《数据库技术》

期末大作业报告

题　　目：数据库技术期末大作业

学生姓名：程宇

学　　号：20211900015

专　　业：计算机科学与技术

任课教师：周小兵

评分（百分制）：

2024 年 1 月

目录

云南大学校园卡管理系统

第一步 需求分析

一、调查用户需求

本系统的最终用户为校园卡管理员，云南大学使用校园卡的老师和学生。校园卡我们日常都在使用，我根据平时使用校园卡的习惯，并对自己学校校园卡管理老师的咨询和对同学和老师的调查，得出用户的下列实际要求：

1、 校园卡的基本情况

平时学生在使用卡的过程中，可以使用登陆、修改密码、查询消费记录、充值、生成二维码、借阅书籍。然后在一开始的时候，后台的校园卡管理人员可以在新生入学使用卡片，进行发卡、补卡、查询、统计和其他的相关信息等，在这个过程中，我们还会使用校园卡借书、宿舍门口门禁刷卡或者实验室刷卡等。

一、 学生的基本信息：

入校时，每位同学都有唯一的学号，然后和名字对应起来，然后还有学生所在的学院，性别（‘男’，‘女’）并被分配到指定的宿舍楼和指定的宿舍，也会有一个宿舍号，还有他的入住时间。另外，为了管理上的方便，同一院系的学生的宿舍一般在一起，相应地会有其所在的院系名称。方便学院的审核和管理。

二、 校园卡的基本信息：

在每一个学生的信息学号被录入的时候，就会有一个对应的校园卡的卡号，然后校园卡还有校园卡的余额，当充值的时候余额就会改变，当然在校园卡使用消费过程也会被记录下来，然后方便下次用户的查询，在挂失的时候还存在校园卡可以用和不可以用的情况，校园卡创建的时间（补办的时间），还有学生如果使用校园卡需要登陆的情况则需要密码(此处参考了云大充值的小程序)。方便在使用过程中用户和管理员对信息的需求。

三、 图书馆借阅的基本信息：

在日常生活中我们经常使用借阅图书，所以每个对应的学号都可以借阅图书的相关信息，包括这个书在数据库中对应的图书编号以及借阅的时间信息，在借阅的时候，经常要一个月内归还，还有目前这本书的借阅状态是借阅中还是已归还。方便用户和图书馆对信息的管理和分析。

四、 学生充值消费的基本信息：

在充值的时候可以通过管理员进行现金或者微信等充值，也可以自己进行充值，充值的时候在学生卡信息中也会改变对应的学生卡信息，还有在食堂或者超市进行消费，对应的金额也会发生变化，如果金额最后小于 0，那么这个操作将不被允许。方便用户对自己使用校园卡情况的分析以及后续的审核。

五、 学生卡门禁的基本信息：

在实验室或者在宿舍需要刷卡进入，这个会和学校的系统进行核

实，如果信息不匹配则不能进入实验室或者宿舍，然后门禁的时候刷卡也需要记录下当前的时间以及地点，方便以后对次操作进行核对以及查询。方便楼栋管理员或者实验室管理员对信息的把握然后分析出勤状况等。

六、 **学生卡信息查询的基本信息：**

在学生的信息比较多的时候，管理起来比较麻烦，所以查询操作也需要进行，可以通过用户的学好进行查询，然后进行分析用户的行为等，可以根据要求进行增删改等，但是不能看到密码和不能删除一些额外的信息，比如用户的姓名或者用户的正在借阅的书籍，防止存在其他不符合用户的操作出现，比如不能删除用户的密码但是可以修改密码。方便用户在平台上对自己的信息进行分析和管理。

七、 **管理员的统计的基本信息：**

当对用户的收支和整个运行状况进行分析，这个月的总消费额以及各个楼栋的刷卡记录，充值记录，金额大小分析，收支分析等等，都需要进行统计，在核实的时候为管理员提供了方便以及合适的通道。方便管理员有序的进行分析和层次结构分明。

2、 用户对系统的要求

一、后台管理人员：

a. 信息要求：

能够查询校园卡的所有相关信息，包括卡号、余额、激活状态、

创建时间等。能查询特定学号或员工编号关联的校园卡的详细信息。管理和查看所有校园卡的充值、消费、补卡等记录。以利于对整个校园卡系统的全面管理。

b.处理要求：

当校园卡信息发生变化时，能对其进行修改，如余额调整、状态更改等，调整前后需记录详细的审核流程和理由，确保透明性和可追溯性。在卡丢失或损坏时，能够及时发行新卡，并更新系统中的相关信息，状态更改需记录原因和执行人信息，以便于审计和追踪。处理充值异常或消费纠纷，针对消费争议，提供详细的交易记录和必要的辅助信息（如终端位置、交易时间），确保每笔交易的正确性。

c．安全性与完整性要求：

安全性要求：

1.系统应设置访问用户的标识以鉴别是否是合法用户，并要求合法用户设置其密码，保证用户身份不被盗用，加密存储的数据应在传输过程中保持加密状态，防止数据在传输过程中被拦截；

2.系统应对不同的数据设置不同的访问级别，限制访问用户可查询和处理数据的类别和内容实施严格的用户认证机制，确保每个用户都有独特的登录凭据。基于角色的访问控制（RBAC），确保员工只能访问与其职责相关的数据和功能。；

3.系统应对不同用户设置不同的权限，区分不同的用户，如区分普通用户（学生），管理员；

4. 系统应记录所有关键操作的日志，包括数据更改、查询和系统

设置的调整。定期审计操作日志以监控可能的不当行为或安全漏洞。必要的时候可以进行回滚的操作。

完整性要求：

1. 数据验证：系统应对输入的数据进行有效性检查，确保数据的准确性和适用性。字段级别的数据完整性检查，如格式校验、范围检查等。确保用户定义的完整性。
2. 数据备份与恢复：定期备份关键数据，确保在系统故障或数据丢失的情况下可以快速恢复。测试并验证数据恢复流程，确保数据完整性和连续性。
3. 参照完整性：确保数据库中的外键关系得到正确维护，防止出现“悬挂”引用。在执行删除或更新操作时，系统应检查并维护数据间的关联性。
4. 数据一致性：在不同的系统组件或模块间保持数据的一致性。实现事务管理机制，确保在发生错误或中断时能够回滚到一致的状态。
5. 数据完整性：在用户自己设置的密码中保证其正确性和完整性，采用加密手段进行加密，防止进入数据库中看到密码等涉及用户的隐私问题，保证了密码的安全，保护用户的基本信息。

二、用户（学生或教职工）：

a. 信息要求：

用户可以通过系统轻松查询自己的校园卡余额、查看详细的消费记录以及充值历史等信息。这不仅提供了一个透明的财务管理工具，还帮助用户更好地控制和监督自己的消费行为。此外，系统还允许用户查询校园卡的激活状态，确保其卡片是可用的，从而避免在关键时刻遭遇不便。为了增强安全性和方便性，系统还提供了生成二维码的功能，用于个人身份验证，这在诸如进入图书馆、使用门禁系统等场景中尤其有用。通过这种方式，学生和教职工可以利用校园卡进行各种活动，包括消费扣款、图书借阅和访问受限区域等。

b. 处理要求：

在云南大学一卡通管理系统中，设计用户享有多项自助服务功能。首先，他们可以方便地在系统内进行自助充值，无需通过第三方进行这一操作，同时还可以自行修改自己的密码，确保账户的安全。若遇到校园卡丢失或损坏的情况，用户可以轻松地在线申请补办新卡或挂失，这极大地提高了处理此类问题的效率和便捷性。

此外，系统还允许用户自助更新个人信息，比如联系方式等，从而确保与用户相关的所有信息都是最新和最准确的。

c. 安全性与完整性要求：

系统应保证个人信息和交易数据的安全，防止未授权访问。保证个人数据的完整性和准确性，避免信息错乱。保持数据在不同平台和记录中的一致性。

二、系统功能的设计和划分

根据如上得到的用户需求，我们将本系统按照所完成的功能分成以下两个部分：

第一部分：学生管理部分

第二部分：管理员管理部分

各部分完成的功能如下：

1、学生管理部分

1）处理学生登录：

学生可以通过校园卡号和密码登录系统。

2）学生可以查询信息：

学生可以查询自己的校园卡余额、消费记录、充值历史等信息。

学生可以查看自己的图书借阅记录和门禁通行记录。

3）学生可以自助充值：

学生可以在系统中进行校园卡的自助充值。

4）学生可以生成二维码进行身份验证：

系统支持生成二维码，用于身份验证和门禁等场合。

5）学生可以修改密码：

学生可以自行修改校园卡的密码。

2、管理员管理部分

1）处理发卡和补卡：

管理员负责新校园卡的发放和遗失或损坏卡的补发。

2）管理员可以进行查询操作：

管理员可以查询校园卡的发放、补卡、消费、充值等记录。

3）管理员可以进行统计分析：

管理员可以对校园卡的使用情况进行统计分析，如消费总额、活跃用户数量等。

4）管理员处理消费扣款：

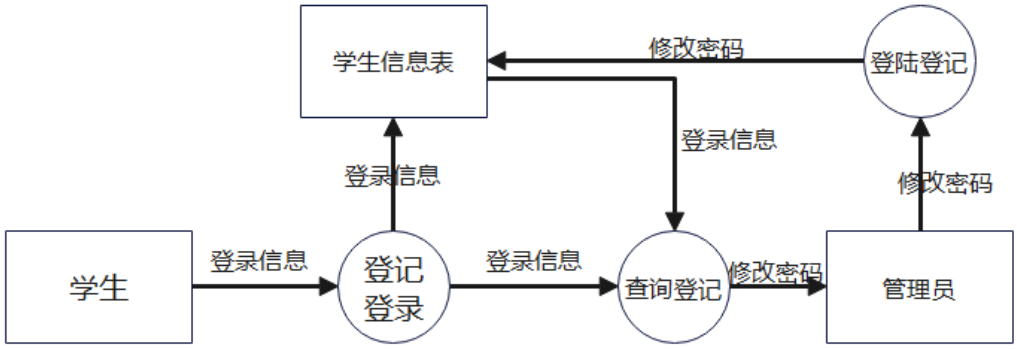
管理消费扣款记录，确保每笔交易的准确性。

5）管理员管理图书借阅和门禁系统：

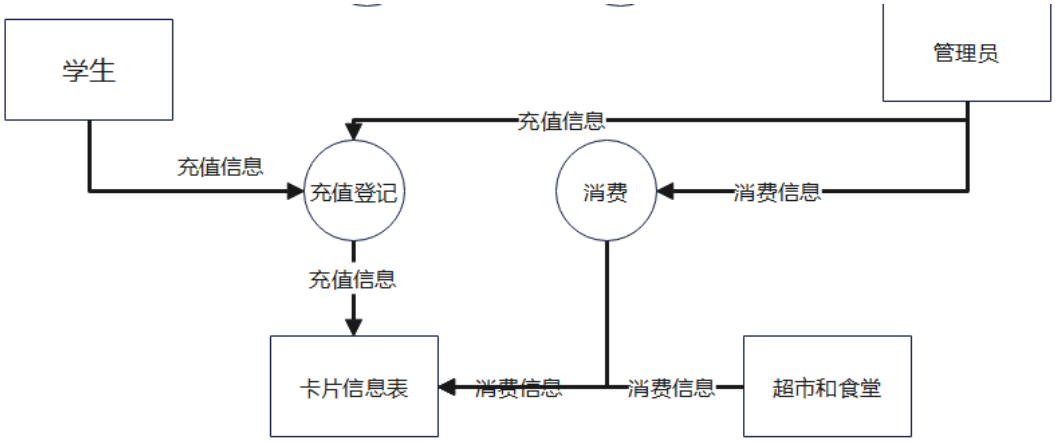
管理图书借阅记录和控制门禁系统，包括记录进出时间和地点。

三、数据流图

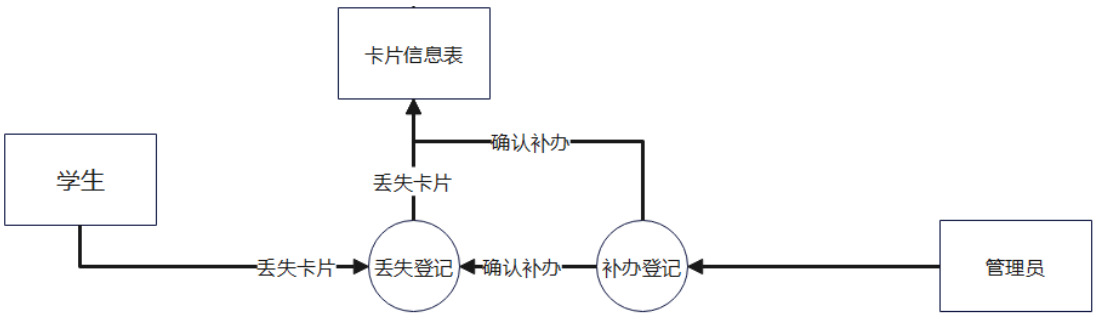
学生登陆数据流图



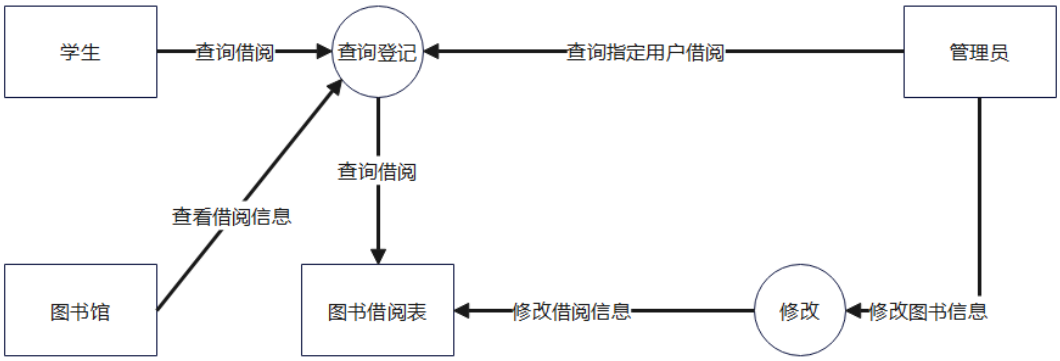
学生消费数据流图：



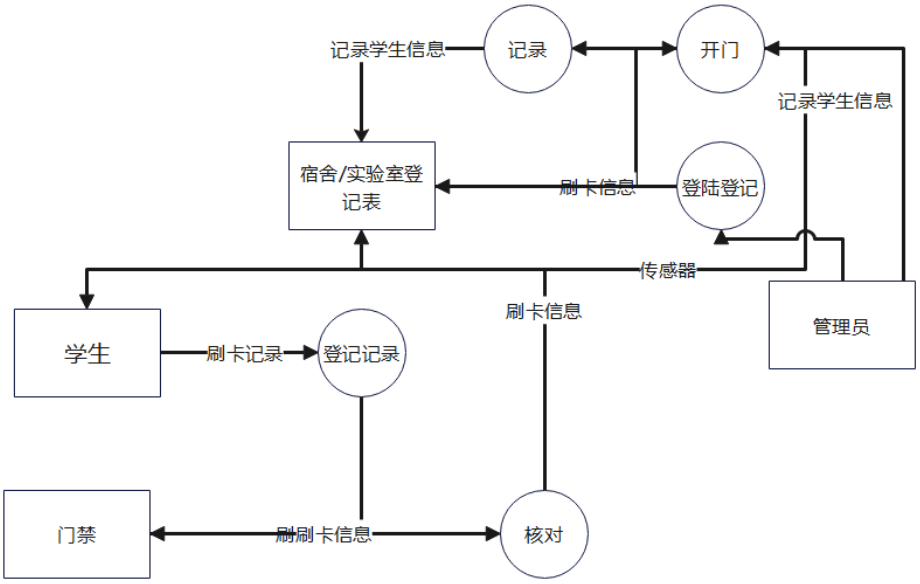
卡片丢失补办登记：



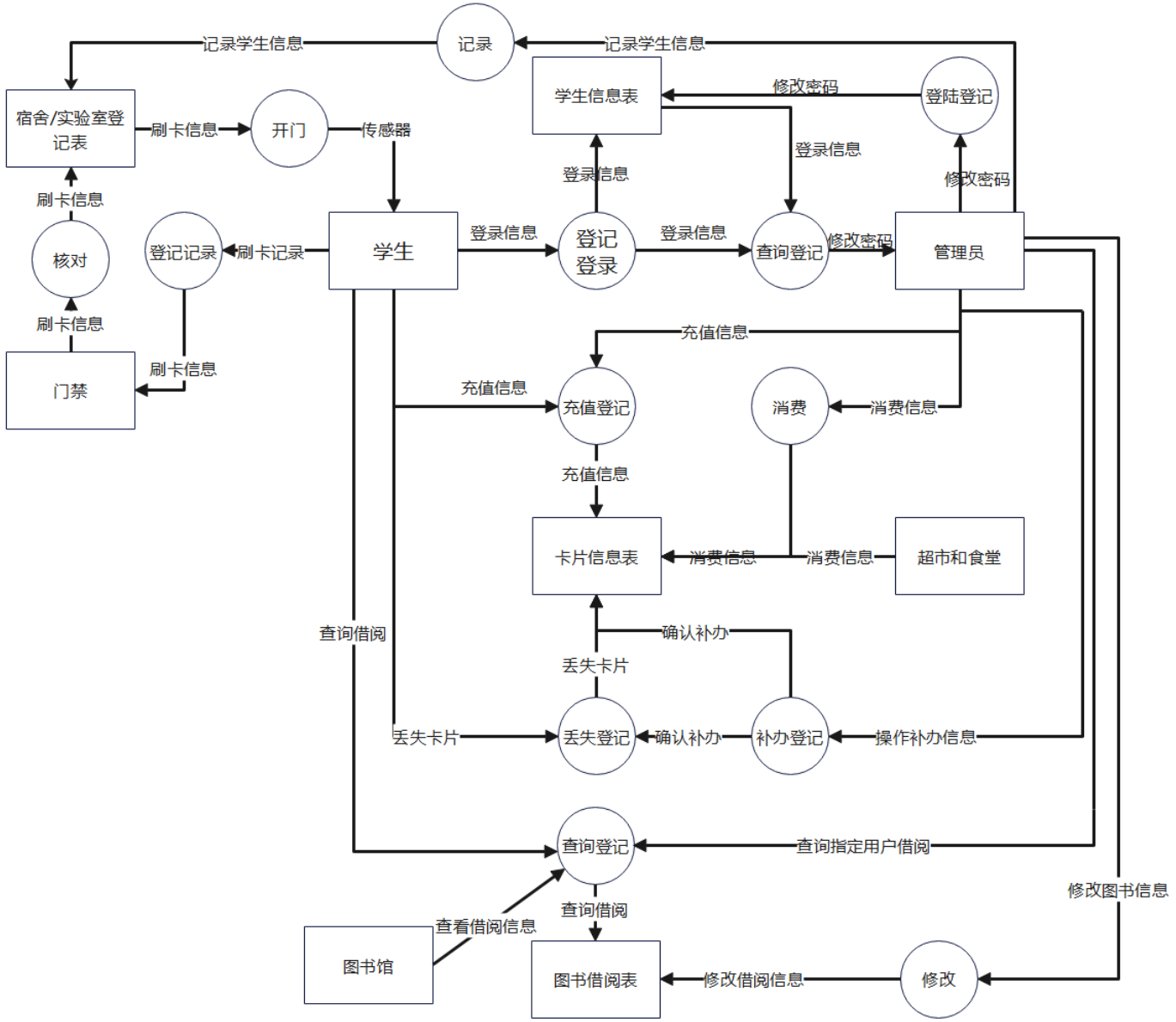
图书借阅数据流图



门禁数据流图



总数据流图


记录
记录学生信息
记录学生信息
学生信息表
修改密码
登陆登记
宿舍/实验室登记表
刷卡信息
开门
传感器
登录信息
登录信息
修改密码
刷卡信息
核对
登记记录
刷卡记录
学生
登录信息
登记登录
登录信息
查询登记
修改密码
管理员
刷卡信息
充值信息
门禁
刷卡信息
充值信息
充值登记
消费
消费信息
充值信息
卡片信息表
消费信息
消费信息
超市和食堂
查询借阅
确认补办
丢失卡片
丢失卡片
丢失登记
确认补办
补办登记
操作补办信息
查询登记
查询指定用户借阅
修改图书信息
查询借阅
查看借阅信息
图书馆
图书借阅表
修改借阅信息
修改

四、数据字典

1.数据项

校园卡数据字典：

属性名	存储代码	类型	长度	备注
卡号	card_id	varchar	20	学生校园卡卡号
余额	balance	double	8	校园卡用户余额
状态	state	varchar	10	校园卡是否可用
创建时间	create_time	datetime	6	管理员创建时间
密码	password	varchar	20	用户设置的密码

学生信息数据字典：

属性名	存储代码	类型	长度	备注
学号	student_id	char	20	进入校园分配学号
学院	college	varchar	40	学生所属学院
性别	sex	char	10	选择男/女
年纪	grade	varchar	10	学生所属年纪
宿舍	dormitory	varchar	20	学生分配的宿舍
卡号	card_id_id	varchar	20	学生的卡号
姓名	name	varchar	20	学生对应的姓名

学生借阅图书信息数据字典：

属性名	存储代码	类型	长度	备注
借阅号	id	int	11	每条记录的 id 编号
价格	value	double	8	书的价格
借阅时间	time	datetime	6	学生借阅时间
模式	mode	varchar	10	是否被借出
卡号	card_id_id	varchar	20	学生的卡号

图书借阅信息数据字典：

属性名	存储代码	类型	长度	备注
借阅号	id	int	11	每次借阅对应的 id
书号	book_id	varchar	10	借阅书的 id
借阅时间	borrow_time	date	8	当前入档时间
理应归还时间	ending_time	data	8	一个月后的时间
借阅状态	state	char	12	借出/归还

卡号	card_id_id	varchar	20	学生的卡号

宿舍/实验室访问记录数据字典：

属性名	存储代码	类型	长度	备注
房间号	id	int	11	地方实验室的 id
进入时间	time	datetime	6	刷卡的时间
地点	location	varchar	40	刷卡进入的地点
卡号	card_id_id	varchar	20	学生的卡号

消费/充值记录数据字典：

属性名	存储代码	类型	长度	备注
消费/充值号	id	int	11	每次消费的 id
金额	value	double	8	消费的金额大小
时间	time	datetime	8	消费时间
地点	location	varchar	40	消费地点
卡号	card_id_id	varchar	20	学生卡卡号

管理员基本信息数据字典：

属性名	存储代码	类型	长度	备注
管理员号	id	int	11	管理员号
密码	password	varchar	128	管理员密码
最近登录	last_login	datetime	8	管理员最近登录
超级管理员	is_superuser	tinyint	1	是否为超级管理员
姓名	username	varchar	150	管理员的姓名
邮箱	email	varchar	254	管理员的邮箱
工作人员	is_staff	tinyint	1	管理员是否在职
有效	is_active	tinyint	1	管理员是否有效
加入时间	date_join	datetime	8	管理员加入时间

管理员分级管理数据字典：

属性名	存储代码	类型	长度	备注
管理员分级号	id	int	11	管理员分级的 id
管理员号	user_id	int	11	管理员对应的 id
管理员登记群号	group_id	int	11	管理员管理的群 id

图书管图书数据字典：

属性名	存储代码	类型	长度	备注
图书号	id	int	11	图书的 id
图书标题	title	varchar	128	图书的标题
借出状态	is_share	tinyint	1	图书是否被借出
用户借阅号	user_id	int	11	用户借阅的号码

2.数据结构

数据结构名	组成
校园卡信息	卡号，余额，状态，创建时间，密码
学生信息	学号，学院，性别，年纪，宿舍，卡号，姓名
学生借阅图书信息	借阅号，借阅时间，卡号
图书借阅信息	借阅号，书号，借阅时间，理应归还时间，借阅状态，卡号
宿舍/实验室访问记录	房间号，进入时间，地点，卡号
消费/充值记录	消费/充值号，金额，时间，地点，卡号
管理员基本信息	管理员号，密码，最近登录，超级管理员，姓名，邮箱，工作人员，有效，加入时间
管理员分级管理	管理员分级号，管理员号，管理员登记群号
图书馆图书信息	图书号，图书标题，借出状态，用户借阅号

3.数据流

登记校园卡信息	学生	校园卡信息表	校园卡信息
查询校园卡信息	校园卡信息表	学生	校园卡信息
更新校园卡信息	管理员	校园卡信息表	校园卡更新信息
登记学生信息	学生	学生信息表	学生信息
查询学生信息	学生信息表	学生/管理员	学生信息
登记图书借阅信息	学生	图书借阅信息表	图书借阅信息
查询图书借阅信息	图书借阅信息表	学生/管理员	图书借阅信息
登记宿舍访问记录	学生	宿舍访问记录表	宿舍访问记录
查询宿舍访问记录	宿舍访问记录表	学生/管理员	宿舍访问记录
登记消费充值记录	学生	消费充值记录表	消费充值记录
查询消费充值记录	消费充值记录表	学生/管理员	消费充值记录
管理员信息更新	管理员	管理员基本信息表	管理员信息
查询管理员信息	管理员基本信息表	管理员	管理员信息
管理员权限管理	管理员	管理员分级管理表	管理员权限信息
查询图书信息	图书馆图书信息表	学生/管理员	图书信息

4.数据存储

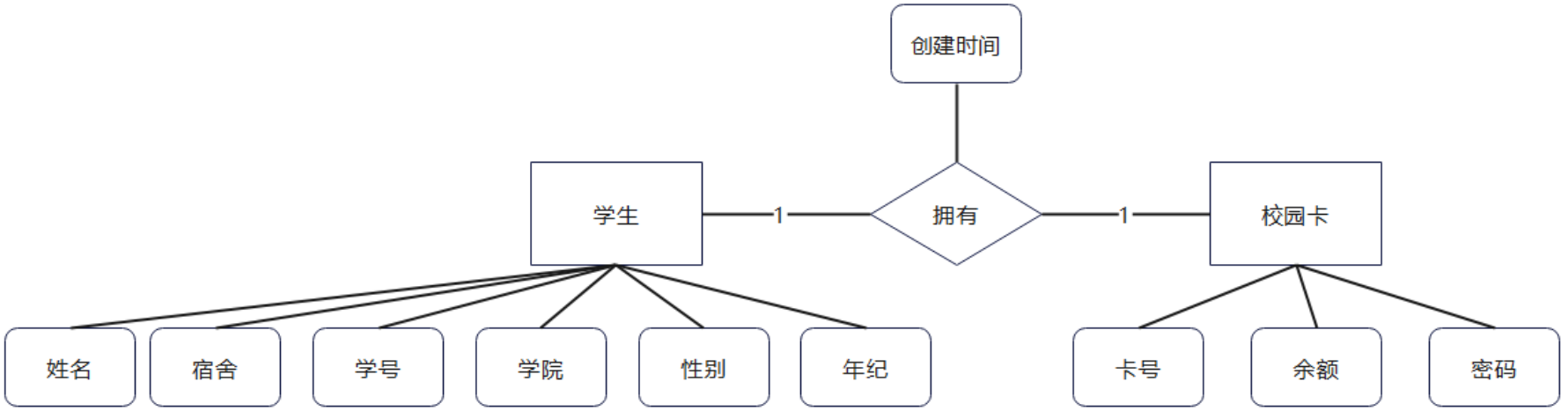
数据存储名	输入的数据流	输出的数据流	组成
校园卡信息表	校园卡信息	校园卡信息	卡号、余额、状态等
学生信息表	学生信息	学生信息	学号、姓名、性别等
学生借阅图书信息表	图书借阅信息	图书借阅信息	借阅号、借阅时间等
图书借阅信息表	图书借阅信息	图书借阅信息	借阅号、书号等
宿舍/实验室访问记录表	宿舍访问记录	宿舍访问记录	房间号、进入时间等
消费充值记录表	消费充值记录	消费充值记录	金额、时间等
管理员基本信息表	管理员信息	管理员信息	管理员号、姓名等
管理员分级管理表	管理员权限信息	管理员权限信息	分级号、管理员号等
图书馆图书信息表	图书信息	图书信息	图书号、标题等

5.处理过程

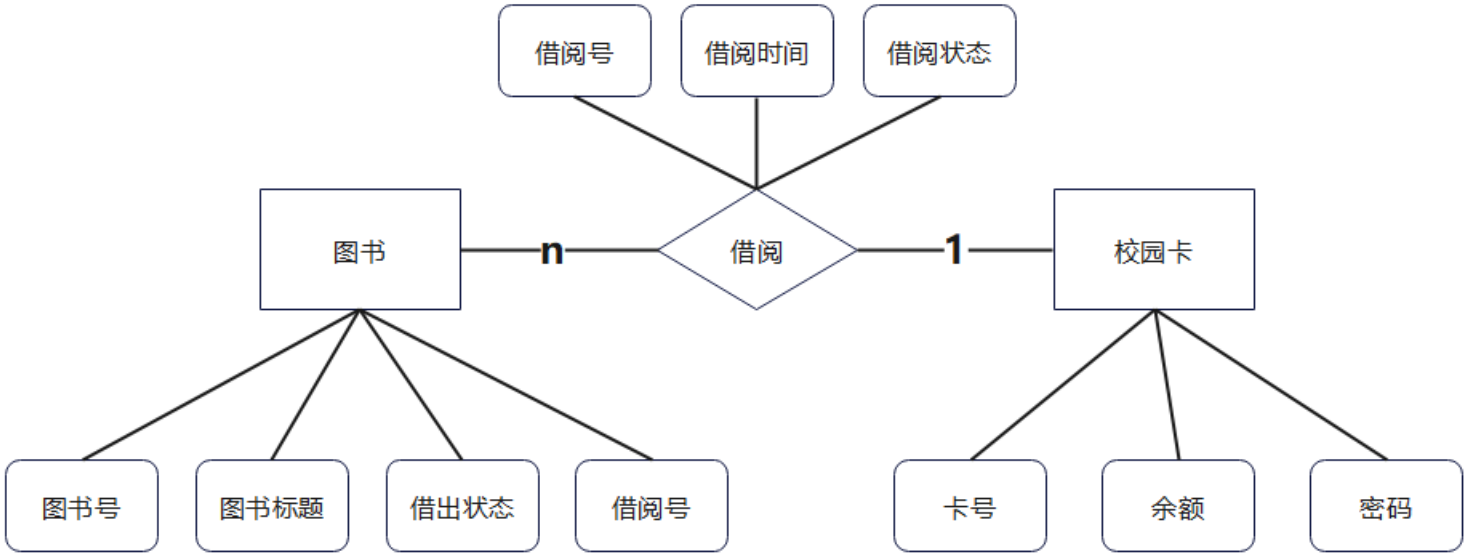
登记校园卡信息	校园卡信息	校园卡信息
查询校园卡信息	校园卡信息	校园卡信息
更新校园卡信息	校园卡更新信息	校园卡更新信息
登记学生信息	学生信息	学生信息
查询学生信息	学生信息	学生信息
登记图书借阅信息	图书借阅信息	图书借阅信息
查询图书借阅信息	图书借阅信息	图书借阅信息
登记宿舍访问记录	宿舍访问记录	宿舍访问记录
查询宿舍访问记录	宿舍访问记录	宿舍访问记录
登记消费充值记录	消费充值记录	消费充值记录
查询消费充值记录	消费充值记录	消费充值记录
管理员信息更新	管理员信息	管理员信息
查询管理员信息	管理员信息	管理员信息
管理员权限管理	管理员权限信息	管理员权限信息
查询图书信息	图书信息	图书信息

第二步、概念结构设计

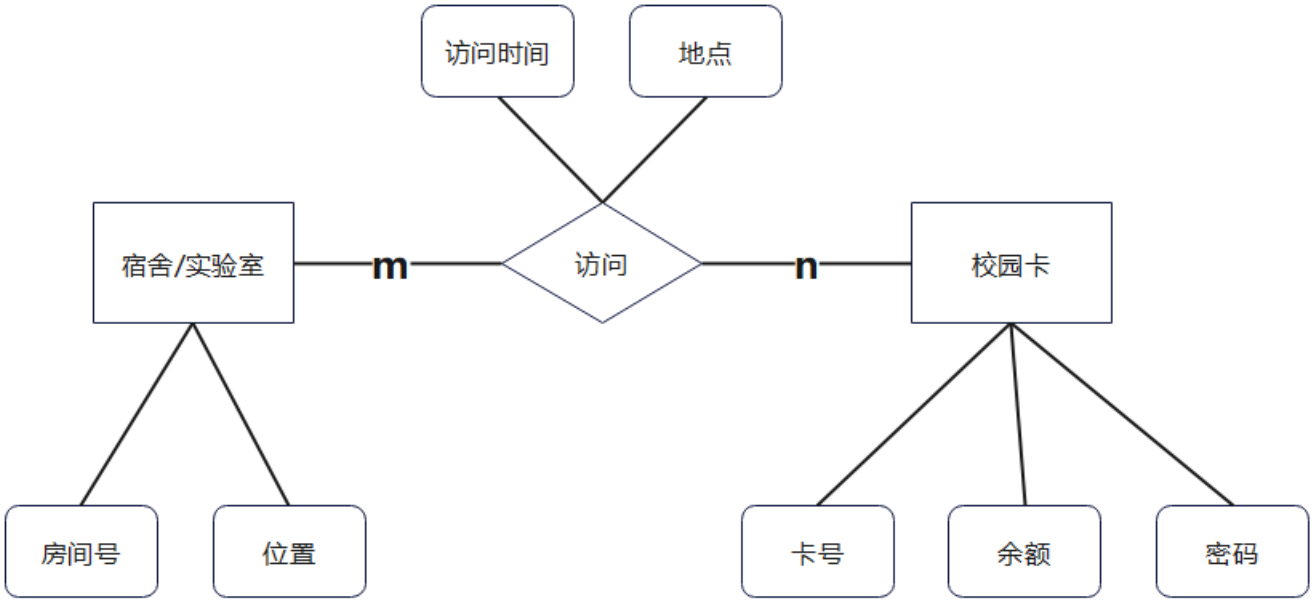
一、学生校园卡信息管理子系统



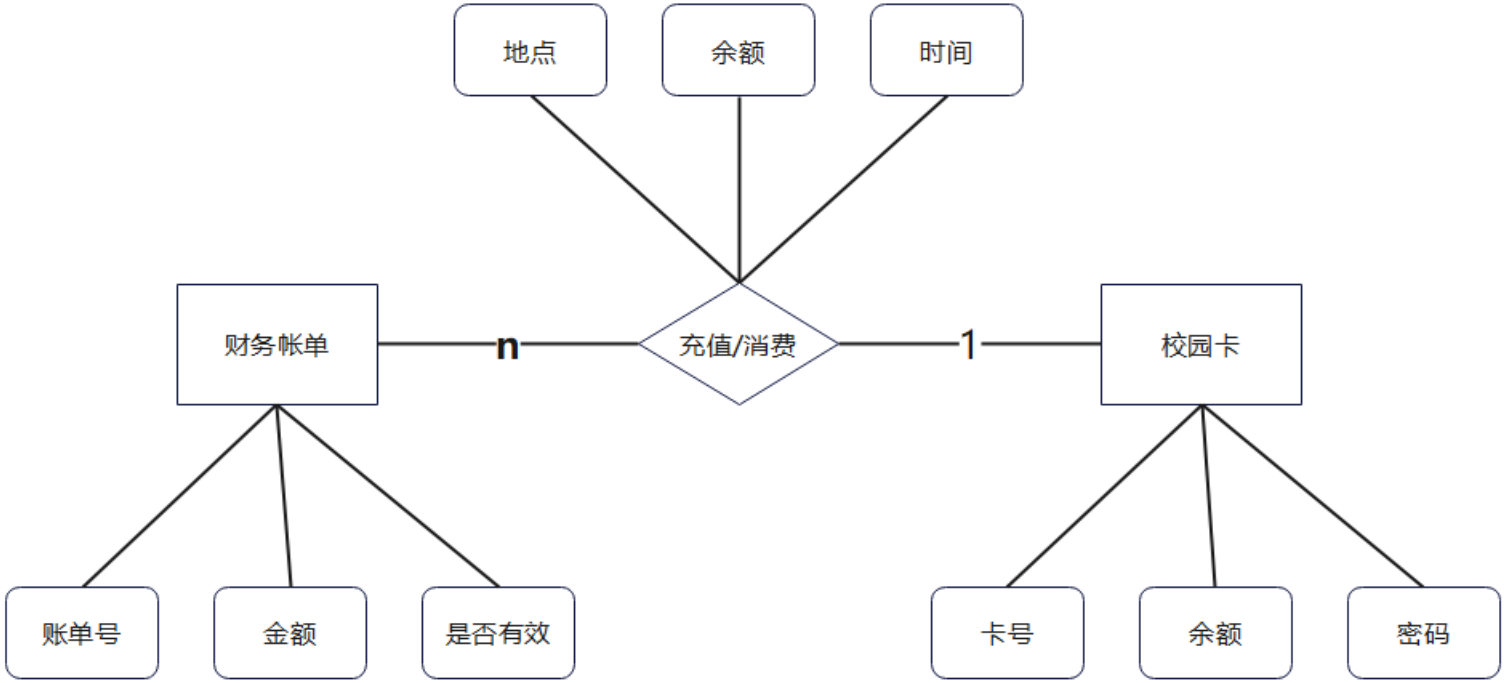
二、图书借阅管理子系统



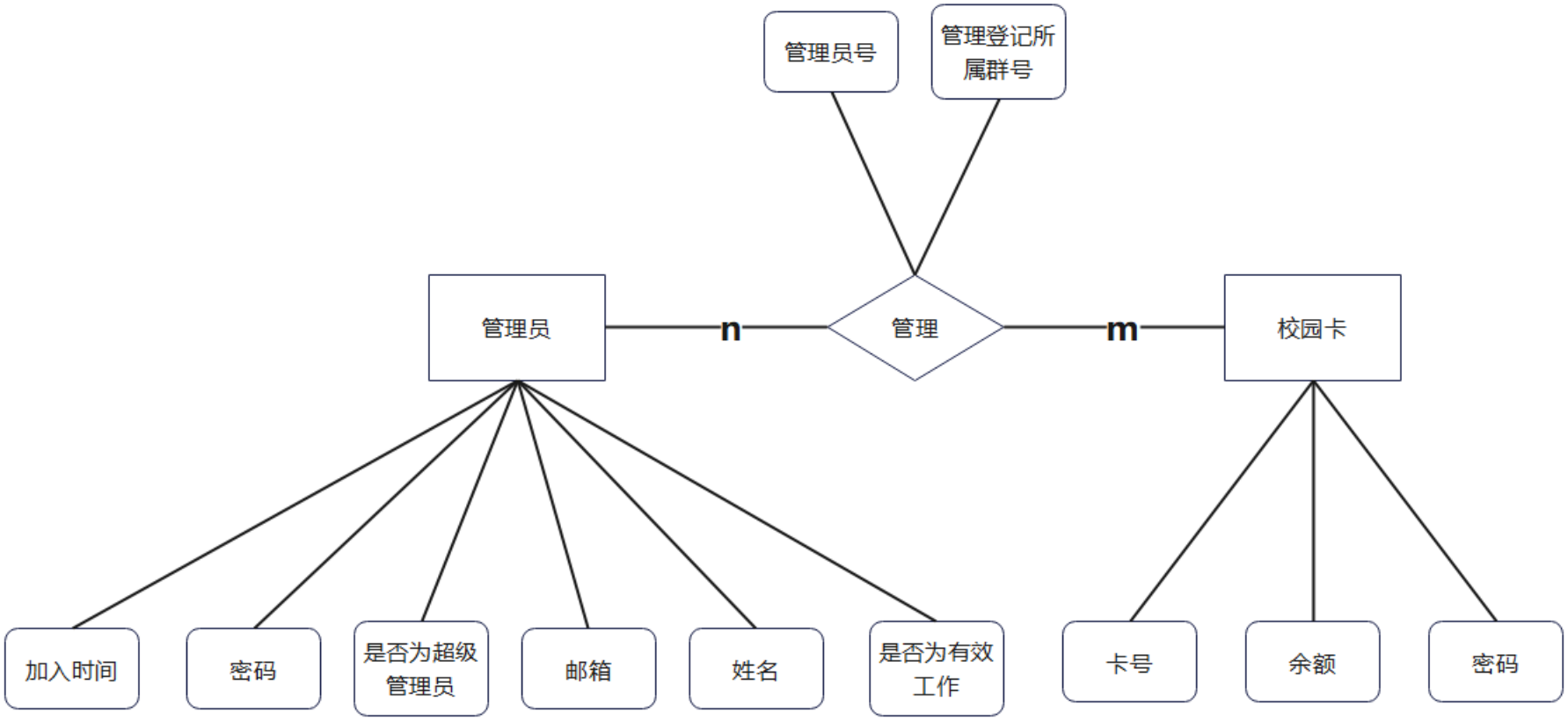
三、宿舍/实验室访问管理子系统



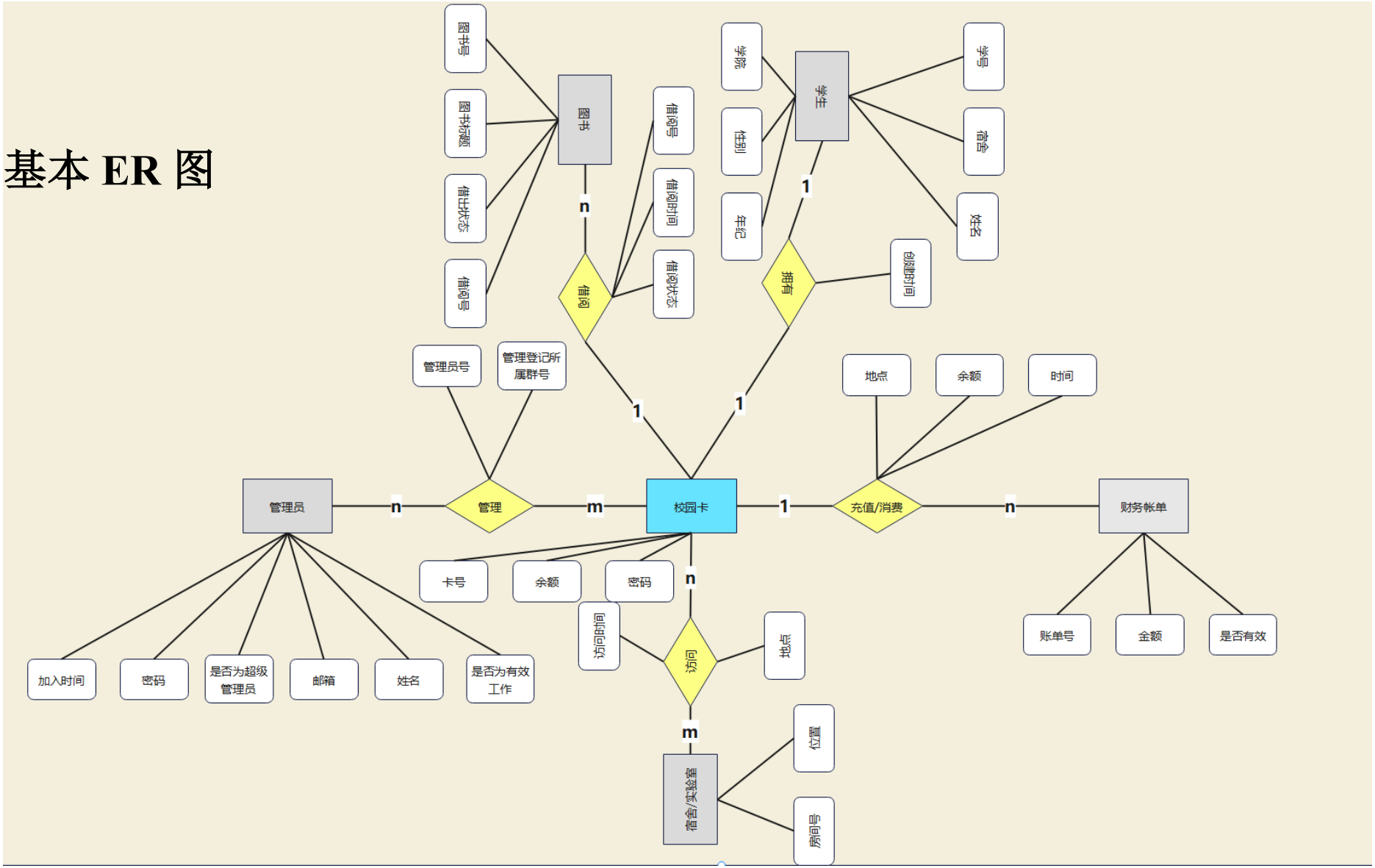
四、财务管理子系统



五、管理员信息管理子系统



基本 ER 图


图书号
图书标题
借出状态
借阅号
图书
借阅号
借阅时间
借阅状态
借阅
n
学院
性别
年纪
学生
学号
宿舍
姓名
拥有
1
创建时间
管理员号
管理登记所属群号
1
1
地点
余额
时间
管理员
n
管理
m
校园卡
1
充值/消费
n
财务帐单
卡号
余额
密码
n
加入时间
密码
是否为超级管理员
邮箱
姓名
是否为有效工作
时间
访问
地点
m
位置
房间号

第三步、逻辑设计

将 ER 图转换为关系模式

关系模式

校园卡信息(卡号, 余额, 状态, 创建时间, 密码)

学生信息(学号, 学院, 性别, 年级, 宿舍, 卡号, 姓名)

图书借阅信息(借阅号, 书号, 借阅时间, 理应归还时间, 借阅状态, 卡号)

宿舍/实验室访问记录(房间记录号, 进入时间, 地点, 卡号)

消费/充值记录(消费/充值号, 金额, 时间, 地点, 卡号)

管理员基本信息(管理员号, 密码, 最近登录, 超级管理员, 姓名, 邮箱, 工作人员, 有效, 加入时间)

管理员分级管理(管理员分级号, 管理员号, 管理员登记群号)

图书馆图书(图书号, 图书标题, 借出状态, 用户借阅号)

（下划线标注的属性为主码）

以上关系模式均为 BCNF。

为了方便程序查询，建立了如下**用户视图：**

学生信息视图（学号，姓名，卡号，密码，学院）

用户余额视图（学号，卡号，余额，姓名，消费时间，消费金额）

学生图书借阅视图（学号，姓名，借阅号，书号，借阅时间，理应归还时间）

宿舍访问视图（学号，姓名，宿舍号，房间记录号，进入时间）

消费记录视图（学号，姓名，卡号，金额，消费时间，地点）

管理员工作视图（管理员号，姓名，最近登录，超级管理员状态，工作人员状态，有效状态）

图书借阅历史视图（学号，姓名，书号，借阅时间，归还时间）

图书馆图书可借阅视图（书号，图书标题，借出状态）

图书借阅历史视图（学号，姓名，书号，借阅时间，归还时间）

这些视图将便于用户从系统中检索有用的信息，通过视图，可以限制用户对基础数据表的直接访问，从而保护敏感数据。因为敏感信息如密码不应该包含在内。视图可以将复杂的查询语句封装起来，用户只需通过查询视图即可获取所需信息，通过视图，可以为不同的用户定制他们所需要的数据展现形式。当底层数据库结构变化时，只需修改视图而不需要更改应用程序代码，视图为用户提供了与物理存储无关的数据逻辑视图，增加了数据库的逻辑独立性。

第四步、物理设计：

1、 为表定义合适的数据类型，以优化存储空间和查询性能（如使用 VARCHAR 替代 CHAR 来节省空间）。

2、 考虑创建复合索引以优化特定的查询操作，为常用的查询字段和排序字段创建索引，以加速查询和报告的速度。例如，对学号、卡号、借阅号等常作为查询条件的字段建立索引，在学生消费时候可以查看相对应的消费记录。

3、 启用数据库的日志记录功能，监控性能指标，以便于分析和优化数据库性能。规划定期备份的策略，包括全备份和增量备份，以及恢复流程，确保数据的安全性。

第五步、数据库实施

1. 管理员界面对用户的操作运行

在 setting.py 中建立数据库的链接，如下图所示：

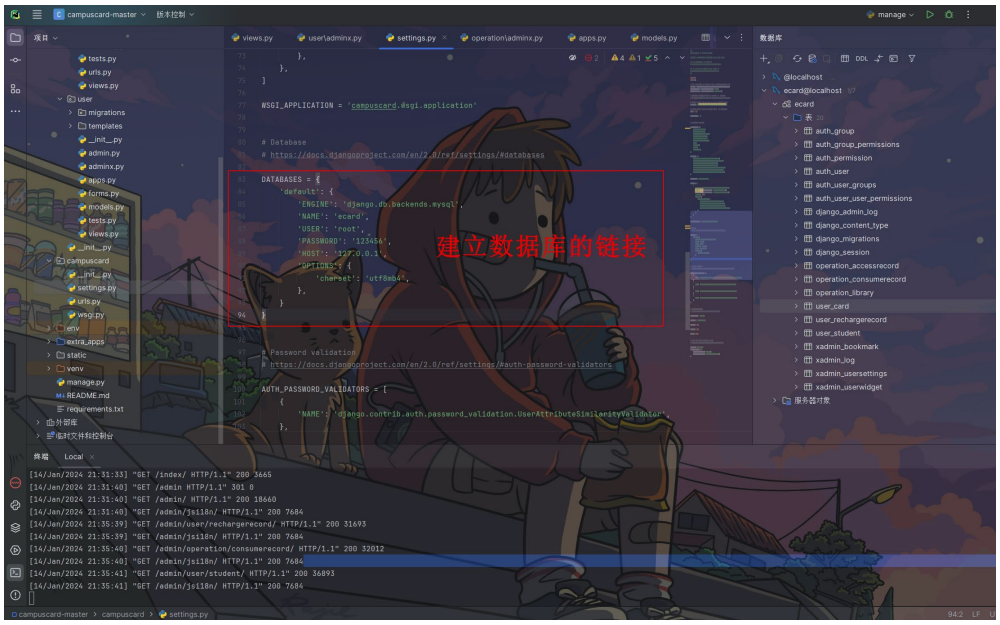


图 1：数据库建立

在登录管理员用户之前我在 MySQL 中加入了这样的一段话，主要是为了方便先登录进去管理员界面进行操作，演示。代码为：

```
INSERT INTO ecard.auth_user(id,username,password,email,is_staff)
VALUES (1,'chengyu','123456','2389122474@qq.com',1);
```

使用 navicat 进行查看：

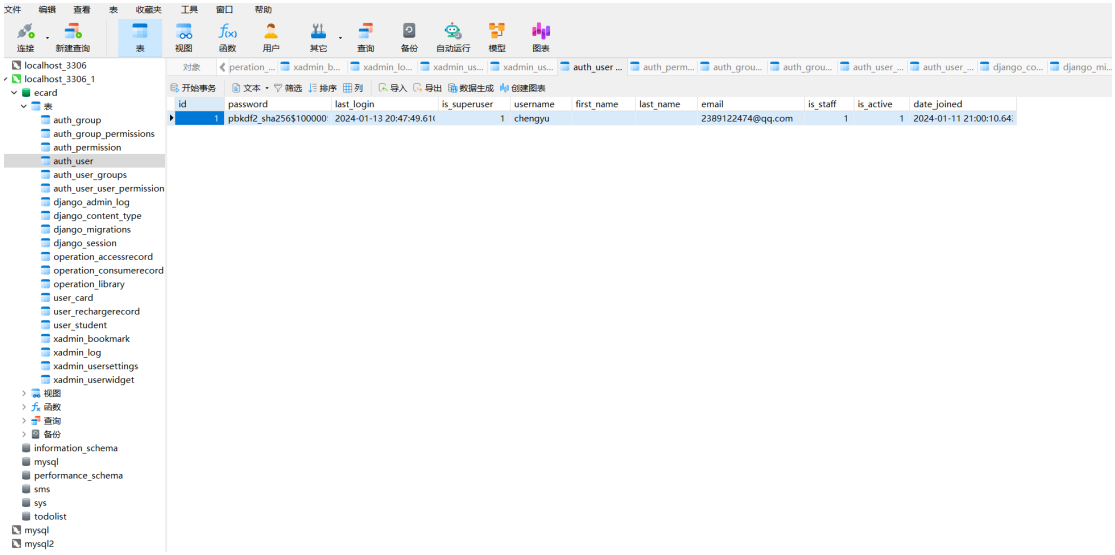


图 2：navicate 进行查看

在用户登录界面的时候，可以看到自己的主页面信息。

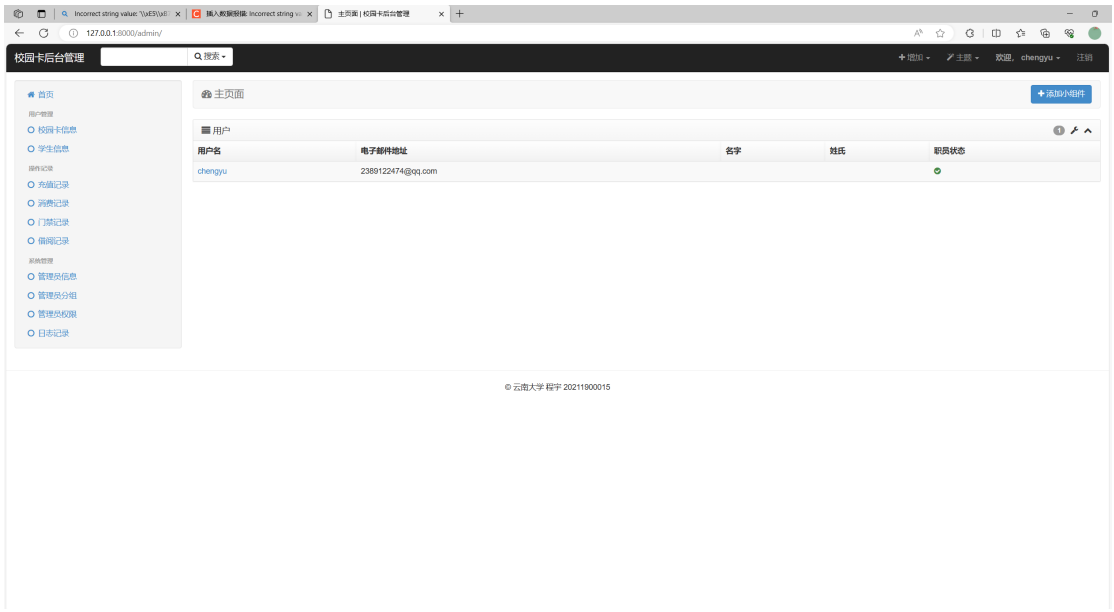


图 3：管理员登陆界面

这是我登陆的时候管理员的界面信息，然后就可以对用户进行增

加操作，但是在增加学生之前要增加卡号的操作，有了卡号才会有学生的信息，因为在数据库定义的过程中，对学生信息的定义代码为：

```
CREATE TABLE user_student (
  student_id varchar(20) NOT NULL,
  college varchar(40) CHARACTER SET utf8mb4 DEFAULT NULL,
  sex varchar(10) CHARACTER SET utf8mb4 DEFAULT NULL,
  grade varchar(10) NOT NULL,
  dormitory varchar(20) CHARACTER SET utf8mb4 DEFAULT NULL,
  card_id_id varchar(20) NOT NULL,
  name varchar(20) CHARACTER SET utf8mb4 DEFAULT NULL,
  PRIMARY KEY (student_id),
  FOREIGN KEY (card_id_id) REFERENCES user_card (card_id)
)
```

定义了卡号为学生的外码，当然其他的比如图书借阅的信息也是基于外码上面的，所以没有外码的情况下就不能进行下面的操作，不能进行一些插入等操作。

如下图所示，对我自己信息进行插入，然后又插入同学的基本进行，为了验证其正确性，我进入 nativate 进行查看之后得到插入是正确的，然后也可以在学生信息中看到数据已经被正常的插入了进去：学生信息插入的代码为：

```
migrations.CreateModel(
                name='Student',
                fields=[
                    ('student_id', models.CharField(max_length=20, primary_key=True, serialize=False, verbose_name='学号')),
                    ('college', models.CharField(max_length=40, verbose_name='学院')),
                    ('sex', models.CharField(choices=[('female', '女'), ('male', '男')], max_length=10, verbose_name='性别')),
                    ('grade', models.CharField(max_length=10, verbose_name='u 年级')),
                    ('dormitory', models.CharField(max_length=20, verbose_name='寝室')),
                    ('card_id', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='user.Card',
```

```
verbose_name='卡号')),
                ],
                options={
                    'verbose_name': '学生信息',
                    'verbose_name_plural': '学生信息',
                    'ordering': ['student_id'],
                },
            ),
```

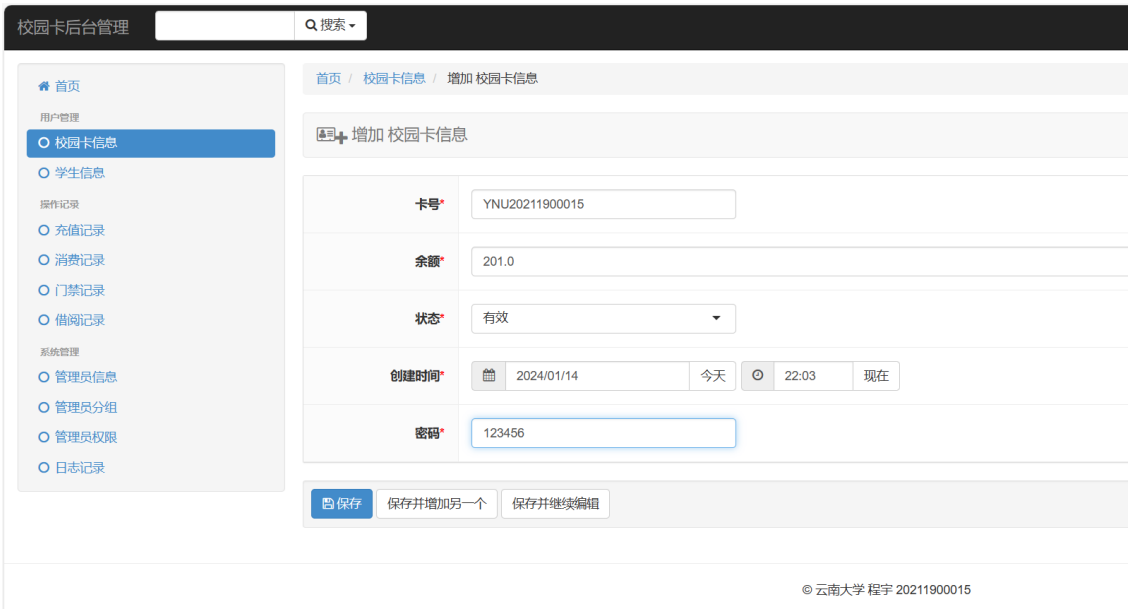


图 4：学生卡的基本信息

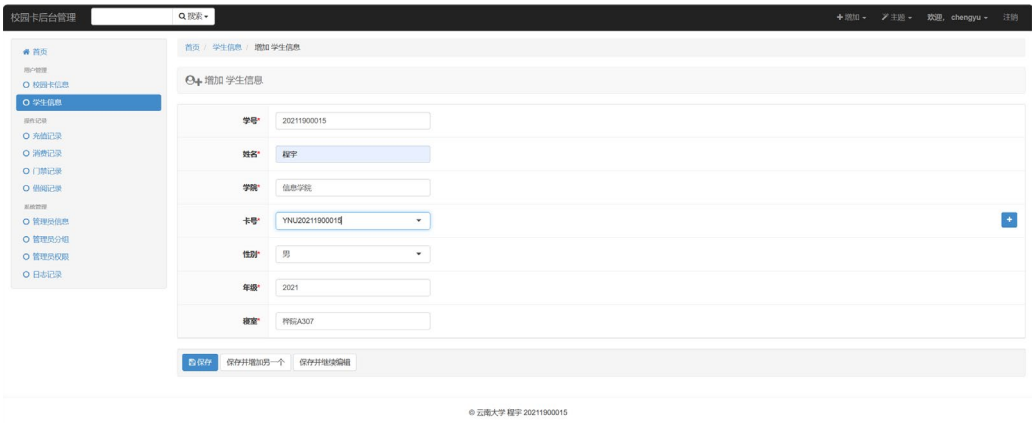


图 5：插入学生的基本信息

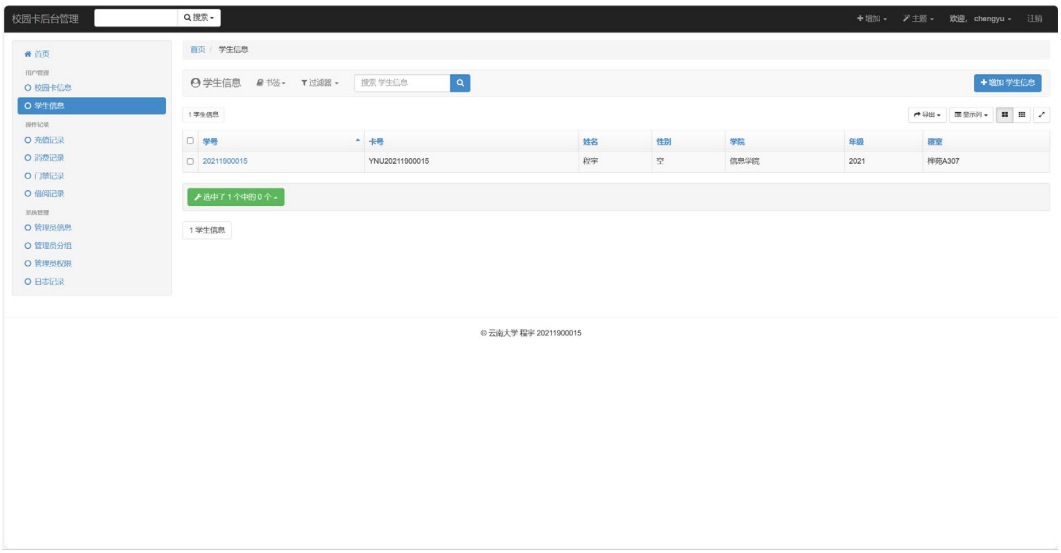


图 6：学生的基本信息查看

然后还要对学生的账户充值进行测试，测试其正确与否，在这个过程中，因为设置了编号会自动地增加，所以用户就会自动的给账单进行编号，方便查询，然后可以从这两个图中看到已经被正常的插入了。

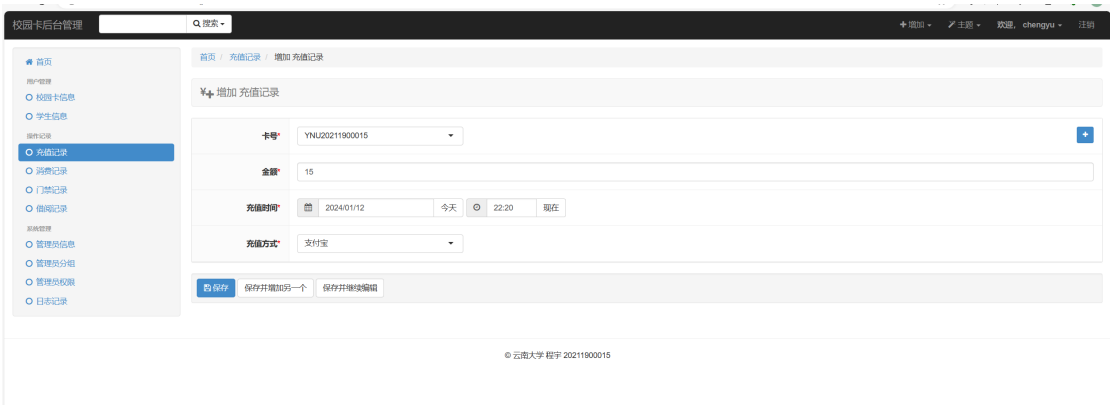


图 7：插入过程

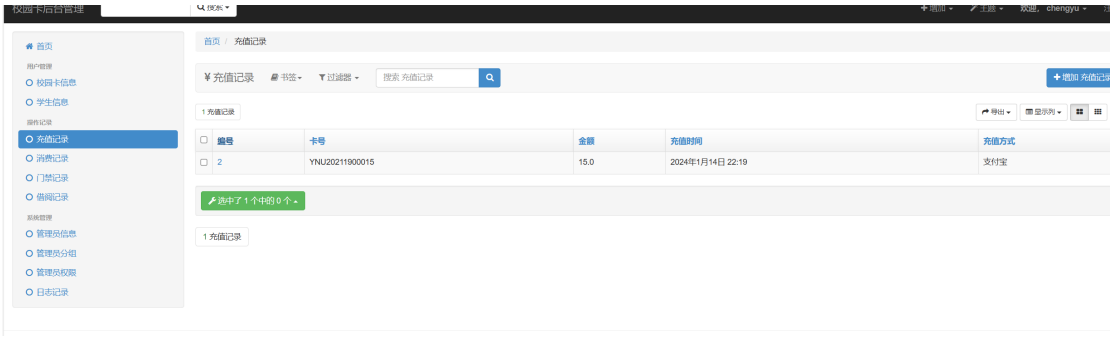


图 8：插入充值结果

其与数据库进行交流的信息代码为：

```
migrations.CreateModel(
                name='RechargeRecord',
                fields=[
                    ('id', models.AutoField(primary_key=True, serialize=False, verbose_name='
编号')),
                    ('value', models.FloatField(verbose_name='金额')),
                    ('time',                models.DateTimeField(default=datetime.datetime.now,
verbose_name='充值时间')),
                    ('mode', models.CharField(choices=[('alipay', '支付宝'), ('wechatpay', '微信
'), ('cash', '现金')], max_length=10, verbose_name='充值方式')),
                    ('card_id',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,              to='user.Card',
verbose_name='卡号')),
                ],
                options={
                    'verbose_name': '充值记录',
                    'verbose_name_plural': '充值记录',
                },
            ),
```

当然也可以对消费记录进行增加，仿照刷卡机的设置，在刷卡的时候自动记录下卡号然后进行扣款的操作，然后在这个过程中接入对应学好的消费记录，这里也同样没有用户的登录编号，这个会自己增加生成，然后可以看到添加后，记录就会自己生成，代表结果运行成功，可以查看相关信息，同理在门禁信息插入的时候也是如此：

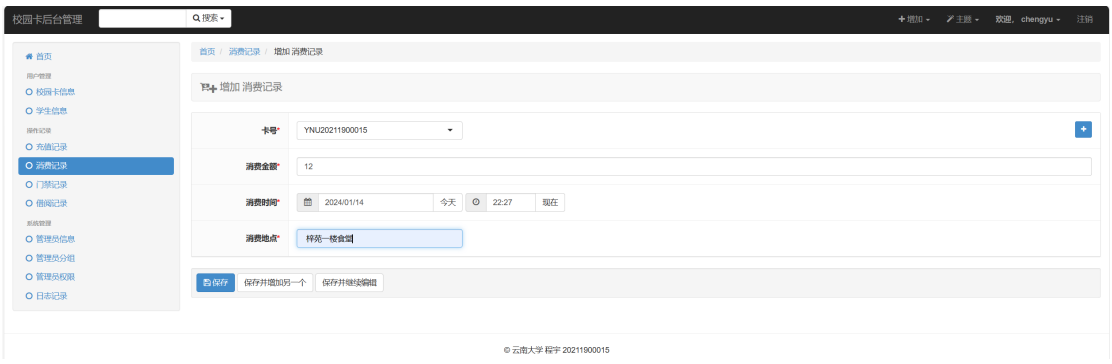


图 9：插入消费记录的过程

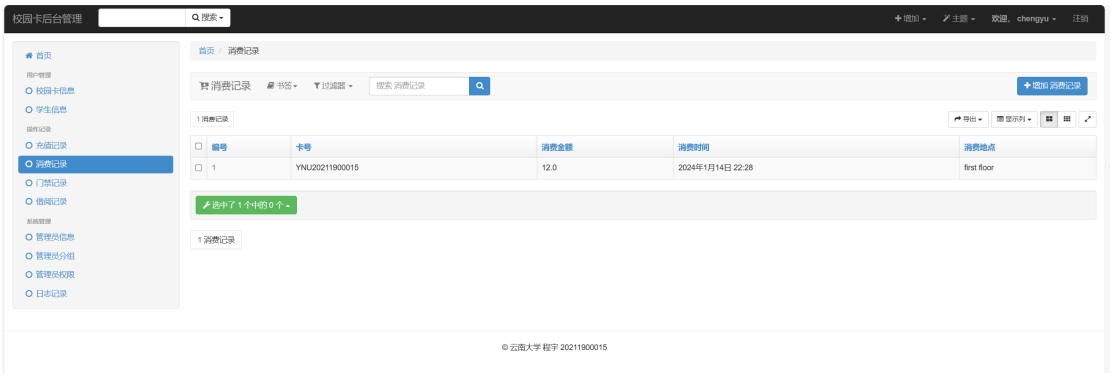


图 10：插入消费记录的结果

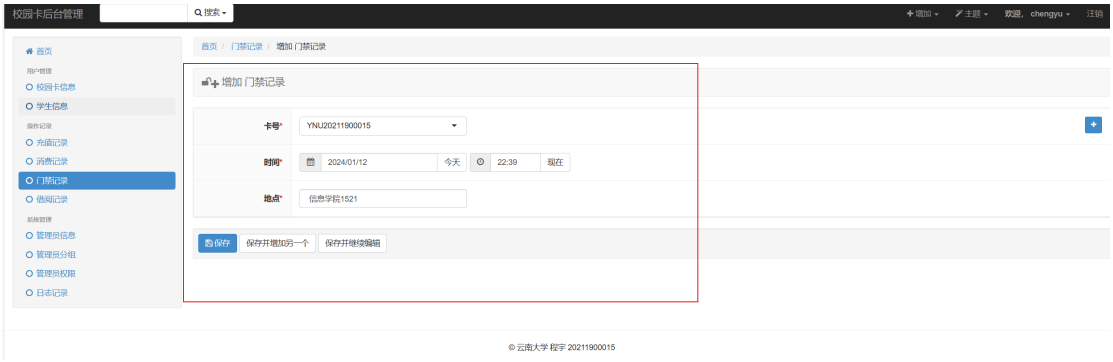


图 11：门禁信息插入

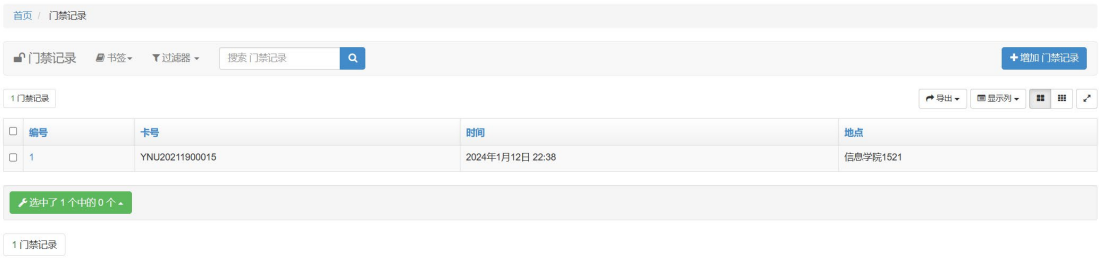


图 12：门禁插入信息结果

代码为：

```
class AccessRecord(models.Model):
    id = models.AutoField(primary_key=True, verbose_name=u"编号")
    card_id = models.ForeignKey(Card, on_delete=models.CASCADE, verbose_name=u"卡号")
    time = models.DateTimeField(verbose_name=u"时间", default=datetime.now)
    location = models.CharField(max_length=40, verbose_name=u"地点", default=" ")

    class Meta:
        verbose_name = "门禁记录"
        verbose_name_plural = verbose_name

    def __str__(self):
        return str(self.card_id)
```

然后对图书的信息进行插入的时候，只需要选定对应的卡号即可，与数据库关联的代码为：

```
class Library(models.Model):
    GENDER_CHOICES = (
        ('returned', '已归还'),
        ('unreturned', '借阅中')
    )
    id = models.AutoField(primary_key=True, verbose_name=u"编号")
    card_id = models.ForeignKey(Card, on_delete=models.CASCADE, verbose_name=u"卡号")
    book_id = models.CharField(max_length=10, verbose_name=u"书本编号")
    borrow_time = models.DateField(verbose_name=u"借书日期", default=datetime.today)
    ending_time = models.DateField(verbose_name=u"应还时间", default=get_endtime())
    state = models.CharField(max_length=12, verbose_name=u"状态", choices=GENDER_CHOICES)

    class Meta:
        verbose_name = u"图书借阅信息"
        verbose_name_plural = verbose_name

    def __str__(self):
        return str(self.card_id)
```

其插入过程如图所示：

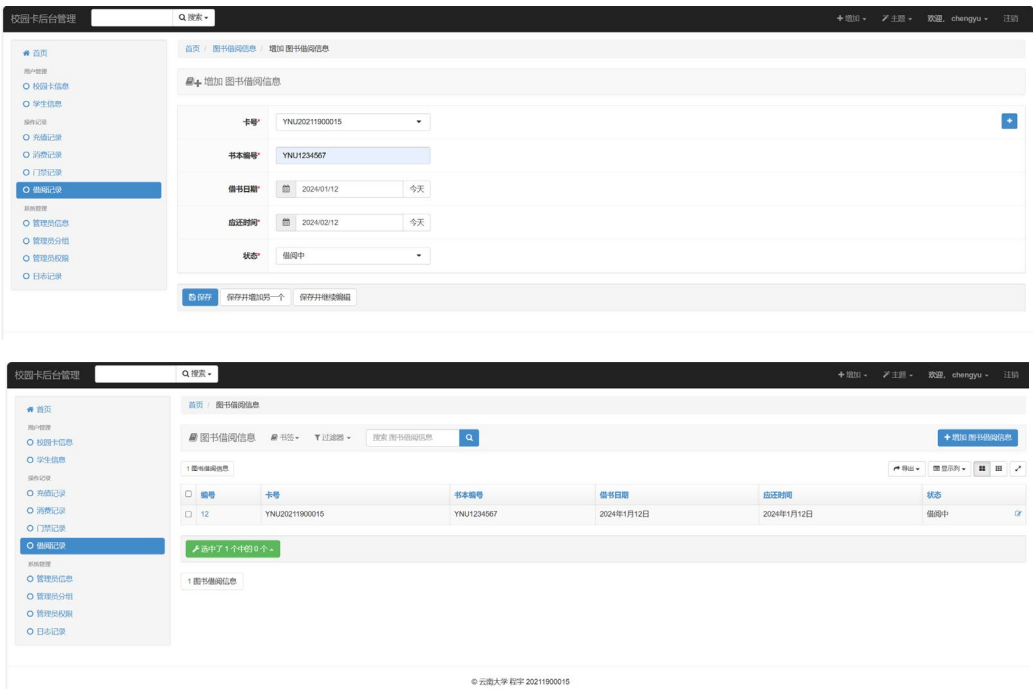


图 13：插入学生的图书借阅

2. 用户对自己界面的查看

当进行完毕上述的操作，现在进入学生端的运行操作，然后对用户的信息进行登录，我刚才创建的 YNU20211900015 和密码 123456，登陆进页面中，其登陆界面如图所示，网址为：127.0.0.1:8000/login：

图 14：用户登录界面

代码为：

```
class LoginView(View):
    def get(self, request):
        return render(request, template_name: 'login.html')

    def post(self, request):
        login_form = LoginForm(request.POST)
        if login_form.is_valid():
            # 获取用户名和密码
            account = request.POST.get('account', '')
            pass_word = request.POST.get('password', '')
            result = Card.objects.filter(card_id=account, password=pass_word)
            # 验证成功
            if len(result) > 0:
                request.session['card_id'] = account
                return HttpResponseRedirect("/index/")
            else:
                return render(request, template_name: 'login.html', context: {'msg': '卡号或密码错误
        else:
            return render(request, template_name: 'login.html', context: {'msg': '卡号和密码不能为空
```

然后登陆后的界面如下图所示，包含了用户的刚才操作的信息，也包括了包括我的头像（此处用静态的照片放入），自己的身份码生成，充值信息，以及借阅信息等，但是我设置这里是看不到自己的刷卡记录的，因为我看云大一卡通里面也没有，然后还有个人信息等。

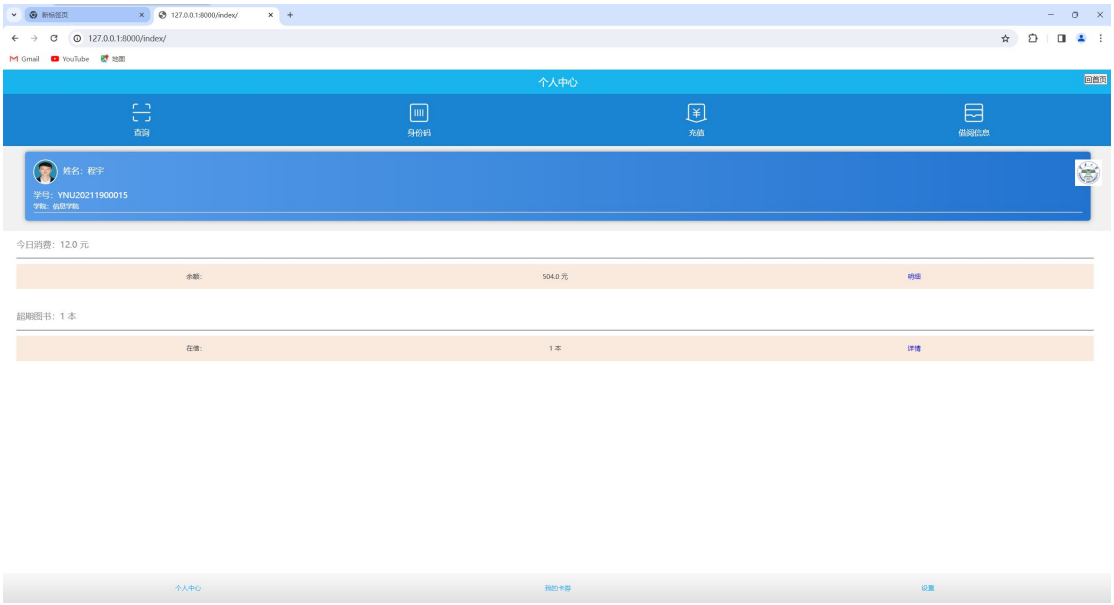


图 15：登录界面

然后我设置了身份码的生成，比如说在刷卡的时候，可以直接调用二维码进行身份验证等，然后依葫芦画，又设置了充值的方式，结果如图所示：

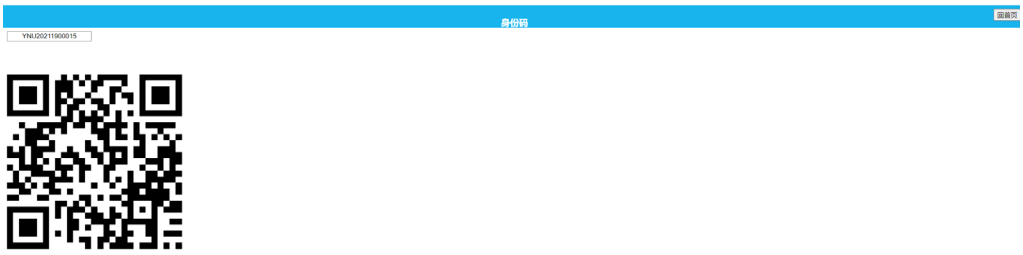


图 16：生成二维码信息

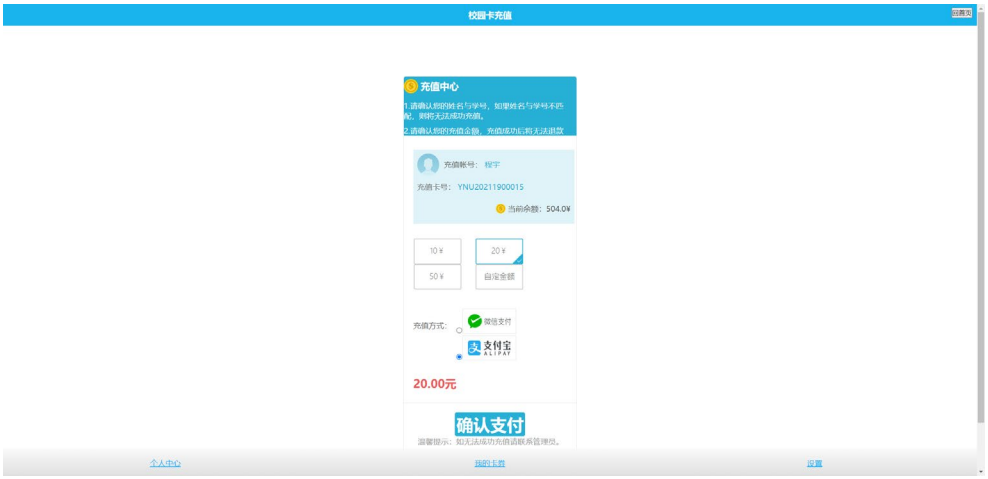


图 17：充值中心

在上面通过管理员进行设置的消费也在这里可以看到，也可以查看本周的消费记录，也可以看自己的借阅书籍，因为在这个过程中我并没有加入图书的书名等这些信息，所以只能看到书的编号。

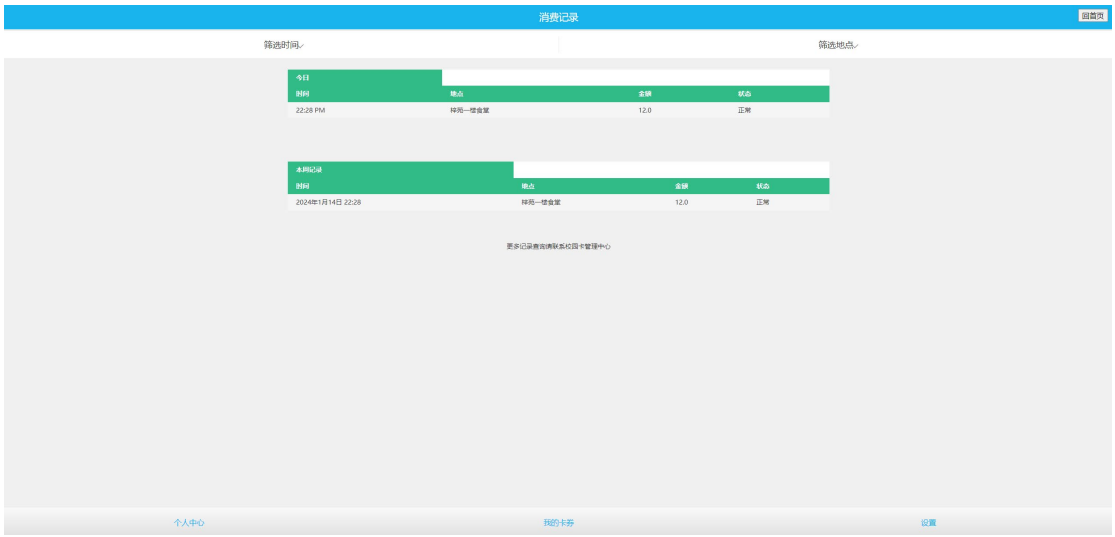


图 18：消费记录

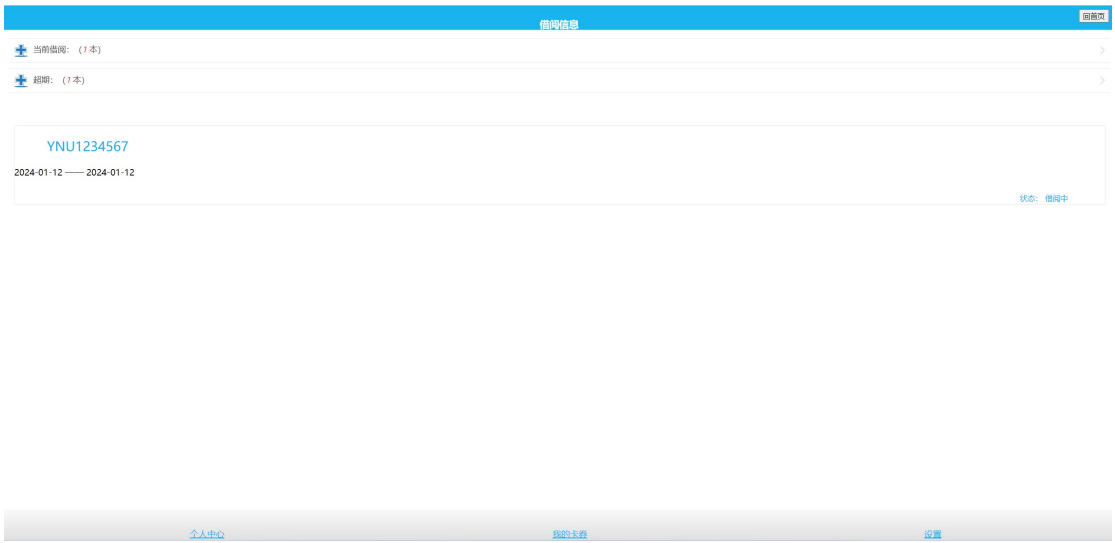


图 19：借阅记录

当然也有修改密码的操作，将自己的密码更改，我这里把原先的 123456 更改为 123456789 进行试验，然后通过 navicate 进行查看，发现信息更正正确，可以顺利的进入到系统中。

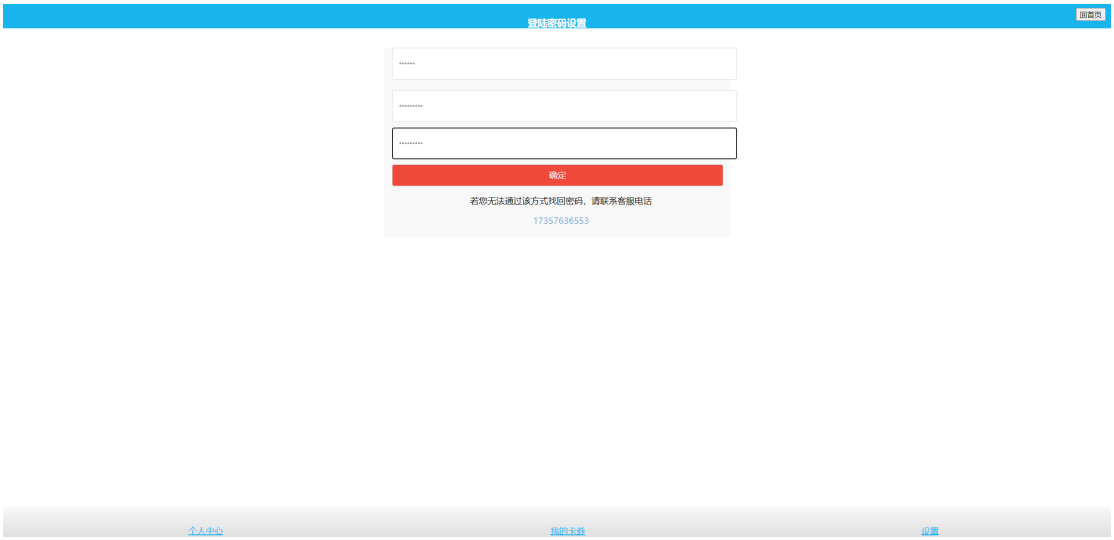


图 20：更改密码

card_id	balance	state	create_time	password
YNU20211900015	504	1	2024-01-13 21:37:02.00(	123456789
YNU20211910103	201	1	2024-01-13 21:37:02.00(	123456

图 21：更改密码结果

其中，消费记录和借阅记录和数据库关联的代码为：

```
# 消费记录查询
class ConsumeRecordView(View):
    @staticmethod
    def processdate(records):
        for record in records:
            record.time = record.time.strftime('%H:%M %p')
        return records

    # get 方法直接返回页面
    def get(self, request):
        account = request.session.get('card_id', None)
        # 已登陆
        if account:
            monday = date.today() - timedelta(days=date.today().weekday())
            consume = ConsumeRecord.objects.filter(card_id=account,
time__gte=monday).order_by('-time')
            consume_today = ConsumeRecord.objects.filter(card_id=account,
time__day=datetime.now().day).order_by('-time')
            consume_today = ConsumeRecordView.processdate(consume_today)
            return render(request, 'consumption-record.html', {
                'consume_today': consume_today,
                'consume': consume,
            })
```

```
            else:
                return HttpResponseRedirect("/login/")
# 借阅记录查询
class BorrowInfoView(View):
    @staticmethod
    def processdate(records):
        for record in records:
            record.borrow_time = record.borrow_time.strftime('%Y-%m-%d')
            record.ending_time = record.ending_time.strftime('%Y-%m-%d')
        return records

    def get(self, request):
        account = request.session.get('card_id', None)
        # 已登陆
        if account:
            books = Library.objects.filter(card_id=account, state='unreturned').order_by('-
borrow_time')
            books_over_time = books.filter(ending_time__lt=date.today())
            books_num = len(books)
            books_over_time_num = len(books_over_time)
            books = BorrowInfoView.processdate(books)
            return render(request, 'borrow-info.html', {
                'books': books,
                'books_num': books_num,
                'books_over_time_num': books_over_time_num,
            })
        else:
            return HttpResponseRedirect("/login/")
```

充值的代码和数据库关联为：

```
# 充值
class RechargeView(View):
    def get(self, request):
        account = request.session.get('card_id', None)
        if account:
            card = Card.objects.get(card_id=account)
            student = Student.objects.get(card_id=account)
            name = student.name
            balance = card.balance
            return render(request, 'recharge.html', {
                'name': name,
                'balance': balance,
                'id': account
            })
```

```
            else:
                return HttpResponseRedirect("/login/")

        def post(self, request):
            # 获取用户名和密码
            account = request.session.get('card_id', None)
            card = Card.objects.get(card_id=account)
            re_id = card.card_id
            re_value = request.POST.get('recharge_value', '')
            pay_mode = request.POST.get('radio', '')
            # 插入记录
            re_value = eval(re_value)
            if re_value > 0:
                obj = RechargeRecord(card_id=card, value=re_value, mode=pay_mode)
                card.balance += re_value
                card.save()
                obj.save()
                return render(request, 'recharge_success.html', {'re_value': re_value})
```

更改密码与数据库关联的代码为：

```
# 更改登陆密码
class UpdatePasswordView(View):
    def get(self, request):
        return render(request, 'login-pssword.html')

    def post(self, request):
        account = request.session.get('card_id', None)
        if account:
            update_form = UpdatePasswordForm(request.POST)
            if update_form.is_valid():
                # 获取表单内容
                old_password = request.POST.get('old_password', '')
                new_password = request.POST.get('new_password', '')
                confirm_password = request.POST.get('confirm_password', '')
                card = Card.objects.get(card_id=account)
                # 符合修改条件
                if old_password == card.password and new_password ==
confirm_password:
                    card.password = confirm_password
                    card.save()
                    return render_to_response('change-password_successfully.html',
locals())
                else:
                    return render(request, 'change-password-failed.html', {'msg': '原密码错
```

```
误或两次新密码不同'})
                else:
                    return render(request, 'change-password-failed.html', {'msg': '请填写原密码
'})
            else:
                return HttpResponseRedirect("/login/")
# 支付方式设置
class PaymentSetView(View):
    def get(self, request):
        account = request.session.get('card_id', None)
        if account:
            card = Card.objects.get(card_id=account)
            return render(request, 'payment-set.html')
        else:
            return HttpResponseRedirect("/login/")
# 支付验证设置
class PayAuthSetView(View):
    def get(self, request):
        account = request.session.get('card_id', None)
        if account:
            card = Card.objects.get(card_id=account)
            return render(request, 'pay-auth-set.html')
        else:
            return HttpResponseRedirect("/login/")
```

其代码的详细地址已经上传到 GitHub 中，地址为：https://github.com/Chengyu0918/YNU-campuscard

第六步、数据库运行和维护

1. 定期备份：定期备份数据库是防止数据丢失的重要步骤。应该设定自动备份计划，并确保备份在安全的地方存储，最好是离线和/或在不同的物理位置。
2. 性能监控：使用工具监控数据库的性能，包括查询速度、响应

时间和资源使用情况。根据监控结果调整数据库配置，优化查询和索引，以提高效率。

3. 安全性管理：定期更新数据库软件，以确保安全性和性能。实施强密码政策和访问控制，以防止未授权访问。配置防火墙和其他安全措施，防止网络攻击。
4. 灾难恢复计划：制定灾难恢复计划，以便在发生数据丢失或损坏时快速恢复。确保备份可用并且可以迅速部署。
5. 数据清理和维护：定期清理数据库，删除过时或不再需要的数据。对数据库进行碎片整理，优化数据存储和访问速度。
6. 更新和迁移：在需要时进行数据库的更新和迁移。确保在执行这些操作之前进行充分的测试，以防止数据丢失或兼容性问题。
7. 监控日志文件：定期检查数据库的日志文件，以便及时发现并解决潜在的问题，如错误的查询、未授权的访问尝试等。
8. 数据库的扩展性和可用性：确保数据库能够处理增长的数据量和访问请求。必要时进行硬件升级或优化软件配置，考虑使用集群或复制来提高可用性。
9. 技术支持和文档：保持数据库的相关文档更新，包括配置信息、自定义脚本和操作指南。确保团队成员了解如何操作和维护数据库，必要时提供培训。
10. 合规性和法律要求：确保数据库的操作和数据处理符合相关的数据保护法律和行业标准。

实验心得

- 这次实验是一个很好的练习实验，之前在暑假实训的时候，就做过一个购物平台的实验，然后做出来的效果并不怎么样，对于数据库的了解也不够透彻,所以最后只是简单的做出来了一个界面，这次我尝试用 django 框架搭建一个框架，让我更加熟悉 python 这门语言，也能够让我在这个学习过程中，了解到数据库映射等问题，帮助我理解在老师上课过程中讲到的数据优化等等相关的问题，其实最难的就是设计数据库的部分，我在这个地方构思了很久然后参看了云大一卡通的小程序设计平台，构想了几个数据库做出来了,在这个实验的过程中,其实用到的 sql 语言并不多，也明白了其上课说到的‘寄生’的意义，对我来说也是很有深刻意义的。
- 关于我的 MySQL 数据表，我在导入的时候，不知道为什么设置了整个表支持 unicode 全部编码，但是插入中文的时候还是出现了很多的错误，以至于现在我对于这个还是不太懂，然后我就把每一个列单独设置支持 Unicode 编码之后才可以，代码为：alter table ecard.operation_consumerecord change location location varchar(40) character set utf8mb4;
- 在设计的过程中,在写报告的过程中我又翻看了一边数据库的书，其实我觉得现在仍然还没有入门，我只是懂了开发的流程而已，还需要大量的实践，这次的阅读并不是像备考一样只看重点，我现在对于数据库设计过程中为何设计，设计的好不好，包括范式

这些都需要有一定的理解之后才能设计出好的表格形式，当然我也觉得不一定 BCNF 范式一定是最好的，还是需要和实际情况结合起来看的。当然在完成这个项目的时候，也许到了很多编程上面的问题，包括图像显示等等，还有密码加密处理和数据的保存形式，以及用户的权限，结合了我这个学期上的《信息安全和实践》的课程尝试实现了一下，就不多赘述了。

- 在使用 navicat 的时候，我个人感觉更符合我设计的直观感受，可以直接看每个表中的数据是什么，数据的定义等等，方便我进行查询，然后进行观察校对，所以在上面的实验中，我很少使用 MySQL workbench 来进行查询，而是 navicat 进行操作。
- 在我写 sql 语言的时候，还是需要反复的观看数据库第三章的内容的，因为自己不够熟悉，写的不够多，所以经验也不是很足够，还需要多多尝试和实践。
- 这个系统中还有很多功能并没有实现，比如说管理员的分级操作，和借阅信息书的管理，数据进行分析。我就着重实现了平时使用比较频繁的功能。其中数据表的定义中，我只展示了一点关于学生表的数据库，其他的也都类似于学生表的数据定义，也没有多家展示占用篇幅。