

## 学生进出管理系统报告

本报告分为：1.基础情况，2.ERD 模型与创建表，3.登录功能，4.菜单功能总述，5.学生功能，6.老师功能，共六项，如需跳转请查看书签

### 基础情况

#### 1. 运行环境

- ✓ 编译 PyCharm 2021.1.2
- ✓ 语言 Python 3.9
- ✓ 数据库 postgresQL 13
- ✓ 数据库连接 psycopg2
- ✓ 界面设计 pyqt5

#### 2. 源代码组成+本报告

Construct.py 建立数据库，表格，插入测试数据

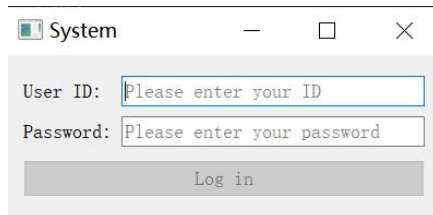
Login.py 用户登录界面

Function.py 用户操作界面

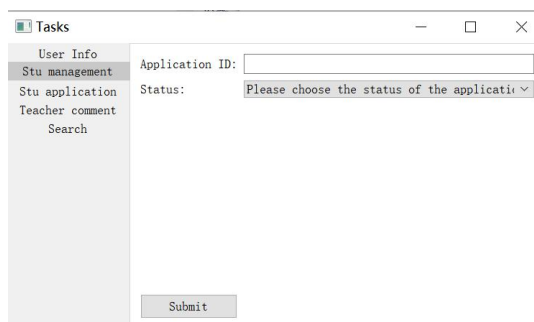
#### 3. 注意事项

- 需要测试请直接执行 Login 文件。
- 在其他计算机中使用时，需要将对源代码中的文件的 conn 参数进行修改。

```
conn = psycopg2.connect(database="Project", user="postgres", password="010504", host="127.0.0.1", port="5432")
```
- 每一次使用软件时都会进行数据库的完全初始化，取消运行文件 Construct.py 可解决。
- 使用系统的用户不可以在界面中自行注册，只能允许数据库中已有的用户进行操作。



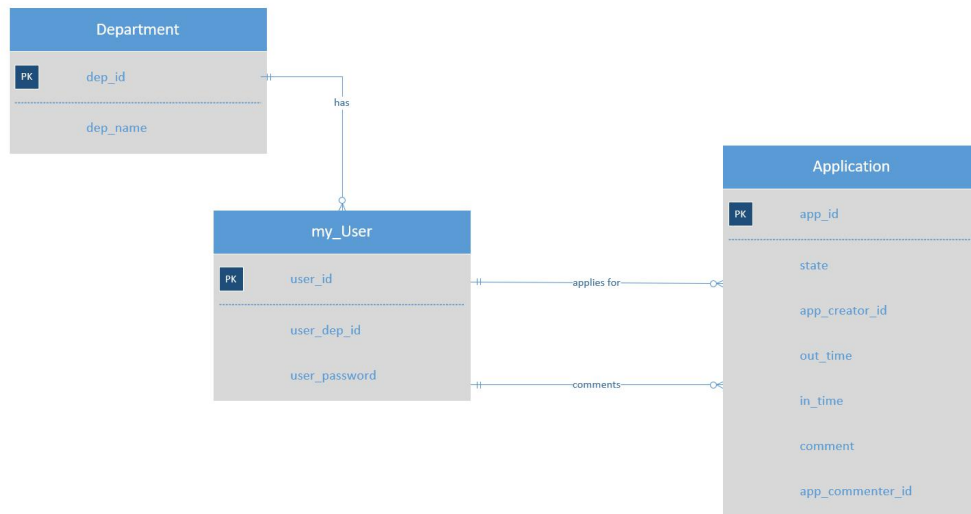
- 用户的身份由用户 ID 进行识别，在本系统中，学生用户的 ID 的第一位数字为 1，老师用户的 ID 的第一位数字为 0。
- 用户可以进行的操作在登录后进入界面方可进行操作。



- 由于每次执行后都进行了查询页面的确认，不再对执行结果做 SQL 中的表格截图
- 本次大作业的设计中，为了便于调试，便于使用测试，考虑到生成大量数据的难度，将用户名和各表的主键 ID 都设置的较小，输入更加方便。

## ERD 模型

## ERD 模型



## 插入测试数据

```
cur = conn.cursor()
sql = '''INSERT INTO Department(dep_id,dep_name)
VALUES ('01','工业工程系'),
('02','自动化系');
'''
cur.execute(sql)

cur = conn.cursor()
sql = '''INSERT INTO my_user(user_id,user_dep_id,user_password)
VALUES ('0001','01','0000'),
('0002','02','0000'),
('1001','01','0000'),
('1002','01','0000'),
('1003','02','0000');
'''
cur.execute(sql)

cur = conn.cursor()
sql = '''INSERT INTO Application(app_id,state,app_creator_id,out_time,in_time,app_commenter_id)
VALUES ('1','passed','1001','2021-06-25 12:00:00','2021-06-25 17:00:00','0001'),
('2','passed','1001','2021-06-26 13:00:00','2021-06-26 19:00:00','0001'),
('3','passed','1001','2021-06-27 13:00:00','2021-06-27 14:00:00','0001');
'''
cur.execute(sql)

cur = conn.cursor()
sql = '''INSERT INTO Application(app_id,state,app_creator_id,out_time,in_time)
VALUES ('4','pending','1002','2021-06-25 14:00:00','2021-06-26 19:00:00'),
('5','pending','1003','2021-06-21 12:00:00','2021-06-21 14:00:00'),
('6','pending','1003','2021-06-02 12:00:00','2021-06-03 12:00:00');
'''
cur.execute(sql)
```

Department		
PK	dep_id	CHAR(2)
	dep_name	VARCHAR(5)

Department(dep\_id, dep\_name) 储存院系的信息  
dep\_id 代表院系 ID，是表的主键，统一长度的字符串  
dep\_name 代表院系的称呼，可变长度的字符串

```
cur = conn.cursor()
sql = '''CREATE TABLE Department(
        dep_id CHAR(2) PRIMARY KEY NOT NULL UNIQUE,
        dep_name VARCHAR(5) NOT NULL
    );'''
cur.execute(sql)
```

```
postgres=# \c Project
您现在已经连接到数据库 "Project", 用户 "postgres".
Project=# SELECT * FROM Department;
 dep_id | dep_name
-----+-----
 01     | 工业工程系
 02     | 自动化系
(2 行记录)

Project=# \d Department;
          数据表 "public.department"
+-----+-----+-----+-----+
|  栏位  | 数据类型  | 校对规则 | 可空的 | 预设 |
+-----+-----+-----+-----+
| dep_id | character(2) |          | not null |      |
| dep_name | character varying(5) |          | not null |      |
+-----+-----+-----+-----+
索引:
    "department_pkey" PRIMARY KEY, btree (dep_id)
由引用:
    TABLE "my_user" CONSTRAINT "my_user_user_dep_id_fkey" FOREIGN KEY (user_dep_id) REFERENCES department(dep_id) ON UPDATE CASCADE
```

my_User		
PK	user_id	CHAR(4)
FK	user_dep_id	CHAR(2) FK >- dep_id
	user_password	VARCHAR(4)

my\_User(user\_id, user\_dep\_id, user\_password) 储存用户信息  
user\_id 代表用户的 ID，是表的主键，统一长度的字符串  
user\_dep\_id 代表用户的院系 ID，外键，指向 Department 表的 dep\_id 属性  
user\_password 代表用户的密码，是可变长度的字符串，用于匹配登录

```
cur = conn.cursor()
sql = '''CREATE TABLE my_User(
        user_id CHAR(4) PRIMARY KEY NOT NULL UNIQUE,
        user_dep_id CHAR(2) NOT NULL REFERENCES Department(dep_id) ON UPDATE CASCADE,
        user_password VARCHAR(4) NOT NULL
    );'''
cur.execute(sql)
```

```
Project=# \d my_User;
数据表 "public.my_user"
+-----+-----+-----+-----+-----+
| 栏位 | 数据类型 | 校对规则 | 可空的 | 预设 |
+-----+-----+-----+-----+-----+
| user_id | character(4) |  | not null |  |
| user_dep_id | character(2) |  | not null |  |
| user_password | character varying(4) |  | not null |  |
+-----+-----+-----+-----+-----+
索引:
    "my_user_pkey" PRIMARY KEY, btree (user_id)
外部键(FK)限制:
    "my_user_user_dep_id_fkey" FOREIGN KEY (user_dep_id) REFERENCES department(dep_id) ON UPDATE CASCADE
由引用:
    TABLE "application" CONSTRAINT "application_app_commenter_id_fkey" FOREIGN KEY (app_commenter_id) REFERENCES my_user(user_id)
    TABLE "application" CONSTRAINT "application_app_creator_id_fkey" FOREIGN KEY (app_creator_id) REFERENCES my_user(user_id) ON UPDATE CASCADE

Project=# SELECT * FROM my_User;
 user_id | user_dep_id | user_password
+-----+-----+-----+
| 0001 | 01 | 0000
| 0002 | 02 | 0000
| 1001 | 01 | 0000
| 1002 | 01 | 0000
| 1003 | 02 | 0000
(5 行记录)
```

Application		
PK	app_id	VARCHAR(4)
FK	app_creator_id	CHAR(4) FK >- user_id
	state	VARCHAR(10)
	out_time	TIMESTAMP
	in_time	TIMESTAMP
	comment	VARCHAR(100)
	app_commenter_id	CHAR(4) FK >- user_id

Application(app\_id, app\_creator\_id, state, out\_time, in\_time, comment) 儲存用戶的申請信息

app\_id 代表申請的 ID，是表的主鍵，統一長度的字符串

state 代表申請所處狀態，pending, passed, refused, cancelled 四種狀態中的一個，可變長度的字符串。

app\_creator\_id 代表申請人，外鍵，指向 my\_User 表的 user\_id 屬性，統一長度的字符串

out\_time 代表申請的出校時間（包括日期和時間），時間戳

in\_time 代表申請的入校時間（包括日期和時間），時間戳

comment 代表評價，可變長度的字符串

app\_commenter\_id 代表對這個申請做審核的老師 ID，外鍵，指向 my\_User 表的 user\_id 屬性，統一長度的字符串

```
cur = conn.cursor()
sql = '''CREATE TABLE Application(
    app_id VARCHAR(4) PRIMARY KEY NOT NULL UNIQUE,
    state VARCHAR(10) NOT NULL CHECK(state in ('pending','passed','refused','cancelled')),
    app_creator_id CHAR(4) REFERENCES my_User(user_id) ON UPDATE CASCADE NOT NULL,
    out_time TIMESTAMP NOT NULL,
    in_time TIMESTAMP NOT NULL,
    comment VARCHAR(100),
    app_commenter_id CHAR(4) REFERENCES my_User(user_id)
);'''
cur.execute(sql)
```

Project=# SELECT * FROM Application;						
app_id	state	app_creator_id	out_time	in_time	comment	app_commenter_id
1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00		0001
2	passed	1001	2021-06-26 13:00:00	2021-06-26 19:00:00		0001
3	passed	1001	2021-06-27 13:00:00	2021-06-27 14:00:00		0001
4	pending	1002	2021-06-25 14:00:00	2021-06-26 19:00:00		
5	pending	1003	2021-06-21 12:00:00	2021-06-21 14:00:00		
6	pending	1003	2021-06-02 12:00:00	2021-06-03 12:00:00		
(6 行记录)						

Project=# \d Application				
栏位	数据类型	校对规则	可空的	预设
app_id	character varying(4)		not null	
state	character varying(10)		not null	
app_creator_id	character(4)		not null	
out_time	timestamp without time zone		not null	
in_time	timestamp without time zone		not null	
comment	character varying(100)		not null	
app_commenter_id	character(4)		not null	

索引:  
 "application\_pkey" PRIMARY KEY, btree (app\_id)

检查约束限制  
 "application\_state\_check" CHECK (state::text = ANY (ARRAY['pending'::character varying, 'passed'::character varying, 'refused'::character varying, 'cancelled'::character varying]::text[]))

外部键(FK)限制:  
 "application\_app\_commenter\_id\_fkey" FOREIGN KEY (app\_commenter\_id) REFERENCES my\_user(user\_id)  
 "application\_app\_creator\_id\_fkey" FOREIGN KEY (app\_creator\_id) REFERENCES my\_user(user\_id) ON UPDATE CASCADE

## 表间的关系

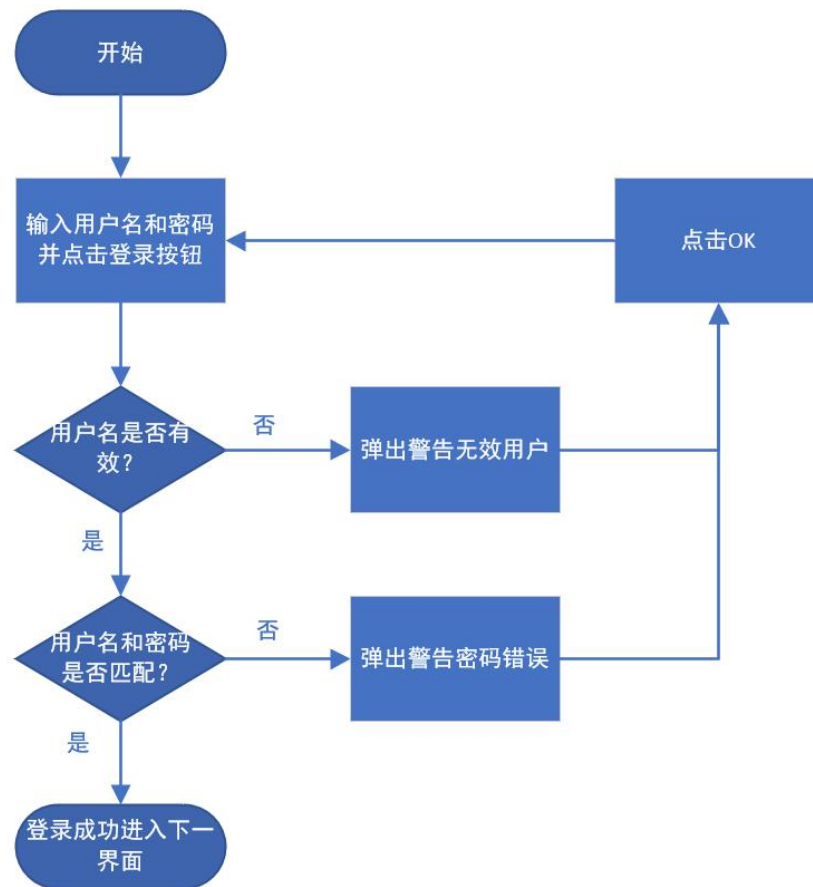
- Department 和 my\_User: 每个用户只能属于一个院系，每个院系可以有多名用户
- my\_User 和 Application:  
 applies for: 一个学生用户可以发起任意多个申请，但每个申请都只能由一个学生用户填写  
 comment: 一个老师用户可以管理多个申请，一个申请只能由一个老师管理

## 如何完成各项操作

- ✓ 对 application 的操作 - 学生
  - 展示不同状态的申请，使用四个状态信息 SELECT 即可
  - 更改申请，首先通过 USER\_ID 查找到某个学生的所有申请，在在某个申请中，可以查询到属性 COMMENT, 定位到当前申请的审批情况，只有当 COMMENT 处于 refused 或者 passed 的状态，学生才可以对该申请进行 update 的操作。
  - 填写申请，INSERT 操作，需要注意对时间冲突的处理，对出入时间进行重叠查询（限于该学生），对时间间隔进行比较，如果出现冲突错误就插入失败。
  - 分类和筛选，使用 SELECT 操作根据当前学生的申请显示。
- ✓ 对 application 的操作 - 老师
  - 对学生的申请进行审批，使用 INSERT 在 COMMENT 表中添加一行数据。
  - 对属于自己院系的申请进行分组查看，使用 SELECT 在对该老师所在院系申请进行展示。
  - 筛查申请，用 SELECT 在对该老师所在院系申请分类排序进行展示。

## 登录页

该部分实现功能需要将界面中用户输入的信息和数据库中的信息进行比对。



对用户名是否存在的检验代码

```

cursor = conn.cursor()
cursor.execute("SELECT COUNT(*) FROM my_User WHERE user_id LIKE %s", (self.user_line.text(),))
number = cursor.fetchall()

```

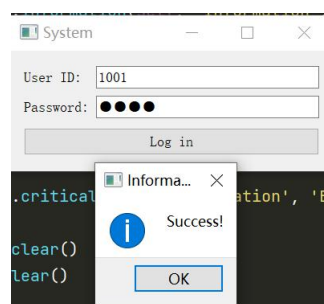
对用户名对应密码的检验代码

```

cursor = conn.cursor()
cursor.execute("SELECT user_password FROM my_User WHERE user_id LIKE %s", (self.user_line.text(),))
results = cursor.fetchall()

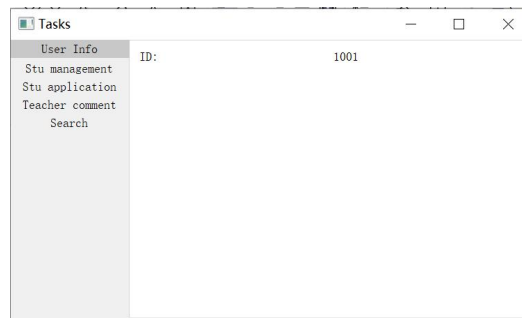
```

用户打开运行 Login 文件后，通过界面输入用户名和密码，点击 Log in 按钮，将 ID 和相应的密码传入，两次检验后将出现登录成功的信息提示。

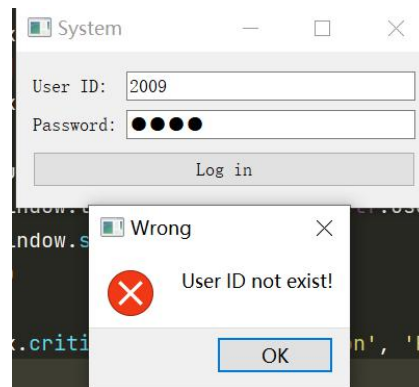




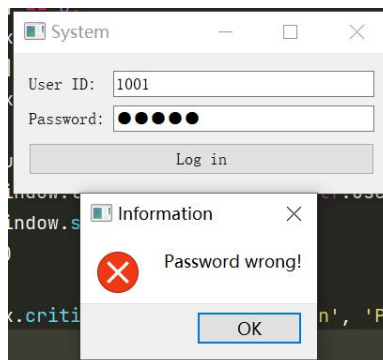
点击 OK，可进入功能界面。



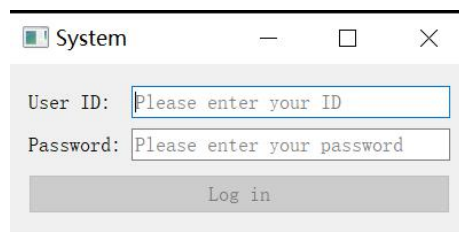
如果输入的用户名在后端数据库中沒有匹配到，就出现报错信息。



如果用户名可以匹配但是密码与后端数据库中的密码不匹配，出现报错信息。



在两种报错信息提示框内，点击报错信息的按钮 OK，就回到登录页面，此时将清空所填数据，需要重新输入。



由于学生进出校管理系统，为针对学生身份和老师身份的固定人群使用，不支持注册新身份，因此没有设置注册用户的功能，所有用户的信息需要在后端导入。

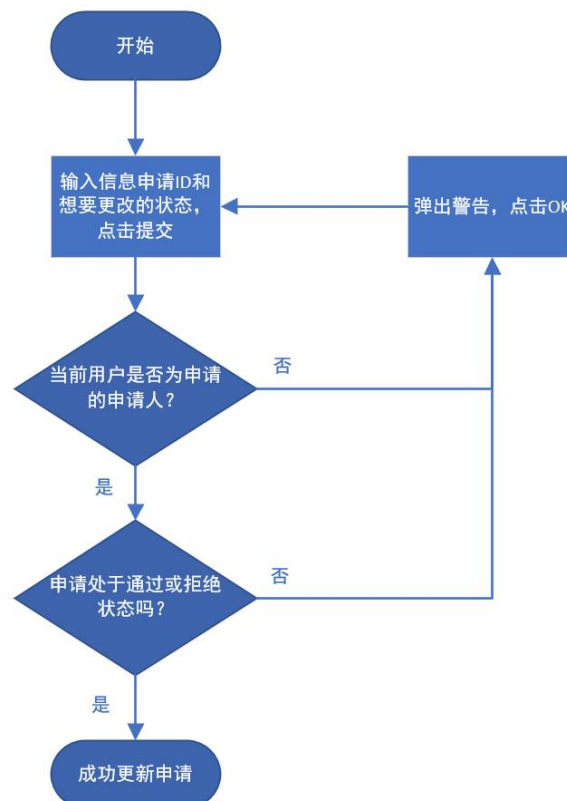
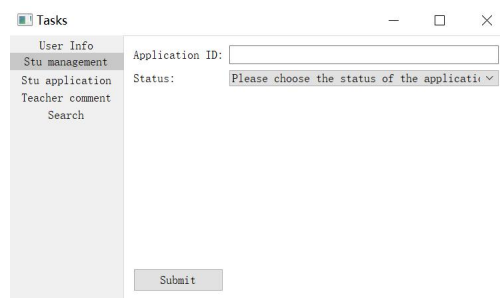
## 功能页面总述

为实现需求中的各项功能，将功能页面的左边设置为五个功能的功能菜单。

在输入正确的用户名和密码后默认打开的是 **User Info** 页面，此页面是显示当前用户的用户名，不允许用户操作，此处的用户 ID 为登录成功时的用户 ID。



菜单栏的 **Stu management** 为学生管理申请界面，根据用户名，只允许输入人为想要管理的申请的申请人操作，其余人执行将报错。





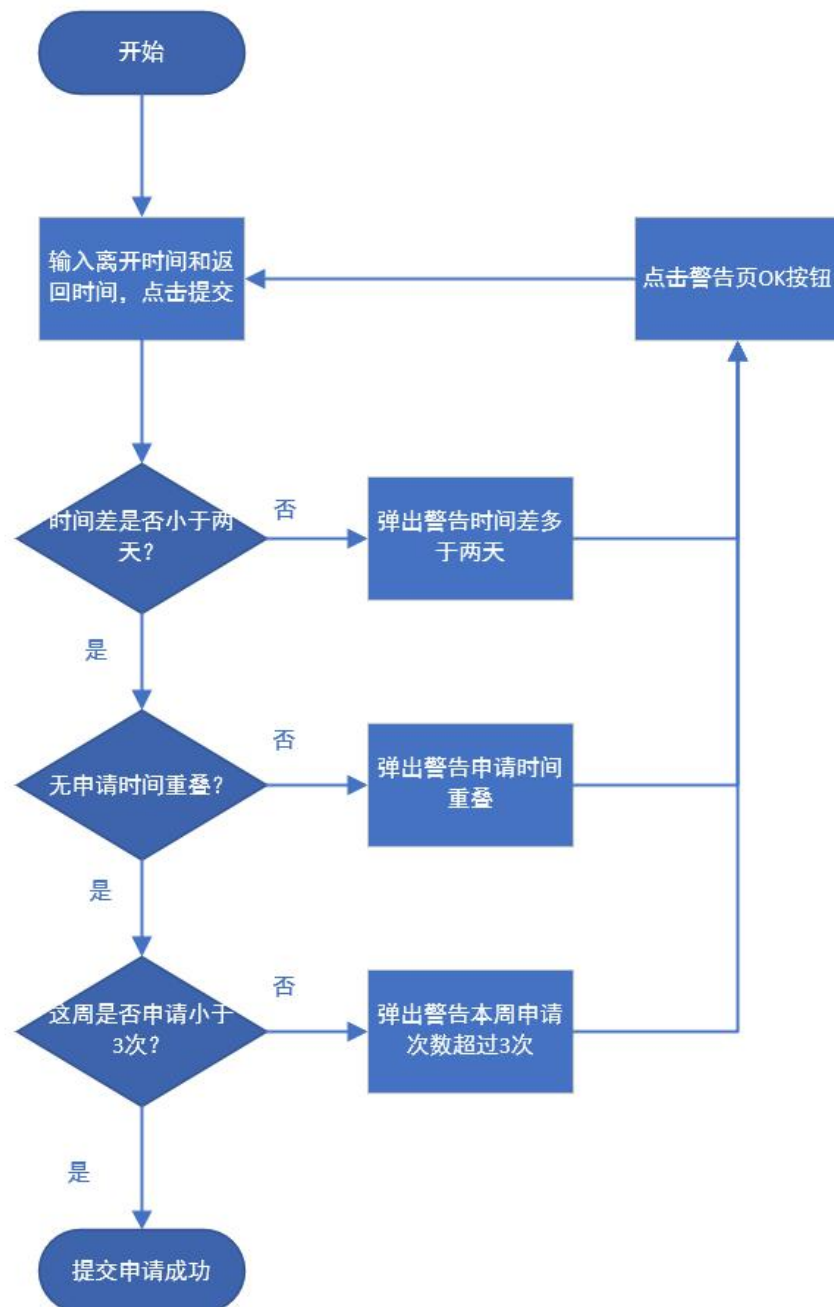
菜单栏的 **Stu application** 为学生填写新的申请界面。

Tasks

User Info  
Stu management  
Stu application  
Teacher comment  
Search

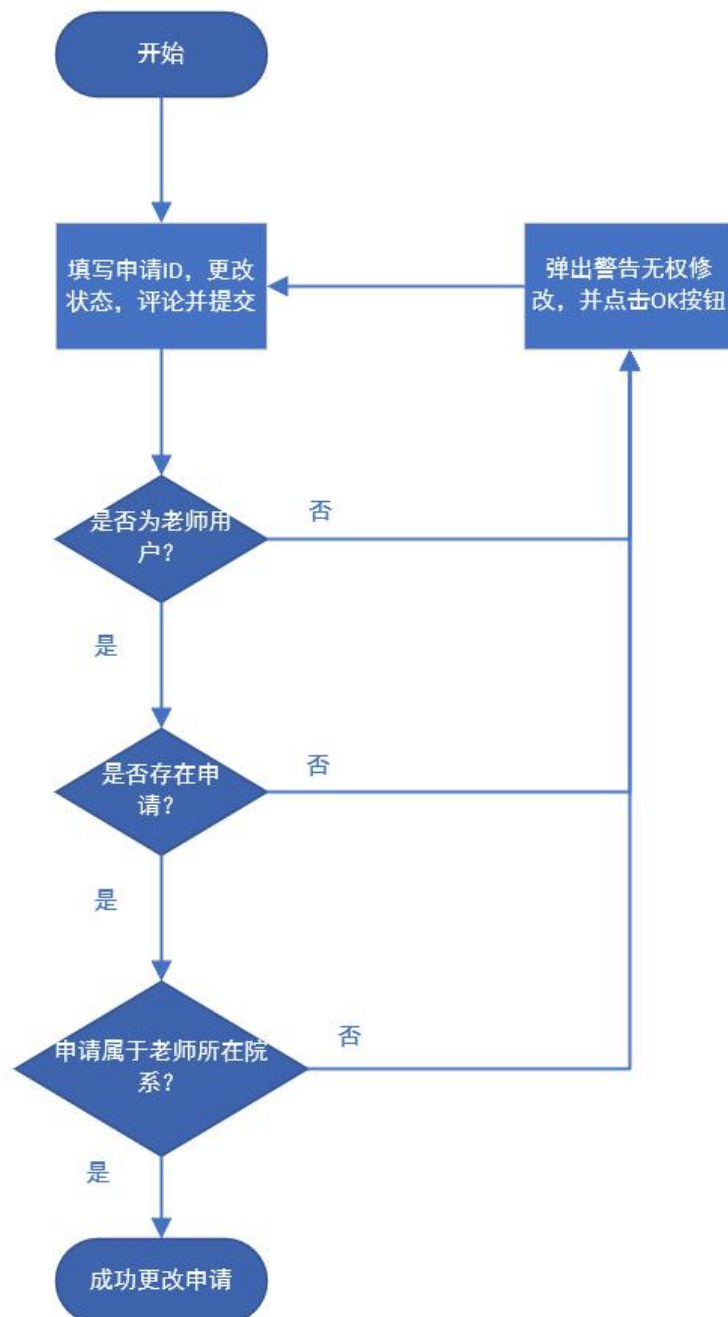
OUT time:   
IN time:

Submit

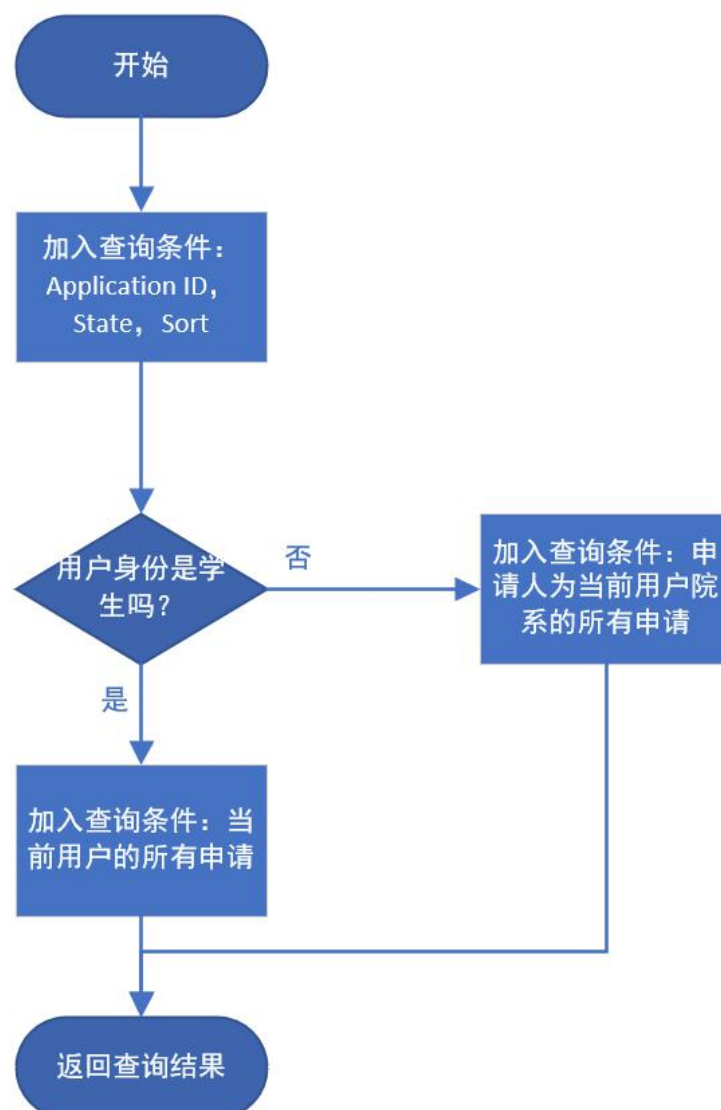
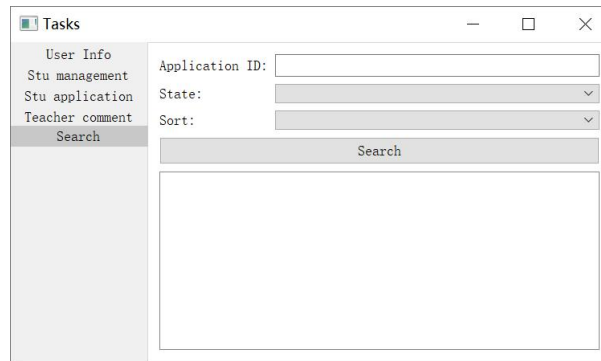


菜单栏的 **Teacher comment** 为老师对申请进行通过不通过和评论的界面,此处将检验输入人为老师用户。

The screenshot shows a web application window titled "Tasks". On the left is a sidebar menu with items: "User Info", "Stu management", "Stu application", "Teacher comment" (which is highlighted), and "Search". The main content area on the right has three input fields: "Application ID:" followed by a text box, "State:" followed by a dropdown menu showing "Please choose the status of the applicati", and "Comment:" followed by a text box. At the bottom center of the main area is a "Submit" button.



菜单栏中的综合搜索栏 **Search**。在这里，学生用户可以实现综合检索自己的申请，老师用户可以实现综合检索与该老师院系相同的申请。



详细的功能实现代码和运行情况如下章节

## 学生功能实现——对申请进行展示

展示属于该学生的申请（根据申请，状态，排序方式），即对 **Application** 表进行 **SELECT** 操作，通过四个状态分别查询，并可根据离开时间正序或逆序排列，实现代码如下：

```
if cur_ID[0] == '1':
    query = f"SELECT * FROM Application WHERE app_creator_id='{cur_ID}' "
    if appID:
        query += f" AND app_id='{appID}'"

    if status:
        query += f" AND state='{status}'"

    if sort:
        print(sort)
        if sort == 'DESC':
            query += " ORDER BY out_time DESC"
        if sort == 'ASC':
            query += " ORDER BY out_time ASC"
```

当执行学生用户 1001



查询该学生全部申请

	1	2	3	4	5
1	1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00
2	2	refused	1001	2021-06-26 13:00:00	2021-06-26 19:00:00
3	3	pending	1001	2021-06-27 13:00:00	2021-06-27 14:00:00

查询状态为 **passed** 的申请

	1	2	3	4	5
1	1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00

查询申请，以离开时间递减排序

	1	2	3	4	5
1	3	pending	1001	2021-06-27 13:00:00	2021-06-27 14:00:00
2	2	refused	1001	2021-06-26 13:00:00	2021-06-26 19:00:00
3	1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00

## 学生功能实现——更改申请状态

通过对 Application 表的 UPDATE 操作完成，（只用申请人才可以更改），且只能更改状态为拒绝或通过的申请。一旦更改则需要重新提交申请。

```
if ((creator_id == self.tab1.id_lab.text())
    and (app_state == 'passed' or app_state == 'refused')
    and (self.tab2.status_combo.currentText() == 'cancelled')):
    cursor.execute("UPDATE Application SET state = %s WHERE app_id LIKE %s",
                  (self.tab2.status_combo.currentText(),
                   self.tab2.app_line.text()))
    cursor.close()
    conn.commit()
    QMessageBox.information(self, 'Information', 'You have updated the state of your application!')
    print('submit')
    self.tab2.app_line.clear()
else:
    QMessageBox.critical(self, 'Wrong', 'Only creator can manage this application in passed or refused state!')
```

执行学生用户 1001，初始情况

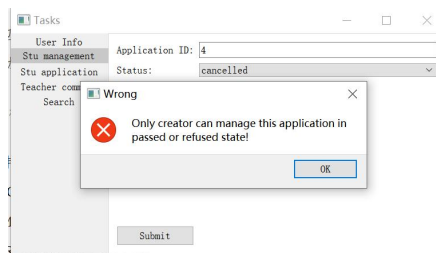
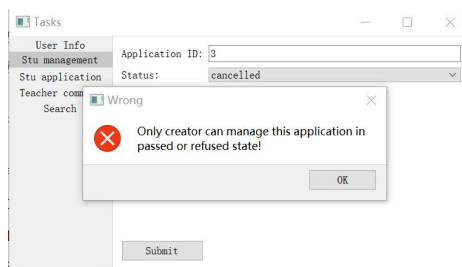
Tasks					
User Info	ID: 1001				
Stu management					
Stu application					
Teacher comment					
Search					

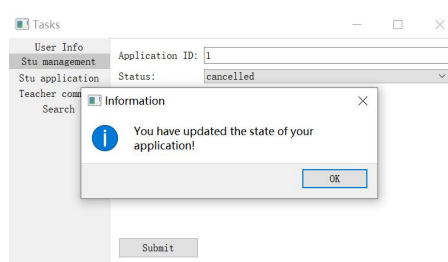
Tasks					
User Info	Application ID:				
Stu management	State:				
Stu application	Sort:				
Teacher comment					
Search	Search				
	1	2	3	4	5
1 1		passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00
2 2		refused	1001	2021-06-26 13:00:00	2021-06-26 19:00:00
3 3		pending	1001	2021-06-27 13:00:00	2021-06-27 14:00:00

修改状态处于 pending 的申请，出现警告

修改不属于自己的申请，出现警告



修改自己的申请，且申请为拒绝或通过



状态改变后

Tasks					
User Info	Application ID:				
Stu management	State:				
Stu application	Sort:				
Teacher comment					
Search	Search				
	1	2	3	4	5
1 2		refused	1001	2021-06-26 13:00:00	2021-06-26 19:00:00
2 3		pending	1001	2021-06-27 13:00:00	2021-06-27 14:00:00
3 1		cancelled	1001	2021-06-25 12:00:00	2021-06-25 17:00:00

## 学生功能实现——填写新的申请

在 Application 表中，需要在插入(INSERT)前检查(SELECT)此处出入时间间隔是否超过 48 小时、原有表中是否有申请时间重叠、一周内申请不可超过三次，只需检查数据库中有没有被申请人取消的申请)。代码如下：

计算时间差

```
cur_ID = self.tab1.id_lab.text()
leaving_time = self.tab3.out_line.text()
return_time = self.tab3.in_line.text()
print(cur_ID)
# 把字符串转为时间类型
leaving_datetime = datetime.datetime.strptime(leaving_time, '%Y-%m-%d %H:%M:%S')
return_datetime = datetime.datetime.strptime(return_time, '%Y-%m-%d %H:%M:%S')
print(leaving_datetime)
print(return_datetime)
# 检查时间差不超过48h
inter = return_datetime - leaving_datetime
flag = (inter <= datetime.timedelta(days=2))
```

查看时间重叠的申请

```
# 检查时间重叠部分
cursor = conn.cursor()
cursor.execute("SELECT app_id FROM Application "
               "WHERE "
               " (state LIKE 'pending' OR state LIKE 'passed' OR state LIKE 'refused') AND "
               " app_creator_id LIKE %s AND "
               " ((out_time <= %s AND in_time > %s) OR "
               " (out_time > %s AND out_time < %s)) ",
               (cur_ID, leaving_time, leaving_time, leaving_time, return_time)
               )
results = cursor.fetchall()
```

查看此处目标周的申请个数

```
# 算出离开周的起始和末尾
leaving_date = leaving_time[:10]
str_s = leaving_date + ' 00:00:00'
str_f = leaving_date + ' 23:59:59'
datetime_s = datetime.datetime.strptime(str_s, '%Y-%m-%d %H:%M:%S')
datetime_f = datetime.datetime.strptime(str_f, '%Y-%m-%d %H:%M:%S')

x = leaving_datetime.weekday() # 离开的星期数-1
week_begin = datetime_s - datetime.timedelta(days=x)
week_end = datetime_f + datetime.timedelta(days=6 - x)
week_begin_str = week_begin.strftime('%Y-%m-%d %H:%M:%S') # 该周起始
week_end_str = week_end.strftime('%Y-%m-%d %H:%M:%S') # 该周结束

# 检查一周离校次数
cursor = conn.cursor()
cursor.execute("SELECT COUNT(*) FROM Application "
               "WHERE "
               " (state = 'pending' OR state = 'passed' OR state = 'refused') AND "
               " app_creator_id LIKE %s AND "
               " out_time >= %s AND "
               " out_time < %s ",
               (cur_ID, week_begin_str, week_end_str)
               )
result1 = cursor.fetchall()
```

对三处检查问题进行分类并对确定是否执行操作或弹出警告

```
if len(results) == 0 and flag and result1[0][0] < 3:
    QMessageBox.information(self, 'Information', 'You have applied!')
    print('Application id = ', self.tab3.app_ID)
    cursor.execute("INSERT INTO Application values(%s, %s, %s, %s,%s)",
                  (self.tab3.app_ID, 'pending', cur_ID, leaving_time, return_time))
    cursor.close()
    conn.commit()
    self.tab3.out_line.clear()
    self.tab3.in_line.clear()
elif flag == 0:
    QMessageBox.critical(self, 'Wrong', 'Check your time interval shorter than 48h!')
elif flag == 1 and len(results) != 0:
    QMessageBox.critical(self, 'Wrong', 'Time overlap!')
elif flag == 1 and len(results) == 0 and result1[0][0] >= 3:
    QMessageBox.critical(self, 'Wrong', 'This week you have been out three times!')
```

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

Application ID:

State:

Sort:

Search

	1	2	3	4	5
1 1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00	
2 2	passed	1001	2021-06-26 13:00:00	2021-06-26 19:00:00	
3 3	passed	1001	2021-06-27 13:00:00	2021-06-27 14:00:00	

申请时间长度超过 48 小时，出现警告

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

OUT time: 2021-06-01 12:00:00

IN time: 2021-06-07 13:00:00

Submit

Wrong

Check your time interval shorter than 48h!

OK

有时间重叠，出现警告

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

OUT time: 2021-06-25 10:00:00

IN time: 2021-06-25 14:00:00

Submit

Wrong

Time overlap!

OK

一周超过 3 次，出现警告

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

OUT time: 2021-06-22 10:00:00

IN time: 2021-06-22 14:00:00

Submit

Wrong

This week you have been out three times!

OK

满足限制后

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

OUT time: 2021-06-07 10:00:00

IN time: 2021-06-08 14:00:00

Submit

Information

You have applied!

OK

Tasks

User Info

Stu management

Stu application

Teacher comment

Search

Application ID:

State:

Sort:

Search

	1	2	3	4	5
1	1	passed	1001	2021-06-25 12:00:00	2021-06-25 17:00:00
2	2	passed	1001	2021-06-26 13:00:00	2021-06-26 19:00:00
3	3	passed	1001	2021-06-27 13:00:00	2021-06-27 14:00:00
4	7	pending	1001	2021-06-07 10:00:00	2021-06-08 14:00:00



## 老师功能实现——对申请进行状态修改和评论

需要在申请表中更新对应的申请，需要检查用户身份，修改的申请所属院系与老师用户院系是否一致，并且此处更改不强制要求评论。检查中需要使用 SELECT 操作，代码：

```

cursor = conn.cursor()
cursor.execute("SELECT COUNT(app_id) FROM Application WHERE app_id LIKE %s",
               (self.tab4.app_line.text(),))
results = cursor.fetchall()

cursor = conn.cursor()
cursor.execute("SELECT app_creator_id FROM Application WHERE app_id LIKE %s",
               (self.tab4.app_line.text(), ))
stu = cursor.fetchall()
stu_id = stu[0][0]

cursor = conn.cursor()
cursor.execute("SELECT user_dep_id FROM my_User WHERE user_id LIKE %s",
               (stu_id,))
stu_dep = stu[0][0]

cursor = conn.cursor()
cursor.execute("SELECT User_dep_id FROM my_User WHERE user_id LIKE %s",
               (self.tab1.id_tab.text(),))
adm = cursor.fetchall()
adm_dep = adm[0][0]

adm_id = self.tab1.id_tab.text()

con = self.tab4.comment_line

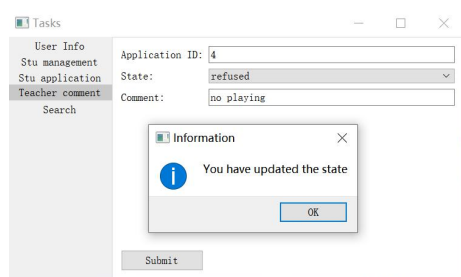
if len(results) != 0 and adm_dep == stu_dep and adm_id[0] == '0':
    cursor = conn.cursor()
    cursor.execute("UPDATE Application SET state = %s WHERE app_id LIKE %s",
                   (self.tab4.state_combo.currentText(),
                    self.tab4.app_line.text()))
    cursor.close()
    conn.commit()
    cursor = conn.cursor()
    cursor.execute("UPDATE Application SET app_commenter_id = %s WHERE app_id LIKE %s",
                   (adm_id,
                    self.tab4.app_line.text()))
    cursor.close()
    conn.commit()
    if con:
        cursor = conn.cursor()
        cursor.execute("UPDATE Application SET comment = %s WHERE app_id LIKE %s",
                       (self.tab4.comment_line.text(),
                        self.tab4.app_line.text()))
        cursor.close()
        conn.commit()
    QMessageBox.information(self, 'Information', 'You have updated the state')
    self.tab4.app_line.clear()
    self.tab4.comment_line.clear()
else:
    self.tab4.app_line.clear()
    self.tab4.comment_line.clear()
    QMessageBox.critical(self, 'Wrong', 'You are not allowed to update the state!')

```

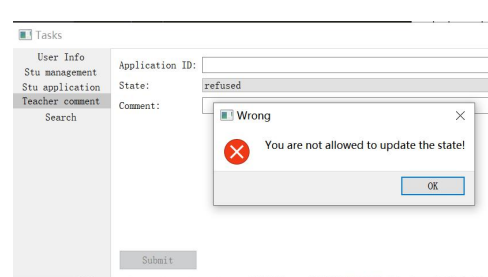
执行老师用户 0001

Search					
1	2	3	4	5	6
1 1	passed	1001	2021-06-25 ...	2021-06-25 ...	None
2 2	passed	1001	2021-06-26 ...	2021-06-26 ...	None
3 3	passed	1001	2021-06-27 ...	2021-06-27 ...	None
4 4	pending	1002	2021-06-25 ...	2021-06-26 ...	None

修改本院系的学生申请



如果修改非本院系的学生申请则出现报错



修改后

Search					
1	2	3	4	5	6
1 1	passed	1001	2021-06-25 ...	2021-06-25 ...	None
2 2	passed	1001	2021-06-26 ...	2021-06-26 ...	None
3 3	passed	1001	2021-06-27 ...	2021-06-27 ...	None
4 4	refused	1002	2021-06-25 ...	2021-06-26 ...	no playing

## 老师功能实现——查看本院系学生情况

即对用户身份进行分类后执行 SELECT 操作，根据用户指定的内容查询申请，代码如下：

```
cursor = conn.cursor()
cursor.execute("SELECT user_dep_id FROM my_User WHERE user_id LIKE %s",
              (cur_ID,))
adm = cursor.fetchall()
adm_dep = adm[0][0]
print(adm_dep)
```

```
else:
    query = f"SELECT * FROM Application JOIN my_User ON app_creator_id = user_id WHERE user_dep_id='{adm_dep}'"
    if appID:
        query += f" AND app_id='{appID}'"

    if status:
        query += f" AND state='{status}'"

    if sort:
        if sort == 'DESC':
            query += " ORDER BY out_time DESC, in_time DESC"
        if sort == 'ASC':
            query += " ORDER BY out_time ASC, in_time DESC"
```

执行老师 0001

Search						
1	2	3	4	5		
1 1	passed	1001	2021-06-25 ...	2021-06-25 ...	None	
2 2	passed	1001	2021-06-26 ...	2021-06-26 ...	None	
3 3	passed	1001	2021-06-27 ...	2021-06-27 ...	None	
4 4	pending	1002	2021-06-25 ...	2021-06-26 ...	None	

无法查看非本院系的申请

Search						
1	2	3	4	5	6	

查看状态为审核中的申请

Search						
1	2	3	4	5	6	
1 4	pending	1002	2021-06-25 ...	2021-06-26 ...	None	

查看状态为已通过的申请，并按离开时间排序

Search						
1	2	3	4	5	6	
1 3	passed	1001	2021-06-27 ...	2021-06-27 ...	None	
2 2	passed	1001	2021-06-26 ...	2021-06-26 ...	None	
3 1	passed	1001	2021-06-25 ...	2021-06-25 ...	None	