上海大学

SHANGHAI UNIVERSITY

数据库原理课程报告

UNDERGRADUATE PROJECT (THESIS)

图书管理系统

学　　院　计算机工程与科学学院

专　　业　计算机科学与技术

组　　号　10

指导教师　郑宇

成　　员　21122783 周孙睿

21122787 吕斯路

21122849 邹凌杰

日　　期　2024-1-20

目录

第1章 引言

§1.1 项目分工

分工和个人贡献度：

21122783 周孙睿：前端部分，10分钟演示视频录制，报告第三章前端部分撰写

21122787 吕斯路：后端部分，报告第三章后端部分、第四章撰写

21122849 邹凌杰：后端部分，报告第一、二章部分撰写，报告汇总排版美化

个人贡献度：

经过小组成员讨论，三人各司其职，共同努力，三人贡献度平分。

§1.2 项目背景

本项目背景为课程设计报告，该项目的背景是某单位资料室需要建立一个图书管理系统，以更有效地管理其图书馆藏和提供服务给读者。资料室的图书管理系统旨在简化图书管理员的工作流程，提高图书借阅和归还的效率，并为读者提供方便的借阅体验。

具体而言，资料室面临以下需求和挑战：

图书管理员管理： 资料室有多名图书管理员，每位管理员负责已购入图书的编目和借还工作。因此，需要一个系统来存储和管理图书管理员的信息，包括工号和姓名，以便有效地分配工作和追踪他们的活动。

读者管理： 读者可在阅览室阅读，也可通过图书流通室借还图书。为了提供个性化的服务，系统需要生成不同的读者ID，并存储读者的基本信息，如姓名、电话和Email。

图书管理： 每部书在系统中对应唯一的一条图书在版编目数据（CIP），其中包括ISBN号、书名、作者、出版商、出版年月等基本信息。此外，需要记录每本书的图书ID、存放位置、当前状态等信息，以确保资料室能够追踪每册图书的位置和状态。

借还书流程： 系统需要支持读者借阅图书的流程，包括图书管理员登记读者ID、所借图书ID、借阅时间和应还时间。同时，要记录读者还书的时间，以便及时更新图书的可借状态。

限制和预约： 系统需要实现一些限制，例如一名读者最多只能借阅十本图书，每本图书最多只能借两个月。当某书目的可借出图书数量为零时，读者可以对其进行预约登记，系统需要记录读者ID、需要借阅的图书的ISBN号和预约时间。

§1.3 编写目的

该图书管理系统的目的在于满足某单位资料室对图书馆藏管理和读者服务的多方面需求，通过引入现代化的信息技术，提升图书馆的管理效率和读者体验。以下是该项目的主要目的：

优化图书管理： 引入图书在版编目数据（CIP）和图书ID等信息，以建立起系统化的图书馆藏管理体系。通过记录每本书的详细信息，如ISBN号、书名、作者、出版商等，资料室能够更精确地追踪馆内图书，并及时进行库存管理。

提高图书管理员效率： 通过数字化管理系统，图书管理员可以更轻松地进行图书编目、借阅和归还等工作。系统能够帮助管理员迅速查找图书信息、跟踪借还记录，从而减轻他们的工作负担，使整个管理流程更加高效。

个性化读者服务： 通过为每位读者生成独特的读者ID，并记录其基本信息，系统可以提供个性化的服务。读者能够方便地借阅图书、查询借还记录，同时系统也可以根据读者的借阅历史提供相关推荐，提升服务的质量和体验。

第2章 系统总体设计

§2.1 系统用途

本项目定义为一个图书管理系统。它部署在图书馆的主机上，向图书管理员与读者提供相应的服务。该系统用Web形式展示，一方面方便图书管理员管理相关图书，另一方面方便读者查询相关图书信息。

§2.2 功能需求与流程概述

该图书管理系统主要面向图书管理员和读者提供服务，旨在简化图书馆的图书馆藏管理和读者借阅服务。在设计系统时，需考虑到图书管理员和读者的使用习惯，使系统交互简便，界面友好，对用户有吸引力。以下按照四部分分析系统的功能需求和流程概述：

（1）图书管理员： 图书管理员是系统的核心使用者，其主要功能需求包括：

编目图书：图书管理员需要能够对新购入的图书进行编目，录入图书的基本信息，并

能够能够创建、编辑、删除数目。

管理图书信息：管理员应能够添加、编辑、删除、修改图书的基本信息，包括ISBN号、书名、作者、出版商、出版年月等。

借阅和归还图书：在管理员通过系统登录之后，为读者登记需要借阅图书的信息，包括读者ID、图书ID、借阅时间和应还时间。同时，系统应支持管理员记录读者归还图书的时间，归还图书操作。

（2）读者： 读者是系统的主要参与者，其功能需求包括：

查询图书信息：读者需要通过系统查询图书的基本信息、当前状态和存放位置，以便找到所需图书。

借阅图书信息查询：在读者在管理员帮助登记借书信息之后，便可以在系统中查看自己的图书归还期限，

预约图书：当某书目的可借出图书数量为零时，读者可以通过系统进行预约登记。

（3）邮箱通知功能： 邮箱通知功能包括一些自动化的通知发送，其功能需求包括：

预约信息通知：一旦其他读者归还这种书，就自动通知该预约读者。系统将自动清除超出预约期限的预约记录并修改相关信息。对于已到期且未归还的图书，系统通过 Email 自动通知读者；若读者预约的书已到，系统则自动通过 Email 通知该读者来办理借书手续。

逾期为归还图书通知：对于超期归还者，系统自动计算罚金（具体的计算过程此处省略）。系统同时自动查询预约登记表，若存在其他读者预约该书的记录，则通过邮件方式通知预约此书的读者，并将该图书的状态修改为“未借出”。

§2.3 系统关系模型设计

图书管理员librarian

属性：工号(主键)，姓名，登录密码

读者reader

属性：读者ID（主键，自增，系统生成）、姓名、电话、Email、登录密码

书目CIP：

ISBN号（主键）、书名、作者、出版商、出版年月，本资料室拥有该书的册、为该书目登记的管理员ID（外键）

书BOOK：

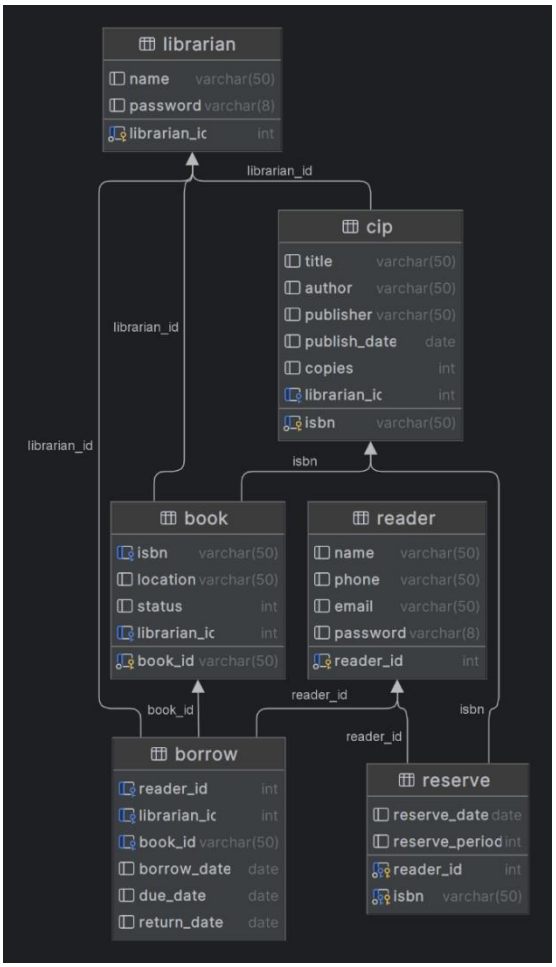
属性：图书ID（主键）、ISBN号（外键）、存放位置、当前状态（用数字表示）、借为该图书登记的管理员ID（外键）

借阅表Borrow：

属性：读者ID（外键）、管理员ID（外键）、所借图书ID（外键）、借阅时间、应还时间、归还时间（可以为空）。

预约表Reserve：

属性：记录读者ID（外键）、需要借阅的图书的ISBN号（外键）、预约时间、邮件已发送flag、预约期限天数(<=10)



图表 2.3-1 系统关系模型设计

§ 2.4 系统业务流程

系统流程概述： 系统的流程主要包括以下步骤：

管理员预先创建账号和图书馆藏信息。

图书管理员进行图书编目。

读者创建账号。

读者通过系统查询图书信息、在管理员帮助下借阅图书和归还图书。

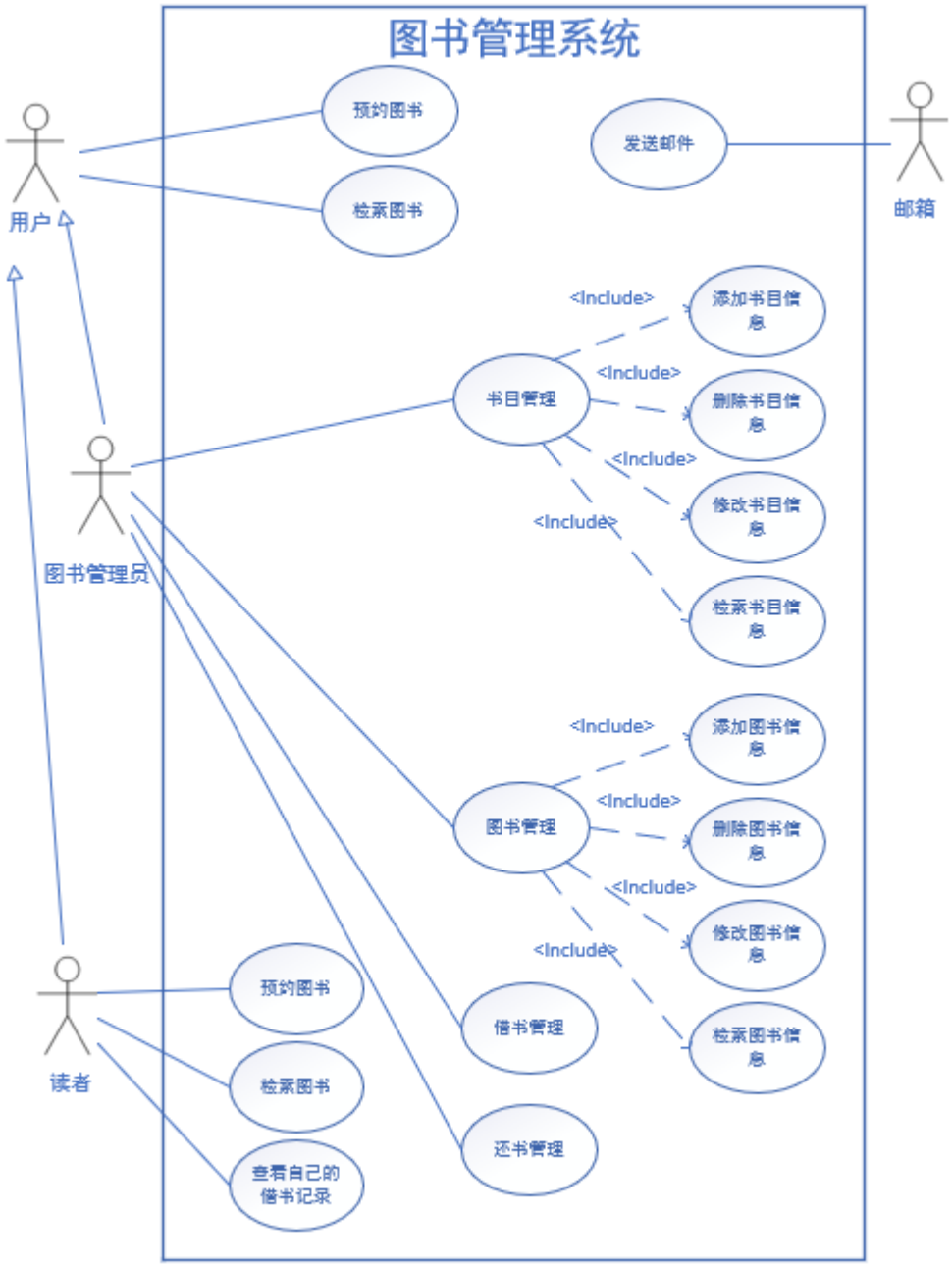
系统支持图书管理员记录借阅和归还的信息，以及处理图书的预约。

在系统中，图书的状态会随着借阅和归还的流程而更新。

通过以上功能需求和流程概述，该图书管理系统旨在提供一个简便、高效的图书馆管理和读者服务平台，使图书馆的运作更加现代化和数字化。

软件工程上通常使用业务流程图或用例图来描述软件业务流程，此处以用例图为例。该教学诊断系统的主要参与者分为三类角色：图书管理员、读者、邮箱。

系统用例图如图表 2.4-1所示。



图表 2.4-1 系统关系模型设计

§ 2.5 开发环境

运行环境为Window10操作系统，浏览器采用Edge浏览器，数据库使用MySQL。前端使用vue，后端使用python。

第3章 系统功能描述

本系统共包含5个功能模块，包括注册模块设计、登录模块设计、图书管理员模块设计和读者模块设计，功能模块列表如表3-0，每个模块中又细分为一些具体的功能，以下将进行主要功能模块介绍。

表 3-0 用户输入框功能表

注册	注册账号	无
登录	账号登录	无
图书管理员	添加书目信息	无
	删除书目信息	无
	修改书目信息	无
	检索书目信息	无
	添加图书信息	无
	删除图书信息	无
	修改图书信息	无
	检索图书信息	无
	借书管理	登记读者借书信息
	还书管理	登记读者还书信息
读者	预约图书	预约暂时没有的图书
	检索图书	无
	查看自己的借书记录	无
邮箱发送	发送邮件	无

§3.1 注册模块设计

§3.1.1 前端设计

1、输入框设计

表 3-1 用户输入框功能表

功能描述	用户输入用户名、密码、确认密码、手机号、邮箱、邮箱验证码进行注册。
异常情况	1、所有输入框不能为空 2、手机号只能输入数字 3、密码不能输入中文，且不少于6位 4、确认密码必须与密码相同

	5、邮箱验证码错误

表 3-2 管理员输入框功能表

功能描述	管理员输入用户名、工号、密码、确认密码进行注册
异常情况	1、所有输入框不能为空 2、工号只能输入数字 3、密码不能输入中文，且不少于6位 4、确认密码必须与密码相同

代码展示：

```
function disablec(v){          //禁止输入中文
    v.value = v.value.replace(/[\u4e00-\u9fa5]/g, '')
}

function onlyNum(v){          //只能输入数字
    v.value = v.value.replace(/[^\d]/g,'')
}

function checkCodeLen(){      //检查密码长度
    if(pw.value.length < 6){
        error_msg.value = "密码长度不能小于 6"
        var error_id = ref("password")
    }
}

function checkCode(){         //检查确认密码
    if(pw.value != confirm_pw.value){
        error_msg.value = "两次密码不相同"
        var error_id = ref("confirm_pw")
    }
}

function checkError(){         //输入框不能为空
    if(user.value == '' || user.value == null){
        error_msg.value = "用户名不能为空"
        error_id.value = "user"
    }
    else if(pw.value == ''){
        error_msg.value = "密码不能为空"
        error_id.value = "password"
    }
}
```

2、注册按钮设计

表 3-3 前端注册功能表

功能描述	用户输入用户名、密码、确认密码、手机号、邮箱、邮箱验证码进行注册。 管理员输入用户名、工号、密码、确认密码进行注册。
请求消息	1、请求类型：POST 2、发送数据：上述所有信息 3、接收数据：检查通过为“1”，否则不通过
异常情况	1、手机号、邮箱已注册 2、工号已注册

代码展示：

```
function register(){
   var req = mod == "1" ? "/registerCheck0" : "/registerCheck1"
    fetch(ip + req, {
        method: "post",
        body: JSON.stringify(account)
    }).then(res=>res.text())
    .then(res => {
         if(res == '1'){                //检查通过
            window.open("login.html", "_self")
        }
        else if(res == 'phoneRepeat'){   //手机号重复
            error_msg.value = "手机号已注册"
            error_id.value = "phone"
        }
        else if(res == 'mailRepeat'){   //邮箱重复
            error_msg.value = "邮箱已注册"
            error_id.value = "mail"
        }
        else if(res == 'workNumRepeat'){   //工号重复
            error_msg.value = "工号已注册"
            error_id.value = "work_num"
        }
    })
}
```

§ 3.1.2 后端设计

表 3-4 后端函数功能表

功能描述	输入项	处理描述	输出项
发送验证码	接收包含邮件信息的数据	生成随机验证码，通过邮件发送给指定邮箱	返回生成的验证码
用户注册检查	接收包含用户信息的数据（用户名、手机号、邮箱、验证码）	在数据库中检查手机号和邮箱是否重复，若不重复则插入用户信息	返回注册成功或重复的信息
管理员注册检查	接收包含管理员信息的数据（用户名、工号、验证码）	在数据库中检查管理员工号是否重复，若不重复则插入管理员信息	返回注册成功或重复的信息

代码展示：

```
# 发送验证码
def sendVerCode(data):
    rand=str(random.randint(10011, 99992))
    sender = EmailSender()
    reciever_list = []
    reciever_list.append(data["mail"])
    sender.send(reciever_list, "注册验证码",
                rand)
    return rand

#用户注册，检查用户手机号、邮箱是否重复（phoneRepeat、mailRepeat）
# {"user": user.value, "phone": phone.value, "mail": mail.value,
"code": pw.value}:
def registerCheck0(data):
    with UsingMysql() as um:
        um.cursor.execute("select 1 from reader where phone=%s",
(data["phone"],))
        check = um.cursor.fetchone()
        if check is not None:
            return "phoneRepeat"
        um.cursor.execute("select 1 from reader where email=%s",
(data["mail"],))
        check = um.cursor.fetchone()
        if check is not None:
            return "mailRepeat"
        insert_sql="""INSERT INTO reader(name,phone,email,password)
                    VALUES (%s,%s,%s,%s)"""
```

```
        um.cursor.execute(insert_sql, (data["user"],data["phone"],
data["mail"], data["code"],))
        return '1'

# 管理员注册，检查管理员工号是否重复（workNumRepeat）
# {"user": user.value, "worknum": work_num.value ,"code": pw.value}
def registerCheck1(data):
    with UsingMysql() as um:
        um.cursor.execute("select 1 from librarian where
librarian_id=%s", (data["work_num"],))
        data = um.cursor.fetchone()
        if data is not None:
            return "workNumRepeat"
        insert_sql="""INSERT INTO
librarian(librarian_id,name,password)
                    VALUES (%s,%s,%s)"""
        um.cursor.execute(insert_sql, (data["work_num"],data["user"],
data["code"],))
        return 1
```

§ 3.2 登录模块设计

§ 3.2.1 前端设计

1、输入框设计

表 3-5 用户输入框功能表

功能描述	输入用户手机号、密码进行登录
异常情况	1、手机号、密码不能为空 2、手机号只能输入数字 3、密码不能输入中文

表 3-6 管理员输入框功能表

功能描述	输入管理员工号、密码进行登录
异常情况	1、工号、密码不能为空 2、工号只能输入数字 3、密码不能输入中文

代码展示：

```
function disablec(v){          //禁止输入中文
    v.value = v.value.replace(/[\u4e00-\u9fa5]/g, '')
```

```
}

function onlyNum(v){          //只能输入数字
    v.value = v.value.replace(/[^\d]/g,'')
}

function checkError(){        //输入框不能为空
    if(user.value == '' || user.value == null){
        error_msg.value = "账号不能为空"
        error_id.value = "user"
    }
    else if(pw.value == ''){
        error_msg.value = "密码不能为空"
        error_id.value = "password"
    }
}
```

2、登录按钮设计

表 3-7 前端登录功能表

功能描述	用户输入手机号、密码进行登录 管理员输入工号、密码进行登录
请求消息	1、请求类型：POST 2、发送数据：用户手机号、密码；管理员工号、密码 3、接收数据：检查通过为“1”，否则不通过
异常情况	1、手机号、密码不匹配 2、工号、密码不匹配

代码展示：

```
function login(){
    var req = mod.value ? "/loginCheck1" : "/loginCheck0"
    fetch(ip + req, {
        method: "post",
        body: JSON.stringify(account)
    }).then(res=>res.text())
    .then(res => {
        if(res == '1'){      //检查通过
            window.open(web, "_self")
        }else{        //检查未通过
            error_msg.value = "账号或密码错误"
            has_error.value = 1;
        }
    })
}
```

§ 3.2.2 后端设计

表 3-8 后端登录功能

功能描述	输入项	处理描述	输出项
用户登录	接收包含手机号和密码的数据	在数据库中检查手机号和密码是否匹配，若匹配则返回登录成功（1），否则返回登录失败（0）	返回登录成功或失败的信息
管理员登录	接收包含管理员 ID 和密码的数据	在数据库中检查管理员 ID 和密码是否匹配，若匹配则返回登录成功（1），否则返回登录失败（0）	返回登录成功或失败的信息

代码展示：

```
# 用户登录 要求输入手机号和密码
def loginCheck0(data):
    phone = data["phone"]
    password = data["code"]
    with UsingMysql() as um:
        um.cursor.execute("select * from reader where phone=%s and
password=%s", (phone, password))
        datas = um.cursor.fetchall()
        if len(datas) == 0:  # 检查用户账号是否正确（正确为 1，否则 0）
            return 0
        else:
            return 1
# 管理员登录 要求输入管理员 ID 和密码
def loginCheck1(data):
    id = data["ID"]
    password = data["code"]
    with UsingMysql() as um:
        um.cursor.execute("select * from librarian where
librarian_id=%s and password=%s", (id, password))
        datas = um.cursor.fetchall()
        if len(datas) == 0:  # 检查用户账号是否正确（正确为 1，否则 0）
            return 0
        else:
            return 1
```

§ 3.3 管理员功能模块设计

§ 3.3.1 前端设计

1、添加书目设计

表 3-9 添加书目功能表

功能描述	管理员添加书目
请求消息	1、请求类型：POST 2、发送数据：书名、作者、出版商、ISBN号、出版年月、管理员工号 3、接收数据：检查通过为“1”，否则不通过
异常情况	1、所有输入不能为空 2、ISBN号已存在

代码展示：

```
function add_cate_btn(){       //点击添加书目
    var req = is_add.value ? "/addCatelog" : "/modCatelog"
    var fdata = formToJson(form);
    for(var i = 0; i < form.length; i++){
    var ele = form[i]
    if(ele.value == ''){
        error = ['书名',"作者","出版商","ISBN 号","出版年月","经办人"]
        error_msg.value = error[i] + "不能为空"
        ele.style.border = "2px solid red"
      }
    }
    fetch(ip + req, {
      method: "post",
      body: JSON.stringify(fdata)
    }).then(res=>res.text())
    .then(res => {
      if(res != '1'){
          error_msg.value = res  //返回错误信息
          has_error.value = true;
      }
    })
}
```

2、修改书目设计

表 3-10 修改书目功能表

功能描述	管理员修改书目
请求消息	1、请求类型：POST 2、发送数据：书名、作者、出版商、旧ISBN号、新ISBN号、出版年月、经办人 3、接收数据：检查通过为“1”，否则不通过
异常情况	1、所有输入不能为空 2、新ISBN号、经办人不存在 3、日期格式不正确

代码展示：

```
function mod_cate_btn(){       //点击修改书目
    var req = is_add.value ? "/addCatelog" : "/modCatelog"
    var fdata = formToJson(form);
    for(var i = 0; i < form.length; i++){
    var ele = form[i]
    if(ele.value == ''){
        error = ['书名',"作者","出版商","ISBN 号","出版年月","经办人"]
        error_msg.value = error[i] + "不能为空"
        ele.style.border = "2px solid red"
      }
    }
    fetch(ip + req, {
      method: "post",
      body: JSON.stringify(fdata)
    }).then(res=>res.text())
    .then(res => {
      if(res != '1'){
          error_msg.value = res  //返回错误信息
          has_error.value = true;
      }
    })
}
```

3、删除书目设计

表 3-11 删除书目功能表

功能描述	管理员删除书目
请求消息	1、请求类型：POST 2、发送数据：ISBN号

	3、接收数据：无
异常情况	无

代码展示：

```
function del_mod_cate(cate){     //删除书目
    var req = "/delCatelog"
    fetch(ip + req, {
        method: "post",
        headers: {'content-type': 'application/json'},
        body: JSON.stringify(cate)
    })
    if(catelog_search.value != '') search()
    else getBookCatelogs()
}
```

4、检索书目设计

表 3-12 检索书目功能表

功能描述	管理员检索书目
请求消息	1、请求类型：POST 2、发送数据：书名 3、接收数据：最大页数、书目信息
异常情况	无

代码展示：

```
function search(){      //检索
    var req = "/searchAdminCatelog"
    fetch(ip + req, {
        method: "post",
        body: JSON.stringify({"name": book_name.value,
             "pageNum":page_num.value})
    }).then(res=>res.json())
    .then(res => {
        if(catelog_show.value){
            catelogs.splice(0, catelogs.length)
            max_catelog_num.value = res[0].max
            for(var i = 1; i< res.length;i++){
                catelogs.push(res[i])
            }
            updateInfo(catelog_load)
        }
```

```
    })
}
```

5、书目翻页设计

表 3-13 书目翻页功能表

功能描述	管理员书目翻页
请求消息	1、请求类型：POST 2、发送数据：当前页数 3、接收数据：最大页数、书目信息
异常情况	1、页数超出范围

代码展示：

```
function pageDown(num){     //1-向前翻页; -1-向后翻页
   if(catelog_show.value) {
       max_page_num.value = max_catelog_num.value
   }
   if((page_num.value == 1 && num == 1) || (page_num.value ==
        max_page_num.value && num == -1)) {
       showHint('没有更多了')
       return
   }
   page_num.value -= num
   if(catelog_show.value) {
       if(catelog_search.value != '') search()
       else getBookCatelogs()
       catelog_num.value -= num
   }
}
```

6、图书增删改查、翻页设计

同书目的增删改查、翻页。

7、借书设计

表 3-14 管理员借书功能表

功能描述	管理员登记借书
请求消息	1、请求类型：POST 2、发送数据：读者ID、ISBN号、管理员ID 3、接收数据：正确为“1”，否则不正确
异常情况	1、读者ID为空 2、读者ID不存在

代码展示：

```
function send_save_btn(){    //借书
    var req = "/sendBook"
    var fdata = formToJson(form);
    var ele = form[2]
    if(ele.value == ''){
        error_msg.value = "读者 ID 不能为空"
    }
    fdata['admin_ID'] = ID.value
    fetch(ip + req, {
        method: "post",
        body: JSON.stringify(fdata)
    }).then(res=>res.text())
    .then(res => {
        if(res != '1'){
            error_msg.value = res  //返回错误信息
            has_error.value = true;
        }
        else{
            showHint('借书成功')
        }
    })
}
```

8、还书设计

表 3-15 管理员还书功能表

功能描述	管理员登记还书，若书目过期，需支付罚金
请求消息	1、请求类型：POST 2、发送数据：读者ID、ISBN号 3、接收数据：无
异常情况	无

代码展示：

```
function return_btn(rt){      //点击还书
    if(rt.penalty == 0){
        var req = "/returnBook"
        fetch(ip + req, {
            method: "post",
            body: JSON.stringify(rt)
        })
```

```
            showHint('还书成功')
        }
        else{
            hint_msg.value = "请支付罚金"
            pay_hint_show.value = true
            showMask(1)
            rt2 = Object.assign(rt)
        }
}
```

§ 3.3.2 后端设计

1、添加书目设计

表 3-16 添加书目功能表

功能描述	输入项	处理描述	输出项
增加书目	接收包含书目信息的数据（名称、作者、出版社、ISBN、日期、办理人员 ID）	检查是否存在相同的 ISBN 号，若存在则返回"ISBN 号已存在"；检查日期是否合法，若不合法则返回"日期非法"；插入新的书目信息到数据库	返回成功插入书目或相关错误信息

代码展示：

```
# 增加书目（接收 name、author、publisher、isbn、time、transactor）
# {'name': '操作系统', 'author': '陈乔乔', 'publisher': '上海大学出版社
', 'isbn': '978-3-16-55556-5', 'time': '2024-01-06', 'transactor': 1}
def addCatalog(data):
    isbn = data["isbn"]
    title = data["name"]
    author = data["author"]
    publisher = data["publisher"]
    publish_date_str = data["time"]
    copies = 0
    librarian_id = data["transactor"]

    # 检查是否存在相同的 ISBN 号
    with UsingMysql() as um:
        check_sql = "SELECT COUNT(*) as count FROM cip WHERE isbn
= %s"
        um.cursor.execute(check_sql, (isbn,))
        result = um.cursor.fetchone()
    if result["count"] > 0:
        return "ISBN 号已存在"
```

```
    else:
        # 检查日期是否合法
        try:
            publish_date =datetime.datetime.strptime(publish_date_str,
'%Y-%m-%d').date()
        except ValueError:
            return "日期非法"
        # 在数据库中插入新的书目信息
        with UsingMysql() as um:
            insert_sql = """
            INSERT INTO cip (isbn, title, author, publisher,
publish_date, copies, librarian_id)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
            """
            um.cursor.execute(insert_sql, (isbn, title, author,
publisher, publish_date, copies, librarian_id))
            return "1"
```

2、修改书目设计

表 3-17 修改书目功能表

功能描述	输入项	处理描述	输出项
修改书目	接收包含书目信息的数据（名称、作者、出版社、旧 ISBN、新 ISBN、日期、办理人员 ID）	检查是否存在指定的旧 ISBN 号，若不存在则返回"ISBN 号不存在"；检查经办人员 ID 是否存在，若不存在则返回"经办人不存在"；更新数据库中指定的书目信息，触发级联更新；发送邮件通知相关读者。	返回成功修改书目、ISBN 不存在或其他错误信息

代码展示：

```
# 修改书目（接收 name、author、publisher、old_isbn、isbn、time、
transactor）
def modCatalog(data):
    old_isbn = data["old_isbn"]
    new_isbn = data["isbn"]
    # 在数据库中删除指定的书目信息
    with UsingMysql() as um:
        # 检查是否存在指定的 ISBN 号
        check_cip_sql = "SELECT COUNT(*) as count FROM cip WHERE isbn
= %s"
        um.cursor.execute(check_cip_sql, (old_isbn,))
        result = um.cursor.fetchone()
        if not check_transactor(data['transactor']):    # 检查
librarian_id 是否存在
```

```
            return("经办人不存在!")

        if result["count"] > 0:
            # update cip, cascade all foreign key
            set_sql = "UPDATE cip SET isbn
= %s,title= %s,author= %s,publisher= %s,publish_date= %s,librarian_id
= %s WHERE isbn = %s"
            um.cursor.execute(set_sql, (
            new_isbn, data['name'], data['author'], data['publisher'],
data['time'], data['transactor'], old_isbn))

            check_borrow_sql = "SELECT email FROM reserve, reader
WHERE reserve.reader_id=reader.reader_id and isbn = %s"
            um.cursor.execute(check_borrow_sql, (new_isbn,))
            emails = list(um.cursor.fetchall())

            check_borrow_sql = "SELECT email FROM borrow,book,reader
WHERE borrow.reader_id=reader.reader_id
and  borrow.book_id=book.book_id and isbn = %s"
            um.cursor.execute(check_borrow_sql, (new_isbn,))
            borrow_emails = um.cursor.fetchall()
            emails.extend(list(borrow_emails))

            if len(emails) > 0:
                # 发送邮件
                sender = EmailSender()
                reciever_list = [item['email'] for item in emails]
                sender.send(reciever_list, "预约提示",
                            "亲爱的读者：\n  您预约/借阅的书目 isbn 号已被
修改，具体信息请登陆查看。")
            return "1"  # 成功删除
        else:
            return "ISBN 号不存在"
```

3、删除书目设计

表 3-18 删除书目功能表

功能描述	输入项	处理描述	输出项
删除书目	接收包含书目信息的数据（名称、作者、出版社、ISBN、日期、数量、经办人员 ID、是否被预约）	检查是否存在指定的 ISBN 号，若不存在则返回"ISBN 号不存在"；检查 borrow 表中是否有该 ISBN 号对应的图书未归还，若存在则返回"此 ISBN 还存在图书未归还，无法删除"；删除数据库中指定的书目信息及相关图书，触发级联删除；发送邮件通知相关读者。	返回成功删除书目、ISBN 不存在、存在未归还图书或其他错误信息

代码展示：

```
# 删除书目（接收 isbn）
def delCatalog(data):
    with UsingMysql() as um:
        # 检查是否存在指定的 ISBN 号
        check_cip_sql = "SELECT COUNT(*) as count FROM cip WHERE isbn
= %s"
        um.cursor.execute(check_cip_sql, (isbn,))
        result = um.cursor.fetchone()
        # 检查 borrow 表中是否有此 isbn
        check_borrow_sql = "SELECT COUNT(*) as count FROM borrow, book
WHERE isbn = %s"
        um.cursor.execute(check_borrow_sql, (isbn,))
        borrowed = um.cursor.fetchone()
        if borrowed["count"] > 0:
            return "此 isbn 还存在图书未归还,无法删除"

        if result["count"] > 0:
            # 如果存在，则执行删除操作
            del_sql = "DELETE FROM cip WHERE isbn = %s"
            um.cursor.execute(del_sql, (isbn,))
            # delete all books on this isbn
            del_book_sql = "DELETE FROM book WHERE isbn =%s"
            um.cursor.execute(del_book_sql, (isbn,))
            return "1"  # 成功删除
        else:
            return "ISBN 号不存在"
```

4、检索书目设计

以还书信息检索为例

表 3-19 检索书目功能表

功能描述	输入项	处理描述	输出项
管理员还书信息检索	接收包含查询条件和当前日期的数据（最大页数、名称、作者、出版社、ISBN、日期、数量、经办人员 ID）	查询数据库中的借书信息，根据查询条件筛选结果，构建返回结果（包括最大页数、每页信息）	返回包含最大页数和每页信息的 JSON 字符串

代码展示：

```
#管理员还书信息检索（最大页数 + name、author、publisher、isbn、time、
num、transactor）
def searchAdminReturn(data,current_date):
    with UsingMysql() as um:
        # 查询借书信息
        search_borrow = '''
                    SELECT
reader_id,title,book.isbn,borrow_date,due_date,return_date,
                    DATEDIFF(%s,due_date) AS 'date_diff'
                    FROM borrow,book,cip
                    WHERE borrow.book_id=book.book_id and
book.isbn=cip.isbn
                    and return_date is not null and title LIKE %s
        '''
        um.cursor.execute(search_borrow,(current_date, '%' + name +
'%',))
        datas = um.cursor.fetchall()
        # 构建返回结果
        # 使用自定义的 JSONEncoder 来处理日期类型的序列化
    return json.dumps(books, cls=CustomEncoder, ensure_ascii=False)
```

5、翻页设计

以书目翻页为例

表 3-20 书目翻页功能表

功能描述	输入项	处理描述	输出项
传输管理员书目信息	接收包含查询条件的数据（最大页数、名称、作者、出版社、ISBN、日期、数量、经办人员 ID）	查询数据库中的图书信息，查询每本书在流通室的可借复本数量，根据查询条件筛选结果，构建返回结果（包括最大页数、每页信息）	返回包含最大页数和每页信息的 JSON 字符串

代码展示：

```
# 传输管理员书目信息（最大页数、name、author、publisher、isbn、time、
num、transactor + 是否预约）
def adminCatelogs(data):
    with UsingMysql() as um:
        # 查询图书信息
        um.cursor.execute("SELECT * FROM cip")
        datas = um.cursor.fetchall()

        # 查询每本书在流通室的可借复本数量
```

```
        num_by_isbn = {}
        for data in datas:
            isbn = data["isbn"]
            um.cursor.execute(
                "SELECT COUNT(*) as num FROM book WHERE isbn = %s AND
location = '图书流通室' AND status = 0", (isbn,))
            result = um.cursor.fetchone()
            num_by_isbn[isbn] = result["num"]

        # 构建返回结果
    return json.dumps(books, cls=CustomEncoder, ensure_ascii=False)
```

6、借书设计

表 3-21 管理员借书功能表

功能描述	输入项	处理描述	输出项
借书	姓名（name） ISBN 号（isbn） 读者 ID（ID） 管理员 ID（admin_ID）	1. 检查该读者的借书记录数量是否已经达到 10 本，如果是则返回"借书过多，每位读者最多借阅 10 本书"。 2. 检查读者号是否存在，如果不存在则返回"读者号不存在"。 3. 从图书表中获取给定 ISBN 的第一本书的 book_id。 4. 如果找到了可借的书籍，则在借书表中插入新的记录，记录包括读者 ID、管理员 ID、书籍 ID、借书日期、到期日期。 5. 更新图书表中对应的 book_id 记录的 status 字段为已借出（1）。 6. 如果存在预约，则删除预约表中对应的信息。	- 如果借书成功，则返回"1"。 - 如果读者借书数量已满，则返回"借书过多，每位读者最多借阅 10 本书"。 - 如果读者号不存在，则返回"读者号不存在"。 - 如果找不到可借的书籍或者无法借出，则返回"该 ISBN 号的图书不存在或者无法借出"。

代码展示：

```
# 借书（接收 ID、isbn、name）、python（借书日期、到期日期、罚金） 一个读者只能借 10 本书
def sendBook(data):
    borrow_date = datetime.datetime.now().strftime('%Y-%m-%d')
    length = 60  # 借书的最大天数
    due_date = (datetime.datetime.now() +
datetime.timedelta(days=length)).strftime('%Y-%m-%d')
```

```
    with UsingMysql() as um:
        # 检查该读者的借书记录数量是否已经达到 10 本
        check_borrow_count_sql = "SELECT COUNT(*) as count FROM borrow
WHERE reader_id = %s"
        um.cursor.execute(check_borrow_count_sql, (rid,))
        borrow_count_result = um.cursor.fetchone()
        borrow_count = borrow_count_result["count"]
        if borrow_count >= 10:
            return "借书过多，每位读者最多借阅 10 本书"
            # 在借书表中插入新的记录
            insert_borrow_sql = """
            INSERT INTO borrow (reader_id, librarian_id, book_id,
borrow_date, due_date)
            VALUES (%s, %s, %s, %s, %s)
            """
            um.cursor.execute(insert_borrow_sql, (rid, lid, book_id,
borrow_date, due_date))
            # 更新图书表中对应的 book_id 记录的 status 字段
            # 如果存在预约，则删除预约表信息
            return "1"  # 借书成功
        else:
            return "该 ISBN 号的图书不存在或者无法借出"  # 在图书表中找不到
ISBN
```

7、还书设计

表 3-22 管理员还书功能表

功能描述	输入项	处理描述	输出项
还书	1. 读者 ID (ID) 2. 书名 (name) 3. ISBN 号 (isbn) 4. 还书时间 (time) 5. 到期日期 (ddl) 6. 罚金 (penalty) 7. 是否已还书 (returned)	1. 检查该读者在指定还书时间是否借阅了指定书籍，如果不存在则返回"此借阅信息不存在"。 2. 更新借书表中对应的记录的还书时间为当前时间。 3. 更新图书表中对应的 book_id 记录的 status 字段为未借出（0）。 4. 查询预约表中是否有对该书的预约记录，如果有则通知相关预约的读者。 5. 更新预约表中对应的记录的 email_sended 字段为已发送邮件（1）。	- 返回包含罚金信息的 JSON 字符串，如果无罚金则罚金为 0。 - 如果借阅信息不存在，则返回"此借阅信息不存在"。

代码展示：

```
# 还书（接收 ID、name、isbn、time、ddl、penalty、returned）
# 返回 penalty，错误返回错误信息(读者 ID 不存在)
def returnBook(data, current_date):
    with UsingMysql() as um:
        # 检查该读者此书的借书情况
        um.cursor.execute(check_borrow_sql,
(current_date,data["time"],data["ID"],data["name"],data["isbn"]))
        borrow_result = um.cursor.fetchone()
        # 还书
        return_book = '''UPDATE borrow SET return_date=%s
                    WHERE borrow.book_id=%s and borrow.borrow_date=%s
                    and reader_id=%s and return_date is null'''
        # 更新图书表中对应的 book_id 记录的 status 字段
        #通知预约的读者
        if len(reserve_list) > 0:
            # 发送邮件
            sender = EmailSender()
            email = [item['email'] for item in reserve_list]
            sender.send(email, "预约提示",
                        "亲爱的读者\n  您预约的《{}》 书目已经可以借阅，具
体信息请登陆查看。".format(data["name"]))
        # 更新 email_sended，保证只发送 1 次邮件
    return json.dumps(book_info, cls=CustomEncoder,
ensure_ascii=False)
```

§ 3.4 读者功能模块设计

§ 3.4.1 前端设计

1、预约图书设计

表 3-23 预约图书功能表

功能描述	读者预约图书
请求消息	1、请求类型：POST 2、发送数据：读者ID、ISBN号 3、接收数据：无
异常情况	无，因为只有当没有剩余书时，才能够预约

代码展示：

```
function res_btn(catelog){  //点击预约
    var req = "/userReserve"
    fetch(ip + req, {
        method: "post",
        headers: {'content-type': 'application/json'},
        body: JSON.stringify({"ID": ID.value, "isbn": catelog.isbn})
    })
    catelog.reserved = "1"
    showHint('预约成功')
}
```

2、取消预约设计

表 3-24 取消预约功能表

功能描述	读者取消预约图书
请求消息	1、请求类型：POST 2、发送数据：读者ID、ISBN号 3、接收数据：无
异常情况	无，因为只有当预约书后，才能够取消预约

代码展示：

```
function cancel_btn(reserve){  //取消预约
    var req = "/cancelReserve"
    fetch(ip + req, {
        method: "post",
        headers: {'content-type': 'application/json'},
        body: JSON.stringify({"ID": ID.value, "isbn": reserve.isbn})
    })
    showHint('取消成功')
    if(reserve_search.value == '') getReserves()
    else search()
}
```

3、检索书目设计

表 3-25 检索书目功能表

功能描述	读者检索书目
请求消息	1、请求类型：POST 2、发送数据：读者ID、检索书名 3、接收数据：最大页数、书目信息
异常情况	无

代码展示：

```
function search(){      //检索
    var req = "/searchUserCatelog"
    fetch(ip + req, {
        method: "post",
        body: JSON.stringify({"ID": ID.value, "name": book_name.value,
             "pageNum":page_num.value})
    }).then(res=>res.json())
    .then(res => {
        if(catelog_show.value){
            catelogs.splice(0, catelogs.length)
            max_catelog_num.value = res[0].max
            for(var i = 1; i< res.length;i++){
                catelogs.push(res[i])
            }
            updateInfo(catelog_load)
        }
    })
}
```

4、书目翻页设计

表 3-26 书目翻页功能表

功能描述	读者书目翻页
请求消息	1、请求类型：POST 2、发送数据：读者ID、当前页数 3、接收数据：最大页数、书目信息
异常情况	1、页数超出范围

代码展示：

```
function pageDown(num){     //1-向前翻页; -1-向后翻页
   if(catelog_show.value) {
       max_page_num.value = max_catelog_num.value
   }
   if((page_num.value == 1 && num == 1) || (page_num.value ==
        max_page_num.value && num == -1)) {
       showHint('没有更多了')
       return
   }
   page_num.value -= num
   if(catelog_show.value) {
```

```
        if(catelog_search.value != '') search()
        else getBookCatelogs()
        catelog_num.value -= num
    }
}
```

§3.4.2 后端设计

1、预约图书设计

表 3-27 用户预约功能表

功能描述	输入项	处理描述	输出项
用户预约	1. 读者 ID (ID) 2. ISBN 号 (isbn) 3. 当前时间 (python)	使用当前时间和默认的预约期限（10 天），向数据库的预约表中插入新的预约记录。	无返回值
用户取消预约	1. 读者 ID (ID) 2. ISBN 号 (isbn)	在数据库的预约表中删除指定读者 ID 和 ISBN 号的预约记录。	无返回值

代码展示：

```
# 用户预约（接收 ID、isbn、当前时间 python）
def userReserve(data):
    length = 10  # 预约期限为 10 天
    sql = """
    INSERT INTO reserve (reader_id, isbn, reserve_date,
reserve_period)
    VALUES
        (%s, %s, %s, %s)
    """
    with UsingMysql() as um:
        um.cursor.execute(sql, (rid, isbn, date, length))
    return
# 用户取消预约（接收 ID、isbn）
def userCancelReserve(data):
    sql = """
    DELETE FROM reserve
    WHERE reader_id = %s AND isbn = %s
    """
    with UsingMysql() as um:
        um.cursor.execute(sql, (rid, isbn))
    with UsingMysql() as um:
        um.cursor.execute("select * from book")
        datas = um.cursor.fetchall()
```

```
        print(datas)
    return
```

2、取消预约设计

表 3-28 取消预约功能表

功能描述	输入项	处理描述	输出项
用户取消预约	1. 读者 ID (ID) 2. ISBN 号 (isbn)	使用数据库连接对象 UsingMysql()，执行 SQL 删除预约表中指定读者 ID 和 ISBN 号的预约记录。	无返回值

代码展示：

```
# 用户取消预约（接收 ID、isbn）
def userCancelReserve(data):
    rid = data["ID"]
    isbn = data["isbn"]
    sql = """
    DELETE FROM reserve
    WHERE reader_id = %s AND isbn = %s
    """
    with UsingMysql() as um:
        um.cursor.execute(sql, (rid, isbn))

    with UsingMysql() as um:
        um.cursor.execute("select * from book")
        datas = um.cursor.fetchall()
        print(datas)
    return
```

3、检索书目设计

表 3-29 检索书目功能表

功能描述	输入项	处理描述	输出项
用户书目信息检索	1. 书名关键词 (name) 2. 读者 ID (ID) 3. 页码 (pageNum)	1. 使用读者 ID 查询预约表，得到该读者预约的书目 ISBN 集合。 2. 使用书名关键词查询图书表，得到相关的图书信息。 3. 查询每本书在流通室的可借复本数量。 4. 根据分页信息，构建返回结果。	返回包含最大页数和图书信息的 JSON 字符串。

代码展示：

```
# 用户书目信息检索（最大页数 + name、author、publisher、isbn、time、num、
transactor + 是否预约）
def searchUserCatelog(data):
        # 查询每本书在流通室的可借复本数量
        num_by_isbn = {}
        for data in datas:
            isbn = data["isbn"]
            um.cursor.execute(
                "SELECT COUNT(*) as num FROM book WHERE isbn = %s AND
location = '图书流通室' AND status = 0", (isbn,))
            result = um.cursor.fetchone()
            num_by_isbn[isbn] = result["num"]

        # 查询读者的预约信息,构建一个 isbn 的集合
        um.cursor.execute("SELECT isbn FROM reserve WHERE reader_id
= %s", (rid,))
        reserved_books = set(row["isbn"] for row in um.cursor.fetchall())

        # 构建返回结果
            flag = isbn in reserved_books  # 判断是否被当前读者预约
    # 使用自定义的 JSONEncoder 来处理日期类型的序列化
    return json.dumps(books, cls=CustomEncoder, ensure_ascii=False)
```

4、书目翻页设计

同管理员端翻页设计

§ 3.5 邮箱发送功能模块设计

1、邮箱发送类封装

表 3-30 邮箱发送类函数表

功能描述	输入项	处理描述	输出项
邮件发送类初始化	无输入项	1. 创建 `EmailSender` 类实例。 2. 连接到 QQ 邮箱服务器。 3. 使用发件人地址和授权码进行登录。	无返回值
生成邮件体	1. 邮件接收人列表 (`email_to_list`) 2. 邮件标题 (`email_title`) 3. 邮件内容 (`email_content`)	组成邮件体，包括标题、发件人地址、收件人地址、以及邮件正文内容。	返回邮件体对象
发送邮件	1. 邮件接收人列表 (`email_to_list`)	1. 调用 `generate_email_body` 方法生成邮件体。	无返回值

	2. 邮件标题（email_title） 3. 邮件内容（email_content）	2. 使用 SMTP 发送邮件。	
退出服务	无输入项	关闭 SMTP 连接。	无返回值

代码展示：

```
class EmailSender:
    def __init__(self):
        self.smtp = smtplib.SMTP()
        # 连接邮箱服务器地址
        self.smtp.connect('smtp.qq.com')
        # 发件人地址及授权码
        self.email_from_username = 'library_10@foxmail.com'
        self.email_from_password =
        self.smtp.login(self.email_from_username,
self.email_from_password)

    def generate_email_body(self, email_to_list, email_title,
email_content):
        email_body = MIMEMultipart('mixed')
        email_body['Subject'] = email_title
        email_body['From'] = self.email_from_username
        email_body['To'] = ",".join(email_to_list)

        text_plain = MIMEText(email_content, 'plain', 'utf-8')
        email_body.attach(text_plain)
        return email_body

    def send(self,email_to_list, email_title, email_content):
        # 发送邮件
        email_body = self.generate_email_body(email_to_list,
email_title,email_content)
        # 注意：此处必须同时指定发件人与收件人，否则会当作垃圾邮件处理掉
        self.smtp.sendmail(self.email_from_username, email_to_list,
email_body.as_string())

    def exit(self):
        self.smtp.quit()
```

2、系统后台检测功能模块设计

表 3-31 邮箱发送类函数表

功能描述	输入项	处理描述	输出项
系统监测线程类	当前日期(current_date)	1. 初始化系统监测线程，设置监测间隔为 n 秒。 2. 实现 process_reserve 方法，处理过期预约和通知相关读者。 3. 实现 process_borrow 方法，处理到期未归还的借书情况并通过邮件通知读者。 4. 在 run 方法中循环执行 process_reserve 和 process_borrow 方法，并在每次执行后等待 n 秒。	无返回值

代码展示：

```
class System(threading.Thread):
    def __init__(self,current_date, n=5):
        self.n = n
        # 每隔 n 秒执行一次系统监测
        super().__init__()
        self.current_date = current_date
    def process_reserve(self):
        with UsingMysql() as um:
            check_reserve = """SELECT email FROM reserve, reader
                    WHERE reserve.reader_id = reader.reader_id
                    and date_add(reserve_date, interval reserve_period
day)< %s"""
            um.cursor.execute(check_reserve, (self.current_date,))
            emails = um.cursor.fetchall()
            if len(emails) > 0:
                # 发送邮件
                sender = EmailSender()
                reciever_list = [item['email'] for item in emails]
                sender.send(reciever_list, "预约提示",
                            "亲爱的读者\n  抱歉，您的预约已过期，具体信息请登陆
查看。")
            # 通知预约读者有书可借
            reserve_list = '''SELECT reader_id, email FROM reader WHERE
reader_id
                                in (SELECT reader_id FROM reserve WHERE
                                email_sended =0 and
                                isbn in (SELECT DISTINCT book.isbn FROM
book WHERE status=0))'''
            um.cursor.execute(reserve_list)
            emails = um.cursor.fetchall()
```

```
            if len(emails) > 0:
                # 发送邮件
            # 对于已到期且未归还的图书，系统通过 Email 自动通知读者
            borrow_list = """SELECT email FROM borrow, reader
                    WHERE borrow.reader_id=reader.reader_id and due_date
< %s
                    and return_date is null
                    """
            um.cursor.execute(borrow_list, (self.current_date,))
            emails = um.cursor.fetchall()
        if len(emails) > 0:
            # 发送邮件
            sender = EmailSender()
            reciever_list = [item['email'] for item in emails]
            sender.send(reciever_list, "借书超期提示",
                        "亲爱的读者\n  您借阅的书目已到期，请及时归还，否则会产
生罚金。")
    def run(self) -> None:
        while True:
            self.process_reserve()
            self.process_borrow()
            time.sleep(self.n)
```

第4章 系统功能展示

§ 4.1 注册功能

新用户可以通过注册功能录入个人信息，读者信息包括读者ID、姓名、电话和Email，如图4.1-1所示，验证没有重复后，系统会自动发送验证码到邮箱，如图4.1-2和4.1-3所示，验证码输入通过后，系统会为不同读者生成不同的读者ID，注册即成功。

图表 4.1-1 注册界面

图表 4.1-2 发送邮箱验证码

图表 4.1-3 读者接收端邮箱

图表 4.1-4 管理员注册界面

§ 4.2 登录功能

注册完成后，用户和管理员就可以输入手机/工号和密码进行登录，如图4.2-1/-2所示。

图表 4.2-1 读者登录界面

图表 4.2-2 管理员登录界面

§4.3 读者模块

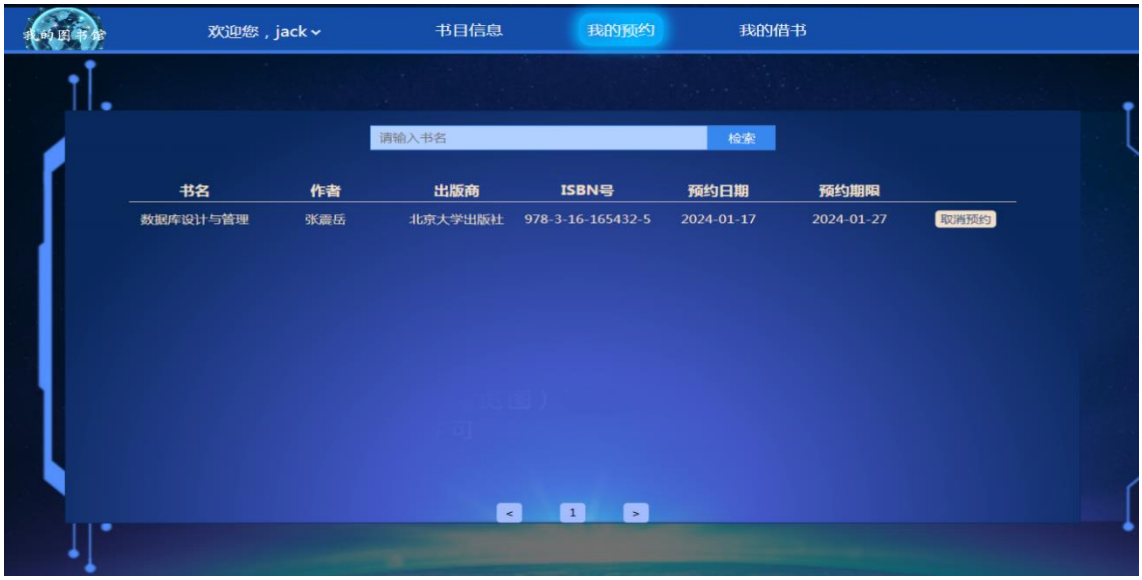
读者首页即为书目信息查看，如图4.3-1，用户可以在此页面进行书目预约，并且进行翻页。

图表 4.3-1 读者主页

图表 4.3-2 翻页功能

用户点击预约后，会提示“预约成功”，如图4.3-3，并且在“我的预约”界面中增加记录，如图4.3-4。

图表 4.3-3 预约功能



图表 4.3-4 查看“我的预约”

此外，读者还可以点击“我的借书”查看本人借阅的图书，如图4.3-5。

图表 4.3-5 查看“我的借书”

§ 4.4 管理员模块

§ 4.4.1 检索功能

管理员也可在所有页面进行检索，只需输入任意位置的部分字段，如图4. 4-1。

图表 4.4-1 搜索“人工”的结果

§ 4.4.2 书目管理

管理员可以在书目管理页面对书目进行信息修改，如图4.4-3，和书目的删除/添加，如图4.4-4。

图表 4.4-2 管理员“书目管理”主页

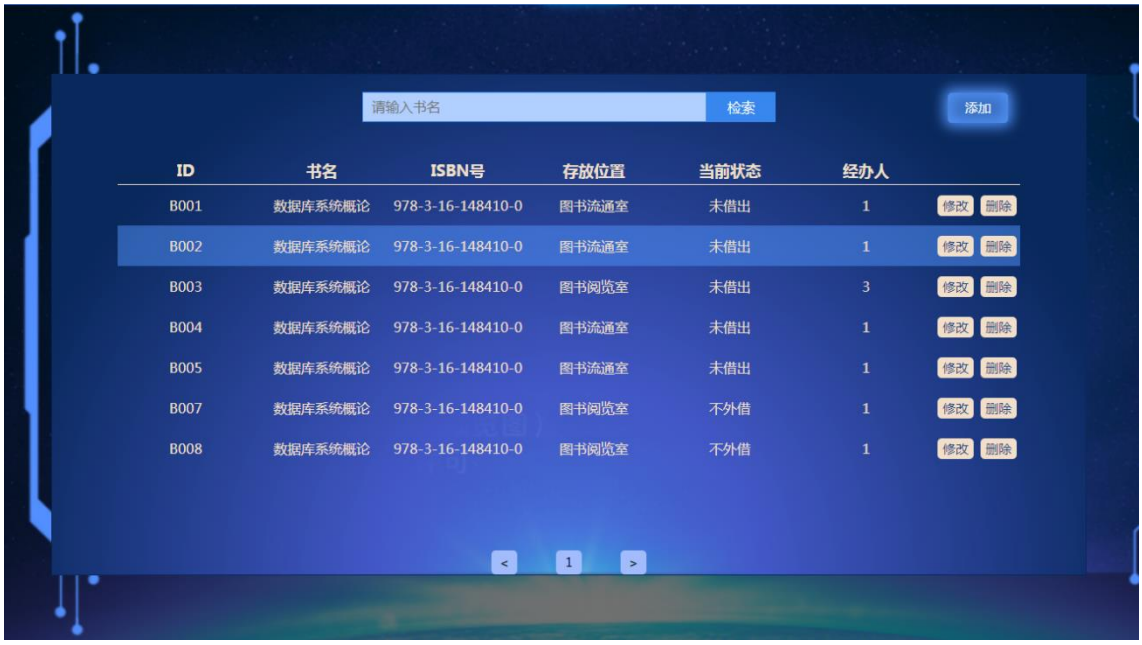
图表 4.4-3 修改书目信息

图表 4.4-4 添加书目

§ 4.4.3 图书管理

管理员可以在图书管理页面对图书进行信息修改，如图4.4-5，和书目的删除，如图4.4-6/7，图书的添加，如图4.4-8/9。

图表 4.4-5 修改图书信息



图表 4.4-6 删除图书B006

图表 4.4-7 删除图书后书目册数减少

图表 4.4-8 添加不存在的图书

图表 4.4-9 新增的图书信息

§4.4.4 借书管理

管理员借书管理界面可以进行借书，登记读者ID，如图4.4-10/11，同时读者端也会新增借书条目，如图4.4-12

图表 4.4-10 借书功能

图表 4.4-11 借书成功

图表 4.4-12 用户界面新增借书信息

§ 4.4.5 还书管理

对于超期归还者，系统自动计算罚金（超时每天罚金0.05），如图4.4-15。系统同时自动查询预约登记表，若存在其他读者预约该书的记录，则通过邮件方式通知预约此书的读者，并将该图书的状态修改为“未借出”。

图表 4.4-13 还书《算法导论》成功

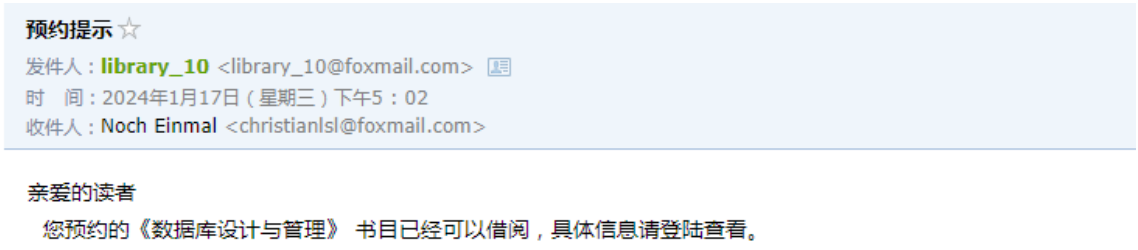
图表 4.4-14 用户端借书信息自动删除

图表 4.4-15 借书超限支付罚金

§ 4.5 系统功能模块

§ 4.5.1 预约提醒

其他用户还书后，系统同时自动查询预约登记表，若存读者预约该书的记录，则通过邮件方式通知预约此书的读者，如图 4.5-1，并将该图书的状态修改为“未借出”。



图表 4.5-1 预约用户收到邮件

§ 4.5.2 预约过期

预约时间为 10 天，到达时限后，系统会邮件通知此读者，如图 4.5-2，同时自动清除超出预约期限的预约记录并修改相关信息，如图 4.5-3。

预约提示 ☆

发件人：library_10 <library_10@foxmail.com>

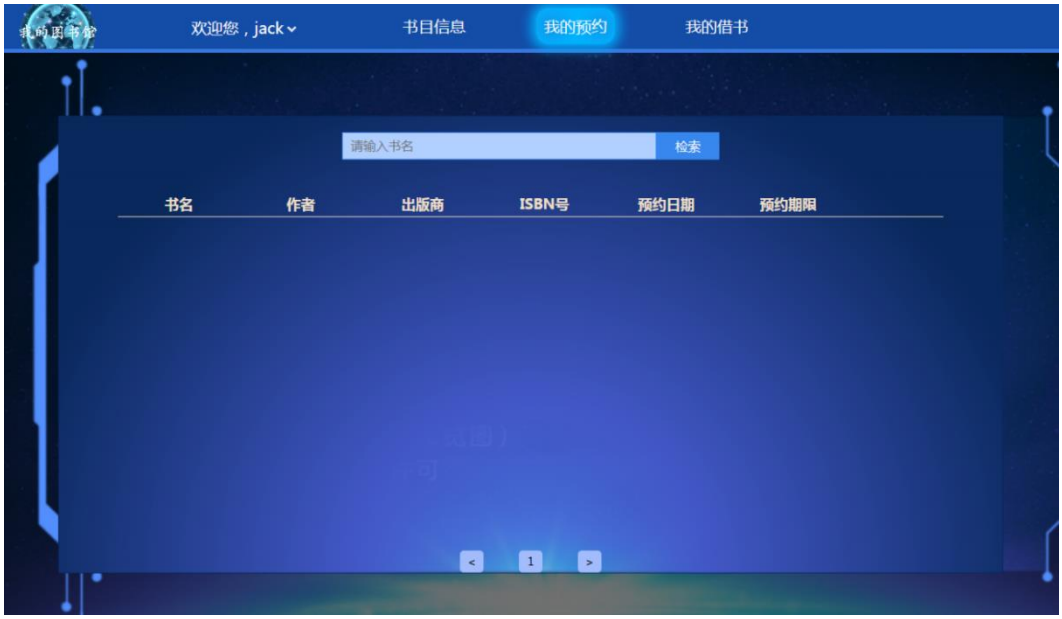
时　间：2024年1月17日（星期三）下午6：15

收件人：Noch Einmal <christianlsl@foxmail.com>

亲爱的读者

抱歉，您的预约已过期，具体信息请登陆查看。

图表 4.5-2 预约超期提醒



图表 4.5-3 预约界面信息自动清除

§4.5.3 借书过期

对于超期归还者，系统自动计算罚金，并且系统会通过 Email 自动通知读者及时归还，如图 4.5-4。

借书超期提示 ☆

发件人：library_10 <library_10@foxmail.com>

时　间：2024年1月17日（星期三）下午6：53

收件人：Noch Einmal <christianlsl@foxmail.com>; lierli <lierli@example.com>;

亲爱的读者

您借阅的书目已到期，请及时归还，否则会产生罚金。

图表 4.5-4 借书过期邮件提醒

第5章 个人小结

§5.1 周孙睿个人小结

在本次图书管理系统的制作过程中，我主要负责前端部分。具体包括用户和管理员的登录、注册界面、用户界面以及管理员界面。

刚开始，我以为前端的开发并不会有多么困难，但是当我实际动手编写时，才发现前端开发远比我想象中的繁琐。

首先，html标签种类繁多，想要完全了解每种标签具体的功能、特性、使用方法，需要不小的学习成本。如果没有完全掌握一种标签的特性，那么十分容易出现“预计界面与实际显现界面不一致”的情况。

其次，css属性多种多样。想要使用哪种属性，就必须先去“开发手册”中寻找该属性的使用方法，需要花费不小的时间成本。

最后，再来谈谈Vue3框架。Vue3框架总体来说还是十分方便的，它能够帮助我构建出结构化、可维护和高性能的前端应用程序。而且，Vue3的响应式数据绑定和虚拟DOM更新机制使得界面更新非常高效，大大简化了原生JS的DOM操作。但是，它的一些特殊语法使用，还是困扰了我一段时间。

在整个开发过程中，我注重与后端开发同学的紧密配合和沟通，确保前后端的数据交互流畅和准确。同时，我也会关注代码的可维护性和可扩展性，尽量将功能模块化，方便日后的维护和升级。我十分感谢两位队友的辛勤付出！

关于课程的体会，我觉得郑老师讲课清晰易懂，课堂氛围也十分活跃。就是研讨方式采用随机选组，会导致一些精心准备的PPT白费，这点不是很好。不过，总体还是很不错的。

§5.2 吕斯路个人小结

在本次数据库课程作业中，我负责开发了一个图书管理系统的后端代码，主要包括用户登录、书目管理、借还书等功能。在开发过程中，我首先进行了文档编写，明确了系统的需求和功能，为后续的开发工作提供了清晰的指导。同时，通过文档编写，我能够更好地与团队成员沟通，确保大家对系统的理解一致。在数据表设计方面，注重表之间的关联关系和数据一致性。通过合理设置主键和外键，确保了数据表之间的关联关系正确，避免了数据冗余和不一致的问题。

其中一个困难是邮件发送端口不匹配的问题。在与邮件发送相关的功能开发中，我发现邮件发送的端口设置不正确导致邮件发送失败。通过查找相关文档和调试，我成功解决了这一问题，确保了系统能够正常发送邮件通知。数据库主键外键的设置也是一个需要注

意的问题。我在设计数据表时，选择了适当的主键和外键设置，采用CASCADE和AUTO INCREMENT等机制，保证了数据的完整性和一致性。

总的来说，通过这次图书管理系统的后端开发，我不仅掌握了一些关键的开发技能，还学到了很多关于团队协作、文档编写和问题解决的经验。在未来的学习和工作中，我将继续努力提升自己的技能，不断优化和改进自己的开发流程，以更好地应对各种挑战。
对于教学的建议：老师的上课讲解都很详细完备，但是我个人不是很喜欢的一点是：这学期的研讨方式，随机抽取导致部分同学无法被抽到，每周花费很多时间认真制作的PPT也都浪费了；同时很多同学为了加分，问一些很偏的奇怪问题，反而没有意义。希望能改进一下研讨方案，不过这只是我作为一个社恐的个人想法，还请老师多多包涵。

§5.3 邹凌杰个人小结

在数据库大作业中，我第一次完整参与到一个系统的设计中，感受数据库与后端代码，前端界面结合的神奇与魅力之处。在图书管理系统的后端部分，我主要负责设计一些增删改查的接口，主要包括数据库交互、函数设计、异常处理等方面的工作。我创建了多个函数来实现系统的各种功能，如添加图书、借书、删除书目等。使用了多线程来处理不同的请求，提高了系统的并发性。实现了自定义的 JSON 编码器，用于将 datetime 对象转换为 ISO 格式的字符串，以便在 JSON 中序列化。

在数据库大作业的实践中，我感受到把所学尽数投入到实践过程中的成就感，从总体设计的需求与可行性分析中，我也感受到一个系统追求完备性的来之不易，写完报告后，有一种来之不易与淋漓尽致感觉，特别感谢和我组队的两位同学，我感觉我们都发挥出了自己百分之两百的热情，制作出了让我们感到很欣慰的系统大作业。

对于课程研讨方式，我有一点小小的需要吐槽的地方，因为我们组是最后一周才有机会研讨，所以我们从第二周开始就每周都做了研讨作业，然而研讨以又是随机抽签的形式展开，导致我们每周做的ppt白费了五六次，有点难受(┬＿┬)。不过就课程教学方式而言，我很感谢老师，在课堂上不囿于浅显的知识点，带着我们做了很多需要思考的题目，加深了我们对于知识点的理解程度，这是我很喜欢的一个点。