Phase 3 Report: Blog Feature Development

Date: October 30, 2024 - November 12, 2024 Group 18

I.Summary of the Work Done to Date

1.1 Blog Post Publishing Feature

We successfully implemented the blog post publishing feature, allowing users to create and publish articles through an intuitive form. The specific tasks completed include:

Blog Model Design: Using Django's Model Class, we defined the data structure for blog posts, including fields such as title, content, publication date, and author.

```
class BlogCategory(models.Model):  5 用法
    name = models.CharField(max_length=200,verbose_name='分类')

    def __str__(self):
        return self.name
    class Meta:
        verbose_name = '博客分类'
        verbose_name_plural = verbose_name

class Blog(models.Model):  8 用法
    title = models.CharField(max_length=200,verbose_name='标题')
    content = models.TextField(verbose_name='内容')
    pub_time = models.DateTimeField(auto_now_add=True)
    category = models.ForeignKey(BlogCategory, on_delete=models.CASCADE,verbose_name='分类')
    author = models.ForeignKey(User, on_delete=models.CASCADE,verbose_name='作者')

    def __str__(self):
        return self.title
    class Meta:
        verbose_name = '博客'
        verbose_name_plural = verbose_name
        ordering = ['-pub_time']
```

Publishing Interface: The blog publishing form was generated using Django's ModelForm, simplifying the development process and ensuring data consistency between the front and back end.

Permission Control: Only authenticated users are allowed to create and publish blog posts, and unauthenticated users are redirected to the login page.

1.2 Rich Text Editor Integration

For the blog post publishing feature, we integrated WangEditor as the rich text editor, allowing users to format their blog content. Users can easily add images, headings, links, and other elements, improving the readability and presentation of the articles.

1.3 Blog Post Comment Feature

We successfully developed the comment feature, enabling users to comment on and reply to blog posts. The main implementation details include:

Comment Model Design: We designed an independent comment model, which is linked to blog posts via foreign keys.

Comment Submission Form: The comment submission form was generated using Django's

ModelForm, ensuring ease of use and accurate data validation.

Comment Display: Users can view and submit comments on the blog post detail page, and the comments are displayed in real-time on the page.

1.4 Search Functionality Implementation

We also developed the search functionality for blog posts, allowing users to search for relevant articles using keywords. The key tasks completed include:

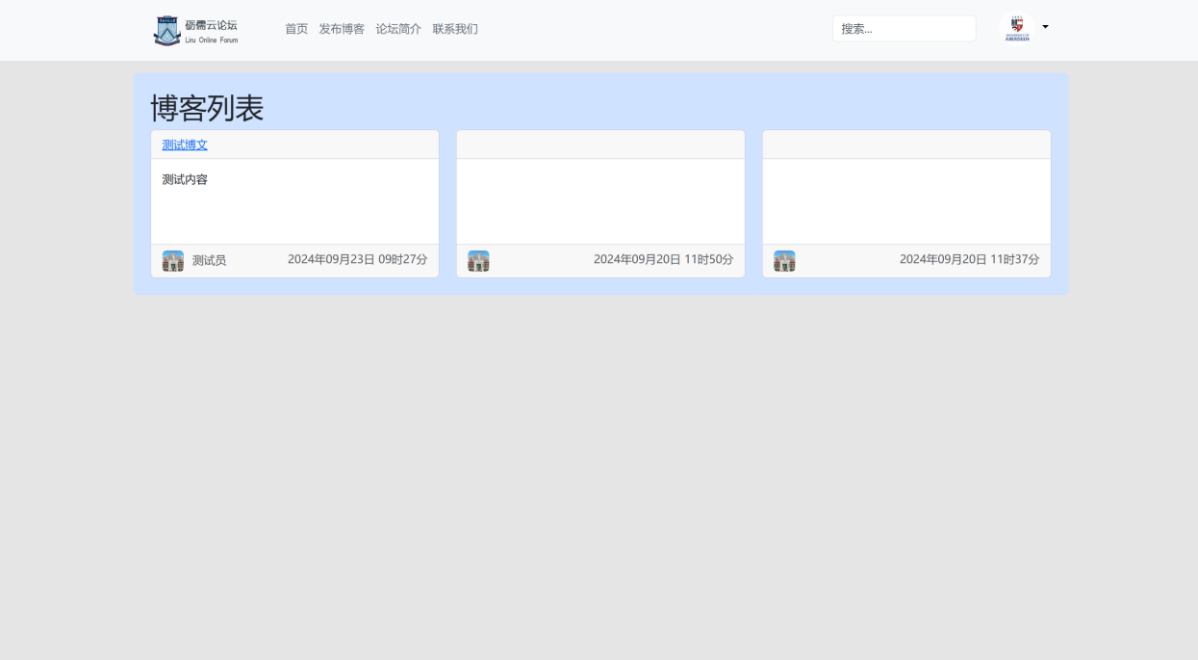
```python
@require_GET
def search(request):
    # /search?q=xxx
    q = request.GET.get('q')
    # 从博客的标题和内容中查找含有q关键字的博客
    blogs = Blog.objects.filter(Q(title__icontains=q)|Q(content__icontains=q)).all()
    return render(request, template_name: 'index.html', context={"blogs": blogs})
```

Full-Text Search: We implemented full-text search across blog titles and content using Django's QuerySet, ensuring users can find content based on specific keywords.

Pagination: Search results are paginated to enhance the browsing experience, allowing users to navigate through large numbers of search results easily.

1.5 Blog Homepage Layout Design

In addition to the core blog features, we also implemented the homepage layout design for the blog. This layout was designed to be user-friendly and visually appealing, showcasing recent blog posts, categories, and featured content in an organized manner, ensuring a seamless user experience from the homepage onwards.



II.Challenges Faced and Solutions

2.1 Customization of WangEditor

During the integration of WangEditor, we encountered compatibility issues with the default configuration, particularly with image uploading and formatting options.

Solution: We customized WangEditor's image upload functionality and other toolbar options to better fit the needs of our website. This customization allowed for smooth handling of formatted text, images, and multimedia content within blog posts.

2.2 Search Performance Optimization

As the amount of data grew, we noticed that the search response time slowed, especially when handling a large number of queries simultaneously.

Solution: We optimized database query execution and introduced indexing mechanisms to improve the performance of the search function. This ensures that even with large amounts of data, search results are returned quickly and efficiently.

III.Next Steps

3.1 Backend Management Features

In the next phase, we will develop the backend management system, providing administrators with tools to manage blog posts, moderate comments, and handle user management to ensure the quality and compliance of platform content.

3.2 system Testing and Optimization

As we complete core functionality, we will conduct comprehensive system testing to ensure the stability of all features across different use cases. Additionally, we will continue to optimize both the front and back end based on user feedback.