系统技术路线文档

系统组
2025 年 4 月 26 日

目录

1. 文档概述

1.1 文档目的

本文档详细定义系统各组件的技术路线，明确 UI 组、System 组、Data 组、Analysis 组的技术选型、实现策略及协作规范，确保技术兼容性与开发效率，为团队提供统一的技术指导框架。

1.2 项目背景

构建多角色医疗监测系统，支持医生、患者、管理员通过网页端进行蓝牙设备连接、数据采集、实时分析与可视化。核心流程包括：

1. 网页端采集蓝牙设备数据并上传至服务器；
2. System 组处理数据并存储至 MySQL 数据库；
3. Analysis 组通过 Java 流式分析生成结果；
4. UI 组可视化展示分析结果。

1.3 读者对象

角色	关注重点
项目管理人员	技术架构全景、里程碑规划
开发人员	技术栈细节、接口规范、模块实现
测试人员	系统交互边界、测试策略
运维人员	部署架构、监控与高可用方案

2. 系统架构概述

2.1 分层架构设计

层级	负责组	核心职责	关键技术
前端层	UI 组	用户交互、蓝牙连接、数据可视化	Vue3、Web Bluetooth API、Element Plus
服务层	System 组	API 网关、服务编排、安全控制、第三方集成	Spring Boot、RESTful API、gRPC
数据层	Data 组	数据建模、存储、质量保障	MySQL、JDBC/HikariCP、Flyway、python-numpy
分析层	Analysis 组	实时分析、算法实现、决策支持	Java Stream API、模块化数据分析类

2.2 核心数据流程

1. **数据采集**：UI 组通过 Web Bluetooth API 获取设备数据，经 RESTful API 发送至 System 组。
2. **数据处理**：System 组接收数据，通过 JDBC 存储至 MySQL（Data 组），并触发 Analysis 组流式分析。
3. **分析与展示**：Analysis 组返回结果至 System 组，UI 组从 API 获取数据并可视化。

3. UI 组技术路线

3.1 技术栈

技术	版本 / 说明	应用场景
Vue 3	Composition API	响应式组件开发
Element Plus	Vue3 专属 UI 库	快速构建管理界面
Web Bluetooth API	W3C 标准	蓝牙设备连接与数据采集
Vite	极速构建工具	开发环境热更新与打包
Pinia	轻量级状态管理	跨组件用户状态（如角色权限）
Axios	封装 API 请求	统一处理接口调用与异常
Mock.js	模拟数据接口	本地开发数据 mock

3.2 项目结构

bash

```
src/
├── api/              # 接口封装（蓝牙、用户、上传等）
├── components/       # 公用组件（表格、表单、弹窗）
```

```
├── views/              # 页面分层
│   ├── Admin/          # 管理员界面
│   ├── Doctor/         # 医生界面
│   ├── Patient/        # 患者界面
│   └── Bluetooth/      # 蓝牙连接与数据采集页
├── stores/             # Pinia 状态管理（用户角色、登录态）
└── mock/               # 开发环境模拟数据（仅本地生效）
```

3.3 关键设计

- **蓝牙通信**：封装 BluetoothService 类，支持设备连接、断开重连、超时处理；
- **权限控制**：登录后通过 Pinia 存储角色标志（如 isDoctor），动态生成路由；
- **数据 mock**：通过 Mock.js 拦截 API 请求，模拟登录、数据上传等场景，生产环境自动禁用。

4. System 组技术路线

4.1 技术栈

- **核心框架**：Spring Boot（Java），构建 RESTful API 与服务编排；
- **API 规范**：OpenAPI 3.0（Swagger），提供接口文档与测试工具；
- **监控与日志**：Micrometer+Prometheus（指标监控）、ELK Stack（日志管理）；
- **安全控制**：Spring Security+JWT（令牌认证）、SSL/TLS 加密数据传输；
- **第三方集成**：封装蓝牙设备注册、数据分析服务调用接口。

4.2 核心功能

- **API 服务**：
  - 支持版本控制（如/api/v1/sensor-data）；
  - 提供设备管理（注册 / 删除）、数据接收（PUT）、结果查询（GET）等接口。
- **数据流转**：
  - 接收 UI 组数据后，同步写入 MySQL（Data 组）并异步触发 Analysis 组分析任务；
  - 使用连接池（HikariCP）管理数据库连接，确保高并发性能。

5. Data 组技术路线

5.1 数据库选型

- **引擎**：MySQL Community Server 8.0+（支持 JSON、角色管理，社区生态成熟）；
- **交互层**：
  - **JDBC**：直接操作 SQL，适合复杂查询与性能优化，使用 HikariCP 连接池；
  - **ORM（备选）**：Java JPA/Hibernate（简化对象映射，适合快速 CRUD）。

5.2 数据管理

- **模式迁移**：Flyway（SQL 脚本版本控制）或 Liquibase（支持 XML/YAML），确保多环境 Schema 一致；
- **表设计**：
  - SensorReadings：存储传感器数据（timestamp、session_id、value），按时间分区（如年度分区）；
  - Session/Patient/Device：通过外键关联，记录会话、患者、设备元信息；
- **索引优化**：对 session_id、timestamp 等高频查询字段建立索引。

5.3 数据处理

- **清洗逻辑**：在应用层（Java/Python）过滤异常值、补偿缺失数据，避免数据库存储过程复杂化；
- **备份策略**：定期 mysqldump 全量备份，结合二进制日志实现时间点恢复，云环境使用 RDS 快照。

6. Analysis 组技术路线

6.1 技术选型

- **语言**：Java（强类型、多线程支持，适合数值计算）；
- **数据结构**：ArrayList 存储原始数据，HashMap 聚合统计结果；
- **模块设计**：
  - Sensor 类：封装传感器数据（角度、时间戳）；

- 动作类（如 Flexion/Extension）：计算单一动作的运动幅度（最大值 - 最小值）；
- AnalysisEngine 类：协调数据输入、调用动作类计算、生成最终统计报告。

6.2 核心流程

1. 接收 System 组传递的传感器数据（54 个数据点）；
2. 实例化 Sensor 对象，预处理负角度（如 a < 0 ? a + 360 : a）；
3. 调用各动作类计算运动幅度，结果存入 ArrayList；
4. 统计分析（平均值、标准差等），生成 JSON 格式报告返回 System 组。

6.3 质量保障

- **单元测试**：使用 JUnit 5 验证单个动作类计算逻辑；
- **集成测试**：测试数据输入到报告生成的完整流程，覆盖率≥85%；
- **代码规范**：遵循 Java 命名规范，使用注释说明算法逻辑，便于扩展新动作类型。

7. 系统集成与交互

7.1 组件间协议

交互场景	协议 / 工具	数据格式	特性
UI ↔ System	RESTful API/WebSocket	JSON	高可靠性（WebSocket 支持实时推送）
System ↔ Data	JDBC	SQL	ACID 事务（保证数据一致性）
System ↔ Analysis	gRPC/HTTP	Protobuf/JSON	低延迟（gRPC 适合流式处理）
Data ↔	JDBC	SQL	批量数据查询（支持

交互场景	协议 / 工具	数据格式	特性
Analysis			历史分析）

7.2 异步处理

- **消息队列**：在 System 组与 Analysis 组间引入 Kafka/ RabbitMQ，解耦数据接收与分析任务，支持流量削峰；
- **事务补偿**：若分析任务失败，通过重试机制或人工干预确保数据最终一致性。

8. 部署架构

8.1 开发环境

- **工具**：IntelliJ IDEA（Java 后端）、VS Code（前端）、Git（版本控制）；
- **测试**：
  - 单元测试：JUnit（Java）、Vue Test Utils（前端）；
  - 集成测试：Spring Test（后端）、Selenium（端到端）；
- **本地部署**：Docker Compose 启动 MySQL、Redis 等服务，Vite 本地运行前端。

8.2 生产环境

- **服务器架构**：
  - **应用层**：Kubernetes 集群部署 System 组与 Analysis 组，Nginx 负载均衡，支持动态扩缩容；
  - **数据层**：MySQL 集群（主从复制）+ Ceph 分布式存储，保障高可用性；
  - **前端层**：静态资源部署至 CDN，Web 服务器（如 Nginx）反向代理 API 请求。
- **监控体系**：
  - 应用指标：Prometheus+Grafana（吞吐量、响应时间）；
  - 日志分析：ELK Stack（错误日志、操作审计）；
  - 报警机制：Alertmanager 实时监控异常并触发通知。

9. 风险与挑战

9.1 主要风险

风险项	影响等级	核心问题
蓝牙兼容性	高	不同浏览器对 Web Bluetooth API 支持不一致
实时性能	中	高并发下数据处理延迟、数据库压力
数据安全	极高	医疗数据传输 / 存储未加密，合规性风险（如 HIPAA）
集成复杂度	中	多技术栈协同导致接口调试成本高

9.2 应对策略

- **兼容性**：建立设备 / 浏览器白名单，提供 CSV 文件导入备选方案；
- **性能优化**：引入 Flink 替代原生 Java 流处理，数据库按时间分区，开启查询缓存；
- **安全性**：传输层使用 TLS 1.3，存储加密敏感字段，定期进行渗透测试；
- **集成效率**：制定统一 API 规范（OpenAPI），通过 Postman 提前进行接口联调，编写集成测试用例。

10. 结论

10.1 核心原则

- **分层解耦**：通过标准接口（REST/gRPC）隔离组件，降低依赖复杂度；
- **数据可靠**：MySQL ACID 事务保证一致性，结合备份策略防止数据丢失；
- **可观测性**：全链路监控（指标 + 日志 + 链路追踪），支持快速故障定位；
- **扩展性**：模块化设计（如 Analysis 组可新增动作类），预留数据库分片、多租户架构接口。

10.2 实施计划

- **2025 Q2**：完成各小组核心模块开发（蓝牙连接、数据存储、基础分析算法）；
- **2025 Q3**：端到端集成测试，优化性能与安全策略；
- **2025 Q4**：通过合规性认证（如 HIPAA），正式上线生产环境。