Technical Roadmap Document

System Team
April 26, 2025

Table of Contents

1. Document Overview

1.1 Document Purpose

This document defines the technical roadmap for each system component, clarifying the technology selection, implementation strategies, and collaboration specifications for the UI Team, System Team, Data Team, and Analysis Team. It ensures technical compatibility and development efficiency, providing a unified technical guidance framework for the team.

1.2 Project Background

A multi-role medical monitoring system supporting doctors, patients, and administrators to connect Bluetooth devices, collect data, perform real-time analysis, and visualize results via a web interface. Core processes include:

- The web interface collects Bluetooth device data and uploads it to the server;
- The System Team processes data and stores it in a MySQL database;

- The Analysis Team generates results through Java stream analysis;
- The UI Team visualizes and displays the analysis results.

1.3 Target Audience

Role	Focus Areas
Project Managers	Panoramic technical architecture and milestone planning
Developers	Technical stack details, interface specifications, module implementation
Testers	System interaction boundaries and testing strategies
Operations Engineers	Deployment architecture, monitoring, and high-availability solutions

2. System Architecture Overview

2.1 Layered Architecture Design

Layer	Responsible Team	Core Responsibilities	Key Technologies
Frontend Layer	UI Team	User interaction, Bluetooth connection, data visualization	Vue3, Web Bluetooth API, Element Plus
Service Layer	System Team	API gateway, service orchestration, security control, third-party integration	Spring Boot, RESTful API, gRPC
Data Layer	Data Team	Data modeling, storage, quality	MySQL, JDBC/HikariCP,

Layer	Responsible Team	Core Responsibilities	Key Technologies
		assurance	Flyway、python-numpy
Analysis Layer	Analysis Team	Real-time analysis, algorithm implementation, decision support	Java Stream API, modular data analysis classes

2.2 Core Data Flow

1. **Data Collection**: The UI Team retrieves device data via the Web Bluetooth API and sends it to the System Team through RESTful APIs.
2. **Data Processing**: The System Team receives data, stores it in MySQL (Data Team) via JDBC, and triggers stream analysis by the Analysis Team.
3. **Analysis and Presentation**: The Analysis Team returns results to the System Team, which are then fetched by the UI Team via APIs for visualization.

3. UI Team Technical Roadmap

3.1 Technology Stack

Technology	Version/Description	Use Case
Vue 3	Composition API	Reactive component development
Element Plus	Vue3-exclusive UI library	Rapid management interface construction
Web Bluetooth API	W3C Standard	Bluetooth device connection and data collection

Technology	Version/Description	Use Case
Vite	High-speed build tool	Development environment hot-reload and packaging
Pinia	Lightweight state management	Cross-component user state (e.g., role permissions)
Axios	API request encapsulation	Unified interface call and exception handling
Mock.js	Simulated data interface	Local development data mocking

3.2 Project Structure

bash

```
src/
├─── api/              # Interface encapsulation (Bluetooth, user, upload, etc.)
├─── components/     # Reusable components (tables, forms, modals)
├─── views/            # Page hierarchy
│     ├─── Admin/      # Administrator interface
│     ├─── Doctor/     # Doctor interface
│     ├─── Patient/    # Patient interface
│     └─── Bluetooth/ # Bluetooth connection and data collection page
├─── stores/           # Pinia state management (user roles, login status)
└─── mock/             # Development environment data mocking (only active locally)
```

3.3 Key Design

- **Bluetooth Communication**: Encapsulated BluetoothService class supporting device connection, disconnection reconnection, and timeout handling;
- **Permission Control**: Store role flags (e.g., isDoctor) via Pinia after login to

dynamically generate routes;

- **Data Mocking**: Intercept API requests with Mock.js to simulate login, data upload, etc., disabled in production.

4. System Team Technical Roadmap

4.1 Technology Stack

- **Core Framework**: Spring Boot (Java) for building RESTful APIs and service orchestration;
- **API Specification**: OpenAPI 3.0 (Swagger) for interface documentation and testing tools;
- **Monitoring & Logging**: Micrometer+Prometheus (metric monitoring), ELK Stack (log management);
- **Security Control**: Spring Security+JWT (token authentication), SSL/TLS for encrypted data transmission;
- **Third-Party Integration**: Encapsulated interfaces for Bluetooth device registration and data analysis service calls.

4.2 Core Functions

- **API Services**:
    - Support version control (e.g., /api/v1/sensor-data);
    - Provide interfaces for device management (registration/deletion), data reception (PUT), and result query (GET).
- **Data Flow**:
    - After receiving data from the UI Team, synchronously write to MySQL (Data Team) and asynchronously trigger analysis tasks for the Analysis Team;
    - Use connection pooling (HikariCP) to manage database connections for high concurrency.

5. Data Team Technical Roadmap

5.1 Database Selection

- **Engine**: MySQL Community Server 8.0+ (supports JSON, role management, mature community ecosystem);

- **Interaction Layer**:
    - **JDBC**: Direct SQL operations for complex queries and performance optimization, using HikariCP connection pooling;
    - **ORM (Alternative)**: Java JPA/Hibernate (simplified object mapping for rapid CRUD).

5.2 Data Management

- **Schema Migration**: Flyway (SQL script version control) or Liquibase (supports XML/YAML) to ensure schema consistency across environments;
- **Table Design**:
    - SensorReadings: Stores sensor data (timestamp, session_id, value), partitioned by time (e.g., annual partitions);
    - Session/Patient/Device: Foreign key associations to record session, patient, and device metadata;
- **Index Optimization**: Indexes on high-frequency query fields like session_id and timestamp.

5.3 Data Processing

- **Cleaning Logic**: Filter outliers and compensate for missing data at the application layer (Java/Python) to avoid complex database stored procedures;
- **Backup Strategy**: Regular mysqldump full backups, combined with binary logs for point-in-time recovery; use RDS snapshots in cloud environments.

6. Analysis Team Technical Roadmap

6.1 Technology Selection

- **Language**: Java (strong typing, multi-threading support for numerical calculations);
- **Data Structures**: ArrayList for raw data storage, HashMap for aggregated statistics;
- **Module Design**:
    - Sensor class: Encapsulates sensor data (angle, timestamp);
    - Action classes (e.g., Flexion/Extension): Calculate motion range for individual actions (max-min);

- AnalysisEngine class: Coordinates data input, invokes action class calculations, and generates statistical reports.

6.2 Core Process

1. Receive sensor data (54 data points) from the System Team;
2. Instantiate Sensor objects and preprocess negative angles (e.g., a < 0 ? a + 360 : a);
3. Invoke action classes to calculate motion ranges, storing results in ArrayList;
4. Perform statistical analysis (average, standard deviation, etc.) and generate JSON reports for the System Team.

6.3 Quality Assurance

- **Unit Testing**: Validate individual action class logic using JUnit 5;
- **Integration Testing**: Test the full data input-to-report generation flow with ⩾85% coverage;
- **Code Standards**: Follow Java naming conventions and comment algorithms for easy extension of new action types.

7. System Integration and Interaction

7.1 Inter-Component Protocols

Interaction Scenario	Protocol/Tool	Data Format	Characteristics
UI ↔ System	RESTful API/WebSocket	JSON	High reliability (WebSocket for real-time push)
System ↔ Data	JDBC	SQL	ACID transactions (data consistency)
System ↔ Analysis	gRPC/HTTP	Protobuf/JSON	Low latency (gRPC for stream processing)

Interaction Scenario	Protocol/Tool	Data Format	Characteristics
Data ↔ Analysis	JDBC	SQL	Batch data queries (support for historical analysis)

7.2 Asynchronous Processing

- **Message Queue**: Introduce Kafka/RabbitMQ between the System and Analysis Teams to decouple data reception and analysis tasks, supporting traffic peak shaving;
- **Transaction Compensation**: Retry mechanisms or manual intervention to ensure eventual consistency if analysis tasks fail.

8. Deployment Architecture

8.1 Development Environment

- **Tools**: IntelliJ IDEA (Java backend), VS Code (frontend), Git (version control);
- **Testing**:
    - Unit Testing: JUnit (Java), Vue Test Utils (frontend);
    - Integration Testing: Spring Test (backend), Selenium (end-to-end);
- **Local Deployment**: Docker Compose for MySQL, Redis, etc., Vite for local frontend runtime.

8.2 Production Environment

- **Server Architecture**:
    - **Application Layer**: Kubernetes cluster for System and Analysis Teams, Nginx load balancing with dynamic scaling;
    - **Data Layer**: MySQL cluster (master-slave replication) + Ceph distributed storage for high availability;
    - **Frontend Layer**: Static assets deployed to CDN, Web server (e.g., Nginx) for API request reverse proxy.
- **Monitoring System**:

- Application Metrics: Prometheus+Grafana (throughput, response time);
- Log Analysis: ELK Stack (error logs, operation auditing);
- Alerting: Alertmanager for real-time anomaly monitoring and notifications.

9. Risks and Challenges

9.1 Key Risks

Risk Item	Impact Level	Core Issue
Bluetooth Compatibility	High	Inconsistent browser support for Web Bluetooth API
Real-time Performance	Medium	Data processing latency and database pressure under high concurrency
Data Security	Extremely High	Unencrypted medical data transmission/storage, compliance risks (e.g., HIPAA)
Integration Complexity	Medium	High interface debugging costs due to multi-technology stack collaboration

9.2 Mitigation Strategies

- **Compatibility**: Establish device/browser whitelists and provide CSV import as an alternative;
- **Performance Optimization**: Replace native Java stream processing with Flink, enable database time-based partitioning, and query caching;
- **Security**: Use TLS 1.3 for transmission, encrypt sensitive fields during storage, and conduct regular penetration testing;
- **Integration Efficiency**: Define unified API specifications (OpenAPI), pre-test interfaces with Postman, and write integration test cases.

10. Conclusion

10.1 Core Principles

- **Layered Decoupling**: Isolate components via standard interfaces (REST/gRPC) to reduce dependency complexity;
- **Data Reliability**: Ensure consistency with MySQL ACID transactions and prevent data loss through backup strategies;
- **Observability**: Full-stack monitoring (metrics + logs + tracing) for rapid fault localization;
- **Scalability**: Modular design (e.g., extendable action classes in Analysis Team), 预留 database sharding and multi-tenant architecture interfaces.

10.2 Implementation Plan

- **2025 Q2**: Complete core module development for each team (Bluetooth connection, data storage, basic analysis algorithms);
- **2025 Q3**: End-to-end integration testing, optimize performance and security strategies;
- **2025 Q4**: Achieve compliance certification (e.g., HIPAA) and launch the production environment.

Note: Architecture diagrams can be generated using Draw.io (system architecture), Mermaid (data flow), or Cloudcraft (deployment topology).