软件设计文档

项目名称：升学指导

2023 年 6 月 11 日

姓名	于海龙
学号	2120220695
完成作业共花费	72小时
ChatGPT 完成任务比例	40%
实现软件功能预计还需要	20天

目录

1 背景介绍

1.1 升学问题现状

当学生决定进入研究生阶段时需要仔细考虑自己的职业目标、兴趣爱好和能力，以及选择最适合自己的学校、专业和导师。这项任务存在一些困难，因为存在太多的选择，并且需要考虑非常多不同的因素，例如学校的声誉、专业的难度、导师的研究领域等等。

当前的学生面临许多的升学问题，这些问题包括了学校选择、专业选择、导师选择等几个方面。作为学生，首先需要确定自己的研究方向，研究方向决定了学生未来的职业发展方向和就业前景；同时学生也会因为选择合适的研究方向更好地发挥自己的优势和兴趣，提高研究效率和成果质量。学校选择也是一项重要任务，因为学校的排名和声誉可以影响到研究生的就业前景和职业发展；与此同时学校所在地的交通、生活条件等也需要考虑，这些因素可能会影响到研究生的生活质量和学习效果。

1.2 软件开发目标

设计一款升学指导软件的目的是帮助学生更好地规划自己的未来，解决他们在升学过程中所面临的各种问题和挑战。目前市面上有很多升学指导的软件，如考研帮、留学Go、升学宝和升学通等。这些市面上的软件具有不同的特色，具体如下：

1. 考研帮：考研帮是一款专门为考研学生提供的软件，它提供了各种考研资讯、历年真题、模拟考试等服务。用户可以通过该软件获取最新的考研政策、报考指南、院校信息等，还可以参加各种在线课程和学习小组，与其他考生交流经验和心得。此外，考研帮还提供了一些实用的工具，如错题本、复习计划、考试倒计时等，帮助学生更好地备考。

2. 留学Go：留学Go是一款为留学生提供服务的软件，它提供了各国留学信息、申请流程、签证办理等方面的指导和建议。用户可以通过该软件了解各国的大学排名、专业设置、学费情况等信息，还可以查看各个大学的官方网站和招生简章，了解具体的申请要求和流程。此外，留学Go还提供了一些实用的功能，如语言测试、文书模板、面试技巧等，帮助学生更好地准备留学申请。

3. 升学宝：升学宝是一款综合性的升学指导软件，它提供了多种服务，包括选校咨询、专业选择、申请指导等。用户可以通过该软件了解各个大学的招生政策、录取标准、课程设置等信息，还可以查看各个专业的就业前景和发展趋势。此外，升学宝还提供了一些实用的功能，如职业测评、志愿填报、面试模拟等，帮助学生更好地规划自己的升学路线。

4. 升学通：升学通是一款为高中生和大学生提供升学指导的软件，它提供了多种服务，

包括选科咨询、专业选择、大学排名等。用户可以通过该软件了解各个大学的招生政策、录取标准、课程设置等信息，还可以查看各个专业的就业前景和发展趋势。

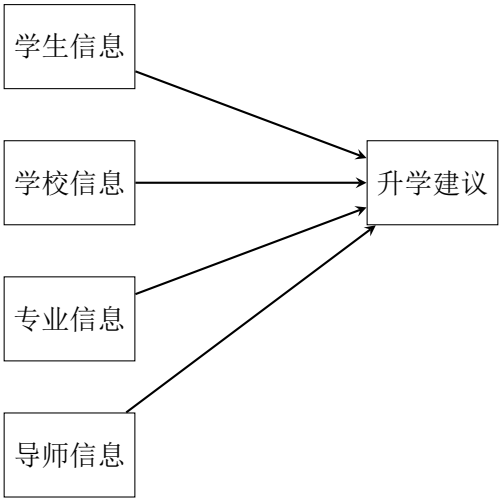

图 1: 获取升学推荐信息

这几款软件存在信息来源不够全面、界面设计不简洁直观和功能不全面等问题。因此我们需要为研究生设计一款简单明了的升学指导软件，如图1所示。这款软件期待的目标包括：简洁明了的界面设计、完备的数据库设计、高效的匹配算法设计和数据安全设计。一款升学指导软件可以帮助学生更好地选择适合自己的研究生阶段的学校、专业和导师，这对于他们的职业发展和人生规划非常重要，因此这款软件具有很高的实用性和价值。

2 项目概述

2.1 系统功能

该软件的主要功能是为用户提供升学建议，并提供学校、专业和导师的信息查询服务。除此之外，还需要提供用户基本信息管理、学校专业信息管理、后台管理和统计分析的功能。为了可以支持后续的学生培养计划，还可以在未来允许职场性格测试、职业发展分析等功能的接入。

2.2 业务描述

2.2.1 用户升学建议查询

用户升学建议查询的主要流程包括：

1. 输入学生信息；包括学习成绩、兴趣爱好、就职意向等，为系统分析数据提供基础支持。
2. 系统分析数据；根据学生输入的信息和数据库中的学校、专业和导师信息，执行匹配算法和推荐算法，对数据进行分析。
3. 生成推荐列表；根据上一步分析的结果，使用Top-K策略选择推荐列表，供用户筛选。
4. 完成选择；用户根据推荐列表完成升学选择。

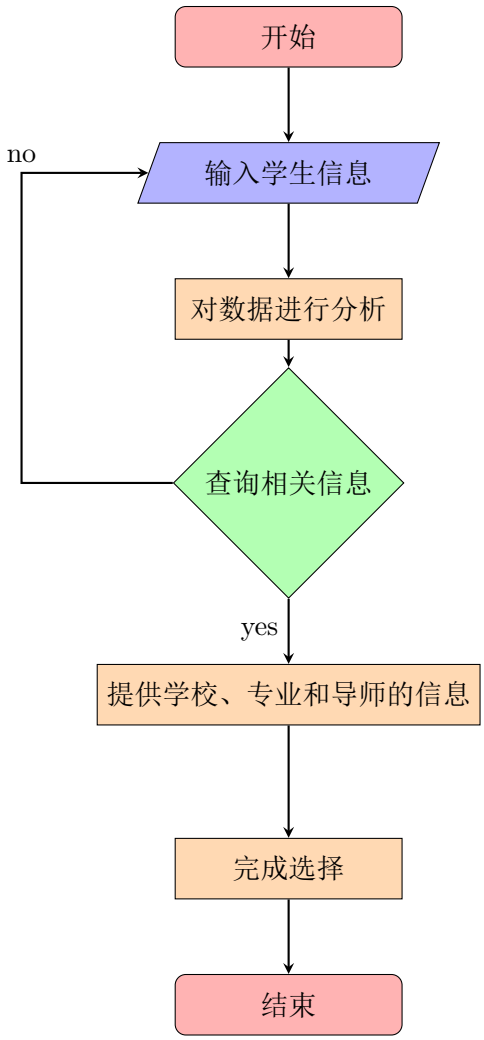

图 2: 输入学生信息查询学校、专业和导师信息流程图

当用户执行一次升学建议查询时，具体的流程如图2所示。

2.2.2 管理员更新学校专业信息

软件管理员更新学校专业和导师信息的主要流程包括：

1. 管理员输入学校信息；用于数据库查询学校信息并执行更新操作。
2. 管理员输入专业信息；用于数据库查询专业信息并执行更新操作。
3. 管理员输入导师信息；用于数据库查询导师信息并执行更新操作。

当管理员用户执行一次学校、专业和导师信息更新时，具体的流程如图3所示。

2.2.3 用户管理

用户注册、登录和登出的主要流程包括：

1. 用户注册；用户需要在注册页面输入用户名、密码、电子邮件地址等信息，并提交注册请求。系统将验证用户输入的信息，如果验证通过，则创建用户账户，并将用户信息保存在用户数据库中。如果验证未通过，则系统将提示用户重新输入信息。
2. 用户登录；用户需要在登录页面输入用户名和密码，并提交登录请求。系统将验证用户输入的信息，如果验证通过，则用户登录系统，可以访问系统提供的服务。如果验证未通过，则系统将提示用户重新输入信息。
3. 用户注销；用户在登录状态下可以选择注销登录，退出系统。用户注销后，系统将清除用户登录信息，确保用户的账户安全。

当用户执行一次账户注册和登录时，具体的流程如图4所示。

2.2.4 统计分析

管理员执行数据统计分析的主要流程包括：

1. 输入登录信息；管理员需要在登录页面输入用户名和密码，并提交登录请求。
2. 验证用户身份；系统将验证管理员输入的信息，如果验证通过，则管理员登录系统，可以访问统计后台数据。如果验证未通过，则系统将提示管理员重新输入信息。
3. 访问统计后台数据；管理员登录后，可以访问统计后台数据，并从数据库中检索相关的统计数据。系统将通过管理员账户和数据库来实现这个过程，以保护管理员的隐私和安全，确保管理员可以方便地查看统计数据。
4. 生成并输出报告；系统将检索相关的统计数据，并生成报告。报告可以显示统计数据的图表、表格和其他相关信息。生成的报告将输出给管理员。管理员可以查看报告，并根据报告中的统计数据做出正确的决策。

当管理员用户执行一次统计数据查看操作时，主要流程如图5所示。

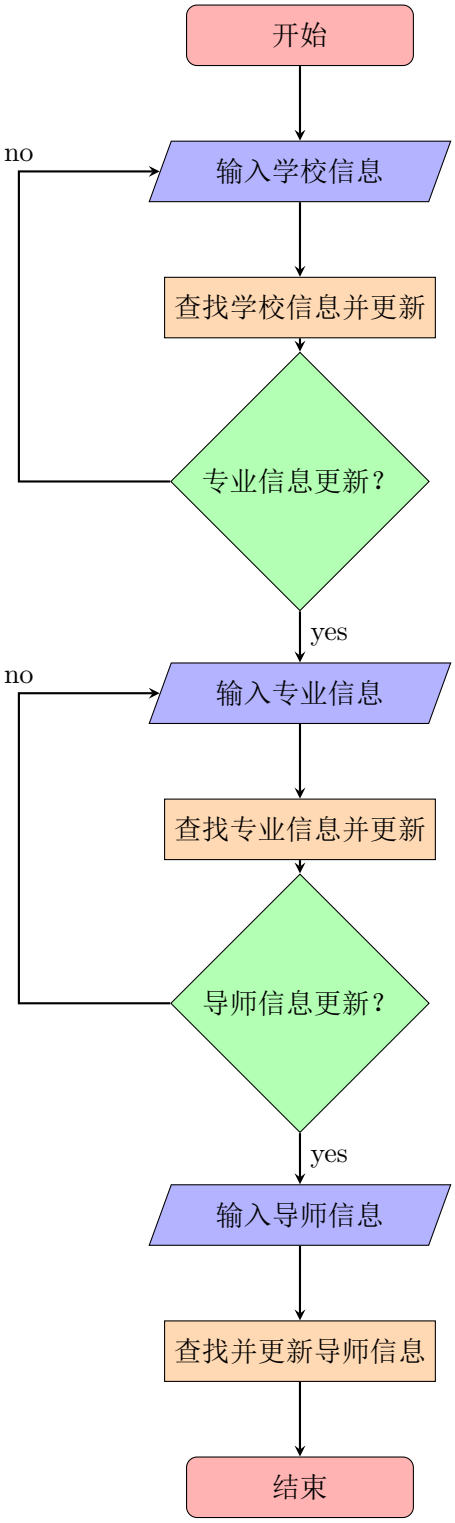

图 3: 更新学校和专业信息流程图

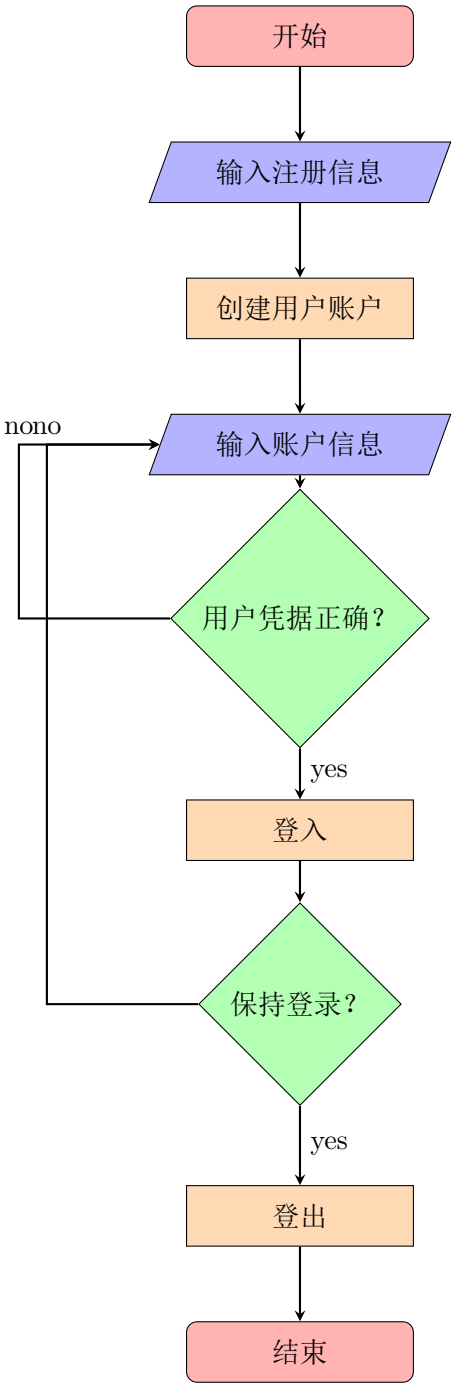

图 4: 用户注册与登录流程图

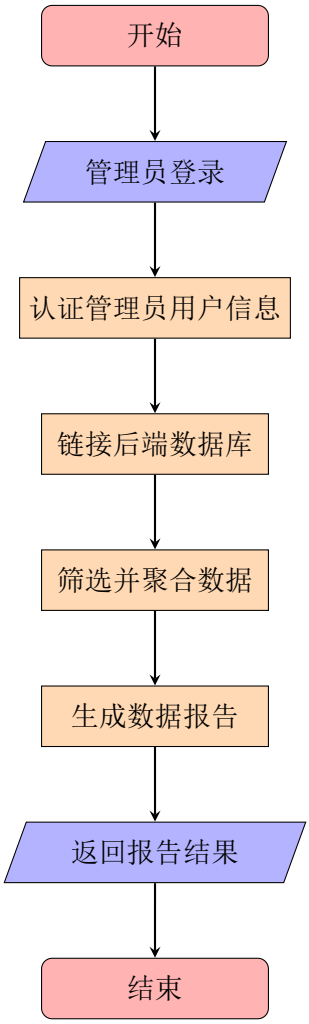

图 5: 管理员查看统计数据

2.3 数据流程描述

2.3.1 数据

在这个软件系统中需要记录保存和管理的数据主要包括：

1. 学生用户信息；学生用户需要提交学生升学相关的数据，包括学生的考试成绩、兴趣爱好、工作目标等内容。除此之外，还需要在数据库中存储用户的凭证信息以供登录使用。

2. 管理员信息；系统数据库需要存储管理员的凭证信息以供登录使用，并且需要存储管理员的操作记录日志，可以在数据出现问题是及时查找问题并恢复。

3. 学校信息；系统数据库需要保存学校信息，以供升学建议的生成。

4. 专业信息；系统数据库需要保存专业信息，专业信息与学校信息关联，以供升学建议的生成。

5. 导师信息；系统数据库需要保存导师信息，导师信息与学校和专业信息关联，以供升学建议的生成。

2.3.2 数据间的关系

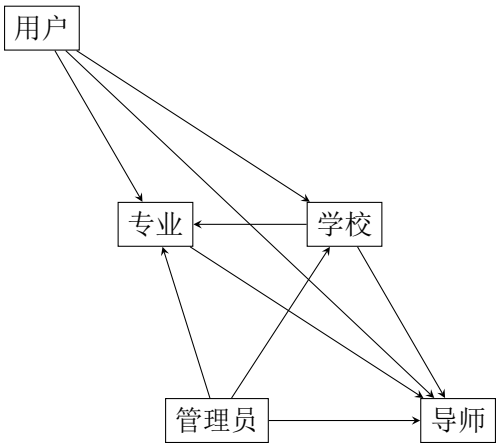

图 6: 存储数据间存在的关系

2.4 用户特点

2.4.1 学生用户

学生用户在升学和职业规划方面的需求可能非常复杂和多样化，例如他们可能需要了解不同学校的录取要求、专业课程设置、就业前景等等。学生用户可能对于数据和统计分析方面的内容比较感兴趣，例如各大学专业的录取率、毕业后就业情况等等，这些数据和统计分析可以帮助他们做出更加明智的决策。

虽然现在的学生都很早就开始接触电脑和互联网，但他们的对软件使用娴熟度仍然各不相同。有些学生可能非常熟练地使用计算机和软件，而另一些学生则可能需要花费更多的时间和精力来适应这些技术。除此之外，学生用户通常喜欢有趣、易于理解的界面和互动方式，因此这个软件系统中的学生用户可能会更喜欢具有可视化效果和游戏化设计的界面和体验。

总体而言，需要给学生用户群体提供简单易懂的可视化界面、清晰全面的推荐信息和稳定可靠的系统平台。

2.4.2 系统管理员

管理员用户通常是教育机构或学校的教育管理人员，具有一定的教育管理和升学规划方面的专业背景和知识。管理员用户需要负责升学指导软件系统的管理和运营，因此需要对这些管理员的操作进行明确完整的日志记录。除此之外，同学生用户一样，他们的对软件使用娴熟度也各不相同。

总体而言，需要给管理员用户群体提供简单易懂的可视化界面、稳定可靠的系统平台和完整的日志记录和恢复系统。

2.5 运行环境要求

2.5.1 前端用户界面

1. Web 网页前端UI，需要基于Blink内核的类Chrome浏览器、基于WebKit的Safari浏览器、基于Gecko内核的FireFox浏览器和IE浏览器。
2. Android 移动应用，需要不低于X版本基于 Linux 内核的 Android 操作系统，或不低于X版本的 HarmonyOS 操作系统。
3. iOS 移动应用，需要不低于X版本的iOS操作系统。
4. 各种小程序应用，与网页前端UI要求相同，需要基于Blink内核的类Chrome浏览器、基于WebKit的Safari浏览器、基于Gecko内核的FireFox浏览器和IE浏览器。

2.5.2 服务器程序

1. 网页服务，需要运行在基于 Linux 内核的操作系统，安装 Apache 和 Nginx 服务完成网站部署、反向代理和负载均衡。
2. 后端服务，需要运行在基于 Linux 内核的操作系统，安装 Java Development Kits 和 Spring Boot 框架。
3. 容器服务，需要运行在基于 Linux 内核的操作系统，安装 docker 支持容器运行、安装 Kubernetes 进行容器管理。
4. 消息队列和缓存，需要运行在基于 Linux 内核的操作系统，安装 Kafka 实现数据流分布式管理、安装 RabbitMQ 实现消息队列处理和事务处理、安装 Redis 实现高速缓存、安装 RocketMQ 实现负载均衡和故障转移。
5. 监控和日志系统，需要监控和日志工具来监控和记录系统的运行状态和性能，以便及时发现和解决问题，常见的监控和日志工具包括Nagios、Zabbix、ELK等。

2.5.3 数据库

1. 数据库管理系统，需要安装数据库管理系统（DBMS）管理数据，包括 MySQL 、PostgreSQL 、Oracle 、SQL Server 等。
2. 数据库连接：需要使用适当的连接方式来连接数据库，常见的连接方式包括 ODB C、JDBC 、ADO.NET等。
3. 编程语言和框架，需要使用编程语言和框架来编写数据库相关的代码，常见的编程语言和框架包括Java和Spring Boot。
4. 数据备份与恢复，需要安装自动备份软件，实现物理备份和逻辑备份，使用完全备份、增量备份、热备份和冷备份结合的方法保障数据库数据的安全。

3 基本设计概述

3.1 用户界面

3.1.1 Web前端界面

这个升学指导软件的前端Web系统需要具备以下要求：

1. 用户友好性：前端Web系统需要具备良好的用户界面和用户体验，易于操作和使用，使用户能够轻松地找到所需信息和功能。

2. 响应式布局：前端Web系统需要具备响应式布局，能够自适应不同的设备和屏幕大小，包括PC端、手机端、平板电脑等，以便用户能够在不同设备上进行访问和使用。
3. 多媒体支持：前端Web系统需要支持多种多媒体格式，包括音频、视频、图片等，以便用户能够获取到更加丰富的信息和资源。
4. 数据可视化：前端Web系统需要支持数据可视化，以便用户能够通过图表、表格等方式更加直观地查看和分析数据和统计信息。
5. 安全性和稳定性：前端Web系统需要具备安全性和稳定性，包括对用户数据和隐私的保护、防止恶意攻击、避免系统崩溃等方面的保障，保证用户的使用体验和数据安全。
6. SEO优化：前端Web系统需要进行搜索引擎优化，包括网站结构、页面内容、关键词等方面的优化，以便更好地被搜索引擎收录和推荐，提高网站的流量和用户数量。
7. 可扩展性和可维护性：前端Web系统需要具备可扩展性和可维护性，以便在未来能够根据需要添加新的功能和模块，同时保证代码的可读性和易于维护。

3.1.2 Android前端界面

1. 用户友好性：Android前端系统需要具备良好的用户界面和用户体验，易于操作和使用，使用户能够轻松地找到所需信息和功能。
2. 响应式布局：Android前端系统需要具备响应式布局，能够自适应不同的设备和屏幕大小，包括手机、平板电脑等，以便用户能够在不同设备上进行访问和使用。
3. 多媒体支持：Android前端系统需要支持多种多媒体格式，包括音频、视频、图片等，以便用户能够获取到更加丰富的信息和资源。
4. 数据可视化：Android前端系统需要支持数据可视化，以便用户能够通过图表、表格等方式更加直观地查看和分析数据和统计信息。
5. 安全性和稳定性：Android前端系统需要具备安全性和稳定性，包括对用户数据和隐私的保护、防止恶意攻击、避免应用程序崩溃等方面的保障，保证用户的使用体验和数据安全。
6. 多语言支持：Android前端系统需要支持多语言，以便用户能够选择自己熟悉的语言进行使用。
7. 可扩展性和可维护性：Android前端系统需要具备可扩展性和可维护性，以便在未来能够根据需要添加新的功能和模块，同时保证代码的可读性和易于维护。

8. 适配不同操作系统版本：Android前端系统需要适配不同的Android系统版本，以便用户能够在不同版本的Android系统上使用应用程序。

3.1.3 iOS前端界面

1. 用户友好性：iOS前端系统需要具备良好的用户界面和用户体验，易于操作和使用，使用户能够轻松地找到所需信息和功能。
2. 响应式布局：iOS前端系统需要具备响应式布局，能够自适应不同的设备和屏幕大小，包括iPhone、iPad等，以便用户能够在不同设备上进行访问和使用。
3. 多媒体支持：iOS前端系统需要支持多种多媒体格式，包括音频、视频、图片等，以便用户能够获取到更加丰富的信息和资源。
4. 数据可视化：iOS前端系统需要支持数据可视化，以便用户能够通过图表、表格等方式更加直观地查看和分析数据和统计信息。
5. 安全性和稳定性：iOS前端系统需要具备安全性和稳定性，包括对用户数据和隐私的保护、防止恶意攻击、避免应用程序崩溃等方面的保障，保证用户的使用体验和数据安全。
6. 多语言支持：iOS前端系统需要支持多语言，以便用户能够选择自己熟悉的语言进行使用。
7. 可扩展性和可维护性：iOS前端系统需要具备可扩展性和可维护性，以便在未来能够根据需要添加新的功能和模块，同时保证代码的可读性和易于维护。
8. 适配不同iOS系统版本：iOS前端系统需要适配不同的iOS系统版本，以便用户能够在不同版本的iOS系统上使用应用程序。

3.1.4 小程序前端界面

1. 用户友好性：小程序前端系统需要具备良好的用户界面和用户体验，易于操作和使用，使用户能够轻松地找到所需信息和功能。
2. 响应式布局：小程序前端系统需要具备响应式布局，能够自适应不同的设备和屏幕大小，包括手机、平板电脑等，以便用户能够在不同设备上进行访问和使用。

3. 多媒体支持：小程序前端系统需要支持多种多媒体格式，包括音频、视频、图片等，以便用户能够获取到更加丰富的信息和资源。

4. 数据可视化：小程序前端系统需要支持数据可视化，以便用户能够通过图表、表格等方式更加直观地查看和分析数据和统计信息。

5. 安全性和稳定性：小程序前端系统需要具备安全性和稳定性，包括对用户数据和隐私的保护、防止恶意攻击、避免应用程序崩溃等方面的保障，保证用户的使用体验和数据安全。

6. 多语言支持：小程序前端系统需要支持多语言，以便用户能够选择自己熟悉的语言进行使用。

7. 可扩展性和可维护性：小程序前端系统需要具备可扩展性和可维护性，以便在未来能够根据需要添加新的功能和模块，同时保证代码的可读性和易于维护。

8. 适配不同小程序平台：小程序前端系统需要适配不同的小程序平台，例如微信小程序、支付宝小程序等，以便用户能够在不同的小程序平台上使用应用程序。

3.2 后端程序设计

3.2.1 网页服务

1. 可靠性和稳定性：Web服务器需要具备高可靠性和稳定性，以保证系统的稳定运行和数据的安全。

2. 高并发处理能力：Web服务器需要具备高并发处理能力，能够同时处理大量用户请求和访问，以保证系统的响应速度和性能。

3. 安全性和防护能力：Web服务器需要具备安全性和防护能力，包括防止恶意攻击、保护用户数据和隐私、防止系统被攻击等方面的保障。

4. 负载均衡：Web服务器需要支持负载均衡，以便将用户请求和访问均匀地分配到不同的服务器节点上，提高系统的并发能力和性能。

5. 高可用性：Web服务器需要具备高可用性，包括故障自动切换、容错机制、备份和恢复等方面的保障，以保证系统的连续性和可用性。

6. CDN加速：Web服务器需要支持CDN加速，以便能够通过CDN分发内容和资源，提高系统的访问速度和性能。

7. 数据备份和恢复：Web服务器需要支持数据备份和恢复，以保证数据的安全和完整性。

8. 高效的日志管理：Web服务器需要具备高效的日志管理，能够对系统运行日志进行收集、存储、分析和展示，以便进行系统管理和优化。
9. 高效的缓存管理：Web服务器需要具备高效的缓存管理，能够对系统的缓存进行管理和优化，提高系统的性能和并发能力。

3.2.2 后端服务

1. 可扩展性和可维护性：后端服务需要具备可扩展性和可维护性，以便在未来能够根据需要添加新的功能和模块，同时保证代码的可读性和易于维护。
2. 安全性和稳定性性：后端服务需要具备安全性和稳定性，包括对用户数据和隐私的保护、防止恶意攻击、避免应用程序崩溃等方面的保障，保证用户的使用体验和数据安全。
3. 高并发处理能力：后端服务需要具备高并发处理能力，能够同时处理大量用户请求和访问，以保证系统的响应速度和性能。
4. 缓存管理：后端服务需要具备缓存管理，能够对数据和资源进行缓存处理，提高系统的响应速度和性能。
5. 数据库管理：后端服务需要对数据库进行管理，包括数据的存储、查询、更新和删除等操作，以便实现数据的持久化和管理。
6. RESTful API设计：后端服务需要设计和实现RESTful API接口，以便前端和其他系统能够通过API接口与后端服务进行通信和数据交互。
7. 消息队列：后端服务需要支持消息队列，以便能够实现异步处理、任务调度和事件驱动等功能。
8. 日志记录和监控：后端服务需要支持消息队列，以便能够实现异步处理、任务调度和事件驱动等功能。
9. 安全认证和授权：后端服务需要进行安全认证和授权，以保证系统的安全性和用户数据的安全。

3.2.3 数据库服务

1. 可靠性和稳定性：数据库服务需要具备高可靠性和稳定性，以保证数据的安全和可靠性。
2. 高性能和高可用性：数据库服务需要具备高性能和高可用性，能够支持大量并发访问和高效的数据处理，同时保证系统的连续性和可用性。
3. 数据库备份和恢复：数据库服务需要支持数据备份和恢复，以保证数据的安全和完整性。
4. 数据库性能优化：数据库服务需要进行性能优化，包括索引优化、查询优化、缓存优化等方面的优化，以提高数据库的响应速度和性能。
5. 数据库安全管理：数据库服务需要进行安全管理，包括用户认证和授权、数据加密、访问控制等方面的管理，以保证数据的安全和隐私。
6. 数据库监控和管理：数据库服务需要进行监控和管理，包括数据库状态监控、性能分析、容量规划等方面的管理，以保证数据库的稳定性和可用性。
7. 数据库升级和迁移：数据库服务需要支持数据库升级和迁移，以便能够根据需要进行系统升级和数据迁移。
8. 数据库备份和恢复策略：数据库服务需要制定数据库备份和恢复策略，包括备份周期、备份方式、备份存储和恢复流程等方面的策略，以保证数据的安全和可靠性。
9. 数据库容灾和故障恢复：数据库服务需要具备容灾和故障恢复能力，包括故障自动切换、容错机制、备份和恢复等方面的保障，以保证数据的连续性和可用性。

3.3 功能需求

3.3.1 升学指导

1. 业务定义与描述：该升学软件是一款提供升学辅助服务的应用程序，旨在帮助用户了解各种高等教育课程和学位的信息，包括申请条件、申请过程和入学要求。该应用程序将提供以下功能：学校搜索、课程搜索、申请流程指南、入学要求和申请条件。
2. 适用的用户类型，指操作本功能所需的授权：该应用程序面向认证的学生用户，通过学生授权即可使用。
3. 功能项的主要页面或样式：该应用程序将提供一个网页界面，其中包括学校搜索、课程搜索、申请流程指南、入学要求和申请条件等功能项，用户可以在该网页上进行操作。

4. 约束条件和特殊考虑：该应用程序不涉及需要计算机化的功能，所有信息都是基于已有的教育机构和课程信息提供的。由于教育机构和课程信息可能会发生变化，因此系统需要定期更新数据以保持准确性。

输入：学校名称、地理位置、学校类型等搜索条件；学生成绩和学生信息，用户的查找优先级设置。

输出：学校搜索结果，包括学校名称、地理位置、学校类型、排名等信息；申请流程指南，包括申请时间、材料提交、面试安排等信息；入学要求，包括学历要求、语言要求、考试要求等信息；申请条件，包括申请资格、申请材料、申请费用等信息。

在异常情况下，例如用户输入的搜索条件不符合要求或者搜索结果为空时，系统将会提示用户重新输入或者提供其他相关信息。

3.3.2 后台管理

1. 业务定义与描述：该升学软件后台管理系统是一款提供升学辅助服务的应用程序，旨在帮助管理员管理学校、专业、导师和申请流程信息。该应用程序将提供以下功能：学校管理、专业管理、导师管理、申请流程管理、用户管理和数据统计。

2. 该应用程序面向管理员用户开放，需要授权才能使用。

3. 功能项的主要页面或样式：该应用程序将提供一个网页界面，其中包括学校管理、专业管理、导师管理、申请流程管理、用户管理和数据统计等功能项，管理员可以在该网页上进行操作。

4. 约束条件和特殊考虑：该应用程序需要保护用户数据的安全性和隐私，因此需要采用安全措施来防止未经授权的访问。另外，由于学校、专业、导师和申请流程信息可能会发生变化，因此需要及时更新数据库信息，以确保数据的准确性和时效性。同时，需要遵守数据保护和隐私法规，保护用户隐私信息。另外，应用程序需要具有良好的可扩展性和可维护性，以适应未来的需求变化和技术升级。

输入：学校信息、专业信息、导师信息、申请流程信息、用户信息。

输出：学校信息、专业信息、导师信息、申请流程信息、用户信息、统计信息。

在异常情况下，例如用户输入的数据格式不正确或者保存失败时，系统将会提示管理员重新输入或者提供其他相关信息。

3.4 非功能需求

3.4.1 系统可用性

1. 系统可靠性：系统需要具备高可靠性和稳定性，以确保系统的持续运行和提供可靠的服务。
2. 用户友好性：系统需要采用用户友好的界面设计和操作流程，以提高用户的使用体验。
3. 可访问性：系统需要具备可访问性，以支持不同类型的用户（如残障人士）使用系统。
4. 系统可维护性：系统需要具备可维护性，以方便后续的维护和升级工作。
5. 系统安全性：系统需要具备高安全性，以保护用户的数据和隐私信息。
6. 系统可扩展性：系统需要具备可扩展性，以应对未来的业务需求变化和技术升级。

3.4.2 系统性能

1. 响应时间：系统响应用户请求的时间需要尽量短，以提高用户体验。
2. 吞吐量：系统需要具备处理大量数据的能力，以确保系统的高效性。
3. 并发处理能力：系统需要具备同时处理多个用户请求的能力，以支持多用户同时访问系统。
4. 资源利用率：系统需要合理利用资源，如CPU、内存、磁盘等，以确保系统的高效性和稳定性。
5. 性能测试：系统需要进行全面的性能测试，包括负载测试、压力测试、性能监控等，以评估系统的性能和稳定性。

3.4.3 安全性

该系统需要确保用户数据的安全性和隐私，防止未经授权的访问和数据泄露等问题。系统应采用安全措施，如加密技术、身份验证、访问控制和审计等，以保护用户数据的安全性和隐私。

1. 用户身份验证：为了保证系统的安全性，需要对用户进行身份验证，以确保只有授权用户才能访问系统。可以采用用户名和密码、短信验证码、指纹识别等多种身份验证方式，以增强系统的安全性。

2. 数据加密：对于敏感数据和隐私信息，需要采用数据加密技术进行保护。可以采用对称加密算法或非对称加密算法对数据进行加密，以确保数据在传输和存储过程中的安全性。

3. 访问控制：为了保护系统的安全性，需要对用户进行访问控制，以确保只有授权用户才能访问系统的特定功能或数据。可以采用基于角色或基于权限的访问控制机制，以确保系统的安全性。

4. 审计日志：为了跟踪系统的操作和异常行为，需要记录系统的审计日志。可以记录用户的登录和退出、操作行为、异常事件等信息，以便对系统进行监控和审计。

5. 异常处理：对于系统的异常情况，需要采取相应的处理措施，以保证系统的安全性。例如，系统出现SQL注入、跨站脚本攻击等安全漏洞时，需要采取相应的修复措施，防止黑客入侵和数据泄露等问题。

6. 数据备份：为了保证数据的安全性和可靠性，需要对数据进行备份。可以采用定期备份、增量备份等方式，以确保数据在系统故障或数据丢失的情况下能够及时恢复。

3.4.4 可靠性

该系统需要保证高可靠性，确保系统运行的稳定性和可用性。系统应具有故障恢复和备份机制，以确保在系统故障或数据丢失的情况下，能够及时恢复系统运行和数据。

1. 故障恢复：为了保证系统的可靠性，系统需要具备故障恢复的能力。当系统出现故障时，应能够及时发现并采取相应的措施进行恢复，以保证系统可以正常运行。例如，可以采用备份机制、冗余设计等方式增强系统的故障恢复能力。

2. 数据备份和恢复：为了保证数据的可靠性和完整性，需要对数据进行备份和恢复。可以采用定期备份、增量备份等方式，以确保数据在系统故障或数据丢失的情况下能够及时恢复。同时，需要测试数据恢复的可行性和正确性，以确保数据能够完全恢复。

3. 系统监控和报警：为了及时发现系统的异常情况，需要对系统进行监控和报警。可以采用实时监控、日志审计等方式，及时发现系统的异常情况并进行处理。同时，需要设置报警机制，当系统出现异常时，能够及时通知相关人员进行处理，以保证系统的可靠性和稳定性。

4. 代码质量和测试：为了保证系统的可靠性，需要具备高质量的代码和测试。可以采用代码规范、代码审查等方式，确保代码的可读性、可维护性和可靠性。同时，需要进行全面的测试，包括单元测试、集成测试、系统测试等，以确保系统的功能和性能均符合要求。

5. 负载均衡：为了保证系统的可靠性和性能，需要采用负载均衡技术，以均衡系统的负载和流量。可以采用硬件负载均衡器或软件负载均衡器，以确保系统的高可用性和稳定性。

6. 高可用性设计：为了保证系统的可靠性和稳定性，需要采用高可用性设计。可以采用冗余设计、灾备设计等方式，以确保系统在出现故障时能够自动切换到备份系统，保证系统的可用性和稳定性。

7. 容错性和可恢复性：为了保证系统的可靠性和容错性，需要采用容错和可恢复性设计。可以采用异常处理机制、事务处理机制、数据校验等方式，以确保系统在出现异常情况时能够自动恢复并保证数据的完整性和正确性。

3.4.5 可维护性

该系统需要具有良好的可维护性，以保证系统的可持续发展。系统应具有清晰的代码结构、易于维护和扩展的架构设计和完善的文档和注释，以方便后续开发和维护工作。

1. 可读性和可维护性：为了保证系统的可维护性，需要具备高质量的代码和易于维护的结构。可以采用代码规范、代码注释等方式，提高代码的可读性和可维护性。同时，需要对代码进行优化和重构，以确保代码的可扩展性和可维护性。

2. 设计模式和架构：为了保证系统的可维护性，需要采用合适的设计模式和架构，以提高系统的可扩展性和可维护性。可以采用MVC、MVP、MVVM等设计模式，以及分层、微服务、SOA等架构模式，以确保系统的可维护性和可扩展性。

3. 测试和调试：为了保证系统的可维护性，需要进行全面的测试和调试工作。可以采用自动化测试、单元测试、集成测试、系统测试等方式，以及调试工具和技术，以确保系统的稳定性和可维护性。

4. 文档和知识库：为了保证系统的可维护性，需要建立完善的文档和知识库，以便后续的维护和升级工作。可以建立系统的设计文档、用户手册、API文档等，以及建立知识库和FAQ，以提供后续维护人员使用。

5. 版本控制和发布管理：为了保证系统的可维护性和升级能力，需要采用合适的版本控制和发布管理机制。可以采用Git、SVN等版本控制工具，以及自动化发布工具和流程，以确保系统的可维护性和升级能力。

6. 代码审查和重构：为了保证系统的可维护性，需要进行代码审查和重构工作。可以采用代码审查工具和流程，以及重构工具和技术，对系统的代码进行优化和重构，以提高代码的可读性、可维护性和可扩展性。

7. 合理的技术选型和架构设计：为了保证系统的可维护性，需要采用合理的技术选型和架构设计。可以采用成熟、稳定、易于维护的技术和工具，以及合理的架构设计，以确保系统的可维护性和稳定性。

3.4.6 可扩展性

该系统需要具有良好的可扩展性，以应对未来的业务需求变化和技术升级。系统应具有模块化的架构设计、可插拔的功能组件和灵活的配置选项，以方便系统的扩展和升级。

1. 设计良好的架构：为了保证系统的可扩展性，需要采用设计良好的架构，以支持系统的可扩展性。可以采用分层架构、微服务架构、SOA架构等，以确保系统的可扩展性和灵活性。
2. 模块化设计：为了保证系统的可扩展性，需要采用模块化的设计方式，以支持模块的独立开发、测试和部署。可以采用模块化的编程方式，以确保系统的可扩展性和灵活性。
3. 弹性设计：为了保证系统的可扩展性，需要采用弹性设计，以支持系统的自适应和自我修复能力。可以采用负载均衡、自动扩展、容错机制等，以确保系统的可扩展性和稳定性。
4. 高可用性设计：为了保证系统的可扩展性，需要采用高可用性设计，以支持系统的持续运行和可靠性。可以采用冗余设计、灾备设计等，以确保系统的可扩展性和稳定性。
5. 采用云计算和容器技术：为了保证系统的可扩展性，可以采用云计算和容器技术，以支持系统的弹性扩展和部署。可以采用云服务器、容器编排工具等，以确保系统的可扩展性和灵活性。
6. 技术选型和架构设计：为了保证系统的可扩展性，需要采用合适的技术选型和架构设计。可以采用成熟、稳定、易于扩展的技术和工具，以及合理的架构设计，以确保系统的可扩展性和稳定性。
7. 预测和规划：为了保证系统的可扩展性，需要进行预测和规划。可以通过对业务需求和用户需求的分析，预测系统未来的需求变化和业务增长，以进行合理的规划和扩展工作。

3.4.7 兼容性

该系统需要具有良好的兼容性，以适应不同的硬件和软件环境。系统应支持不同的操作系统、浏览器和设备，以确保系统能够在不同的环境下正常运行和展示。

1. 浏览器和操作系统兼容性：为了保证系统的可兼容性，需要考虑不同浏览器和操作系统的兼容性。可以采用标准化的Web技术和CSS样式，以确保系统在不同的浏览器和操作系统上都能正常运行。

2. 移动设备兼容性：为了保证系统的可兼容性，需要考虑不同移动设备的兼容性。可以采用响应式设计或移动优先的设计，以确保系统在不同的移动设备上都能正常运行。

3. 数据库兼容性：为了保证系统的可兼容性，需要考虑不同数据库的兼容性。可以采用标准化的SQL语句和ORM框架，以确保系统在不同的数据库上都能正常运行。

4. 第三方组件和集成兼容性：为了保证系统的可兼容性，需要考虑与第三方组件和服务的兼容性。可以进行充分的测试和集成工作，以确保系统与第三方组件和服务的兼容性。

5. 跨平台和跨语言兼容性：为了保证系统的可兼容性，需要考虑跨平台和跨语言的兼容性。可以采用标准化的API和协议，以确保系统在不同的平台和语言上都能正常运行。

6. 版本兼容性：为了保证系统的可兼容性，需要考虑不同版本之间的兼容性。可以进行版本管理和测试，以确保系统在不同版本之间的兼容性。

7. 安全兼容性：为了保证系统的可兼容性，需要考虑安全方面的兼容性。可以采用标准化的加密算法和安全协议，以确保系统在不同安全环境下都能正常运行。

3.4.8 隐私

隐私是指个人的个人信息、行为和身份等不被他人获取、使用和泄露的权利。在软件开发中，隐私是指系统需要保护个人信息和隐私，以确保用户的隐私权不受侵犯。

1. 数据保护和加密：为了保护用户的隐私，需要对用户的个人信息和敏感数据进行保护和加密。可以采用数据加密技术、SSL协议等方式，以确保用户的个人信息和敏感数据不被他人获取和使用。

2. 访问控制和权限管理：为了保护用户的隐私，需要对用户的个人信息和敏感数据进行访问控制和权限管理。可以采用RBAC、ABAC等访问控制机制，以确保用户的个人信息和敏感数据只被授权的人员访问和使用。

3. 匿名化和脱敏：为了保护用户的隐私，需要对用户的个人信息进行匿名化和脱敏处理。可以采用哈希算法、脱敏工具等方式，以确保用户的个人信息不被他人获取和使用。

4. 合规和法律保护：为了保护用户的隐私，需要确保系统符合相关的隐私法规和标准。可以采用GDPR、CCPA等隐私法规和标准，以确保系统的合规性和用户的隐私权得到法律保护。

5. 安全审计和监控：为了保护用户的隐私，需要对系统进行安全审计和监控，以发现和防止潜在的隐私问题和漏洞。可以采用安全审计工具、防火墙等方式，以确保系统的安全性和用户的隐私权得到保护。

6. 用户知情权和选择权：为了保护用户的隐私，需要尊重用户的知情权和选择权。可以提供明确的隐私政策和用户协议，以告知用户系统如何收集、使用和保护用户的个人信息，以及提供用户选择是否参与的权利。

7. 数据删除和销毁：为了保护用户的隐私，需要对用户的个人信息进行删除和销毁。可以采用数据删除工具和安全销毁机制，以确保用户的个人信息得到有效删除和销毁，避免信息泄露和滥用。

3.5 数据存储与共享

对于医疗信息这种涉及个人隐私的数据来说，使用类似于比特币区块链网络一样的直接存储方式明显是不现实的，因此，区块链网络只能作为一个数据分布式存储的平台提供医疗数据共享服务，并且确保这些数据难以被某些节点恶意篡改。因此，这样的一个系统需要通过一些合理的加密手段以确保存储的数据难以被外界人员恶意读取。

为了提供可靠的、防篡改的医疗数据存储，可以使用对数据进行非对称加密存储的联盟链区块链网络对医疗数据和个人健康数据等进行存储，通过管理推送软件对数据进行加密解密、读写和传输，并对数据的读写行为进行监管、记录，以便于为普通用户和医务人员提供即时可靠的医疗数据访问服务。同时，为保障用户个人信息不被恶意泄露，在非特殊情况下，（特殊情况指患者需要紧急救治，并且无相关人员能够及时全面提供患者的有效信息），用户的医疗信息需要在用户或其法定监护人授意的情况下才能被医院获取。

3.6 数据安全与制约机制

在一般情况下，医院读取用户的医疗数据需要用户或其法定监护人同意并授权才能访问，管理软件会对用户的个人身份信息进行一定程度的脱敏处理，在完成整个就诊流程后，数据访问将会被禁用，用户也可以在就诊流程中结束医院的数据访问，因此在这样的流程中，数据的安全可以得到一定程度上的保障。

考虑到特殊情况的存在，（如患者等待接受急救等），用户无法及时为医院读取数据提供授权，医院又急需用户的医疗数据，可以允许医院临时调用用户的重要医疗信息和近期病历记录。如果无限制地允许医院临时调用用户的医疗信息，这无异于明文的比特币区块

链网络。因此对于医院临时对读取数据而言，可以设计一种制约机制，使得医院临时读取数据的行为受到合理的约束。对于医院的监管，需要一个可信任的组织通过软件管理平台根据不同医院的实际情况，如医院的类型、规模、就诊人次等信息进行分析判断，并为不同的医院设置不同的单位时间临时读取上限。软件管理平台可以根据医院的历史操作行为为管理者提供相对应的调整建议，根据历史记录为医院进行评分。用户在结束在某医院的治疗后，可以核实医院的数据读取流程，恢复医院的临时读取限额，用户不能完成操作，该院则可以通过向管理者申诉等恢复限额。借助用户、管理者、数据读取记录、评分的制约机制可以在一定程度上避免信息的恶意读取。

4 组织与用户

组织与用户是软件管理平台的一项重要管理内容，将不同的用户划分至不同的组织可以为用户提供不同的权限。

4.1 组织

4.1.1 医院

医院组织拥有对医疗数据的读取和写入的权限，临时读取可以使用任何可识别患者身份的信息获取，而普通的医疗数据读取则只允许使用软件提供的不包含个人身份信息的识别代码进行读取写入。需要医院提供有效的认证等信息，经过可信任组织认证后医院组织生效。医院组织接受可信任组织的管理，在非有效状态下，医院内的全部人员不能对他人的医疗数据进行任何操作。

4.1.2 管理机构

管理机构拥有对医院、科研机构和健康设备提供商的管理权限，拥有对医务人员、普通用户等用户的管理权限。管理机构是可信任组织，管理机构下的管理员可以通过软件平台直接对管理者进行管理。

4.1.3 科研机构

科研机构在授权状态下可以读取脱敏后的医疗数据，需要提供有效的认证信息，经过可信任组织的认证后该组织生效。在非有效状态下，科研机构内的全部人员，不能对他人的医疗数据进行任何操作。

4.1.4 健康设备供应商

健康设备提供商包括了医疗场所的检测设备制造商，个人用户健康检测设备提供商等。健康设备提供商需要提供有效的认证信息，经过可信任组织的认证后该组织生效。在非有效状态下，该供应商提供的所有设备不能执行医疗数据的写入。

4.2 用户与设备

4.2.1 医务人员

医务人员，涵盖了从护理人员到主任医师等属于医院组织的全部人员，这些人员需要提供有效的认证信息，经过管理机构可信任组织的认证后生效，可以在一定情况下读取或写入用户的医疗数据。医务人员提供的医疗数据包括但不限于挂号信息、病历信息、用药记录、检查记录、手术记录等等。针对于不同身份的医务人员，软件管理平台会赋予不同的权限，例如部分科室的护理人员、主任医师等可以拥有临时调用医疗数据的权限，而其他的医务人员则不具备这种权限。当医务人员本人的工作情况改变时，管理机构会同步更新医务人员的状态。

4.2.2 普通用户

普通用户，拥有对自身医疗数据的读取权限，可以对个人的健康检测设备进行关联或解除关联。普通用户需要提供有效的个人认证信息，经过管理机构认证后成为有效状态。普通用户可以授权或停止医院对个人的数据进行读取，确认或否认医院对个人数据临时读取的合理性，标定自己认为可以提供给科研的医疗数据。

4.2.3 管理人员

管理人员，拥有对其他各类人员认证、管理的权限，是由可信任组织认证的管理者。组织内的所有者可以管理组织内的管理员，管理员可以管理组织内的管理者。当其他各个类型的用户在实际状态发生变化时，管理人员需要对用户的状态进行同步更新。

4.2.4 科研人员

科研人员，可以在授权的情况下，获取脱敏处理后的医疗数据进行科学研究，获得的信息只包含用户认可标定为可用于医疗科研的数据，不包含任何的个人身份信息。

4.2.5 医院设备

医院设备，由健康设备提供商制造提供的设备关联医院组织并通过管理组织认证后生效。根据设备的不同功能，可以读取或写入用户的医疗数据，这些数据与医疗人员读取和写入的方式和格式一致。

4.2.6 健康监测设备

健康检测设备，由健康设备提供商制造提供的设备关联个人用户并通过管理组织认证后生效，根据设备的不同功能，可以写入用户不同的医疗数据。此类设备数据的置信度低于医院设备。

5 组建网络

5.1 区块链网络

5.1.1 医院服务器集群

根据医院所具有的特点，医院服务器集群可以包括参与记录数据读写记录的节点、参与记录个人健康数据的节点和参与记录病历数据的节点。数据存储在服务器节点上，但由于数据是加密存储的，医院对于这部分内容无法直接读取。

5.1.2 管理平台服务集群

根据管理组织所具有的特点，管理平台的服务器集群可以包括参与记录用户信息的节点、参与记录医院数据读写记录的节点和参与记录个人健康数据的节点。在组织与用户的真实状态更新后，管理机构的管理者可以更新位于管理平台集群上用户信息记录的状态、同时可以通过查看医院数据读写记录监控医疗数据的恶意读取等行为。

5.2 管理与信息服务网络

5.2.1 网络管理

由于基于区块链开发框架的网络组建需要修改大量的配置文件，执行很多重复性的命令，因此需要为管理者提供一个便于理解和操作的工具，简化过程的复杂度，降低失误率。

网络管理部署在管理者服务器上，管理者可以通过软件管理平台访问获取当前的组织与用户状态、各节点状态等属性信息，可以创建、编辑或删除组织信息或用户信息，为不同的区块链网络增删节点、通道，修改区块链网络配置文件与智能合约。

5.2.2 信息服务

当进行医疗数据的读取和写入时，由于存储在区块链网络中的个人医疗信息需要经过加密解密的过程才能形成真正可以阅读的医疗信息，又不能够将加密解密使用的公钥、私钥直接提供给使用者，因此需要信息服务来完成信息的交互操作。

信息服务部署于云服务器，通过客户端向云服务器发送请求信息，云服务器读取请求信息，向所需数据对应的服务器进行请求，并在对数据处理后返回至与该请求相关的所有客户端。

6 数据管理

6.1 数据分类

存储在软件中的全部数据包括组织下的组织识别代码、组织认证信息、组织状态标识、组织属性信息、机构和企业评分、医院评分；用户下的用户识别码、用户认证信息、用户密码、人员绑定关系、设备绑定关系、个人文本信息加密公钥、个人文本信息解密私钥、个人多媒体数据加密密钥，用户状态标记、设备状态标记信息；个人健康指标数据、个人病历数据、医疗数据读写记录等文本医疗数据和多媒体文件与哈希校验码等多媒体医疗数据。

6.2 数据存储

6.2.1 组织

针对于不同的组织，需要提供对应的认证信息，软件管理平台也为不同的组织设置了可供识别的组织类别代码。其具体的格式如下。

组织类别	类别代码	认证信息	状态标识	属性信息	其他信息
医院	11	医院识别码	有效/无效	详情	医院评分
管理机构	10	分支机构识别码	有效/无效	详情	管理员识别码
科研机构	00	机构识别码	有效/无效	详情	机构评分
健康设备供应商	01	企业识别码	有效/无效	详情	企业评分

其中识别代码的第一位代码代表了组织内最高权限成员在有效状态下，可否不经所获取信息对象的授权临时访问部分数据。对于不同的组织和用户具体可以访问的数据类型将会在权限管理部分进行具体描述。识别代码的第二位代码则代表了组织内最高权限成员在有效状态下是否具有数据写入的权限。

6.2.2 用户

针对于不同的用户，会存储或绑定不同的数据，软件管理平台为用户增加了不同的读写权限设置，结合用户所属的组织识别代码，组成不同用户的权限代码。读写权限设置分两位，第一位表示用户的读取权限，第二位表示用户的写入权限，具体权限代码可以访问的数据将在权限管理部分详细描述，管理人员权限代码代表的读写内容与其他类别用户不同，为用户、组织的信息与医疗数据读写记录。

用户信息的具体格式如下。

用户类别	权限代码	认证信息	状态标识	属性信息	识别码
医务人员	11xx	身份证	有效/无效	执业医师资格证	程序生成
普通用户	0010	身份证	有效/无效	公钥私钥密钥等	程序生成
管理人员	10xx	身份证	有效/无效	所在分支识别码	程序生成
科研人员	0000	身份证	有效/无效	所在机构识别码	程序生成
医院设备	01xx	设备ID	有效/无效	所在医院识别码	程序生成
健康监测设备	0100	设备ID	有效/无效	所属用户识别码	程序生成

其中针对于管理人员的识别码生成，有别于其他用户的程序随机生成，这与管理机构分支机构设置的管理员识别码有关，管理人员的识别码与其他用户的位数不同。

6.2.3 文本医疗数据

文本数据包括了基本健康数据、病历数据和医疗数据读写记录。基本健康数据涵盖了基础的体格检查数据、特殊的过敏源等信息；病历数据则包括了全部依据临床文档架构(CDA)格式保存的文字病历信息、多媒体病历的哈希校验码等基于文本格式存储的医疗信息。全部的内容以经过非对称加密的键值对的形式进行保存。具体涵盖的指标参数等内容可以参考附录。

对于医院的临时访问，软件设计可以将其认定为医院在紧急情况下对患者基本健康数据信息（包括各项身体指标和既往史等信息）和近期医疗、用药记录内容重要医疗信息提供给医院在紧急情况下的数据访问。

6.2.4 多媒体医疗数据

多媒体医疗数据包括但不限于仪器化验结果、影像资料和用药清单等，使用简要文字难以准确表达的复杂医疗数据。这部分数据需要进行AES对称加密后才能存储至部署有HDFS(Hadoop Distributed File System) 的文件服务器上，并且将原始文件数据的哈希校验码存储至病历数据中以便于后续的数据完整性验证。

6.3 数据写入

用户请求写入数据后首先要经过信息服务平台验证用户权限与当前状态的验证，当审查通过后，信息服务平台会开始执行数据写入的操作并将通知推送给数据被写入的对象，当数据完成写入将结果信息返回给提交写入请求的用户。临时写入的相关细则将会在安全性制约机制部分进行详细阐述。

6.4 数据读取

用户请求读取数据后首先要经过信息服务平台对验证用户权限与当前状态的验证，当审查通过后，信息服务平台会开始执行数据读取的操作并将通知推送给数据被读取的对象。当数据完成读取将结果信息犯规给提交读取请求的用户。临时读取的相关细则将会在安全性制约机制部分进行详细阐述。

6.5 授权机制

在非特殊情况下，读取或者写入数据首先需要获取被访问者的授权，在被访问者停止授权或者授权到期之前，可以持续进行读写操作，全部操作将会被平台记录。

在特殊情况下，临时读取写入采取制约机制，医院无需得到用户授权即可访问数据，但在特殊情况结束后，需要重新获得授权，用户可以通过或者拒绝，全部操作同样会被平台记录。

6.6 数据验证

鉴于医疗数据的特性，被读取数据应该确保无误。存储于区块链网络上的数据，可以确保数据的正确性，但是对于存储在分布式文件系统服务器上的文件来说，需要我们对服务器上存储的文件进行校验，即针对于这些文件，对解密前的文件进行CheckSum校验，对解密后的文件进行哈希校验码校验，借以确保文件的正确性。

7 权限管理

7.1 医务人员权限

医务人员从属于医院组织，当医院的状态转变为失效时，该医院组织下的全部医务人员的特殊权限将会失效。当医务人员被管理人员调整为无效状态时，医务人员的特殊权限将会失效。失效状态的医务人员仍然保有作为普通用户的使用权限。

对于所属不同医院组织和所属不同职位、不同科室或不同医院组织的医务人员来说，管理机构的管理者可以为医务人员提供不同的数据读写权限。具体的权限代码与对应权限见下。软件平台默认为医务工作者提供的权限参见附录。针对于临时读取的类型Ⅰ类、Ⅱ类的详细规则将会在安全性制约机制部分进行详细描述。

权限代码	授权读取	授权写入	临时读取	临时写入
1111	允许	允许	Ⅰ类	Ⅰ类
1110	允许	禁止	Ⅰ类	Ⅱ类
1100	禁止	禁止	Ⅱ类	Ⅱ类
1101	禁止	允许	Ⅱ类	Ⅰ类

7.2 用户权限

普通用户在有效状态下可以读取关联的普通用户数据，能够授权他人进行访问、通过或拒绝临时访问、关联、解绑个人健康设备、设置可供医学研究的医疗数据。用户在任何状态下都可以获取个人的医疗数据，通过或拒绝临时访问。用户的权限代码为0010，对于他人医疗数据，只能以授权形式访问，不能进行临时读写。

7.3 管理者权限

7.3.1 管理者

管理者维护着整个网络平台的组织、用户认证和状态信息。在管理者处于有效状态时，可以将组织的服务器节点添加到区块链网络中，可以认证组织和用户的信息，修改组织、用户的状态，可以协助用户修改当前的密码，用户的密钥对管理者不透明。

权限代码	管理员权限	组织管理	用户管理	记录管理
1011	是	允许	允许	允许
1010	否	禁止	允许	允许
1000	否	禁止	禁止	允许
1001	否	允许	禁止	允许

7.3.2 管理员

管理员具有普通管理者的全部权限，并在此基础上额外拥有了管理其所在分支机构下管理员的权限。软件管理平台存在一个超级管理员，由管理机构指定，可以任命或卸任管理员，在半数管理员通过后生效。同普通的管理者一样，全部的操作行为被平台记录。

7.4 科研人员权限

科研人员可以使用医学数据进行医学研究，在科研人员所在的组织有效且个人状态有效的情况下，可以向管理者申请调用与科研内容相关且用户标记为可以用于科学研究的数据，经管理者验证通过后科研人员可以获取经过脱敏处理、不含任何个人信息的医疗数据。

7.5 医院设备权限

一般情况下，在有效状态下的医院设备不具有授权读取用户医疗数据的权限。对用户的数据写入一般来说存在于检测类型的设备上，理论上非特殊情况下每一次检测均应当获取用户的授权。

权限代码	授权读取	授权写入	临时读取	临时写入
0111	禁止	允许	Ⅰ类	Ⅰ类
0110	禁止	禁止	Ⅰ类	Ⅱ类
0100	禁止	禁止	Ⅱ类	Ⅱ类
0101	禁止	允许	Ⅱ类	Ⅰ类

7.6 健康监测设备权限

健康监测设备在被用户绑定，且设备确认绑定至用户后，设备可以写入相应的健康监测数据。设备的权限代码为0100，意味着不允许任何的个人健康设备进行任何临时的读写操作。

8 共享性

8.1 共享流程

一般情况下医疗数据采取授权共享的模式，用户在医院就诊时，院方向用户请求所需授权的项目，告知用户授权项目的应用环节，用户同意授权后，医院有关医务人员和设备将会拥有对用户医疗数据读取或写入的权限，用户完成就诊后或在其他特殊情况下可以提前中止授权，数据将会不再允许被上述用户访问。

8.2 交互策略

医疗数据以加密后的形式存储在区块链网络或分布式文件存储网络上，但是将密钥提供给访问者依然容易导致数据的泄露，因此采用信息服务代理的方式，由请求者提供希望请求数据归属者的识别码和数据筛选条件，代理服务器代为请求数据，将完成解密并且不含有任何个人信息的数据传送回请求者的客户端。这样的交互策略同样适用于医疗数据写入的过程，仅改变数据筛选为需要写入的数据类型与详细内容。

9 安全性

9.1 制约机制

针对医院需要及时读写患者的医疗数据的特殊情况，软件管理平台允许医院内有关医务人员和医疗设备临时访问。管理机构可以根据医院的类型、就诊人流量、不同科室的类型和医务人员的身份等信息为医务人员和医院设备设置临时访问的权限，访问的类型可以分为Ⅰ类、Ⅱ类两种，在特殊情况下，医务人员和医院设备临时读写用户的病历数据时，Ⅰ类不会导致医院的评分下降，在超出Ⅰ类访问权限限额后变为Ⅱ类权限，此时在进行临

时访问和写入用户的病历数据时会导致医院的评分下降，两类访问均会被软件管理平台自动记录，当用户确认全部读写记录后，对于 I 类则恢复限额，对于二类则恢复评分。用户否认读写记录后，医院可以申请由管理机构的管理者进行仲裁，由管理者最终确定医院的读写行为是否合理。

9.2 文本数据加密

对于文本类型的数据采用非对称加密，这里可以使用椭圆曲线加密(ECC)的方法进行加密，将公钥私钥存储在用户信息中，当数据写入时，将公钥提供给数据写入方进行加密，读取时则由信息服务平台读取私钥将解密后的数据返回给请求方。

9.3 多媒体数据加密

对于多媒体类型的数据采用对称加密，这里可以使用高级加密标准(AES)的算法，将密钥存储在用户信息中，当数据写入时，将文件上传至分布式文件系统，由软件服务平台读取密钥加密，当数据读取时，由软件服务平台读取密钥解密，并将数据返回给请求方。

9.4 数据安全增强

面对可能存在的手写病历，可以采用卷积神经元网络(CNN)的方法验证手写病历是否为本人笔迹，借以增强数据的可靠性。

9.5 数据传输时协议

如果在数据传输的过程中使用应用层中传统的TCP/IP 协议传输数据请求和数据，这些数据将会面临被截获的风险。因此对于这些数据在网络中的传输不应当采用明文传输的方式，而对于传输的具体格式也和请求的具体内容也不应当以明文传输的方式进行。

10 附录

10.1 CDA

临床文档架构 (Clinical Document Architecture, CDA)，文本医疗数据信息中的个人病历、医疗机构就医记录等信息依据CDA的规范进行存储，并进行非对称加密。

10.2 文本医疗数据

10.3 医务人员默认权限

10.4 ECC

椭圆曲线加密算法(Elliptic Curve Cryptography, ECC)，非对称加密算法。用于对文本类型医疗数据的加密存储工作。在此处使用了以太坊的ECIES加密方法实现ECC加密。

```
package ECC

import (
   "crypto/ecdsa"
   "crypto/elliptic"
   "crypto/rand"
   "crypto/x509"
   "encoding/hex"
   "encoding/pem"
   "fmt"
   "github.com/ethereum/go-ethereum/crypto/ecies"
   uuid "github.com/satori/go.uuid"
   mathRand "math/rand"
   "os"
   "strings"
   "time"
)

func GenerateRandomKeyOfECC() (*ecies.PrivateKey, *ecies.PublicKey, error) {
   randKey := createRandomSalt(55)
   privateKey, err := ecdsa.GenerateKey(elliptic.P256(), strings.NewReader(randKey))
   if err != nil {
      return nil, nil, err
   }
   privateBytes, err := x509.MarshalECPrivateKey(privateKey)
   if err != nil {
```

```
    return nil, nil, err
}
privateBlock := pem.Block{
    Type: "ecc private key",
    Bytes: privateBytes,
}
var path string
path = "./"
path += uuid.Must(uuid.NewV4()).String()
path += "-private.pem"
privateFileOutput, err := os.Create(path)
if err != nil {
    return nil, nil, err
}
defer privateFileOutput.Close()
err = pem.Encode(privateFileOutput, &privateBlock)
if err != nil {
    return nil, nil, err
}
privateFileInput, err := os.Open(path)
if err != nil {
    return nil, nil, err
}
defer privateFileInput.Close()
privateKeyContent, err := privateFileInput.Stat()
if err != nil {
    return nil, nil, err
}
privateKeyBuffer := make([]byte, privateKeyContent.Size())
_, err = privateFileInput.Read(privateKeyBuffer)
if err != nil {
    return nil, nil, err
}
privateReaderBlock, _ := pem.Decode(privateKeyBuffer)
privateKeyBytes, err := x509.ParseECPrivateKey(privateReaderBlock.Bytes)
if err != nil {
    return nil, nil, err
}
privateKeyECIES := ecies.ImportECDSA(privateKeyBytes)
go RemovePemFile(path)
publicKey := privateKey.PublicKey
publicBytes, err := x509.MarshalPKIXPublicKey(&publicKey)
```

```
if err != nil {
   return nil, nil, err
}
publicBlock := pem.Block{
   Type: "ecc public key",
   Bytes: publicBytes,
}
path = "./"
path += uuid.Must(uuid.NewV4()).String()
path += "-public.pem"
publicFileOutput, err := os.Create(path)
if err != nil {
   return nil, nil, err
}
defer publicFileOutput.Close()
err = pem.Encode(publicFileOutput, &publicBlock)
if err != nil {
   return nil, nil, err
}
publicFileInput, err := os.Open(path)
if err != nil {
   return nil, nil, err
}
defer publicFileInput.Close()
publicKeyContent, err := publicFileInput.Stat()
if err != nil {
   return nil, nil, err
}
publicKeyBuffer := make([]byte, publicKeyContent.Size())
_, err = publicFileInput.Read(publicKeyBuffer)
if err != nil {
   return nil, nil, err
}
publicReaderBlock, _ := pem.Decode(publicKeyBuffer)
publicKeyBytes, err := x509.ParsePKIXPublicKey(publicReaderBlock.Bytes)
if err != nil {
   return nil, nil, err
}
publicKeyInner := publicKeyBytes.(*ecdsa.PublicKey)
publicKeyECIES := ecies.ImportECDSAPublic(publicKeyInner)
go RemovePemFile(path)
return privateKeyECIES, publicKeyECIES, nil
```

```
}

func EncryptECC(srcData string, publicKey *ecies.PublicKey) (cryptData string, err
    error) {
   encryptBytes, err := ecies.Encrypt(rand.Reader, publicKey, []byte(srcData), nil, nil)
   if err != nil {
      return "", err
   }
   cryptData = hex.EncodeToString(encryptBytes)
   return
}

func DecryptECC(cryptData string, privateKey *ecies.PrivateKey) (srcData string, err
    error) {
   cryptBytes, err := hex.DecodeString(cryptData)
   srcByte, err := privateKey.Decrypt(cryptBytes, nil, nil)
   if err != nil {
      return "", err
   }
   srcData = string(srcByte)
   return
}
```

10.5 AES

高级加密标准(Advanced Encryption Standard, AES)，对称加密算法。用于对多媒体医疗数据的加密存储工作。

```
package AES

import (
   "bytes"
   "crypto/aes"
   "crypto/cipher"
   "fmt"
   uuid "github.com/satori/go.uuid"
)

func GenerateRandomKeyOfAES() []byte {
   return []byte(uuid.Must(uuid.NewV4()).String()[:16])
}
```

```go
func decryptAES(cipherText, key []byte) []byte {
    block, _ := aes.NewCipher(key)
    blockSize := block.BlockSize()
    blockMode := cipher.NewCBCDecrypter(block, key[:blockSize])
    originData := make([]byte, len(cipherText))
    blockMode.CryptBlocks(originData, cipherText)
    originData = PKCS7UnPadding(originData)
    return originData
}

func PKCS7UnPadding(originData []byte) []byte {
    length := len(originData)
    position := int(originData[length-1])
    return originData[:length-position]
}

func encryptAES(originData, key []byte) []byte {
    block, _ := aes.NewCipher(key)
    originData = PKCS7Padding(originData, block.BlockSize())
    blockMode := cipher.NewCBCEncrypter(block, key[:block.BlockSize()])
    cipherText := make([]byte, len(originData))
    blockMode.CryptBlocks(cipherText, originData)
    return cipherText
}

func PKCS7Padding(originData []byte, blockSize int) []byte {
    padding := blockSize - len(originData)%blockSize
    padText := bytes.Repeat([]byte{byte(padding)}, padding)
    return append(originData, padText...)
}
```

10.6 架构图示

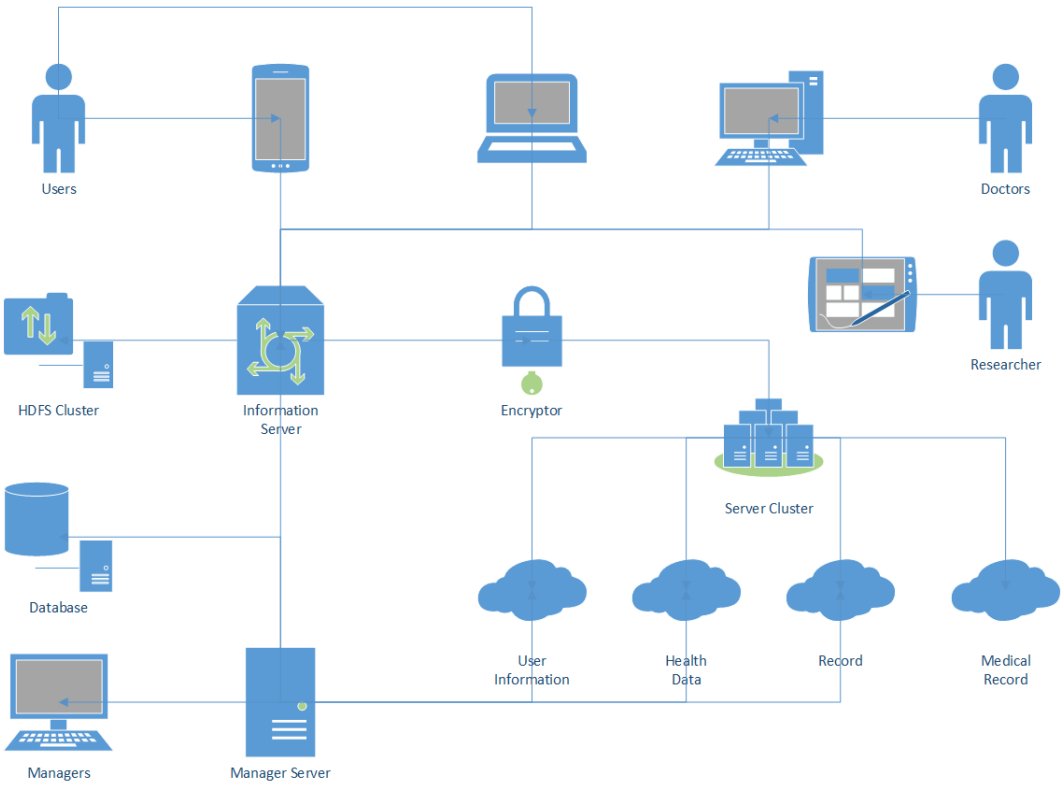

图 7: Architecture