1.1.3 后端结构

后端使用Django框架实现。文件结构如下：

- **myapp**
  - **migrations**

    这个文件夹中是Django生成的数据库迁移代码，用于修改模型类后将修改执行到数据库。
  - **models.py**

    该文件定义了模型类，对应数据库的表，具体内容间本文档第二部分。
  - **url.py**

    该文件定义路由，通过设置文件中的路由，可以将不同的请求发送到不同的视图函数中进行处理。
  - **views.py**

    该文件定义视图函数，通过与数据模型进行交互，实现项目的相关功能。
- **backend**

  后端基本文件，用于进行一些配置，主要有Django框架生成。
- **manage.py**

  脚本，用于后端的部署、数据库的迁移等。

二、数据库基本表的定义

数据库共包含10个实体，11张表，采用Django提供的数据库访问接口实现数据库表的定义以及访问

2.1 用户管理部分

这部分内容有关用户的管理，主要包括用户的登录、注册、修改密码等功能。该系统中有三类用户，分别为学生、家教和管理员，其中管理员无法注册，只能由超级管理员（网站管理者）手动添加。考虑到老师和学生具有某些共同的性质（例如账号和密码），但是又有一些不同的属性（例如学生有成绩，家教有学生评分等），因此我们将Student和Tutor抽象为User统一管理，通过identity字段区分学生和家教，并通过**user_id**将User与其他表关联。

```
# 用户表 BCNF
class User(models.Model):
    # 主码、主属性
    user_id = models.BigAutoField(primary_key=True)
    # 主属性
    username = models.CharField(max_length=255)
    password = models.CharField(max_length=255)
    identity = models.IntegerField()  # 0: 管理员,1: 家教, 2: 学生
    registration_date = models.DateField()

    def __str__(self):
        return self.username

    class Meta:
        ordering = ['-registration_date']
```

这种关系的设计可以在不需要继承的前提下使得学生和家教的共同属性可以在User表中维护，而不需要在Student和Tutor中重复定义。同时，这种设计也使得我们可以方便地通过user_id找到对应的学生或家教。

Student

属性名	数据类型	备注
user_id	int	主码、主属性
age	int	-
gender	int	0：男 1：女
contact	varchar	主属性
email	email	主属性
grade	int	-

```
# 学生表 BCNF
class Student(models.Model):
    # 主码、主属性
    user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_student')
    age = models.IntegerField(blank=True,null=True)
    gender = models.CharField(max_length=255,blank=True,null=True)
    # 主属性
    telephone = models.CharField(max_length=255,null=True,blank=True)
    # 主属性
    email = models.EmailField(max_length=255,blank=True,null=True)
    grade = models.CharField(max_length=255,blank=True,null=True)
    address = models.CharField(max_length=255,blank=True,null=True)
    intro= models.TextField(blank=True,null=True)
    personalSignature = models.CharField(max_length=255,blank=True,null=True)


    def __str__(self):
        return self.user_id.username
```

Tutor

属性名	数据类型	备注
user_id	int	主码、主属性
age	int	-
gender	int	0：男 1：女
contact	varchar	主属性
email	email	主属性
rating	int	-

```
# 家教表 BCNF
class Tutor(models.Model):
    # 主码、主属性
    user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_tutor')
    age = models.IntegerField(blank=True,null=True)
    gender = models.CharField(max_length=255,blank=True,null=True)
    # 主属性
    telephone = models.CharField(max_length=255,blank=True,null=True)
    # 主属性
    email = models.EmailField(max_length=255,blank=True,null=True)
    rate = models.FloatField(default=0)
    rateNum = models.IntegerField(default=0)
    address = models.CharField(max_length=255,blank=True,null=True)
    intro= models.TextField(blank=True,null=True)
    personalSignature = models.CharField(max_length=255,blank=True,null=True)
    degree = models.CharField(max_length=255,blank=True,null=True)

    def __str__(self):
        return self.user_id.username
```

Student与Tutor中存在部分共同的属性，但我们并未将其统一放至User中，这是因为我们认为User应当面向登录部分，而不应该包含过多的个人信息。在项目设计中，个人信息部分在**账户管理**页面进行补充，可以直接对应Student和Tutor表，不会带来冗余操作。

2.2 师生关系管理部分

这部分的表展示师生关系，其具体维护逻辑由于与事务逻辑强相关，我们将其放在下一节中详细讨论，此处仅给出表的定义。

Link

属性名	数据类型	备注
link_id	int	主码、主属性
student_id	int	外码
tutor_id	int	外码

```
# 师生关系表 BCNF
class Link(models.Model):
    # 主码、主属性
    link_id = models.BigAutoField(primary_key=True)
    # 外码
    student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_link')
    # 外码
    tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_link')

    def __str__(self):
        return f"Link {self.link_id}"
```

师生关系Link与学生、家教这两个用户通过外码关联，保证了数据的一致性。

2.3 帖子管理部分

这部分内容有关帖子的管理。帖子是家教平台的核心内容，学生与家教的关系通过帖子建立，后续一系列逻辑也从帖子展开。由于学生和家教发布的帖子内容字段相同，我们将其统一放在Post表中，通过**user_id**区分发布者是学生还是家教。

Post

属性名	数据类型	备注
post_id	int	主码、主属性
user_id	int	外码
title	varchar	-
post_date	date	-
start_date	date	-
end_date	date	-
content	test	-
is_completed	boolean	-
is_approved	boolean	-

```
# Post表 BCNF
class Post(models.Model):
    # 主码、主属性
    post_id = models.BigAutoField(primary_key=True)
    # 外码
    user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_post')
    title = models.CharField(max_length=255,blank=True,null=True)
    post_date = models.DateTimeField(blank=True,null=True)
    start_date = models.DateField(blank=True,null=True)
    end_date = models.DateField(blank=True,null=True)
    content = models.TextField(blank=True,null=True)
    is_completed = models.BooleanField()
    is_approved = models.BooleanField(default=False)

    def __str__(self):
        return self.title

    class Meta:
        indexes = [
            models.Index(fields=['post_id']),
            models.Index(fields=['title']),
        ]
```

PostSubject

这个属性的设置是由于每条帖子的subject是一个列表，为规避表中套表的嫌疑，将关系模式规范到3NF及以上，我们将subject单独拿出来，通过外码与Post关联。

属性名	数据类型	备注
post_id	int	外码
subject	varchar	与post_id一同组成主属性

```
# Post表的科目表 BCNF
class PostSubject(models.Model):
    # 主码、主属性
    post_id = models.ForeignKey(Post,
on_delete=models.CASCADE,related_name='post_id_subject')
    subject = models.CharField(max_length=255,blank=True,null=True)

    def __str__(self):
        return self.subject
```

2.4 其他事务部分

这部分内容有关其他事务的管理，主要包括通知、学习资料、评价、待办事项、管理员公告等功能。

Notification

属性名	数据类型	备注
notification_id	int	主码、主属性
user_id	int	外码
notification_date	date	-
title	varchar	-
description	text	-
is_read	boolean	-

```
# 通知表 BCNF
class Notification(models.Model):
    # 主码、主属性
    notification_id = models.BigAutoField(primary_key=True)
    # 外码
    user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_notice')
    notificationDate = models.DateTimeField(blank=True,null=True)
    title = models.CharField(max_length=255,blank=True,null=True)
    description = models.TextField(blank=True,null=True)
    is_read = models.BooleanField()

    def __str__(self):
```

```
        return self.title

    class Meta:
        ordering = ['-notificationDate']
        indexes = [
            models.Index(fields=['notification_date']),
        ]
```

StudyMaterial

属性名	数据类型	备注
material_id	int	主码、主属性
tutor_id	int	外码
student_id	int	外码
file_name	varchar	-
download_link	varchar	-
upload_ate	date	-

```
# 学习资料表 BCNF
class StudyMaterial(models.Model):
    # 主码、主属性
    material_id = models.BigAutoField(primary_key=True)
    # 外码
    tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_material')
    # 外码
    student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_material')
    file_name = models.CharField(max_length=255)
    download_link = models.CharField(max_length=255)
    upload_date = models.DateField()

    def __str__(self):
        return f"Material {self.material_id}"

    class Meta:
        ordering = ['-upload_date']
        indexes = [
            models.Index(fields=['upload_date']),
        ]
```

Review

属性名	数据类型	备注
review_id	int	主码、主属性
student_id	int	外码

属性名	数据类型	备注
tutor_id	int	外码
rating	int	-
content	text	-
date	date	-

```
# 评价表 BCNF
class Review(models.Model):
    # 主码、主属性
    review_id = models.BigAutoField(primary_key=True)
    # 外码
    student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_review')
    # 外码
    tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_review')
    rating = models.FloatField(blank=True,null=True)
    content = models.TextField(blank=True,null=True)
    date= models.DateField(blank=True,null=True)

    def __str__(self):
        return f"Review {self.review_id}"

    class Meta:
        ordering = ['-date']
        indexes = [
            models.Index(fields=['date']),
        ]
```

Todo

属性名	数据类型	备注
todo_id	int	主码、主属性
owner_id	int	外码
accepter_id	int	外码
accepter_post_id	int	外码
is_completed	int	-

```
# 待办事项表 BCNF
class Todo(models.Model):
    # 主码、主属性
    todo_id = models.BigAutoField(primary_key=True)
    # 外码
    owner_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='owner_id_todo')
    accepter_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='sender_id_todo')
    accept_post_id = models.ForeignKey(Post,
on_delete=models.CASCADE,related_name='source_post_id_todo')
    is_completed = models.BooleanField()

    def __str__(self):
        return f"Todo {self.todo_id}"
```

Announcement

属性名	数据类型	备注
announcement_id	int	主码、主属性
announcement_date	date	-
title	varchar	-
description	text	-

```
class Announcement(models.Model):
    # 主码、主属性
    announcement_id = models.BigAutoField(primary_key=True)
    announcement_date = models.DateTimeField(blank=True,null=True)
    title = models.CharField(max_length=255,blank=True,null=True)
    description = models.TextField(blank=True,null=True)

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['-announcement_date']
        indexes = [
            models.Index(fields=['announcement_date']),
        ]
```

3.7 存储过程

3.7.1 用户管理（注册/登录）

1. 用户注册

- **涉及的基本表**：`User`，`Student`，`Tutor`

- **过程描述**：检查用户是否被注册。如果未被注册，则将用户信息插入到User表中，再根据身份信息插入到Student或Tutor表中，并给用户发送一条注册成功的通知。值得注意的是，管理员账号只能由超级管理员手动添加，因此不会出现管理员注册的情况。

```
@csrf_exempt
def register(request):
    if request.method == 'POST':
        body = json.loads(request.body)
        username = body.get('name')
        password = body.get('password')
        role = body.get('role')
        try:
            # 检查用户是否已经存在
            user = User.objects.get(username=username)
            result={'data':{}}
            result['code'] = -1
            return JsonResponse(result)
        except User.DoesNotExist:
            # 创建用户
            user = User(username=username,
                        password=password,
                        identity=-1,

 registration_date=datetime.now(pytz.timezone('Asia/Shanghai')))
            result={'data':{}}
            if role == "admin":
                raise Exception("管理员账号只能由超级管理员手动添加")
            elif role == "teacher":
                user.identity = 1
                user.save()
                tutor=Tutor(user_id=user)
                tutor.save()
                result['data']['roles']=[{'id': 'teacher'}]
            elif role == "student":
                user.identity = 2
                user.save()
                student=Student(user_id=user)
                student.save()
                result['data']['roles']=[{'id': 'student'}]
            else:
                raise Exception("未知身份")

            request.session['user_id'] = user.user_id
            sendNotice(user,"注册成功","欢迎加入家教综合服务平台！")

            result['code'] = 0
            result['data']['token']="Authorization:" + str(random.random())
            result['data']['id'] = str(user.user_id)
            return JsonResponse(result)
    else:
        return JsonResponse({'code': -1, 'message': '仅支持POST请求'})
```

2. 用户登录

- **涉及的基本表**：`User`

- **过程描述**：检查用户是否存在，如果存在则返回用户的身份信息，否则返回错误信息。值得注意的是，后续有相关业务需要知道当前用户的身份，因此在登录成功后将用户的user_id存入session中。

```
@csrf_exempt
def login(request):
    if request.method == 'POST':
        body = json.loads(request.body)
        username = body.get('name')
        password = body.get('password')
        result = {'data': {}}
        try:
            # 查询数据库中是否存在对应的用户
            user = User.objects.get(username=username, password=password)
            if user.identity == 0:
                result['data']['roles'] = [{'id': 'admin'}]
            elif user.identity == 1:
                result['data']['roles'] = [{'id': 'teacher'}]
            elif user.identity == 2:
                result['data']['roles'] = [{'id': 'student'}]
            else:
                raise Exception("未知身份")
            result['data']['id'] = str(user.user_id)

            request.session['user_id'] = user.user_id

            # 登录成功
            result['code'] = 0
            result['data']['token'] = "Authorization:" + str(random.random())
            result['message'] = "登录成功"
        except User.DoesNotExist:
            # 用户不存在或密码错误
            result['code'] = -1
            result['message'] = "账户名或密码错误"
        return JsonResponse(result)
    else:
        return JsonResponse({'code': -1, 'message': '仅支持POST请求'})
```

3.7.2 帖子管理

1. 发帖

- **涉及的基本表**：`Post`，`PostSubject`
- **过程描述**：检查帖子的内容是否完整，如果不完整则返回错误信息。如果内容完整，则将帖子信息插入到Post表中，再将帖子的科目信息插入到PostSubject表中。

```
@csrf_exempt
def sendPost(request):
    if request.method == 'POST':
        body = json.loads(request.body)
        request_id=int(body.get('id'))
        title = body.get('data').get('title')
        startDate = body.get('data').get('startDate')
        endDate = body.get('data').get('endDate')
```

```
        subjects=body.get('data').get('subjects')
        location=body.get('data').get('location')
        fullLocation=body.get('data').get('fullLocation')
        telephoneNumber=body.get('data').get('telephoneNumber')
        email=body.get('data').get('emailAddress')
        content=body.get('data').get('content')
        is_complete = all([title, startDate, endDate, subjects, location,
fullLocation, telephoneNumber, content])
        if not is_complete:
            return JsonResponse({'code': 0})

        try :
            user=User.objects.get(user_id=request_id)
        except User.DoesNotExist:
            return JsonResponse({'code': -1, 'message': '用户不存在'})

        if user.identity ==0:
            raise Exception("管理员不通过sendPost发帖")
        else:
            post=Post(
                user_id=user,
                title=title,
                postDate=datetime.now(pytz.timezone('Asia/Shanghai')),
                startDate=startDate,
                endDate=endDate,
                location=location,
                fullLocation=fullLocation,
                telephoneNumber=telephoneNumber,
                emailAddress=email,
                content=content,
                is_completed=True,
                is_approved=False,
            )
            post.save()

            postSubjects = [
                PostSubject(post_id=post, subject=subject)
                for subject in subjects
            ]
            PostSubject.objects.bulk_create(postSubjects)

            result={'data':{}}
            result['code'] = 0
            return JsonResponse(result)
    else:
        return JsonResponse({'code': -1, 'message': '仅支持POST请求'})
```

2. 保存模板

- **涉及的基本表**：`Post`，`PostSubject`
- **过程描述**：将帖子内容保存为模板，如果用户已经有模板，则更新模板内容，否则新建一个模板。模板与帖子用is_completed字段区分。

```
@csrf_exempt
def savePost(request):
```

```
    if request.method == 'POST':
        body = json.loads(request.body)
        request_id=int(body.get('id'))
        title = body.get('data').get('title')
        startDate = body.get('data').get('startDate') or None
        endDate = body.get('data').get('endDate') or None
        subjects=body.get('data').get('subjects')
        location=body.get('data').get('location')
        fullLocation=body.get('data').get('fullLocation')
        telephoneNumber=body.get('data').get('telephoneNumber')
        email=body.get('data').get('emailAddress') or None
        content=body.get('data').get('content')

        try:
            user=User.objects.get(user_id=request_id)
        except User.DoesNotExist:
            return JsonResponse({'code': -1, 'message': '用户不存在'})

        if user.identity ==0:
            raise Exception("管理员无法保存帖子")
        else:
            try:
                post=Post.objects.get(user_id=user,is_completed=False)
                post.title=title
                post.startDate=startDate
                post.endDate=endDate
                post.location=location
                post.fullLocation=fullLocation
                post.telephoneNumber=telephoneNumber
                post.emailAddress=email
                post.content=content
                post.save()
                PostSubject.objects.filter(post_id=post).delete()
                postSubjects = [
                    PostSubject(post_id=post, subject=subject)
                    for subject in subjects
                ]
                PostSubject.objects.bulk_create(postSubjects)
                result={'data':{}}
                result['code'] = 0
                return JsonResponse(result)
            except Post.DoesNotExist:
                post=Post(
                    user_id=user,
                    title=title,
                    startDate=startDate,
                    endDate=endDate,
                    location=location,
                    fullLocation=fullLocation,
                    telephoneNumber=telephoneNumber,
                    emailAddress=email,
                    content=content,
                    is_completed=False,
                    is_approved=False,
                )
                post.save()
```

```
                postSubjects = [
                    PostSubject(post_id=post, subject=subject)
                    for subject in subjects
                ]
                PostSubject.objects.bulk_create(postSubjects)
                result={'data':{}}
                result['code'] = 0
                return JsonResponse(result)
    else:
        return JsonResponse({'code': -1, 'message': '仅支持POST请求'})
```

3. 获取模板

- **涉及的基本表**：`Post`，`PostSubject`
- **过程描述**：获取用户的模板，如果用户没有模板，则返回空数据。

```
@csrf_exempt
def getSavedPost(request):
    if request.method == 'GET':
        try:
            user_id=request.GET.get('id')
            user=User.objects.get(user_id=int(user_id))
            if user.identity==0:
                raise Exception("管理员无法获取帖子")
            else:
                try:
                    post=Post.objects.get(user_id=user,is_completed=False)
                    result={'data':{}}
                    result['code'] = 0
                    result['data']['title']=post.title
                    result['data']['startDate']=post.startDate
                    result['data']['endDate']=post.endDate
                    location=post.location.strip('"')
                    location=json.loads(location.replace("'",'"'))
                    result['data']['location']=location
                    result['data']['fullLocation']=post.fullLocation
                    result['data']['telephoneNumber']=post.telephoneNumber
                    result['data']['emailAddress']=post.emailAddress
                    result['data']['content']=post.content
                    subjects=PostSubject.objects.filter(post_id=post)
                    result['data']['subjects']=[subject.subject for subject in
subjects]
                except:
                    result={'data':{}}
                    result['code'] = 0
                return JsonResponse(result)
        except User.DoesNotExist:
            result={'data':{}}
            result['code'] = 0
            return JsonResponse(result)
    else:
        return JsonResponse({'code': -1, 'message': '仅支持GET请求'})
```

4. 在帖子广场上获取帖子

- **涉及的基本表**：`Post`，`PostSubject`
- **过程描述**：获取帖子广场上的帖子，并根据搜索关键词以及用户身份进行筛选。具体来说，首先学生只会看到家教发的帖子，家教会看到学生发的帖子，管理员会看到所有尚未审核的帖子。然后根据搜索关键词进行筛选，匹配字段包含科目、用户名、标题、内容。如果搜索关键词为空，则返回所有符合条件的帖子。

```
@csrf_exempt
def getPosts(request):
    if request.method == 'GET':
        request_id = int(request.GET.get('id'))
        request_page = int(request.GET.get('page'))
        request_query = request.GET.get('query')
        start = (request_page - 1) * 10
        end = request_page * 10
        returnSubjects = []

        try:
            user = User.objects.get(user_id=request_id)
            filter_id = 2 if user.identity == 1 else 1

            if request_query == "":
                if user.identity != 0:
                    posts = Post.objects.filter(is_completed=True,
user_id__identity=filter_id, is_approved=True)
                else:
                    posts = Post.objects.filter(is_completed=True,
is_approved=False)
            else:
                subjectsFits =
PostSubject.objects.filter(subject=request_query).values_list('post_id',
flat=True)
                userFits =
User.objects.filter(username=request_query).values_list('id', flat=True)
                OtherFits = Post.objects.filter(
                    models.Q(title__icontains=request_query) |
                    models.Q(content__icontains=request_query)
                ).values_list('id', flat=True)

                all_posts = Post.objects.all()

                allFits = set(subjectsFits) | set(userFits) | set(OtherFits)
                if user.identity != 0:
                    posts = Post.objects.filter(id__in=allFits,
is_completed=True, user_id__identity=filter_id, is_approved=True)
                else:
                    posts = Post.objects.filter(id__in=allFits,
is_completed=True, is_approved=False)

            post_ids = posts.values_list('id', flat=True)
            post_subjects =
PostSubject.objects.filter(post_id__in=post_ids).values('post_id', 'subject')
            post_subject_map = {}
            for ps in post_subjects:
```

```
                post_subject_map.setdefault(ps['post_id'],
[]).append(ps['subject'])

            return_posts = []
            now = datetime.now(pytz.timezone('Asia/Shanghai'))
            for post in posts:
                post_date =
post.postDate.astimezone(pytz.timezone('Asia/Shanghai'))
                time_diff = now - post_date
                if time_diff < timedelta(hours=24):
                    date_display = f"{time_diff.seconds // 3600}小时前"
                elif time_diff < timedelta(days=3):
                    date_display = f"{time_diff.days}天前"
                else:
                    date_display = post_date.strftime("%Y-%m-%d")

                location = post.location.strip('"')
                location = json.loads(location.replace("'", '"'))
                location = location[0] if isinstance(location, list) and location
else location

                return_posts.append({
                    'id': str(post.post_id),
                    'title': post.title,
                    'tags': post_subject_map.get(post.id, []),
                    'content': post.content,
                    'author': post.user_id.username,
                    'authorId': post.user_id.user_id,
                    'date': date_display,
                    'location': location,
                })

            result = {'posts': return_posts[start:end], 'total': len(posts)}
            return JsonResponse(result)
        except User.DoesNotExist:
            return JsonResponse({'code': -1, 'message': '用户不存在'})
    else:
        return JsonResponse({'code': -1, 'message': '仅支持GET请求'})
```