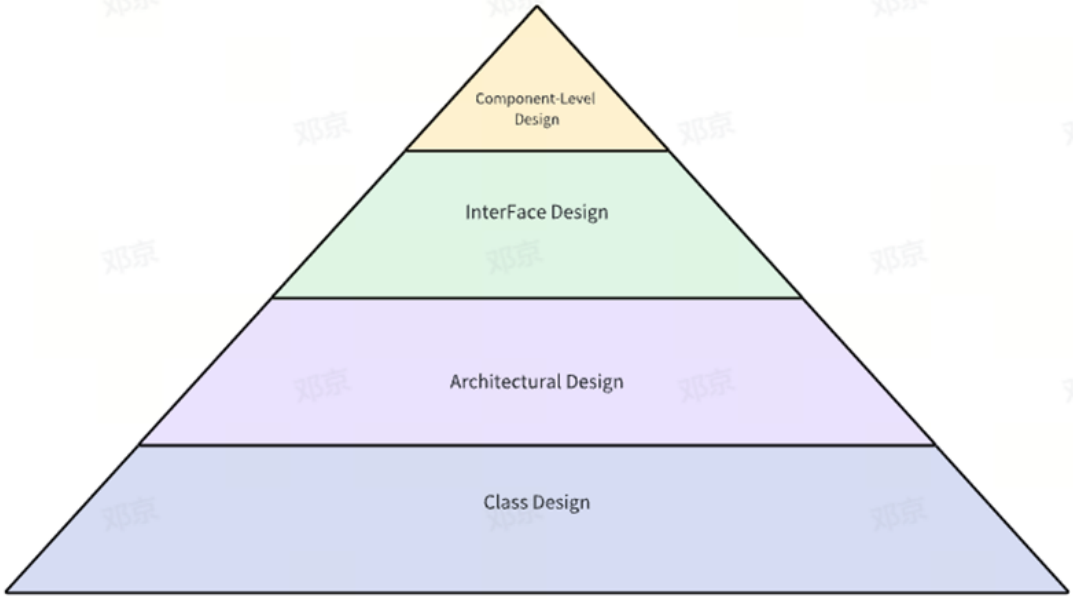
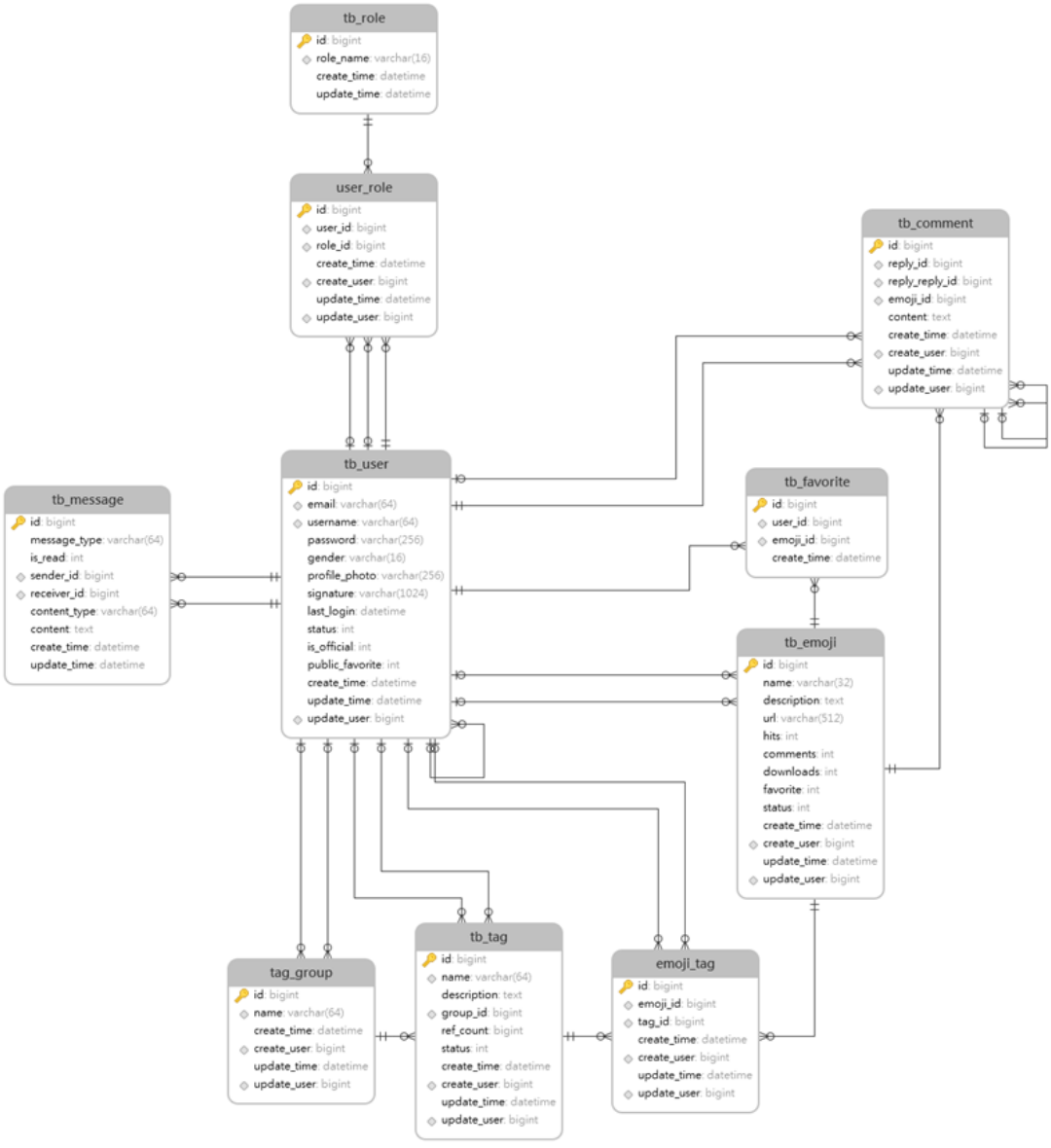
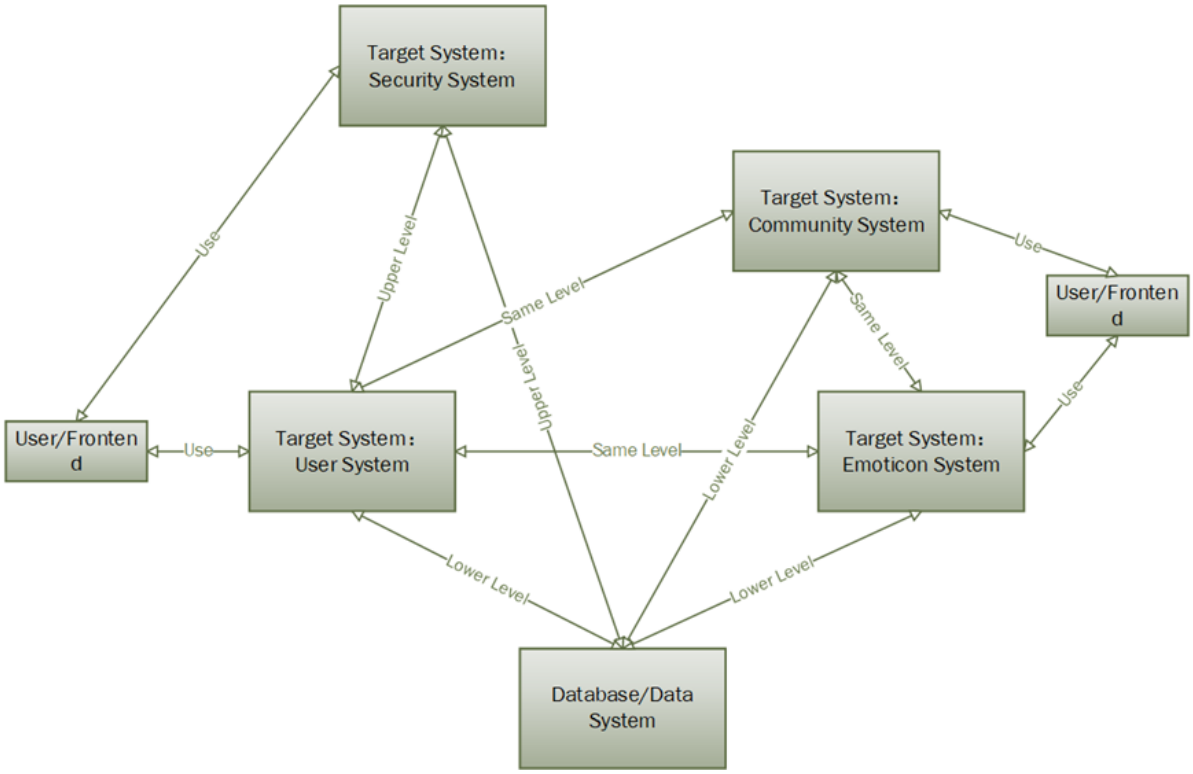
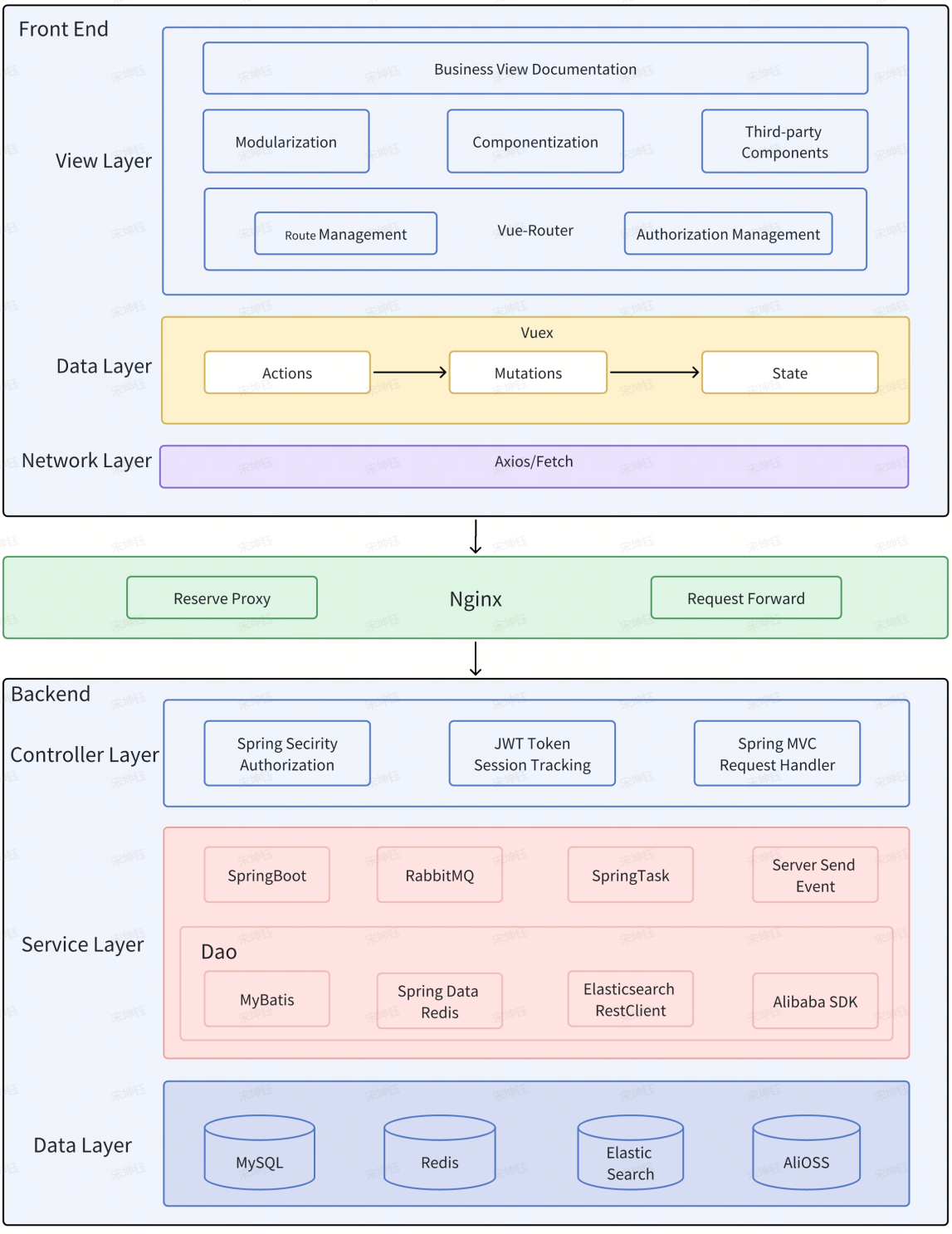


Chapter 1-Class Design



Chapter 2-Architecture Design





1. View Layer

This is the interface layer of the front-end application.

1. Bussiness View Documentation: A document about the business logic and view layer, used to guide developers on how to build front-end interfaces to meet business requirements.
2. Modularization: Divide front-end code into modules to improve maintainability and scalability.
3. Composition: Divide the front-end interface into components, each responsible for a specific function or view.
4. Third party comp: Use third-party components, libraries, or plugins to accelerate the development process.

5. Vue router: The routing manager of the Vue. js framework, used to handle navigation between front-end pages.

(1) Route Management: Manage front-end routing and determine which pages and components are displayed on specific URLs.

(2) Authorization Management: Manage user authentication and authorization, ensuring that users can only access content they have permission to.

2. FrontEnd Data Layer

This is the data layer of the front-end application.

\1. Vuex: Vue.js state management library, used to manage the global state of front-end applications, typically including the storage, management, and update of data and states.

\2. Actions ->mutations ->state: This is the data flow architecture in Vuex. Actions are used to trigger changes to the state, mutations are used to actually modify the state, and state is the application's state data.

3. Network Layer

This is the network layer of the front-end application, using Axios or fetch libraries for HTTP requests.

4. Nginx:

Web server, acting as a reverse proxy server, is responsible for forwarding requests to backend servers.

1. Reverse Proxy: The reverse proxy server serves as the intermediate layer between the client and backend servers, providing functions such as load balancing, caching, and security.
2. Request Forward: The reverse proxy server forwards the client's request to the backend server for processing.

5. HTTP Protocol:

A protocol used to transfer data between clients and servers.

6. Controller Layer

The control layer of the backend application, handling request routing and business logic.

1. Spring Security: A framework for handling authentication and authorization.
2. JWT Token: JSON Web Token, a token used for authentication.
3. Session Tracking: Tracks the status of user sessions.
4. Spring MVC: The MVC (Model View Controller) layer of the Spring framework, used to process HTTP requests and build web applications.

7. Service Layer

1. SpringBoot: Part of the Spring framework used to create standalone, self-contained Java applications.
2. Exception Handler: Handle exceptions in the application to ensure that errors do not cause the application to crash.

3. Junit Test: Use the JUnit framework for automated unit testing.
4. Code Standards: Coding standards and best practices to ensure code quality and consistency.
5. Dao: The main purpose is to separate data access logic from business logic

(1) MyBatis: A persistence framework for Java applications, used to map database operations to Java objects.

(2) Spring Data Redis: A Spring framework module used to interact with Redis databases.

(3) Spring Cache: Cache support for the Spring framework to improve performance.

(4) Alibaba SDK: Alibaba's software development toolkit used to interact with services such as Alibaba Cloud.

8. Backend Data Layer:

This is the layer related to data storage and management in backend applications.

1. MySQL Database: A relational database management system used to store data.
2. Redis Cache: An in memory cache database used for fast data retrieval.
3. Alioss: Alibaba Cloud Object Storage Service, used to store and manage files and data.

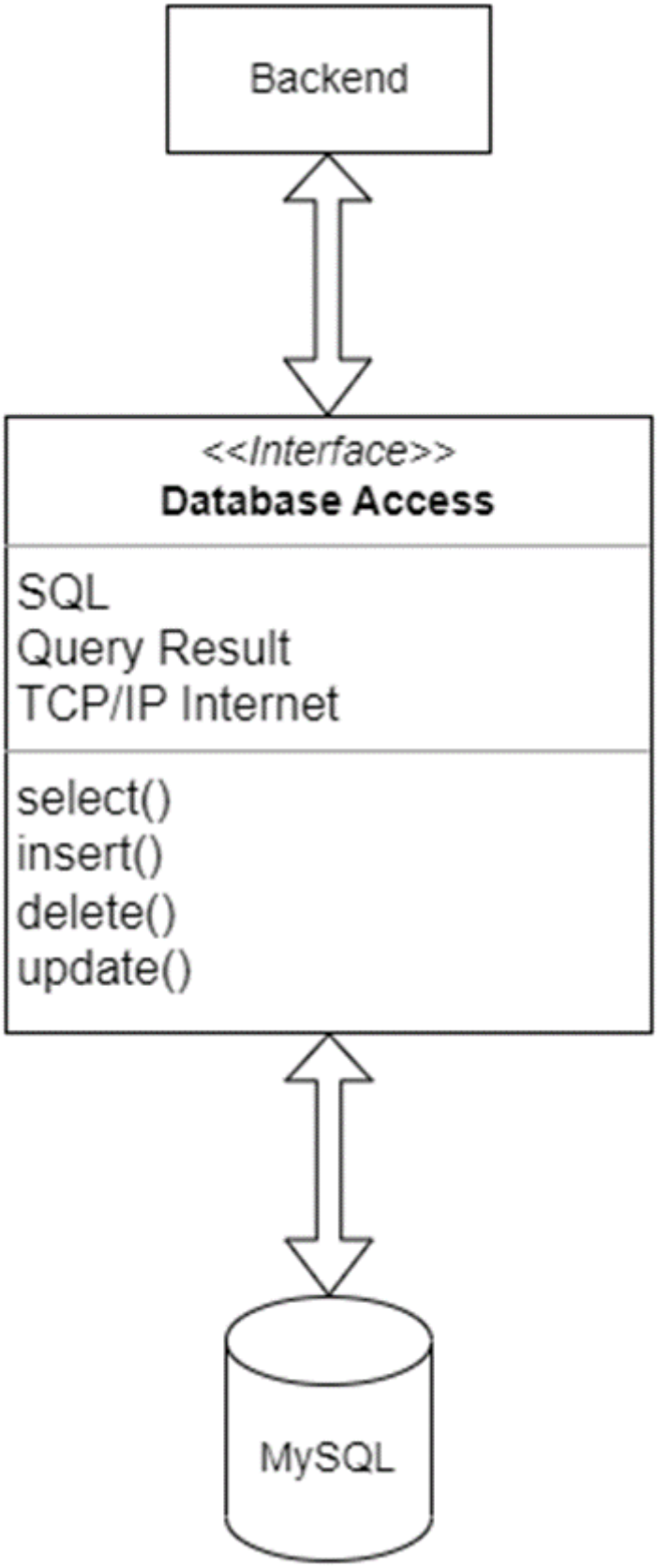
Chapter 3-Interface Design

1.Interface between front and back ends

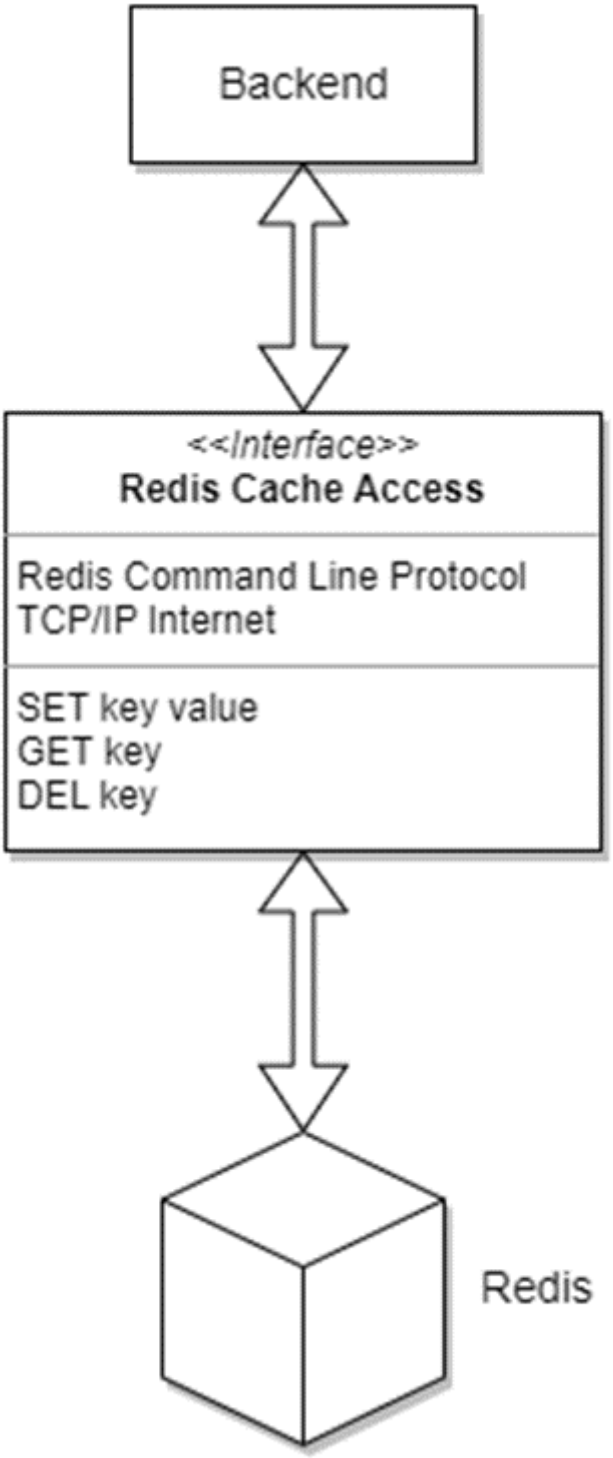
Endpoint	Description	Request Method
/user	Modify user information	PUT
/user/register	User registration	POST
/user/login	User login	POST
/user/{id}	Get user information by ID	GET
/favorite	Add emoji to favorites	POST
/favorite	View favorites list	GET
/favorite	Unfavorite emoji	DELETE
/favorite/status	Modify favorites' public status	PUT
/tag	Get emoji tag list	GET
/tag	Add a new tag	POST
/tag	Get tag list by group ID	GET
/tag/{id}	Delete tag by ID	DELETE
/tagGroup	Add a tag group	POST
/tagGrouplist	View tag group list	GET

Endpoint	Description	Request Method
/tagGroup/{id}	Delete a tag group by ID	DELETE
/emoji	Paginated query for emoji information	GET
/emoji	Upload emoji	POST
/emoji/{id}	Get detailed information by ID	GET
/emoji/uploaded	Query uploaded emojis by the user	GET
/emoji/queryByUserId	Query emojis uploaded by a user	GET
/common/upload	File upload	POST
/common/connect	Establish SSE connection	GET
/common/close	Close SSE connection	GET
/comment	Comment on an emoji	POST
/comment/reply	Reply to a comment	POST
/comment/{emojiId}	Get comments for a specific emoji	GET
/comment/reply/{commentId}	Get replies for a specific comment	GET
/message/unread	Query unread message count	GET

2.The interface between Backend and Database



3.The interface between Backend and Redis Cache



4.Interface between User and System

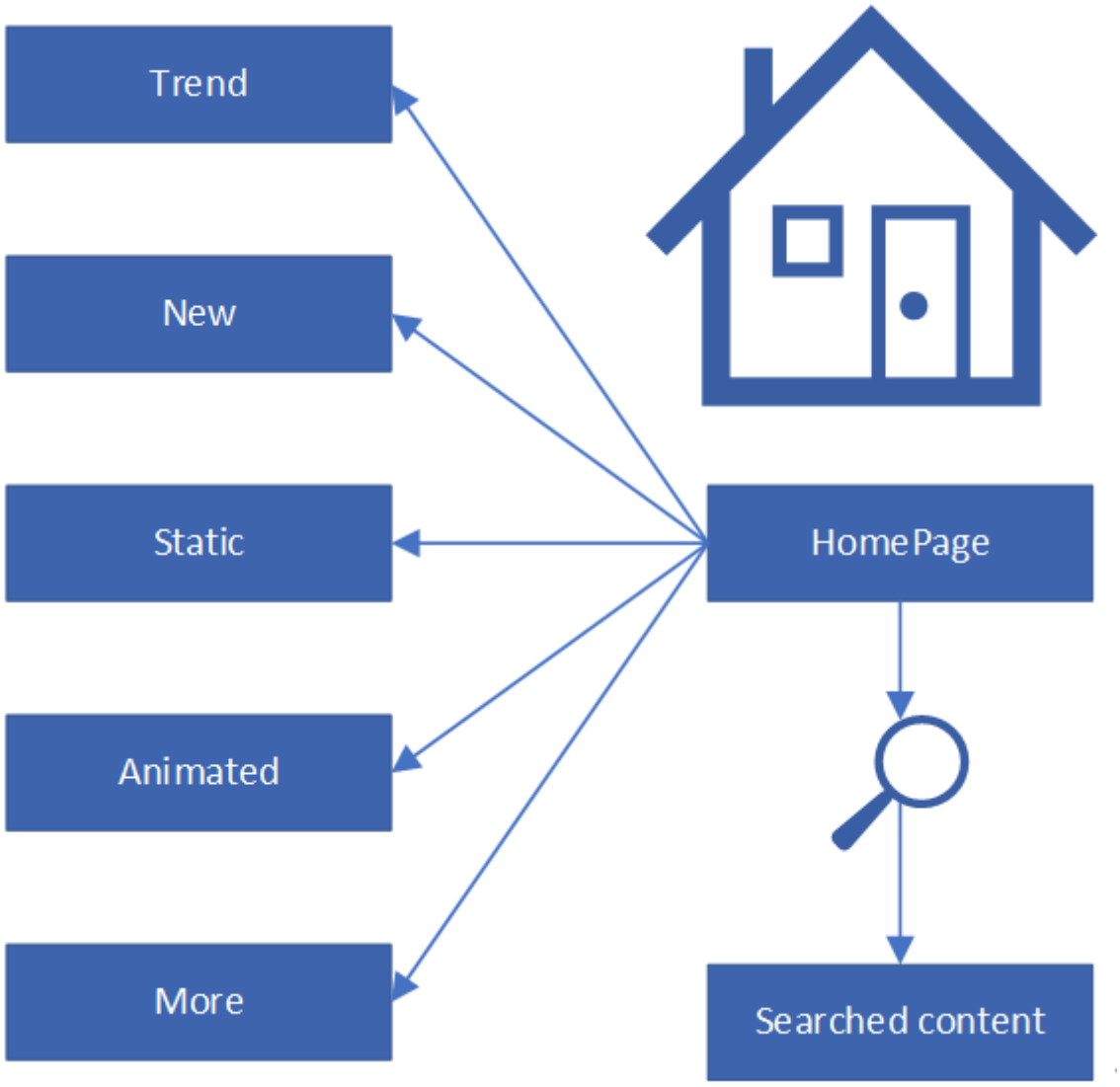
Interface	User-System Interaction Description
User Registration/Login	Users interact by registering and logging in using email, phone number, or social media accounts
Emoji Upload	Users engage with the system by uploading their own emojis
Emoji Management	Users interact with the system to view and manage their uploaded emojis, including editing, deleting, and managing visibility settings

Interface	User-System Interaction Description
Search and Classification	Users interact with the search and classification system to find emojis based on tags, keywords, and popularity
Like and Comment	Users express interaction by liking emojis and leaving comments, fostering social engagement
User Notifications	The system interacts with users by sending notifications about likes, comments, and important platform updates
Social Sharing	Users interact by sharing their favorite emojis on social media platforms
Reporting and Feedback	Users provide feedback and interact with the system through a reporting system for inappropriate content and feedback channels for suggestions
User Statistics	Users engage with the system through statistics and analytics, understanding the popularity of emojis and other platform usage data
Security and Privacy	Users trust the system to ensure the security of their data and implement privacy protection measures, complying with relevant regulations
Developer API (Optional)	Developers interact with the system through API documentation, contributing emojis seamlessly

Chapter 4-Component-level Design

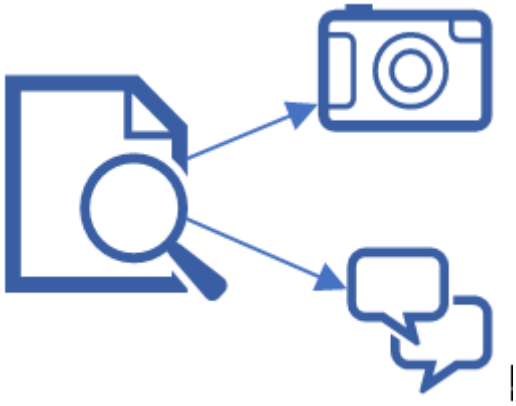
Component-level design for WebApps often incorporates elements of content design and functional design.

The homepage display interface is part of the entire WebApp. The home page display page provides a convenient way for users to view emoji, and is used to display emoji of different categories. Users select an emoticon package to view on this page. A well-designed homepage display interface can be reused to display different categories of emoji by simply modifying its content model.



With careful design, user and author information interfaces can dynamically adjust their appearance to ensure that text, images, and interface control elements are displayed correctly.

The well-designed emoticon display page can be reused, you can view emoji and related information, and it also has a comment function.



The carefully designed upload page can be reused. You can paste or select emoticon packages to upload, add emoticon package description information (name, description, tags, etc.) to adapt to emoticon packages of different sizes and file types.