1. 项目概述

1.1 项目背景

图书管理系统是一种用于管理图书馆图书及借阅流程的系统，目的是简化图书管理的复杂流程，提升图书馆的工作效率。

1.2 系统功能概述

系统应具备如下功能：

- 用户注册、登录
- 图书查询
- 图书借阅、归还
- 图书信息管理（增、删、改、查）
- 借阅记录查看
- 用户权限管理（管理员和普通用户）

---

2. 系统设计

2.1 架构设计

系统采用B/S（Browser/Server）架构，基于Web技术构建，主要由前端、后端和数据库三部分组成。

- **前端**：HTML、CSS、JavaScript实现网页的展示和交互。
- **后端**：使用RestFul API处理用户请求，连接数据库，并返回处理结果。
- **数据库**：MySQL用于存储图书信息、用户信息、借阅记录等。

2.2 主要功能模块划分

- 用户模块
- 图书管理模块
- 借阅管理模块

- 管理员模块

2.3 页面设计

系统页面主要包括：

- 首页
- 用户注册/登录页面
- 图书列表页
- 图书详情页
- 借阅记录页面
- 管理员后台页面（添加、删除图书等操作）

---

3. 数据库设计

3.1 数据库概述

数据库使用MySQL，包含以下主要表：

- 用户表（存储用户信息）
- 图书表（存储图书信息）
- 借阅记录表（记录借阅情况）

3.2 数据库表设计

3.2.1 用户表

字段名	类型	描述
id	INT	主键，用户ID
username	VARCHAR(50)	用户名
password	VARCHAR(50)	密码
role	ENUM('user', 'admin')	用户角色

3.2.2 图书表

字段名	类型	描述
book_id	INT	主键，图书ID
title	VARCHAR(100)	书名
author	VARCHAR(50)	作者
publish_year	YEAR	出版年份
status	ENUM('available', 'borrowed')	状态

3.2.3 借阅记录表

字段名	类型	描述
record_id	INT	主键，借阅记录ID
user_id	INT	用户ID（外键）
book_id	INT	图书ID（外键）
borrow_date	DATE	借阅日期
return_date	DATE	归还日期

---

4. 前端设计

4.1 页面布局

使用HTML5和CSS3设计前端页面，主要页面包括：

- **主页**：显示图书馆简介、热门图书推荐。
- **用户登录页面**：输入用户名和密码，支持注册新用户。
- **用户列表页面**：显示所有用户信息，提供搜索功能。
- **图书列表页面**：显示所有图书信息，提供搜索功能。

- **图书详情页面**：显示选定图书的详细信息，并提供借阅功能。
- **借阅记录页面**：显示用户个人的借阅记录。

4.2 交互设计

前端主要使用JavaScript实现用户交互，包括：

- 表单验证（登录、注册、添加图书）
- 图书搜索（实时筛选图书）
- 图书借阅与归还

---

5. 后端设计

5.1 后端技术栈

后端使用Springboot框架，提供API接口进行数据处理和逻辑实现。

5.2 API设计

5.2.1 用户模块

- **注册**：`POST /api/register`，用户注册接口。
- **登录**：`POST /api/login`，用户登录接口。
- **获取用户信息**：`GET /api/user/{id}`，获取指定用户信息。

5.2.2 图书管理模块

- **获取图书列表**：`GET /api/books`，获取所有图书信息。
- **添加图书**：`POST /api/books`，管理员添加图书。
- **修改图书信息**：`PUT /api/books/{id}`，管理员修改图书信息。
- **删除图书**：`DELETE /api/books/{id}`，管理员删除图书。

5.2.3 借阅管理模块

- **借阅图书**：`POST /api/borrow`，用户借阅图书。
- **归还图书**：`POST /api/return`，用户归还图书。

- **查看借阅记录**：`GET /api/records`，查看用户的借阅记录。

---

6. 安全性设计

6.1 用户身份验证

使用JWT（JSON Web Token）加密和验证用户身份。服务端通过签名验证JWT的有效性，而无需存储用户会话信息。

6.2 用户权限管理

- 普通用户只能借阅和归还图书，无法修改图书信息。
- 管理员具有增加、修改、删除图书信息的权限。

---

7. 测试方案

7.1 单元测试

对各个模块进行单元测试，确保各个功能模块能够独立正常工作。

7.2 集成测试

将前后端集成起来进行功能测试，确保系统整体运行流畅。

7.3 用户测试

邀请用户参与系统测试，收集反馈，优化用户体验。

---

8. 部署方案

8.1 服务器选择

选择合适的云服务器（如阿里云、AWS），搭建Web服务器。

8.2 服务器配置

配置Node.js运行环境和MySQL数据库，确保前后端的顺利运行。

8.3 系统维护

定期备份数据库，监控系统运行状态，确保系统的稳定性。