Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 7, No. 3, 2024

Design and Implementation of a Primary Healthcare Cloud Platform

Haishan Wang *, Binbin Wang, Siyu Wang

University of Science and Technology Liaoning, Anshan, Liaoning, 114051, China

*** Corresponding author:** Haishan Wang (Email: 2650321241@qq.com)

Abstract: This paper presents the design and implementation of a primary healthcare cloud platform that aims to optimize primary medical services through informatization. The cloud platform integrates multiple functional modules including patient information management, diagnostic and treatment processes, pharmaceutical inventory, online appointments, financial management, and data analysis to meet the daily operational needs of primary healthcare institutions. Starting with the project background and current situation analysis, the paper details user needs research and functional design, and gives key discussions on the selection of system architecture and critical technology modules. During system development, technologies such as Spring Boot, Spring Cloud, and MySQL were used to build an efficient, stable, and secure platform. The system's reliability was ensured through unit testing, integration testing, system testing, and user acceptance testing. The paper concludes with a look into the future development direction of the platform, including further optimization of user experience, introduction of natural language processing and artificial intelligence technologies, and expansion of collaborative functions between primary healthcare institutions.

Keywords: Spring Cloud; Primary Healthcare; Cloud Platform; Shiro.

1. Introduction

The rapid development of information technology is driving profound changes in the medical service industry, especially for primary healthcare institutions, where informatization has become key to enhancing service quality and work efficiency. The Medical Cloud Platform [1-3] project was timely initiated with the goal of providing a comprehensive informatization solution for primary healthcare institutions. By constructing an integrated one-stop internet medical service system that connects clinics, doctors, and patients, the project aims to delve into and meet the needs of primary healthcare institutions.

The core purpose of this project is to address specific problems and challenges encountered by primary healthcare institutions during the informatization [4] process, in order to achieve seamless connection and convenience in medical service processes, and thereby comprehensively improve management quality and operational effectiveness. The project's technical architecture is modern and forward-looking, adopting a front-end and back-end separation design concept, and has implemented a stringent API authorization mechanism based on the Shiro framework, ensuring the security and stability of the system. Additionally, the project leverages Spring Cloud Alibaba to build a distributed architecture, taking into consideration the system's scalability and high concurrency processing capabilities, laying a solid foundation for future development.

The Medical Cloud Platform is more than just a project name; it symbolizes the solution to the challenges of primary healthcare informatization, representing the spirit of exploration and practical steps towards modernizing medical services.

2. Project Background

2.1. Industry Background

With the deepening application of information technology in the medical field, significant changes have occurred in various aspects such as clinical diagnosis and treatment, and medical management. Particularly, advancements in cloud computing and big data technologies have made medical informatization a key driving force for the development of the entire industry. Currently, the focus of medical informatization not only lies in the establishment of electronic medical record systems and the integration of medical data but is also gradually expanding to emerging areas such as optimization of medical resource allocation, remote medical services, and intelligent diagnostic assistance. at the primary healthcare institutions [5], informatization construction is still lagging. These institutions are often limited by resource shortages, a lack of professional talent, and limited service capabilities. The application of informatization technology has great potential to change this situation. By improving data processing efficiency, optimizing diagnostic and treatment processes, and achieving resource sharing, it can not only enhance service quality and work efficiency but also effectively improve the overall service level of primary healthcare institutions.

Nevertheless, primary healthcare institutions also face numerous challenges on the road to informatization construction, including insufficient integration of medical information systems with actual work processes, data silos, and a lack of targeted, customized service solutions. These issues have become significant bottlenecks in improving the quality of primary healthcare services.

2.2. Analysis of Existing Similar Systems

With the medical industry's growing demand for informatization technology, many medical information

systems have emerged on the market, such as Electronic Medical Records (EMR), Hospital Information Systems (HIS), and Personal Health Records (PHR). These systems have significantly optimized the storage and management of patient information and effectively improved the efficiency of medical management. In several large medical institutions, mature medical information products like Epic and Cerner have been widely adopted.

Although existing systems have promoted the management of medical information, the application of these systems in primary healthcare institutions has encountered multiple challenges. Most medical information systems are designed to meet the needs of large hospitals, and their complexity and high costs often exceed the capacity of primary healthcare institutions. Furthermore, the high standardization of systems limits their ability to meet the personalized needs of primary institutions.

Functionally, while existing systems can handle basic medical data, they fall short in advanced functions such as data analysis, decision support, and cloud services. Primary healthcare institutions urgently need a medical platform that can handle daily medical management tasks while supporting advanced data processing and remote services.

In terms of user experience, the complex operations and steep learning curve of current systems pose challenges for primary healthcare staff. Usability and accessibility have become their focus because these directly relate to work efficiency and patient satisfaction.

2.3. Expected Usage and Target User Group

The Medical Cloud Platform project aims to comprehensively enhance the informatization management level of primary healthcare institutions. Its primary function is to realize the electronic management of patient information, thereby improving the efficiency of data retrieval and processing. Additionally, the platform supports the automation of medical service processes, including online appointments, electronic prescriptions, and remote diagnoses, aiming to reduce the daily workload of medical personnel and improve overall service quality. Moreover, the built-in data analysis tools in the platform will assist doctors in making more accurate treatment decisions and provide powerful operational data support for administrators to optimize resource allocation and service processes.

The main target user group of this project is primary healthcare institutions, such as township health centers, community health service centers, and village health clinics. These institutions often face limitations in resources, experience in informatization construction, and technical support. In response to this situation, the Medical Cloud Platform particularly emphasizes the system's ease of use and affordability to ensure it can be widely adopted and rapidly popularized.

The scope of the Medical Cloud Platform is not limited to primary healthcare institutions; small and medium-sized medical entities such as individual clinics and specialized hospitals can also benefit. These users can utilize the platform's high flexibility and customization capabilities for personalized configuration according to their specific needs, without bearing the high costs and complex operations that traditional large medical information systems may bring.

3. Requirement Analysis

3.1. User Requirements

Primary healthcare institutions urgently need information management systems, seeking solutions that can enhance the efficiency and convenience of daily medical activities. They widely expect to achieve electronic management of patient data, medical records, and pharmaceutical inventory through a medical cloud platform, replacing traditional manual entry and reducing errors.

Automated workflows are one of the core user needs. For example, an automated appointment registration system can effectively alleviate crowded conditions within the facility, while electronic prescription and report generation systems simplify the daily workflow for doctors, optimizing medical service efficiency and patient experience. Moreover, with the widespread application of mobile internet technology, users increasingly expect to expand service coverage through remote medical services, especially providing more timely and convenient medical assistance to remote areas.

While managing patient information, data security and privacy protection are issues of great concern to users. Facing an increase in data breaches, users have set higher standards for the platform's ability to protect sensitive patient information. Therefore, strengthening data security measures and enhancing the system's protective capabilities are key aspects of the platform's design.

The platform's usability is directly linked to the work efficiency of medical personnel, especially for users with weaker technical backgrounds. Thus, users strongly expect the platform to have an intuitive interface and simple operations. This not only requires the platform to be highly user-friendly but also necessitates effective training and technical support from the project team to help users overcome technical barriers.

Meanwhile, primary healthcare institutions are usually subject to strict budget constraints, and users are particularly sensitive to cost-effectiveness. They hope that the costs incurred will bring significant improvements in operational efficiency and service quality, ensuring a reasonable return on investment. Therefore, the platform's pricing and service model design need to consider affordability, ensuring that users can obtain the services they need within their financial capacity.

3.2. Functional Requirements

The design requirements of the medical cloud platform include several key functions:

Patient Information Management: The platform should provide comprehensive patient information management tools, including the entry, retrieval, and editing of patient personal information, as well as electronic storage and management of medical history records. It also needs to support various data types, including text, imaging, and laboratory reports.

Diagnosis and Treatment Process Management: The platform needs to offer management of diagnostic and treatment activities such as doctor's appointment scheduling, diagnostic record filling, pharmaceutical prescription issuance, and treatment plan formulation. Additionally, it should support prescription printing and medical record export functions.

Pharmaceutical and Resource Inventory Management: The system should be able to monitor pharmaceutical

inventory in real time, providing expiration warnings and low inventory alerts, as well as recording the usage of medical supplies to optimize inventory management.

Online Appointment System: For the convenience of patients, the platform should integrate an online appointment system, allowing patients to book appointments via the internet or mobile devices and enabling medical staff to manage appointment schedules.

Financial Management: The platform needs to integrate financial settlement tools, including setting fee standards, settlement, invoice printing, and financial report generation, reducing the workload of financial and front desk personnel.

Data Analysis and Reporting: The platform should have data analysis capabilities to optimize service processes and improve medical quality by analyzing medical data, as well as providing operational reports for management decisions.

Data Security: Given the sensitivity of medical information, the platform must implement strict security measures, including access control, data encryption, security auditing, and data backup and recovery, to ensure data security and integrity.

By realizing these functionalities, the medical cloud platform will greatly improve the operational efficiency and service quality of primary healthcare institutions and ensure the security of sensitive medical data.

3.3. Performance Requirements

During the development of the medical cloud platform, performance requirements are directly related to system usability and end-user satisfaction. To this end, the platform must meet a set of predefined standards across several key dimensions.

Response Speed: A system with quick responses is crucial for medical institutions, significantly enhancing work efficiency and shortening patient waiting times. The medical cloud platform should ensure rapid data processing, page loading, and function execution capabilities, especially for commonly used operations such as patient information queries and pharmaceutical inventory checks, which should provide feedback within seconds.

System Stability: Considering the continuous nature of medical services, the platform needs to ensure high system availability and minimize system failures and maintenance time. Effective fault recovery mechanisms should be designed to ensure rapid return to normal operation in case of failures.

High Concurrency Processing: The platform should possess good scalability, especially when facing potential surges in access volume (e.g., during a pandemic). The system architecture needs to consider load balancing and dynamic resource allocation capabilities to ensure stable operation during spikes in user volume.

Security: Complying with data protection regulations (such as HIPAA or GDPR) is a basic requirement for medical information systems to strictly safeguard patient personal health information. This involves using data encryption technologies, implementing identity verification and access control, periodically conducting security audits, and vulnerability scanning.

Maintainability and Testability: The development team should be able to conveniently carry out system upgrades and maintenance, and the system should support automated testing to ensure stability and performance after updates.

The meticulous design of the medical cloud platform's performance requirements aims to provide users with a responsive, stable, and secure medical informatization solution, thereby improving medical service quality and patient satisfaction and ensuring the security of medical data. Based on this, the platform will regularly conduct performance and security assessments, continuously optimizing and enhancing system performance.

4. Design and Implementation

4.1. System Architecture

The system architecture of the medical cloud platform seeks clarity and a high degree of modularity in design to ensure that the medical information system operates stably and efficiently. The core design goals are stability and scalability, based on a distributed microservices architecture pattern. Different business logics are divided into independent service units with clear divisions of labor, which communicate and exchange data through predefined interfaces.

At the data level, a relational database is used to store key medical data, such as patient profiles, diagnostic results, and treatment records, with multiple backups and fault recovery mechanisms in place to ensure data consistency and security.

On the service level, each microservice is built around a specific business domain, such as user authentication, appointment management, and electronic medical record management. This design enhances the flexibility of development, testing, deployment, and expansion, as well as the maintainability of the system.

The interface layer serves as a unified entry point for system interaction with external entities, providing a standardized channel for front-end applications and third-party systems to access microservices. Additionally, the API gateway offers functions such as request routing, load balancing, and security control at this layer.

The front-end level includes web interfaces and mobile applications, responsible for providing an intuitive and friendly user interaction experience. Front-end applications implement business logic and user interaction by invoking the APIs of the interface layer.

In terms of performance, security, stability, and maintainability, the design and implementation of the system always prioritize these factors. The requirements are met through containerized deployment, continuous integration/ continuous deployment (CI/CD), and numerous security measures, such as data encryption, access control, and security audits.

System Architecture Diagram (Fig 1):

4.2. Key Module Design

In the construction of the medical cloud platform, the consultation module serves as a core component, responsible for handling a series of medical sub-services including appointments, diagnoses, and treatments. The following sections will elaborate on the design concepts, implementation methods, and key code of several main sub-modules.

4.2.1. Doctor Scheduling Sub-module

This module aims to create an efficient and flexible scheduling management system that allows doctors to view and manage their work schedules while giving administrative staff the ability to adjust schedules. The implementation of real-time update functionality ensures the accuracy and immediacy of scheduling information.

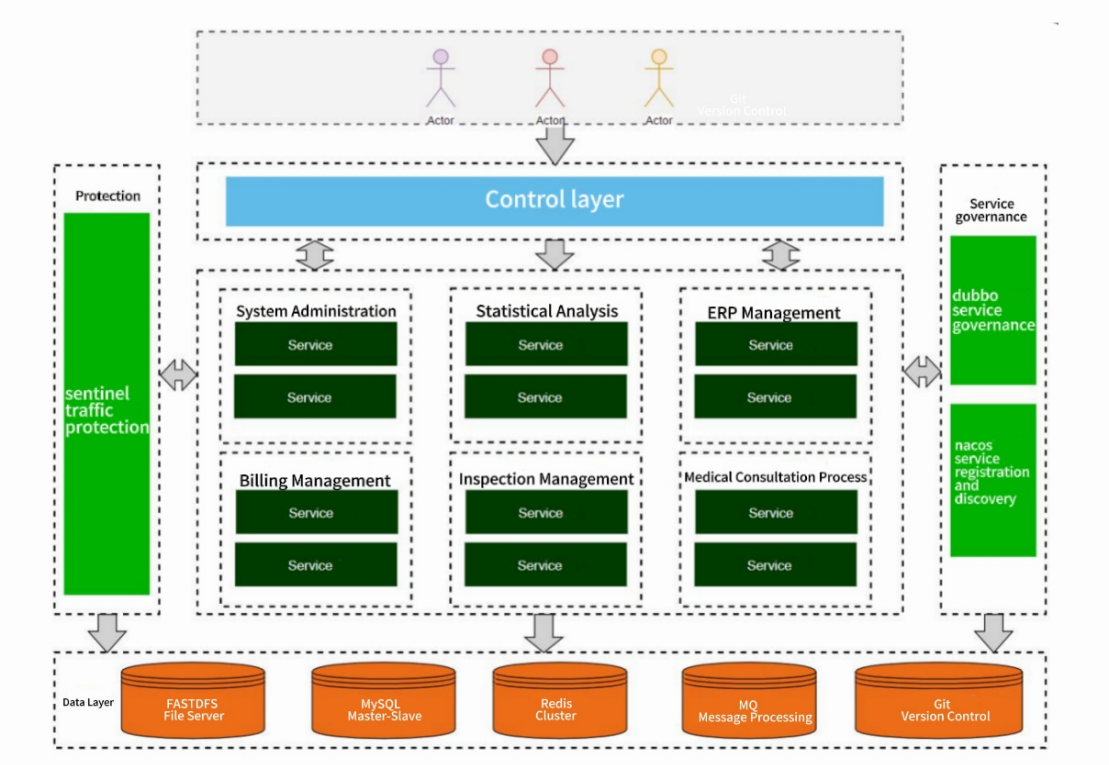

Fig 1. System Architecture Diagram

A RESTful service is built using the Spring Boot framework, with its built-in Tomcat server for server operations, and JPA for database operations, enabling persistent storage and retrieval of doctor scheduling information. Spring Security is used for access control to ensure that only authorized users can perform operations. Additionally, the introduction of transaction management ensures data consistency and integrity during scheduling, with data creation and updates managed by Spring's @Transactional annotation.

Key Code Example:

```
@RestController@RequestMapping("/doctor/scheduling")public class SchedulingController {
    private final SchedulingService schedulingService;

    @Autowired
    public SchedulingController(SchedulingService schedulingService) {
        this.schedulingService = schedulingService;
    }

    @Transactional
    @PostMapping("/saveScheduling")
    public ResponseEntity<?> saveScheduling(@RequestBody ScheduleDTO scheduleDTO) {
        schedulingService.saveSchedule(scheduleDTO);
        return ResponseEntity.ok().build();
    }}
```

4.2.2. Patient Information Management Sub-module

This module is designed to ensure the security, privacy, and usability of patient data. It encompasses the functions of entering, querying, editing, and deleting patient basic information while adhering to medical industry standards such as HIPAA. Patient data is stored in a MySQL database, with database operations simplified by Spring Data JPA. Multi-level security measures, such as database encryption, data anonymization, and role-based access control with Spring Security, ensure the protection of information [7].

Key Code Example:

```
@RestController@RequestMapping("/patient")public class PatientInfoController {
    private final PatientInfoService patientInfoService;

    @Autowired
    public PatientInfoController(PatientInfoService patientInfoService) {
        this.patientInfoService = patientInfoService;
    }

    @GetMapping("/{patientId}")
    public ResponseEntity<PatientDTO> getPatientInfo(@PathVariable Long patientId) {
        PatientDTO patientDTO = patientInfoService.getPatientInfo(patientId);
        return ResponseEntity.ok(patientDTO);
    }}
```

4.2.3. Diagnostic and Treatment Services Sub-module

This module focuses on providing efficient management of the diagnostic and treatment process, including symptom recording, diagnosis, treatment plan formulation, and prescription management. The design emphasizes the accuracy of medical records and the smooth communication of information between doctors and patients. It utilizes the Spring MVC architecture, combined with the Thymeleaf template engine and Hibernate ORM, to support user interface and database interactions. The integration of a rules engine allows for the formulation of treatment plans based on clinical guidelines.

Key Code Example:

```
@RestController@RequestMapping("/treatment")public class TreatmentController {
```

```
    private final TreatmentService treatmentService;

    @Autowired
    public TreatmentController(TreatmentService treatmentService) {
        this.treatmentService = treatmentService;
    }

    @PostMapping("/addRecord")
    public ResponseEntity<?> addTreatmentRecord(@RequestBody TreatmentRecordDTO recordDTO) {

treatmentService.addTreatmentRecord(recordDTO);
        return new ResponseEntity<>("Treatment record added successfully.", HttpStatus.CREATED);
    }}
```

4.2.4. Financial Settlement Sub-module

The financial settlement module is designed to automate the calculation and collection of medical fees, accurately account for service charges, and provide patients with convenient payment methods. The module implements complex fee calculation logic and processes financial computations using the Spring Framework. It integrates payment gateway APIs to offer patients a variety of payment options.

Key Code Example:

```
@RestController@RequestMapping("/finance")public class FinanceController {
    private final FinanceService financeService;

    @Autowired
    public FinanceController(FinanceService financeService) {
        this.financeService = financeService;
    }

    @GetMapping("/calculateBill/{patientId}")
    public ResponseEntity<BillDTO> calculateBill(@PathVariable Long patientId) {
        BillDTO billDTO = financeService.calculateBill(patientId);
        return ResponseEntity.ok(billDTO);
    }}
```

4.3. Database Design

Table 1. Patient Form

Field Name	Data Type	Constraints
PatientID	INT	PK
FullName	VARCHAR (255)	NOT NULL
Gender	VARCHAR (10)	
DateOfBirth	DATE	
Address	VARCHAR (255)	
ContactNumber	VARCHAR (15)	
InsuranceDetails	VARCHAR (255)	

In the medical cloud platform project, the key to the database design of the consultation module is the efficient storage and processing of data during the patient's medical process. Here are the design details for the key data tables in this module:

Table 2. Appointment Form

Field Name	Data Type	Constraints
AppointmentID	INT	PK
PatientID	INT	FK
DoctorID	INT	FK
ScheduledDate	DATE	
ScheduledTime	TIME	
AppointmentType	VARCHAR (50)	
tatus	VARCHAR (50)	

Table 3. Doctor Form

Field Name	Data Type	Constraints
DoctorID	INT	PK
FullName	VARCHAR (255)	NOT NULL
Specialization	VARCHAR (50)	
ContactNumber	VARCHAR (15)	
Qualifications	TEXT	

Table 4. Medical Record Form

Field Name	Data Type	Constraints
RecordID	INT	PK
AppointmentID	INT	FK
Symptoms	TEXT	
Diagnosis	TEXT	
PrescribedTests	TEXT	
TreatmentPlan	TEXT	
FollowUpRequired	BOOLEAN	

Table 5. Prescription Form

Field Name	Data Type	Constraints
PrescriptionID	INT	PK
RecordID	INT	FK
MedicationName	VARCHAR (100)	NOT NULL
Dosage	VARCHAR (50)	NOT NULL
Frequency	VARCHAR (50)	NOT NULL
Duration	INT	NOT NULL

In these tables, 'PK' stands for Primary Key, which ensures the uniqueness of each record in the table; 'FK' stands for Foreign Key, which is used to link data between tables. For example, in the 'Appointment' table, 'PatientID' acts as a

foreign key and connects to the 'PatientID' in the 'Patient' table, ensuring that appointment information is correctly associated with the specific patient.

Table 6. Financial Record Form

Field Name	Data Type	Constraints
BillingID	INT	PK
AppointmentID	INT	FK
TotalCharges	DECIMAL (10,2)	NOT NULL
PaymentStatus	VARCHAR (50)	
PaymentMethod	VARCHAR (50)	

For query performance optimization, appropriate indexes will be set on each table. For instance, a composite index based on 'ScheduledDate' and 'DoctorID' will be created on the 'Appointment' table to quickly retrieve doctor scheduling information for a specific date. Sensitive information will be encrypted and stored, and the database will be regularly backed up and maintained to ensure the security of the data and the high availability of the system.

4.4. Technology Selection

For backend service development, this project adopts the Spring Boot and Spring Cloud frameworks. Spring Boot greatly simplifies the development process for Spring-based applications through its auto-configuration feature and significantly reduces the complexity of configuration and deployment due to its standalone running nature. Spring Cloud provides a suite of microservices-related tools, such as service discovery, configuration management, and load balancing. Its deep integration with Spring Boot facilitates the construction and management of a microservices architecture.

In terms of data persistence, MySQL is chosen as the primary relational database system. Its mature stability, open-source freeness, and broad community support fully meet the project's requirements for data storage and management. Additionally, to optimize system performance, Redis is used as a caching and session storage solution to enhance data retrieval speed and user experience.

The development of the frontend application is based on the React framework. Its component-based construction method, efficient DOM rendering capabilities, and rich ecosystem make it possible to maintain and develop large single-page applications (SPAs). At the same time, the project also integrates the UI component library of Ant Design, which speeds up the interface construction process and ensures interface consistency and aesthetics.

On the deployment and operation level, the project chooses Docker container technology and Kubernetes container orchestration platform. Docker's containerization capability ensures the consistency of applications and their dependencies in different environments, while Kubernetes provides powerful functions for the automatic deployment, scaling, and management of container applications, significantly improving the reliability and scalability of applications.

Security, as a key consideration of the project, incorporates the Spring Security framework. Spring Security offers comprehensive security functions, including authentication and authorization, and can effectively withstand common security threats such as CSRF and XSS, ensuring the security of medical information system data.

5. Implementation Process

5.1. Development Environment Setup

For backend development, IntelliJ IDEA was selected as the integrated development environment due to its deep support for Java and its efficient features such as code auto-completion, refactoring tools, and integration with version control systems, which significantly improve development efficiency and code quality. Spring Cloud Alibaba was used to build the microservices architecture, simplifying the complexity of service governance and configuration management, and providing a solid support foundation for communication between microservices. The Shiro framework was integrated to strengthen permission control and ensure the security of system interfaces.

For databases, MySQL was chosen as the primary relational database management system. Its stability and maturity meet all the project's data storage needs. MySQL Workbench, with its intuitive user interface and feature set, greatly improved the efficiency of database operations.

The frontend development environment utilized the highly customizable Visual Studio Code. Its rich plugin ecosystem provided convenience for writing and testing Web development-related work, accelerating the frontend code development process.

5.2. Programming Practice

During the project construction, the team focused on the challenges of building a distributed microservices architecture, particularly the issues of inter-service communication and management. In an environment where multiple services work together, effectively managing service dependencies and communication is especially crucial. The use of the Spring Cloud Alibaba framework, with its built-in Nacos acting as the service registration and configuration center, ensured high availability and flexible configuration of services. Dubbo was used to implement efficient RPC remote calls between services, while Sentinel was responsible for traffic control and circuit breaking protection, collectively ensuring the stability and reliability of the system.

In terms of API security implementation, since the system needed to handle sensitive medical data and personal privacy information, ensuring the security of these data was of utmost importance. The introduction of the Shiro framework achieved fine-grained permission control and provided flexible security strategies. Faced with complex security requirements, the development team further strengthened the system's security defenses by customizing permission verification logic.

The issue of interface docking under a front-end and back-end separation architecture was efficiently resolved with the introduction of Swagger. Swagger not only automatically generated API documentation for the front-end and back-end but also provided an interface with intuitive testing functions, simplifying the debugging process and ensuring collaborative efficiency between front-end and back-end development.

As the project scale expanded, the complexity of the code and the difficulty of maintenance gradually increased. The introduction of Mybatis-plus effectively simplified the code of the data persistence layer, reducing a lot of repetitive work. At the same time, by writing unit tests and integration tests,

the team ensured high quality and robustness of the code, improving code maintainability and laying a good foundation for subsequent iteration upgrades.

In scenarios dealing with large volumes of data and high concurrency, the use of Redis as a caching method greatly reduced the pressure on the database. The application of RocketMQ for asynchronous message processing not only optimized the user experience but also improved the system's processing capability and response speed.

5.3. Function Implementation

The medical cloud platform project successfully created a comprehensive information solution for primary healthcare institutions. The system management module utilized the efficient support of the SpringBoot framework, combined with Shiro's security control and API authorization, to provide a solid management foundation for the entire medical information system. The inventory management module not only optimized the efficiency of medical material circulation but also effectively coped with high-concurrency data processing needs through the joint use of MySQL and Mycat middleware.

The consultation module, as the core of the system, handled key business processes such as registration and scheduling, using SpringCloud's microservices architecture and RocketMQ's message queues to achieve high availability and data consistency. The automated billing system in the charge management module provided patients with accurate and efficient cost calculations. The automated process of the examination management module further improved the efficiency of medical operations.

The data statistics module implemented rapid and in-depth data analysis through the application of Redis and Elasticsearch, providing strong data support for medical decision-making. In terms of deployment and testing, Docker's containerization technology and Nginx's load balancing strategy significantly enhanced the system's concurrent processing capability and stability.

In conclusion, the medical cloud platform project integrated numerous advanced technologies, not only meeting the information needs of primary healthcare institutions but also playing an important role in improving the quality of medical services and work efficiency.

6. Testing and Feedback

6.1. Testing Methods

During the development of the medical cloud platform project, comprehensive testing was conducted to ensure the system's reliability and stability. The testing was divided into several stages: unit testing, integration testing, system testing, and user acceptance testing (UAT).

In the unit testing phase, each module's functionality was verified through writing test cases and using the JUnit framework for automated testing to ensure the code performed as expected. With the help of the Mockito framework to mock external dependencies, the accuracy and efficiency of the tests were ensured.

During the integration testing phase, the interactions and data flow between modules were verified using the Spring framework to simulate a real system environment, identifying and fixing issues that occurred during integration.

In the system testing phase, end-to-end tests were executed, simulating complete business scenarios, including exception handling and boundary condition testing. Selenium automation tools were used to simulate user operations to evaluate whether the overall system behavior met business requirements, and stress and load testing were conducted to verify the system's performance under high loads.

In the UAT phase, actual users were invited to participate, and the system functionality and user interface were adjusted based on user feedback. System adjustments were made to meet user needs and increase user satisfaction based on actual operations and feedback.

6.2. Test Results

Issues such as the strictness of permission verification and the completeness of data validation were identified during the testing process. These issues were resolved by refactoring related logic and strengthening data validation mechanisms. Integration testing revealed inconsistencies in API calls, which were fixed after unifying interface documentation and data exchange processes. During the system testing phase, database query optimizations were made to address performance bottlenecks, and indexing and read-write separation strategies were introduced to improve system processing capabilities. In UAT, the UI and operational processes underwent multiple optimizations based on user feedback, enhancing the system's usability.

6.3. User Feedback

During the UAT phase, users reported that the system significantly improved work efficiency, especially with the automated inventory management and billing calculation functions. The implementation of electronic health records was also well-received by medical staff. In response to user suggestions for improvements, such as adding reporting features and improving the user interface, the development team made corresponding system optimizations. A custom report generator was introduced, and the user interface design was improved to better align with medical professionals' operating habits. These adjustments made the medical cloud platform more aligned with actual workflows, earning widespread user approval.

7. Conclusion and Outlook

7.1. Project Summary

This medical cloud platform project has successfully built an efficient, stable, and user-friendly medical informatics solution. The platform has improved the efficiency of medical resource management, enhanced service quality, streamlined medical processes, reduced the risk of errors, and alleviated the workload of medical staff, allowing them to focus more on patient care. Additionally, the system's data statistics and reporting features have provided strong data support for medical decision-making.

However, the project also has some limitations and challenges. For instance, the system's performance in handling a large number of concurrent user requests, especially in terms of data synchronization and real-time updates, needs to be improved. Moreover, although the user interface is already user-friendly, there is still room for improvement in the intuitiveness and ease of operation of certain features. For medical personnel without a technical background, the learning curve for some advanced features of the system is steep, suggesting a need for further optimization of user experience and simplification of operation processes

in future versions.

From a long-term development perspective, the platform has already demonstrated its immediate value in business improvement and has laid the foundation for the future development of medical institutions. Nevertheless, with the continuous advancement of technology and the evolving industry demands, the system requires ongoing iteration and optimization to ensure its long-term adaptability and competitiveness. By integrating emerging technologies such as big data and artificial intelligence, the medical cloud platform has the potential to provide more in-depth and intelligent medical support.

7.2. Looking to the Future

The medical cloud platform will continue to enhance user experience and optimize the user interface and processes to make them more intuitive and user-friendly. It plans to introduce natural language processing technology to simplify user interactions. In terms of big data and artificial intelligence [6], it will integrate analytical tools and algorithms to offer disease prediction analysis and personalized treatment plans. The integration of mobile medical and remote monitoring devices will extend the coverage of medical services, particularly in home rehabilitation and chronic disease management.

At the same time, to promote collaboration between medical institutions, there is a plan to develop standardized interfaces to achieve the interoperability of medical data and improve resource utilization efficiency. With technological development, continuous education and training programs will be established to ensure that medical staff are proficient in new features, enhancing work efficiency.

References

[1] Xinyi Huang, Xiaoping Zhou, Zhongmin Wang. Design and Implementation of Maternal and Child Health Care Service Cloud Platform [J]. Journal of China Health Information Management, 2023, 20(05):814-818.

[2] Xuefeng Han, Fei Cheng. Design and Implementation of Hospital Medical Collaboration Cloud Platform [J]. Journal of China Health Information Management, 2023, 20(03):442-447.

[3] Zaixin Wu, Lichao Chen, Dapeng Chen, et al. Design and Implementation of Regional Medical Service Information Cloud Platform [J]. Medical Health Equipment, 2022, 43(01):40-44. DOI: 10.19745/j.1003-8868.2022009.

[4] Weikang Wang. Analysis on the Application of Network Technology in the Construction of Hospital Informationization [J]. Yangtze Information and Communication, 2023, 36(12): 143-145.

[5] Yanru Li, Lefei Zhang, Li Zhang, et al. Construction of Budget Monitoring Platform for Special Health Funds in Primary Medical Institutions: A Case Study of Socially Operated Medical Institutions in Futian District, Shenzhen [J]. Friends of Accounting, 2022(24):107-113.

[6] Yuan Xu, Lingfeng Wan, Tong Yang, et al. Research and Construction of Clinical Big Data Mining and Artificial Intelligence Modeling Cloud Platform [J]. China Digital Medicine, 2022, 17(06):31-36.

[7] Jinan Shen. Research on Data Security Assurance Mechanism in Medical Cloud Environment [D]. Huazhong University of Science and Technology, 2022. DOI:10.27157/d. cnki. ghzku. 2019. 005231.