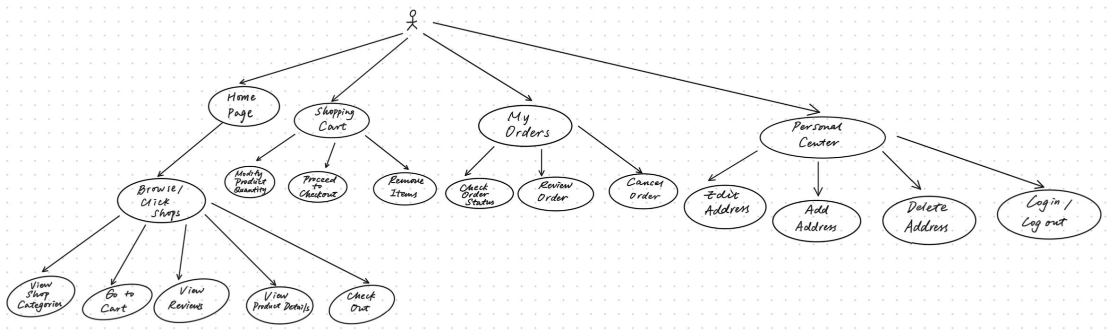
Functional Requirements Analysis

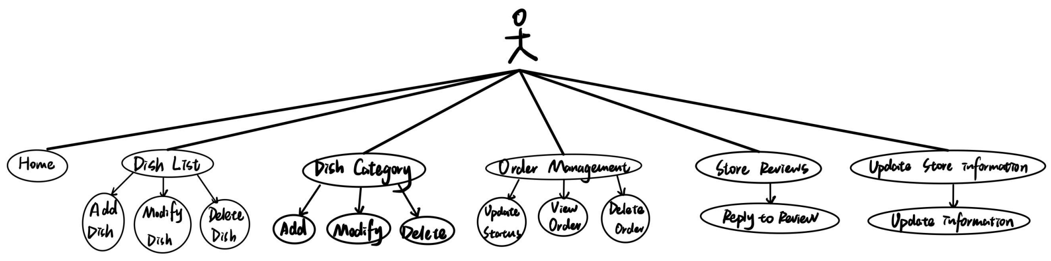
Basic functional requirements of the web page

Users can browse stores, view dishes, add shopping carts, view orders, manage addresses, and other functions on the web page. The details are as follows:
1) Login web page: After entering the web page, you will not log in directly. When placing an order, you will jump to the login page. After logging in successfully, you can select the address.
2) Browse stores: After scanning the code to enter the web page, you can browse the store list and introduction on the homepage.
3) Select a store: After entering the store, view the beverage classification, beverage list, beverage pictures, beverage introduction, etc., and you can add your favorite beverage to the shopping cart or place an order directly.
4) Shopping cart management: Users can adjust, delete, or directly order beverages in the shopping cart.
5) Order page: Users can view all their order details, evaluate completed orders, and delete orders.
6) Personal center: You can log in or exit the web page, and manage addresses in the personal center at the same time.



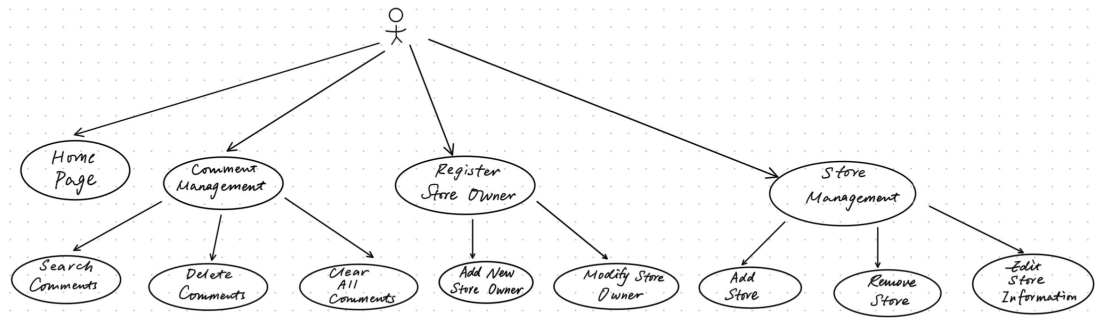
Merchant management terminal functional requirements

The merchant management terminal can modify store information, view orders, modify order status, reply and view comments, classify drinks, change drinks, etc. The store can set recommended drinks, which will be displayed on the user terminal; it also has the function of changing drink information and pictures at any time, updating order status, etc.



Administrator management function requirements

The administrator side is mainly responsible for managing the entire webpage, including managing stores and deleting illegal comments. Stores can also be added or removed.



Non-functional requirements

In addition to functional requirements, the system also focuses on non-functional requirements such as maintainability, usability, and security. It adopts a front-end and back-end separation approach, with the front-end utilizing responsive design to ensure compatibility across various devices and browsers. The back-end follows a layered architecture (Controller, Service, Dao layers) to simplify business logic, enhance code reusability, and improve development efficiency. Regarding usability, the system provides a simple and intuitive interface with comprehensive functionality to ensure a user-friendly experience. For security, the system employs JWT (JSON Web Token) for user authentication, using encrypted tokens for identity verification. Additionally, it encrypts user information and passwords in the database to prevent data breaches and ensure the security of user data.