# Group Project Report（Group Y）

# Post Forum System Project Report

| Name | Student ID |
|---|---|
| Shenghan Gao | 24428078 |
| Lin Yuan | 24447897 |
| Jianxi Guan | 24429163 |

# I. Project Introduction

This project is a front-end and back-end separation of the post forum system, for users to create an efficient, convenient and secure information exchange platform. The front-end is built based on Vue framework, with its rich component development model and responsive features, and waterfall flow as the dominant, to achieve a smooth and beautiful user interface interaction; the back-end is based on Node.js Express and MongoDB is responsible for the storage of data, processing, and business logic control, to ensure that the system operates stably and reliably. The system covers a series of functions such as user registration, login, post publishing, browsing and management, and fully meets the diversified needs of users by fully implementing CRUD operations.

# II. Responsibilities of team members

[Jianxi Guan]
Responsibilities: Responsible for the overall layout and design of the front-end page, using Vue technology to achieve the development of post display, user interaction interface and other functional modules, including post list rendering, user interaction buttons. At the same time, participate in the front-end page style adjustment and optimisation to ensure the compatibility and aesthetics of the page on different devices.
Achievements: Completed the development of the post display area and user interface on the home page, and realised the dynamic display of information such as post titles, authors and images, as well as the initial implementation of interactive functions such as user likes and comments.
[Shenghan Gao]
Scope of Responsibilities:Undertake back-end data model design and database management work, defining the structure of data collections such as posts and users, and establishing the correlation relationship between them. Implement the back-end interface for CRUD operations to ensure the correct storage, reading, updating and deletion of data. At the same time, responsible for the development and integration of Token - based authentication mechanism to ensure the security of the system.
Achievements: Successfully designed and built the data table related to posts and users, implemented the CRUD operation interface for posts and user information, and completed the Token - based authentication function to effectively prevent unauthorised access.Develop the

function of invoking Google AI and extracting post keywords.
[Yuan Lin]
Responsibilities: Responsible for the overall coordination and testing of the project, supervising the work progress of team members to ensure the project progresses according to plan. Conduct system functional testing, performance testing and security testing, find and feedback problems in time, and assist team members to fix them. Actively participate in the code review work, according to the code quality standards, code specification, readability, maintainability and other aspects of the code to carry out a rigorous review, put forward optimisation proposals, to ensure that the code quality in line with the project requirements. At the same time, responsible for writing the project result report.
Work Achievements: Organise several project progress meetings to solve problems arising from teamwork in a timely manner. Completed several rounds of system testing, found and assisted in fixing several functional and performance problems, and improved the stability and reliability of the system.

## III. Demonstration of the system

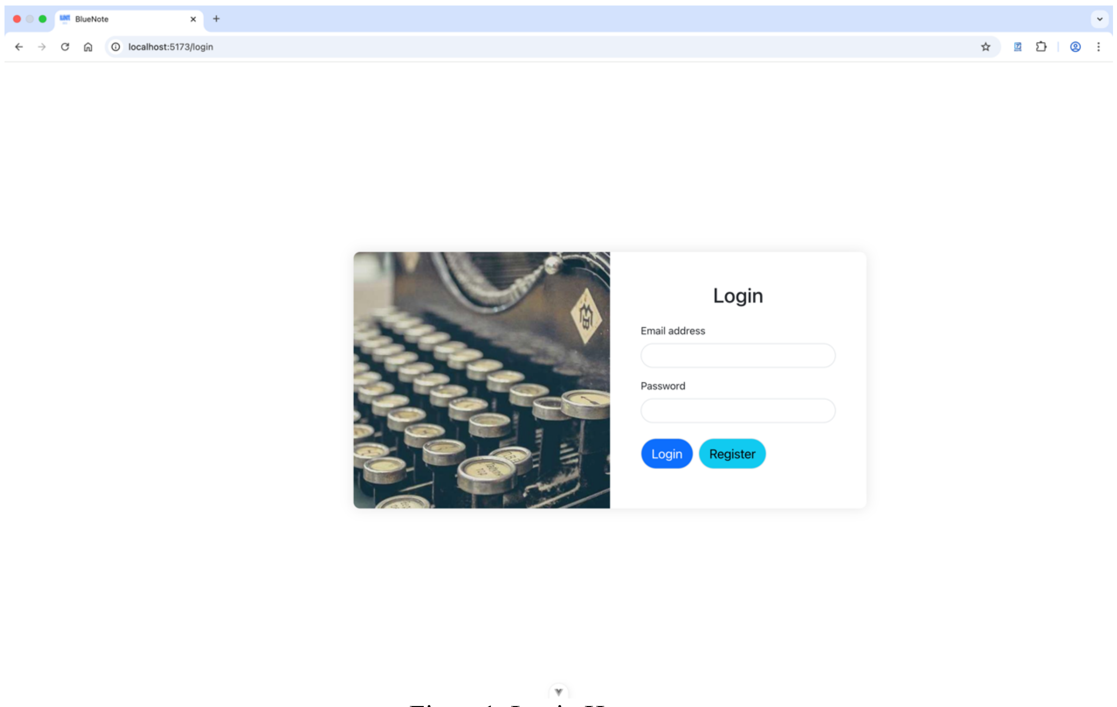
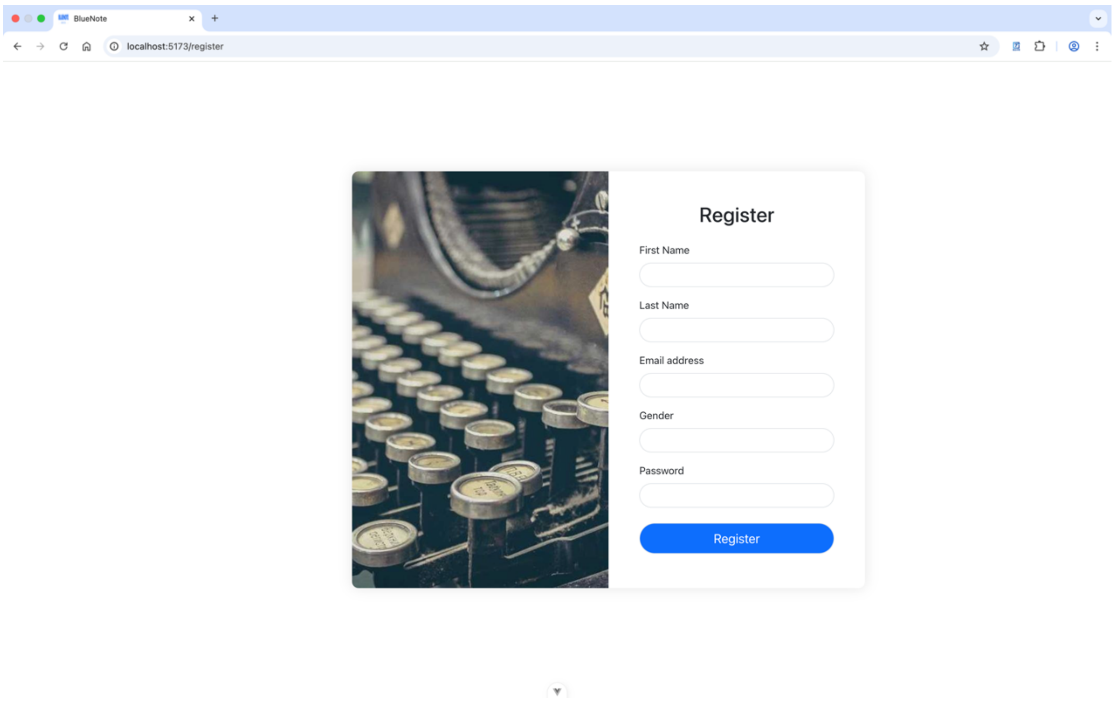


Figure1. Login Homepage


BlueNote

localhost:5173/register

Register

First Name

Last Name

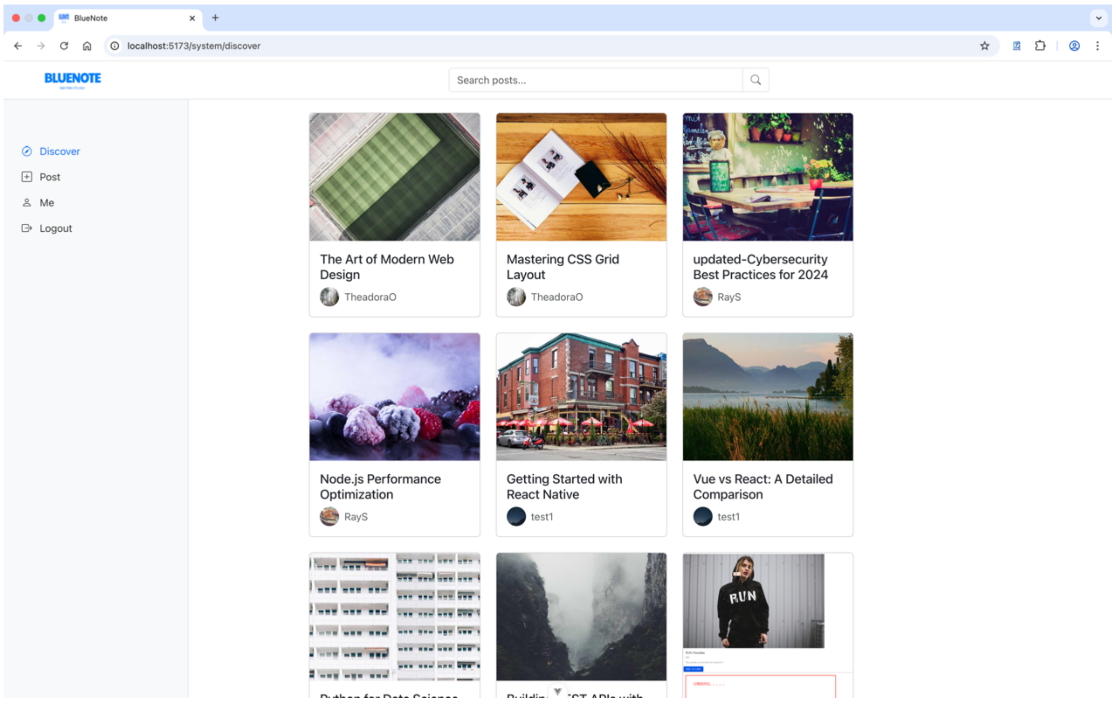
Email address

Gender

Password

Register

Figure2. Register Page


BlueNote

localhost:5173/system/discover

BLUENOTE

Search posts...

Discover

Post

Me

Logout

The Art of Modern Web Design
TheadoraO

Mastering CSS Grid Layout
TheadoraO

updated-Cybersecurity Best Practices for 2024
RayS

Node.js Performance Optimization
RayS

Getting Started with React Native
test1

Vue vs React: A Detailed Comparison
test1

Python for Data Science

Buildin ST APIs with

Figure3. System Homepage

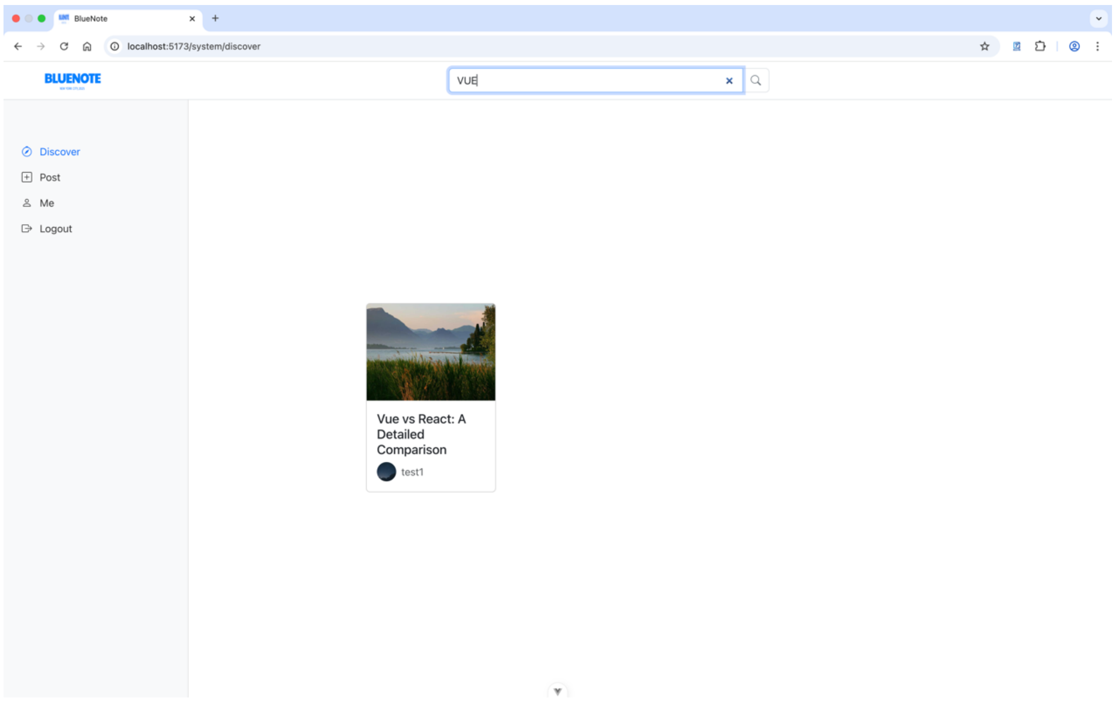

BlueNote
localhost:5173/system/discover
BLUENOTE
VUE
Discover
Post
Me
Logout
Vue vs React: A Detailed Comparison
test1

Figure4. Search Page

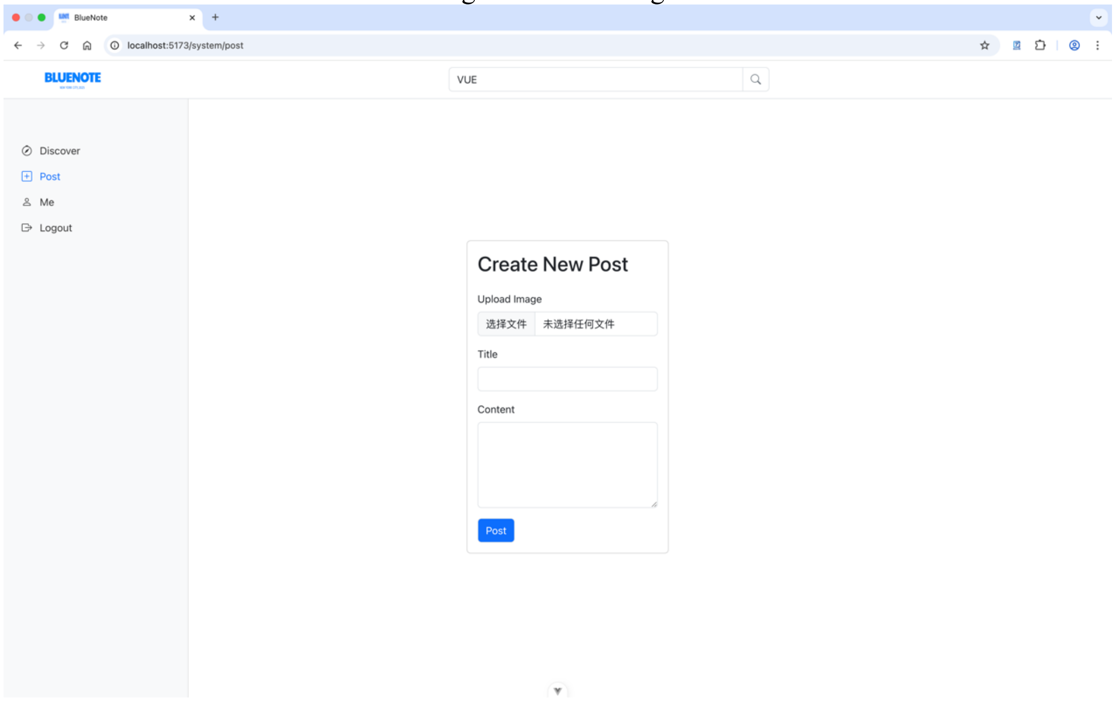


Figure5. Post Page

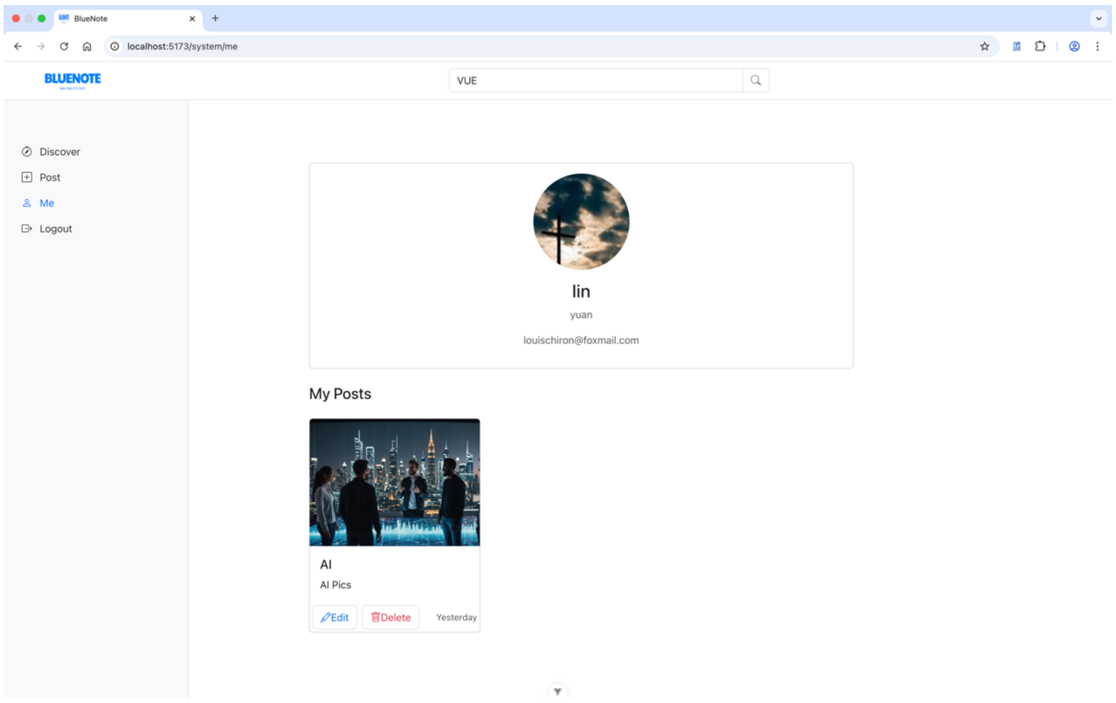


Figure6. Profile Page

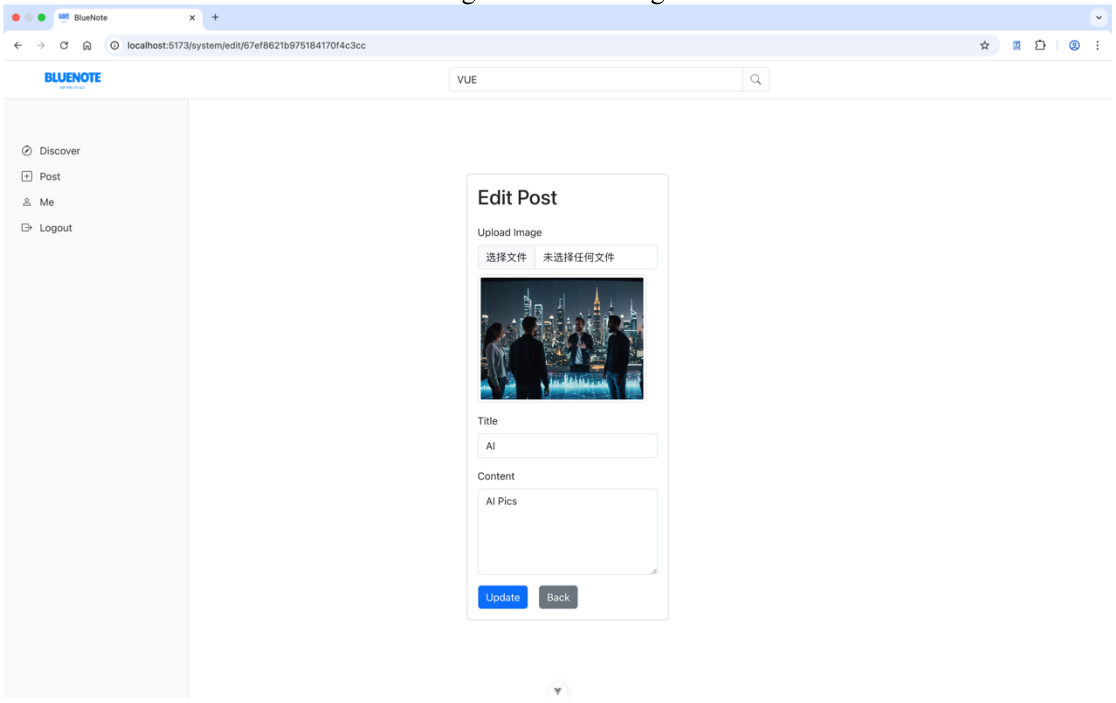


Figure7. Edit Post Page

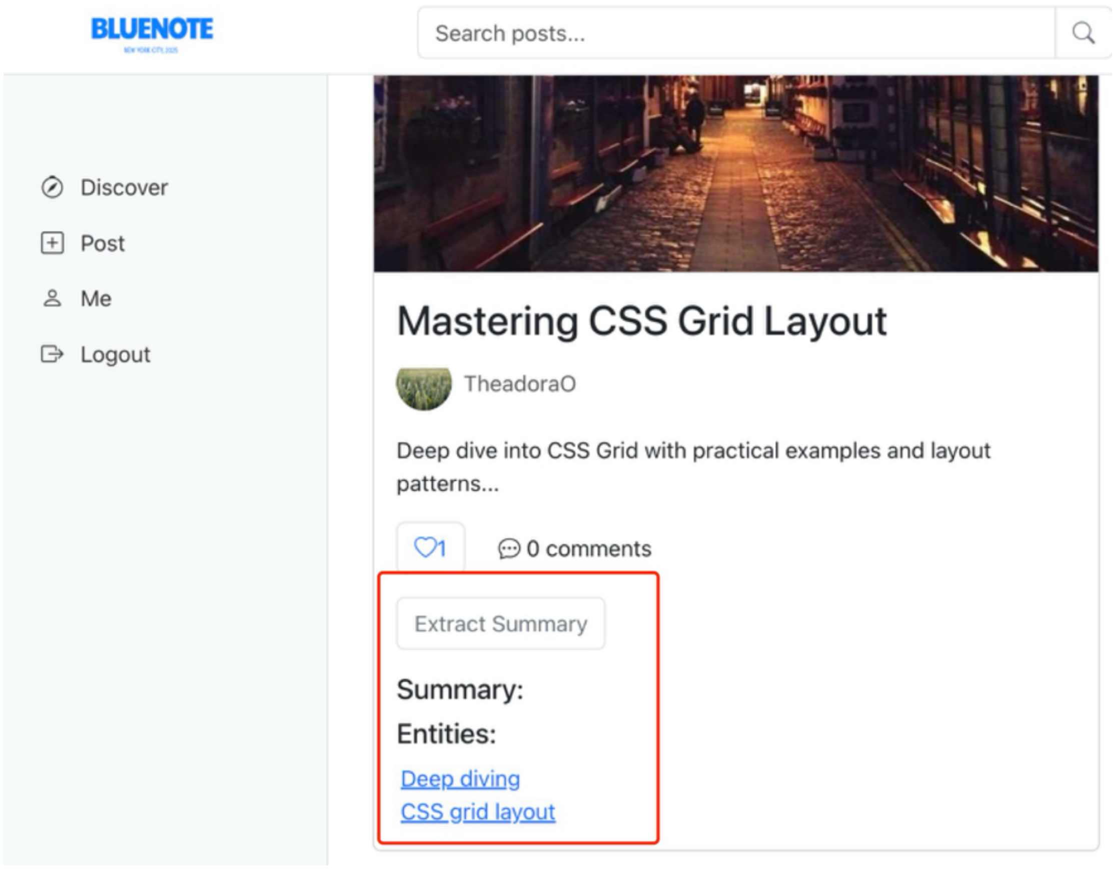


Figure8. AI analysis function

# IV. Details related to assessment criteria

## (i) CRUD operation implementation

Post data collection

Create: After clicking the ‘Publish Post’ button on the front-end page, the user will enter the post editing page. In this page, the user can fill in the title of the post, write detailed content, and upload related images or attachments. The front-end checks the format of the data entered by the user and sends it to the back-end via API. After receiving the data, the back-end further verifies and processes the data, and then stores the post information accurately and correctly in the post table of the database, and records the author ID of the post, release time and other related information.

Read: When the user visits the home page or the post list page, the front-end will send a request to the back-end to get the post list. The back-end queries the post information from the database according to the request parameters, such as sorting by time, sorting by hotness (hotness can be calculated based on the number of likes, comments, and other comprehensive calculations), and returns the query results to the front-end in JSON format. The front-end receives the data, parses and renders it, and displays the list of posts on the page. When the user clicks on a post to view the details, the front-end will send a request to the back-end again, and the back-end will get the detailed information of the post from the database according to the post ID, including the complete content, author information, number of likes, comment list, etc., and return it to the front-end for display.

Update: Under certain conditions (e.g. not locked by administrator, etc.), users can click the 'Edit' button on the post details page for posts published by users. The front-end will fill the original data of the post into the edit form, and after the user modifies the relevant content, the front-end will check the data again and then send the updated data to the back-end. After receiving the data, the back-end will find the corresponding database record according to the post ID and update the corresponding fields to achieve the modification of the post content.
Delete: Users can delete their own posts on the post details page or post list page. The front-end sends a delete request to the back-end, and the back-end deletes the corresponding post record from the database according to the post ID. At the same time, the system will check other data related to the post (such as comments, likes, etc.) and process them according to the business rules (such as deleting comments and likes related to the post) to ensure data consistency.
User Data Collection
Create: On the registration page, the user enters the user name, password, email address and other registration information. The front-end checks the basic format of the user input (e.g., whether the format of the email is correct, whether the length of the password meets the requirements, etc.) and then sends the data to the back-end. The back-end further verifies whether the user name has been registered, whether the mailbox is valid, etc. If the verification passes, the user password will be encrypted (e.g., using the hash algorithm), and then the user information will be stored in the user table of the database, and at the same time generate the user's unique identification and other related data.
Read: When the user logs in, the front-end sends the user name and password to the back-end. The back-end queries the user information from the database according to the user name, obtains the user's encrypted password and other data, then encrypts the password entered by the user and compares it with the password stored in the database to verify the user's identity. After the user has successfully logged in, the back-end can also return the user's personal information (e.g., user name, avatar, user level, etc.) to the front-end for display according to the user's request. In addition, the administrator can also query the list of users and related details in the management background and other scenarios.
Update: After logging in, users can modify their personal information, such as user name (under certain conditions, such as not occupied by other users), avatar, email address, etc. on the profile page. The front-end sends the user's modified data to the back-end, which finds the corresponding database record according to the user ID and performs the update operation on the corresponding fields to achieve the modification of the user's information. If the user modifies the password, the back-end will encrypt the new password before storing it.

Association relationship processing: By recording the user ID field in the post data collection, the association relationship between posts and users is established. When displaying a post, the backend queries the corresponding user information (e.g., user name, avatar, etc.) from the user table based on the user ID in the post, and returns it together with the post information to the frontend, so that the author information of the post can be accurately displayed on the frontend page. On the user's personal page, the back-end queries all the posts published by the user from the post table according to the user ID and returns them to the front-end for display, which realises two-way querying and displaying of the associated data between the user and the post, making it convenient for the user to check his posting history, and for other users to understand the relevant information of the post's author.

## (ii) Token - based authentication

This system uses JWT (JSON Web Token) based Token - based authentication mechanism. The specific process is as follows:
User login: the user enters the user name and password on the login page and clicks the 'Login' button, the front-end encapsulates the user name and password into request parameters and sends them to the back-end login interface.
Authentication and Token Generation: After receiving the login request, the back-end queries the user information from the database according to the user name and obtains the user's encrypted password. Then the password entered by the user is encrypted (using the same hash algorithm as during registration) and compared with the password stored in the database. If the comparison is successful, it means that the user authentication has passed. At this point, the back-end generates a JWT Token containing user-related information (such as user ID, user name, user role, etc.) In the process of generating the Token, the user information will be signed with the system's pre-set key to ensure that the integrity of the Token and tamper-proof.
Token return and storage: the back-end will return the generated JWT Token to the front-end. After the front-end receives the Token, it will be stored locally (such as the browser's Local Storage or Cookie, the specific storage method is determined according to the project requirements and security policies).
Subsequent request verification: In the subsequent operations after the user has successfully logged in, each time the front-end sends a request to the back-end, it will carry the JWT Token in the request header (usually in the Authorisation field, in the format of 'Bearer [Token value]'). When the backend receives the request, it first extracts the Token from the request header, and then verifies the Token with the pre-set key, including verifying whether the signature of the Token is correct, whether the Token is expired, etc. If the Token passes, the request is considered to have passed. If the authentication passes, the request is considered legitimate and allowed to access the relevant resources and based on the user information carried in the Token (e.g., user role) to determine whether the user has the authority to access the resource; if the authentication fails, an error message is returned, and the request is rejected.

Through this Token - based authentication mechanism, it effectively guarantees secure access to the system, prevents unauthorised users from illegally accessing the system resources, and also facilitates the user's identity maintenance and permission control between different pages and operations.

## (iii) Responsive Web Design and User Interface

Responsive design:
Layout technology: The front-end page uses modern CSS layout technology. In the overall layout of the page, Flexbox is used to achieve a flexible arrangement of containers and items, enabling page elements to automatically adjust their position and size according to the screen width. For

example, in the home page post list display area, the use of Flexbox will post cards in accordance with the appropriate spacing and arrangement of the layout, in different screen sizes can maintain a neat and beautiful display effect. For some complex page structures, such as pages containing multiple sidebars and main content areas, Grid layout is used for precise rows and columns division and positioning to ensure that each area can be reasonably displayed on different devices.

Media Queries: Media Queries technology is used extensively to dynamically adjust the style and layout of page elements according to the screen width of the device. Aiming at the common screen sizes of different devices such as mobile phones, tablets and computers, multiple breakpoints are set. For example, when the screen width is less than 768px (generally the range of mobile phone screen size), the page will automatically adjust to a single-column layout to hide some elements that are not very important on a small screen, and increase the size of the text and buttons, so as to facilitate the user's touch operation; when the screen width is between 768px - 991px (generally the range of the screen size of a flat screen), the page will be in a two-column layout, so that a reasonable allocation of space! When the screen width is greater than 991px (generally the range of computer screen size), the page will adopt multi-column layout to make full use of the screen space and display more contents and function modules. By this way, it ensures that the system can present good visual effect and operation experience on various devices.

User Interface:

Interface style: The overall interface design upholds a simple and intuitive style, adopting clear typography and a soft, coordinated colour scheme to ensure that users do not feel fatigued when browsing the post content and operation tips. Buttons, icons and other elements of the design is simple and clear, using a unified style and specifications, easy for users to identify and operate.

Functional Layout: The home page displays posts in a card layout. Each post card contains information such as the post title, author's avatar and name, release time, content summary, and related images (if any), so that the user is able to quickly browse through the key content of the post and decide whether or not to further view the details. The operation buttons (e.g. like, comment, share, etc.) are reasonably located, easy for users to click, and have corresponding animation effects or prompt messages (e.g. the like button will change colour when clicked and show the dynamic update of the number of likes) when the user operates them, so as to enhance the user's interactive experience. On the user's personal page, a split-column layout is used to display the user's information, the list of published posts, favourite posts, etc., which is convenient for the user to manage his/her personal data and related operations. On the registration and login pages, the form design is simple and clear, the arrangement of input boxes and buttons is in line with the user's operating habits, and clear error messages are provided to help users quickly correct input errors.

## (iv) Keyword extraction and wiki link function

A new ‘Extract Summary’ button has been added to the post details page. When the user clicks this button, the front-end will send the post content to the back-end. The back-end will call the Google AI related interface to perform natural language processing on the post content and extract key information and keywords. For example, for a post about ‘Mastering CSS Grid Layout’, keywords such as ‘CSS grid layout’ ‘layout patterns ‘ and other keywords. Then, the backend will process these keywords and convert them into clickable links to the corresponding Wikipedia pages. The front-end receives the processed data and displays the keyword link on the page, and the user can click on the link to jump to the Wikipedia page and get detailed information about the keyword. This feature not only enriches users' access to knowledge, but also enhances the expandability and usefulness of the forum content.

# V. Code quality assessment

Reasonable Routing:

Front-end routing: the front-end uses Vue Router for routing management. According to the functional modules and page structure of the system, different routes are clearly divided. For example, the home page, post detail page, user personal page, registration page, login page, etc. are defined as different routing components respectively. In the routing configuration, routing parameters, navigation guards and other functions are set reasonably. Through the routing parameters, it is convenient to pass data between different pages, such as in the post details page routing, the post ID is passed through the parameter so that the back-end can get the post details data according to the ID. Navigation guards are used to implement some permission control and page jump logic, such as preventing users from accessing some pages that require login privileges when they are not logged in, and guiding users to the login page. Through reasonable front-end routing design, clear page navigation and component loading logic is achieved, which improves the maintainability of the front-end code and user experience.

Back-end routing: The back-end is also routed according to business functions. User-related interfaces (e.g., register, login, get user information, etc.) and post-related interfaces (e.g., publish post, get post list, edit post, etc.) are categorised into different routing modules. Each routing module corresponds to a specific business logic processing function and adopts a unified request method.