Introduction to Software Engineering

AI Group 18: Liru Online Forum

Technical Report

Date: November 26, 2024

Authors:

Siyan Wu

ID:50091024

Chenghua Zhu

ID: 50090983

Lintao Ouyang

ID: 50090993

Group Members:

Siyan Wu （50091024）

Chenghua Zhu （50090983）

Lintao Ouyang （50090993）

Yunqi Chen （50091018）

Chenchen Ming （50091020）

Xueyi Wang （50090995）

Jiarui Feng （50090979）

Chenxi Zhao （50090990）

Abstract

This paper presents the development process of the campus blog forum, encompassing requirement analysis, software design, testing, and user interface design across multiple dimensions. First, we identified the functional requirements of the system, including user management, blog post publishing, commenting, and search functionalities. These features will provide users with a comprehensive blogging experience, facilitating the exchange of information and sharing of knowledge. Additionally, we defined non-functional requirements that address security, reliability, and performance, ensuring a positive user experience during system usage.

In terms of technical implementation, we utilized the Django framework and Python language for their rich functionalities and flexible design, supporting rapid development. By combining modular design with the MVC architecture, we effectively enhanced development efficiency while minimizing system maintenance complexity. To ensure the quality of the system, we implemented unit testing, component testing, performance testing, and interface testing during the software testing phase. This series of tests confirmed the functionality of the system, meeting the diverse needs of users while significantly improving reliability and stability.

In user interface design, we adopted a responsive layout to ensure users could have a good experience across different devices. Our design focuses on user interaction, ensuring the interface is friendly and usable, enabling users to easily browse, publish, and manage blog conten.

Future work directions include enhancing interactive features, adding entertainment functionalities, optimizing page animations, and introducing user feedback mechanisms to further improve system usability and user satisfaction. The successful implementation of this project will provide a more interactive and engaging platform for the campus community, with the expectation of promoting further development of campus culture.

Context

Chapter 1 Introduction

With the rapid development of the internet, campus blogging platforms have become

increasingly popular among students as tools for information exchange and knowledge sharing. These platforms not only provide students with a space to freely express their thoughts but also foster interaction and learning among peers, making them an essential part of modern campus culture. Therefore, we decided to develop a campus blog forum aimed at providing users with an open, free, and secure environment for publishing, sharing, and managing blog content.

In this project, we have chosen the Incremental Model as the primary development methodology. This choice is based on its flexibility and adaptability, which effectively address the ever-changing user requirements and project complexities. By breaking the entire project down into multiple small, manageable functional modules, the team can gradually implement key features such as user management, blog publishing, and commenting, ensuring that each function is fully developed and rigorously tested within the development cycle. The design of the Incremental Model enables us to identify issues early in the development process and make necessary adjustments, thereby reducing development risks and improving the quality and stability of the software .

The goal of this project is to provide campus users with a feature-rich, user-friendly, and highly interactive blogging platform. To achieve this, we comprehensively considered the actual needs of users in the system design, drawing upon successful experiences and design principles from related fields. Throughout the development and implementation phases, we will maintain continuous communication with users to gather feedback, ensuring that the platform evolves and ultimately meets its intended goals.

Chapter 2 Software Design Methodology

In this project, we have chosen the Incremental Model as the primary methodology for software development. Compared to the traditional Waterfall model, the Incremental Model offers greater flexibility and iterative development capabilities, effectively handling changes in requirements and project complexity. According to Ian Sommerville's Software Engineering, employing the Incremental Model helps establish a high-quality software product.

2.1 Why Choose the Incremental Model

2.1.1 Iterative Development

The Incremental Model allows us to break the entire project down into several small, manageable functional modules, each of which can be developed and tested independently. For the development of the campus blog forum, we sequentially implemented user management, blog publishing, and comment functionalities. This iterative development approach enables us to identify issues early and make timely adjustments during the initial stages of the project.

2.1.2 Flexibility and Adaptability

The design of the Incremental Model allows us to adapt flexibly to feedback and changes in requirements during the development process. For instance, during the development of user management features, we decided to integrate CAPTCHA and email verification features in response to evolving security requirements. This adaptability enhances our ability to meet dynamic user needs, thus improving system reliability.

2.1.3 Risk Reduction

By gradually delivering fully functional modules, the Incremental Model effectively reduces the risks associated with large-scale development. Each module undergoes independent testing and validation, allowing us to detect potential issues early on. As emphasized in quality management, early independent validation contributes to ensuring consistency and stability across different phases of development .

2.2 Comparison with Waterfall Model

2.2.1 Parallel Development

The Incremental Model permits team members to simultaneously develop different modules. For instance, during various stages of the project, some members focused on the development of blog functionalities, while others handled user management and backend management systems. This parallel development approach enhances efficiency and maximizes the utilization of human resources, aligning with efficient team collaboration principles in project management.

2.2.2 Early Feedback

The Incremental Model enables us to receive early feedback from users and stakeholders during the development process. Each functional module undergoes internal testing upon completion, and necessary adjustments are made based on the testing results. This process ensures that the system can be continuously optimized and improved, ultimately meeting the expected quality standards.

2.2.3 Flexible Handling of Requirement Changes

At various stages of project development, requirements may change. The Incremental Model allows us to make flexible adjustments based on these changes. For example, when implementing the comment feature, we added a "like" functionality for comments, a requirement that was not explicitly stated in the early stages but was incorporated in subsequent phases.

2.3 Why Choose the Incremental Model Over Waterfall

2.3.1 Dynamic Changing Requirements

For a complex project like ours, requirements tend to evolve based on user feedback and results from system testing. The Incremental Model allows us to respond flexibly to these changes within each iteration, avoiding the rigid handling of requirements

typical in the Waterfall Model. This flexibility is also highlighted in software quality management .

2.3.2 Early Delivery and Validation of Features

The Incremental Model enables us to deliver portions of functionality earlier and conduct user testing and validation, ensuring that the final product meets user requirements. After completing the user management feature, comprehensive system testing was conducted to ensure the stability of the user system for subsequent blog functionality development.

2.3.3 Optimized Resource Utilization

The project team can utilize members' skills to perform modular division of labor. The Incremental Model allows for parallel handling of multiple modules, thus maximizing resource utilization. For instance, while one part of the team was implementing the blog comment feature, other members were optimizing the integration of the rich text editor, ensuring that each feature could be delivered on time.

Chapter 3 Software Requirement

In this project, the focus is on implementing distinct features that differentiate the blog forum from existing social media platforms. The primary objective is to offer users a secure, open, and highly customizable environment where they can freely express themselves without restrictions.

3.1 Functional Requirements

The functional requirements of the campus blog forum are essential to providing a seamless user experience. Below are the key functional features that will shape the core of the system.

3.1.1 Comment System

To enhance user interaction, the blog integrates a dynamic and visually engaging commenting system. Unlike traditional platforms, our system features a scrolling "danmu" style for real-time comment display, making interactions livelier. This real-time element injects spontaneity and fun into the platform, ensuring the blog community feels vibrant and active. While adding entertainment, we also incorporate mechanisms for order and quality control. Users can report inappropriate comments or like comments to show support, creating a community-driven moderation system that balances creative freedom with responsibility.

3.1.2 Customization Options

A core differentiator of our blog forum is the extensive customization feature. Users can personalize their blogs through multiple themes, color schemes, and font choices, allowing them to reflect their personal style. This functionality goes beyond mere

appearance, as users can also categorize their posts to make content organization more intuitive, helping their audience find relevant posts more easily.

3.1.3 Privacy Protection

Privacy is a fundamental requirement of our platform. Users can set their blogs' visibility, ranging from fully public to private, ensuring full control over who can access their content. Furthermore, users are empowered to manage their data autonomously—whether it's modifying, accessing, or deleting content. This transparency strengthens the trust between users and the platform.

3.1.4 Email Verification

To enhance account security, the system integrates an email verification feature using QQ Mail to send verification codes. When users register for an account, the system sends a verification email containing a code to the user's registered email address. The user must enter the code to complete the registration process, ensuring the authenticity and security of the account. This feature helps prevent malicious registrations and fake accounts, further strengthening the platform's security.

3.1.5 Multilingual Support

To serve a diverse user base, the system incorporates multilingual support, ensuring that users can easily switch between Chinese and English. Our comprehensive translation approach ensures that every aspect of the platform is accessible, fostering an inclusive environment for a global audience.

3.2 Non-Functional Requirements

In addition to the core functionalities, the system is designed to meet stringent non-functional requirements to ensure performance, security, scalability, and overall usability.

3.2.1 Security

To safeguard user data and ensure confidentiality, the platform employs advanced security measures such as encryption, identity verification, and firewall protocols. The system's email verification feature adds an extra layer of security, ensuring each registered account is authentic by sending verification codes via QQ Mail. These measures aim to prevent unauthorized access and data breaches, providing a secure environment where users can feel confident sharing content.

3.2.2 Performance and Scalability

To accommodate growing user numbers and high-traffic periods, the platform is optimized for performance, with features such as load balancing and database sharding. These improvements ensure the blog operates smoothly under heavy use without compromising performance.

3.2.3 Compatibility

The system is compatible with various browsers and devices, leveraging responsive design principles to provide a seamless experience across platforms. Whether accessed from desktops or mobile devices, users will experience a consistent and optimized interface.

3.2.4 Maintainability

To support long-term maintenance and future updates, the platform follows modular design principles. This structure facilitates ease of maintenance, allowing for future expansions, bug fixes, and system upgrades to be implemented with minimal disruption.

3.3 Hardware and Software Requirements

For optimal performance, the blog system relies on the following hardware and software configuration.

3.3.1 Hardware

Processor: At least Intel Core i5 or equivalent

RAM: Minimum 8GB

Storage: 500GB SSD

3.3.2 Software

Operating System: Windows 11 or Ubuntu 20.04

Programming Language: Python 3.11.x with Django 4.x framework

Database: MySQL 8.x

Web Browser: Google Chrome 116.x for testing and compatibility

Development Tools: Navicat for MySQL database management, Git for version control

Chapter 4 Software Design

In the development of the campus blog forum, the software design phase is crucial to ensuring that the system meets its requirements and operates reliably. This phase primarily focuses on object-oriented design methodologies and leverages the features of the Django framework to implement the overall architecture of the system.

4.1 Programming Language Selection

We selected Python as the programming language for this project for several compelling reasons:

Ease of Learning and Use: The syntax of Python is clear and straightforward, which reduces the learning curve for developers, especially for new team members.

Rich Libraries and Frameworks: Python boasts a plethora of powerful libraries and frameworks, with Django providing numerous built-in functionalities that facilitate rapid development.

Community Support: Python has a large and active developer community, making it easy to obtain technical support and share resources.

4.2 Design Aspects

During the design process, we considered several key aspects:

Modular Design: The system is divided into multiple modules, including user management, blog functionalities, comment features, and backend management. This modular design enhances the maintainability and extensibility of the system.

MVC Architecture: We adopted Django's Model-View-Controller (MVC) architecture, which clarifies the separation between business logic, user interface, and data models. This design facilitates modifications and expansions of system functionalities .

4.3 Software Architecture

The architecture of our system comprises several layers:

Presentation Layer: This layer is responsible for user interaction, displaying blog articles, and managing the user interface. We used Django's templating engine to render dynamic pages and integrated WangEditor as a rich text editor, making it easier for users to create and edit content.

Business Logic Layer: This layer handles user requests, executes business logic, and processes data. We utilized Django views to manage requests and implement the relevant business logic.

Data Access Layer: This layer is responsible for interacting with the database. We used Django's Object-Relational Mapping (ORM) feature to simplify database operations, enhancing data processing efficiency.

The modular design and layered architecture not only streamline the development process but also ensure that the system can evolve to meet future needs.

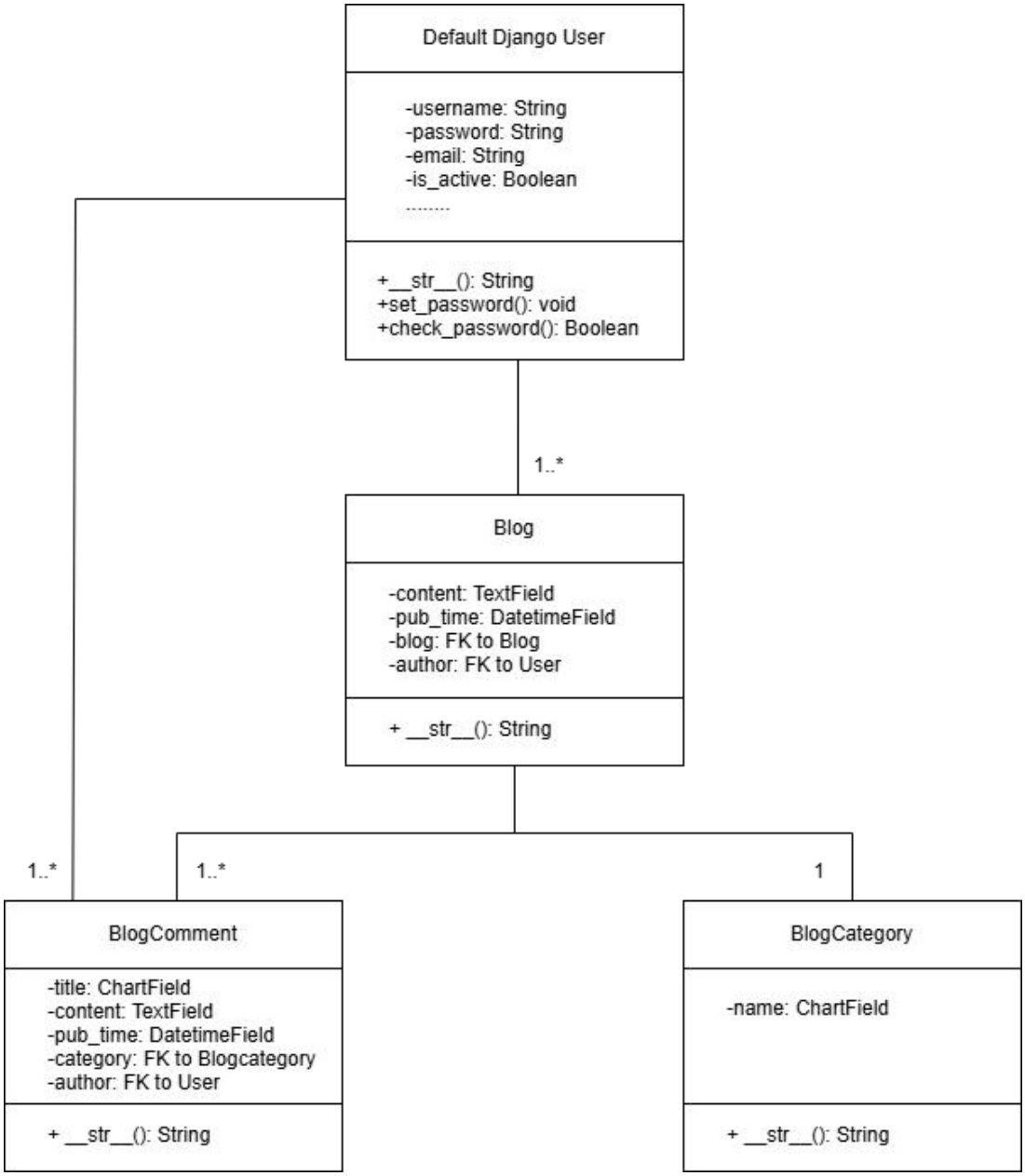
Chapter 5 Software Analysis

The software analysis phase is critical in the software development process, as it not only helps us understand the design structure of the system but also provides a foundation for subsequent development and testing activities. This phase primarily focuses on the design of class diagrams and sequence diagrams, along with a detailed analysis of their content.

5.1 Class Diagrams

Class diagrams are essential tools in object-oriented design that illustrate the

structure of the system and the relationships between classes. For the campus blog forum, we designed several key classes:



The design of these class diagrams ensures that the relationships between different classes are clear, facilitating easier subsequent development and maintenance .

5.1.1 Class Relationship

I.Relationship between Default Django User and Blog

The User class represents the users of the Django system.Each Blog object has an author field, which is a ForeignKey referencing the User model, indicating that a blog post is created by a specific user.

Relationship: A User can be associated with multiple Blog posts, forming a one-to-many relationship, where one user can create multiple blog posts (1..*

relationship).

II.Relationship between Blog and BlogCategory

The BlogCategory class represents the category of blog posts.Each Blog object has a category field, which is a ForeignKey referencing the BlogCategory, indicating that each blog post belongs to a specific category.

Relationship: A BlogCategory can contain multiple Blog posts, forming a one-to-many relationship, where one category can have multiple blog posts (1..* relationship), but each blog post can only belong to one category.

III.Relationship between Blog and BlogComment

The BlogComment class represents the comments on blog posts.Each BlogComment object has a blog field, which is a ForeignKey referencing the Blog, indicating that the comment is related to a specific blog post.

Relationship: A Blog can have multiple BlogComments, forming a one-to-many relationship (1..* relationship), where one blog post can have multiple comments.

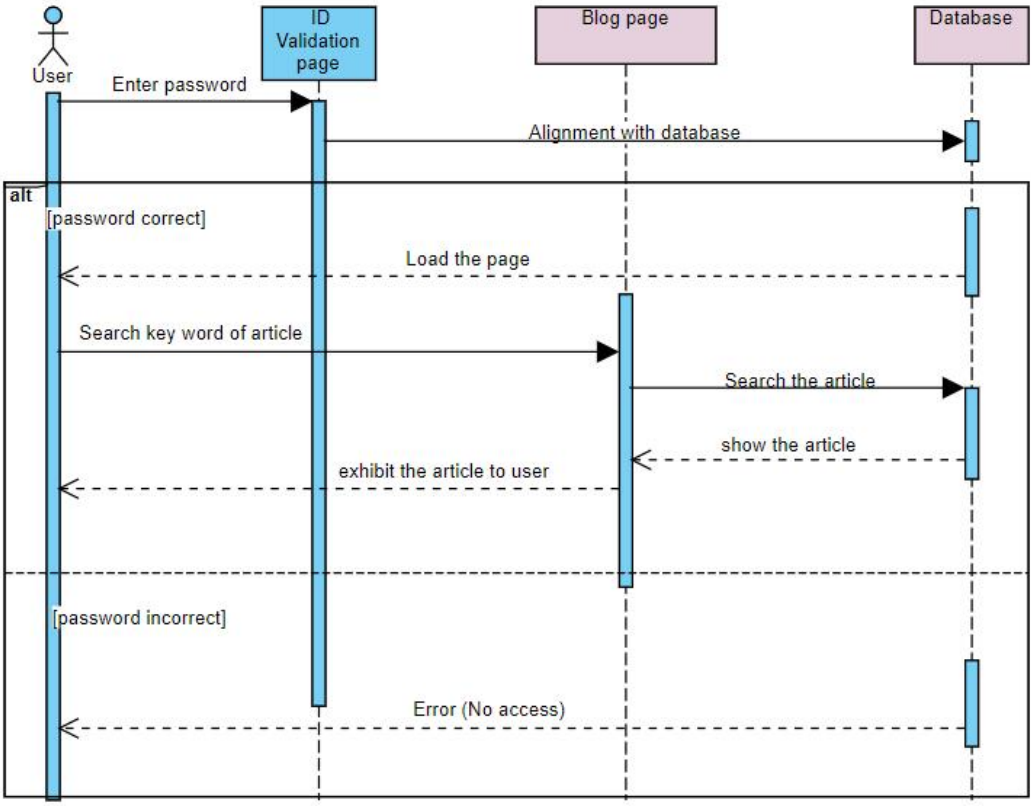
IV.Relationship between BlogComment and Default Django User

The BlogComment class has an author field, which is a ForeignKey referencing the User model, indicating the author of the comment.

Relationship: A User can write multiple BlogComments, forming a one-to-many relationship (1..* relationship), where one user can write multiple comments.

5.2 Sequence Diagrams

5.2.1 Sequence Diagrams



These sequence diagrams clearly depict the interaction processes between users and the system, ensuring the integrity and logic of system functionalities .

5.2.2 Explanation

I.User Login Verification

At the beginning of the sequence diagram, the user initiates a login request by entering a password on the login page. This request is passed to the ID validation page, where the system verifies the credentials provided by the user. If the password is correct, the system allows the user to proceed and loads the blog page for access. On the other hand, if the password validation fails, the system returns an error message, informing the user of the incorrect password and preventing access to the system. This step illustrates the complete user authentication process, ensuring that only authorized users can access the blog page.

II.Keyword Search and Database Interaction

Once the user has successfully logged in, they can use the search function on the blog page by entering keywords to find articles. After receiving the search request, the system constructs a query and sends it to the database for processing. The database

is responsible for searching for articles that match the entered keywords. During this process, the interaction between the blog page and the database ensures that users can quickly retrieve relevant content based on the input keywords. This step outlines the entire search functionality, involving user input, system query processing, and database retrieval operations.

III.Displaying Search Results

After the database completes the search, it returns the matching articles to the blog page. The system organizes the search results and presents them to the user, allowing the user to view a list of articles related to the entered keywords on the blog page. This process is the final step in the search function, demonstrating how the system handles the user's search request and ultimately displays the results. Through the database query and the return of results, the system enables the user to retrieve articles as needed.

5.3 User Password Input Handling

In the processes of user registration and login, the system needs to handle user password input securely. We implement the following steps to ensure security:

Password Encryption: When storing user passwords in the database, we employ hashing techniques to prevent plain text storage, thereby enhancing security.

Validation Mechanism: During user login, the system hashes the entered password and compares it with the stored hash value in the database to ensure security.

This design aligns with the security requirements outlined in software quality management, effectively mitigating potential security risks.

Chapter 6 Software Testing

Software testing is a crucial phase in ensuring that the campus blog forum functions correctly and meets user requirements. This phase primarily focuses on unit testing, component testing, performance testing, interface testing, and release testing to guarantee the quality and reliability of the system.

6.1 Unit Testing

Unit testing involves validating the smallest testable units within the software, ensuring that each component functions correctly when run independently. In the development of the campus blog forum, we focused on the following areas for unit testing.

6.1.1 Testing Environment

I.Operating System

Windows 11 Pro (64-bit): As the host operating system, Windows 11 provides a

stable and powerful environment for development and testing. It supports various development tools, database management tools, and testing frameworks, offering great compatibility for the project.

II.Development Framework

Django 4.x: As the main framework, Django offers an efficient and secure web development environment. It includes a wide range of built-in functionalities and third-party libraries, allowing for rapid environment setup and ensuring system robustness. The latest version of Django ensures support for the newest technologies and features.

III.Database

MySQL 8.x: MySQL is used as the primary database management system. Known for its high performance and reliability, MySQL supports concurrent operations, making it suitable for data storage and query handling in medium to large-scale projects. Combined with Django's ORM, MySQL can easily manage complex queries and data interactions.

IV.Database Management Tool

Navicat 16.x for MySQL: As the database management tool, Navicat provides an intuitive and powerful GUI interface, allowing developers to create, modify, manage databases, and import/export data. In this project, Navicat is used to manage the MySQL database, monitor data changes, and maintain test data.

V.Programming Language

Python 3.11.x: Since Django is a Python framework, the latest version of Python is essential for development. Python 3.11 offers improved performance, enhanced syntax, and better support for third-party libraries. In the testing phase, we will use Python to write test scripts and leverage Django's built-in testing tools for comprehensive tests.

VI.Browser

Google Chrome version 116.x: Google Chrome is the primary browser used during testing. Chrome provides robust developer tools (DevTools), supporting network request monitoring, performance optimization, and page debugging. It is our preferred browser for frontend interaction testing.

6.1.2 Test Case Design

User Management Module: Testing the user registration functionality to validate the effectiveness and security of input data. Ensuring that the login functionality correctly processes valid and invalid user credentials.

Blog Functionality Module: Testing the creation, editing, and deletion operations of blog posts to ensure data is accurately stored and updated. Verifying that the

categorization and tagging functionalities operate as intended.

Comment Functionality Module: Ensuring that comments can be added and removed effectively, and that they are correctly associated with the respective posts .

6.1.3 Testing Execution

Test Module	Test Case	Inputs	**Expected Results**
User Management Module	User Registration Function	Valid username, password, and email	Registration successful; user information should be stored in the database, and a success message should be returned.
	User Login Function	Valid username and password	Login successful; a user session should be created.
	Invalid Login Credentials	Invalid username and password	Returns an error message: "Username or password is incorrect."
Blog Functionality Module	Create Blog Post	Post title, content, and tags	Post successfully created and stored in the database.
	Get Blog Post List	/	Returns a list of all blog posts.
	Publish Blog Post (Boundary Condition Test)	Empty title and content	eturns an error message: "Title and content cannot be empty."
Comment Functionality Module	Add Comment	Post ID and comment content	Comment should be successfully adde
	Get Comments for Post	Post ID	Returns a list of all comments under that post.
	Add Comment (Boundary	Post ID and empty comment content	Returns an error message: "Comment content

	Condition Test)		cannot be empty."

6.1.4 Test Results Analysis

1.User Management Module

User Registration Function: Testing results indicate that when valid username, password, and email are provided, registration is successful, and user information is correctly stored in the database, returning a success message.

User Login Function: Logging in with valid username and password is successful, and the user session is created properly.

Invalid Login Credentials: When invalid username and password are entered, the system returns an error message: "Username or password is incorrect," which meets expectations.

2.Blog Functionality Module

Create Blog Post: All provided inputs for post title, content, and tags can be successfully created, and the post is correctly stored in the database.

Get Blog Post List: The system can correctly return all blog posts, and the functionality is normal.

Publish Blog Post (Boundary Condition Test): When empty title and content are inputted, the system returns an error message: "Title and content cannot be empty," and the functionality verification meets expectations.

3.Comment Functionality Module

Add Comment: When valid post ID and comment content are inputted, the comment is successfully added, and the system returns a success message.

Get Comments for Post: The system can normally return all comments under a specific post.

Add Comment (Boundary Condition Test): When post ID and empty comment content are provided, the system returns an error message: "Comment content cannot be empty," and the functionality verification meets expectations.

6.2 Component Testing

Component testing focuses on the interaction between independent components of the software to ensure that they work together as intended. In this phase, we conducted the following tests:

Blog Article Component Testing: We conducted comprehensive testing of the blog article display functionality to ensure that all articles' content, titles, and tags are correctly displayed on the user interface after publishing. During the testing process,

we verified the display effects of different types of articles (such as text, images, and videos) and ensured that users do not encounter any formatting errors or missing content while browsing. Additionally, we checked the accuracy of the publication date and author information to ensure that this information is appropriately displayed on the user interface. The success of this test provides users with a friendly reading experience, enhancing the usability of the system.

Comment Component Testing: In testing the commenting functionality, we focused on the interface between the commenting feature and the blog module to ensure that user-submitted comments are accurately displayed under the corresponding articles. The testing included verifying the display order of user comments, the integrity of the content, and the timestamps of the comments. We also tested the display of multiple comments to ensure that the system maintains a good display effect even with a high number of comments. Additionally, we performed boundary condition tests to verify that the system can handle invalid comment content or empty comments correctly and return appropriate error messages. Through these tests, we ensured that the commenting functionality operates normally, allowing users to interact conveniently.

Navigation Bar Component Testing: The testing of the navigation bar aimed to ensure that users can smoothly switch between different functional pages. We tested the links to the homepage, article pages, user management pages, and other functional modules to ensure that each link responds quickly and directs users to the correct page. Furthermore, we focused on the responsiveness and user experience of the navigation bar, ensuring that the layout and functionality remain usable when users access the system on mobile devices or different screen sizes. Through these tests, we confirmed that the design of the navigation bar is reasonable and effectively supports user operations within the system.

6.3 Performance Testing

Performance testing aims to evaluate the system's performance under specific load conditions, including response time, stability, and resource consumption. In the development of the campus blog forum, we conducted the following aspects of performance testing in a local database environment:

6.3.1 Load Testing

We simulated multiple users accessing the system simultaneously to assess the performance under multi-user operations. In the local environment, we set up 10 users to access the blog homepage concurrently, monitoring page load times and system response times.

The test results indicated that the system could load the homepage within 2 seconds, displaying all the latest articles and functionalities correctly, providing a good user experience.

6.3.2 Stress Testing

To evaluate the system's capacity, we gradually increased the number of concurrent users to observe performance under high load. During the tests, 15 users simultaneously performed registration, login, and commenting operations.

The results showed that the system maintained a reasonable response time without noticeable performance degradation when handling 15 concurrent users.

6.3.3 Stability Testing

We conducted long-duration tests to observe the system's performance during sustained use. In the local environment, we allowed the system to run continuously for 24 hours, simulating actual user access while recording response times and resource usage.

The results indicated that the system remained stable over the extended period, with no significant increase in response time, maintaining smooth performance.

6.3.4 Resource Usage Testing

We monitored the system's resource usage under different load conditions, including CPU and memory. During high-concurrency access tests, we used local monitoring tools to record peak resource usage.

The test results showed that the CPU usage remained below 50%, with stable memory consumption, indicating that the system operated smoothly without performance bottlenecks.

6.3.5 Conclusion

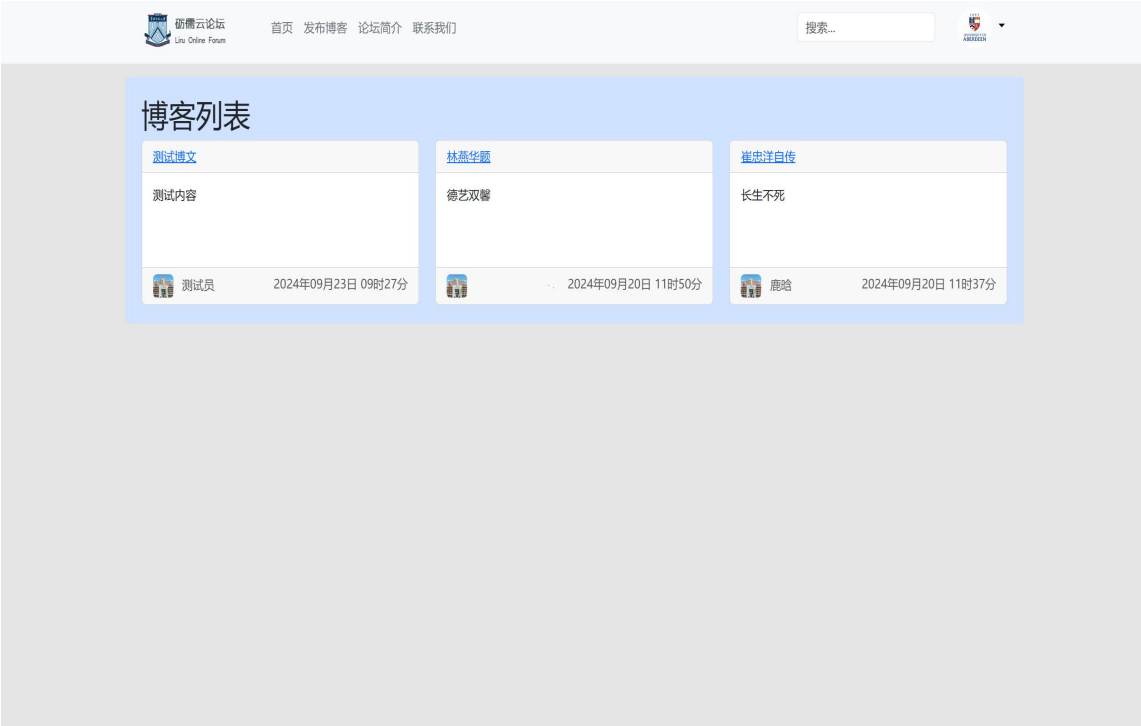
Through these performance tests, we gained a comprehensive understanding of the system performance of the campus blog forum. The test results demonstrate that the system can operate smoothly in a local database environment, meeting user needs and providing important insights for future optimizations and improvements.

6.4 Interface Testing

In this project, we implemented several functional modules for the user interface of the campus blog forum and conducted integration testing to ensure these modules operate and interact correctly. This section showcases two main interface modules: the blog homepage and the article display page, featuring custom designs and the WangEditor rich text editor.

6.4.1 Blog Homepage

As shown in the screenshot below, the blog homepage is designed with simplicity, displaying a list of the latest published articles, a category directory, and popular tags. Users can click on any article title, and the system will automatically navigate to the detailed article page, where the full content is displayed. This design ensures that users can quickly browse and access content of interest.



We tested the following functionalities:

Article List Display: Ensure that the article list on the homepage displays correctly in order of publication, with the newest articles at the top.

Category Directory and Tags: Users can filter content by clicking on categories and tags, ensuring that the correct articles are displayed based on the user's selection.

Navigation Functionality: When users click on an article title, the system correctly navigates to the article detail page, where the content is fully displayed.

6.4.2 Article Detail Page

When a user clicks on an article title, the system navigates to the article detail page, where the full content of the article is displayed. This page integrates WangEditor as the rich text editor, allowing users to easily create and edit article content, supporting various text formatting features.

We tested the following functionalities:

Article Content Display: Ensure that the article detail page correctly displays the title, content, publication date, and author information.

Commenting Feature: Users can leave comments at the bottom of the article, and the system will display the comments in real time, allowing users to edit or delete them.

Editor Functionality Testing: Tested WangEditor's text formatting, image insertion, and other features to ensure users can create and edit content smoothly.

6.4.3 Search and Filtering Functionality

At the top of the blog homepage, a search feature is available, allowing users to quickly find articles by entering keywords. Additionally, the system supports filtering articles based on tags and categories, ensuring users can easily locate the content they need. We tested the following functionality:

Keyword Search: Ensure that after entering a keyword in the search box, the system returns the correct articles and sorts them by relevance.

Through these interface tests, we verified that the system's modules work collaboratively and ensure a smooth user experience. These tests provide a solid foundation for the system's functional stability.

6.5 Release Testing

In the final phase of the system, we performed release testing to ensure that all functionalities operate correctly after integration. The release testing included the following steps.

6.5.1 Functional Verification

Verifying that all functional modules (such as user management, blog article publishing, and commenting functionality) work as required. We checked the system's responses to various inputs through test cases to ensure its behavior meets the specifications. This process ensures that the system provides all necessary functionalities to users at the time of release.

6.5.2 Error Checking

Ensuring that no new errors or faults have been introduced into the system during integration. By executing regression tests, we check the stability and integrity of existing functionalities to confirm that they remain unaffected by recent changes. This aligns with a key principle of software quality management: ensuring that existing functionalities still work when new features are introduced.

6.5.3 Compatibility Testing

Testing across various devices and browsers to guarantee good compatibility. We utilized multiple mainstream browsers (such as Chrome, Firefox, and Safari) as well as different types of devices (such as PCs and mobile devices) to ensure a consistent user experience across various environments. This process is a critical aspect of ensuring user satisfaction and software quality.

6.5.4 Performance Monitoring

Monitoring the system's performance post-release to ensure that it can maintain good response times and stability under high load conditions. We used performance testing tools to record system performance under high concurrent user access, ensuring that the system can handle the load requirements encountered in actual usage.

6.5.5 User Acceptance Testing

In the final stage of release testing, we invited real users to participate in testing to gather their feedback on the system. The actual user experience is crucial to the success of the system, so we further optimized the system based on user feedback to enhance overall user satisfaction.

6.5.6 Conclusion

Through these steps of release testing, we ensured that all functional modules of the campus blog forum could run stably and securely at the time of release, meeting the actual needs of users.

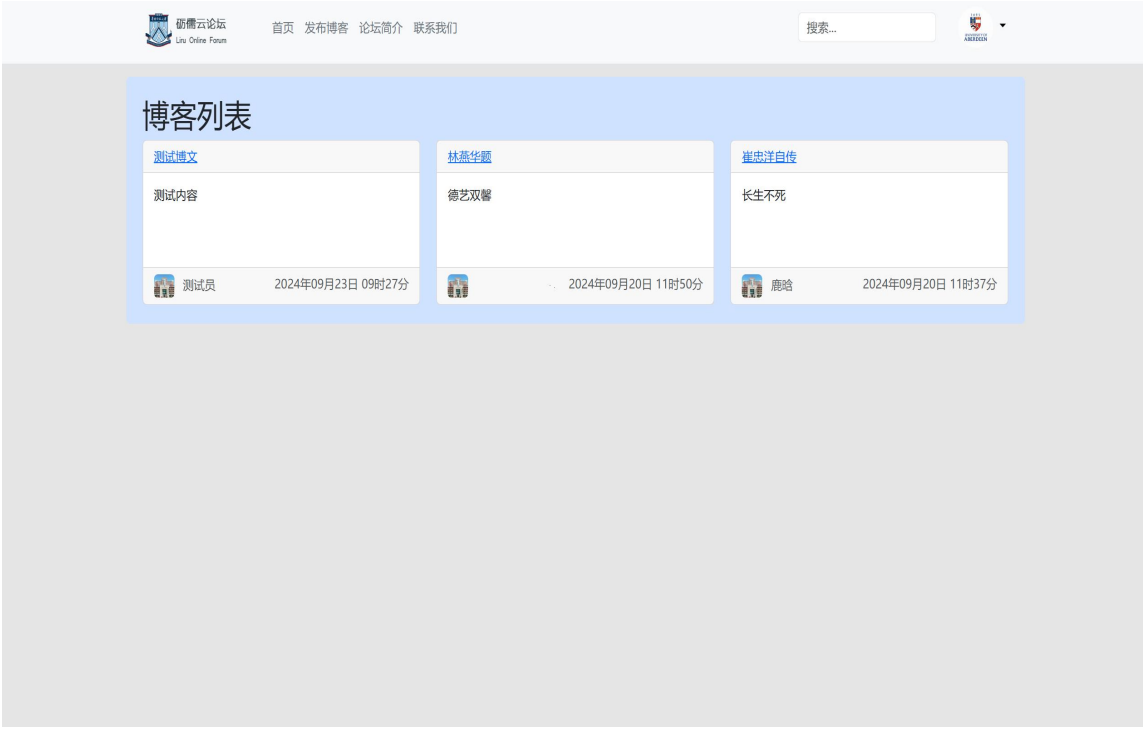
Chapter 7 Graphical User Interface Design

The design of the graphical user interface (GUI) plays a crucial role in the development of the campus blog forum, as it aims to provide a user-friendly and intuitive interaction experience. This phase focuses on the layout of the interface, user interactions, and aesthetics to ensure that users can conveniently access and utilize the various features of the system.

7.1 Layout Structure

The overall layout of the interface adopts a responsive design, ensuring that it provides a good user experience across various devices, including PCs, tablets, and smartphones.

I.Homepage



Displays the latest blog posts, categories, and tags prominently.

Provides a search box to allow users to quickly find content of interest.

II.Article Page

Shows the article's title, content, author information, and publication date.

Includes a comment section where users can interact with the content.

III.User Management Page

Provides entry points for user registration and login, as well as a section for users to view and edit their personal information.

This layout design ensures that information is presented clearly, allowing users to navigate effortlessly to different functions.

7.2 Article Display

In the article display section, we employed a clear and concise typography that makes

the content easy to read. The layout of each article includes:

Title: Highlighted in a large font to attract user attention.

Content: Utilizes appropriate line spacing and paragraph formatting to ensure readability.

Author Information: Provides the author's name along with a link to their profile, enhancing the visibility of the authors.

7.3 User Interaction

We designed various user interaction methods to enhance the user experience:

Buttons and Links: Clear and visually distinct buttons guide users in actions such as registration, login, and article publishing. The colors and text of the buttons ensure that users can quickly identify their functions.

Dynamic Feedback: Implemented dynamic loading effects and interaction feedback, such as displaying comments immediately after submission and updating the comment count, to enhance interactivity.

7.4 Multilingual Support

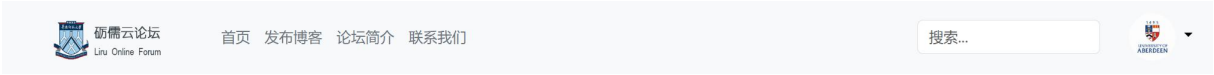
To cater to the needs of different users, the system features multilingual support. Users can easily switch between Chinese and English versions through a simple interface toggle, thus improving accessibility for a diverse user base. This design ensures that our platform can serve a wide range of users .

7.5 Page Aesthetics

The aesthetic appeal of the interface is a significant component of the user experience. We utilized a fresh color palette and a clean design style to create a welcoming environment. Key design elements include:

Color Scheme: A selection of soft colors is used to reduce visual fatigue for users during prolonged use.

Font Selection: Easy-to-read fonts ensure that the text is clear and visible, contributing to a better overall user experience.



Through thoughtful visual design, we aim to provide users with a comfortable and enjoyable experience when using the forum .

Chapter 8 Conclusion and Future Work

In this project, we successfully designed and implemented a campus blog forum that meets the basic needs of users. By employing the Incremental Model and modular design, we were able to flexibly respond to changes in requirements and ensure high quality throughout the system. The following are the main conclusions of the project and directions for future work.

8.1 Conclusion

System Implementation: Our campus blog forum successfully implemented functionalities such as user management, blog publishing, commenting, and searching. Each module underwent rigorous testing and validation, ensuring the system's stability and reliability.

User Experience: With a thoughtfully designed user interface, users can easily browse, publish, and manage blog content, leading to a positive user experience. This achievement demonstrates the success of our efforts in graphical user interface design.

Technical Choices: The decision to utilize the Django framework and Python language made the development process more efficient and reduced the complexity of system maintenance. This decision had a positive impact on the smooth progression of the project .

8.2 Future Work

To further enhance the system's functionality and user experience, we plan to implement the following improvements in future work.

I.Enhanced Interaction Features

Private Messaging System: One of our primary goals is to introduce a private messaging feature. This will allow users to communicate directly with each other, fostering a closer community and encouraging more personal interaction. Users will have the ability to exchange messages, fostering deeper connections within the platform's community.

Liking and Sharing Functions: In addition, we plan to implement features that allow users to like and share articles. These functionalities will help promote more engagement on the platform, encouraging users to support content they appreciate while increasing the visibility of articles. The sharing feature will also allow users to distribute content across other social media platforms, further extending the platform’s

reach.

II.Adding Entertainment Features

Mini-Games and Video Sharing: To create a more engaging user experience, we will develop mini-games and integrate video-sharing functionalities. This will provide users with entertaining content, diversifying the types of activities available on the platform. The integration of video-sharing will give users the ability to share multimedia content, further enriching their interaction with the platform.

Event and Contest Features: We also plan to design events or contests that encourage user participation. Competitions, themed events, and community challenges will be used to attract more users and increase overall platform activity. These events will provide incentives such as rewards or recognition, further motivating user participation.

III.Page Animation Optimization

Enhanced User Interaction Animations: To make the user interface more dynamic and visually appealing, we plan to incorporate more interactive animations in response to user actions. For instance, animations will be triggered when users click buttons, navigate between pages, or hover over specific elements. This will not only add aesthetic value but also provide visual feedback, making the platform more engaging and user-friendly.

Optimizing Loading Animations: We aim to reduce waiting times for users by improving the efficiency of loading animations. While minimizing load times is a priority, we also want to ensure that any loading processes are visually appealing, keeping users engaged during the brief wait periods. These animations will be subtle yet effective in maintaining user interest, thereby creating a smoother and more seamless browsing experience.

IV.User Feedback Mechanism

Introducing a Comprehensive Feedback System: One of our main objectives is to implement a system that allows users to provide direct feedback on the platform's features, usability, and interface. This feedback mechanism will be embedded across various parts of the platform, enabling users to report bugs, suggest new features, or comment on their overall experience. By regularly reviewing this feedback, we can prioritize updates and modifications based on user input, ensuring that the platform evolves in line with user expectations.

Conducting User Satisfaction Surveys: In addition to the feedback system, we will periodically conduct user satisfaction surveys. These surveys will help us measure how well the platform meets the needs of its user base, assessing areas like usability, performance, and feature satisfaction. The insights gathered from these surveys will guide future development and ensure that improvements are user-driven. Regular

user satisfaction checks will allow us to stay aligned with our community's evolving demands and preferences.

Through these future endeavors, we look forward to further enhancing the user experience of the campus blog forum, increasing user engagement, and creating greater user value.