浙江大学

本科实验报告

课程名称：	B/S 体系软件设计
实验名称：	商品比价网站
姓　名：	周玥儿
学　院：	计算机学院
系：	计算机科学与技术
专　业：	计算机科学与技术
学　号：	3220102179
指导教师：	胡晓军

2024 年　12 月　23 日

目录

实验记录

用户注册与登录

使用令牌进行用户状态管理。令牌是一种无状态的用户认证方式，它通常采用 JSON Web Token（JWT）格式。这种机制将用户认证信息直接嵌入令牌中，从而无需服务器存储会话信息。

登陆流程

1. **用户登录**：用户通过登录接口提交用户名和密码。
2. **生成 Token**：后端验证用户身份成功后，生成一个令牌。令牌通常包含用户信息、有效期、签名等。
3. **返回 Token**：令牌返回给客户端，由客户端（如浏览器或移动应用）保存在 `localStorage` 中。

后续请求

1. **请求携带 Token**：客户端每次请求时，将 Token 放在 HTTP 请求头（`Authorization: Bearer <token>`）中发送到服务器。
2. **Token 验证**：后端验证 Token 的签名和有效期，确认用户身份。
3. **Token 失效处理**：当 Token 过期后，用户需要重新登录以获取新的令牌，或通过刷新 Token 机制（如 Refresh Token）来延长会话。

```
// 登陆的信息接口
export interface LoginInfo {
  uid: string,
  username: string,
  password: string,
  email: string,
  avatar: string,
```

```
  isLogin: boolean,
}

// 用户登陆时发送的请求以及收到的回应信息接口
export interface LoginData{
    name: string;
    password: string;
}

export interface LoginResponse{
    token: string;
}

export interface RegisterInfo{
    name: string;
    password: string;
    email: string;
}
```

用户搜索

前端通过搜索栏输入 keyword 后按下搜索按钮发起搜索请求，请求被封装成 SearchPramas 的形式，包含关键词、结果排序、分页数、分类、筛选等条件。

后端接收请求后，进行搜索。

- 首先调用创建好的爬虫对象更新搜索关键字，然后从网站抓取商品数据获取商品信息列表。
- 检查每个商品的 SKU 是否已经存在于数据库中。如果存在，则更新商品的价格信息；如果不存在，则添加新商品进数据库，在添加。
- 如果在爬虫抓取数据过程中发生异常（比如当前时间段网站繁忙），就从数据库中查找匹配关键字的商品。

全文搜索

分词处理

mysql 全文检索是以词为基础的。MySQL 默认的分词实现把**所有非字母和数字的特殊符号**都看作分词符。为了实现关键词的全文搜索，在加入商品时首先要对商品标题进行**分词处理**。

这里使用了 `jieba`，并且以空格作为示意 mysql 分词的提示。

```
def segment_text(text):
    return ' '.join(jieba.cut(text))
```

在数据库中的存储类似于。

	title	search_title
115	小米笔记本电脑 红米 Redmi Book 14 焕新版 12代酷睿标压	watch 小米 笔记本电脑 红米 Redmi Book 14 焕 新版 12 代酷 睿 标压 2.8 K 屏 高性能 轻薄 本
116	ROG魔霸新锐 2024 酷睿i9 16英寸星云屏游戏本笔记本电脑	watch ROG 魔霸 新锐 2024 酷睿 i9 16 英寸 星云 屏 游戏 本 笔记本电脑 (i9 - 13980HX 液金 导热
117	七彩虹新品上市14代酷睿i9 14900KF/i7 14700KF/i5 1460	watch 七彩虹 新品 上市 14 代酷 睿 i9 14900KF / i7 14700KF / i5 14600KF / i5 14400F + RTX40
118	荣耀 MagicBook X16 战斗版 12代酷睿标压i5 16G 512G 16	watch 荣耀 MagicBook X16 战斗 版 12 代酷 睿 标压 i5 16G 512G 16 吋 高清 护眼 屏 轻薄 本
119	ThinkPad 联想 ThinkBook 14 英特尔酷睿i5 14英寸轻薄办公	watch ThinkPad 联想 ThinkBook 14 英特尔 酷睿 i5 14 英寸 轻薄 办公 笔记本电脑 13 代 i5 - 1350
120	HAEWI 国行【2024款 15代英特尔酷睿i7】大屏笔记本电脑	watch watch HAEWI 国行 【 2024 款 15 代 英特尔 酷睿 i7 】大屏 笔记本电脑 高性能 轻薄 本 大学生 上
121	ThinkPad联想笔记本电脑 ThinkBook 14+ 2024 AI全能本	watch ThinkPad 联想 笔记本电脑 ThinkBook 14 + 2024 AI 全能 本 英特尔 酷睿 Ultra5 125H 1
122	华硕 追影i5 12600KF/14600KF/4060Ti黑神话悟空电竞游	watch 华硕 追影 i5 12600KF / 14600KF / 4060Ti 黑 神话 悟空 电竞 游戏 设计师 专项 补贴 台式 电
123	惠普 政府补贴 战99 高性能台式电脑 (14代i5-14500 16G	watch 惠普 政府 补贴 战 99 高性能 台式 电脑 (14 代 i5 - 14500 16G 1T) 23.8 护眼 大屏 14
124	联想笔记本电脑小新 Pro14 2024 高性能标压锐龙7 8745H	watch 联想 笔记本电脑 小新 Pro14 2024 高性能 标压 锐龙 7 8745H 14 英寸 轻薄 本 24G 1T
125	联想笔记本电脑小新 Pro14 2024 高性能标压酷睿 Ultra5 14	watch 联想 笔记本电脑 小新 Pro14 2024 高性能 标压酷 睿 Ultra5 14 英寸 AI 轻薄 本 32G 1T 2
126	戴尔 政府立减20% 成就3030S 台式电脑主机 (12代i3-121	watch 戴尔 政府 立减 20% 成就 3030S 台式 电脑主机 (12 代 i3 - 12100 8G DDR5 512GSSD
127	微星 i5 12400F升12490F/RTX3060/4060/4060Ti黑神话	watch 微星 i5 12400F 升 12490F / RTX3060 / 4060 / 4060Ti 黑 神话 悟空 游戏 主机 电脑 台式机
128	联想 (Lenovo) 畅玩 黑神话悟空官方合作 拯救者Y9000P A	watch 联想 (Lenovo) 畅玩 黑 神话 悟空 官方 合作 拯救 者 Y9000P AI 元启 游戏 笔记本电脑 i9
129	华为 MateBook D 14 SE 2024笔记本电脑 13代酷睿/14英	watch 华为 MateBook D 14 SE 2024 笔记本电脑 13 代酷 睿 / 14 英寸 护眼 全面 屏 / 轻薄 办公

关联优化

在 mysql 中有两种模式的全文搜索可选：

- **NATURAL LANGUAGE MODE**：按相关性排序，基于词频，适合一般的全文搜索。
- **BOOLEAN MODE**：可以使用布尔运算符来精确控制查询，适合需要更复杂条件的查询。

为了优化搜索结果的相关性，每次搜索后，在搜索索引中加入查询关键词再更新/加入数据库，以提高搜索结果的关联度。

```
search_title = (keyword+' '+segment_text(result['title']))[:255]
```

全文搜索操作通过封装 fulltextsearch 函数实现

```
def fulltextsearch(keyword):
    query = text("SELECT * FROM items WHERE MATCH(search_title) AGAINST
(:keyword IN NATURAL LANGUAGE MODE)")
    result = db.session.execute(query, {'keyword': keyword})
    return result.fetchall()
```

由于 Flask-SQLAlchemy 不支持创建全文索引，需要在初始化数据库的时候手动执行 mysql 语句创建索引

```
# app/__init__.py


def db_init(app):
    with app.app_context():
        db.drop_all()
        db.create_all()  # 数据库初始化
        db.session.execute(text('CREATE FULLTEXT INDEX idx_title ON ite
ms(search_title);'))
    print('Database initialized')
```

爬虫处理

爬虫有两种方案：

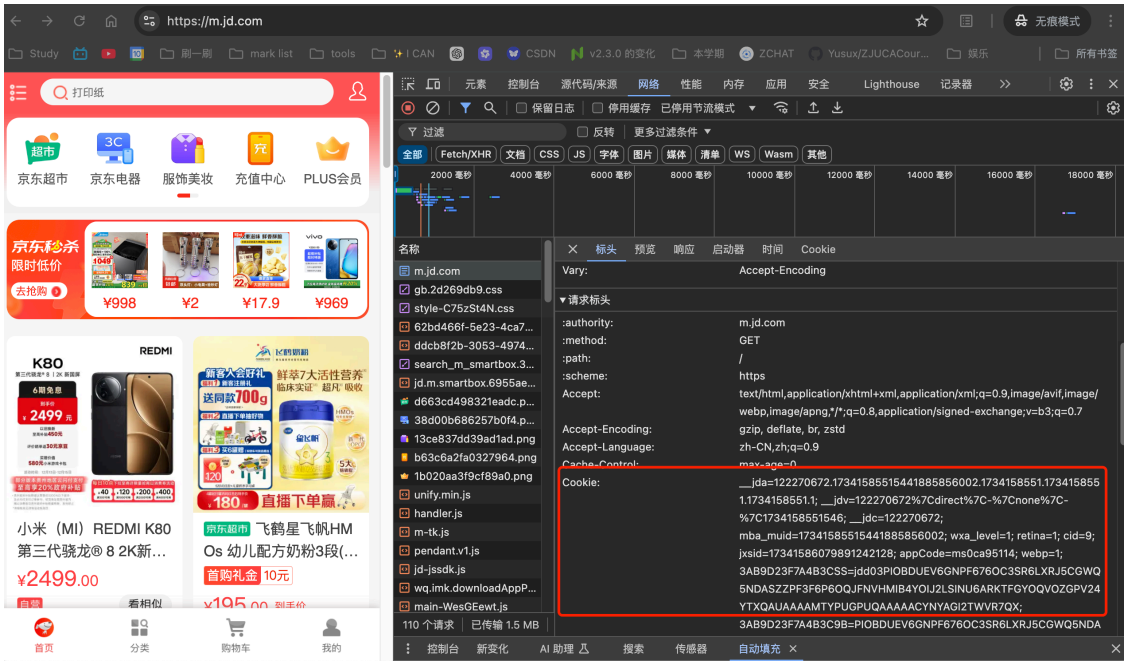
- 第一种方式是使用 request 模拟请求，并使用 bs4 解析 respond 得到数据。
- 第二种是使用 selenium 和无头浏览器，selenium 自动化操作无头浏览器，由无头浏览器实现请求，对得到的数据进行解析。

京东

- 京东的登录通常需要模拟用户输入用户名、密码以及验证码。可以手动登录京东，获取登录后 cookie，然后将 cookie 传递给爬虫使用。**使用程序自动**

登录（例如使用 requests 或 selenium）：模拟登录过程，自动获取 cookie。但自动登录涉及处理验证码，这需要复杂的机制，可以用 selenium 实现模拟登录。

- 根据用户提供的关键词，构建搜索 URL 并请求搜索页面。京东的搜索链接通常包含 keyword（关键词）和 page（页数）参数。




```
keyword = "手机"
start_url = 'https://search.jd.com/Search?keyword=' + keyword + '&enc=utf-8&wq=' + keyword
print("搜索链接: ", start_url)
```

- 使用 request 向搜索链接发送请求，需要将 cookie 添加到请求头中，以确保爬虫模拟登录后的状态。

```
response = requests.get(start_url, headers={
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36',
    'cookie': cookie  # 使用登录后的cookie
```

```
})
response.encoding = 'utf-8'
```

- 使用 BeautifulSoup 解析获取的 HTML 内容，以便从中提取需要的商品信息。在解析后的页面中，定位到包含商品信息的标签，例如商品的 sku、名称、价格 等。通过选择器来定位到包含商品数据的 HTML 元素。 京东的商品信息通常位于 `#J_goodsList > ul` 标签下，每个商品的信息会在 `li.gl-item` 标签中。通过 `soup.select()` 或 `soup.find_all()` 方法，定位到所有商品元素。

```
# 使用 BeautifulSoup 解析页面内容
soup = BeautifulSoup(response.text, 'html.parser')
 # 定位到商品列表
goods_list = soup.select('#J_goodsList > ul > li.gl-item')
```

价格提醒

Celery 检查价格

通过设置一个 celery 队列 + reddis，来周期性地执行价格更新任务，并且根据更新的价格、提醒的设置决定是否要给用户发送价格提醒。

```
@celery.task(name='alert')
def check_price():
    results = []
    with db.session() as session:  # 使用 Flask 的 SQLAlchemy
        alerts = session.query(PriceAlert).all()  # 查询所有 alertlist 条目
        for alert in alerts:
            if alert.enable is False:
                continue
            current_price = get_item_details(alert.item_id)['current_price']  # 调用爬取价格的逻辑
```

```
            if current_price <= alert.target_price:
                if alert.notification_method == 'email':
                    notify_result = email_notify(alert.user_id, alert.i
tem_id, current_price)
                elif alert.notification_method == 'sms':
                    notify_result = sms_notify(alert.user_id, alert.ite
m_id, current_price)
                results.append({
                    ...
                })
    return results
```

每次使用时需要启动 Reddis 服务器、Celery 任务和周期性 Beat 任务

邮件发送

使用 Flask 为用户发送邮件提醒可以通过 `Flask-Mail` 插件实现。在 Flask 应用中配置邮件服务器的参数，SMTP 地址、端口和认证信息。这里我使用的是网易 163 邮箱提供的服务。

记录相关信息到配置中

```
MAIL_SERVER="smtp.163.com"
MAIL_PORT=465
```

```
MAIL_USE_SSL=True
MAIL_USERNAME="MyPrice2025@163.com" # 发送邮箱
MAIL_PASSWORD="****" # 客户端授权码
```

在 app 的初始化时加入 Mail 功能，在需要发送邮件的场合里调用 flask 相关函数即可。

```
def email_notify(user_id, item_id, price):
    title = "[My Price Notification] 您收藏的商品价格下降啦！"
    email = get_user_email(user_id)
    product = get_item_details(item_id)
    alert = query_alert(user_id=user_id, item_id=item_id)
    content =  f"您收藏的商品 {product['title']} 价格已经降到 {price} 元！您预期的价格曾经是 {alert.target_price} 元。\n 点击链接查看详情：{product['link']}"
    print(f"Sending notification to {email} for product {item_id} at price {price}, title: {product['title']}, content: {content}")
    message = Message(title, recipients=[email], sender=Config.MAIL_USERNAME, body=content)
    with current_app.app_context():
        mail = Mail()
        mail.send(message)
    # 这里写通知逻辑，例如发送邮件或短信
    alert.enable = False # 关闭提醒
    add_alert_history({
        'alert_id': alert.id,
        'price_after': price
    })
    print(f"User {user_id} notified for product {item_id} at price {price}.")
    return True
```

短信发送

使用短信宝提供的服务，从短信服务提供商的管理界面获取必要的 API 密钥、账户 ID 等信息。使用的 API 接口为

```
# coding=utf-8
import urllib
import urllib.request
import hashlib

def md5(str):
    import hashlib
    m = hashlib.md5()
    m.update(str.encode("utf8"))
    return m.hexdigest()

statusStr = {
      ...
}

smsapi = "http://api.smsbao.com/"
```

```
# 短信平台账号
user = '***'
# 短信平台密码
password = md5('****')
# 要发送的短信内容
content = '短信内容'
# 要发送短信的手机号码
phone = '*****'

data = urllib.parse.urlencode({'u': user, 'p': password, 'm': phone, 'c
': content})
send_url = smsapi + 'sms?' + data
response = urllib.request.urlopen(send_url)
the_page = response.read().decode('utf-8')
print (statusStr[the_page])
```

在 celery task 中调用提醒函数即可

```
def sms_notify(user_id, item_id, price):
    product = get_item_details(item_id)
    alert = query_alert(user_id=user_id, item_id=item_id)
    if len(product['title']) > 15:
        product['title'] = product['title'][:15] + '...'
    content = f"【云比价】商品{product['title']}价格当前为 {price} 元！您
的预期价格为 {alert.target_price} 元。"
    print(f"Sending SMS notification to user {user_id} for product {ite
m_id} at price {price}, content: {content}")

    # 获取用户的手机号码
    phone_number = get_user_phone(user_id)
    if not phone_number:
        print(f"User {user_id} has no phone number, skip sending SMS.")
        return False
```

```
    # 使用 Twilio 发送短信
    data = urllib.parse.urlencode({'u': Config.SMS_USER, 'p': Config.SM
S_PASSWORD, 'm': phone_number, 'c': content})
    send_url = Config.SMSAPI + 'sms?' + data
    response = urllib.request.urlopen(send_url)
    the_page = response.read().decode('utf-8')
    print (Config.SMS_STATUS[the_page])
    alert.enable = False  # 关闭提醒
    add_alert_history({
        'alert_id': alert.id,
        'price_after': price
    })
    print(f"User {user_id} notified for product {item_id} at price {pri
ce}.")
    return True
```


14:21
64
145
10684562990000019621
信息・短信
今天 14:16
【云比价】您收藏的商品 Apple/苹果 iPhone 13 (A2634) 128GB 午夜色 支持移动联通电信 5G 双卡双待手机 价格已经降到 3517.0 元！ 您预期的价格曾经是 4000.0 元。

遇到的困难

跨域问题

localhost：5000 收不到换成 127.0.0.1:5000 即可

withCredential

token 加在 header 里的格式，需要前后端一致。比如说在 authorization 字段下。

页面设计

flask 和 celery 模块化后循环引入的问题

分为两次注册 app，第一次用于注册 celery，第二次在主程序中提供上下文环境并且注册蓝图

```
# app/__init__,py


def register_celery(celery, app):
    class ContextTask(celery.Task):
        abstract = True
        def __call__(self, *args, **kwargs):
            with app.app_context():
                return self.run(*args, **kwargs)
    celery.Task = ContextTask


def create_app(celery=None,db_init=False,register_celery_blueprint=Fals
e):
    app = Flask(__name__)
```

```
    app.config.from_object(Config)
    db.init_app(app)
    CORS(app, supports_credentials=True, resources={r"/*": {"origins":
"http://localhost:5173"}})
    if celery:
        register_celery(celery=celery, app=app)  # >> 注册 celery
    if register_celery_blueprint:
        from celery_route import celery_bp
        app.register_blueprint(celery_bp)
    app.register_blueprint(api)  # /api/search, /api/login # 注册蓝图
    if db_init:
        with app.app_context():
            db.drop_all()
            db.create_all()  # 数据库初始化
            # db.session.execute(text('CREATE FULLTEXT INDEX idx_title
ON items(search_title);'))
            db.session.execute(text('ALTER TABLE items ADD FULLTEXT IND
EX ngram_index (title) WITH PARSER ngram;'))
            print('Database initialized')
    return app


def make_celery(app_name):
    celery = Celery(app_name,
                    broker=celeryconfig.broker_url,
                    backend=celeryconfig.result_backend)
    celery.config_from_object(celeryconfig)
    return celery

celery = make_celery(__name__)
app = create_app(celery=celery, db_init=False, register_celery_blueprin
t=False)
```

notes/flask/03-2 使用 Celery 执行周期任务.md at master · newbieof410/notes · GitHub

```python
# run.py
import os
project_root = os.path.abspath(os.path.dirname(__file__))
os.environ['PROJECT_ROOT'] = project_root

from app import create_app, celery

app = create_app(db_init=False, register_celery_blueprint=True)

if __name__ == '__main__':
    for rule in app.url_map.iter_rules():
        print(rule)
    app.run(debug=True)
```

短信发送

- 一开始准备使用 twilio 提供的服务，但是不能正常接收到短信，后面换成使用短信宝提供的服务。
- 服务签名一开始使用形如【比价提醒】的格式，但是过于中性，被要求换成公司商标。经测试，【云比价】可以正常通过发送。

try...catch 无法捕捉 promise.reject 异常

根据逻辑用最外层的处理函数来处理，不能过早地在内部就直接进行捕捉后处理。

京东爬虫失效

12/17 京东疑似调整了页面结构和登陆手段，导致爬虫失效，换成使用比价网站的内容。

心得体会

在选课之前就已经对 B/S 体系软件设计有所耳闻，学长们都称这是一门真正的计算机专业课，一门“开局一个大作业，其他全靠自己写”的课程。

我在上这门课之前接触过一些 vue 的前端知识——曾经我加入了学校的某个学生组织，在里面参与过一些网站开发的小练手。在部门学长的建议下学习了 vue3 的教程，但这些也都是“纸上谈兵”。理论知识过不了多久在没有实践的情况下就被忘在了脑后。

后来就是数据库系统课上被要求写一个“图书管理系统”，并且完成相应的前后端。那次开发是在学长提供的框架之下，后端只用按需完成部分填空即可。前端的灵活度也被限制在了仅仅更换某些组件。也没有给出完整的开发框架。

所以本次开发可以算是我第一次完完整整地从零开始架构，独立分析需求、拟定设计文档、开发、测试、部署的项目经历。以下将从开发的各个阶段来简短抒发下心得。（因为其他期末考试还没复习呜呜呜，要来不及了）

需求分析

在拿到需求（作业要求）之后，我对项目有了粗浅的理解，认为这并不是一个非常复杂的项目，起码每个功能都有实现的思路。随后，我便开始了挑选前后端的开发技术。

在这一阶段，我也花了不少时间。先前对前后端的开发技术并不了解，只知道有很多种奇奇怪怪的搭配，但从来不知道他们各自在项目中起了什么作用。我先搜索了一些开发者论坛，确定了前端的框架。本来是想尝试下使用 react 来拓宽自己的技术栈，但是后面还是不想为难自己，准备沿用之前有一定基础的 vue3 框架。现在想想，也幸亏当时没有选择去学习新的技术，否则完成项目的难度又将提升一个阶梯。

对于后端的语言无非是在 java 和 python 之间做出选择。因为我有一定的 python 基础，所以果断选用了 python。

在选择好技术栈之后，我的第一想法是先从 github 上找到一些开源的、和我使用相同技术栈的项目，学习他们的项目架构来进行改编。如果能找到类似的商城开发项目更是能为我锦上添花。不过很可惜，因为 vue3 是近几年更新的技术，所以找到的大部分项目使用的都是 vue2，或者就是项目架构不够完整。接着，我又转念是否能找到商城框架的静态 html，来自己把它转换成 vue3 的框架。后来，我很幸运地找到了一个别人跟着 b 站改编的商城项目练手框架，在学习之后我就在这个框架上进行前端开发。

前端开发

我的前端开发先从改动一些小组件开始，比如说 header、搜索框、footer 等。在读懂了别人的项目架构之后，我尝试按照自己的理念来编写页面，在此过程中逐渐熟悉了 vue3 和 typescript 的语法。但还是会有一些技术性要求比较高的地方一知半解，比如说 vue 的响应式处理。

在掌握完编程理论知识后，就要思考项目的逻辑层面需求了。根据以往的经验，我知道联系前后端最重要的便是——“接口设计”。我又去学习了一系列知识，花了一段时间分清前端的路由路径、后端的 api 路径、互联网 url 等等概念。后续参照别人的一些博客教程，尝试按照 RESTFUL 规范来设计。在这个阶段，我的每一次设计其实都充满了自我怀疑，因为我很担心在后端代码的编写中无法和前端的接口匹配而导致需要不停地修改代码。

我起初以为前端只需要负责思考页面的呈现即可，但是发现在引入 pinia 和 axios 这些功能之后还要去考虑参数的存储、和后端的交互。学习这一部分也花费了我不少时间，尤其是处理各个 store 的状态。

后端开发

实际上，我的前后端开发是同步进行的，因为比较担心全部写好之后再进行测试的 debug 工作量会很大。我的后端代码也经历了很多版本的迭代，从一开始的单文件，到后来的多文件，再到最后的不同文件模块下的多文件格式。代码的架构逐渐复杂化，也提高了我的开发难度。

我先根据需求文档设计好了数据库结构并且定义了 modles，事实证明一开始很难想得非常全面，这个数据库的格式在后续开发中还是经历了多轮迭代修改。

我最初尝试实现的是登录和注册功能。为了更加规范，在这里我接触到了我很多不了解的概念，比如说登录后的 JWT 验证。我也不知道该怎么处理这些验证。不过在后续的学习中，我也做到了对登录信息进行了一定的处理，比如说对密码进行加密后传输、赋予登录的用户一个 JWT token 来验证其身份信息等等。

接着就来到本项目最难实现的部分了——爬虫。我早就对各大电商网站的反爬机制有所耳闻。先在网上找了一圈爬虫代码后，发现都无法处理淘宝的情况（request 一直失败、selenium 又依赖于手动过于缓慢），于是我转战京东。幸运的是，很快就发现了使用 requsest 就能成功处理京东的场合。但是根据这个原理我试图在淘宝上复现，还是无果。多次尝试后还是放弃了从淘宝获取数据。准备先用京东搭好基础框架。

其他的前后端交互（价格提醒，历史价格，详情页面）部分完成地都比较顺利，只需要确定好和前端之间数据传递的格式，返回前端需要使用的数据即可。需要注意的是，数据的格式真的非常非常重要，一不小心就会匹配错误！这段时间的开发非常消耗体力和耐心。

后来，就到了要实现异步处理价格提醒的部分了。这一阶段我使用了 celery 队列来完成，准备让他在后台定期执行任务。这又要求我学习新的技术，不过我也已经习惯了（笑）。在粗略地了解了 celery 和 redis 之后，我就尝试把它们揉和起来。过程中还是遇到了很多困难的，尤其是和 flask app 初始化循环引用的问题。不过

在网上参照了别人的博客之后还是顺利结局了。在这段开发过程中，我还使用了 postman 不断进行测试，节省了很大的精力。

这一部分我还学习了如何使用 python 借助互联网协议发送 email，如何调用其他短信运营商提供的接口发送短信等技术，收获颇丰。（但也真的很花时间😭）

终于，在项目的后期，我尝试接入另一个电商平台——亚马逊。亚马逊的反爬做得并不严格，所以处理起来比京东更快一些。其实后来还打算接入更多的平台，例如苏宁易购、唯品会等等。我分析了苏宁的接口请求感觉还是比较好爬的，不过没太多时间就先暂时搁置了。

docker 部署

这又是一个我完全没接触过的领域！唉，从头学习。

不过这一阶段我遇到比较大的问题就是的前端和后端请求一直匹配不上，以及后端和 redis 等环境不能成功对接。在 debug 的过程中我倒是对了网络交互有了更多的认识，包括 nginx 反向代理、localhost、端口、宿主机代理、docker 网络等等有关计算机网络的知识。最后成功打包并且运行起来的时候还是很有成就感的。

最后尝试并且打包发布在了 Docker Hub 上。由于全部都是自己在网上学习，也不知道做的是否规范。

存在的不足

开发过程中最让人崩溃的无非就是不断失灵的爬虫！由于有时候的过多请求或者电商网站更换机制，爬虫总是会间歇性地失效，要求我不断去获取新的 cookie 或者调整其他代码。这一点也是我部署之后最担心的一点，也导致了我的网站无法长期的自主运营下去。需要我或者用户不断自己维护。

同时，由于并没有使用很好的搜索算法，只是简单地使用了 mysql 本身支持的全文索引，商品的搜索结果往往不尽如人意，无法按照相关度进行结果的优先级排序展示。

总的来说，我的网站总有很多还能优化的部分，比如对高并发的支持，以及一些细节处的维护（以免用户试出 bug）。我总是对我的网站充满了担心，觉得他非常地破碎，害怕他运行着运行着就崩溃了。

收获

最大的感受是，一个人毫无经验的网页开发真的好难做到规范化，因为会缺少很多“常识性”的指导。从一开始，我就不停地在网上去搜索一些“开发规范”，但似乎大部分都没有一个明确的规定。找了很久才找到类似于，“设计文档怎么写”，“接口怎么设计”，这样的“授人以渔”的教程。我想，大部分的网页开发应该是在一个已经具有一定规模的团队内部进行的，而团队内部会从更加宏观的层面基于前人经验规定好开发要求。但是对于我这样一个还未入门的初学者来说，很难获取到相关信息，也不愿意投入很大的时间进行文档学习。所以大部分情况我都是借助 ChatGPT 等工具，希望他们为我提供一个“常识性”的解答。比如说，前端的架构该如何设计、后端的模块化应该怎么划分等等。

此外，我还领悟到，学习就是要在动手中学。不要等到学完了所有理论知识再动手，完全可以借助各种各样的工具、前人经验，先动手开始写，在一次次修改代码和 debug 中就能学到很多知识。（但是如果 debug 效果实在不好，还是得从理论层面来分析错误的）

总结

希望老师测试的时候手下留情呜呜呜！

以及真的非常感谢老师这学期的付出和解答，从这个作业中学习到了很多网页开发知识，也算是一种外力让我入门了一点点网页开发。总的来说是一次很有趣很有趣

的项目经历，学习新技术很开心，但是真的很花时间。在这个过程中真的提升了很多个人的学习能力和代码能力！