Healthcare Monitoring System using IoT and Cloud Computing

Handi Yan
College of Science, Methmatics and Technology
Wenzhou-kean University
Shijiazhuang, China
1308013@wku.edu.cn

Feige Fang
College of Science, Methmatics and Technology
Wenzhou-kean University
Ji'an, China
1308214@wku.edu.cn

Kehan Zhu
College of Science, Methmatics and Technology
Wenzhou-kean University
Hangzhou, China
1306119@wku.edu.cn

***Abstract*—The project develops a healthcare monitoring system using IoT device and cloud computing technique. The system enables patients to check their real-time vital signs, and allows the doctors to receive medical alerts, which are computed through data collected from patients.**

Keywords—healthcare monitoring, internet of things (IoT), cloud computing, vital signs, medical alerts, health data analysis, wearable devices

I. INTRODUCTION

A. Background Information and Motivation

Nowadays, monitoring healthcare is facing challenges from the aging society, demands of telemedicine, and the threat of pandemic.

By 2050, there will be 2 billion people aged 60 or over, accounting for 21% of the world's population [1]. The huge burden of taking care of older adults is transferred to the medical professionals. To ease that burden, programmed healthcare monitoring systems are promoted and developed by scientist.

The issue of uneven will be a lifelong problem to the humanity. The uneven of medical resources give birth to the telemedicine. The top ranked doctors could diagnose patient who is far away by the use of health care monitoring system.

After the pandemic, the governments all over the world attach importance to handling the threaten of infectious disease. To solve the most crucial part of curing infectious disease, the healthcare monitoring systems is carried out to achieve non-contact therapeutic process.

B. Project objective

This project aims to develop a healthcare monitoring system. Through the system, the vital health signs from patients can be monitored, and the health alerts can be sent to healthcare professionals. To develop the healthcare monitoring system, the Internet of Things (IoT) technique is used to interconnect devices that collect vital signs from wearer. The signs are computed and monitored through cloud computing technique. Through the cloud computing, the medical alerts can be sent to doctors who make treatment decisions.

II. REQUIREMENTS

A. Functional Requirements

The healthcare monitoring system must provide services are as followed.

- The IoT device shall collect real-time vital signs from wearer.
- The IoT device shall transmit the collected data securely to the cloud server.
- The system shall store the data collected in the cloud database.
- The system shall compute the data collected and detect abnormal signs changes.
- The system shall send an alert email to the corresponding doctor when an abnormal sign is detected.
- The system shall provide a login page for the user, admin, and doctor.
- The system shall provide a web page for the user to view their real-time vital signs, and the daily variation diagram of the signs.
- The system shall allow the user to upload their physical report, including gender, height, history of chronic diseases, and other relevant health data.
- The system shall allow the admin to change the responsibility relationship between doctors and patients.
- The system shall allow the admin to add and delete patients and doctors.
- The system shall allow the doctors to check the real-time vital signs, and daily variation diagram of their patients.
- The system shall send an alert email to the doctor when there is an abnormal change of the vital signs of the patient.

According to the must provide services, the use case diagram are drawn.

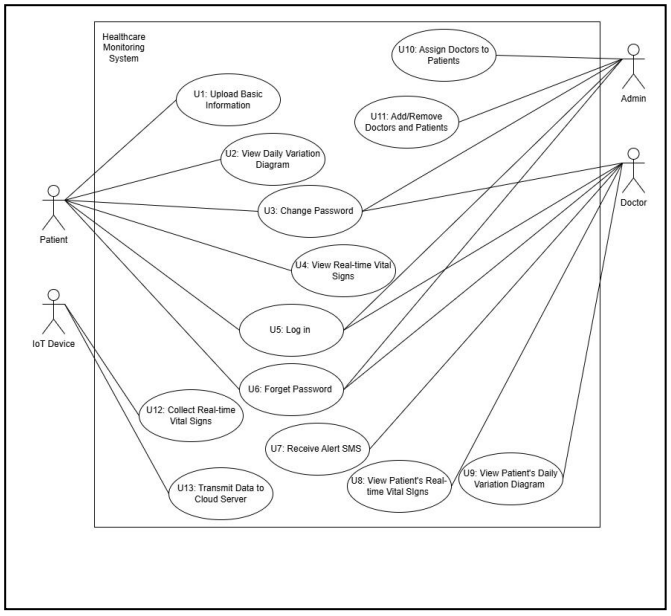

Fig. 1. Use case diagram of health care monitoring system

According to the use case 7, the SMS messages receiving function between the doctor and the healthcare monitoring system can be shown as a communication diagram.

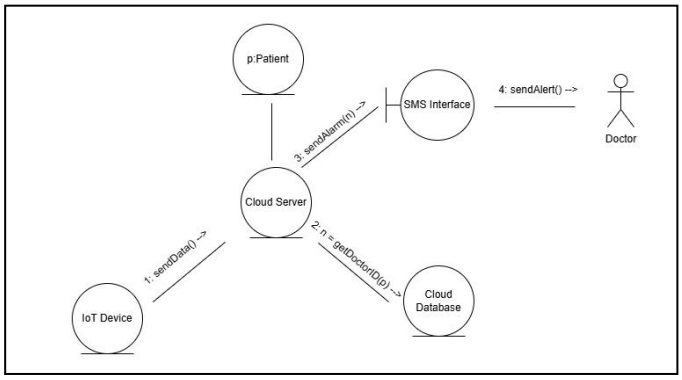

Fig. 2. Communication diagram for use case 7

The crucial activity that patients check their vital signs is picked and displayed as an activity diagram as followed.

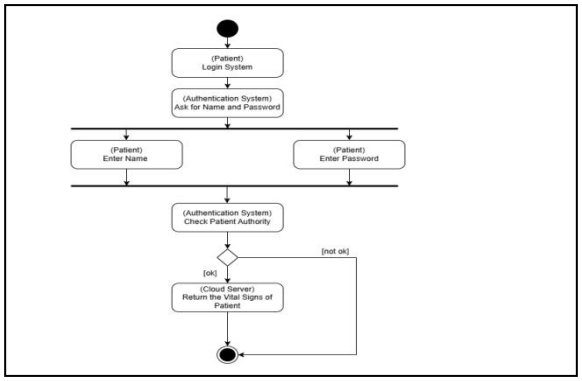

Fig. 3. Activity diagram for patient checking their vital signs

B. *Nonfunctional Requirements*

Everything else that needs to be specified are as followed.

- The system shall update the vital signs data at intervals no longer than 10 seconds.
- The system shall protect the patient privacy. The access to the health data of patients should be limited only for patient themselves and the responsible doctor.
- The web interface shall support multiple browsers.

III. ANALYSIS

A. *Static Analysis*

Through the system requirements model, the candidate classes are found.

- Patient
- Doctor
- Admin
- VitalSign
- Altert
- Authentification
- HealthRecord
- IoTDevice
- CloudServer
- Account
- Association_Doctor_Patient
- HealthDashboard
- OpticalCharacterRecogniction (OCR)
- RiskEvaluator
- AuditLog

According to the candidate classes, the class diagram at analysis level is constructed.

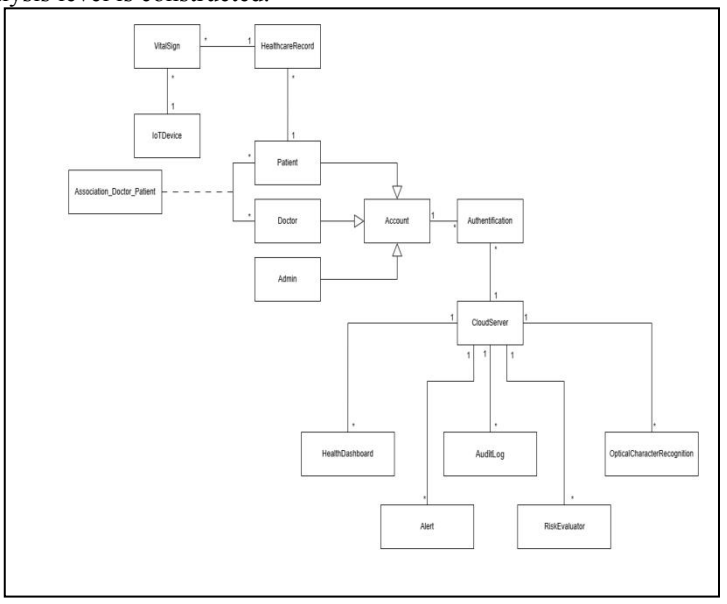

Fig. 4. Class diagram at analysis level for healthcare monitoring system

The class diagram shows what classes exits and how they are related.

According to Fig. 4, the Account class is the parent class of its child class Patient, Admin, and Doctor. An association class is created because it is existed based on the responsibility relationship between doctors and patients. A doctor could have any number of patients. Also, a patient could have any number of doctors.

The CloudServer class has association relationships between Authentification, HealthDashboard, Alert, RiskEvaluator, and OpticalCharacterRecogniciton (OCR) class. The CloudServer class could generate multipul request to these classes. So, the association relationships are marked as one to many relationships.

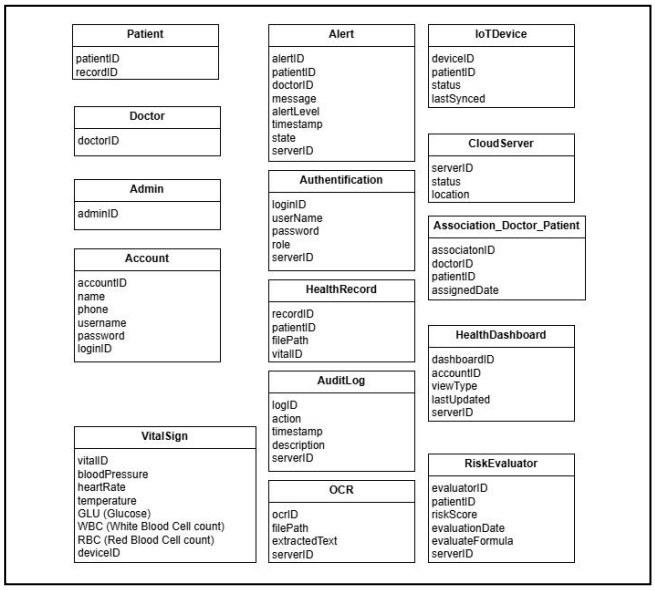

Fig. 5. Attributes for healthcare monitoring system

Besides the relationships between classes, the attributes of the classes have also been found. According to Fig. 5, the attributes example is present in a run-time object in Fig. 6.

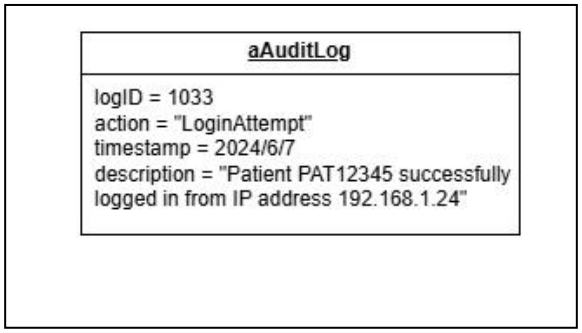

Fig. 6. An example of run-time object for healthcare monitoring system

B. *Dynamic Analysis*

Confirming the class diagram, a comprehensive dynamic analysis has been done in this project. And the process of doing dynamic analysis gives confidence to the project team.

To implement the most important part of dynamic analysis, the project team carefully conduct use case realization. By introducing the operations on the objects that receive messages, and adding classes that represent boundaries, the project team construct a communication diagram that reflect an important function of the healthcare monitoring system.

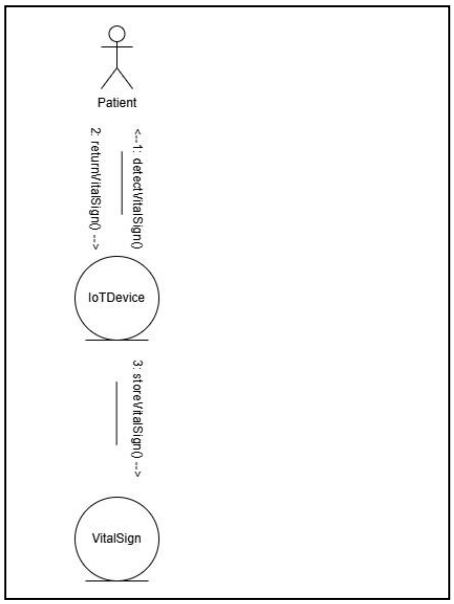

Fig. 7. Communication diagram for IoT device send data to Cloud Server

Based on Fig. 7, the communication class gives the insights for the operations in class.

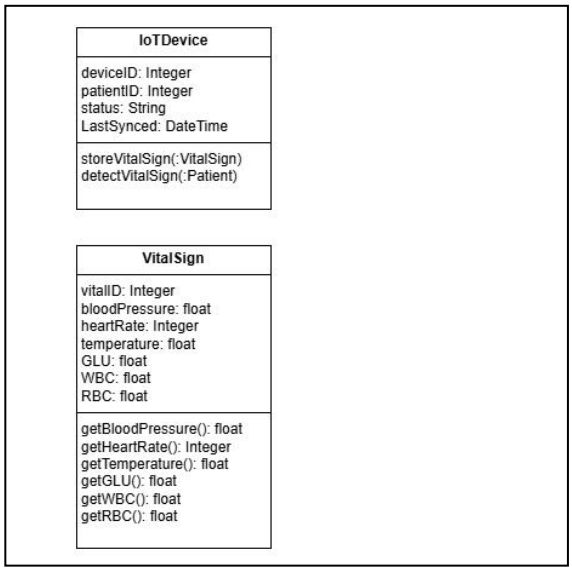

Fig. 8. Operations in classes

Because of the complexity of the entity alert's life cycle, and the importance of the entity alert's crucial role in the healthcare monitoring system, the state machine diagram is created to explain the working mechanism.

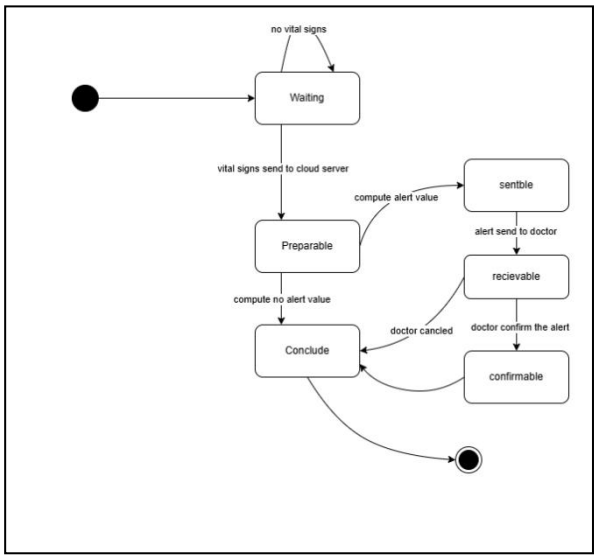

Fig. 9. State machine diagram for alert entity

IV. DESIGN

A. Designing the System and Choosing Techinology

1) Choosing A System Toplogy:

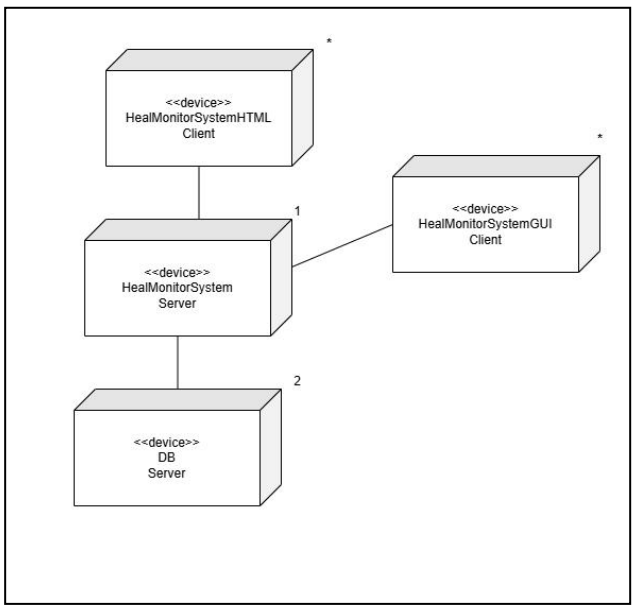

Fig. 10. A basic deployment diagram for Healthcare Monitoring System

This project adopts the classical three-tier architecture to build the network-based system, including client tier, middle tier, and the data tier.

The healthcare system comprises two database servers to guarantee the reliability of the system. The two database creates redundancy prepared for accidental damages of one of the database servers.

The middle tier functioning as communication bridge to the data tier is set to be one, due to the financial considerations.

The multiple client nodes represent the multiple amounts of users, such as doctors, patients, and admins, in the healthcare monitoring system.

2) Making Technology Choices:

The programming language used in this project is java. MySQL is chosen to be the database manage system (DBMS) technique. The communication protocol is set to be HTTP.

3) Designing A Concurrency Policy:

Based on the condition of intensive requests are sending to the system to receive and upload the vital sign data of the patient to the database. The system adopts multithreading policy. In case of concurrent requests, the health care monitoring system introduce queuing mechanisms or rate limiting to prevent system overload.

4) Designing A Security Policy:

The class variables are all set as private to prevent unauthorized access.

The privacy of the patient users is strictly protected. Irrelevant user account cannot get access to the private data, except patient themselves and the responsible doctor.

5) Partitioning the Subsystems into Layers:

The network layer and the server layer encapsulate the complex code implementation and provide simple interface for invoking functionality.

6) Deciding How Machines, Subsystems and Layers Will Communicate:

HTTP is adopted as the communication protocol between devices, while subsystems and layers communicate through well-defined interfaces.

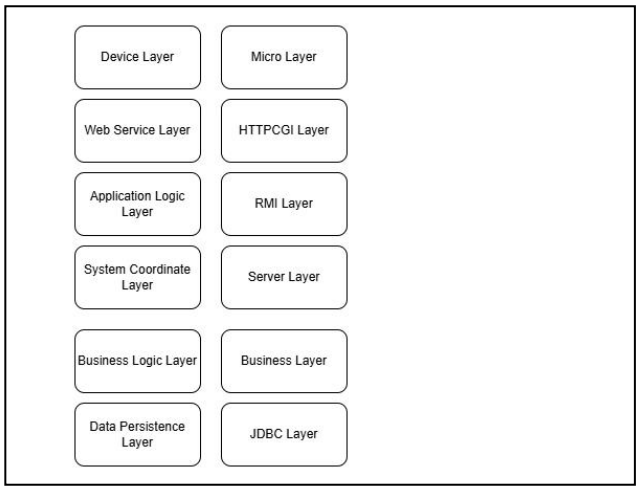

Fig. 11. Layers for healthcare monitoring system

B. Designing the Subsystems

After the analysis phase, the project has to map the conceptual analysis model into implementable classes.

1) Mapping Analysis to Design Class Mode:

To conduct system design, this project comprehensively designs the classes and the fields. A sample java code for the IoT device is presented as followed.

```java
public class IoTDevice{

private int deviceID;
private int patientID;
private String status;
private java.time.LocalDateTime lastSynced;

public void storeVitalSign(VitalSign v) {}
public void detectVitalSign(Patient p) {}
```

Fig. 12. An example of IoTDevice class java code

The data field of the IoTDevice class is all set to be private to protect data safety, and the privacy of the patient.

To access the private data, getters and setters are introduced in the java code of this healthcare monitoring system project.

```java
public class VitalSign{

private int vitalID;
private float bloodPressure;
private int hearRate;
private float temperature;
private float GLU;
private float WBC;
private float RBC;

public float getBloodPressure(){}
public int getHeartRate(){}
public float getTemperature(){}
public float getGLU(){}
public float getWBC(){}
public float getRBC(){}

}
```

Fig. 13. An example of VitalSign class java code

2) Middle Tier to Data Tier Technologies:

In the desing phase, this project will demonstrate its fasibility through mapping the one-to-many association relationship.

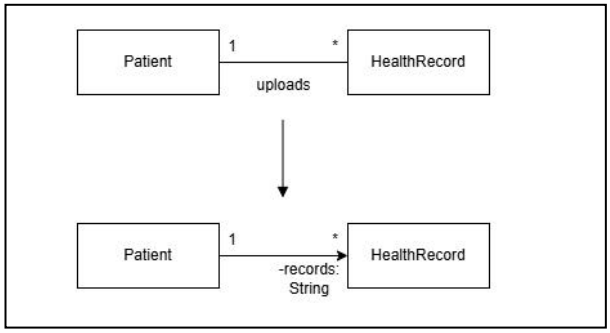

Fig. 14. Mapping one-to-many relationship

3) Universal Identifier:

All the classes in this project to be transformed into data frames are equipment with universal identifiers. The unique id that can function as primary key and foreign key plays an important role in database organizing. A sample diagram of the class that implements a universal identifier is presented as followed.

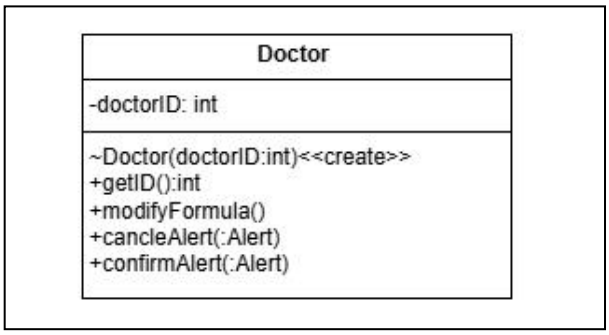

Fig. 15. Implementing a universal identifier

4) Handling Persistence with A Relational Database:

This project uses database management system named MySQL. To visualize the database structure, this project uses WampServer to construct the database structure sample. To make the project come to fruition, the Aphache web server which provides the database service will be deployed on the cloud computing platform.

The transformation from class diagrams to a table is shown as below.

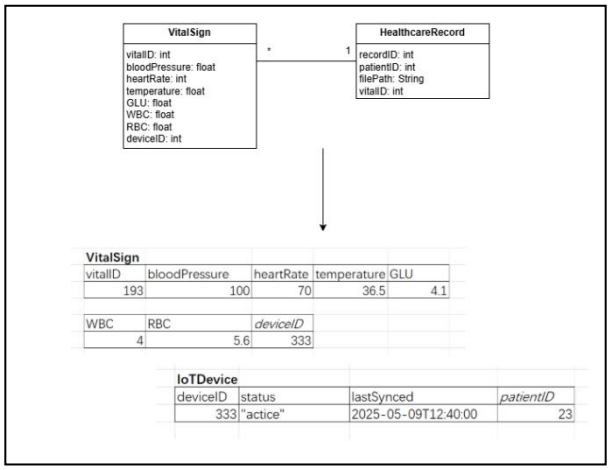

Fig. 16. Mapping class diagram to table

Through Fig. 10 to Fig. 14, this project demonstrates its feasibility on the design level.

V. SPECIFICATION

1) Objective: The healthcare monitoring system aims to dectect the vital signs from patients and collect their medical report. Through the programmed cloud computing platform, the system could detect abnormal vital signs and generate an alert. The alert will automatically send to the doctors who is responsible for the patient.

2) Target User: Patients who needs medical monitoring, and doctors.

3) System Range: Within the rage of the collaberative hospital.

4) System Interface: The system should support visiting from web and mobile terminal.

5) Data Flow Specification: The healthcare monitoring system collect patients' vital sign, such as, heart rate, blood pressure, temperature, GLU, WBC, and RBC, through IOT device. The data of vital sign is transmitted to the cloud computing platform through HTTP protacol. The cloud

computing platform will analyze the vital sign data by using the formula given and modified by the doctors. The abnormal value will trigger the alert. The alert will be sent to the doctor through SMS. And the doctor will confirm whether the alert is validate. The healthcare monitoring system is developed under the premise of minimizing the delay and the error of the system.

6) Doctor Function Specification: The doctor could modify the threshold of the alert according to the different health condition of the patient. Hence, the doctor could modify the alert evaluatoin formula.

VI. Implementation

After completing the system analysis and design phases, the project entered the implementation stage. In this phase, each module was developed according to the design models using appropriate programming languages, and integrated into a complete health monitoring system. The backend was developed using Java, while the frontend was built with HTML, CSS, and JavaScript. MySQL was used for data storage. The system was deployed on a cloud server and communicated via HTTP. Key technologies include OCR, multithreading, and a plugin-based architecture.

A. Module Implementation

1) IoT Data Collection Module

This module uses embedded sensor devices (e.g., smart wristbands) to collect vital signs such as heart rate, blood pressure, and body temperature in real time. The embedded programs were written in C and periodically upload data to the cloud platform using HTTP.

a) Data is collected every 10 seconds;

b) Each device is linked to a specific patient via a unique device ID;

c) Local caching mechanisms prevent data loss in case of network interruptions.

2) Data Upload and Cloud Processing Module

The server receives data from IoT devices, preprocesses it (e.g., noise removal and normalization), stores it in the MySQL database, and invokes the health evaluation module.

a) Implemented using Java Servlets or the Spring Boot framework;

b) Multithreading enables high-concurrency data handling;

c) Abnormal values trigger the alert system automatically.

3) User and Doctor Management Module

There are three types of users: patients, doctors, and administrators. Each has a dedicated login portal and access permissions.

a) OAuth 2.0 is used for secure user authentication;

b) Administrators can add/remove users and bind doctors to patients;

c) Doctors can customize evaluation models and thresholds for each patient.

4) Vital Sign Visualization and Report Module

Upon logging in, users can view real-time health data and visualized charts such as radar and line graphs.

a) Data visualization is implemented using ECharts or similar libraries;

b) Medical reports can be uploaded as images and processed via OCR for text extraction;

c) Users can view data by day, week, or month.

5) Alert Notification and Risk Evaluation Module

When a patient's vital signs exceed predefined thresholds, the system automatically generates alerts and notifies the assigned doctor via SMS or app notifications.

a) The alert system uses a strategy pattern for flexible evaluation logic;

b) Doctors can manually adjust thresholds and evaluation formulas;

c) Message queues (e.g., RabbitMQ) ensure reliable alert delivery.

6) Interface and Plugin System

The system uses interfaces and a plugin mechanism to support modular expansion and future integration of new health indicators or evaluation models.

a) The EvaluationPlugin interface defines a standard protocol for evaluation algorithms;

b) The NormalizationStrategy interface handles unit normalization for different indicators;

c) Plugins are dynamically loaded using Java reflection to ensure flexibility and maintainability.

VII. Testing

A. Functional Testing

The aim of functional testing is to verify whether the function of the system satisfies the use case expected, ensure smooth user interaction processes, and correct system response.

1) Upload physical examination report test: After users upload PDF/image files, can the system correctly call the OCR module to extract key information.

2) Health physical examination test: After adding data, can system correctly call RiskEvaluator module, return the expected score and generate a radar chart.

3) Warning notification triggering test: When the indicator exceeds the threshold, can the system notify users in a timely manner through SMS/APP based on the warning level.

4) Doctor adjustment model test: After the doctor modifies the evaluation model, the system recalculates the user's historical data and updates the results.

B. Unit Testing

Performing independence verification for core classes and methods.

1) RiskEvaluator module: Verify the logical accuracy of its scoring algorithm.

2) AlertSystem module: Verify whether the warning types triggered by different thresholds are classified correctly.

3) Interface verification: Verify whether the interface, such as NormalizationStrategy, EvaluationPlugin, executed correctly.

4) Exception module: Verify whether AuditLog can properly record abnormal information.

C. Integration Testing

Simulate users using an intelligent blood pressure monitor to verify whether the system can collect relevant data and upload it, while correctly identifying hypertension and promptly pushing reminders.

D. Performance and Strss Testing

Verify whether the system completes parsing within 3 seconds when 2000 users upload reports simultaneously.

VIII. Deployment

After passing the system test, it enters the deployment phase. This system supports multiterminal use (APP, Web), ensuring service stability, security, and easy scalability.

The deployment diagram is created driven by actual situations.

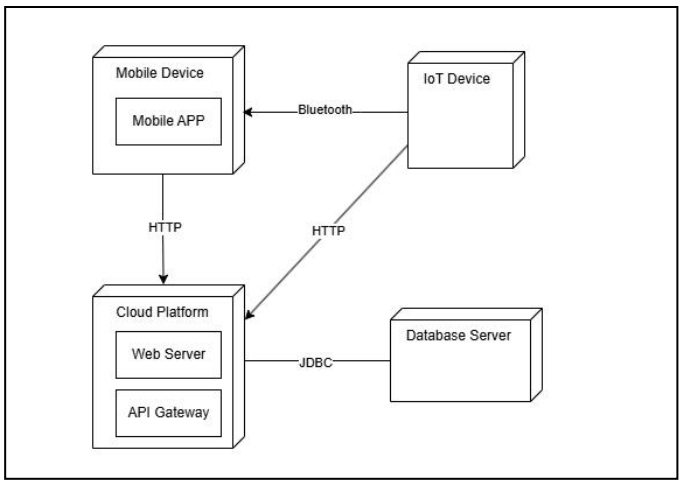

Fig. 17. Deployment diagram for healthcare monitoring system

A. System Deployment Phase

- The backend is deployed on cloud servers, using Docker containerization and CI/CD tools to automatically build and publish.
- Front end applications (web and apps) are deployed on Nginx servers and application markets, respectively.

B. Data Security Deployment

- Use the HTTPS protocol to ensure communication security.
- Use the OAuth 2.0 authentication mechanism for user authorization, and make sure sensitive medical data is encrypted and stored.
- Set up a database access audit mechanism and permission control.

C. Third Party Device Access

Configure IoT data access API to support standard Bluetooth/HTTP protocols.

Enable local caching mechanism in abnormal situations (such as network disconnection) to ensure that data is not lost within 24 hours.

D. Documentation and Training

Provide detailed system installation, configuration, and recovery manuals.

At the same time, provide model configuration and evaluation interpretation training for doctors, ensure administrators can manage user permissions, and maintain indicator libraries.

IX. Maintenance

After the system is put into use, maintenance work becomes a key link to ensure that the system can operate stably for a long time. The maintenance phase covers defect repair, functional expansion, model optimization, and system monitoring.

A. Defect Repair

Configure an automatic exception logging system to immediately record errors such as OCR recognition failure, model misjudgment, and device data loss.

Collect user feedback through customer service and feedback modules, regularly evaluated and repaired.

B. Functional Expansion

Through the interface HealthIndicatorRegistry, dynamically support the addition of indicators.

Support the introduction of new risk assessment algorithms based on plugin architecture and integrate them by implementing the EvaluatePlugin interface.

C. Model Optimization

Doctors fine tune models for specific patients or groups, and the system automatically generates historical reports.

During long-term operation, model iteration and accuracy optimization are carried out based on the user's big data.

D. System Monitoring

The database is regularly backed up automatically to ensure quick recovery in the event of a disaster.

References

[1] S. Harper, "Economic and social implications of aging societies," *Science*, vol. 346, no. 6209, pp. 587–591, Oct. 2014, doi: https://doi.org/10.1126/science.1254405.