

浙江大学



Techhub 后端知识库搜索引擎系统 测试报告书

Group 08

黄亦非、傅诤哲、胡瀚丹、陈鑫

2019/07/17

目录

| | |
|-----------------------------|----|
| 1. 引言 | 4 |
| 1.1 编写目的 | 4 |
| 1.2 背景 | 4 |
| 1.3 定义 | 5 |
| 1.4 参考资料 | 5 |
| 2. 测试概要 | 7 |
| 2.1 软件说明 | 7 |
| 2.2 测试内容 | 7 |
| 2.3 测试一：模块测试 | 7 |
| 2.3.1 进度安排 | 7 |
| 2.3.2 条件 | 8 |
| 2.3.3 测试资料 | 8 |
| 2.4 测试二：边界值 | 9 |
| 2.4.1 进度安排 | 9 |
| 2.4.2 条件 | 10 |
| 2.4.3 测试资料 | 10 |
| 2.5 测试三：压力测试 | 11 |
| 2.5.1 进度安排 | 11 |
| 2.5.2 测试条件 | 11 |
| 2.5.3 说明 | 11 |
| 3. 模块功能测试 | 12 |
| 3.1 爬虫系统及数据衍生系统测试 | 12 |
| 3.1.1 爬虫测试 | 12 |
| 3.1.2 数据处理测试 | 14 |
| 3.2 前端模块测试 | 18 |
| 3.2.1 控制 | 18 |
| 3.2.2 基本数据 | 19 |
| 3.2.3 输入和预期输出 | 19 |
| 3.2.4 测试结果 | 20 |
| 3.2.5 测试结果分析 | 20 |
| 3.3 后端 WEB 项目测试 | 21 |
| 3.3.1 /search 接口测试 | 21 |
| 3.3.2 /getAllTag 接口测试 | 25 |
| 4. 边界值测试 | 27 |
| 4.1 控制 | 27 |
| 4.2 输入和预期输出 | 27 |
| 4.3 测试结果 | 27 |
| 5. 压力测试 | 29 |

| | |
|-----------------------|-----------|
| 5.1 测试简介 | 29 |
| 5.2 控制..... | 29 |
| 5.3 输入..... | 29 |
| 5.4 测试结果 | 30 |
| 5.5 测试结果与分析..... | 31 |
| 6. 分析摘要 | 31 |
| 6.1 能力 | 31 |
| 6.2 缺陷和限制 | 31 |
| 7. 测试资源消耗..... | 32 |

1. 引言

1.1 编写目的

本测试分析报告文档编写的目的在于测试我们编写的软件工程课程网站系统，检测在我们的代码中是否还存在错误。本测试的分析有助于完善我们的软件工程课程网站系统，提供我们对于我们的系统的一些测试样例，保证系统的正确性，更好的服务于用户。

本文档主要面向的读者包括：网站系统开发人员、系统项目管理人员、系统测试人员等。

1.2 背景

Techhub 后端知识库搜索引擎系统主要是面向在校大学生用户，致力于在互联网上学习后端的相关技术，能够帮助用户快速的搜索相关领域的技术文档，包括该技术的描述、使用手册、教学视频、使用当中的相关问题等方面。对于用户而言，可以根据输入的问题或者关键字，准确的定位用户的意图，并且返回最准确的结果给用户。同时用户也可以根据提供的几个过滤条件对返回的结果进行对应的过滤，实现更加精确的结果定位。

同时系统对于数据的来源和处理也非常的重视，致力于提供最准确、最完整、最精确的知识库架构。对于数据我们会进行相关的过滤、去重、结构化信息的提取等等数据处理的动作，保证系统数据的稳定和准确。

1.3 定义

- MongoDB:系统服务器所使用的非关系型数据库（DBMS）。
- React: 一种 JavaScript MVC 框架。
- Material-UI: 一个实现了 Google's Material Design 设计规范的 react 组件库。
- HTML(HyperText Markup Language): 超文本标记语言.
- UML (Unified Modeling Language): 统一建模语言、是一套用来设计软件蓝图的标准建模语言, 是一种从软件分析、设计到编写程序规范的标准建模语言。
- Spring-Boot: 基于 Java 的微服务框架
- Solr: 分布式搜索引擎框架
- scrapy: 基于 python 语言的分布式爬虫框架

1.4 参考资料

《软件设计文档国家标准》

《软件工程项目开发文档范例》

《Software Requirements edition2》 Karl E. Wiegers

《软件需求》 刘伟琴、刘洪涛 译

2. 测试概要

2.1 软件说明

本软件为 TechHub 后端知识库搜索引擎软件系统。根据需实现的各项功能，我们将该软件分为三个子系统，其中包括前端子系统、后端 web 与搜索引擎子系统、爬虫与数据处理子系统。

2.2 测试内容

| 测试名称 | 目的 | 内容 | 与实际差别 |
|------------|---------------------------|------------------------------|-------|
| 模块功能测试 | 检测各个模块是否符合实际的要求 | 针对各个模块，根据《需求说明书》里的要求进行测试 | 未做改变 |
| 边界测试 | 检测系统能否正常的处理边界值 | 在存在边界值的情况下测试边界值输入下系统能否正常工作 | 未做改变 |
| 压力测试 | 检测系统的承受能力 | 对系统导入大量的数据，观察系统的各个模块是否正常工作 | 未做改变 |
| 与其他模块的接口测试 | 测试与其他模块是否能够对接，能否实现最后的集成测试 | 运行系统，与上下游系统对接，观察数据库内数据是否正常变更 | 未做改变 |

2.3 测试一：模块测试

2.3.1 进度安排

| 阶段 | 内容 | 时间 |
|------------|----------------------|------|
| 第一阶段（预备阶段） | 测试人员阅读本子系统的需求说明书，了解每 | 7.16 |

| | | |
|------------|---|-------|
| | 一个模块需要实现的功能 | |
| 第二阶段（准备阶段） | 根据各个模块的功能，编写测试用例，准备好测试时使用的数据 | 7. 16 |
| 第三阶段（测试阶段） | 测试人员根据已经开发好的系统，用测试样例对各个模块进行测试，找出系统中仍存在的错误 | 7. 17 |
| 第四阶段（后期阶段） | 开发人员根据测试人员发现的错误修改代码，修复存在的错误 | 7. 18 |


2.3.2 条件

设备：已联网的笔记本电脑

软件：MongoDB

2.3.3 测试资料

| 功能 | 输入数据 | 显示结果 |
|--------|-------|---------------------|
| 数据获取 | 无 | 从网站爬取信息插入到目标数据库当中 |
| 数据去重 | 无 | 去除重复或者相似度接近 100%的文档 |
| 数据摘要生成 | 无 | 从抓取的文档当中生成摘要并且更新数据库 |
| 输入搜索内容 | 搜索关键字 | 显示搜索结果列表（第一页） |

| | | |
|--|------------|--------------|
| 点击搜索结果标题 | 按钮点击 | 打开源网址 |
| 点击分页页码 | 鼠标点击 | 显示对应页数的搜索结果 |
| 选择资源类别 | 鼠标点击 | 显示该类的搜索结果 |
| 选择发布时间 | 鼠标点击相对应的时间 | 显示筛选时间内的搜索结果 |
| 点击菜单栏-标签 | 鼠标点击 | 显示所有标签名称与数量 |
| 点击单个标签  | 鼠标点击 | 显示该标签的所有匹配结果 |

2.4 测试二：边界值

2.4.1 进度安排

| 阶段 | 内容 | 时间 |
|------------|---|-------|
| 第一阶段（预备阶段） | 测试人员阅读本子系统的需求说明书，了解每一个模块需要实现的功能 | 7. 16 |
| 第二阶段（准备阶段） | 根据各个模块的功能，编写测试用例，准备好测试时使用的数据 | 7. 16 |
| 第三阶段（测试阶段） | 测试人员根据已经开发好的系统，用测试样例对各个模块进行测试，找出系统中仍存在的错误 | 7. 17 |
| 第四阶段（后期阶段） | 开发人员根据测试人员发现的错误修改代码，修复存在的错误 | 7. 18 |

2.4.2 条件

设备：已联网的笔记本电脑

软件：MongoDB

2.4.3 测试资料

本系统的边界值测试主要是导入过程中的关键字输入等的边界值测试。这些填写的信息涵盖范围大（有个类型的值），对于这些数据也需要进行一些合法性检测（大小、长度等），因此存在着一个特殊情况的测试情况。对于处在边界的输入数据，系统能否做出正确判断，回答这个问题正是我们进行边界值测试的原因。

本测试采用的测试方法主要是人工输入一些边界值数据，然后观察系统能否出正确的判断和操作。

2.5 测试三：压力测试

2.5.1 进度安排

| 阶段 | 内容 | 时间 |
|------------|--|------|
| 第一阶段（预备阶段） | 查阅压力测试相关资料，找到相关测试工具 | 7.16 |
| 第二阶段（准备阶段） | 学习该测试工具的使用方法，根据本系统子模块，确定需要做压力测试的模块并准备测试数据。 | 7.16 |
| 第三阶段（测试阶段） | 测试人员根据已经开发好的系统，进行压力测试，并对结果进行分析 | 7.17 |
| 第四阶段（后期阶段） | 开发人员根据测试人员发现的错误修改代码，修复存在的错误 | 7.18 |

2.5.2 测试条件

设备：已联网的笔记本电脑

软件：MongoDB

2.5.3 说明

本系统中需要做压力测试的模块主要是用户信息导入和开课信息导入部分，以及上传下载资料的时间。

3. 模块功能测试

3.1 爬虫系统及数据衍生系统测试

3.1.1 爬虫测试

3.1.1.1 控制

控制操作顺序为：

1. 预先安装必要的依赖包

2. 以项目管理员身份执行爬虫命令

3.1.1.2 基本数据

```
scrapy crawl xxx
```

3.1.1.3 输入和预期输出

| 序号 | 测试用例 ID | 输入 | 预期输出 |
|----|---------|---------------------|--|
| 1 | FN_1 | scrapy crawl xxx | Mongodb 数据库指定 collection 新增爬取内 容 |

3.1.1.4 测试结果

| Scenario | Test Case No. | Test Case Description | Test Step | Test Data | 预 期 输 出 | Actual Result |
|----------|------------------|--------------------------|--------------|--------------|------------------|---------------|
| | 1 | | 1. 预 先安装必 | 指定 的网站如 | Mongodb 数据库指定 | pass |

| | | | | | | |
|--------------|--|--------------|-----------------------------------|------------------------------|--------------------------|--|
| 验证爬虫 爬取能力 | | 验证爬虫 爬取能力 | 要的依赖包 | GitHub, CSDN, Coursera | collection 新增爬取内 容 | |
| | | | 2. 以 项目管理 员身份执 行爬虫命 令 | | | |

3.1.1.5 测试结果分析

数据库结果符合预期，由此推断这个功能已经基本实现

3.1.2 数据处理测试

3.1.2.1 控制

控制操作顺序为：

1. 预先安装必要的依赖包
2. 以项目管理员身份执行去重命令

3. 重复数据处理命令

3.1.2.2 基本数据

数据库中的内容:

```
1 {  
2   "_id" : ObjectId("5d2be40fc357e109b25caaf9"),  
3   "author" : "",  
4   "catalog" : 1.0,  
5   "content" : "A query language for your API GraphQL is a query language for APIs at  
6   "date" : "",  
7   "source" : "官网",  
8   "summary" : "A query language for your API GraphQL is a query language for APIs at  
9   "tags" : [  
10    "GraphQL"  
11  ],  
12  "title" : "GraphQL",  
13  "url" : "https://graphql.org/"  
14 }
```

由_id, author, catalog, content, date, source, summary, tags, title, url 几个部分组成。

3.1.2.3 输入和预期输出

| 序号 | 测试用例 ID | 输入 | 预期输出 |
|----|------------|--------|------------------|
| 2 | FN_2 | 脚本执行语句 | 第一遍有提示重复第 二遍无 |

| | | | |
|---|------|--------|---------------------|
| 3 | FN_3 | 脚本执行语句 | 误差小于一定阈值 (如 10%) |
|---|------|--------|---------------------|

3.1.2.4 测试结果

| Scenario | Test Case No. | Test Case Description | Test Step | Test Data | 预 期输出 | Actual Result |
|----------|---------------------|--------------------------|--------------|--------------|----------|---------------|
|----------|---------------------|--------------------------|--------------|--------------|----------|---------------|

| | | | | | | |
|----------|---|----------|---------------------|---------------------------------|-----------------|------|
| 验证信息去重能力 | 2 | 验证信息去重能力 | 1. 预先安装必要的依赖包 | 已存在 collection 的博客等技术资料等信息 | 第一遍有提示重复第二遍无 | pass |
| | | | 2. 以项目管理员身份执行去重命令 | | | |
| | | | 3. 重复去重命令 | | | |
| 验证信息提取能力 | 3 | 验证信息提取能力 | 1. 预先安装必要的依赖包 | 已存在 collection 并执行去重的博客等技术资料等信息 | 误差小于一定阈值（如 10%） | pass |
| | | | 2. 以项目管理员身份执行信息提取命令 | | | |

| | | | | | | |
|--|--|--|---|--|--|--|
| | | | 3. 重 复信息提 取过程， 但只与数 据库中信 息最比 对，记录 误差 | | | |
|--|--|--|---|--|--|--|

3.1.2.5 测试结果分析

数据处理均符合预期测试结果，由此推断这个功能已经基本实现

3.2 前端模块测试

3.2.1 控制

控制操作顺序为：

1. 进入搜索引擎主页

2. 在搜索框输入搜索内容，敲击回车或点击搜索按钮，等待网页跳转至搜索结果页面
3. 进入搜索结果页面，等待显示全部资源的内容列表
4. 点击某条搜索结果的标题或外链，等待打开源网址
5. 拉至页面底部，点击分页页数，等待显示某页搜索结果
6. 点击左侧筛选分类条目（比如技术博客），等待搜索结果更新
7. 选择右侧筛选发布时间（比如最近一个月），等待搜索结果更新
8. 点击左侧菜单栏-标签，等待显示所有标签与数量
9. 点击标签，等待显示该标签的内容列表

3.2.2 基本数据

不为空、长度不超过 20 的字符串，后端技术知识相关内容（比如 Spring Boot、Docker）

3.2.3 输入和预期输出

| 序号 | 输入 | 预期输出 |
|----|----------|---------------|
| 1 | 输入搜索内容 | 显示搜索结果列表（第一页） |
| 2 | 点击搜索结果标题 | 打开源网址 |
| 3 | 点击分页页码 | 显示对应页数的搜索结果 |

| | | |
|---|----------|--------------|
| 4 | 选择资源类别 | 显示该类的搜索结果 |
| 5 | 选择发布时间 | 显示筛选时间内的搜索结果 |
| 6 | 点击菜单栏-标签 | 显示所有标签名称与数量 |
| 7 | 点击某个标签 | 显示该标签的所有匹配结果 |

3.2.4 测试结果

| Scenario | Test Case No. | Test Case Description | Test Step | Test Data | Expected Result | Actual Result |
|----------|---------------|-----------------------|-------------------------------|---------------------|--------------------------------|---------------|
| 搜索内容 | 1 | 普通搜索 | 进入主页后，在搜索框中输入搜索内容，敲击回车或点击搜索按钮 | 搜索内容：Docker | 跳转至搜索结果页面，显示Docker相关的搜索结果（第一页） | Pass |
| | 2 | 搜索结果分页 | 显示搜索结果后，拉至页面底部，点击分页页码 | 搜索内容：Docker，页数：2 | 显示Docker相关搜索结果的第二页内容 | Pass |
| | 3 | 筛选资源类别 | 已输入搜索内容，点击左侧资源分类条目 | 搜索内容：Docker，分类：技术博客 | 显示Docker技术博客的搜索结果 | Pass |
| | 4 | 筛选发布时间 | 已输入搜索内容，选择发布时间 | 搜索内容：Docker，时间：一个月内 | 显示在一个月内发布的Docker相关搜索结果 | Pass |
| | 5 | 显示所有标签 | 点击左侧菜单栏-标签 | 标签名称与数量 | 显示所有标签名称与每个标签的资源数量（第一页） | Pass |
| | 6 | 显示标签 | 在所有标签列表，或搜索结果的卡片中，点击某个标签 | 标签：Spring Boot | 显示所有标签为Spring Boot的资源（第一页） | Pass |
| | 7 | 显示词云 | 进入搜索结果页面 | 标签名称与数量 | 在页面右侧栏中显示标签词云 | Pass |

3.2.5 测试结果分析

前端搜索内容的结果均符合预期，由此推断该功能已经基本实现。

3.3 后端 Web 项目测试

3.3.1 /search 接口测试

3.3.1.1 控制

控制操作顺序为：

1. 打开 Postman
2. 输入/search 接口的 URL
3. 点击 send，等到返回包

3.3.1.2 基本数据

/search 接口直接从 Solr 服务器获取数据，以下为 Solr 的搜索示意图。可以看到一共有 56513 条数据，每一条的数据的格式如下。

```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "_":"1563257482816"}},
  "response":{"numFound":56513,"start":0,"docs":[
    {
      "summary":"Now, this doesn't work. If I request /calc/a the method Sampler.isSampled(Span) is being called before the
      "date":"2016-11-10",
      "catalog":5,
      "author":"Danny",
      "_id":"5d26b8f2acc78b4a2fc186d5",
      "source":"StackOverflow",
      "title":"Spring Sleuth - Tracing Failures",
      "url":"https://stackoverflow.com/questions/40525453/spring-sleuth-tracing-failures/40534423?r=SearchResults#40534423"
      "content":"In a microservice environment I see two main benefits from tracing requests through all microservice insta
      "tags":["spring",
        "spring-cloud-sleuth"],
      "_version_":1639096884178976768},
```

3.3.1.3 输入和预期输出

| 序号 | 测试用例ID | 输入 | 预期输出 |
|----|--------|---|-------------------------------|
| 1 | FN_1 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=-1 | 返回跟Docker相关的前10条数据 |
| 2 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=0 | 返回跟Docker相关的前10条数据 类别为使用手册 |
| 3 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=1 | 返回跟Docker相关的前10条数据 类别为源码分析 |
| 4 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=2 | 返回跟Docker相关的前10条数据 类别为Demo |
| 5 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=3 | 返回跟Docker相关的前10条数据 类别为使用手册 |
| 6 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=4 | 返回跟Docker相关的前10条数据 类别为教学视频 |
| 7 | FN_2 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=0&catalog=5 | 返回跟Docker相关的前10条数据 类别为相关问题 |
| 8 | FN_3 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=1&catalog=-1 | 返回跟Docker相关的前10条数据 时间为1天内 |
| 9 | FN_3 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=2&catalog=-1 | 返回跟Docker相关的前10条数据 时间为1周内 |
| 10 | FN_3 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=3&catalog=-1 | 返回跟Docker相关的前10条数据 时间为1月内 |
| 11 | FN_3 | http://10.214.213.43:9999/search?key=Docker&page=1&size=10&delta=4&catalog=-2 | 返回跟Docker相关的前10条数据 时间为1年内 |
| 12 | FN_4 | http://10.214.213.43:9999/search?key=Docker&page=1&size=5&delta=0&catalog=-1 | 返回跟Docker相关的前5条数据 |
| 13 | FN_4 | http://10.214.213.43:9999/search?key=Docker&page=2&size=10&delta=0&catalog=-1 | 返回跟Docker相关的第二页数据 每页10条 |

3.3.1.4 测试结果

| Scenario | Test Case No. | Test Case Description | Test Step | Test Data | Expected Result | Actual Result |
|-------------|---------------|----------------------------|---|---|---------------------------|---------------|
| 测试/search接口 | 1 | 根据key进行搜索，不选择catalog和delta | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: -1 | 返回跟Docker相关的前10条数据 | Pass |
| | 2 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 0 | 返回跟Docker相关的前10条数据类别为使用手册 | Pass |
| | 3 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 1 | 返回跟Docker相关的前10条数据类别为源码分析 | Pass |
| | 4 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 2 | 返回跟Docker相关的前10条数据类别为Demo | Pass |
| | 5 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 3 | 返回跟Docker相关的前10条数据类别为使用手册 | Pass |
| | 6 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 4 | 返回跟Docker相关的前10条数据类别为教学视频 | Pass |
| | 7 | 根据key进行搜索，用catalog进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: 5 | 返回跟Docker相关的前10条数据类别为相关问题 | Pass |
| | 8 | 根据key进行搜索，用delta进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 1 catalog: -1 | 返回跟Docker相关的前10条数据时间为1天内 | Pass |
| | 9 | 根据key进行搜索，用delta进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 2 catalog: -1 | 返回跟Docker相关的前10条数据时间为1周内 | Pass |
| | 10 | 根据key进行搜索，用delta进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 3 catalog: -1 | 返回跟Docker相关的前10条数据时间为1月内 | Pass |
| | 11 | 根据key进行搜索，用delta进行筛选 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 4 catalog: -1 | 返回跟Docker相关的前10条数据时间为1年内 | Pass |
| | 12 | 根据key进行搜索，分页 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: -1 | 返回跟Docker相关的前5条数据 | Pass |
| | 13 | 根据key进行搜索，分页 | 1. 打开Postman 2. 输入/search接口的URL 3. 点击send，等到返回包 | key: Docker page: 1 size: 10 delta: 0 catalog: -1 | 返回跟Docker相关的第二页数据每页10条 | Pass |

3.3.1.5 测试结果分析

/search 接口均符合预期测试结果，由此推断这个功能已经基本实现

3.3.2 /getAllTag 接口测试


3.3.2.1 控制



控制操作顺序为：

1. 打开 Postman
2. 输入/getAllTag 接口的 URL
3. 点击 send，等到返回包

3.3.2.2 基本数据

如下图，可以看到存储 tag 的 MySQL 表格 infos 一共有 155606 条数据。infos 表格的结构比较简单，由一个 tag 和一个_id 组成。tag 为标签的名字，_id 为 MongoDB 中的 document 的_id

| Name | Engine | Version | Row Format | Rows |
|---|--------|---------|------------|--------|
|  infos | InnoDB | 10 | Dynamic | 155606 |

| Table | Column | Type | Default Value | Nullable |
|-------|---|--------------|---------------|----------|
| infos |  tag | varchar(128) | | NO |
| infos |  _id | varchar(128) | | NO |

3.3.2.3 输入和预期输出

| 序号 | 测试用例ID | 输入 | 预期输出 |
|----|--------|---|----------------------|
| 1 | FN_5 | http://10.214.213.43:9999/getAllTag?page=1&size=10&key= | 返回数量前10的标签 |
| 2 | FN_6 | http://10.214.213.43:9999/getAllTag?page=1&size=10&key=Docker | 返回标签为Docker的数量前10的记录 |
| 3 | FN_7 | http://10.214.213.43:9999/getAllTag?page=1&size=5&key= | 返回数量前5的标签 |
| 3 | FN_7 | http://10.214.213.43:9999/getAllTag?page=2&size=10&key= | 返回跟第二页数据，每页10条 |

3.3.2.4 测试结果

| Scenario | Test Case No. | Test Case Description | Test Step | Test Data | Expected Result | Actual Result |
|----------------|---------------|-----------------------|--|---------------------------------|----------------------|---------------|
| 测试/getAllTag接口 | 1 | 得到所有tag信息 | 1. 打开Postman 2. 输入/getAllTag接口的URL 3. 点击send，等到返回包 | key: page:1 size:10 | 返回数量前10的标签 | Pass |
| | 2 | 得到tag信息 | 1. 打开Postman 2. 输入/getAllTag接口的URL 3. 点击send，等到返回包 | key:Docker page:1 size:10 | 返回标签为Docker的数量前10的记录 | Pass |
| | 3 | 得到tag信息，分页 | 1. 打开Postman 2. 输入/getAllTag接口的URL 3. 点击send，等到返回包 | key: page:1 size:5 | 返回数量前5的标签 | Pass |
| | 4 | 得到tag信息，分页 | 1. 打开Postman 2. 输入/getAllTag接口的URL 3. 点击send，等到返回包 | key: page:2 size:10 | 返回跟第二页数据，每页10条 | Pass |

3.3.2.5 测试结果分析

/getAllTag 接口均符合预期测试结果，由此推断这个功能已经基本实现。

4. 边界值测试

4.1 控制

本模块的测试步骤控制方式是人工输入信息、文件加载信息。

控制操作信息为：

- (1) 在需要填写数据框当中填入需要填入的数据边界值，并且予以确认。
- (2) 等待系统反应。

4.2 输入和预期输出

| 序号 | 输入 | 预期输出 |
|----|-------------|----------------------|
| 1 | 用户输入搜索关键字为空 | 弹出警告⚠️，输入不能为空 |
| 2 | 用户输入搜索关键字过长 | 弹出警告⚠️，输入过长 |
| 3 | 用户输入非法字符 | 弹出警告⚠️，输入的字符当中存在非法字符 |

4.3 测试结果

| 序号 | 输入 | 预期输出 | 实际结果 |
|----|----|------|------|
|----|----|------|------|

| | | | |
|---|-------------|----------------------|-----------------------|
| 1 | 用户输入搜索关键字为空 | 弹出警告⚠️，输入不能为空 | As expected passed |
| 2 | 用户输入搜索关键字过长 | 弹出警告⚠️，输入过长 | As expected passed |
| 3 | 用户输入非法字符 | 弹出警告⚠️，输入的字符当中存在非法字符 | As expected passed |

5. 压力测试

5.1 测试简介

本次压力测试的目的是为了测试该 TechHub 后端知识库搜索引擎软件系统的承载能力。测试内容包括测试在上传文件包含大量数据时系统的负载能力、响应时间，以及数据处理能力。

本次压力测试设计的部分主要是测试在大规模用户同时发送请求时的最大负载能力、相应时间等性能。

5.2 控制

本次压力测试的输入控制方式为自动输入控制，利用的是 siege 工具进行负载压力测试。通过设置相关参数之后，使用负载测试来聚合测试。并且使用虚拟用户在模拟负载当中同时运行他们。

5.3 输入

本次压力测试所选择的工具主要是 siege，使用命令行参数对指定的 URL 和接口进行相关的压力测试，可以同时测试 GET 方法或者是 POST 方法，例如：`siege -c 200 -r 10 http://localhost:3000`

表示的是对前端的主页面进行并发量为 200，重复数量为 10 的压力测试，可以得到如下的结果：

```

Transactions:      2000 hits
Availability:      100.00 %
Elapsed time:      41.07 secs
Data transferred: 610.87 MB
Response time:     2.04 secs
Transaction rate:  48.70 trans/sec
Throughput:        14.87 MB/sec
Concurrency:       99.33
Successful transactions: 2000
Failed transactions: 0
Longest transaction: 8.11
Shortest transaction: 0.02

```

5.4 测试结果

| 编号 | 模块 | Availability | Response time | Transaction rate | Thoughtput | Concurrency |
|----|-----------|--------------|---------------|------------------|------------|-------------|
| | 单位 | % | Sec | Trans/sec | MB/src | 无 |
| 1 | 网站主页 | 100.00 | 2.04 | 48.70 | 14.87 | 99.33 |
| 2 | 搜索接口 | 100.00 | 0.19 | 52.77 | 16.12 | 99.4 |
| 3 | Tags 显示界面 | 100.00 | 1.87 | 53.01 | 16.19 | 99.29 |
| 4 | 搜索接口 | 100.00 | 10.46 | 0.80 | 0.01 | 88.37 |
| 5 | 标签接口测试 | 100.00 | 5.08 | 1.35 | 0.08 | 66.87 |

5.5 测试结果与分析

测试结果如上，总共对一些主要的页面和后端提供的接口进行了压力测试。用户负载在 90 左右，系统当中查询结果并且返回几乎占据了全部的相应时间，原因可能是与网络有关，可能也与从数据库中查询数据的信息较多有关。总体而言能够承载一定数量用户的同时使用。

6. 分析摘要

6.1 能力

经过模块功能测试、界面测试、压力测试、接口测试等测试工作之后，本次项目团队所开发的网站能够实现所有的功能，能够满足网站在不同的特殊请求下的网站请求，且能够合理的处理各种异常出现的情况，符合现实当中能够正常运行的网站。

6.2 缺陷和限制

本网站对并发访问处理能力较弱，功能实现较简单，没有考虑一些复杂处理情况。

7. 测试资源消耗

测试由本小组在 3 台 PC 上历时两天共同完成。模块测试、边界测试、压力测试、接口测试实现了上千条数据的处理。