COM668 Computing Project

Assessment Task 3 – Project Review Report

Project title:	Design and Implementation of Personal Bog System
Student name:	Yang Hao
Student ID number:	B01006068
PSG identifier:	PSG-S02
Mentor name:	Yang Yuexin
Course title:	Final Year Project
Year of submission:	2024/2025
Word Count (9000 Maximum):	8183

You are required to self-declare the word count in this report.
A falsified word count may be considered as academic misconduct.
This is in keeping with Ulster University policy.
The Module Handbook and guidance has advice regarding what must be included in the word count.

Contents

1. Introduction

1.1. Concept

The research topic of this thesis is the design and implementation of a personal blog system based on Vue3 and Spring Boot. This system is a modern personal blog platform, aiming to provide users with high-quality blog writing and reading experiences. The core functions of the system include article management, category and tag management, user authentication and authorization, article search, and article detail display, etc. The research scope mainly focuses on front-end interface design and user experience optimization, back-end API design and implementation, database design and performance optimization, as well as the guarantee of system security.

1.2. Aim

The research topic of this thesis is the design and implementation of a personal blog system based on Vue3 and Spring Boot. This system is a modern personal blog platform, aiming to provide users with high-quality blog writing and reading experiences. The core functions of the system include article management, category and tag management, user authentication and authorization, article search, and article detail display, etc. The research scope mainly focuses on front-end interface design and user experience optimization, back-end API design and implementation, database design and performance optimization, as well as the guarantee of system security[1].

Figure1.1 Project Feature Implementation Keywords

1.3. Justification for changes (if any)

The original intention of the project was to create a multi-user blog system that supports multiple users posting, managing, and interacting within a community. It involves complex features such as advanced permissions, data isolation, and concurrent control, which have extremely high requirements for architecture, database, and security. During the implementation, challenges such as user authentication, data consistency, scalability, and performance optimization were encountered, which require profound technical accumulation. Due to time, resource, and technical limitations, it was adjusted to a single-user blog, focusing on article management, display, and experience, simplifying complexity. It retains the potential for expansion, and in the future, multi-user functions can be added. Although advanced features were reduced, it is more practical and feasible, and better meets the needs of personal blogs.

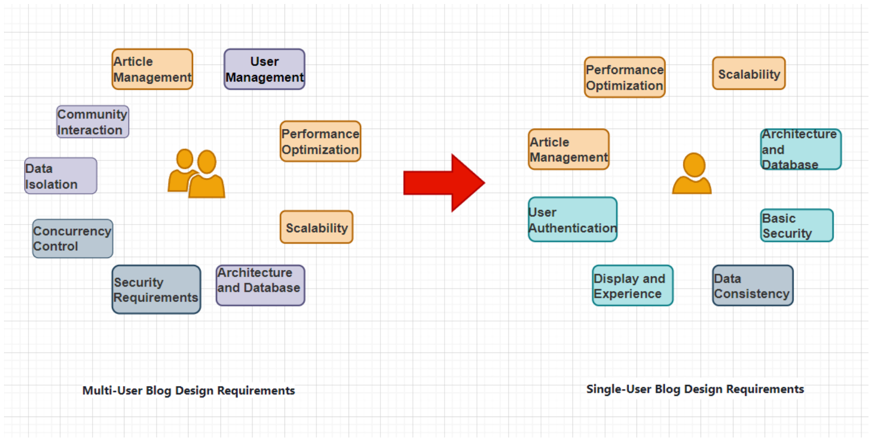


Figure 1.2 Focus of Implementation in Multi-user Blog System and Single-user System

1.4. Outcomes

This research is expected to complete a fully functional personal blog system, achieving a good user experience, with system performance meeting the expected goals and code quality conforming to standards. These achievements not only have practical value, providing references for personal blog development and demonstrating modern web development technologies, but also have theoretical value, exploring the front-end and back-end separation architecture and studying the best practices of modern web development [2]. However, the research also has some limitations, such as a limited development cycle and limited personal development resources, which may prevent the realization of some advanced functions. Despite these, this research will still provide valuable references and guidance for the development of personal blog systems.

2. Plan

2.1. Plans

The goal of the project is to provide high-quality article management and presentation capabilities while maintaining the scalability of the system and leaving room for possible future upgrades. In terms of function planning, the system will achieve user authentication and authorization, article management, category label management, article search, article details display and other core functions[3]. Together, these features will build a complete blogging ecosystem that provides users with a great writing and reading experience

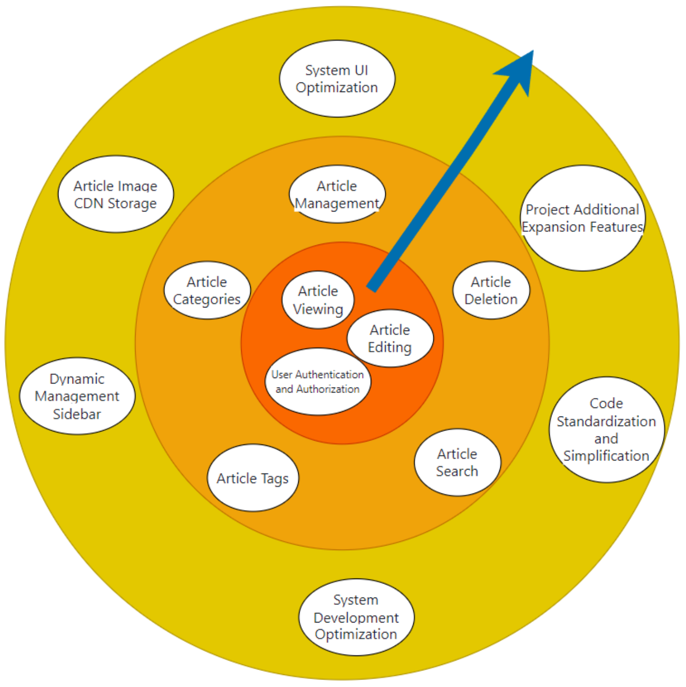

System UI Optimization
Article Image CDN Storage
Article Management
Project Additional Expansion Features
Article Categories
Article Viewing
Article Deletion
Article Editing
User Authentication and Authorization
Dynamic Management Sidebar
Code Standardization and Simplification
Article Search
Article Tags
System Development Optimization

Figure 2.1 Feature Implementation Priority

In terms of technology selection, the front-end uses Vue3 framework with TypeScript and Element Plus component libraries, the back-end uses Spring Boot framework and MyBatis Plus persistence layer framework, and the data storage uses MySQL database. These technologies were selected based on their maturity, community support, performance, and development efficiency[4]. Vue3 provides better performance and a more flexible Composition API, TypeScript improves code quality and maintainability, Spring Boot simplifies back-end development, and MySQL provides reliable data storage.

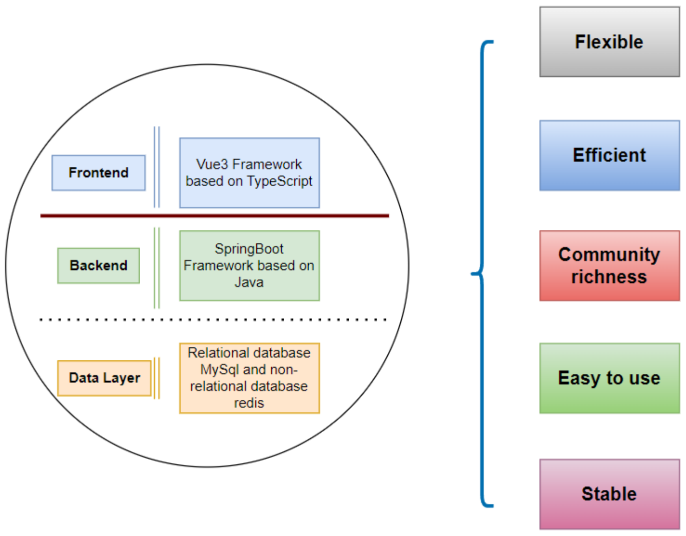

Frontend
Vue3 Framework based on TypeScript
Backend
SpringBoot Framework based on Java
Data Layer
Relational database MySql and non-relational database redis
Flexible
Efficient
Community richness
Easy to use
Stable

Figure 2.2 Technology Selection Advantages

The development of the project strictly follows the standard software development cycle, including requirements analysis, system design, coding implementation, testing and verification, deployment and online and other stages, each stage has a clear schedule and quality requirements. Agile development strategy is adopted, functional requirements are prioritized based on the MoSCoW model, and business value, technical complexity and risk management are comprehensively considered to ensure that core functions are implemented first, while additional functions are taken into account to improve user experience [5]. When resources are limited, we focus on implementing high-priority functions to ensure stable and efficient project delivery

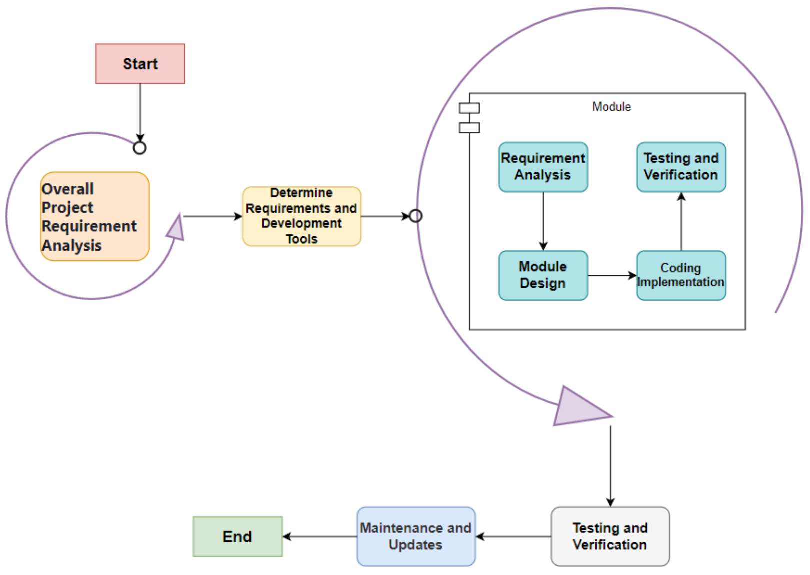


Figure 2.3 Software Development Process

Table2.1 MoSCoW Prioritization

Feature ID	Dependent Features	Feature Description	Estimated Time (Days)	Priority
1	None	Project blueprint planning, design database structure and specific implementation goals	7	**MUST**
2	Feature 1	System architecture setup, complete project environment configuration, implement user registration and login module, integrate JWT authentication and front-end back-end testing	7	**MUST**
3	Feature 2	Homepage layout, information display module	7	**MUST**
4	Feature 2	User registration	2	**MUST**
5	Feature 2	User login	2	**MUST**
6	Feature 4, Feature 5	JWT authentication integration	2	**MUST**
7	Feature 2	Article publishing	3	**MUST**
8	Feature 7	Article editing	3	**MUST**
9	Feature 7	Article deletion	2	**SHOULD**
10	Feature 7	Simple text upload for blog	4	**SHOULD**
11	Feature 10	Markdown support	7	**SHOULD**
12	Feature 10	Image upload to remote image hosting	3	**SHOULD**

13	Feature 7	Comment functionality	7	**MUST**
14	Feature 13	Comment reply reception	3	**SHOULD**
15	Feature 13	Comment management	7	**SHOULD**
16	Feature 7	Article search	3	**SHOULD**
17	Feature 7	Tag functionality	3	**SHOULD**
18	Feature 7	Article category management	4	**SHOULD**
19	Feature 7	Popular article recommendation	4	**COULD**
20	Feature 7	User profile management	3	**SHOULD**
21	Feature 7	Article archiving	4	**COULD**
22	Feature 7	Admin system	7	**MUST**
23	Feature 7	Article modification	3	**SHOULD**
24	Feature 22	Article backend management	7	**MUST**
25	Feature 22	Display module	7	**COULD**
26	Feature 7	User role and permission management	4	**SHOULD**
27	Feature 7	Article like functionality	3	**WON'T**
28	Feature 7	Social sharing functionality	4	**WON'T**
29	Feature 7	Email notification functionality	3	**COULD**
30	Feature 7	Friendship links module	3	**WON'T**
31	Feature 7	SEO optimization	7	**SHOULD**
32	Feature 7	API documentation	3	**COULD**

2.2. Metrics

In order to comprehensively evaluate the effect of the implementation of the plan, this project constructs a multi-dimensional index system. In terms of performance, the system must ensure that the page load time is less than 2 seconds, and the system response time must be controlled within 500 milliseconds except for certain apis [6]. In terms of user experience, the system will provide an intuitive and convenient interface, allowing users to easily publish and manage articles, while enabling efficient content reading and search. In terms of content support, the system will be compatible with a variety of article formats, especially Markdown format, and ensure that the content display effect is excellent.

In addition, the project will also consider the integrity of the documentation, the quality of the code and the scalability of the system to ensure that the system has excellent maintainability and sustainable development. In terms of security, the system will strengthen the authentication and authorization mechanism, improve the overall protection capability, and ensure the security of user data and the stability of the system. Through these comprehensive and detailed indicators, we will conduct an in-depth evaluation of the overall performance of the system to ensure the smooth and successful implementation of the program[7].

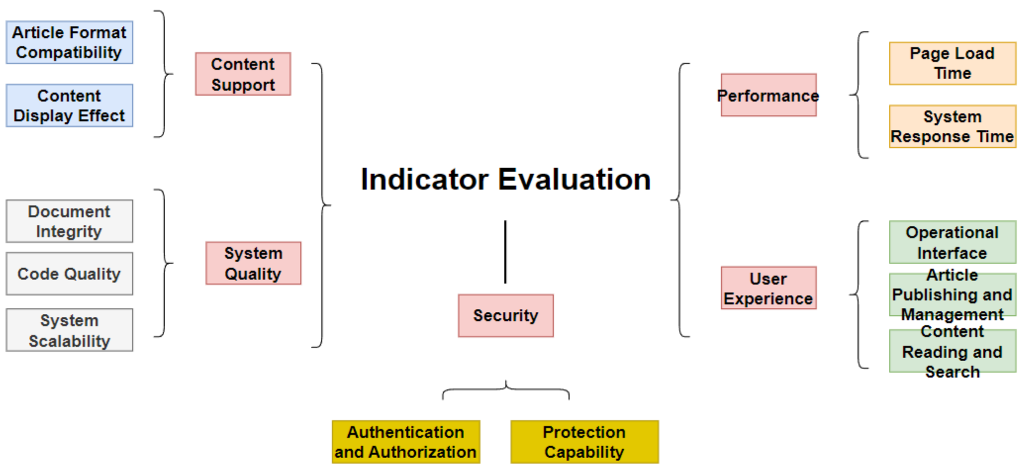


Figure 2.4 System Design Evaluation Criteria

2.3. Justification for planning decisions

In this project, the front-end adopts Vue3 and TypeScript technology stack, and TypeScript's strong typing features enhance the readability and predictability of code, effectively improving the standardization of development and the maintainability of code. The back-end is developed based on the Spring Boot framework, which provides out-of-the-box project structure and automatic configuration mechanism, simplifying the construction process of the back-end system[8]. In terms of database, MySQL is selected as the core data storage scheme. With its high reliability, good performance and mature ecosystem, it can stably support the complex requirements of the system in data reading and writing, transaction processing and other aspects [9].

In addition, both Vue and Spring Boot have extensive community support and rich learning resources, and the relevant development documents, best practices and technical solutions are highly mature, providing a solid guarantee for the efficient promotion and problem solving of the project. The integration of these technologies not only meets the requirements of current business functions, but also takes into account the future scalability and maintainability of the system, laying a solid foundation for the sustainable development of the project.

In terms of functional design, the system is built around the core usage scenario of users to ensure that each module has actual usability and good user experience. User authentication and authorization mechanism effectively ensure the system access security; Article management module provides a convenient creation and management platform for content creators; The detailed presentation of the article optimizes the reading and interactive experience of the content. These functional modules work together to build a secure, easy to use and efficient blog ecosystem[10].

This project has clear objectives and tasks in each stage of development process management. Demand analysis focuses on user needs and business logic, the system design clearly defines the architecture and module scheme, the coding stage strictly follows the design specifications, and the test ensures the stability of the system through multi-layer verification. The overall process is efficient and standardized, ensuring that the project is completed on time and with high quality.

3. Technical Solution

3.1. Architectural model

When building a blog-like project, it is important for visitors to access blog details. In order to make every visitor can easily and smoothly browse the blog content, the blog system chooses to be designed in B/S architecture. B/S architecture, with its unique advantages, brings the dual guarantee of easy access and high cross-platform compatibility [11]. Regardless of the device or network environment, visitors can access blogs anytime, anywhere through a browser, without worrying about additional software installation or configuration issues.

However, in individual development projects, developers often face the challenge of highly coupled code, which not only reduces development efficiency, but also increases the difficulty of subsequent maintenance. In order to break this dilemma, the project adopts the development model of front and back end separation. The front-end and back-end separation effectively decouples the front-end and back-end code, allowing developers to focus more on their respective areas of development work, which significantly improves development efficiency. At the same time, this mode also greatly improves the maintainability and expansibility of the project, and lays a solid foundation for the long-term stable operation and sustainable development of the blog system[12].

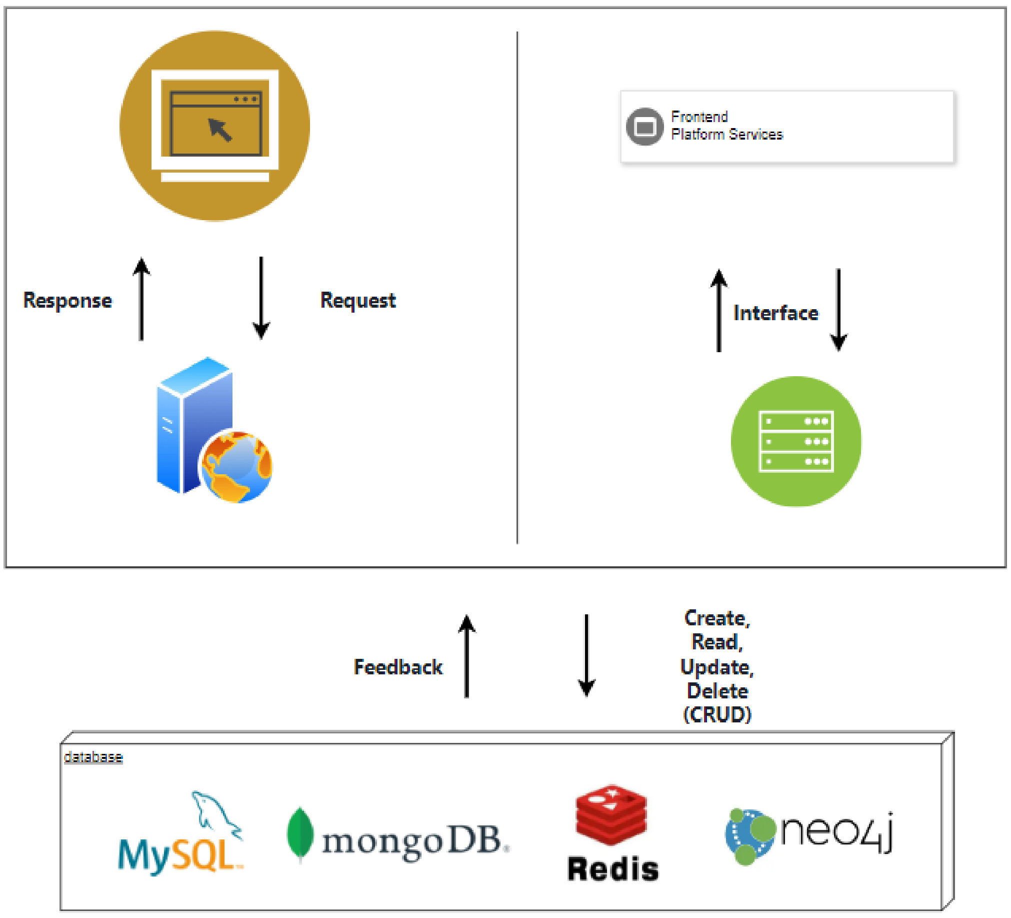


Figure 3.1 B/S Architecture (left) and Frontend-Backend Separation (right)

On this basis, this project further introduces the design concept of layered architecture. By dividing the system into presentation layer, business logic layer and data access layer, the responsibility of each layer is clear and independent. The presentation layer is responsible for presenting the user interface and interaction logic, the business logic layer handles specific business rules and processes, and the data access layer is responsible for interacting with the database. This layered design not only reduces the coupling degree between layers, but also improves the reusability of code and the flexibility of the system.

Under the guidance of a layered architecture, developers can organize their code in a more orderly manner, which is convenient for subsequent modifications and extensions. For example, when the user interface needs to be updated, the code in the presentation layer can only be adjusted without affecting the business logic layer and the data access layer. Similarly, changes to business rules only need to be made at the business logic level and do not affect other levels. This modular design method simplifies the development process and improves the development efficiency[13].

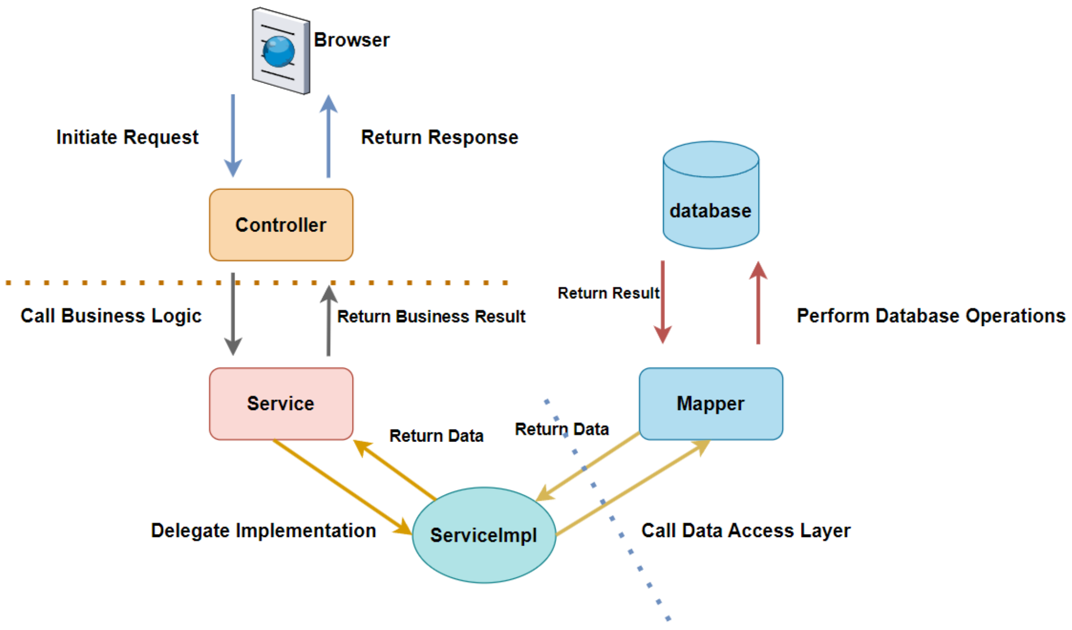


Figure 3.2 Multi-tier Architecture

Through the combination of B/S architecture, front end separation and layered architecture, the project not only achieves easy access and wide compatibility, but also has the advantages of efficient development, easy maintenance and flexible expansion, bringing visitors a better reading experience. This comprehensive architecture design not only reflects the technical trend of modern Web development, but also lays a solid foundation for the future development of blog system.

3.2. System design

The front-end of the blog system is based on Vue3 and TypeScript, combining Element Plus, Pinia and Axios to achieve interface construction and data interaction, and using localStorage to improve user experience. The back end uses SpringBoot framework, integrates Spring Security and JWT to achieve permission control, uses MyBatis Plus to simplify data operations and supports scheduled tasks. Redis is introduced to cache hotspot data to improve system response speed, and MySQL is used to implement data persistence in the underlying database. The system has clear structure, good performance and expansibility.

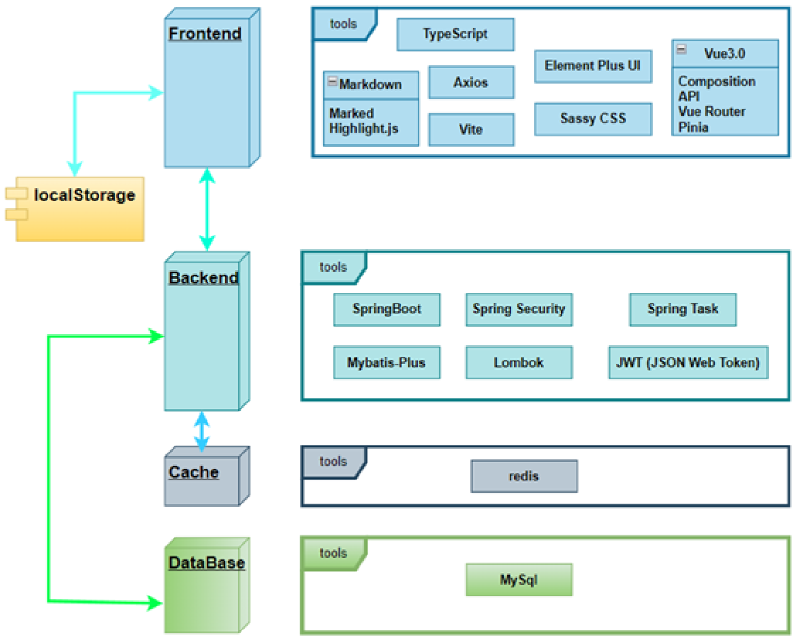


Figure 3.3 Overall Architecture of the Blog System

In order to ensure efficient design and development of projects, a set of specifications is developed to unify programming development and the use of tools. First, Git will be adopted as a version control tool and the style of "Conventional Commits" will be strictly followed to ensure the clarity and consistency of the commits [14]. When the project is launched, a storage warehouse will be created on GitHub for cloud storage and data backup.

In terms of programming style, in addition to Java code, others need to extensively use serpentine nomenclature for variable naming in the project to improve the readability and maintainability of the code. Front-end development will take place in a Windows environment, using the VSCode compiler with the front-end plug-in installed. It mainly uses TypeScript, SCSS, and Vue frameworks for development, and uses PNPM as a package management tool. Back-end development is also carried out in the Windows environment, using IDEA, and following Alibaba development specifications (Alibaba Cloud Developer Community, 2024) to ensure code quality and development efficiency. The database development will be carried out in the Linux environment, and the MySql field naming will also adopt the serpentine naming method to maintain the consistency of naming style. Through these specifications, we can improve the efficiency of project development and ensure the quality of code.

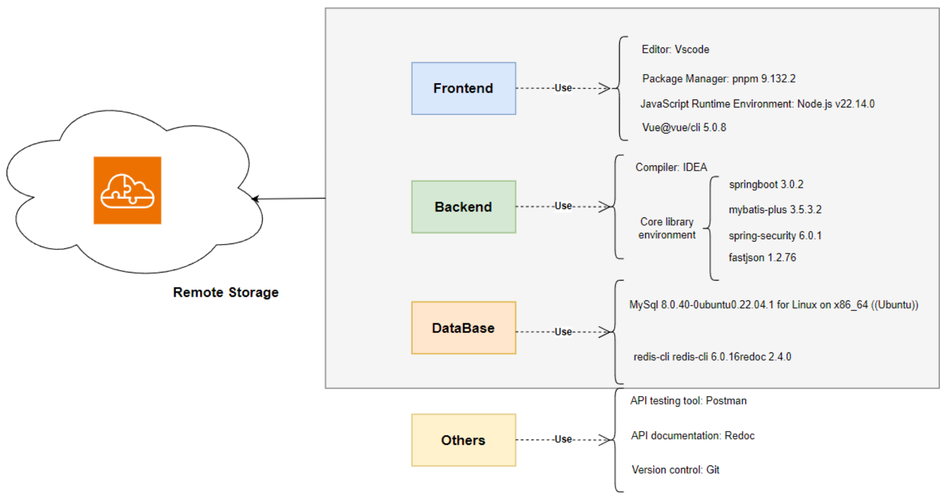


Figure 3.4 Scope of Tools Used in Project Development

Table 3.1 Tools Used in Project Development and Their Versions

Name	Version
Visual Studio Code	-
IntelliJ IDEA 2023.1.4	-
Navicat Premium 15	-
postman	-
Windows 10	-
wsl2-Ubuntu-22.04	-
MySql	8.0.40-0ubuntu0.22.04.1 for Linux on x86_64 ((Ubuntu))
Redis-cli	redis-cli 6.0.16
Git	2.41.0.windows.1
Node.js	v22.14.0
pnpm	9.12.2
java	17
springboot	3.0.2
Mybatis-plus	3.5.3.2
spring-security	6.0.1
fastjson	1.2.76
vue	@vue/cli 5.0.8
redoc	2.4.0

The front-end of the blog system adopts templating and component-based design, combined with Pinia for status management, Vue Router for route management, functions are divided into article reference, editing and modification, management, classification label, login registration and search modules. Componentized design divides the system into independent, reusable modules, each responsible for a specific function and communicating through a clear interface. This design method simplifies the development process, improves the reusability and maintainability of the code, and also enhances the scalability and flexibility of the system [16]. The functions and responsibilities of each module are clear. The communication between components is realized through Props and Events to ensure the modularity and scalability of the system. Each component follows the principle of single responsibility, and realizes complex functions through combination, simplifying development, improving code maintenance, reusability, and efficient operation of the system, providing convenience for users and developers.

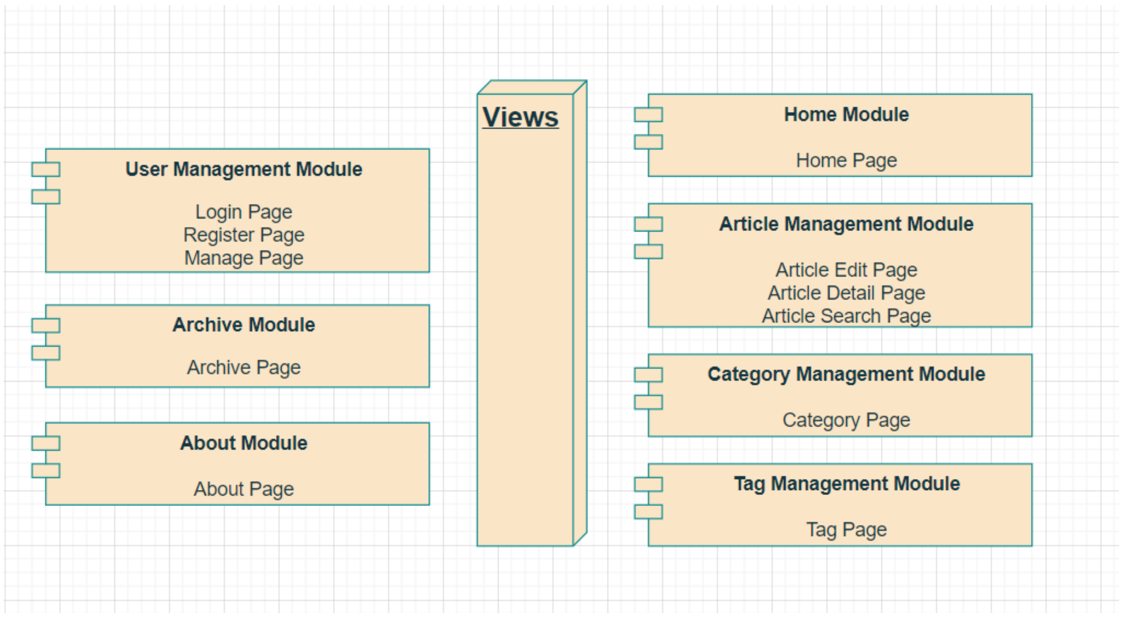


Figure 3.5 View Design and Requirement Description

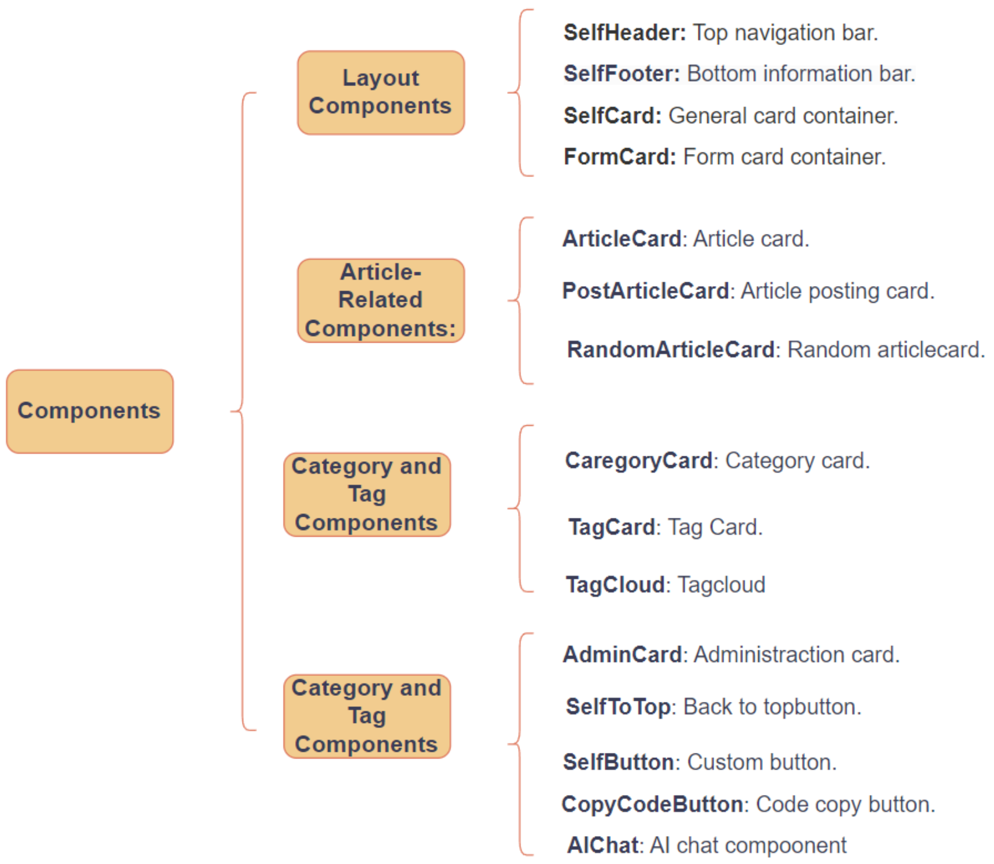

Components
Layout Components
SelfHeader: Top navigation bar.
SelfFooter: Bottom information bar.
SelfCard: General card container.
FormCard: Form card container.
Article-Related Components:
ArticleCard: Article card.
PostArticleCard: Article posting card.
RandomArticleCard: Random articlecard.
Category and Tag Components
CagoryCard: Category card.
TagCard: Tag Card.
TagCloud: Tagcloud
Category and Tag Components
AdminCard: Administraction card.
SelfToTop: Back to topbutton.
SelfButton: Custom button.
CopyCodeButton: Code copy button.
AIChat: AI chat compoonent

Figure 3.6 Component Design and Requirement Description

The back-end of this blog system is built based on Spring Boot and adopts a typical hierarchical architecture design, which is divided into control layer, service layer, data access layer, tool layer and configuration layer. The request first goes through the Filter layer for cross-domain and authentication preprocessing, and then the Controller layer receives the request, performs parameter verification, and distributes the request to the Service layer. The Service layer hosts the core business logic, and combines transaction control and caching mechanisms to ensure data consistency and system performance. The data access layer implements database operations based on MyBatis Plus and supports paging and complex queries [17]. The tools layer provides common functions such as JWT authentication, file processing, and time formatting for reuse across layers. The configuration layer centrally manages security, cross-domain, and component registration to enhance system flexibility. The global exception handling mechanism provides a unified response format to ensure interface consistency. It is also integrated with Scheduled tasks and provides monitoring support. The overall design has a clear structure and a single responsibility, which reflects good maintainability and expansibility.

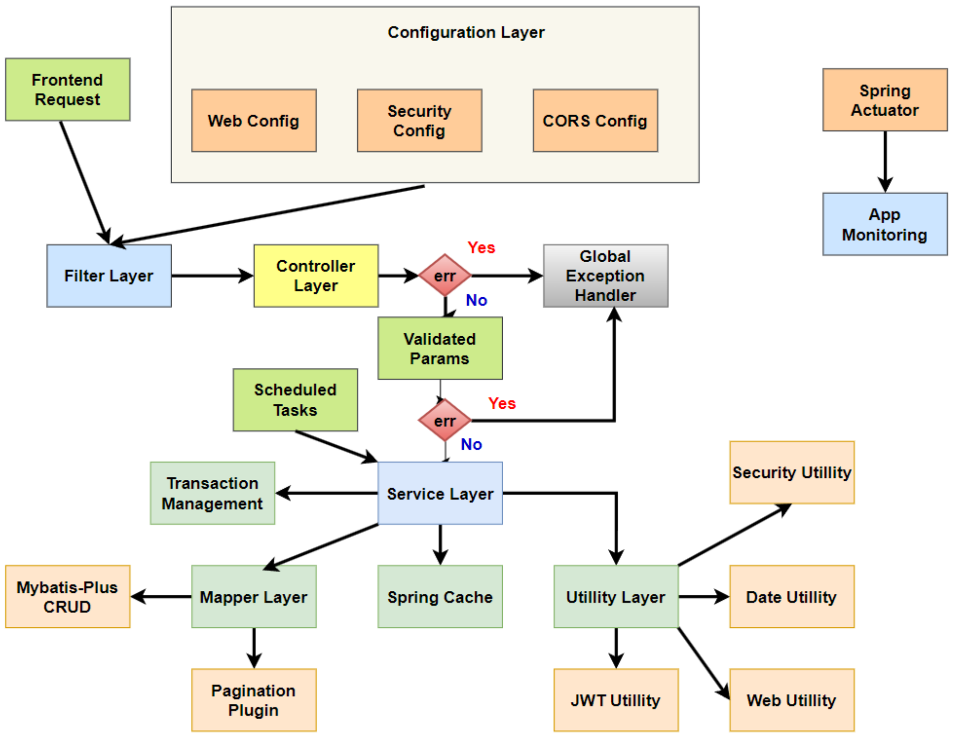
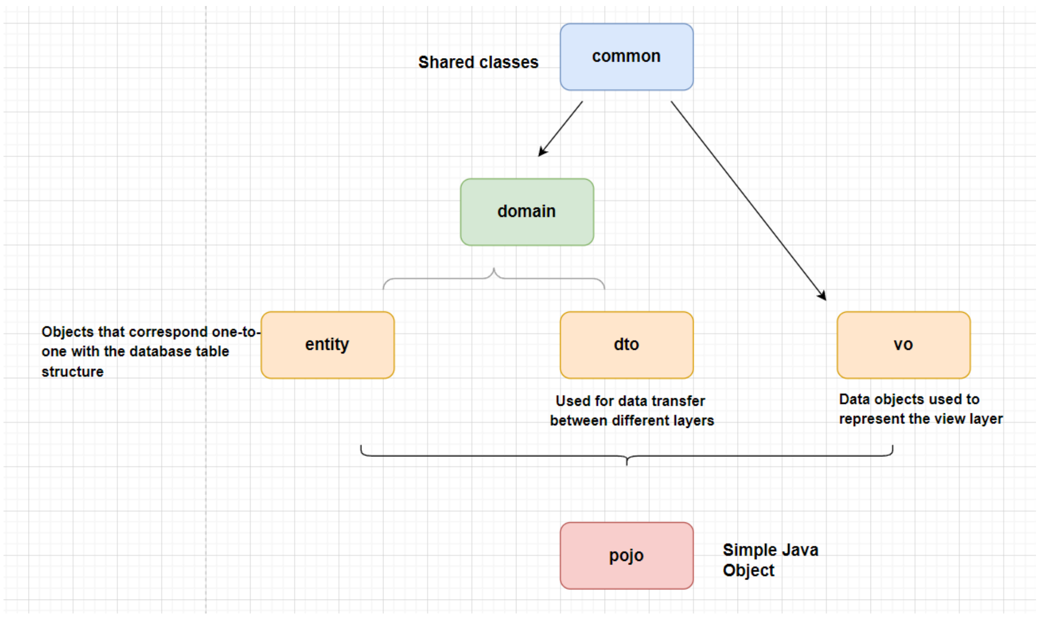
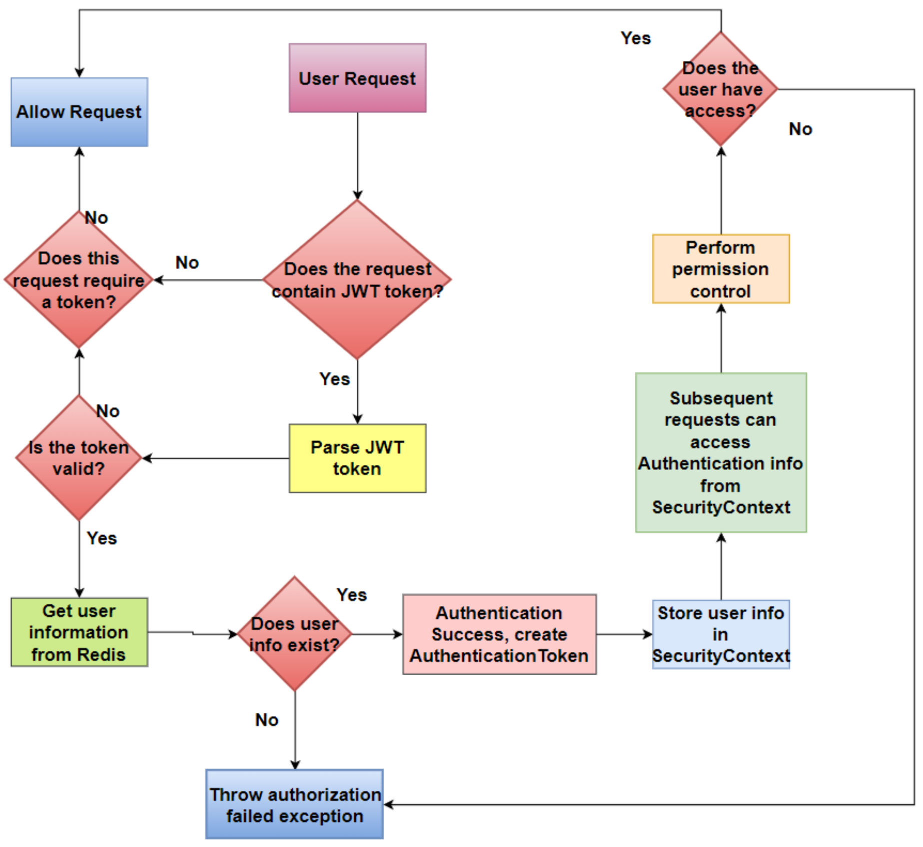


Figure 3.7 Backend Module Design Process Diagram

In the back-end design of this system, hierarchical data object management mechanism is adopted to improve the modularity, maintainability and expansibility of the system. The overall structure mainly includes three parts: common, domain and pojo. The common module is used to store globally common classes, such as unified response structure, exception handling, and enumeration types, so that modules can be reused easily. The domain module is further subdivided into entity, dto and vo, which assume different responsibilities respectively. entity is used to map the database table structure and is the core model of interaction between the persistence layer and the data table. Data Transfer Object (dto) is used to transmit data between different system layers, and performs functions such as parameter receiving and structure adaptation. vo (View Object) is used to encapsulate the data structure returned to the front end to ensure the clarity and security of the interface response data [18]. Pojos (Plain Old Java Objects) provide support for simplified data objects that encapsulate the underlying data structures. Through this clear hierarchical mechanism of data objects, not only the effective decoupling of business logic and data structure is achieved, but also the development efficiency and readability of the system are improved, in line with the best practices of modern enterprise Java projects.


Shared classes
common
domain
Objects that correspond one-to-one with the database table structure
entity
dto
Used for data transfer between different layers
vo
Data objects used to represent the view layer
pojo
Simple Java Object
Figure 3.8 Layered Data Object Management Mechanism

In order to improve the Security of blog system, the authentication mechanism based on Spring Security and JWT is designed and implemented. The system configures specific URL access permissions to allow anonymous access to login interfaces. Other interfaces require user authentication. Passwords are encrypted using BCryptPasswordEncoder for enhanced security. JWT certification by custom JwtAuthenticationTokenFilter filters, parsing the token request and verify its effectiveness. After successful authentication, the system creates UsernamePasswordAuthenticationToken object, storing user information to SecurityContext, ensure that subsequent requests authentication and authorization[19]. In addition, a custom AuthenticationEntryPoint and AccessDeniedHandler are configured to handle authentication and authorization failure exceptions. Through this design, the system realizes stateless JWT authentication mechanism, which improves the security and system performance.



User Request
Does the request contain JWT token?
No
Does this request require a token?
No
Allow Request
Yes
Parse JWT token
Is the token valid?
No
Yes
Get user information from Redis
Does user info exist?
Yes
No
Authentication Success, create AuthenticationToken
Store user info in SecurityContext
Subsequent requests can access Authentication info from SecurityContext
Perform permission control
Does the user have access?
Yes
No
Throw authorization failed exception

Figure 3.9 Secure Access Implementation

The system adopts a URL-based access control mechanism, where access paths are centrally managed in the SecurityConfig class. By combining HTTP methods with wildcard URL patterns, the system enables refined, hierarchical access control. Paths that are not explicitly configured are by default open for access, ensuring the normal operation of basic features.

The user login method uses the Transactional annotation to ensure the atomicity and consistency of the login process. Although the database is not directly modified, the login process involves identity authentication and writing to the Redis cache. If an error occurs midway, it could lead to a mismatch between the system state and the user's expectations. The transaction mechanism helps to handle exceptions uniformly, preventing inconsistencies such as "token issued but login failed," thereby enhancing the security and robustness of the system.

The system employs an IP-based global rate limiting mechanism, controlling the request frequency through a custom filter. This filter applies to the entire Spring Boot application, intercepting all HTTP requests, including web pages, APIs, and static resource access. The system limits each IP to a maximum of 50 requests within 60 seconds. Requests exceeding this limit result in a 429 status code and are denied access. This strategy effectively prevents malicious traffic and enhances the system's security and stability.

To enhance the security of password transmission on the frontend, the MD5 hashing algorithm is used to hash the user’s input password. This is implemented through a custom md5Encrypt function, which utilizes the CryptoJS library to process the plaintext password[20]. The resulting hashed password is then transmitted over the network to the backend system. On the backend, Spring Security's BCryptPasswordEncoder is used to perform a second layer of encryption and verification. This encoder is configured in the SecurityConfig class to ensure consistency in password handling. During user registration, the backend receives the hashed password and applies passwordEncoder.encode() to further encrypt it before storing it in the database. During login verification, the system uses passwordEncoder.matches() to compare the user's input with the encrypted password stored in the database, thereby ensuring the security of the system.

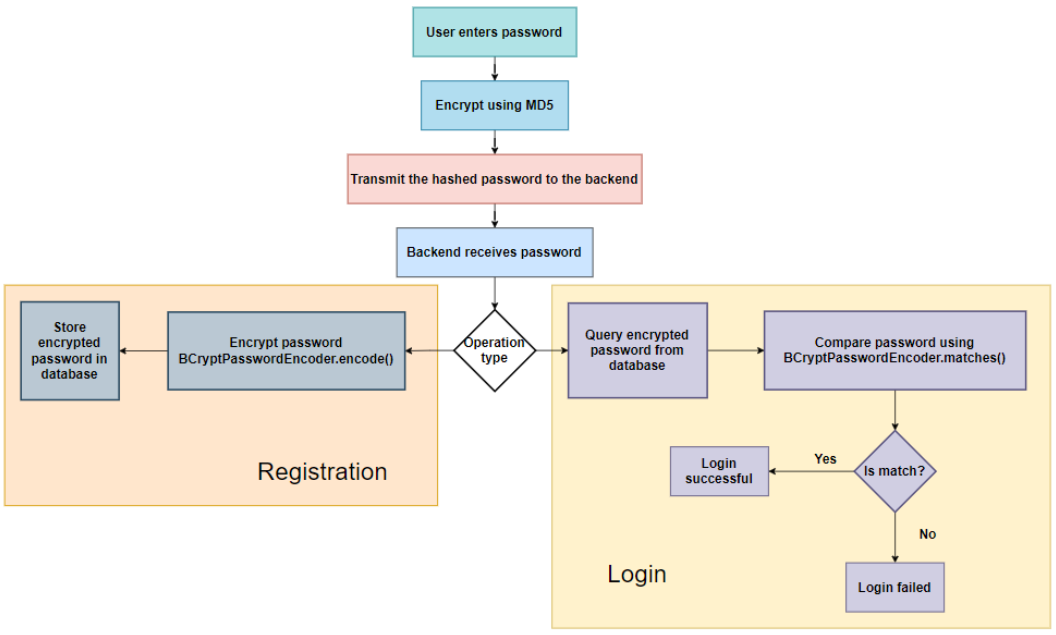

User enters password
Encrypt using MD5
Transmit the hashed password to the backend
Backend receives password
Operation type
Store encrypted password in database
Encrypt password BCryptPasswordEncoder.encode()
Registration
Query encrypted password from database
Compare password using BCryptPasswordEncoder.matches()
Is match?
Yes
Login successful
No
Login failed
Login

Figure 3.10 Password Encryption and Retrieval

3.3. Data design

RBAC (Role-Based Access Control) decouples users from permissions through "roles," enabling centralized management of permissions. In this model, users acquire permissions through the roles assigned to them, simplifying permission configuration and enhancing the system's security and maintainability. When adjusting permissions, it is only necessary to modify the role's permissions or the user's role assignments, rather than directly altering user permissions, thus improving management efficiency and flexibility.

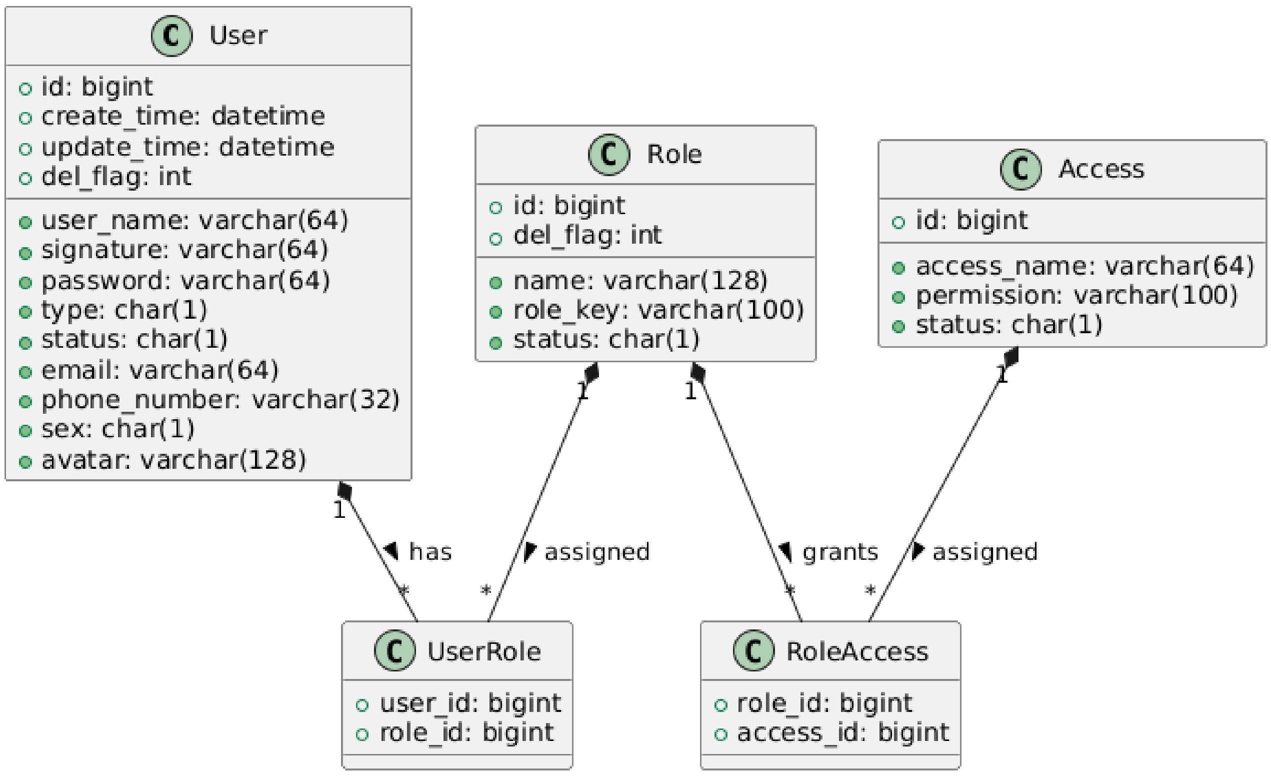


Figure 3.11 RBAC Implementation in Table Design

This system uses the relational database MySQL for data storage, according to the system function module, design a number of core data tables, covering the user, role, article, classification, authority and other entities. The logical association between data tables is established by foreign keys to ensure data integrity and scalability. The following describes the structure and design logic of some core data tables:

1) User table

The user table stores basic information about registered users, including user names, passwords, email addresses, and roles. Each user is associated with the role table by the user id.

Table 3.2 User Table Structure Design

Field Name	Data Type	Description
id	bigint	Primary Key
user_name	varchar(64)	Username
signature	varchar(64)	Signature
password	varchar(64)	Password
type	char(1)	User Type (0 Regular User, 1 Administrator)
status	char(1)	Account Status (0 Active, 1 Disabled)
email	varchar(64)	Email
phone_number	varchar(32)	Phone Number
sex	char(1)	Gender (0 Male, 1 Female, 2 Unknown)
avatar	varchar(128)	Avatar
create_time	datetime	Creation Time
update_time	datetime	Update Time
del_flag	int	Deletion Flag (0 Not Deleted, 1 Deleted)

2) Article table

It is used to store basic information about blog posts, including the title, content, summary, category, status, etc., and to record the number of views.

Table 3.3 Article Table Structure Design

Field Name	Data Type	Description
id	bigint	Primary key
title	varchar(256)	Title
content	longtext	Article Content
summary	varchar(1024)	Article Summary
category_id	bigint	Category ID
status	char(1)	Status (0 published, 1 draft)
view_count	bigint	Views
create_by	bigint	Creator ID
create_time	datetime	Creation time
update_by	bigint	Updater ID
update_time	datetime	Update time
del_flag	int	Deletion flag (0 not deleted, 1 deleted)

3) Category table

This table is used to implement the classification function of articles, supports parent-child classification structure, and facilitates the expansion of hierarchies.

Table 3.4 Category Table Structure Design

Field Name	Data Type	Description
id	bigint	Primary Key
name	varchar(128)	Category Name
pid	bigint	Parent Category ID
status	char(1)	Status (0 normal, 1 disabled)
create_by	bigint	Creator ID
create_time	datetime	Creation Time
del_flag	int	Deletion Flag (0 not deleted, 1 deleted)

4) Roles and Permissions Table

The system adopts the RBAC permission model, which assigns permissions through roles, and the many-to-many relationship between roles and users is realized through intermediate tables user_role, and the relationship between roles and permissions is realized through role_access tables.

Table 3.5 Role Table Structure Design

Field Name	Data Type	Description
id	bigint	Primary Key
name	varchar(128)	Role Name
role_key	varchar(100)	Role Permission String
status	char(1)	Role Status (0 active, 1 inactive)
del_flag	int	Deletion Flag

Table 3.6 Access Table Structure Design

Field Name	Data Type	Description
id	bigint	Primary Key
access_name	varchar(64)	Role Name
permission	varchar(100)	Role Permission String
status	char(1)	Role Status (0 active, 1 inactive)

Table 3.7 Role Access Table Structure Design

Field Name	Data Type	Description
role_id	bigint	Role ID
access_id	bigint	Permission ID

5) User role association table

This table is used to store the relationship between users and roles, and implements one-to-one relationship mapping

Table 3.8 User-Role Association Table Structure Design

Field Name	Data Type	Description
user_id	bigint	User ID
role_id	bigint	Role ID

6) Label table

This table is used to store tag information in your blog. Tags are used to categorize and tag articles, making it easy for users to find them based on their interests.

Table 3.9 Label table structure design

Field Name	Data Type	Description
id	bigint	Primary Key
name	varchar(128)	Tag Name
create_time	datetime	Creation Time
del_flag	int	Deletion Flag (0 not deleted, 1 deleted)

7) Article tag association table

This table is used to store the relationship between articles and tags to implement many-to-many relationship mapping. Each post can have multiple tags, and each tag can be owned by multiple posts.

Table 3.10 Article label association table structure design

Field Name	Data Type	Description
article_id	bigint	Article ID
tag_id	bigint	Tag ID

This system is designed based on MySQL and covers core entities such as users, articles, categories, tags, permissions, and roles. The RBAC model is implemented in conjunction with many-to-many relationship tables for user management and access control. The tables are linked through foreign keys, ensuring data consistency and scalability. The structure is clear, making the system easy to maintain and expand with additional features.

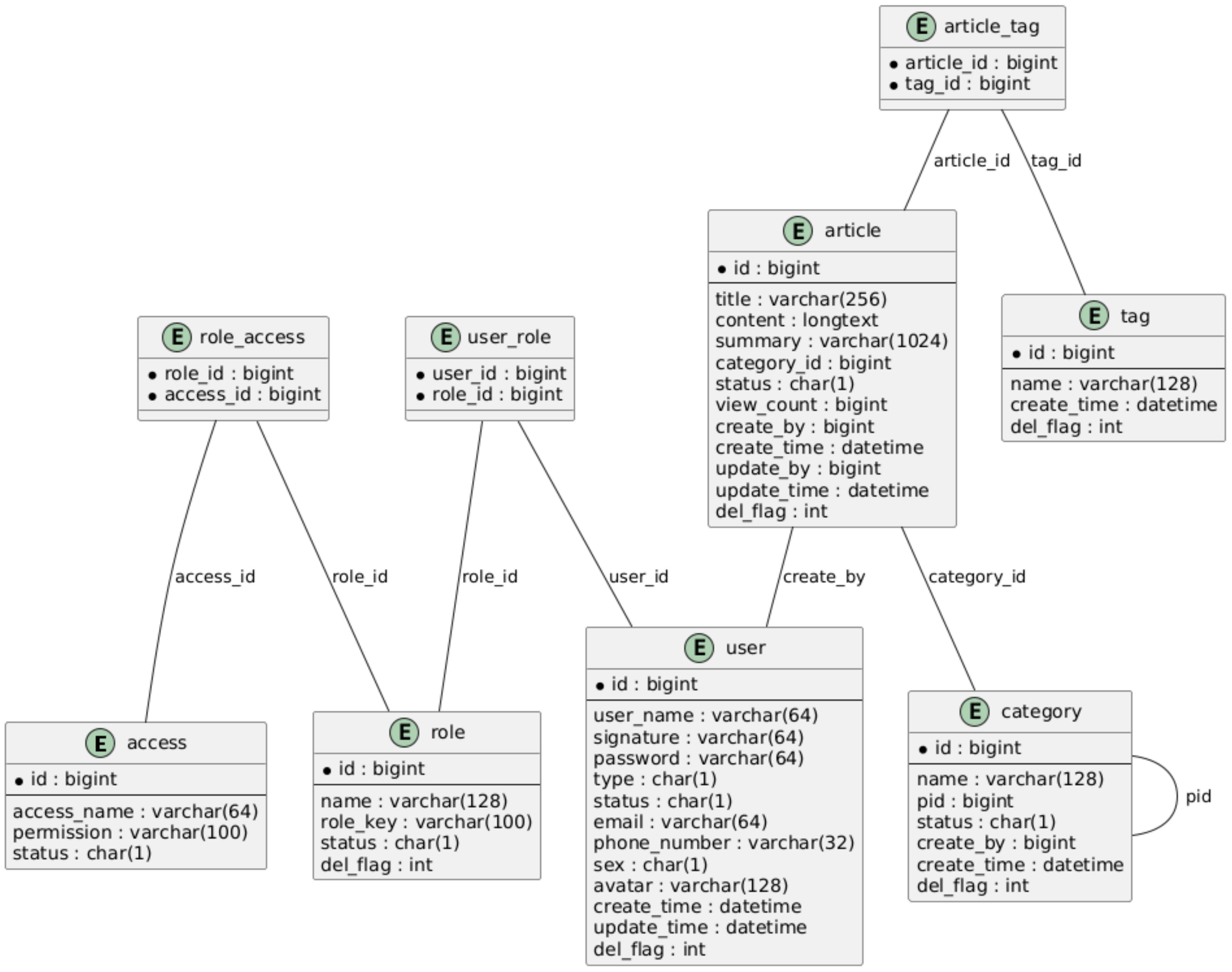


Figure 3.12 UML Diagram of Table Relationships in the Database

3.4. User Experience Design

In single-user blog design, user experience is a core element. The overall design follows the principle of simplicity and unity, aiming to provide a clear interface, convenient operation and coordinated use environment. The functional design focuses on practicality and meets the basic operation needs of users; Aesthetic design creates a comfortable visual experience by coordinating color matching and restrained style [21].

The webpage adopts a partitioned layout, and the navigation bar and the main content area are reasonably divided. The bottom footer provides information such as copyright to improve the integrity of the structure. The design incorporates the art of white space, and by consciously retaining the white space, the sense of space and hierarchy of the page is enhanced, and the reading fatigue is alleviated and the overall experience is optimized [22].

To sum up, on the basis of simplicity and unity, the blog system integrates functional practicability and visual aesthetics to form a high-quality page design with clear structure, prominent focus and clear priorities.

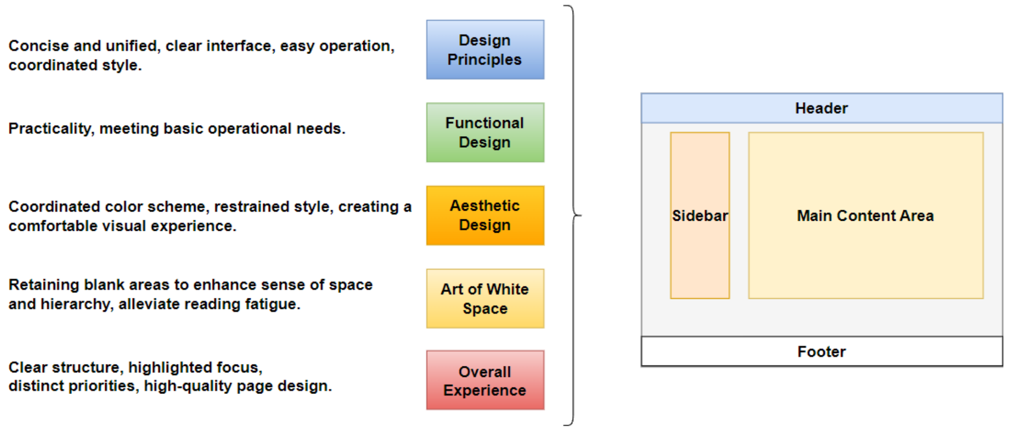

Concise and unified, clear interface, easy operation, coordinated style.
Design Principles
Practicality, meeting basic operational needs.
Functional Design
Coordinated color scheme, restrained style, creating a comfortable visual experience.
Aesthetic Design
Retaining blank areas to enhance sense of space and hierarchy, alleviate reading fatigue.
Art of White Space
Clear structure, highlighted focus, distinct priorities, high-quality page design.
Overall Experience
Header
Sidebar
Main Content Area
Footer

Figure 3.13 User Experience Design

4. Testing, Verification and Validation

4.1. Testing

1) Unit testing

In this blog system project, based on the SpringBoot back-end module, the article service is unit tested, covering core functions such as article query, statistics, detail acquisition, and page view update. Mockito simulates the dependency behavior to verify the correctness and invocation logic of each method to ensure the stability and reliability of key modules of the system [23].

Table 4.1 Unit test

Test Method Name	Test Target Description	Simulated Behavior	Assertion and validation logic explanation
testListNormalArticle	Test Function to Get List of Articles in Normal Status	articleMapper.selectList() returns a list containing testArticle.	Assert non-empty result, list length is 1, article IDs match, verify selectList() is called
testGetNormalArticleCount	Test to Get Count of Articles in Normal Status	articleMapper.selectCount() returns 1L.	Assert result is 1L, verify selectCount() is called
testGetArticleDetail	Test fetching article details	articleMapper.selectById() returns testArticle. categoryService.getById() returns testCategory. articleTagService.list() returns testArticleTag. tagService.list() returns testTag.	Assert non-empty result, status code 200, verify all four dependent methods are called
testUpdateViewCount	Test updating article view count	redisCache.increaseCacheMapValue() returns 1L.	Assert non-empty result, status code 200, verify increaseCacheMapValue() parameters and values
testGetHotArticleList	Test fetching paginated list of popular articles	articleMapper.selectPage() returns a Page object containing testArticle.	Assert non-empty result, status code 200, verify all three count methods are called

testGetArticleCount	Test fetching counts of articles, categories, and tags	articleMapper.selectCount() returns 10L. categoryService.count() returns 5L. tagService.count() returns 20L.	Assert non-empty result, status code 200, verify selectList() is called
testGetRandomArticleList	Test fetching a random list of articles	**articleMapper.selectList() returns a list of three articles.**	Assert non-empty result, status code 200, verify selectList() is called

2) Interface testing

Content interface testing is the process of functional verification, performance evaluation, and security testing of each API interface of the system [24]. In this project, the interface test covers the functions of adding, deleting, modifying, and querying articles, such as obtaining article lists, article details, article editing, deletion, and statistical information. It also includes the acquisition of tags and classifications to ensure that the system is able to respond correctly to requests and return the appropriate data. Through comprehensive testing of interfaces, the stability, performance, and security of the system are ensured, potential vulnerabilities and performance bottlenecks are avoided, and the overall user experience is improved.

Table 4.2 Interface testing

API Interface	**Description**	**Authentication method**	**Request Parameters**	**Whether successful**
GET http://localhost:8080/article/articleList?pageNum=1&pageSize=10&categoryId=2&title=df&tagId=1&date=2024/01	Get article list (with pagination and query conditions)	None	None	Success
GET http://localhost:8080/article/1	Get article details	None	None	Success
POST http://localhost:8080/article/add	Add article	Requires token	{ "title": "Test Article", "content": "Article Content", "category": "Technology", "tags": ["Java", "Spring"], "isDraft": false }	Success
PUT http://localhost:8080/article/edit	Edit article	Requires token	{ "id": 1, "title": "Modified Article", "content": "Modified Content", "category": "Technology", "tags": ["Java", "Spring", "Vue"], "isDraft": false }	Success

GET http://localhost:8080/article/count	Get article statistics	None	None	Success
PUT http://localhost:8080/article/updateViewCount/1	Update article views	None	None	Success
GET http://localhost:8080/article/previousNextArticle/1	Get adjacent articles	None	None	Success
DELETE http://localhost:8080/article/1	Delete article	Requires token	None	Success
GET http://localhost:8080/article/randomArticleList	Get random article list	None	None	Success
GET http://localhost:8080/article/search?keyword=df	Search articles by title	None	None	Success
GET http://localhost:8080/tag/tagCountList	Get tag list and article count	None	None	Success
GET http://localhost:8080/category/categoryCountList	Get category list and article count	None	None	Success

3) Front-end interaction testing

The interface test part covers the core functions of the blog system, including article addition, deletion, modification and search, pagination and condition query, page view update, adjacent article acquisition, classification and tag statistics, etc. During the test, each interface can respond correctly in both non-authentication and authentication scenarios, and the function execution results meet expectations, and the system performance is stable and reliable.

Table 4.2 Front-end interaction testing

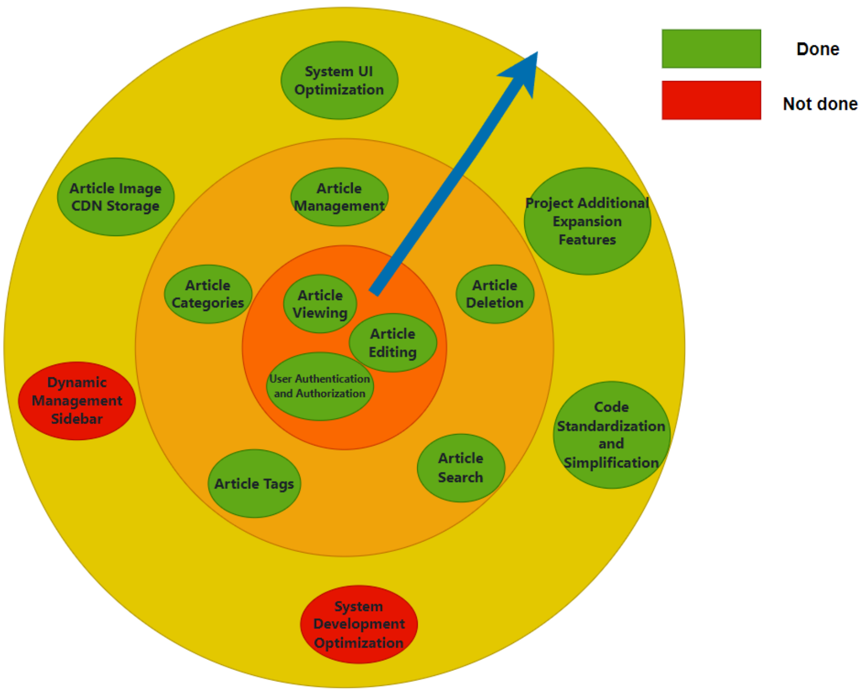
Test Module	**Test Content**	**Implementation Steps**	**Whether successful**
Login/Register Section	Login Form Test	Enter the correct username and password, verify whether successful login can be achieved	Success
		After successful login, verify whether the correct redirection to the home page occurs	Success
		Whether the login status is correctly saved in localStorage	Success

Login/ Registration Section	Registration Form Test	Validation rules for all required fields (username, email, password, confirm password)	Success
		Error message when password and confirm password do not match	Success
		Whether automatically logs in and redirects after registration	Success
Article Management	Article Filter Test	Whether the article list is correctly displayed with pagination	Success
		Whether the article list is correctly displayed with pagination	Success
Article Management	Article Filter Test	Whether the total number of articles is correctly displayed	Success
		Search by title functionality	Success
		Filter by tag functionality	Success
Article Management	Article Operation Test	Reset filter conditions functionality	Success
		Redirect functionality of the 'Write Article' button	Success
		Functionality of the article edit button	Success
Article Edit Page	Article Form Test	Article delete confirmation dialog and delete functionality	Success
		Validation for required fields of title, content, and category	Success
		Multiple selection of tags and functionality to create new tags	Success
Article Edit Page	Editor Function Test	Optional entry for summary	Success

		Basic input functionality of Markdown editor	Success
Article Detail Page	Article Display Test	Whether the preview functionality correctly renders Markdown content	Success
		Correct rendering of article content	Success
		Article views statistics functionality	Success
Article Detail Page	Article Display Test	Verification of article copyright information display	Success
		Navigation functionality for previous/next article	Success
		Click-through for article tags	Success
Access Control	Access Control Test	Display conditions for admin edit/delete buttons	Success
		Access restriction to admin pages without login	Success
		Automatic redirection after login expiration	Success
Tag Management Interaction	Tab Bar Function Test	Display control for admin privilege features	Success
Article Search Interaction	Search Box Function Test	Adding new tags on the article edit page	Success

4.2. Verification

During the development process of this project, the priority of the required functions is strictly implemented in accordance with the pre-planned requirements, so as to ensure that the project starts from the key core functions, gradually iteratively expands to the secondary function modules, and always focuses on the user experience and system stability. In the actual coding stage, strictly follow the established specifications and development systems to improve the overall project quality and consistency.



🟩 Done
🟥 Not done
System UI Optimization
Article Image CDN Storage
Article Management
Project Additional Expansion Features
Article Categories
Article Viewing
Article Editing
Article Deletion
User Authentication and Authorization
Dynamic Management Sidebar
Code Standardization and Simplification
Article Tags
Article Search
System Development Optimization

Figure 4.1 Functional completion status

The front-end part uses its static type checking feature to improve development efficiency and code security, and implements modular and component-based design ideas. All reusable interface elements and business logic components are abstracted and encapsulated into independent components and managed in a unified manner to improve code reuse rate and maintenance efficiency. Type declarations are centrally placed in the 'types' directory, which is clearly structured and type-based, making it easy for teams to collaborate and develop. In addition, the introduced third-party JavaScript libraries have been adapted to type declarations to ensure the type integrity and development consistency of the TypeScript project.

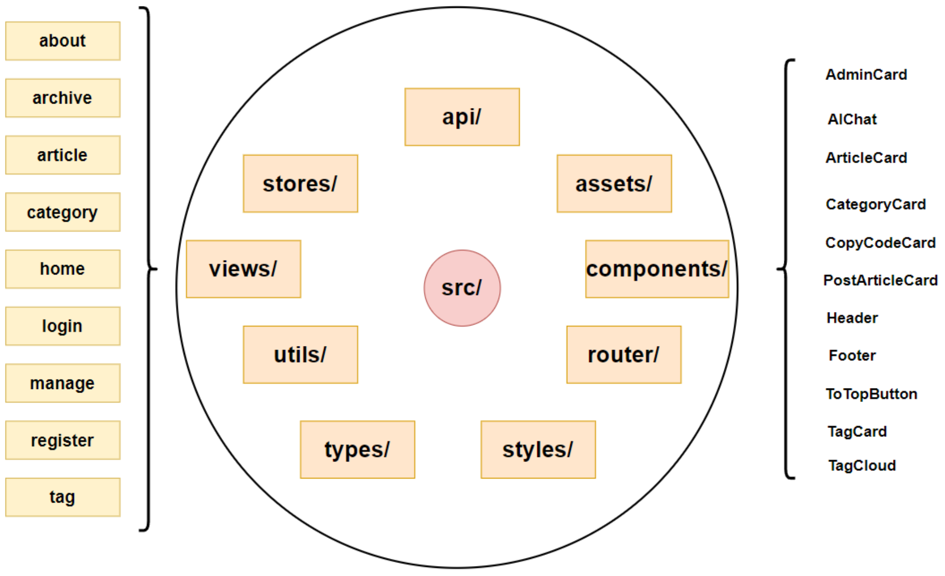


Figure 4.2 Front-end development directory structure

The back-end strictly follows the Alibaba Java Development Manual to standardize the project structure and coding style. Whether it is the division of responsibilities of the controller layer, the service layer, and the data access layer, or in the exception handling, logging, parameter verification, etc., it reflects the idea of standardization and engineering, which lays a good foundation for the maintainability and scalability of the system.

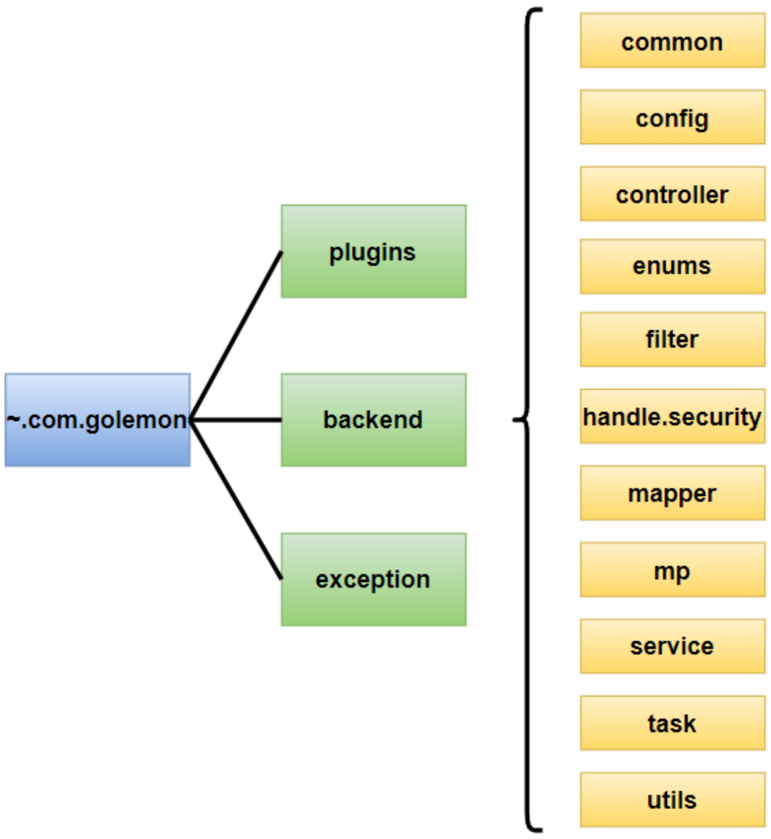
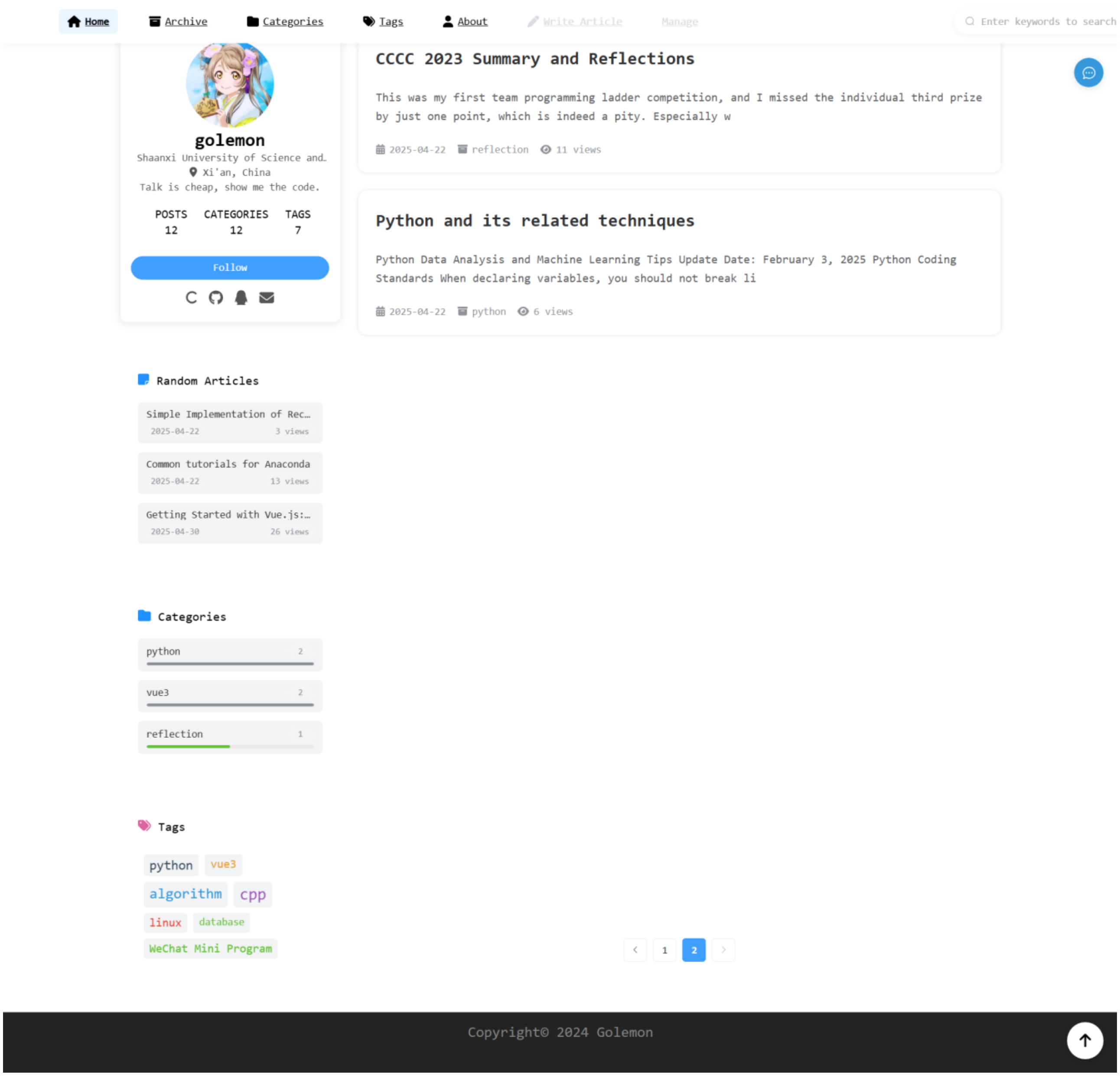


Figure 4.3 Backend development directory structure

Git is used as a version control tool throughout the project, standardizing commit information and clear branch management, ensuring orderly development and version consistency in the process. At the same time, combined with the above-mentioned development standards and process specifications, the project has reached a good level in terms of function implementation, system performance, maintainability, etc., reflecting good software engineering practice ability.

4.3. Validation

To verify the functionality and user experience of the system, this project validates the system's correctness and usability by deploying it and demonstrating the main functional pages.
When visitors access the blog system, they can browse the public content of the website without logging in. The homepage will display a list of the most recently published articles, arranged in reverse chronological order. The content is concise and clear, allowing users to quickly obtain information about the latest blog updates.


Home Archive Categories Tags About Write Article Manage Enter keywords to search

golemon
Shaanxi University of Science and…
Xi'an, China
Talk is cheap, show me the code.

POSTS	CATEGORIES	TAGS
12	12	7

Follow

CCCC 2023 Summary and Reflections

This was my first team programming ladder competition, and I missed the individual third prize by just one point, which is indeed a pity. Especially w

2025-04-22 reflection 11 views

Python and its related techniques

Python Data Analysis and Machine Learning Tips Update Date: February 3, 2025 Python Coding Standards When declaring variables, you should not break li

2025-04-22 python 6 views

Random Articles

Simple Implementation of Rec… 2025-04-22 3 views
Common tutorials for Anaconda 2025-04-22 13 views
Getting Started with Vue.js:… 2025-04-30 26 views

Categories

python 2
vue3 2
reflection 1

Tags

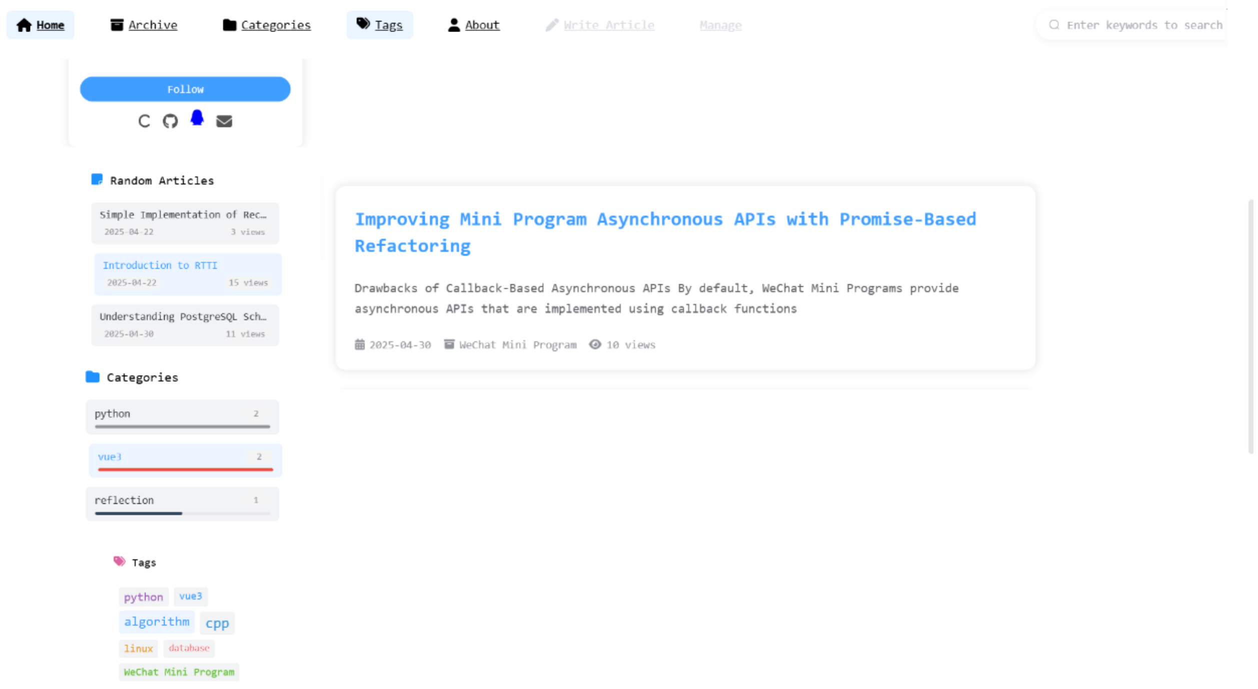
python vue3 algorithm cpp linux database WeChat Mini Program

1 2

Copyright© 2024 Golemon

Figure 4.4 Blog System Home Page

The main page includes a navigation bar, a left sidebar, a main content area, and a footer. The navigation bar is located at the top and provides basic navigation functions. The "Write Article" and "Admin Panel" options are grayed out and unavailable for non-logged-in administrators, while other options can be accessed normally to view content. The left sidebar displays the administrator's information, a list of random articles, and categories. The main content area shows the article content, and the footer displays copyright and other supplementary information. It also supports article pagination with prominent indicators to guide click actions. The administrator information, random articles, and other cards can be clicked to navigate to their respective detail pages for viewing.



Simple Implementation of Rec… 2025-04-22 3 views

Introduction to RTTI 2025-04-22 15 views

Understanding PostgreSQL Sch… 2025-04-30 11 views

python 2

vue3 2

reflection 1

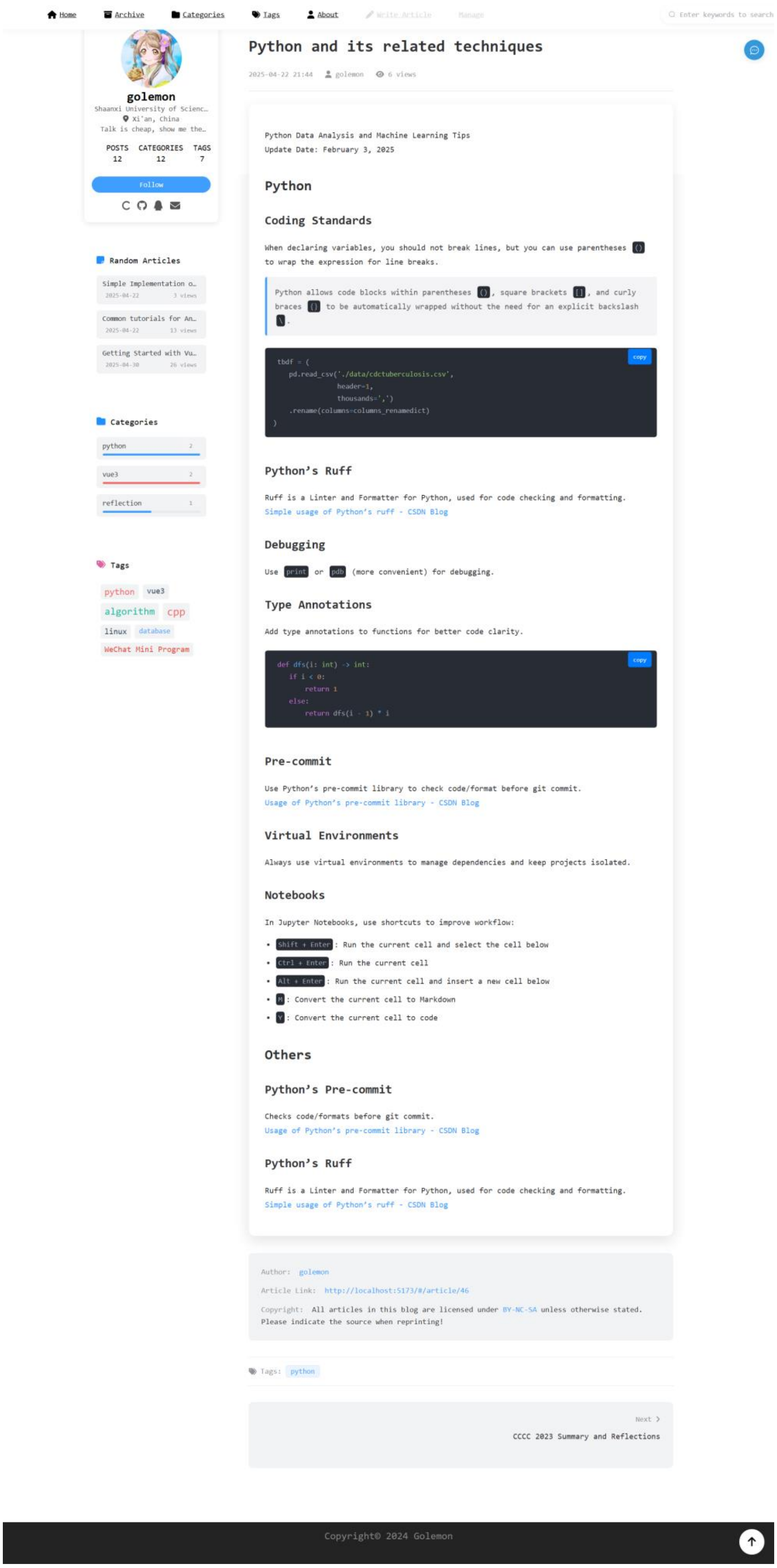
Improving Mini Program Asynchronous APIs with Promise-Based Refactoring

Drawbacks of Callback-Based Asynchronous APIs By default, WeChat Mini Programs provide asynchronous APIs that are implemented using callback functions

2025-04-30 WeChat Mini Program 10 views

Figure 4.5 Home Page Click Effects

The article detail page displays the full content, including the publication date, view count, and copyright information. The content is rendered using Markdown, supporting citations and syntax highlighting for code blocks.



Random Articles

Simple Implementation o…	
2025-04-22	3 views
Common tutorials for An…	
2025-04-22	13 views
Getting Started with Vu…	
2025-04-30	26 views

Categories

python	2
vue3	2
reflection	1

Tags

python vue3 algorithm cpp linux database WeChat Mini Program

Python and its related techniques

2025-04-22 21:44 golemon 6 views

Python Data Analysis and Machine Learning Tips
Update Date: February 3, 2025

Python

Coding Standards

When declaring variables, you should not break lines, but you can use parentheses `()` to wrap the expression for line breaks.

> Python allows code blocks within parentheses `()`, square brackets `[]`, and curly braces `{}` to be automatically wrapped without the need for an explicit backslash `\`.

```
tbdf = (
   pd.read_csv('./data/cdctuberculosis.csv',
               header=1,
               thousands=',')
   .rename(columns=columns_renamedict)
)
```

Python’s Ruff

Ruff is a Linter and Formatter for Python, used for code checking and formatting.
Simple usage of Python’s ruff - CSDN Blog

Debugging

Use `print` or `pdb` (more convenient) for debugging.

Type Annotations

Add type annotations to functions for better code clarity.

```
def dfs(i: int) -> int:
   if i < 0:
      return 1
   else:
      return dfs(i - 1) * i
```

Pre-commit

Use Python’s pre-commit library to check code/format before git commit.
Usage of Python’s pre-commit library - CSDN Blog

Virtual Environments

Always use virtual environments to manage dependencies and keep projects isolated.

Notebooks

In Jupyter Notebooks, use shortcuts to improve workflow:

- `Shift + Enter`: Run the current cell and select the cell below
- `Ctrl + Enter`: Run the current cell
- `Alt + Enter`: Run the current cell and insert a new cell below
- `M`: Convert the current cell to Markdown
- `Y`: Convert the current cell to code

Others

Python’s Pre-commit

Checks code/formats before git commit.
Usage of Python’s pre-commit library - CSDN Blog

Python’s Ruff

Ruff is a Linter and Formatter for Python, used for code checking and formatting.
Simple usage of Python’s ruff - CSDN Blog

Author: golemon
Article Link: http://localhost:5173/#/article/46
Copyright: All articles in this blog are licensed under BY-NC-SA unless otherwise stated. Please indicate the source when reprinting!

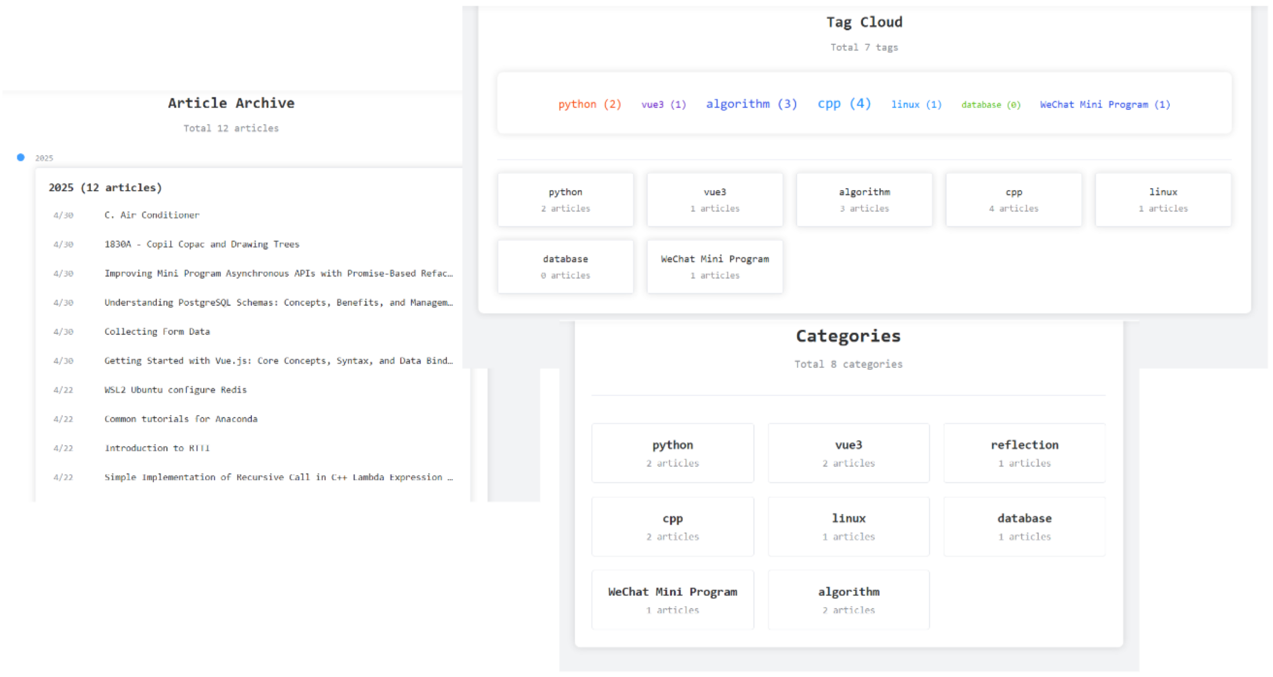
Tags: python

Next
CCCC 2023 Summary and Reflections

Copyright© 2024 Golemon

Figure 4.6 Article Detail Page

By clicking the Archive, Categories, and Tags buttons in the navigation bar, users can access the corresponding pages to view the content.



Article Archive

Total 12 articles

2025

2025 (12 articles)

```
4/30    C. Air Conditioner
4/30    1830A - Copil Copac and Drawing Trees
4/30    Improving Mini Program Asynchronous APIs with Promise-Based Refac…
4/30    Understanding PostgreSQL Schemas: Concepts, Benefits, and Managem…
4/30    Collecting Form Data
4/30    Getting Started with Vue.js: Core Concepts, Syntax, and Data Bind…
4/22    WSL2 Ubuntu configure Redis
4/22    Common tutorials for Anaconda
4/22    Introduction to RTTI
4/22    Simple Implementation of Recursive Call in C++ Lambda Expression …
```

Tag Cloud

Total 7 tags

python (2) vue3 (1) algorithm (3) cpp (4) linux (1) database (0) WeChat Mini Program (1)

python	2 articles
vue3	1 articles
algorithm	3 articles
cpp	4 articles
linux	1 articles
database	0 articles
WeChat Mini Program	1 articles

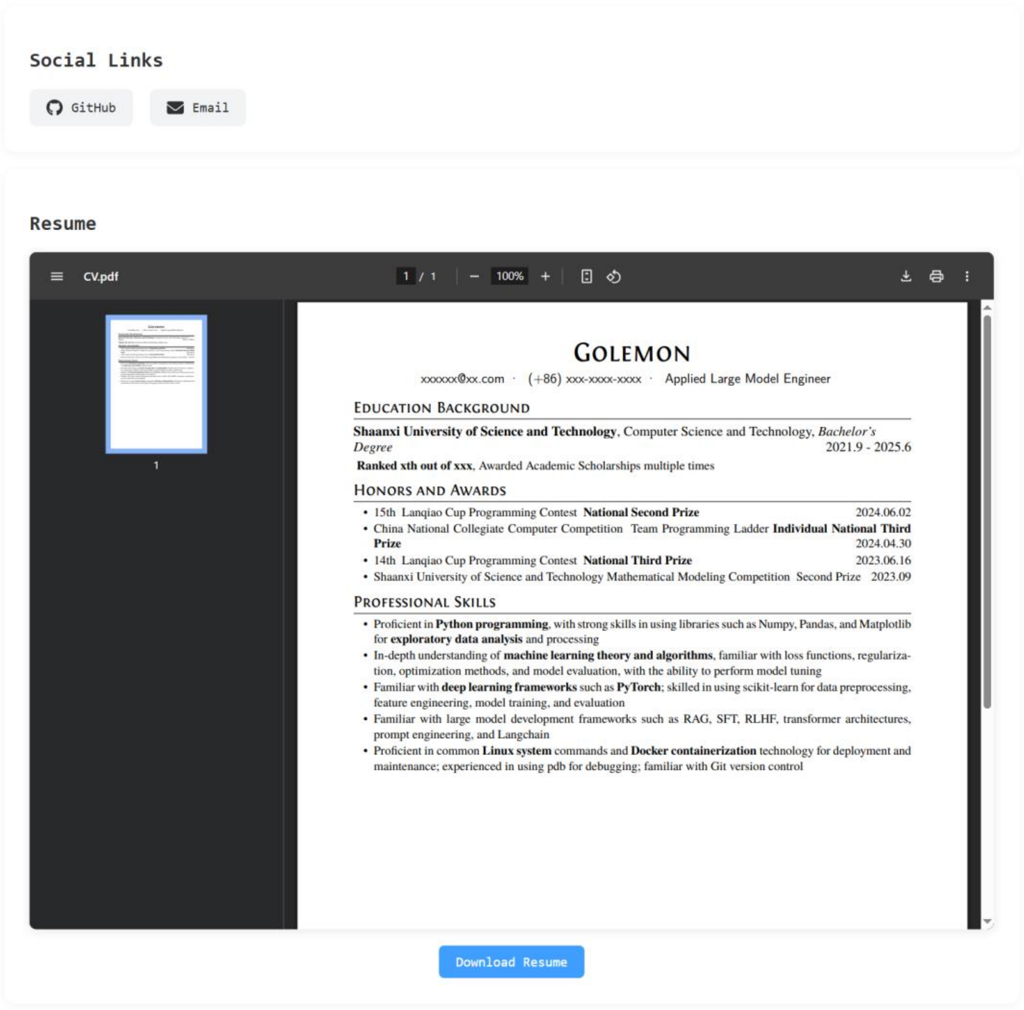
Categories

Total 8 categories

python	2 articles
vue3	2 articles
reflection	1 articles
cpp	2 articles
linux	1 articles
database	1 articles
WeChat Mini Program	1 articles
algorithm	2 articles

Figure 4.7 Blog Archive, Categories, and Tags Pages

The navigation bar includes an "About" button. Clicking it will take users to the About page, where they can view the author's basic, non-sensitive information, helping visitors get to know the site administrator.



GOLEMON

xxxxxx@xx.com · (+86) xxx-xxxx-xxxx · Applied Large Model Engineer

EDUCATION BACKGROUND

Shaanxi University of Science and Technology, Computer Science and Technology, *Bachelor's Degree* 2021.9 - 2025.6

Ranked xth out of xxx, Awarded Academic Scholarships multiple times

HONORS AND AWARDS

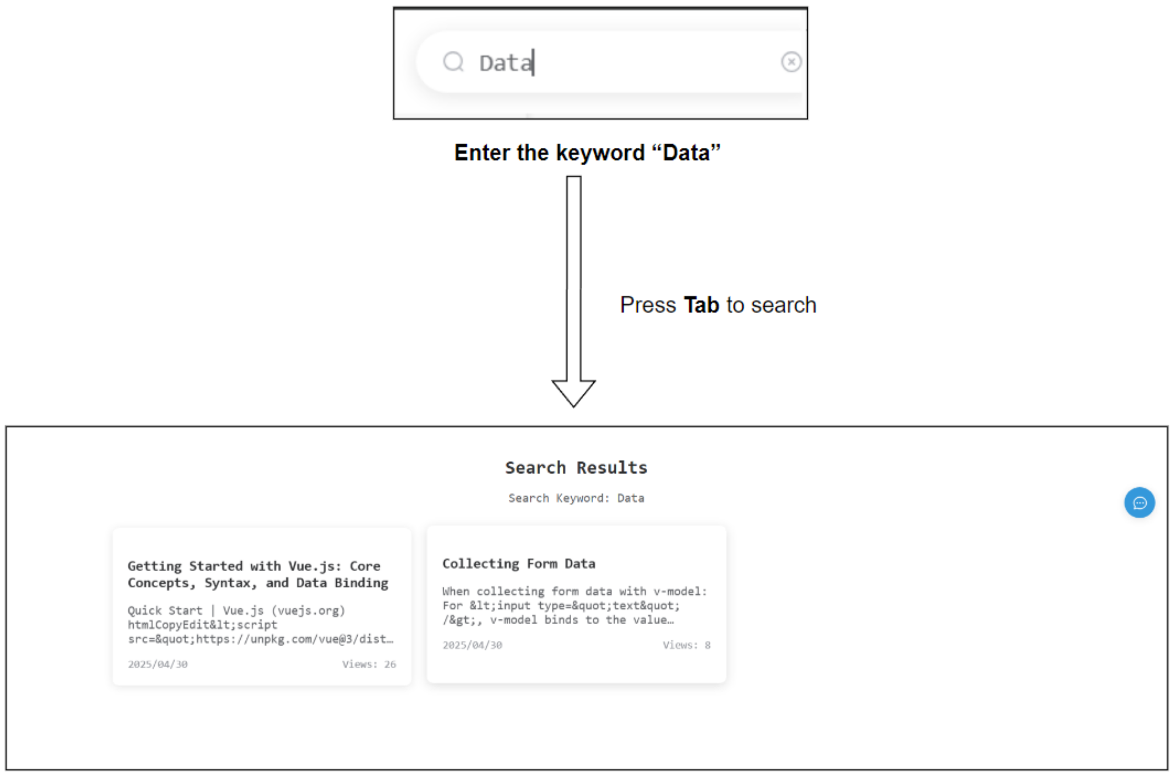
- 15th Lanqiao Cup Programming Contest **National Second Prize** 2024.06.02
- China National Collegiate Computer Competition Team Programming Ladder **Individual National Third Prize** 2024.04.30
- 14th Lanqiao Cup Programming Contest **National Third Prize** 2023.06.16
- Shaanxi University of Science and Technology Mathematical Modeling Competition Second Prize 2023.09

PROFESSIONAL SKILLS

- Proficient in **Python programming**, with strong skills in using libraries such as Numpy, Pandas, and Matplotlib for **exploratory data analysis** and processing
- In-depth understanding of **machine learning theory and algorithms**, familiar with loss functions, regularization, optimization methods, and model evaluation, with the ability to perform model tuning
- Familiar with **deep learning frameworks** such as **PyTorch**; skilled in using scikit-learn for data preprocessing, feature engineering, model training, and evaluation
- Familiar with large model development frameworks such as RAG, SFT, RLHF, transformer architectures, prompt engineering, and Langchain
- Proficient in common **Linux system** commands and **Docker containerization** technology for deployment and maintenance; experienced in using pdb for debugging; familiar with Git version control

Figure 4.8 Blog "About" Page

The right side of the navigation bar features a search bar, which supports fuzzy matching of article titles and displays brief information about related articles.



Search Results

Search Keyword: Data

Getting Started with Vue.js: Core Concepts, Syntax, and Data Binding

Quick Start | Vue.js (vuejs.org) htmlCopyEdit&lt;script src=&quot;https://unpkg.com/vue@3/dist…

2025/04/30 Views: 26

Collecting Form Data

When collecting form data with v-model: For &lt;input type=&quot;text&quot; /&gt;, v-model binds to the value…

2025/04/30 Views: 8

Figure 4.9 Blog Article Search

When clicking the "Write Blog" or "Admin Panel" buttons, if the user is not logged in, they will be automatically redirected to the login page for administrator authentication.

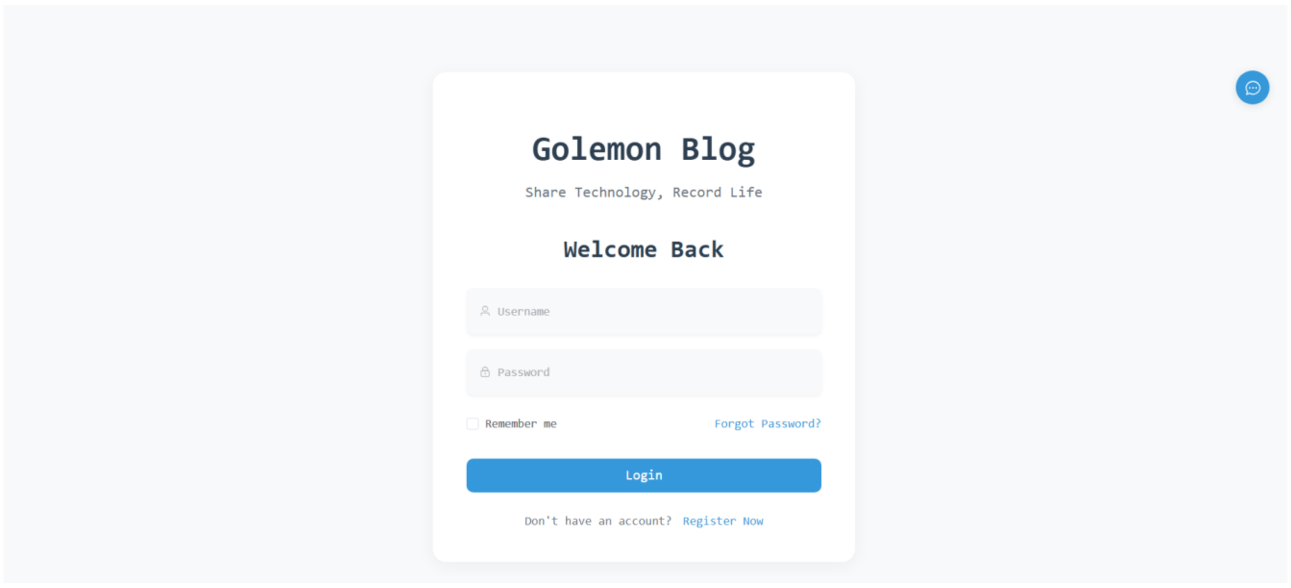
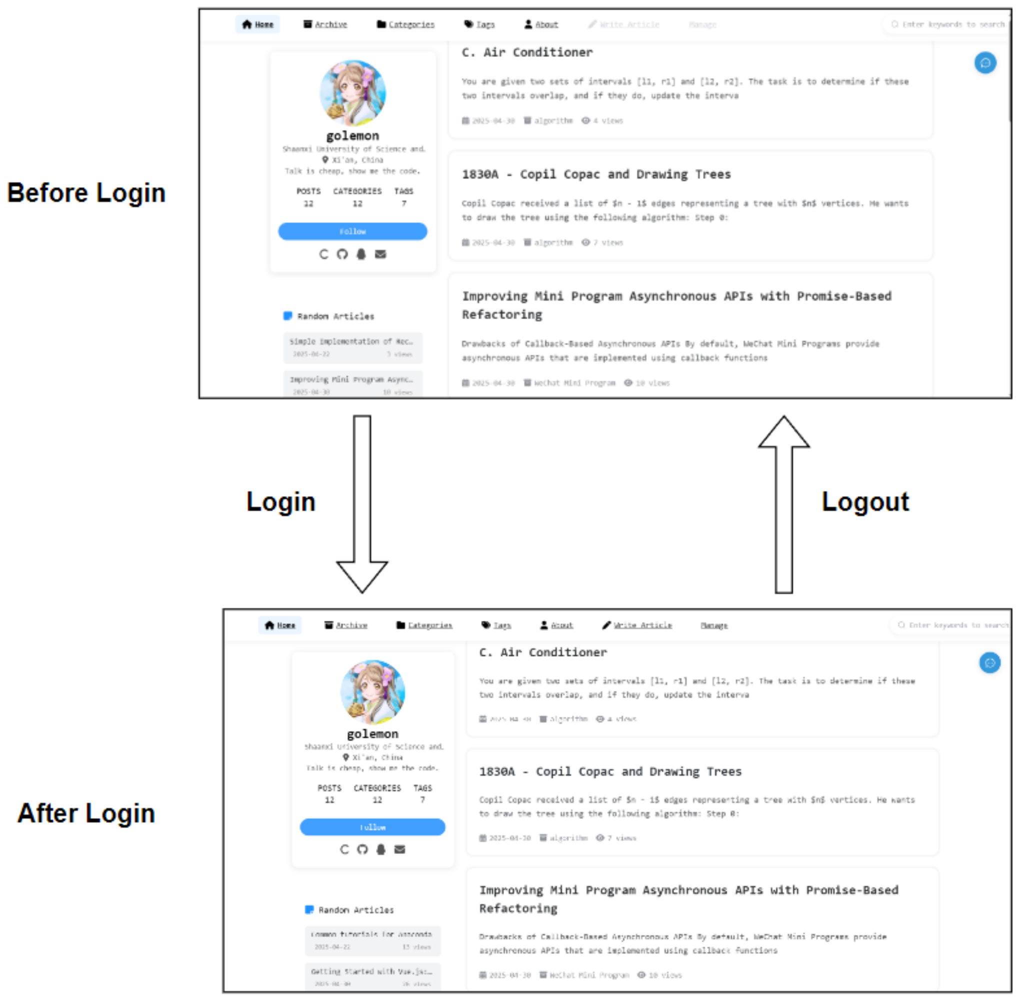


Figure 4.10 Blog Login Page

After entering the administrator username and password, the system will automatically redirect to the main page. At this point, the "Write Blog" and "Admin Panel" buttons will become available again and will no longer be displayed as grayed out.



C. Air Conditioner

You are given two sets of intervals [l1, r1] and [l2, r2]. The task is to determine if these two intervals overlap, and if they do, update the interva

2025-04-30 algorithm 4 views

1830A - Copil Copac and Drawing Trees

Copil Copac received a list of $n - 1$ edges representing a tree with n vertices. He wants to draw the tree using the following algorithm: Step 0:

2025-04-30 algorithm 7 views

Improving Mini Program Asynchronous APIs with Promise-Based Refactoring

Drawbacks of Callback-Based Asynchronous APIs By default, WeChat Mini Programs provide asynchronous APIs that are implemented using callback functions

2025-04-30 WeChat Mini Program 10 views

golemon
Shaanxi University of Science and.
Xi'an, China
Talk is cheap, show me the code.

POSTS	CATEGORIES	TAGS
12	12	7

Simple implementation of Rec...
2025-04-22 3 views

Improving Mini Program Async...
2025-04-30 10 views

Common tutorials for Anaconda
2025-04-22 13 views

Getting Started with Vue.js:...
2025-04-09 26 views

Figure 4.11 Comparison of the Blog Home Page Before and After Login

Not only does the main page change, but after successfully logging in as an administrator, the article detail page will also display "Edit" and "Delete" buttons, allowing for modification and deletion of the article content.

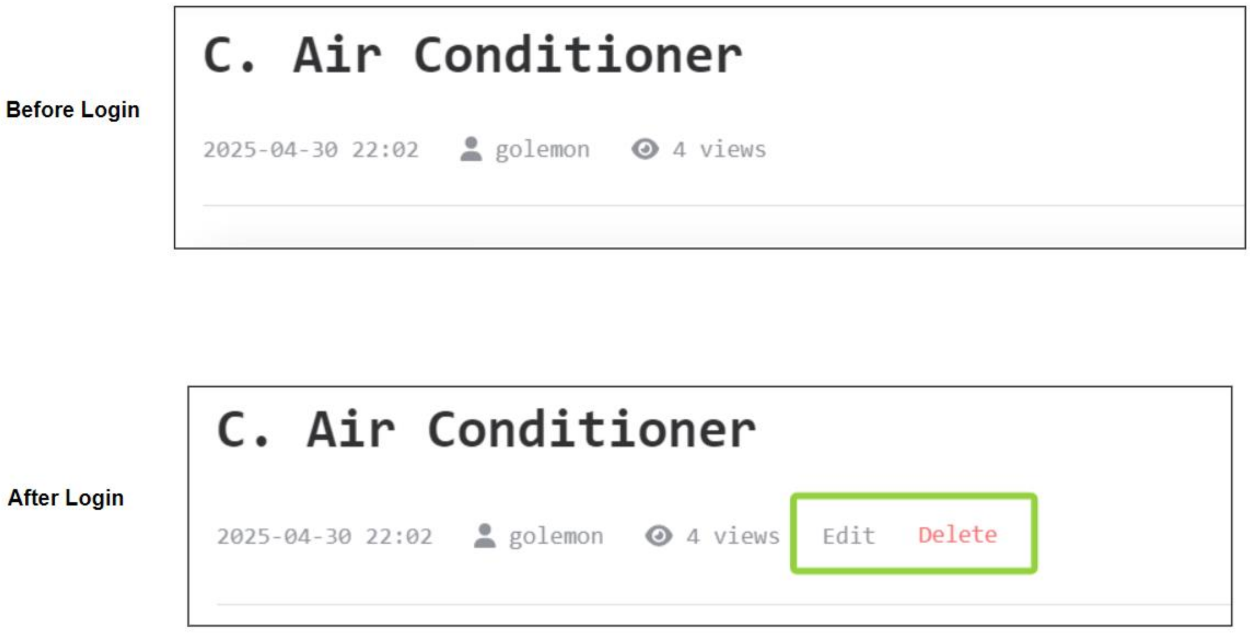
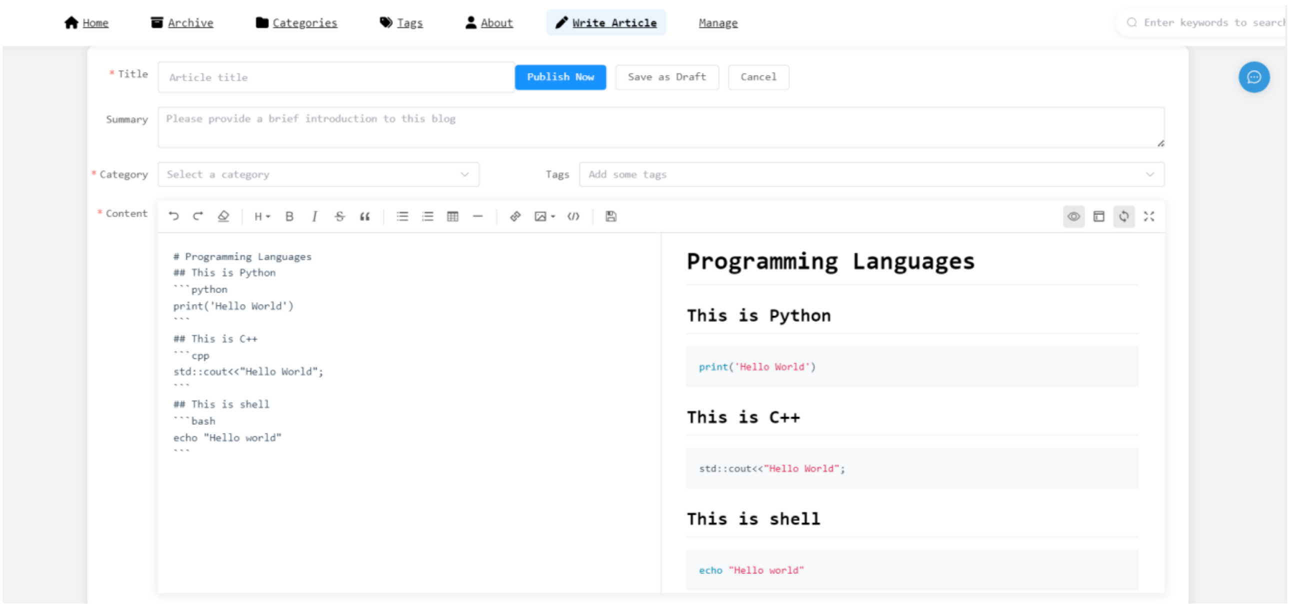

Before Login
C. Air Conditioner
2025-04-30 22:02 golemon 4 views
After Login
C. Air Conditioner
2025-04-30 22:02 golemon 4 views Edit Delete

Figure 4.12 Comparison of the Blog Article Detail Page Before and After Login

After clicking the "Write Blog" button, users can enter the blog writing interface. In this interface, Markdown syntax editing is supported, along with a visual Markdown editing feature.



~~~~
# Programming Languages
## This is Python
```python
print('Hello World')
```
## This is C++
```cpp
std::cout<<"Hello World";
```
## This is shell
```bash
echo "Hello world"
```
~~~~

Programming Languages

This is Python

```
print('Hello World')
```

This is C++

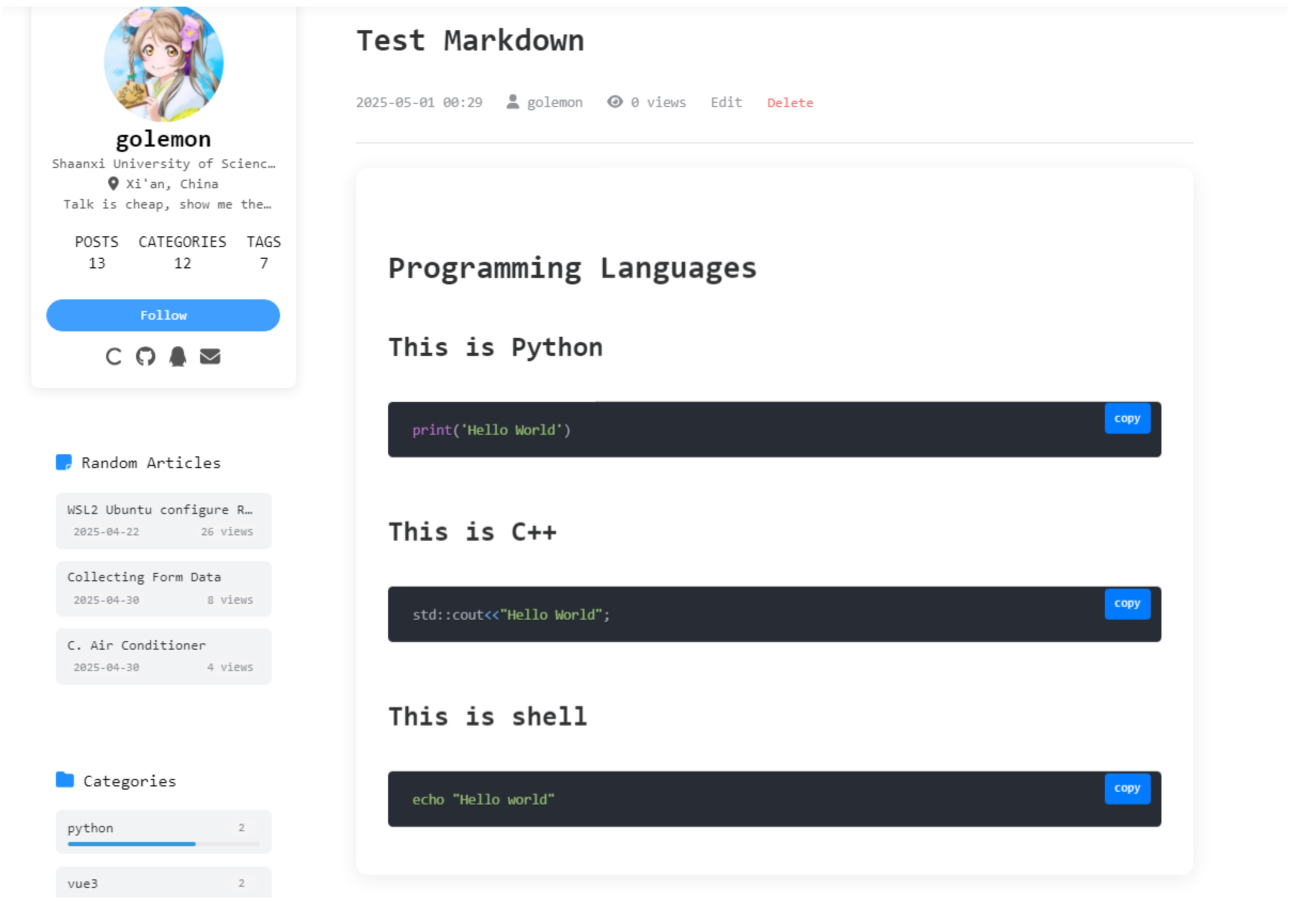
```
std::cout<<"Hello World";
```

This is shell

```
echo "Hello world"
```

Figure 4.13 Article Editing Page

After clicking the "Publish" button, users will be redirected to the newly published blog page.


golemon
Shaanxi University of Scienc…
Xi'an, China
Talk is cheap, show me the…

POSTS	CATEGORIES	TAGS
13	12	7

Random Articles

WSL2 Ubuntu configure R… 2025-04-22 26 views
Collecting Form Data 2025-04-30 8 views
C. Air Conditioner 2025-04-30 4 views

Categories

python 2
vue3 2

Test Markdown

2025-05-01 00:29 golemon 0 views Edit Delete

Programming Languages

This is Python

```
print('Hello World')
```

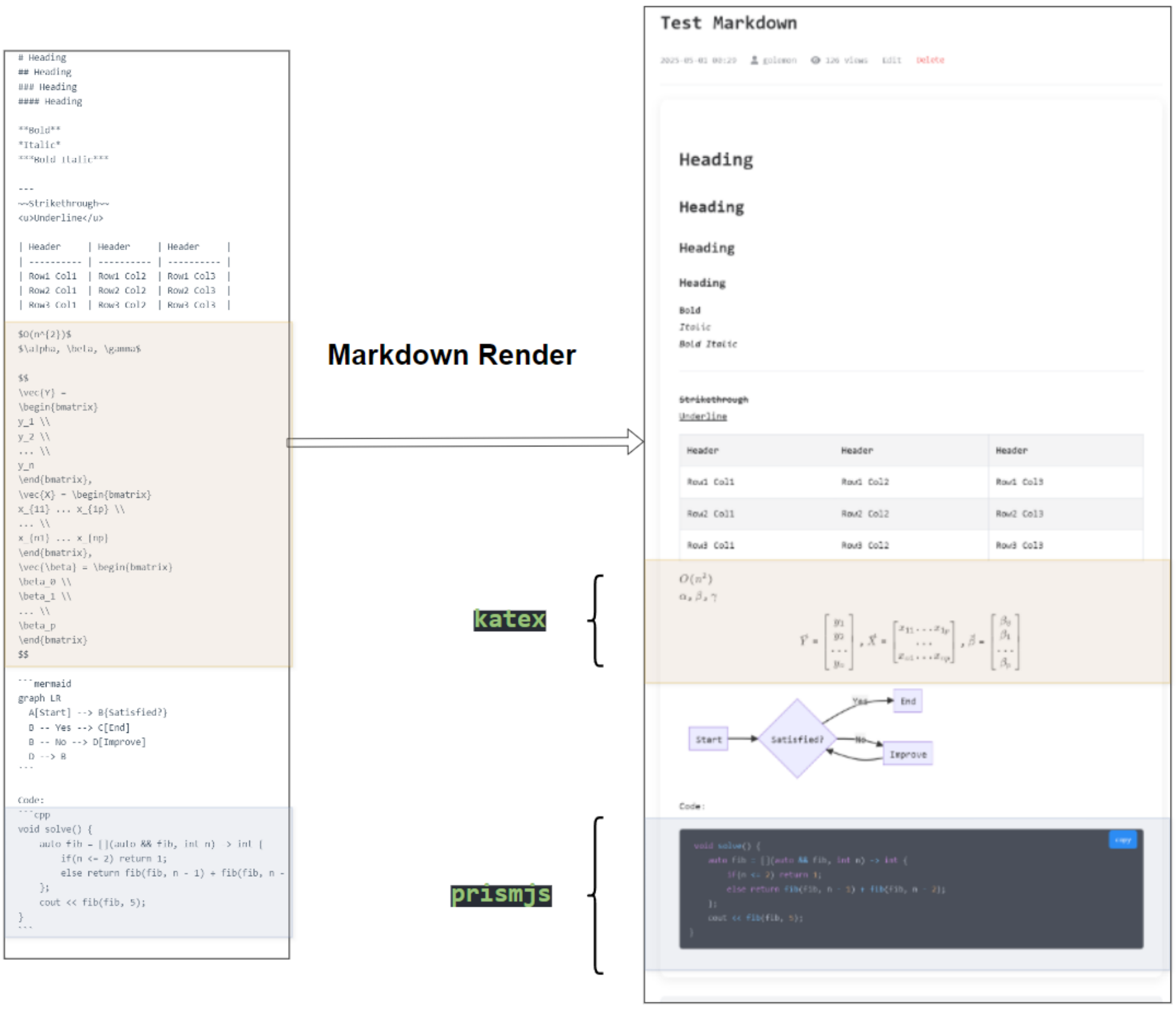
This is C++

```
std::cout<<"Hello World";
```

This is shell

```
echo "Hello world"
```

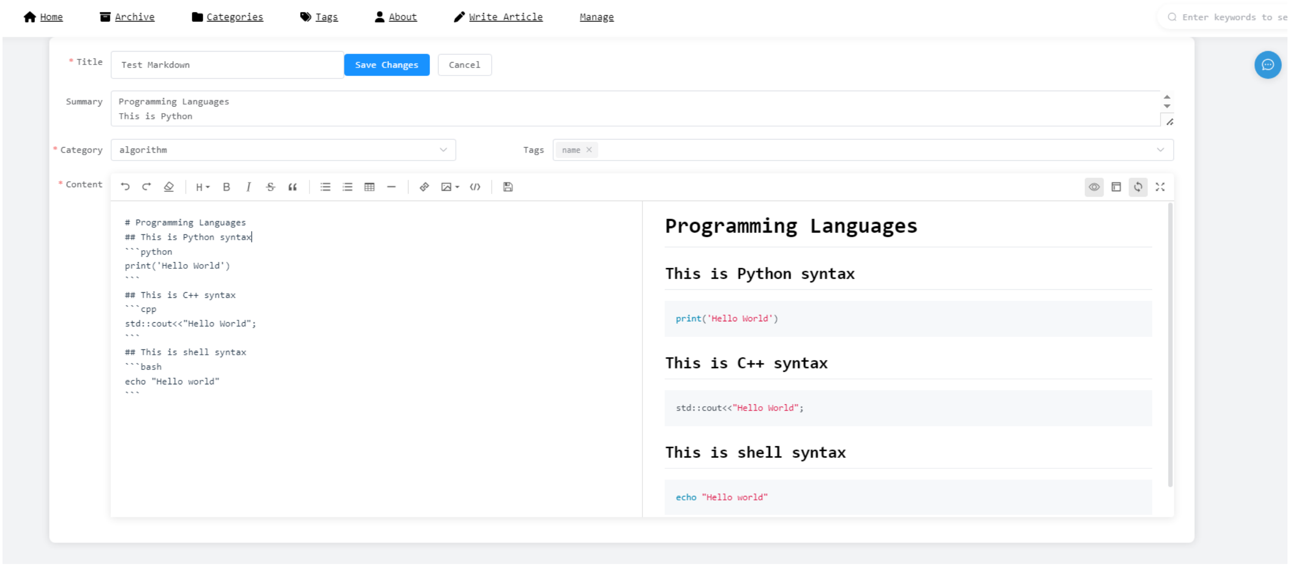
Figure 4.14 Published Article Detail Page



~~~
# Heading
## Heading
### Heading
#### Heading

**Bold**
*Italic*
***Bold Italic***

---
~~Strikethrough~~
<u>Underline</u>

| Header     | Header     | Header     |
| ---------- | ---------- | ---------- |
| Row1 Col1  | Row1 Col2  | Row1 Col3  |
| Row2 Col1  | Row2 Col2  | Row2 Col3  |
| Row3 Col1  | Row3 Col2  | Row3 Col3  |

$O(n^{2})$
$\alpha, \beta, \gamma$

$$
\vec{Y} =
\begin{bmatrix}
y_1 \\
y_2 \\
... \\
y_n
\end{bmatrix},
\vec{X} = \begin{bmatrix}
x_{11} ... x_{1p} \\
... \\
x_{n1} ... x_{np}
\end{bmatrix},
\vec{\beta} = \begin{bmatrix}
\beta_0 \\
\beta_1 \\
... \\
\beta_p
\end{bmatrix}
$$

```mermaid
graph LR
  A[Start] --> B{Satisfied?}
  B -- Yes --> C[End]
  B -- No --> D[Improve]
  D --> B
```

Code:
```cpp
void solve() {
    auto fib = [](auto && fib, int n) -> int {
        if(n <= 2) return 1;
        else return fib(fib, n - 1) + fib(fib, n -
    };
    cout << fib(fib, 5);
}
```
~~~

Figure 4.15 Markdown Rendering Preview

Clicking the "Edit" button on the blog detail page allows users to modify the blog.



* Title: Test Markdown

Summary: Programming Languages
This is Python

* Category: algorithm

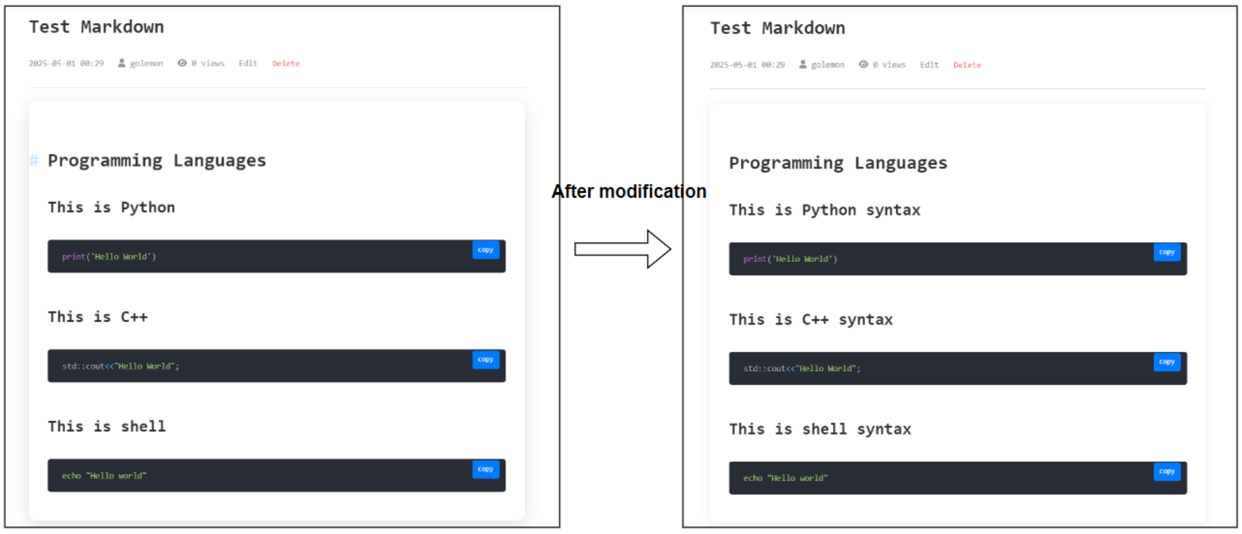
Tags: name ×

* Content:

~~~
# Programming Languages
## This is Python syntax
```python
print('Hello World')
```
## This is C++ syntax
```cpp
std::cout<<"Hello World";
```
## This is shell syntax
```bash
echo "Hello world"
```
~~~

Figure 4.16 Article Editing Page

After making modifications and publishing, the system will redirect to the article page, where the content will be updated to the latest version.



Test Markdown

2025-05-01 00:29 golemon 0 views Edit Delete

Programming Languages

This is Python

```
print('Hello World')
```

This is C++

```
std::cout<<"Hello World";
```

This is shell

```
echo "Hello world"
```

Test Markdown

2025-05-01 00:29 golemon 0 views Edit Delete

Programming Languages

This is Python syntax

```
print('Hello World')
```

This is C++ syntax

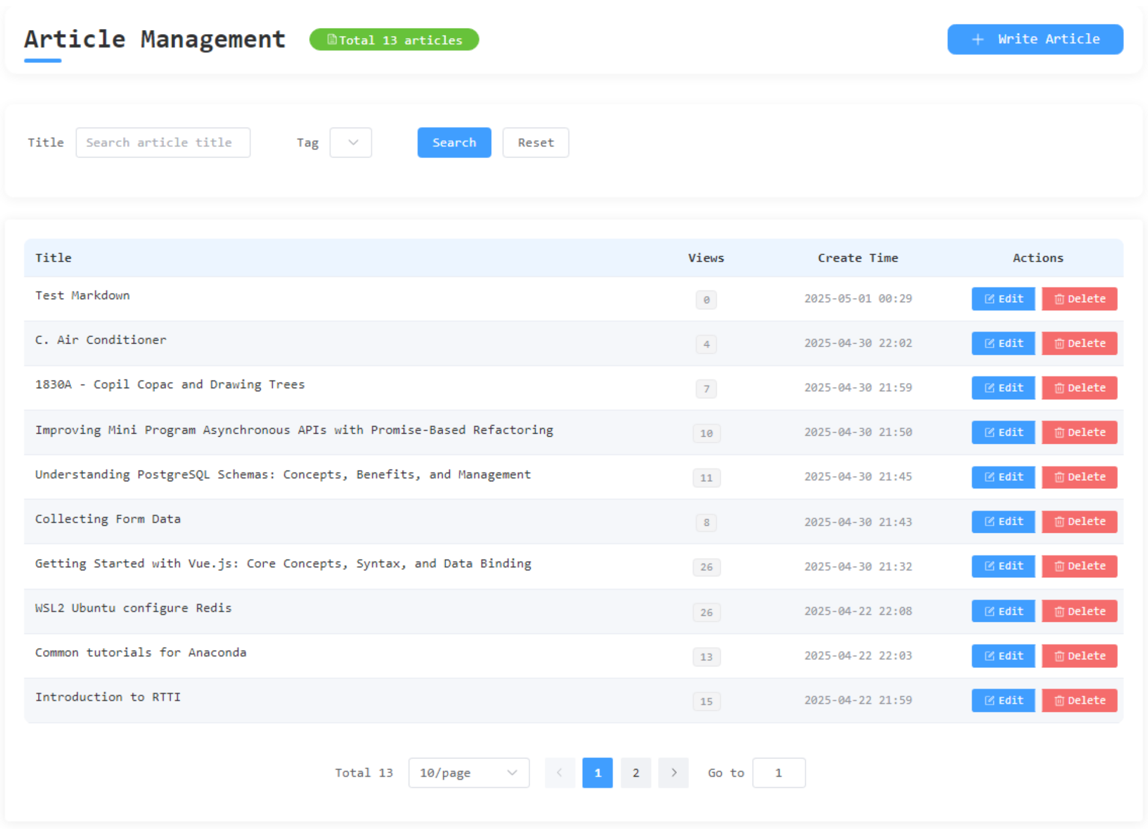
```
std::cout<<"Hello World";
```

This is shell syntax

```
echo "Hello world"
```

Figure 4.17 Comparison of the Article Before and After Editing

Clicking "Admin Panel" allows access to the backend, where detailed blog data can be viewed, and operations such as deletion can be performed.

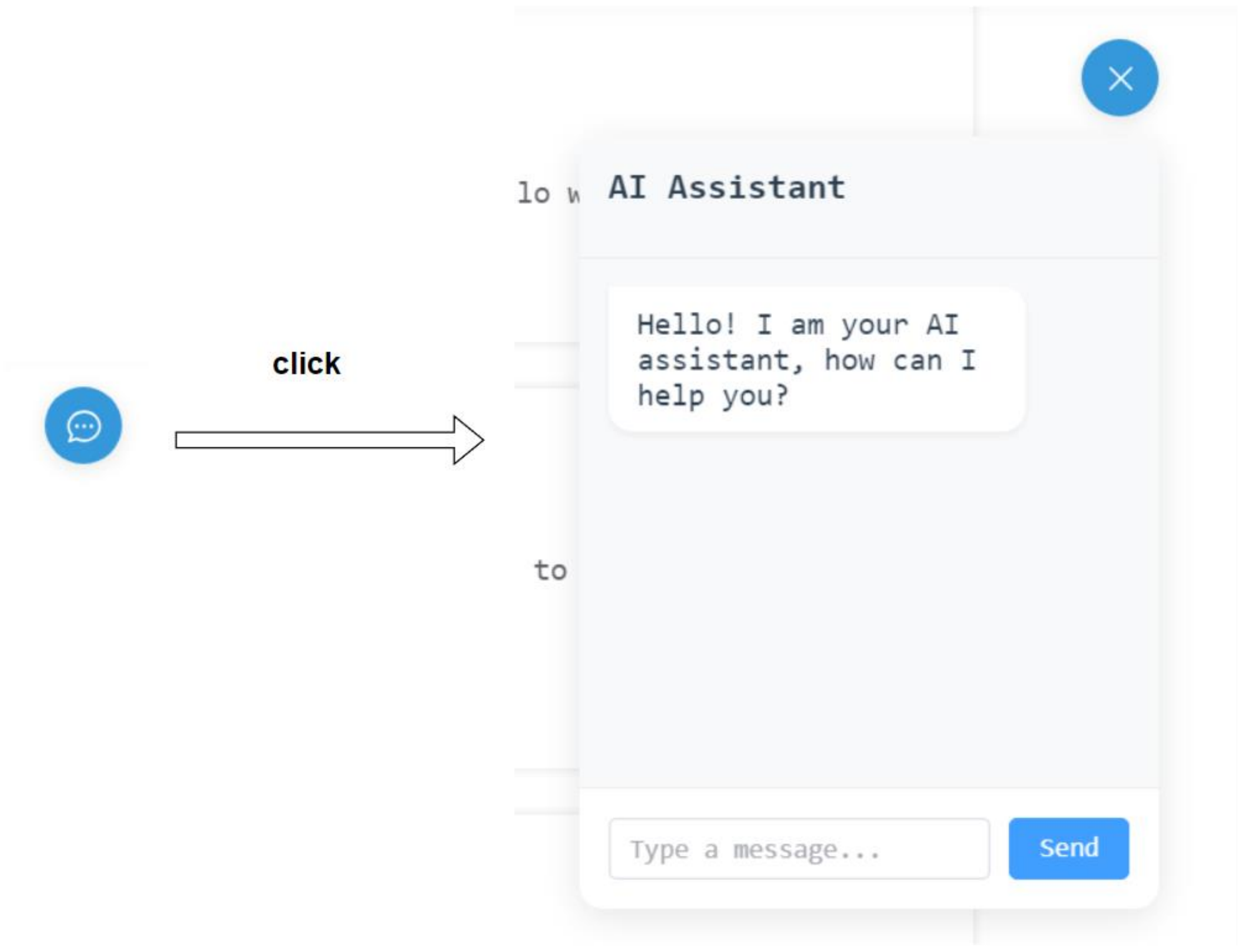


Title	Views	Create Time	Actions
Test Markdown	0	2025-05-01 00:29	Edit Delete
C. Air Conditioner	4	2025-04-30 22:02	Edit Delete
1830A - Copil Copac and Drawing Trees	7	2025-04-30 21:59	Edit Delete
Improving Mini Program Asynchronous APIs with Promise-Based Refactoring	10	2025-04-30 21:50	Edit Delete
Understanding PostgreSQL Schemas: Concepts, Benefits, and Management	11	2025-04-30 21:45	Edit Delete
Collecting Form Data	8	2025-04-30 21:43	Edit Delete
Getting Started with Vue.js: Core Concepts, Syntax, and Data Binding	26	2025-04-30 21:32	Edit Delete
WSL2 Ubuntu configure Redis	26	2025-04-22 22:08	Edit Delete
Common tutorials for Anaconda	13	2025-04-22 22:03	Edit Delete
Introduction to RTTI	15	2025-04-22 21:59	Edit Delete

Figure 4.18 Blog System Admin Panel Page

The blog system also integrates a large model API conversation feature, set as a global component. A message button is located on the right side of the page, and clicking it expands the chat interface.

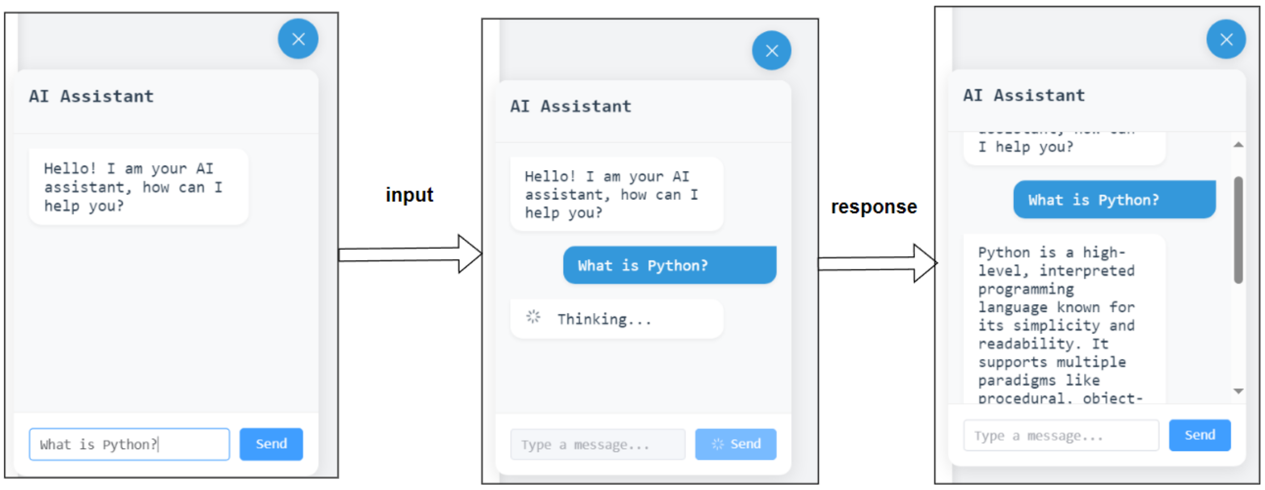
Through the backend prompt engineering, the system can effectively answer questions related to programming skills, enhancing the user interaction experience.



Hello! I am your AI assistant, how can I help you?

Figure 4.19 Large Model Chat Component

After entering the information, a brief response time will yield an answer generated by the large model.



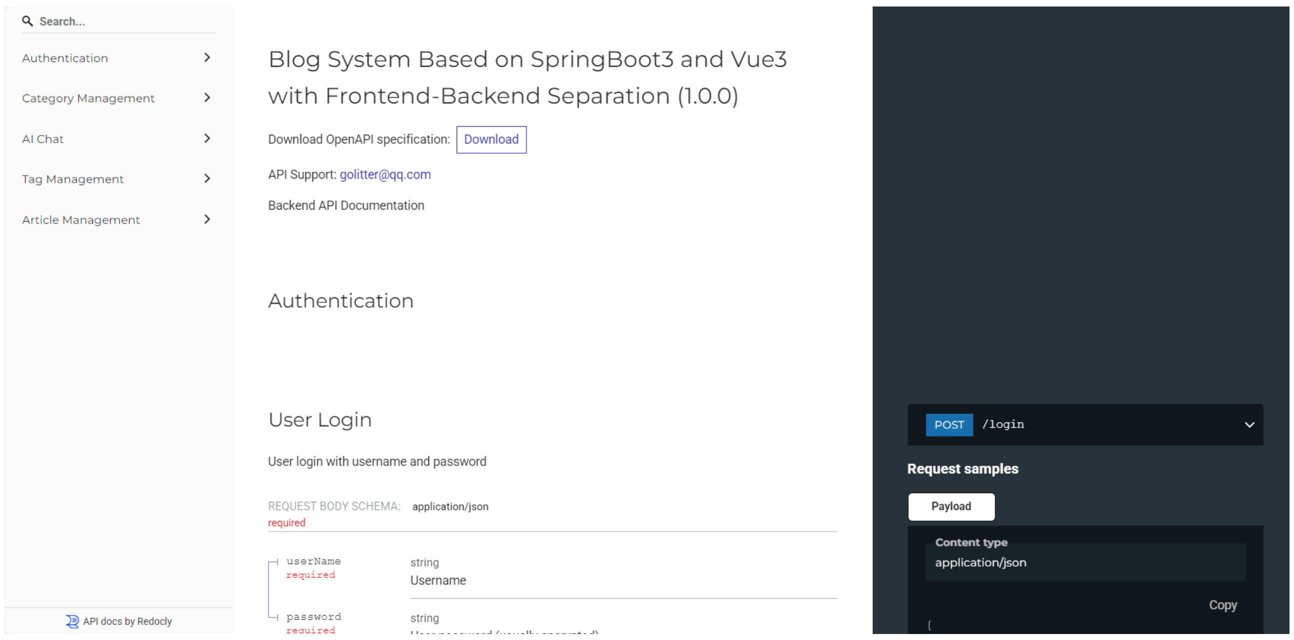
Hello! I am your AI assistant, how can I help you?

What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple paradigms like procedural, object-

Figure 4.20 Dialogue Process of the Large Model Chat Component

In addition, the API documentation is visually presented using the API archiving feature provided by Redocly.



Blog System Based on SpringBoot3 and Vue3 with Frontend-Backend Separation (1.0.0)

Download OpenAPI specification: Download

API Support: golitter@qq.com

Backend API Documentation

Authentication

User Login

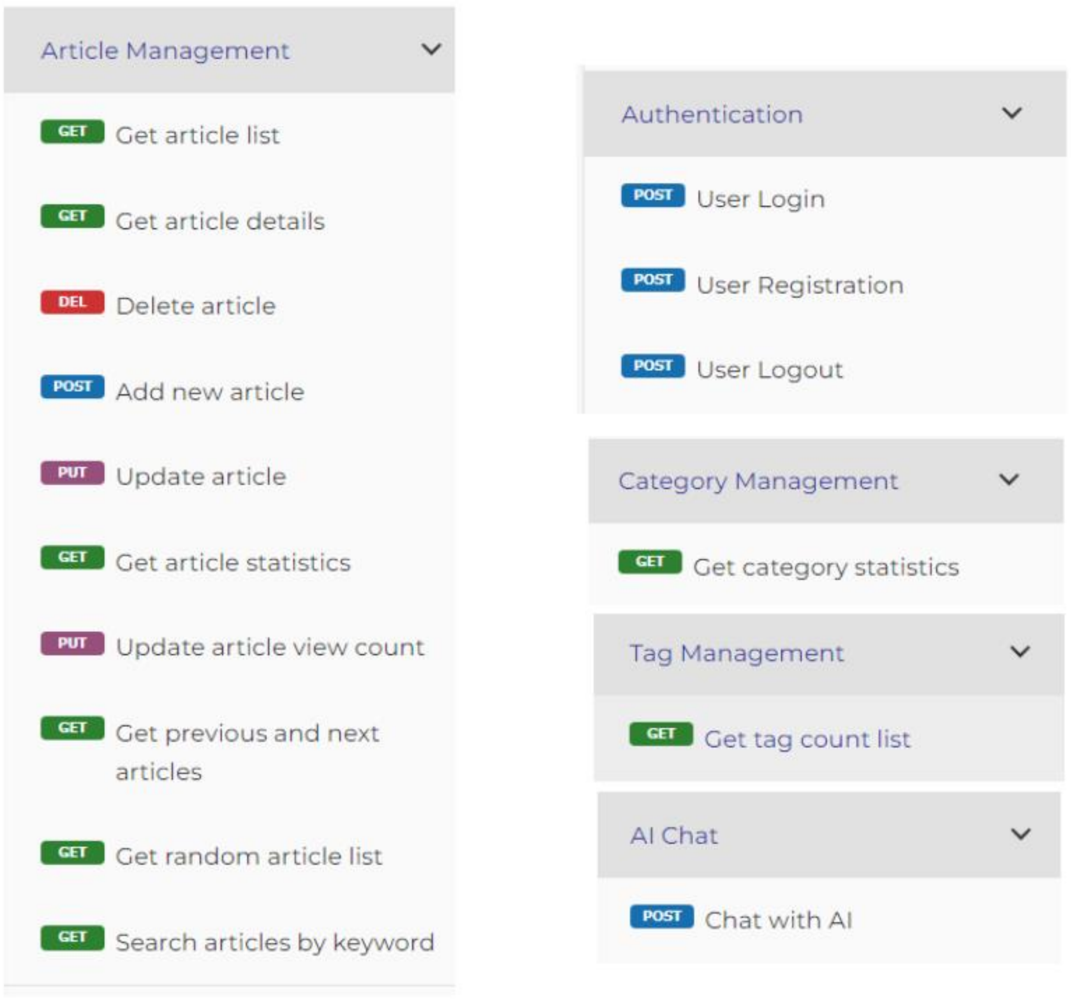
User login with username and password

REQUEST BODY SCHEMA: application/json
required

userName required	string Username
password required	string

Content type
application/json

Figure 4.21 Redocly API Documentation Homepage


Article Management

GET Get article list

GET Get article details

DEL Delete article

POST Add new article

PUT Update article

GET Get article statistics

PUT Update article view count

GET Get previous and next articles

GET Get random article list

GET Search articles by keyword

Authentication

POST User Login

POST User Registration

POST User Logout

Category Management

GET Get category statistics

Tag Management

GET Get tag count list

AI Chat

POST Chat with AI

Figure 4.22 API Documentation Modules and Content

In the security verification module of this system, the implementation status of various security mechanisms has been comprehensively validated. The system successfully integrates a stateless authentication framework based on Spring Security and JWT, effectively ensuring reliable verification of administrator identities. Fine-grained access control is achieved through URL-based permission configuration. On the front end, user passwords are encrypted using the MD5 algorithm, while on the back end, additional encryption and verification are performed with BCryptPasswordEncoder, ensuring the security of user credentials during transmission and storage. Furthermore, the system is equipped with customized authentication and authorization exception handlers to standardize error responses and prevent leakage of sensitive data. An IP-based rate-limiting filter has also been implemented to restrict access frequency within a specified time frame, thereby mitigating malicious request flooding. The login process uses the Transactional annotation to ensure consistency and atomicity of all operations. Functional validation confirms that all these security mechanisms have been successfully deployed and are operating stably as designed.

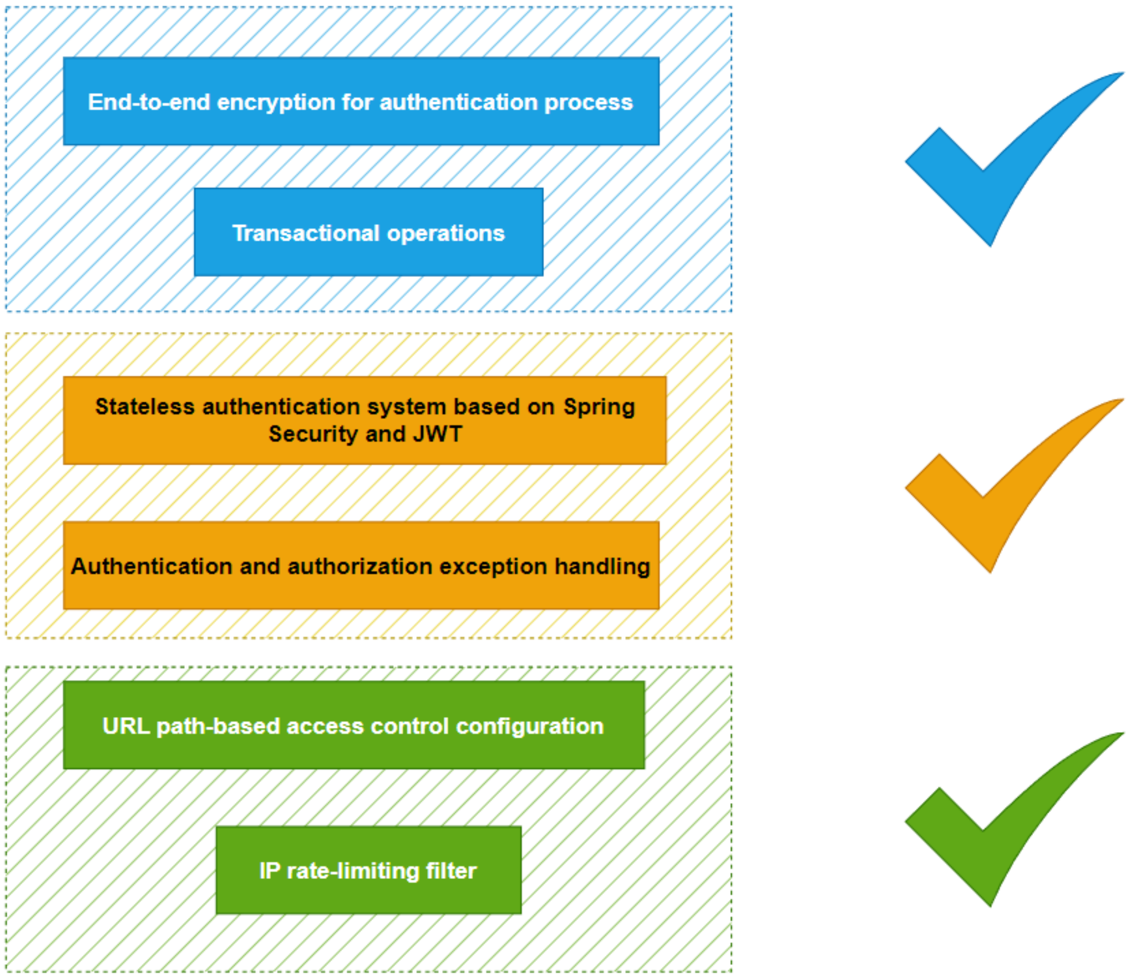

End-to-end encryption for authentication process
Transactional operations
Stateless authentication system based on Spring Security and JWT
Authentication and authorization exception handling
URL path-based access control configuration
IP rate-limiting filter

Figure 4.23 Implemented Security Mechanisms

In addition, the Redis caching mechanism plays a vital role in enhancing the security of the blog system. On one hand, caching user information reduces the exposure risk of the database, while encryption and access control strategies further enhance data security during transmission and storage. On the other hand, Redis is used to store non-critical data such as page view counts, which effectively reduces the load on the database and achieves logical data isolation. Redis also handles IP-based access rate limiting, thereby strengthening the system's resistance to malicious attacks at the architectural level. Its persistence mechanism ensures that data can be recovered after unexpected system crashes or restarts, improving the overall reliability and fault tolerance of the system.

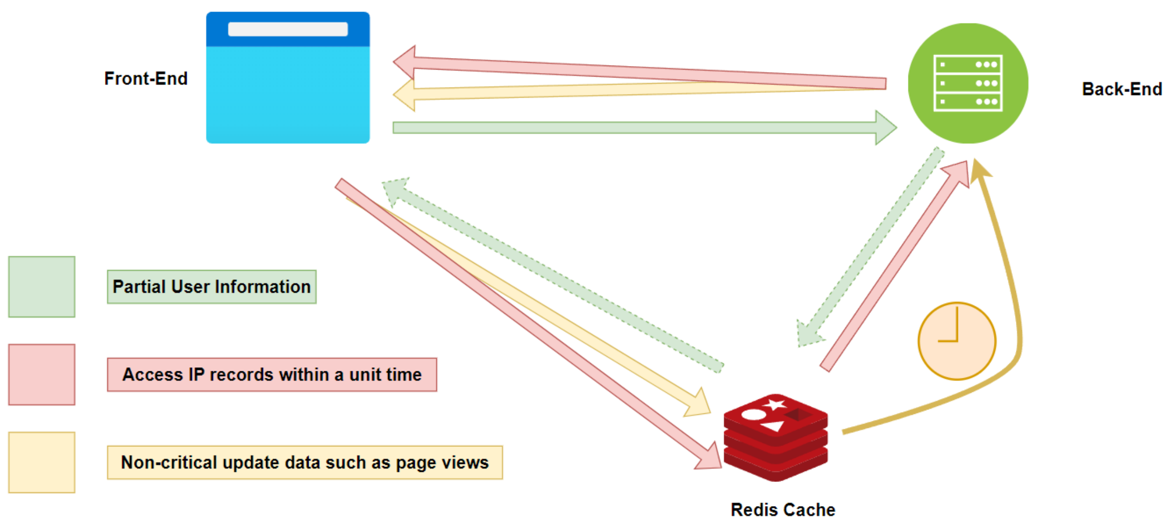

Front-End
Back-End
Redis Cache
Partial User Information
Access IP records within a unit time
Non-critical update data such as page views

Figure 4.24 The Use of Redis Caching Mechanism in the System

The corresponding code implementations and security testing diagrams are provided in the appendix for further reference.

5. Evaluation

5.1. Outcomes

The main goal of this project was to design and implement a secure blog system with efficient functionality, user-friendliness, and excellent security. Overall, the project achieved most of its goals, especially in terms of feature implementation and security, although there is still room for improvement in some details.

The key success of the project was that the system was completed on time and delivered with full functionality. In the initial stage, the individual development capacity was overestimated, and it was planned to develop a blogging system that supports multiple users, and a brief project analysis was conducted. However, there are many unforeseen challenges encountered in the actual development process, such as high concurrency, complex user interfaces, and user interactions. Fortunately, agile development strategies were used to make adjustments, and finally the project design and development were completed on time.

The blog system realizes the core functions such as article publishing, comment management, and user rights control, and effectively prevents common security threats through the security mechanism of Spring Security, ensuring the stability and security of the system.

However, the project also faced some challenges. The range of features expanded during development, especially in terms of user experience, and the original timeframe did not fully meet the development needs of all the expected features. The final delivery system met the basic needs and had good stability, but failed to implement the evaluation function of blog posts, which limited the interaction between users and the content.

5.2. Project management

Agile development methodologies were used in the development of this project. In the project planning stage, through demand analysis and function division, the core functions that the system needs to achieve are clarified, such as article publishing, comment management, user authority control, etc. The initial planning of the project is consistent with the final outcome in most respects, especially the implementation of the core functionality. However, during the development process, the scope of functions was expanded, and some details were not fully covered in the original plan, especially in terms of user experience optimization, resulting in some functions not being completed on time or being simplified.

In terms of time management, the project as a whole was completed and delivered within the scheduled timeframe. Although the overestimation of individual development capabilities in the early stage of development led to a lag in the implementation of the original functions, by adjusting the priority of tasks, simplifying some non-core functions, and adopting an iterative development model, the time pressure was effectively alleviated, and the core goals were finally delivered on time.

In terms of version control, the project uses Git tools for code version management throughout the project, ensuring the orderly change of Chinese parts and the ability to trace back versions in the development process. The project is hosted in a cloud Git repository, which ensures the security of the code and the convenience of collaborative development, and can quickly roll back to the stable version even if there is a problem during the development process, improving development efficiency and reliability.

5.3. Tools

A variety of development, collaboration, and deployment tools are used during the development process, which play an important role in improving development efficiency, code quality, and project maintainability. The core development technology is the front-end Vue3 framework and the back-end Spring Boot framework, the former is based on the concept of reactive programming, and the component-based design improves the reusability and maintainability of the front-end module, while the latter simplifies the configuration and structure of Java back-end development, and combines the MyBatis-Plus framework to significantly improve the efficiency of database access. In terms of database, MySQL is used to meet the data storage and management needs of small and medium-sized projects, and the cache optimization of some functions is realized with Redis to improve the system response performance.

In terms of development tools, using IntelliJ IDEA to write back-end code has greatly improved development efficiency with its powerful code hints, debugging, and plug-in support [25]. The front-end part is developed using VS Code, which is lightweight, fast, and plug-in-rich to meet the needs of Vue3 projects. In terms of database management, Navicat is used for visual database structure design and data operation, which improves the convenience of database management. Redis operation and debugging use the redis-cli tool, combined with the command line to improve the testing efficiency of the cache logic. The overall runtime environment of the project is based on Windows Subsystem for Linux 2 (WSL2), which realizes a Linux development experience closer to the production environment and enhances the portability and compatibility of the system [26].

In terms of collaboration and version management, Git is used for version control and GitHub repositories for code hosting. Although the project is developed by individuals, Git still plays an important role in branch management, feature iteration, and version rollback, ensuring the orderly development process and the stability of the code. In addition, Markdown is used to write interface descriptions, module designs, and functional records during the development process, which improves the quality of project documentation and maintainability.

5.4. Innovation (if any)

This project introduces a global AI chat component based on large language models in terms of technical application and user experience design. This component is presented in the form of a global floating chat box on the front end, and users can call up the chat window at any time while browsing the blog system, ask questions related to the current content to the connected large model API, and get real-time intelligent answers. This feature greatly improves the user's interaction depth and knowledge acquisition efficiency.

At the implementation level, the AI chatbox is encapsulated into a module that can be embedded in any page through the front-end component reuse mechanism to maintain the lightness and responsiveness of the interface. The backend encapsulates the general service class that calls the large model interface to realize message forwarding and context management to ensure the coherence and semantic relevance of the model response. In addition, in order to ensure security and performance, the system adopts interface current limiting and identity verification mechanisms to prevent abuse and ensure the stability of user access.

5.5. Context

This project demonstrates adaptability and social responsibility in terms of data security, accessibility, and legal compliance. The system uses Spring Security to achieve user permission control and encrypted storage to effectively ensure data security, and designs back-end logic with reference to relevant laws and regulations to ensure legal compliance. The interface is responsive and intuitive, improving user accessibility and user experience. The development process follows open source protocols, uses third-party resources reasonably, and reflects technical ethics. By introducing the large-model AI Q&A component, the system further enhances the interaction between users and content, helps to promote knowledge dissemination and public learning, and has a certain social influence.

6. Conclusion

With the theme of "Personal Blog System Based on Vue3 and Spring Boot", this project has successfully implemented a modern blog platform with complete functions, excellent performance and user-friendly interface. The system adopts a front-end and back-end separation architecture, builds a responsive front-end interface through Vue3 TypeScript, combines Spring Boot MyBatis-Plus to achieve efficient and stable back-end services, and works with MySQL and Redis to ensure data durability and access performance. Through the introduction of JWT authentication mechanism and front-end and back-end permission control system, the security of the system is effectively guaranteed.

The project strictly follows the software engineering process during the development process, and proceeds in an orderly manner from requirements analysis, system design, coding and implementation to testing and verification. In terms of function implementation, it covers key modules such as article management, classification and tag management, user authentication, content display and search, etc., which meets the core use scenarios of the personal blog system. In terms of technical practice, he has accumulated experience in the application of mainstream front-end and back-end frameworks, and improved the ability of system architecture design and performance optimization.

Although advanced features such as multi-user collaboration and social interaction cannot be implemented due to time and resource constraints, the current version of the system is stable and reliable, and has a good foundation for expansion. The project not only improves the development and problem-solving capabilities, but also lays a solid foundation for the subsequent in-depth study of front-end and back-end development and system security design, which has high learning and practical value.

References

[1] Zhong, Y. & Guo, Y. (2021). Design and implementation of a blog management system based on Springboot. *Modern Information Technology*, 5(7), pp.18–20, 24.
[2] Mo, W., et al. (2024). Research and application of a personal blog platform based on Spring Boot technology. *Science and Technology Wind*, (14), pp.94–96.
[3] Wang, H. & Zhang, L. (2024). Design and implementation of a web-based microblogging system. *Computer Knowledge and Technology*, 20(17), pp.65–68, 77.
[4] Dong, Z. & Pereira, C. (2024). Full-stack website independent development technology development. *Wireless Internet Technology*, 21(24), pp.51–56.
[5] Li, J. (2023). Analysis of computer software testing technology and development application strategies. *Information Recording Materials*, 24(3), pp.50–52.
[6] Wang, Q. & Chen, L. (2025). The application of value engineering in software engineering. *Hebei Enterprises*, (1), pp.69–73.
[7] Cui, P. (2019). Security issues of Java platform and application Java technology. *Information and Computer (Theory Edition)*, (15), pp.160–161.

[8] Spring Team. (2025). *Spring Boot documentation*. Retrieved from https://docs.spring.io/spring-boot/index.html
[9] Wei, C. & Zhang, R. (2024). Design and implementation of a student information management system based on Vue 3 and SpringBoot. *Computer Programming Techniques and Maintenance*, (10), pp.3–6, 20.
[10] Lu, Y. (2021). A brief discussion on the design and implementation of personal blog websites. *Inner Mongolia Science and Technology and Economy*, (17), pp.78–79, 81.
[11] Zeng, G. (2016). *Design and implementation of a personal blog system based on web front-end modularization* [Master's thesis, Huazhong University of Science and Technology].
[12] Ma, X., et al. (2022). Research and implementation of a front-end and back-end separation web platform technology. *Electronic Technology and Software Engineering*, (08), pp.70–73.
[13] Song, X. (2025). Application analysis of layered technology in computer software development. *Information and Computer*, 37(01), pp.141–143.
[14] Conventional Commits. (2025). *Conventional Commits 1.0.0*. Retrieved from https://www.conventionalcommits.org/en/v1.0.0/
[15] Aliyun Developer Community. (2024). *Ali official code specifications*. Retrieved from https://developer.aliyun.com/article/1587468
[16] Qin, D. (2024). A brief analysis of the application of the Vue framework in front-end development. *Information and Computer (Theoretical Edition)*, 36(13), pp.61–63.
[17] Mao, Z. (2019). *Construction of a product quality tracking system based on a front-end and back-end separation architecture* [Master's thesis, Fudan University].
[18] Wei, Q. & Ren, W. (2011). Application research based on Java storage mechanisms. *Modern Computer (Professional Edition)*, (20), pp.8–11.
[19] Lv, Y. (2024). Implementing authentication and authorization in microservice architecture using Spring Security + JWT. *Computer Knowledge and Technology*, 20(22), pp.60–63.
[20] Sun, Q., et al. (2024). Implementation of information system security functions based on Java security library Spring Security. *Industrial Control Computer*, 37(05), pp.104–105, 108.
[21] Gong, Y. (2022). Research on web interaction design principles and pathways based on user experience. *Footwear Technology and Design*, 2(10), pp.29–31.
[22] Kang, Q. & Xin, H. (2019). Research on differentiated web layout application based on user experience. *Rural Economy and Technology*, 30(24), pp.237–238.
[23] Zhihu. (2023). *A brief analysis of Java unit testing (JUnit + Mockito)*. Retrieved from https://zhuanlan.zhihu.com/p/608775174
[24] Dai, X., et al. (2024). Analysis of typical defects in software interface testing and optimization of test design. *Industrial Control Computer*, 37(09), pp.25–27.
[25] JetBrains. (n.d.). *IntelliJ IDEA | Features*. Retrieved from https://www.jetbrains.com/idea/features/
[26] Microsoft. (2023). *What is Windows Subsystem for Linux*. Retrieved from https://learn.microsoft.com/en-us/windows/wsl/about?ref=hackernoon.com

Appendix A - Code Manifest

Back-End:

File Name	File Path	Nature of Contribution	Purpose within the software
application.yml	WorkSpace\blog-backend\src\main\resources\application.yml	The project was automatically created and modified	Provide basic configuration for functions such as database, Redis, file upload, log output, MyBatis, and AI model invocation for SpringBoot projects.
dto/ ArticleDTO.java ArticleQueryDTO.java LoginUserDTO.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\common\domain\dto	Created and modified by developers	The directory of the data transmission object is used to define the data structure and verification rules when the front-end and back-end interact.
entity/ Access.java Article.java ArticleTag.java Category.java Tag.java User.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\common\domain\entity	Created and modified by developers	It corresponds one-to-one with the database tables and is used to represent the core business objects in the system. Each entity class contains the basic attributes and relationship mappings of the business object.
vo/ ArticleCountVo.java ArticleDetailsVo.java ArticlesVo.java CategoryCountVo.java HotArticleVo.java LoginUserVo.java PageVo.java PreviousNextArticleVo.java ResponseResult.java TagCountVo.java TagVo.java UserInfo.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\common\vo	Created and modified by developers	It is used to define view objects and is mainly used to encapsulate the data structure returned by the back end to the front end.

LoginUser.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\common\LoginUser.java	Created and modified by developers	Encapsulate the basic information and permissions of the currently logged-in user for identity authentication and authorization by Spring Security.
PageConfig.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\config\PageConfig.java	Created and modified by developers	A pagination plugin for setting up MyBatis-Plus.
RedisConfig.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\config\RedisConfig.java	Created and modified by developers	Configure the Redis serialization method and create the RedisTemplate Bean.
SecurityConfig.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\config\SecurityConfig.java	Created and modified by developers	Configure Spring Security to handle authentication, authorization and exception handling.
WebConfig.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\config\WebConfig.java	Created and modified by developers	Configure Spring Security to handle authentication, authorization and exception handling.

ArticleController.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\controller\ArticleController.java	Created and modified by developers	Handle requests related to articles, such as adding, editing, querying, etc.
CategoryController.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\controller\CategoryController.java	Created and modified by developers	Handle requests related to classification, such as obtaining classification statistics.
TagController.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\controller\TagController.java	Created and modified by developers	Handle requests related to labels, such as obtaining label statistics.
UserController.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\controller\UserController.java	Created and modified by developers	Handle user registration, login, logout and information query.
BlogSystemConstantsEnum.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\enums\BlogSyst	Created and modified by developers	Define the constants of the blog system, such as article status, user type, etc.

	emConstantsEnum.java		
HttpStatusCodeEnum.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\enums\HttpStatusCodeEnum.java	Created and modified by developers	Define the constants of the blog system, such as article status, user type, etc.
IpRateLimitFilter.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\filter\IpRateLimitFilter.java	Created and modified by developers	Limit the frequency of IP requests to prevent flooding.
JwtAuthenticationTokenFilter.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\filter\JwtAuthenticationTokenFilter.java	Created and modified by developers	Parse the JWT and set the user authentication information.
AccessDeniedHandlerImpl.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\handle\security\AccessDeniedHandlerImpl.java	Created and modified by developers	Handle abnormal responses for unauthorized access.
AuthenticationEntryPointImpl.java	WorkSpace\blog-backend\src\main\java\co	Created and modified by developers	Handle the abnormal response of authentication failure.

	m\golemon\blogbackend\handle\security\AuthenticationEntryPointImpl.java		
mapper/ AccessMapper.java Article.Mapper.java ArticleTagMapper.java CategoryMapper.java TagMapper.java UserMapper.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\mapper	Created and modified by developers	The data access layer of MyBatis-Plus contains the database operations corresponding to each entity class.
MyMetaObjectHandler.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\mp\MyMetaObjectHandler.java	Created and modified by developers	Automatically fill the creation and update time fields.
service/ ArticleService.java ArticleTagService.java CategoryService.java TagService.java UserService.java Impl/ArticleServiceImpl.java Impl/ArticleTagServiceImpl.java	E:\Undergraduate\School\Courses\毕设\WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\service	Created and modified by developers	Service interfaces including various business modules

Impl/CategoryServiceImpl.java Impl/TagServiceImpl.java Impl/UserServiceImpl.java			
ViewCountSyncTask.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\task\ViewCountSyncTask.java	Created and modified by developers	Synchronize the page views to the database at that time.
Assert.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\Assert.java	Created and modified by developers	Make unified assertions and throw business exceptions.
BeanCopyUtils.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\BeanCopyUtils.java	Created and modified by developers	Simplify the object property copy operation.
FastJsonRedisSerializer.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\FastJsonRedisSerializer.java	Created and modified by developers	FastJson serialization and deserialization tool classes.
JwtUtil.java	WorkSpace\blog-backend\src\	Created and modified by developers	JWT generation and parsing tool class.

	main\java\com\golemon\blogbackend\utils\JwtUtil.java		
LocalDateTimeUtil.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\LocalDateTimeUtil.java	Created and modified by developers	Obtain the start and end dates of the specified month.
RedisCache.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\RedisCache.java	Created and modified by developers	A utility class that provides Redis cache operations.
SecurityUtils.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\SecurityUtils.java	Created and modified by developers	Obtain the relevant information of the currently logged-in user.
WebUtils.java	WorkSpace\blog-backend\src\main\java\com\golemon\blogbackend\utils\WebUtils.java	Created and modified by developers	Render the string into the client response.
BlogSystemException.java	WorkSpace\blog-backend\src\main\java\com\golemon\ex	Created and modified by developers	Handle business exceptions in the blog system.

	ception\BlogSystemException.java		
AIChatController.java	WorkSpace\blog-backend\src\main\java\com\golemon\plugins\controller\AIChatController.java	Created and modified by developers	Process AI chat requests and return response results.
AIChatServiceImpl.java	WorkSpace\blog-backend\src\main\java\com\golemon\plugins\service\impl\AIChatServiceImpl.java	Created and modified by developers	Process AI chat requests and return response results.
AIChatService.java	WorkSpace\blog-backend\src\main\java\com\golemon\plugins\service\AIChatService.java	Created and modified by developers	Define the interface of the AI chat service.
BlogBackendApplication.java	WorkSpace\blog-backend\src\main\java\com\golemon\BlogBackendApplication.java	Created and modified by developers	Start the blog backend application and enable the scheduled tasks.
AccessMapper.xml	WorkSpace\blog-backend\src\main\resources\mapper\AccessMapper.xml	Created and modified by developers	Query the permission list based on the user ID.

ArticleServiceImplTest.java	WorkSpace\blog-backend\src\test\java\com\golemon\blogbackend\service\impl\ArticleServiceImplTest.java	Created and modified by developers	Test the various functions in the article service class.
BlogBackendApplicationTests.java	WorkSpace\blog-backend\src\test\java\com\golemon\blogbackend\BlogBackendApplicationTests.java	Created and modified by developers	Test the loading of the Spring Boot application context.
pom.xml	WorkSpace\blog-backend\pom.xml	The project was automatically created and modified	Maven project configuration file.

Front-End:

File Name	File Path	Nature of Contribution	Purpose within the software
vite.config.ts	WorkSpace\blog-frontend\vite.config.ts	The project was automatically created and modified	Vite configuration file, set aliases and style preprocessing.
settings.ts	WorkSpace\blog-frontend\settings.ts	Created and modified by developers	Project configuration and encryption salt Settings.
README.md	WorkSpace\blog-frontend\README.md	The project was automatically created and modified	Vue 3 + TypeScript + Vite project template description.
index.html	WorkSpace\blog-frontend\index.html	The project was automatically created and modified	The HTML template file of the project.

gitdev.bat	WorkSpace\blog-frontend\gitdev.bat	Created and modified by developers	Automate Git commit and push, and start the development environment.
.gitignore	WorkSpace\blog-frontend\.gitignore	The project was automatically created and modified	Configure Git to ignore files and development environment-related files.
.eslintrc.cjs	WorkSpace\blog-frontend\.eslintrc.cjs	The project was automatically created and modified	This file is used to configure ESLint rules.
.eslintignore	WorkSpace\blog-frontend\.eslintignore	The project was automatically created and modified	This file is used to specify the files that ESLint ignores.
.eslint.cjs	WorkSpace\blog-frontend\.eslint.cjs	The project was automatically created and modified	This file is used to configure the operation rules of ESLint.
vite-env.d.ts	WorkSpace\blog-frontend\src\vite-env.d.ts	Created and modified by developers	This file is used to provide type declarations for the Vite project.
main.ts	WorkSpace\blog-frontend\src\main.ts	The project was automatically created and modified	This file is the entry file of the project and initializes the application.
App.vue	WorkSpace\blog-frontend\src\App.vue	Created and modified by developers	This file is the root component of the Vue application.
tagList.vue	WorkSpace\blog-frontend\src\views\tag\tagList.vue	Created and modified by developers	This file presents the tag cloud and tag statistics.
tagDetails.vue	WorkSpace\blog-frontend\src\	Created and modified by developers	This file shows the list of articles with specific tags and the pagination function.

	views\tag\tagDetails.vue		
index.vue	WorkSpace\blog-frontend\src\views\register\index.vue	Created and modified by developers	This file implements the user registration function, including form validation and submission.
index.vue	WorkSpace\blog-frontend\src\views\manage\index.vue	Created and modified by developers	The article management interface supports article filtering, editing and deletion.
index.vue	WorkSpace\blog-frontend\src\views\login\index.vue	Created and modified by developers	Realize the user login function, verify the identity and navigate to the homepage.
index.vue	WorkSpace\blog-frontend\src\views\home\index.vue	Created and modified by developers	Display the article list, supporting pagination and sidebar content display.
categoryList.vue	WorkSpace\blog-frontend\src\views\category\categoryList.vue	Created and modified by developers	Display the category list. Click to jump and view the specific category content.
categoryDetails.vue	WorkSpace\blog-frontend\src\views\category\categoryDetails.vue	Created and modified by developers	Display the list of articles under the category and support pagination browsing.
SearchView.vue	WorkSpace\blog-frontend\src\views\ArticleSearch\SearchView.vue	Created and modified by developers	Display search results and support article click to view details.

index.scss	WorkSpace\blog-frontend\src\views\ArticlePage\index.scss	Created and modified by developers	Display the article content and related navigation, and support the style of code copying function.
index.vue	WorkSpace\blog-frontend\src\views\ArticlePage\index.vue	Created and modified by developers	Display article details and support editing, deletion, and page view updates.
index.vue	WorkSpace\blog-frontend\src\views\ArticleEditPage\index.vue	Created and modified by developers	The article editing function supports publishing, saving drafts and editing.
index.vue	WorkSpace\blog-frontend\src\views\ArticleChangePage\index.vue	Created and modified by developers	Provide article editing and update functions.
index.vue	WorkSpace\blog-frontend\src\views\archive\index.vue	Created and modified by developers	Display articles archived and classified by year.
index.vue	WorkSpace\blog-frontend\src\views\about\index.vue	Created and modified by developers	Display social links and resume files.
awesomeIcon.ts	WorkSpace\blog-frontend\src\utils\awesomeIcon.ts	Created and modified by developers	Import and register the FontAwesome icon library for global use.
code-block.ts	WorkSpace\blog-	Created and modified by developers	Add highlighting and copying functions to the code block.

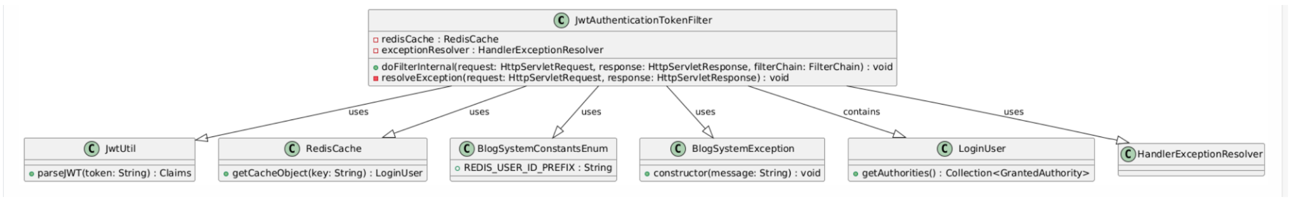
	frontend\src\utils\code-block.ts		
encrypt.ts	WorkSpace\blog-frontend\src\utils\encrypt.ts	Created and modified by developers	The MD5 encryption function is implemented using CryptoJS.
markdown-it-setup.ts	WorkSpace\blog-frontend\src\utils\markdown-it-setup.ts	Created and modified by developers	Configure and initialize the Markdown parser to support plugin extensions.
request.ts	WorkSpace\blog-frontend\src\utils\request.ts	Created and modified by developers	Encapsulate Axios, handle request and response interception, and enhance error alerts.
storage.ts	WorkSpace\blog-frontend\src\utils\storage.ts	Created and modified by developers	Manage the storage operations of user tokens and user information.
types/ markdown-js-d-ts archive.ts article.ts axios.d.ts crypto-js.d.ts index.ts prismjs.d.ts shims-v-md-editor.d.ts user.ts	WorkSpace\blog-frontend\src\types	Created and modified by developers	Store the type definition files in the project.
styles/ index.scss variable.scss	WorkSpace\blog-frontend\src\styles	Created and modified by developers	Store the style files in the project.

stores/ admin.ts category.ts categoryAbout.ts index.ts tag.ts tagAbout.ts web.ts	WorkSpace\blog-frontend\src\stores	Created and modified by developers	Store Vuex state management-related files.
router/ index.ts routes.ts	WorkSpace\blog-frontend\src\router	Created and modified by developers	Store the routing configuration and management files.
components/ AdminCard.vue AIChat.vue ArticleCard.vue CategoryCard.vue CopyCodeButton.vue FormCard.vue PostArticleCard.vue RandomArticleCard.vue SelfButton.vue SelfCard.vue SelfFooter.vue SelfHeader.vue SelfToTop.vue TagCard.vue TagCloud.vue	WorkSpace\blog-frontend\src\components	Created and modified by developers	Store reusable UI components.
pdfs/ CV.pdf	WorkSpace\blog-frontend\src\assets\pdfs	Created and modified by developers	Store PDF files.
images/ avatar.jpg register.png	WorkSpace\blog-frontend\src\assets\images	Created and modified by developers	Store image files.

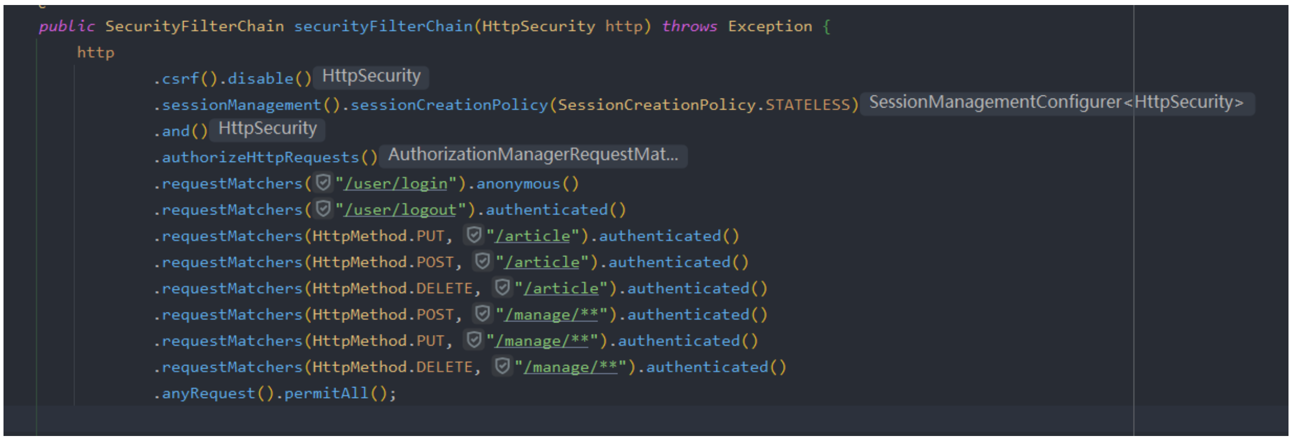
css/ code-block.scss index.scss markdown.scss	WorkSpace\blog-frontend\src\assets\css	Created and modified by developers	Store CSS files.
api/ archive.ts article.ts category.ts comment.ts image.ts tag.ts user.ts	WorkSpace\blog-frontend\src\api	Created and modified by developers	Encapsulate the interface for interaction with the back end.
logo.png	WorkSpace\blog-frontend\public\logo.png	Created and modified by developers	The Logo image of the blog project.
openapi/ aichat.yaml article.yaml category.yaml openapi.yaml tag.yaml user.yaml	WorkSpace\blog-frontend\docs\openapi	Created and modified by developers	Back-end API interface documentation.

Additional appendices

a) JWT authentication mechanism


JwtAuthenticationTokenFilter
redisCache : RedisCache
exceptionResolver : HandlerExceptionResolver
doFilterInternal(request: HttpServletRequest, response: HttpServletResponse, filterChain: FilterChain) : void
resolveException(request: HttpServletRequest, response: HttpServletResponse) : void
uses
uses
uses
uses
contains
uses
JwtUtil
parseJWT(token: String) : Claims
RedisCache
getCacheObject(key: String) : LoginUser
BlogSystemConstantsEnum
REDIS_USER_ID_PREFIX : String
BlogSystemException
constructor(message: String) : void
LoginUser
getAuthorities() : Collection<GrantedAuthority>
HandlerExceptionResolver

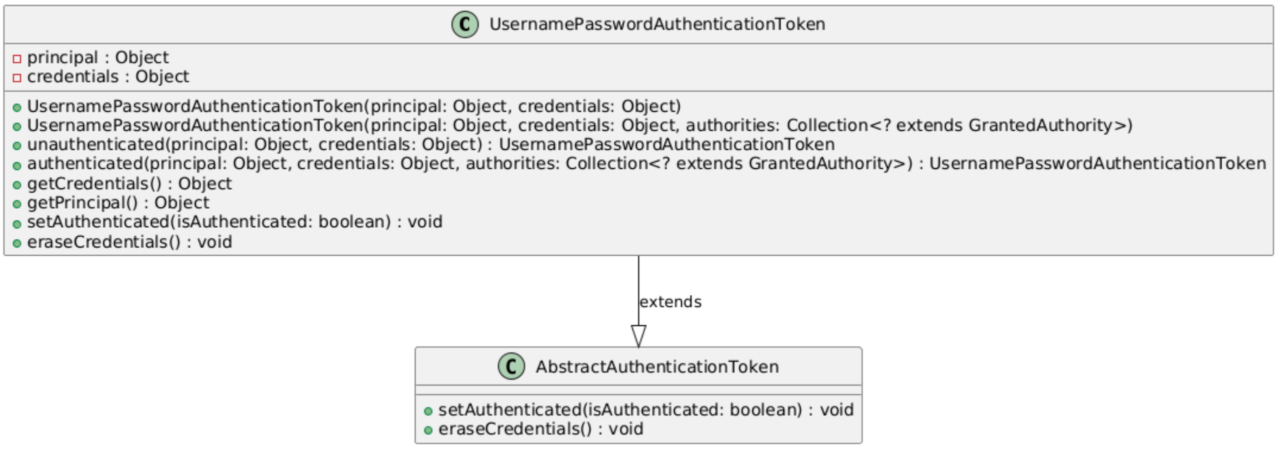
b) Access control configuration



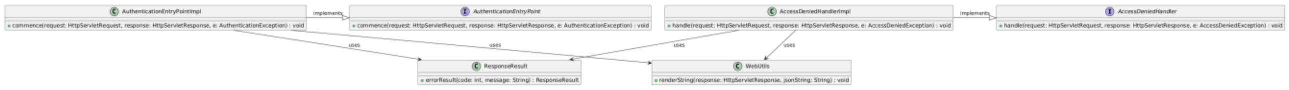
```java
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
            .csrf().disable()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeHttpRequests()
            .requestMatchers("/user/login").anonymous()
            .requestMatchers("/user/logout").authenticated()
            .requestMatchers(HttpMethod.PUT, "/article").authenticated()
            .requestMatchers(HttpMethod.POST, "/article").authenticated()
            .requestMatchers(HttpMethod.DELETE, "/article").authenticated()
            .requestMatchers(HttpMethod.POST, "/manage/**").authenticated()
            .requestMatchers(HttpMethod.PUT, "/manage/**").authenticated()
            .requestMatchers(HttpMethod.DELETE, "/manage/**").authenticated()
            .anyRequest().permitAll();
```

HttpSecurity
SessionManagementConfigurer<HttpSecurity>
HttpSecurity
AuthorizationManagerRequestMat...

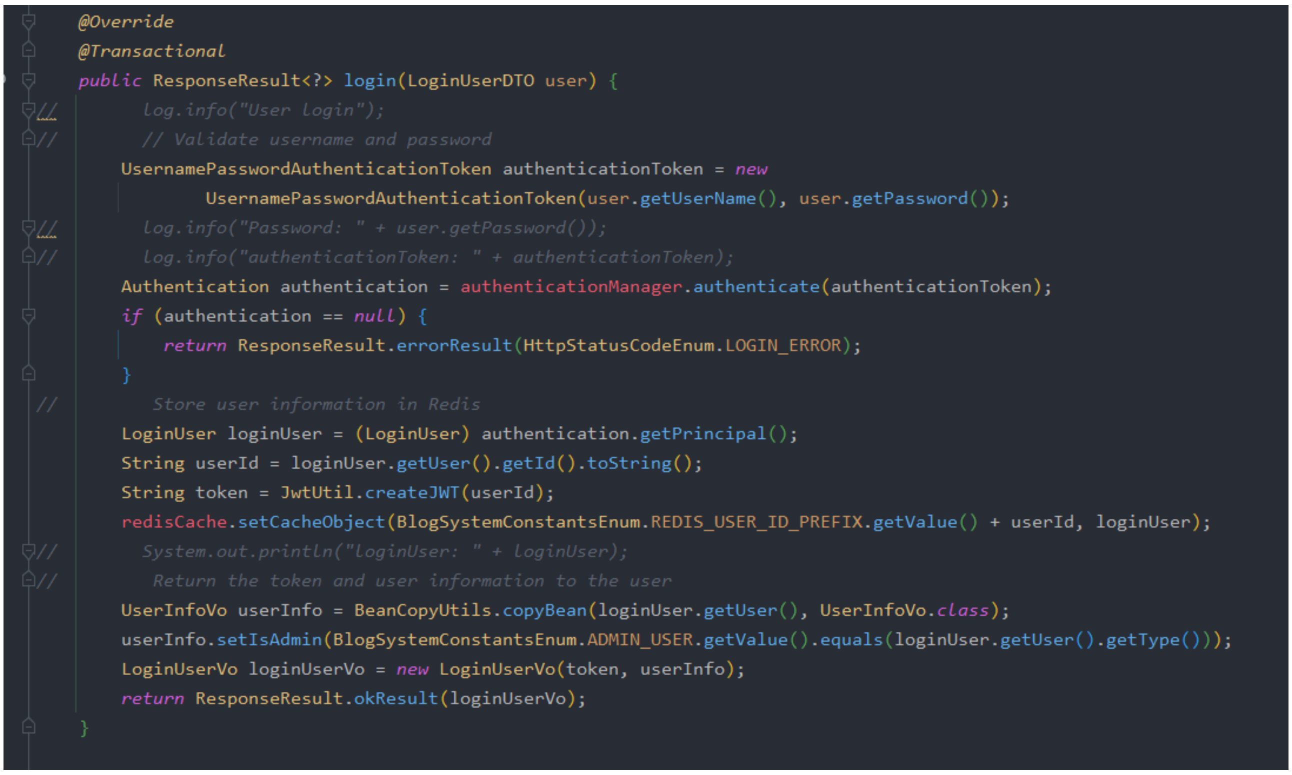
c) Password encryption using BCryptPasswordEncoder on the backend


UsernamePasswordAuthenticationToken
principal : Object
credentials : Object
UsernamePasswordAuthenticationToken(principal: Object, credentials: Object)
UsernamePasswordAuthenticationToken(principal: Object, credentials: Object, authorities: Collection<? extends GrantedAuthority>)
unauthenticated(principal: Object, credentials: Object) : UsernamePasswordAuthenticationToken
authenticated(principal: Object, credentials: Object, authorities: Collection<? extends GrantedAuthority>) : UsernamePasswordAuthenticationToken
getCredentials() : Object
getPrincipal() : Object
setAuthenticated(isAuthenticated: boolean) : void
eraseCredentials() : void
extends
AbstractAuthenticationToken
setAuthenticated(isAuthenticated: boolean) : void
eraseCredentials() : void

d) Unified handling of authentication and authorization exceptions

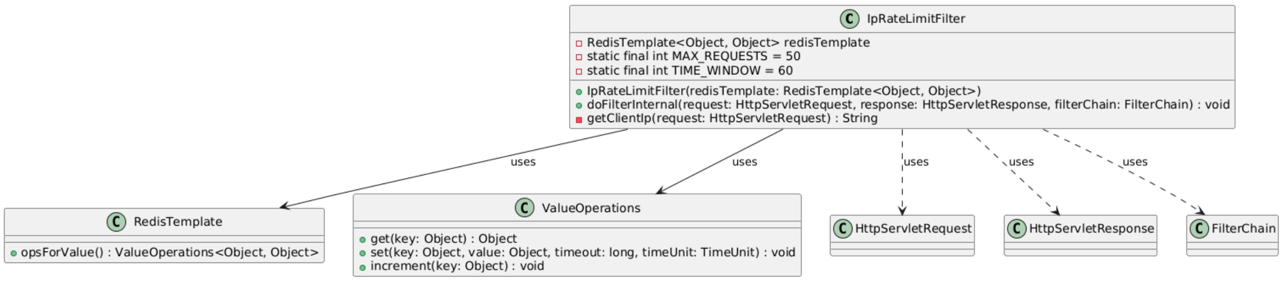

AuthenticationEntryPointImpl
commence(request: HttpServletRequest, response: HttpServletResponse, e: AuthenticationException) : void
implements
AuthenticationEntryPoint
commence(request: HttpServletRequest, response: HttpServletResponse, e: AuthenticationException) : void
AccessDeniedHandlerImpl
handle(request: HttpServletRequest, response: HttpServletResponse, e: AccessDeniedException) : void
implements
AccessDeniedHandler
handle(request: HttpServletRequest, response: HttpServletResponse, e: AccessDeniedException) : void
uses
uses
uses
uses
ResponseResult
errorResult(code: int, message: String) : ResponseResult
WebUtils
renderString(response: HttpServletResponse, jsonString: String) : void

e) Transaction management



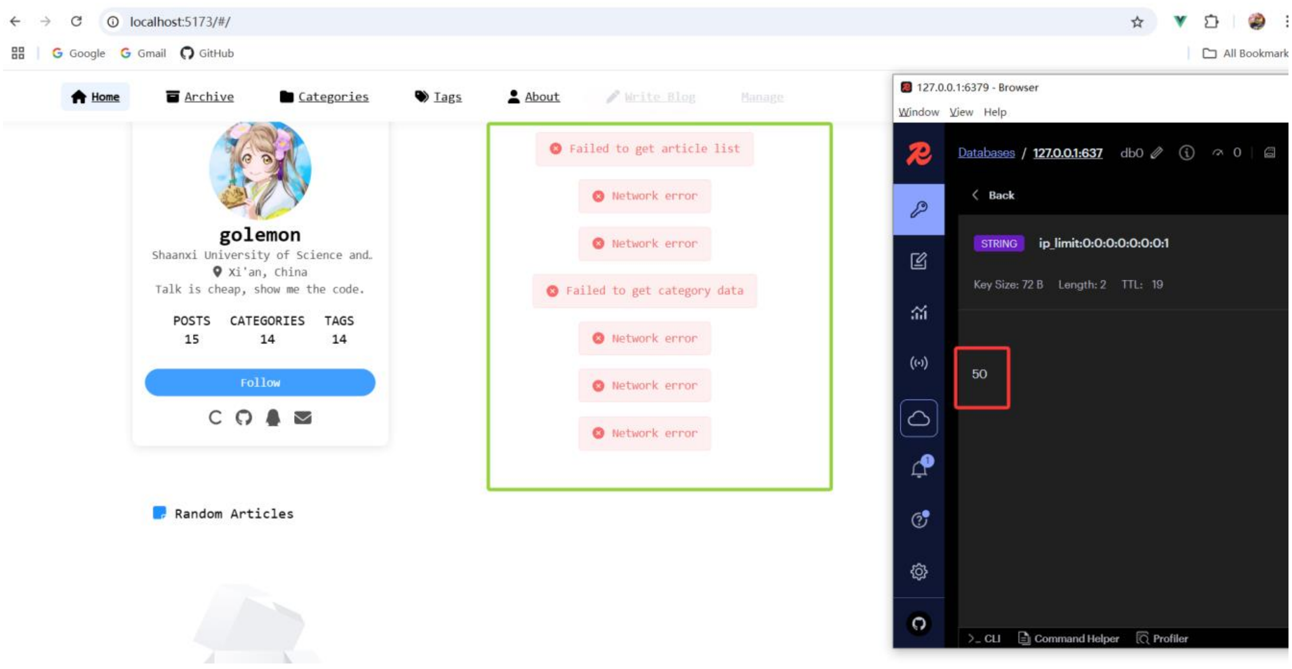
```java
    @Override
    @Transactional
    public ResponseResult<?> login(LoginUserDTO user) {
//        log.info("User login");
//        // Validate username and password
        UsernamePasswordAuthenticationToken authenticationToken = new
                UsernamePasswordAuthenticationToken(user.getUserName(), user.getPassword());
//        log.info("Password: " + user.getPassword());
//        log.info("authenticationToken: " + authenticationToken);
        Authentication authentication = authenticationManager.authenticate(authenticationToken);
        if (authentication == null) {
            return ResponseResult.errorResult(HttpStatusCodeEnum.LOGIN_ERROR);
        }
//         Store user information in Redis
        LoginUser loginUser = (LoginUser) authentication.getPrincipal();
        String userId = loginUser.getUser().getId().toString();
        String token = JwtUtil.createJWT(userId);
        redisCache.setCacheObject(BlogSystemConstantsEnum.REDIS_USER_ID_PREFIX.getValue() + userId, loginUser);
//        System.out.println("loginUser: " + loginUser);
//         Return the token and user information to the user
        UserInfoVo userInfo = BeanCopyUtils.copyBean(loginUser.getUser(), UserInfoVo.class);
        userInfo.setIsAdmin(BlogSystemConstantsEnum.ADMIN_USER.getValue().equals(loginUser.getUser().getType()));
        LoginUserVo loginUserVo = new LoginUserVo(token, userInfo);
        return ResponseResult.okResult(loginUserVo);
    }
```

f) Global IP-based rate limiting mechanism



IpRateLimitFilter
RedisTemplate<Object, Object> redisTemplate
static final int MAX_REQUESTS = 50
static final int TIME_WINDOW = 60
IpRateLimitFilter(redisTemplate: RedisTemplate<Object, Object>)
doFilterInternal(request: HttpServletRequest, response: HttpServletResponse, filterChain: FilterChain) : void
getClientIp(request: HttpServletRequest) : String
uses
uses
uses
uses
uses
RedisTemplate
opsForValue() : ValueOperations<Object, Object>
ValueOperations
get(key: Object) : Object
set(key: Object, value: Object, timeout: long, timeUnit: TimeUnit) : void
increment(key: Object) : void
HttpServletRequest
HttpServletResponse
FilterChain

After more than 50 access attempts, the server refuses to respond.



localhost:5173/#/

Home Archive Categories Tags About Write Blog Manage

golemon
Shaanxi University of Science and.
Xi'an, China
Talk is cheap, show me the code.

POSTS 15 CATEGORIES 14 TAGS 14

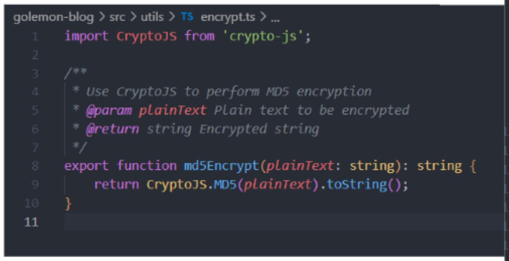
Follow

Random Articles

Failed to get article list
Network error
Network error
Failed to get category data
Network error
Network error
Network error

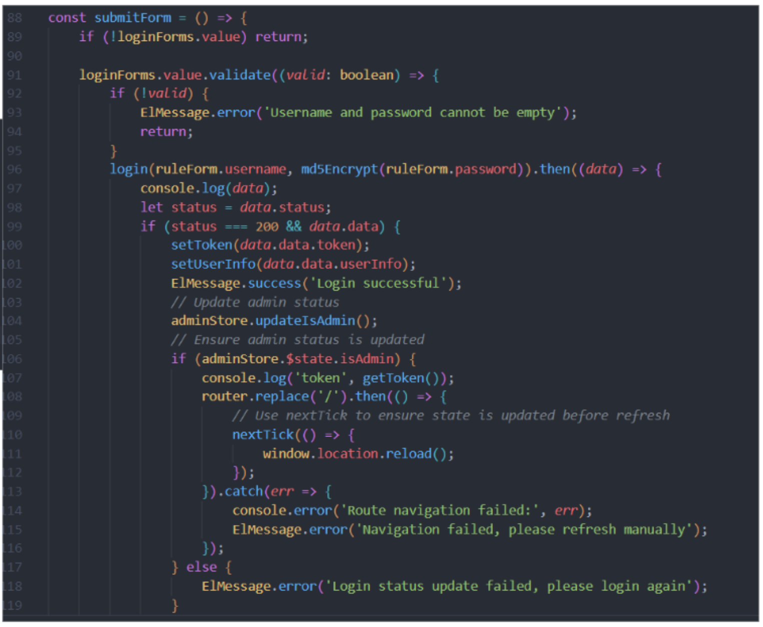
127.0.0.1:6379 - Browser
Window View Help
Databases / 127.0.0.1:637 db0
Back
STRING ip_limit:0:0:0:0:0:0:0:1
Key Size: 72 B Length: 2 TTL: 19
50
>_ CLI Command Helper Profiler

g) Password encryption using MD5 on the frontend



golemon-blog > src > utils > TS encrypt.ts > ...

```ts
import CryptoJS from 'crypto-js';

/**
 * Use CryptoJS to perform MD5 encryption
 * @param plainText Plain text to be encrypted
 * @return string Encrypted string
 */
export function md5Encrypt(plainText: string): string {
    return CryptoJS.MD5(plainText).toString();
}

```



```ts
const submitForm = () => {
    if (!loginForms.value) return;

    loginForms.value.validate((valid: boolean) => {
        if (!valid) {
            ElMessage.error('Username and password cannot be empty');
            return;
        }
        login(ruleForm.username, md5Encrypt(ruleForm.password)).then((data) => {
            console.log(data);
            let status = data.status;
            if (status === 200 && data.data) {
                setToken(data.data.token);
                setUserInfo(data.data.userInfo);
                ElMessage.success('Login successful');
                // Update admin status
                adminStore.updateIsAdmin();
                // Ensure admin status is updated
                if (adminStore.$state.isAdmin) {
                    console.log('token', getToken());
                    router.replace('/').then(() => {
                        // Use nextTick to ensure state is updated before refresh
                        nextTick(() => {
                            window.location.reload();
                        });
                    }).catch(err => {
                        console.error('Route navigation failed:', err);
                        ElMessage.error('Navigation failed, please refresh manually');
                    });
                } else {
                    ElMessage.error('Login status update failed, please login again');
                }
```