深圳技术大学
SHENZHEN TECHNOLOGY UNIVERSITY

本科毕业论文（设计）

题目：基于医学图像的大数据管理平台系统设计与开发

姓　　名	李奕键
学　　院	大数据与互联网学院
专　　业	计算机科学与技术
学　　号	202011100217
指导教师	黄炳顶
职　　称	特聘教授
提交日期	2024 年 5 月 20 日

深圳技术大学本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的毕业论文（设计），题目《基于医学图像的大数据管理平台系统设计与开发》是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：李奕键

日期：　　2024 年 5 月 20 日

目录

基于医学图像的大数据管理平台系统设计与开发

【摘要】随着当今经济不断发展，科技发展的步伐也越来越快，助力了医学的快速发展。与此同时，为了更有利于疾病的发现和治疗，医学影像技术也出现阶梯式进展，医学图像数据量呈现出爆炸式增长。据统计，一台高端医疗设备的日均产生的图像数据量可达数 GB 甚至数百 GB，而一个三甲医院每年产生的医学图像数据更是可达数 TB。如此庞大的数据量，对医学图像的管理和分析带来了极大的挑战。因此，我们需要一种基于医学图像的大数据管理平台系统来提高科研人员对于众多医学图像管理的效率。本项目使用 Vue 框架作为前端开发框架，Springboot 作为后端开发框架，基于医学图像数据库实现对于医学图像的可视化和管理功能整合，简化了对于医学图像的管理操作流程，为科研人员管理研究医学图像提供了便利。

【关键词】医学图像；大数据管理平台；Vue；MySQL；Springboot

Design and Development of a Big Data Management Platform System Based on Medical Images

【**Abstract**】With today's continuous economic development, technology is advancing rapidly, which has greatly benefited the medical field. Medical imaging technology has progressed significantly, aiding in the detection and treatment of diseases. As a result, the amount of medical image data is growing explosively. Statistics show that high-end medical devices can generate several gigabytes to hundreds of gigabytes of image data daily. A tertiary hospital can produce several terabytes of medical image data annually. This massive volume of data presents significant challenges in managing and analyzing medical images. Therefore, a big data management platform for medical images is necessary to improve the efficiency of researchers in handling this data. This project uses the Vue framework for front-end development and Springboot for back-end development. It integrates visualization and management functions for medical images, simplifying the management process and aiding researchers in their work.

【**Key words**】Medical imaging; Big data management platform; Vue; MySQL; Springboot

1. 前言

1.1 课题研究背景及意义

医学图像技术是一项发展多年的临床医学技术，其简单方便、高特异性和非侵入性等优势收到了医学领域的重点关注。随着多年来的科学技术发展，医学图像相关技术已经取得了日新月异的创新与突破，在临床，在病灶识别和诊断、疗效评估等方面辅助医师做出了出色的成果[1]。医学影像从过去的 X 光片和透视逐渐转变，从平面到立体多维图像，从反映解剖结构的形态学图像转为反映脏器功能的“功能性成像”。计算机断层扫描(Computed Tomography，CT)、磁共振成像(Magnetic Resonance Imaging, MRI)、正电子发射计算机断层显像(Positron Emission Computed Tomography, PET)等一系列大型医学影像设备能够帮助诊断及治疗，具有良好的定位和定性过程，能够更早更精准地发现病变，并对病情情况的严重程度进行判断，以便选择较好的治疗方法。

目前，传统的医学图像技术依旧是采用人工诊断的方式，而近年人工智能(Artificial Intelligence,AI)和大数据的蓬勃发展给医学图像提供了不同的诊断方式。基于人工智能和大数据的医学影像方法突破了传统方法的技术壁垒，大大提高了医师的诊断效率[2]。同时，数据的爆发性增长驱动着健康医疗大数据在临床科研、健康管理、公共卫生等核心领域的广泛应用[3,4,5]。但是，随着医学图像大数据的爆发性增长，关于数据的管理和应用也愈加麻烦，如何在数百 GB 甚至 TB 的数据中快速便捷地进行搜索、删减、添加、分类等管理操作成为一个问题。因此，本项目基于医学图像设计开发大数据管理平台系统，旨在解决医学图像数据的存储、管理和分析问题，为临床诊疗、科研和教学提供高效、便捷的支持。

首先，在临床技术研究、医药卫生、基础医学、新型药物研发、疾病治疗等方面，都涉及到医学图像数据获取、处理和分析等技术。现代医学研究的关键集中在如何高效的使用医学技术手段从海量的数据中获取所需要的有价值的数据。而基于医学图像的大数据管理系统可以通过对医学图像大数据的管理与分析，挖掘出潜在的医学信息，为临床诊断、治疗和预后评估提供更为精准的依据。

其次，尽管当今我国的医疗服务系统机构已经日益完善和庞大，但仍然存在诸多问题和不足：患者病情复杂、医疗费用过高、优质医疗资源短缺、部分科室医生工作强度过大、药品研发的速度跟不上不断出现的新型疾病的要求等等[6]。对此，医学图像大数据管理系统可以有效地弥补和解决上诉所说的大部分问题。比如医生利用医学图像管理系统进行医院间的数据互通，综合可以帮助尽快做出有效地诊断，并且在这过程中可以让患者做到全程参与医疗患者的具体情况和既往病史，参考系统内以往的医学图像和诊断记录；再比如建立基于医学图像大数据的管理系统可以通过大数据统计、计算机辅助诊断和精准的生物医学数据有效地获取患者的疾病情况，并对其实施分组指导，降低患者的治疗费用的同时满足患者个性化的医疗服务需求，能够减少医患矛盾发生的可能，并提供给患者更加高效可靠的医疗服务。

此外，随着医学图像相关技术的日渐成熟，医学影像设备产商和技术公司涌现，而高效的医学图像管理系统可以促进医疗机构间的信息共享，医学图像大数据管理平台可以整合多种影像设备的数据，为设备厂商提供更为广阔的市场空间，提升医疗服务质量，推动智慧医疗的发展和创新。

1.2 国内外医学图像大数据管理系统研究进展

医学图像大数据管理系统的研发与应用，是近年来全球医疗信息化领域的重要发展趋势。这些系统旨在高效、安全地存储、处理、分析和共享海量医学影像数据，以支持精准诊断、个性化治疗及科研创新。

1.2.1 国外研究进展

欧美等发达国家在医学图像大数据管理方面走在前列，强调数据的统一标准与格式化。例如，DICOM（Digital Imaging and Communications in Medicine）已成为国际通用的医学影像数据交换标准，确保了不同设备产生的影像数据能够无缝接入管理系统[7]；同时，HL7 FHIR（Fast Healthcare Interoperability Resources）等标准促进了临床信息与影像数据的深度融合[8]。而在云存储和计算方面，众多国外云服务提供商如 Google Cloud Healthcare API、Amazon HealthLake、Microsoft Azure Healthcare APIs 等针对医学影像的大数据解决

方案，提供弹性存储空间、高速传输网络以及强大的并行计算能力，支持深度学习模型训练、远程诊疗、多中心研究等高级应用[9]。

对于医学图像技术和人工智能大数据技术结合的研究，国外也有掌握先进技术的科研公司进行研究。IBM Watson Health、Arterys、Zebra Medical Vision 等公司推出基于 AI 的医学图像分析平台[10],[11]，集成于大数据管理系统中，可自动识别病灶、量化病变特征、预测疾病发展，显著提升医生工作效率与诊断准确性。

鉴于医学影像数据的敏感性，国外研究着重强化数据加密、访问控制、匿名化处理等技术，在隐私保护和数据安全上做到一定的保障。如 GDPR（General Data Protection Regulation）等法规推动了数据脱敏、差分隐私等先进技术在医疗领域的应用，确保大数据共享与挖掘过程中的患者隐私保护[12]。

1.2.2 国内研究进展

我国高度重视医学图像技术的发展和创新，在政策引导和标准建设方面，我国发布的《“十三五”国家信息化规划》《健康中国 2030 规划纲要》等文件明确提出推进医学影像信息系统建设[13]。同时，制定和完善了如 GB/T 30149-2013《医学数字影像通信》等国家标准，促进数据互通互认。此外，我国的自主研发平台也迅速崛起，以联影智慧医疗云、东软医疗 NeuMiva、翼展科技 eMedCloud 等为代表的国产医学影像大数据管理系统，具备大规模数据存储、智能分析、远程协作等功能，已在多家医疗机构落地应用，助力我国医疗信息化自主可控。

在人工智能赋能诊断和科研的研究，我国有众多的企业如阿里云 Link 医疗影像平台、腾讯觅影、深睿医疗 Dr.Wise 运用深度学习、计算机视觉等技术，实现对 CT、MRI 等多种影像的智能化分析，已在肺结节检测、眼底病变识别等领域取得显著成果。此外，这些平台还支持科研数据标注、模型训练等，推动医学影像 AI 的产学研用一体化发展。

为解决医疗资源分布不均问题，国内积极推进区域医联体建设，通过搭建医学影像大数据平台实现各级医疗机构间的数据互联互通。如广东省远程医疗平台、江苏影像云等项目，实现了基层检查、上级诊断、结果互认，极大提升了医疗服

务效率[14]。

国内外医学图像大数据管理系统研究已取得显著进展，体现在数据标准化与集成、云技术应用、AI 辅助诊断、隐私保护等方面。未来，随着 5G、区块链等新技术的应用，以及数据立法的完善，医学图像大数据管理系统将在提升医疗服务效能、驱动医学科研创新、保障患者隐私权益等方面发挥更大作用。

1.3 主要工作

系统需求分析，基于该管理平台所面对的用户，研究目前市面上现有的数据管理平台，分析医学图像大数据管理平台的功能需求、性能需求和安全需求，为系统设计提供依据，以提高和完善用户使用体验。

系统页面设计，根据需求分析，设计医学图像大数据管理平台的整体架构、模块划分和接口定义，使用 Axure 工具设计用户友好的平台界面，实现数据的可视化和操作便捷性。

系统开发,采用 Vue+Springboot 作为前后端开发框架，结合 MySQL 数据库、Mybatis 等技术实现医学图像大数据管理平台对数据库基本的增删查改功能及用户登录功能。

测试与优化,使用 apipost 等工具对医学图像大数据管理平台进行功能测试、性能测试和安全测试，确保系统满足需求，并对系统进行优化，提高系统稳定性和可靠性。通过一定的用户使用测试进行优化意向和意见的收集，对系统进行一些功能的添加和优化，完善用户体验。

数据安全与隐私保护,由于医疗图像大数据是一个充满隐私的数据集，因此，对于医疗影像大数据的隐私和安全问题，必须确保其安全性和隐私保护。在这个方面，通过加密通讯、安全传输、身份认证和授权管理等方式来确保数据的安全。同时，对于医疗图像大数据的隐私问题还需要通过数据匿名化、掩盖等隐私保护方式来减少隐私泄漏的风险。

1.4 论文结构

本篇从整体上分为八大板块，并对各板块的主要内容作了如下整理：

第一章：绪论。阐释设计本科教学文档管理系统前端子系统的背景，论述该系统的重要意义，并梳理国内外相关领域的研究现状，指出现有系统的不足之处。

第二章：系统相关技术简介。介绍系统前端开发所应用的关键技术，包括 Vue.js 框架、MVVM 架构模式、虚拟 DOM 概念、Ant Design UI 库等，对这些技术的原理、特点及应用场景进行说明。

第三章：系统需求分析。对系统需要实现的用户管理、医学图像信息管理、系统公告、用户反馈等功能模块，对系统可行性需求、功能需求、非功能需求进行全面分析，明确系统目标用户群，并进行具体确定。

第四章 系统设计。阐述系统的整体设计思路和技术架构，以及总体功能设计，简单介绍了系统拥有的功能模块以及其对应的子模块，明确系统的功能模块。

第五章 前端模块设计与实现。重点叙述系统各个功能的具体设计以及核心模块的实现方法，包括功能模块设计、系统各个模块的详细设计与实现等，并通过界面设计细节及真实运行界面截图，展现系统的视觉效果和用户交互体验。

第六章 后端功能设计与实现。主要介绍了后端程序的框架构成、所采用的技术、数据库的构成和对各个功能的实现，并对后端程序的目录结构、数据库结构和功能实现进行了详细介绍。

第七章 系统测试。介绍了系统测试使用的环境，并对系统做了完整的功能性测试，保证系统的开发质量以及用户的使用体验。

第八章 总结与展望。全面总结系统设计与实现过程中的创新之处，分析存在的不足，并对未来的发展方向进行展望和安排。

通过以上 8 个部分的阐述，旨在全方位介绍基于医学图像大数据的管理系统的设计理念、技术路线、实现过程及最终效果。

2. 系统相关技术简介

2.1 MVVM 架构

传统前端架构采用的是一种常见的软件架构模式：模型-视图-控制器（Model-View-Controller，MVC）模式。MVC 模式可以将应用程序分为三个层级。最高层是视图层（View），负责界面的显示和用户的交互功能；中间层是控制层（Controller），用来接受视图层发送来的请求，通过操作数据层中的数据完成对应操作。最底层是核心数据层（Model），用于接受视图层数据的请求，并返回最终的处理结果。这三个层级彼此独立，内部的变化不会互相影响[15]。图 2-1 为 MVC 模式的结构：

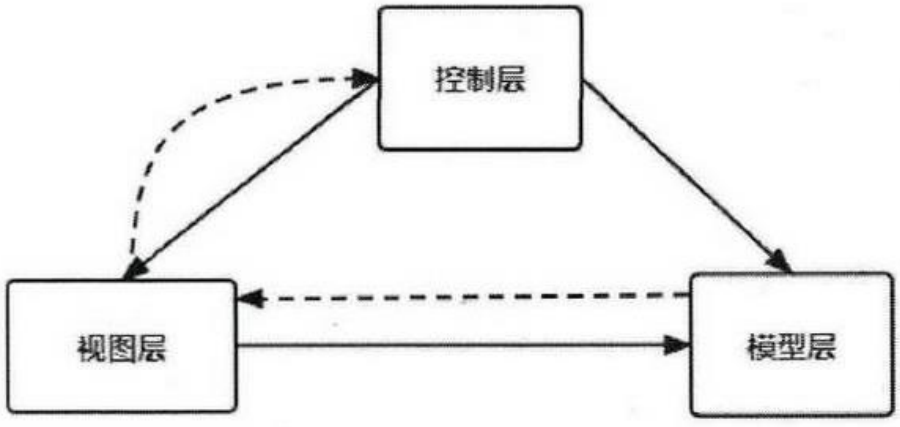

图 2-1 MVC 模型

虽然 MVC 模式在过去被广泛应用，但是 MVC 的缺点也很明显：控制层的复杂性、维护困难以及与视图层的高耦合度和测试难度。为了解决以上问题，产生了一种新的设计模型架构，即模型-视图-视图模型（Model-View-ViewModel，MVVM）模式。MVVM 模式与 MVC 模式的主要区别在于将控制层（Controller）替换为视图模型层（ViewModel），实现了数据层与视图层的双向绑定。当数据层发生变化时，会通知视图层更新，而视图层的变化也会反馈到数据层进行修改。因此在 MVVM 架构下，开发人员可以专注于数据的使用，通过数据的变化来改变视图的状态，而无需频繁操作文档对象模型（Document Object Model，DOM），从而提高了开发效率和软件性能[16]。MVVM 模式实现了数据和视图的解耦，使开发者能够更专注于数据的处理和状态的变化，而不需要直接操作 DOM。这样不仅提高了

开发效率，还改善了软件的性能。

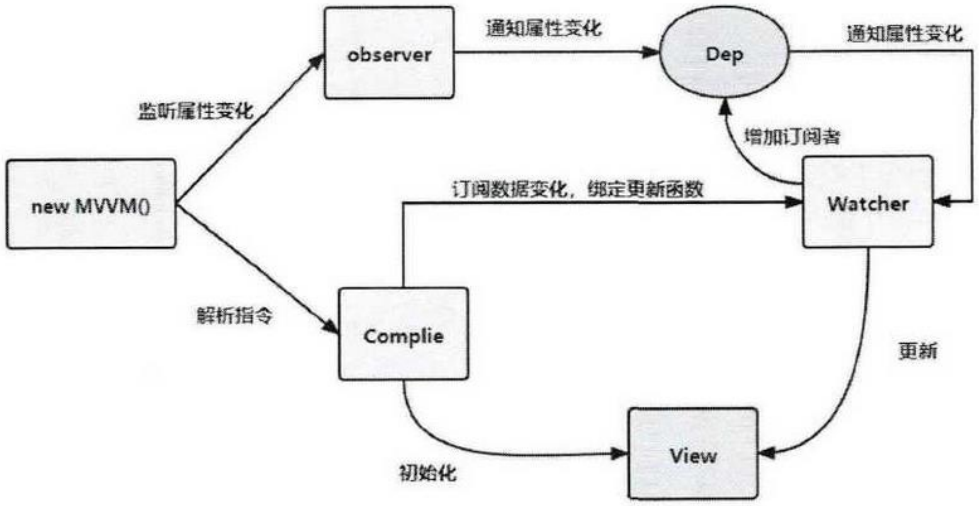

图 2-2 Vue 双向绑定原理图

2.2 Vue 生命周期

Vue 是一款流行的前端 JavaScript 框架，其核心特性之一是组件化开发和丰富的生命周期钩子函数。Vue 组件的生命周期可以大致划分为创建（Initialization）、挂载（Mounting）、更新（Updating）和销毁（Destruction）四个主要阶段，每个阶段都提供了对应的钩子函数供开发者在特定时刻执行自定义逻辑[17]。

在创建阶段的生命周期钩子函数有 beforeCreate 函数和 created 函数。beforeCreate 函数是组件实例被创建后，数据观测（data observer）和事件配置之前触发的钩子。此时，实例的 this 上下文已就绪，但数据、方法、计算属性等尚未初始化，无法访问。created 钩子函数在实例创建完成后被调用。在这个阶段，可以进行数据、计算属性和方法的检测。数据对象 data 完成初始化，但挂载元素 el 仍然不可见。通常在 created 阶段会获取计算属性或对实例进行预处理等操作。

在挂载阶段的钩子函数有 beforeMount 函数和 mounted 函数。beforeMount 钩子函数在编译模板并将其挂载到 DOM 前触发，此时虚拟 DOM 已经生成，但尚未

与实际的 DOM 进行关联。beforeMount 钩子函数在实例挂载前被调用，此时 el 和 data 都已经初始化，模板编译也已完成。最后，在挂载完成后触发 mounted 函数。此时页面已经完成了从模板中渲染 HTML 的步骤。通常在这个阶段会执行一些 Ajax 请求等操作。

当组件数据发生变化时，即为更新阶段，会触发 beforeUpdate、updated、beforeUnmount、unmounted 函数。beforeUpdate 函数在数据变化触发视图重新渲染前调用，此时新的虚拟 DOM 已生成，但尚未与旧的 DOM 进行对比和更新。Updated 函数在数据更新导致的视图重渲染完成后触发，此时 DOM 已经同步到最新状态，可以在此进行基于新 DOM 的操作。需要注意的是，避免在此处频繁操作 DOM 或进行复杂的计算，以免陷入无限更新循环。beforeUnmount 函数在组件卸载前触发。用于清理组件内部的定时器、取消订阅外部资源、解除事件绑定等操作，确保组件能够干净地退出。Unmounted 函数在组件从 DOM 中移除后触发。与 mounted 相对应，此时组件已经完全脱离了 DOM 树，无法再通过 this.$el 访问到相关元素。

除了上述生命周期钩子外，Vue 还提供了错误处理相关的钩子：errorCaptured 函数，以用来捕获一个来自子孙组件的错误。此钩子可以接收三个参数：错误对象、发生错误的组件实例以及包含错误信息的 Vue 插槽。它为开发者提供了一个集中处理组件树中错误的机会，有助于提升应用的健壮性。

2.3 虚拟 DOM 技术

Vue.js 作为一款流行的前端框架，其高效的数据响应与视图更新机制在很大程度上得益于其采用的虚拟 DOM（Virtual DOM）技术[18]。虚拟 DOM 并非 Vue 所独创，而是由 React 首次引入并广为接受的一种现代 Web 开发理念，但在 Vue 中得到了巧妙且深入的应用。虚拟 DOM，顾名思义，是一种对实际 DOM（Document Object Model）的抽象表现，它以 JavaScript 对象的形式来描述 DOM 结构。在 Vue 中，每当数据发生变化时，Vue 会根据当前组件的状态重新生成对应的虚拟 DOM 树。这个树状结构由一系列节点对象组成，每个节点对象包含了元素类型、属性、子节点等信息，精确地模拟了真实 DOM 节点的各项特征，但并未直接操作浏览器的 DOM API 进行渲染。

虚拟 DOM 的核心价值在于减少对实际 DOM 的操作次数。由于 DOM 操作通常比 JavaScript 计算昂贵得多，尤其是在大型应用中，频繁的全量更新可能导致明显的卡顿和性能瓶颈。而虚拟 DOM 通过 Diff 算法找出最小化更新集，极大提升了视图更新的效率。

虚拟 DOM 本质上是对 DOM 结构的抽象描述，与具体的渲染环境无关。这种抽象性使得 Vue 能够轻松地实现服务器端渲染（SSR）、Web Workers 渲染，甚至适配非 Web 平台（如 Weex、Electron 等），只需提供相应的 DOM“实现”（如 Node.js 的 jsdom、Weex 的原生渲染等）即可[19]。而对于开发者而言，无需直接操作或深度理解复杂的 DOM API，只需关注数据和模板。Vue 自动处理了数据变化到视图更新的映射过程，极大地提高了开发效率，降低了心智负担。

2.4 Ant Design 组件库

随着 Web 应用的快速发展，前端的功能性愈发丰富和复杂，为了解决复杂的逻辑交互和 UI 效果展示，给开发者提供更为方便快捷的开发方式，一大批优秀的基于 Vue.js 的前端 UI 框架和组件库不断出现，如 iView、Element UI、Vant UI、Mint UI、Vuex 等。这些组件库的出现大大降低了开发的时间和人工成本，推动了前端技术的发展。

随着前端商业化的趋势，众多企业级产品对更好的用户体验有了进一步的要求。蚂蚁集团通过大量项目实践和总结，逐步打磨出一个服务于企业级产品的设计体系—— Ant Design（简称 AntD）。Ant Design 是一款面向企业级产品的设计体系与开源 UI 组件库，专注于提升 Web 应用的用户体验与开发效率。它以其优雅、一致的设计风格，丰富、完善的组件种类，以及高度可定制、易于使用的特性，在全球范围内受到广泛欢迎，尤其在企业级项目中应用颇多[20]。因此本项目选择 Ant Design 组件库作为前端系统组件库进行开发。

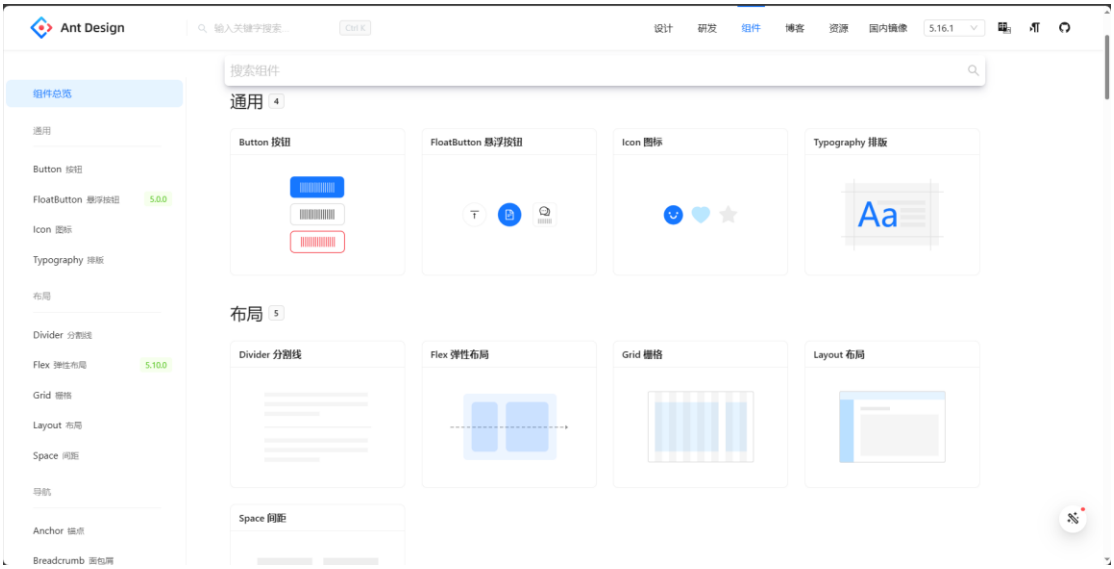

图 2-3 Ant Design 组件库

2.5 本章小结

本章介绍了前端框架相关的技术，重点讨论了 MVVM 模式、Vue 的双向绑定原理、Vue 的生命周期、虚拟 DOM 技术以及 Ant Design 组件库。

MVVM 模式通过将控制层替换为视图模型层实现了数据层与视图层的双向绑定，提高了开发效率和软件性能。

Vue 框架采用了数据劫持和发布/订阅者模式的开发方式，实现了双向绑定。Vue 实例在不同阶段具有完整的生命周期，通过生命周期钩子函数可以在实例的不同阶段进行相应的操作。

虚拟 DOM 技术将页面的更新抽象成 JavaScript 对象表示，并通过 diff 算法计算出需要更新的节点，减少了 DOM 操作的次数，开发者只需关注数据和模板，提高了渲染效率和开发效率。

Ant Design 组件库是一款面向企业级产品的设计体系与开源 UI 组件库，专注于提升 Web 应用的用户体验与开发效率。

3. 系统需求分析

3.1 系统需求分析概述

本系统主要服务对象是医生、医学图像管理员以及研究专家，系统的核心需求是为医疗机构提供一个集中的医学图像资源平台，并减轻医学图像管理员繁重的管理负担。

对于医生来说，涉及到上传医学图像、查看和修改已上传图像信息的功能。因此，本系统需要提供一套完备的图像上传、修改流程以满足医生对医学图像上传的基本需求。

对于管理员来说，涉及到发布公告、管理医生上传的医学图像和医学图像信息，管理用户账号等功能。因此，系统也需要提供完备的公告发布以及账号管理功能。

3.1.1 用户角色

根据对医学图像及其信息的使用权限，我们将系统用户角色划分如下：

（1）系统管理员：拥有拥有网站的全部管理权限，包括用户管理、配置管理（系统设置与代码管理）、医学图像文件管理、站内公告、日志管理等功能。

（2）医生：所有医生负责管理所在科目的医学图像资料，包括上传、更新和查阅医学图像；

（3）研究专家：在对医学图像进行科研工作期间，我们会临时设置一些研究专家账号，以方便他们查阅相应的医学图像信息和资料。在科研工作结束后，这些账号将被禁用。

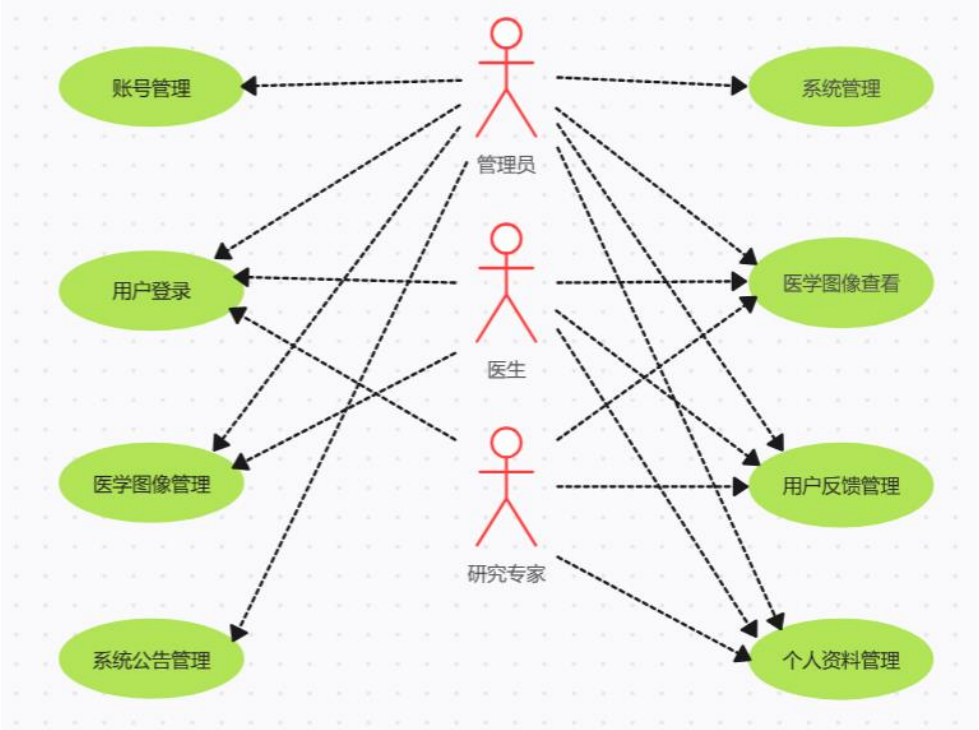

图 3-1 用户角色图

3.2 系统功能需求分析

本系统主要包含用户管理、网站公告、用户反馈、账号管理、系统管理、医学图像管理这 6 大模块。

3.2.1 用户管理

用户管理模块主要负责用户的账号登录、修改个人信息、修改密码、退出登录等操作；

① 注册：网站不提供用户在线注册功能，只有在管理员预先创建账号后，将账号提供给用户才能登陆网站。

② 登录：用户首次登录需要使用管理员提供的用户 ID 进行登录，初始密码与用户 ID 一致。每次登录的时候，如果用户的密码为初始密码，则每次都会在用户首页上提示“当前密码为初始密码，为保证安全，请尽快修改”。用户成功登录后，会直接跳转到个人首页。如果用户一直未修改个人密码，个人主页的工作区将提示用户修改密码，并弹出窗口提醒用户尽快进行修改。如果用户登录验

证错误，系统会重新加载登录页面，并显示提示信息：″用户登录错误，请重新登录″。

③ 修改个人信息：用户可以在修改信息界面编辑个人的基本资料，包括以下数据项：手机号码、电子邮件地址、密码和备注。

④ 修改密码：修改密码必须符合以下要求：至少 14 字或以上的长度，且必须包含至少 1 个大写字母、至少 1 个小写字母、1 个数字、1 个专用字符。

⑤ 退出登录：用户可以通过退出登录，清除登录状态，回到账号登录界面。

用户管理模块用例图如图 3-2 所示：

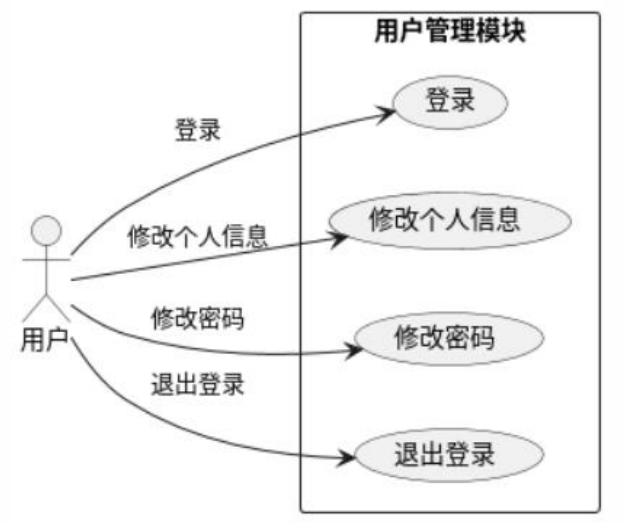

图 3-2 用户管理用例图

3.2.2 系统公告

系统公告模块主要有发布系统公告、查看系统公告以及删除系统公告的功能。

① 发布系统公告功能：系统管理员可以撰写系统公告，并通过提交公告发布在系统上。

② 查看系统公告功能：系统公告以按发布时间逆序的方式显示在公告列表中。所有用户均可查看目前发布的所有公告。

③ 系统公告删除功能：系统管理员可以查看所有系统公告列表，并可选择

删除指定公告记录。系统管理员可以删除所有公告记录。

系统公告模块具体用例图如图 3-3 所示：

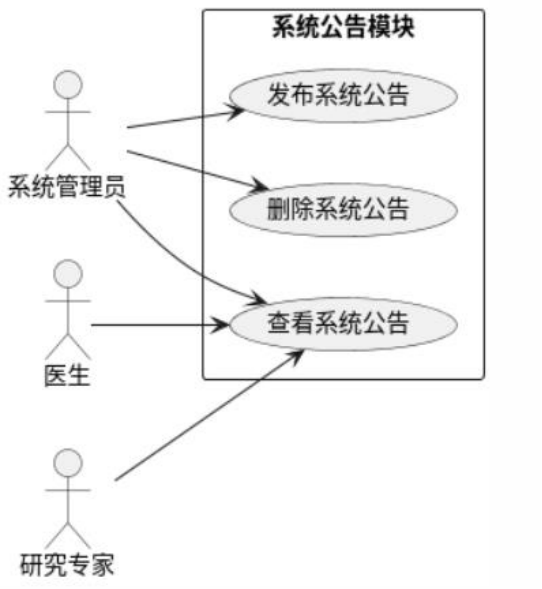

图 3-3 系统公告模块用例图

3.2.3 用户反馈

用户反馈模块主要有提交反馈意见、查看反馈意见、删除反馈意见的功能。

① 提交反馈意见功能：提供反馈意见表单，医生和研究人员可以填写反馈意见并提交；

② 查看反馈意见功能：系统管理员可以看到以提交时间逆序的形式显示所有反馈意见的列表；

③ 删除反馈意见功能：系统管理员可以查看反馈意见列表，并选择删除指定的反馈意见记录。

用户反馈模块用例图如图 3-4 所示：

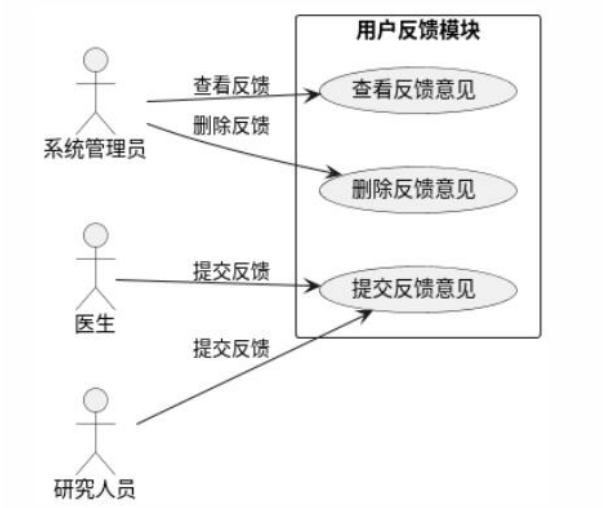

图 3-4 用户反馈模块用例图

3.2.4 账户管理

账户管理模块主要包括添加、修改、删除用户账户和重置账号密码的功能，并且该功能模块只有系统管理员可以使用。

① 添加用户账号：系统默认内置一个系统管理员账户，通过此内置账户创建其他用户账户，包括系统管理员、医生、研究人员这三种角色；新增账号的密码默认与用户 ID 相同。

② 修改用户账号：除登录 ID 外，系统管理员可以修改账号的所有其他数据项。

③ 重置账户密码 ：当用户遗忘密码时，可以向管理员申请恢复初始密码。初始密码与卡号/登录 ID 相同，并将初始密码状态设置为有效。

④ 删除用户账号：指定的账号记录可以删除，但内置的系统管理员账户（admin）不能被删除

账户管理模块用例图如图 3-5 所示：

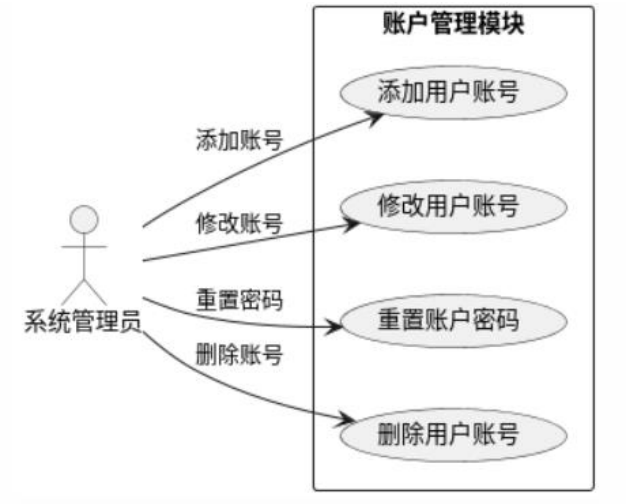

图 3-5 账户管理模块用例图

3.2.5 系统管理

系统管理模块主要包括专业代码管理、医学图像信息管理，以及系统日志管理的功能，并且只有系统管理管理员可以使用。

图 3-6 为系统管理模块的用例图：

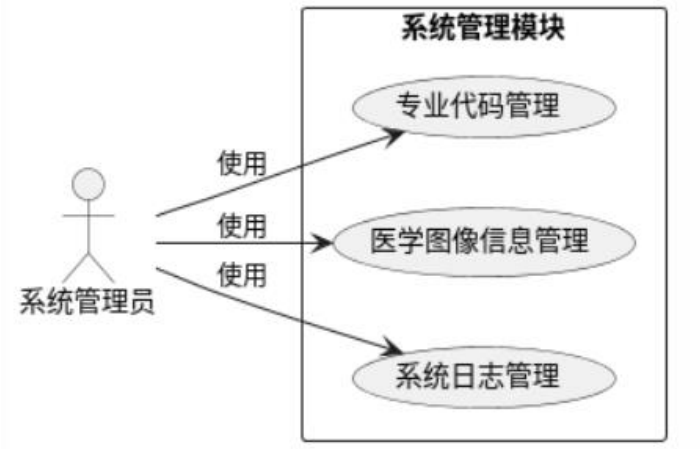

图 3-6 系统管理模块用例图

① 系统代码管理：

a. 添加和修改与医学图像相关的专用名词与代码，例如需求代码（Require）和采集表代码（Collection）等；

b. 查询代码：可以选择某种代码类型，并以列表的形式显示所有代码记录。

c. 删除代码：如果某种代码已被使用，不能删除其代码记录，建议通过修改代码名称来反映代码变更的情况。如果代码未被实际使用，可以删除其记录。

图 3-7 为系统管理模块的用例图：

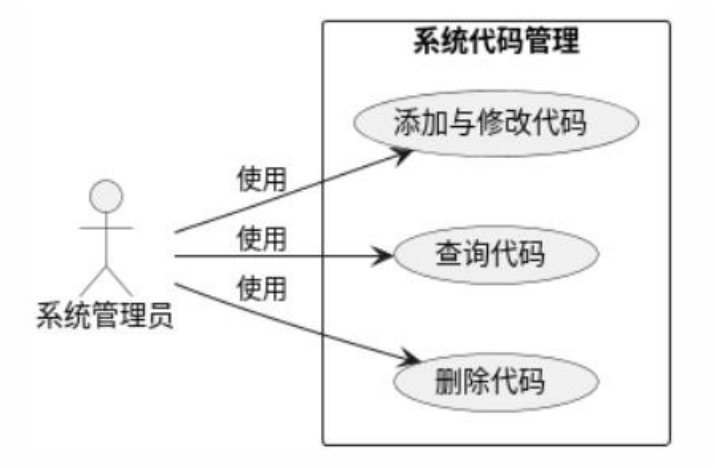

图 3-7 系统代码管理模块用例图

② 系统日志管理:

a. 日志查询：日志查询可以通过后端部署在云端后，在后端查看生成的对应日期和时间的 TXT 日志文件。

b. 日志删除与导出：可以删除部分或全部日志记录，并且可以部分或全部日志记录导出为 TXT 文件。

图 3-8 为系统管理模块的用例图：

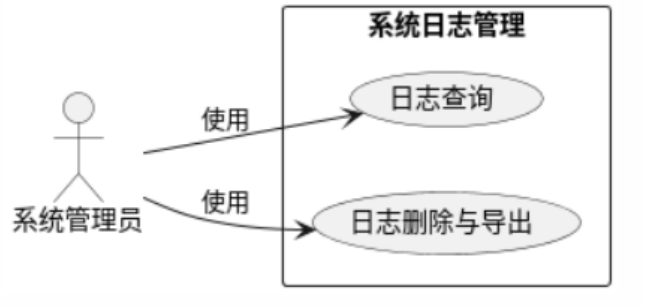

图 3-8 系统日志管理模块用例图

3.2.6 医学图像管理

医学图像管理模块主要包含创建数据采集表、查询采集数据表、创建 CT 标注标准表、查询 CT 标注标准表、查询 CT 标注标签表、创建 CT 筛选数据表、查询 CT 筛选数据表、查询 CT 筛选标准表、查询 CT 清洗标准表。

① 创建数据采集表：在医学图像数据收集后，需要录入系统进行保存，这个功能允许用户创建新的数据采集表，用于收集医学图像数据相关的信息。该功能只有管理员和医生可以使用。

② 查询采集数据表：用户可以使用此功能查询已创建的数据采集表，以查看已经收集的医学图像数据相关的信息。同时可以从云端下载已经上传的医学图像对应文件进行研究。该功能所有用户都可以使用。

③ 创建 CT 标注标准表：这个功能用于创建新的 CT 标注标准表，其中包括需要对 CT 图像进行标注的结构和标签，以及标注标准的创建时间。该功能只有管理员可以使用。

④ 查询CT标注标准表:用户可以使用此功能查询已创建的CT标注标准表，以查看需要标注的结构、标签以及标注标准的创建时间。该功能所有用户都可以使用。

⑤ 查询 CT 标注标签表：该功能用于查询已经创建的 CT 标注标签，可以了解该标签的对应标签 ID 和创建时间。该功能所有用户都可以使用。

⑥ 创建 CT 筛选数据表：这个功能允许管理员和医生创建新的 CT 筛选数据表，用于存储需要筛选的 CT 图像数据的相关信息，例如 CT 影像时期、数据不可用说明等。

⑦ 查询 CT 筛选数据表：所有用户可以使用此功能查询已创建的 CT 筛选数据表，以查看已筛选的 CT 图像数据相关的信息。

⑧ 查询 CT 筛选标准表：此功能用于查询 CT 筛选标准表，其中包含了用于筛选 CT 图像的标准和要求，有助于确保筛选的一致性和准确性。该功能所有用户都可以使用。

⑨ 查询 CT 清洗标准表：用户可以使用此功能查询 CT 清洗标准表，其中包含了清洗 CT 图像数据的标准和方法，有助于确保数据的质量和准确性。该功能所有用户都可以使用。

图 3-9 为医学图像管理模块的用例图：

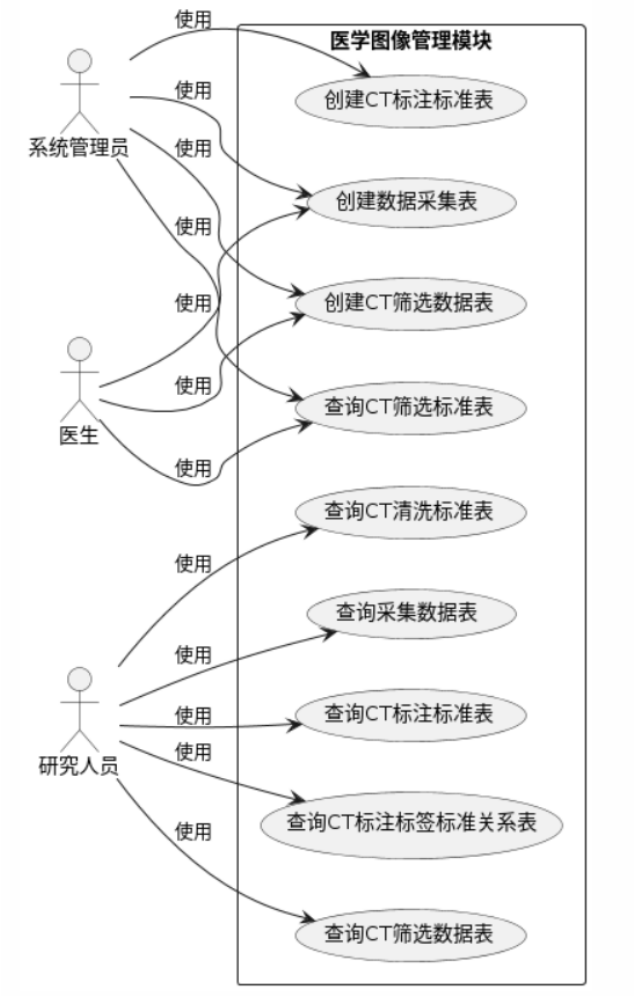

图 3-9 医学图像管理模块用例图

3.3 系统功能需求分析

3.3.1 安全性需求

通过用户身份认证、权限管理、数据加密传输、数据存储加密、后端操作监控等安全预防机制，来保证系统数据和操作的安全性，防止信息泄密和非法访问。

3.3.2 可维护性需求

系统应具备清晰的架构和模块划分，采用前后端分离框架，以提高系统的可扩展性和开发效率。代码应遵循规范，易于阅读，开发文档应完备，以便于后续维护和功能拓展。

3.3.3 兼容性需求

前端界面应具备良好的跨浏览器兼容性，可通过遵循 Web 标准、进行兼容性测试，并采用渐进式增强策略等方式，以支持各种主流浏览器和移动设备的访问，从而提供更优质的浏览体验给用户。

后端代码应具备较好的跨操作系统兼容性，并且能与不同版本的数据库管理系统（DBMS）兼容。确保数据格式和协议的一致性，与前端保证接口版本兼容。

3.4 系统的可行性分析

3.4.1 技术可行性

该前端子系统采用了经验丰富的前端开发技术，运用了 Vue 框架和 Ant Design 作为 UI 框架。这些技术拥有广泛的社区支持、丰富的资源和强大的生态系统，能够帮助开发人员迅速上手并确保技术的可行性。

3.4.2 经济可行性

本系统作为非商用的供医学图像科研公司或医疗机构使用的医学图像管理系统，其开发和运维成本相对可控。作为非商用系统，其开发和维护的成本相对较低，因为不需要考虑商业利润的最大化。针对医学图像科研领域的专业需求，系统的功能和性能定位明确，避免了不必要的开发成本和资源浪费。此外，医学图像管理系统的应用范围广泛，能够为医疗机构和科研单位提供高效的图像管理和数据分析服务，从而为医学研究和临床实践提供支持，进一步提升了其经济可行性。该医学图像管理系统在合理的投入下，能够实现良好的经济效益和社会效益。

3.4.3 操作可行性

系统的主要用户群体为医生、研究人员，他们普遍具备一定的计算机操作水平，能适应系统的正常使用。并且，系统界面操作简单直观，配有详细的使用说明文档，用户无需进行专门的培训就能快速上手。

3.4.4 社会可行性

该系统通过有效管理和分析医学图像数据，有助于加强医学科研领域的数据共享与合作，推动医学科学的进步。同时该系统能够提高医疗机构的工作效率和服务质量，为医护人员提供更准确、便捷的图像诊断和治疗方案，从而提升了患者的医疗体验和治疗效果。故该医学图像管理系统的社会价值巨大，对推动医学科研、提升医疗服务水平和促进人才培养具有重要意义。

3.5 本章小结

本章主要详细分析了系统的需要。首先，描述了系统的背景和目的，并明确定义了用户角色。接着对系统的前后端和具体功能需求进行了详细的分析。此外，还对系统的其他需求进行了细致的分析，包括性能需求，安全性需求，可维护性需求以及兼容性需求等。最后，全面分析了系统的可行性，包括技术可行性、经济可行性、操作可行性、社会可行性这四个方面。

4. 系统设计

4.1 系统设计目标

本系统旨在为医学图像科研公司或医疗机构提供一个非商用的医学图像大数据管理系统，实现对医学图像数据集和资源的高效管理。设计目标有以下几点：

（1） 提高医学图像管理效率，减少人力成本；

（2） 规范图像信息收集、提交流程；

（3） 界面友好，操作简单，提升用户体验；

（4） 确保信息传输安全和操作安全，保护用户隐私和信息保密。

4.2 系统体系架构

本系统使用前后端分离技术。前端主要采用的技术架构为 Vue、Vuex、Vue-Router、Vue-cil、Webpack、Mock。表现层主要用于展示系统各功能页面。Store 采用 Vuex 状态资源管理技术，路由使用 Vue-Rounter 技术。业务层主要负责与后端 API 交互，而 API 层封装所有接口请求，为医学图像管理系统提供完整的 API 支持。Util 存放系统所有工具类。

本系统后端主要采用的技术架构为 Springboot、MyBaits、Tomcat、Spring Data JPA。系统通过 Dao 层实现与数据库的交互，使用 MyBatis 提升了数据访问的性能和稳定性。控制层（Controller）接受前端发来的请求，调用业务逻辑层的服务，最后将业务逻辑层处理后返回的数据传给前端。数据库层使用 MySQL5.8 数据库，同时为了提高访问速度，采用缓存 Redis。Nginx 将作为请求转发和负载均衡的前端子系统和后端系统之间的中间层。配置 Nginx 可以监听前端地址，转发请求到后端系统。用户通过在浏览器中前端地址访问网站，前端地址是前端端子系统的入口。如图 4-1 所示为系统架构图：

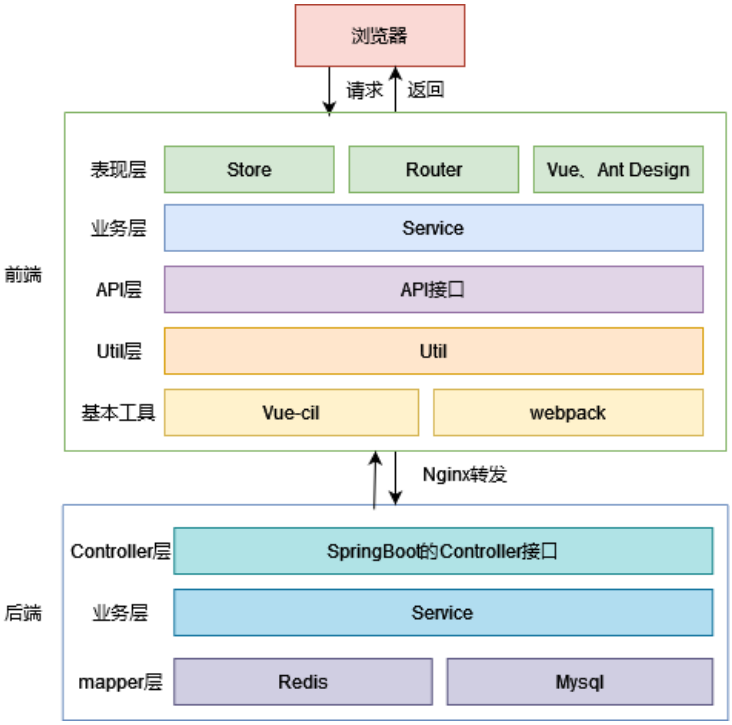

图 4-1 医学图像管理系统体系架构

4.3 总体功能设计

该医学图像大数据管理系统网站主要由 6 个主要功能模块和 25 个子功能模块组成.6 个主要功能模块分别是医学图像管理、系统管理、网站公告、账号管理、用户反馈、用户管理。医学图像管理模块由创建数据采集表、查询采集数据表、创建 CT 标注标准表、查询 CT 标注标准表、查询 CT 标注标签表、创建 CT 筛选数据表、查询 CT 筛选数据表、查询 CT 筛选标准表、查询 CT 清洗标准表组成；系统管理模块由系统代码管理和系统日志管理组成；网站公告模块由发布公告、查看系统公告、修改系统公告、删除系统公告组成；账号管理模块由查询用户账号、添加用户账号、删除用户账号、重置用户账号密码组成；用户反馈模块由查看反馈意见、提交反馈意见、删除反馈意见组成；用户管理模块由用户登录、修改个人信息、修改密码组成。如图 4-2 为系统功能结构图：

图 4-2 医学图像管理系统系统功能结构图

4.4 系统开发与运行环境

本系统前端开发选用的前端框架为 Vue，以及 Ant Design 的 UI 组件库。

前端开发环境：

（1）编译器：VScode

（2）调试工具：Chrome 浏览器

（3）版本控制工具：Git

（4）前端框架和库：@vue/cli (5.0.8)

（5）包管理工具：npm (8.19.2)、yarn (1.22.21)、Node（16.18.0）

（6）操作系统：Windows 10

本系统后端开发选用的后端框架为 Springboot，以及 MyBatis 的数据库连接工具。

后端开发环境：

（1）编译器：IntelliJ IDEA

（2）调试工具：ApiPost7

（3）版本控制工具：Git

（4）后端框架和库：Springboot (2.7.2)、MyBatis (3.4.2)

（5）操作系统：Windows 10

（6）数据库：MySQL(5.7)

4.5 本章小结

本章主要介绍了医学图像大数据管理系统前后端子系统的设计目标、系统体系设计、总体功能模块设计以及系统开发与运行环境首先，设计目标包括提高医学图像管理效率、规范医学图像数据资源的传输流程、加强安全监控和提升用户体验；接着介绍了系统前后端分离和分别对应的技术架构；然后再总体功能模块设计中介绍了系统拥有的模块以及其对应的子模块和功能；最后分别介绍了前端和后端系统的开发和运行环境。

5. 前端模块设计与实现

5.1 基础框架实现

5.1.1 前端目录结构搭建

前端采用了 Vue 框架开发，并通过调用 Ant Design 的 UI 组件库，整体的目录结构如图 5-1 所示：

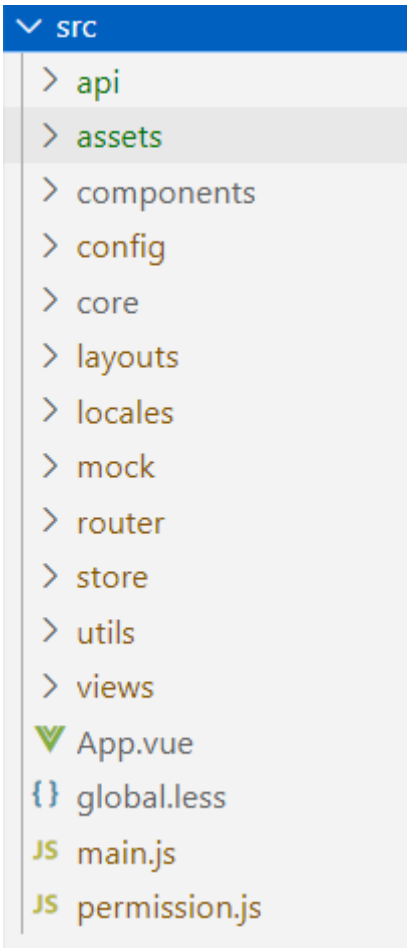

图 5-1 前端项目目录结构图

顶层的目录关键路径和说明如表所 5-1 所示：

表 5-1 顶层目录表

路径	说明
config	用于存放如 Webpack 配置、环境变量配置等项目的配置文件

node_modules	存放通过 npm 安装的项目依赖包
dist	构建后的生产环境代码会输出到这个目录中
docs	存放项目的文档材料，如 API 文档、设计文档等。
public	存放纯静态资源文件，在构建过程中会被完整拷贝到输出目录中
src	存放项目的源代码文件，包括组件、视图、路由、状态管理等
tests	存放项目的单元测试代码文件
package.json	npm 项目的核心文件，用于存储和管理项目元数据、依赖项、脚本等信息
README.md	用于提供项目的概览性说明
babel.config.js	将 ES6+代码转换为 JavaScript 语法，向后兼容
.gitignore	用于指定需要忽略的文件或目录，用于 Git 版本控制
.env	用于储存诸如开发环境、生产环境等环境变量的配置信息

其中最核心的部分就是 src 目录下的，src 目录结构如表 5-2：

表 5-2 src 目录表

路径	说明
api	用于存放与后端服务器进行交互的 API 请求函数，使用 Axios 工具库来实现
assets	用来储存本地静态资源，如图片、字体文件等项目
config	存放项目的基础配置文件，包括路由配置、全局环境变量配置等
components	存储项目中的通用组件，可在不同的页面或模块中重复使用该组件
core	存放项目的引导文件，如项目依赖的全局配置、插件引入等
router	存放 Vue-Router 相关的路由配置文件
store	存放 Vuex 相关的状态管理文件
utils	存放项目中的工具库函数，如日期格式化、数据处理等
locales	存储国际化资源文件，实现多语言支持
views	存放项目的业务页面入口文件和通用模板文件
App.vue	Vue 项目的模板入口文件，一般用于设置全局布局
main.js	Vue 项目的 JavaScript 入口文件，用于初始化 Vue 实例并挂载相关

	依赖
permission.js	用于实现路由守卫，控制不同角色的路由访问权限
global.less	存放全局样式文件，定义项目的通用样式规则

5.1.2 页面路由设计

路由设计页面如表 5-3：

表 5-3 路由设计表

模块	页面地址	页面名称	需要登录	权限	备注
登录	/user/login	登录页面	否	所有用户	进行登录操作
个人设置	/account/settings/basic	基本设置	是	所有用户	进行个人信息修改
用户首页	/dashboard/workplace	个人首页	是	所有用户	进行系统相关导航，可以查看公告
账号管理	/manage/useraccount	用户账号管理	是	系统管理员	管理用户的账号
医学图像信息管理	/collection/add	创建采集表	是	系统管理员、医生	新建采集表，可以上传采集的医学图像
	/collection/table	查询采集表	是	所有用户	高级搜索查询采集表
	/ctmarkstd/add	创建 CT 标注标准表	是	系统管理员	新建 CT 标注标注表
	/ctmarkstd/table	查询 CT 标注标准表	是	所有用户	查看所有 CT 标注标准表
	/ctlabel/table	查询 CT 标注标签表	是	所有人	查看所有查询 CT 标注标签表
	/ctselect/add	创建 CT 筛选数据表	是	系统管理员、医生	新建 CT 筛选数据表

	/ctselect/table	查询 CT 筛选标准表	是	所有用户	查看所有 CT 筛选标准表
	/ctwash/table	查询 CT 清洗标准表	是	所有用户	查看所有 CT 清洗标准表
公告模块	/notice/manage	网站公告管理	是	系统管理员	管理网站公告
	/notice/detail	公告详情	是	所有用户	查看公告详情页面
反馈模块	/feedback/submit	提交反馈意见	是	医生、研究人员	提交反馈意见
	/feedback/see	查看反馈意见	是	系统管理员	查看反馈意见
系统帮助	/help	系统帮助	是	所有用户	查看系统帮助

5.1.3 前端框架入口实现

前端入口为 main.js，首先导入了项目所需的各种依赖模块，包括 Vue 框架本身、应用主文件（App.vue）、路由、状态管理（store）、国际化配置（i18n）、网络请求封装（VueAxios）、Ant Design Vue 组件库的布局组件（ProLayout 和 PageHeaderWrapper）等。接着，对 Vue 框架的一些配置进行了设置，安装并使用 VueAxios 插件、注册全局组件等。在创建 Vue 实例之前，通过执行 bootstrap 函数进行初始化操作。最后创建了 Vue 实例，并将其挂载到了 id 为 app 的 DOM 元素上。关键代码如图 5-2 所示：

```
Vue.config.productionTip = false

// mount axios to `Vue.$http` and `this.$http`
Vue.use(VueAxios)
// use pro-layout components
Vue.component('pro-layout', ProLayout)
Vue.component('page-container', PageHeaderWrapper)
Vue.component('page-header-wrapper', PageHeaderWrapper)

window.umi_plugin_ant_themeVar = themePluginConfig.theme

new Vue({
  router,
  store,
  i18n,
  // init localstorage, vuex, Logo message
  created: bootstrap,
  render: h => h(App)
}).$mount('#app')
```

图 5-2 main.js 代码图

5.1.4 接口函数封装

utils/request.js 对 Axios 进行了二次封装，创建了一个带有请求拦截器和响应拦截器的 Axios 实例，并将其导出供 Vue 实例使用。首先，导入了必要的库和模块，包括 axios、storage、notification、VueAxios、ACCESS_TOKEN 和 router。然后，通过 Axios.create()方法创建了一个新的 Axios 实例 request，并设置了请求网址前缀和请求超时时间。接着，定义了异常处理函数 ERRORHANDLER 来处理错误情况。当响应存在且状态码为 403 或 401 时，会显示相应的通知提示信息。如果是 401 且没有登录状态，会刷新前端页面并要求用户重新登录。同时，在请求拦截器中，从 localStorage 中获取 token 并在请求头中携带该 token。在响应拦截器中，如果后端返回了 2001 状态码（表示登录已过期），则会清除 token 并跳转到登录页。如果请求设置了 fullResponse 选项，则返回完整的响应对象，否则只返回响应数据。最后，定义了一个 installer 对象，用于将 request 实例作为 VueAxios 插件安装到 Vue 实例上，并导出 request 实例本身以及 installer 对象作为 VueAxios 插件。如图 5-3 所示为创建实例的代码：

```
import axios from 'axios'
//  import store from '@/store'
import storage from 'store'
import notification from 'ant-design-vue/es/notification'
import { VueAxios } from './axios'
import { ACCESS_TOKEN } from '@/store/mutation-types'
import router from '@/router'

// 创建 axios 实例
const request = axios.create({
  // API 请求的默认前缀
  baseURL: process.env.VUE_APP_API_BASE_URL,
  timeout: 60000 // 请求超时时间
})
```

图 5-3 request.js 代码图

在对 Axios 进行封装后得到一个 Axios 实例对象，在 src/api 目录下的 js 文件中调用这个对象，以 account_api.js 文件为例（即用户管理接口），如图 5-4 是部分代码：

```
import request from '@/utils/request'

const userApi = {
  UserList: '/api/accountm/userlist',
  AddUser: '/api/accountm/adduser',
  SearchUser: '/api/accountm/searchuser',
  ManageReset: '/api/accountm/managereset',
  ManageDelete: '/api/accountm/managedelete'
}
// 查询所有用户
export function getuserlist () {
  return request({
      url: userApi.UserList,
      method: 'get'
  })
}
```

图 5-4 account_api.js 代码图

5.1.5 主组件的实现

App.vue 是应用的根组件，主要负责配置语言环境和页面渲染。它使用 a-config-provider 组件提供 Ant Design 组件库的语言环境。在根容器中，包含一个 id 为“app”的 div 元素，用于承载应用程序。在这个容器内，使用 router-view 组件渲染 Vue 路由中配置的组件。如图 5-5 所示：

```
<template>
  <a-config-provider :locale="locale">
    <div id="app">
      <router-view/>
    </div>
  </a-config-provider>
</template>
```

图 5-5 App.Vue 代码图

最终实现的系统页面整体框架如图 5-6 所示：

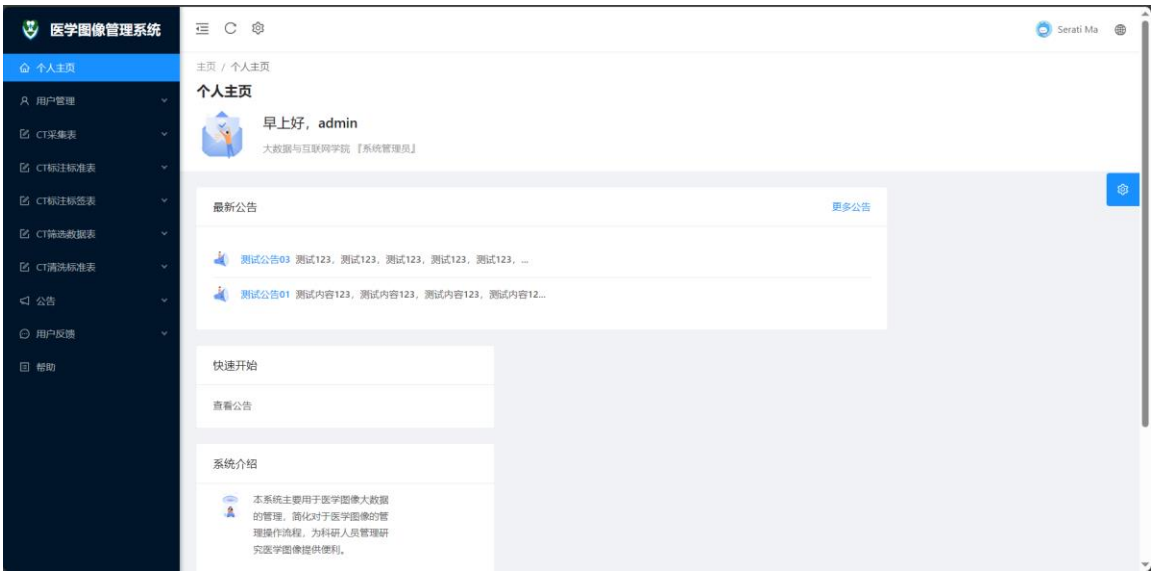

图 5-6 个人首页图

5.2 模块功能实现

5.2.1 用户管理

（1） 功能描述

用户管理模块包括登录、修改密码、个人资料修改和退出系统功能。用户登录系统后，输入用户名和密码进行验证。如果验证失败，系统显示“密码错误，请重新输入”。登录成功后，系统检查用户密码是否为初始密码，若是，则提示用户修改初始密码，并转至个人信息页面。如图 5-7 所示为用户管理的功能结构图：

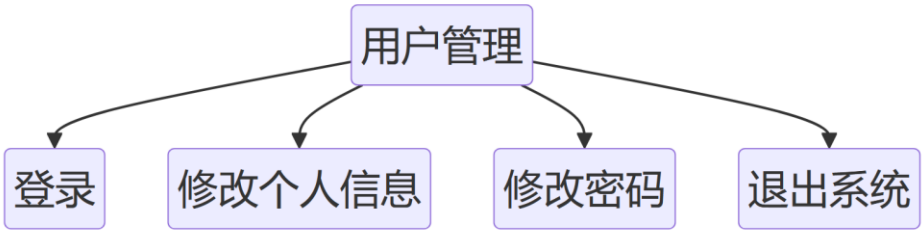

图 5-7 用户管理功能结构图

（2） 模块接口设计

用户管理模块的 API 接口如表 5-4、表 5-5、表 5-6 所示。

表 5-4 登录接口说明

功能	登录		
请求方法	POST		
URL	/api/user/login		
输入参数	参数名	类型	说明
	userId	String	用户账号
	password	String	用户密码
返回参数	data	Object	包含用户信息、token
	code	Int	状态码
	msg	String	状态信息

表 5-5 修改用户信息接口说明

功能	修改用户信息		
请求方法	PUT		
URL	/api/user/manage/update/personal		
输入参数	参数名	类型	说明
	email	String	电子邮箱

	memo	String	备注
	mobile	String	手机号码
	oldPassword	String	旧密码：更新密码所用
	password	String	密码
	roleType	Int	角色类型
	userID	Int	用户 id
	username	String	用户名
返回参数	data	Object	包含用户信息、token
	code	Int	状态码
	msg	String	状态信息

表 5-6　用户退出登录接口说明

功能	用户退出登录		
请求方法	GET		
URL	/api/user/logout		
返回参数	code	Int	状态码
	msg	String	状态信息

（3）模块界面设计

如图 5-8 所示为登录页面：

图 5-8 用户登录页面图

用户主页为用户登陆后进入的页面，首页显示平台功能导航栏左侧为平台功

能导航栏，不同的用户角色会有不同的菜单栏，如最新公告、上次登陆信息、系统介绍等主要信息。

针对不同的用户类型，例如系统管理员、医生、研究人员，平台会给予不同的菜单权限，确保用户只能访问具备访问权限的功能，避免用户误操作或越权访问平台。

图 5-9 是医生角色所展示的首页：

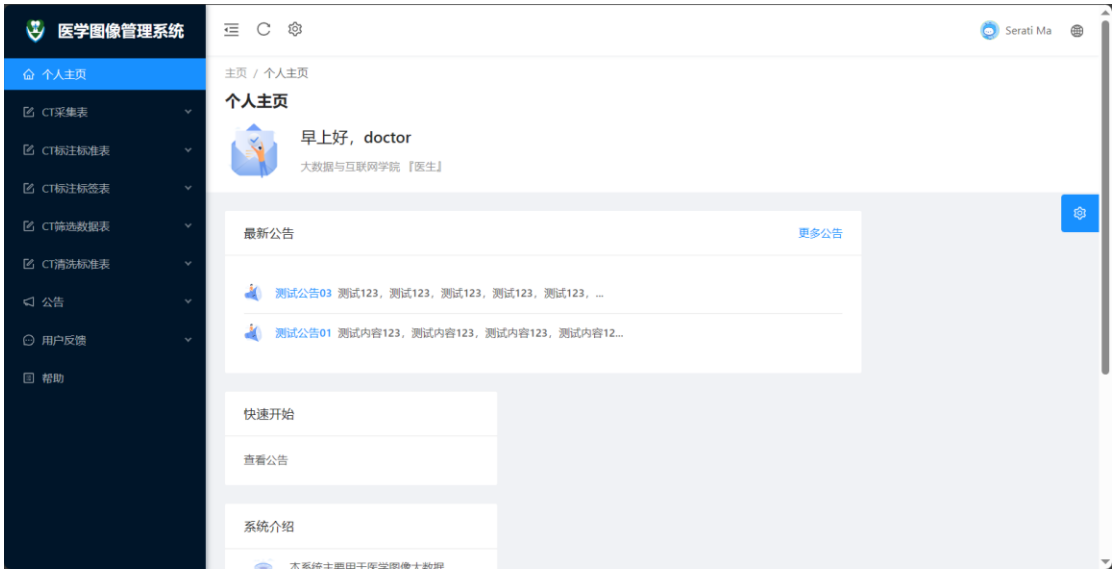

图 5-9 医生用户个人首页图

（4）程序流程图

用户管理的流程是：用户在访问到系统后先进入登录界面，通过输入用户 ID 和密码进行登录。系统会检验用户的登录信息是否正确，如果登录失败，则显示“账号或密码错误”，用户需要重新输入账户和密码进行登录；如果登录成功，则系统会判断用户密码是否为初始密码，如果密码是初始密码，用户会收到提示要求修改密码并跳转至个人信息修改页面；如果不是初始密码则留在个人主页。在个人信息页面，可以进行个人资料或者密码的修改。当用户成功更新个人信息后，系统会显示更新成功的消息，用户也可以选择在个人首页退出登录，返回到用户登录页面。

用户管理流程图如图 5-10 所示：

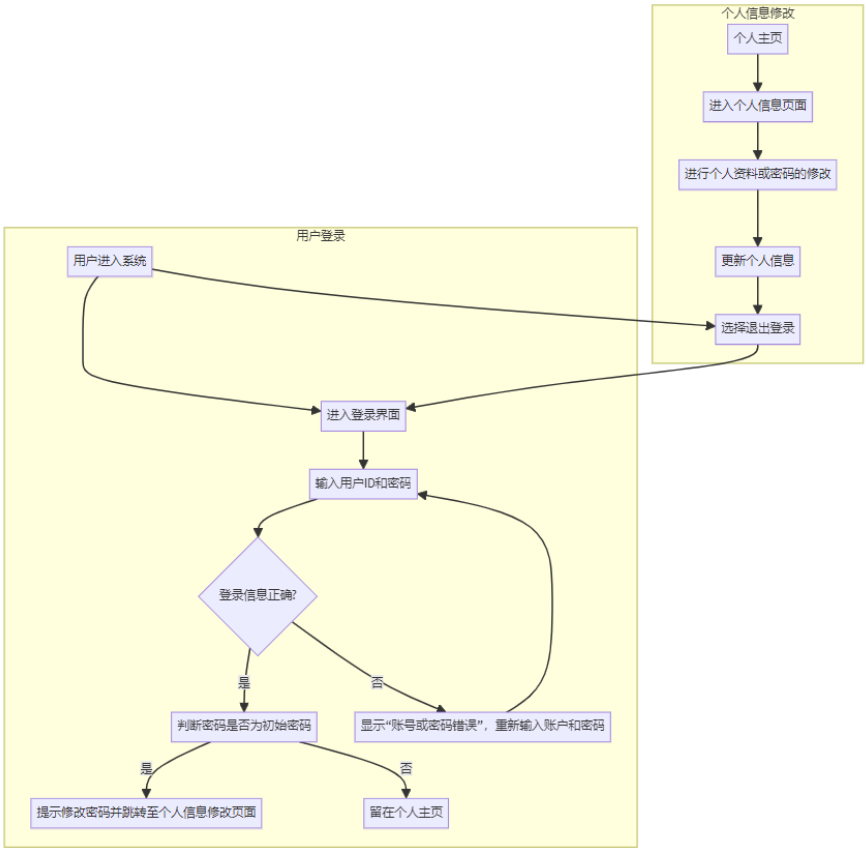

图 5-10 用户管理模块流程图

（5）关键代码实现

以下是 store/modules/user.js 的部分代码。其中，定义了一个 user 对象，包含了 Vuex store 中的状态和状态变更的方法。state 对象包含了一组状态属性，包括用户令牌（token）、登录时间（loginTime）、登录 IP（loginIP）、角色类型（roleType）、用户名（username）、用户 ID（userId）、角色列表（roles）、姓名（name）和欢迎语（welcome）。如图 5-11 所示为 user.js 部分代码：

```
const user = {
  state: {
    token: '',
    username: '',
    userID: '',
    roleType: '',
    roles: [],
    name: '',
    welcome: ''
  },

  mutations: {
    SET_TOKEN: (state, token) => {
      state.token = token
    },
    SET_USERNAME: (state, username) => {
      state.username = username
    },
```

图 5-11 user.js 部分代码图

在登录发送请求后，调用 login 方法将用户信息发送至后端进行判断，若成功则后端返回 code 为 1000，前端再进行判断是否登录成功；同时如果登录成功，后端会生成用户令牌 token，前端系统将 token 通过调用 SET_TOKEN 的方法将用户 token 存放至 Vuex store 中，以便在后续的请求中将 token 存至请求头 Header 里面发送给后端进行权限判断。无论登录是否失败，系统都会将登录请求的结果传递给调用者。如图 5-12 所示为 Login 函数：

```
actions: {
  // 登录
  Login ({ commit }, userInfo) {
    return new Promise((resolve, reject) => {
      login(userInfo).then(response => {
        if (response.code === 1000) {
          const result = response.data
          storage.set(ACCESS_TOKEN, result.token, new Date().getTime() + 1 * 24 * 60 * 60 * 1000)
          commit('SET_TOKEN', result.token)
          resolve(response)
        } else {
          resolve(response)
          // reject(response.msg)
        }
      }).catch(error => {
        reject(error)
      })
    })
  },
```

图 5-12 user.js 函数代码图

登录成功后，Vue Router 的全局前置守卫（beforeEach）执行以下逻辑：

开始进度条（NProgress.start()）显示页面加载进度；根据目标路由的元信息设置页面标题；检查是否存在用户令牌 token，若存在，则判断目标路由是否为登录页，若是则重定向到登录页面，否则检查角色列表（store.getters.roles），若为空则请求 GetInfo 动作获取用户信息并生成可访问的路由表，若不为空则直接访问；若不存在用户令牌，且目标路由不在免登录名单中，则重定向到登录页并携带重定向参数（to.fullPath）；结束进度条（NProgress.done()）表示页面加载完成。图 5-13 是 permission.js 的部分代码：

```
if (store.getters.roles.length === 0) {
  // request login userInfo
  store
    .dispatch('GetInfo')
    .then(res => {
      // console.log('res', res)
      // 根据用户权限信息生成可访问的路由表
      store.dispatch('GenerateRoutes', { token, ...res }).then(() => {
        // 动态添加可访问路由表
        // VueRouter@3.5.0+ New API
        resetRouter() // 重置路由 防止退出重新登录或者 token 过期后页面未刷新，导致的路由重复添加
        store.getters.addRouters.forEach(r => {
          router.addRoute(r)
        })
        // 请求带有 redirect 重定向时，登录自动重定向到该地址
        const redirect = decodeURIComponent(from.query.redirect || to.path)
        if (to.path === redirect) {
          // set the replace: true so the navigation will not leave a history record
          next({ ...to, replace: true })
        } else {
          // 跳转到目的路由
          next({ path: redirect })
        }
      })
    })
```

图 5-13 permission.js 部分代码图

接着，user.js 会调用 GetInfo 函数来获取用户信息，并保存到本地并对照权限表进行用户权限的设定，并根据不同角色展示不同的菜单界面。如图 5-14 为 user.js 的部分代码：

```
let role = {}
// 在这里配置路由以及权限
if (result.roleType === 1) {
  role = { id: 'admin', name: '系统管理员', permissions: [ { 'roleId': 'admin', 'permissionId': 'admin' } ] }
  role.permissionList = ['dashboard', 'manage', 'add', 'stdadd', 'search', 'fbsearch', 'noticeadd', 'help']
} else if (result.roleType === 2) {
  role = { id: 'doctor', name: '医生', permissions: [ { 'roleId': 'doctor', 'permissionId': 'doctor' } ] }
  role.permissionList = ['dashboard', 'add', 'search', 'fbadd', 'help']
} else if (result.roleType === 3) {
  role = { id: 'expert', name: '研究专家', permissions: [ { 'roleId': 'expert', 'permissionId': 'expert' } ] }
  role.permissionList = ['dashboard', 'search', 'fbadd', 'help']
}
```

图 5-14 user.js 部分代码图

退出登录的代码逻辑为：首先，清空用户角色的令牌以及清空角色列表，接着，调用后端接口 logout 请求，将指定 token 设置为失效状态。如图 5-15 所示：

```
// 登出
Logout ({ commit, state }) {
  commit('SET_TOKEN', '')
  commit('SET_ROLES', [])

  return new Promise((resolve) => {
    const params = {}
    params['token'] = state.token
    logout(params).then(() => {
      storage.remove(ACCESS_TOKEN)
      resolve()
    }).catch(() => {
      // // console.log('', err)
    }).finally(() => {
      storage.remove(ACCESS_TOKEN)
      resolve()
    })
  })
}
```

图 5-15 user.js 部分代码图

5.2.2 账号管理模块

（1）功能描述

账号管理模块仅提供给系统管理员使用，任何账户（除最初的管理员账号外）都要通过系统管理员添加账户的方式进行注册。系统管理员可以通过用户列表查看所有用户的数据，或通过用户 ID 查找对应用户数据，并可以重置用户密码和删除用户。

（2）模块接口设计

该模块的 API 接口如表 5-7、表 5-8、表 5-9、表 5-10、表 5-11 所示。

表 5-7　添加用户接口说明

功能	添加用户		
请求方法	POST		
URL	/api/accountm/adduser		
输入参数	参数名	类型	说明
	userRO	Object	用户信息
返回参数	data	Object	包含用户信息、token
	code	Int	状态码
	msg	String	状态信息

表 5-8　查询所有用户接口说明

功能	查询所有用户		
请求方法	GET		
URL	/api/accountm/userlist		
输入参数	参数名	类型	说明
	userROPageVO	Object	用户信息相关查询
返回参数	data	Object	包含具体的反馈信息
	code	Int	状态码
	msg	String	状态信息

表 5-9　根据 id 查询用户信息详情接口说明

功能	根据 id 获取用户信息
请求方法	POST

URL	/api/accountm/searchuser		
输入参数	参数名	类型	说明
	id	Int	用户 id
返回参数	data	Object	包含具体的用户信息信息
	code	Int	状态码
	msg	String	状态信息

表 5-10　重置用户密码接口说明

功能	重置用户密码		
请求方法	PUT		
URL	/api/accountm/managereset		
输入参数	参数名	类型	说明
	id	Int	用户 id
	userRO	Object	用户信息
返回参数	code	Int	状态码
	msg	String	状态信息

表 5-11　删除用户信息接口说明

功能	根据 id 删除用户信息		
请求方法	DELETE		
URL	/api/accountm/managedelete		
输入参数	参数名	类型	说明
	id	Int	用户 id
返回参数	code	Int	状态码
	msg	String	状态信息

（3）程序流程图

账户管理模块的流程为：系统管理员登录后，进入账户管理页面。在账户管理页面，管理员可以进行查找所有用户信息、根据用户 ID 查找用户信息、添加账户、删除账户和重置账户密码。如果管理员选择添加用户账号，则需要填写相关信息，包括账户的用户名和账号角色类型，然后点击“确定”即可添加成功；如果管理员选择根据用户 ID 查找用户信息，则在输入框内填写用户 ID 并点击搜索，即可在下方表格内获取对应用户信息；如果管理员选择删除账户，则需要填

写删除的账户 ID，并在二次确认后即可成功删除账户；如果管理员选择重置账户密码，则需要填写需要重置密码的用户 ID，并在点击“确认”后成功重置。

账户管理模块流程图如图 5-16 所示：

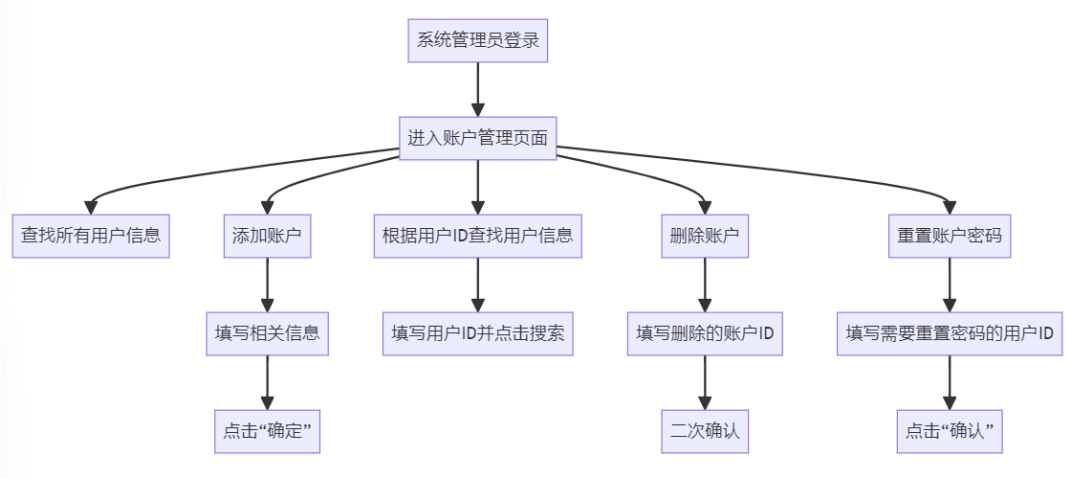

图 5-16 账户管理模块流程图

（4）模块界面设计

账号管理界面如图 5-17 所示：

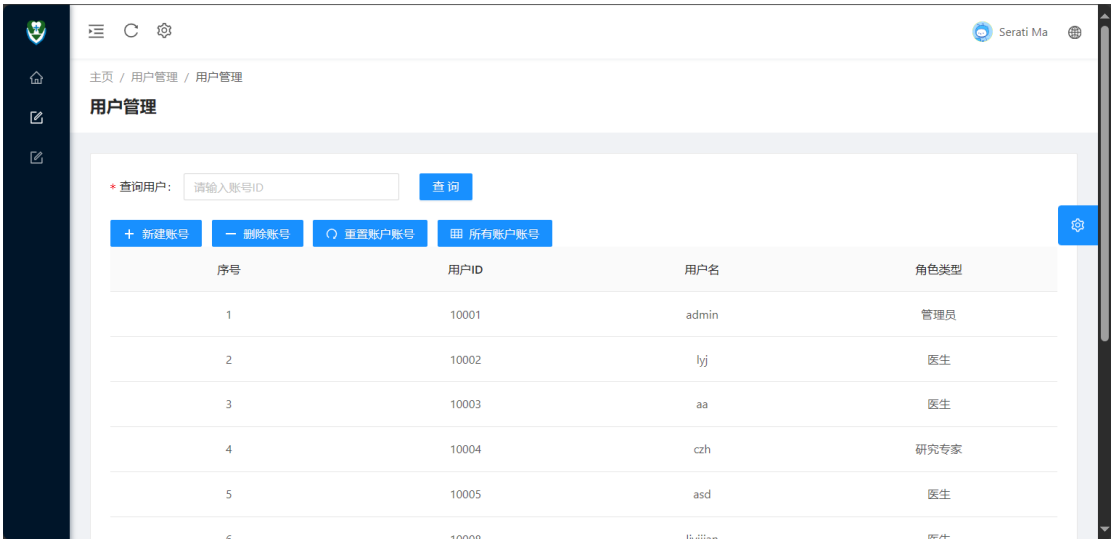

图 5-17 账号管理界面图

5.2.3 网站公告模块

（1）功能描述

在系统的网站公告模块中，系统管理员可以随时发布、修改系统公告，所有用户都可以获取和查看公告。系统管理员通过公告向其他用户传达重要的信息和通知。公告内容可以是富文本的形式，满足自定义样式的需要，同时还可以添加附件，并将其呈现在主页上，方便用户对最新动态消息的快速浏览和了解。所有用户可以通过点击查看或者点击标题的操作进入具体的公告页面。

（2）模块接口设计

该模块的 API 接口如表 5-12、表 5-13、表 5-14、表 5-15、表 5-16 所示。

表 5-12 添加系统公告接口说明

功能	添加系统公告		
请求方法	POST		
URL	/api/bulletin/insert		
输入参数	参数名	类型	说明
	issueSTime	String	公告发布时间
	operator	String	公告作者
	subject	String	标题
返回参数	data	Object	包含用户信息、token
	code	Int	状态码
	msg	String	状态信息

表 5-13 分页查询系统公告接口说明

功能	分页查询系统公告		
请求方法	GET		
URL	/api/bulletin/get/page		
输入参数	参数名	类型	说明
	bulletinROPageRO	Object	系统公告相关的查询条件，比如标题、作者等
返回参数	data	Object	包含具体的系统公告
	code	Int	状态码
	msg	String	状态信息

表 5-14 根据 id 查询系统公告详情接口说明

功能	根据 id 获取系统公告

请求方法	POST		
URL	/api/bulletin/get/single		
输入参数	参数名	类型	说明
	id	Int	系统公告 id
返回参数	data	Object	包含具体的系统公告信息
	code	Int	状态码
	msg	String	状态信息

表 5-15 更新系统公告接口说明

功能	更新系统公告		
请求方法	PUT		
URL	/api/bulletin/update		
输入参数	参数名	类型	说明
	id	Int	系统公告 id
	issueSTime	String	公告发布时间
	expiredTime	String	公告过期时间
	operator	String	公告作者
	subject	String	标题
	annexPath	String	附件
返回参数	code	Int	状态码
	msg	String	状态信息

表 5-16 删除系统公告接口说明

功能	根据 id 删除系统公告		
请求方法	DELETE		
URL	/api/bulletin/delete		
输入参数	参数名	类型	说明
	id	Int	系统公告 id
返回参数	code	Int	状态码
	msg	String	状态信息

（3）模块界面设计

网站公告管理界面如图 5-18、图 5-19 所示：

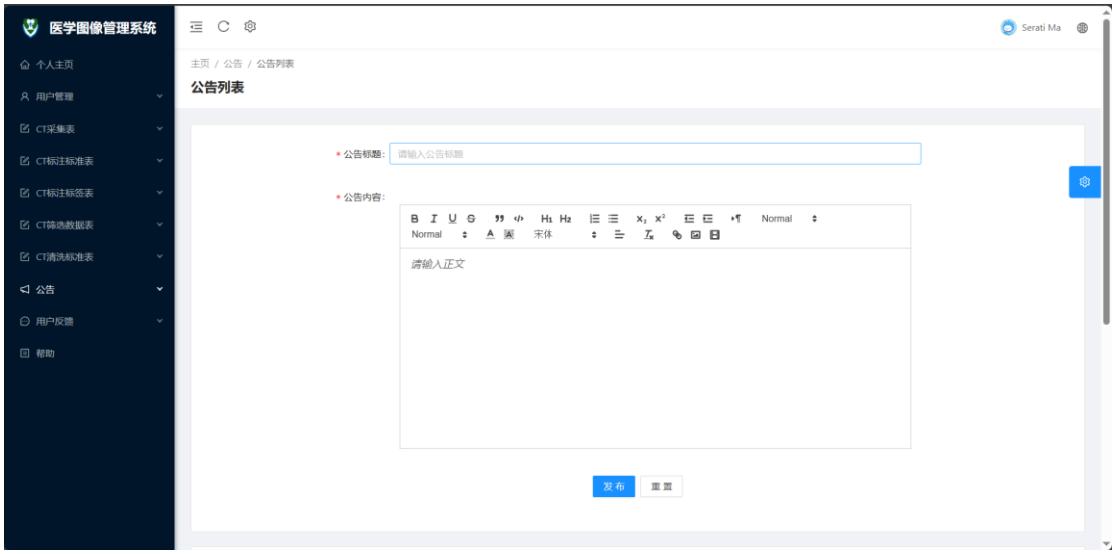

图 5-18 网站公告管理界面图

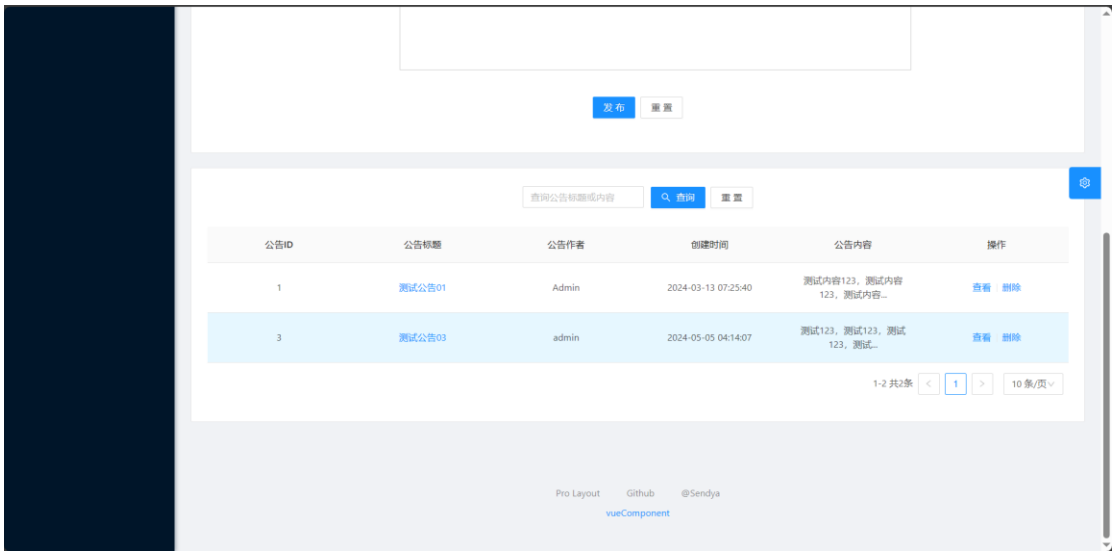

图 5-19 网站公告管理界面图

（4）程序流程图

系统公告模块的流程为：用户登录后，根据用户角色的不同，会有不同的操作页面，如果用户是系统管理员或者文档管理员，将进入公告管理页面。在公告管理页面，管理员可以选择发布公告、查看公告列表。如果管理员选择发布公告，将进入发布公告页面，填写相关的公告信息，包括公告标题、公告内容、发布时间这些关键信息，然后点击“发布”按钮，即可完成公告的发布。如果管理员选择查看公告列表，系统将显示所有公告列表。管理员还可以选择删除公告. 如果

用户是医生或研究人员，进入的是公告列表页面，可以查看所有的公告列表。如图 5-20 为系统公告模块流程图：

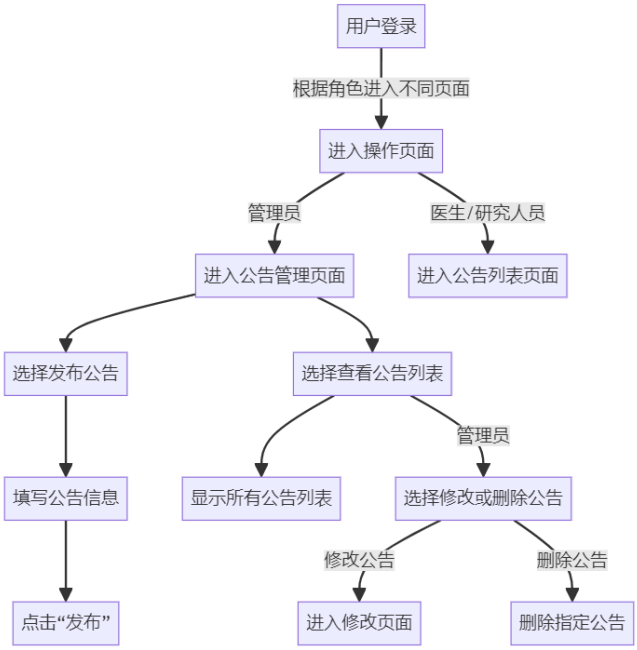

图 5-20 系统公告模块流程图

（5）关键代码实现

发布公告是通过调用后端接口，将填写的数据传送给后端，并调用公告列表分页查询来更新公告列表，同时公告内容使用的是 QuillEditor 富文本组件，保证管理员在发布公告时，可以添加图片以及使用自己想要的公告文本格式。如图 5-21 和图 5-22 所示：

```
<a-form-item
  label="公告内容"
  :labelCol="{ lg: { span: 5 }, sm: { span: 5 } }"
  :wrapperCol="{ lg: { span: 15 }, sm: { span: 17 } }"
>
  <QuillEditor
    v-decorator="[
      'content',
      { initialValue: '<p></p>', rules: [{ required: true, message: '公告内容不能为空' }] },
    ]"
  ></QuillEditor>
</a-form-item>
```

图 5-21 系统公告模块发布公告代码图

```
noticeAdd(tmp)
  .then((res) => {
    if (res.code === 1000) {
      this.$message.success( content: '公告添加成功')
      this.publishForm.resetFields()
      const param = {
        content: '<p></p>'
      }
      this.$nextTick( cb: () => {
        this.publishForm.setFieldsValue(param) //
      })
      // 清空文件上传组件的数据
      if (this.isFile === true) {
        this.$refs.fileUpload.clearFilePath()
        this.docPath = ''
      }
      this.preloadTime()
      this.getNoticeList()
```

图 5-22 系统公告模块发布公告代码图

删除公告，会先调用 confirm 弹窗来进行二次确认，确认无误后，才会将公告 id 传给后端接口，从而删除指定公告内容。如图 5-23 所示：

```
this.$confirm({
  title: '提示',
  content: '确认删除该公告',
  okText: '确认',
  cancelButtonText: '取消',
  onOk: () => {
    // 这里写删除请求的代码
    const params = {}
    params['id'] = id
    noticeDelete(params)
      .then((res) => {
        console.log(res)
        if (res.code === 1000) {
          this.$message.success( content: '删除成功')
          // 刷新表格
          this.getNoticeList()
        } else {
          this.$message.error( content: '删除失败, ' + res.msg)
        }
      })
```

图 5-23 系统公告模块删除公告代码图

5.2.4 用户反馈模块

（1）功能描述

在用户反馈模块中，医生和研究人员可以反馈本系统中存在的问题和需求，比如系统 bug、功能缺陷、用户体验、功能需求等。系统管理员可以定期查看和处理这些反馈意见，并根据这些反馈内容对系统进行更新和维护，从而完善和优化用户的体验。

（2）模块接口设计

该模块的 API 接口如表-17、表-18、表-19、表-20 所示。

表 5-17 添加反馈信息接口说明

功能	添加反馈信息		
请求方法	POST		
URL	/api/feedback/create		
输入参数	参数名	类型	说明
	content	String	反馈内容
	title	String	反馈标题
返回参数	code	Int	状态码
	msg	String	状态信息

表 5-18 分页查询反馈信息接口说明

功能	分页查询反馈信息		
请求方法	GET		
URL	/api/feedback/list		
输入参数	参数名	类型	说明
	feedbackPOPageRO	Object	反馈信息相关的查询条件，比如标题、作者等
返回参数	data	Object	包含具体的反馈信息
	code	Int	状态码
	msg	String	状态信息

表 5-19 根据 id 查询反馈信息详情接口说明

功能	根据 id 查询反馈信息详情		
请求方法	GET		
URL	/api/feedback/detail		
输入参数	参数名	类型	说明
	id	Int	反馈信息 id
返回参数	data	Object	包含具体的反馈信息
	code	Int	状态码
	msg	String	状态信息

表 5-20 删除反馈信息接口说明

功能	根据 id 删除反馈信息		
请求方法	DELETE		
URL	/api/bulletin/delete		
输入参数	参数名	类型	说明
	id	Int	公告 id
返回参数	code	Int	状态码
	msg	String	状态信息

（3）模块界面设计

医生或研究人员提交反馈和管理员查看反馈信息界面分别如图 5-24 和图 5-25 所示：

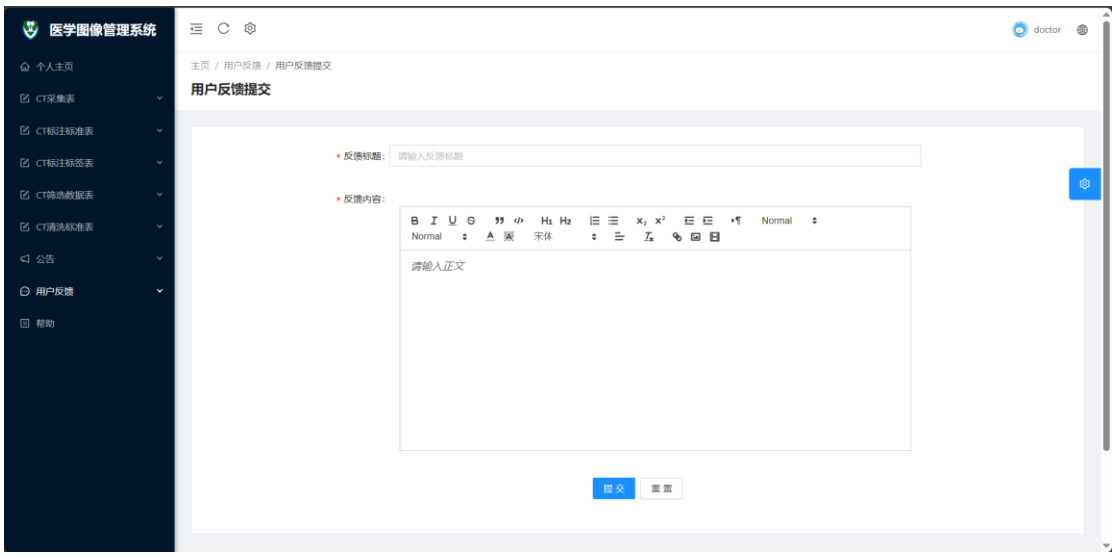

图 5-24 医生或研究人员提交反馈页面图

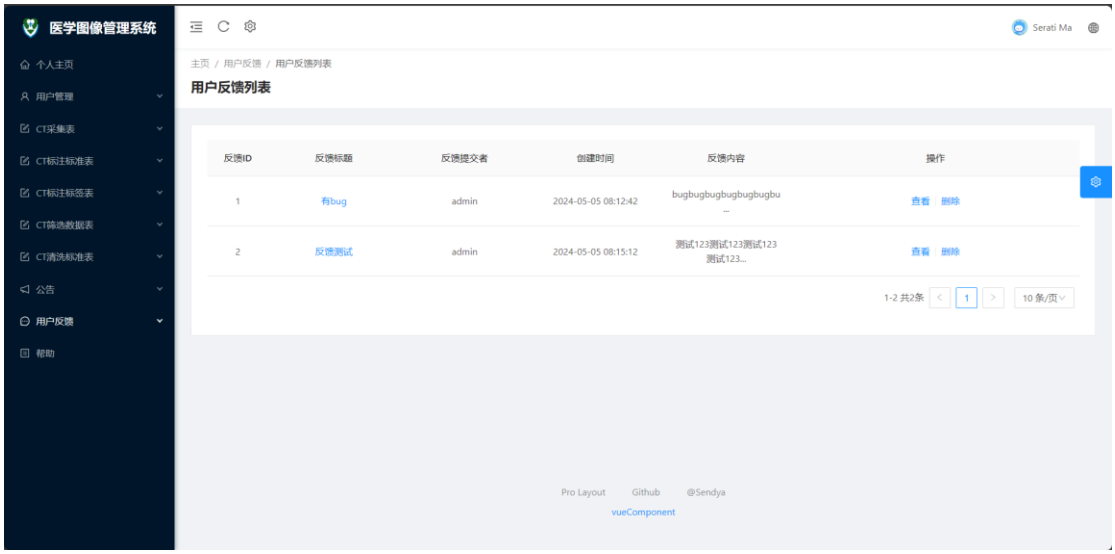

图 5-25 管理员查看反馈意见页面图

（4）程序流程图

用户反馈模块的流程：用户登录后，根据用户角色的不同，会有不同的操作页面。如果用户是医生或研究人员，将进入提交反馈意见页面，需要填写反馈表单，包括标题、内容这些信息，接着点击“提交”，即反馈提交成功；如果用户是系统管理员，将进入查看反馈意见列表页面，列表中的反馈意见按照提交时间的逆序排列，系统管理员还可以选择删除反馈意见，点击确认删除操作，即可将反馈意见从列表中移除。

用户反馈模块流程图如图 5-26 所示：

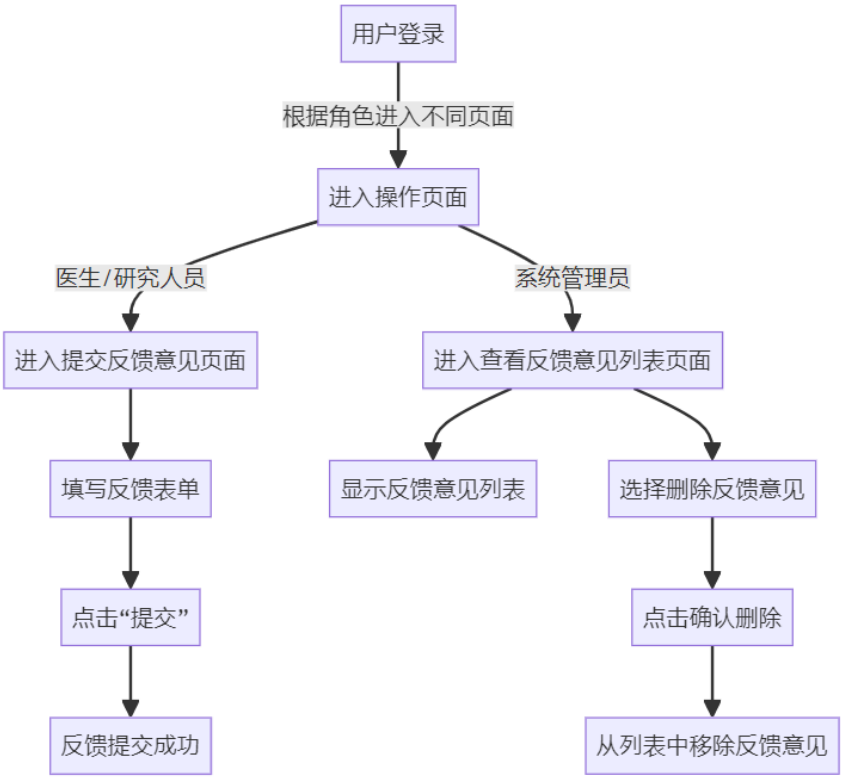

图 5-26 用户反馈模块流程图

5.2.5 医学图像信息管理模块

（1）功能描述

在这个模块中，医生可以将采集的数据信息提交并保存至数据库中，系统管理员可以根据需求添加 CT 标注标准表和 CT 筛选数据表。所有用户均可查看现有的数据采集表、CT 标注标准表、CT 标注标签表、CT 筛选标准表、CT 清洗标准表。以对医学图像数据集或数据库有更加便捷的查询和互动。

（2）模块接口设计

该模块的 API 接口如表-21、表-22、表-23、表-24、表-25、表-26、表-27、表-28 所示。

表 5-21 创建采集表接口说明

功能	创建采集表
请求方法	POST
URL	/api/collection/add

	参数名	类型	说明
输入参数	collectionDate	String	采集日期
	collectionHospital	String	采集单位
	collectionPlace	String	采集地点
	collectionMethod	String	采集方式
	collectionRequire	Int	关联需求 ID
返回参数	code	Int	状态码
	msg	String	状态信息

表 5-22 查询采集表接口说明

功能	查询采集表		
请求方法	GET		
URL	/api/collection/all		
输入参数	参数名	类型	说明
	coPOPageRO	Object	查询表相关的查询条件
返回参数	data	Object	包含所有的采集表信息
	code	Int	状态码
	msg	String	状态信息

表 5-23 创建 CT 标注标准表接口说明

功能	创建 CT 标注标准表		
请求方法	POST		
URL	/api/ctmark/add		
输入参数	参数名	类型	说明
	data	Object	CT 标注标准表的详细信息
返回参数	msg	String	状态信息
	code	Int	状态码

表 5-24 查询 CT 标注标准表接口说明

功能	查询 CT 标注标准表		
请求方法	GET		
URL	/api/ctmark/standard		
输入参数	参数名	类型	说明
	ctPOPageRO	Object	查询表相关的查询条件
返回参数	data	Object	包含所有的 CT 标注标准表信息

	code	Int	状态码
	msg	String	状态信息

表 5-25 查询 CT 标注标签表接口说明

功能	查询 CT 标注标签表		
请求方法	GET		
URL	/api/ctmark/label		
输入参数	参数名	类型	说明
	ctPOPageRO	Object	查询表相关的查询条件
返回参数	data	Object	包含所有的 CT 标注标签表信息
	code	Int	状态码
	msg	String	状态信息

表 5-26 创建 CT 筛选数据表接口说明

功能	创建 CT 筛选数据表		
请求方法	POST		
URL	/api/ctselect/add		
输入参数	参数名	类型	说明
	data	Object	CT 筛选数据表的详细信息
返回参数	msg	String	状态信息
	code	Int	状态码

表 5-27 查询 CT 筛选数据表接口说明

功能	查询 CT 筛选数据表		
请求方法	GET		
URL	/api/ctselect/search		
输入参数	参数名	类型	说明
	ctPOPageRO	Object	查询表相关的查询条件
返回参数	data	Object	包含所有的 CT 筛选数据表信息
	code	Int	状态码
	msg	String	状态信息

表 5-28 查询 CT 清洗标注表接口说明

功能	查询 CT 清洗标注表
请求方法	GET
URL	/api/ctwash/all

输入参数	参数名	类型	说明
	ctPOPageRO	Object	查询表相关的查询条件
返回参数	data	Object	包含所有的 CT 清洗标注表信息
	code	Int	状态码
	msg	String	状态信息

（3）模块界面设计

管理员或医生提交和查看数据采集表的界面如图 5-27、图 5-28 所示：

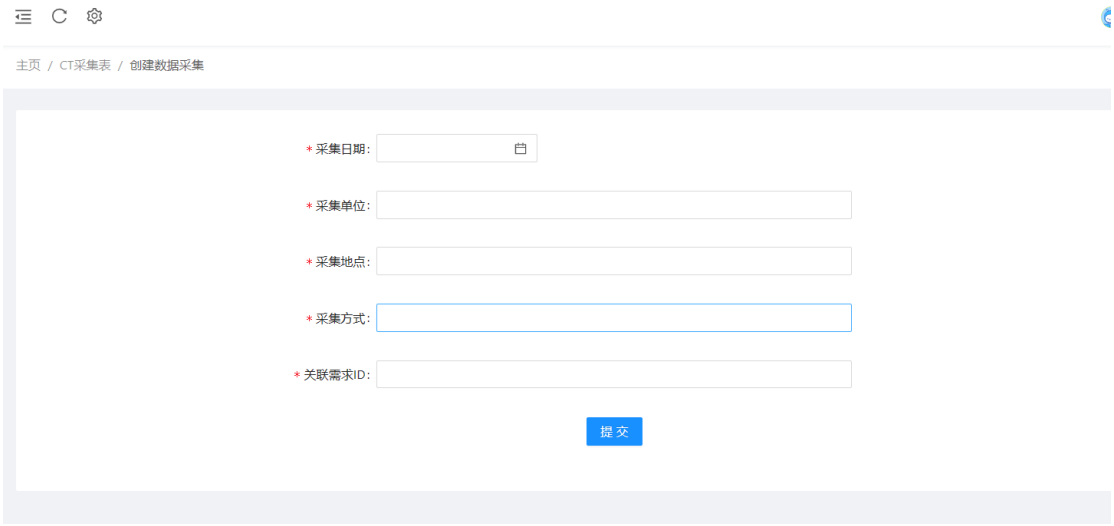

图 5-27 创建数据采集页面图

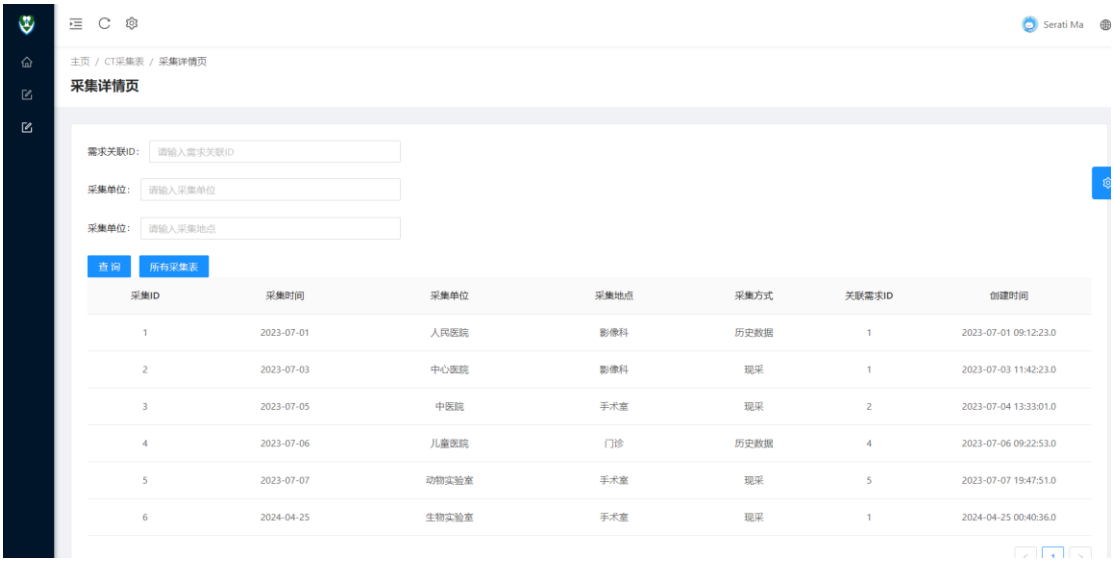

图 5-28 数据采集查询页面图

5.3 本章小结

本章从基础框架和功能模块两方面介绍了医学图像大数据管理系统前端系统的设计与实现，并且展示了部分关键代码及相关实现页面，其中分别介绍了系统前端的各个模块以及功能，包含了 6 大功能模块，涵盖了用户管理、网站公告、用户反馈、账号管理、系统管理、医学图像管理等需求。针对关键功能的实现，本章还展示了部分关键逻辑和代码的实现，满足了前期分析的功能需求。

6. 后端功能设计与实现

6.1 基础框架实现

6.1.1 后端目录结构搭建

后端采用 springboot 框架和 Mybatis 搭建而成，其整体结构目录如图 6-1 所示：

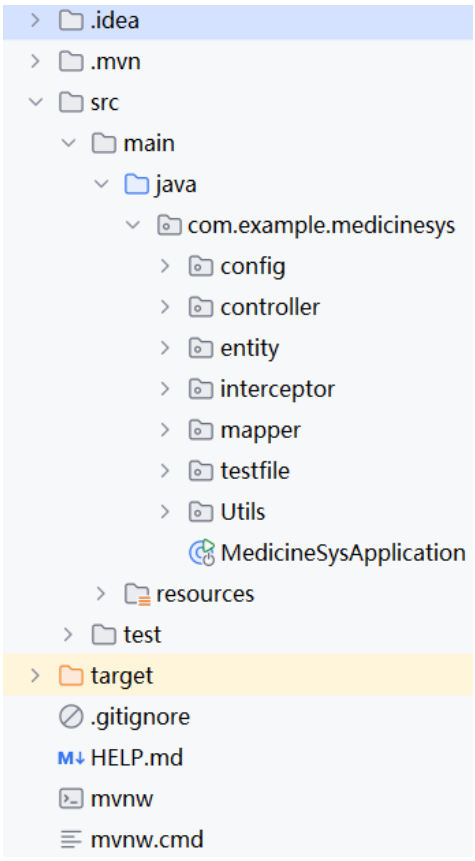

图 6-1 后端项目目录结构图

其中，后端项目主要文件存放于 src/main/java/com.example.medicinesys 目录下，该目录下的文件路径说明如表 6-1 所示：

表 6-1 目录表

路径	说明
config	用于存放如 Swagger 配置、拦截器配置等项目的配置文件
controller	存放控制器文件

entity	存放项目构建中的各个实体类
interceptor	存放拦截器文件
mapper	存放 mybatis 的各个 mapper 文件
testfile	存放项目测试文件
Utils	存放项目所用工具类

6.1.2 数据库结构搭建

本系统数据库采用 MySQL 搭建，具体库和表如图 6-2 所示：

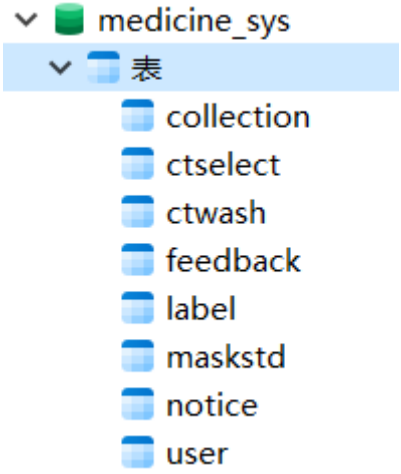

图 6-2 数据库结构图

其中，collection 存放采集表内容，ctselect 存放 CT 筛选数据表内容，ctwash 存放 CT 清洗标准表内容，feedback 存放用户反馈内容，label 存放 CT 标注标签表内容，maskstd 存放 CT 标注标准表内容，notice 存放公告内容，user 存放系统用户信息内容。

6.2 功能接口实现

6.2.1 用户管理

（1）接口设计

用户管理功能主要实现用户的登录、退出登录和修改密码。具体使用的方法

和接口如图 6-3 所示：

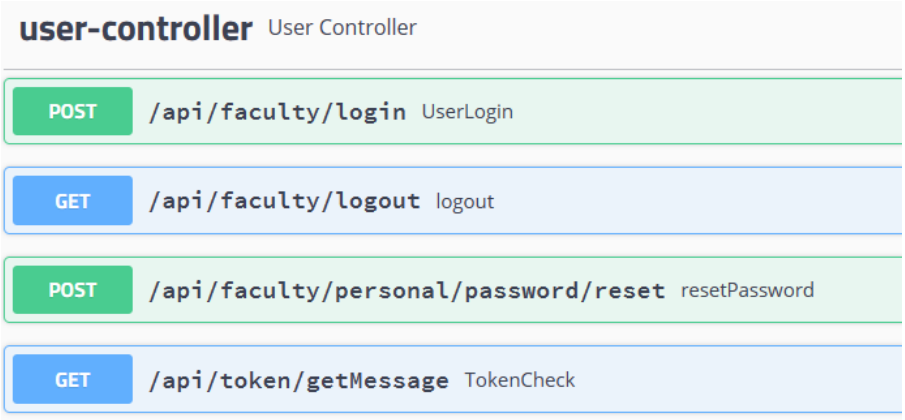

图 6-3 用户管理后端接口图

（2）程序流程

后端将前端传输的用户登录数据使用 POST 方法获取后，通过 Mybatis 提供的方法在数据库中搜索对应的信息，如果正确则向前端发送值为 1000 的 code 状态码，错误则发送值为 1001 的状态码，用于前端判断是否登录成功。退出登录则是使用 GET 方法将用户的 token 设置为过期并返回至前端。用户修改密码是使用 POST 方法将前端发送的用户信息表单进行处理，首先验证用户的旧密码是否正确，后再将新密码通过使用 Mybatis 提供的方法修改至数据库中，并向前端返回值为 1000 的 code 状态码。

6.2.2 账号管理

（1）接口设计

账号管理功能主要涉及账号的添加、查询、删除、用户密码的重置，具体使用的方法和接口如图 6-4 所示：

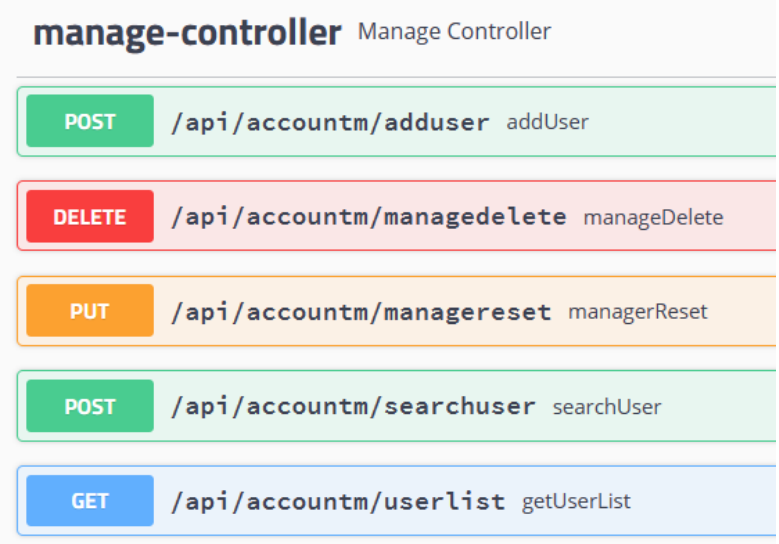

图 6-4 账号管理后端接口图

（2）程序流程

后端通过前端发送的不同的请求和接口进行需求的判断，将前端发送的关于账户信息的表单在处理后进行对应操作，并使用 Mybaits 的 mapper 将数据提交到 user 表中进行对应操作，以实现账号管理的目的。

6.2.3 医学图像信息管理

（1）接口设计

医学图像信息管理功能主要实现对前端发送的各个医学图像信息的表格进行对应需要的操作，如添加和查看，具体使用的方法和接口如图 6-5、图 6-6、图 6-7、图 6-8、图 6-9 所示：

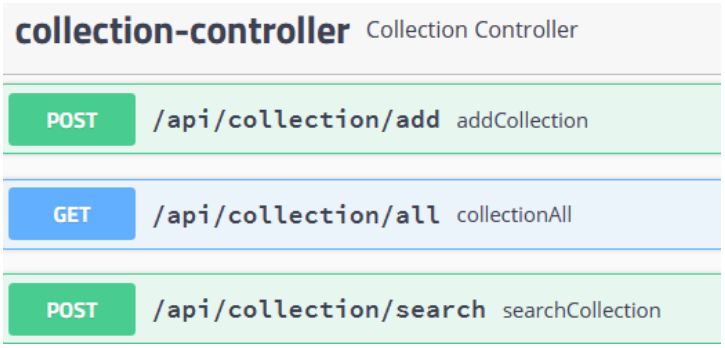

图 6-5 采集表后端接口图

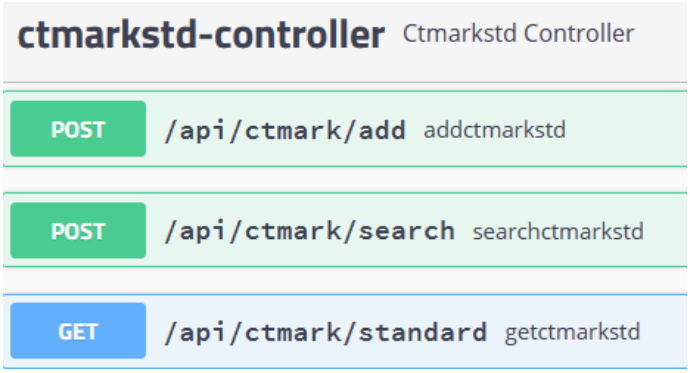

图 6-6 CT 标注标准表后端接口图

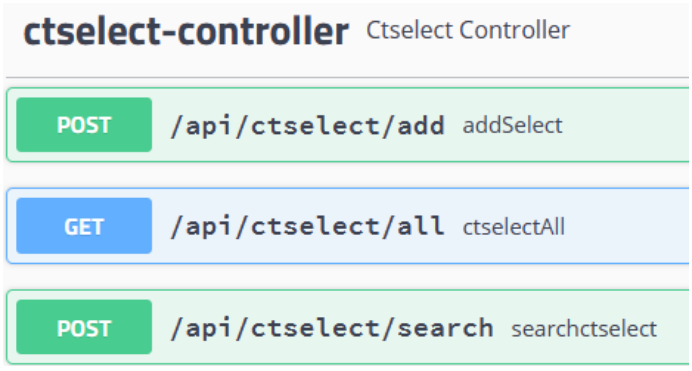

图 6-7 CT 筛选标准表后端接口图

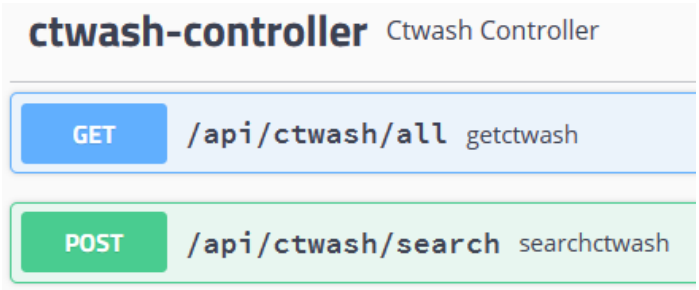

图 6-8 CT 清洗标准表后端接口图

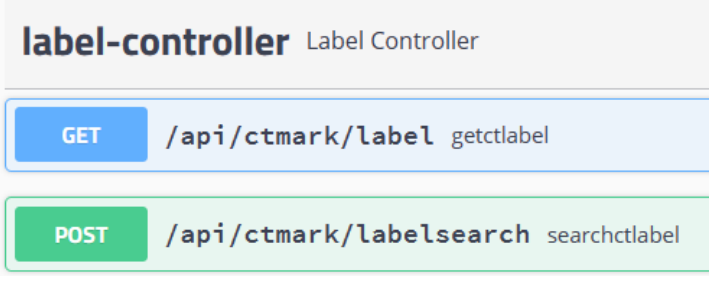

图 6-9 CT 标注标签表后端接口图

（2）程序流程

后端系统将前端发送的对应接口的表单进行处理，并调用对应的方法将表单内的数据添加至数据库中，从而实现医学图像信息数据的添加。当前端调用后端的查询数据接口时，后端系统在 mapper 中使用 select 方法从数据库中获取信息并转换为 json 格式发送到前端。

6.2.4 账户加密处理

（1）程序流程：

账户的加密处理是用户登录系统的重要功能，如果用户信息未经过加密和保护将对系统的安全和用户信息的安全造成威胁。本系统后端通过将用户在登录成功后的用户信息保存并加密于 token 中，并在每次前端调用后端接口时（除了用户登录）都会将存于请求头中的 token 进行验证，其中包括对用户权限的测试、信息的正确和过期时间的验证，并在验证无误后才对用户的调用进行对应的操作。

（2）关键代码：

生成 token 采用 JWT 的方法，将发行者、发行时间、签名类型和密钥等信息在保存后进行加密而生成，关键代码如图 6-10 所示：

```
public String generateToken(){  1 usage
    String token = jwtBuilder
            .setSubject(subObject) // 发行者
            .setIssuedAt(new Date()) // 发行时间
            .setExpiration(new Date(System.currentTimeMillis() + expired))
            .signWith(SignatureAlgorithm.HS256,secret) // 签名类型 与 密钥
            .compressWith(CompressionCodecs.DEFLATE)// 对载荷进行压缩
            .compact(); // 压缩一下
    return token;
}
```

图 6-10 后端生成 token 代码图

解析 token 则调用 JWT 自带的方法，在使用对应的加密密钥等信息后将保存在 token 内的加密信息提取出来，关键代码如图 6-11 所示：

```
public Claims check(String token){
    try{
        final Claims claims = Jwts.parser() JwtParser
                .setSigningKey(secret)
                .parseClaimsJws(token) Jws<Claims>
                .getBody();
        return claims;
    }catch (Exception e){
        return null;
    }
}
```

图 6-11 后端解析 token 代码图

在前端每次调用后端端口后，后端端口对应函数将调用 TokenCheck 类的方法将 token 解析并进行信息的验证，关键代码如图 6-12、图 6-13 所示：

```
public class TokenCheck {  22 usages
    //权限检测
    public static boolean tokencheck(String token){
        JwtUtils jwt = JwtUtils.getInstance();
        Claims claims = jwt.check(token);
        if (claims != null){
            return true;
        }else {
            return false;
        }
    }
}
```

图 6-12 后端验证 token 代码图

```
@PostMapping("/api/feedback/create")
public Map<String, Object> add(HttpServletRequest request, @RequestBody Feedback feedback) {
    Map<String, Object> map = new HashMap<>();
    if(TokenCheck.tokencheck(request.getHeader( s: "token"))){
        int aa =feedbackMapper.insert(feedback);
        if (aa==1){
            map.put("code",1000);
            map.put("msg","添加成功");
        }else{
            map.put("code", 1001);
            map.put("msg","添加失败");
        }
    }else{
        map.put("code", 1001);
        map.put("msg","非法token");
    }
    return map;
}
```

图 6-13 后端调用验证 token 代码图

6.3 本章小结

在本章中主要介绍了后端程序的框架构成、所采用的技术、数据库的构成和对各个功能的实现，并对后端程序的目录结构、数据库结构和功能实现进行了详细介绍。在对功能实现的介绍中，展示了用户管理、账号管理、医学图像信息管理和账户加密功能所对应的接口和使用的传输方法，以及各个功能的实现流程和部分关键代码，并满足了前期的功能需求。

7. 系统测试

7.1 测试环境

软件环境：Node.js、Chrome 浏览器、JDK8

硬件环境：客户端 PC 主机、服务器主机

网络：内网

7.2 功能测试

下面的功能测试均在医学图像大数据管理系统中进行测试，如表 6-1 所示为整体系统功能测试：

表 6-1 整体系统功能测试

测试模块	测试内容	功能要求	用户角色	测试结果
登录模块	用户登录	已导入信息的用户能够正常登录	全体用户	正常
个人首页模块	页面展示	不同用户角色对应不同权限的导航栏，首页展示内容清晰完整	全体用户	正常
账号管理模块	管理用户账号	能够手工录入用户账号信息，支持批量导入用户账号，显示具体的用户账号信息列表	系统管理员	正常
医学图像管理	创建采集表	能够正常添加采集表信息，并返回成功提示	系统管理员、医生	正常
	查询采集表	能够正常显示所有采集表信息，能够成功实现采集表的高级搜索	全体用户	正常
	创建 CT 标注标准表	能够正常添加 CT 标注标准表信息，并返回成功提示	系统管理员	正常
	查询 CT 标注标准	能够正常显示所有 CT 标注标准表信息	全体用户	正常

	表			
	查询 CT 标注标签表	能够正常显示所有 CT 标注标签表信息	全体用户	正常
	创建 CT 筛选标准表	能够正常添加 CT 筛选标准表信息，并返回成功提示	系统管理员、医生	正常
	查询 CT 筛选标准表	能够正常显示所有 CT 筛选标准表信息	全体用户	正常
	查询 CT 清洗标准表	能够正常显示所有 CT 清洗标准表信息	全体用户	正常
网站公告模块	发布系统公告	能够发布并修改系统公告，设置公告标题、内容、发布时间	系统管理员	正常
	查看系统公告	能够查看系统公告列表	全体用户	正常
用户反馈模块	提交反馈意见	用户能够提交使用过程中的问题和需求	医生 研究专家	正常
	查看反馈意见	管理员能够查看反馈问题列表、删除反馈意见	系统管理员	正常
系统帮助模块	查看系统帮助	能够以问答形式呈现常见问题和注意事项	全体用户	正常

7.2.1 登录模块测试

如表 6-2 所示为登录模块的功能测试，经测试，登录模块可正常运行，并且满足功能需求。

表 6-2 登录模块功能测试

测试模块	登录模块			
测试目的	测试登录功能是否正常			
预置条件	无			
用例编号	测试步骤	输入数据	预期结果	测试结果
1	输入正确的系统管理员账号密码	账号：admin 密码：admin	登录成功并跳转至系统管理员用户首页	符合预期
2	输入正确的文档管理员账号密码	账号：lyj 密码：lyj	登录成功并跳转至医生用户首页	符合预期
3	输入正确的任课教师账号密码	账号：abc 密码：abc	登录成功并跳转至研究人员用户首页	符合预期
4	输入错误的账号和正确的密码	账号：a2min 密码：admin1	登录失败并提示用户“账号或密码错误”	符合预期
5	输入正确的账号和错误的密码	账号：admin 密码：ad2in	登录失败并提示用户“账号或密码错误”	符合预期

7.2.1 医学图像数据上传模块测试

如表 6-3 所示为医学图像数据上传模块的功能测试，经测试，上传该模块可正常运行，并且满足功能需求。

表 6-3 医学图像数据上传模块测试

测试模块	医学图像文件上传模块			
测试目的	测试上传医学图像文件功能是否正常			
预置条件	无			
用例编号	测试步骤	输入数据	预期结果	测试结果
1	正常上传数据	输入每个应有的正确数据	上传成功”	符合预期
2	上传数据时缺少必须的数据	少输入一个必须的数据	上传失败	符合预期
3	上传时输入错误的数据格式	输入不符合要求的数据	上传失败	符合预期

7.3 兼容性测试

该系统具备跨设备、跨浏览器的兼容性，即在不同设备和浏览器上都能够正常使用和显示。该兼容性测试主要对不同浏览器、以及不同设备进行了测试，其中浏览器包括 Chrome、Firefox、Edge、QQ 浏览器等常用浏览器，设备则包括电脑、手机、平板这三种双设备。

本前端子系统采用了 Vue 以及 Ant Design 框架，采用了 Webpack 作为代码构建工具，并通过 Babel 将 JavaScrpt 代码转换为符合 ES5 语法规范、具有更好兼容性的代码。如下表所示，目前主流的浏览器均可兼容本系统，系统界面能够正常显示，各项功能也能正常使用，如表 6-4 所示，并推荐用户使用 Firefox 或 Chrome 浏览器。

表 6-4 浏览器兼容测试

浏览器	版本	测试结果
Google Chrome	112.0.5615.49	正常
Mozilla Firefox	111.0.1	正常
Microsoft Edge	112.0.1722.34	正常
Apple Safari	16.4	正常
QQ 浏览器	12.1.3106.400	正常
夸克浏览器	6.2.1.240	正常
UC 浏览器	13.9.7.1448	正常

7.4 本章小结

本章主要介绍了医学图像大数据管理系统的系统测试，测试分为功能测试和兼容性测试。

在功能测试中对系统各模块的所有内容进行了整体测试，对登录模块、个人主页模块、系统管理模块、账号管理模块、医学图像信息管理模块、网站公告模块、用户反馈模块、系统帮助模块等功能进行了整体系统的功能测试，同时也针对登录模块和医学图像数据上传模块进行了详细的功能测试，测试成绩较好，针

对不同内容在测试过程中达到了相应的功能要求。

兼容性测试主要针对不同设备、不同浏览器上系统的兼容性进行测试。通过测试结果可以看出，这套系统可以在主流浏览器上正常运行（如谷歌浏览器、火狐浏览器、QQ 浏览器和夸克浏览器等），并可以适应不同设备的屏幕尺寸和自动调整布局，保证用户在不同终端装置上的正常使用。

通过本章的测试，可以证明本系统在功能性和兼容性方面有良好表现，用户可以正常使用，通过这些测试不仅可以保证用户的体验，并且也可以保证系统的质量。

8. 总结与展望

8.1 结论

本文的研究工作是设计一个非商用的面对医学图像科研公司和医疗机构的医学图像大数据管理系统，旨在提供一个可学习的科研平台和提高医学图像管理效率。为了实现这一目标，本文采用了 Vue 框架搭建前端子系统，Springboot 框架搭建后端子系统，并应用了 MVVM 架构模式、Vuex 状态管理和 Vue=Rounter 路由管理等成熟的前端技术，构建了一套完整的系统方案。在界面设计引入了成熟的 Ant Design 组件库，搭建了简洁美观、交互性强的用户界面。在数据库交互使用了 MyBatis 库，确保了数据传输的安全和快捷。

通过使用该系统，医生和研究人员可以快速方便地上传、查阅、修改各个医学图像的信息表单；系统管理员可以更加方便地对账户进行管理，同时保证了系统的安全性，还可以发布通知公告实现关键信息的发布和传输。这一系统实现了对医学图像管理的科学化和规范化，为促进医疗信息规范化管理提供了参考。

总之，本研究工作通过设计和实现基于医学图像大数据的管理系统，为医学图像科研公司和医疗机构的规范管理医学图像数据提供了有力支持。系统运用方便快捷，提高了管理效率，简化了管理流程和方法。

8.2 展望

随着前端技术的不断发展，前后端框架不断更新和演进、移动端和响应式设计不断优化、WebAssembly 的应用等。这些技术的发展将为前后端分离系统提供更多创新的可能性，提升用户体验和性能。下一步的研究工作如下：

（1）性能进一步优化：通过代码分割、资源预加载和缓存策略等手段，提升前端应用的加载速度和响应性能，提供流畅的用户体验；

（2）框架优化：将现有的 Vue 框架进行框架迁移和重构，改造成微前端架构，这不仅确保模块间的独立性和可扩展性，也可以保持良好的开发体验和性能表现，同时也为后续的功能扩展提供了方便；

（3）安全性进一步加强：随着安全技术的不断发展，网站安全性也会随着

降低，所以需要定期对本系统进行安全漏洞扫描和代码审查，避免潜在的安全漏洞和弱点被利用，保证系统的安全性和稳定性；

（4）移动端优化：通过响应式设计和移动端优化策略，在注重移动端性能和用户体验的同时，确保在不同设备上提供良好的用户体验。

参考文献

【1】陈文.医学影像技术研究进展及其发展趋势[J].实用医学影像杂志,2016,17(03):254-257.DOI:10.16106/j.cnki.cn14-1281/r.2016.03.027.

【2】吴绯红,赵煌旋,杨帆,等. 医学影像+人工智能的发展、现状与未来[J].临床放射学杂志,2022,41(04):764-767.DOI:10.13437/j.cnki.jcr.2022.04.022.

【3】舒影岚,陈艳萍,吉臻宇,等.健康医疗大数据研究进展[J].中国医学装备,2019,16(01):143-147.

【4】代涛.健康医疗大数据发展应用的思考[J].医学信息学杂志,2016,37(02):2-8.

【5】蔡佳慧,张涛,宗文红.医疗大数据面临的挑战及思考[J].中国卫生信息管理杂志,2013, 10 (04): 292-295.

【6】Rahnama A H A. Distributed real-time sentiment analysis for big data social streams[C]// Proc of Int Conf on Control, Decision and Information Technologies (CoDIT). Piscataway, NJ: IEEE, 2014: 789-794.

【7】Contino S ,Cruciata L ,Gambino O , et al. IODeep: An IOD for the introduction of deep learning in the DICOM standard [J]. Computer Methods and Programs in Biomedicine, 2024, 248 108113-.

【8】岳和欣,桂路婷,湛永乐,等. 基于FHIR模型和CDASH模型的食管癌筛查队列数据共享技术规范的建立 [J]. 中华疾病控制杂志, 2021, 25 (10): 1198-1205+1219.DOI:10.16462/j.cnki.zhjbkz.2021.10.015.

【9】Life Image; Life Image Announces New Value-Added Solution That Further Enhances the Google Cloud Healthcare API to De-Identify Patient Information in Medical Imaging Data [J]. Journal of Engineering, 2019,

【10】吴佳男.IBM Watson Health:“大象”自有步调[J].中国医院院长, 2018,(23):81.

【11】Zebra Medical Vision; Apollo Hospitals Group Integrates Zebra Medical Vision's AI for COVID-19 Detection and Disease Progression Tracking [J]. Medical Letter on the CDC & FDA, 2020,

【12】Wenkai L ,Paul Q .The European Health Data Space: An expanded right to data portability? [J]. Computer Law & Security Review: The International Journal

of Technology Law and Practice, 2024, 52

【13】国务院印发《“十三五”国家信息化规划》[J].电子政务,2017,(01):40.

【14】谢成元.新冠病毒感染无症状患者的人口学、病毒学特征及预后调查[D].南方医科大学, 2023. DOI:10.27003/d.cnki.gojyu.2023.001158.

【15】李佳昌,张晨星,宋丕伟,等. 一种基于MVC架构的智能储物柜交互系统设计 [J]. 物联网技术, 2022, 12 (10): 64-65. DOI:10.16667/j.issn.2095-1302.2022.10.018.

【16】彭显雯.基于MVVM模式的响应式轻量级前端组件设计与实现[D].华中科技大学, 2017.

【17】胡新敏.Vue框架的渲染算法优化[D].大连理工大学,2020. DOI:10.26991/d.cnki.gdllu.2020.000807.

【18】季甜甜,刘冬冬. 基于Vue前端性能的研究与分析 [J]. 阜阳师范大学学报(自然科学版), 2024, 41 (01): 15-22. DOI:10.14096/j.cnki.cn34-1069/n/2096-9341(2024)01-0015-08.

【19】刘泓杰.基于 Vue.js 的视频营销创作平台的研究与设计[D]. 北京邮电大学, 2023. DOI:10.26969/d.cnki.gbydu.2023.000563.

【20】毛炎,任福,王功存,等.基于新型 Web 脚本样式框架构建城市规划编制信息平台——以ReactJS和Ant Design为例 [J]. 测绘与空间地理信息, 2017, 40 (08): 81-84.

致谢

本文能顺利完成，首先最要感谢的是我的导师黄炳顶教授，在整个毕业设计的过程中，黄炳顶老师给予了我悉心的指导和无私的支持，不仅在学术方面提供了宝贵的建议和指导，还在心理上给予了我极大的鼓励和帮助。没有黄炳顶老师的耐心指导和关怀，我将无法顺利完成这篇毕业设计。再次感谢黄炳顶老师对我的支持和关心！

此外，我还要感谢所有在我完成本文工作中为我提供过帮助的老师、同学、朋友和亲人，他们为我提供了宝贵的指导、支持和鼓励，为我提供了不断前行的动力和力量。特别是我的父母，他们始终支持我追求知识，给予我无尽的鼓励和关爱。感谢你们一直以来对我的鼓励和支持，让我能在人生的道路上勇往直前。

最后，我还要感谢过去的自己，正是因为过去的我在学习专业技术上的不断前行和探究，让我抓住每一个机遇，才使得这项研究工作得以顺利展开并最终取得成功。

衷心感谢以上所有人的付出和帮助，正是有了你们的支持和鼓励，我才能顺利完成这篇毕业设计。再次向大家表示深深的感谢和敬意！