# 一、系统结构设计

# 二、数据库基本表的定义

该数据库共有七个实体，八张表（除了七张实体表以外，有一张多对多的关系表，其余的一对多的表均建立为外键关系）。

由于后端是用django编写，且已连接华为云数据库，因此在`model.py`中实现数据库基本表的定义。定义如下。

## 用户管理部分

用户管理部分有学生和管理员两个实体表。

### 学生

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| Sid | varchar(8) | NO | 账号，主键 |
| Spassword | varchar(20) | NO | 密码 |
| Sname | varchar(10) | NO | 姓名 |
| Semail | varchar(100) | YES | 邮箱 |
| Smajor | varchar(20) | YES | 专业 |
| Sgrade | int | YES | 年级 |

后端代码如下

```
class Student(models.Model):
    Sid = models.CharField(max_length=8)
    Spassword = models.CharField(max_length=20)
    Sname = models.CharField(max_length=10)
    Semail = models.CharField(max_length=100, null=True)
    Smajor = models.CharField(max_length=20, null=True)
    Sgrade = models.IntegerField(null=True)
```

### 管理员

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| Aid | varchar(8) | NO | 账号，主键 |
| Apassword | varchar(20) | NO | 密码 |
| Aname | varchar(10) | NO | 姓名 |
| Aemail | varchar(100) | YES | 邮箱 |

后端代码如下

```
class Admin(models.Model):
    Aid = models.CharField(max_length=5)
    Apassword = models.CharField(max_length=20)
    Aname = models.CharField(max_length=10)
    Aemail = models.CharField(max_length=100, null=True)
```

# 课程管理部分

课程管理部分有课程和文件两个实体表。

## 课程

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| CourseId | varchar(10) | NO | 课程号，主键 |
| CourseName | varchar(20) | NO | 课程名称 |
| CourseType | varchar(10) | NO | 课程种类 |
| CourseDescription | varchar(2) | YES | 课程描述 |
| Aid | varchar(10) | NO | 管理者账号，外键 |

后端代码如下

```
class Course(models.Model):
    CourseId = models.CharField(max_length=30)
    CourseName = models.CharField(max_length=20)
    CourseType = models.CharField(max_length=10)
    CourseDescription = models.CharField(max_length=200, null=True)
    Aid = models.CharField(max_length=8)
```

## 文件

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| Fid | varchar(10) | NO | 文件序号，主键 |
| Fname | varchar(20) | NO | 文件名称 |
| Faddr | varchar(200) | NO | 文件地址 |
| CourseId | varchar(10) | NO | 课程号，外键 |

后端代码如下

```
class File(models.Model):
    Fid = models.CharField(max_length=30)
    FName = models.CharField(max_length=20)
    Faddr = models.CharField(max_length=200)
    CourseId = models.CharField(max_length=10)
    Sid = models.CharField(max_length=8)
```

# 社区管理部分

社区管理部分有社区、帖子和评论三个实体表，并且有一个学生-帖子点赞关系表。

## 社区

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| CommunityId | varchar(10) | NO | 社区序号，主键 |
| CommunityName | varchar(20) | NO | 社区名称 |
| Aid | varchar(10) | NO | 管理者账号，外键 |

后端代码如下

```
class Community(models.Model):
    CommunityId = models.CharField(max_length=30)
    CommunityName = models.CharField(max_length=20)
    Aid = models.CharField(max_length=8)
```

## 帖子

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| Pid | varchar(10) | NO | 帖子序号，主键 |
| Sid | varchar(8) | NO | 发布者账号，外键 |
| Ptitle | varchar(30) | NO | 帖子标题 |
| Plabel | varchar(10) | YES | 标签 |
| Pcontent | varchar(500) | NO | 帖子内容 |
| CommunityId | varchar(10) | NO | 社区序号，外键 |

后端代码如下

```
class Post(models.Model):
    Pid = models.CharField(max_length=30)
    Sid = models.CharField(max_length=8)
    Ptitle = models.CharField(max_length=30)
    Plabel = models.CharField(max_length=10, null=True)
    Pcontent = models.CharField(max_length=500)
    Plikes = models.IntegerField(default=0)
    CommunityId = models.CharField(max_length=30)
```

## 评论

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| CommentId | varchar(10) | NO | 评论序号，主键 |
| Sid | varchar(8) | NO | 评论者账号，外键 |
| Pid | varchar(10) | NO | 帖子序号，外键 |
| CommentContent | varchar(200) | NO | 评论内容 |

后端代码如下

```
class Comment(models.Model):
    CommentId = models.CharField(max_length=30)
    Sid = models.CharField(max_length=8)
    Pid = models.CharField(max_length=30)
    CommentContent = models.CharField(max_length=200)
```

## 学生-帖子点赞关系表

| 字段名称 | 数据类型 | 可否为空 | 说明 |
| --- | --- | --- | --- |
| Sid | varchar(8) | NO | 学号，外键 |
| Pid | varchar(10) | NO | 帖子序号，外键 |

后端代码如下

```
class StudentPostLike(models.Model):
    Sid = models.CharField(max_length=8)
    Pid = models.CharField(max_length=30)
```

# 三、系统重要功能实现方法

## 数据库连接

通过修改`django`项目中`settings.py`中的内容，实现与课程组提供的华为云数据库的连接。

```
DATABASES = {
    'default':
    {
        'ENGINE': 'django.db.backends.mysql',     # 数据库引擎
        'NAME': '......', # 数据库名称，db+学号
        'HOST': '......', # 数据库地址
        'PORT': '....', # 端口
        'USER': '......',  # 数据库用户名，学号
        'PASSWORD': '....', # 数据库密码，课程组提供
    }
}
```

## 跨域访问

由于Ssstudy学习平台前端使用vue框架，域名为http://localhost:8080/；后端使用django框架，域名为http://127.0.0.1:8000/。当前端通过HTTP请求与后端进行通信时，需要进行跨域访问。解决方法如下：

1. 在后端使用第三方库`django-cors-headers`，以确保允许来自Vue项目的跨域请求。

   首先执行指令

   ```
   pip install django-cors-headers
   ```

   在django框架中的`settings.py`添加配置如下

   ```
   CORS_ALLOWED_ORIGINS = [
       "http://localhost:8080",  # 前端地址
   ]

   INSTALLED_APPS = [
       #...
       'corsheaders', #corsheaders软件包
       'StudyApp',
   ]

   MIDDLEWARE = [
       'django.middleware.csrf.CsrfViewMiddleware', #csrf令牌
       'corsheaders.middleware.CorsMiddleware',
       #...
   ```

```
]
```

2. 使用`@csrf_exempt`装饰器

   在django项目中的`views.py`的每个后端函数前都添加代码

   ```
   @csrf_exempt
   ```

   Django会跳过对视图函数的CSRF检查，允许没有CSRF令牌的请求，实现跨域访问。

3. 前端使用`axios`库

   运行以下命令安装`axios`库

   ```
   npm install axios
   ```

   然后在前端页面和组件中，添加以下代码

   ```
   import axios from 'axios';
   ```

   这样即可使用`axios`向后端发送请求

   ```
   const response = await axios.get('http://127.0.0.1:8000/Ssstudy/......');
   //针对不同views中的函数发送请求
   ```

通过实现以上三步，完成前后端的连接，实现跨域访问。

# token机制鉴定用户身份

Token机制是一种身份验证（authentication）和授权（authorization）的方式，通常用于在客户端和服务器之间进行安全的用户身份验证。

在用户登陆时，通过使用`token`机制，在用户登录时，身份核验通过后赋予token唯一标识。具体实现在`login_student`和`login_admin`函数中。

# 存储过程

## 用户管理方面

### 学生注册

**涉及的基本表：**Student

**过程描述：**传入学生账号和密码，检查是否有相同的学生账号在Student表中，及学生账号是否为8位。通过检查后，添加新学生到Student表中。

**代码：**

```
@csrf_exempt
def create_student(request):
    if request.method == 'POST':
```

```
        # 解析请求的 JSON 数据
        data = json.loads(request.body)

        # 判断学生账号是否为8位数字
        if not (data['studentNumber'].isdigit() and len(data['studentNumber']) ==
8):
            return JsonResponse({'msg': 'Invalid student number format. It must be
an 8-digit number.', 'error_num': 2})

        try:
            # 尝试获取学生对象，获取成功说明有同账号学生，需要报错
            student = Student.objects.get(Sid=data['studentNumber'])
            response = {'msg': 'Has a same Sid student', 'error_num': 1}
        except Student.DoesNotExist:
            # 没有同账号学生，创建成功
            Student.objects.create(Sid=data['studentNumber'],
Spassword=data['password'], Sname=data['name'])
            response = {'msg': 'success', 'error_num': 0}
        except Exception as e:
            response = {'msg': 'Error creating student: {}'.format(str(e)),
'error_num': 1}

        return JsonResponse(response)
    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

## 管理员注册

**涉及的基本表**：Admin

**过程描述**：传入管理员账号和密码，检查是否有相同的管理员账号在Admin表中，及管理员账号是否为5位。通过检查后，添加新管理员到Admin表中。

**代码**：

```
@csrf_exempt
def create_admin(request):
    if request.method == 'POST':
        # 解析请求的 JSON 数据
        data = json.loads(request.body)

        # 判断管理员账号是否为5位
        if not (data['id'].isdigit() and len(data['id']) == 5):
            return JsonResponse({'msg': 'Invalid admin ID format. It must be a 5-
digit number.', 'error_num': 2})

        try:
            # 尝试获取管理员对象，获取成功说明有同账号管理员，要报错
            admin = Admin.objects.get(Aid=data['id'])

            response = {'msg': 'Has a same Aid admin', 'error_num': 1}
        except Admin.DoesNotExist:
```

```
            # 如果找不到管理员对象，则创建新的管理员对象
            Admin.objects.create(Aid=data['id'], Apassword=data['password'],
Aname=data['name'])
            response = {'msg': 'success', 'error_num': 0}
        except Exception as e:
            response = {'msg': 'Error creating admin: {}'.format(str(e)),
'error_num': 1}

        return JsonResponse(response)
    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

## 学生登录

**涉及的基本表**：Student

**过程描述：**传入学生账号和密码，从Student表中获取信息，判断密码是否正确；若正确，生成JWT令牌并返回带有token的响应，登录成功。

**代码：**

```
SECRET_KEY = secrets.token_urlsafe(32)

@csrf_exempt
def login_student(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        try:
            student = Student.objects.get(Sid=data['id'])
            if data['password'] == student.Spassword:
                # 生成 JWT 令牌
                token_payload = {
                    'user_id': student.id,
                    'username': student.Sid,
                    'exp': datetime.utcnow() + timedelta(days=30)
                }
                token = jwt.encode(token_payload, SECRET_KEY, algorithm='HS256')
                # 返回带有 token 的响应
                response_data = {
                    'error_num': 0,
                    'msg': 'Login successful',
                    'data': {
                        'token': token,
                        'expireAt': (datetime.utcnow() +
timedelta(days=30)).strftime('%Y-%m-%d %H:%M:%S'),
                        'user': {
                            'id': student.Sid,
                            'password': student.Spassword,
                            'name': student.Sname
                        },
                        'permissions': [],
                        'roles': []
```

```
                    }
                }
                return JsonResponse(response_data)
            else:
                # 密码错误
                return JsonResponse({'error_num': 1, 'msg': 'Incorrect password'})
        except Student.DoesNotExist:
            # 没找到对应学生
            return JsonResponse({'error_num': 2, 'msg': 'Student does not exist'})
        except Exception as e:
            return JsonResponse({'error_num': 3, 'msg': 'Error: {}'.format(str(e))})
    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

### 管理员登录

**涉及的基本表**：Admin

**过程描述**：传入管理员账号和密码，从Admin表中获取信息，判断密码是否正确；若正确，生成JWT令牌并返回带有token的响应，登录成功。

**代码**：

```
@csrf_exempt
def login_admin(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        try:
            admin = Admin.objects.get(Aid=data['id'])
            if data['password'] == admin.Apassword:
                # 生成 JWT 令牌
                token_payload = {
                    'user_id': admin.id,
                    'username': admin.Aid,
                    'exp': datetime.utcnow() + timedelta(days=30)
                }
                token = jwt.encode(token_payload, SECRET_KEY, algorithm='HS256')
                # 返回带有 token 的响应
                response_data = {
                    'error_num': 0,
                    'msg': 'Login successful',
                    'data': {
                        'token': token,
                        'expireAt': (datetime.utcnow() +
timedelta(days=30)).strftime('%Y-%m-%d %H:%M:%S'),
                        'user': {
                            'id': admin.Aid,
                            'password': admin.Apassword,
                            'name': admin.Aname
                        },
                        'permissions': [],
                        'roles': []
```

```
                    }
                }
                return JsonResponse(response_data)
            else:
                # 密码错误
                return JsonResponse({'error_num': 1, 'msg': 'Incorrect password'})
        except Admin.DoesNotExist:
            # 没找到对应管理员
            return JsonResponse({'error_num': 2, 'msg': 'Admin does not exist'})
        except Exception as e:
            return JsonResponse({'error_num': 3, 'msg': 'Error: {}'.format(str(e))})
    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

### 获取学生信息

**涉及的基本表**：Student

**过程描述**：传入学生账号，从Student表中获取对应学生的全部信息

**代码**：

```
@csrf_exempt
def get_student_info(request):
    if request.method == 'GET':
        try:
            student_id = request.GET.get('studentNumber')

            # 查询数据库，获取学生信息
            student = Student.objects.get(Sid=student_id)

            # 构建学生信息的字典
            student_info = {
                'Sid': student.Sid,
                'Sname': student.Sname,
                'Semail': student.Semail,
                'Smajor': student.Smajor,
                'Sgrade': student.Sgrade
            }

            return JsonResponse({'student_info': student_info})

        except Student.DoesNotExist:
            return JsonResponse({'error': 'Student not found.'}, status=404)

        except json.JSONDecodeError:
            return JsonResponse({'error': 'Invalid JSON data.'}, status=400)

    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

### 获取管理员信息

**涉及的基本表**：Admin

**过程描述：**传入管理员账号，从Admin表中获取对应管理员的全部信息

**代码：**

```
@csrf_exempt
def get_admin_info(request):
    if request.method == 'GET':
        try:
            admin_id = request.GET.get('adminId')  # 假设管理员ID通过请求参数传递

            # 查询数据库，获取管理员信息
            admin = Admin.objects.get(Aid=admin_id)

            # 构建管理员信息的字典
            admin_info = {
                'Aname': admin.Aname,
                'Aemail': admin.Aemail,
                'AdminId': admin.Aid
            }

            return JsonResponse({'admin_info': admin_info})

        except Admin.DoesNotExist:
            return JsonResponse({'error': 'Admin not found.'}, status=404)

    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

### 更改学生信息

**涉及的基本表**：Student

**过程描述：**传入学生账号和相关信息，从Student表中获取对应学生并修改信息

**代码：**

```
@csrf_exempt
def update_student_info(request):
    if request.method == 'POST':
        # 解析请求的 JSON 数据
        data = json.loads(request.body)

        # 获取学生对象，如果找不到则创建一个新的学生对象

        student = Student.objects.get(Sid=data['Sid'])

        # 更新学生信息
        student.Sname = data['Sname']
```

```
        student.Sgrade = data['Sgrade']
        student.Smajor = data['Smajor']
        student.Semail = data['Semail']

        # 保存更新
        student.save()

        return JsonResponse({'message': 'Student information updated
successfully.'})
    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```

## 更改管理员信息

**涉及的基本表**：Admin

**过程描述：**传入管理员账号和相关信息，从Admin表中获取对应管理员并修改信息

**代码：**

```
@csrf_exempt
def update_admin_info(request):
    if request.method == 'POST':
        # 解析请求的 JSON 数据
        data = json.loads(request.body)

        try:
            # 获取管理员对象，如果找不到则创建一个新的管理员对象
            admin = Admin.objects.get(Aid=data['AdminId'])  # 假设管理员ID通过请求数据传
递

            # 更新管理员信息
            admin.Aname = data['Aname']
            admin.Aemail = data['Aemail']

            # 保存更新
            admin.save()

            return JsonResponse({'message': 'Admin information updated
successfully.'})

        except Admin.DoesNotExist:
            return JsonResponse({'error': 'Admin not found.'}, status=404)

    else:
        return JsonResponse({'error': 'Invalid request method.'}, status=400)
```