# 肺动脉介入手术计划管理系统需求文档

## 1. 项目概述

### 1.1 项目背景

心血管疾病是全球主要的致死原因之一，精确的诊断和手术规划对患者的治疗效果至关重要。本项目旨在开发一个专业的肺动脉介入手术计划管理系统，为心血管外科医生提供高精度的诊断和手术规划工具。

### 1.2 项目目标

开发一个基于B/S架构的肺动脉介入手术计划管理系统，通过先进的图像处理和3D建模技术，实现肺动脉的精确三维重建，并提供一系列自动化和手动诊断工具，辅助医生进行诊断和手术规划。

### 1.3 目标用户

医院心血管科室的外科医生

## 2. 功能需求

### 2.1 肺动脉三维建模

* 支持DICOM格式医学影像数据导入
* 自动分割肺动脉区域
* 生成高精度三维模型
* 支持模型调整和优化
* 模型保存与导出功能

### 2.2 自动提取功能

* 自动识别并提取肺动脉关键解剖结构
* 病变区域自动标记
* 提取结果可视化展示
* 提取过程参数可调整

### 2.3 自动测量功能

* 血管直径自动测量
* 血管长度自动测量
* 病变区域尺寸自动测量
* 测量结果自动标注在模型上
* 生成测量数据报告

### 2.4 自动绘制功能

* 自动绘制血管中心线
* 自动绘制横截面
* 自动生成血管树状结构图
* 绘制结果可编辑和调整

### 2.5 手动测量功能

* 支持用户在模型上进行手动测量
* 提供直线、曲线、角度、面积等多种测量工具
* 测量结果可保存并生成报告
* 支持与自动测量结果比较

### 2.6 智能内窥镜功能

* 虚拟内窥镜视角浏览血管内部
* 自动路径规划功能
* 内窥镜视角下的测量功能
* 可调节视角和导航速度
* 内窥镜过程录制功能

### 2.7 诊断意见记录

* 诊断发现记录表单
* 支持附加模型截图
* 诊断历史记录查询
* 支持多人协作诊断
* 诊断报告生成与导出

### 2.8 手术规划方案记录

* 手术路径规划工具
* 手术风险区域标记
* 手术步骤记录
* 手术方案比较功能
* 手术规划报告生成与导出

## 3. 非功能需求

### 3.1 性能需求

* 系统响应时间：普通操作响应时间不超过1秒
* 3D模型渲染帧率不低于30FPS，包括在中等配置PC的浏览器环境中
* 支持至少500MB大小的DICOM数据处理
* 并发用户支持：支持20名以上医生同时在线使用
* 支持主流现代浏览器（Chrome、Firefox、Edge等）的WebGL渲染

### 3.2 安全需求

* 用户身份认证与权限管理
* 患者数据加密存储和传输
* 操作日志完整记录
* 符合医疗数据隐私保护法规
* 安全的会话管理机制

### 3.3 可用性需求

* 直观的用户界面设计
* 关键功能不超过3次点击可达
* 详细的用户操作手册
* 完整的上下文帮助信息
* 浏览器兼容性检测和提示机制

### 3.4 可靠性需求

* 系统稳定运行时间达到99.9%
* 自动数据备份功能
* 系统崩溃时的数据恢复机制
* 关键操作确认机制
* 断线重连和会话恢复功能

### 3.5 扩展性需求

* 模块化系统架构，支持功能扩展
* 标准化API设计，便于组件互操作
* 插件机制，支持功能增强
* 配置驱动的系统行为，减少硬编码
* 清晰定义的前后端接口，支持独立升级

## 4. 技术架构

### 4.1 总体架构

* B/S架构，支持多终端访问
* 前后端分离设计，明确职责边界
* 计算密集型任务在服务端完成，渲染和交互在客户端完成

### 4.2 前端技术

* 3D渲染核心：Three.js/WebGL
* 医学可视化：vtk.js
* 医学影像处理：Cornerstone.js/OHIF框架/dcmjs
* 前端框架：React + TypeScript
* UI组件：Ant Design
* 状态管理：Redux
* 性能优化：WebAssembly、Web Workers
* 客户端存储：IndexedDB

### 4.3 后端技术

* 开发语言：Java
* 框架：Spring Boot
* 安全框架：Spring Security
* ORM：MyBatis
* 项目管理：Maven
* 图像处理：ITK、VTK
* API设计：RESTful

### 4.4 数据库

* MySQL关系型数据库
* 缓存机制：Redis

## 5. 第一阶段开发计划（调整版）

### 5.1 浏览器端高性能3D渲染基础

* WebGL优化实现：利用Three.js和WebGL进行高效率的3D渲染
* 渐进式加载策略：实现数据和模型的分块加载，优先显示关键区域
* 显卡适配优化：针对不同显卡性能自动调整渲染质量
* 内存管理：实现高效的内存使用和释放机制，避免内存泄漏
* 兼容性检测：在启动时检测浏览器WebGL能力，提供降级渲染方案
* 技术选型：Three.js作为主要渲染引擎，结合vtk.js处理医学数据
* 分辨率自适应：根据设备性能调整模型细节级别(LOD)

### 5.2 DICOM数据处理优化

* 客户端解析：使用Cornerstone.js或OHIF框架进行浏览器端DICOM解析
* 流式传输：实现DICOM数据的分块流式传输，减少初始加载时间
* 预处理分流：计算密集型预处理在服务器端完成，渲染在客户端完成
* 缓存策略：实现有效的客户端缓存机制，减少重复数据传输

### 5.3 肺动脉三维重建

* 服务端分割算法：将计算密集型的分割算法部署在服务端
* 客户端渲染优化：优化分割后模型的渲染效率
* 简化模型传输：通过简化网格和压缩技术优化模型传输
* 扩展接口预留：为未来更高级的分割算法预留接口

### 5.4 交互性能优化

* 异步交互处理：确保界面响应不被渲染阻塞
* 操作反馈优化：提供即时的视觉反馈，减少感知延迟
* 事件节流与防抖：优化鼠标和触摸事件处理，提高交互流畅度
* WebWorkers应用：将计算密集型操作迁移到后台线程

### 5.5 基础测量工具

* 直线测量：测量两点间距离
* 曲线测量：测量沿血管中心线的距离
* 横截面测量：自动生成并测量血管横截面
* 模块化设计：以插件形式实现测量工具，方便后期扩展

### 5.6 简化版自动分析

* 计算分离策略：分析算法在服务端实现，结果在客户端展示
* 增量分析模式：支持分步执行分析，避免长时间阻塞
* 分析结果缓存：缓存分析结果，避免重复计算
* 扩展接口设计：为后期更复杂分析算法预留标准化接口

### 5.7 架构扩展性设计

* 模块化架构：采用松耦合的模块化设计，便于功能扩展
* 插件系统：实现插件机制，便于后期功能增强
* 标准化API：设计标准化的内部API，确保各组件间的互操作性
* 前后端分离设计：清晰定义前后端接口，便于独立升级和扩展
* 配置驱动设计：通过配置文件控制系统行为，减少硬编码

### 5.8 数据管理与安全

* 轻量级授权：实现基本的用户认证和授权机制
* 传输加密：确保数据传输安全
* 会话管理：实现可靠的会话管理，支持断线重连
* 数据隔离：确保患者数据的隔离和保密
* 日志记录：记录关键操作，便于调试和审计

## 6. 第二、三阶段开发计划

### 6.1 第二阶段（1个月）

* 完善肺动脉模型导入与处理
* 增强自动提取与测量功能
* 深化模型交互操作功能
* 开发内窥镜基础功能
* 用户界面优化

### 6.2 第三阶段（1个月）

* 诊断意见与手术规划记录功能
* 高级自动测量与绘制功能
* 系统性能深度优化
* 用户体验全面提升
* 完整测试与部署

## 7. 验收标准

### 7.1 功能验收

* 第一阶段核心功能按调整后需求实现并通过测试
* 浏览器3D渲染流畅稳定，符合帧率要求
* 基本测量工具准确可用

### 7.2 性能验收

* 系统响应时间符合要求
* 在推荐配置的PC上通过浏览器实现30FPS以上的渲染性能
* 数据处理能力符合要求
* 通过主流浏览器兼容性测试

### 7.3 安全验收

* 通过安全性测试
* 符合数据隐私保护要求
* 传输加密机制有效

### 7.4 架构验收

* 模块化设计合理
* API设计标准化
* 扩展机制可用

## 8. 风险评估

### 8.1 技术风险

* WebGL在复杂医学模型下的渲染性能可能不达标
* 不同浏览器和硬件环境下兼容性问题
* 大型DICOM数据处理效率问题
* 客户端内存管理风险

### 8.2 应对策略

* 实施渐进式渲染和LOD（细节层次）技术
* 建立全面的浏览器兼容性测试矩阵
* 优化数据传输和处理流程，采用流式传输
* 实现智能内存管理和垃圾回收机制

### 8.3 项目风险

* 开发周期较紧，可能面临延期风险
* 需求变更可能影响开发进度
* 前后端结合的技术挑战

### 8.4 风险缓解措施

* 采用迭代开发方式，优先实现核心功能
* 提前进行关键技术研究和验证，建立技术原型
* 清晰定义前后端接口规范，采用并行开发
* 定期与用户沟通，及时调整需求