人信息，如地址、电话号码等。
- **图书管理：**
 - **图书录入：** 管理员添加新书入库，录入书名、作者、出版社、ISBN号、数量等信息。
 - **图书分类：** 对图书按照类型、主题等标准进行分类。
- **借阅管理：**
 - **借书操作：** 读者选择想要借阅的书籍后，通过系统完成借书流程。

- **还书操作**：读者归还书籍时，管理员在系统中记录归还信息。
- **续借功能**：允许读者在线申请续借。
- **查询服务**：
 - **图书查询**：支持按书名、作者、关键词等多种方式查找图书。
 - **借阅记录查询**：读者可以查看自己的借阅历史。
- **通知提醒**：
 - **到期提醒**：自动发送邮件或短信提醒读者按时归还书籍。
 - **新书通知**：向读者推送新书上架的信息。

4. 数据结构/数据库设计

4.1 数据库表设计

- **用户表 (users)**
 - **id** (INT, 主键, 自增)
 - **username** (VARCHAR, 用户名)
 - **password** (VARCHAR, 密码)
 - **name** (VARCHAR, 真实姓名)
 - **email** (VARCHAR, 邮箱)
 - **phone** (VARCHAR, 手机号)
 - **address** (VARCHAR, 地址)
- **图书表 (books)**
 - **id** (INT, 主键, 自增)
 - **title** (VARCHAR, 书名)
 - **author** (VARCHAR, 作者)
 - **publisher** (VARCHAR, 出版社)
 - **isbn** (VARCHAR, ISBN号)
 - **category_id** (INT, 外键, 关联分类表)
 - **quantity** (INT, 数量)
- **分类表 (categories)**
 - **id** (INT, 主键, 自增)
 - **name** (VARCHAR, 分类名称)
- **借阅记录表 (borrow_records)**

- `id` (INT, 主键, 自增)
- `user_id` (INT, 外键, 关联用户表)
- `book_id` (INT, 外键, 关联图书表)
- `borrow_date` (DATE, 借书日期)
- `due_date` (DATE, 应还日期)
- `return_date` (DATE, 归还日期, 可为空)
- `renew_times` (INT, 续借次数)
- 日志表 (logs)
 - `id` (INT, 主键, 自增)
 - `action` (VARCHAR, 操作类型)
 - `details` (TEXT, 操作详情)
 - `timestamp` (DATETIME, 时间戳)

5. 用户界面设计

5.1 系统主操作流程

!!! (系统主操作流程, 用流程图表示) !!!



5.2 用户主界面设计

- **首页**：展示最新上架的图书、热门图书等。
- **个人中心**：显示用户的个人信息，提供修改功能。
- **图书查询**：提供搜索框，支持按书名、作者、关键词等条件查询。
- **借阅记录**：显示用户的借阅历史，包括借书日期、应还日期等。
- **借书**：选择图书后，填写借书信息，提交借书请求。
- **还书**：输入图书编号，确认归还信息。
- **续借**：选择已借图书，申请续借。

6. 详细设计

6.1 模块说明书

- **用户注册与登录模块**

- **功能描述：** 用户注册账号，通过用户名和密码登录系统。
- **输入：** 用户名、密码
- **输出：** 登录成功提示或错误信息
- **算法描述：**
 - 验证用户名和密码是否符合格式要求。
 - 查询数据库，验证用户名是否存在。
 - 如果用户名存在，验证密码是否正确。
 - 登录成功后，生成会话信息，返回登录成功提示。

- **图书录入模块**

- **功能描述：** 管理员录入新书信息。
- **输入：** 书名、作者、出版社、ISBN号、数量、分类
- **输出：** 录入成功提示或错误信息
- **算法描述：**
 - 验证输入信息是否完整。
 - 插入新书信息到数据库。
 - 更新分类表中的图书数量。
 - 返回录入成功提示。

- **借书模块**

- **功能描述：** 读者借阅图书。
- **输入：** 用户ID、图书ID
- **输出：** 借书成功提示或错误信息
- **算法描述：**
 - 验证用户ID和图书ID是否有效。
 - 检查图书库存是否充足。
 - 插入借阅记录到数据库。
 - 更新图书表中的库存数量。
 - 返回借书成功提示。

- **还书模块**

- **功能描述：** 读者归还图书。
- **输入：** 用户ID、图书ID
- **输出：** 还书成功提示或错误信息

- **算法描述：**
 - 验证用户ID和图书ID是否有效。
 - 检查借阅记录是否存在。
 - 更新借阅记录中的归还日期。
 - 更新图书表中的库存数量。
 - 返回还书成功提示。

7. 软件实现情况

7.1 前端实现

- **技术栈：** Vue.js
- **主要功能：**
 - 用户登录和注册页面
 - 图书查询页面
 - 借阅记录页面
 - 借书、还书、续借页面

7.2 后端实现

- **技术栈：** Spring Boot
- **主要功能：**
 - 用户管理接口
 - 图书管理接口
 - 借阅管理接口
 - 查询服务接口
 - 通知提醒接口

7.3 数据库实现

- **技术栈：** MySQL
- **主要表结构：**
 - 用户表
 - 图书表
 - 分类表

- 借阅记录表
- 日志表

7.4 测试情况

- **单元测试：**使用JUnit进行单元测试，确保每个模块的功能正确。
- **集成测试：**使用Postman进行API测试，确保前后端接口的正常通信。
- **系统测试：**进行全面的系统测试，确保整个系统的稳定性和性能。

五、思考题

1. 在软件工程活动中如何进行项目管理和质量控制？

项目管理：

- **项目计划：**明确项目目标、范围、时间表和资源分配。制定详细的项目计划，包括里程碑和关键任务。
- **团队管理：**组建合适的团队，明确每个成员的职责和任务。定期召开团队会议，跟踪项目进展。
- **风险管理：**识别潜在的风险因素，制定应对措施。定期评估风险，及时调整项目计划。
- **变更管理：**建立变更管理流程，确保所有变更都能得到适当的审批和记录。
- **沟通管理：**保持与项目干系人的良好沟通，定期汇报项目进展和问题。使用有效的沟通工具，如邮件、会议、项目管理软件等。
- **版本控制：**使用版本控制系统（如Git）管理代码和文档，确保版本的一致性和可追溯性。

质量控制：

- **需求管理：**确保需求的准确性和完整性，定期进行需求评审和验证。
- **代码审查：**定期进行代码审查，确保代码质量和一致性。使用静态代码分析工具辅助审查。
- **单元测试：**编写单元测试用例，确保每个模块的功能正确。使用自动化测试工具提高测试效率。
- **集成测试：**进行集成测试，确保各个模块之间的协同工作正常。使用持续集成工具（如Jenkins）自

动化集成测试。

- **系统测试**：进行全面的系统测试，确保整个系统的功能和性能符合要求。使用性能测试工具评估系统性能。
- **用户验收测试**：邀请用户进行验收测试，确保系统满足用户需求。收集用户反馈，进行必要的改进。
- **文档管理**：编写和维护详细的项目文档，包括需求文档、设计文档、测试文档等。确保文档的准确性和完整性。

2. UML建模有什么优势？

UML（统一建模语言）的优势：

- **可视化表达**：UML提供了丰富的图形符号，能够直观地表达系统的结构和行为，便于团队成员之间的沟通 and 理解。
- **标准化**：UML是一种国际化的建模语言，广泛应用于软件开发领域，有助于团队成员之间的协作和交流。
- **多视图支持**：UML支持多种视图（如用例图、类图、序列图、状态图等），可以从不同角度描述系统，帮助开发者全面理解系统。
- **灵活性**：UML支持不同的建模层次，从高层次的概念模型到低层次的详细设计模型，适用于软件开发的各个阶段。
- **工具支持**：有许多成熟的UML建模工具（如Enterprise Architect、Rational Rose等），提供强大的建模和文档生成功能，提高开发效率。
- **变更管理**：UML模型可以方便地进行修改和更新，有助于管理需求变更和设计变更，保持模型的一致性和准确性。

3. 结构化设计的主要原理是什么？

结构化设计的主要原理：

- **自顶向下**：从整体到局部，先设计系统的总体结构，再逐步细化每个模块的设计。
- **逐步求精**：将复杂的问题分解成若干简单的小问题，逐步解决每个小问题，最终完成整个系统的开发。
- **模块化**：将系统划分为多个独立的模块，每个模块负责完成一个特定的功能。模块之间通过明确定义的接口进行通信。
- **抽象**：通过抽象隐藏复杂的细节，只暴露必要的信息，减少模块之间的依赖，提高系统的可维护性和可扩展性。

- **信息隐藏**：每个模块内部的数据和实现细节对外部是隐藏的，其他模块只能通过模块提供的接口与其交互。
- **功能分解**：将系统的主要功能分解为多个子功能，每个子功能由一个或多个模块实现。
- **层次结构**：系统的设计通常具有层次结构，高层模块调用低层模块，形成一个有序的模块体系。

4. 详细设计的图形工具有哪些？

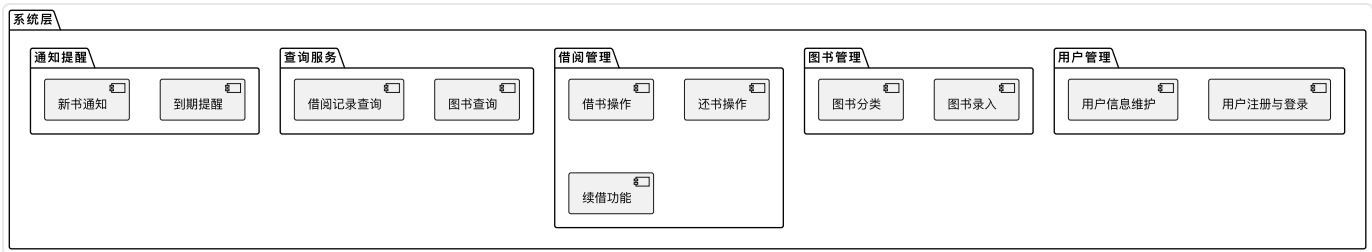
详细设计的图形工具：

- **流程图**：用于描述程序的执行流程和控制结构。常用的流程图符号包括起始/结束符、处理符、判断符、输入/输出符等。
- **数据流图（DFD）**：用于描述系统的数据流动和处理过程。DFD分为上下文图和详细图，分别描述系统的整体数据流和具体的数据处理过程。
- **状态图**：用于描述对象的状态及其状态之间的转换。状态图常用于表示系统的动态行为，特别是在事件驱动的系统。
- **活动图**：用于描述系统的活动流程和并发行为。活动图类似于流程图，但更侧重于表示并发活动和决策点。
- **类图**：用于描述系统的静态结构，包括类、属性、方法和它们之间的关系。类图常用于面向对象的设计中。
- **序列图**：用于描述对象之间的交互顺序，表示对象之间的消息传递。序列图常用于描述系统的动态行为。
- **协作图（也称为通信图）**：用于描述对象之间的交互和协作关系。协作图以对象为中心，表示对象之间的消息传递和关系。
- **实体关系图（ER图）**：用于描述数据库的逻辑结构，包括实体、属性和关系。ER图常用于数据库设计中。
- **组件图**：用于描述系统的组件及其依赖关系。组件图常用于表示系统的模块化结构和组件之间的交互。
- **部署图**：用于描述系统的物理部署结构，包括节点、组件和连接。部署图常用于表示系统的硬件和软件配置。

图形

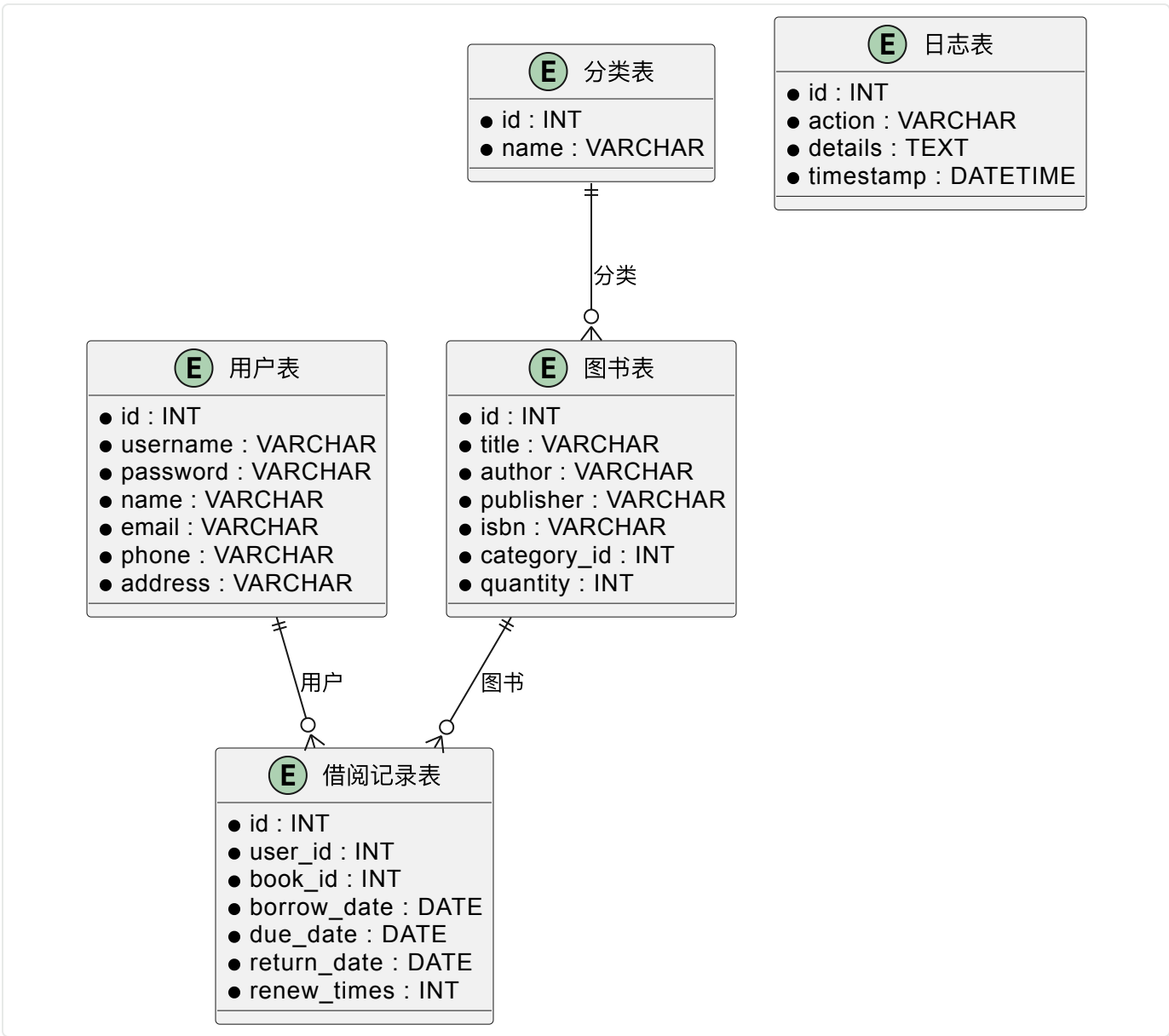
1. 软件功能结构图 (Software Function Structure Diagram)

此图用于展示系统的整体功能划分。



2. 数据库 ER 图 (Entity-Relationship Diagram)

此图用于展示数据库表之间的关系。



3. 系统主操作流程图 (Main Operation Flow Diagram)

此图用于展示系统的主要操作流程。

```
PlantUML 1.2024.4

<b>This version of PlantUML is 225 days old, so you should
<b>consider upgrading from https://plantuml.com/download

[From string (line 6) ]

@startuml
start
:用户登录;
if (登录成功?) then (yes)
:进入主页面;
while (操作未结束) do
Syntax Error?
```

4. 模块说明书 – 用户注册与登录模块流程图

此图用于详细描述用户注册与登录模块的工作流程。

