软件系统设计说明书

**Software System Design Specification**

**编号**：**TMP-SSDS**

**版本 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| 作者： | 黄家意 | 日期： | 2024/5/14 |
| 审批： |  | 日期： |  |

**变更记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 变更说明 | 作者 |
| 2024/5/14 | 1.0 | 创建 | 黄家意 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**本阶段工作及具体分工**

|  |  |
| --- | --- |
| 工作任务 | 主要负责人 |
| 编辑设计书大概框架以及其他一些基本信息 | 黄家意 |
| 开发与运行环境 | 齐润东 |
| 系统业务流程图 | 齐润东 |
| 功能模块划分部分 | 陈凯域 |
| 数据库设计概述 | 黄家意 |
| 用户界面设计（原型系统设计） | 陈宇兆 |
| 大概类的构想 | 陈宇兆 |

# **1.引言**

## **1.1编写目的**

编写本文档主要用于说明关于问卷系统的具体系统架构设计。其中，具体说明了系统架构设计的目标和约束需求，规定了系统的开发环境和运行环境，阐述了逻辑架构设计的整体思想和具体设计的实现，介绍了系统的主要业务流程和系统的功能模块的划分以及模块之间的联系，表明了系统采用存储的数据库类型以及数据库又采用了什么设计编辑工具和数据库中存储的各种表的具体信息，最后是设计了系统核心功能的界面原型。

本文档是对系统架构设计的详细说明，通过编写该系统架构设计说明书，有利于了更好地理解系统逻辑架构和系统相关功能，也便于系统管理员更好地管理系统以及在出现问题后能够更好地对系统进行维护。

## **1.2 读者对象**

系统管理者：在系统正式运行后，系统的管理者，主要用来维护系统的正常运行，保证系统数据库信息的安全，能够及时应对系统在运行时出现的一些小问题。

系统开发人员：整个系统的主要开发人员，也是系统架构和功能的主要设计人员，负责系统的web前端和后端的开发。

系统测试人员：在系统完成了初步实现后，对系统进行各方面的测试，其中包括系统的功能是否全面，系统的响应和事件的处理是否快捷，系统的安全性以及系统整体的业务流程等方面。

# **2.系统概述**

问卷系统是一款能够设计问卷、发布问卷、收集问卷以及对问卷收集信息分析的多功能应用型系统。用户登陆系统，可以根据自己的需求设计问卷的主题性质和问卷的题目，题目可定义为单选题、多选题、判断题以及主观意见题；在完成问卷的设计后，可发布问卷收集数据，收集后可在系统查看相关数据

该问卷系统主要满足的功能需求如下(详细见需求分析文档)：

（1）用户注册与登录：新用户能够注册独立用户，旧用户能登入账户

（2）问卷生成：在问卷编辑界面可以生成几个类别的问题，最后生成属于用户的问卷。

（3）问卷发布：某个用户的问卷在发布界面能生成分享链接从而能异地填写

（4）数据收集分析：系统能把被填写的问卷的数据汇总，展示。

（5）个人中心：展示该用户编辑、保存、发布的问卷

# **3. 架构设计目标和约束**

## **3.1架构设计目标**

架构设计的主要目的是为了解决软件系统复杂度带来的问题。

架构设计是宏观性考量，在整体上理解问题的复杂性，给出方案，并论证方案的可行性，提供一系列准则指导执行。

因此，架构设计的目标主要有以下几点：

（1）提前识别问题的复杂性和关注点，提供可行的经过论证的解决方案；

（2）建立服务质量指标，确定设计方案可以满足指定的质量指标；

（3）规划整体设计，提供长远的可扩展性；

（4）为实际开发确定基本框架。

## **3.2 约束需求**

### **3.2.1 软件、硬件环境（包括运行环境和开发环境）的约束**

软件开发和运行是基于64位Windows操作系统，所以软件可在64位Windows操作系统上的网页浏览器中正常打开和运行。采用vue框架完成Web前端的开发，node.js+express框架完成后端开发，通过mongoose对MongoDB进行数据库的设计和编辑，最终在Windows 10及以上操作系统上的Google Chrome浏览器完成系统的测试和最终运行。

### **3.2.2 接口/协议的约束**

（1）内部接口/协议的约束

在内部接口方面，各模块之间采用函数调用、参数传递、返回值等方式进行信息传递。接口传递的信息将是以数据结构封装的数据，以参数传递或返回值的形式在各模块间传输。

（2）外部接口/协议的约束

在输入方面，对于键盘、鼠标等外部设备的输入，通过web的内置模块自动对输入进行处理。

在输出方面，如数据的显示也由web浏览器通过前端编码，后端获取数据展示给用户

### **3.2.3 用户界面的约束**

UI的设计开发基于html5、css3和TypeScript等前端开发语言。在UI的设计中要具备适用性，易于用户理解；当错误操作时会有提示信息和防错处理，整体的布局和色彩的适用要合理，在事件的响应方面要最高效率，整体界面可复用，控件设置满足单一应对原则。网络通信协议采用 HTTP和TCP/IP协议，调用接口采用 Web Service，系统通过内嵌浏览器技术与扩展对象插件接口集成。

### **3.2.4 软件质量的约束**

（1）正确性

UI上的控件能够正确且及时响应，实现该控件对应功能，错误操作会进行信息提示，同时添加防错处理，及时应对出现的错误操作。

（2）可靠性

设置备用服务器保证系统正常工作。当遇到断电重启的情况，能够实现系统断电后的自动重启，且较少人工维护量和实现系统稳定运行。

（3）效率

数据信息的采集通讯支持10秒以上的实时数据传输（受网络环境影响），并且要求系统的响应时间不超过5秒。

（4）安全性

用户注册/登录系统，关键数据操作，需要验证码进行验证，关于数据采用MD5加密技术。

（5）可扩展性

系统在整体设计上要充分考虑可扩展性，当用户的需求变化时，系统可进行相应的处理；当系统在运行中，客户如果对原有功能作调整或变更，系统应该能够灵活地修改。同时，预留接口，能扩展到其他监测系统。

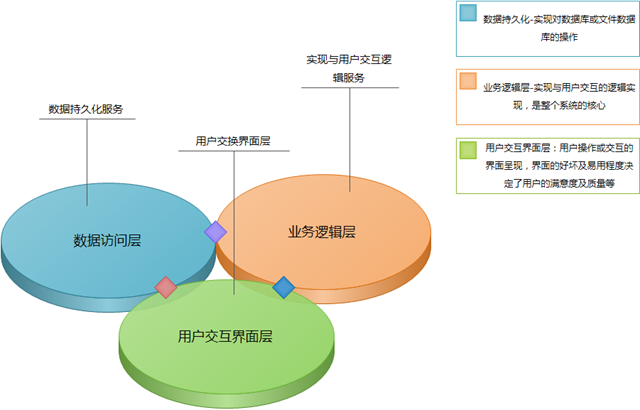
## **3.3 开发与运行环境**

|  |  |  |
| --- | --- | --- |
| **类别** | **标准配置** | **最低配置** |
| **开发环境** | Windows 11操作系统 | Windows 10操作系统 |
| Visual Studio Code | Visual Studio Code |
| MongoDB 5.0.26 | MongoDB 5.0.26 |
| **测试环境** | Windows 11操作系统 | Windows 10操作系统 |
| Google Chrome（64位） | Google Chrome（32位） |
| **运行环境** | Windows 11操作系统 | Windows 10操作系统 |
| Google Chrome（64位） | Google Chrome（32位） |

# **4. 系统总体设计**

## **4.1 设计思想**

三层架构(3-tier architecture) 通常意义上的三层架构就是将整个业务应用划分为：界面层（User Interface layer）、业务逻辑层（Business Logic Layer）、数据访问层（Data access layer）。区分层次的目的即为了“高内聚低耦合”的思想。3个层次中，系统主要功能和业务逻辑都在业务逻辑层进行处理。



**三层架构图**

所谓三层体系结构，是在客户端与数据库之间加入了一个“中间层”，也叫组件层。这里所说的三层体系，不是指物理上的三层，不是简单地放置三台机器就是三层体系结构，也不仅仅有B/S应用才是三层体系结构，三层是指逻辑上的三层，即使这三个层放置到一台机器上。三层体系的应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。通常情况下，客户端不直接与数据库进行交互，而是通过COM/DCOM通讯与中间层建立连接，再经由中间层与数据库进行交互。

## **4.2 软件逻辑架构设计**

分层的原因：

（1）分层设计允许你分割功能进入不同区域。换句话说，层在设计中就是逻辑组件的分组。例如，A层可以访问B层，但B层不能访问A 层。

（2）用分层的方法，以提高应用程序的可维护性，并使其更容易扩展，以提高性能。

设计分层的原则

（1）层意味着组建的逻辑分组。例如，对用户界面，业务逻辑和数据访问组建应该使用不同的不同的层。

（2）在一个层内组建应该聚合的。如业务层组建仅应提供与业务逻辑相关的操作，而不是提供其他操作。

（3）在设计的每一个层接口时要考虑好物理边界。如果通信扩展了物理边界，使用基于消息操作；否则使用基于对象操作。

（4）考虑使用接口类型（interface）来定义每层的接口。这将允许你创建该接口的不同实现，提高可测性。

（5）对于Web应用程序，在表示层和业务逻辑层之间实现基于消息的接口是一个好主意，即使这两层没有跨越物理边界。基于消息的接口更适合于无状态的Web操作。

·数据访问层：

有时候也称为是持久层，其功能主要是负责数据库的访问，可以访问数据库系统、二进制文件、文本文档或是XML文档。简单的说法就是实现对数据表的Select，Insert，Update，Delete的操作。主要是对原始数据（数据库或者文本文件等存放数据的形式）的操作层，而不是指原始数据，也就是说，是对数据的操作，而不是数据库,具体为业务逻辑层或表示层提供数据服务。

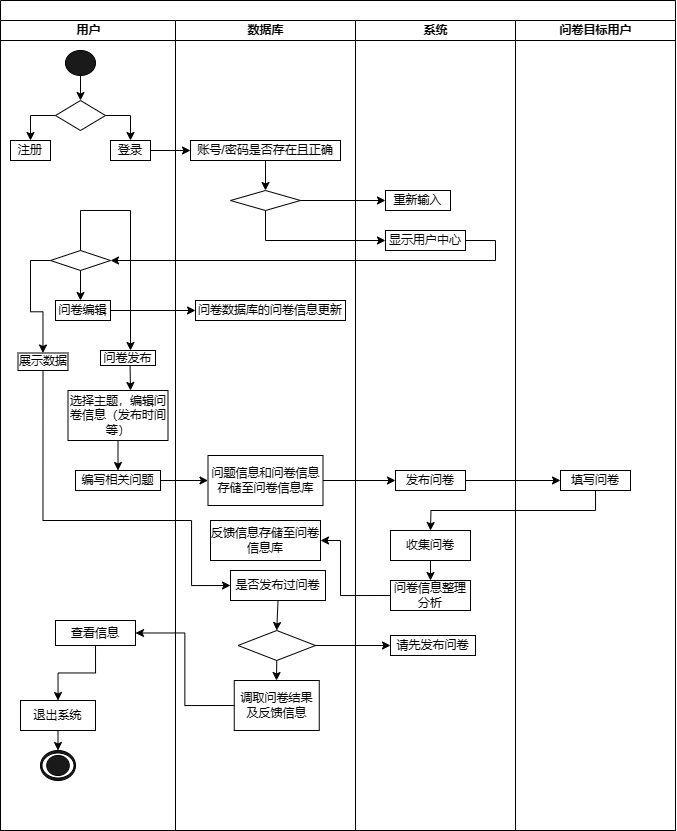
·业务逻辑层：

主要是针对具体的问题的操作，也可以理解成对数据层的操作,对数据业务逻辑处理，如果说数据层是积木，那逻辑层就是对这些积木的搭建。业务逻辑层（Business Logic Layer）无疑是系统架构中体现核心价值的部分。它的关注点主要集中在业务规则的制定、业务流程的实现等与业务需求有关的系统设计，也即是说它是与系统所应对的领域（Domain）逻辑有关，很多时候，也将业务逻辑层称为领域层。业务逻辑层在体系架构中的位置很关键，它处于数据访问层与表示层中间，起到了数据交换中承上启下的作用。由于层是一种弱耦合结构，层与层之间的依赖是向下的，底层对于上层而言是“无知”的，改变上层的设计对于其调用的底层而言没有任何影响。如果在分层设计时，遵循了面向接口设计的思想，那么这种向下的依赖也应该是一种弱依赖关系。因而在不改变接口定义的前提下，理想的分层式架构，应该是一个支持可抽取、可替换的“抽屉”式架构。正因为如此，业务逻辑层的设计对于一个支持可扩展的架构尤为关键，因为它扮演了两个不同的角色。对于数据访问层而言，它是调用者；对于表示层而言，它却是被调用者。依赖与被依赖的关系都纠结在业务逻辑层上，如何实现依赖关系的解耦，则是除了实现业务逻辑之外留给设计师的任务。

·表示层：

位于最外层（最上层），离用户最近。用于显示数据和接收用户输入的数据，为用户提供一种交互式操作的界面。主要表示为WEB方式，如果逻辑层相当强大和完善,无论表现层如何定义和更改,逻辑层都能完善地提供服务。

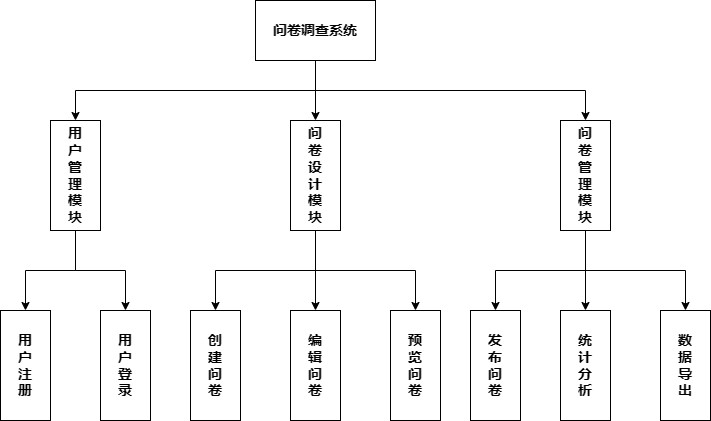
## **4.3 系统主要业务流程设计**



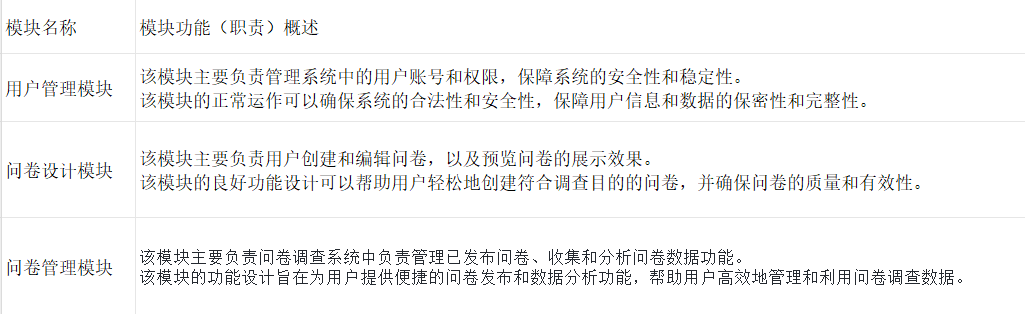
**业务流程图**

## **4.4 功能模块划分**

功能模块图



### **4.4.1.模块—描述**



### **4.4.2.模块—接口设计**

**用户管理模块接口设计：**

**用户注册接口：**

输入：用户名、密码、手机号等个人信息。

输出：注册成功/失败信息。

**用户登录接口：**

输入：用户名、密码。

输出：登录成功/失败信息，用户权限等。

**问卷设计模块接口设计：**

创建问卷接口：

输入：问卷标题、问题内容、问题类型等。

输出：问卷创建成功/失败信息。

**编辑问卷接口：**

输入：问卷ID、修改内容。

输出：问卷编辑成功/失败信息。

**预览问卷接口：**

输入：问卷ID。

输出：问卷预览页面。

**问卷管理模块接口设计：**

**发布问卷接口：**

输入：问卷ID、发布时间、发布方式。

输出：问卷发布成功/失败信息。

**统计分析接口：**

输入：问卷ID、统计参数。

输出：数据统计、分析结果。

**数据导出接口：**

输入：问卷ID、导出格式。

输出：导出的数据文件。

# **5. 数据库设计概述**

## **5.1 数据库环境说明**

MongoDB是一个基于分布式文件存储的数据库。由[C++](https://baike.baidu.com/item/C++/99272?fromModule=lemma_inlink)语言编写。旨在为WEB应用提供可扩展的高性能数据[存储解决方案](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E8%A7%A3%E5%86%B3%E6%96%B9%E6%A1%88/10864850?fromModule=lemma_inlink)。MongoDB是一个介于[关系数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93/1237340?fromModule=lemma_inlink)和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的[数据结构](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84/1450?fromModule=lemma_inlink)非常松散，是类似[json](https://baike.baidu.com/item/json/0?fromModule=lemma_inlink)的[bson](https://baike.baidu.com/item/bson/0?fromModule=lemma_inlink)格式，因此可以存储比较复杂的[数据类型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B/10997964?fromModule=lemma_inlink)。Mongo最大的特点是它支持的[查询语言](https://baike.baidu.com/item/%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/2661811?fromModule=lemma_inlink)非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

它的特点是高性能、易部署、易使用，存储数据非常方便。

主要功能特性有：

\*面向集合存储，易存储对象类型的数据。

\*模式自由。

\*支持动态查询。

\*支持完全索引，包含内部对象。

\*支持查询。

\*支持复制和故障恢复。

\*使用高效的二进制数据存储，包括大型对象（如视频等）。

\*自动处理碎片，以支持云计算层次的扩展性。

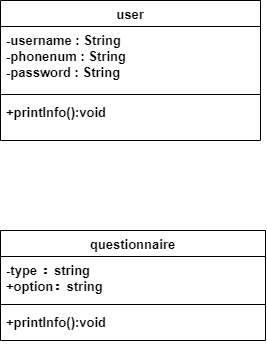
\*支持 Golang，RUBY，PYTHON，JAVA，C++，PHP，C#,JS等多种语言。

\*文件存储格式为BSON（一种JSON的扩展）。

\*可通过网络访问。

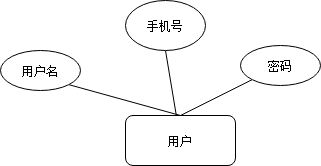
## **5.2系统概念架构设计**

基于用例模型创建系统概念模型，创建数据类图

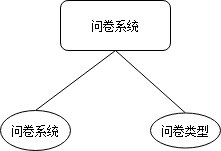


## **5.3数据库逻辑架构设计**

根据系统概念模型，以及根据需求分析和系统逻辑架构，进行数据库逻辑架构设计



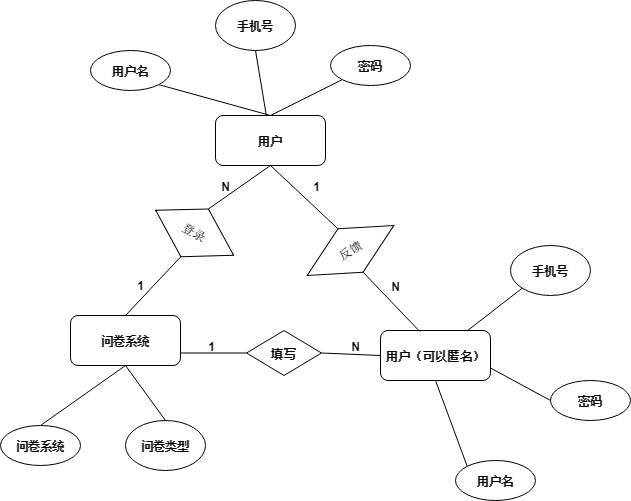
**用户E-R图**

****

**问卷系统E-R图**

用户可以进行问卷设计和发布，并能够得到问卷反馈的信息。

用户也可以是问卷的填写人员，在规定时间内完成问卷的填写并提交问卷。问卷系统是数据库的一个模型，主要区分问卷类型，问卷的发布者登录系统从而发布具体类型的问卷，而发布的问卷由用户填写完毕后反馈给问卷发布者，整体关系由E-R图展示如下。



**概念模型关系图**

# **6. 用户界面设计**

见原型系统

# **7.大概类的构想**

