北工大教材资源整合平台TextbookGenius

测试计划与分析报告

团队编号：1

团队成员：张嘉茵、王林诺、彭叶、

李锶、李懿璇、张瑞涵

完成日期：

目录

[1．引言 3](#_Toc43488912)

[1.1编写目的 3](#_Toc43488913)

[1.2项目背景 3](#_Toc43488914)

[1.3定义 3](#_Toc43488915)

[1.4参考资料 3](#_Toc43488916)

[2．测试计划 3](#_Toc43488917)

[2.1测试条件 3](#_Toc43488918)

[2.2测试方案设计 3](#_Toc43488919)

[3．测试执行情况 4](#_Toc43488920)

[3.1 软硬件环境 4](#_Toc43488921)

[3.2 测试结果 4](#_Toc43488922)

[4．评价 4](#_Toc43488923)

[4.1范围 4](#_Toc43488924)

[4.2准则 4](#_Toc43488925)

[4.3软件能力 4](#_Toc43488926)

[4.4缺陷和限制 4](#_Toc43488927)

[4.5建议 4](#_Toc43488928)

# 1．引言

## 1.1编写目的

【阐明编写测试计划的目的，指明读者对象。】

确保系统经过全面的测试和验证，以满足预期的质量标准和用户需求。

主要目的包括：

1. 明确测试范围和目标：定义需要测试的功能、特性和系统组件，以及测试的具体目标和预期结果。
2. 规划测试活动和资源：确定测试所需的时间、人力资源、工具和设备，以有效地执行测试活动。
3. 制定测试策略和方法：确定测试的方法论、技术和流程，确保测试全面、系统化、可重复。
4. 评估风险和制定风险管理策略：识别和分析可能的风险，制定相应的风险缓解和应对策略，确保在项目各阶段中控制和管理风险。
5. 验证遵从性：确保测试活动符合相关的标准、规定和行业最佳实践。

测试计划的读者对象包括开发团队、以及其他相关利益相关者。

## 1.2项目背景

【说明项目的来源、委托单位及主管部门。】

本组成员在过去的学期中经常出现以下问题：购买教材时因教材使用情况无处得知，购买后教材在课程中完全不被使用、在第三方网站购买教材远比在学校购买便宜，因此带来经济上的损失。通过调查问卷可知，这是同学们共同的疑惑与困扰，参与本次调查文件共29人，全部参与者出现购买教材后不使用的情况。



正逢软件工程课设，我们针对教材使用情况无处得知的痛点，考虑到老师或许存在利益相关问题，因此准备开发一个基于北工大学生对教材使用情况进行评价的资源整合平台。

考虑信息时代web开发的发展与繁荣、web开发涉及技术的广泛应用，本组成员认为进行web开发可以将学习到的知识最大化，因此选择web应用作为开发目标，学习并使用当前流行的技术进行开发。

系统将调用豆瓣、淘宝等系统的API，获得教材基本信息及价格等信息，并通过评论区收集用户（北工大学生）对教材的评价。

## 1.3定义

【列出测试计划中所用到的专门术语的定义和缩写词的原意。】

1. API, Application Programming Interface，即应用程序编程接口。
2. React, 是一个用于构建用户界面的 Javascript 库，主要用于构建 UI.
3. Django，Django 是一个高级的 Python 网络框架，可以快速开发安全和可维护的网站。由经验丰富的开发者构建，Django 负责处理网站开发中麻烦的部分，因此开发者可以专注于编写应用程序，而无需重新开发。
4. Material UI，是React的一个UI框架，它提供了各种现成的组件，可以在React项目中使用。
5. FR：功能需求（Functional Requirement）
6. C：约束（Constraint）

## 1.4参考资料

【列出有关资料的作者、标题、编号、发表日期、出版单位或资料来源】

Django官方文档：<https://docs.djangoproject.com/zh-hans/4.2/>

Tim的Music controller项目：

<https://github.com/techwithtim/Music-Controller-Web-App-Tutorial/tree/main>

Material UI 官方文档：<https://mui.com/material-ui/getting-started/>

# 2．测试计划

## 2.1测试条件

【给出测试对资源的特殊要求，如设备、软件、人员等】

设备要求： windows系统电脑

软件：

前端测试：

Jest和React Testing Library：

单元测试、集成测试

Selenium/Cypress：

端到端的自动化测试

后端测试：

Django自带的测试框架unittest：

单元测试、集成测试

Selenium/Cypress：

端到端的自动化测试

综合测试

postman测试工具：

接口功能测试、接口参数测试、接口性能测试、安全性测试、集成测试、监控测试

人员：李懿璇 王林诺

## 2.2测试方案设计

【说明所使用的测试方法，并系统地生成相应的测试用例；测试用例应包含输入数据、输入操作、预期输出数据以及测试负责人等】

**2.2.1教材管理模块**

1. **CreateBookPage 测试**
2. **功能测试**

测试目标： 确保基本的创建书籍功能和ISBN搜索功能能够正常工作。

测试步骤和预期结果：

1. **ISBN搜索功能测试：**
   1. 有效ISBN号码

输入数据： 9787506419880

操作： 发送GET请求至 /api/get-douban-book?isbn=9787506419880

预期输出数据： 返回的JSON数据包含书籍信息：标题为《围城》，作者为茅盾，出版社为上海译文出版社，封面图像URL等。

* 1. 无效ISBN号码

输入数据： 1234567890 (不存在的ISBN号码)

操作： 发送GET请求至 /api/get-douban-book?isbn=1234567890

预期输出数据： 返回的JSON数据为空或错误信息："未找到相关书籍信息"。

* 1. 格式错误的ISBN号码

输入数据: 97875064A9880 (包含非数字字符)

操作: 发送GET请求至 /api/get-douban-book?isbn=97875064A9880

预期输出数据: 返回的JSON数据为空或错误信息："ISBN格式错误"。

* 1. ISBN号码位数不匹配（过短）

输入数据: 97875064 (位数不够)

操作: 发送GET请求至 /api/get-douban-book?isbn=97875064

预期输出数据: 返回的JSON数据为空或错误信息："ISBN格式错误"。

* 1. ISBN号码位数不匹配（过长）

输入数据: 97875064546231564562 (位数过多)

操作: 发送GET请求至 /api/get-douban-book?isbn=97875064

预期输出数据: 返回的JSON数据为空或错误信息："ISBN格式错误"。

* 1. 空ISBN号码

输入数据: 空字符串

操作: 发送GET请求至 /api/get-douban-book?isbn=

预期输出数据: 返回的JSON数据为空或错误信息："ISBN不能为空"。

1. **创建书籍功能测试**
   1. 正确输入
      1. **输入数据：**

ISBN号码：9787536692930

标题：《活着》

作者：余华

出版社：作家出版社

出版日期：2012-08

封面：URL地址

* + 1. 操作： 输入以上信息后，点击创建按钮。
    2. 预期输出数据： 成功创建书籍后，页面导航到新创建书籍的页面 /book/9787536692930。
  1. 错误输入
     1. 输入数据： 仅填写ISBN号码，其他信息留空
     2. 操作： 输入ISBN号码后，点击创建按钮。
     3. 预期输出数据： 页面显示错误信息："请填写完整的书籍信息"。

1. **接口测试**

测试目标： 确保前端与后端API的交互正常，包括数据的发送和接收。

测试步骤和预期结果：

1. **API Get 请求测试：**
   1. 有效ISBN号码
      1. 输入数据： 9787506419880
      2. 操作： 发送GET请求至 /api/get-douban-book?isbn=9787506419880
      3. 预期输出数据： 返回的JSON数据包含书籍信息：标题为《围城》，作者为茅盾，出版社为上海译文出版社，封面图像URL等。
   2. 无效ISBN号码
      1. 输入数据： 1234567890 (不存在的ISBN号码)
      2. 操作： 发送GET请求至 /api/get-douban-book?isbn=1234567890
      3. 预期输出数据： 返回的JSON数据为空或错误信息："未找到相关书籍信息"。
   3. 格式错误的ISBN号码
      * 1. 输入数据: 97875064A9880 (包含非数字字符)
        2. 操作: 发送GET请求至 /api/get-douban-book?isbn=97875064A9880
        3. 预期输出数据: 返回的JSON数据为空或错误信息："ISBN格式错误"。
   4. ISBN号码位数不匹配（过短）
      * 1. 输入数据: 97875064 (位数不够)
        2. 操作: 发送GET请求至 /api/get-douban-book?isbn=97875064
        3. 预期输出数据: 返回的JSON数据为空或错误信息："ISBN格式错误"。
   5. ISBN号码位数不匹配（过长）
2. **输入数据**: 97875064546231564562 (位数过多)
3. **操作**: 发送GET请求至 /api/get-douban-book?isbn=97875064
4. **预期输出数据**: 返回的JSON数据为空或错误信息："ISBN格式错误"。
   1. 空ISBN号码
5. **输入数据: 空字符串**
6. **操作: 发送GET请求至 /api/get-douban-book?isbn=**
7. **预期输出数据: 返回的JSON数据为空或错误信息："ISBN不能为空"。**
8. **API Post 请求测试：**
   1. 完整信息创建：
      1. 输入数据：

{

"isbn": "9787536692930",

"title": "活着",

"author": "余华",

"publisher": "作家出版社",

"pubdate": "2012-08",

"cover": "URL地址",

"douban\_url": "URL地址"

}

* + 1. 操作： 发送POST请求至 /api/create-book，包含以上JSON格式的书籍信息。
    2. 预期输出数据： 返回的JSON数据包含新创建书籍的ISBN号码 "isbn": "9787536692930"，并能正确导航到新书籍的页面。
  1. 缺少必填信息创建：
     1. 输入数据：

{

"isbn": "9787536692930"

}

* + 1. 操作： 发送POST请求至 /api/create-book，仅包含ISBN号码，其他信息为空。
    2. 预期输出数据： 返回的JSON数据包含错误信息："请填写完整的书籍信息"。

1. **UI测试**

测试目标： 确保用户界面的布局和交互符合设计和预期。。

测试步骤和预期结果：

1. **页面布局测试：**
   * 1. 操作： 打开页面，观察页面布局。
     2. 预期结果： 页面元素（输入框、按钮、文本框等）正确布局，无重叠或错位。
2. **错误处理测试**
   * 1. 输入数据： 无效的ISBN号码或缺少必填字段。
     2. 操作： 在输入框中输入无效的ISBN号码或不完整的书籍信息，尝试进行搜索或创 建操作。
     3. 预期结果： 页面能够正确显示错误提示信息，如红色文本或弹出提示框，提醒用 户输入有效信息。
3. **边界条件测试**

测试目标： 测试在极端或边界条件下组件的行为。

测试步骤和预期结果：

1. **空数据处理测试：**
   1. 空ISBN搜索：
      1. 输入数据： 空的ISBN号码或其他输入框
      2. 操作： 在ISBN输入框中不输入任何内容，点击搜索按钮。
      3. 预期输出数据： 页面显示错误信息："请输入有效的ISBN号码"。确保系统能够正确识 别并处理空白输入，不出现异常状态。
   2. 部分信息创建：
      1. 输入数据： 仅填写ISBN号码，其他信息留空。
      2. 操作： 输入ISBN号码后，点击创建按钮。
      3. 预期输出数据： 页面显示错误信息："请填写完整的书籍信息"。确保系统在缺少必要 信息时能够提醒用户填写完整信息。
2. **大数据量测试**
   1. 高并发请求测试：

并发搜索请求：

* + 1. 输入数据： 不同的有效ISBN号码，可以是随机生成的一组数据。
    2. 操作： 使用性能测试工具（如Apache JMeter或Postman的集成）发送多个 并发的ISBN搜索请求。
    3. 预期输出数据： 系统能够在高并发请求下稳定运行，所有请求都能够得到正 确的书籍信息响应，响应时间在可接受范围内。
  1. 大量数据创建测试：

批量创建书籍数据：

* + 1. 输入数据： 一组包含多个完整书籍信息的JSON数据。

[

{

"isbn": "9787536692930",

"title": "活着",

"author": "余华",

"publisher": "作家出版社",

"pubdate": "2012-08",

"cover": "URL地址1",

"douban\_url": ""

},

{

"isbn": "9787506419880",

"title": "围城",

"author": "茅盾",

"publisher": "上海译文出版社",

"pubdate": "2020-01",

"cover": "URL地址2",

"douban\_url": ""

},

...

]

* + 1. 操作： 使用自动化脚本或测试工具，批量发送POST请求至 /api/create-book，包含上述JSON数据。
    2. 预期输出数据： 所有书籍均能成功创建，并返回正确的ISBN号码和导航到对应书籍的页面。

1. **Book 测试**

注：对于 Book 组件，可以省略重复的测试，其功能和 CreateBookPage 有些相似，主要是展示书籍信息。两者的测试策略可以有所重叠，但重点略有不同：

CreateBookPage 的重点在于输入合法ISBN后，能够正确地从后端获取书籍信息并展示，然后创建新书籍的功能。

Book 组件则是在已知的ISBN下，负责展示从后端获取的书籍详细信息。因此，其测试重点更侧重于展示逻辑、接口调用的正确性以及界面的正确渲染。

1. **功能测试**

测试目标：获取书籍详情功能测试：

测试步骤和预期结果：

1. **获取书籍详情功能测试：**
   1. 有效ISBN号码：
      1. 输入数据： 9787506419880
      2. 操作： 访问 /book/9787506419880 页面。
      3. 预期输出数据： 页面显示书籍标题为《围城》。
   2. 无效ISBN号码：
      1. 输入数据： 1234567890 (不存在的ISBN号码)
      2. 操作： 访问 /book/1234567890 页面。
      3. 预期输出数据： 页面显示错误信息或导航回主页。

③空白ISBN号码：

* + 1. 输入数据： 空白或未输入任何内容。
    2. 操作： 访问 /book/ 页面。
    3. 预期输出数据： 页面显示错误信息或导航回主页。

1. **接口测试**

测试目标： 确保前端与后端API的交互正常，包括数据的发送和接收。

测试步骤和预期结果：

1. **API Get 请求测试：**
   1. 有效ISBN号码：
      1. 输入数据： 9787506419880
      2. 操作： 模拟向 /api/get-book?isbn=9787506419880 发送GET请求。
      3. 预期输出数据： 返回的JSON数据包含书籍信息：标题为《围城》。
   2. 无效ISBN号码：
      1. 输入数据： 1234567890 (不存在的ISBN号码)
      2. 操作： 模拟向 /api/get-book?isbn=1234567890 发送GET请求。
      3. 预期输出数据： 返回的JSON数据为空或错误信息。
2. **UI测试**

测试目标： 测试目标： 确保用户界面的布局和交互符合设计和预期。

测试步骤和预期结果：

1. **页面渲染测试：**
   1. 元素展示测试：
      1. 操作： 访问 /book/:isbn 页面。
      2. 预期输出数据： 页面能正确展示书籍的标题和其他相关信息。
   2. 状态变化测试：
      1. 操作： 使用模拟数据调用 getBookDetails 方法。
      2. 预期输出数据： 确保页面能根据返回的书籍信息正确更新标题和其他相 关信息。
2. **边界条件测试**

测试目标： 测试在极端或边界条件下组件的行为。

测试步骤和预期结果：

1. **空数据处理测试：**
   1. 空白ISBN号码处理：
      1. 输入数据： 空白或未输入任何内容。
      2. 操作： 访问 /book/ 页面。
      3. 预期输出数据： 页面显示错误信息或导航回主页。
   2. 无效ISBN号码：
      1. 输输入数据： 1234567890 (不存在的ISBN号码)
      2. 操作： 访问 /book/1234567890 页面。
      3. 预期输出数据： 页面显示错误信息或导航回主页。

**2.2.2用户信息管理模块**

1. **RegisterPage测试**
2. **功能测试**

测试步骤和预期结果：

1. **有效注册测试：**
   * 1. 输入数据：

name: "AliceExample"

password: "Alic3Ex@mple!"

email: "alice@example.com"

pic: "http://example.com/alicepic.jpg"

class: 4

major: "物理学"操作步骤： 用户在注册表单填写以上信息并提交。

* + 1. 预期结果：

页面显示“注册成功！请登录。”

数据库中添加了Alice的信息。

页面自动跳转到登录页面。

1. **缺少必要信息**
   * 1. 输入数据： 缺少 email

id:”1234567890”

name: "TestUser"

password: "TestPassword123!"

email: " "

pic: "http://example.com/userpic.jpg"

class: 3

major: "计算机科学"

* + 1. 操作步骤： 用户尝试提交不完整的注册信息。
    2. 预期结果： 页面显示错误信息“邮箱是必填项”，注册不成功。

1. **无效注册测试：**
   1. 邮箱已存在
      1. 输入数据： 邮箱重合了

id:”1234567890”

name: "TestUser"

password: "TestPassword123!"

email: " 23177977947@qq.com"

pic: "http://example.com/userpic.jpg"

class: 3

major: "计算机科学"

* + 1. 操作步骤： 用户尝试提交已存在的用户信息
    2. 预期结果： 页面显示错误信息，注册不成功。
  1. 注册内容不合法
     1. 输入数据：

id:”1234567890”

name: "TestUser"

password: "TestPassword123!"

email: "12456789 "

pic: "http://example.com/userpic.jpg"

class: 3

major: "计算机科学"

* + 1. 操作步骤： 用户尝试提交不合法的邮箱预期结果： 页面显示错误信息，注册不 成功。

1. **UI测试**

测试步骤和预期结果：

* + 1. 操作步骤：

在桌面浏览器上打开注册页面，调整浏览器窗口大小。

* + 1. 预期结果：

页面元素如文本框、按钮、标签在所有设备和窗口大小上均清晰可见且易于操作。布局调整以适应不同屏幕尺寸，无覆盖或不当重叠。

1. **UserInfoPage测试**
2. **功能测试**
3. **展示用户信息**
   * 1. 输入数据：用户ID “1234567890”
     2. 操作步骤：访问特定用户的详细页面
     3. 预期结果：显示用户的详细信息，如：昵称、邮箱、头像等等
4. **修改用户信息**
   * 1. 输入数据：新昵称 "TestUser123"
     2. 预期结果：用户信息更新成功，页面显示成功消息并且显示新的用户信息。
5. **UI测试**

测试步骤和预期结果：

* + 1. 操作：在桌面浏览器上打开注册页面，调整浏览器窗口大小。
    2. 预期结果：

页面元素如文本框、按钮、标签在所有设备和窗口大小上均清晰可见且易于操作。布局调整以适应不同屏幕尺寸，无覆盖或不当重叠。

1. **LoginPage测试**
2. **功能测试**
3. **登录测试**
   1. 有效登录
      1. 输入数据

name: "AliceExample"

password: "Alic3Ex@mple!"

操作步骤：

打开登录页面。

在用户名输入框中输入"AliceExample"。

在密码输入框中输入"Alic3Ex@mple!"。

点击登录按钮

* + 1. 预期结果

登录成功，页面显示“登录成功！”。

* 1. 无效登录
     1. 输入数据

**name: "AliceExample"**

**password: "wrongAlic3Ex@mple!"**

* + 1. 操作步骤：

用户输入正确的用户名和错误的密码。

点击“登录”按钮。。

* + 1. 预期结果

登录失败，页面显示“用户名或密码错误，请重新输入。”。

1. **接口测试**
2. **登录接口测试**
   1. 有效登录接口测试

**API调用详情：**

* + 1. 方法： POST
    2. URL： /api/user-login
    3. 测试步骤：

构造包含有效用户名和密码的POST请求。

发送请求到服务器。

验证响应体中是否包含成功消息

* + 1. 输入数据：

**{**

**"name": "AliceExample",**

**"password": "Alic3Ex@mple!"**

**}**

* + 1. 预期结果：

"message": "登录成功",

"token": "some\_secure\_token"

* 1. 错误的接口测试

**API调用详情：**

* + 1. 方法： POST
    2. URL： /api/user-login
    3. 测试步骤：

构造包含正确用户名和错误密码的POST请求。

发送请求到服务器。

验证响应体中是**否正确显示错误信息。**

* + 1. 输入数据：

{

"name": "AliceExample",

"password": "WrongPassword123"

}

* + 1. 预期结果：

"error": "用户名或密码错误"

**2.2.3 检索模块测试**

1. **HomePage 测试**
2. **功能测试**
3. **用户界面交互测试**
   1. 导航链接点击测试
      1. 步骤：打开主页。

点击“创建书籍”按钮。

点击“修改书籍信息”按钮。

点击“注册”按钮。

点击“登录”按钮。

点击“用户信息”按钮。

* + 1. 预期结果：

每次点击按钮后，页面跳转到对应的路径，显示相应的页面或组件。

点击“用户信息”按钮时，应该能根据当前登录用户的信息跳转到对应 的用户信息页面。

1. **登录功能测试**
   1. 有效登录测试
      1. 输入数据：

用户名： "AliceExample"

密码： "Alic3Ex@mple!"

* + 1. 步骤：

打开登录页面。

输入上述用户名和密码。

点击登录按钮。

* + 1. 预期结果

登录成功，页面显示“登录成功！”。

* 1. 无效登录测试
     1. 输入数据：

用户名： "AliceExample"

错误的密码

* + 1. 步骤：

打开登录页面。

输入上述用户名和错误的密码。

点击登录按钮。

* + 1. 预期结果：

登录失败，页面显示“用户名或密码错误，请重新输入。”。

1. **API集成测试**
2. **关键词检索接口测试**
   1. 关键词搜索
      1. 接口： GET /api/search?q=软件测试
      2. 步骤：

构造包含关键词“软件测试”的GET请求。

发送请求到服务器端。

验证返回的结果是否包含相关书籍和课程信息。

* + 1. 预期结果返回包含相关书籍和课程信息的响应

**2.2.4 论坛模块 测试**

1. **功能测试**
   1. **获取评论信息**
      1. 目标：验证系统能正确返回评论信息、点赞数、和教材实用性评分
      2. 输入数据：无需特定输入
      3. 操作步骤：

发送GET请求到/api/forum以获取所有评论信息。

* + 1. 预期结果：

返回数据包含教材评分、评论列表及点赞数

* 1. 发表评论：
     1. 目标：验证用户能成功发表评论。
     2. 输入数据：

userId: "user123"

content: "这是一个测试评论"

materialId: "material456"

* + 1. 操作步骤：

发送POST请求到/api/forum/commentThread，包含用户ID、评论内容和材料ID。

* + 1. 预期结果：

数据库中新增一条评论记录

* 1. 点赞评论
     1. 目标：验证用户可以对评论进行点赞。
     2. 输入数据：

commentId: "comment789"

* + 1. 操作步骤：

发送POST请求到/api/forum/like，包含评论ID。

* + 1. 预期结果：

应评论的点赞数增加

# 3．测试执行情况

## 3.1 软硬件环境

【操作系统、测试工具、硬件配置等】

## 3.2 测试结果

【按顺序给出每一测试项目的：

1. 实测结果数据；
2. 与预期结果数据的偏差；
3. 该项测试表明的事实；
4. 该项测试发现的问题。】

# 4．评价

## 4.1范围

【说明所完成的各项测试说明问题的范围及其局限性】

## 4.2准则

【说明评价测试结果的准则】

## 4.3软件能力

【经过测试所表明的软件能力】

## 4.4缺陷和限制

【说明测试所揭露的软件缺陷和不足，以及可能给软件运行带来的影响】

## 4.5建议

【提出为弥补上述缺陷的建议】