《数据库系统原理》大作业

系统实现报告

题目名称： 航吧-北航学生自己的贴吧

学号及姓名： 22373500 孙锐毅（组长）
22371430 黄弘烨
22371157 尹祺霖

2024年 12 月 10日

一、系统结构设计

1. 1 体系结构

本项目采用了前后端分离的架构设计，前端与后端通过 API 接口进行数据交互。具体而言，前端向后端发送请求以获取所需的数据，后端的控制器根据不同的请求类型，执行相应的数据库操作，并将处理后的数据返回给前端。接收到数据后，前端将其渲染到页面上，从而实现了前后端的顺畅交互和数据传递。这种架构的设计使得前端与后端的职责更加明确，增强了系统的可维护性与扩展性。

1.1.1 前端体系结构

本项目采用了vue3框架搭建。有着清晰的模块化结构：

- **组件系统**：
  - 布局组件 (layout/)
  - 对话框组件 (dialogs/)
  - 帖子相关组件 (post/)
  - 用户档案组件 (profile/)
  - 通用组件 (common/)
- **视图层**：
  - 主页视图
  - 用户档案视图
  - 帖子详情视图
  - 标签页视图
- **服务层**：
  - API 服务封装

1.1.2 前端实现环境

技术栈包括：

- **核心框架**：
  - Vue 3 (使用 Composition API)
  - Vite 作为构建工具
- **状态管理**：

    - Vuex
    - 模块化的状态管理（如主题管理）
- **UI组件库**：
    - Element Plus
    - MDI 图标库

核心环境与工具

- **开发服务器**：
    - Vite 开发服务器
    - 端口：8080
    - 代理配置支持
- **构建工具**：
    - Vite
    - Rollup
- **样式处理**：
    - CSS 变量系统

需要导入的包：

```
# 核心依赖
npm install vue@^3.5.12
npm install vite@^6.0.3
npm install vuex@next
npm install vue-router@^4.4.5
npm install axios@^1.7.7

# UI组件
npm install element-plus@^2.9.0
npm install @mdi/font@^7.4.47

# Markdown相关
npm install markdown-it@^14.1.0
npm install highlight.js@^11.10.0

# 开发依赖
```

npm install -D @vitejs/plugin-vue@^5.2.1

npm install -D rollup@^4.24.3

1.1.3 后端体系结构

本系统采用Django后端框架，主要结构由url.py解析路由并分发请求给views.py中的具体方法，由views.py中的方法接受前端传入的Json数据和文件，views.py根据传入信息拿到目标数据并以Json形式传回。

1.1.4 后端实现环境

- 数据库：GaussDB
- 后端框架Django 5.1.2
- 启动命令

python manage.py runserver

1.2 功能结构

用户模块
1. 发布帖子到各个模块中，其他用户可以在帖子下发回复。
2. 注册、登录账号。
3. 在帖子下发布评论。
4. 修改个人信息，如头像、邮箱、用户名、密码。
5. 查看自己的帖子。

帖子模块
1. 用户可以发布帖子，帖子格式为Markdown，支持图片和代码块
2. 用户可以对帖子进行评论

评论模块
1. 可以对帖子进行评论，此为一级评论。
2. 评论之后会出现通知

板块
1. 管理员可以发布公告，通知所有用户一些重要事件。
2. 不同板块之间有不同的颜色与内容，丰富内容。

二、数据库基本表的定义

2.1 用户表

属性名	中文	数据类型	备注
user_id	ID	bigint	主键
user_name	用户名	varchar	
user_email	用户邮箱	varchar	
user_password	用户密码	varchar	
user_student_id	用户学号	bigint	
user_experience	用户经验	bigint	
user_sign_date	注册时间	date	
user_birthday	生日	date	
user_uploaded	头像上传状态	int	
user_user_role	用户角色（管理员等）	int	
user_post_cnt	发表贴数	int	
user_info_cnt	提示消息数	int	
user_introducntion	用户简介	text	

2.2 帖子表

属性名	中文	数据类型	备注
post_id	ID	bigint	主键
post_title	帖子标题	varchar	
post_content	帖子内容	text	
post_tag_id	帖子标签	bigint	外键
post_heat	帖子热度	int	
post_time	帖子创建时间	date	
post_user_id	帖子作者ID	bigint	外键
post_isTop	是否置顶	bool	

2.3 一级评论表

属性名	中文	数据类型	备注
FLC_id	ID	bigint	主键
FLC_content	评论内容	text	
FLC_time	评论创建时间	date	
FLC_post_id	评论的帖子ID	bigint	外键
FLCuser_id	评论发布者ID	bigint	外键

2.4 图片表

属性名	中文	数据类型	备注
PC_id	ID	bigint	主键
PC_path	图片路径	varchar	
PC_category	类型，贴图/用户图	int	
PC_author_id	上传者id	bigint	外键

2.5 标签表

属性名	中文	数据类型	备注
LB_id	ID	bigint	主键
LB_tag_name	频道名	varchar	

2.6 通知表

属性名	中文	数据类型	备注
IF_id	ID	bigint	主键
IF_content	通知内容	text	
IF_receiver_id	收到内容的id	bigint	外键
IF_sender_id	发送内容的id	bigint	外键

三、系统重要功能实现方法

3.1触发器的设计与实现说明

在论坛中帖Post子包含了依附于他的FirstLayerComment评论，当删去帖子的时候需要将他的所有评论一同删除

触发器函数

```
CREATE OR REPLACE FUNCTION delete_related_comments()
RETURNS TRIGGER AS $$
BEGIN
   DELETE FROM comments WHERE post_id = OLD.post_id;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

创建触发器

```
CREATE TRIGGER delete_comments_before_post_delete
BEFORE DELETE ON posts
FOR EACH ROW EXECUTE FUNCTION delete_related_comments();
```

3.2存储过程的设计和实现说明

使用Django中的save方法进行存储数据

3.2.1 用户模块

注册

- 请求：传入username,password,email,student_id,传出code,id
- 后端看这个学号是否有人注册了来返回code和随机生成的id

```
def studentRegistration(request):
   res = {"code": 1, "message": "", "data": None,"id": -1}
   if request.method == 'POST':
      try:
         data = JSONParser().parse(request)
         user_id = generate_unique_user_id()
         user_name = data.get("username")
         user_password = data.get("password")
         user_email = data.get("email")
         user_student_id = data.get("student_id")
         user_experience = 0
         user_sign_date = datetime.now().date()
         user_birthday = datetime(2024, 1, 1)
         # 头像上传状态先默认为0
         user_uploaded = 0
         # 用户状态先默认为0
```

```
            user_role = 0
            user_post_cnt = 0
            user_info_cnt = 0
            user_introduction = "快来填写简介吧"
            # hash存储密码更安全
            user = User(user_id=user_id, user_name=user_name, user_password=user_password,
user_email=user_email,
                        user_student_id=user_student_id, user_experience=user_experience,
                        user_sign_date=user_sign_date, user_birthday=user_birthday,
user_uploaded=user_uploaded,
                        user_role=user_role,
                        user_post_cnt=user_post_cnt,
user_info_cnt=user_info_cnt,user_introduction=user_introduction)
            PC_id = generate_unique_picture_id()
            # 默认路径
            PC_path =
"https://img0.baidu.com/it/u=1864154549,3150614661&fm=253&fmt=auto&app=138&f=JPEG?
w=285&h=285"
            # 图片种类默认为0
            PC_category = 0
            PC_author_id = user
            picture = Picture(PC_id = PC_id, PC_path = PC_path, PC_author_id = PC_author_id,
PC_category = PC_category)
            if User.objects.filter(user_student_id = user_student_id).exists():
                res["code"] = -1
                res["message"] = "学号已被注册"
            else:
                res["code"] = 1
                res["message"] = "注册成功"
                res["id"] = user_id
                user.save()
                picture.save()
        except Exception as e:
            res["code"] = -1
            res["message"] = "服务器错误：用户创建失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

登录

- 请求：传入student_id,password,传出code,path,username,bio,date
- 后端根据传入的学号来看密码是否正确，正确的话传出相应的信息供前端展示

```
def studentLogin(request):
    res = {"code": 1, "message": "", "data": None,"id": -1,"path": None,"username":
None,"bio":None,"date":None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            print(data)
            user_student_id1 = data.get("student_id")
            user_password = data.get("password")
            if User.objects.filter(user_student_id=user_student_id1).exists():
```

```
            # 进行hash比较登录更加安全
            print("!!")
            if user_password == User.objects.get(user_student_id=user_student_id1).user_password:
                print("b")
                res["code"] = 1
                res["message"] = "登陆成功"
                user = User.objects.get(user_student_id=user_student_id1)
                picture = Picture.objects.get(PC_author_id = user)
                res["id"] = user.user_id
                print(user.user_id)
                res["bio"] = user.user_introduction
                print(user.user_introduction)
                res["username"] = user.user_name
                print(user.user_name)
                res["date"] = user.user_sign_date
                print(user.user_sign_date)
                res["path"] = picture.PC_path
                print(picture.PC_path)
            else:
                res["code"] = -1
                res["message"] = "登陆失败，密码不正确"
        else:
            res["code"] = -1
            res["message"] = "登录失败，不存在这个学号的用户"
    except Exception as e:
        res["code"] = -1
        res["message"] = "服务器错误：用户创建失败" + str(e)
else:
    res["code"] = -1
    res["message"] = "请使用POST方法"
return JsonResponse(res)
```

创建帖子

- 传入title,content,user_id,image,传出code
- 因为前端支持了markdown，所以在传入后端的帖子的content中包含了临时地址，后端将图片文件数组上传到图床拿到url按照正则表达式的匹配文法替换content中的临时地址

```
def createPost(request):
    res = {"code": 1, "message": "", "data": None, "heat": 1,"id": -1,"created_at":
datetime.now().date()}
    if request.method == 'POST':
        try:
            data = request.POST
            post_id = generate_unique_post_id()
            post_title = data.get("title")
            post_content1 = data.get("content")
            post_user_id = data.get("user_id")
            post_images = []
            images = request.FILES.getlist("image[]")
            s = []
            for image1 in images:
                headers = {
                    "Authorization": "RoQRscR3iQAQQ4aAgPxaJuEzZWgDn3b3"
```

```
            }
            files = {
                'smfile': (image1.name, image1.file)  # 发送文件
            }
            response = requests.post('https://smms.app/api/v2/upload', headers=headers, files=files)
            response_data = response.json()
            if response_data.get('code') == 'success':
                data1 = response.json()
                s.append(data1['data']['url'])
            elif response_data.get('code') == 'image_repeated':
                data1 = response.json()
                s.append(data1['images'])
            else:
                print("b!!!")

        post_label = 'aaa'
        post_isTop = False
        user = User.objects.get(user_student_id=post_user_id)
        LB_tag_name = data.get("section")
        if Label.objects.filter(LB_tag_name=LB_tag_name).exists():
            label = Label.objects.get(LB_tag_name=LB_tag_name)
        else:
            LB_id = generate_unique_label_id()
            label = Label(LB_tag_name=LB_tag_name, LB_id = LB_id)
            label.save()
        post_heat = 1
        res["heat"] = post_heat
        post_time = localtime(timezone.now())
        post_content = modifyContentPicture(post_content1,s)
        post = Post(post_id=post_id, post_title=post_title, post_content=post_content, post_tag_id
= label,
                    post_heat = post_heat, post_time = post_time,post_user_id = user, post_isTop =
post_isTop,
                    post_label = post_label)
        post.save()
        for image in post_images:
            PP_id = generate_unique_postpicture_id()
            PP_path = image
            PP_post_id = post
            postpicture = PostPicture(PP_id=PP_id, PP_path=PP_path, PP_post_id=PP_post_id)
            postpicture.save()

        PL_id = generate_unique_postandlabel_id()
        PL_tag_id = label
        PL_post_id = post
        postandlabel = PostAndLabels(PL_id=PL_id, PL_tag_id=PL_tag_id,
PL_post_id=PL_post_id)
        postandlabel.save()
        # 用户帖子数+1
        user.user_post_cnt = user.user_post_cnt + 1
        # 经验+3
        user.user_experience = user.user_experience + 3
        user.save()
        res["code"] = 1
        res["message"] = "成功创建帖子"
        res["id"] = post_id
        res["created_at"] = post_time
```

```
        except Exception as e:
            res["code"] = -1
            res["message"] = "服务器错误：用户创建失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

修改密码

- 请求：传入old_password,new_password，传出code
- 后端根据旧密码是否正确返回code，正确的话存入新的密码

```
def modifyPassword(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            old_password = data.get("old_password")
            new_password = data.get("new_password")
            user = User.objects.get(user_student_id = user_id)
            if old_password == user.user_password:
                user.user_password = make_password(new_password)
                user.save()
                res["code"] = 1
                res["message"] = "修改成功"
            else:
                res["code"] = -1
                res["message"] = "修改失败，旧密码错误"
        except Exception as e:
            res["code"] = -1
            res["message"] = "服务器错误：用户创建失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

删除帖子

- 请求：传入user_id,post_id,传出code
- 后端根据传入的post_id找到对应的帖子，看这个帖子的author_id(创作者)的id是否为user_id来返回code

```
def deletePost(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            post_id = data.get("post_id")
            post = Post.objects.get(post_id=post_id)
            user = User.objects.get(user_student_id = user_id)
            if post.post_user_id == user:
                user.user_post_cnt = user.user_post_cnt - 1
```

```
            user.save()
            post.delete()
            res["code"] = 1
            res["message"] = "帖子删除成功"
        else:
            res["code"] = -1
            res["message"] = "帖子删除失败，发帖用户不对"
    except Exception as e:
        res["code"] = -1
        res["message"] = "帖子删除失败,帖子id不存在" + str(e)
else:
    res["code"] = -1
    res["message"] = "请使用POST方法"
return JsonResponse(res)
```

创建评论

- 请求：传入content,user_id,post_id,image,传出code
- 因为前端支持了markdown，所以在传入后端的评论的content中包含了临时地址，后端将图片文件数组上传到图床拿到url按照正则表达式的匹配文法替换

```
def createFirstLayerComment(request):
    res = {"code": 1, "message": "", "data": None,"comment_id": -1,"created_at":
datetime.now().date(),"heat": 1,"path": None}
    if request.method == 'POST':
        try:
            data = request.POST
            content = data.get("content")
            user_id = data.get("user_id")
            post_id = data.get("post_id")
            images = request.FILES.getlist("image[]")
            s = []
            for image in images:
                headers = {
                    "Authorization": "RoQRscR3iQAQQ4aAgPxaJuEzZWgDn3b3"
                }
                files = {
                    'smfile': (image.name, image.file)  # 发送文件
                }
                response = requests.post('https://smms.app/api/v2/upload', headers=headers, files=files)
                response_data = response.json()
                if response_data.get('code') == 'success':
                    data1 = response.json()
                    s.append(data1['data']['url'])
                elif response_data.get('code') == 'image_repeated':
                    data1 = response.json()
                    s.append(data1['images'])
                else:
                    print("b!!!")

            user = User.objects.get(user_student_id = user_id)
            post = Post.objects.get(post_id=post_id)
            receiver = post.post_user_id
            IF_id = generate_unique_inform_id()
            IF_content = "你有新的消息"
            IF_receiver_id = receiver
```

```
            IF_sender_id = user
            inform = Inform(IF_id = IF_id,IF_content = IF_content,IF_receiver_id =
IF_receiver_id,IF_sender_id = IF_sender_id)
            inform.save()
            FLC_id = generate_unique_picture_id()
            FLC_time = localtime(timezone.now())
            FLC_content = modifyContentPicture(content,s)
            FLC_post_id = post
            FLC_author_id = user
            # 经验+3
            user.user_experience = user.user_experience + 3
            user.save()
            # 帖子热度+1
            post.post_heat = post.post_heat + 1
            post.save()
            firstlayercomment = FirstLayerComment(FLC_id = FLC_id, FLC_time = FLC_time,
FLC_content = FLC_content, FLC_post_id = FLC_post_id
                                                  , FLC_author_id = FLC_author_id)
            firstlayercomment.save()
            res["code"] = 1
            res["message"] = "成功发表一级评论"
            res["comment_id"] = FLC_id
            res["created_at"] = FLC_time
            res["heat"] = post.post_heat
            res["path"] = Picture.objects.get(PC_author_id = user).PC_path
        except Exception as e:
            res["code"] = -1
            res["message"] = "发布评论失败，该用户不存在" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    print(res["code"])
    return JsonResponse(res)
```

修改邮件

- 请求：传入user_id,newEmail
- 后端根据传入的user_id找到用户并修改他的Email为newEmail

```
def modifyEmail(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            newEmail = data.get("newEmail")
            user = User.objects.get(user_student_id = user_id)
            user.user_email = newEmail
            user.save()
            res["code"] = 1
            res["message"] = "修改邮件成功"
        except Exception as e:
            res["code"] = -1
            res["message"] = "修改邮件失败，用户id不存在" + str(e)
    else:
        res["code"] = -1
```

```
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

修改昵称

- 请求：传入user_id,newUsername
- 后端根据传入的user_id找到用户并修改他的username为newUsername

```
def modifyUserName(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            newUsername = data.get("newUsername")
            user = User.objects.get(user_student_id = user_id)
            user.user_name = newUsername
            user.save()
            res["code"] = 1
            res["message"] = "修改用户名成功"
        except Exception as e:
            res["code"] = -1
            res["message"] = "修改邮件失败，用户id不存在" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

修改简介

- 请求：传入user_id,new_bio
- 后端根据传入的user_id找到用户并修改他的bio为new_bio

```
def modifyIntroduction(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            new_bio = data.get("new_bio")
            user = User.objects.get(user_student_id = user_id)
            user.user_introduction = new_bio
            user.save()
            res["code"] = 1
            res["message"] = "修改简介成功"
        except Exception as e:
            res["code"] = -1
            res["message"] = "修改邮件失败，用户id不存在" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

修改头像

- 请求：传入user_id,image(头像文件)

- 后端根据传入的user_id找到用户并找到picture，即他对应的头像，将图片上传到图床后修改picture的path为图床返回的url

```
def modifyPicture(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = request.POST
            user_id = data.get("user_id")
            image_file = request.FILES.get('image')
            user = User.objects.get(user_student_id = user_id)
            picture = Picture.objects.get(PC_author_id = user)
            # 获取 SM.MS 的 API Token
            api_token = 'RoQRscR3iQAQQ4aAgPxaJuEzZWgDn3b3'

            # 查询图片是否存在
            headers = {
                "Authorization": "RoQRscR3iQAQQ4aAgPxaJuEzZWgDn3b3"
            }
            files = {
                'smfile': (image_file.name, image_file.file)  # 发送文件
            }
            response = requests.post('https://smms.app/api/v2/upload', headers=headers,files=files)
            response_data = response.json()
            if response_data.get('code') == 'success':
                data1 = response.json()
                picture.PC_path = data1['data']['url']
                picture.save()
                res["code"] = 1
                res["message"] = "修改头像成功"
            elif response_data.get('code') == 'image_repeated':
                data1 = response.json()
                picture.PC_path = data1['images']
                picture.save()
                res["code"] = 1
                res["message"] = "修改头像成功"
            else:
                print("b")
                res["code"] = -1
                res["message"] = "获取图床url失败"
        except Exception as e:
            res["code"] = -1
            res["message"] = "修改头像失败，用户id不存在" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    print(res["code"])
    return JsonResponse(res)
```

拿到全部帖子的简要信息

- 请求：传入section,sort,传出data数组,包含了post_id,post_title,post_heat,post_time,username,avatar

- 后端根据前端传入的版块和排序方式来返回相应的信息,展示了帖子的标题，创建时间，板块，作者昵称和头像，热度

```
def getPostBriefInfo(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            part = data.get("section")
            sort = data.get("sort")
            if part == "all":
                if sort == "new":
                    posts = Post.objects.order_by("-post_time").values('post_id', 'post_title', 'post_heat',
                                                       'post_time', 'post_user_id')
                    post_list = list(posts)
                    for post in post_list:
                        post['post_time'] = timezone.localtime(post['post_time']).strftime('%Y-%m-%d
%H:%M:%S')
                        post_user_id = post['post_user_id']
                        post['username'] = User.objects.get(user_id=post_user_id).user_name
                        post['avatar'] = Picture.objects.get(PC_author_id=post_user_id).PC_path
                    res["data"] = post_list
                else:
                    posts = Post.objects.order_by("-post_heat").values('post_id', 'post_title', 'post_heat',
                                                       'post_time', 'post_user_id')
                    post_list = list(posts)
                    for post in post_list:
                        post['post_time'] = timezone.localtime(post['post_time']).strftime('%Y-%m-%d
%H:%M:%S')
                        post_user_id = post['post_user_id']
                        post['username'] = User.objects.get(user_id=post_user_id).user_name
                        post['avatar'] = Picture.objects.get(PC_author_id=post_user_id).PC_path
                    res["data"] = post_list
            else:
                if sort == "new":
                    posts = Post.objects.order_by("-post_time").values('post_id', 'post_title', 'post_heat',
                                                       'post_time', 'post_user_id', 'post_tag_id')
                    post_list = []
                    for post in posts:
                        post['post_time'] = timezone.localtime(post['post_time']).strftime('%Y-%m-%d
%H:%M:%S')
                        post_user_id = post['post_user_id']
                        post['username'] = User.objects.get(user_id=post_user_id).user_name
                        post['avatar'] = Picture.objects.get(PC_author_id=post_user_id).PC_path
                        LB_id = post['post_tag_id']
                        if Label.objects.get(LB_id=LB_id).LB_tag_name == part:
                            post_list.append(post)
                    res["data"] = post_list
                else:
                    posts = Post.objects.order_by("-post_heat").values('post_id', 'post_title', 'post_heat',
                                                       'post_time', 'post_user_id', 'post_tag_id')
                    post_list = []
                    for post in posts:
                        post['post_time'] = timezone.localtime(post['post_time']).strftime('%Y-%m-%d
%H:%M:%S')
                        post_user_id = post['post_user_id']
                        post['username'] = User.objects.get(user_id=post_user_id).user_name
```

```
                post['avatar'] = Picture.objects.get(PC_author_id=post_user_id).PC_path
                LB_id = post['post_tag_id']
                if Label.objects.get(LB_id=LB_id).LB_tag_name == part:
                    post_list.append(post)
            res["data"] = post_list
        except Exception as e:
            res["code"] = -1
            res["message"] = "获取帖子简要信息失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    print(res["data"])
    return JsonResponse(res)
```

拿到一个帖子全部信息

- 请求：传入post_id,传出content,post_title,heat,post_time,username,avatar,section,user_id,picture以及data数组，其中包含了content,reply_time,username,avatar,user_id
- 后端根据前端传入的post_id返回帖子的内容，标题，热度，创建时间，创作者头像url和昵称，板块，还有帖子的评论的数组，包含了时间，内容，评论者昵称和头像url

```
def getPostAllInfo(request):
    res = {"code": 1, "message": "", "data": None,"content": None,"post_title":
None,"heat":None,"post_time":None,"username":None,"avatar":None
          ,"section": None,"user_id": None,"picture":None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            post_id = data.get("post_id")
            post = Post.objects.get(post_id=post_id)
            user = post.post_user_id
            picture = Picture.objects.get(PC_author_id=user)
            label = post.post_tag_id
            res["code"] = 1
            res["content"] = post.post_content
            res["post_title"] = post.post_title
            res["heat"] = post.post_heat
            res["post_time"] = post.post_time
            res["username"] = user.user_name
            res["user_id"] = user.user_id
            res["avatar"] = picture.PC_path
            res["section"] = label.LB_tag_name
            pictureBack = PostPicture.objects.filter(PP_post_id = post).values('PP_path')
            pictureBackList = list(pictureBack)
            for picture1 in pictureBackList:
                picture1['path'] = picture1['PP_path']
            res["picture"] = pictureBackList
            comment_list = FirstLayerComment.objects.filter(FLC_post_id =
post).values('FLC_id','FLC_content','FLC_time','FLC_author_id')
            comment_list1 = list(comment_list)
            for comment in comment_list1:
                comment['id'] = comment['FLC_id']
                comment['content'] = comment['FLC_content']
                comment['reply_time'] = comment['FLC_time']
```

```
            comment_user_id = comment['FLC_author_id']
            user = User.objects.get(user_id=comment_user_id)
            picture = Picture.objects.get(PC_author_id=user)
            comment['username'] = user.user_name
            comment['avatar'] = picture.PC_path
            comment['user_id'] = user.user_id
        res["data"] = comment_list1
    except Exception as e:
        res["code"] = -1
        res["message"] = "获取帖子全部信息失败" + str(e)
else:
    res["code"] = -1
    res["message"] = "请使用POST方法"
print(res["code"])
print(res["post_title"])
print(res["content"])
print(res["username"])
print(res["picture"])
print(res["data"])
return JsonResponse(res)
```

拿到经验值榜单前三

- 请求：传出榜单经验值前三
- 计算方式是发帖+3，评论+3，删帖不扣

```
def getTopThreeUserInfo(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            # data = JSONParser().parse(request)
            users = User.objects.order_by('-
user_experience').values('user_name','user_id','user_experience')
            users_list = []
            for user in users:
                user_id = user['user_id']
                picture = Picture.objects.get(PC_author_id=user_id)
                user['avatar'] = picture.PC_path
                user['username'] = user['user_name']
                user['experience'] = user['user_experience']
                if len(users_list) < 3:
                    users_list.append(user)
            res["data"] = users_list
            print("!!")
            print(res["data"])
        except Exception as e:
            res["code"] = -1
            res["message"] = "拿到经验值前三信息失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

获取用户所有的帖子

- 请求：传出user_id,返回data数组，包含了post_title,post_time,post_id

- 后端根据传入的user_id找到user，并以此在Post中查到所有帖子，返回帖子的创建时间，标题和id

```
def getUserPost(request):
    res = {"code": 1, "message": "", "data": None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            user = User.objects.get(user_student_id=user_id)
            post_list = Post.objects.filter(post_user_id=user).values('post_title','post_time','post_id')
            res["data"] = list(post_list)
        except Exception as e:
            res["code"] = -1
            res["message"] = "获取个人帖子失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

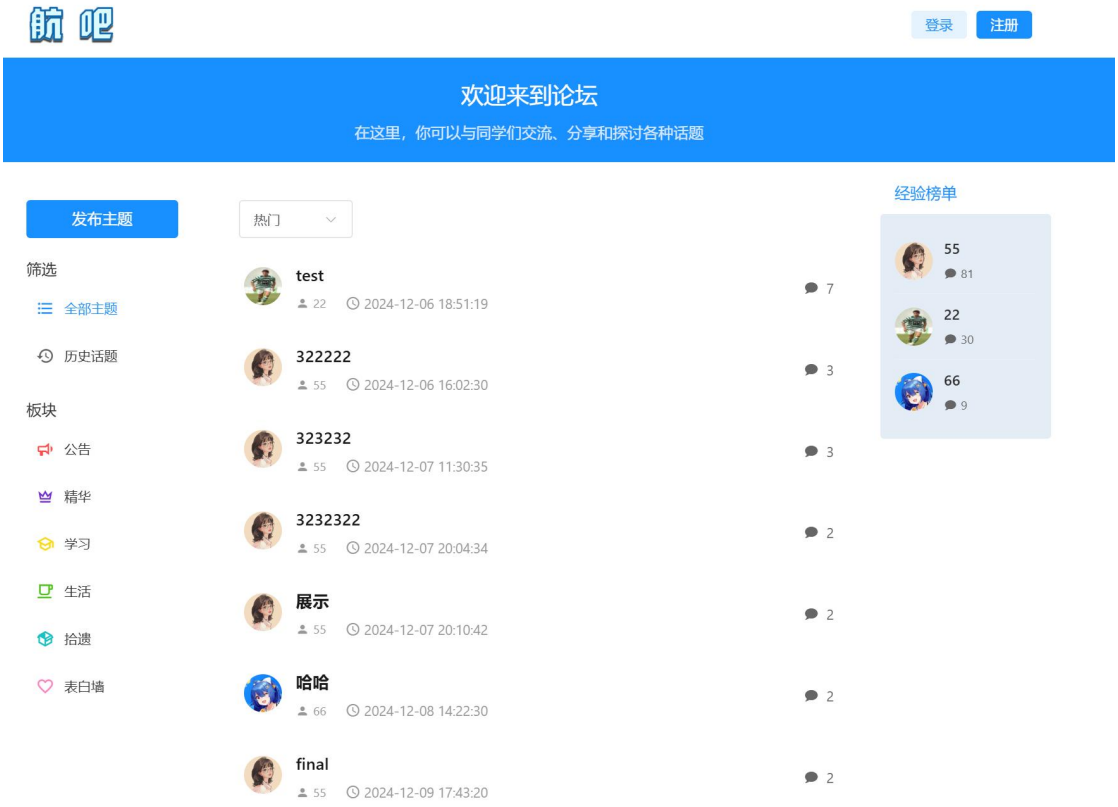
获得用户头像

- 请求：传入user_id,传出avatar
- 后端根据传入的user_id找到user,并返回他的头像的url

```
def getUserAvatar(request):
    res = {"code": 1, "message": "", "data": None,"avatar":None}
    if request.method == 'POST':
        try:
            data = JSONParser().parse(request)
            user_id = data.get("user_id")
            user = User.objects.get(user_student_id=user_id)
            picture = Picture.objects.get(PC_author_id=user)
            res["avatar"] = picture.PC_path
        except Exception as e:
            res["code"] = -1
            res["message"] = "获取个人信息失败" + str(e)
    else:
        res["code"] = -1
        res["message"] = "请使用POST方法"
    return JsonResponse(res)
```

四、系统实现结果

4.1 首页



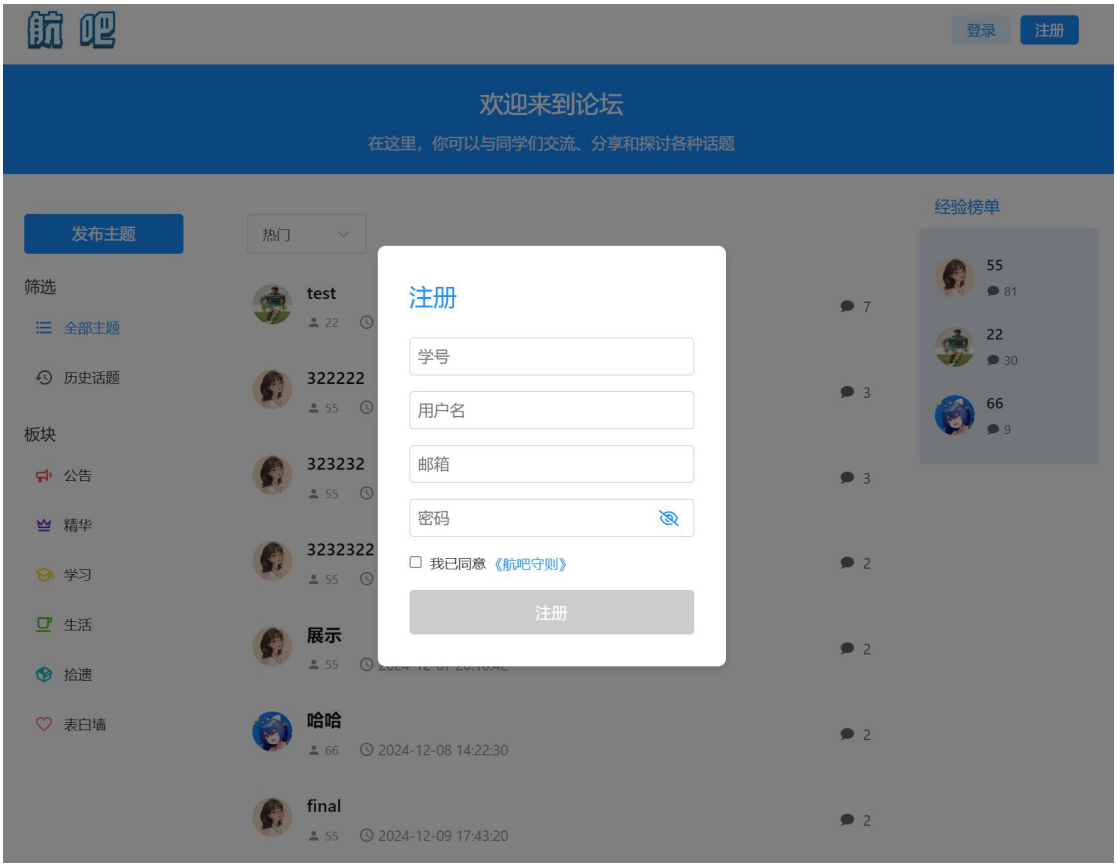
首页顶栏左侧是“航吧”logo，点击可以返回首页，右侧是登录、注册按钮，点击能进行相应操作

左侧功能栏最上方是发布主题按钮，用户点击即可进行帖子的发布。下方是板块选择，用户能根据自己想要的板块来进行选择查看对应的帖子。

中间是帖子列表，用户能在这里查看所有的帖子，点击对应的帖子即可进入查看帖子界面。上方有一个帖子排序选择下拉框，用户能根据自己的需求进行帖子排序的选择。

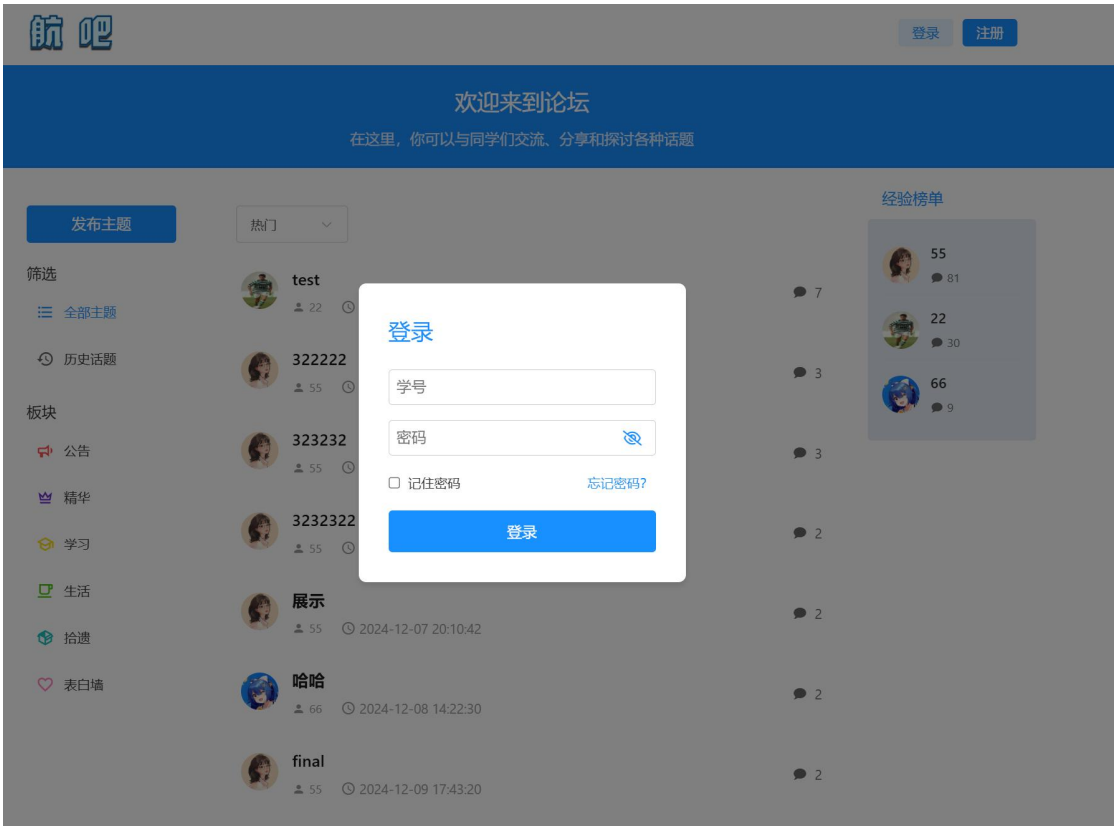
右侧是经验榜单，能在这里查看所有用户中经验值前三的用户

4.1.1 注册



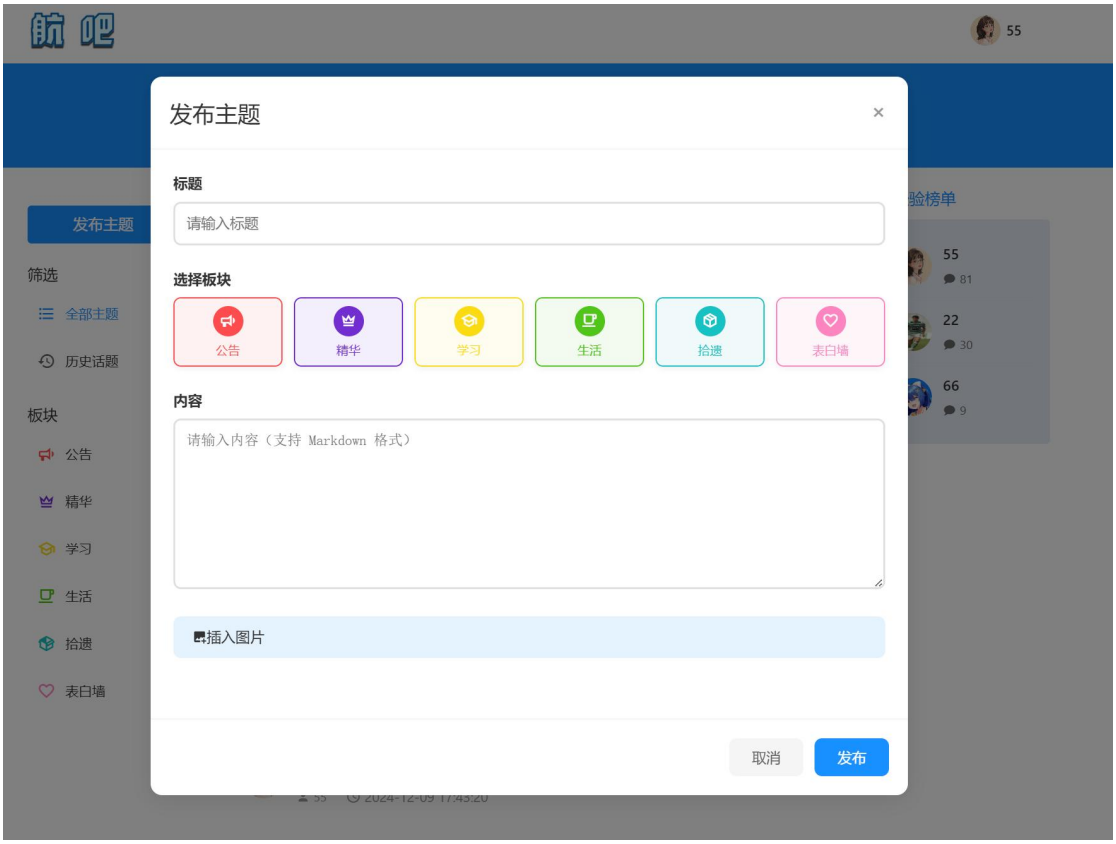
点击注册按钮后，会弹出一个对话框，用户能在填写完毕信息后进行注册，同时必须要同意《航吧守则》

4.1.2 登录



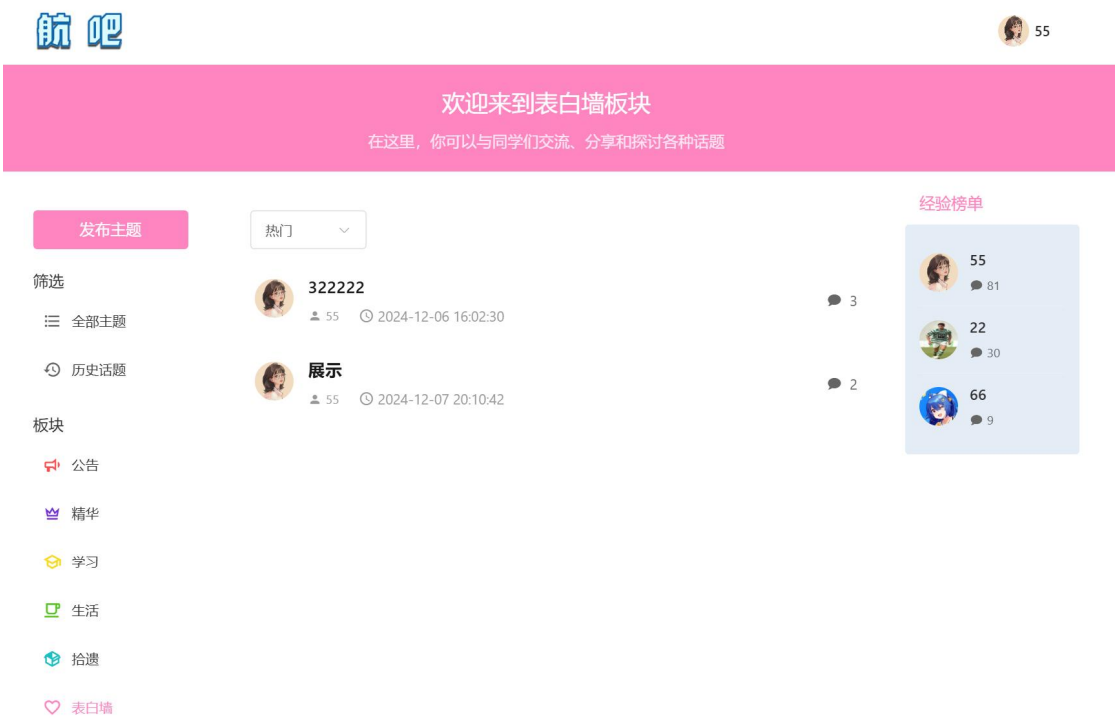
点击登录按钮后，会弹出一个对话框，用户能在输入学号和密码后进行登录

4.1.3 发布帖子



登录后，点击发布主题按钮能进行帖子发布，用户可以进行输入帖子标题，选择发布的板块，帖子内容，插入图片等操作，帖子内容支持markdown格式。填写完毕后点击发布即可发布帖子

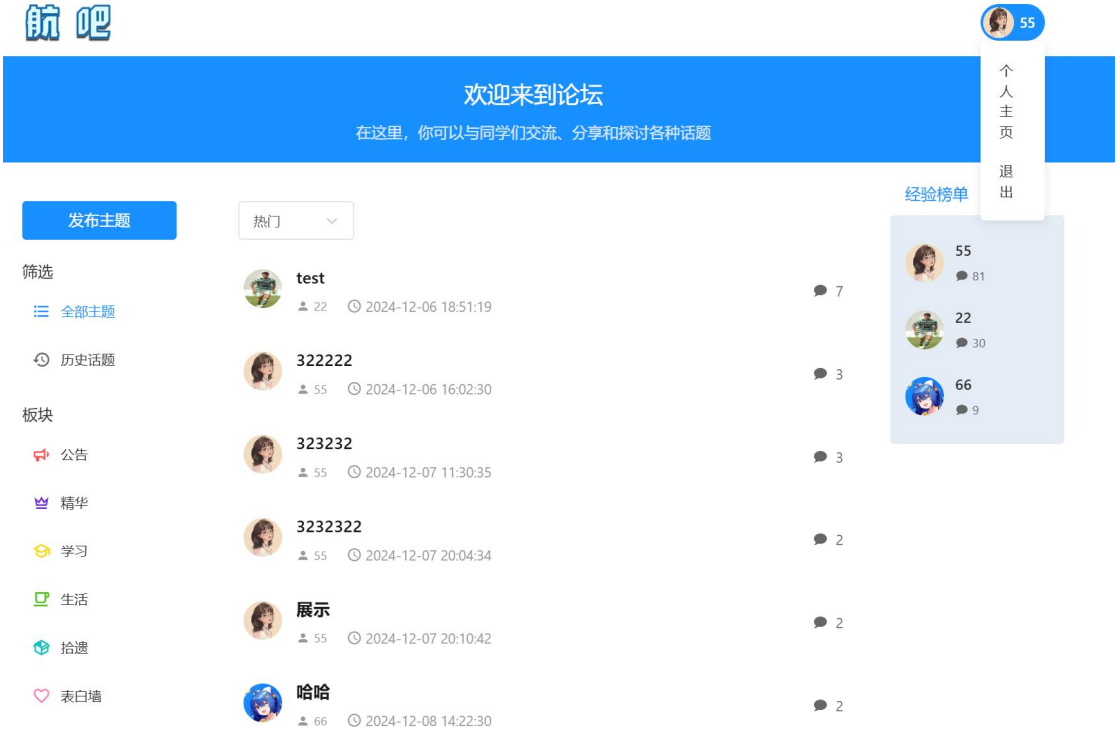
4.1.4 选择板块



通过左边功能栏可以选择查看板块。选择后帖子列表将只出现对应板块的帖子，可以方便用户选择、查看想要关注的内容

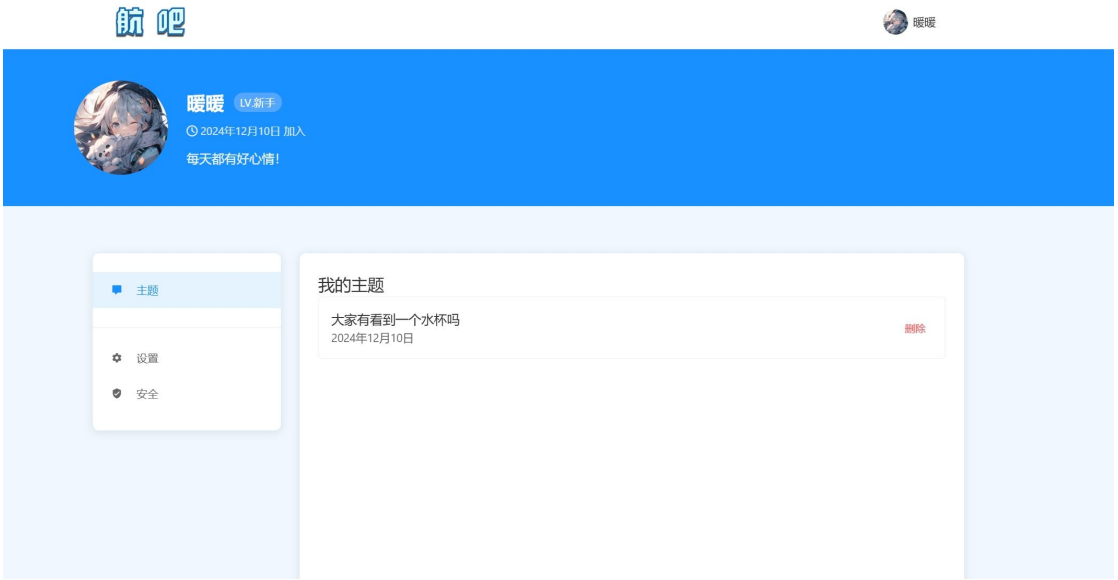
4.2 个人主页

4.2.1 进入个人主页



登录后点击头像会出现下拉框，子这里可以选择进入个人主页，或者退出登录

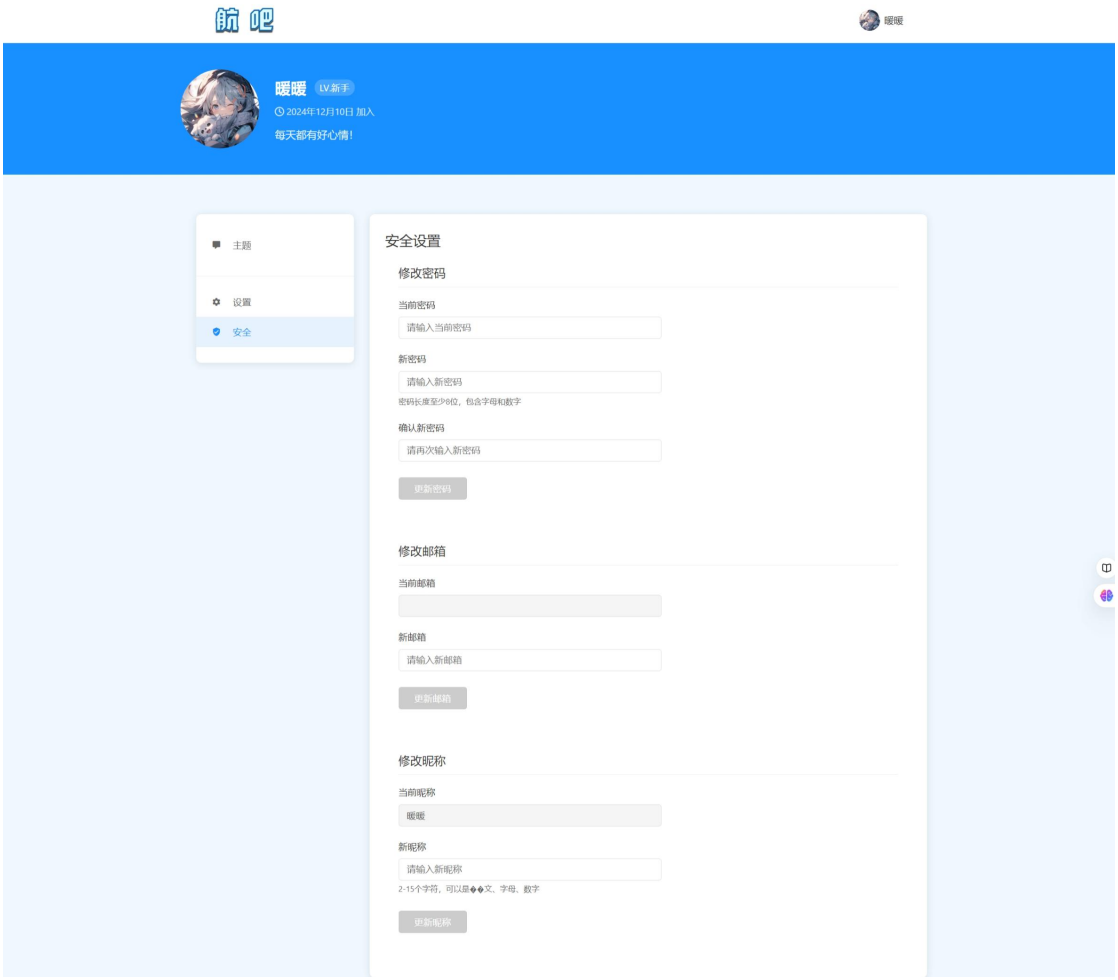
4.2.2 个人主页



个人主页上方是用户个人信息。点击头像可以进行更换，点击简介可以进行修改。

下方默认显示自己已经发布的帖子，可以点击右边的删除按钮删除帖子

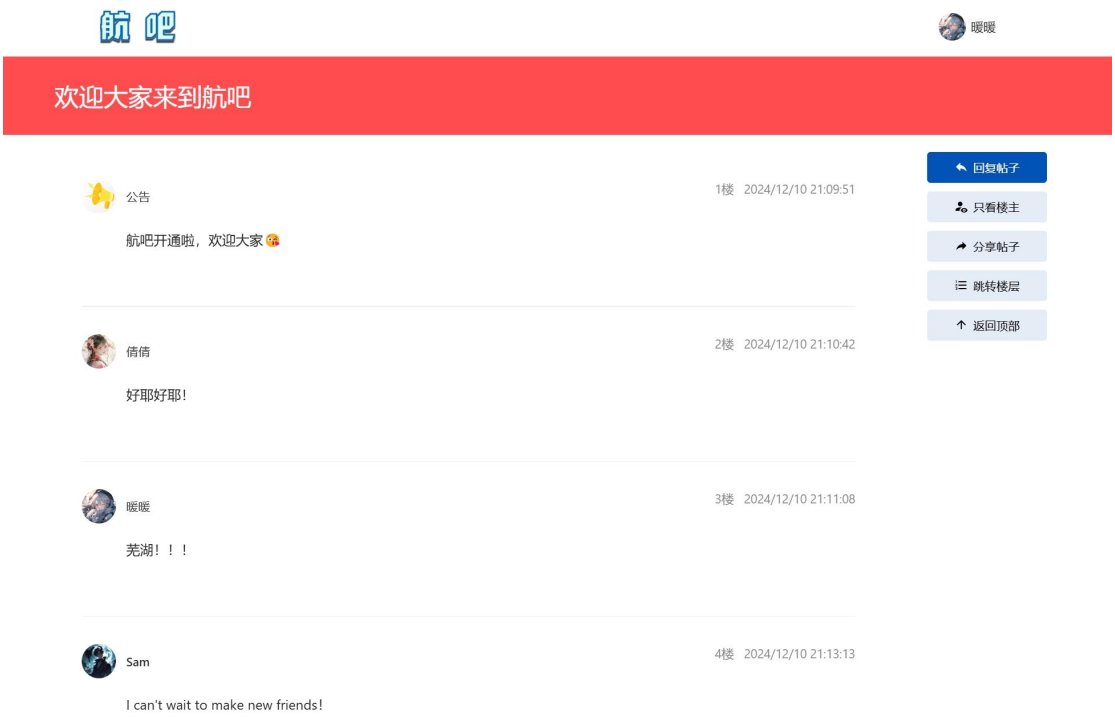
4.2.3 修改信息



在安全界面，可以修改用户自己的密码，邮箱，昵称等内容

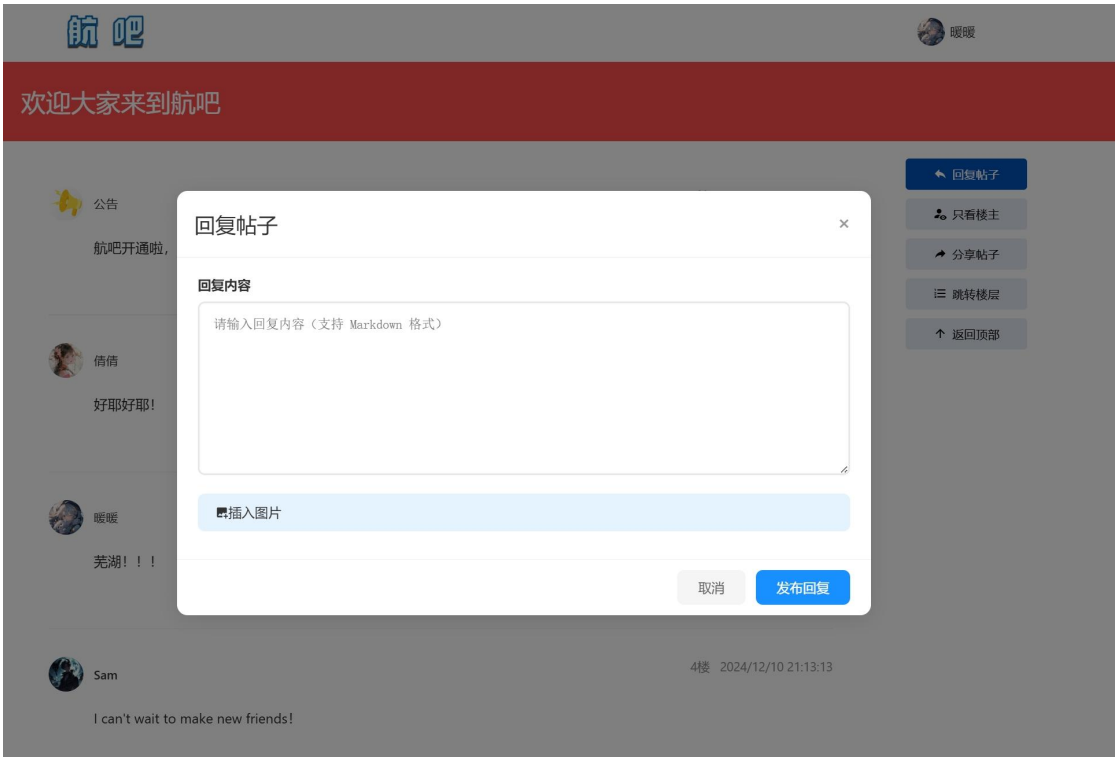
4.3 帖子内容界面

4.3.1 整体样式



帖子界面上方是标题，下面是帖子内容，右侧是功能栏

4.3.2 回复帖子



点击回复帖子按钮能对帖子进行回复，内容填写完后点击发布回复即可

五、组员大作业总结

（小组内每位同学对所承担任务完成情况进行总结，包括任务内容概述、任务的难点以及解决难点的方法、任务完成结果、收获与体会。）

孙锐毅:在这个项目中，我的主要任务包括接口文档的敲定、前后端API通信的工作和前后端联合测试。首先，我负责根据需求文档与前后端开发人员的讨论，制定详细且清晰的接口文档，确保前后端在接口的定义、数据格式、请求方式和返回值等方面达成共识，以便后续的开发和对接工作能够顺利进行。在API通信方面，我不仅要确保前后端的数据能够正确传输，还要保证数据格式、字段和错误处理机制的统一性，避免因为细节上的不一致导致功能出现异常。最后，在完成前后端接口对接后，我与团队一起进行了全面的测试，确保接口能够稳定运行。

在任务执行过程中，遇到了几项较为复杂的挑战。首先，接口文档的准确性是一个难点，特别是在需求不完全明确的情况下，如何预见到所有可能的使用场景并设计出高质量的接口文档，避免后期修改，是一个重要问题。为了克服这一难点，我在文档中尽可能地涵盖了所有可能的请求和返回情况，并与前后端开发人员进行了多轮讨论，确保接口定义没有歧义。其次，前后端API

通信中的兼容性问题也是一个挑战，前端和后端使用不同的技术栈，导致数据格式和传输方式上可能出现差异，需要与组员之间紧密沟通。

通过这次任务，我不仅提升了自己在接口设计、前后端协作和系统测试方面的能力，还深入理解了团队合作和跨部门沟通的重要性。在整个过程中，我体会到，良好的文档和规范化的接口设计能够有效减少开发中的误解和重复工作，而细致的测试和前后端密切的配合则是保证项目顺利交付的关键。通过这些经验，我深刻认识到，技术细节的把控与团队协作的无缝对接是成功完成项目的基础，这些收获对我未来在技术架构和团队合作方面将产生深远的影响。

尹祺霖:在本次作业中我负责了后端部分、需求设计分析、接口文档敲定和前后端API通信和测试。任务的难点是和接口文档的敲定，在需求并不明确的情况下这是不容易定死的，并且需要统一大家的命名习惯。图片上传的部分也是难点，最终在经历了一番研究后选择了图床，并且我在面对烤漆和编译实验的情况下平衡了时间分配，更好的处理了压力，本次开发的过程还是比较愉快的，期待和同学们的下一次合作。

黄弘烨：这次作业中，我负责了前端的工作。搭建整个网页、处理路由跳转、处理核心交互逻辑、美化网页外观与显示效果等等部分。并从中收获到了很多新的知识，积累了许多经验。

开发的过程从来都不是一帆风顺的。总是面临许多挑战。我遇见的困难与bug犹如璀璨星河，数不胜数。我是在同学、网络与现代化工具的帮助下战胜这些困难的。而克服这些技术难题的过程，也令我收获了诸多感悟。

通过这个项目，我深刻理解了组件化开发的重要性。将复杂的页面拆分成可复用的组件不仅提高了代码的可维护性，也大大提升了开发效率。最开始搭建项目时，我的组件划分是高度耦合且不清晰的，在一次次重构中，逐渐将组件拆分，才达到了清晰可调试的地步。

使用 Vuex 进行状态管理让我理解到，在复杂应用中，一个良好的状态管理方案对于保持数据一致性和提高开发效率至关重要。我们项目中间遇见过数次有关刷新和状态记录的bug。原先的登录状态消失了、刷新之后的网页又回到主页等等。合理的状态管理是前端的必修课。

在实现功能的同时，我开始更加注重用户体验的细节，比如添加了加载状态提示，优化了错误处理机制，实现了平滑的动画过渡。

与后端的紧密协作也让我明白了接口文档的重要性，以及如何更好地处理前后端的数据交互。

互联网的技术学习也是一道坎。许多最新的、最好用的技术，在网上的学习资料是难以找到的。而各类组件的css美化技术则是五花八门，同样的效果可能有好几种实现方式，但只有一种实现方式最适合某一个项目。在搭建前端网页的过程中，除了学习到的各种知识，更重要的是我学习知识和提炼知识的能力提升了。

这个项目让我在技术能力和项目管理能力上都有了显著提升。特别是在组件化开发、状态管理、用户体验优化等方面获得了宝贵的经验。同时，也深刻体会到了团队协作和技术文档的重要性。这次的开发体验非常愉快，感谢我的团队!