文本复制检测报告单(全文标明引文)

№:BC20200508140535117399295 3 检测时间:2020-05-08 14:05:35

检测文献: 三稿(定稿)-蒋芳

作者: 蒋芳

检测范围: 中国学术期刊网络出版总库

中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库

中国重要会议论文全文数据库

中国重要报纸全文数据库

中国专利全文数据库

图书资源

优先出版文献库

大学生论文联合比对库

互联网资源(包含贴吧等论坛资源)

英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)

港澳台学术文献库

互联网文档资源

CNKI大成编客-原创作品库

时间范围: 1900-01-01至2020-05-08

检测结果

总文字复制比：10.5% 跨语言检测结果：0%

去除引用文献复制比：10.1% 去除本人已发表文献复制比：10.5%

单篇最大文字复制比：3.5%（6676977_谢成昱_基于vue.js的在线答题系统）

重复字数： [1641] 总段落数： [2]

总字数： [15592] 疑似段落数： [2]

单篇最大重复字数： [549] 前部重合字数： [60]

疑似段落最大重合字数：[1520] 后部重合字数： [1581]

疑似段落最小重合字数：[121]

指 标：☐疑似剽窃观点 ☑疑似剽窃文字表述 ☐疑似自我剽窃 ☐一稿多投 ☐疑似整体剽窃 ☐过度引用 ☐重复发表

表格：0 公式：- 疑似文字的图片：0 脚注与尾注：0

3.3%（121） 三稿(定稿)-蒋芳.doc_第1部分（总3653字）

12.7%（1520） 三稿(定稿)-蒋芳.doc_第2部分（总11939字）

（注释：无问题部分 文字复制比部分 引用部分）

1. 三稿(定稿)-蒋芳.doc_第1部分

总字数：3653

相似文献列表 文字复制比：3.3%(121) 疑似剽窃观点：(0)

1	社交网络用户知识共享研究:特征、内容与展望 耿瑞利;申静; -《图书情报知识》- 2018-02-27 1	1.6%（60） 是否引证：是
2	福建省石化行业固体废物管理问题及对策建议 张鸿斌; -《资源节约与环保》- 2019-12-25	0.8%（30） 是否引证：否
3	习近平总书记理想信念教育重要论述研究 姬亚男(导师：荆世群) -《湘潭大学硕士论文》- 2019-06-09	0.8%（29） 是否引证：否

原文内容

湖南人文科技学院
本科生毕业设计说明书

项目名称：	大学生知识分享问答平台		
学院名称：	信息学院		
学生姓名：	蒋芳	学号	16436213
专业年级：	软件工程2016级		
指导教师：	付又香/陈洁		
教师职称：	高级实验师/副教授		

项目名称： 大学生知识分享问答平台
学院名称： 信息学院
学生姓名： 蒋芳学号 16436213
专业年级： 软件工程2016级
指导教师： 付又香/陈洁
教师职称： 高级实验师/副教授
湖南人文科技学院教务处
二〇二〇年制
毕业设计原创性及知识产权声明
本人郑重声明：所呈交毕业设计是本人在导师指导下独立进行研究所取得的成果，对本设计的研究做出重要贡献的个人和集体，均已在文中明确标明。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。
本毕业设计成果归湖南人文科技学院所有。
作者签名： 201 年月日
导师签名： 201 年月日
目录
摘要：.....I
Abstract：.....II
1 绪论.....1
1.1 背景和意义.....1
1.2 国内外研究现状.....1
1.3 本内容安排.....2
2 系统需求分析及总体设计.....3
2.1 功能需求.....3
2.1.1 业务逻辑：.....3
2.1.2 系统功能.....3
2.1.3 系统流程图.....3
2.2 操作和开发选择.....4
2.2.1 页面操作.....4
2.2.2 开发选择.....4
3 系统详细设计及实现.....7
3.1 数据库设计.....7
3.1.1 E-R图.....7
3.1.2 数据库需求分析.....8
3.2 前端功能实现.....8
3.2.1 前端分析.....8
3.2.2 页面路由.....9
3.2.3 请求跨域.....11
3.2.4 富文本.....12
3.2.5 前端输入校验.....13
3.3 后台功能实现.....15
3.3.1 后台分析.....15
3.3.2 用户注册.....15
3.3.3 用户登录.....17
3.3.4 退出登录.....18
3.3.5 操作文章.....19

3.3.6 操作提问.....28
4 系统测试.....33
4.1 测试目的.....33
4.2 测试方法.....33
4.3 测试用例.....33
4.3.1 用户测试.....33
4.3.2 文章测试.....34
4.3.3 提问测试.....37
4.3.4 权限测试.....39
总结.....41
参考文献.....42
致谢.....44

大学生知识分享问答平台

摘要：

知识共享不仅是知识积累的需求，也是知识管理的重要部分。在知识共享的过程中，在将自己所拥有的知识积累起来的同时必然会发现自己知识面的缺陷，并完善自己所欠缺的地方。知识分享问答平台给是用户提供一个根据自身需求提出问题，并寻求解答的平台。其他用户提供该问题的答案。随着知识分享范围的扩大，用户在分享的同时，知识面也在不断的增加。近年来，随着科学与信息计算的进步，以及人们对信息资源需求的增加，在这个互联网时代，通过网络平台进行学习已经成为一种常态。

本文对大学生知识分享平台进行整体分析，明确了系统的功能需求，规划了系统功能模块。本设计前端部分采用Vue搭建项目，实现数据的双向绑定。后台选用Node.js 和MongoDB来设计一个大学生知识分享问答平台。给大学生提供一个知识交流学习，问题讨论，技术交流、拓宽知识面、知识共享的社交类平台。在平台上，用户可以对不懂的问题进行提问，等待其他用户的回答，或者在平台上搜索相关的知识进行学习了解；当然，用户也可以对他人提出的问题进行答复；除此之外，用户还可以将平时积累的知识发表到平台上，供他人参考学习，以此来实现用户之间的知识分享。

关键词：知识分享，Vue.js，Node.js，MongoDB、数据双向绑定

A question-and-answer platform for college students to share their knowledge

Abstract：

Knowledge sharing is not only the requirement of knowledge accumulation, but also an important part of knowledge management. In the process of knowledge sharing, when we accumulate our knowledge, we will inevitably find the defects of our knowledge and improve what we lack. Knowledge sharing q&a platform is a platform for users to ask questions according to their own needs and seek answers. Other users provide answers to this question. With the expansion of the scope of knowledge sharing, users are sharing at the same time, the scope of knowledge is also increasing. In recent years, with the progress of science and information computing, as well as the increase of people's demand for information resources, learning through network platform has become a norm in this Internet era.

This paper makes an overall analysis of the knowledge sharing platform for college students, defines the functional requirements of the system, and plans the functional modules of the system. The front-end part of the design adopts Vue to build the project and realize the two-way data binding. In the background, node. js and MongoDB are selected to design a question-and-answer platform for college students to share knowledge. It provides a social platform for college students to exchange knowledge, study, discuss problems, exchange technologies, broaden the scope of knowledge and share knowledge. On the platform, users can ask questions that they don't understand, wait for answers from other users, or search for relevant knowledge on the platform for learning. Of course, users can also respond to questions from others; In addition, users can also publish their accumulated knowledge on the platform for others to learn for reference, so as to realize knowledge sharing among users.

Key Words: Knowledge sharing, vue. js, node. js, MongoDB、Data bidirectional binding

1 绪论

1.1 背景和意义

在"互联网+"的知识经济时代，知识已成为获得竞争优势和重要资源的重要要素。知识分享平台它以其独有的个性、方便而广泛应用。许多的知识分享类平台开始蓬勃发展。无论是聊天工具，还是知乎、B站等，甚至于企业公告等都或多或少地将其运用成为知识分享交流的重要手段。知识共享平台为用户提供了一个良好的交流与分享的平台，并且吸引着志趣相投的用户参与进来，并在平台中建立联系，进行相互的交流学习。在平台中，用户成为知识共享的受益者和知识构建的促进者。

随着科学技术的飞速发展，人们对知识和信息资源的需求大幅上升，互联网已经成为快速获取，发布和传递信息的重要渠道。知识分享平台的出现，提供了一个知识交流学习，问题讨论，技术交流、拓宽知识面的社交类平台。在平台中用户可以将自己平时积累的知识分享到平台上，在这个对知识整理、记录的过程中，不但加深了自己对知识的理解与运用，也可能对某个问题产生进的想法和思考。在平台中用户也可以针对自身的需要通过搜索问题查找与答案相关的文章解决疑难，亦可以向他人寻求解答，与其他用户进行交流探讨。有利促进学习、记录学习收获、交流学习心得看法，知识共享。

1.2 国内外研究现状

近年来，随着信息技术与经济的快速发展，以及人们对信息资源需求的增加，在这个互联网时代，人们可以通过网络平台快速的获取到自己所需的知识，也可以快速的分享和传播自身所获取的知识，通过网络平台进行学习已经成为一种常态。

在我国，网络社区平台蓬勃发展，鼓励用户在会联网上进行互动，加强用户之间的交流与沟通，促进用户之间的知识交换；肯定用户在知识共享上的劳动成果，使知识共享成为互联网发展的源泉动力[14]。在如今，通过网络进行交流和知识共享已逐渐成为人们共享知识的重要渠道；国内外2000年至2017年有关社交网络用户知识共享研究的478篇文献进行全面深入分析

；对该领域的"日常知识共享"实证研究的186篇文献进行内容分析,构建了实证研究、理论基础编码表,并对知识共享行为的"影响因素"进行抽取和归纳分析；内容分析结果表明,当前社交网络用户知识共享研究呈现文献数量增长迅速、实证研究取向突出、研究主题较为集中、*研究框架初显成型的特征;研究内容主要包括日常知识共享的行为特性、*单个社交平台的知识共享、*基于社会资本理论探讨、知识共享行为的前置影响因素等;未来将*呈现研究热度继续增加、研究视角学科交叉性、研究方法多元化、理论创新不断增强的发展趋势[13]。。

1.3 本内容安排

第一章主要介绍了大学生知识分享问答平台的背景、意义以及国内外关于该系统的研究状态等信息，对文章的安排进行了介绍；

第二章对系统的需求和功能进行了分析，同时介绍开发工具的选择；

2. 三稿(定稿)-蒋芳.doc_第2部分

总字数：11939

相似文献列表　文字复制比：12.7%(1520)　疑似剽窃观点：(0)

1	6676977_谢成昱_基于vue.js的在线答题系统 谢成昱 - 《大学生论文联合比对库》- 2019-04-18	4.6%（549） 是否引证：否
2	网络工程1501-吴文文-151608020109-基于html5的移动端网站的设计与实现 吴文文 - 《大学生论文联合比对库》- 2019-03-08	3.6%（431） 是否引证：否
3	基于APICloud的宠物信息管理系统的设计与实现 张根 - 《大学生论文联合比对库》- 2019-04-18	3.6%（427） 是否引证：否
4	2014876782_曹可鑫_三相逆变器的设计（通信及监控部分） 曹可鑫 - 《大学生论文联合比对库》- 2018-06-18	3.6%（424） 是否引证：否
5	P20_史淳焘_上海证券交易所技术运营监测平台的设计与实现 史淳焘 - 《大学生论文联合比对库》- 2018-03-10	3.5%（420） 是否引证：否
6	“吃货大搜罗”系统的设计与实现 王庆冉 - 《大学生论文联合比对库》- 2018-05-19	3.5%（416） 是否引证：否
7	“吃货大搜罗”系统的设计与实现 王庆冉 - 《大学生论文联合比对库》- 2018-05-31	3.5%（416） 是否引证：否
8	“吃货大搜罗”系统的设计与实现 王庆冉 - 《大学生论文联合比对库》- 2018-06-04	3.5%（416） 是否引证：否
9	校园快递服务系统的分析与设计 翟露 - 《大学生论文联合比对库》- 2018-05-19	3.5%（415） 是否引证：否
10	网络工程1502-孙璐鹏-151608020235-移动电商系统 孙璐鹏 - 《大学生论文联合比对库》- 2019-03-08	3.5%（414） 是否引证：否
11	网络工程1502-孙璐鹏-151608020235-移动电商系统 孙璐鹏 - 《大学生论文联合比对库》- 2019-03-22	3.5%（414） 是否引证：否
12	基于VUE Nodejs的网上书店系统的设计与实现 邹旻磊 - 《大学生论文联合比对库》- 2018-05-08	3.4%（410） 是否引证：否
13	20146067_樊子微_小区物业管理系统的开发与实现 樊子微 - 《大学生论文联合比对库》- 2018-05-29	3.4%（401） 是否引证：否
14	1533140526_张宇翔_音乐网站的设计与实现_蒋树清 张宇翔 - 《大学生论文联合比对库》- 2019-05-14	3.4%（401） 是否引证：否
15	4_袁晓芳_基于Vue架构的音乐APP的设计与实现 袁晓芳 - 《大学生论文联合比对库》- 2019-04-18	3.3%（391） 是否引证：否
16	基于web的课程平时成绩管理系统设计与实现 谭坪盛 - 《大学生论文联合比对库》- 2018-03-24	3.1%（369） 是否引证：否
17	186_萧志新_基于Vue.js框架的相册定制商城的设计与实现 萧志新 - 《大学生论文联合比对库》- 2018-05-03	3.0%（355） 是否引证：否
18	基于公钥加密的即时通信系统 杨占强 - 《大学生论文联合比对库》- 2016-05-25	2.9%（346） 是否引证：否
19	基于Node.js的大学照片分享交友平台的设计与实现 徐文臻 - 《大学生论文联合比对库》- 2019-05-24	2.7%（325） 是否引证：否
20	高校毕业论文选题系统的设计与实现 李华夏(导师：牛新征;赵明友) - 《电子科技大学硕士论文》- 2011-09-01	2.7%（320） 是否引证：否
21	基于Web的医院内部基本信息管理系统的设计与实现 魏妮 - 《大学生论文联合比对库》- 2018-06-18	2.6%（316） 是否引证：否
22	轻松短租网的设计与实现	2.5%（303）

王振 - 《大学生论文联合比对库》- 2017-04-27	是否引证：否
23 基于vue的韩衣都舍mobileAPP开发	2.5%（294）
童鑫 - 《大学生论文联合比对库》- 2018-05-22	是否引证：否
24 毕业设计	2.4%（288）
周怡 - 《大学生论文联合比对库》- 2016-05-13	是否引证：否
25 基于Nodejs的多功能聊天室	2.3%（271）
徐洋 - 《大学生论文联合比对库》- 2016-05-08	是否引证：否
26 吴豪_基于nodejs技术的小型博客类应用设计与实现	2.2%（262）
吴豪 - 《大学生论文联合比对库》- 2016-01-04	是否引证：否
27 在线视频播放系统	1.2%（139）
池健敏; - 《福建电脑》- 2013-01-25	是否引证：否
28 图书管理信息系统的设计与实现	0.9%（102）
马立;杨静; - 《科技视界》- 2012-11-15	是否引证：否
29 成人高校论文指导交流系统设计与实现	0.7%（83）
蒋彭(导师：郭文明) - 《北京邮电大学硕士论文》- 2010-10-01	是否引证：否
30 单孔腹腔镜在育龄期女性卵巢良性肿瘤患者中的应用	0.6%（69）
王丹莹(导师：刘海元) - 《北京协和医学院硕士论文》- 2019-04-01	是否引证：否
31 血浆单采机软件的设计与实现及抗凝剂精确控制的研究	0.6%（69）
郑亚斌(导师：廖勇) - 《电子科技大学硕士论文》- 2016-07-01	是否引证：否
32 基于WFMS的办公自动化应用系统设计	0.5%（58）
赵黎华(导师：任长明) - 《天津大学硕士论文》- 2003-06-01	是否引证：否
33 基于实时数据流处理的http数据分析可视化系统	0.4%（50）
潘冬(导师：刘玓) - 《电子科技大学硕士论文》- 2016-07-01	是否引证：否
34 基于OCL的Web服务测试方法研究	0.3%（37）
冯细光(导师：刘建勋) - 《湖南科技大学硕士论文》- 2010-04-01	是否引证：否
35 面向铁塔公司的共建共享系统设计	0.3%（32）
俞昌虹;叶敏; - 《电信工程技术与标准化》- 2015-10-15	是否引证：否
36 软件测试方法概述	0.3%（32）
郭文梁; - 《科技与企业》- 2012-08-22	是否引证：否

原文内容

第三章详细的介绍了前后端搭配数据库实现实现功能；

第四章系统测试部分，主要对功能进行了具体分析和测试，并提出解决方案，实现系统功能

2 系统需求分析及总体设计

2.1 功能需求

2.1.1 业务逻辑：

1. 首次使用的用户可以查看文章、评论和问题。若想进行其他操作则需要进行注册登录。

2. 新用户登录后可发表文章，并且可在个人中心对发表的文章进行查看、修改、删除；在浏览其他用户查看文章时可以对文章进行评论；亦可对其他用户提出的问题进行回答。

3. 用户可查看不同类型的文章

4. 用户在发表评论后，也可对自己或者其他用户发表的评论进行查看、回复

5. 用户可提出问题，等带其他用户回答

6. 用户可以对文章进行点赞、收藏

7. 用户可退出登录

2.1.2 系统功能

1. 对文章进行发表、 删除、修改、删除

2. 提出问题、答复问题、删除问题、查看问题及问题答复信息

3. 发表评论，答复评论、查看评论信息

4. 点赞、验证、收藏

5. 查看用户信息

2.1.3 系统流程图

进入平台页面中，若是用户未登录可以对文章、评论、提问、回答进行查看。若是登录用户，除了未登录用户的一些操作以外，还可以发表自己的文章，并在个人中心页面对其进行修改、删除等操作；也可以发表自己的疑问，等待他人的回答，或者回答他人的问题，同样可以在个人中心页面对自己发表的问题进行操作。查浏览文章时，若是有认为写的不错的，可收藏和点赞，方便以后再次浏览这篇文章。

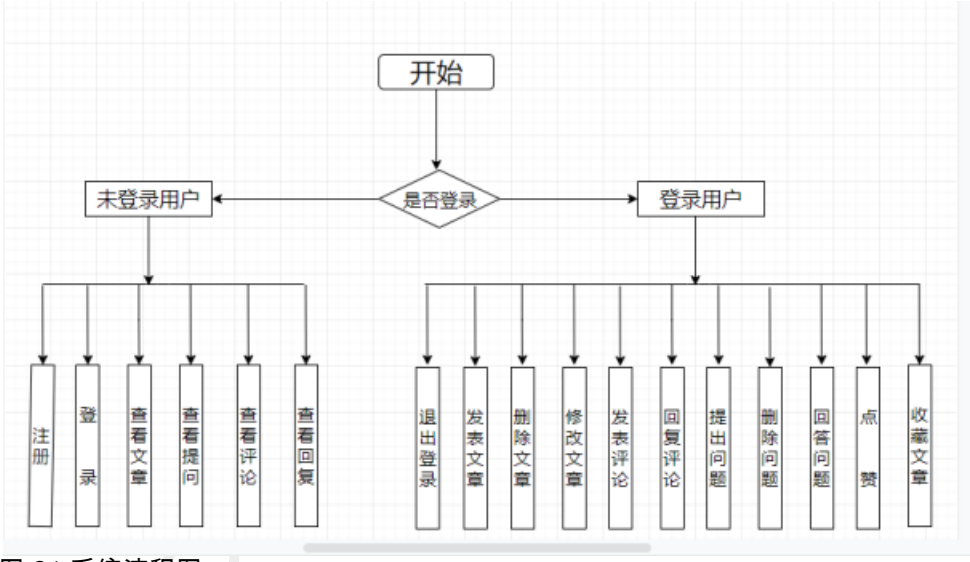

图 21 系统流程图

2.2 操作和开发选择

2.2.1 页面操作

1. 允许用户进行登录、注册、退出登录
2. 允许用户查看文章、问题、评论
3. 允许用户发表文章、发表评论、提出问题、回答问题
4. 允许用户删除文章、删除问题
5. 允许用户对已发表的文章进行修改
6. 允许用户点赞、收藏

2.2.2 开发选择

1、MVVM模式

MVVM是Model-View-ViewModel的简写，由经典的软件架构MVC衍生而来，最早是由微软公司提出并运用，是MVP（Model-View-Presenter）模式与WPF结合的应用方式时发展演变过来的一种新型架构架构；MVVM有助于将图形用户界面的开发与业务逻辑或后端逻辑（数据模型）的开发分离开来 [15]。

MVVM是一种数据双向绑定的响应式框架。主要把每个页面分层了M(模型)、V(视图)、VM(视图模型)，VM(视图模型)是MVVC模式的核心思想，将视图和模型关联起来。Model(模型):即数据层，负责页面中数据存储。View(视图)：即视图层，从ViewModel层获取数据，然后显示。ViewModel: 视图模型层，侦听视图和模型更改，并封装处理业务逻辑、封装网络处理和数据缓存。View(视图)和ViewModel(视图模型)之间建立联系通过双向绑定。

在 MVVM 架构下，View和Model之间没有直接联系，他们之间的交互通过ViewmModel进行。模型(Model) 和视图模型(ViewModel) 双向交互， 因此 ，View上数据的发生改变会同步到 Model 中，而 Model中数据的变化也会马上反应到 View 上。

ViewModel 通过数据双向绑定将 View 层和 Model 层进行连接，View 和 Model 之间工作自动同步，不需要人为去干涉，所以，开发者只需专注于业务逻辑，不需要手动去操作 DOM, 也不需要注意数据状态同步的问题，复杂的数据状态完全由 MVVM 来统一管理、维护。

2、Vue.js

Vue.js是一个轻巧、高性能、可组件化的MVVM库，同时拥有非常容易上手的API，是一个构建数据驱动的Web界面的库，是一套构建用户界面的渐进式框架，与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用[2]。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合，另一方面，Vue 完全有能力驱动采用单文件组件和 Vue 生态系统支持的库开发的复杂单页应用，Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件[2]。

Vue.js具有插件化、数据双向绑定、轻量级架构、提供指令和过滤器等特点。Vue 的核心是一个 "响应式的数据绑定系统"。在项目开发过程中，数据和视图绑定后,数据将保持同步。每当修改数据后， DOM 也进行相应的更新；反之，在视图上进行操作，相关的数据也会随之改变。即关于DOM的操作都由Vue执行了，我们只需操作数据即可。除此之外，在项目的开发过程中，使用Vue.js 可以引入插件和第三方库，能够很方便的搭建前端页面和实现一部分功能，并且完成与后台的交互功能，便于开发。

3、Node.js

Node.js是一个基于Chrome V8 引擎的 JavaScript 运行环境；也是运行在服务端的JavaScript，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装[5]。Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。 V8引擎执行Javascript的速度非常快，性能非常好。Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用，Node.js使用事件驱动， 非阻塞I/O模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用[5]。

Node.js的出现打破了前端脚本语言和后台开发语言之间的屏障，使JavaScript不仅能用于前端开发，而且还能用于后台开发，让Web开发变得更简单；更重要的是，Node.js在性能、实时处理和高并发等多个领域都有非常优异的表现。Node.js提供了一种简单的、用于创建高性能服务器及可在该服务器中运行的各种应用程序的开发工具[8]。

4、MongoDB

与传统的关系型数据库不同，MongoDB是一种面向文档的数据库；是一个基于分布式文件存储的开源数据库系统；MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成；MongoDB 文档类似于 JSON 对象；字段值可以包

含其他文档，数组及文档数组[3]。

在MongoDB中多个文档组成集合，同样的是多个集合组成数据库。其中文档是一组键值(key-value)对(即 BSON)的形式，即{ "name":"MongoDB" }。MongoDB文档中的值支持不同的字段，相同的字段支持不同的数据类型，这与关系数据库不同。当然，相同key不能同时存在于文档中，不同的类型和大小写对应的key也不一样。若是说文档类似于数据库中的行，那么集合是如同数据库的表。MongoDB中集合是无模式的，也就是说集合中的文档不但值的类型可以不同，键也可以不同。当然也可以将不同类型的文档放到不同的集合中。

在本项目设计开发的过程中可以使用MongoDB丰富的表达式对数据进行操作。在搭配Node.js进行后台开发时，使用update修改器: $inc $set $unset $push $pull，对数据或指定的数据字段进行增加、修改、删除等操作。同样可以使用json样式的指令来轻松地查询文档中的数组和对象。。并且在向数据库中插入数据时，不需要事先在数据库的集合中建立字段，只需将前端页面需要的字段与后台对接上即可。

3 系统详细设计及实现

3.1 数据库设计

3.1.1 E-R图

系统在数据库设计上使用了六个数据库集合来实现系统功能

1. 问题集合：主要是记录用户提问的相关信息
2. 答复集合：记录对问题的答复，一个问题对应多个答复
3. 用户集合：记录用户信息
4. 文章集合：记录系统的文章数据
5. 评论集合：记录每篇文章的评论，与文章实体通过外键一对多关联
6. 收藏集合：记录用户收藏的文章，一个用户可收藏多篇文章

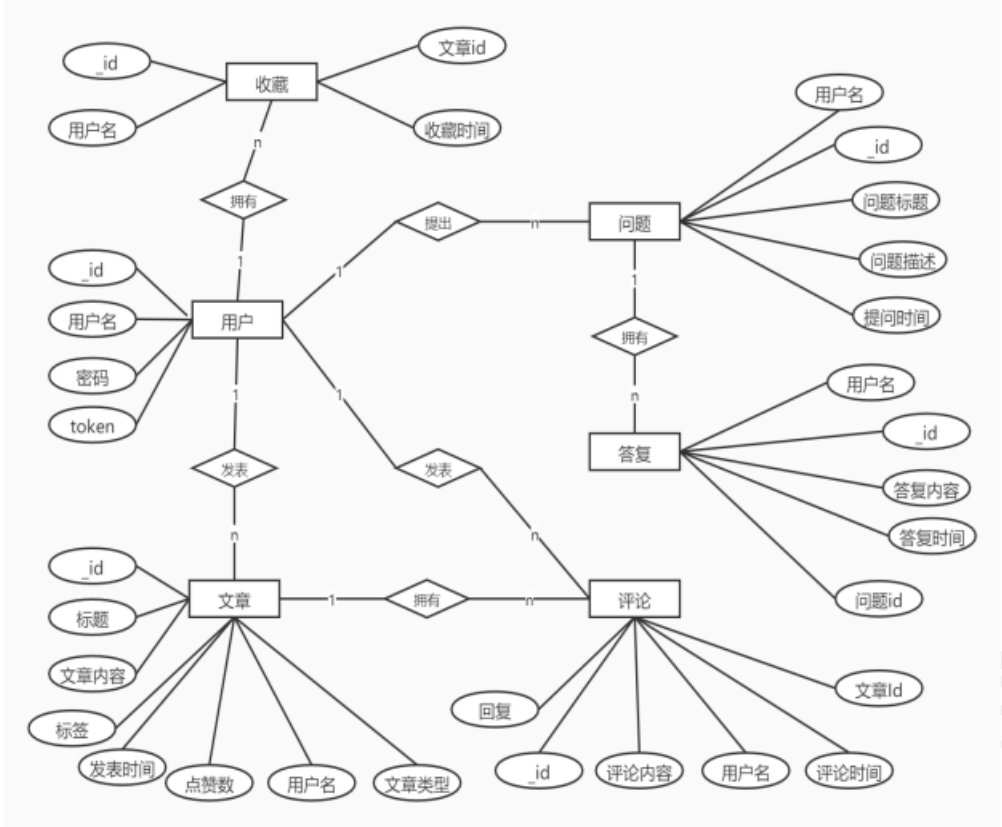

图 31 E-R图

3.1.2 数据库需求分析

1. 用户分为登录用户和未登录用户
2. 未登录用户可注册、查看文章、评论、问题等
3. 登录用户可以对文章进行发布、查看、修改、删除；亦可对评论进行发表、删除、回复、查看；还可以提问、答复、点赞、收藏
4. 一篇文章可拥有多评论
5. 一个提问可以有多答复
6. 一种类型对应多篇文章
7. 一个问题对应多个答复
8. 一个评论可以有多个答复

实体属性：

1. 用户：唯一_ID、用户名、密码
2. 文章：唯一ID、文章标题、文章内容、文章标签、发表时间、点赞数、评论、文章类型、用户名
3. 评论：唯一ID、评论内容、评论时间、评论文章Id、用户名
4. 问题：唯一ID、提问内容、提问时间、答复、用户名
5. 回复：唯一ID、回复内容、回复时间、用户名、回复问题id / 回复评论id

3.2 前端功能实现

3.2.1 前端分析

前端项目使用vue-cli脚手架搭建项目，使用Vue，并在项目中引入组件和第三方库，将不同页面分成不同的组件进行开发。

目录位置：src/

1. api: 存放前端与后台进行请求交互的接口，
2. assets：资源目录，放置图片或者公共js、公共css。文件夹下的资源会被webpack构建；
3. components：组件目录，项目组件所放位置；
4. components/home/Index.vue 项目的首页，显示文章列表信息，可文章进行点赞功能
5. components/nav/Nav.vue 项目头部导航栏，包括登录、注册、退出登录、搜索、发表文章、提问等功能
6. components/register/Register.vue： 注册组件,点击导航栏的注册填写用户名和密码完成注册功能，其中用户名是唯一的
7. components/login/Login.vue: 登录组件，用户点击导航栏的登录完成登录功能
8. components/edit/Edit.vue: 发表文章组件，用户点击导航栏的发表文章进入该组件填写项对应的内容，完成发表文章的功能
9. components/details/: 详情组件，
10. Detail.vue在文章列表点击某篇文章，进入该组件查看文章的内容信息
11. QuestionDetail.vue 进入该组件可查看问题详细信息，以及回答问题、查看所有关于该问题的回答
12. components/personal/Personal.vue： 个人中心组件，用户可在该组件中查看自己个个人信息，以及所发表的文章
13. components/category/category.vue: 专栏组件，包括不同类型的文章
14. components/common/： 公共组件，供其他组件复用
15. components/comments/Comments.vue: 评论组件，当用户想要评论一篇文章时，点击文章页面上的评论按钮，输入相关内容来完成对文章的评论功能router：前端路由，实现路由跳转。配置的路由路径写在index.js里面；
16. store: vuex状态管理文件，管理共享数据
17. utils: 组件所需的公共方法
18. App.vue：根组件；
19. main.js：入口js文件

3.2.2 页面路由

Vue中使用vue-router插件实现页面之间的跳转。下面是部分路由信息：

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)
const routerPush = Router.prototype.push
Router.prototype.push = function push (location) {
  return routerPush.call(this, location).catch(error => error)
}
export default new Router({
  // 设置 链接激活时使用的 CSS 类名。默认值可以通过路由的构造选项 linkActiveClass 来
  linkActiveClass: 'active',
  routes: [
    {
      path: '/',
      beforeEnter (to, from, next) {
        if (to.path === '/') {
          next('/index')
        } else {
          next()
        }
      }
    },
    {
      path: '/index',
      name: 'Index',
      component: () => import('@/components/home/Index.vue')
```

图 32 前端路由

```
{
  path: '/edit',
  name: 'Eidt',
  component: () => import('@/components/edit/Edit.vue')
},
{
  path: '/personal',
  name: 'Personal',
  component: () => import('@/components/personal/Personal.vue'),
  meta: { toPersonal: false },
  beforeEnter (to, from, next) {
    if (to.name === 'Personal') {
      next({name: 'PersonArticle'})
    } else {
      next()
    }
  },
  children: [{
    path: 'personArticle',
    name: 'PersonArticle',
    component: () => import('@/components/personal/component/PersonArticle.vue')
  },
  {
    path: 'waitReply',
    name: 'WaitReply',
    component: () => import('@/components/common/WaitReply.vue')
```

图 33 前端路由

3.2.3 请求跨域

在前端领域中，跨域是指浏览器允许向服务器发送跨域请求，从而克服Ajax只能同源使用的限制。

前端设置跨域：在config/index中设置代理

```
proxyTable: {
  '/api': {
    target:'http://localhost:8081', // 你请求的第三方接口
    changeOrigin:true, // 在本地会创建一个虚拟服务端，然后发送请求的数据，并同时接收请求的数据
    pathRewrite:{  // 路径重写，
      '^/api': ''  // 替换target中的请求地址，
    }
  }
},
```

图 34 前端跨域

后端使用cors （跨域资源共享）和设置请求头

```
const cors = require('cors')
// 全局配置
app.use(cors())
```

```
// 设置跨域访问
app.all('*', function (req, res, next) {
  // 设置允许跨域的域名，*代表允许任意域名跨域
  res.header('Access-Control-Allow-Origin', '*')
  // 允许的header类型
  res.header('Access-Control-Allow-Headers', 'content-type')
  // 跨域允许的请求方式
  res.header('Access-Control-Allow-Methods', 'DELETE,PUT,POST,GET,OPTIONS')
  // 让options尝试请求快速结束
  if (req.method.toLowerCase() === 'options') { res.send(200) } else { next() }
})
```

图 35后台跨域

3.2.4 富文本

在项目中，通过使用富文本插件mavon-editor对文章文章、问题进行编辑和显示

编辑文章页面：

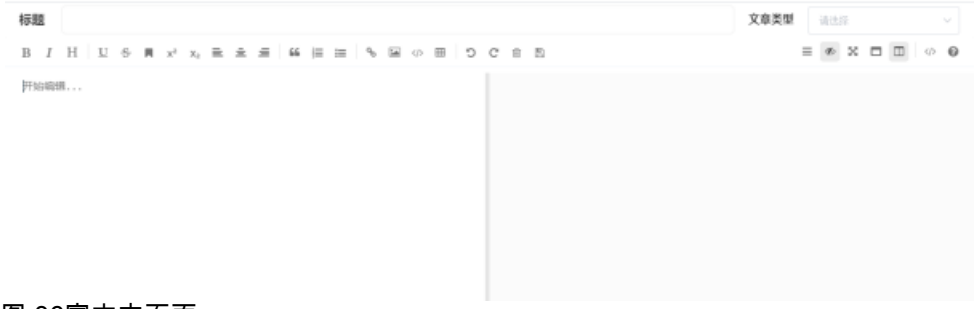

图 36富文本页面

在前端组件中使用富文本组件：

富文本组件绑定的值分别为：

:value=".content"：引入要转换的内容

:subfield = "false"：开启单栏模式

:defaultOpen = "'preview'"：默认展示预览区域
:toolbarsFlag = "false"：关闭工具栏
:editable="false"：不允许编辑
scrollStyle="true"：开启滚动条样式(暂时仅支持chrome)
:ishljs = "true"：开启代码高亮

```
<mavon-editor
  class="md"
  :value="content"
  :subfield = "false"
  :defaultOpen = "'preview'"
  :toolbarsFlag = "false"
  :editable="false"
  :scrollStyle="true"
  :ishljs = "true">
</mavon-editor>
```

图 37富文本组件
使用富文本进行编辑：
其中@change="change" 绑定change事件，在该事件中，所有操作都会被解析并重新渲染

```
<mavon-editor v-model="content"
  ref="md"
  :ishljs = "true"
  @change="change"
  @imgAdd="handleAddImg"
  style="min-height: 560px"/>
```

图 38 组件中使用富文本
3.2.5 前端输入校验
采用elementUI插件中的form进行布局，并绑定自定义规则对输入的内容进行校验，校验成功则执行登录、注册功能

```
<el-form :model="registerRuleForm" status-icon :rules="rules" ref="registerRuleForm"
  <el-form-item label="用户名" prop="username">
    <el-input
      type="text"
      v-model="registerRuleForm.username"
      autocomplete="off"
      @keyup.enter.native="handleRegiterInputFocus('password')">
    </el-input>
  </el-form-item>
  <el-form-item label="密码" prop="userpwd">
    <el-input
      type="password"
      ref="password"
      v-model="registerRuleForm.userpwd"
      autocomplete="off"
      @keyup.enter.native="handleRegiterInputFocus('checkpwd')"></el-input>
  </el-form-item>
```

图 39 HTML代码

```
// 校验用户名
var validateUsername = (rule, value, callback) => {
  var reg = /^[a-zA-Z0-9]{4,11}$/
  if (value === '') {
    callback(new Error('请输入用户名'))
  } else if (!(reg.test(value))) {
    callback(new Error('请输入4-11位数字或字符串的用户名'))
  } else {
    callback()
  }
}
// 校验密码
var validatePass = (rule, value, callback) => {
  var reg = /^[a-zA-Z]\w{4,17}$/
  if (value === '') {
    callback(new Error('请输入密码'))
  } else if (!(reg.test(value))) {
    callback(new Error('以字母开头，长度在4-18之间，只能包含字符、数字和下划线'))
  } else {
    callback()
  }
```

图 310 输入校验

```
rules: {
  username: [
    { required: true, validator: validateUsername, trigger: 'blur' }
  ],
  userpwd: [
    { required: true, validator: validatePass, trigger: 'blur' }
  ],
  checkpwd: [
```

图 311 校验规则

3.3 后台功能实现

3.3.1 后台分析

后台使用Node来搭建服务器，使用MongoDB作为数据库， 完成数据的增删改查

后台目录位置： server/

1. server/base-server.js: 搭建服务器，写后台接口，使用bodyParser解析中间件
2. server/dev-server.js: 设置请求头，实现跨域，开发的后台这此运行，进入server 文件夹后运行 node dev-server.js
3. server/handle.js: 具体实现一系列的后台接口，并暴露出去
4. server/token.js：生成登录时的token，有效期一周，校验token
5. server/utils.js：获取客户端ip地址方法

3.3.2 用户注册

组件Register.vue 用于实现用户注册功能。用户点击注册，输入用户名和密码，实现注册功能。用户名唯一，在点击注册时，会先从数据库中查询该用户是否注册过，若是没有注册则向数据库插入一条数据，完成注册，否则提示用户名已存在。注册完成后，进入Login组件来登录并执行其他的操作。

// 引入MongoDB数据库
const MongoClient = require('mongodb').MongoClient
// 将前端拿到的String类型 _id 转为ObjectId类型
const ObjectID = require('mongodb').ObjectID
const url = 'mongodb://localhost:27017/'
const config = { useUnifiedTopology: true }

数据库用户集合

_id<ObjectId>: 必选唯一ID
user<String>: 必选用户名
password<String>: 必选密码
token<String>: 必选验证登录

表 31 user集合

集合名称	文档对象
user	{ "_id" : ObjectId("5e99742f39deb4347c7aeef1"), "user" : "admin", "password" : "admin", "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 " }

集合名称文档对象

user {
"_id" : ObjectId("5e99742f39deb4347c7aeef1"),
"user" : "admin",
"password" : "admin",
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 "
}

实现注册功能的代码如下：

```
register (req, res) {
  MongoClient.connect(url, config, (err, db) => {
    if (err) throw err
    const dbo = db.db('project')
    const userinfo = req.body
    const collection = dbo.collection('user')
    collection.findOne({user: userinfo.user}).then(result => {
      if (result) {
        res.send({
          code: -1,
          msg: '该用户已存在'
        })
```

```
    } else {
      collection.insertOne(userinfo, err => {
        if (err) {
          res.send({
            code: 1,
            msg: '注册失败'
          })
        } else {
          res.send({
            code: 0,
            msg: '注册成功'
          })
        }
        db.close()
      })
    }
  })
})
```

图 312 后台注册代码

3.3.3 用户登录

用户在注册成功后，会进入登录组件，用户可以在此输入用户名和密码进行登录，除此之外可以点击页面的登录按钮，输入用户名和密码来完成登录。点击登录时，首先根据输入的用户名在数据库中查询它的存在。如果不存在，提示该用户不存在。若是用户名和密码都正确，登录成功则生成一个token，将其插入数据库并返回给前端，否则提示登录失败。

实现登录功能代码如下：

```
collection.findOne({ user, password }).then(result => {
  if (!result) {
    res.send({
      code: 1,
      msg: '没有查询到该用户'
    })
  } else {
    const token = generateToken(user)
    const updateContent = {
      $set: {
        token
      }
    }
```

```
    collection.updateOne({ user, password }, updateContent, err => {
      if (err) {
        res.send({
          code: 1,
          msg: '登陆失败，请重试'
        })
      } else {
        res.send({
          code: 0,
          data: token
        })
      }

      db.close()
    })
```

图 313 后台登陆

3.3.4 退出登录

用户在点击退出登录后，会删除数据库和保存在前端localStorage中的token，完成退出登录功能。

```
    const updateContent = {
      $unset: {
        token: 1
      }
    }
    collection.updateOne({user: user}, updateContent, err => {
      if (err) {
```

图 314 退出登陆

3.3.5 操作文章

用户可以点击发表文章按钮进入Edit.vue组件。在点击发表文章时会进行验证，若是已登录则直接进入编辑页面进行编辑，否则提示请先登录。进入编辑组件后填写相关的文章信息，点击发布文章后可填写文章标签，然后完成发布文章功能同。在发布文章时也会验证用户是否登录。发布文章后，用户可以在首页搜索、或者进入个人中心查看发布的文章，除此之外还可以在个人中心页面对所发布的文章进行删除、修改操作。若是查看文章，则进入Detail.vue 组件查看文章详细内容，也可以在文章详细页面底部的评论区对文章进行评论。同时，也可对其他用户关于这篇文章的评论进行回复。除此之外，若是觉得某一篇

文章写得不错，可以点赞和收藏。点赞过后，会将点赞用户的用户名存放到文章集合中。当然，评论、回复、点赞与收藏同样需要进行验证。

数据库文章集合

_id<ObjectId>： 必选唯一ID

articleTitle<String>：必选文章标题

articleType<String>： 必选文章类型

articleContent<String>：必选文章内容

articleTags<Array>： 可选标签

username<String>： 必选发表人

date<Number>：必选时间

likes<Array>：可选点赞

表 32 articles集合

集合名称	文档对象
articles	{ "_id" : ObjectId("5ea636850f59713cd40cf8d0"), "articleTitle" : "7大原始类型与Object类型", "articleType" : "软件工程", "articleContent" : " 7大原始类型与Object类型 \n "articleTags" : ["js","前端"], "username" : "admin", " date " : 1587951237253, "likes" : ["admin", "root"] }

集合名称文档对象

```
articles {
"_id" : ObjectId("5ea636850f59713cd40cf8d0"),
"articleTitle" : "7大原始类型与Object类型",
"articleType" : "软件工程",
"articleContent" : "<p>7大原始类型与Object类型</p>\n
"articleTags" : ["js","前端"],
"username" : "admin",
" date " : 1587951237253,
"likes" : ["admin", "root"]
}
```

评论集合

_id<ObjectId>： 必选唯一ID

articleId<String>： 必选文章ID

comment<String>： 可选评论

username<String> ：必选评论人

date<Number>： 必选时间

reply<Array>： 可选回复

username<String>： 必选回复人

content<String>：必选回复详情

date<Number>：必选回复时间

表 33 comments集合

集合名称	文档对象
comments	{ "_id" : ObjectId("5ea689033e8afc3954b48c4e"), "articleId" : "5ea631220f59713cd40cf8cd", "comment" : "Vuex的5个核心属性是什么？", "username" : "admin", "date" : 1587972355676, "reply" : [{ "username" : "root", "content" : "分别是 state、getters、mutations、actions、modules ", "date" : 1587972390018 }] }

集合名称文档对象

```
comments {
"_id" : ObjectId("5ea689033e8afc3954b48c4e"),
"articleId" : "5ea631220f59713cd40cf8cd",
"comment" : "Vuex的5个核心属性是什么？",
"username" : "admin",
"date" : 1587972355676,
"reply" : [
{
"username" : "root",
"content" : "分别是 state、getters、mutations、actions、modules ",
"date" : 1587972390018
}
]
}
```

数据库收藏集合

_id<ObjectId>： 必选唯一ID

articleId<String>： 必选文章id

username<String>：必选收藏人

date<Number>：必选时间

表 3-4 collect集合

集合名称	文档对象
collect	{ "_id" : ObjectId("5eaa9307499b600d38c2b7b3"), "username" : "admin", "articleId" : "5ea8e6d3b022c9455021c59b", "date" : 1588237063591 }

集合名称文档对象

collect {

"_id" : ObjectId("5eaa9307499b600d38c2b7b3"),

"username" : "admin",

"articleId" : "5ea8e6d3b022c9455021c59b",

"date" : 1588237063591

}

验证：

```
const dbo = db.db('project')
const token = req.get('userToken')
// 验证 token
const vaild = await isVaildToken(dbo, token)
const collection = dbo.collection(articleCollection)
if (!vaild) {
  res.send({
    code: 1,
    msg: 'token 失效，请重新登陆'
  })

  db.close()
  return
}
```

图 315 校验

发布文章：

```
const date = (new Date()).getTime()
const articleData = {
  ...req.body,
  date: date,
  likes: []
}

collection.insertOne(articleData, err => {
  if (err) {
    res.send({
      code: 1,
      msg: '发布失败'
    })
  } else {
    res.send({
      code: 0,
      data: '发布成功'
    })
  }

  db.close()
```

图 316 发布文章

修改文章：

```
if (req.body.id) {
  const query = { _id: ObjectID(req.body.id) }
  const body = req.body
  const updateContent = {
    $set: {
      articleTitle: body.articleTitle,
      articleType: body.articleType,
      articleContent: body.articleContent,
      articleTags: body.articleTags,
      username: body.username
    }
  }

  collection.updateOne(query, updateContent, err => {
    if (err) {
      res.send({
        code: 1,
        msg: '更新失败'
      })
    } else {
      res.send({
        code: 0,
```

图 317 修改文章

删除文章：

```
  // 验证 token
  const vaild = await isVaildToken(dbo, token)
  const query = { _id: ObjectID(req.body.id) }
  if (!vaild) {
    res.send({
      code: 1,
      msg: 'token 失效，请重新登陆'
    })

    db.close()
    return
  }

  dbo.collection(articleCollection).deleteOne(query, err => {
    if (err) {
      res.send({
        code: 1,
        msg: '删除失败'
      })
    } else {
```

图 318 删除文章

查看文章：

```
const query = req.query
// ~~取整
const size = ~~query.pageSize
const index = ~~query.currentPage
const queryObj = {}
const collection = dbo.collection(articleCollection)
if (query.articleType) {
  queryObj.articleType = query.articleType
} else if (query.articleTags) {
  let tag = query.articleTags
  queryObj.articleTags = {$elemMatch: {$eq: tag}}
} else if (query.articleTitle) {
  let title = query.articleTitle
  queryObj.articleTitle = {$regex: title, $options: '$i'}
} else if (query.username) {
  queryObj.username = query.username
}
```

```
collection.find(queryObj).count((err, num) => {
  if (err) throw err
  collection.find(queryObj).sort({'date': -1}).skip(size * (index - 1)).limit(size).toArray(
    if (err) {
      res.send({
        code: 1,
        msg: '查找失败',
        data: []
      })
    } else {
      res.send({
        code: 0,
        data: result,
        total: num
      })
    }

    db.close()
  })
})
```

图 319 查看文章

发表评论：

```
    if (err) throw err
    const dbo = db.db('project')
    const token = req.get('userToken')
    // 验证 token
    const vaild = await isVaildToken(dbo, token)
    const collection = dbo.collection(commentCollection)
    if (!vaild) {
      res.send({
        code: 1,
        msg: 'token 失效，请重新登陆'
      })

      db.close()
      return
    }
```

```
    const date = (new Date()).getTime()
    const commentData = {
      ...req.body,
      date: date,
      reply: []
    }

    collection.insertOne(commentData, err => {
      if (err) {
        res.send({
          code: 1,
          msg: '评论失败'
        })
      } else {
        res.send({
          code: 0,
          data: date
        })
      }

      db.close()
    })
```

图 320 发表评论

回复评论：

```
    const date = (new Date()).getTime()
    const replyData = {
      $push: {
        reply: {
          username: req.body.username,
          content: req.body.content,
          date
        }
      }
    }
    collection.updateOne(query, replyData, err => {
      if (err) {
        res.send({
          code: 1,
          msg: '回复失败'
        })
      } else {
        res.send({
          code: 0,
          data: '回复成功'
        })
      }
      db.close()
```

图 321 回复评论

点赞、取消点赞：

```
const query = { _id: ObjectID(req.body.id) }
const body = req.body
if (body.liked) {
  likedContent = {
    $pull: {
      likes: body.username
    }
  }
} else {
  likedContent = {
    $push: {
      likes: body.username
    }
  }
}
collection.updateOne(query, likedContent, err => {
  if (err) {
    res.send({
      code: 1,
      msg: '点赞失败'
    })
  } else {
    res.send({
      code: 0,
      data: '点赞成功'
    })
  }
  db.close()
})
```

图 322 点赞

收藏功能

```
if (req.body.id) {
  const query = { _id: ObjectID(req.body.id) }
  collection.deleteOne(query, err => {
    if (err) {
      res.send({
        code: 1,
        msg: '取消收藏失败'
      })
    } else {
      res.send({
        code: 0,
        msg: '取消收藏成功'
      })
    }
    db.close()
  })
} else {
  const date = (new Date()).getTime()
  const collectData = {
    username: req.body.username,
    articleId: req.body.articleId,
    date
  }
  collection.insertOne(collectData, err => {
    if (err) {
      res.send({
        code: 1,
        msg: '收藏失败'
```

图 323 收藏功能

3.3.6 操作提问

用户点击提问进入AskQuestion.vue组件进行对问题进行编辑。在填写完相关的信息后，点击按钮发表问题。在发表问题时，同样需要进行验证。发布成功后可在当前组件查看所有问题，也可进入个人中心的等待回答中查看个人发布的问题，也可对提出的问题进行删除，删除问题的同时也会将关于该问题的回答一并删除。也可进入QuestionDetail.vue 组件查看问题详细信息，对提出的问题进行回答，并在问题详情页查看对于该问题的回答。

数据库问题集合

_id<ObjectId>： 必选唯一ID

questionTitle<String>：必选问题标题

questionContent<String>：可选问题描述

date<Number>：必选时间

username<String>：必选提问人

表 3-5 questions集合

集合名称	文档对象
questions	{ "_id" : ObjectId("5ea2a0c47f73002364e4669f"), "questionTitle" : "Node.js写后端的优势？", "questionContent" : "" "date" : 1587716292386 "username" : "admin" }

集合名称文档对象

questions {

"_id" : ObjectId("5ea2a0c47f73002364e4669f"),

"questionTitle" : "Node.js写后端的优势？",

"questionContent" : ""

"date" : 1587716292386

"username" : "admin"

}

数据库问题答复集合

_id<ObjectId>： 必选唯一ID

questionId<String>： 必选问题ID

content<String>：必选回复内容

username<String>：必选回复人

date<Number>：必选时间

表 3-6 replys集合

集合名称	文档对象
replys	{ "_id" : ObjectId("5ea2a0137f73002364e4669e"), "questionId" : "5ea0f79ac63ccf0b68f5df99", "content" : " node.js是使用非阻塞、事件驱动I/O模型使其轻量、高效，能够构建数据密集型、实时的跨平台应用 \n", "username" : "fangfang", "date" : 1587716115360 }

集合名称文档对象

replys {

"_id" : ObjectId("5ea2a0137f73002364e4669e"),
"questionId" : "5ea0f79ac63ccf0b68f5df99",
"content" : " node.js是使用非阻塞、事件驱动I/O模型使其轻量、高效，能够构建数据密集型、实时的跨平台应用 \n",
"username" : "fangfang",
"date" : 1587716115360
}

发表问题页面：

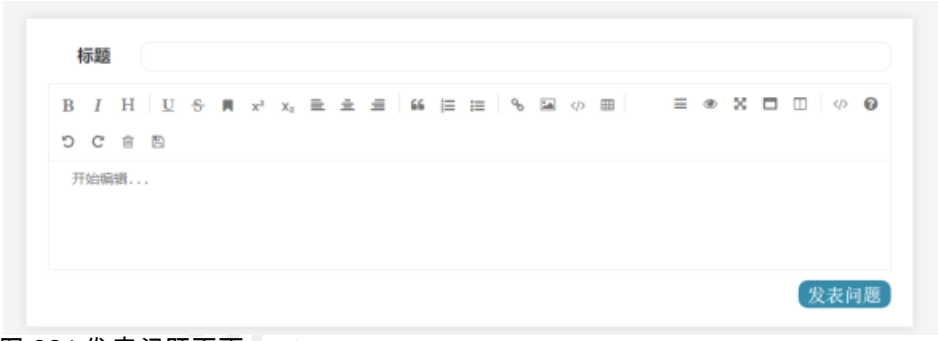

图 324 发表问题页面

发表问题：

```
const date = (new Date()).getTime()
const questionData = {
  ...req.body,
  date: date
}

collection.insertOne(questionData, err => {
  if (err) {
    res.send({
      code: 1,
      msg: '发布失败'
    })
  } else {
    res.send({
      code: 0,
      data: '发布成功'
    })
  }

  db.close()
})
```

图 325 发表问题

查询所有问题：

```
const dbo = db.db('project')
const query = req.query
const queryObj = {}
const collection = dbo.collection(questionCollection)

if (query.username) {
  queryObj.username = query.username
}
collection.find(queryObj).count((err, num) => {
  if (err) throw err
  collection.find(queryObj).sort({'date': -1}).toArray((err, result) => {
    if (err) {
      res.send({
        code: 1,
        msg: '查找失败',
        data: []
      })
    } else {
      res.send({
        code: 0,
        data: result,
        total: num
      })
    }

    db.close()
```

图 326 查询问题

删除问题：

```
    const query = { _id: ObjectID(req.body.id) }
    const queryObj = { questionId: req.body.id }
```

```
dbo.collection(questionCollection).deleteOne(query, err => {
  if (err) {
    res.send({
      code: 1,
      msg: '删除失败'
    })
  } else {
    delReply(queryObj)
    res.send({
      code: 0,
      msg: '删除成功'
    })
  }

  db.close()
```

图 327 删除问题

删除问题的相关回答：

```
// 删除问题的同时删除回复内容
function delReply (query) {
  MongoClient.connect(url, config, (err, db) => {
    if (err) throw err
    const dbo = db.db('project')
    dbo.collection(replyCollection).deleteMany(query, (err, res) => {
      if (err) throw err
      console.log('删除数据条数 ' + res.deletedCount)
      db.close()
    })
  })
}
```

图 328 删除问题相关回复

4 系统测试

4.1 测试目的

测试是在程序投入运行前，对程序的需求分析、设计规格说明和编码的最终复审，是保证程序质量的关键步骤，软件测试时指为没发现错误面执行程序的过程，根据各个阶段的规格说明书、内部结构设置一些测试用例，用这些测试用例去执行程序，以发现程序错误的过程[16]。

测试的目的就是在软件投入生产运行之前，尽可能多地发现软件中存在的错误，在开发大型软件系统的过程中，难免会遇到许多错综复杂的问题，在软件生存周期的每个阶段都不可避免地会产生错误，因此，在每个阶段结束之前通过严格的技术审查，以便尽可能早的发现并纠正错误[16]。

4.2 测试方法

主要方法为黑盒测试。

黑盒测试：也称为功能测试，不考虑软件的内部结构和特性，测试功能的实现。根据软件的需求规格说明书设计测试用例，从程序的输入和输出特性上测试程序是否与功能要求一致。只考虑软件功能要求，不涉及程序的内部结构和实现细节[16]。

4.3 测试用例

4.3.1 用户测试

测试内容：输入用户名和密码，验证注册、登录功能

操作：

填写用户名和密码，进行注册，若是成功

输入正确的用户名和密码

输入错误的用户名和密码

结果：

登录成功

提示用户名或者密码错误

登录页面：

图 41 用户登录页面

4.3.2 文章测试

测试内容：对文章进行增删改查、以及评论、回复操作

操作：

增加：进入系统首页，点击发表文章按钮进入文章编辑页面，填写完相关内容后点击发布文章，输入标签后点击确定按钮

删除：进入个人中心页面，找到文章点击删除按钮

修改： 点击文章，进入页面详情页，对文章内容进行修改后点击发表文章按钮

查看：点击文章，查看文章详细内容

评论：在文章详情页底部，输入评论内容，点击评论

回复：点击用户评论部分的回复按钮，输入回复内容进行回复

首页：显示所有文章

图 42 首页

专栏： 显示不同类别的文章

图 43 专栏

文章详情：查看文章详细内容以及评论回复

图 44文章详情页

图 45 评论

个人中心： 显示用户文章和提出的问题及用户收藏的文章

图 46个人中心

结果： 测试成功

4.3.3 提问测试

测试内容：提问、回复、删除问题

操作：进入提问页面，输入问题标题和描述，点击按钮发表问题；点击写回答，进入问题详情页，进行编辑回答，填写完成点击提交按钮。在个人中心页面，点击删除按钮删除文章。

提问：提出问题、显示问题

图 47 提问

问题详情页： 查看问题详细信息、写回答、查看他人回答

图 48问题详情及写回答

图 49 回答列表

图 410 问题列表

结果： 测试成功

4.3.4 权限测试

测试内容：用户不登录，只能进行查看功能

操作：在用户不登录的条件下，分别进行发表文章、评论、提问、答复功能

测试结果：请先登录

权限验证功能代码：

获取请求头中的token值，并使用jsonwebtoken验证token，来判断用户是否已经登录

```
const Authorization = req.get('userToken')
if (Authorization !== undefined && Authorization != null && Authorization !== '') {
  const token = Authorization
  // 验证 token
  const vaild = await isVaildToken(dbo, token)
  if (!vaild) {
    res.send({
      code: 1,
      msg: 'token 失效，请重新登陆'
    })
  } else {
    res.send({
      code: 0,
      msg: ''
    })
    db.close()
  }
```

```
async function isVaildToken (dbo, token) {
  let result
  try {
    result = jwt.verify(token, key)
  } catch (e) {
    console.log(e)
    return false
  }

  const { exp } = result
  const current = Math.floor(Date.now() / 1000)
  if (current > exp) {
    return false
  }

  const res = await dbo.collection('user').findOne({ token })
  if (res) {
    return true
  }

  return false
}
```

图 411 验证代码

图 412 验证

总结

大学生知识分享问答平台主要实现了对文章的发表、删除、修改、查看、评论、回复和对问题的发表、回答、删除、查看功能。完成了一个可以供大学生记录、交流、分享、解决问题的平台。

以知乎为例，对平台的开发进行探讨和实践。主要针对数据双向绑定的响应式框架MVVM三层开发模式的特点分析。并在此基础上对项目进行开发。在发阶段的技术选择上，我并没有按以前系统设计的经验来，而是选择使用市场上比较前沿的技术开发。系统前端开发使用Vue，它是一套用于构建用户界面的渐进式JavaScript框架，它出色的性能和组件化的设计让我选择使用它来开发系统前端，为了使得系统变得更加轻量，具有更好的性能[1]。在数据库选择上使用了mongdb，它是一款出色的缓存数据库，同时支持数据的持久化。后台的服务搭建上使用node.js作为技术支持，它能让 JavaScript 运行在服务端的开发平台，同时提供对各种底层组件的支持，比如操作数据库等。以此来完成系统功能，保证系统稳定性。

在这次的完成这次设计的过程中，尽管在测试过程中还存在一些未解决的问题，但是还是能够解决的。在项目开发的过程中，更加认识到对于学习过的知识应该加以实践，并做到熟练掌握和运用。并对vue，node，MongoDB进行了学习和掌握，更加发现了自己知识方面的欠缺，在改正的同时加以整理。更加的完善了自己在专业方面的知识积累

参考文献

[1] 梁灏. Vue.js实战[M]. 清华大学出版社,2017.
[2] Vue.js. https://cn.vuejs.org/
[3] 菜鸟教程. https://www.runoob.com/
[4] 弗拉纳根. JavaScript权威指南[M]. 机械工业出版社,2007.
[5] 陆凌牛. Node.js 权威指南[M]. 北京机械工业出版社，2014.4.
[6] GItHu开源. node.js最佳实践.
[7] 沈剑翘,陈泽椿.Vue.js在构建系统前端SPA的应用[J].科技创新与应用,2020(03):181-182.
[8] 朱晓阳,刘苑如,范仲言.基于Node.js的学习平台后端系统设计与实现[J].电脑知识与技术,2019,15(13):116-118.
[9] 任明飞,李学军,崔蒙蒙,杨双龙,孙小奇.基于MongoDB的非关系型数据库的设计与开发[J].电脑知识与技术,2019,15(34):1-2.
[10] 杨锦.关于网页设计与制作方法研究[J].计算机产品与流通,2019(08):210
[11] Robert Hoekman,Jr.，[美] Jared Spool. 网站设计解构[M]. 人民邮电出版社,2010-09.
[12] 闫安. 虚拟社区中知识获取方式对获取结果的影响研究——以知乎社区为例[J]. 图书馆理论与实践. 2020年.(1):G252
[13] 耿瑞利.申静.A Research on Knowledge Sharing of Social Networking Users: Features, Contents, and Prospects%社交网络用户知识共享研究:特征、内容与展望[J]. 图书情报知识, 2018, 000(001):16-26.
[14] 徐浩东. 网络社群虚拟奖励对用户知识共享意愿的影响研究[J]. 中文信息, 2017(9).
[15] 刘佩, 林如鹏. 网络问答社区“知乎”的知识分享与传播行为研究[J]. 图书情报知识, 2015, No.168(06):111-121
[16] 马瑞芳,王会燃. 计算机软件测试方法的研究[J]. 小型微型计算机系统(12):2210-2213.
[17] 朱二华. 基于Vue.js的Web前端应用研究[J]. 科技与创新, 2017(20):119-121.
[18] 宁菁菁. 知乎网用户知识共享研究[D]. 北京邮电大学, 2014
[19] 李广宏.vue.js前端应用技术分析[J].中国新通信,2019,21(20):115.
[20] Vue.js前端开发快速入门与专业应用[M]. 人民邮电出版社，陈陆扬, 2017

致谢

本课题在选题及研究过程中得到陈洁老师的指导。感谢陈洁、侯海良等老师在论文工作的各方面给予我的关心、指导和帮助。老师严谨的治学作风、认真的工作态度和勤奋的工作精神深深地影响着我，使我受益终身。在此我对他们表示由衷的感谢。

感谢在大学期间专业老师们对我的指导，让我学到了很多的专业知识，为此次毕业论文设计奠定了基础，还让我树立了正确的价值观，收货很多，是一辈子的财富，学校和领导们给我们创建的学习环境也使论文在一个很好的学术氛围中完成。除此之外，感谢同学在学习和生活上的帮助。

最后，向审阅我论文和参加答辩的老师们表示我衷心的感谢，感谢你们抽出宝贵的时候参加我的论文答辩会，感谢你们对论文不当之处提出的宝贵意见和建议。

指　标
疑似剽窃文字表述

1. Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。 V8引擎执行Javascript的速度非常快，性能非常好。
2. 考虑软件的内部结构和特性，测试功能的实现。根据软件的需求规格说明书设计测试用例，从程序的输入和输出特性上测试程序是否
3. 用户测试
   测试内容：输入用户名和密码，验证注册、登录功能
   操作：
   填写用户名和密码，进行注册，若是成功
   输入正确的用户名和密码
   输入错误的用户名和密码
4. 工作的各方面给予我的关心、指导和帮助。老师严谨的治学作风、认真的工作态度和勤奋的工作精神深深地影响着我，使我受益终身。在此我对他们表示由衷的感谢。
   感谢在大学期间专业老师们对我的指导，
5. 最后，向审阅我论文和参加答辩的老师们表示我衷心的感谢，感谢你们抽出宝贵的时候参加我的论文答辩会，感谢你们对论文不当之处提出的宝贵意见和建议。

说明：1.总文字复制比：被检测论文总重合字数在总字数中所占的比例

2.去除引用文献复制比：去除系统识别为引用的文献后，计算出来的重合字数在总字数中所占的比例

3.去除本人已发表文献复制比：去除作者本人已发表文献后，计算出来的重合字数在总字数中所占的比例

4.单篇最大文字复制比：被检测文献与所有相似文献比对后，重合字数占总字数的比例最大的那一篇文献的文字复制比

5.指标是由系统根据《学术论文不端行为的界定标准》自动生成的

6.红色文字表示文字复制部分;绿色文字表示引用部分

7.本报告单仅对您所选择比对资源范围内检测结果负责