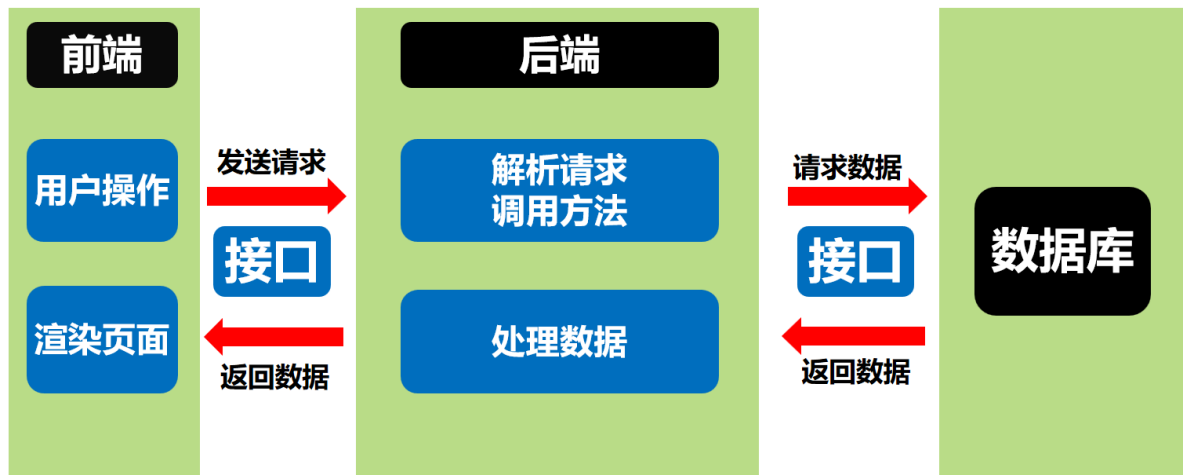


一、系统结构设计

1.1 体系结构

1.1.1 总体结构

本项目采用前后端分离的体系结构，前后端通过api来交换数据。前端通过api向后端请求数据，后端控制器负责根据前端的不同请求，来对数据库进行存取操作，进而将数据返回给前端。前端得到数据后将数据渲染在网页上，进而实现前后端的交互，如下图所示。



1.1.2 前端结构

assets:

用于存储静态资源，例如网站logo、图片、字体。

components:

包含多次重复使用的或功能较为独立的Vue组件，例如固定按钮、侧边栏、导航栏、密码修改表单等。

router:

存放控制应用路由的JS文件，具体路由配置可参考config.js。

page:

存放应用界面的Vue文件，每个子文件夹对应一个具体功能页面。

services:

存放api相关JS文件，用于与后端交互。

mock:

存放模拟数据和服务器响应，扮演一个临时的、轻量级的服务端角色，帮助我们在没有真实后端API或是在后端API尚未完成的情况下进行前端开发和测试。

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，具体内容见本文档第二部分。

- **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**

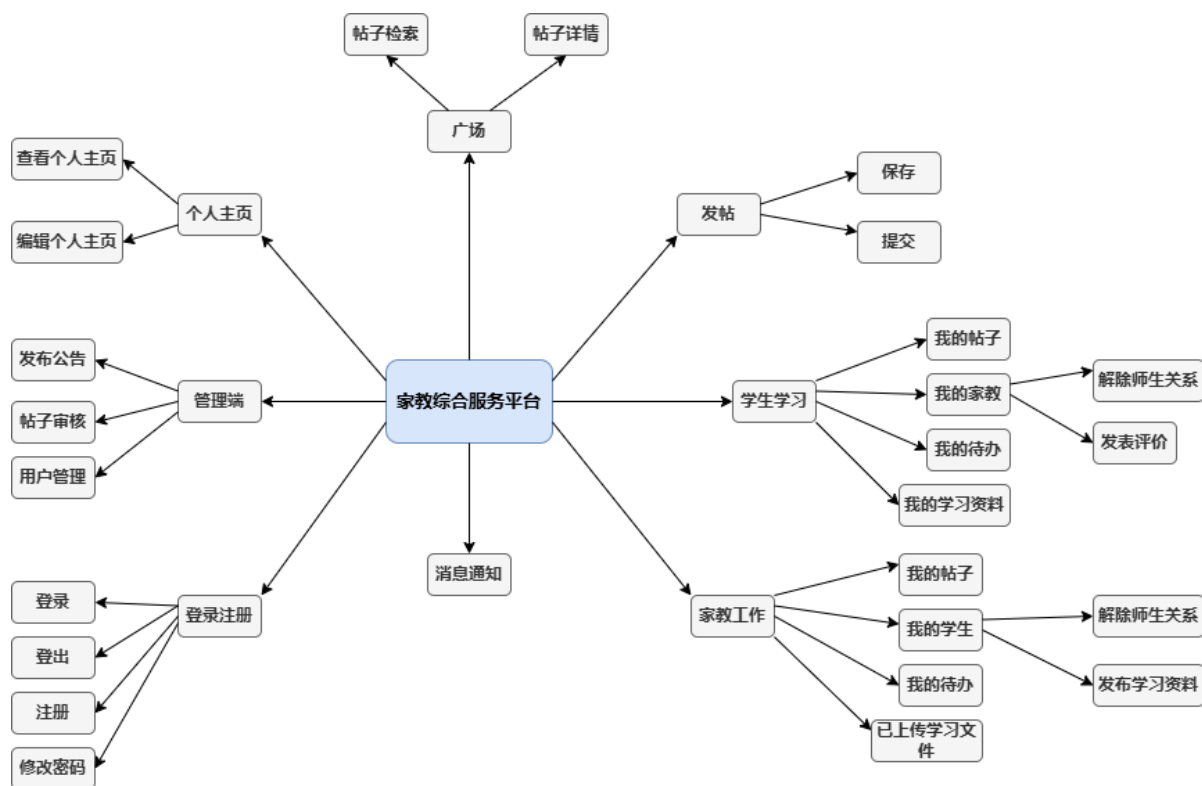
后端基本文件，用于进行一些配置，主要有Django框架生成。

- **manage.py**

脚本，用于后端的部署、数据库的迁移等。

1.2 功能结构

功能架构示意图如下：



1.2.1 登录注册

- 登录：可以通过输入用户名和对应的正确密码，登录进入功能页面，否则无法进入功能页面。
- 注册：可通过输入用户名、密码，选择家教或老师的身份，当后台查验无误的时候，即可注册账户。
- 登出：用户登录后可以选择登出，登出后用户失去访问功能页面的权限。
- 修改密码：在登陆后可以选择修改密码，用户的密码会被设置成新的密码。

1.2.2 个人主页

- 查看个人主页：学生和家教可以查看自己的主页，家教主页会额外显示自己的学生对自己的评价。
- 编辑个人主页：用户可以随时编辑自己的主页，修改例如年龄、邮箱、联系方式等个人信息。

1.2.3 管理端

- 发布公告：管理员可以对所有用户发布公告，用户可以在公告页面查看。
- 帖子审核：学生和家教发布的帖子需要被管理员审核通过后才会在广场上显示，管理员可驳回待通过的帖子。
- 用户管理：管理员可以删除任意用户，将其加入黑名单。

1.2.4 广场

- 帖子检索：广场提供搜索功能，并支持基本的模糊搜索。
- 帖子详情：在招聘帖中，家教可以选择接受招聘；在求聘帖中，学生可以选择接收求聘。

1.2.5 发帖

学生（家教）用户可以通过在侧边栏选择我要招聘（我要求聘）进入发帖页面，在这个页面用户可以填写自己的基本信息以及需求。

- 保存：用户可以对已填写内容进行保存。
- 提交：用户可以提交帖子，提交后的帖子经管理员审核通过后便会出现在广场界面。

1.2.6 消息通知

当发生“帖子通过审核”“有新的公告”“家教发布学习资料”“请求建立师生关系”“成功建立师生关系”等多种情况时都会产生通知信息，以使用户及时获取重要信息。

1.2.7 学生学习

- 学生可以查看自己发布过的帖子，可以选择删除之前发布的帖子。
- 学生可以查看自己的家教信息，选择与其解除师生关系，对家教发表评价。
- 学生可以浏览自己的待办事项，并选择接受或者拒绝。
- 学生可以查看家教给自己上传的学习资料。

1.2.8 家教工作

- 家教可以查看自己发布过的帖子，可以选择删除之前发布的帖子。
- 家教可以查看自己的学生信息，选择与其解除师生关系，给学生上传学习资料。
- 家教可以浏览自己的待办事项，并选择接受或者拒绝。
- 家教可以查看自己已上传的学习资料。

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。

属性名	数据类型	备注
user_id	int	主码、主属性
username	char	主属性
password	char	-
identity	int	-
registration_date	date	-
avatar	char	-

```
1  # 用户表 BCNF
2  class User(models.Model):
3      # 主码、主属性
4      user_id = models.BigAutoField(primary_key=True)
5      # 主属性
6      username = models.CharField(max_length=255)
7      password = models.CharField(max_length=255)
8      identity = models.IntegerField()
9      registration_date = models.DateField()
10     avatar = models.CharField(max_length=255, blank=True, null=True)
11
12     def __str__(self):
13         return self.username
14
15     class Meta:
16         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

属性名	数据类型	备注
email	email	主属性
grade	int	-

```

1  # 学生表 BCNF
2  class Student(models.Model):
3      # 主码、主属性
4      user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_student')
5      age = models.IntegerField(blank=True,null=True)
6      gender = models.CharField(max_length=255,blank=True,null=True)
7      # 主属性
8      telephone = models.CharField(max_length=255,null=True,blank=True)
9      # 主属性
10     email = models.EmailField(max_length=255,blank=True,null=True)
11     grade = models.CharField(max_length=255,blank=True,null=True)
12     address = models.CharField(max_length=255,blank=True,null=True)
13     intro= models.TextField(blank=True,null=True)
14     personalSignature =
models.CharField(max_length=255,blank=True,null=True)
15
16
17     def __str__(self):
18         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	-

```

1  # 家教表 BCNF
2  class Tutor(models.Model):
3      # 主码、主属性
4      user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_tutor')
5      age = models.IntegerField(blank=True,null=True)
6      gender = models.CharField(max_length=255,blank=True,null=True)
7      # 主属性
8      telephone = models.CharField(max_length=255,blank=True,null=True)
9      # 主属性
10     email = models.EmailField(max_length=255,blank=True,null=True)

```

```

11     rate = models.FloatField(default=0)
12     rateNum = models.IntegerField(default=0)
13     address = models.CharField(max_length=255,blank=True,null=True)
14     intro= models.TextField(blank=True,null=True)
15     personalSignature =
models.CharField(max_length=255,blank=True,null=True)
16     degree = models.CharField(max_length=255,blank=True,null=True)
17
18     def __str__(self):
19         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

```

1  # 师生关系表 BCNF
2  class Link(models.Model):
3      # 主码、主属性
4      link_id = models.BigAutoField(primary_key=True)
5      # 外码
6      student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_link')
7      # 外码
8      tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_link')
9
10     def __str__(self):
11         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	-

```
1  # Post表 BCNF
2  class Post(models.Model):
3      # 主码、主属性
4      post_id = models.BigAutoField(primary_key=True)
5      # 外码
6      user_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='user_id_post')
7      title = models.CharField(max_length=255,blank=True,null=True)
8      post_date = models.DateTimeField(blank=True,null=True)
9      start_date = models.DateField(blank=True,null=True)
10     end_date = models.DateField(blank=True,null=True)
11     content = models.TextField(blank=True,null=True)
12     is_completed = models.BooleanField()
13     is_approved = models.BooleanField(default=False)
14
15     def __str__(self):
16         return self.title
17
18     class Meta:
19         indexes = [
20             models.Index(fields=['post_id']),
21             models.Index(fields=['title']),
22         ]
```

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

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

```

1 # Post表的科目表 BCNF
2 class PostSubject(models.Model):
3     # 主码、主属性
4     post_id = models.ForeignKey(Post,
5     on_delete=models.CASCADE,related_name='post_id_subject')
6     subject = models.CharField(max_length=255,blank=True,null=True)
7
8     def __str__(self):
9         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	-

```

1 # 通知表 BCNF
2 class Notification(models.Model):
3     # 主码、主属性
4     notification_id = models.BigAutoField(primary_key=True)
5     # 外码
6     user_id = models.ForeignKey(User,
7     on_delete=models.CASCADE,related_name='user_id_notice')
8     notificationDate = models.DateTimeField(blank=True,null=True)
9     title = models.CharField(max_length=255,blank=True,null=True)
10    description = models.TextField(blank=True,null=True)
11    is_read = models.BooleanField()
12
13    def __str__(self):
14        return self.title
15
16    class Meta:
17        ordering = ['-notificationDate']
18        indexes = [
19            models.Index(fields=['notification_date']),
20        ]

```


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	-

```
1  # 学习资料表 BCNF
2  class StudyMaterial(models.Model):
3      # 主码、主属性
4      material_id = models.BigAutoField(primary_key=True)
5      # 外码
6      tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_material')
7      # 外码
8      student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_material')
9      file_name = models.CharField(max_length=255)
10     download_link = models.CharField(max_length=255)
11     upload_date = models.DateField()
12
13     def __str__(self):
14         return f"Material {self.material_id}"
15
16     class Meta:
17         ordering = ['-upload_date']
18         indexes = [
19             models.Index(fields=['upload_date']),
20         ]
```

Review

属性名	数据类型	备注
review_id	int	主码、主属性
student_id	int	外码
tutor_id	int	外码
rating	int	-
content	text	-
date	date	-

```

1  # 评价表 BCNF
2  class Review(models.Model):
3      # 主码、主属性
4      review_id = models.BigAutoField(primary_key=True)
5      # 外码
6      student_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='student_id_review')
7      # 外码
8      tutor_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='tutor_id_review')
9      rating = models.FloatField(blank=True,null=True)
10     content = models.TextField(blank=True,null=True)
11     date= models.DateField(blank=True,null=True)
12
13     def __str__(self):
14         return f"Review {self.review_id}"
15
16     class Meta:
17         ordering = ['-date']
18         indexes = [
19             models.Index(fields=['date']),
20         ]

```

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	-

```

1  # 待办事项表 BCNF
2  class Todo(models.Model):
3      # 主码、主属性
4      todo_id = models.BigAutoField(primary_key=True)
5      # 外码
6      owner_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='owner_id_todo')
7      accepter_id = models.ForeignKey(User,
on_delete=models.CASCADE,related_name='sender_id_todo')
8      accept_post_id = models.ForeignKey(Post,
on_delete=models.CASCADE,related_name='source_post_id_todo')
9      is_completed = models.BooleanField()
10
11     def __str__(self):
12         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	-

```
1 class Announcement(models.Model):
2     # 主码、主属性
3     announcement_id = models.BigAutoField(primary_key=True)
4     announcement_date = models.DateTimeField(blank=True,null=True)
5     title = models.CharField(max_length=255,blank=True,null=True)
6     description = models.TextField(blank=True,null=True)
7
8     def __str__(self):
9         return self.title
10
11     class Meta:
12         ordering = ['-announcement_date']
13         indexes = [
14             models.Index(fields=['announcement_date']),
15         ]
```

三、系统重要功能实现

3.1 用户鉴权

我们参照现代网页前端设计，实现了token鉴权机制。当用户尝试进行登录时，服务端会鉴别用户身份并返回一个token。客户端会将这个token保存在cookie中。在登录成功后用户每次向服务端发出请求时都将附上这个token。服务端可以依据校验该token，判断用户是否成功登录，并进一步获取用户的具体身份。用户鉴权的主要实现逻辑位于 `frontend/src/utlis/request.js`。

```
1 // 用户登录成功后，调用该函数保存用户认证信息：保存用户token，以及设置token过期时间
2 function setAuthorization(auth) {
3     Cookie.set(xsrfHeaderName, 'Bearer ' + auth.token, {expires:
4 auth.expireAt})
5 }
6 // 用户退出登录后，调用该函数移除用户认证信息
7 function removeAuthorization() {
8     Cookie.remove(xsrfHeaderName)
9 }
```

3.2 路由守卫

为了防止未获取权限的用户直接通过网址跳转到需要对应权限的页面，我们设计了全局路由守卫。即在用户进入每个页面之前都对用户的认证信息进行检测：如果检测通过，则跳转到对应页面；如果检测不通过，则跳转到错误处理页面。路由守卫主要处理逻辑位于 `frontend/src/router/guards.js` 中。

我们会在页面路由设置文件(`frontend/src/touter/config.js`)中规定进入相应页面所需的权限，如下所示：

```
1  {
2    path: 'announcementPage',    // 这是公告界面，所有登录用户均可查看
3    name: '公告',
4    meta: {
5      icon: 'dashboard',
6      page: {
7        closable: false
8      },
9    },
10   component: () => import('@pages/announcementPage'),
11 },
12 {
13   path: 'jobSeekPage', // 这是针对教师的求职页面
14   name: "广场",
15   meta: {
16     icon: "team",
17     authority: {
18       role: 'teacher',    // 设置该页面只有role为'teacher'的用户才能进入
19     }
20   },
21   component: () => import('@pages/jobSeekPage')
22 },
23 {
24   path : "hirePage", // 这是针对学生的招聘页面
25   name: "广场",
26   meta: {
27     icon: "team",
28     authority: {
29       role: 'student'    // 设置该页面只有role为'student'的用户才能进入
30     }
31   },
32   component: () => import('@pages/hirePage')
33 }
```

当用户尝试进入某个页面时，首先会经过登录守卫的检测

```
1  const loginGuard = (to, from, next, options) => {
2    const {message} = options
3    if (!loginIgnore.includes(to) && !checkAuthorization()) {
4      message.warning('登录已失效，请重新登录')    // 鉴权失败，重定向到登录界面
5      next({path: '/login'})
6    } else {
7      next()    // 鉴权成功，跳转到目标界面
8    }
9  }
```

用户通过登录守卫验证，表示用户已经成功登录。之后用户还需要经过权限守卫的检测

```
1  const authorityGuard = (to, from, next, options) => {
2    const {store, message} = options
3    const roles = store.getters['account/roles']
4    if (!hasAuthority(to, roles)) {
5      message.warning(`对不起，您无权访问页面：${to.fullPath}，请联系管理员`) // 鉴
      权失败，跳转到错误处理页面
6      next({path: '/403'})
7      // NProgress.done()
8    } else {
9      next() // 鉴权成功，跳转到目标页面
10   }
11 }
12
13 // 检测用户是否具有role权限
14 function hasAuthority(route, roles) {
15   const authorities = [...route.meta.pAuthorities, route.meta.authority]
16   for (let authority of authorities) {
17     if (!hasRole(authority, roles)) {
18       return false
19     }
20   }
21   return true
22 }
23
24 // 检测用户的roles列表中是否含有页面所需的role权限
25 function hasRole(authority, roles) {
26   let required = undefined
27   if (typeof authority === 'object') {
28     required = authority.role
29   }
30   return hasAnyItem(required, roles, (r, t) => !(r === t || r === t.id))
31 }
```

用户通过权限守卫，表示用户拥有目标页面所需的权限。

用户通过登录守卫与权限守卫后，即可进入目标页面。

3.3 动态路由匹配

我们有时需要把某种模式匹配到的所有路由全都映射到同样的组件中。例如**学生主页**页面，**家教主页**页面，**帖子详情**页面，这时我们可以采用动态路由匹配：一组页面的组件设计相同，客户端通过具体指定的参数来决定具体跳转的目标页面。如下所示：

```
1  {
2    path: 'user/studentHomePage/:id',
3    name: '学生主页',
4    meta: {
5      invisible: true
6    },
7    component: () => import('@/pages/studentHomePage')
8  }
```

在“学生主页”页面中，我们使用了动态路由参数，该页面的具体路径将由 `$route.params.id` 决定。

当我们需要跳转到 `id` 为 `22373090` 的页面时，可以这么编写代码

```
1 | this.$router.push({ name: '学生主页', params: { id: "22373090" } })
```

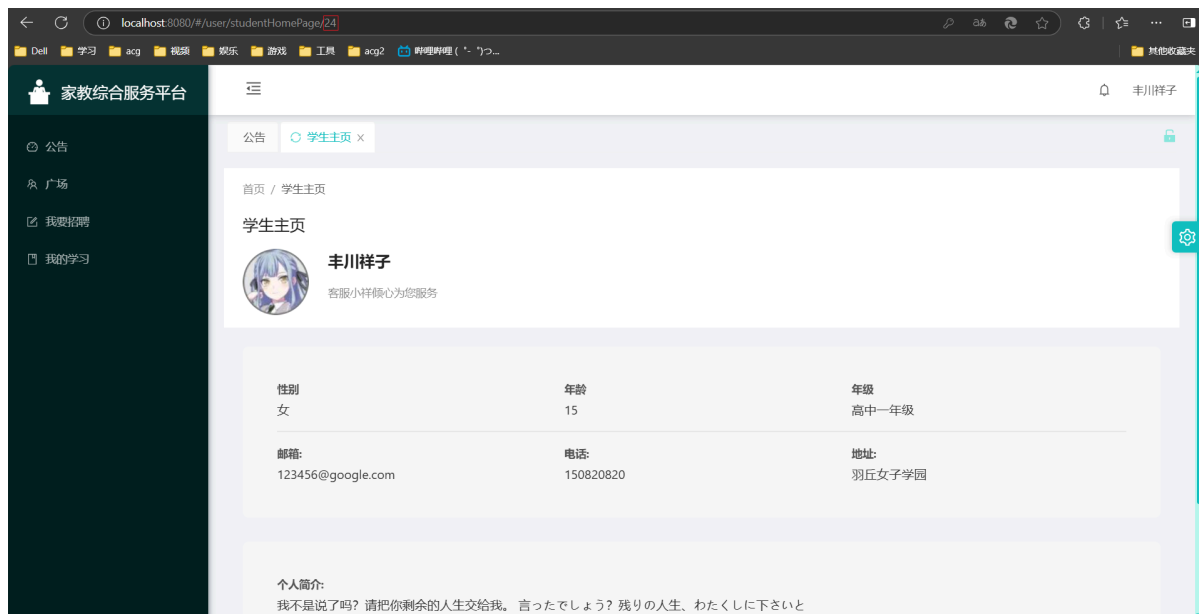
动态路由参数通过 `route` 直接获取

```
1 | userId: this.$route.params.id // 获取路由参数中的id
```

页面的具体信息可以通过 `api` 从服务端调取

```
1 | created() { // 创建“学生主页”页面时，调用fetchUserInfo获取用户信息
2 |   this.fetchUserInfo();
3 | },
4 | methods: {
5 |   fetchUserInfo() {
6 |     // 调用 getUserInfo api, 获取用户信息
7 |     getStudentInfo(this.userId)
8 |       .then(response => {
9 |         // ...
10 |      })
11 |     .catch(error => {
12 |       // ...
13 |     });
14 |   }
15 | }
```

具体实现效果如下



3.4 文件上传

在我们使用的Vue2中，文件需要包装在 `FormData` 对象的 `file` 段中，才能与后端进行通信。如果此外还有其它数据，可以以 `json` 的形式包装在 `FormData` 对象的 `data` 段中，如下所示

```
1 | submitLearningMaterial() {
2 |   const formData = new FormData();
```

```

3      formData.append('file', this.fileList[0]);    // 将文件添加到formData的file
   段
4      const jsonData = {
5          id: this.currUser.id,
6          teacher: this.currUser.name,
7          studentId: this.curStudentId,
8      };
9      formData.append('data', JSON.stringify(jsonData));    // 将其他数据以json形式
   添加到data段中
10
11      submitLearningMaterial(formData).then(res => {    // 通过api与后端进行通信
12          // ...
13      }).catch(error => {
14          // ...
15      });
16  }

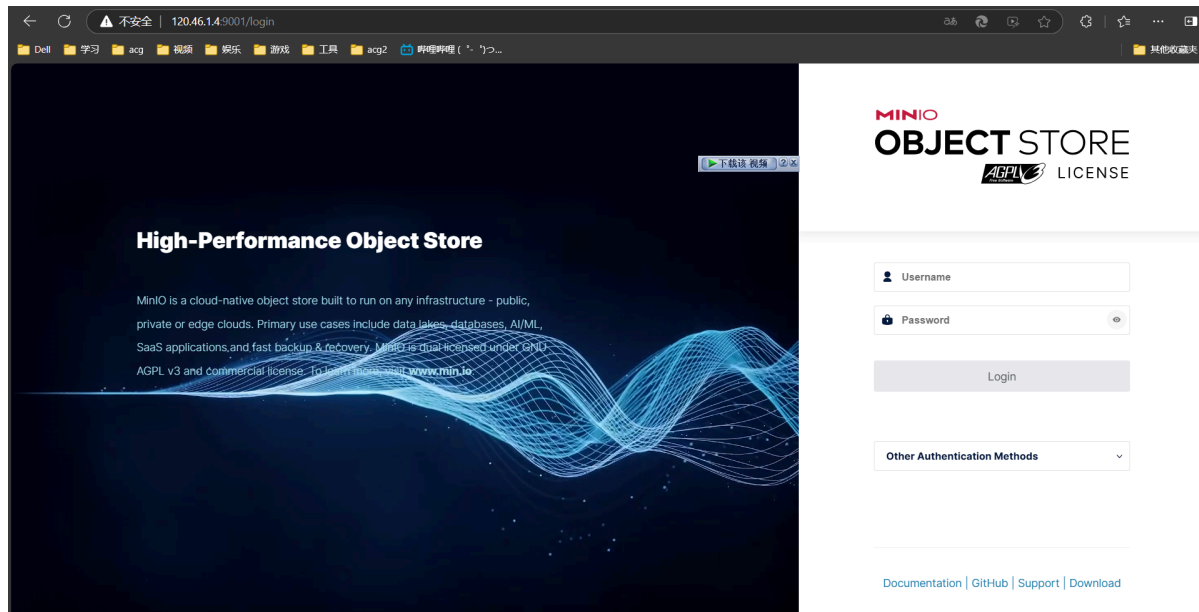
```

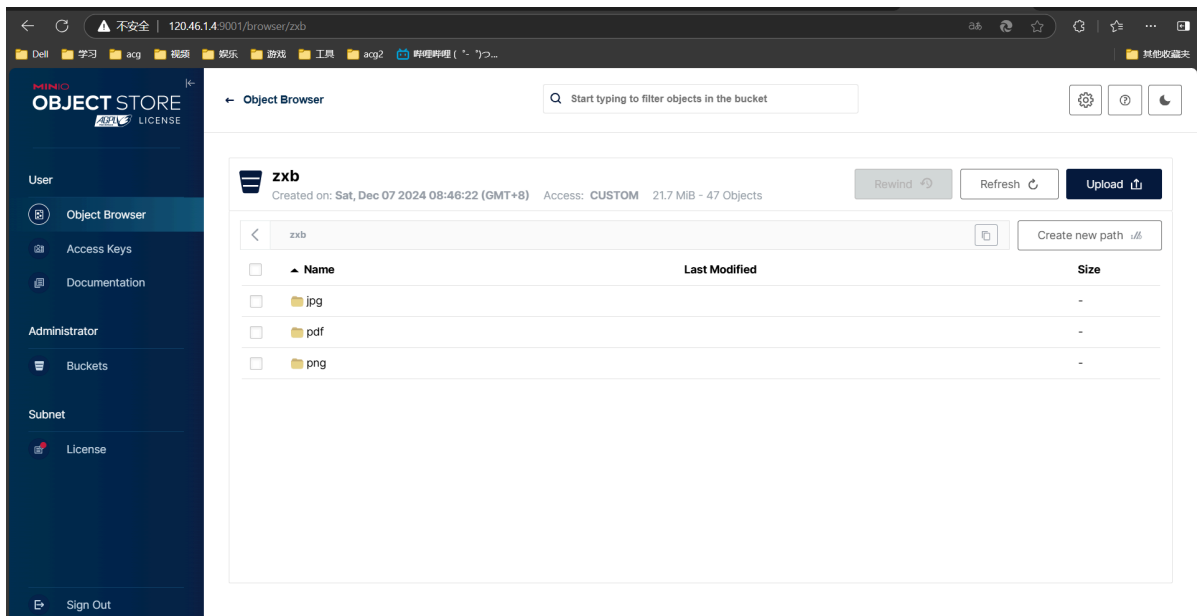
3.5 文件下载

为了实现对用户上传的头像/文件进行云端存储，我们使用了MinIO进行对象存储。

MinIO是一个对象存储解决方案，它提供了与Amazon Web Services S3兼容的API，并支持所有核心S3功能。使用文档参照[MinIO Windows中文文档](#)

我们从华为云租借了服务器（120.46.1.4），并将MinIO部署在该服务器上。实现效果如下





我们还对上传文件以及下载文件的接口进行了包装，使其使用起来更加方便

```
1  from minio import Minio
2
3  minio_url = "120.46.1.4:9000"
4  minio_access_key = "*****"
5  minio_secret_key = "*****"
6
7  def get_download_url(file_name, type) -> str:
8      """
9      获取文件名为file_name文件的下载链接
10     @param file_name: 文件名称
11     @param type: 文件类型
12     """
13     return "http://" + minio_url + "/zxb/" + type + "/" + file_name
14
15
16 class MinioClient:
17     def __init__(self):
18         self.client = Minio(endpoint=minio_url, access_key=minio_access_key,
19                             secret_key=minio_secret_key, secure=False)
20
21     def upload_file(self, data, file_name, type)
22     """
23     上传文件
24     @param data: 文件的二进制流数据
25     @param file_name: 文件存储在云端的名称
26     @param type: 文件类型
27     """
```

我们可以调用 `upload_file` 上传文件，调用 `get_download_url` 获取文件的下载链接，使用示例如下

```
1  def submitLearningMaterial(request):
2      # ...
3      file = request.FILES['file'] # 获取上传的文件
4      file_name = file.name
5      file_type = file_name.split('.')[0]
6
```



```

7         unique_file_name = f"{uuid.uuid4()}.{file_type}" # 使用uuid生成唯一文件名，
          避免文件名重复发生覆盖
8
9         minio = MinioClient()
10        file_data = file.read() # 获取文件的二进制数据
11        minio.upload_file(file_data, unique_file_name, file_type) # 上传文件
12        download_link = get_download_url(unique_file_name, file_type) # 获取下载链
          接
13
14        # ...

```

我们通过这种设计，成功实现了“学习资料”以及“用户头像”相关功能。

3.6 存储过程

3.6.1 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。

```

1  @csrf_exempt
2  def register(request):
3      if request.method == 'POST':
4          body = json.loads(request.body)
5          username = body.get('name')
6          password = body.get('password')
7          role = body.get('role')
8          try:
9              # 检查用户是否已经存在
10             user = User.objects.get(username=username)
11             result={'data':{}}
12             result['code'] = -1
13             return JsonResponse(result)
14         except User.DoesNotExist:
15             # 创建用户
16             user = User(username=username,
17                         password=password,
18                         identity=-1,
19
20             registration_date=datetime.now(pytz.timezone('Asia/Shanghai')))
21             result={'data':{}}
22             if role == "admin":
23                 raise Exception("管理员账号只能由超级管理员手动添加")
24             elif role == "teacher":
25                 user.identity = 1
26                 user.save()
27                 tutor=Tutor(user_id=user)
28                 tutor.save()
29                 result['data']['roles']=[{'id': 'teacher'}]
30             elif role == "student":

```

```

30         user.identity = 2
31         user.save()
32         student=Student(user_id=user)
33         student.save()
34         result['data']['roles']=[{'id': 'student'}]
35     else:
36         raise Exception("未知身份")
37
38     request.session['user_id'] = user.user_id
39     sendNotice(user,"注册成功","欢迎加入家教综合服务平台!")
40
41     result['code'] = 0
42     result['data']['token']="Authorization:" + str(random.random())
43     result['data']['id'] = str(user.user_id)
44     return JsonResponse(result)
45 else:
46     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。

```

1  @csrf_exempt
2  def login(request):
3      if request.method == 'POST':
4          body = json.loads(request.body)
5          username = body.get('name')
6          password = body.get('password')
7          result = {'data': {}}
8          try:
9              # 查询数据库中是否存在对应的用户
10             user = User.objects.get(username=username, password=password)
11             if user.identity == 0:
12                 result['data']['roles'] = [{'id': 'admin'}]
13             elif user.identity == 1:
14                 result['data']['roles'] = [{'id': 'teacher'}]
15             elif user.identity == 2:
16                 result['data']['roles'] = [{'id': 'student'}]
17             else:
18                 raise Exception("未知身份")
19             result['data']['id'] = str(user.user_id)
20
21             request.session['user_id'] = user.user_id
22
23             # 登录成功
24             result['code'] = 0
25             result['data']['token'] = "Authorization:" +
str(random.random())
26             result['message'] = "登录成功"
27         except User.DoesNotExist:
28             # 用户不存在或密码错误
29             result['code'] = -1

```

```

30         result['message'] = "账户名或密码错误"
31         return JsonResponse(result)
32     else:
33         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

3. 重置密码

- **涉及的基本表：** User
- **过程描述：** 检查用户是否存在，如果存在则将用户的密码重置为新密码，否则返回错误信息。

```

1  def resetPassword(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          user_id=int(body.get('id'))
5          user=User.objects.get(user_id=user_id)
6          old_password=body.get('oldpassword')
7          new_password=body.get('password')
8          if user.password!=old_password:
9              return JsonResponse({'code': -1, 'message': '密码错误'})
10         user.password=new_password
11         user.save()
12         return JsonResponse({'code': 0})
13     else:
14         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

3.6.2 帖子广场管理

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。

```

1  @csrf_exempt
2  def sendPost(request):
3      if request.method == 'POST':
4          body = json.loads(request.body)
5          request_id=int(body.get('id'))
6          title = body.get('data').get('title')
7          startDate = body.get('data').get('startDate')
8          endDate = body.get('data').get('endDate')
9          subjects=body.get('data').get('subjects')
10         location=body.get('data').get('location')
11         fullLocation=body.get('data').get('fullLocation')
12         telephoneNumber=body.get('data').get('telephoneNumber')
13         email=body.get('data').get('emailAddress')
14         content=body.get('data').get('content')
15         is_complete = all([title, startDate, endDate, subjects, location,
16                             fullLocation, telephoneNumber, content])
17         if not is_complete:
18             return JsonResponse({'code': 0})
19         try :
20             user=User.objects.get(user_id=request_id)

```

```

21     except User.DoesNotExist:
22         return JsonResponse({'code': -1, 'message': '用户不存在'})
23
24     if user.identity == 0:
25         raise Exception("管理员不通过sendPost发帖")
26     else:
27         post=Post(
28             user_id=user,
29             title=title,
30             postDate=datetime.now(pytz.timezone('Asia/Shanghai')),
31             startDate=startDate,
32             endDate=endDate,
33             location=location,
34             fullLocation=fullLocation,
35             telephoneNumber=telephoneNumber,
36             emailAddress=email,
37             content=content,
38             is_completed=True,
39             is_approved=False,
40         )
41         post.save()
42
43         postSubjects = [
44             PostSubject(post_id=post, subject=subject)
45             for subject in subjects
46         ]
47         PostSubject.objects.bulk_create(postSubjects)
48
49         result={'data':{}}
50         result['code'] = 0
51         return JsonResponse(result)
52     else:
53         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。

```

1  @csrf_exempt
2  def savePost(request):
3      if request.method == 'POST':
4          body = json.loads(request.body)
5          request_id=int(body.get('id'))
6          title = body.get('data').get('title')
7          startDate = body.get('data').get('startDate') or None
8          endDate = body.get('data').get('endDate') or None
9          subjects=body.get('data').get('subjects')
10         location=body.get('data').get('location')
11         fullLocation=body.get('data').get('fullLocation')
12         telephoneNumber=body.get('data').get('telephoneNumber')
13         email=body.get('data').get('emailAddress') or None
14         content=body.get('data').get('content')
15

```

```

16         try:
17             user=User.objects.get(user_id=request_id)
18         except User.DoesNotExist:
19             return JsonResponse({'code': -1, 'message': '用户不存在'})
20
21     if user.identity ==0:
22         raise Exception("管理员无法保存帖子")
23     else:
24         try:
25             post=Post.objects.get(user_id=user,is_completed=False)
26             post.title=title
27             post.startDate=startDate
28             post.endDate=endDate
29             post.location=location
30             post.fullLocation=fullLocation
31             post.telephoneNumber=telephoneNumber
32             post.emailAddress=email
33             post.content=content
34             post.save()
35             PostSubject.objects.filter(post_id=post).delete()
36             postSubjects = [
37                 PostSubject(post_id=post, subject=subject)
38                 for subject in subjects
39             ]
40             PostSubject.objects.bulk_create(postSubjects)
41             result={'data':{}}
42             result['code'] = 0
43             return JsonResponse(result)
44         except Post.DoesNotExist:
45             post=Post(
46                 user_id=user,
47                 title=title,
48                 startDate=startDate,
49                 endDate=endDate,
50                 location=location,
51                 fullLocation=fullLocation,
52                 telephoneNumber=telephoneNumber,
53                 emailAddress=email,
54                 content=content,
55                 is_completed=False,
56                 is_approved=False,
57             )
58             post.save()
59             postSubjects = [
60                 PostSubject(post_id=post, subject=subject)
61                 for subject in subjects
62             ]
63             PostSubject.objects.bulk_create(postSubjects)
64             result={'data':{}}
65             result['code'] = 0
66             return JsonResponse(result)
67     else:
68         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。

```

1  @csrf_exempt
2  def getSavedPost(request):
3      if request.method == 'GET':
4          try:
5              user_id=request.GET.get('id')
6              user=User.objects.get(user_id=int(user_id))
7              if user.identity==0:
8                  raise Exception("管理员无法获取帖子")
9              else:
10                 try:
11                     post=Post.objects.get(user_id=user,is_completed=False)
12                     result={'data':{}}
13                     result['code'] = 0
14                     result['data']['title']=post.title
15                     result['data']['startDate']=post.startDate
16                     result['data']['endDate']=post.endDate
17                     location=post.location.strip('')
18                     location=json.loads(location.replace("'",''))
19                     result['data']['location']=location
20                     result['data']['fullLocation']=post.fullLocation
21                     result['data']['telephoneNumber']=post.telephoneNumber
22                     result['data']['emailAddress']=post.emailAddress
23                     result['data']['content']=post.content
24                     subjects=PostSubject.objects.filter(post_id=post)
25                     result['data']['subjects']=[subject.subject for subject
in subjects]
26                 except:
27                     result={'data':{}}
28                     result['code'] = 0
29                     return JsonResponse(result)
30             except User.DoesNotExist:
31                 result={'data':{}}
32                 result['code'] = 0
33                 return JsonResponse(result)
34         else:
35             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、地址。如果搜索关键词为空，则返回所有符合条件的帖子。采用模糊匹配。

```

1  @csrf_exempt
2  def getPosts(request):
3      if request.method == 'GET':
4          request_id=int(request.GET.get('id'))
5          request_page=int(request.GET.get('page'))
6          request_query=request.GET.get('query')

```

```

7         start=(request_page-1)*10
8         end=request_page*10
9         returnSubjects=[]
10        user=User.objects.get(user_id=request_id)
11        try:
12            if user.identity==1:
13                filter_id=2
14            else:
15                filter_id=1
16
17            if request_query=="":
18                if user.identity!=0:
19
20                posts=Post.objects.filter(is_completed=True,user_id__identity=filter_id,is_
21                approved=True)
22            else:
23                posts=Post.objects.filter(is_completed=True,is_approved=False)
24            else:
25                all_posts = Post.objects.all()
26                matching_posts = []
27                for post in all_posts:
28
29                subjects=PostSubject.objects.filter(post_id=post).values_list('subject',
30                flat=True)
31
32                subjects_str = ' '.join(subjects)
33                username_str = post.user_id.username
34                title_str = post.title
35                content_str = post.content
36                location_str = post.location.strip('')
37                location_list = json.loads(location_str.replace("'",
38                '''))
39
40                location_str = ' '.join(location_list) if
41                isinstance(location_list, list) else location_str
42                long_str = f"{subjects_str} {username_str} {title_str}
43                {content_str} {location_str}".lower()
44
45                request_query = request_query.lower()
46                if request_query in long_str:
47                    matching_posts.append(post)
48                if user.identity != 0:
49                    posts = [post for post in matching_posts if
50                    post.is_completed and post.user_id.identity == filter_id and
51                    post.is_approved]
52            else:
53                posts = [post for post in matching_posts if
54                post.is_completed and not post.is_approved]
55
56            return_posts=[]
57            for post in posts:
58                user=post.user_id
59                now = datetime.now(pytz.timezone('Asia/Shanghai'))
60                post_date =
61                post.postDate.astimezone(pytz.timezone('Asia/Shanghai'))
62                time_diff = now - post_date
63                if time_diff < timedelta(hours=24):

```

```

51         date_display = f"{time_diff.seconds // 3600}小时前"
52     elif time_diff < timedelta(days=3):
53         date_display = f"{time_diff.days}天前"
54     else:
55         date_display = post_date.strftime("%Y-%m-%d")
56
57     return_subject_lines=PostSubject.objects.filter(post_id=post)
58     returnSubjects=[]
59     returnSubjects = [subject.subject for subject in
60     return_subject_lines]
61     location=post.location.strip('')
62     location=json.loads(location.replace("'", ''))
63     location=location[0]
64     return_posts.append({
65         'id':str(post.post_id),
66         'title':post.title,
67         'tags':returnSubjects,
68         'content':post.content,
69         'author':user.username,
70         'authorId':user.user_id,
71         'date':date_display,
72         'location':location,
73     })
74     result={'posts':return_posts[start:end], 'total':len(posts)}
75     return JsonResponse(result)
76 except User.DoesNotExist:
77     raise Exception("用户不存在")
78 else:
79     return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

5. 获取用户所发的帖子

- **涉及的基本表：** Post , PostSubject
- **过程描述：** 根据帖子id直接获取帖子内容，如果帖子不存在则返回错误信息。

```

1  @csrf_exempt
2  def getPost(request):
3      if request.method == 'GET':
4          post_id=int(request.GET.get('id'))
5          try:
6              post=Post.objects.get(post_id=post_id)
7              result={'data':{}}
8              result['code'] = 0
9              result['data']['title']=post.title
10             result['data']['author']=post.user_id.username
11             location=post.location.strip('')
12             location=json.loads(location.replace("'", ''))
13             result['data']['location']=location
14             result['data']['fullLocation']=post.fullLocation
15             result['data']['telephoneNumber']=post.telephoneNumber
16             result['data']['emailAddress']=post.emailAddress
17             result['data']['startDate']=post.startDate
18             result['data']['endDate']=post.endDate
19             subject_lines=PostSubject.objects.filter(post_id=post)
20             subjects=[]

```



```

21         subjects = [subject.subject for subject in subject_lines]
22         result['data']['subjects']=subjects
23         result['data']['content']=post.content
24         return JsonResponse(result)
25
26     except Post.DoesNotExist:
27         return JsonResponse({'code': -1, 'message': '帖子不存在'})

```

3.6.3 师生个人信息管理

1. 更新学生信息

- **涉及的基本表：** Student
- **过程描述：** 更新学生的个人信息。

```

1  @csrf_exempt
2  def updateStudentInfo(request):
3      if request.method == 'POST':
4          body = json.loads(request.body)
5          request_id=int(body.get('id'))
6          grade = body.get('data').get('grade')
7          gender = body.get('data').get('gender')
8          age=body.get('data').get('age')
9          email=body.get('data').get('email')
10         telephone=body.get('data').get('telephone')
11         address=body.get('data').get('address')
12         intro=body.get('data').get('intro')
13         personalSignature=body.get('data').get('personalSignature')
14
15         user=User.objects.get(user_id=request_id)
16         student=Student.objects.get(user_id=user)
17         student.grade=grade
18         student.gender=gender
19         student.age=age
20         student.email=email
21         student.telephone=telephone
22         student.address=address
23         student.intro=intro
24         student.personalSignature=personalSignature
25         student.save()
26
27         result={'data':{}}
28         result['code'] = 0
29         return JsonResponse(result)
30     else:
31         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

2. 获取学生信息

- **涉及的基本表：** Student
- **过程描述：** 获取学生的个人信息。

```

1  @csrf_exempt
2  def getStudentInfo(request):

```

```

3     if request.method == 'GET':
4         request_id=int(request.GET.get('id'))
5         result={'data':{}}
6         result['code'] = 0
7         user=User.objects.get(user_id=request_id)
8         student=Student.objects.get(user_id=user)
9         result['data']['name']=user.username
10        result['data']['email']=student.email
11        result['data']['telephone']=student.telephone
12        result['data']['address']=student.address
13        result['data']['personalSignature']=student.personalSignature
14        result['data']['intro']=student.intro
15        result['data']['age']=student.age
16        result['data']['gender']=student.gender
17        result['data']['grade']=student.grade
18        return JsonResponse(result)
19    else:
20        return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

3. 更新家教信息

- **涉及的基本表：** Tutor
- **过程描述：** 更新家教的个人信息。

```

1  @csrf_exempt
2  @csrf_exempt
3  def updateTeacherInfo(request):
4      if request.method == 'POST':
5          body = json.loads(request.body)
6          request_id=int(body.get('id'))
7          degree=body.get('data').get('degree')
8          gender=body.get('data').get('gender')
9          age=body.get('data').get('age')
10         email=body.get('data').get('email')
11         telephone=body.get('data').get('telephone')
12         address=body.get('data').get('address')
13         intro=body.get('data').get('intro')
14         personalSignature=body.get('data').get('personalSignature')
15
16         user=User.objects.get(user_id=request_id)
17         tutor=Tutor.objects.get(user_id=user)
18         tutor.degree=degree
19         tutor.gender=gender
20         tutor.age=age
21         tutor.email=email
22         tutor.telephone=telephone
23         tutor.address=address
24         tutor.intro=intro
25         tutor.personalSignature=personalSignature
26         tutor.save()
27
28         result={'data':{}}
29         result['code'] = 0
30         return JsonResponse(result)
31     else:

```

4. 获取家教信息

- **涉及的基本表:** Tutor
- **过程描述:** 获取家教的个人信息。注意在评论区需要获取评论学生的头像，如果学生没有头像则使用默认头像。

```

1  @csrf_exempt
2  def getTeacherInfo(request):
3      if request.method == 'GET':
4          request_id=int(request.GET.get('id'))
5
6          result={'data':{}}
7          result['code'] = 0
8          user=User.objects.get(user_id=request_id)
9          tutor=Tutor.objects.get(user_id=user)
10         result['data']['name']=user.username
11         result['data']['email']=tutor.email
12         result['data']['telephone']=tutor.telephone
13         result['data']['address']=tutor.address
14         result['data']['personalSignature']=tutor.personalSignature
15         result['data']['intro']=tutor.intro
16         result['data']['gender']=tutor.gender
17         result['data']['age']=tutor.age
18         result['data']['degree']=tutor.degree
19         result['data']['rate']=round(tutor.rate,1)
20         result['data']['rateNum']=tutor.rateNum
21         comments=Review.objects.filter(tutor_id=user)
22         return_comments=[]
23         for comment in comments:
24             return_comments.append({
25                 'id':str(comment.review_id),
26                 'authorName':comment.student_id.username,
27                 'rating':comment.rating,
28                 'content':comment.content,
29                 'date':comment.date,
30                 'avatar':comment.student_id.avatar if
comment.student_id.avatar else 'http://120.46.1.4:9000/zxb/png/Akkarin.png',
31             })
32         result['data']['comments']=return_comments
33         print(result)
34         return JsonResponse(result)
35     else:
36         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

5. 更新头像

- **涉及的基本表:** User
- **过程描述:** 更新用户的头像。注意为了避免图片重名，我们使用uuid生成唯一文件名，将文件上传至容器后，将访问链接保存至数据库中。

```

1  def uploadAvatar(request):
2      if request.method == 'POST':

```

```

3         id_request=json.loads(request.POST.get('data'))
4         # 使用 request.POST 和 request.FILES 解析 FormData 对象
5         user_id = int(id_request.get('id'))
6         file = request.FILES['file'] # 获取上传的文件
7         file_name = file.name
8         file_type = file_name.split('.')[-1]
9
10        unique_file_name = f"{uuid.uuid4()}.{file_type}"
11
12        try:
13            user = User.objects.get(user_id=user_id)
14        except User.DoesNotExist:
15            return JsonResponse({'code': 1, 'message': '用户不存在'})
16
17        # 使用 MinioClient 上传文件并获取下载链接
18        minio_client = MinioClient()
19        file_data = file.read()
20        try:
21            minio_client.upload_file(file_data, unique_file_name, file_type)
22            download_link = get_download_url(unique_file_name, file_type)
23        except ValueError as e:
24            return JsonResponse({'code': 1, 'message': str(e)})
25
26        # 更新用户头像链接
27        user.avatar = download_link
28        user.save()
29
30        result = {'code': 0, 'message': ''}
31        return JsonResponse(result)
32    else:
33        return JsonResponse({'code': 1, 'message': '仅支持POST请求'})

```

6. 获取头像

- **涉及的基本表：** User
- **过程描述：** 获取用户的头像。如果用户没有头像，则使用默认头像。

```

1 def getAvatar(request):
2     if request.method == 'GET':
3         user_id = int(request.GET.get('id'))
4         try:
5             user = User.objects.get(user_id=user_id)
6         except User.DoesNotExist:
7             return JsonResponse({'code': 1, 'message': '用户不存在'})
8
9         if user.avatar:
10            avatar_url = user.avatar
11        else:
12            avatar_url = 'http://120.46.1.4:9000/zxb/png/Akkarin.png'
13        print(avatar_url)
14        result = {'code': 0, 'avatar': avatar_url}
15        print(result)
16        return JsonResponse(result)
17    else:

```

3.6.4 个人工作管理

1. 获取本人发布的所有帖子

- **涉及的基本表：** Post , PostSubject
- **过程描述：** 获取用户发布的所有帖子。注意，有别于帖子广场，未经审核的帖子也会在这里显示。

```

1 def getUserPosts(request):
2     if request.method == 'GET':
3         request_id=int(request.GET.get('id'))
4         user=User.objects.get(user_id=request_id)
5         posts=Post.objects.filter(user_id=user,is_completed=True)
6
7         return_posts=[]
8         result={}
9         result['code'] = 0
10        for post in posts:
11            return_posts.append({
12                'id':str(post.post_id),
13                'title':post.title,
14                'content':post.content,
15            })
16        result['posts']=return_posts
17        return JsonResponse(result)
18    else:
19        return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

2. 删除自己发布的帖子

- **涉及的基本表：** Post , PostSubject
- **过程描述：** 用户可以自行删除自己发布的帖子。

```

1 def deletePost(request):
2     if request.method == 'POST':
3         body = json.loads(request.body)
4         post_id=int(body.get('postId'))
5         post=Post.objects.get(post_id=post_id)
6         post.delete()
7         result={'data':{}}
8         result['code'] = 0
9         return JsonResponse(result)
10    else:
11        return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

3. 家教获取自己的所有学生

- **涉及的基本表：** Link
- **过程描述：** 家教可以获取自己的所有学生。

```

1 def getStudents(request):
2     if request.method == 'GET':

```

```

3         request_id=int(request.GET.get('id'))
4         students=Link.objects.filter(tutor_id=request_id)
5         return_students=[]
6         result={}
7         result['code'] = 0
8         for student in students:
9             return_students.append({
10                 'id':str(student.student_id.user_id),
11                 'name':student.student_id.username,
12             })
13         result['students']=return_students
14         return JsonResponse(result)
15     else:
16         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

4. 学生获取自己的家教

- **涉及的基本表：** Link
- **过程描述：** 学生可以获取自己的家教。

```

1 def getTutors(request):
2     if request.method == 'GET':
3         request_id=int(request.GET.get('id'))
4         tutors=Link.objects.filter(student_id=request_id)
5         return_tutors=[]
6         result={}
7         result['code'] = 0
8         for tutor in tutors:
9             return_tutors.append({
10                 'id':str(tutor.tutor_id.user_id),
11                 'name':tutor.tutor_id.username,
12             })
13         result['tutors']=return_tutors
14         return JsonResponse(result)
15     else:
16         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

5. 发送通知

- **涉及的基本表：** Notification
- **过程描述：** 用户在执行相关工作后，会发送信息给涉及到的其他用户。这是一个工具方法，前端并不会直接调用这一api。

```

1 def sendNotice(user,title,description):
2     notification=Notification(
3         user_id=user,
4         notificationDate=datetime.now(pytz.timezone('Asia/Shanghai')),
5         title=title,
6         description=description,
7         is_read=False,
8     )
9     notification.save()

```

6. 获取通知

- **涉及的基本表:** Notification
- **过程描述:** 用户可以获取自己的通知。

```

1  def getNotices(request):
2      if request.method == 'GET':
3          request_id=int(request.GET.get('id'))
4          user=User.objects.get(user_id=request_id)
5          notifications=Notification.objects.filter(user_id=user)
6
7          return_notifications=[]
8          result={'data':{}}
9          result['code'] = 0
10         for notification in notifications:
11             return_notifications.append({
12                 'title':notification.title,
13                 'description':notification.description,
14             })
15         result['data']['notices']=return_notifications
16         new_len=len(Notification.objects.filter(user_id=user,is_read=False))
17         Notification.objects.filter(user_id=user).update(is_read=True)
18         result['data']['newNum']=new_len
19         return JsonResponse(result)
20     else:
21         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

7. 完成待办事项以建立师生关系

- **涉及的基本表:** Todo , Link
- **过程描述:** 家教和学生通过完成待办事项建立关系。

```

1  def link(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          tutor_id=int(body.get('teacherId'))
5          student_id=int(body.get('studentId'))
6          tutor=User.objects.get(user_id=tutor_id)
7          student=User.objects.get(user_id=student_id)
8
9          link=Link(tutor_id=tutor,student_id=student)
10         link.save()
11
12         user_id=request.session['user_id']
13         if user_id==tutor_id:
14             sendNotice(student,"招聘成功",f"您正式成为{tutor.username}的学生")
15             todos=Todo.objects.filter(owner_id=tutor,accepter_id=student)
16             for todo in todos:
17                 todo.delete()
18         else:
19             sendNotice(tutor,"应聘成功",f"您正式成为{student.username}的家教")
20             todos=Todo.objects.filter(owner_id=student,accepter_id=tutor)
21             for todo in todos:
22                 todo.delete()
23
24         result={'data':{}}

```

```

25         result['code'] = 0
26         return JsonResponse(result)
27     else:
28         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

8. 拒绝待办事项

- **涉及的基本表:** `Todo`
- **过程描述:** 家教和学生通过拒绝待办事项拒绝建立关系。

```

1  def refuseLink(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          tutor_id=int(body.get('teacherId'))
5          student_id=int(body.get('studentId'))
6          tutor=User.objects.get(user_id=tutor_id)
7          student=User.objects.get(user_id=student_id)
8
9          user_id=request.session['user_id']
10         if user_id==tutor_id:
11             sendNotice(student,"招聘失败",f"很抱歉，{tutor.username}未同意您的招
聘")
12             todos=Todo.objects.filter(owner_id=tutor,accepter_id=student)
13             for todo in todos:
14                 todo.delete()
15         else:
16             sendNotice(tutor,"应聘失败",f"很抱歉，{student.username}未同意您的应
聘")
17             todos=Todo.objects.filter(owner_id=student,accepter_id=tutor)
18             for todo in todos:
19                 todo.delete()
20
21         result={'data':{}}
22         result['code'] = 0
23         return JsonResponse(result)
24     else:
25         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

9. 取消师生关系

- **涉及的基本表:** `Link`
- **过程描述:** 家教和学生通过取消师生关系解除关系。

```

1  def unlink(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          print(body)
5          tutor_id=int(body.get('id'))
6          student_id=int(body.get('studentId'))
7          tutor=User.objects.get(user_id=tutor_id)
8          student=User.objects.get(user_id=student_id)
9
10         link=Link.objects.get(tutor_id=tutor,student_id=student)
11         link.delete()

```



```

12
13     user_id=request.session['user_id']
14     if user_id==tutor_id:
15         sendNotice(student,"解除招聘",f"您已经不再是{tutor.username}的学生")
16     else:
17         sendNotice(tutor,"解除应聘",f"您已经不再是{student.username}的家教")
18
19     result={'data':{}}
20     result['code'] = 0
21     return JsonResponse(result)
22 else:
23     return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

10. 学生给家教评价

- **涉及的基本表：** `Review`
- **过程描述：** 学生给家教评价。

```

1  def submitComment(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          student_id=int(body.get('studentId'))
5          tutor_id=int(body.get('teacherId'))
6          rating=int(body.get('rate'))
7          comment=body.get('comment')
8          user=User.objects.get(user_id=student_id)
9          tutor=User.objects.get(user_id=tutor_id)
10         teacher=Tutor.objects.get(user_id=tutor)
11         if teacher.rateNum!=0:
12             rate_sum=teacher.rate*teacher.rateNum
13         else:
14             rate_sum=0
15         old_review=Review.objects.get(student_id=user,tutor_id=tutor)
16         if old_review:
17             teacher.rateNum-=1
18             rate_sum-=old_review.rating
19             old_review.delete()
20         review=Review(
21             student_id=user,
22             tutor_id=tutor,
23             rating=rating,
24             content=comment,
25             date=datetime.now(pytz.timezone('Asia/Shanghai')),
26         )
27         review.save()
28
29         sendNotice(tutor,"收到新评价",f"您收到了来自{user.username}的新评价")
30         teacher.rateNum+=1
31         rate_sum+=rating
32         teacher.rate=rate_sum/teacher.rateNum
33         teacher.save()
34
35         result={'data':{}}
36         result['code'] = 0
37         return JsonResponse(result)

```

```

38     else:
39         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

11. 家教给学生上传学习资料

- **涉及的基本表：** StudyMaterial
- **过程描述：** 家教给学生上传学习资料。为避免重名文件发生覆盖，这里在上传容器前使用uuid生成唯一文件名，然后将文件名和下载链接存入数据库。

```

1  def submitLearningMaterial(request):
2      if request.method == 'POST':
3          body = json.loads(request.POST.get('data'))
4          tutor_id = int(body.get('id'))
5          student_id = int(body.get('studentId'))
6          file = request.FILES['file'] # 获取上传的文件
7          file_name = file.name
8          file_type = file_name.split('.')[-1]
9          user=User.objects.get(user_id=student_id)
10         tutor=User.objects.get(user_id=tutor_id)
11
12         unique_file_name = f"{uuid.uuid4()}.{file_type}"
13
14         minio = MinioClient()
15         file_data = file.read()
16         minio.upload_file(file_data, unique_file_name, file_type)
17         download_link = get_download_url(unique_file_name, file_type)
18
19         material=StudyMaterial(
20             tutor_id=tutor,
21             student_id=user,
22             file_name=file_name,
23             download_link=download_link,
24             upload_date=datetime.now(pytz.timezone('Asia/Shanghai')),
25         )
26         material.save()
27         sendNotice(user,"收到新学习资料",f"您收到了来自{tutor.username}的新学习资
料")
28         result={'data':{}}
29         result['code'] = 0
30         return JsonResponse(result)
31     else:
32         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

12. 学生获取学习资料

- **涉及的基本表：** StudyMaterial
- **过程描述：** 学生获取家教上传的学习资料。

```

1  def getLearningMaterials(request):
2      if request.method == 'GET':
3          request_id=int(request.GET.get('id'))
4          user=User.objects.get(user_id=request_id)
5          materials=StudyMaterial.objects.filter(student_id=user)
6

```

```

7         return_materials=[]
8         result={}
9         result['code'] = 0
10        for material in materials:
11            return_materials.append({
12                'id':str(material.material_id),
13                'filename':material.file_name,
14                'publisher':material.tutor_id.username,
15                'downloadLink':material.download_link,
16                'date':material.upload_date,
17            })
18        result['materials']=return_materials
19        return JsonResponse(result)
20    else:
21        return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

13. 发送待办事项

- **涉及的基本表：** Todo
- **过程描述：** 用户在执行相关工作后，会发送待办事项给涉及到的其他用户。这是一个工具方法，前端并不会直接调用这一api。

```

1 def sendTodo(user,post,accepter):
2     todo=Todo(
3         owner_id=user,
4         accepter_id=accepter,
5         accept_post_id=post,
6         is_completed=False,
7     )
8     todo.save()

```

14. 获取待办事项

- **涉及的基本表：** Todo
- **过程描述：** 用户可以获取自己的待办事项。

```

1 def getTodos(request):
2     if request.method == 'GET':
3         request_id=int(request.GET.get('id'))
4         user=User.objects.get(user_id=request_id)
5         todos=Todo.objects.filter(owner_id=user)
6
7         return_todos=[]
8         result={}
9         result['code'] = 0
10        for todo in todos:
11            return_todos.append({
12                'id':str(todo.todo_id),
13                'postId':str(todo.accept_post_id.post_id),
14                'postTitle':todo.accept_post_id.title,
15                'accepterName':todo.accepter_id.username,
16                'accepterId':str(todo.accepter_id.user_id),
17            })
18        result['todos']=return_todos

```

```

19         print(result)
20         return JsonResponse(result)
21     else:
22         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

15. 获取站点公告

- **涉及的基本表:** Announcement
- **过程描述:** 用户可以获取站点公告。

```

1  def getAnnouncements(request):
2      if request.method == 'GET':
3          try:
4              return_announcements=[]
5              for announcement in Announcement.objects.all():
6                  now = datetime.now(pytz.timezone('Asia/Shanghai'))
7                  announcement_date =
announcement.announcementDate.astimezone(pytz.timezone('Asia/Shanghai'))
8                  time_diff = now - announcement_date
9                  if time_diff < timedelta(hours=24):
10                     date_display = f"{time_diff.seconds // 3600}小时前"
11                 elif time_diff < timedelta(days=3):
12                     date_display = f"{time_diff.days}天前"
13                 else:
14                     date_display = announcement_date.strftime("%Y-%m-%d")
15                 return_announcements.append({
16                     'title':announcement.title,
17                     'content':announcement.description,
18                     'date':date_display,
19                 })
20                 result={'data':{}}
21                 result['code'] = 0
22                 result['data']['announcements']=return_announcements
23                 return JsonResponse(result)
24             except User.DoesNotExist:
25                 raise Exception("用户不存在")
26         else:
27             return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

3.6.5 管理员事务管理

1. 获取用户身份

- **涉及的基本表:** User
- **过程描述:** 管理员可以获取用户的身份。

```

1  def getUserRole(request):
2      if request.method == 'GET':
3          post_id=int(request.GET.get('id'))
4          post=Post.objects.get(post_id=post_id)
5          user=post.user_id
6          result={'data':{}}
7          result['code'] = 0
8          if user.identity==0:

```

```

9         raise Exception("管理员不发帖子")
10    elif user.identity==1:
11        result['data']['userRole']="家教"
12    else:
13        result['data']['userRole']="学生"
14    print(result)
15    return JsonResponse(result)
16 else:
17    return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

2. 审核通过帖子

- **涉及的基本表：** Post
- **过程描述：** 管理员可以审核通过帖子。

```

1  def approvePost(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          print(body)
5          post_id=int(body.get('postId'))
6          post=Post.objects.get(post_id=post_id)
7          post.is_approved=True
8          post.save()
9          sendNotice(post.user_id,"帖子通过审核",f"您的帖子《{post.title}》已通过审
核")
10         result={}
11         result['code'] = 0
12         return JsonResponse(result)
13     else:
14         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

3. 审核拒绝帖子

- **涉及的基本表：** Post
- **过程描述：** 管理员可以审核拒绝帖子。

```

1  def rejectPost(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          post_id=int(body.get('postId'))
5          post=Post.objects.get(post_id=post_id)
6          post.delete()
7          sendNotice(post.user_id,"帖子未通过审核",f"您的帖子《{post.title}》未通过
审核")
8          result={}
9          result['code'] = 0
10         return JsonResponse(result)
11     else:
12         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

4. 获取所有家教

- **涉及的基本表：** Tutor
- **过程描述：** 管理员可以获取所有家教。

```

1  def getAllTeachers(request):
2      if request.method == 'GET':
3          teachers=User.objects.filter(identity=1)
4          return_teachers=[]
5          result={}
6          result['code'] = 0
7          for teacher in teachers:
8              return_teachers.append({
9                  'id':str(teacher.user_id),
10                 'name':teacher.username,
11             })
12         result['teachers']=return_teachers
13         return JsonResponse(result)
14     else:
15         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

5. 获取所有学生

- **涉及的基本表：** Student
- **过程描述：** 管理员可以获取所有学生。

```

1  def getAllStudents(request):
2      if request.method == 'GET':
3          students=User.objects.filter(identity=2)
4          return_students=[]
5          result={}
6          result['code'] = 0
7          for student in students:
8              return_students.append({
9                  'id':str(student.user_id),
10                 'name':student.username,
11             })
12         result['students']=return_students
13         return JsonResponse(result)
14     else:
15         return JsonResponse({'code': -1, 'message': '仅支持GET请求'})

```

6. 发表站点公告

- **涉及的基本表：** Announcement
- **过程描述：** 管理员可以发表站点公告。

```

1  def makeAnnouncement(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          title=body.get('title')
5          content=body.get('content')
6          announcement=Announcement(
7              title=title,
8              description=content,
9              announcementDate=datetime.now(pytz.timezone('Asia/Shanghai')),
10          )
11         announcement.save()
12         result={}

```

```

13         result['code']=0
14         return JsonResponse(result)
15     else:
16         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

7. 删除用户

- **涉及的基本表：** User
- **过程描述：** 管理员可以删除用户。

```

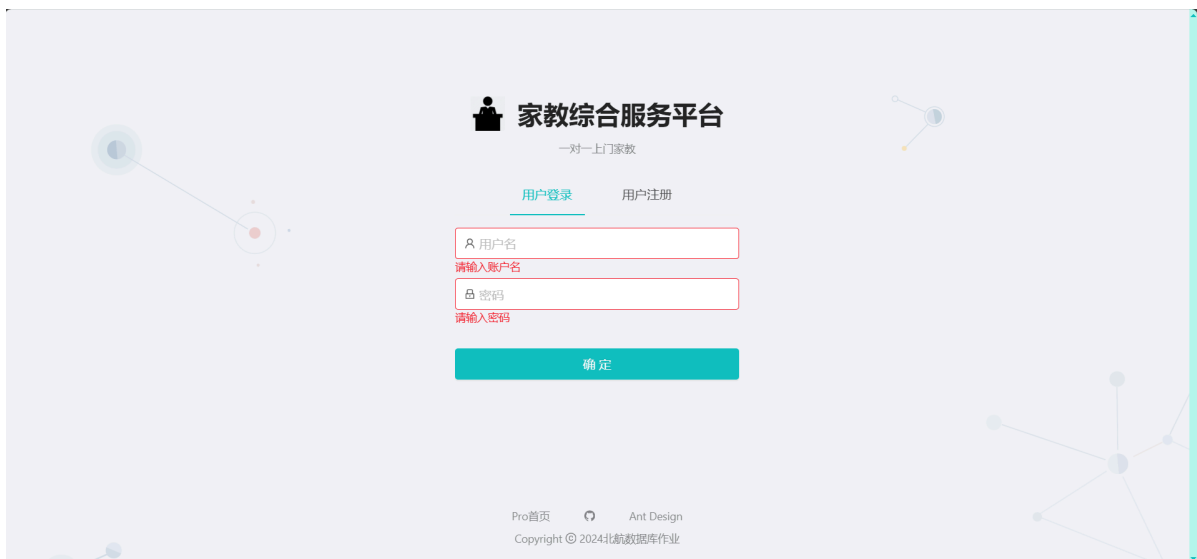
1  def deleteUser(request):
2      if request.method == 'POST':
3          body = json.loads(request.body)
4          user_id=int(body.get('id'))
5          user=User.objects.get(user_id=user_id)
6          user.delete()
7          result={}
8          result['code']=0
9          return JsonResponse(result)
10     else:
11         return JsonResponse({'code': -1, 'message': '仅支持POST请求'})

```

四、系统实现效果

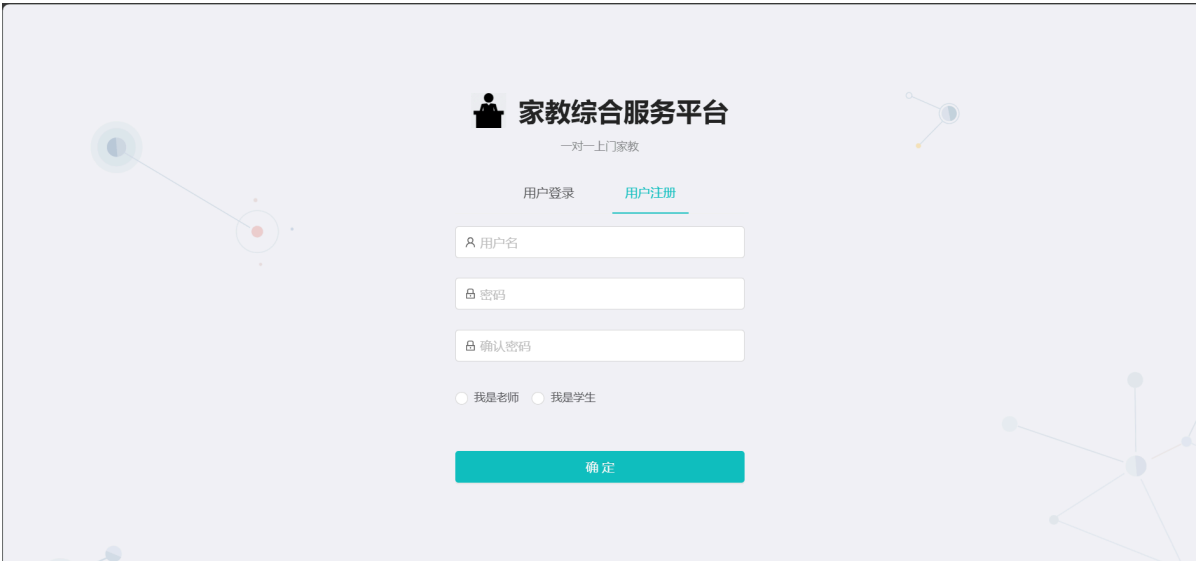
4.1 登录注册

4.1.1 登录



进入网站，或者在未登录的情况下访问网站页面都将跳转到首页，在这个页面用户可以进行登录，输入正确的用户名以及密码即可登录成功。

4.1.2 注册



用户可以在首页的标签页进行注册。填写用户名、密码，以及确认密码、身份之后即可成功注册。本系统的管理员账号由后端分配，不可直接注册。

4.1.3 退出登录

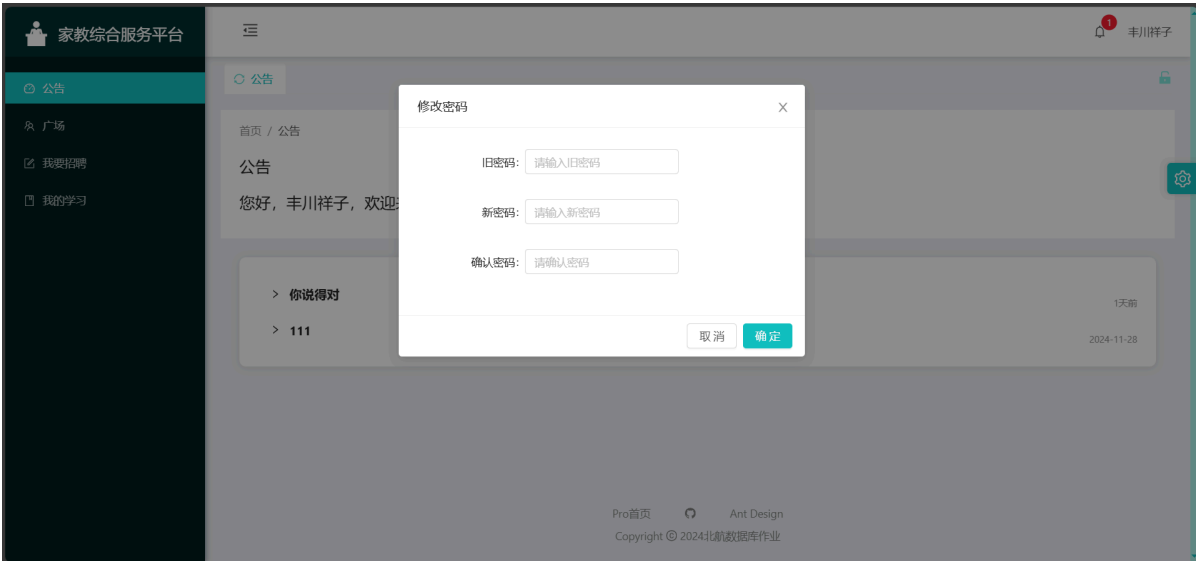


在登录成功后的主页面中，点击右上角的用户名可以展开下属的选择框可以选择退出登录。随后页面便会删除当前用户的登录认证信息。用户将会跳转到登录页面。

4.1.4 修改密码



在相同的选择框中可以选择**修改密码**，之后便会弹出修改密码对话框。



在经过前端验证“新密码”和“确认密码”一致、后端验证旧密码一致后即可修改密码。之后页面将会删除当前用户的登录认证信息。用户将会跳转到登录界面。

4.2 个人主页



在登录成功后的主页面中，点击右上角的用户名可以展开下属的选择框可以选择**个人主页**以查看个人主页。对于学生和家教两种不同的身份，个人主页的展示内容有所不同，主要区别在于是否有评论展示区以及评分展示区。

4.2.1 学生个人主页

公告编辑学生主页 ×学生主页 ×

首页 / 学生主页

学生主页



丰川祥子
客服小样倾心为您服务

性别	年龄	年级
女	15	高中一年级
邮箱:	电话:	地址:
123456@google.com	150820820	羽丘女子学园


个人简介:

我不是说了吗？请把你剩余的人生交给我。言ったでしょう？残り的人生、わたくしに下さいと

学生个人主页按照从上到下，从左到右的顺序展示姓名、个性签名、性别、年龄、年级、邮箱、电话、地址、以及个人简介。

4.2.2 家教个人主页

家教主页



五条悟
没关系，我是最强的！


评分3.7 | 评分人数3


性别	年龄	学历
男	28	大专
邮箱:	电话:	地址:
111111111111@163.com	123456789	东京都立咒术高等专门学校


个人简介:

昨天居然有人质疑我的实力，对于我不能打败宿傩这件事表示质疑，还问我“会输吗”，废话，我可是现代最强咒术师，区区史上最强的宿傩我根本不放在眼里。不管什么时候，我的回答只有一句话——“会赢的”，不说了，我要去打宿傩了。

用户评论:

 **丰川祥子** ★★★★★ 2028-12-08
你这个人，真是总想着自己呢

 **融斗花园第一突破手牛爷爷** ★★★ 2028-12-08
优秀学生，体验下墓，令人感动

 **虎杖悠仁** ★★★ 2028-12-08
会赢吗？

家教个人主页按照从上到下，从左到右的顺序展示姓名、个性签名、评分、评分人数、性别、年龄、学历、邮箱、电话、地址、个人简介以及用户评论。

4.2.3 编辑个人主页




在用户名下属的选择框中选择**编辑个人主页**可以对个人主页信息进行编辑。

首页 / 编辑家教主页

编辑家教主页

在这里你可以修改你的个人信息



Select File

Start Upload

学历

大专

性别

男

年龄

28

邮箱

1111111111@163.com

电话

123456789

地址

东京都立咒术高等专门学校

个性签名

没关系，我是最强的！

个人简介


昨天居然有人质疑我的实例，对于我不能打败宿雉这件事表示质疑，还问我“会输吗”。废话，我可是现代最强咒术师，区区史上最强的宿雉我根本不放在眼里。不管什么时候，我的回答只有一句话——“会赢的”。不说了，我要去打宿雉了。

保存

首页 / 编辑学生主页

编辑学生主页

在这里你可以修改你的个人信息



Select File

Start Upload

年级

高中一年级

性别

女

年龄

15

邮箱

123456@google.com

电话

150820820

地址

羽丘女子学园

个性签名

客服小样倾心为您服务

个人简介

我不是说了吗？请把你剩余的人生交给我。 言ったでしょう？残りの人生、わたくしに下さいと

保存

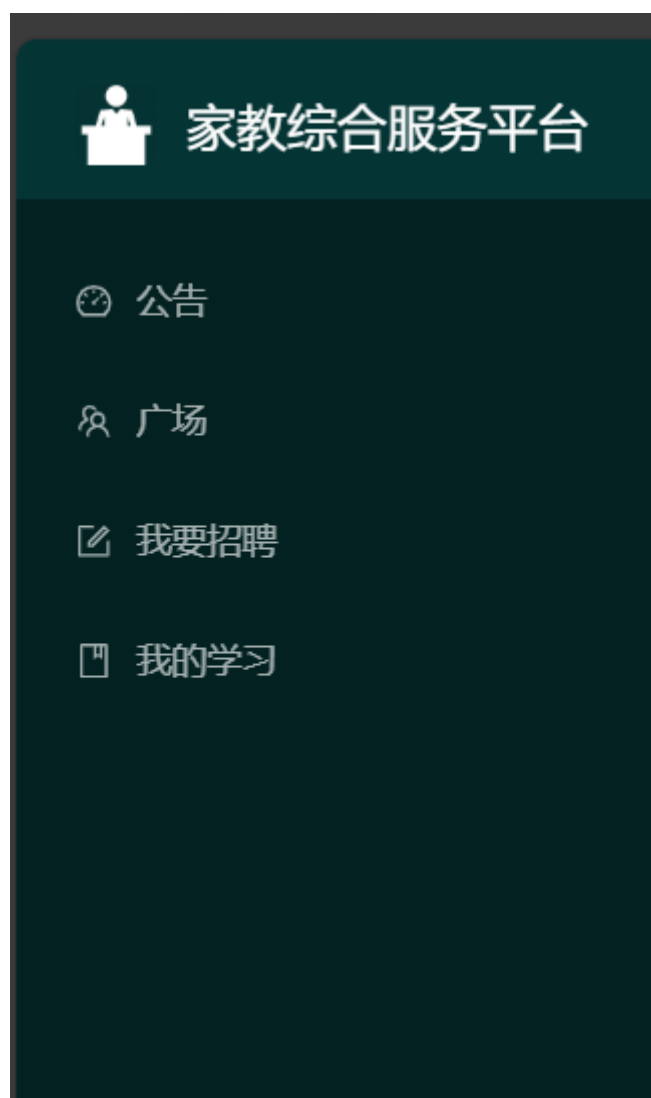
学生和家教的“编辑个人主页”页面大致相同。唯一的区别在于家教页面的“学历”一栏，在学生页面中显示为“年级”。

4.3 消息通知



点击顶栏的“闹钟”图标即可展开通知。当发生“帖子通过审核”“有新的公告”“家教发布学习资料”“请求建立师生关系”“成功建立师生关系”等多种情况时都会产生通知信息，以使用户及时获取重要信息。

4.4 侧边栏



在每个页面的左侧都存在侧边栏，用户可以在侧边栏中选择页面进行跳转。对于不同的身份（学生，家教，管理员），侧边栏包含的页面不同。公有的页面为“公告”与“广场”。学生独有的页面为“我要招聘”和“我的学习”。家教独有的页面为“我要求聘”和“我的工作”。管理员页面独有的页面为“用户管理”与“发布公告”。

4.5 公告

4.5.1 查看公告



在登录成功或者注册成功后用户会跳转到**公告页面**，在欢迎信息之下便是公告展示框，用户可以对公告标题进行点击以选择需要查看的公告内容。

4.5.2 发布公告



在用户以管理员身份登入后，可以在侧边栏选择**发布公告**“跳转到发布公告界面。在该界面管理员可以填写公告的内容并进行发布。

4.6 广场

4.6.1 帖子检索

请输入搜索内容

搜索

secondSave

编译技术

123

 aba 2024-12-09 北京市

剪门徒文

语文 数学 物理 化学 生物 历史 政治 地理

道生一,一生二,二生三,三生万物。万物负阴而抱阳,冲气以为和

 鬼谷子 2024-12-06 山东省

暑假太无聊了！想招一些可爱的学生捏

心理 火焰使

没有什么要求呢，只要学生够可爱够努力就可以了

 月咏小萌 2024-12-06 国外

招募弟子，随性而为

仙术 道法

相遇即是缘

 须菩提祖师 2024-12-06 国外

招一辈子乐队队友

音乐

。。。不敢说话

 高松灯 2024-12-06 上海市

草地绘画大师，诚心招募弟子

美术

我是未来人，外星人以及超能力者，精通多种古怪技能，欢迎前来咨询

 约翰·史密斯 2024-12-06 国外

s

英语 物理

ssss

 五条悟 2024-12-06 天津市

名侦探柯南家教启动

数学 物理

都说叫柯南了，可知水平如何

 柯南 2024-12-05 国外

吉他老师

吉他

青梅竹马大小姐家里破产了，出来赚钱。

 若叶睦 2024-12-04 国外

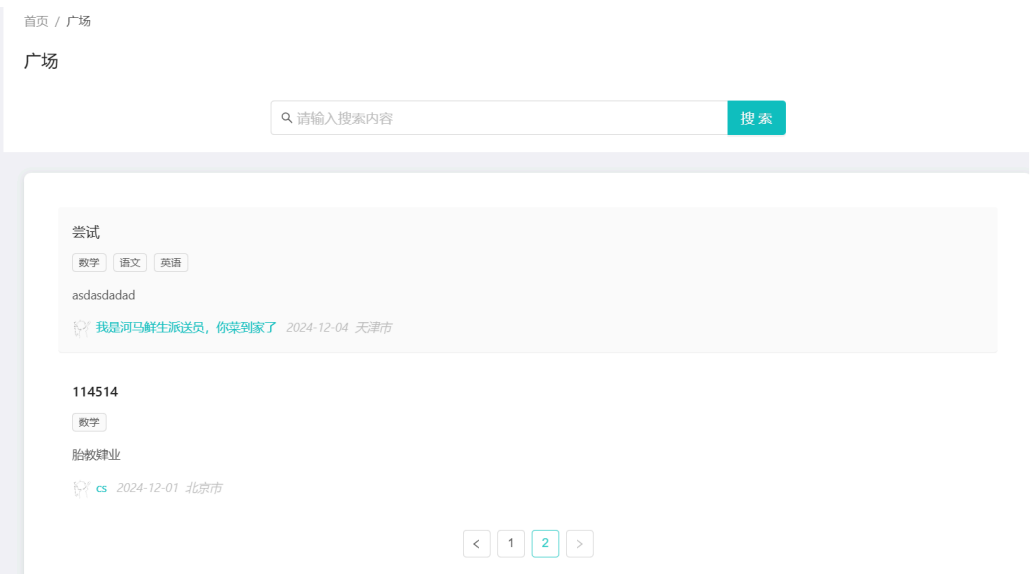
海淀地区求家教

美术 体育 音乐 信息技术

本人曾经在十个月内成功获得胎教学位，在三年内掌握了包括吃饭，睡觉，说话，跑步等多种专业技能，目前正在考虑进一步深造——冲...

 我是河马鲜生派送员，你菜到家了 2024-12-04 北京市

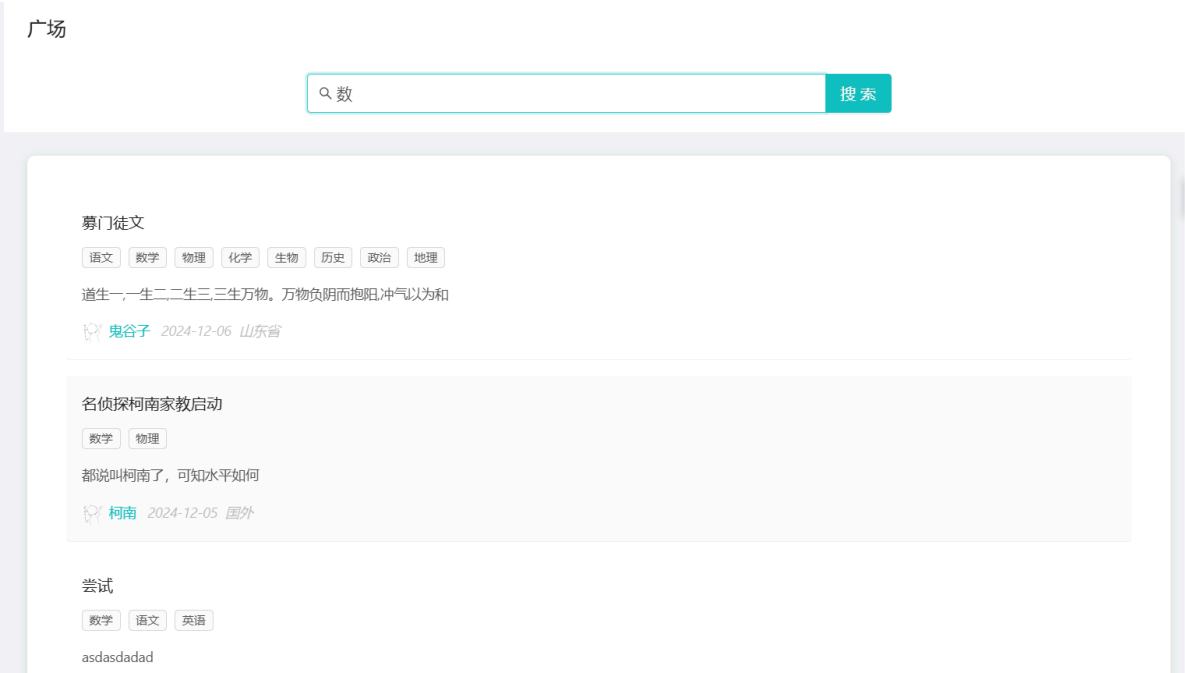
用户可以在侧边栏中选择跳转到**广场**页面。在广场页面，不同的角色所看到的帖子不同。学生可以看到家教们发的求聘帖；家教可以看到学生们发的招聘帖；管理员可以看到所有帖子。



网页每次至多加载10篇帖子。用户需要点击页面下方的页码去查看更多的帖子。这样的设计可以使用户界面更加简介，同时减少网页负担。

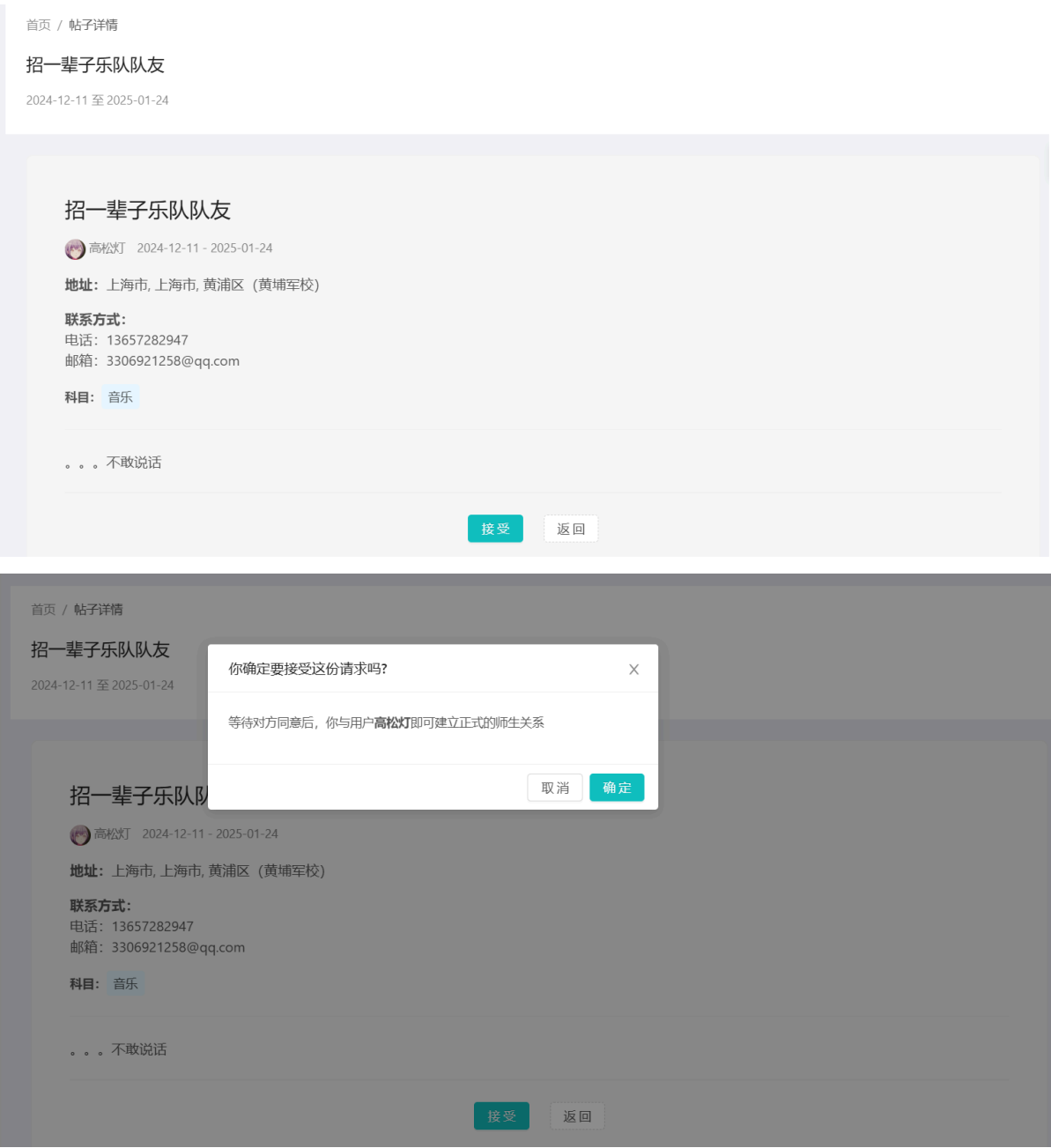


对于每个帖子，点击作者名（蓝字部分）即可查看作者的个人主页，点击除此之外的任意部分即可查看帖子详情。



广场页面的最上方存在搜索框，提供搜索功能，并支持基本的模糊搜索（搜索“北”字，可以检索出所有内容包含“北”字的帖子）

4.6.2 帖子详情



在广场页面点击帖子即可跳转到**帖子详情**页面, 在该页面家教 (学生) 可以选择接受“**招聘**” (“**求聘**”), 管理员可以选择“**通过**”或者“**驳回**”该帖子。只有管理员审核通过的帖子才会进入广场页面。

4.7 发帖界面

标题:

请输入标题

日期:

开始日期 ~ 结束日期

科目:

请输入或选择科目

地址:

请选择地址

请输入详细地址

联系方式:

电话号码 (必填)

电子邮箱 (选填)

详情:

可以在这里描述你的具体要求，薪资水平，学生的学习情况等

提交

保存

学生（家教）用户可以通过在侧边栏选择“我要招聘（我要求聘）”跳转到发帖界面。在这个页面用户可以填写自己的基本信息以及需求，并可以对已填写内容进行保存和提交。提交后的帖子经管理员审核通过后便会出现在广场界面。

4.8 学生学习界面

我的帖子

我的家教

我的待办

我的学习资料

学生用户可以通过点击左边侧边栏“我的学习”进入学生学习界面。在该界面用户可以通过切换标签页进入“我的帖子”“我的家教”“我的待办”“我的学习资料”四个小标签页。

4.8.1 我的帖子

公告广场我的学习家教主页帖子详情

首页 / 我的学习

我的学习

我的帖子

我的家教

我的待办

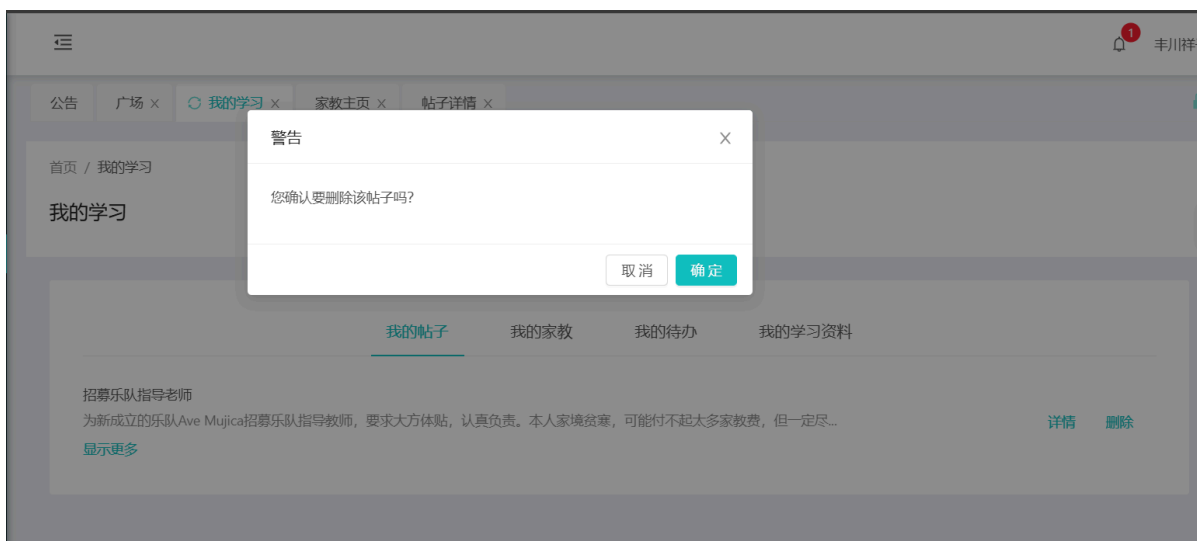
我的学习资料

招募乐队指导老师

为新成立的乐队Ave Mujica招募乐队指导教师，要求大方体贴，认真负责。本人家境贫寒，可能付不起太多家教费，但一定尽...

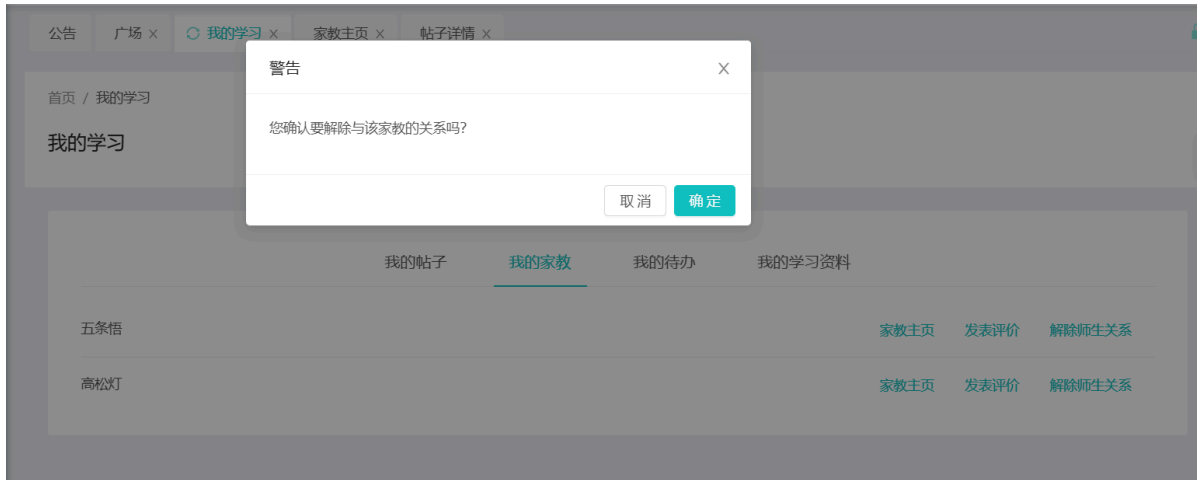
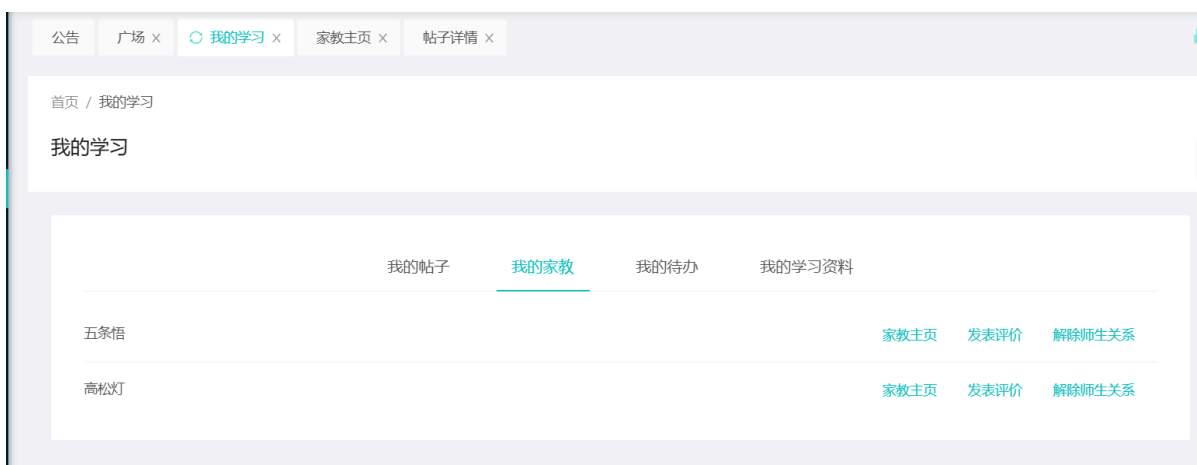
详情 删除

显示更多

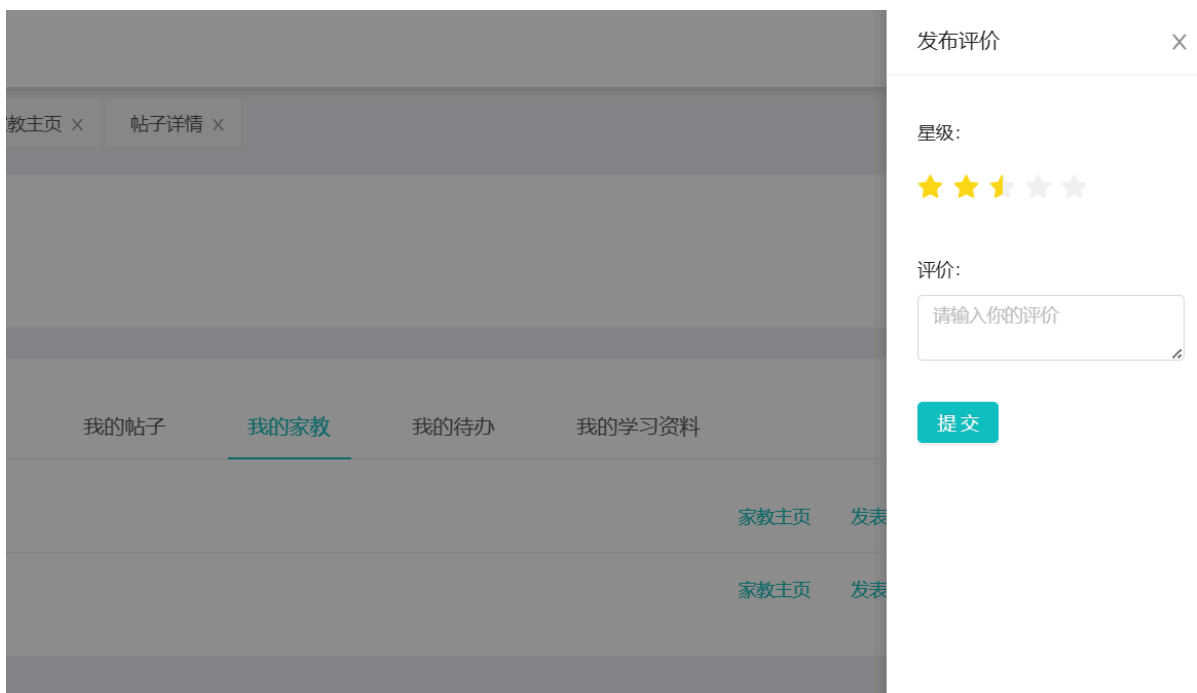


在“我的帖子”小标签页中，用户可以浏览自己发布过的帖子，选择查看帖子详情或者删除帖子。

4.8.2 我的家教



在“我的家教”小标签页中，学生用户可以查看自己的家教，浏览家教主页，对家教发表评价或者解除师生关系。



在用户点击“发表评价”后，页面右侧将会弹出抽屉以供用户填写评价具体信息。用户提交的评价将会出现在对应家教的个人主页上。

4.8.3 我的待办



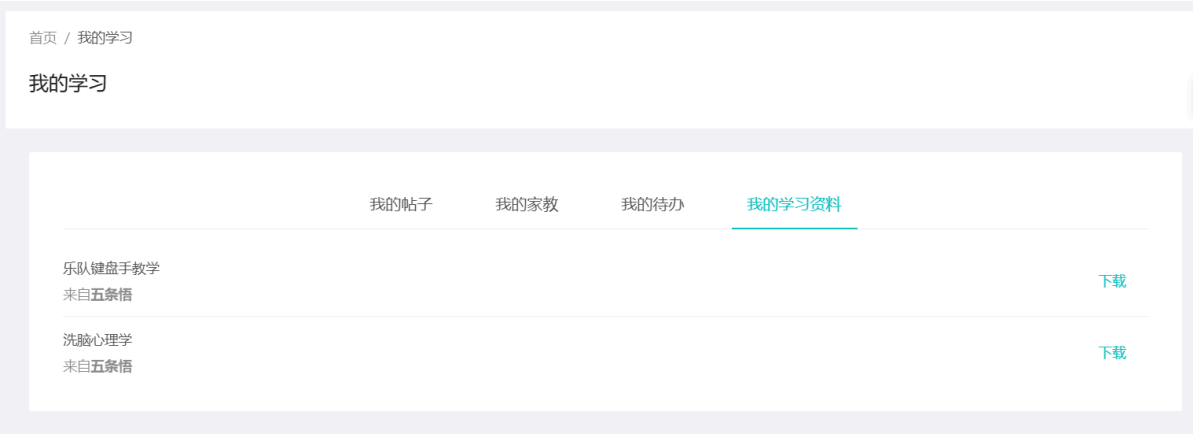
在“我的待办”小标签页中，用户可以浏览自己的待办事项，并选择接受或者拒绝。



待办事项中的蓝色部分为可跳转链接。以上图为例，当用户点击红框后，页面将分别跳转到对应的帖子详情页面和

对应的家教主页。

4.8.4 我的学习资料

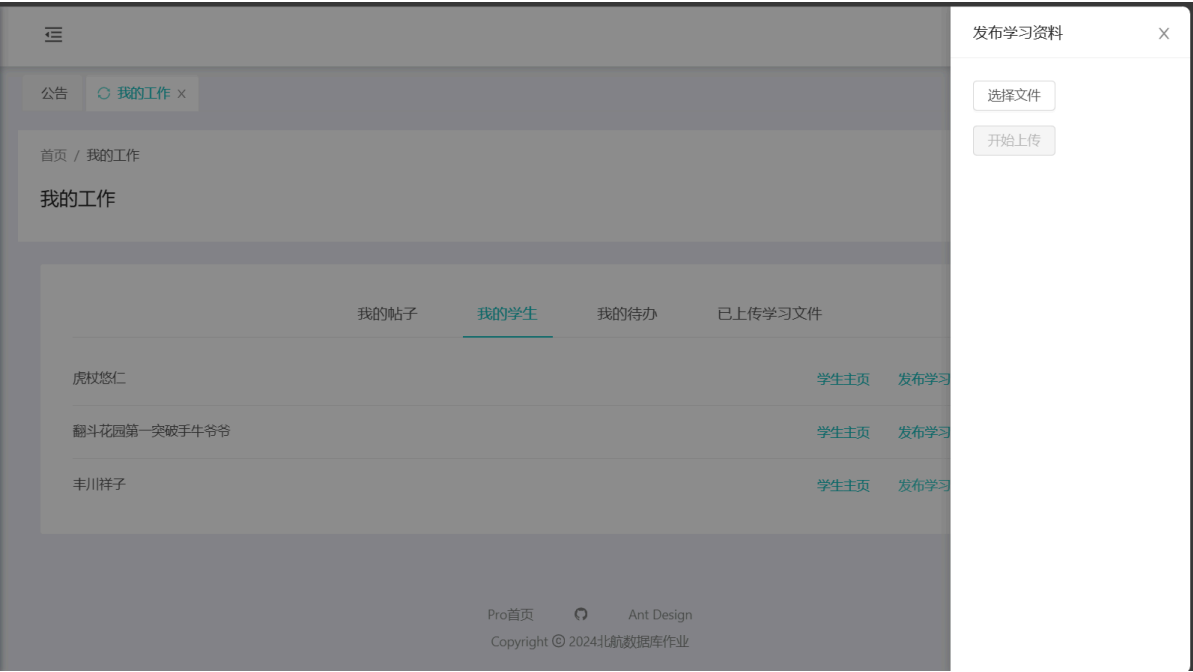


在"我的学习资料"小标签页中，学生用户可以查看家教发布的学习资料，并选择进行下载。

4.9 家教工作界面

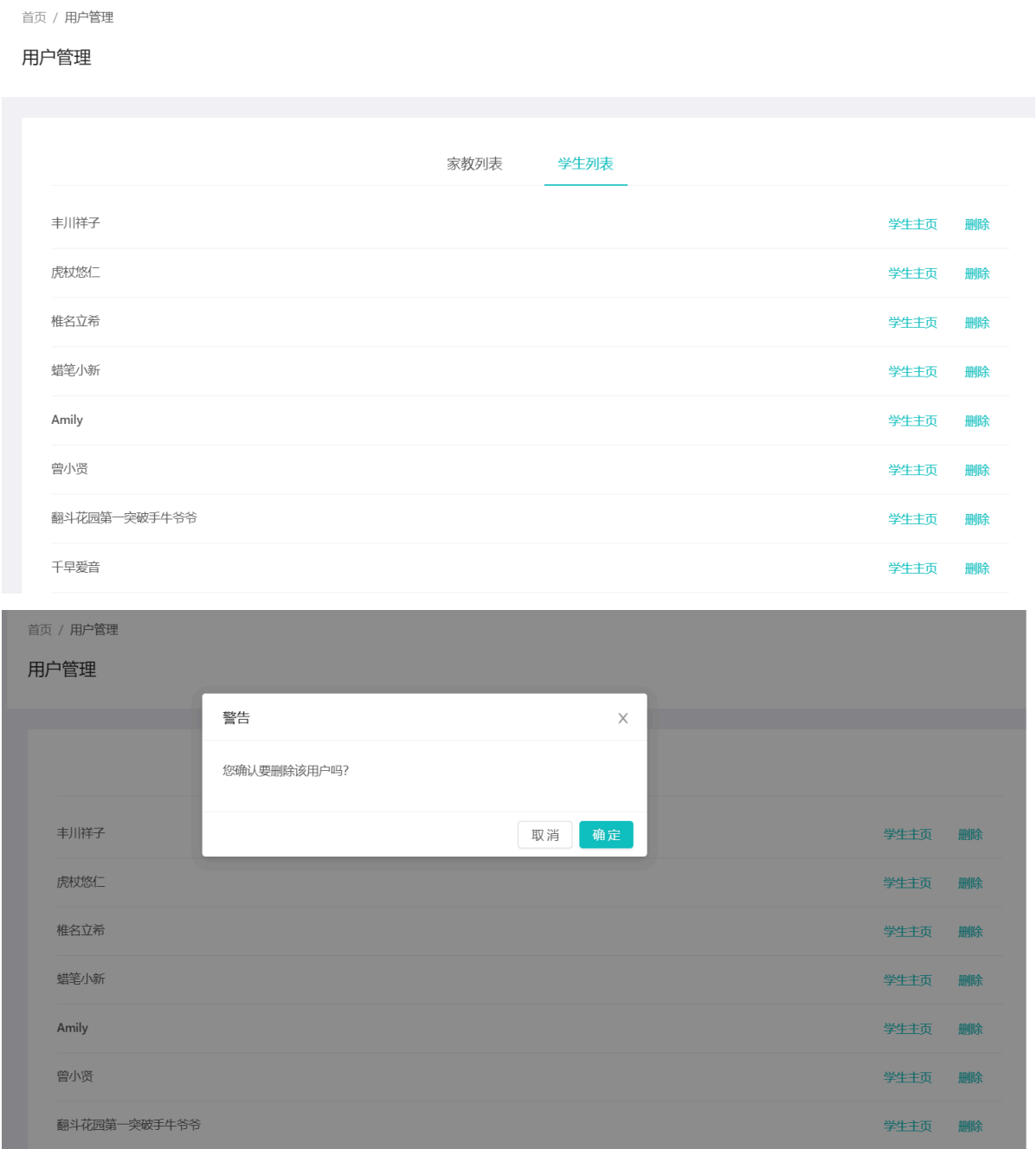


家教的“我的工作”界面与学生的“我的学习”界面界面大致相同，区别在于家教在“我的学生”标签页下存在“发布学习资料”按钮。



在该界面家教可以填写需要上传学习资料，以供学生进行下载。

4.10 用户管理



在用户以管理员身份登入后，用户可以点击侧边栏的“**用户管理**”按钮进入“用户管理”界面。在该界面，管理员可以查看所有用户的信息并删除用户（将用户加入黑名单）。

五、组员大作业总结

个人总结_张哲（组长）

我在这次数据库大作业中和张鑫同学共同负责前端的实现。我主要负责管理员页面的实现和部分美化工作，以及和周泽同同学共同负责最终的展示汇报。

首先我想谈谈在历时一整个学期的大作业完成中，我个人的一些收获。首先就是对于web开发前端设计的知识的学习，前端涉及html、css、javaScript、vue的方面的知识，知识面要求很广。所幸张鑫同学承担起了底层架构的实现，我只需要在他的基础上进行页面的添加就可以，不过我也对这些知识有了较深的理解。除此之外我还学习mysql语言的一些知识，这在链接GaussDB时起到了帮助。有了一定对前端开发的基础认识，我认为在下学期的软工学习中，我可以更加得心应手，并继续增强我的前端知识学习。最后我深刻认识到了大作业完成过程中团队合作的重要性，只有前后端及时沟通，才能避免很多bug

的产生，也会提高整体开发的效率，在完成自己任务的同时，也要考虑如何为另一部分的同学减轻任务，这就要求我们仔细设计接口。

总而言之，这学期的数据库大作业具有挑战但又充满收获，我感到十分自豪和快乐。

个人总结_周泽同

我在这次数据库大作业中承担后端的实现。由于张鑫同学提出想要写前端的想法，我和张哲本来打算一起进行后端的实现。后来发现前端的工作量似乎更大，因此张哲同学决定去和张鑫一起写前端，我来承担后端的工作。后端使用django框架进行开发，主要负责数据模型的设计和前端api的实现。除此之外，我还负责了最终答辩的演讲。

最开始写后端的时候，我并没有网页开发的经验，也没有接触过django框架，因此需要自学相关知识。不过付出了一定学习成本后，我发现django的使用还是非常方便的，与一般的python代码编写过程基本类似，多出的数据库迁移等操作也只需要按部就班地完成即可。令我印象比较深刻的是数据模型的设计。出于我个人的工作习惯，我后端的基本框架搭建在学期刚开始时就已经逐步进行了，当时理论课尚未讲到数据库设计部分，因此我数据模型的设计完全是靠感觉，没有经过严谨细致的思考。后来随着需求的逐步调整和细化，以及实验对于关系模式3NF以上的要求，我不得不对数据模型进行了多次调整。如果模型经历了大改，django的自动迁移可能无法完成冲突的消除，因此我还进行了几次删掉本地数据库重新构建的过程。不过好在经过不断的打磨，我们数据模型的设计总体还是比较简洁合理的，也全部达到了BCNF。

由于后端负责需求的具体实现，系统的bug大多会在我这里测试发现。通过打印、调试、F12开发者工具等手段，我对bug进行“分锅”，将前端的bug反映给前端的同学进行处理，后端的bug则自己进行修复。这里需要感谢jqz同学，在我们系统大致完工后义务担任我们的“测试员”一职，发现了许多或许“刁钻”但又十分合理的bug，让我们进行了不情愿但又快乐的返工。经过不断地“磨”，我们的系统也臻于完善了。

总而言之，言而总之，大作业也是有惊无险地结束了。有了这次的开发经历，或许我也逐渐从一个计算机专业的学生，变成了一个小小的菜鸟“程序员”吧。

个人总结_张鑫

本人在本次数据库大作业中与张哲同学共同承担前端的实现。个人工作具体体现为对“学生”以及“家教”两种身份的页面以及相应逻辑的设计、服务器的租借、以及存储容器的实现。除此之外，本人还负责大作业的中期汇报。

从最开始敲定选题到完成整个任务，总共历时三个半月。我出于“想要走出舒适圈，挑战一下未知的领域”的想法，自告奋勇地接下了“前端设计”的任务。结果，我一下子就被应当学习的知识量惊呆了：html、JavaScript、css、Vue，以及路由设置、axios通信、组件设计.....毫无疑问，数据库作业的难度是巨大的。我甚至可以说，与数据库大作业斗争的三个月是我上大学以来学习知识容量最大的三个月。****但同时，这也是我最充实的三个月****，在面向需求的学习过程中，我对于前端框架越来越熟悉，前端技能越来越娴熟。在最后，我甚至学习了计算机网络的部分内容，成功在自己租借的服务器上实现了MinIO存储容器。

团队合作完成大作业的过程，更是让我明白了团队合作的重要性：每个人不仅仅要对自己的部分负责，还要注重与负责下一环节的组员的沟通。我已经不记得有多少次因为api文档描述不清或者前端测试强度不足导致对接出现bug，进而导致整个项目的停摆。

但万幸的是，最后终究是按时完成了。最后的结果虽然算不上多么出类拔萃，但我已然问心无愧。