济南大学

软件工程课程设计

题　　目　某大学校园一卡通系统用户管理子系统研究与设计

学　　院　信息科学与工程学院

专　　业　计算机科学与技术

班　　级　计科 1703

学　　生　刁承坤

学　　号　20171222129

指导教师　杜立新

二〇二〇 年 六 月 三十 日

摘　　要

随着社会的进步与变革，各学校原有的消费和管理模式已不能适应新的发展要求，基于目前现状“校园卡”应运而生。所谓“校园卡”即在学校内，凡有现金、票证或需要识别身份的场合均采用卡来完成。通过高校一卡通管理系统的设计与实现，不仅使得校园实现了管理的数字化和自动化，还会是一卡通管理者信息管理的好帮手，提高了管理的水平，实现迅速、全方位的一卡通信息获取与处理，给全校的师生的日常生活提供了便利，而且具有很好的应用前景和经济效益，此种管理模式代替了传统的消费管理模式，提升了学校管理的高效性、方便性与安全性。

建立先进的信息管理系统是实现高等教育现代化的必由之路，而需要识别身份的场合均采用卡来完成。此种管理模式代替了传统的消费管理模式，为学校管理带来了高效、方便与安全的同时，也是推进高校信息化管理的重要举措之一。学生只需在相关银行开设帐户并存入金额，即可通过本软件的接口去向一卡通中充值，学生可使用校园卡通进行消费，借阅图书，申请宿舍等。实现 “一卡在手，走遍校园”。

关键词：校园卡；一卡通；方便；安全；高效

目　　录

1 可行性分析

1.1 要求

（1） 系统的整体性

系统开发需要内部的整体性，需要与各个阶段和各个模块相互配合，包括管理员子系统，图书管理子系统，教学管理子系统，卡务管理子系统等系统之间的整体联系。

（2） 系统的实用性

我们的目标是该卡应用到实际情况中时，在校区的所有学生都可以使用，能够用来识别学生的身份信息、消费，借阅图书等多种功能，确保学生能够实现一卡在手，走遍校园的最终目的。

（3）系统的可靠性

该系统比较稳定，数据备份在服务器上，做好日志备份，一旦服务器数据库数据丢失能够及时的恢复，不影响学生校园的正常生活。可靠性是该系统的首要条件。

（4）人机交互

使用简介的界面，方便学生的使用，便于管理更加智能。所有信息都有相应的密码保护，所有的密码都采用 MD5 码加密，而后在保存到数据库中，确保用户信息的安全，一旦数据库泄露，能够最小程度的减少学生和学校的损失。

1.2 目标

本项目的目标旨在为学生设计通用的一卡通系统，能够让学生通过一卡通取代原来繁杂的水卡、澡卡、借书卡等证件，同事也能方便学校进行管理，能够更好的保障学生的生命安全和财产安全。

1.3 可行性研究的方法及评价尺度

从技术角度来说，我们打算采用 C/S 架构，将数据库部署在服务器端，能够方便用户在个人终端上进行安装操作，针对不同身份的用户提供相应的权限。

从用户角度来说，在项目施行的前期我们做好调研工作，针对不同职能部门和学生的不同需求完善系统的功能，确保能够为学校的大部分师生所接受。

评价尺度是要看用户使用学校一卡通系统是否方便和安全，以及管理人员管理起来是否简便和它出错情况以及软件的安全性。

2 需求分析

针对校园内通用的校园卡需要统一管理这一需求，我们需要较方便的实现用户的登陆以及管理等服务，同时针对不同的身份有不同的使用界面，能够实现对所有用户信息及账户信息的管理。

2.1 系统的功能性需求

随着校园卡的快速发展，各部门逐步建立起了各自的用卡系统，比如图书馆管理系统、食堂管理系统、机房管理系统等等，每个单位都独立发卡，独立结算。这样使得学校师生每人手中都持有多张证卡，例如学生证、借书证、工作证、饭卡、上机卡，证件功能单一，多种证件共存，师生携带和使用很不方便，也造成资源的极大浪费。同时，由于各部门用卡系统相互独立，应用分散、数据分散，互不兼容容易给学校增加一些不必要的人力、财力的开销，同时也不方便广大师生。 为了解决这一问题，方便广大师生，提高学校的管理服务水平，整合学校现有资源，借助校园网建立校园一卡通系统，实现应用集中、数据集中、设备集中，实现各校区、各类收费和各种身份识别的一卡通行，取代原有的各种证件，支持交易支付、身份识别、图书借阅、水控等功能。系统将采用银行卡与校园卡校务管理功能相整合的方式，以校园卡为主、不涉及银行卡的金融功能，由学校发行校园卡。其功能主要是首先要能替代现有的各种证件实现身份识别及校园内的各种消费场所，从而实现“一卡多用”的目标。每个学生在入学时得到唯一的个人账号和相应的卡片。该管理系统不只单纯收费，还要具有学校内部管理的功能。系统的建成后，将在最大限度上使学校的现有资源得到合理利用，避免资源浪费，减少投资金额。

整个用户管理子系统分为普通用户和系统中心管理员和子系统管理员，其中系统中心管理员权限最高，可以修改子系统管理员的信息，对子系统管理员进行增删改查等操作，是系统中权限最高的实体。再次子系统管理员可以对其相应的子系统的普通用户进行管理，进行增删改查等操作，不过涉及到学校信息的时候要经过系统中心管理员的审核。最后权限最低的，使用人数最多的是普通用户也就是学生，他们只能修改一些常用的基本属性。可以进行消费查询等基本操作，其操作涉及到子系统信息时需要经过子系统管理员的审核。

系统中心管理员可以对子系统管理员、学生用户进行批量删除增加，办卡等一系列操作，方便学校统一管理。子系统管理员在系统中心管理员的审核下进行日常子系统的维护和管理工作，与学生交互。

用户管理子系统的系统结构图如图 2-1 所示。

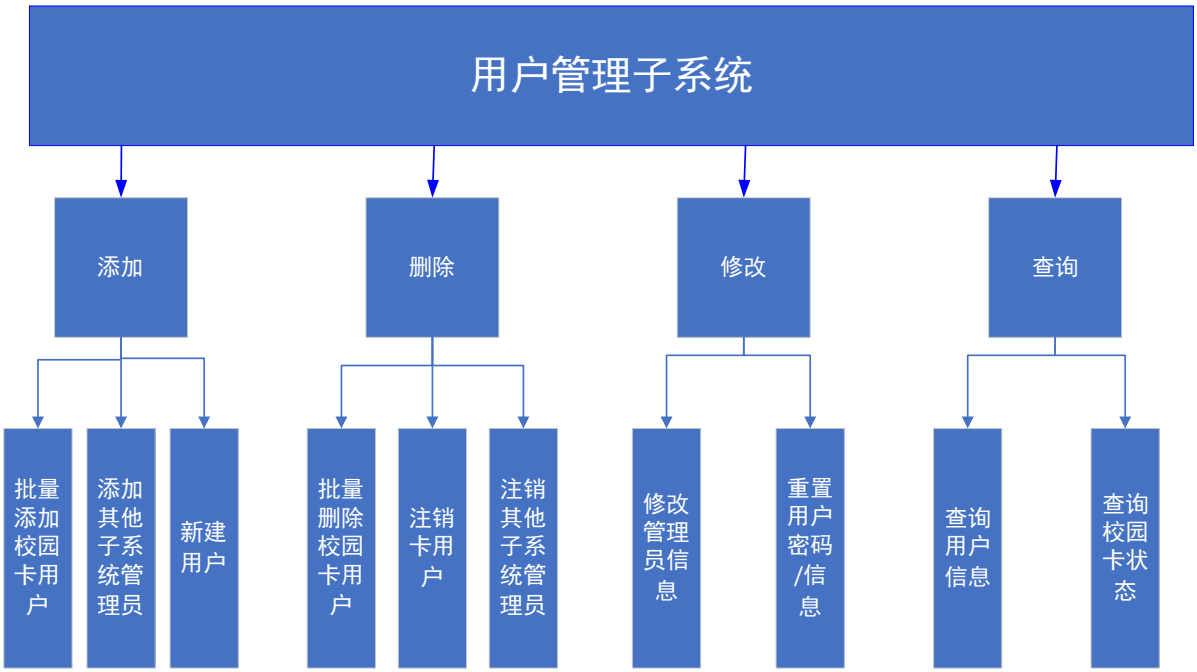

图 2-1 系统结构图

系统的类图如图 2-2 所示。

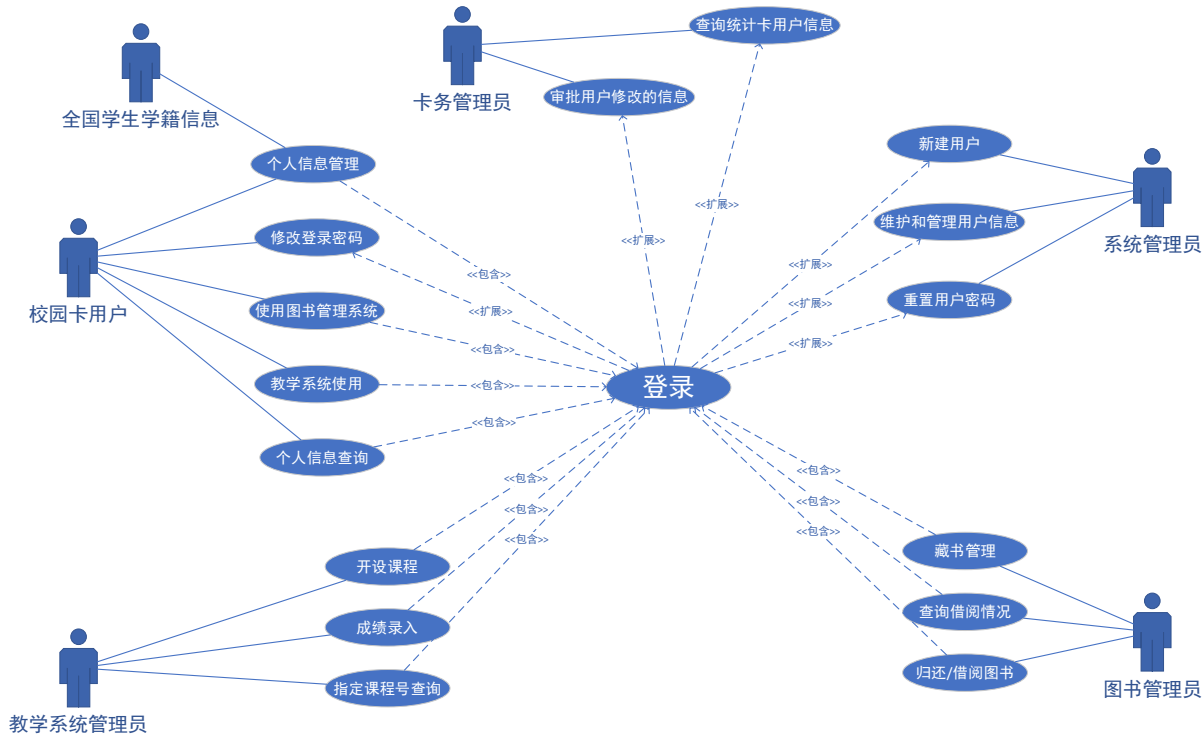

图 2-2 系统总用例图

在用上述用例图中，可以将“个人信息管理”用例细化，如图 2-3 所示。

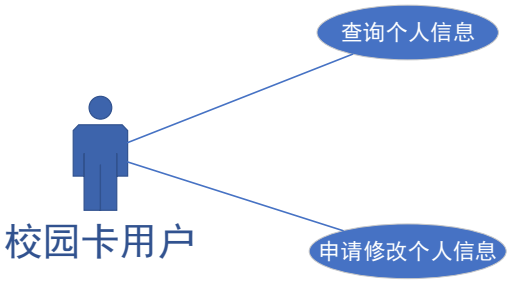

图 2-3 个人信息管理用例细化

将“图书系统使用”用例细化，如图 2-4 所示。

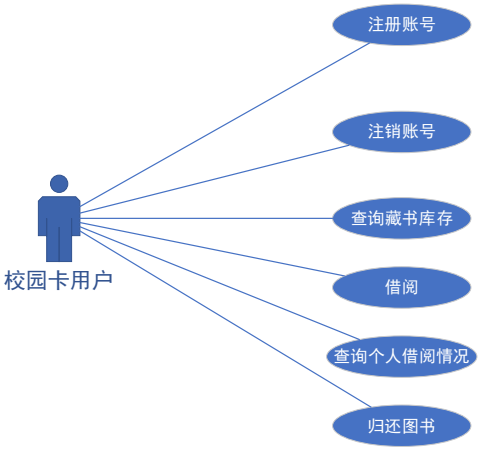

图 2-4 图书系统使用用例细化

将“教学系统使用”用例细化，如图 2-5 所示。

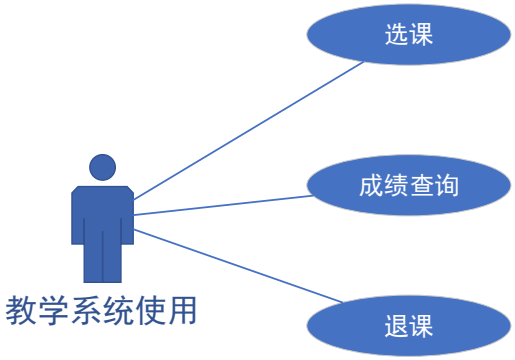

图 2-5 教学系统用例细化

将系统中心管理员的“新建用户”用例细化，如图 2-6 所示。

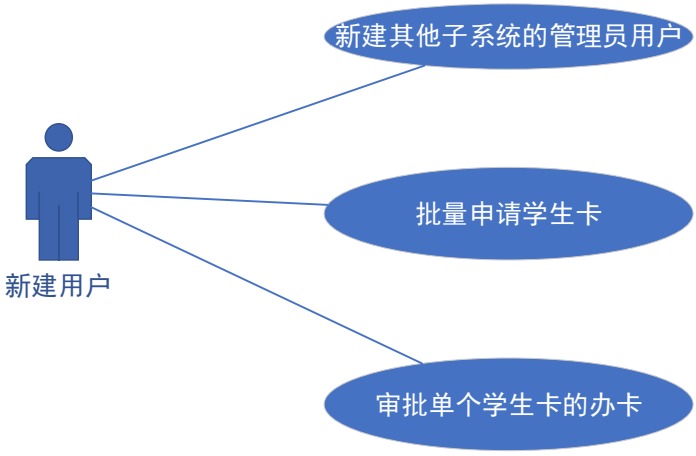

图 2-6 新建用户用例细化

将系统中心管理员的“维护用户信息”用例细化，如图 2-7 所示。

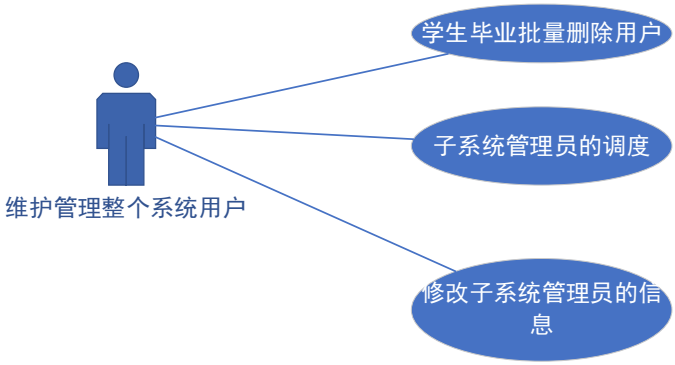

图 2-7 维护用户信息用例细化

建立相应的用例规约描述：

用例名称	基本事件流
登录	当管理员或卡用户登录系统时，用例启动。 （1） 系统提示用户输入用户名和密码，选择用户类型（子系统管理员或普通用户）。 （2） 管理员或用户输入用户名和密码。 （3） 系统验证输入的用户名和密码，若正确则管理员或用户登录到系统中。若错误则返回到登录界面并提示错误信息。
查询用户个人信息	当用户查询用户个人信息时，用例启动。 （1） 点击查询用户个人信息。 （2） 系统检验当前用户是否有相应的权限执行操作。 （3） 从数据库中获取信息并显示。

续表：

修改相关信息	当用户选择修改相关信息时，用例被启动。 （1） 用户选择“修改相关信息”。 （2） 系统检查当前用户的可修改权限和可修改项。从数据中获取可修改的原始信息（邮箱、电话、地址等）。 （3） 用户填写新的要修改的信息。 （4） 点击保存，更新数据库信息。
修改登录信息	当用户选择修改登录信息时，用例被启动 （1） 用户点击“修改登录信息”。出现更改密码界面。 （2） 用户填写旧密码，和两次新密码。 （3） 系统从数据库中获取旧密码进行验证。 （4） 若验证通过，系统修改用户登录信息数据库，并提示修改成功，返回登录界面重新登录。
批量增加用户	当系统中心管理员点击批量添加时，用例启动。 （1） 管理员从新生入学系统中导入所有新生的学生信息。 （2） 将这些信息与全国身份认证系统核实，无误后将这些信息导入学校校园卡用户信息数据库。 （3） 通知制卡商制作相应数量的校园卡并通知缴费。 （4） 将校园卡发放到学生手中。
批量删除用户	当学生毕业，系统管理员点击批量删除时，用例启动。 （1） 管理员收到学生毕业信息，并检查毕业手续是否齐全合格。 （2） 从校园卡数据库中查出所有满足条件的学生信息，并检查是否有欠费、违规等未交费现象。 （3） 检查无误后，从校园卡信息数据库中删除满足条件的所有校园卡信息，校园卡失效。

用户管理子系统的数据流图如图 2-8 所示。

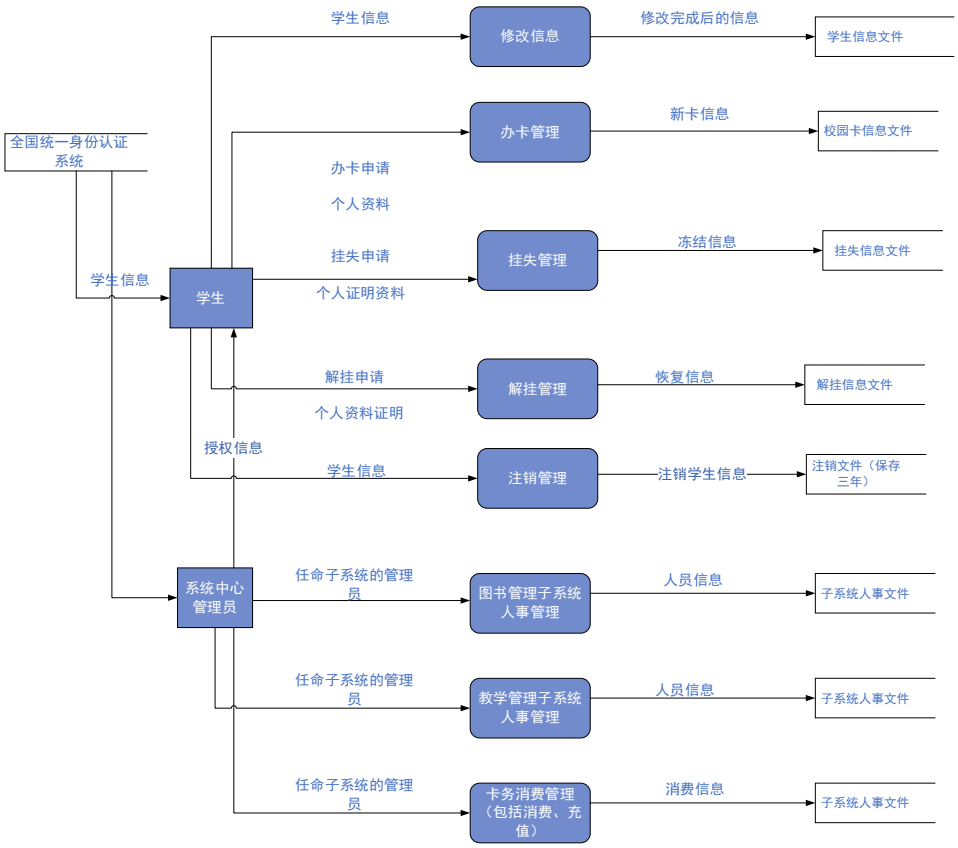

图 2-8 数据流图

用户管理子系统类图如图 2-9 所示。

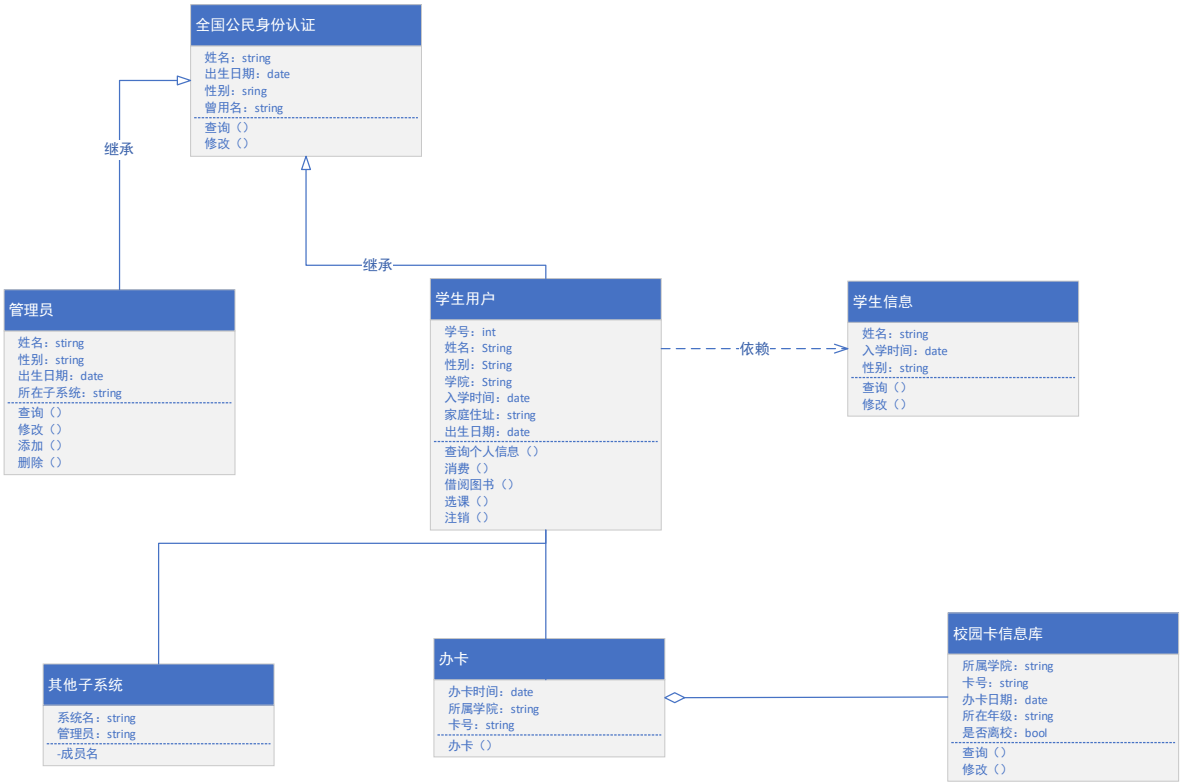

图 2-9 类图

用户管理子系统中，学生修改个人信息的活动图如图 2-10 所示。

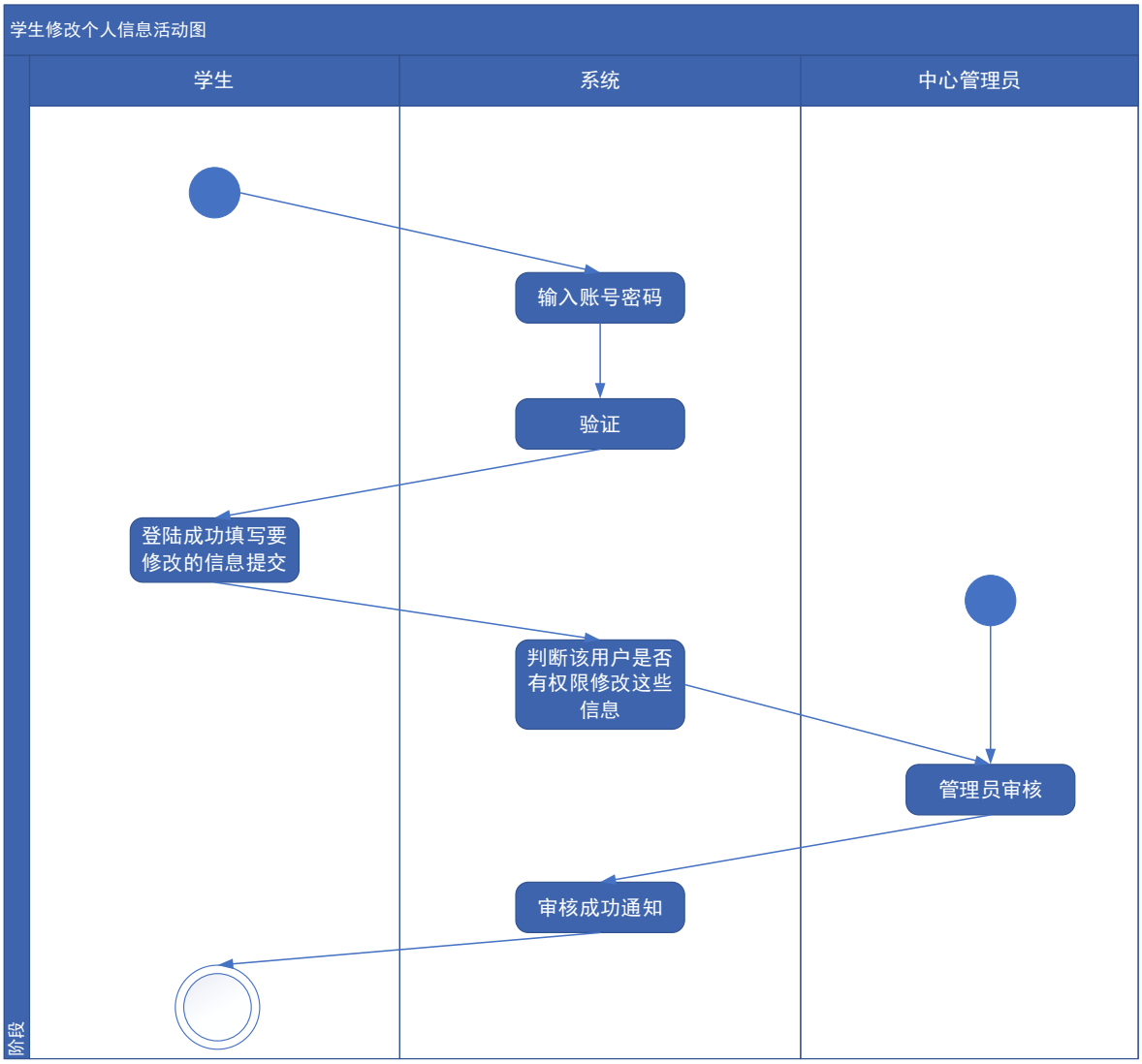

图 2-10 学生修改个人信息活动图

因学生毕业，系统管理员注销学生校园卡的活动图如图 2-11 所示。

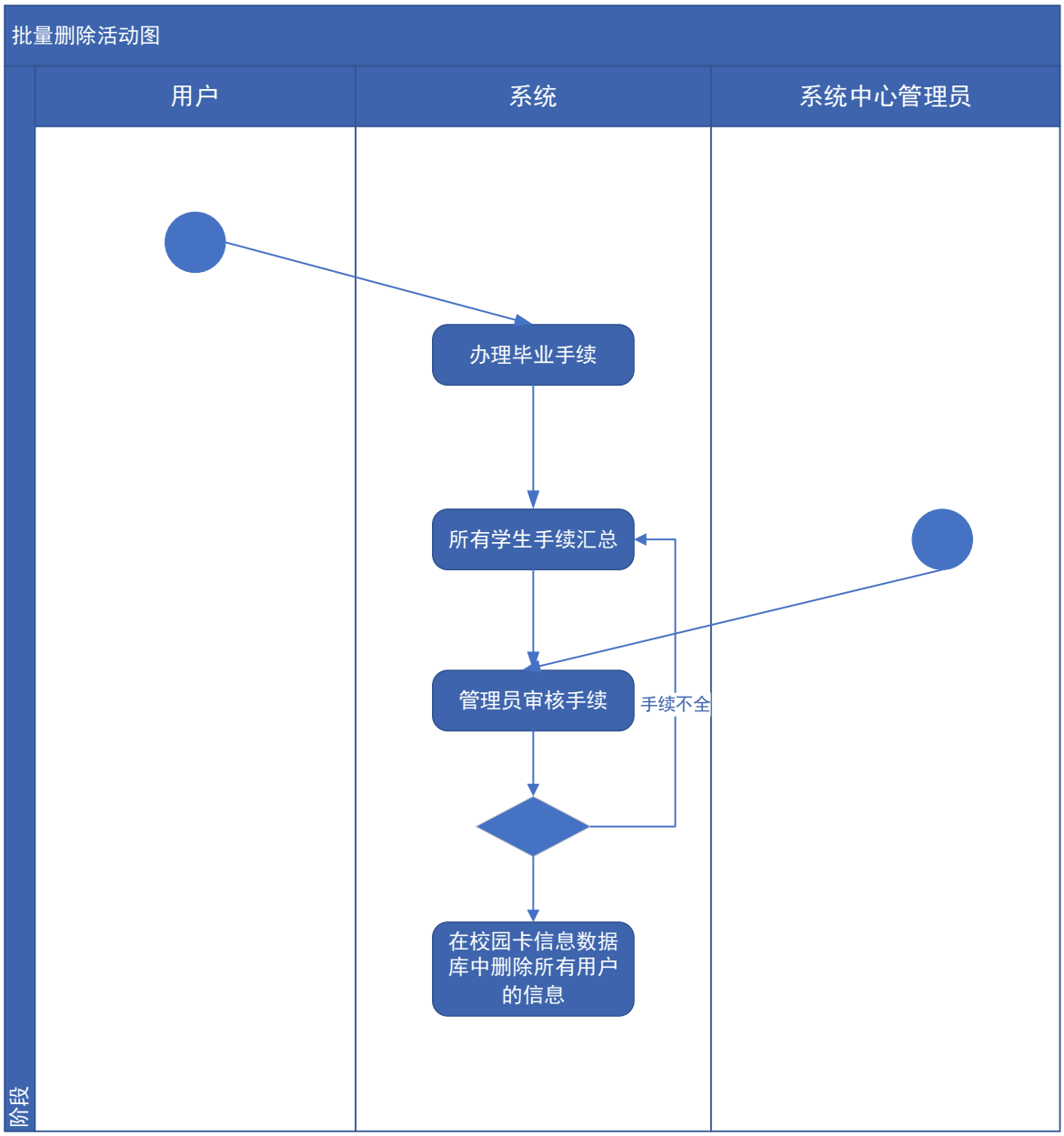

图 2-11 注销校园卡活动图

2.2 系统的性能需求

易用性、可靠性、安全性等方面的问题，是该系统中的性能需求。这种需求不会直接影响系统功能，而是为了保证系统能够给在较长时间内快速、稳定、安全、高效的运行，本系统采用传统的 C/S 系统架构，能够防止内存泄漏等问题，在计算机硬件日益发展的环境下性能差距越来越小。

2.3 系统的安全需求

增加权限管理功能，此系统对不同的用户使用权限进行分类，针对不同的用户，赋

予不同的权限，设置系统中心管理员，系统中心管理员可以设置其他子系统的管理员，包括图书子系统管理员、卡务子系统管理员、教学管理子系统管理员等，然后不同的子系统管理员有对应的操作界面和不同的权限，能够有效地避免产生不同权限用户之间的误操作。

3 系统设计

3.1 系统架构

整个系统基于校园网这个基础平台上，一卡通系统通过公用数据平台的建立连接到数字化校园平台上从而实现数据共享。系统采用 3 层架构，如图 3-1 所示。

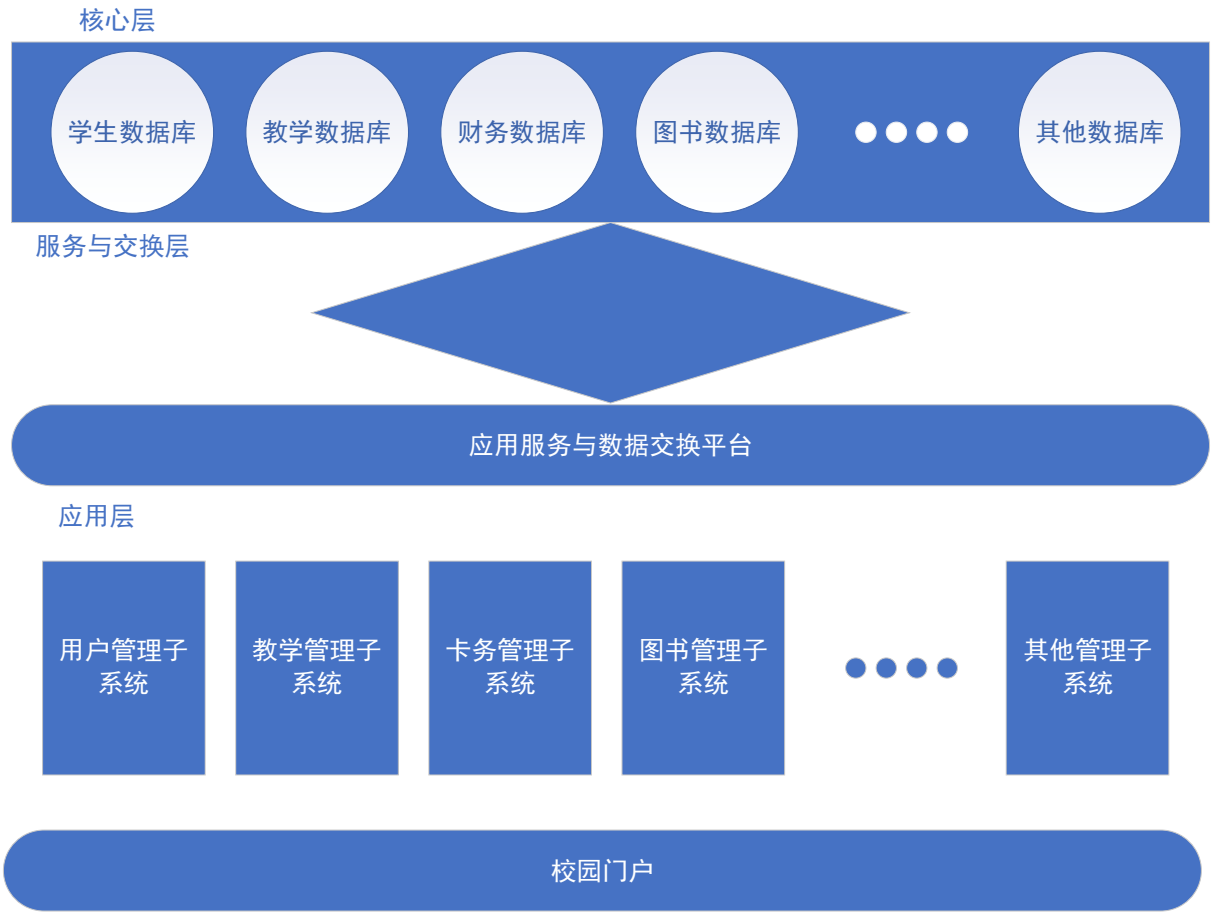

图 3-1 系统架构图

相应的，在此整体系统架构图之下，完成用户管理子系统的设计与开发，其子系统的系统架构图如下图 3-2 所示。

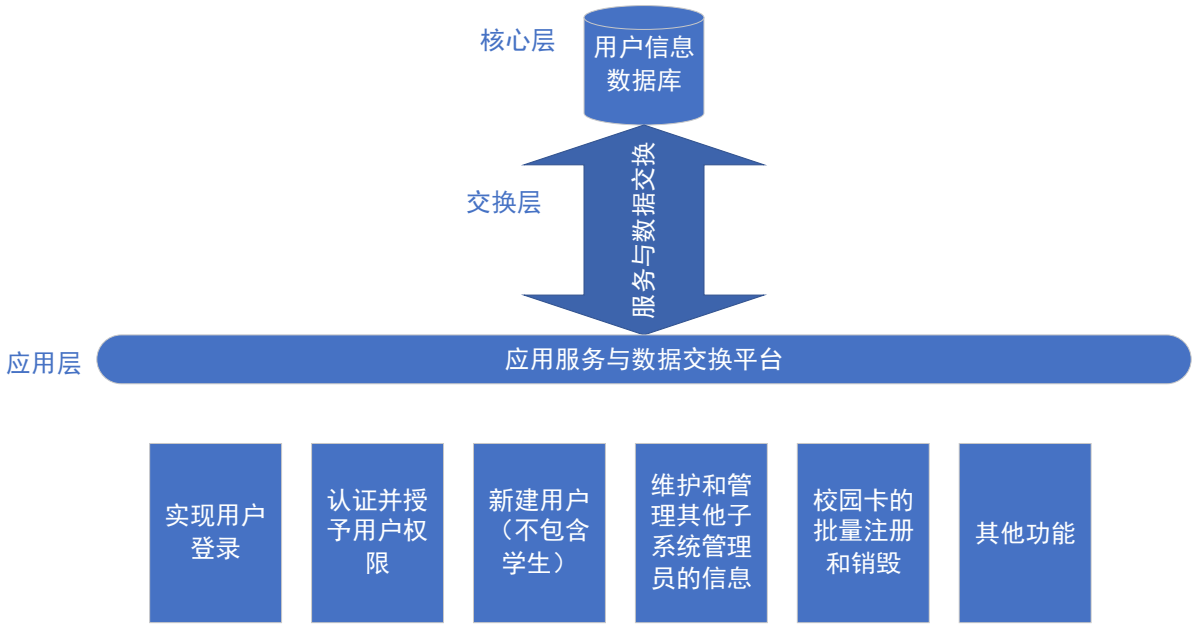

图 3-2 用户管理子系统架构图

3.2 系统功能

3.2.1 用户卡注册

用户卡注册的流程图如图 3-3 所示：

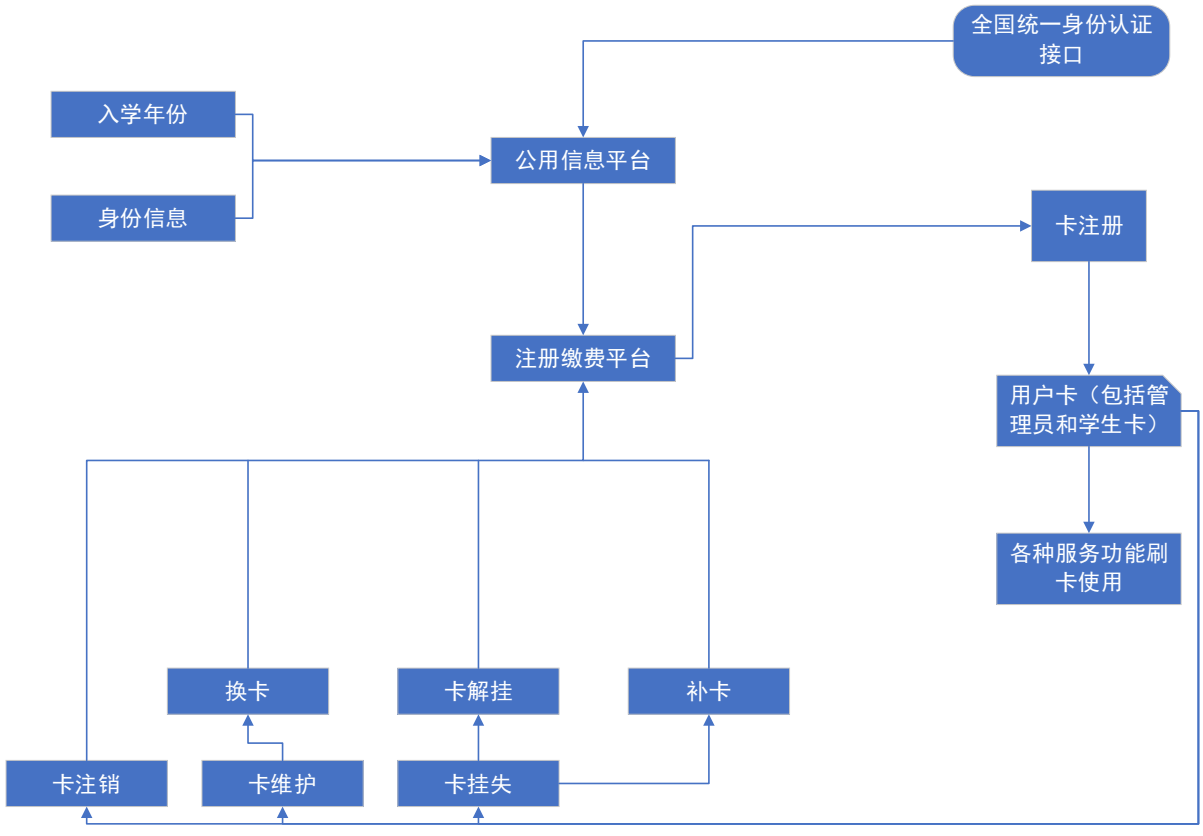

图 3-3 用户卡注册流程

3.2.2 用户卡修改信息

用户卡修改信息的流程图如图 3-4 所示：

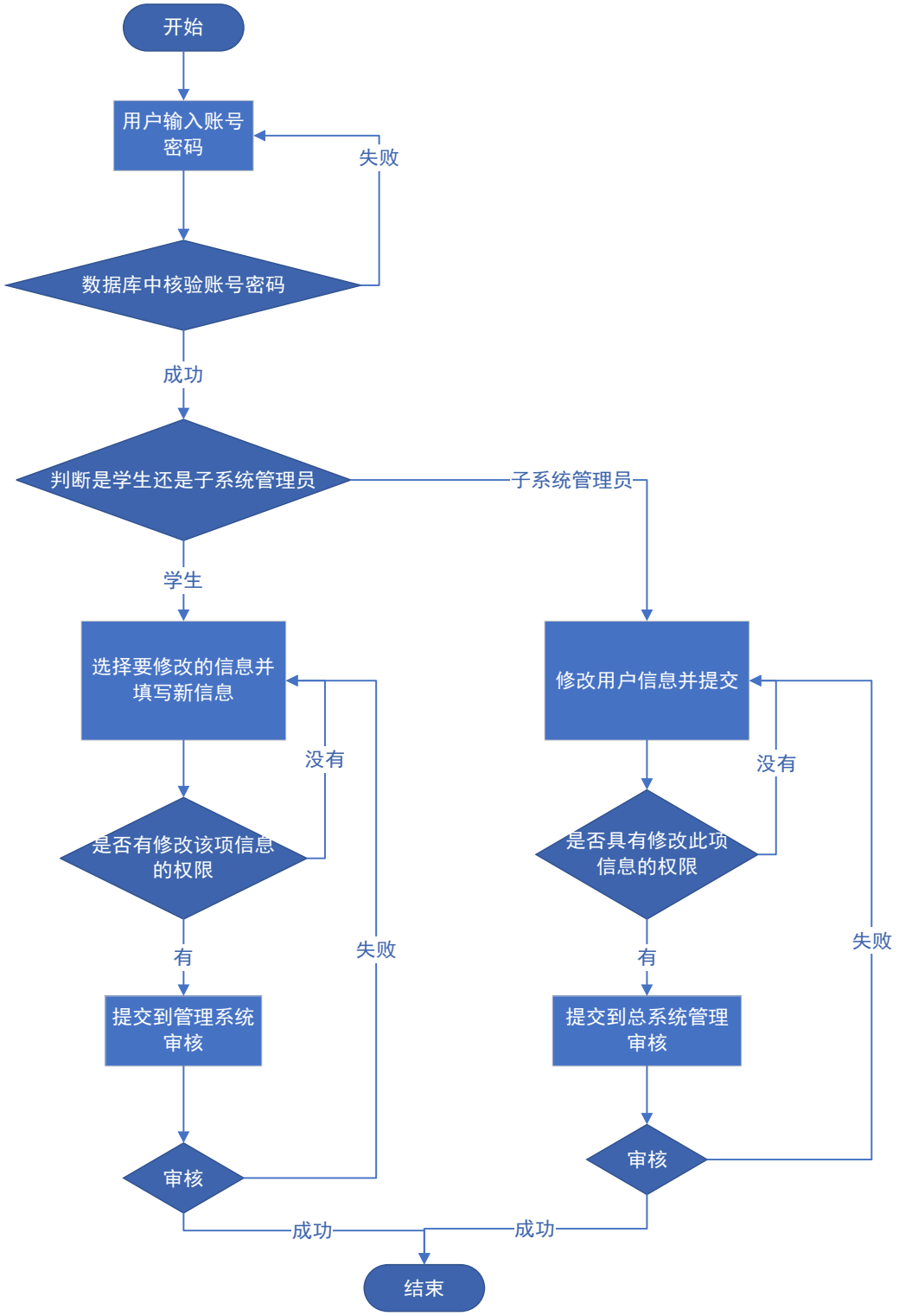

图 3-4 用户卡信息修改流程图

3.2.3 用户卡信息查询

用户卡查询信息流程图如图 3-5 所示。

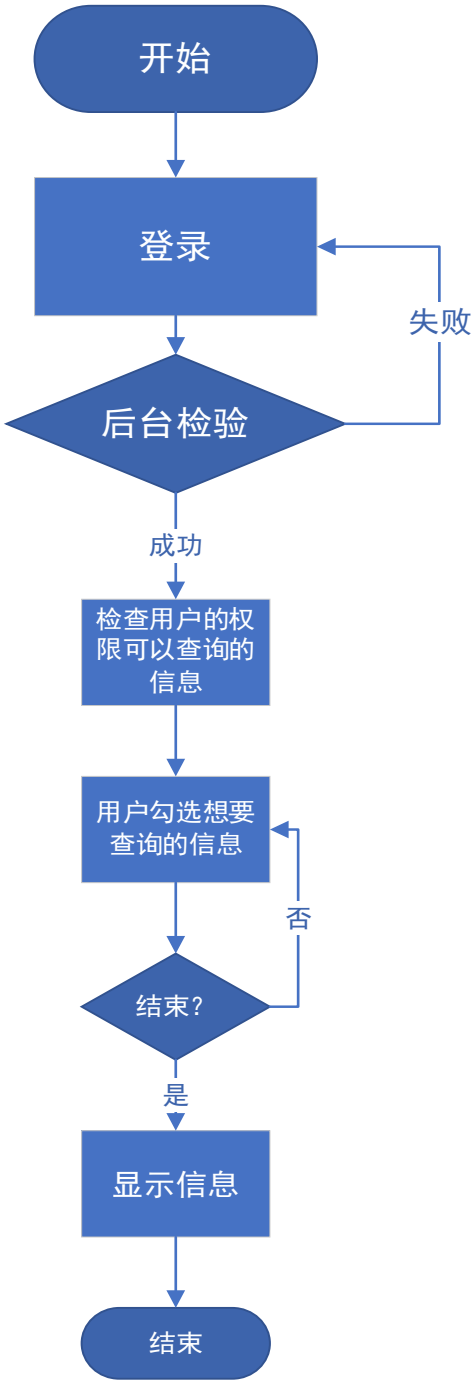

图 3-5 用户卡信息查询流程图

3.2.4 用户卡批量删除

用户卡批量删除的数据流图如图 3-6 所示。

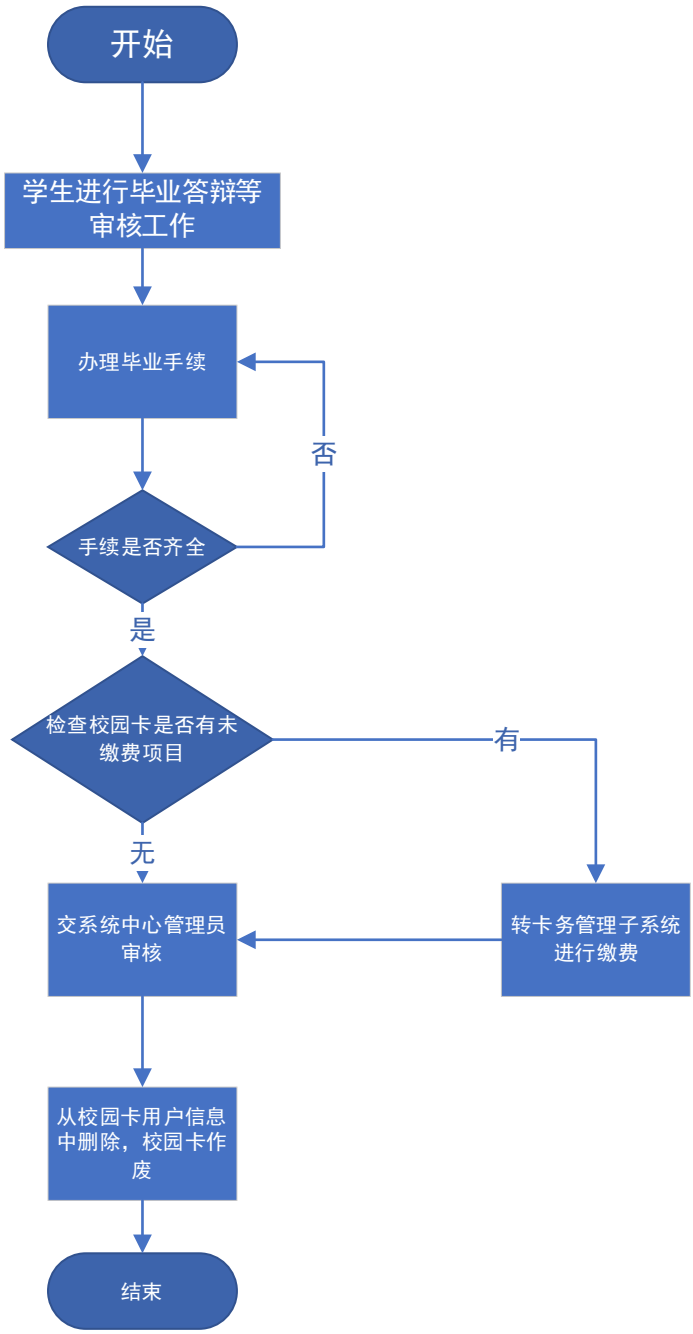

图 3-6 批量删除校园卡信息流程图

3.3 数据设计

表 3.1 学生信息表

字段名	数据类型	允许非空	备注
stu_id	int	NO	学生编号
stu_name	string	NO	学生姓名
in_date	date	NO	学生的入学时间
stu_sex	string	NO	学生的性别
stu_telephone	string	NO	学生的联系方式
stu_address	string	NO	学生的住址
stu_email	string	NO	学生的邮箱
stu_password	string	NO	学生的密码
stu_balance	string	NO	一卡通余额
stu_school	string	NO	所属学院
stu_class	string	NO	所属班级

表 3.2 修改信息请求表

字段名	数据类型	允许非空	备注
query_id	int	NO	请求编号
queryUser_id	int	NO	用户编号
new_telephone	string	YES	新电话
new_email	string	YES	新邮箱
new_address	string	YES	新家庭住址
isReact	bool	NO	是否已经生效

表 3.3 校园卡状态信息表

字段名	数据类型	允许非空	备注
card_id	int	NO	校园卡编号
IsUse	bool	NO	是否使用状态
MissingCount	int	NO	挂失次数
Authority	string	NO	用户权限
Query_list	string	YES	查询记录

3.4 界面设计

图 3-7 用户登录界面

图 3-8 个人信息维护界面

图 3-9 开卡界面

用户编号	用户类型	用户状态	用户姓名	一卡通余额	组织机构	电话
1000	系统管理员	正常	管理员	15.62	10	18621703545
1001	学生	正常		0.00	13	18621703545
1002	学生	正常	李四	0.00	12	16666666666
1003	学生	正常	王五	0.00	12	11211113333
1004	学生	正常	张二三	0.00	12	22222222222
1005	学生	正常	张三三	0.00	12	55555555555
1006	学生	正常	张思三	0.00	12	55555555555
1007	学生	正常	张五三	0.00	12	44444444444
1008	一卡通管理员	正常	严唯嘉	0.00	10	18621703545
2001	一卡通管理员	正常	尼古拉斯	0.00	16	1237767899

图 3-10 查询统计卡用户信息

图 3-11 增加用户界面

4 关键模块算法设计

4.1 登录模块

在启动该系统后，必须在登录界面选择登录身份和提供用户名密码进行验证，正确后方可执行后续操作，系统设置为普通用户、子系统管理员和系统中心管理员。

用户登录部分实现算法。

```
/**
     * 登录
     * @throws IOException
     */
    private void login() throws IOException {
        String cardNo = textField_cardNo.getText();
        String password = CodeUtils.getMD5(passwordField.getText());
        //验证用户是否可用
        SqlSession sqlSession = DBAccess.getSqlSession();
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        User user = userMapper.selectByPrimaryKey(cardNo);
        sqlSession.close();
        if(user==null || !password.equals(user.getPassword())){
```

```
                AlertInfoDialog("信息","信息","用户名密码不匹配,请重新输入!");
                passwordField.selectAll();
                passwordField.requestFocus();
                return;
            }
            if(!"正常".equals(user.getAvailable())){
                AlertInfoDialog("信息","信息","用户状态异常,无法登陆系统.\n 具体原
因:"+user.getAvailable());
                return;
            }
            CurrentUser.userID = user.getId();          //设置全局变量
            CurrentUser.password = user.getPassword();
            CurrentUser.type = user.getType();

            //TODO:判断用户是否勾选了维护信息的 checkbox
            if(checkBox_changeInfo.isSelected()){
                FXHelper.newStage(getClass(),"/resources/fxml/ChangeUserInfo.fxml","个人信息维
护!");
                FXHelper.getStage().close();      //关闭系统第一个窗口
                return;
            }
```

算法 4.1 用户登录界面部分实现算法

4.2 重置用户信息模块

```
/**
     * 重置用户密码
     */
    private void resetUserPassword(){
        String userID = textFieldResetPwUserID.getText();
        if("".equals(userID)){
```

```
            FXHelper.showWarningDialog("请输入用户编号");
            textFieldResetPwUserID.requestFocus();
            return;
        }
        SqlSession sqlSession = DBAccess.getSqlSession();
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        User user = userMapper.selectByPrimaryKey(userID);
        if(user==null){
            FXHelper.showWarningDialog("编号为 0"+userID+"的用户不存在!");
            textFieldResetPwUserID.requestFocus();
            return;
        }
        user.setPassword(CodeUtils.getMD5("123456"));
        FXHelper.showInfoDialog("编号为 0"+userID+"的用户密码已被重置为:\n123456\n 请通知该用户及时登录系统修改密码!");
        userMapper.updateByPrimaryKey(user);
        sqlSession.commit();
        sqlSession.close();
    }
```

算法 4.2 重置密码部分实现算法

4.3 修改用户信息模块

```
/**
     * 确认更新用户信息
     */
    @FXML
    private void confirmUpdateUserInfo(){
        String userID = textFieldUpdateUserID.getText();
        String userName = textFieldUpdateUserName.getText();
        String organizationName = choiceBoxUpdateUserOrganization.getValue();
        String sectionName = choiceBoxUpdateUserSection.getValue();
        if("".equals(userID)){
            FXHelper.showWarningDialog("请输入用户编号!");
            textFieldUpdateUserID.requestFocus();
            return;
        }
        if("".equals(userName)){
            FXHelper.showWarningDialog("请输入用户姓名!");
            textFieldUpdateUserName.requestFocus();
            return;
```

```
            }
            SqlSession sqlSession = DBAccess.getSqlSession();
            User user = sqlSession.getMapper(UserMapper.class).selectByPrimaryKey(userID);
            user.setId(userID);
            user.setName(userName);
            user.setAvailable(choiceBoxUpdateUserAvailable.getValue());
            user.setType(choiceBoxUpdateUserType.getValue());

            OrganizationMapper organizationMapper = sqlSession.getMapper(OrganizationMapper.class);
            SectionMapper sectionMapper = sqlSession.getMapper(SectionMapper.class);
            OrganizationExample organizationExample = new OrganizationExample();
            SectionExample sectionExample = new SectionExample();

            if(!"".equals(organizationName)){
                organizationExample.or().andNameEqualTo(organizationName);
                Integer organizaionID =
organizationMapper.selectByExample(organizationExample).get(0).getOrganization_id();
                user.setOrganization(organizaionID);
            }else {
                user.setOrganization(null);
            }

            if(!"".equals(sectionName)){
                sectionExample.or().andNameEqualTo(sectionName);
                Integer sectionID =
sectionMapper.selectByExample(sectionExample).get(0).getSectionId();
                user.setSection(sectionID);
            }else{
                user.setSection(null);
            }

            sqlSession.getMapper(UserMapper.class).updateByPrimaryKey(user);
            FXHelper.showInfoDialog("修改成功!");
            sqlSession.commit();
            sqlSession.close();
        }
```

算法 4.3 修改信息部分实现算法

4.4 查询信息模块

```
/**
```

```
     * 查询用户并显示在表格上
     */
    private void queryUsersAndShow(){
        String userID = textFieldSearchUserID.getText();
        String userType = choiceBoxSearchUsertype.getValue();
        String userName = textFieldSearchUserName.getText();
        String userOrganization = choiceBoxSearchOrganization.getValue();
        String userSection = choiceBoxSearchUserSection.getValue();

        SqlSession sqlSession = DBAccess.getSqlSession();
        OrganizationMapper organizationMapper = sqlSession.getMapper(OrganizationMapper.class);
        SectionMapper sectionMapper = sqlSession.getMapper(SectionMapper.class);
        OrganizationExample organizationExample = new OrganizationExample();
        SectionExample sectionExample = new SectionExample();

        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        UserExample userExample = new UserExample();
        UserExample.Criteria criteria =  userExample.or();
        criteria.andIdLike((userID!=null)?"%"+userID+"%":"%")
                        .andNameLike("%"+userName+"%");
        if(!"".equals(userType)){
            criteria.andTypeEqualTo(userType);
        }
        if(!"".equals(userOrganization)) {
            organizationExample.or().andNameEqualTo(userOrganization);
            Integer organizationID =
organizationMapper.selectByExample(organizationExample).get(0).getOrganization_id();
            criteria.andOrganizationEqualTo(organizationID);
        }
        if(!"".equals(userSection)) {
            sectionExample.or().andNameEqualTo(userSection);
            Integer sectionID =
sectionMapper.selectByExample(sectionExample).get(0).getSectionId();
            criteria.andSectionEqualTo(sectionID);
        }
        List<User> userList = userMapper.selectByExample(userExample);

        tableViewSearch.setItems(FXCollections.observableList(userList));

        sqlSession.close();
    }
```

算法 4.4 查询信息部分实现算法

5 系统测试方案

5.1 测试计划描述

本测试计划主要为功能测试。功能测试计划覆盖子系统中的所有功能模块，测试整个子系统是否达到预计功能。

5.2 测试方案

1、测试用户登录模块

测试用例：

测试序号	测试用户名	测试密码	预期结果	说明
1	admin	123456	登录成功	符合命名规则
2	Admin	Null	登录失败	密码不符合规则
3	Null	123456	登录失败	用户名违规
4	1+1	=2	登录失败	违规的信息

2、删除用户模块

测试用例：

测试序号	要删除的编号	预期结果	说明
1	20170001	删除成功	符合数据条件
2	管理员：1001	删除失败	无权限操作
3	张三	删除失败	违规的删除主键

3、重置用户密码

测试序号	要重置的编号	预期结果	说明
1	20170001	重置成功	存在的的用户
2	20210001	重置失败	不存在的用户编号
3	张三	重置失败	违规的重置条件

4、查询用户信息模块

测试序号	要查询的编号	预期结果	说明
1	20170001	查询成功	存在的用户信息
2	20210001	查询失败	不存在的用户信息
3	张三	查询失败	违规的查询条件

5、修改信息模块

测试序号	修改信息	预期结果	说明
1	用户类型：学生 学院改为：信息科学与工程学院	修改失败	学生没有权限执行操作
2	用户类型：管理员	修改成功	有权限修改基本属性

6 参考文献

[1] 于丽,田园. 基于 JavaEE 架构的校园一卡通系统设计与实现[M]. 大连: 大连理工大学, 2015.3

[2] 吴昊, 刘东旭. 数字化校园一卡通系统设计[M]. 无线互联科技, 2018.5: 10

[3] 杨玉坤, 邵世栋. 校园一卡通系统设计与实现[M]. 河北: 河北科技大学, 2013.12

[4] 张海藩. 软件工程导论（第五版）[M]. 北京: 清华大学出版社, 1999.5: 20-25

[5] 陈莉. 人月神话[M]. 北京: 清华大学出版社, 2015.4

[6] 彭鑫. 软件工程[M]. 北京: 机械工业出版社, 2016.12

[7] 李忠利, 李淳. 软件需求（第三版）[M]. 北京: 清华大学出版社,2016.2: 20-25

[8] 黄文娟. 基于 Java 和 MySQL 的图书馆信息化管理系统设计[M]. 电子设计工程, 2019.2: 20-24

[9] 倪海顺. 计算机软件开发的 Java 编程语言应用探讨[M]. 信息与电脑（理论版）, 2019.2: 60-61

[10] 赵一凡, 卞良. 基于 MD5 的加盐消息摘要 Java 实现[M]. 软件导刊, 2018.3: 214-216

[11] 魏志军. 基于 Java EE 多层框架的实时监测系统设计与实现[M]. 电子设计工程, 2018.5: 47-50

[12] Sabela Ramos, Guillermo L. Nonblocking collectives for scalable Java communications [J]. Concurrency Computat, 2015.5: 10-25

[13] Xiaoyan Zhu, An analysis of programming language statement frequency in C, C++, and Java source code [M]. Softw. Pract, 2015.11: 45-50

[14] Ronald A, Olsson. a Java package providing JR-like concurrent programming [J]. Softw. Pract, 2016.5: 46-48

[15] Ortin, The Runtime Performance of invokedynamic: An Evaluation with a Java Library [M]. IEEE Software,2014.4: 31-33