Project 4: Django博客系统（Blog）

姓名：陈铄涵 **学号：**2024140014 **课程：**Python程序设计 **日期：**2025/12/28

1 项目概述与工作内容

本项目基于Django 5.2框架开发，**完整实现了Project4所有要求**，并进行了多项功能扩展。项目涵盖：**BlogPost数据模型**（包含title、text、date_added、owner字段）、**用户认证系统**（注册/登录/登出）、**文章CRUD操作**（创建、查看、编辑、删除）、**权限保护机制**（仅作者可编辑自己的文章）。在此基础上，额外实现了**标签系统、评论功能、点赞机制、图片上传、全文搜索、用户主页**等高级特性。

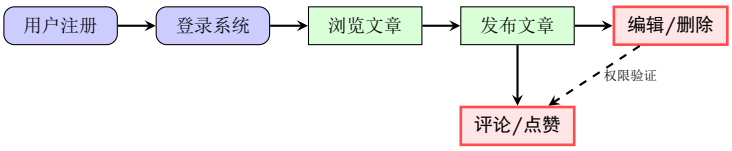


图 1: 项目工作流程图（红框为扩展功能：评论点赞系统）

2 项目架构与系统设计

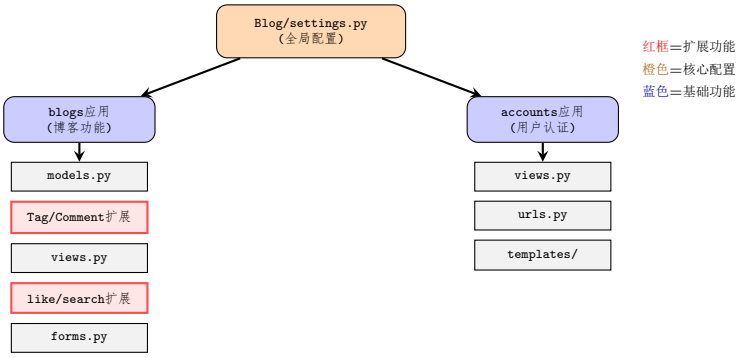


图 2: 项目架构图（红框标注扩展功能模块）

3 核心代码与技术亮点

1. BlogPost模型

```python
class BlogPost(models.Model):
    # === Required fields ===
    title = models.CharField(max_length=200)
    text = models.TextField()
    date_added = models.DateTimeField(
        auto_now_add=True)
    owner = models.ForeignKey(
        User, on_delete=models.CASCADE)

    # === Extended fields ===
    tags = models.ManyToManyField(
        Tag, blank=True)
    likes = models.ManyToManyField(
        User, blank=True,
        related_name='liked_posts')
    views = models.PositiveIntegerField(
        default=0)
    image = models.ImageField(
        upload_to='post_images/',
        blank=True)

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

```python
@login_required
def edit_post(request, post_id):
    post = get_object_or_404(
        BlogPost, id=post_id)

    # === Permission check ===
    if post.owner != request.user:
        raise Http404

    if request.method != 'POST':
        form = BlogPostForm(
            instance=post)
    else:
        form = BlogPostForm(
            instance=post,
            data=request.POST,
            files=request.FILES)
        if form.is_valid():
            form.save()
            messages.success(
                request, 'Updated!')
            return redirect(
                'blogs:post_detail',
                post_id=post.id)

    return render(request,
        'blogs/edit_post.html',
        {'post': post, 'form': form})
```

2. 权限保护（19-5核心）

3. 点赞功能（扩展）

```python
@login_required
def like_post(request, post_id):
    post = get_object_or_404(
        BlogPost, id=post_id)

    # Like/Unlike toggle
    if post.likes.filter(
        id=request.user.id).exists():
        post.likes.remove(request.user)
        liked = False
    else:
        post.likes.add(request.user)
        liked = True

    # AJAX response support
    if request.headers.get(
        'X-Requested-With') ==
        'XMLHttpRequest':
        return JsonResponse({
            'liked': liked,
            'total_likes':
                post.total_likes()
        })

    return redirect(
        'blogs:post_detail',
        post_id=post_id)
```

4. 全文搜索（扩展）

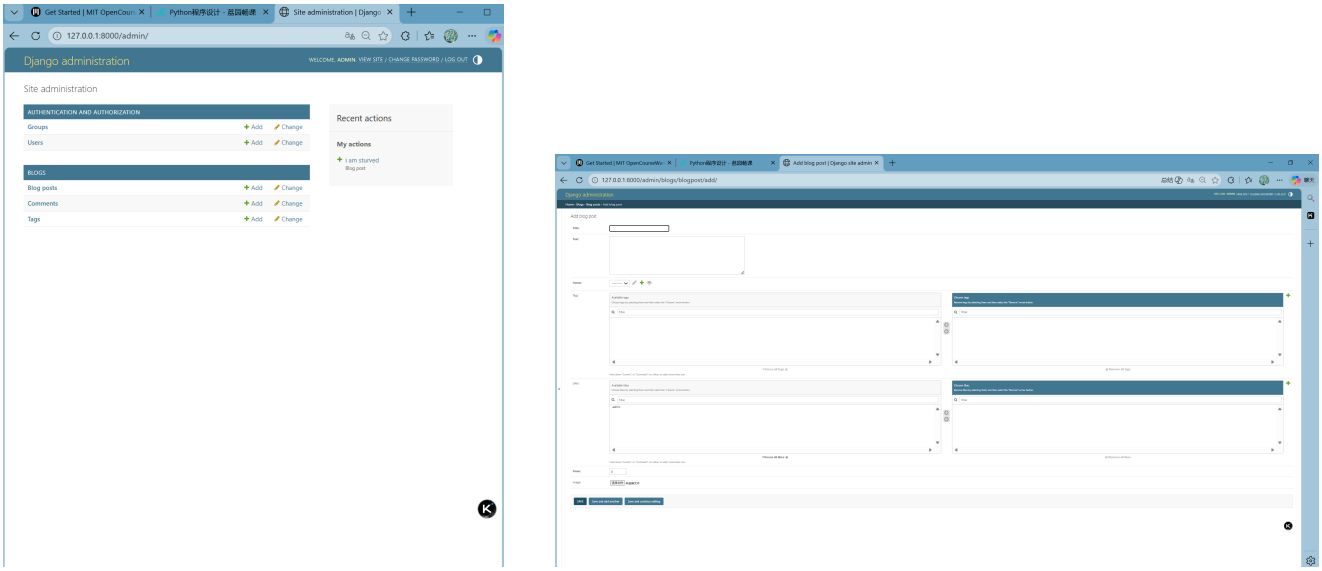
```
def search(request):
    query = request.GET.get('q', '')
    posts = []

    if query:
        # Q object multi-field query
        posts = BlogPost.objects.filter(
            Q(title__icontains=query) |
            Q(text__icontains=query)
        )

    return render(request,
                  'blogs/search_results.html',
                  {'posts': posts,
                   'query': query})
```

4 功能演示截图

本项目完整实现了作业要求的所有功能，并进行了多项扩展，以下为关键功能的演示截图。

4.1 核心功能：Admin后台与文章管理

Django Admin后台提供完善的管理界面，支持对Blog posts、Comments、Tags的完整CRUD操作。图3展示了Admin首页和添加文章页面，可以看到BlogPost模型包含了作业要求的基础字段（title、text、owner）以及扩展字段（tags、likes、views、image）。

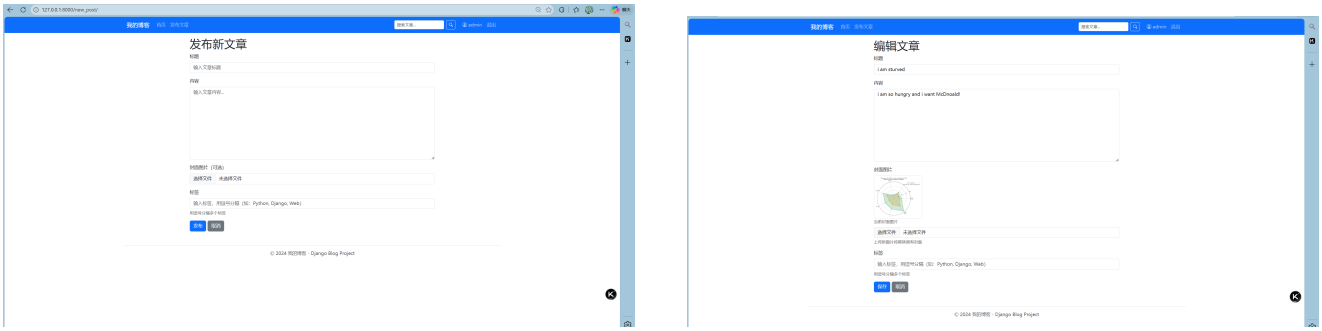


(a) Admin后台首页 (b) Admin添加文章页面

图 3: Django Admin后台管理界面

4.2 核心功能：文章发布与编辑

前台用户界面提供友好的文章发布和编辑功能。图4左图展示发布文章页面，支持输入标题、内容、上传封面图、添加标签；右图为编辑文章页面，**系统会验证用户是否为文章作者，只有作者才能编辑和删除自己的文章（4-5核心要求）**。

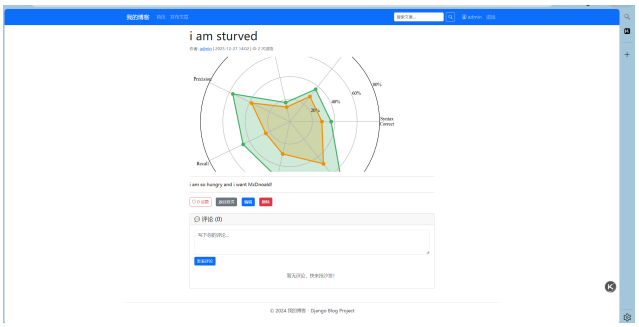


(a) 发布新文章表单 (b) 编辑文章页面（权限保护）

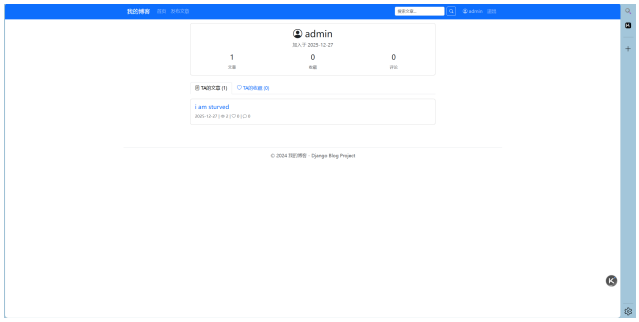
图 4: 文章发布与编辑功能

4.3 扩展功能：社交互动与用户系统

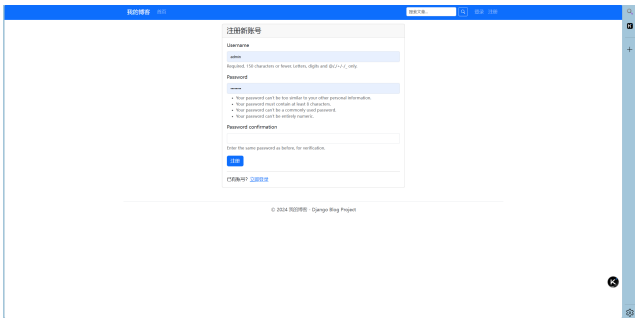
扩展功能包括：(1) 点赞功能（显示点赞数，支持实时点赞/取消）；(2) 浏览量统计；(3) 评论系统；(4) 标签展示；(5) 用户注册/登录；(6) 用户个人主页（统计信息）。图5展示了文章详情页（点赞、评论、浏览量、标签）、用户注册、用户登录和用户个人主页。



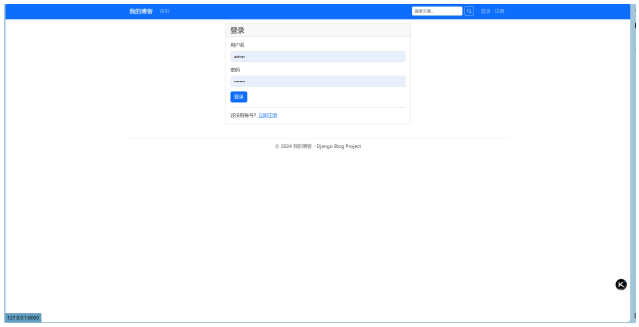
(a) 文章详情（点赞/评论/标签）



(b) 用户个人主页



(c) 用户注册页面



(d) 用户登录页面

图 5: 扩展功能展示：社交互动与用户系统

5 技术总结与作业完成情况

1. **作业完成度**：已实现project4所有要求。BlogPost模型包含title、text、date_added、owner字段；实现了完整的用户认证系统；文章按时间倒序显示；支持新建和编辑文章；编辑前验证文章所有权（`if post.owner != request.user: raise Http404`）。
2. **扩展功能**（红色标注）：标签系统（ManyToMany关系）、评论功能（Comment模型）、点赞机制（AJAX实时更新）、图片上传（ImageField + 媒体文件管理）、全文搜索（Q对象多字段查询）、用户主页（统计信息）、分页显示。
3. **技术栈与架构**：Django 5.2 + SQLite + Bootstrap 5 + Pillow；采用MVT模式（Model-View-Template）；模块化设计（blogs应用+accounts应用）；使用Django内置认证系统和Admin后台。
4. **代码质量**：使用`@login_required`装饰器保护视图；表单验证（`form.is_valid()`）；消息提示（Django messages框架）；遵循DRY原则（Don't Repeat Yourself）；充足的注释说明。

技术特点：Django MVT架构 — 用户认证与权限 — ManyToMany关系 — AJAX异步交互 — 媒体文件管理

6 安装与运行说明

项目结构：详见README.md文档，完整说明了目录结构、数据模型、功能特性和使用指南。
GitHub：https://github.com/Csh0601/project4forpy

7 教师评语

评分：_____ 日期：_____