

毕 业 设 计（论文）

题目 基于认知的心理测评系统的

*（二号、黑体、居中）*

设计与实现

学 院 计算机科学与工程

专 业 计算机科学与技术

班 级 116030701班

学生姓名 王孟鹏 学号11603990325

指导教师 陆艳军 职称

时 间 2020年3月

*（三号、仿宋、居中）*

**毕业设计(论文)格式规范化要求及装订顺序**

**一、**毕业设计（论文）装订要求（必须按以下顺序，用计算机打印）：

1、目录

2、摘要（300字左右的汉字，用中、英文书写，中文在前，英文在后）

3、正文（应含引言、结论等部分）

4、参考文献

5、附录

**二、**字数要求

毕业设计（论文）字数理工科、文科类均不少于1.2万字。

**三、**文字、图表要求

1、文字通顺，语言流畅，无错别字，不准请他人代写。

2、工程设计类题目的图纸应全部用计算机绘制，所有图纸均应符合国家技术标准规范。图表整洁，布局合理，文字注释规范。

3、毕业设计（论文）可双面打印或单面打印。

4、软件工程类课题应有程序清单（光盘）。

四、文献翻译

1、外文翻译可用钢笔书写，也可用计算机打印。

2、译文内容必须与课题（或专业内容）相关，并需注明详细出处。

3、译文原文（或复印件）应附在译文后备查，并与译文一起装订成册。

目录

[摘要 I](#_Toc38389308)

[Abstract II](#_Toc38389309)

[1 绪论 1](#_Toc38389310)

[1.1 研究背景和意义 1](#_Toc38389311)

[1.1.1 背景 1](#_Toc38389312)

[1.1.2 意义 1](#_Toc38389313)

[1.2 国内外现状 1](#_Toc38389314)

[1.3 相关技术 1](#_Toc38389315)

[1.3.1 MySQL 1](#_Toc38389316)

[1.3.2 SpringBoot 2](#_Toc38389317)

[1.3.3 undertow 2](#_Toc38389318)

[1.3.4 MyBatis 3](#_Toc38389319)

[1.3.5 vue 6](#_Toc38389320)

[1.3.6 Restful API 8](#_Toc38389321)

[2 系统需求分析 9](#_Toc38389322)

[2.1 可行性分析 9](#_Toc38389323)

[2.2 功能需求 9](#_Toc38389324)

[2.2.1 题库管理 9](#_Toc38389325)

[2.2.2 量表管理 10](#_Toc38389326)

[2.2.3 账号信息管理 10](#_Toc38389327)

[2.2.4 管理员权限及功能 10](#_Toc38389328)

[2.2.5 历史记录查看和打印 10](#_Toc38389329)

[2.2.6 结果打印功能 11](#_Toc38389330)

[2.2.7 即时通讯 11](#_Toc38389331)

[2.3 非功能需求 11](#_Toc38389332)

[2.3.1 系统稳定性 11](#_Toc38389333)

[2.3.2 界面友好性 11](#_Toc38389334)

[2.3.3 信息安全 12](#_Toc38389335)

[2.3.4 可维护性 12](#_Toc38389336)

[2.3.5 可扩展性 12](#_Toc38389337)

[3 系统设计 13](#_Toc38389338)

[3.1 设计原则 13](#_Toc38389339)

[3.2 网络拓扑图 13](#_Toc38389340)

[3.3 功能模块设计 13](#_Toc38389341)

[4 数据库设计 14](#_Toc38389342)

[4.1 E-R图 14](#_Toc38389343)

[4.2 表设计 14](#_Toc38389344)

[5 UI设计 15](#_Toc38389345)

[5.1 目的 15](#_Toc38389346)

[5.2 工具 15](#_Toc38389347)

[6 系统实现与测试 16](#_Toc38389348)

[6.1 开发环境搭建 16](#_Toc38389349)

[6.2 主要功能模块 16](#_Toc38389350)

[6.3 功能测试 16](#_Toc38389351)

[6.4 性能测试 16](#_Toc38389352)

[总结 17](#_Toc38389353)

[参考文献 18](#_Toc38389354)

[附录 19](#_Toc38389355)

# 摘要

# Abstract

# 1 绪论

## 1.1 研究背景和意义

### 1.1.1 背景

人口老龄化趋势在21世界将愈演愈烈，国内外形势令人担忧。世界大部分发达国家均已经开展相应国家计划，我国政府也对此高度重视，2013年9月国务院公布《关于加快发展养老服务的若干意见》，明确指出要努力使养老服务成为积极应对老龄化、保障和改善民生的重要举措，成为扩大内需、增加就业、促进服务行业发展、推动经济转型升级的重要力量。

根据联合国标准，目前已有60多个国家进入老龄化社会。老龄化已成为全球瞩目的问题。老龄化带来了其相关疾病如老龄痴呆，也称阿尔茨海默患病率的攀升，其起病隐匿，致病机制复杂，治疗费用昂贵，难治愈性、致残率、致死率高，给患者、家属及社会带来了沉重的负担。

### 1.1.2 意义

老龄化问题的关键是解决好养老，是老年人能够“老有所养、老有所医、老有所为、老有所乐、老有所学”，而病因复杂、病程漫长的老年痴呆、脑卒中、帕金森病等神经系统疾病发病率的上升加重了养老的难度。对于这类病程长、发病机制复杂的疾病，若能提早加大预防力度，弘扬中医学“上工治未病”的原创思想，防患于未然，不仅是老年人重视脑健康进而早期预防疾病发生的最佳策略，也是现实国情的需要。中国自古有敬老传统，尊重老人，就是尊重自己的未来，就是尊重民族的未来。俗话说家有一老，如有一宝。老年人群犹如社会历史的经验和智慧的活宝藏，对中华文化的延续意义深远。通过积极预防，不仅可以减少疾病，还可延长大脑耐力，是老年群体发挥积极作用，在发展经济的浪潮中进行老龄资源的社会化。

## 1.2 研究现状

## 1.3 相关技术

### 1.3.1 MySQL

MySQL采用的是客户/服务器体系结构，因此在实际使用的时候有两个程序：一个是MySQL服务器程序，指的是mysqlld程序，运行在数据库服务器上，负责在网络上监听并处理来自客户端的服务请求，然后根据这些请求去访问数据库的内容，再把相关信息回传给客户端；另一个是MySQL客户端程序，负责连接到数据库服务器，并通过发出命令来告知服务器它想要的操作。

MySQL的逻辑结构如下：



图 1.3.1

从逻辑结构图可以看出，MySQL内部大致分为三层：最上层是大部分基于网络的C/S服务都有的部分，不如连接处理、授权认证、安全等；第二层包括MySQL的很多核心服务功能，包括查询解析、分析、优化、缓存以及所有的内置函数（如时间、日期、数字和加密函数），所有的跨存储引擎的功能都在这一层实现，如存储过程、视图、触发器等；第三层包含了存储引擎，存储引擎负责MySQL中数据的存储和提取，是数据库中非常重要非常核心的部分，也是MySQL区别于其他数据库的一个重要特征。

MySQL支持插入式的存储引擎，可以根据实际情况选择最合适的存储引擎。不过目前对于绝大部分应用来说，MySQL默认的存储引擎InnoDB应该就是其最佳选择。

### 1.3.2 SpringBoot

Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作《Expert One-On-One J2EE Development and Design》。Spring是为了解决企业级应用开发的复杂性而创建的，使用Spring可以让简单的JavaBean实现之前只有EJB才能完成的事情。但是Spring不仅仅局限于服务器端开发，任何Java应用都能在简单性、可测试性和松耦合性等方面从Spring中获益。

使用Spring可以极大的简化Java开发，在使用了基于POJO的轻量级和最小侵入性编程、基于切面（AOP）和惯性生命式编程、通过切面和模板减少样式代码、通过依赖注入（DI）和面向接口实现松耦合后降低了Java开发的复杂性，提升了程序员的开发效率。

SpringBoot是所有基于Spring开发的项目的起点。SpringBoot的设计是为了让程序员能尽可能快的将spring应用程序跑起来并且尽可能少的减少配置文件。意思就是说Springboot不是新框架，它默认就配置了很多框架的使用方式，也就是说springboot是其它框架的一个承载体，通过使用springboot可以减少很多的配置项，对开发人员来说是十分友好的。

Springboot有四个重要的特性。第一个，SpringBoot Starter可以将常用的依赖分组进行整合，将其合并到一个依赖中，这样就可以一次性添加到项目的Maven或Gradle构建中；第二个，它拥有自动配置的功能，利用Spring4对条件化配置的支持，合理地推测应用所需要的Bean并自动化配置他们；第三，它有命令行接口功能（Command-line-interface，CLI），CLI发挥了Groovy编程语言的优势，并结合自动配置进一步简化Spring应用的开发；第四，Actuatir为SpringBoot应用的所有特性构建一个小型的应用程序。

### 1.3.3 undertow

Undertow是红帽公司开发的一款基于NIO的高性能Web嵌入式服务器，它有着轻量级、Servlet3.1支持、Websocket支持、嵌套性、长连接等特点，他是JBoss赞助的的一个Web服务器，是Wildfly应用程序服务器中的默认Web服务器。

为什么说他是一个轻量级的？他是一个Web服务器，但并不像传统的Web服务器有容器的概念，它由两个核心的Jar包组成，加载一个Web应用可以小于10M内容，对于性能有高要求的服务器来说是一个非常好的选择。在Websocket上是完成支持的，并能够实现Web应用巨大数量的客户端请求。为什么说它具有嵌套性，因为它不需要容器，只需要通过API即可快速搭建Web服务器。它的长连接表现在可以通过keep-alive的response header来支持，通过重用连接信息（connection details）来改善长连接的性能。

在默认情况下SpringCloud使用Tomcat作为内嵌Servlet容器，可启动一个Tomcat的SpringBoot程序与一个Undertow的SpringBoot程序，通过VisualVM工具进行比较就会发现Undertow的性能优于Tomcat。

Undertow是一个用java编写的、灵活的、高性能的Web服务器，提供基于NIO的阻塞和非阻塞API。它的架构是组合式的，可以通过组合各种小型的目的单一的处理程序来构建Web服务器，所以可以很灵活地选择完整的JavaEEServlet3.1容器或初级非阻塞程序处理。它的设计是可以完全嵌入的，具有简单用的编译接口，其生命周期完全有嵌入的应用程序控制。

### 1.3.4 MyBatis

**Mybatis介绍：**

MyBatis是支持定制化SQL、存储过程以及高级映射的一个非常优秀的持久层框架。它避免了几乎所有的JDBC代码和手动设置参数以及获取结果集。MyBatis可以对配置和原生Map使用简单的XML或注解，将接口和Java的POJOs（Plain Old Java Objects，普通的Java对象）映射成数据库中的记录。同时MyBatis是目前非常流行的ORM框架，功能也十分的强大，然而事实上实现却是很简单、优雅。

MyBatis的功能架构可以分为三层：API接口层提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理；数据处理层负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。他主要的目的是根据调用的请求完成一次数据库操作；基础支撑层负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为基础的组件，为上层的数据处理层提供最基础的支撑。

MyBatis的设计图如下：



图 1.3.2

**MyBatis和数据库交互方式：**

MyBatis和数据库的交互方式有通过传统的MyBatis提供的API和使用Mapper接口两种方式。

传统的交互方式通过传递Statement Id和查询参数给SqlSession对象，使用SqlSession对象完成和数据的交互，Mybatis提供了非常方便和简单的API，供用户实现对数据库的增删改查数据操作，以及对数据库连接信息和MyBa自身配置信息的维护操作。



图 1.3.3 MyBatis的工作模式

图1.3.3是创建一个和数据打交道的SqlSession对象，然后根据Statement Id和参数来操作数据库，这种方式固然很简单和实用，但是不符合面向对象语言的概念和面向接口编程的编程习惯。由于面向接口的编程是面向对象的大趋势，MyBatis为了适应这一趋势，增加了第二种使用MyBatis支持接口（Interface）调用方式。

使用Mapper接口的方式，将配置文件中的每一个<Mapper>节点抽象为一个Mapper接口，而这个接口中声明的方法和跟<Mapper>节点中的<select|updata|delete|insert>节点对应，即<select|updata|delete|insert>节点的id值为Mapper接口的方法名称，parameterType值表示Mapper对应方法的传入参数类型，而resultMap值对应Mapper接口表示返回值的类型或返回结果集的元素类型。

**数据处理层：**

数据处理层可以说是MyBatis的核心，从大方面将它主要完成两个功能：

1. 通过传入参数构建动态SQL语句。动态语句可以说是MyBatis框架非常优雅的一个设计，MyBatis通过传入参数值，使用Ognl来动态地构造SQL语句，使得MyBatis有很强的灵活性和扩展性。在这个过程中还有个非常重要的环节，那就是参数映射，参数映射是指对于Java数据类型和jdbc数据类型之间的转换，包括连个过程，分别为将java类型的数据转换成jdbc类型数据的查询阶段，可以通过preparedStatement。SetXXX（）来设置值和对resultset查询结果集的jdbcType数据转换成java数据类型。
2. SQL语句的执行以及封装查询结果集成List<E>。动态SQL语句生成之后，MyBatis将执行SQL语句，并将可能返回的结果集转换成List<E>列表。MyBatis在对结果集处理中，支持结果集关系一对多和多对一的转换，并且有两种支持方式，一种为嵌套查询语句的查询，还有一种是嵌套结果集的查询。

**框架支撑层：**

1. 事务管理机制

事务管理机制对于ORM框架而言是不可缺少的一部分，事务管理机制的质量也是考量一个ORM框架是否优秀的一个标准。对于数据库的事务而言，应该具有以下几点：创建（create）、提交（commit）\回滚（rollback）、关闭（close）。对应地，MyBatis将事务抽象成了Transaction接口。

1. 连接池管理机制

由于创建一个数据库连接所占用的资源比较大，对于数据吞吐量达和访问量非常大的应用而言，连接池的设计就显得非常重要。MyBatis将数据源DataSource分为三种分别为不使用连接池的数据源（UNPOOLED）、使用连接池数据源（POOLED）和使用JNDI实现的数据源（JNDI）。

1. 缓存机制

为了提高数据利用率和减小服务器以及数据库的压力，MyBatis会对于一些查询提供会话几倍的数据缓存，会将对某一次查询放置到SqlSession中，在允许的时间间隔内，杜宇完全相同的查询，MyBatis会直接将缓存结果返回给用户，而不用再到数据库中查找。

1. SQL语句的配置方式

传统的MyBatis 配置SQL 语句方式就是使用XML文件进行配置的，但是这种方式不能很好地支持面向接口编程的理念，为了支持面向接口的编程，MyBatis 引入了Mapper接口的概念，面向接口的引入，对使用注解来配置SQL 语句成为可能，用户只需要在接口上添加必要的注解即可，不用再去配置XML文件了，但是，目前的MyBatis 只是对注解配置SQL 语句提供了有限的支持，某些高级功能还是要依赖XML配置文件配置SQL 语句。

**MyBatis的优缺点：**

1. 优点：
2. 简单易学。本身很小且简单，没有任何第三方依赖，最简单安装只要两个Jar文件+配置几个sql映射文件，易于学习，易于使用，通过文档和源码，可以比较完全的掌握它的设计思路和实现。
3. 灵活。Mybatis不会对应用程序或者数据库的现有设计强加任何影响，sql写在xml里面，便于统一管理和优化。通过sql基本可以实现我们不使用数据访问框架就能实现的所有功能，甚至更多。
4. 解除sql与程序代码的耦合。通过提供DAL层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更容易维护，更容易进行单元测试，sql和代码分离，提高了可维护性。
5. 提供映射标签。支持对象与数据库的orm字段关系映射。
6. 提供对象关系映射标签，支持对象关系组建维护。
7. 提供xml标签，支持编写动态sql。
8. 缺点：
9. 编写SQL语句时工作量大，尤其是字段多、关系表多时。
10. SQL语句依赖于数据库，导致数据库移植性查，不能更换数据库。
11. 框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写，工作量也比较大，而且不太容易适应快速数据库修改。
12. 二级缓存机制不佳。

### 1.3.5 vue

Vue是一款用于构建用户界面的渐进式框架，与其他大型框架不同的是，Vue被设计为可以自底向上逐层应用。Vue的核心库值关注视图层，不仅易于上手，还很方便区与第三方库或已经存在的项目进行整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue也完全能够为复杂得单页应用提供驱动。



图 1.3.5 vue生命周期

### 1.3.6 Restful API

Restful架构风格最开始是由Roy T.Fielding（HTTP/1.1协议专家组的负责人）在2000年的博士学位论文中提出来的。HTTP就是这个架构风格的一个典型应用。从他出现的时候开始，他就因为其可以扩展的特性和简单的特性受到了广大开发者和架构师的青睐。一方面随着移动计算和云计算的兴起，有很多的企业都愿意在互联网上共享自己的数据、功能，在企业中Restful API（Restful Web服务）也逐渐超越SOAP成为实现SOAP的重要手段之一。REST即Respresentational State Transfer的缩写，可以翻译为“表现层状态转化”。他最大的几个特点为：资源和接口统一、URI和无状态。

**资源：**

所谓“资源”，就是网络上的一个实体，或者说是网络上的一个具体信息。他可以是一段文本、一收歌曲、一章图片、一种服务，总之就是一个具体的实际存在的事务。资源总要通过某种载体来反映出其中的内容，文本可以用.txt的格式来表现，也可以用.HTML的格式、.XML的格式，甚至可以采用二进制格式；图片可以用.jpg的格式，也可以用.PNG的格式；JSON是现在最常用的资源表示格式。

资源事宜JSON（或者其它Representation）为载体、面向用户的一组数据集，资源对信息的表达倾向于概念模型中的数据即资源总是以某种Representation为载体显示。

**统一接口：**

Restful架构风格有规定，数据的源操作即CRUD（create，raad，update，delete）分别对于HTTP方法：GET用来获取资源，POST一般用于新建资源（也可以用于更新资源），PUT用来更新资源，DELETE一般用于删除资源，这样就统一了数据操作的接口，仅仅通过HTTP方法，就可以完成对数据的所有增删改查等工作。

**URI：**

可以用一个URI（统一资源定位符）指向资源，即是说每个URI都有着一个对应的特定资源。要想获取这个资源，就要访问它的URI，因此URI就成了每一个资源地址的唯一识别码。

**无状态：**

所谓的无状态，即所有的资源都可以通过URI来进行定位，而且这个定位与其他资源的位置无关，也不会因为其他资源发生了改变而改变。有状态和无状态的区别就好比查询一个员工的工资，如果查询工资需要登录系统，进入查询工资页面，执行相关操作后，获取工资的多少数值，这情况就是有转态的，因为查询的每一个步骤都是依赖于前一个步骤，只要前置操作不成功，后续操作就无法执行；如果输入一个url就可以得到指定员工的工资，那这种就是无状态的，因为获取数据不依赖其他任何操作和资源。

### 1.3.7 Element UI

Element UI是一款构建用户界面的UI组件库，可以配合Vue、React、Angular前端框架构建出一个很好的界面环境，它按照以下原则进行设计：

* 1. 一致性原则
* 与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；
* 在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置等。
  1. 反馈原则
* 控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；
* 页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。
  1. 效率原则
* 简化流程：设计简洁直观的操作流程；
* 清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；
* 帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负担。
  1. 可控性原则
* 用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行决策；
* 结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。

按照这些原则，无论是对开发人员还是用户都是极为的友好，开发人员可以很灵活的选择自己的样式，而且其中还有很多已经写好了的组件，我们只需要稍作修改就可以使用，极大的提升了开发的效率。

# 2 系统需求分析

## 2.1 可行性分析

1. **需求可行性：**人口老年龄化趋势在21世纪将愈演愈烈，国内外形势令人担忧。老龄化问题的关键是要解决好养老，而病因复杂、病程漫长的老龄痴呆、脑卒中、帕金森病等神经系统疾病发病率的上升加重了养老的难度。对于这类病程长、发病机制复杂的疾病，若能提早加大预防力度，弘扬中医学“上工治末病”的原创思想，防患于未然，不仅是老年人重视脑健康进而早期预防疾病发生的最佳策略，也是现实国情的需要。不仅仅是老龄人，随着社会的发展与进步，在青少年人群中，其心理出现问题的概率也增加不小，这无疑不给社会增加了不小的负担，但是心理上的问题一般情况都看不出来，表明上看上去和正常人无异，针对这些种种问题设计了一套以心理认知为主要内容的测评系统，虽然不能做到100%治疗效果，却可以尽早的发现，越早发现就越有机会改善，很大程度的减小了爆发的概率，同时也减小了社会压力。
2. **法律可行性：**本系统为自主研发，没有涉及他人成果。
3. **操作可行性**：本系统从实际情况出发，考虑到使用对象从事职业，尽最大可能地简化了操作的复杂性，最大化对用户的友好性。
4. **环保可行性：**在环境保护这一块，该系统不会对周围环境产生污染，所用设备均均为互联网设备。
5. **技术可行性：**为了后期维护方便且易于拓展其它项目，采用前后端分离的方式，后端使用springboot为基础架构编写java代码；mybatis作为与数据库操作的关键工具，他可以做到轻松将数据修改的内容一一对应到服务上，大大减少了因数据改变而需要修改代码的开销；阿里的Druid作为数据库的连接池可以在改变数据库之后能够轻松切换数据库，并且Druid带有多维度的统计和分析功能；redis做为服务器的高性能缓存工具，可以加快前端获取数据的速度，同时也减小了数据承受的压力；Prince工具可以很好的将前端页面转为pdf页面，在参考了诸多html转pdf文档的方式后，在保证文档质量的前提下，这是最佳的工具；前端页面主要以vue框架开发为主，使用vue开发可以用更短时间产出高质量的代码，且编写界面简洁，用户使用过度效果较好。

## 2.2 功能需求

### 2.2.1 题库管理

题库管理使用对象为系统管理员，该管理功能提供了题目信息修改和查看的功能，当一套题目用了很久之后，难免会遇到与现代人的思想略微有点偏差，为了弥补这些，开发了该功能来加以进行修正。为了保证数据的完整性，在进行题目内容修改是需参考修改说明，以防发生不可挽救的损失。

### 2.2.2 量表管理

该功能专为系统管理员定制开发，主要任务为查看数据库中已经存在的量表信息，如有不正确或预设字符有错误时可以进行更正。另外，如需要从数据库中移除某一个量表可以使用该功能进行操作，对应的题目信息也将永久从数据库中删除。该操作为不可逆性操作，需谨慎执行。

### 2.2.3 账号信息管理

账号信息管理功能为后台管理功能，通过该功能可很轻松的实现账号信息的管理。该功能共划分为两个区域，分别为医生账号管理和病人账号管理。

医生账号管理共有密码重置、信息修改和查看两个功能，目的是为了解决内网不支持短信验证，设置密保可能会遇到其它安全隐患。因此统一由管理员进行密码重置。在查看信息的时候如果发现某些信息填写错误可以修改之后并提交，其中某些信息不支持修改，如账号。

病人账号设置了密码重置、信息修改和查看、答题历史记录三部分。密码重置功能可以在病人忘记账号的情况下进行重置找回；信息修改和查看功能可以通过输入对应用户的账号查询到相应的资料信息情况，对那些不正确的记录可以进行修改。答题历史记录只能进行查看每一个量表完成的时间，由谁来完成的，不支持按账号或其它信息进行查询，显示结果为历史所有用户的全部结果，可以按照需要进行排序显示。

### 2.2.4 管理员权限及功能

该系统专为管理员提供了一个单独的平台，负责管理后台的一些事务。主要有医生和病人账号管理，答题记录管理，量表管理和题目管理。这些功能可以很好的帮助管理员进行系统管理。

### 2.2.5 历史记录查看和打印

该功能在医生端进行实现，当医生进入主页之后，可以选择是进入测评还是查看历史报告。查看历史报告功能可以为医生提供很方便的历史记录查询，历史记录默认为显示全部内容，按日期排序，同时医生也可以按需要进行精确查找，共有用户id、用户姓名、答题时间三个选项，其中时间为一个范围，开始时间和结束时间都是按0点计算，即输入2020.01.01到200.01.03表示2020.01.01的0点到2020.01.03的0点，这三个选项可以单选，也可以多选，但是时间必须有开始时间和结束时间，输入的选项越多得到的结果越精确。

为了方便医生在之后的打印需要，在查询处添加了一个打印功能，通过点击查看可以打印出相应的PDF文件，文件格式为A4纸大小，默认是预览状态，由医生自行决定是下载为PDF文件还直接通过打印机打印。

### 2.2.6 结果打印功能

该功能和答题部分直接连接，为了满足医院需求，需要在所有测评完成后自动打印出对应的报告，并且系统需自动回到首页。

### 2.2.7 即时通讯

这一功能主要体现在医生和病人的互动环节，目的是为了减小服务器的压力和传输的高效进行。

在医生端和平板端需要相互给对方传输消息，而且是需要立即显示，所以这一锅称就涉及到了服务主动推送消息的功能，以往的方式为不断的向服务器发送http请求，每次请求都去查看服务器是否有数据需要返回，这种通过轮询的方式十分的浪费资源和消耗内存，为了很好的解决这一问题使用了websocket来建立即时通讯，即时通讯是一种新的协议，类似于http和http有对应的ws和wss协议两种，不同的是通过这种协议可以达到长连接的效果，通过后台服务器可以设置具体的长连接最大时长。通过这样一种方式极大的减轻了服务器的压力，提升了传输效率，尤其是在传输图片文件的时候。

在本系统中，有很多时候都需要用到传输图片，但是图片并不是静态文件，所以没办将同一张图片放在两个端的项目之中，因此需要将图片转化为base64的字符编码，处于系统的质量要求，传输的图片有质量要求，每张图片的base64编码都有几十万的字符长度，这无疑对系统性能造成影响，通过使用websocket即时通讯方式可以很好的解决这一问题，达到两端图片或文字显示一致的结果。

## 2.3 非功能需求

### 2.3.1 系统稳定性

该系统为商业使用的系统，所有在投入使用后肯定会有大量的用户请求，为了达到系统稳定运行的效果，web服务端需要使用Nginx做负载均衡处理，已达到最大的并发量请求而不崩掉，数据服务端需要使用高速缓存而不是将数据直接存储到硬盘上，即便是固态硬盘也不利于资源的读取，数据服务器缓存服务使用redis服务，这样既保证了数据的高速存储，也能保证高并发的正常执行。

### 2.3.2 界面友好性

在界面设计上需要考虑到使用对象是什么类型的人，本次使用的对象为心理认知医生和可能有认知问题的病人，因此在界面设计上功能不能过于复杂，多余的功能和样式应该全部去掉，类似通过鼠标的hover事件来弹出一个菜单功能应该减少使用或尽量不用，在给病人使用的设备中，应该有适当的提示信息，同时医生端也要有当前题目的提示信息以防部分病人为文盲不能阅读的情况。

### 2.3.3 信息安全

系统开发对象为医院机构，所以对于病人的信息需要严格进行保密，需要保证每次测评结果信息的安全性以防外漏。测评结果涉及到了个人隐私问题，为了能更好的解决这一问题，在数据服务采用了UUID和MD5的方式对文件进行了加密处理。

### 2.3.4 可维护性

系统的最大价值不是在于当前系统拿出去有多好使用，而是在于维护是否方便，如果维护起来特别麻烦那最终就可能导致该系统最终被淘汰，因此使用前后端完全分离是最好处理方式，通过这种方式大大降低了后期的维护成本，而且如果遇到前端需要大改而但是服务都不变的情况，完全可以选择抛弃现有前端项目，后端项目保持不变，虽然前期工作量大了一定，但是最后结果却是非常完美的。

### 2.3.5 可扩展性

本系统需要有足够的扩展价值，因为在以后可能会接入第三方服务，如广告、工具插件等，不能因为前期觉得暂时没有这些必要就不做，要知道互联网是快速发展的，现在没有的不代表以后也不会有，所以在系统结构上不能将结构写死，要预留其它业务位置以便以后逻辑业务可以很方便的进行扩展。

# 3 系统设计

## 3.1 设计原则

在设计系统的时候，不能盲目的去凑功能，必须要充分对系统的实用性、稳定性、易用性进行全面的考虑，以满足用户的需求。具体如下：

1. 实用性

当今社会是一个网络社会，所以网络对于普通人来说也是一个普通东西，几乎每个人都可以在任何时候任何地点都会使用到，但是当我们在浏览网页的时候可能会遇到某个网页会弹出很多的广告，或者要经过很过的操作才能看到我们想看到的，这并不是一个用户想要看到的，对于本次设计的系统来说，实用才是根本，只为用户添加他们想要的功能，对于其它无关的内容应该要完全避免，对于他们经常会使用的功能要放在最显眼的地方，并且还进行重点优化提升用户体验，这才是用户最希望看到的。

1. 稳定性

系统运行最重要的一点就是系统的稳点性。系统稳定性是指在任何环境都能正常运行，即便是在并发量大的高负荷状态下，如果不能很好的处理这些问题，那么就会很容的造成服务器瘫痪。服务器瘫痪对程序员来说就是系统崩了，停止工作了，但是用户不会在乎到底是怎么个情况，更不会关系系统是怎么出的问题，他只是知道现在用不了了，页面打不开了，最后得出的结果就是这个产品不是一个合格的产品。所以在进行系统设计的时候必须将系统设计的健壮一点，保证系统对硬件的完美支持，偶尔的逻辑错误或是其它异常也是在所难免的，但是即便中间出了一些错误也想办法不让系统停止运行，或者是卡在那里就不动了，如果真的出现某些重要信息错误这种情况，要及时提示当前用户，告诉用户接下来应该怎么操作，是刷新页面还是退出重登等。

1. 可维护性

系统的可维护性是非常重要的，如果在设计系统的时候不加以考虑，产生的后果将会无法估量。可维护主要体现在开发文档上，所以编写开发文档很重要，完成一个系统需要写的代码肯定是越来越多的，即便是开发者本人时间久了也会忘记自己之前写过的一些内容和逻辑，这个时候如果有开发文档就可以很快定位到相应的功能上。如果没有开发文档，时间久了自己也会忘记，并且如果以后将项目交给其他人去管理，对方根本知道里面都是写的什么，哪一部分分别是干嘛的，逻辑结构怎样完全不清楚，这样造成的后果就是项目不能被维护，出了问题也不能解决，最后被抛弃，这样的结果是很不好的。

1. 可扩展性

设计项目必须考虑项目的扩展能力，不能将结构完全写死，现在的网络发展很快，必须要与时俱进才行，要赋予系统成长的属性的，这样才能生存的更加长久。可扩展不仅可以赋予系统更多的生气，还能提升开发人员的开发效率，当开发的时候使用某些插件的时候，可以通过简单的调用就可以使用了，如果能在进一步进行封装，当我们在使用多个插件的时候可以以同一种方式进行调用却不冲突，这对在以后无论更换插件或增加插件都无疑是最好的结果。

1. 环保性

对于某些需要耗材的部分要尽可能的考虑到环保性，不能对周围的环境造成污染，就像汽车制造行业对尾气的排放有要求一样。比如在本系统中有一个打印报告的功能，为了更加的环保可以为用户提供两种打印方式，保存为pdf文件或选择直接打印在纸上，由用户自己选择是否需要消耗纸张，在进行纸张打印的时候还需要进一步优化，尽可能的压缩内容空间，以节约纸张，最终达到更加环保的效果。

## 3.2 网络拓扑图

## 3.3 功能模块设计及结构分布



图 3.3模块结构分布图

* 1. 登录模块

该模块为系统的入口位置，登录成功之后会将用户信息保存，同时会与服务器保持连接，目的是为了建立即时通讯网络。

* 1. 发布测评模块

该模块实现测评的发布任务，由医生选择将要发布的量表有哪一些，可以选择一个量表或是多个量表进行发布，发布成功之后会将数据保存到服务器并等待患者端登录后进行操作。

* 1. 打印报告模块

该模块实现一个得出结论并打印报告的功能。结论来源主要是从两个方面得出，在每一次发布的所有测验完成之后可以自动跳转报告的打印界面，如果有多个量表就会进行分页显示；还可以在查询记录时候进行报告打印。

* 1. 消息同步模块

该模块主要实现的功能为pc端与平板的数据同步管理功能。由于该系统为两个端相互互动操作，因此存在大量的数据相互更新，如果采用http轮询的方式将会占用大量的服务资源，通过该模块可以很轻松的实现这一功能，且不会占用太多资源，对系统的有着良好的提升效果。

* 1. 记录查询模块

查询模块提供所有信息的查询功能，包括量表信息，题目信息，用户信息，历史记录信息等的查询。设计该模块的目的是为了方便系统的管理，同时为用户创建一个良好的体验环境。该模块的注入减少了医生对用户频繁询问等操作，需要的信息可以从记录之中全部查看到，避免了某些信息的遗漏，如果发现某些信息与当前情况不相符合也可以立即作出修改，大大简化了用户们的操作，且操作简单，很适合操作性不强的用户使用。

# 4 数据库设计

## 4.1 E-R图

E-R图是实体和联系的一种具体的展现图，特提供了表示实体类型、属性和联系的方法，通过这些方式来描述现实世界的概念模型，同时也是开始设计数据库的第一步，只有完成了这一步才能继续设计数据库，若这一步没设计好，那么将会面临大量修改数据库的麻烦。本系统的E-R如下图所示。



图 4.1.1 系统E-R图



图 4.1.2 系统E-R图

## 4.2 表设计

数据库表的设计在数据库的操作中有着十分重要的作用，一个好的表设计可以很好的保证数据库内容的完整性。在建立数据库表的时候肯定有一些表的结构是复杂得，这个时候就需要对其进行优化，增强系统的性能。本系统的数据库表结构如下所示。

（1）管理员表

表4.2.1 管理员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_admin\_id | 11 | 否 | 主键 | 管理员id号 |
| ai\_admin\_pwd | 16 | 是 |  | 管理员登录密码 |
| ai\_admin\_unit | 40 | 是 |  | 管理员所属单位 |
| ai\_admin\_department | 40 | 否 | 主键 | 管理员所在部门 |
| ai\_admin\_name | 12 | 是 |  | 管理员名字 |
| ai\_admin\_limit | 11 | 是 |  | 管理远权限等级 |
| ai\_user\_type | 12 | 是 |  | 账号类型 |
| ai\_id\_card | 20 | 是 |  | 省份证 |

（2）医生表

表4.2.2 医生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_doc\_id | 11 | 否 | 主键 | 医生账号id |
| ai\_doc\_pwd | 16 | 是 |  | 医生账号密码 |
| ai\_doc\_age | 11 | 是 |  | 医生年龄 |
| ai\_doc\_sex | 2 | 是 |  | 医生性别 |
| ai\_doc\_name | 12 | 是 |  | 医生名字 |
| ai\_doc\_unit | 40 | 是 |  | 医生所属单位 |
| ai\_doc\_department | 40 | 是 |  | 医生所属部门 |
| ai\_doc\_phone | 12 | 是 |  | 电话号码（座机） |
| ai\_doc\_superior\_id | 11 | 是 |  | 直属领导id号 |
| ai\_doc\_photo | 60 | 是 |  | 医生证件照 |
| ai\_user\_type | 12 | 是 |  | 账号类型 |
| ai\_id\_card | 20fou | 是 |  | 省份证号 |

（3）病人表

表 4.2.3 病人表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_user\_id | 11 | 否 | 主键 | 病人账号id |
| ai\_user\_pwd | 16 | 否 |  | 病人账号密码 |
| ai\_user\_type | 16 | 否 |  | 账号类型 |

（4）病人信息表

表 4.2.4 病人信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_user\_id | 40 | 否 | 主键/外建 | 病人id号 |
| ai\_user\_patient\_number | 40 | 是 |  | 病人门诊号 |
| ai\_user\_department | 40 | 是 |  | 所在部门 |
| ai\_doc\_name | 40 | 是 |  | 主治医生名字 |
| ai\_user\_id\_num | 40 | 是 |  | 病人省份证号码 |
| ai\_user\_office | 40 | 是 |  | 所属科室 |
| ai\_user\_from | 40 | 是 |  | 来源 |
| ai\_user\_qq | 40 | 是 |  | qq号码 |
| ai\_user\_email | 40 | 是 |  | 电子邮箱 |
| ai\_user\_home | 40 | 是 |  | 家庭住址 |
| ai\_user\_address | 40 | 是 |  | 目前居住地 |
| ai\_user\_emergency\_con\_phone | 40 | 是 |  | 紧急联系电话 |
| ai\_user\_age | 40 | 是 |  | 病人年龄 |
| ai\_user\_sex | 40 | 是 |  | 病人性别 |
| ai\_user\_name | 40 | 是 |  | 病人名字 |
| ai\_user\_nation | 40 | 是 |  | 民族 |
| ai\_user\_native\_place | 40 | 是 |  | 生源地 |
| ai\_user\_edu\_bg | 40 | 是 |  | 教育背景 |
| ai\_user\_politics\_status | 40 | 是 |  | 政治面貌 |
| ai\_user\_health\_con | 40 | 是 |  | 健康状态 |
| ai\_user\_height | 40 | 是 |  | 体重 |
| ai\_user\_birth | 40 | 是 |  | 出生日期 |
| ai\_user\_unit | 40 | 是 |  | 单位 |
| ai\_user\_marriage | 40 | 是 |  | 婚姻 |
| ai\_user\_profession | 40 | 是 |  | 职业 |
| ai\_user\_duty | 40 | 是 |  | 职位 |
| ai\_user\_children | 40 | 是 |  | 子女 |
| ai\_user\_bloodType | 40 | 是 |  | 血型 |

（5）测试结果表

表 4.2.5测试结果表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_user\_id | 20 | 否 | 主键/外建 | 病人id号 |
| ai\_scale\_id | 20 | 否 | 主键/外建 | 量表id号 |
| ai\_test\_name | 100 | 否 |  | 测试名称 |

（6）历史记录表

表 4.2.6历史记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_user\_id | 20 | 否 | 主键 | 病人id号 |
| ai\_scale\_id | 20 | 否 |  | 量表id号 |
| ai\_test\_time |  | 否 | 主键/外建 | 内容发布时间 |
| ai\_scale\_name | 100 | 否 | 主键 | 量表名字 |
| ai\_doc\_id | 100 | 否 |  | 发布医生账号 |
| ai\_doc\_name | 100 | 否 |  | 发布医生名字 |
| report\_content |  | 否 |  | 报告主要内容 |
| ai\_user\_name | 255 | 否 |  | 病人名字 |
| msg | 1000 | 否 |  | 报告其它内容 |
| start\_time |  | 否 |  | 保存时间 |

（7）量表信息表

表 4.2.7量表基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_scale\_id | 11 | 否 | 主键/外建 | 量表id号 |
| ai\_scale\_name | 40 | 是 |  | 量表名称 |
| ai\_scale\_overview | 5000 | 是 |  | 量表说明 |
| ai\_scale\_notice | 5000 | 是 |  | 注意事项 |

（8）题目内容表

表 4.2.8题目内容表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 长度 | 是否为空 | 备注 | 说明 |
| ai\_scale\_id | 11 | 否 | 主键/外建 | 量表id号 |
| ai\_question\_id | 11 | 否 | 主键 | 题目编号 |
| ai\_question\_content | 8000 | 是 |  | 题目内容 |
| ai\_question\_option | 1000 | 是 |  | 题目选项 |
| ai\_question\_answer | 500 | 是 |  | 题目答案 |
| ai\_question\_img |  | 是 |  | 题目图片 |
| ai\_question\_type | 100 | 是 |  | 题目类型 |
| ai\_question\_score | 100 | 是 |  | 问题得分 |
| ai\_question\_notice | 1000 | 是 |  | 注意事项 |
| ai\_question\_guid | 1000 | 是 |  | 指导语 |
| ai\_question\_answer\_describe | 1000 | 是 |  | 答案描述 |
| ai\_score\_method\_name | 40 | 否 | 主键 | 评分方式名称 |
| ai\_score\_method | 2000 | 是 |  | 评分方式 |
| ai\_scale\_name | 255 | 是 |  | 量表名称 |

# 5 UI设计

## 5.1 目的

为了使得整个系统的结构更加的清晰和直观，在系统层次设计好之后开始进行UI设计，通过这种方式可以很好的提升程序开发的效率，并且最后系统的体验效果也会很好。

通过UI设计，我们可以提前了解到系统的整体布局，对相应需求功能的位置以及如何实现都会有一个大致的了解，当我们在编写代码的时候，通过先前对系统的了解，可以提前设置好全局变量和局部变量，避免了同一个值会重复在多个变量里面，造成内存的浪费和资源的消耗，同时编写的界面不是很整洁，极不利于后期的维护工作。但是当我们有了一个统一的处理之后，修改起来就方便了很多，同时代码也不会显得十分的冗余复杂。

## 5.2 工具

# 6 系统实现与测试

## 6.1 开发环境搭建

## 6.2 功能测试

## 6.3 性能测试

# 总结

# 参考文献

# 附录