

**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY**

Faculty of cybersecurity and software engineering
Software engineering department

ADMIT TO DEFENSE
Head of Department
_____ Oleksiy GORSKI
“ _____ ” _____ 2023

**QUALIFICATION WORK
(EXPLANATORY NOTE)**

GRADUATE OF EDUCATIONAL MASTER’S DEGREE

Theme: “Service-oriented means of obtaining the name of a disease using an international code”

Performer: Liu Heyan

Standart controller: Doctor of technical sciences, Associate Professor, Mykola Fyodorovych Radishevskiy

Supervisor: Ph.D, Mykhailo Viktorovich Olenin

Kyiv 2023

NATIONAL AVIATION UNIVERSITY

Faculty cybersecurity and software engineering

Department Software Engineering

Degree of education master

Speciality 121 Software engineering

Education-professional program Software engineering

APPROVED

Head of Department

_____ Oleksiy GORSKI

“ _____ ” _____ 2023

6. Calendar plan-schedule

№	Task	Deadline	Performance note
1.	Familiarization with the statement of the problem and the study of literature Writing 1 section, presentation to the supervisor	14.10.2023- 31.10.2023	
2.	Preprint of section 1 and auxiliary pages (draft) - title, task, schedule, abstract, list of abbreviations, content, introduction, source list. First standard control.	15.10.2023- 22.10.2023	
3.	Writing 2 section, presentation to the supervisor	22.10.2023- 01.11.2023	
4.	Writing 3 section, presentation to the supervisor	01.11.2023- 14.11.2023	
5.	General editing and printing of an explanatory note, graphic material	14.11.2023- 20.11.2023	
6.	Passing standard control	20.11.2023- 26.11.2023	
7.	Development of the text of the report. Creating of graphic material for presentation	26.11.2023- 27.11.2023	
8.	Get feedback from the supervisor, reviews.	27.11.2023- 13.12.2023	
9.	Preparation of materials for transmission to the secretary of the DEC (software, GM, CD-R with electronic copies of software, GM, presentations, supervisors review, review, certificate of progress, 2 folders, 2 envelopes)	13.12.2023- 19.12.2023	
10.	Graduation project presentation	19.12.2023- 31.12.2023	

Date of issue of the assignment 05.09.2023 p.

Supervisor: _____ Ph.D, Mykhailo Olenin
Task accepted for execution: _____ Liu Heyan

ABSTRACT

Explanatory note to the topic «Service-oriented means of obtaining the name of a disease using an international code.»

The object of research is - The name of a disease is obtained by an international code. This means that a specific international code can be provided to the system to obtain the official name of the disease associated with that code. This method is commonly used in healthcare information systems and the health field to quickly and accurately identify and retrieve the names of specific diseases for diagnosis, reporting, and documentation. This helps medical professionals and health institutions to use uniform standards to describe and track diseases, thereby facilitating the standardization and comparison of global health data.

The purpose of the thesis - The main development objective of the development is to provide a standardized way to enable users to query and obtain the corresponding disease name through the International disease Code, supporting the medical information system to enable medical professionals to quickly find and identify diseases based on the International disease Code for diagnosis and treatment. It is used in disease statistics and epidemiological research to help health departments and disease control agencies track and analyze diseases in different regions and populations. In the field of health insurance, it is used to process claims, authorizations and reimbursements and to identify the illness of the insured through an international disease code. It is used in medical research, academic research and clinical trials to more easily identify diseases and related information. Support data exchange between different healthcare information systems to ensure that healthcare information can be shared seamlessly.

Development type - is usually a service-oriented architecture or a more modern microservices architecture. This service architecture is characterized by the partitioning of functions into small independent services that can communicate with each other and provide interfaces to the outside world over a network. This service can communicate using standard Web protocols, such as HTTP, and can be

deployed and extended independently, helping to improve the flexibility and maintainability of the system.

Therefore, the type of development of this project falls under the category of service-oriented architecture or microservice architecture, which aims to provide users with a service to obtain disease names through international disease codes. This service can be integrated with other applications or systems to obtain information about diseases.

Hardware and software – PC with Windows 11 or Win10 operating system; environment for object-oriented programming VS Code. The use of artificial intelligence methodology is impossible without an Internet connection

The predicted assumption about the development of tools - The predicted assumption about the development of tools can involve a number of aspects, depending on the tool being developed and the application area.

Market demand:

Prediction: Whether the market demand for new tools will grow steadily or whether there will be high demand in a specific period of time.

Assumption: User or industry demand for this tool is based on a specific issue or trend, such as digital transformation, health concerns, or environmental sustainability.

Technology Trends:

Prediction: Will technological developments have a significant impact on the functionality, performance, or safety of the tool?

Hypothesis: The success of the tool may depend on its integration with the latest technological trends, such as artificial intelligence, iot, or blockchain.

Regulations and Compliance:

Prediction: Will the regulatory environment impose restrictions or requirements on the development and use of tools?

- Assumption: Compliance with regulations and compliance standards is required to ensure the legality and security of the tool.

Business model:

Prediction: How will the tool's business model generate revenue? License fees, subscription models, advertising, etc.?

Assumption: Different business models may work for different types of tools and markets.

User experience:

Prediction: Will the quality of the user experience affect tool adoption and user retention?

- Assumption: Optimizing the user interface, response time, and availability may be the key to success.

Sustainability:

- Forecast: How sustainable and long-term is the tool?

Assumption: Sustainability strategies, including funding, talent, and growth plans, may influence the future of the tool.

TABLE OF CONTENT

catalogue

LIST ACRONYMS AND ABBREVIATIONS	8
INTRODUCTION	10
CHAPTER I	12
Analyze the main features of a web service that obtains disease names based on international codes. Implementation of major web functions.	12
1.1. Background	12
1.2. Research Objectives	12
1.3. Introduction to the basics of ICD11	13
1.3.1. Overview of the service	13
1.3.2. ICD-11 API	16
1.4. Practical studies of use.	17
1.4.1. Overview	17
1.4.2. Requirements	18
1.4.3. Development and usage issues	18
1.5. Develop ideas with expected implementation features.	20
1.6. Content that may need to be refined after development and subsequent content modules that need to be maintained	22
1.6.1. Testing and validation	22
1.6.2. Documentation and support	23
1.6.3. Choice of development tools	25
1.7. Programming language	26
1.8. Challenges	27
CHAPTER 2	30
Requirements for the code generation system and improvements to the web page module	30
2.1. Web Design Requirements	30
2.2. Specific requirements for database in the development of ICD11API	32
2.2.1. Capabilities for backend part of web app applications	32
2.2.2. Capabilities for frontend part of web app applications	34
2.3. Non-functional optimization of the ICD11API	36
2.3.1. Requirements for backend part of web app applications	36
2.3.2. An introduction to the non-functional usability of the ICD11API	37
2.4. Detailed Description	38
2.4.1. Capabilities for backend part of web app applications	38
2.4.3. Requirements for backend part of web app applications	44
2.4.4. Requirements for frontend part of web app applications	45
Conclusion	49
CHAPTER 3	50
3.1. Database selection and structure of the API system of ICD11	50
3.2. Flow chart of registration and cooperation of users with different identities	52
3.3. ICD11API WEB side background database overview and database key data components schematic	53
3.4. ICD11API WEB side key entity class diagram	54
3.5. Establish a physical layer connection	56
conclusion	58
CHAPTER 4	60
4.1. The development of ICD11 coding API display system is briefly introduced	60
4.2. ICD11API WEB side results display and source code display	61
4.3. Web results display and function detailed introduction	75

Conclusion	85
CONCLUSIONS	87
LIST OF REFERENCES	89
APPENDIX A	91
APPENDIX B	93

LIST ACRONYMS AND ABBREVIATIONS

ICD-11: International Classification of Diseases, 11th Revision (ICD-11) - An international standard developed by the World Health Organization (WHO) for diagnosing diseases and health statistics.

ICD-11-CM: International Classification of Diseases, 11th Revision, clinical Modification For clinical diagnosis and reporting in the United States.

ICD-11-AM: International Classification of Diseases, 11th Revision, Australian Modification (ICD-11) International Classification of Diseases, 11th Revision, Australian Modification (ICD-11) For clinical diagnosis and medical statistics in Australia.

Mortality and Morbidity Statistics: ICD-11 for Mortality and Morbidity Statistics - A specific version of ICD-11 for recording and reporting statistics on mortality and morbidity.

ICD-11-CM/PCS: International Classification of Diseases, 11th Revision, Clinical Modification/Procedure Coding System (International Classification of Diseases, 11th Revision, Clinical Modification/Procedure Coding System) Clinical Modified Edition/Operational Coding System) - a set of standards used for medical diagnostics and medical operational coding in the United States. When programming, you often come across abbreviations and acronyms. Here are some common programming abbreviations and what they mean:

HTML: HyperText Markup Language - The markup language used to create web pages.

CSS: Cascading Style Sheets - Used to define the style and layout of a web page.

JS: JavaScript - a scripting language for web page interaction and dynamic content.

API: Application Programming Interface - A specification that allows different software to communicate and interact with each other.

SQL: Structured Query Language - A language used to manage and query databases.

IDE: Integrated Development Environment - An integrated development tool that provides coding, debugging, and build tools.

OOP: Object-Oriented Programming - A programming paradigm that emphasizes the concepts of objects and classes.

URL: Uniform Resource Locator - An address used to identify the location of resources on the Internet.

HTTP: Hypertext Transfer Protocol (HyperText Transfer Protocol) - A protocol used to transfer data over the Web.

FTP: File Transfer Protocol (FTP) - The protocol used to transfer files on the network.

DNS: Domain Name System - The system used to translate domain names into IP addresses.

JSON: JavaScript Object Notation - a lightweight format for data interchange.

API: Application Programming Interface - An interface that allows different software components to communicate with each other.

SDK: Software Development Kit - A collection of tools, libraries, and documentation for developing a piece of software.

GUI: Graphical User Interface - A user interface for interacting with a computer through graphical elements such as buttons and Windows.

MVC: Model-View-Controller - A design pattern used to organize the structure of an application.

DBMS: Database Management System - Software used to manage a database.

CMS: Content Management System - Software used to create and manage the content of a website.

IoT: Internet of Things (iot) - refers to the network of physical devices and objects that connect and interact.

SSH: Secure Shell - A protocol used to secure remote access to computers on a network.

INTRODUCTION

Project Overview: Service-oriented means of obtaining the name of a disease using an international code is a project designed to provide disease name query services for the medical industry and related fields. The goal of this project is to provide the medical field with a useful tool to help professionals more easily access disease information and is expected to have a positive impact in the fields of medicine, research and health management. It allows users to quickly obtain the corresponding disease name via international disease codes such as ICD-10 or other standard codes. This service will be provided through a network interface (API) so that other applications and systems can easily integrate and access disease information.

Project objective: The main objective of the project is to provide a convenient, efficient and standardized way to query disease names against international disease codes. This helps medical professionals, health information systems, health research organizations and insurance companies, among others, use disease coding more effectively in a variety of applications.

Main functions:

1. Provide an easy-to-use API that returns the corresponding disease name by entering the international disease code.
2. Database management: Store the mapping relationship between disease codes and names to ensure the accuracy and real-time data.
3. Security: Use encrypted communications and authentication to protect user data.
4. High performance and scalability: able to handle a large number of queries and provide fast responses.
5. Documentation and support: Provide detailed documentation and support to help developers integrate the service.

Audience: Key audiences for the project include medical professionals, health information system developers, disease statisticians, medical researchers, health insurance companies, and health care decision makers.

Technical architecture: The project will use a modern service-oriented architecture with RESTful apis for communication. The data will be stored in a relational or NoSQL database and deployed on a cloud server to ensure scalability.

Business model: The project's business model may include both free and

paid subscription plans, with paying users enjoying more advanced features and support.

Regulatory compliance: The project will comply with regulatory and compliance requirements related to the privacy of medical information and personal data to ensure the security and privacy of user data.

Future plans for the project include expanding support for additional disease coding standards, improving the quality and timeliness of data, and providing more advanced features such as historical data tracking and multilingual support.

CHAPTER I

Analyze the main features of a web service that obtains disease names based on international codes. Implementation of major web functions.

1.1. Background

The ability to obtain disease names according to the International Classification of Diseases codes plays an important role in the field of medicine and health information management. The development prospects of this function include improving the accuracy of medical diagnosis, promoting medical research and health statistics. With the proliferation of digital medical records, this feature is expected to play a greater role in medical information systems to provide patients with more accurate medical services.

Contextually, the International Classification of Diseases (ICD) is a globally recognized standard for identifying, tracking, and counting a wide range of health problems and diseases. By associating diseases with unique codes, ICDs provide a common language for medical professionals, researchers, and health policymakers, facilitating the sharing and comparison of global health information. Therefore, the ability to obtain disease names according to international codes is part of the more efficient information management within this system of standards.

1.2. Research Objectives

By achieving these goals, you can ensure that your web application meets the needs of your users and maintains the maintainability of your system while providing accurate information, including:

- Accuracy and up-to-dateness: Provide an accurate and up-to-date system to ensure that users can obtain the most up-to-date disease information in line with the International Classification of Diseases standard.
- Ease of use: Develop a user-friendly interface that enables medical professionals and other interested personnel to easily enter or query ICD codes to obtain the corresponding disease names.

- **Reliability and stability:** Establish a stable and reliable system to ensure consistent service under various network conditions, and avoid system failures or interruptions as much as possible.
- **Security:** Ensure the privacy and security of information entered and obtained by users, and take appropriate security measures, such as data encryption and authentication.
- **Cross-platform and scalability:** Design your system to accommodate different platforms and devices, while being scalable so that you can easily add new features or upgrade your system in the future.
- **Internationalization & Localization:** Consider global use, support multi-language and multi-regional needs, and ensure that the system can meet the needs of users in various cultures and contexts.
- **Integration:** Integrate with other medical information systems or health management platforms to provide more comprehensive medical information support.

1.3. Introduction to the basics of ICD11

1.3.1. Overview of the service

ICD-11 API Web refers to a web-based programming interface that allows developers to access and interact with data and functionality related to the International Classification of Diseases (11th Revision) (ICD-11) via the Internet. ICD-11 is a standard classification system for diseases and health-related issues published by the World Health Organization (WHO).

ICD-11 is a redesigned classification system that better supports data collection in IT systems, and has the advantages of being more systematic and interoperable than previous coding systems. It can not only inherit the expression of the previous ICD-10 for diagnosis and other information, but also connect with ICF, ICHI and other coding systems. It refers to the concept and method of SNOMED, deconstructs information such as diagnosis to more refined information, and reliably realizes information sharing between people and between humans and machines.

It can express: Illness, Service Contacts, Episode of Care, Course and other information. The main features of ICD-11 are "Extended" and "Combinatorial". "Extension" refers to an extension in the sense that comes with additional code (i.e., extension code). The extension code itself does not contain diagnostic information, but does describe additional information about the disease/health condition. By

setting the extension code, the coding surge caused by adding dimension codes to the classification is avoided, and the classification is more flexible. When you want to express the diagnostic information in a simpler and more flexible way, you can use the "&"/"/" operator to "combine" the ICD-11 diagnosis code together to form a diagnostic statement to better express the clinical information.



Fig. 1.1. “World Health Organization (WHO) presentation page on ICD11”

The ICD-11's new technical architecture makes coding even simpler. The digital fabric allows coding tools to be embedded into local digital medical records and IT systems using a local version of a WHO-provided system or a web-based version (known as an application programming interface or API). Clinicians can search for diagnoses using natural language or preferred terms, thus associating them with the correct technical codes (without requiring the clinician to memorize these codes). Its integration with existing digital medical record systems combines recording with coding, reducing the number of steps required to obtain a complete record, improving user compliance, and reducing the cost and time of training. The core content of the ICD-11 ontology can be quickly expanded to include new terms, synonyms and concepts in all language versions, or to improve user guides. The customized version of the specialty facilitates its use in the field of a specialty (e.g. mental health department). For the use of paper medical records, paper indexes or related subsets can provide a quick lookup of the code.

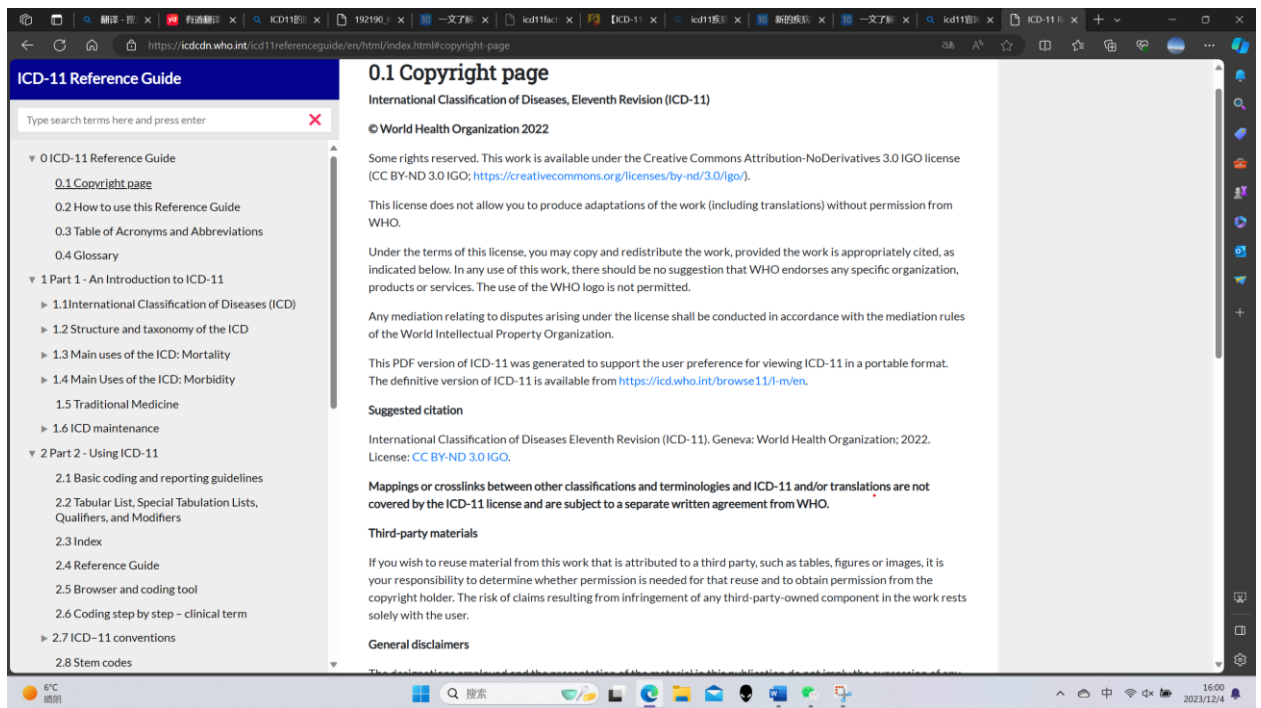


Fig. 1.2. “Reference Guide for ICD11 (WTO)”

ICD-11 is rich in content.

ICD-11 covers 54,000 concepts of entities, which can be diseases, disorders, causes of trauma, signs and symptoms, etc., and each entity has 13 dimensions of attributes to describe, representing system structure, clinical manifestations, causal attributes, etc.

The international code mentioned in this article is the International Classification of Diseases (ICD) code, which is the basis for health statistics and maps the human condition from birth to death: injuries or illnesses we encounter in our lives and anything that could lead to death, are encoded. Moreover, the ICD captures some of the factors that affect health, or external causes of death and morbidity, giving a holistic view of all aspects of life that have an impact on health.

The ICD-11 API Web service allows developers to retrieve disease codes, descriptions, classifications, and more from the ICD-11 database for use in a variety of applications, websites, and systems. This API allows medical professionals, researchers, and developers to integrate ICD-11 standard medical data into their applications to aid in areas such as diagnostics, statistics, research, and health management.

Through the ICD-11 API Web, users can programmatically access ICD-11 data to better understand and utilize this globally accepted medical classification system. This helps promote standardization and interoperability of healthcare data, as well as improve disease management and health research on a global scale.

1.1.1.3.2. ICD-11 API

The ICD-11 API (Application Programming Interface) is a programming interface that allows developers to access and integrate data and functionality from the International Classification of Diseases, Release 11 (ICD-11) into their applications, websites, or systems, and is an international standard classification system for standardizing the recording of medical diagnoses and statistical diseases. This API enables developers to programmatically interact with ICD-11's databases and services, allowing ICD-11's standard disease coding and related information to be used in a variety of application scenarios.

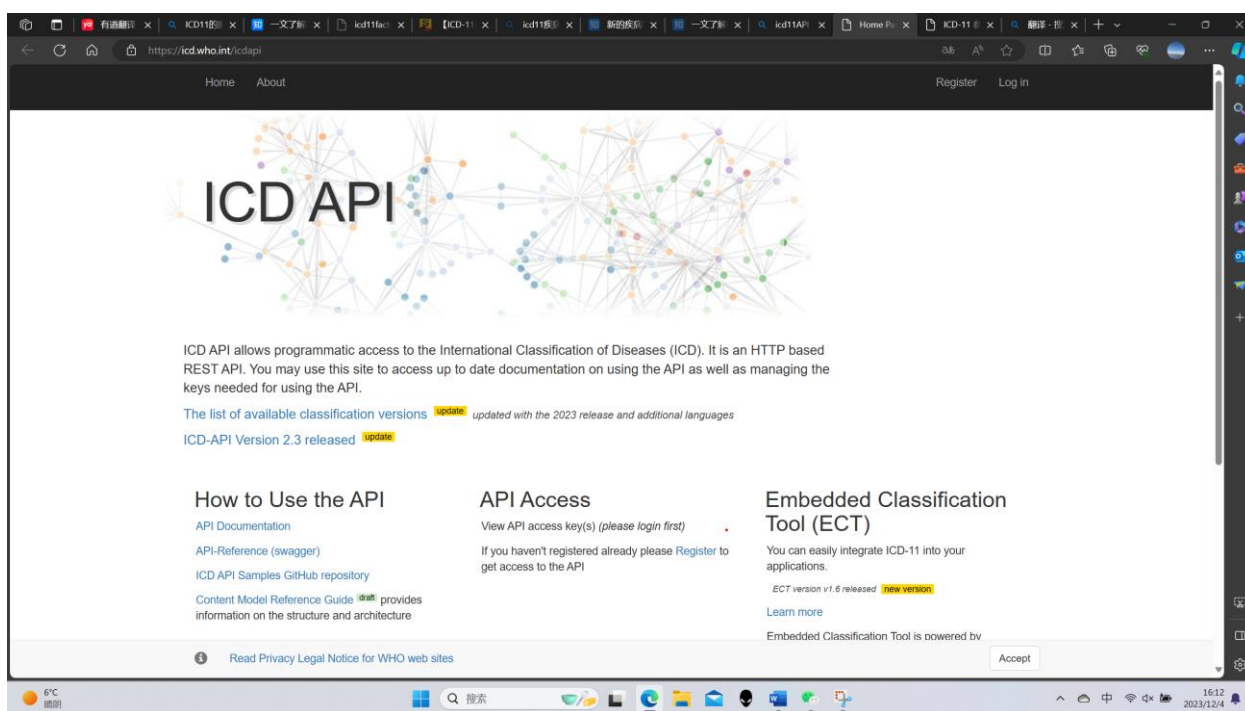


Fig. 1.3. “A general introduction to the API”

According to the tip, the main functions usually provided by the current ICD-11 API include but are not limited to the following:

The first is the data retrieval function, where ICD-11 allows developers to retrieve information related to a disease based on a specific ICD-11 code or other identifier, such as the corresponding name of the disease, a detailed description of the disease, and other attributes of the disease. Then there is the classification browsing function, ICD-11 provides the ability to programmatically browse the ICD-11 classification system to view individual disease categories and subcategories. There is also a search function, which allows developers to perform advanced searches to find disease codes related to keywords, symptoms, or other queries. And to keep up to date and synchronized, we assume to include information about ICD-11 updates and changes in order to keep the application up to date with the latest standards. ICD-11 also has an integrated support feature that provides technical support and

documentation for integrating ICD-11 data into healthcare information systems, health applications, research tools, and more.

Developers can use the ICD-11 API to create medical applications, data analysis tools, clinical information systems, and more to better manage and leverage ICD-11's standardized medical information. This helps foster innovation and data exchange in healthcare. To use the ICD-11 API, you usually need to register and obtain an API key in order to access the relevant services.

1.4. Practical studies of use.

1.4.1. Overview

The development of the ICD-11 API can provide broader and more practical support for your healthcare services and information management, enabling more precise, efficient and advanced medical practice:

The ICD-11 API provides easy access to the latest ICD information to provide reliable data support for medical practice, research and decision-making, using the API to quickly and accurately identify and obtain information on specific diseases, helping to improve the accuracy of clinical diagnosis, through the API, you can analyze and count disease data, providing an important data foundation for public health research, integrating the ICD-11 API into your medical information system can make the system more comprehensive and flexible and improve work efficiency, provide detailed information about ICD-11 coding to help healthcare institutions and decision-makers make more informed decisions, improve medical services and health policies, through the use of ICD-11 APIs will comply with international medical standards and help ensure smoother collaboration with other medical institutions and research teams, develop ICD-11 APIs to make healthcare information systems easier to use and flexible, thereby improving the user experience (doctors, researchers, etc.), The API is developed to keep the ICD-11 standard up to date and keep the system up to date with the latest medical knowledge.

1.4.2. Requirements

1. Access rights: Developers need to register or obtain access to ICD-11 APIs. This typically involves requesting access keys or credentials from the API provider in order to access ICD's API services.
2. Technical requirements: Ensure that our computer or application meets the technical requirements of the API provider. This may include support for specific operating systems, programming languages, network protocols, or other technical standards.
3. Terms and Conditions: You need to agree to the ICD-11 API Terms and Conditions. These agreements typically include provisions regarding data use, privacy, compliance, and access control.
4. API documentation: Read the documents and instructions provided by the API provider to understand how to use the API correctly, including request format, response format, endpoint and other information.
5. Authentication and authorization: When using the API, it is often necessary to authenticate and provide appropriate authorization credentials as required by the API provider to ensure that only legitimate users can access the API.
6. Purpose of Use: Ensure that the use is consistent with the design purpose and lawful use of the API and does not violate any regulations or policies.
7. Security considerations: Take appropriate security measures when using the API to protect the security of data and systems.
8. Maintenance and monitoring: After using the API, it is often necessary to maintain and monitor the application regularly to ensure its proper functioning and performance optimization.

We need to visit the official website of the ICD-11 API or contact the API provider for detailed information on how to register, gain access, and properly use the API, as these requirements may vary depending on the specific provider and service of the ICD-11 API. However, following the API provider's guidelines and regulations will help ensure that we can successfully use the ICD-11 API Web service.

1.4.3. Development and usage issues

While developing the ICD-11 API, there are a variety of issues and challenges that can be encountered, and we will focus on those that cover the technical aspects. The Coding Rules of ICD-11 are a set of guidelines and specifications for assigning unique codes to various health problems and diseases. ICD-11 uses a multi-axis structure that includes the primary diagnosis, associated etiology, symptoms, pathology, and social context. These axes allow for a more detailed description of

the patient's health problems. ICD-11 uses a linear coding system, which means that each code is unique and can accurately represent a particular health problem. The code is usually composed of letters and numbers in the form "A00.0".

Primary diagnostic code: The primary diagnostic code is used to indicate the primary cause or primary disease that causes a patient to seek medical attention. This is often the most important diagnosis a doctor records at a patient's visit.

Additional codes: In addition to the primary diagnostic codes, ICD-11 allows additional codes to be provided for related causes, symptoms, complications, etc. This helps to give a more complete picture of the patient's health problems. In some cases, two codes can be used at the same time to represent a disease or health problem to provide more information. For example, it is possible to encode both the anatomical site and the cause of a disease.

Usage Specifications: Medical professionals need to follow ICD-11 coding rules to accurately record patient diagnostic information. This helps ensure consistency and comparability of medical data. The ICD-11 coding system is regularly updated to reflect the latest advances and discoveries in medical science. This helps keep the coding system accurate and useful.

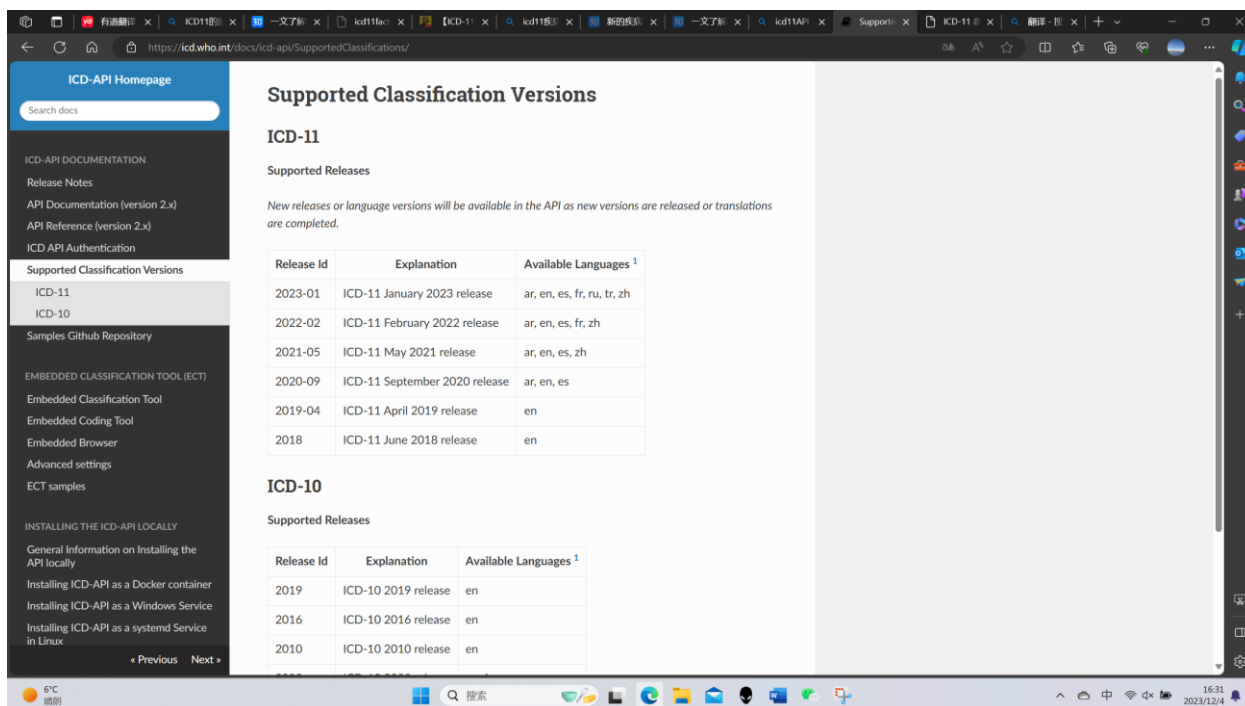


Fig. 1.4. "Classified versions supported by ICD11"

Overall, the ICD-11 coding Rules are a set of standardized guidelines designed to help medical professionals and health organizations accurately record and exchange disease data to support effective medical diagnosis, health policy, and statistical analysis.¹

Data quality issues: Medical data is often complex and may have inconsistencies, missing data, or erroneous data. Ensuring the accuracy and completeness of data can be a challenge.

2. Privacy and compliance: Handling medical data requires compliance with strict privacy regulations such as HIPAA, and ensuring data security and compliance can require complex technical and policy measures.

3. Data standardization: ICD-11 uses specific criteria to represent disease and health-related information. Standardizing data to ICD-11 format can require significant data cleansing and conversion efforts.
4. Performance issues: If the API needs to process large amounts of medical data, performance can become an issue. Performance optimization and caching strategy considerations are required.
5. Security issues: Handling medical data requires strict security measures, including access control, data encryption, and security audits. Protecting data from malicious attacks can be a challenge.
6. Multilingualism and localization: If your API will be available to users on an international scale, you need to consider multilingual support and localization issues, including translation and cultural differences.
7. Version control: If the ICD-11 standard changes, you need to ensure that the API can support multiple versions and that the upgrade does not break existing integrations.
8. Documentation and support: Providing clear, accurate API documentation and providing support is key, but it can also require a significant investment of time and resources to maintain documentation and provide support.
10. Regulatory Review: In some regions, medical APIs may be subject to regulatory review to ensure they comply with regulatory requirements. This can lead to additional complexity and time costs.
11. Testing and Validation: Ensuring the accuracy and performance of APIs requires extensive testing and validation, including testing for functionality, performance, security, and privacy.

1.5. Develop ideas with expected implementation features

The user interface design of ICD-11 disease classification coding system involves several key aspects, and the following are my ideas and ideas

1. Search function: Search function is the core of ICD-11 user interface. Users should be able to enter keywords, such as disease names, symptoms, or codes, into the search box to find relevant information. The search function should have auto-completion to help users find the information they need more quickly. This is important for medical professionals in the actual diagnosis and documentation process.
2. Browsing function: Users should be able to browse the classification structure of ICD-11. This means providing a hierarchical menu or tree structure so that users can click on different categories and subcategories to gain insight into relevant

information. This is very helpful for users to understand the relationships between diseases and the hierarchy of classifications.

3. Detailed information: For each code and classification, the user should be able to obtain detailed information. This may include a complete definition of the code, relevant causes, symptoms, clinical criteria, and treatment recommendations. This helps medical professionals understand the characteristics of a particular disease or health problem to support better diagnosis and treatment decisions.

4. Multi-language support: ICD-11 is an international standard, so the user interface should provide multi-language support. This allows users to access the system from different countries and regions to meet the needs of global users. Multilingual support helps ensure that all users can understand and use the coding system.

5. Interactivity: The user interface should have a certain degree of interactivity, allowing users to perform various actions. This may include saving the code of interest, adding comments, exporting the code as a document, or sharing information. This increases user flexibility and personalizes the experience.

6. Version control: If the ICD-11 coding system is updated frequently, the user interface should provide different versions of the coding system so that the user can view the data of the old version and understand the changes of the latest version. This is very useful for research and data comparison.

7. User assistance and training: Provide users with help documents, training materials and online support to help them understand how to properly use the ICD-11 coding system. Training materials can include video tutorials, FAQs, and user manuals to reduce the learning curve.

8. Filtering and sorting: Users should be able to sort and filter codes according to different criteria to meet their specific needs. This is useful for quickly finding specific types of coding or data analysis.

9. Data import and export: The user interface should support the import and export of data so that users can easily integrate the ICD-11 encoding system into their healthcare information system. This facilitates the exchange and integration of data.

10. Accessibility: Ensure that the user interface is easy to access, including user friendliness for people with visual impairments or other special needs. This includes providing high contrast options, screen reader compatibility and other accessibility features.

11. Security: Protecting the privacy and security of user data is very important. Ensure that the user interface meets data privacy regulations and only allows authorized users access to sensitive information.

User interface design is crucial to the successful application of ICD-11 coding system. A user-friendly, fully functional, and easy-to-navigate interface will help healthcare professionals use the coding system more effectively, thereby improving medical diagnostics, data management, and health policy making.

The development of the ICD-11 API Web can facilitate digital transformation in the medical and health sector, improve the availability and accessibility of medical information, and contribute to better medical care, disease surveillance and global health research. This has potential benefits for medical professionals, researchers, developers and patients alike.

1.6. Content that may need to be refined after development and subsequent content modules that need to be maintained

1.6.1. Testing and validation

Once the ICD-11 API has been developed, thorough testing and validation is required to ensure that the API functions properly and performs well, while meeting the expected requirements and standards. Here are some of the key aspects to test and validate and how to test them:

Functional testing: Ensure that the basic functionality of the API is functioning properly. This includes testing that requests and responses from each API endpoint work as expected, including error handling for normal requests and exceptions.

Integration testing: Test the integration of APIs with other systems or services to ensure that the individual components work together. This can be done by simulating communication with other systems.

Performance testing: Evaluate the performance of your API, including response time, throughput, and load capacity. Commonly used performance testing tools include Apache JMeter, LoadRunner, etc.

Security testing: Conduct security testing, including vulnerability scanning, penetration testing, and cross-site scripting (XSS) checks to ensure that APIs are free of potential security risks.

Reliability testing: Testing the stability and reliability of APIs, including long-running, high-load testing to ensure that APIs function properly in a variety of situations.

Load testing: Test the performance of an API under high load conditions to determine its scalability and performance limits. This helps determine if horizontal scaling is required.

Data conformance testing: Ensure that the API returns consistent results for the same request and does not cause data inconsistencies or conflicts.

Version control testing: If your API supports multiple versions, test compatibility between different versions to ensure that older versions of the client still work correctly.

Document validation: Ensure that API documentation behaves consistently with the actual API. This includes verifying that the request and response are as described in the documentation.

Privacy testing: Ensure that APIs follow privacy regulations when handling sensitive data and do not disclose the privacy information of patients or users.

Anomaly testing: Test the behavior of your API under unusual conditions, such as network failures, database failures, or other error conditions.

Cross-platform testing: Test API compatibility across different operating systems, browsers, and devices to ensure broad client support.

Regression testing: After making changes or fixes, run regression tests to ensure that new code does not introduce new issues or break existing functionality.

User Acceptance Testing (UAT): Hand an API to the end user or a team on behalf of the user to have them verify that the API meets their needs and expectations.

Performance monitoring: Set up performance monitoring and logging in production to monitor how your API is running at any time.

Testing methods can include manual testing, automated testing, and the use of third-party testing services, depending on the size and requirements of the project. Testing is a critical step in ensuring API quality and reliability and should not be overlooked.

1.6.2. Documentation and support

API documentation design:

Clear entry points: Define the entry points (endpoints) of the API, stating the purpose and function of each endpoint. Provide a concise URL and description of the HTTP method for each endpoint.

Request and response examples: Sample requests and responses are provided for each endpoint so that developers understand how to structure requests and process responses.

Parameter description: For each endpoint, detail the supported parameters, including path parameters, query parameters, request headers, and request body

parameters. Provide the data type, optional values, default values, and example values for each parameter.

Error handling: Describes possible error status codes and error messages, and provides workarounds. It is recommended to provide a list of common errors and solutions.

Authentication and authorization: Explains the authentication and authorization mechanisms of the API, including how to obtain access tokens or API keys, and how to use them.

Limits and quotas: If there are access limits or quota limits, clearly describe those limits and how to request higher limits.

Versioning: If we plan to support multiple API versions, please provide information about your versioning strategy so that developers can choose the appropriate version.

Data model and field description: Provide a clear description of the data model and fields returned by the API, including the name, data type, meaning, and example values of each field.

Providing clear, comprehensive API documentation and multiple support channels can help developers use and integrate APIs more easily. Documentation should be easy to understand and contain enough examples and information to answer common questions from developers. Respond to issues and feedback in a timely manner, and actively improve APIs and documentation to improve the user experience.

Regarding support, we can make the following decisions to increase effective use and praise:

Email support is essential, by providing a dedicated email address for developers to send questions, feedback, and request help. Make sure to respond to emails in a timely manner. Add an online community to create an online community or forum where developers can exchange experiences, ask questions, and get answers. It is also possible to set up an FAQ page, by building an FAQ page with FAQs and answers, where developers can find answers without having to wait for a support response. And add live chat support, allowing developers to communicate with the support team in real time. Actively update the documentation regularly, starting with ensuring that the API documentation is always up to date. Then update the documentation whenever the API changes, and send notifications to developers.

Tutorials and samples for users: Tutorials and sample code are provided to help developers get started and use APIs faster. To plan regular webinars or trainings, hold regular webinars or online trainings to help developers gain insight into the advanced features and best practices of the API. With social media, use social

media platforms to stay in touch with developers and share important updates and resources.

1.6.3. Choice of development tools

When developing the ICD-11-API Web, we need to use a range of tools to streamline the development process, improve efficiency, and ensure the quality of the API. You can choose the following steps and methods for development

Programming languages and frameworks: Choose back-end programming languages and frameworks based on our preferences and needs, such as Python (Django or Flask), Node.js (Express.js), Ruby (Ruby on Rails), etc. These tools provide basic web development functionality.

Database Management System: Choose the right database for our data storage needs, such as PostgreSQL, MySQL, MongoDB, etc. The database is used to store and retrieve ICD-11 data.

API frameworks: Use mature API frameworks to simplify API creation and management, such as the Django REST framework (Python), Express.js Middleware (Node.js), Ruby on Rails (Ruby), etc.

Document Generation Tools: Use tools to automatically generate API documentation so that developers can understand the endpoints, parameters, sample requests, and responses of the API. Some commonly used tools include Swagger, OpenAPI, API Blueprint, etc.

Version control system: Use a version control system such as Git to track and manage versions of your code. This helps with multi-person collaboration and maintainability of the code.

Integrated Development Environment (IDE): Use the appropriate IDE to write, debug, and test code. Some popular IDEs include Visual Studio Code, PyCharm, WebStorm, etc., depending on the programming language we choose.

Testing tools: Use unit, integration, and performance testing tools to ensure the stability and usability of your APIs. For example, you can use unittest for Python, Mocha and Chai for Node.js etc.

Security tools: Use tools to scan and detect potential security vulnerabilities in APIs to ensure the security of your data. Some tools include OWASP ZAP, Nessus, and others.

Monitoring tools: Set up a monitoring system to monitor the performance and availability of your APIs in real time. Tools such as Prometheus and Grafana can help us monitor and visualize performance metrics.

Deployment and hosting services: Choose the right cloud hosting platform or server to deploy our APIs. AWS, Azure, Heroku, etc. are some of the commonly used hosting options.

1.7. Programming language

There are many programming languages that can be used to develop the ICD11-API Web, but I think python and Node .js may be the better choice at the moment, and I have summarized a few advantages and disadvantages

Merit:

- Easy to learn and use: Python is considered to be an easy to learn and get started programming language with a clear syntax and a large number of development libraries.

- Lots of libraries and frameworks: Python has a rich ecosystem, including Django and Flask frameworks for web development, as well as libraries for data processing and analysis (e.g. NumPy, Pandas).

- Strong community support: Python has a large global developer community that makes it easy to find resources to solve problems.

-Shortcoming:

- Relatively low performance: Python is generally slower than some compiled languages and is not suitable for some applications that require high performance.

2. Node.js (JavaScript):

-Merit:

- High performance: Node .js runs on non-blocking I/O and is suitable for handling high concurrent requests, so it performs well for real-time applications and APIs.

- Front-end consistency: If we already develop the front-end using JavaScript, using Node.js can achieve front-end consistency.

-Shortcoming:

- Callback hell: The asynchronous programming model can lead to complex nested callbacks, known as "callback hell."

- Not suitable for CPU-intensive tasks: Node .js is suitable for I/O intensive tasks, but may not be the best choice for CPU-intensive tasks.

1.8. Challenges

When developing and using ICD-11 API Web services, may encounter some common issues and challenges:

1.Data acquisition and updating: Access to ICD-11 data may require specific authorization, and data may need to be regularly updated to reflect the latest standards.To ensure that can obtain data lawfully and to establish an update mechanism.

2.Data quality and consistency: Data quality and consistency are critical to healthcare data.Ensure the accuracy of ICD-11 data and deal with possible errors or inconsistencies.

3.Security and privacy: Handling medical data involves sensitive information, so strict security measures must be taken to protect the data. Ensure that data transmission and storage are encrypted and comply with relevant privacy regulations.

4.Performance: Large scale data queries and high concurrent access may affect API performance. Optimize database queries, use techniques such as caching and load balancing, and handle high load situations.

5.Authentication and authorization: Design and implement effective authentication and authorization mechanisms to ensure that only legitimate users can access sensitive medical data.

6.Documentation and user support: Provide clear and detailed API documentation and set up support channels to answer user questions and solve problems.

7.Compliance: Compliance with regulatory and legal requirements, especially when it comes to medical data. Understand international and domestic regulatory requirements for medical data.

8. Monitoring and troubleshooting: Set up a monitoring system to monitor the performance and availability of APIs in real time. Develop a troubleshooting plan for possible issues.

9. Data Update Management: Manage updates to ICD-11 versions and data to ensure APIs are up to date with the latest standards.

10. User feedback and improvement: Collect user feedback, consider user needs, and continuously improve the API to meet user expectations.

11. Caching and data storage: Consider how to efficiently cache data to reduce the burden on the database, and choose the appropriate data storage scheme.

These issues need to be carefully considered during the development, deployment, and maintenance of APIs. Work with relevant medical and legal professionals to ensure that our APIs meet industry standards and regulatory requirements while providing high-quality medical data services.

Conclusion

The development of ICD-11 Web has brought a more convenient, accurate and globally standardized way of disease information management to the medical field, which will help promote the development of medical science and technology, and promote the development of medical practice and health decision-making in a more advanced and collaborative direction.

As a global medical standard, ICD-11 provides a unified disease classification system for the medical field, and the development of its web version will help to promote and apply this standard more widely, ICD-11 Web provides a convenient way for medical professionals to accurately consult and understand information on various diseases, supporting the improvement of medical practice, and providing ICD-11 Web enables medical institutions and health decision-makers to make more informed decisions based on accurate disease data, To promote the optimization of public health and medical policies, the development of ICD-11 Web promotes the analysis and research of health data on a global scale, and provides support for international health cooperation.

ICD-11 Web has been developed with an emphasis on internationalization, making the system easily accessible to users around the world by supporting multi-language and multi-regional needs, overcoming technical challenges in the development process, such as real-time data updates, multi-language support, security, etc., to help improve the reliability and performance of the system.

CHAPTER 2

Requirements for the code generation system and improvements to the web page module

2.1. Web Design Requirements

1. Home/Overview:

Provide a brief introduction and description of the ICD-11 classification system.

An overview of the main functions and uses of the API.

Provide a quick entry or search box for quick access to data.

2. Search function:

Allows users to search for information about ICD-11 by keyword, disease name, or code.

Provide advanced search options, such as filtering, sorting, etc., so that users can accurately find the information they need.

3. Browse by category:

Shows the hierarchical structure of the ICD-11 classification system, enabling users to browse disease information by chapter, block or category.

Provides interactive collapsible/expandable structures to give users a deeper understanding of categories.

4. Details page:

Provide detailed information about each disease or disease group, including coding, definition, clinical description, etc.

Display other information related to the disease, such as clinical guidelines, treatments, etc.

5. API Documentation:

Provide detailed API documentation, including request examples, parameter descriptions, response formats, etc., to help developers use the API properly.

May also include information on authentication and authorization to ensure secure use of the API.

6. Data export/download:

Allows users to export specific ICD-11 data to files (e.g. CSV, JSON, etc.).

Provide download links or API endpoints for easy access to large volumes of data.

7. User authentication and authorization:

If necessary, implement user authentication and authorization mechanisms to ensure that sensitive information is only available to authorized users.

8. Interactive charts and statistics:

Display charts and statistics of ICD-11 data so that users can intuitively understand the incidence, correlation, etc., of different diseases.

9. Comparison of historical versions:

If applicable, provide a historical version comparison function for ICD-11 so that users can understand the changes between different versions.

10. Feedback and support:

Provide a feedback channel for users to report problems and make suggestions.

Provide support information, including FAQs, contact details, etc.

11. Multi-language support:

Support multi-language interface, so that users of different regions and languages can easily use the web page.

12. Mobile adaptation:

Ensure that web pages are mobile-friendly and provide a good mobile user experience.

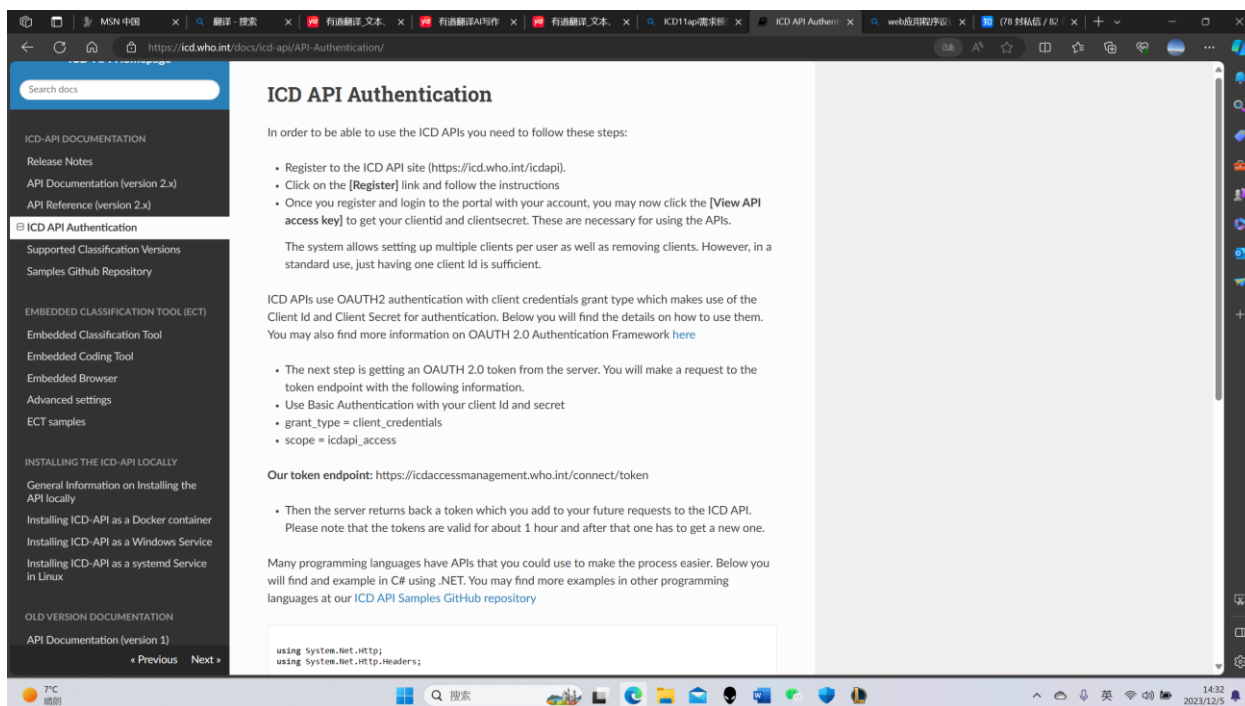


Fig. 2.1. “Detailed introduction and requirements of wto ICD11API ”

2.2. Specific requirements for database in the development of ICD11API

Use an appropriate data model to represent the various concepts and relationships of ICD-11. This may include diseases, symptoms, diagnostic codes, etc., compliance with ICD-11 standards, ensuring that the database structure is compliant to the specification for interaction and integration with ICD-11 data, considering the performance requirements of the database, especially for large-scale queries and data retrieval. The use and optimization of the index is a key aspect in the design, with appropriate security measures in place to ensure the confidentiality and integrity of ICD-11 data. This may include encrypting data, access control, etc., designing interfaces that comply with RESTful or other API design criteria so that applications can communicate effectively with the ICD-11 database, and considering how to handle updates and revisions to the ICD-11 standard to ensure that the database remains up to date.

2.2.1. Capabilities for backend part of web app applications

System requirements:

1. Clearly defined API endpoints, including data retrieval, search, category browsing and other functions.
2. Design RESTful APIs that are easy to understand and use.
3. Provide sufficient API documentation, including request examples, parameter descriptions, response formats, etc.
4. Select an appropriate database, such as a relational database (such as MySQL or PostgreSQL) or a NoSQL database (such as MongoDB), to store ICD-11 data.
5. Establish the corresponding database table or document model according to the structure of ICD-11.
6. Considering the scale of ICD-11 data, optimize the query performance of the database, and use indexes and other means to improve the retrieval efficiency.
7. Consider the cache mechanism to reduce the burden on the server and improve the response speed.
8. Implement authentication and authorization mechanisms to ensure that only authorized users have access to sensitive data.
9. Effectively verify and filter the input data to prevent potential security vulnerabilities.
10. Build scalable systems that can easily add new features or adapt to larger amounts of data later.
11. Consider using a microservice architecture to divide the system into separate services and reduce coupling.

User data requirements:

1. Implement a powerful search engine, allowing users to retrieve ICD-11 data by keyword, coding and other ways.
2. Provide fuzzy search, automatic completion and other functions to enhance user experience.
3. Design classification browsing function with hierarchical structure, so that users can intuitively understand the hierarchical relationship of ICD-11.

4. Provide an interactive interface that allows users to expand/collapse categories.
5. Provide detailed disease information, including coding, definition, clinical description, etc.
6. Display other information related to the disease, such as treatment options, related research, etc.
7. Allow users to export specific ICD-11 data to common formats (CSV, JSON, etc.).
8. Provide download links or API endpoints for easy access to large amounts of data.
9. Provide statistical information and charts to visually present disease incidence, correlation and other data.
10. May also include the option to support user-defined statistics features.
11. Set up feedback channels for users to report problems and make suggestions.
12. Provide support information, including frequently asked questions and contact information..

2.2.2. Capabilities for frontend part of web app applications

User pages:

1. Add interactive elements, such as buttons, drop-down menus, etc., to improve the user experience.
2. Asynchronous loading is implemented to ensure that pages remain responsive when data is loaded.
3. Use appropriate colors, fonts, and ICONS to improve page readability and appeal.
4. Implement a powerful search function that allows users to enter disease names, codes, or keywords.
5. Provide auto-complete recommendations to help users find the information they need faster.
6. When you click on a disease or category, detailed relevant information is displayed, including definitions, symptoms, treatments, and more.

7. Use clear layouts and diagrams to make the information easy to understand.
8. Provides a clear navigation structure that makes it easy for users to find the information they need.
9. Provide help documents or prompts to guide users through the various features of the application.

System main:

1. Create page layouts, including navigation bars, sidebars, and other elements to ensure a user-friendly interface.
2. Design your pages with HTML and CSS, ensuring a responsive design to suit different devices.
3. Integrate ICD11API calls to get disease classifications and related information.
4. Present the obtained data on a page, for example in the form of a list, table, or graph.
5. Provides disease browsing by different categories of ICD11.
6. Use intuitive navigation and expandable menus to give users insight into specific areas.
7. The main language interface is provided to meet the needs of most users.
8. Automatically adjust the content displayed based on the user's language.
9. Make sure the app can be adapted to different devices and screen sizes, including mobile, tablet and desktop devices.
10. Consider possible error scenarios, such as API calls that fail or no data is returned.
11. Provide clear error messages and feedback to help users understand problems and provide solutions.

User side:

1. Consider a user's possible query needs, such as searching for a specific disease, viewing a specific category, or obtaining relevant statistics.
2. Implement a search function that allows users to easily find and access information that interests them.

3. Filtering and sorting options are provided to enable users to customize their displayed data according to specific criteria.
4. Allows users to create personal accounts to save preferences, bookmark important information, or record their search history.
5. Provides setting options, such as theme selection or font size adjustment, to enhance the user experience.
6. Use charts and graphs to present disease statistics such as prevalence, geographic distribution, etc.
7. Provides interactive text descriptions that allow users to customize the data they view.
8. Regularly obtain the latest data from the ICD11API to ensure that the application reflects the latest International Classification of Diseases standards.
9. Provide notifications or update alerts to keep users informed of new or updated content.

2.3. Non-functional optimization of the ICD11API

Non-functional project requirements are those related to the operation and performance of the system, but do not involve specific functionality or specific behavior. While these requirements often focus on the performance, availability, security, maintainability, and so on of the system rather than the specific functionality that the system provides at the user level, these non-functional project requirements are critical to ensuring that the system meets user and business expectations in all aspects. They play a key role in the overall performance and quality of the system.

2.3.1. Requirements for backend part of web app applications

1. Response time: Set a maximum time for page loading and API response to ensure users get prompt feedback.
2. Throughput: Specifies the maximum number of users that the system should support to ensure that the system remains stable as the load increases.

3. Data encryption: The HTTPS protocol is required to encrypt data transmission to protect user privacy.
4. Authentication and authorization: Specify criteria for user access control to ensure that only authorized users have access to sensitive information.
5. Interface responsiveness: Ensure that the website works properly on different devices, including mobile, tablet and desktop devices.
6. User friendliness: Set interface design standards to ensure that users can easily navigate and understand the functionality of the site.
7. Code readability: Emphasize code clarity and comments so that team members can easily maintain and modify the code.
8. Modular design: Adopt modular design principles to make the system easy to expand and update.
9. Translation and localization: Specify multilingual support standards to accommodate global users, especially medical professionals.
10. Test coverage: Set standards for test coverage to ensure system reliability and stability.
11. Automated testing: Emphasis on automated testing to detect potential problems early in the development process.
12. Privacy Policy: Sets out the standards for the processing of user data to ensure compliance with relevant privacy legislation.

2.3.2. An introduction to the non-functional usability of the ICD11API

1. Failover: Implement a failover mechanism to ensure that the system can switch to the standby device in the event of a server or component failure, ensuring continuity.
2. Automatic recovery and automatic saving: Set the automatic recovery and automatic saving mechanism to minimize manual intervention in case of system failure and improve the self-repair ability of the system.
3. Accessibility Design: Follow accessibility design principles to ensure that the website is accessible to users with visual, hearing or other impairments

4. Keyboard navigation: Provides keyboard navigation options to meet the needs of users who use a keyboard instead of a mouse to navigate.
5. Clear labels and guidelines: Provide clear labels and guidelines to enable users to understand the functionality and navigation structure of the website.
6. Information hierarchy: Design a hierarchy of information so that users can drill down to more details.
7. User input validation: Implement front-end validation of user input to reduce the negative impact on users due to incorrect input.
8. Friendly Error messages: Provide friendly and clear error messages to help users understand the problem and provide solutions.
9. Real-time feedback: Provide real-time feedback as the user performs an action, ensuring that the user knows whether their action was successful.
10. Notification Center: Provides a notification center that displays the status of important system updates, events, or user actions.
11. Loading time prompt: Provides a loading time prompt during page loading to let users know the page loading progress.
12. Network Connection status: Provides a reminder of network connection status so that users know the status of their network connection.
13. Smart Search Suggestions: Use smart search suggestions to help users quickly find content they might be interested in.
14. Navigation path: The navigation path of the user's current location is displayed on the page to make it easy for the user to know where they are.

2.4. Detailed Description

2.4.1. Capabilities for backend part of web app applications

System requirements:

1. Design principle: Among many programming language options, Java is chosen as a programming language with cross-platform characteristics. By simply installing the Java Virtual Machine (JVM) on a specific platform, Java applications can be executed on a variety of computers and devices, improving portability. In addition, Java is an object-oriented programming language that supports object-

oriented development principles such as encapsulation, inheritance, and polymorphism. This helps to write code that is modular, maintainable, and extensible. In addition, strict type checking at compile time can reduce runtime errors and improve the stability and reliability of your code. In addition, Java provides an automatic memory management mechanism (garbage collector), which relieves developers of the burden of memory management and avoids the problem of memory leaks. At the same time, extensive and active community support makes it easy for developers to access support, documentation, and tools, and facilitates the development of rich third-party libraries and frameworks to improve development efficiency. In addition, Java also supports multi-threaded application development to achieve concurrent performance; Has a rich standard class library to simplify the basic functions of repeated coding; Ideal for powerful solvers for high applications; Support dynamic loading and dynamic code execution to increase flexibility and scalability; A large ecosystem of tools, integrated development environments (ides), frameworks, and libraries facilitates building applications of all types.

2. Testing: Software testing is essential for users to ensure that the product is correct, verifying code coverage and helping other developers understand possible problems and how to resolve them.

3. Code: In the development process of the back-end part of the web, referring to the mature ICD11 website query mode will shorten the idea and opening time of the software module.

4. Connection: The connection between the front-end part and the back-end part enables the online exchange of important case information required by users.

5.API endpoints: The connection data exchange between the server part and third-party apis is very critical, and third-party products are used to provide link shortening functions.

6. Code generation: Back-end applications need to obtain pathological codes from the front end and convert them to shorter international codes for data storage.

7. Judgment processing: Select and perform operations according to different user identities, such as super administrator, doctor or patient permissions, resulting in differences in page display data and use functions.
8. Data storage: Saving user data is the most important task of software; If not done properly, users will be reluctant to use pages that may cause data loss.
9. Access to the database must capture the entire array in order to retrieve relevant information from the database; The system to store/present information by request needs to be configured to interconnect with doctor-patient interaction/Doctor interaction to quickly update website pages.
10. The application needs to add a certain amount of encoded data on request, as well as a database form buffer to store the existing complete name path, subject classification, etc., and gradually increase the new content over time.
11. If the information is added incorrectly or entered incorrectly or only some fields are updated, the relevant data needs to be changed; If user-related information is involved, it must be possible to modify this data in the background.
12. The ability to view information is reflected both in the web interface and in the user number management section
13. We try to avoid lag when a large number of users are using the application at the same time because we introduce an additional update record table that is relatively immediate but not as frequent because the special parallelism requirements are not met by the slower update method.

User data requirements:

1. Role-Based Authentication: Implement role-based authentication to distinguish between doctors and patients, ensuring that each user has access to the appropriate functionalities based on their role.
2. Data Extraction: Develop algorithms to extract relevant information from user data arrays, allowing for the identification of pathology trends without compromising user rights. For example, analyze frequent pathology codes used by doctors to monitor future trends.

3. **Registration Auditing:** Implement an authentication option for user registration, including an additional verification scheme for users requiring auditing. This enhances the accuracy of user registration and ensures that only authorized individuals gain access.
4. **Identity Verification:** Integrate robust identity verification measures during the registration process to prevent unauthorized access and maintain the integrity of user roles within the system.
5. **Data Encryption:** Utilize strong encryption protocols to protect sensitive pathological data during transmission and storage. This prevents unauthorized access and ensures the confidentiality of user information.
6. **Access Controls:** Implement strict access controls to restrict data access to authorized personnel only. This helps prevent data breaches and unauthorized usage.
7. **Regular Security Audits:** Conduct regular security audits to identify and address potential vulnerabilities in the system. This proactive approach enhances the overall security posture.
8. **Compliance with Privacy Laws:** Ensure compliance with relevant privacy laws and regulations to safeguard user rights. Clearly define and communicate the privacy policy governing the handling of pathological data.
9. **Backup and Recovery:** Establish robust backup and recovery mechanisms to mitigate the risk of data loss. Regularly test these mechanisms to guarantee their effectiveness in case of unforeseen events.
10. By incorporating these measures, the web application can provide a secure and compliant environment for users, protecting their sensitive data while enabling meaningful analysis for pathology trend monitoring.

2.4.2. Capabilities for frontend part of web app applications:

User pages:

1. **Account authentication and Identity selection:** Provides the option to select an identity through account authentication on the registration page, so that the client can log in using an account that has been created.

2. Super administrator page: The super administrator can view all user information, including identity selection and authentication status, on the background service page. Present identity verification function to check user information when necessary or to conduct future statistics and registration of user information.
3. Login and Registration Panel: The main page of the WEB application can be accessed only after the user has logged in, otherwise it should be redirected to the registration panel. Limit the ability of unknown users to interact with the product and ensure that only registered users can use the WEB.
4. Selection panel and diagnosis code: The main page provides a selection panel for the user (doctor) to select the condition he has diagnosed. Combined with the information module, users can quickly select coded information and learn more about how to register coded information.
5. Send codes to the database API: A button is provided on the home page that allows users to send pathology codes to the database API. Add additional supplementary codes to record additional information about the pathology code.
6. User deletion operations: Allow deletion operations to be performed, but only for exceptional reasons, and ensure that there is a strict review process. Avoid improper operations and the risk of obtaining profit or confidential information.
7. Correct user information: If the user enters incorrect account information, provide a correction mechanism for the user to correct the true information. Handles cases where registration information has expired or changed to ensure that the user's pathology code is kept up to date.
8. Administrator Add users: Allows users to add by contacting the application administrator, so that users can still use the system even if they cannot create an account themselves.

System main:

1. Since the application does not require a complex architecture, it is recommended to use the simplest programming language, such as Javascript, to implement the website quickly and efficiently.
2. Combine the framework with the Javascript programming language to create dynamic and extensible web pages that are easier to further develop and improve after the initial release.
3. Leverage the AI that generates code to accelerate the front-end development of the application, reducing the effort of developing the complete architecture from scratch, allowing developers to incrementally supplement the parts that generate pathology coding.
4. The user will be required to have an underlying pathology code, which will be converted to their name version.
5. At the same time, codes that can be locked in the same way as the underlying coding are retained to provide predictive and analytical analysis of the underlying coding usage.

User side:

1. Create a user-friendly interface that allows users to quickly browse and select various parts of the application and get the necessary information through simple interactions.
2. Implement a user authentication system to ensure that users can accurately log in to their accounts in order to record and interact with their own pathology codes.
3. Design different access levels for different types of users, provide different permissions, distinguish access options, and ensure the security of the system.
4. The data is displayed in a structured manner, such as through longitudinal complexity selection of conditions and coding of condition determination, helping users to clearly understand the history of the system and providing a comfortable interactive experience.

5. Allows (doctor) users to modify and update data, ensuring that doctor users do not experience errors or reprocessing when performing operations in the system, improving the user experience.
6. Providing the web with a clear description of the pathology code selection, or showing pages that are loading, gives registered users an understanding that the system needs more time to perform a large number of requests so that the work continues to perform correctly. Make sure registered users have clear guidance when they encounter errors so they can operate correctly.

2.4.3. Requirements for backend part of web app applications

1. Ensure the ICD11API is responsive with efficient database and application caching. Through rational configuration and connection of program modules, rapid response to user requests is realized, so that users can quickly browse and obtain necessary information in various parts of the application.
2. For the scalability of the ICD11API, implement policies to migrate the system to a more efficient server or address possible architectural issues, taking into account possible performance degradation. By distributing some parts of the system on different servers, you can cope with the heavy load that WEB pages can face.
3. Establish a support mechanism for the ICD11API to ensure that the system can recover quickly in the event of an emergency such as a server restart. The reliability of the system is critical, as any data error or loss can have a significant impact on the accuracy and timeliness of the user's pathologically encoded medical information.
4. The architecture of the ICD11API is implemented to support the flexibility of easy configuration or addition of software modules. This allows the system to be modified when the ICD-11 standard is updated or other requirements change, ensuring long-term maintainability.
5. Security is emphasized in the software module architecture of the ICD11API, measures are taken to protect against potential hacking attacks

and protect the confidentiality of users' medical data. Ensure systems comply with relevant privacy and security standards.

6. For the ICD11API, Internet access is critical so that medical professionals can have fast, secure access through the API. Ensure that the API's Web functionality is complete so that users can effectively use and transform pathological codes for information interaction.

2.4.4. Requirements for frontend part of web app applications

1. Performance is one of the key considerations in the development of the ICD11API. Architectural requirements require ensuring that the system is implemented correctly so that the application can interact error-free between its software parts. To achieve this, we first need to optimize data interaction and use efficient algorithms. In the context of the ICD-11 standard, this involves the rapid processing of medical coding and routing data to ensure that users can quickly navigate through the system and access critical information. By establishing an efficient database structure and application cache, the ICD11API provides fast response times, ensuring that healthcare professionals can use the system efficiently.
2. The architecture of the ICD11API not only incorporates best build principles and rules, but also ensures that the system is well scalable. This means that the system has the flexibility to support, change, and remove parts of the software without introducing problems. During the evolution of the ICD-11 standard, the system needed to be able to adapt to the introduction of new coding and routing standards without compromising its performance. By adopting a flexible architecture and technologies such as cloud computing, the ICD11API ensures that the system can easily scale to respond to changing environments as the demands of the healthcare sector grow.
3. The ICD11API is dedicated to providing users with 24/7 reliability. Medical professionals may need access to coding and routing information at any time, so the system must be highly available. By implementing redundancy and failback mechanisms, the ICD11API ensures that the system can recover

quickly and reliably in the event of an emergency server restart or other abnormal situations. This high level of reliability is critical to the real-time needs of the healthcare industry, ensuring that users can always rely on the system.

4. The ICD11API has been developed with a focus on maintainability, especially when it comes to updated versions. By configuring and using the version control system, the ICD11API is able to quickly switch between old and new software module versions and efficiently send the new version of the system to the server. This maintainability ensures that when the ICD-11 standard changes or new medical perceptions emerge, the system can be updated to maintain consistency with the latest standards.
5. The ICD11API provides users with the ability to access different authentication options to suit their needs. In a medical setting, safety is Paramount. The authentication mechanism of the system must be flexible and maintainable to meet user preferences and security needs. This means that the ICD11API can adapt to the different security expectations of healthcare professionals, ensuring that they can access and use the system safely.
6. Since the initial release of the ICD11API uses the system as a simple API, you need to consider the software modules that will have to change as you transition to a full-fledged, full-time application. During this transition, the ICD11API must maintain API compatibility while gradually introducing more mature features to improve system usability. This ensures a smooth transition for healthcare professionals as the system evolves and can always rely on the medical coding and routing information provided by the ICD11API

Interface:

When designing the interface of ICD11API, it is crucial to focus on good usability, flexibility and performance. The interface is the portal of the system, which directly affects the user experience, system scalability and overall performance. To make the system as comfortable as possible, ICD11API needs to maintain communication with users and developers throughout the interface design

and creation process to continuously optimize the interface to adapt to the needs and changing standards of the medical industry. With such a design, ICD11API will become a reliable resource in the field of medical coding and disease classification. Therefore, a simple web application interface needs to be developed, which needs to be followed up :

- 1) Authentication and authorization: Implement a robust authentication and authorization mechanism to ensure that only authorized users can access sensitive information. Standard authentication protocols can be used to provide secure access control. In the ICD11API, medical professionals may need special permissions to access certain coded or classified information, so a well-designed authorization system is critical.
- 2) Query parameters and filtering: Provide flexible query parameters and filtering options to meet the needs of different users. For example, when querying medical codes, the ICD11API can support filtering by disease name, code category, or associated label. This design enables users to pinpoint the information they need.
- 3) Error handling mechanism: Design robust error handling mechanism to provide users with clear error information to help them understand and solve the problem. Good error handling helps to improve the user experience and reduce user confusion when using the API.
- 4) Version control: Implement a version control mechanism to ensure that existing applications can continue to run even after API updates. Include version information in the URI so that users can choose to use a specific version of the API. Version control of the ICD11API is a key factor in ensuring stability and backward compatibility.
- 5) Documentation and API reference: Provide detailed documentation and API reference to help users understand the function and use of the interface. Clear documentation can make it much easier for users to get started and drive wider API adoption.

- 6) Under the main container there should be a dynamic generated list that displays all the converted links that the user sent, and so that the next links sent by him do not erase the previous ones.
- 7) Current limiting mechanism; Implement appropriate current limiting mechanisms to prevent abuse and ensure system stability. By setting request rate limits, the ICD11API can effectively manage high traffic volumes and maintain high performance.

Design:

Web design involves multiple aspects, from user experience to visual presentation, that require careful consideration to ensure that the site is easy to navigate and that users can easily find the information they need. Consider the user's behavior path, simplify the process, improve user satisfaction, and ensure that the website can adapt to different devices and screen sizes to provide a consistent user experience. Choose the right color combination, consistent with the brand. Provide interactive elements such as buttons, forms, micro-interaction effects, etc., to enhance the user experience, ensure that users get clear feedback when interacting with the site, adopt good typography, ensure that text is easy to read, line spacing and font size are appropriate, responsive design is critical for mobile device compatibility, optimize the performance of the site, ensure fast load times. Compress images, use browser cache, and process code properly to improve user experience and search engine rankings.

Create clear, intuitive navigation menus that make it easy for users to find the information they need. Consider using breadcrumb navigation and page links to improve the user's navigation experience. Develop a good content structure, using elements such as headings, paragraphs, and lists to make the content easy to understand and scan. Make sure key information is highlighted, do cross-browser testing, make sure the site works well in all major browsers (Chrome, Firefox, Safari, Edge, etc.), and optimize the site to improve its ranking in search engines

Conclusion

In this section, I highlight the key design requirements for the web application I want to develop. I have selected and investigated all aspects of the WEB in advance, divided the main part of the program into WEB page modules, and the display of the final product will be released at the back.

Through the study of materials, I made it clear that WEB page development is a complex task involving design, coding and implementation. When conducting the analysis, we need to consider several aspects, including user experience, functional requirements, technology selection, and so on. In terms of user experience, a successful WEB page should be able to provide a concise interface design, and have a good response speed. Through reasonable layout and intuitive operation, users can easily find the required information and complete various interactive actions. We also need to pay attention to page loading time and compatibility issues to ensure that different devices and browsers can be displayed normally. In terms of functional requirements, we need to determine the functional modules needed to be implemented according to the project requirements, and carry out detailed planning and design.

This includes front-end page presentation, back-end data processing, and interaction with the database. At the same time, in the development process to pay attention to code quality and maintainability, the use of appropriate frameworks or libraries to improve efficiency and reduce the probability of error, in the analysis of WEB page development, we need to fully consider the user experience, functional requirements and technology selection and other factors, and flexible use of relevant knowledge and tools to achieve the expected goals. Only through in-depth analysis and comprehensive thinking can we provide customers with high-quality products that meet their expectations.

CHAPTER 3

Database annotation of WEB system for ICD11API

3.1. Database selection and structure of the API system of ICD11

In the database of the WEB end system of ICD11API, my idea was to store the gender information of the patient, the time information, and the information of accessing the system separately. Later, I found that there were not so many separate data tables in the development, so I inserted these information into the large form of the established database. As a sub-table header for data storage, because the development of the database is based on the premise of network applications, so the production will be slightly more detailed. This method can reduce the time to build a large database form and the follow-up work, can work to a certain extent -- also relative to the same type of data can be stored together, it would be easier to when looking for a little.

Here I show my database form. The blue database stores the main data content, which is also the most critical and core data in this project. Since I downloaded it from China, there will be some Chinese characters. The core data here are: The patient's ICD11 coded record, that is, the patient's ICD11 coded record, is public to the patient and the doctor, but private to other patients and other users, and the patient can save the form of this code to let the doctor record the development of the disease, so as to better track the pathology in the first time. So it's a very critical piece of data; Then is the expansion of ICD11 extra code, the main role of extra code is ICD-11 used to extend the coding, can also be understood as a part of ICD-10 extended coding. ICD-11 added the additional code function. The supplementary codes cover several dimensions such as severity, time, anatomy, description of diagnostic codes, etc. Here I specially made a table to store the supplementary codes. The ICD11 backbone code is used to indicate a patient's primary health condition and is a code that can be used alone in a specific linear combination. The design of the backbone code is to ensure that the most meaningful minimum information can be obtained from each medical record when only one code is needed. I also made a separate table to record the storage of the ICD11 backbone code, but the supplementary code and the backbone code are different database tables, because in my idea, both can be independently expanded, and the supplementary code will be updated more frequently. Convenient back end personnel to schedule data, in the review time will be a little easier.

Then the introduction of the core data is the User's common information, such as the user's registration information, including the registration name, registration identity, registration time, registration gender and registration password, these belong to the basic and public data port content, so I put these data in a whole independent user form, used to represent the user registration needs to use the data. It's a little more detailed.

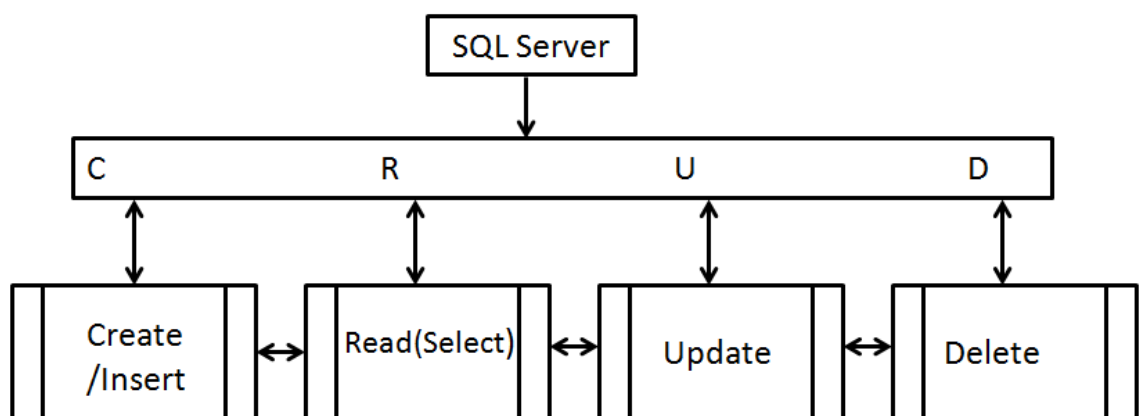
Here is the form shown in my database catalog form, the blue form is the key data form

QRTZ_TRIGGERS	0	16 KB	InnoDB	2023-11-06 11:27:44	utf8mb4_0900_ai...	触发器详细信息表	
sys_case	7	16 KB	InnoDB	2023-11-10 14:04:19	utf8mb4_0900_ai...	病例	
sys_case_icdAdd	6	16 KB	InnoDB	2023-11-10 14:03:12	utf8mb4_0900_ai...		
sys_config	6	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	参数配置表	
sys_dept	10	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	部门表	
sys_dict_data	29	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	字典数据表	
sys_dict_type	10	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	字典类型表	
sys_icd	827	176 KB	InnoDB	2023-11-06 14:06:45	utf8mb4_0900_ai...		
sys_icd_add	135	48 KB	InnoDB	2023-11-06 14:06:45	utf8mb4_0900_ai...		
sys_job	3	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	定时任务调度表	
sys_job_log	0	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	定时任务调度日志表	
sys_logininfor	60	16 KB	InnoDB	2023-11-06 11:27:34	2023-12-07 14:25:20	utf8mb4_0900_ai...	系统访问记录
sys_menu	85	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	菜单权限表	
sys_notice	2	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	通知公告表	
sys_oper_log	111	112 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	操作日志记录	
sys_post	4	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	岗位信息表	
sys_role	4	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	角色信息表	
sys_role_dept	3	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	角色和部门关联表	
sys_role_menu	95	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	角色和菜单关联表	
sys_user	12	16 KB	InnoDB	2023-11-06 13:24:29	2023-12-07 14:25:20	utf8mb4_0900_ai...	用户信息表
sys_user_post	2	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	用户与岗位关联表	
sys_user_role	8	16 KB	InnoDB	2023-11-06 11:27:34	utf8mb4_0900_ai...	用户和角色关联表	

Key Data form

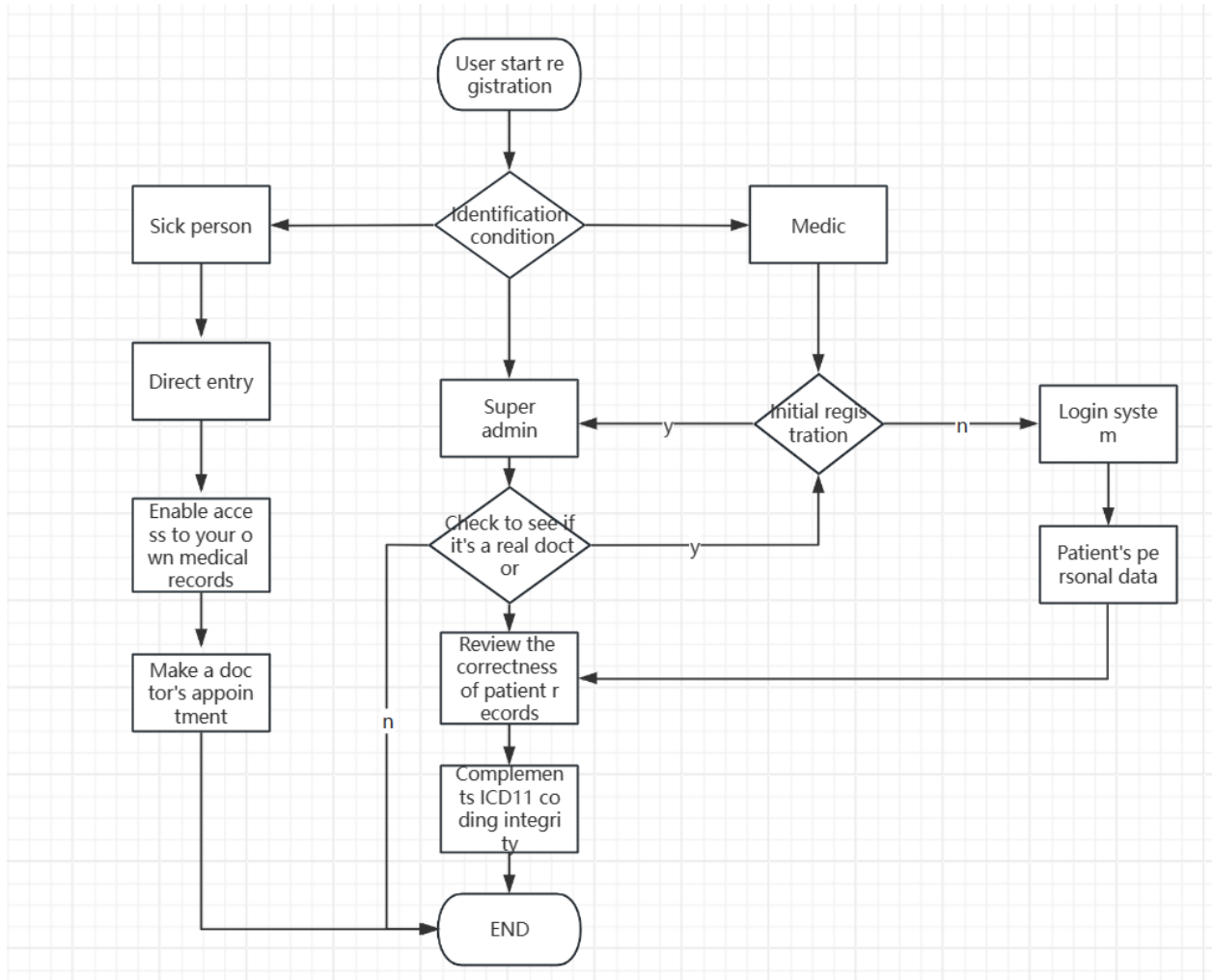
In this project, I mainly borrowed four functions of the database CRUD, which are relatively basic but very important functions of the database. Here, I borrowed the main functions of the database to expand, so as to improve my own functions. CRUD is often used for anything related to the database and database design. Many contents of my project benefit from the inspiration and help of CRUD. In addition to the basic function I use in the operation of the backbone code and additional code of ICD11API, CRUD is also important for the final registered users. Without it, it will be a huge workload for users to query personal information and modify medical information. Most of the ones I've used in this WEB project allow administrators in the project to add or create new entries, search for existing entries, make changes to them, or delete them.

The following is a detailed illustration of this function



C.R.U.D.

3.2. Flow chart of registration and cooperation of users with different identities



Flowchart of registration and work

This figure is to express the work undertaken by users with different identities in this project. I firmly believe that the improvement of software is the communication and interaction between users and developers, and mutual learning and mutual help can make a software gradually flawless. Therefore, users are also the perfect way that I must consider in the design. Their permissions are also different, among which the purest users are patients, because they only have the basic functions of viewing, querying and registration, do not have the ability to modify the case or improve the ICD11 code library, and do not have the permission to modify their own case code.

The second identity in this figure is the doctor. The special difference between the doctor and the patient is that the doctor needs the administrator to review his identity. Because the doctor has the right to prescribe and prescribe drugs, he cannot directly register and use his account, and the administrator must open his

authority in the background to login. In addition, after landing, they should participate in the entry of ICD11 code. In the process of ICD11 coding, they should be familiar with the condition of patients and increase their understanding and cognition of ICD11. In this way, doctors can also expand and learn in their professional fields, further extend their understanding of cases, and facilitate the internationalization of medical treatment. Promote the development of medical technology, strengthen the pace of medical internationalization, so that medical code can help patients in other countries to better and more directly understand their condition.

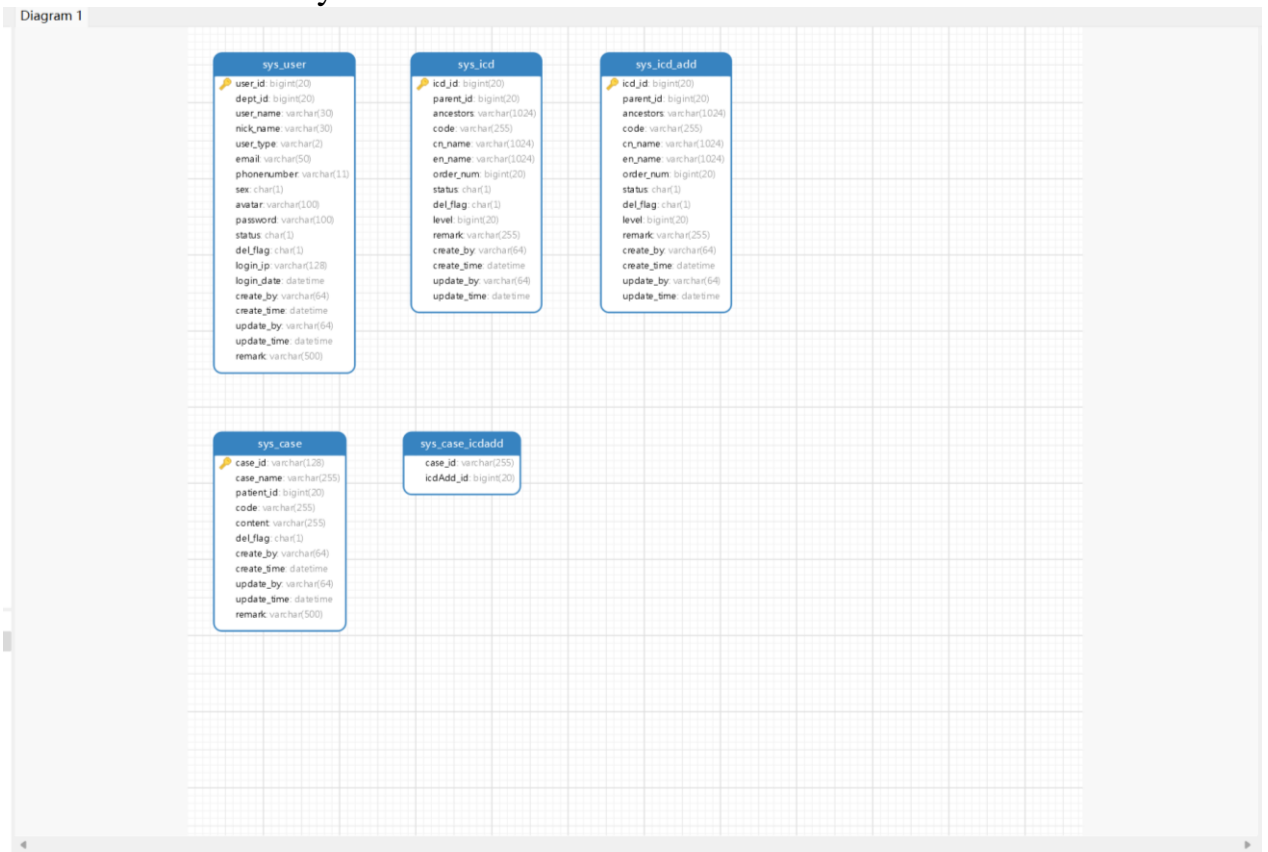
The last and most important identity is the super administrator, the reason why he is called the super administrator is because it has all the permissions, including the review of the conditions of registered users of the above two identities, including the detailed description and entry of cases, and the administrator will always monitor and manage the data, but also to ensure the security of the data. Therefore, the identity of the super administrator can not be registered directly, and the account needs to be issued after the background audit, and the account of the super administrator can not select the account, only after the background editing is sent, in this system by the highest detection and management of the account is the super administrator account. Because of the particularity of its account identity, it is the most critical existence in the entire account and the most core user demand record in the entire system.

It is easy to see from the above introduction that the design I made is basically carried out around users. Administrators and actual users work together to improve the ICD11API web page system, so that the system can better learn what each user needs in the evolution of the system and what the specific needs are. According to this system, patients can well protect their own information and do not want anyone other than doctors to know their case information. Moreover, because the cases are digitized and the ICD11 internationally recognized code is used, I believe that patients will have better records and more detailed records of their cases, and doctors will have more communication platforms.

3.3. ICD11API WEB side background database overview and database key data components schematic

In the component diagram, you can see more detailed statistical relationships between databases (databases of critical data), as well as the types of data that databases of critical data store in the system and the names of the data they store. Through the content components displayed in the database shown in this figure, the storage location of each data can be identified from the entire system, and the linear relationship between each data can be roughly estimated. The key information of each data can be intuitively seen in the first time to grasp the detailed statistical content of each data. If it can be commercialized in the later stage of the system, we can try to add some data protection measures here, so as to protect the key information and key data content in the database. The figure shows

the core data graph in the system, and you can see the specific content of the core data and how each layer is related to each other.



Core data graph

3.4. ICD11API WEB side key entity class diagram

Class diagram is a kind of structural diagram in Unified Modeling Language (UML), which is used to describe the classes in a system and the relationships between them. Class diagram is a powerful visualization tool that can graphically present complex system designs. With graphical representations, developers can more easily understand the structure, relationships, and design decisions of the system, and class diagrams are also a modeling tool through which abstract models of the system can be created. This model not only has a guiding role for developers, but also can be used as a tool to document and communicate with themselves, helping them better understand the WEB project and develop the WEB project.

The main elements in a class diagram are classes, which represent abstract entities in the system. A class is a way of abstracting a group of objects that have similar characteristics and behavior. Abstractness means that class diagrams are concerned with concepts and models, while concreteness is reflected in the fact that these abstract concepts correspond to concrete objects in a real system. In the design of this WEB, the balance between abstractness and materiality is very critical. Abstractness helped me understand the nature of the WEB problem I was

developing, while materiality ensured that the model corresponded to the real system. Through class abstraction, more general and extensible system models can be built, while materiality ensures that these models have practical meaning in practical development.

- Use data received during the registration phase or format standard data fields;
- Changing field contents or even attributes by modifying accepted data;
- This method is used to obtain certain key fields.

There is also polymorphism, which means that an object can exhibit multiple forms. In the class diagram I use, polymorphism is usually implemented through method overloading and overriding. Overloading allows a class to have multiple methods with the same name but different argument lists, while overloading allows subclasses to reimplement methods inherited from the parent class.

The advantage of polymorphism here is also to increase the flexibility and scalability of the system. Through polymorphism, objects of different kinds can be handled in a unified way, and the specific subtypes can be ignored in the process of processing. This makes it easier for the system to adapt to changes and new features. All service classes are divided into interfaces (basic representations without adding logic) and their implementations (complete descriptions of internal mechanisms) to facilitate the operation of the WEB system.

The User class is responsible for allowing users to select their own specific identity in the web application and then waiting for further authentication by the super administrator (who cannot be registered by the user alone). Each user has the right to choose their own login and password. In addition, each user is assigned unique data, including first name, last name, gender, age account name and password, in the extended use we will use a number of measures to ensure the security of user information due to the sensitive and private nature of patient information.

The Case class is responsible for communicating with patients and doctors. This entity class was created because patients need to provide medical records for doctors to write prescriptions, which can also be divided into: The backbone code of ICD11 and the additional code of ICD11, here also schedule the creation, modification and deletion conditions of user cases, here you can see that each patient will get its corresponding soldier line data and ICD11 complete code.

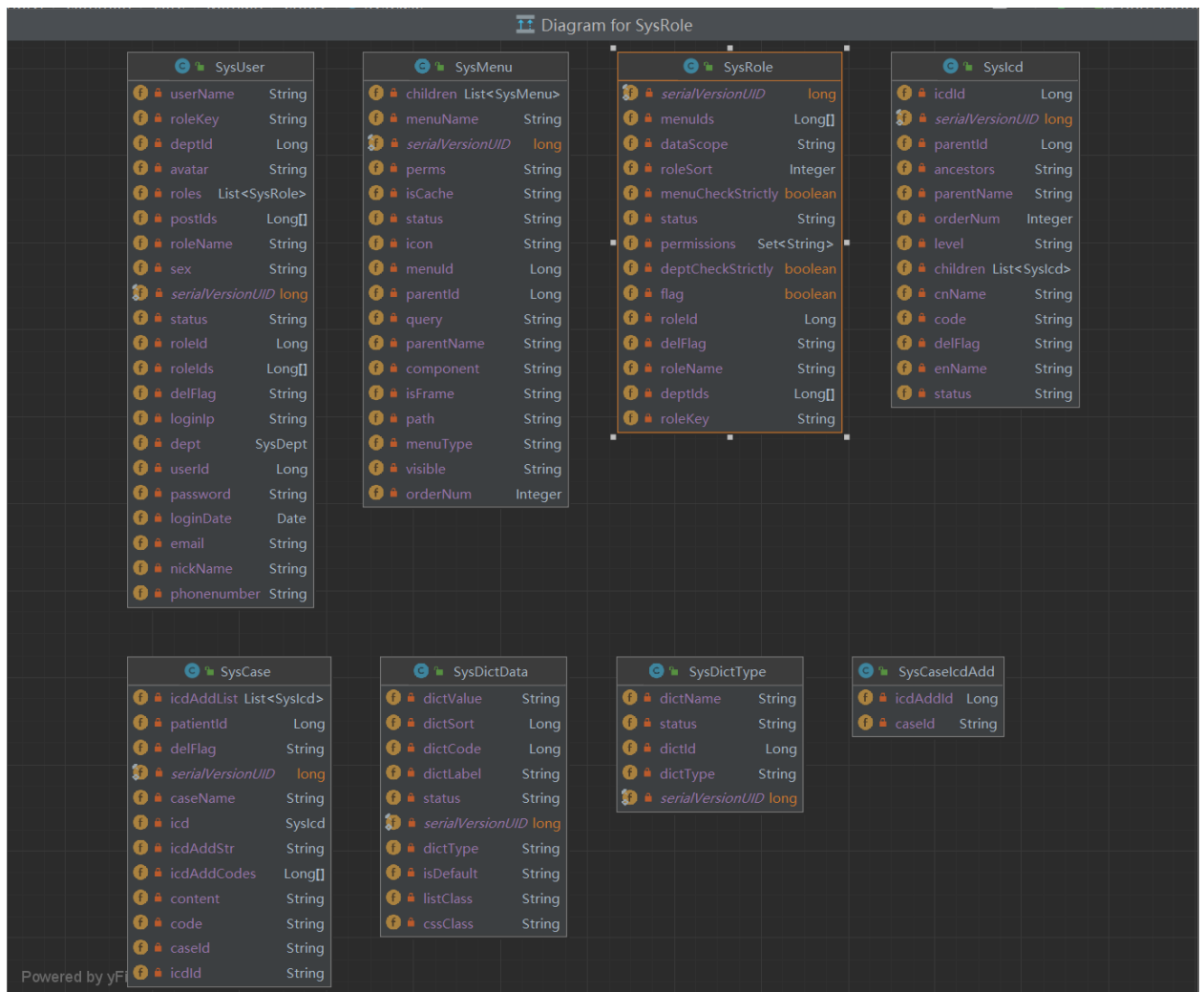
The ICD class is the class responsible for recording the details of the ICD11 backbone code statistically, including but limited to the following situations:

1. Super administrator and doctor users need to edit the ICD11 trunk code
2. The super administrator and the doctor need to synchronize the update of the ICD11 backbone code with the information on the WEB

3. Super administrator and doctor identity users need to modify and adjust the ICD11 backbone code (mainly delete function), because the ICD11 code may appear in the input code error may occur, so here we need to set up such input error content and retrograde change

4. Users with super administrator and doctor status need to edit the ICD11 backbone code5. Information on the registration of the modifier (function as originally envisaged)

In the entity class of ICDADD I mainly implement a total of um that is ICD11 with extra code to expand the function, this function is relatively simple.



Class diagram

3.5. Establish a physical layer connection

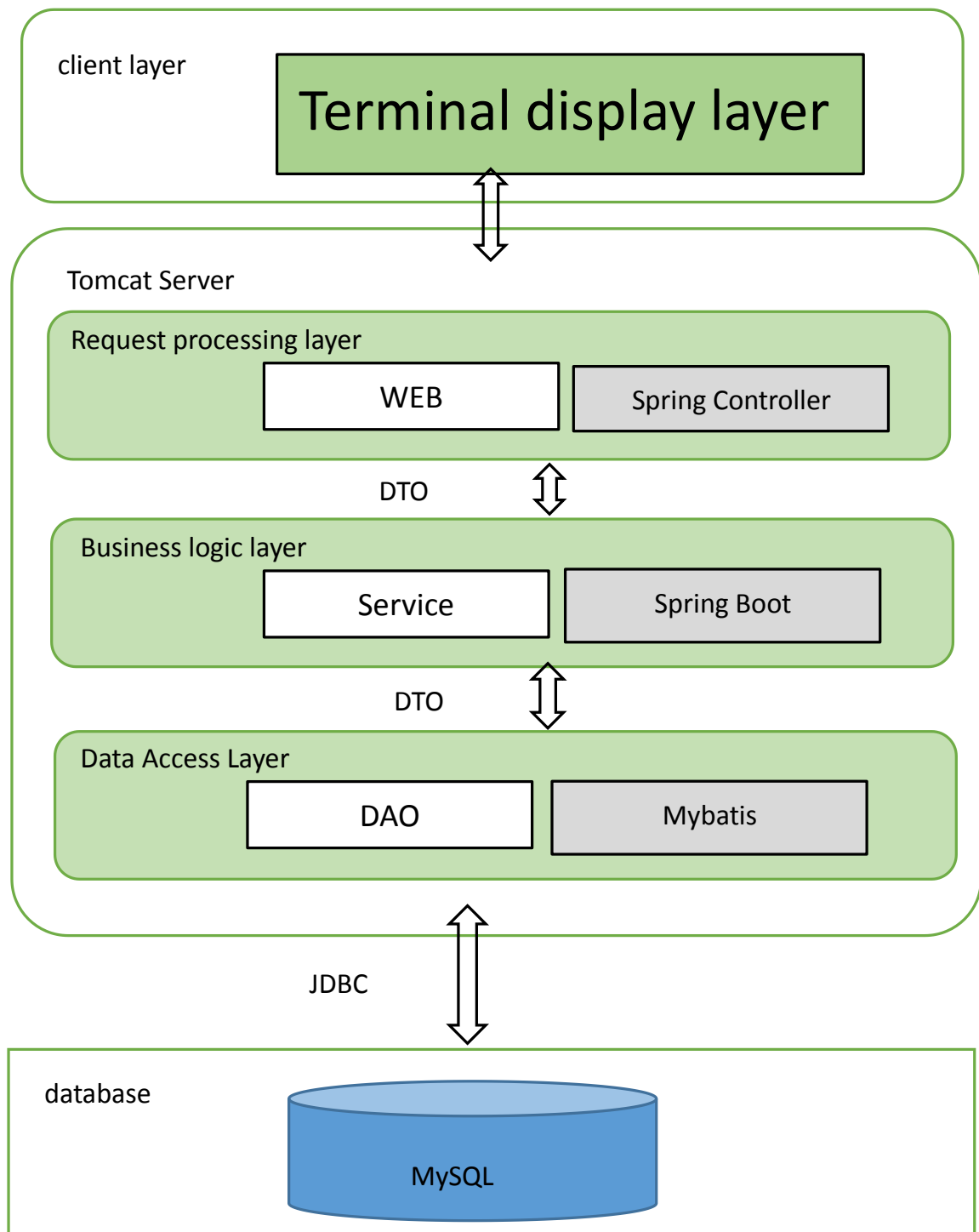
In this ICD11API service, I decompose the physical layer into three levels of physical hardware synthesis, from the top down are: The user layer, Tomcat Server

and database (in fact, if there is a fourth layer, the fourth layer can be divided into the operating system level, but I did not choose to use another operating system in this project), the user layer can be commonly understood as the display layer, which is the most intuitive feeling that can be displayed to users. In addition, most of the functions can be added, deleted and modified directly through the buttons on the web side, which aims to reduce personnel training. Users (doctors and super administrators) can modify directly on the web side, and there is no need to go to the background to increase the content of the database by moving code. I believe this is very useful for non-programmers.

Then there is the module of TomCat Server, which is divided into three parts according to the logical way of reference. For example, I will first collect the requirements of users and the functions that users want to achieve, and then enter the first layer with these requirements, which I call the request processing layer. At this level, our main purpose is not to realize the needs of users, but to collect statistics and delegate the needs of users. At this time, we enter the second layer of services, that is, the business logic layer. In this layer, we will do a similar way to jump to show the ideas of users, but only send the ideas of users to the next layer like the post office. In fact, the business logic layer actually plays a more role in the user's ideas into the system for logical differentiation, in order to better reference the next layer of functions, that is, the final execution level, I named it the data access layer, in this layer the system will issue an application to view the database command, in fact, here is the real execution of the command level. It is also the beginning of matching the system with the user's idea and the user's operation. In this set of three steps, the complete walk down is actually the complete face of TomCat Server service, so that the user's needs are the formal beginning.

The last part is the database. After the complete set of TomCat Server operations, the system will go to the database to access the required data or store the user's data to complete the detailed operation of the database. Originally, it should enter the database for data storage, modification and deletion. However, because I added visual data operation, this step became easier and our learning cost was reduced. As people who can modify the database, they are not so familiar with the code. In other words, a simple aspect is opened up, so that people without basic knowledge can make changes directly. In this era of rapid talent circulation, I think it is still necessary to save the cost of talent.

The following figure is the detailed physical layer specific contact policy I made, and it is possible to see my idea here more clearly in the form of pictures.



Detailed physical layer specific contact policy

conclusion

conclusion

In this module, I introduced the general structure of the WEB program system based on the development of ICD11API in detail; Here I also put forward my understanding of this project, and pointed out the main aspects that need special attention and improvement in the development process; From a very important point of view, I reviewed the part of the system that I designed and how it would continue to work later; I also specially show the key data in the system database

and the important and detailed class diagram in the system; First, it shared the progressive relationship between each layer of the physical layer, and then explained how to advance between each layer of the physical layer, the working principle and working mode of the physical layer. Finally, it introduced the construction of physical hardware connection in detail and concretely, and put forward my improved part in this layer.

CHAPTER 4

ICD11 API WEB page system prototype display

4.1. The development of ICD11 coding API display system is briefly introduced

Develop a clear project plan, including goals, tasks, timelines, and resource allocations. Make sure the team understands and is committed to achieving these goals. Effective learning is the key to successful thinking about project management. Think clearly and in a timely manner to ensure that information is communicated correctly, identify potential risks in the project, and develop a risk management plan accordingly. Resolve issues in a timely manner to ensure that the project is progressing smoothly, address changes in the project, and ensure that these changes do not negatively impact the project objectives. Flexibly respond to changes, adjust plans, focus on the quality of project development, and ensure that expected standards are met. Establish appropriate quality control and quality assurance mechanisms. Review the experience of the page regularly, draw lessons learned, and identify opportunities for improvement. Continue to learn and update their project management knowledge

In the SSM framework of my choice, I usually refer to a combination of Spring + Spring MVC + MyBatis. In this framework, I have selected various components that are common. The Controller is a core component in the Spring MVC framework that processes user requests and returns responses. It usually contains the method for handling the request and is responsible for forwarding the request to the appropriate Service layer for processing, and then returning the processing result to the front-end page. Service is the business logic layer of my choice, which contains the core business logic of the application and is responsible for handling the business-related logic and processes. Services are typically designed as interfaces that define the methods of the business logic. The Service implementation class is responsible for concretely implementing the business logic defined by the Service interface. In the SSM framework, the Service interface and its implementation class are generally separated to decouple the business logic from the invocation relationship. Mapper is an important interface used to perform database operations in MyBatis framework, which defines the required methods to interact with the database, and does not contain specific SQL statements. Mapper XML file is used as MyBatis to store SQL mapping statements and database interaction configuration information, including contents related to database interaction such as SQL statements, parameter mappings, result mappings, and so on. These XML files are used to establish mappings between Mapper interfaces and database operations. These components work together to form the core of the SSM framework and effectively support Web application functionality: Controller is responsible for receiving and processing user requests; Service carries the core

business logic. Mapper and its corresponding XML files focus on efficient interaction with the database.

4.2. ICD11API WEB side results display and source code display

This part of the content is to show that I have implemented the module of adding, deleting and correcting functions, which is composed of four parts in general, and my explanation is also carried out in this order:

1. Add, modify and delete ICD main code;
2. Adding, modifying and deleting additional ICD codes;
3. Add, modify and delete registered users;
4. Addition, modification and deletion of patient's medical information.

The above are the specific functions I realized in this program development, and I refer to some fresh technologies and achievements in the market now. The modification and deletion functions in all project implementation need to be reviewed before deletion, so as to ensure that no accidental deletion will occur during deletion, so as to protect the integrity of data

Source Code Inventory 1

Class ICD-create.java

```
1. @PostMapping
2.     public AjaxResult add(@Validated @RequestBody SysIcd icd)
3.     {
4.         if (!icdService.checkCodeUnique(icd))
5.         {
6.             return error("新增 Icd'" + icd.getCode() + "'失败, Code 已
存在");
7.         }
8.         icd.setCreateBy(getUsername());
9.         return toAjax(icdService.insertIcd(icd));
10. }
11.
12. public int insertIcd(SysIcd icd);
13.
14. @Override
15.     public int insertIcd(SysIcd icd)
16.     {
17.         SysIcd info = icdMapper.selectIcdById(icd.getParentId());
18.         // 如果父节点不为正常状态,则不允许新增子节点
19.         if (!UserConstants.DEPT_NORMAL.equals(info.getStatus()))
20.         {
21.             throw new ServiceException("Icd 停用, 不允许新增");
22.         }
23.         icd.setAncestors(info.getAncestors() + "," + icd.getParentId
());
24.         return icdMapper.insertIcd(icd);
25.     }
26. public int insertIcd(SysIcd icd);
27.     <insert id="insertIcd" parameterType="SysIcd">
```

```

28.      insert into sys_icd(
29.          <if test="icdId != null and icdId != 0">icd_id,</if>
30.          <if test="parentId != null and parentId != 0">parent_id,
    </if>
31.          <if test="cnName != null and cnName != ''">cn_name,</if>
32.          <if test="enName != null and enName != ''">en_name,</if>
33.          <if test="code != null and code != ''">code,</if>
34.          <if test="ancestors != null and ancestors != ''">ancesto
rs,</if>
35.          <if test="orderNum != null">order_num,</if>
36.          <if test="status != null">status,</if>
37.          <if test="level != null">level,</if>
38.          <if test="createBy != null and createBy != ''">create_by
    ,</if>
39.          create_time
40.      )values(
41.          <if test="icdId != null and icdId != 0">#{icdId},</if>
42.          <if test="parentId != null and parentId != 0">#{parentId
    },</if>
43.          <if test="cnName != null and cnName != ''">#{cnName},</i
    f>
44.          <if test="enName != null and enName != ''">#{enName},</i
    f>
45.          <if test="code != null and code != ''">#{code},</if>
46.          <if test="ancestors != null and ancestors != ''">#{ances
    tors},</if>
47.          <if test="orderNum != null">#{orderNum},</if>
48.          <if test="status != null">#{status},</if>
49.          <if test="level != null">#{level},</if>
50.          <if test="createBy != null and createBy != ''">#{createB
    y},</if>
51.          sysdate()
52.      )
53.  </insert>

```

Under this code is mainly to do the ICD main code input code, when the super administrator starts the system and starts to create a new main code is to use this way to input, the scheduling content is the same.

Source Code Inventory 2

Class ICD-update.java

```

1.  @PostMapping
2.  public AjaxResult edit(@Validated @RequestBody SysIcd icd)
3.  {
4.      Long icdId = icd.getIcdId();
5.      if (!icdService.checkCodeUnique(icd))
6.      {
7.          return error("修改 Icd'" + icd.getCode() + "'失败, Code 已
    存在");
8.      }

```

```

9.         else if (icd.getParentId().equals(icdId))
10.        {
11.            return error("修改 Icd'" + icd.getCnName() + "'失败, 上级
Icd 不能是自己");
12.        }
13.        else if (StringUtils.equals(UserConstants.DEPT_DISABLE, icd.
getStatus()) && icdService.selectNormalChildrenIcdById(icdId) > 0)
14.        {
15.            return error("该 Icd 包含未停用的子 Icd! ");
16.        }
17.        icd.setUpdateBy(getUsername());
18.        return toAjax(icdService.updateIcd(icd));
19.    }
20. public int updateIcd(SysIcd icd);
21. @Override
22.    public int updateIcd(SysIcd icd)
23.    {
24.        SysIcd newParentIcd = icdMapper.selectIcdById(icd.getParentI
d());
25.        SysIcd oldIcd = icdMapper.selectIcdById(icd.getIcdId());
26.        if (StringUtils.isNotEmpty(newParentIcd) && StringUtils.isNot
Null(oldIcd))
27.        {
28.            String newAncestors = newParentIcd.getAncestors() + ","
+ newParentIcd.getIcdId();
29.            String oldAncestors = oldIcd.getAncestors();
30.            icd.setAncestors(newAncestors);
31.            updateIcdChildren(icd.getIcdId(), newAncestors, oldAnces
tors);
32.        }
33.        int result = icdMapper.updateIcd(icd);
34.        if (UserConstants.DEPT_NORMAL.equals(icd.getStatus()) && Str
ingUtils.isNotEmpty(icd.getAncestors())
35.            && !StringUtils.equals("0", icd.getAncestors()))
36.        {
37.            // 如果该 Icd 是启用状态, 则启用该 Icd 的所有上级 Icd
38.            updateParentIcdStatusNormal(icd);
39.        }
40.        return result;
41.    }
42. public int updateIcd(SysIcd icd);
43.    <update id="updateIcd" parameterType="SysIcd">
44.        update sys_icd
45.        <set>
46.            <if test="parentId != null and parentId != 0">parent_id
= #{parentId},</if>
47.            <if test="cnName != null and cnName != ''">cn_name = #{c
nName},</if>
48.            <if test="enName != null and enName != ''">en_name = #{e
nName},</if>
49.            <if test="ancestors != null and ancestors != ''">ancesto
rs = #{ancestors},</if>
50.            <if test="code != null and code != ''">code = #{code},</
if>
51.            <if test="orderNum != null">order_num = #{orderNum},</if
>
52.            <if test="status != null and status != ''">status = #{st
atus},</if>

```



```

53.         <if test="level != null and level != ''">level = #{level
    },</if>
54.         <if test="updateBy != null and updateBy != ''">update_by
    = #{updateBy},</if>
55.         update_time = sysdate()
56.     </set>
57.     where icd_id = #{icdId}
58. </update>

```

In this code is mainly to do the ICD main code input update code, when the identity of the super administrator and doctor start the system and start to create a new main code is to use this way to make new data modification, scheduling content is also the code of this area .

Source Code Inventory 3

Class ICD-delete.java

```

1. @DeleteMapping("/{icdId}")
2. public AjaxResult remove(@PathVariable Long icdId)
3. {
4.     if (icdService.hasChildByIcdId(icdId))
5.     {
6.         return warn("存在下级 Icd,不允许删除");
7.     }
8.     return toAjax(icdService.deleteIcdById(icdId));
9. }
10. public int deleteIcdById(Long icdId);
11. @Override
12. public int deleteIcdById(Long icdId)
13. {
14.     return icdMapper.deleteIcdById(icdId);
15. }
16. public int deleteIcdById(Long icdId);
17. <delete id="deleteIcdById" parameterType="Long">
18.     update sys_icd set del_flag = '2' where icd_id = #{icdId}
19. </delete>

```

In this code is mainly to do the ICD main code deletion code, when the super administrator and the doctor's identity to start the system and start to delete the old main code is to use this way to modify the new data, to do the deletion of the schedule content is also the code of this area.

Source Code Inventory 4

Class ICDADD-create.java

```

1. @PostMapping
2. public AjaxResult add(@Validated @RequestBody SysIcd icdAdd)
3. {
4.     if (!icdAddService.checkCodeUnique(icdAdd))
5.     {

```

```

6.         return error("新增 IcdAdd'" + icdAdd.getCode() + "'失败,
Code 已存在");
7.     }
8.     icdAdd.setCreateBy(getUsername());
9.     return toAjax(icdAddService.insertIcdAdd(icdAdd));
10. }
11. public int insertIcdAdd(SysIcd icd);
12.     @Override
13.     public int insertIcdAdd(SysIcd icdAdd)
14.     {
15.         SysIcd info = icdAddMapper.selectIcdAddById(icdAdd.getParent
Id());
16.         // 如果父节点不为正常状态,则不允许新增子节点
17.         if (!UserConstants.DEPT_NORMAL.equals(info.getStatus()))
18.         {
19.             throw new ServiceException("IcdAdd 停用, 不允许新增");
20.         }
21.         icdAdd.setAncestors(info.getAncestors() + "," + icdAdd.getPa
rentId());
22.         return icdAddMapper.insertIcdAdd(icdAdd);
23. }
24.
25. public int insertIcdAdd(SysIcd icd);
26.     <insert id="insertIcdAdd" parameterType="SysIcd">
27.         insert into sys_icd_add(
28.             <if test="icdId != null and icdId != 0">icd_id,</if>
29.             <if test="parentId != null and parentId != 0">parent_id,
</if>
30.             <if test="cnName != null and cnName != ''">cn_name,</if>
31.             <if test="enName != null and enName != ''">en_name,</if>
32.             <if test="code != null and code != ''">code,</if>
33.             <if test="ancestors != null and ancestors != ''">ancesto
rs,</if>
34.             <if test="orderNum != null">order_num,</if>
35.             <if test="status != null">status,</if>
36.             <if test="level != null">level,</if>
37.             <if test="createBy != null and createBy != ''">create_by
,</if>
38.             create_time
39.         )values(
40.             <if test="icdId != null and icdId != 0">#{icdId},</if>
41.             <if test="parentId != null and parentId != 0">#{parentId
},</if>
42.             <if test="cnName != null and cnName != ''">#{cnName},</i
f>
43.             <if test="enName != null and enName != ''">#{enName},</i
f>
44.             <if test="code != null and code != ''">#{code},</if>
45.             <if test="ancestors != null and ancestors != ''">#{ances
tors},</if>
46.             <if test="orderNum != null">#{orderNum},</if>
47.             <if test="status != null">#{status},</if>
48.             <if test="level != null">#{level},</if>
49.             <if test="createBy != null and createBy != ''">#{createB
y},</if>
50.             sysdate()

```

```
51.     )
52. </insert>
```

Under this code is mainly to do the ICD additional code new input code code, when the super administrator and doctor start the system and start to create additional code is to use this way to input, scheduling content is also the same.

Source Code Inventory 5

Class ICDADD-update.java

```
1. @PostMapping
2.     public AjaxResult edit(@Validated @RequestBody SysIcd icdAdd)
3.     {
4.         Long icdAddId = icdAdd.getIcdId();
5.         if (!icdAddService.checkCodeUnique(icdAdd))
6.         {
7.             return error("修改 IcdAdd'" + icdAdd.getCode() + "'失败,
Code 已存在");
8.         }
9.         else if (icdAdd.getParentId().equals(icdAddId))
10.        {
11.            return error("修改 IcdAdd'" + icdAdd.getCnName() + "'失败,
上级 IcdAdd 不能是自己");
12.        }
13.        else if (StringUtils.equals(UserConstants.DEPT_DISABLE, icdA
dd.getStatus()) && icdAddService.selectNormalChildrenIcdAddById(icdA
ddId) > 0)
14.        {
15.            return error("该 IcdAdd 包含未停用的子 IcdAdd! ");
16.        }
17.        icdAdd.setUpdateBy(getUsername());
18.        return toAjax(icdAddService.updateIcdAdd(icdAdd));
19.    }
20.
21. public int updateIcdAdd(SysIcd icd);
22.     @Override
23.     public int updateIcdAdd(SysIcd icdAdd)
24.     {
25.         SysIcd newParentIcdAdd = icdAddMapper.selectIcdAddById(icdAd
d.getParentId());
26.         SysIcd oldIcdAdd = icdAddMapper.selectIcdAddById(icdAdd.getI
cdId());
27.         if (StringUtils.isNotEmpty(newParentIcdAdd) && StringUtils.is
NotNull(oldIcdAdd))
28.         {
29.             String newAncestors = newParentIcdAdd.getAncestors() + "
," + newParentIcdAdd.getIcdId();
30.             String oldAncestors = oldIcdAdd.getAncestors();
31.             icdAdd.setAncestors(newAncestors);
32.             updateIcdAddChildren(icdAdd.getIcdId(), newAncestors, ol
dAncestors);
33.         }
34.         int result = icdAddMapper.updateIcdAdd(icdAdd);
35.         if (UserConstants.DEPT_NORMAL.equals(icdAdd.getStatus()) &&
StringUtils.isNotEmpty(icdAdd.getAncestors()))
```

```

36.         && !StringUtils.equals("0", icdAdd.getAncestors()))
37.     {
38.         // 如果该 IcdAdd 是启用状态, 则启用该 IcdAdd 的所有上级
        IcdAdd
39.         updateParentIcdAddStatusNormal(icdAdd);
40.     }
41.     return result;
42. }
43.
44. public int updateIcdAdd(SysIcd icd);
45.     <update id="updateIcdAdd" parameterType="SysIcd">
46.         update sys_icd_add
47.         <set>
48.             <if test="parentId != null and parentId != 0">parent_id
        = #{parentId},</if>
49.             <if test="cnName != null and cnName != ''">cn_name = #{c
        nName},</if>
50.             <if test="enName != null and enName != ''">en_name = #{e
        nName},</if>
51.             <if test="ancestors != null and ancestors != ''">ancesto
        rs = #{ancestors},</if>
52.             <if test="code != null and code != ''">code = #{code},</
        if>
53.             <if test="orderNum != null">order_num = #{orderNum},</if
        >
54.             <if test="status != null and status != ''">status = #{st
        atus},</if>
55.             <if test="level != null and level != ''">level = #{level
        },</if>
56.             <if test="updateBy != null and updateBy != ''">update_by
        = #{updateBy},</if>
57.             update_time = sysdate()
58.         </set>
59.         where icd_id = #{icdId}
60.     </update>

```

Under this code is mainly to do the ICD supplementary code input update code, when the super administrator and doctor identity start the system and start to create additional code is to use this way to make new data modification, in this area we can review the content of the additional code and correct the correctness of the additional code .

Source Code Inventory 6

Class ICDADD-delete.java

```

1. @DeleteMapping("/{icdAddId}")
2.     public AjaxResult remove(@PathVariable Long icdAddId)
3.     {
4.         if (icdAddService.hasChildByIcdAddId(icdAddId))
5.         {
6.             return warn("存在下级 IcdAdd, 不允许删除");
7.         }

```

```

8.         return toAjax(icdAddService.deleteIcdAddById(icdAddId));
9.     }
10. public int deleteIcdAddById(Long icdId);
11.     @Override
12.     public int deleteIcdAddById(Long icdId)
13.     {
14.         return icdAddMapper.deleteIcdAddById(icdId);
15.     }
16. public int deleteIcdAddById(Long icdId);
17.     <delete id="deleteIcdAddById" parameterType="Long">
18.         update sys_icd_add set del_flag = '2' where icd_id = #{icdId}
19.     </delete>

```

Under this code is the code for ICD plus code deletion. When the super administrator and doctor start the system and start to delete the old main code, this is the way to modify the new data. The content of scheduling when deleting is also the code of this area. It is not possible to delete large modules with subsets below them.

Source Code Inventory 7

Class User-create.java

```

1. public AjaxResult add(@Validated @RequestBody SysUser user)
2.     {
3.         if (!userService.checkUserNameUnique(user))
4.         {
5.             return error("新增用户" + user.getUserName() + "'失败，登
录账号已存在");
6.         }
7.         else if (StringUtils.isNotEmpty(user.getPhonenumber()) && !u
serService.checkPhoneUnique(user))
8.         {
9.             return error("新增用户" + user.getUserName() + "'失败，手
机号码已存在");
10.        }
11.        else if (StringUtils.isNotEmpty(user.getEmail()) && !userSer
vice.checkEmailUnique(user))
12.        {
13.            return error("新增用户" + user.getUserName() + "'失败，邮
箱账号已存在");
14.        }
15.        user.setCreateBy(getUsername());
16.        user.setPassword(SecurityUtils.encryptPassword(user.getPassw
ord()));
17.        return toAjax(userService.insertUser(user));
18.    }
19. public int insertUser(SysUser user);
20. @Override
21.     @Transactional
22.     public int insertUser(SysUser user)
23.     {
24.         // 新增用户信息
25.         int rows = userMapper.insertUser(user);
26.         // 新增用户岗位关联
27.         insertUserPost(user);
28.         // 新增用户与角色管理

```

```

29.         insertUserRole(user);
30.         return rows;
31.     }
32. public int insertUser(SysUser user);
33.     <insert id="insertUser" parameterType="SysUser" useGeneratedKeys
34.         ="true" keyProperty="userId">
35.         insert into sys_user(
36.             <if test="userId != null and userId != 0">user_id,</if>
37.             <if test="deptId != null and deptId != 0">dept_id,</if>
38.             <if test="userName != null and userName != ''">user_name
39.             ,</if>
40.             <if test="nickName != null and nickName != ''">nick_name
41.             ,</if>
42.             <if test="email != null and email != ''">email,</if>
43.             <if test="avatar != null and avatar != ''">avatar,</if>
44.             <if test="phonenummer != null and phonenummer != ''">pho
45.             nenummer,</if>
46.             <if test="sex != null and sex != ''">sex,</if>
47.             <if test="password != null and password != ''">password,
48.             </if>
49.             <if test="status != null and status != ''">status,</if>
50.             <if test="createBy != null and createBy != ''">create_by
51.             ,</if>
52.             <if test="remark != null and remark != ''">remark,</if>
53.             create_time
54.         )values(
55.             <if test="userId != null and userId != ''">#{userId},</i
56.             f>
57.             <if test="deptId != null and deptId != ''">#{deptId},</i
58.             f>
59.             <if test="userName != null and userName != ''">#{userNam
60.             e},</if>
61.             <if test="nickName != null and nickName != ''">#{nickNam
62.             e},</if>
63.             <if test="email != null and email != ''">#{email},</if>
64.             <if test="avatar != null and avatar != ''">#{avatar},</i
65.             f>
66.             <if test="phonenummer != null and phonenummer != ''">#{p
67.             honenummer},</if>
68.             <if test="sex != null and sex != ''">#{sex},</if>
69.             <if test="password != null and password != ''">#{passwor
70.             d},</if>
71.             <if test="status != null and status != ''">#{status},</i
72.             f>
73.             <if test="createBy != null and createBy != ''">#{createB
74.             y},</if>
75.             <if test="remark != null and remark != ''">#{remark},</i
76.             f>
77.             sysdate()
78.         )
79.     </insert>

```

Under this code is the code for the user to create a new user. When the user starts to register again, it is entered in this way, and the user's judgment is added. For example, the same account cannot be registered twice directly. And the detailed information of the user is also sorted out, so is the scheduling of content.

Source Code Inventory 8

Клас User-update.java

```
1. @PostMapping
2.     public AjaxResult edit(@Validated @RequestBody SysUser user)
3.     {
4.         userService.checkUserAllowed(user);
5.         userService.checkUserDataScope(user.getUserId());
6.         if (!userService.checkUserNameUnique(user))
7.         {
8.             return error("修改用户" + user.getUserName() + "'失败，登
录账号已存在");
9.         }
10.        else if (StringUtils.isNotEmpty(user.getPhonenumber()) && !u
serService.checkPhoneUnique(user))
11.        {
12.            return error("修改用户" + user.getUserName() + "'失败，手
机号码已存在");
13.        }
14.        else if (StringUtils.isNotEmpty(user.getEmail()) && !userSer
vice.checkEmailUnique(user))
15.        {
16.            return error("修改用户" + user.getUserName() + "'失败，邮
箱账号已存在");
17.        }
18.        user.setUpdateBy(getUsername());
19.        return toAjax(userService.updateUser(user));
20.    }
21. public int updateUser(SysUser user);
22.     @Override
23.     @Transactional
24.     public int updateUser(SysUser user)
25.     {
26.         Long userId = user.getUserId();
27.         // 删除用户与角色关联
28.         userRoleMapper.deleteUserRoleByUserId(userId);
29.         // 新增用户与角色管理
30.         insertUserRole(user);
31.         // 删除用户与岗位关联
32.         userPostMapper.deleteUserPostByUserId(userId);
33.         // 新增用户与岗位管理
34.         insertUserPost(user);
35.         return userMapper.updateUser(user);
36.     }
37. public int updateUser(SysUser user);
38.     <update id="updateUser" parameterType="SysUser">
```

```

39.         update sys_user
40.         <set>
41.             <if test="deptId != null and deptId != 0">dept_id = #{de
ptId},</if>
42.             <if test="userName != null and userName != ''">user_name
= #{userName},</if>
43.             <if test="nickName != null and nickName != ''">nick_name
= #{nickName},</if>
44.             <if test="email != null ">email = #{email},</if>
45.             <if test="phoneNumber != null ">phoneNumber = #{phonenum
ber},</if>
46.             <if test="sex != null and sex != ''">sex = #{sex},</if>
47.             <if test="avatar != null and avatar != ''">avatar = #{av
atar},</if>
48.             <if test="password != null and password != ''">password
= #{password},</if>
49.             <if test="status != null and status != ''">status = #{st
atus},</if>
50.             <if test="loginIp != null and loginIp != ''">login_ip =
#{loginIp},</if>
51.             <if test="loginDate != null">login_date = #{loginDate},<
/if>
52.             <if test="updateBy != null and updateBy != ''">update_by
= #{updateBy},</if>
53.             <if test="remark != null">remark = #{remark},</if>
54.             update_time = sysdate()
55.         </set>
56.         where user_id = #{userId}
57.     </update>

```

Under this code is the main user registration update code, when the super administrator, doctor and patient identity start the system and start the registration and login is to use this way to modify the new user registration data, in this area we can review the content of the new user and correct the correctness of the attached code.

Source Code Inventory 9

Клас User-delete.java

```

1. @DeleteMapping("/{userIds}")
2.     public AjaxResult remove(@PathVariable Long[] userIds)
3.     {
4.         if (ArrayUtils.contains(userIds, getUserId()))
5.         {
6.             return error("当前用户不能删除");
7.         }
8.         return toAjax(userService.deleteUserByIds(userIds));
9.     }
10. public int deleteUserByIds(Long[] userIds);
11. @Override
12. @Transactional
13. public int deleteUserByIds(Long[] userIds)

```



```

14.     {
15.         for (Long userId : userIds)
16.         {
17.             checkUserAllowed(new SysUser(userId));
18.             checkUserDataScope(userId);
19.         }
20.         // 删除用户与角色关联
21.         userRoleMapper.deleteUserRole(userIds);
22.         // 删除用户与岗位关联
23.         userPostMapper.deleteUserPost(userIds);
24.         return userMapper.deleteUserByIds(userIds);
25.     }
26. public int deleteUserByIds(Long[] userIds);
27.     <delete id="deleteUserByIds" parameterType="Long">
28.         update sys_user set del_flag = '2' where user_id in
29.         <foreach collection="array" item="userId" open="(" separator
30.             ="," close=")">
31.             #{userId}
32.         </foreach>
33.     </delete>

```

Under this code is the main code to delete the registered user, when the super administrator starts the system and begins to delete old users or zombie users is to use this way to modify the new user data, to delete user information when the scheduling content is the code of this area, it should be noted that the code judgment is added in this area.

Source Code Inventory 10

Class Case-create.java

```

1. @PostMapping
2. public AjaxResult add(@Validated @RequestBody SysCase sysCase)
3. {
4.     sysCase.setCreateBy(getUsername());
5.     return toAjax(sysCaseService.insertCase(sysCase));
6. }
7. public void insertCaseIcdAdd(SysCase user);
8. @Override
9. public void insertCaseIcdAdd(SysCase sysCase) {
10.     Long[] icdAddCodes = sysCase.getIcdAddCodes();
11.     if (StringUtils.isNotEmpty(icdAddCodes))
12.     {
13.         List<SysCaseIcdAdd> list = new ArrayList<SysCaseIcdAdd>(
14.             icdAddCodes.length);
15.         for (Long icdAddId : icdAddCodes)
16.         {
17.             SysCaseIcdAdd scia = new SysCaseIcdAdd();
18.             scia.setIcdAddId(icdAddId);
19.             scia.setCaseId(sysCase.getCaseId());
20.             list.add(scia);
21.         }
22.         caseIcdAddMapper.batchCaseIcdAdd(list);
23.     }

```

```

24. public int insertCase(SysCase sysCase);
25.     <insert id="insertCase" parameterType="SysCase" useGeneratedKeys
    = "true" keyProperty="caseId">
26.         insert into sys_case(
27.             <if test="caseId != null and caseId != ''">case_id,</if>
28.             <if test="caseName != null and caseName != ''">case_name
    ,</if>
29.             <if test="patientId != null and patientId != ''">patient
    _id,</if>
30.             <if test="code != null and code != ''">code,</if>
31.             <if test="content != null and content != ''">content,</i
    f>
32.             <if test="delFlag != null and delFlag != ''">del_flag,</
    if>
33.             <if test="createBy != null and createBy != ''">create_by
    ,</if>
34.             <if test="remark != null and remark != ''">remark,</if>
35.             create_time
36.         )values(
37.             <if test="caseId != null and caseId != ''">#{caseId},</i
    f>
38.             <if test="caseName != null and caseName != ''">#{caseNam
    e},</if>
39.             <if test="patientId != null and patientId != ''">#{patie
    ntId},</if>
40.             <if test="code != null and code != ''">#{code},</if>
41.             <if test="content != null and content != ''">#{content},
    </if>
42.             <if test="delFlag != null and delFlag != ''">#{delFlag},
    </if>
43.             <if test="createBy != null and createBy != ''">#{createB
    y},</if>
44.             <if test="remark != null and remark != ''">#{remark},</i
    f>
45.             sysdate()
46.         )
47.     </insert>
48.
49. </delete>

```

Under this code is the code that mainly does the user's new case information. When the user starts to register and enter the case, it is entered in this way, and the judgment of the doctor user is added. For example, the same account cannot be registered twice directly. And the detailed information of the user is also sorted out, so is the scheduling of content.

Source Code Inventory 11

Class Case-update.java

```

1. @PutMapping
2.     public AjaxResult edit(@Validated @RequestBody SysCase sysCase)
3.     {
4.         sysCase.setUpdateBy(sysCase.getUpdateBy());

```

```

5.         return toAjax(sysCaseService.updateCase(sysCase));
6.     }
7.     public int updateCase(SysCase sysCase);
8.     @Override
9.     @Transactional
10.    public int updateCase(SysCase sysCase)
11.    {
12.        String sysCaseId = sysCase.getCaseId();
13.        // 删除 Case 与 icdAdd 关联
14.        caseIcdAddMapper.deleteCaseIcdAddByCaseId(sysCaseId);
15.        // 新增 Case 与 icdAdd 管理
16.        insertCaseIcdAdd(sysCase);
17.        return caseMapper.updateCase(sysCase);
18.    }
19.    public int updateCase(SysCase sysCase);
20.    <update id="updateCase" parameterType="SysCase">
21.        update sys_case
22.        <set>
23.            <if test="caseId != null and caseId != '>case_id = #{c
aseId},</if>
24.            <if test="caseName != null and caseName != '>case_name
= #{caseName},</if>
25.            <if test="patientId != null and patientId != '>patient
_id = #{patientId},</if>
26.            <if test="code != null and code != '>code = #{code},</
if>
27.            <if test="content != null and content != '>content = #
{content},</if>
28.            <if test="delFlag != null and delFlag != '>del_flag =
#{delFlag},</if>
29.
30.            <if test="updateBy != null and updateBy != '>update_by
= #{updateBy},</if>
31.            <if test="remark != null">remark = #{remark},</if>
32.            update_time = sysdate()
33.        </set>
34.        where case_id = #{caseId}
35.    </update>

```

Under this code is the code that mainly updates the patient's case information. When the super administrator, doctor and patient's identity start the system and modify the user's case data in this way, we can modify and check the contents of the already issued case information and correct the correctness in this area.

Source Code Inventory12

Class Case-delete.java

```

1.     @DeleteMapping("/{sysCaseIds}")
2.     public AjaxResult remove(@PathVariable String[] sysCaseIds)
3.     {
4.         return toAjax(sysCaseService.deleteCaseByIds(sysCaseIds));
5.     }
6.     public int deleteCaseByIds(String[] sysCaseIds);
7.     @Override

```

```

8.     @Transactional
9.     public int deleteCaseById(String sysCaseId)
10.    {
11.        // 删除 Case 与 icdAdd 关联
12.        caseIcdAddMapper.deleteCaseIcdAddByCaseId(sysCaseId);
13.        caseMapper.deleteCaseById(sysCaseId);
14.        return caseMapper.deleteCaseById(sysCaseId);
15.    }
16. public int deleteCaseById(String sysCaseId);
17. <delete id="deleteCaseById" parameterType="String">
18.     delete from sys_case where case_id = #{caseId}
19. </delete>

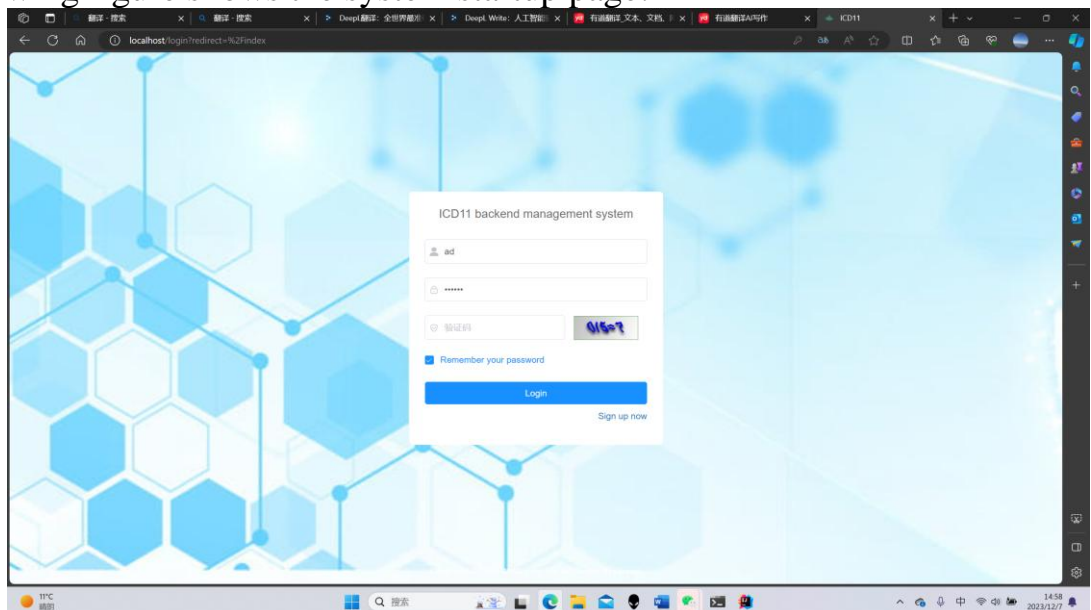
```

Under this code is the code for registered users to delete the wrong cases. When the super administrator and doctor start the system and start to delete the wrong cases or zombie user cases, this is the way to modify the new case data. When deleting the pathological information of users, the scheduling content is the code of this area. It should be noted that the deletion of the case message judgment has been added to this area.

4.3. Web results display and function detailed introduction

ICD11API Web application startup page, users can start from here to register to perform operations.

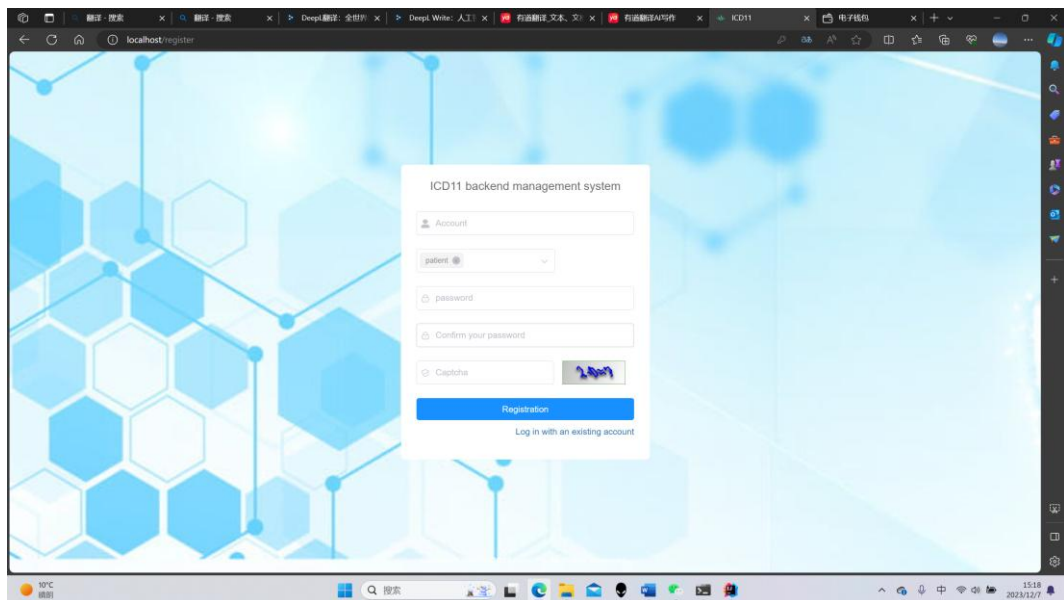
The following figure shows the system startup page.



System Launch Page

The user needs to register on the page again. The user can choose different identities when registering (super administrator identity is not optional, only doctor or patient identity can be selected). After entering the account and password, the user will enter the main page. The user must meet the requirements of each field, that is, no data outside the setting can be entered. At this stage, data verification has been carried out, because if it is found that the entered user does not have the required "user" role, it will not be allowed to log in to the system. In this function, I added the function of verification code to protect the user's privacy and prevent the robot from logging in.

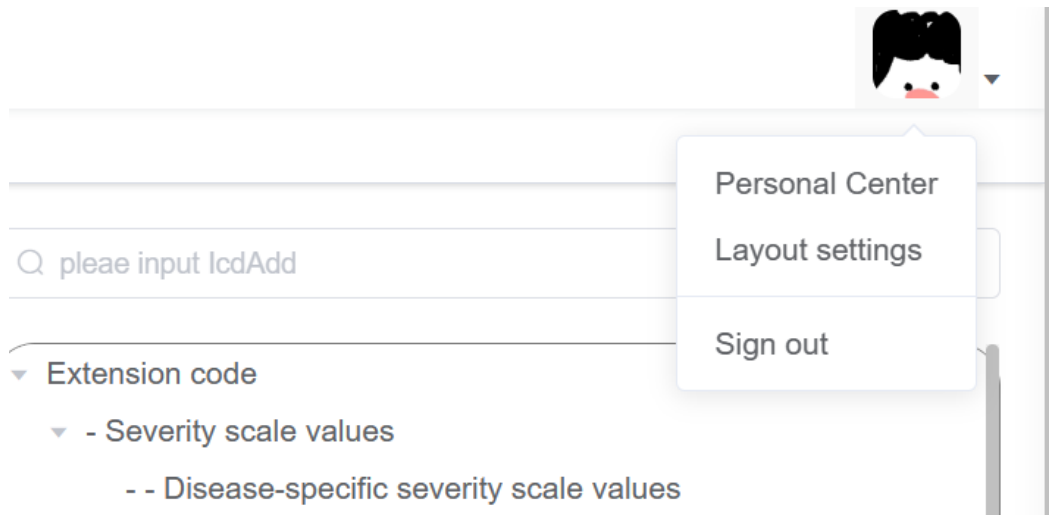
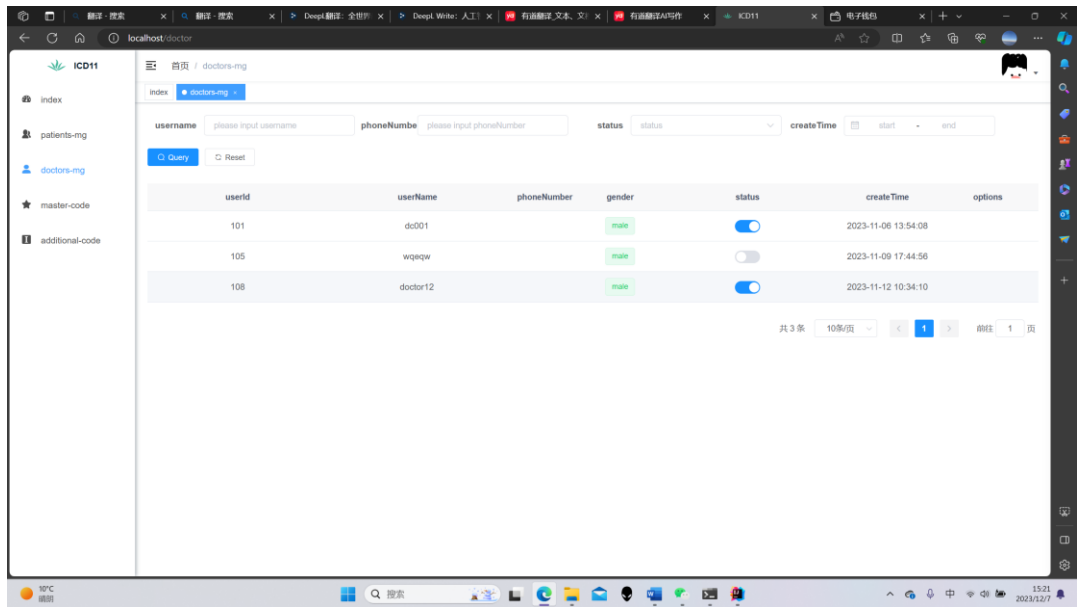
The figure shows the authorization page



authorization page

If the user is a patient, you can directly register to log in. If the user is a doctor, you need to wait for the super administrator's audit after submitting the registration information, and you need to wait for a longer time, because the administrator's audit is manual and may be slow. If the user is an administrator, you will see a special user (doctor) registration login request, the administrator has the right to review whether the other party is a doctor, if the other party is determined to open the login channel for the other party will not change, nor allow the other party to access the system.

The figure shows the administrator review page。

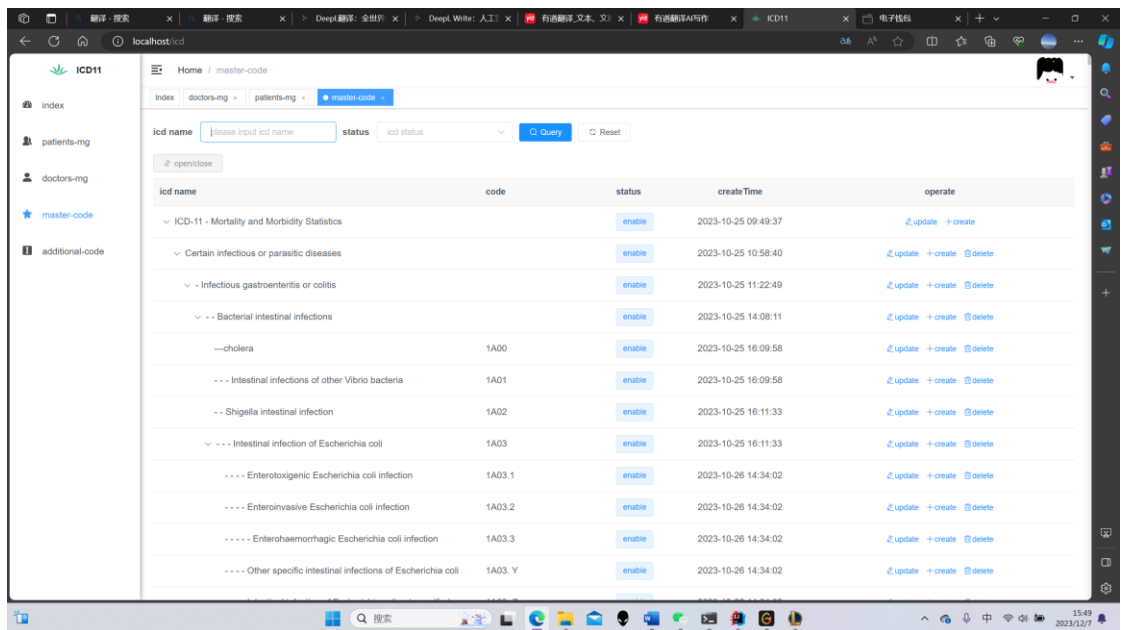


Administrator audits doctor identity page

After confirming the identity data on the administrator page, the registered user will be opened to enter this page using the doctor privileges. You can view all the ICD11 encoded data and patient information data, and you can add and modify the patient information data, the administrator does not know that you can view the patient data can also view the doctor's data, and then open many options.

In addition, at the top right of the page, there is already a button that opens the secondary interface by clicking on the user name, including the option to log out and change the background. It also added a data retrieval function, which searches for certain information and sorts it to make faster choices about the patient's situation.

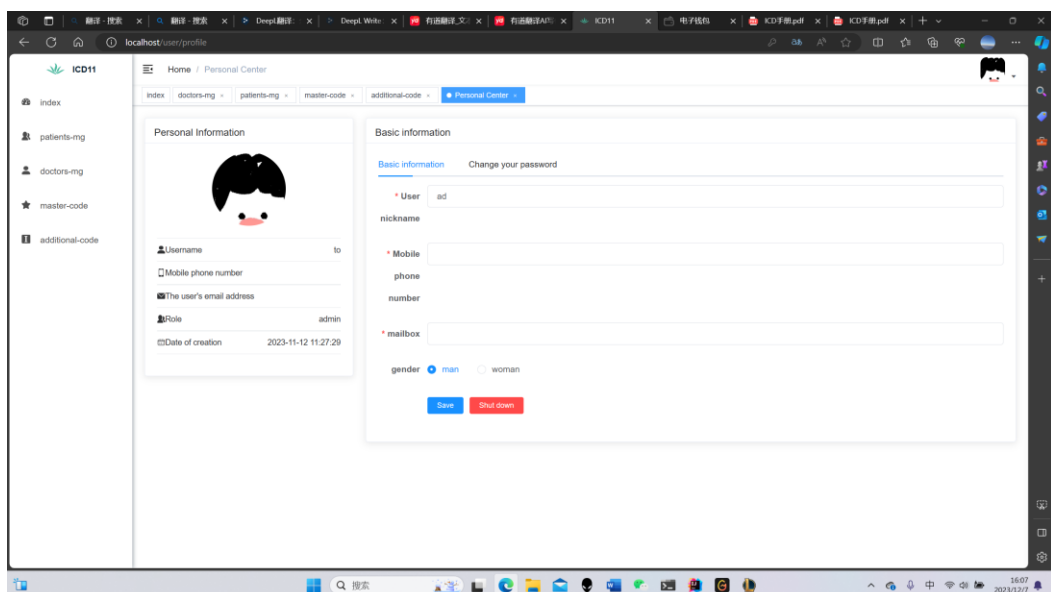
The figure shows the administrator's code management page



Administrator's code management page.

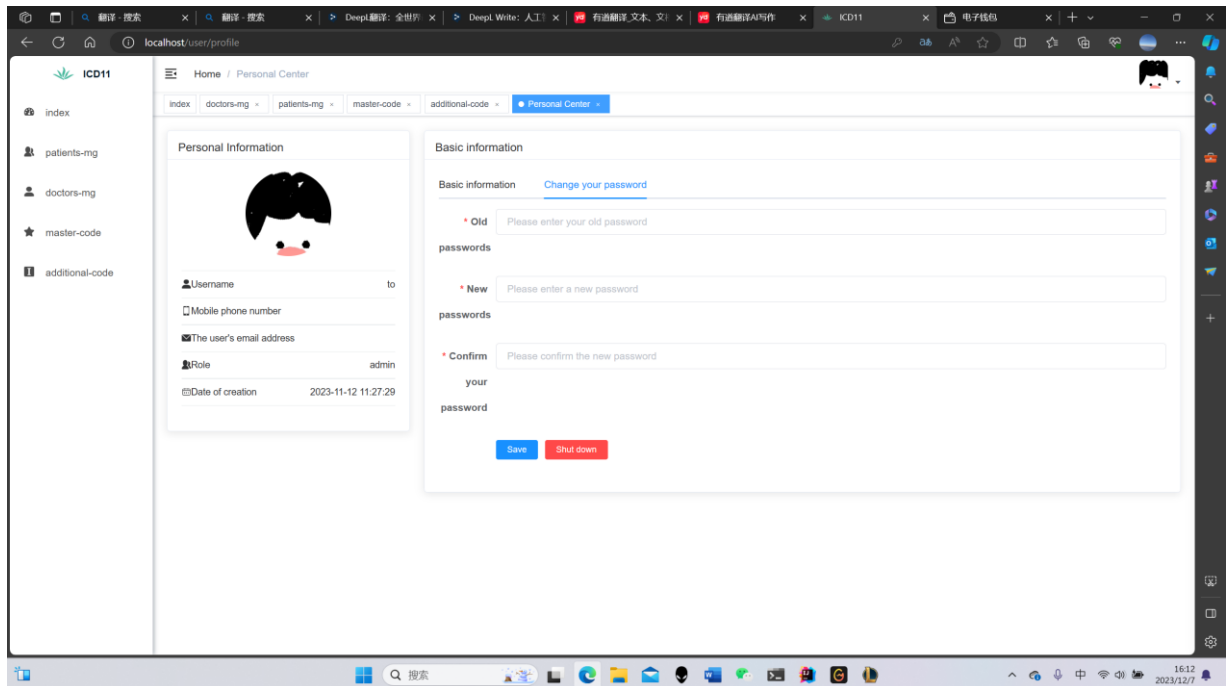
The administrator can perform the following operations on the user: In addition to reviewing new users, changing user information or deleting user information, the administrator can also operate the ICD11 code. In order to facilitate the update of the ICD11 code, I have done visual work and directly added the increase of ICD11 code on the webpage, so that the administrator can directly add it through the webpage and reduce the cost of manual learning. All operations can be carried out through data filling forms, which can reduce the cost of manual cultivation and manual modification costs.

The figure shows user management of accounts and passwords.



User management of accounts and passwords

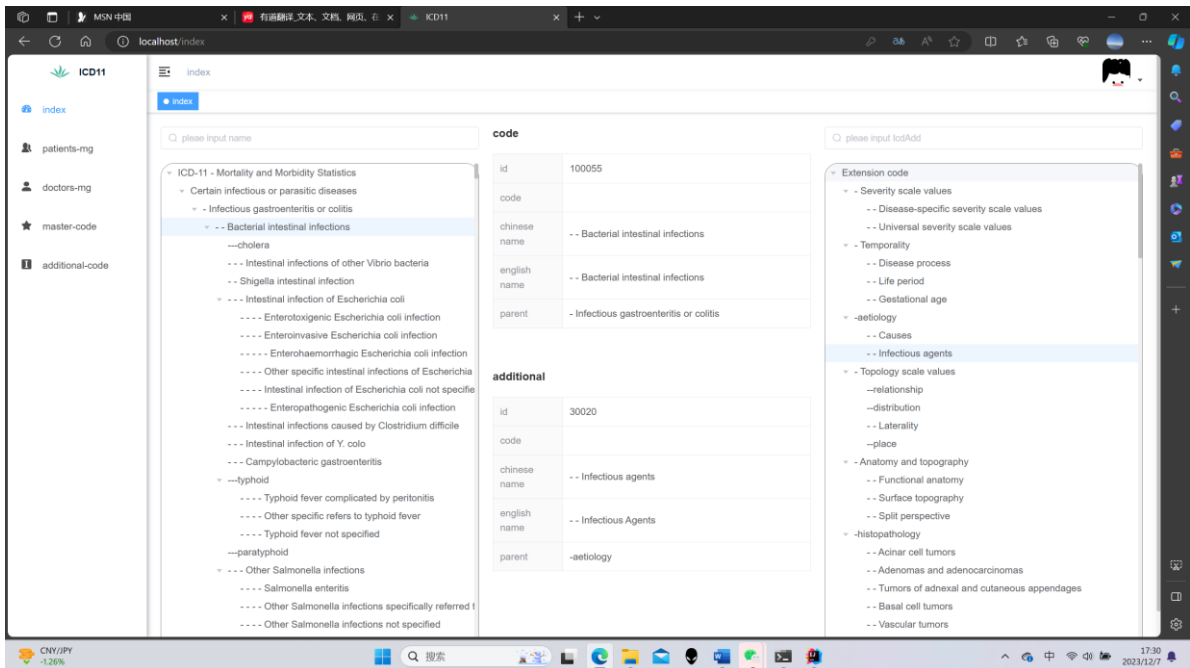
The user changes the account password as shown in the figure below. It should be noted that I have specifically added the confirmation of the user's old password here to ensure that the user who is operating is himself.



Password change interface

Back to the main page of the largest functional area, the current display is the super administrator permissions can see the main interface, we can check the patient's situation here, in the form of text to describe you can see that it can automatically generate ICD11 code to the patient when the system will be directed to the patient's detailed disease. The left side is the trunk code, the right side is a detailed description of the trunk code of the additional code, our choice of trunk code is necessary, can not be skipped, the current library is still improving.

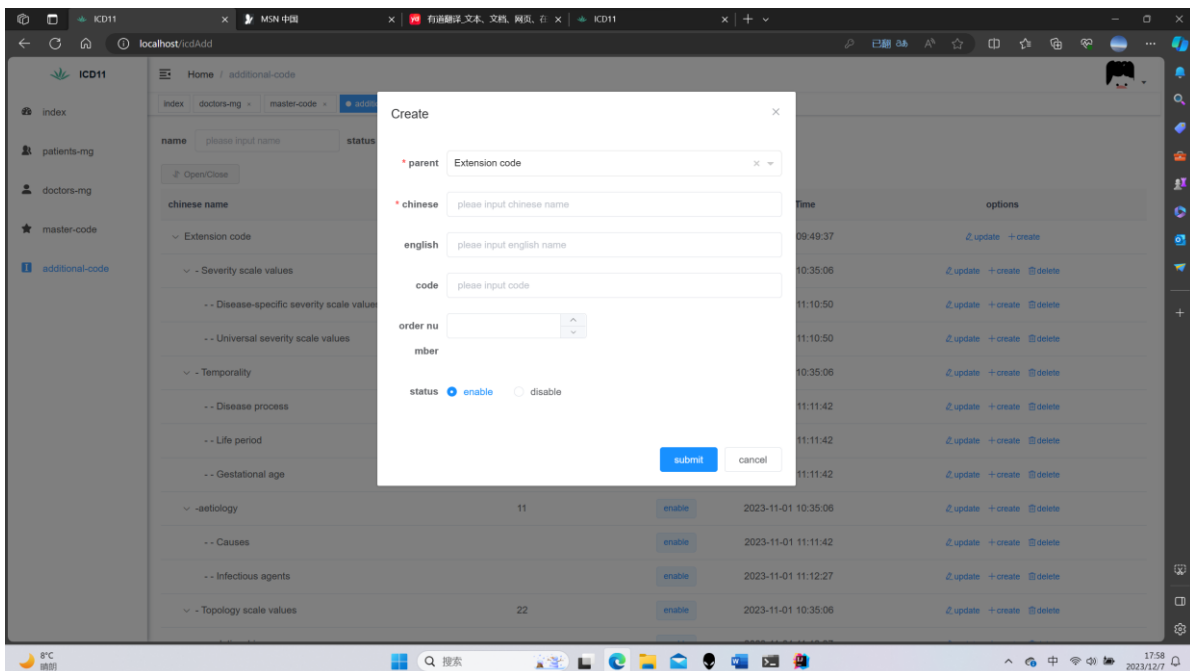
Figure shows the pathology code page at the administrator level.



Admin level pathology code page

The super administrator can perform the following operations on the pathology code of ICD11: Create a new code, change the information of an existing code, or delete an error code (expired code). Including deletion, I have made all the visual design, so that you can directly edit the pathological code information, the advantage is that the operation is simple, but also become convenient.

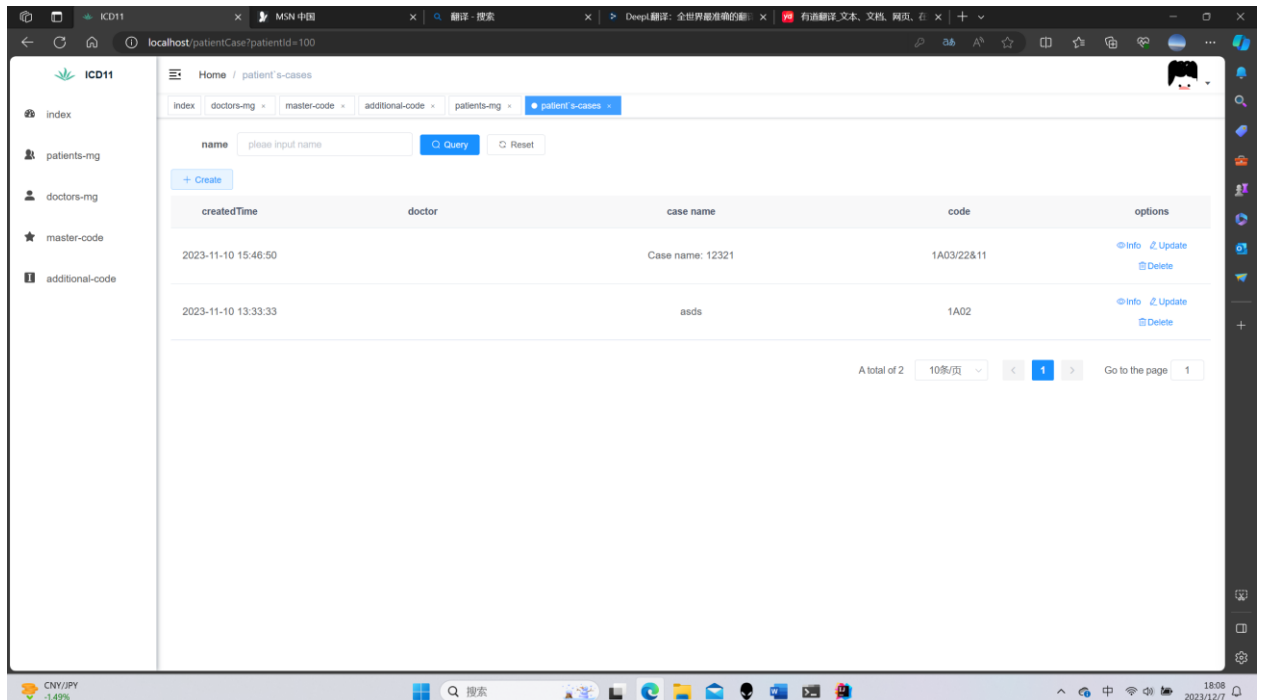
The figure shows the page for creating a new pathology code.



Create a page for a new pathology code

The super administrator can choose the following actions for a patient's permissions: change an existing patient's error message or delete an incorrect patient's message (expired code). I also designed everything visually so that I could directly manage the patient's information.

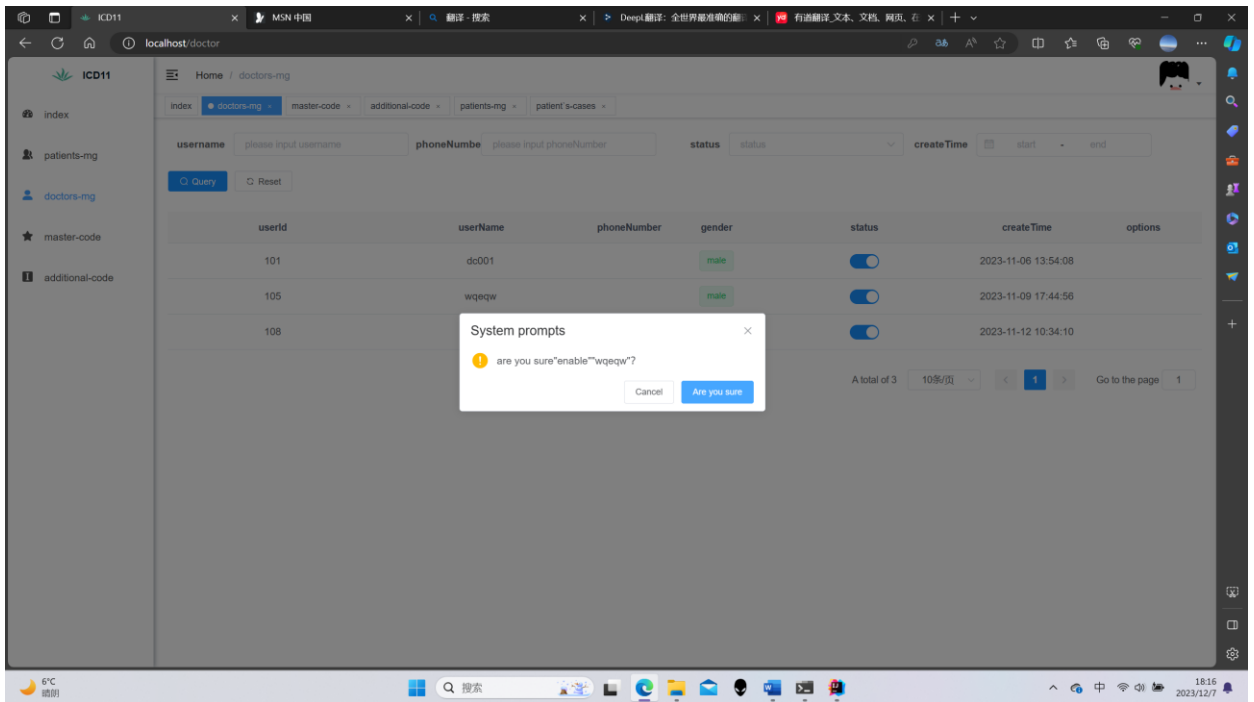
Figure shows the page for changing patient information.



Change the patient information page

The super administrator can choose the following operations for the doctor's review: open the doctor's power, allow the doctor's account registration, or block the doctor's account if it detects an anomaly, and see the doctor's application registration time. I also designed it visually so that I could directly manage the doctor's information and situation.

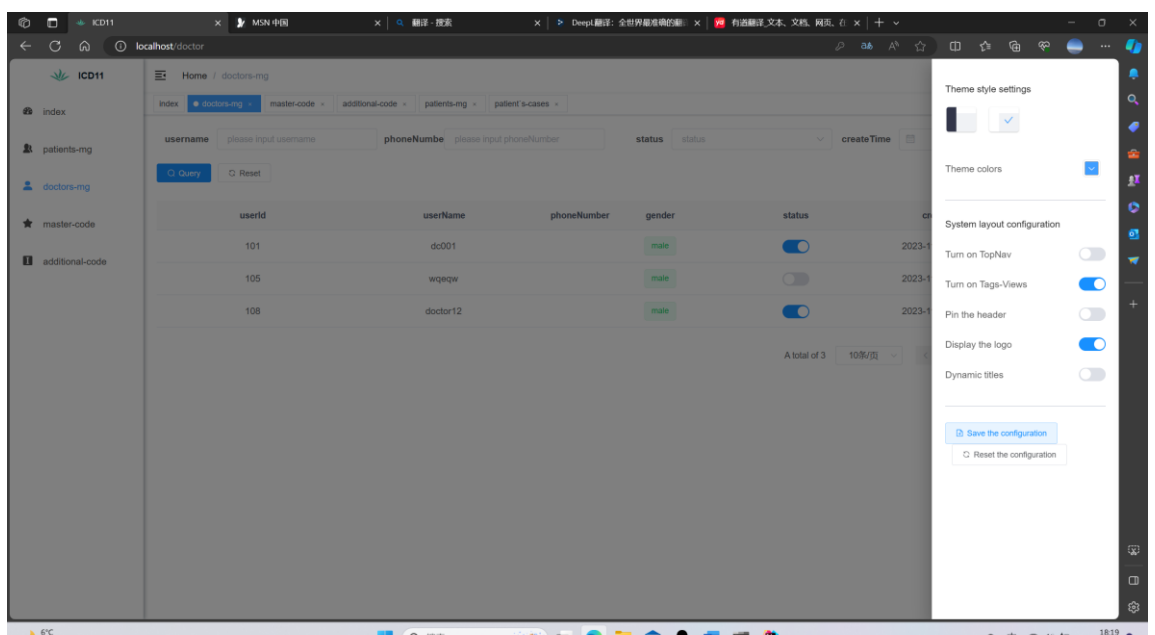
Super administrator audits the Doctor identity interface.



The super administrator reviews the doctor identity interface

Click the "Layout Settings" button at the top right of the page to open the "Modify layout" page. All users can make their own theme switching and selection here, even I have added the WEB header update here. Users can view all modifiable theme information on the main page or template view page, for example, users can select a dynamic title to know which page they are currently on, and users can also add TOPNAV to make the interface look cleaner.

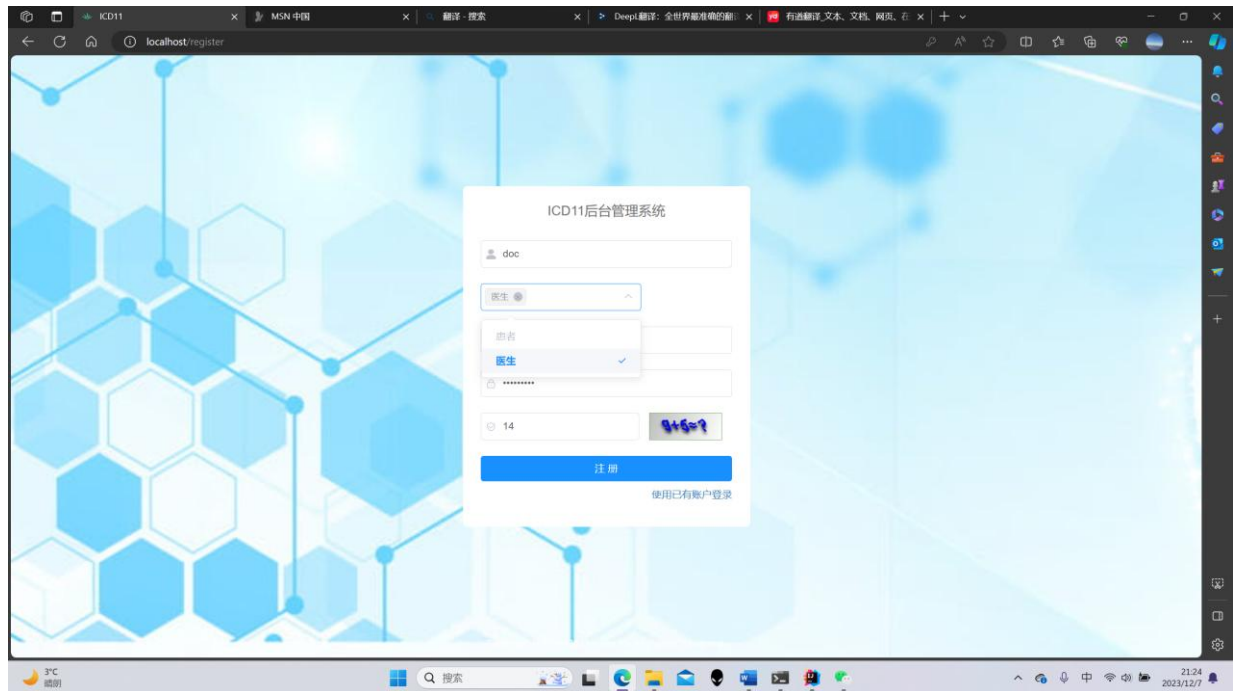
Figure user layout management page.



The user layout administration page

What I'm showing here is a page that I designed to use specific information for doctor registration. Registrants can choose their own identity, according to the selected identity will be different rights management, in order to create a special identity, now show the doctor's identity registration and interface.

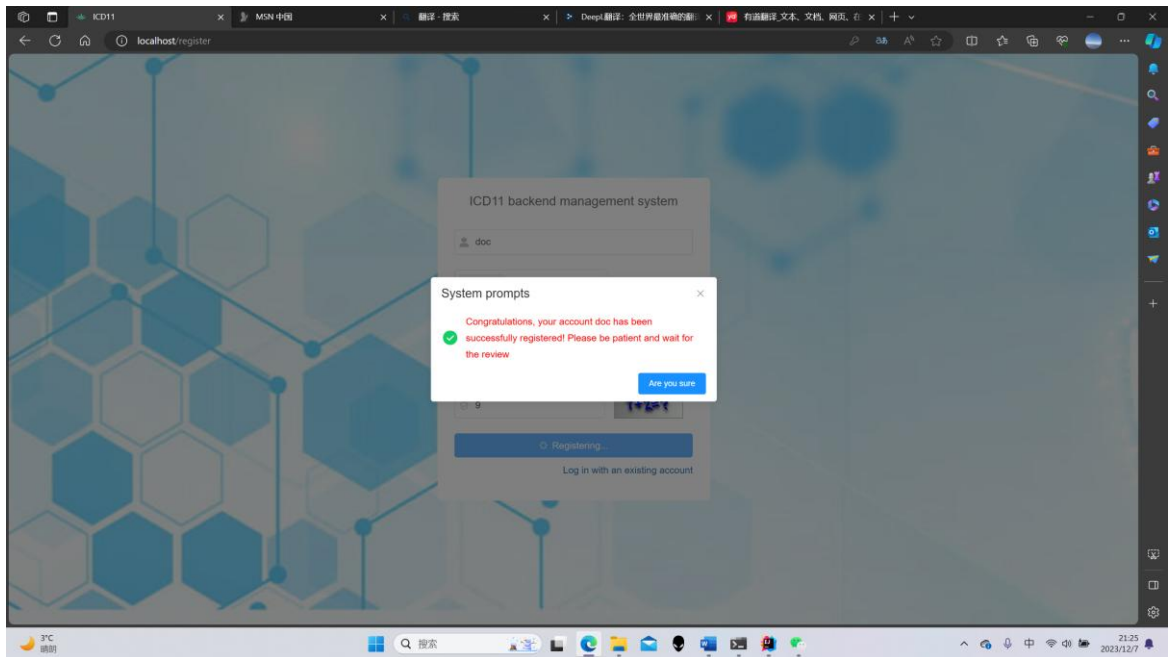
Doctor Graph registration interface



The doctor's registration screen

What is displayed now is the prompt information after the doctor submits the registration message. Because the doctor's identity information is special, we need to add manual audit. After the manual audit is passed, the doctor's identity user can normally use the WEB for information operation.

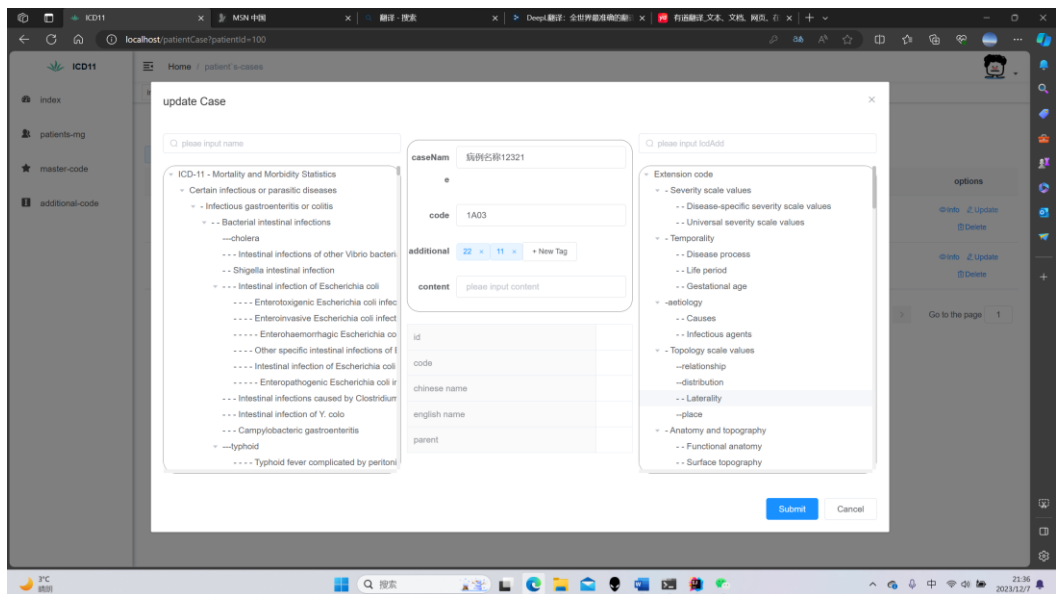
Dr. Tu's registration review interface



Doctor registration review interface

What is displayed now is the patient information after the doctor submitted the registration message. Because the doctor's identity information is special, he can see the patient's information, and he can also choose the patient's pathological code here, so he can also change the selected information. After the approval, the doctor can directly visualize the patient and reduce the lifetime of learning tasks and learning costs.

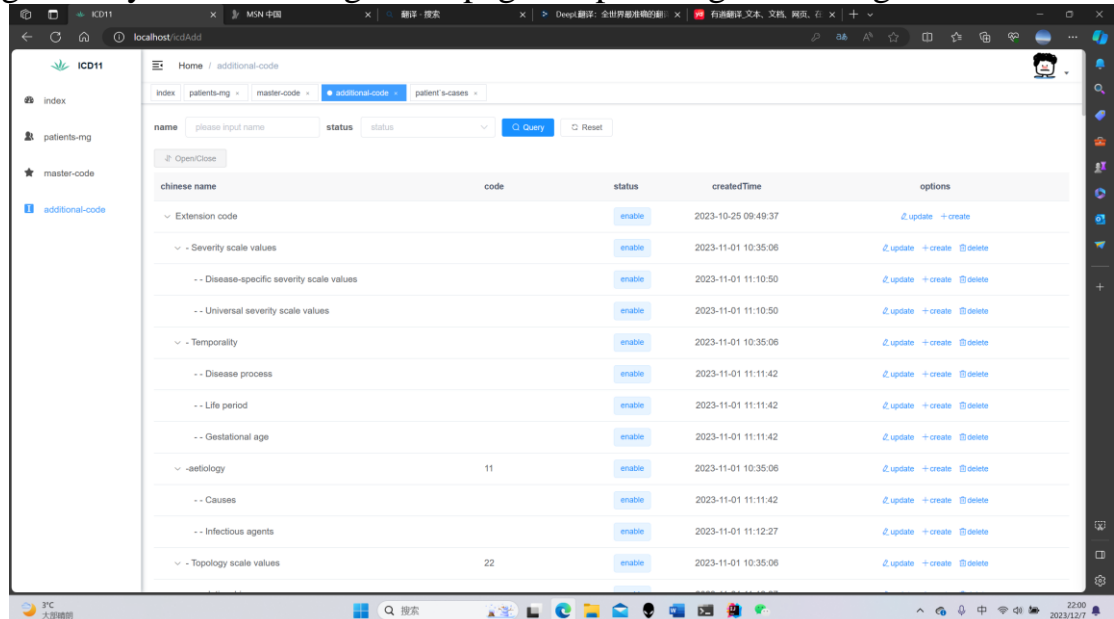
Dr. Tu's management page for patient pathology coding.



Dr. management page for patient pathology coding.

Because I hope that the whole people can work together to increase the ICD11 database, I have opened this permission to users who are doctors, and I hope that everyone can work together to promote my database to get better and better。

Figure: Physician's management page for pathological coding



Physician's management page for pathological coding

Conclusion

In this section, we have made a prototype of the WEB system of ICD11API and described its specific functions in detail. In my opinion, the highlights of the design are as follows:

I divided the users according to their identities, and divided the functions according to their identities and different needs, so that users can have a better positioning of their needs, although it may increase some workload to a certain extent, but at the root of it, I think it is worthwhile for users to get more accurate services.

I use the verification code of person and person identification in the login interface to ensure that the user's ICD11 diagnostic code is private, which can protect the privacy of patients to the greatest extent, solve the security problems of patients, and ensure the privacy of patients to the greatest extent.

In addition, after the study of this project, I understand that the development of a complete project must have clear goals and ideas, and carry out necessary summary and demonstration in each stage.

The development stages of a complete project include: vision scope planning and use case specification, project structure and risk assessment, business function specification, detailed design specification, code implementation, testing and installation package, and so on. The development of a project requires a lot of financial resources and manpower. If there is no good long-term plan, it will have a

great impact on the development progress in the future, and even appear that the project cannot be completed within the scheduled time or the completed project is different from the original expectation. A good project structure, business functions and detailed design specification have a clear guiding effect on the development of a project, which can make developers have a clearer understanding of the functions to be achieved in the project as a whole, and can also reduce unnecessary trouble in the development process. The implementation of the code is the key to the success of a project development, that is to say, the early work is to prepare for the implementation of the code.

I will continue to work hard to make my project more and more perfect, and strive to help many people.

CONCLUSIONS

Don't limit yourself to superficial use of a technology, even once or twice. "Don't worry about everything" is a quality that engineers in any industry should not have. Develop windows applications, look at the design, loading, execution principles of windows programs, analyze the pe file format, and try to develop a windows application from scratch with sdk development; Use, delphi, java, net to develop applications, spend time to study mfc, vcl, j2ee, net their framework design or source code; In addition to using excellent open source products or frameworks such as j2ee, jboss, spring, hibernate, etc., take a look at how the gurus abstract, analyze, design, and implement common solutions to similar problems. Try to do this, to learn more about other industries, but also to learn and understand the high-tech within their own industry, so that you can

Program in a language, but don't let it bind your mind. In a code book says: "In-depth programming in a language, do not float on the surface." In-depth development of a language is far from enough, the existence of any programming language has its own reasons, so no language is a "panacea". Examples abound of the impact and constraints that programming languages have on the way developers think and solve specific problems.

develop the habit of summary and reflection, and consciously refine the results of daily work, form their own personal source code library, solve a type of general system architecture, and even evolve into a framework. As we all know, for software developers, there is a significant difference between inexperienced and experienced: inexperienced people complete any task from scratch, while experienced people often solve problems by reorganizing their reusable modules and libraries (in fact, this conclusion should not be limited to the field of software development, can be extended to many aspects). This is not to say that everything that can be reused must be implemented on its own, and the mature and tested results of others can also be collected, organized, and integrated into their own knowledge base. However, it is best to achieve their own, so that there is no intellectual property rights, copyright and other issues, the key is to truly master this knowledge point after their own implementation, have this skill.

Pay equal attention to theory and practice, both inside and outside. The connotation of engineer is: to observe and analyze things and the world with the eyes of an engineer. A qualified software engineer is a person who truly understands the nature of software products and the essence of software product development (personal opinion). Mastering software development language, applying language tools to solve specific problems in work, and completing target tasks are the main tasks of software engineers, but from the perspective of software engineers, this is only external things, not important and essential work. To learn and master the theoretical knowledge and methodology of software product development, and to understand and apply the analysis, design and realization ideas of software products to solve specific problems of software product development in practice, is the real work of software engineers. Standing on the height of mature theory and reliable methodology, thinking, analyzing and solving

problems, and verifying and revising these ideas and ways in concrete practice, and finally forming their own theoretical system and practical methodology.

Don't limit your knowledge to just technical aspects. The research results of Professor Simon, a Nobel Prize winner in economics, show that: "For a person with a certain foundation, he can master any subject within six months as long as he is really willing to work hard." The educational psychology community thanks Professor Simon's research results, so named Simon learning method. It can be seen that mastering a strange knowledge is far from as difficult and profound as I think. Absorb from many, widely dabbled. Try to consolidate their circle of influence, as far as possible to expand their circle of attention. Financial, economic, tax, management and other knowledge, have time to take a look, bide your time, prepare for a rainy day.

What I learned the most and what impressed me the most was:

Don't become skilled unless your goal is to do so. Although this article is about suggestions for improving software development knowledge, being a master of technology is something I have always disagreed with. You can improve your expertise, but only if you are competent.

Improving software knowledge and technology is only the surface of the problem, the essence is to improve their understanding of the problem, analysis of the problem, to solve the problem of the ideological height. Many of the methods and principles of software expertise can be easily extended and applied to other aspects of life.

On the basis of being able to work, immediately dabble in other fields of professional knowledge, enrich their knowledge system, improve their overall quality, especially those who are not the goal of technical friends.

LIST OF REFERENCES

1. 11th revision of the International Statistical Classification of Diseases and Related Health Problems, [Electronic resource] – Mode of access:
<https://icd.who.int/en/>
2. World Health Organization. Classifications, [Electronic resource] – Mode of access:
<http://www.who.int/classifications/icd/en/>
3. International Statistical Classification of Diseases and Related Health Problems (ICD), [Electronic resource] – Mode of access:
<https://www.who.int/standards/classifications/classification-of-diseases>
4. Research on heterogeneous data integration based on JSON, [Electronic resource] – Mode of access:
<http://cdmd.cnki.com.cn/Article/CDMD-10673-1017215915.htm>
5. Research on MySQL database protection technology based on security agent [Electronic resource] – Mode of access:
https://link.zhihu.com/?target=https%3A//www.zhangqiaokeyan.com/academic-degree-domestic_mphd_thesis/020316312346.html%3Ffrom%3Dlzh-0-2-1-4p-29499
6. Design and implementation of JAVA programming language examination system based on WEB, [Electronic resource] – Mode of access:
https://www.zhangqiaokeyan.com/academic-degree-domestic_mphd_thesis/020312862068.html
7. Web front-end development technology and optimization strategy based on website production, [Electronic resource] – Mode of access:
services/services/http://qikan.cqvip.com/Qikan/Article/Detail?id=7102757884&from=Qikan_Article_Detail
8. Research and application of web front-end performance optimization, [Electronic resource] – Mode of access:
<https://xueshu.baidu.com/usercenter/paper/show?paperid=af39f22139f02747b1a544cda602a333>

9. From numerical optimization to learning optimization, [Electronic resource] – Mode of access:

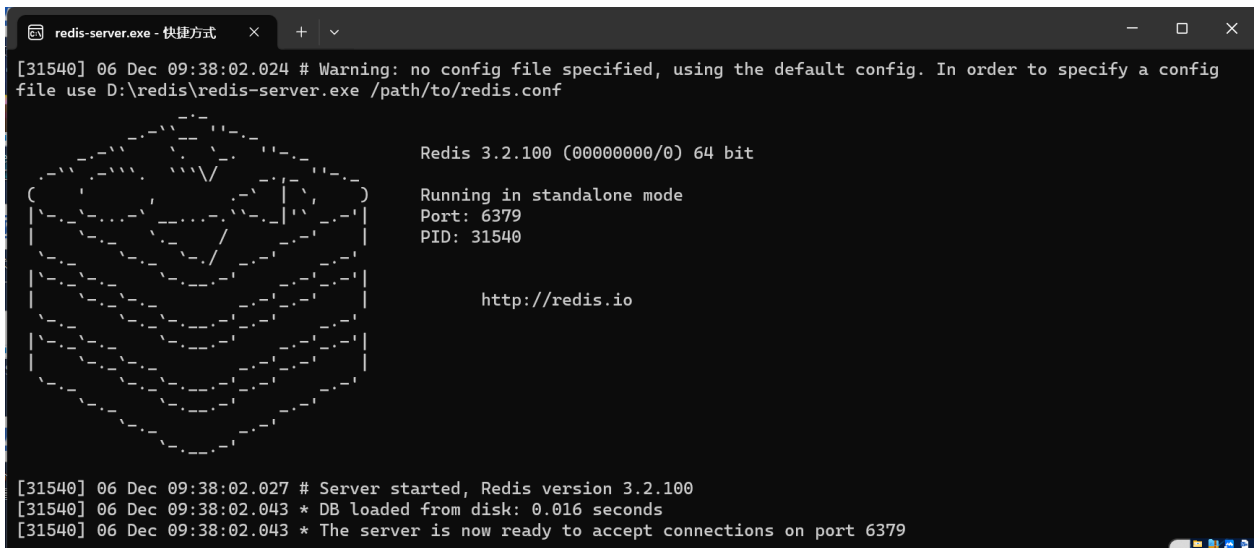
https://link.zhihu.com/?target=https%3A//www.zhangqiaokeyan.com/academic-journal-cn_operations-research-transactions_thesis/0201277589075.html%3Ffrom%3D01-005-04-19475

10. Review and summary of various Optimizer gradient descent optimization algorithms, [Electronic resource] – Mode of access:

<https://zhuanlan.zhihu.com/p/343564175>

ADDITIONAL PICTURES

This step is to start the SEVER before starting the system through IDEA



```

redis-server.exe - 快捷方式
[31540] 06 Dec 09:38:02.024 # Warning: no config file specified, using the default config. In order to specify a config
file use D:\redis\redis-server.exe /path/to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit

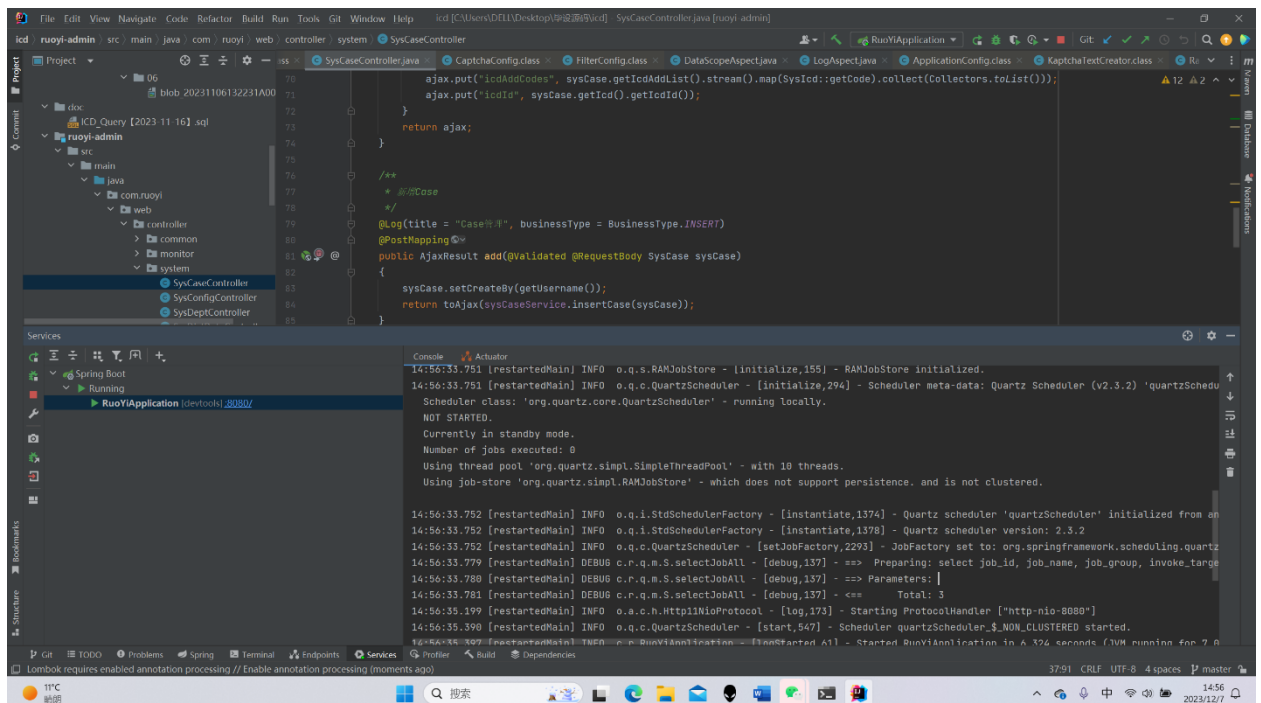
Running in standalone mode
Port: 6379
PID: 31540

http://redis.io

[31540] 06 Dec 09:38:02.027 # Server started, Redis version 3.2.100
[31540] 06 Dec 09:38:02.043 * DB loaded from disk: 0.016 seconds
[31540] 06 Dec 09:38:02.043 * The server is now ready to accept connections on port 6379

```

The following are the steps and procedures for launching the backend and frontend code. When the progress bar loads to 100%, it starts successfully, but I added an emoticon in order to see if it starts successfully in the first place. If the emoticon can appear on the device, it can start, otherwise it can't start, so I need to make a small adjustment to the application



```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Idea [C:\Users\DELL\Desktop\项目\ruoyi-admin\src\main\java\com\ruoyi\web\controller\system\SysCaseController.java [ruoyi-admin]
Project
  06
  blob_20231106132231A00
  dxs
  ICD_Query [2023 11 16].sql
  ruoyi-admin
    src
    main
      java
        com.ruoyi
          web
            controller
              common
              monitor
              system
                SysCaseController
                SysConfigController
                SysDeptController
  SysCaseController.java
  CaptchaConfig.class
  FilterConfig.class
  DataScopeAspectJava
  LogAspectJava
  ApplicationConfig.class
  KaptchaTextCreator.class
  ajax.put("icdAddCodes", sysCase.getIcdAddList().stream().map(SysIcd::getIcdCode).collect(Collectors.toList()));
  ajax.put("icdIcd", sysCase.getIcd().getIcdId());
  }
  return ajax;
  }
  /**
   * 添加病例
   */
  @Log(title = "添加病例", businessType = BusinessType.INSERT)
  @PostMapping
  public AjaxResult add(@Validated @RequestBody SysCase sysCase)
  {
    sysCase.setCreateBy(getUsername());
    return toAjax(sysCaseService.insertCase(sysCase));
  }
}
Services
  Spring Boot
  Running
  RuoYiApplication [devtools:8080]
Console
  14:56:33.751 [restartedMain] INFO o.q.s.RAMJobStore - [initialize,155] - RAMJobStore initialized.
  14:56:33.751 [restartedMain] INFO o.q.c.QuartzScheduler - [initialize,294] - Scheduler meta-data: Quartz Scheduler (v2.3.2) 'quartzSchedu
  Scheduler class: 'org.quartz.core.QuartzScheduler' - running locally.
  NOT STARTED.
  Currently in standby mode.
  Number of jobs executed: 0
  Using thread pool 'org.quartz.simpl.SimpleThreadPool' - with 10 threads.
  Using job-store 'org.quartz.simpl.RAMJobStore' - which does not support persistence, and is not clustered.
  14:56:33.752 [restartedMain] INFO o.q.i.StdSchedulerFactory - [instantiate,1374] - Quartz scheduler 'quartzScheduler' initialized from an
  14:56:33.752 [restartedMain] INFO o.q.i.StdSchedulerFactory - [instantiate,1378] - Quartz scheduler version: 2.3.2
  14:56:33.752 [restartedMain] INFO o.q.c.QuartzScheduler - [setJobFactory,2293] - JobFactory set to: org.springframework.scheduling.quartz
  14:56:33.779 [restartedMain] DEBUG c.p.q.m.s.selectJobAll - [debug,137] - ==> Preparing: select job_id, job_name, job_group, invoke_tange
  14:56:33.780 [restartedMain] DEBUG c.p.q.m.s.selectJobAll - [debug,137] - ==> Parameters: []
  14:56:33.781 [restartedMain] DEBUG c.p.q.m.s.selectJobAll - [debug,137] - <== Total: 3
  14:56:35.199 [restartedMain] INFO o.a.c.h.Http11NioProtocol - [log,173] - Starting ProtocolHandler ["http-nio-8080*"]
  14:56:35.390 [restartedMain] INFO o.q.c.QuartzScheduler - [start,547] - Scheduler quartzScheduler_$_NON_CLUSTERED started.
  14:56:35.390 [restartedMain] INFO o.a.c.h.Http11NioProtocol - [start,547] - Started RuoYiApplication in 6.372 seconds (JVM running for 7.0
  Lombok requires enabled annotation processing // Enable annotation processing (moments ago)
  37/91 CRLF UTF-8 4 spaces master
  11°C
  14:56
  2023/12/7

```

```

package com.ruoyi.framework.aspectj;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

import java.util.Arrays;

/**
 * 操作日志记录处理
 *
 * @author ruoyi
 */
@Aspect
@Component
public class LogAspect {

    @Usage
    private static final Logger log = LoggerFactory.getLogger(LogAspect.class);

    /**
     * 排除敏感属性字段
     */
    @Usage
    public static final String[] EXCLUDE_PROPERTIES = { "password", "oldPassword", "newPassword", "confirmPassword" };

    /**
     * 计算操作耗时时间
     */
    @Usage
    private static final ThreadLocal<Long> TIME_THREADLOCAL = new NamedThreadLocal<>("Cost Time");

    /**
     * 处理请求前执行
     */
    @Before(value = "@annotation(controllerLog)")
    public void boBefore(JoinPoint joinPoint, Log controllerLog) {
        TIME_THREADLOCAL.set(System.currentTimeMillis());
    }

    /**
     * 处理请求后执行
     */
    @After(joinPoint)
}

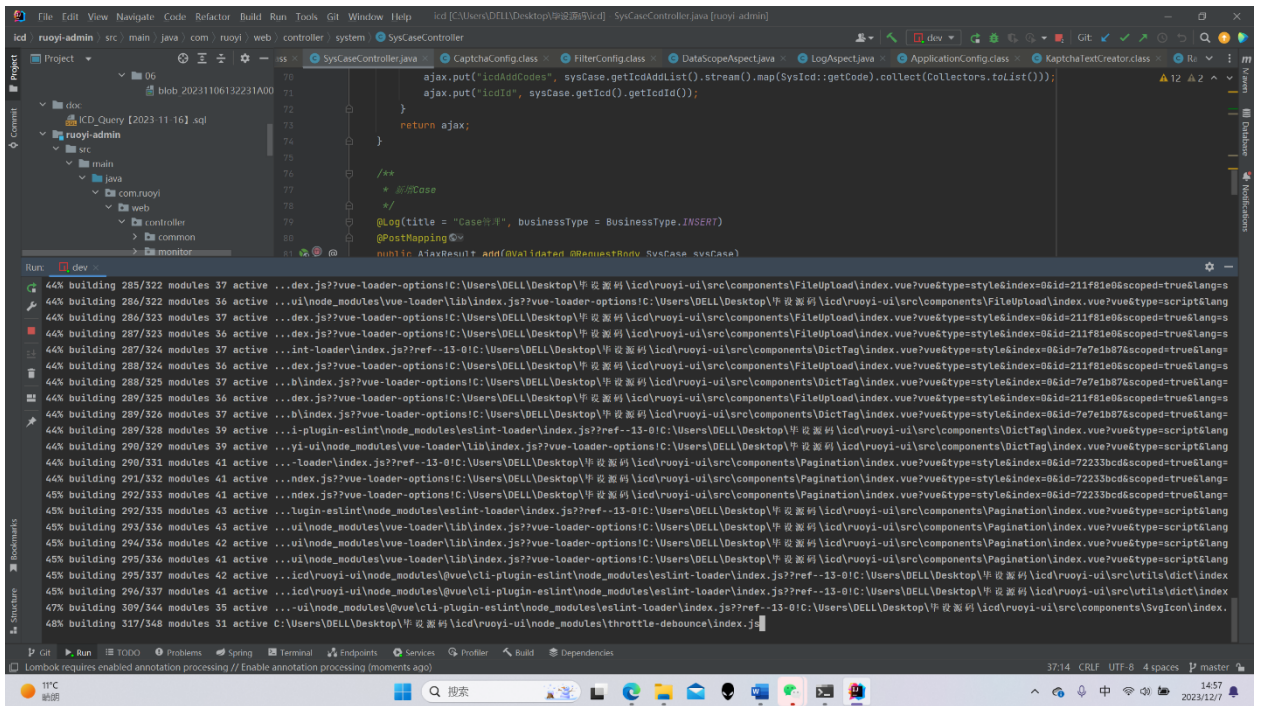
```

Services

```

09:42:54.319 [restartedMain] DEBUG c.r.q.m.s.selectJobAll - [debug,137] - ==> Preparing: select job_id, job_name, job_group, invoke_targe
09:42:54.328 [restartedMain] DEBUG c.r.q.m.s.selectJobAll - [debug,137] - ==> Parameters:
09:42:54.326 [restartedMain] DEBUG c.r.q.m.s.selectJobAll - [debug,137] - <== Total: 3
09:42:55.707 [restartedMain] INFO o.a.c.n.Http11NioProtocol - [log,173] - Starting ProtocolHandler ["http-nio-8080"]
09:42:55.915 [restartedMain] INFO o.q.c.QuartzScheduler - [start,547] - Scheduler quartzScheduler_$_NON_CLUSTERED started.
09:42:55.922 [restartedMain] INFO c.r.RuoYiApplication - [logStarted,61] - Started RuoYiApplication in 6.735 seconds (JVM running for 9.5
(♥~♥) ICD11启动成功 (☺☺☺)

```



APPENDIX B.

LINK TO SIMILAR ANALOGUES

Here are the functions implemented by other websites that I referred to when I was thinking about. I was inspired after reading their designs, so I optimized this function and implemented it on my own system

1. ICD11 Official browser

<https://icd.who.int/browse11/l-m/zh>

2. Pumch

<https://icd11.pumch.cn/>

3. Medsci

<https://www.medsci.cn/sci/icd-10.asp>

