University of Macau
Nov 10th, 2024

澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

CISC3014 Information Retrieval and Web Search

Final Report

ebay's automated price comparison system:

a solution to young people's difficulty in making choices

By

Li Shitian
DC126892
&
Chen Yu
BC104414

Introduction

In the digital age where e-commerce has become a cornerstone of retail, consumers are increasingly looking for the most cost-effective purchasing options available online. To aid in this pursuit, our team has developed a comparative shopping tool that leverages web scraping technologies to identify the lowest priced options for specific products on eBay, one of the world's largest online marketplaces. This innovative tool is designed to provide a competitive edge to savvy shoppers by automating the process of price comparison across multiple product listings.

The primary objective of this project is to simplify the user's shopping experience by reducing the time and effort spent in searching for the best deal. By inputting the name of a product, users can deploy our dual web crawlers that concurrently search through eBay listings to fetch the lowest priced item for each product queried. The tool then compares these prices and presents the user with the item that offers the best value for money.

Our report outlines the development process of our comparative shopping tool, from conceptualization to execution, emphasizing the technical strategies employed, the challenges faced, and the solutions devised to overcome these hurdles. By automating the comparison of product prices, our tool not only enhances the shopping experience but also promotes a more economical approach to online purchasing, reflecting the project's alignment with the needs of cost-conscious consumers in today's e-commerce landscape.

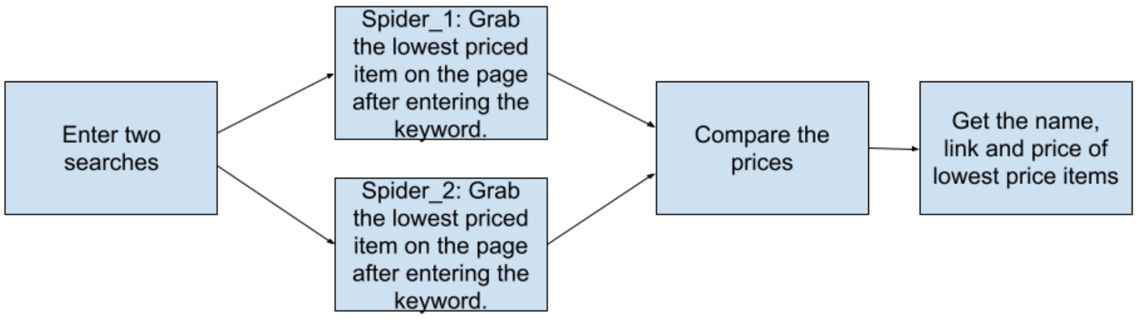
Background

In recent years, an important trend has emerged among young consumers, characterized by difficulties in decision-making and a strong preference for low-cost products. This phenomenon, often described as “choice paralysis”, is attributed to the plethora of products available in today's consumer marketplace, coupled with the economic pressures faced by this demographic. Therefore, we have designed an automated price comparison system that allows us to find the lowest priced of these products that best meets the expectations of young people by simply entering the brand of shoe type they want and some keywords.

The impetus for choosing eBay as the focus of our project stems from its accessibility and popularity, which make it a practical and impactful target for a price comparison tool. Additionally, eBay's vast inventory and varying seller pricing strategies create a dynamic environment where prices can fluctuate significantly, thereby presenting a prime opportunity for significant savings on consumer goods.

Our project was specifically conceived to support low-income consumers who are often the most affected by price variances and may not have the time or resources to manually search for the best deals. By automating the process of price comparison, our tool aims to empower this demographic, ensuring they achieve the best possible prices with minimal effort. This focus is rooted in a broader commitment to economic equity, striving to provide all individuals with the capability to make cost-effective purchasing decisions, regardless of their economic status.

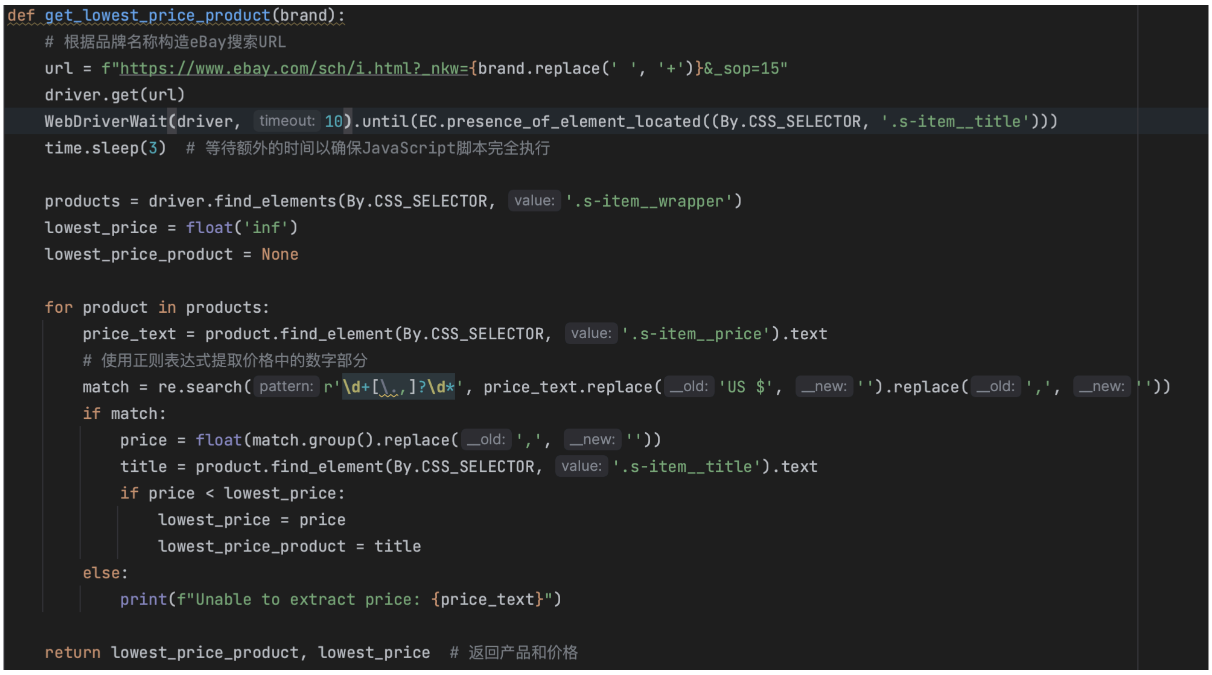
Architecture



Enter two searches
Spider_1: Grab the lowest priced item on the page after entering the keyword.
Spider_2: Grab the lowest priced item on the page after entering the keyword.
Compare the prices
Get the name, link and price of lowest price items

Specify the path to the ChromeDriver via the Service object and set up the Chrome user agent to simulate browser access. Then set up the get_lowest_price_product function, which is the core of the crawler and is used to open the eBay search page, enter the brand name, and search for related products.

The search URL is then constructed based on the brand name and navigates to that page. And it uses WebDriverWait to wait for the product title element to finish loading to make sure the page is fully loaded. Finally, iterate through the product elements on the page and use regular expressions to extract values from the product prices. Compare to find the lowest priced item.

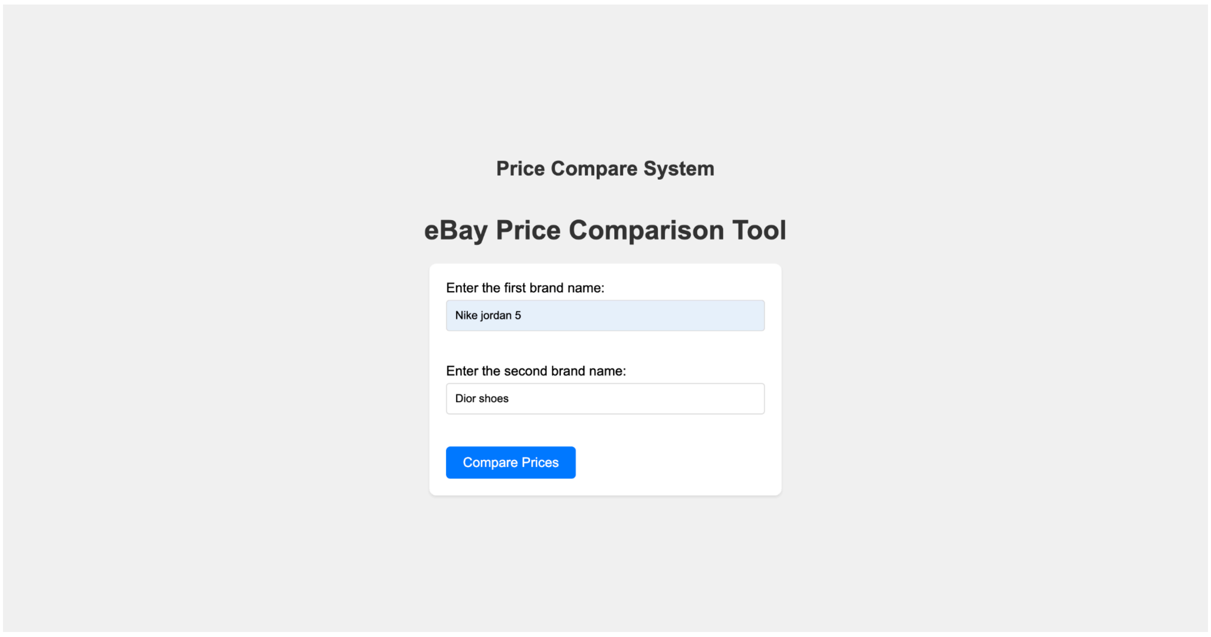


```python
def get_lowest_price_product(brand):
    # 根据品牌名称构造eBay搜索URL
    url = f"https://www.ebay.com/sch/i.html?_nkw={brand.replace(' ', '+')}&_sop=15"
    driver.get(url)
    WebDriverWait(driver, timeout: 10).until(EC.presence_of_element_located((By.CSS_SELECTOR, '.s-item__title')))
    time.sleep(3)  # 等待额外的时间以确保JavaScript脚本完全执行

    products = driver.find_elements(By.CSS_SELECTOR, value: '.s-item__wrapper')
    lowest_price = float('inf')
    lowest_price_product = None

    for product in products:
        price_text = product.find_element(By.CSS_SELECTOR, value: '.s-item__price').text
        # 使用正则表达式提取价格中的数字部分
        match = re.search( pattern: r'\d+[\.,]?\d*', price_text.replace( __old: 'US $', __new: '').replace( __old: ',', __new: ''))
        if match:
            price = float(match.group().replace( __old: ',', __new: ''))
            title = product.find_element(By.CSS_SELECTOR, value: '.s-item__title').text
            if price < lowest_price:
                lowest_price = price
                lowest_price_product = title
        else:
            print(f"Unable to extract price: {price_text}")

    return lowest_price_product, lowest_price  # 返回产品和价格
```

In the main program, the user is prompted to enter two brand names. The get_lowest_price_product function is called for each brand and gets the returned lowest price product and its price. Then, the prices of the products of these two brands are compared and the information about the lowest price product is output.

Sample

First, after the program runs, you can open our front-end app.html file, whose ip address is http://127.0.0.1:5000/



Price Compare System

eBay Price Comparison Tool

Enter the first brand name:

Nike jordan 5

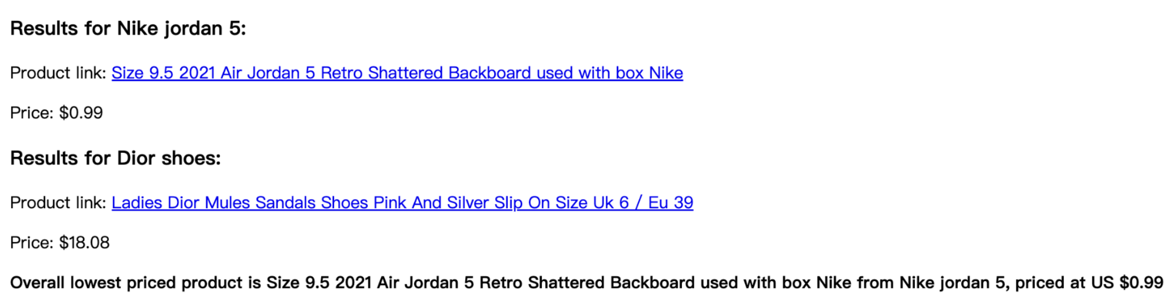
Enter the second brand name:

Dior shoes

Figure 1: Front end of our system

In this case, we entered some specific product details to ensure that what we found was what we actually wanted. Nike jordan 5 and Dior shoes, as shown in the picture, are specific to the brand as well as the style.

After the crawler crawls the content, we can get the conclusion as shown in the figure.



Results for Nike jordan 5:

Product link: Size 9.5 2021 Air Jordan 5 Retro Shattered Backboard used with box Nike

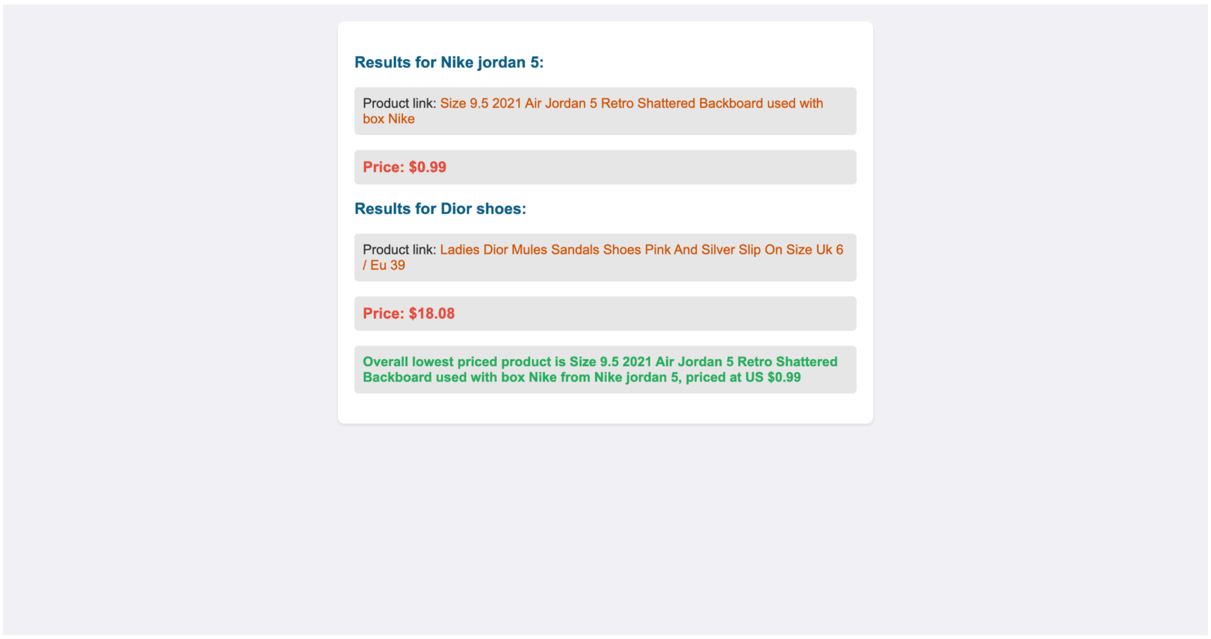
Price: $0.99

Results for Dior shoes:

Product link: Ladies Dior Mules Sandals Shoes Pink And Silver Slip On Size Uk 6 / Eu 39

Price: $18.08

Overall lowest priced product is Size 9.5 2021 Air Jordan 5 Retro Shattered Backboard used with box Nike from Nike jordan 5, priced at US $0.99


Results for Nike jordan 5:

Product link: Size 9.5 2021 Air Jordan 5 Retro Shattered Backboard used with box Nike

Price: $0.99

Results for Dior shoes:

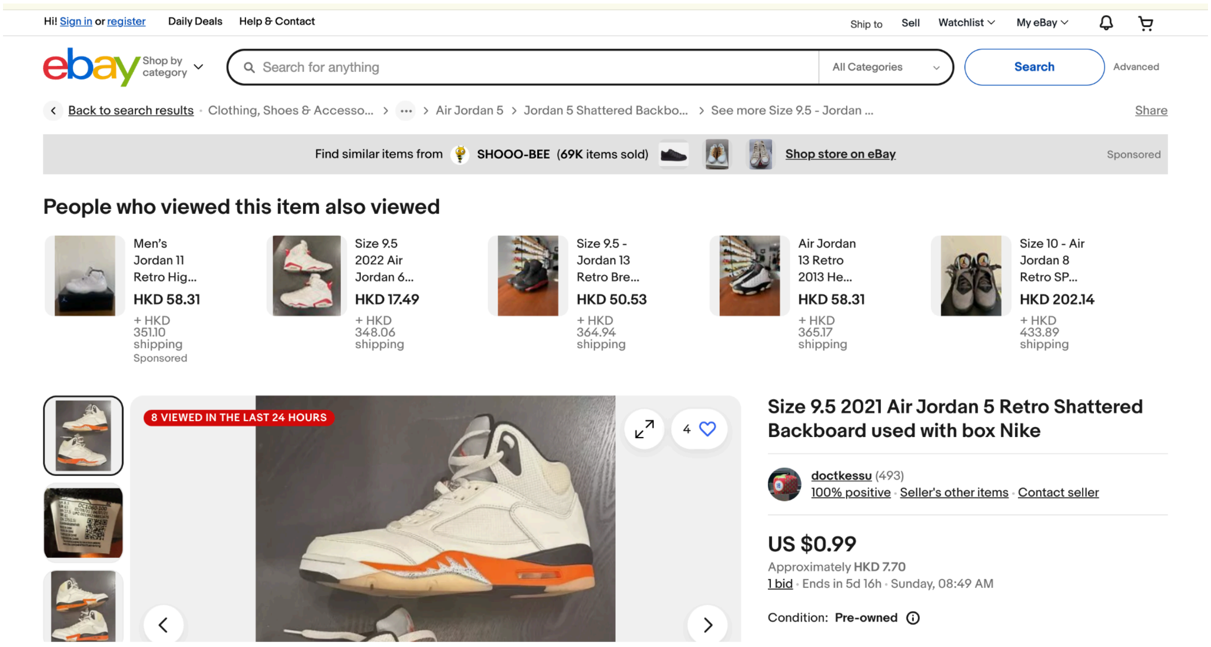
Product link: Ladies Dior Mules Sandals Shoes Pink And Silver Slip On Size Uk 6 / Eu 39

Price: $18.08

Overall lowest priced product is Size 9.5 2021 Air Jordan 5 Retro Shattered Backboard used with box Nike from Nike jordan 5, priced at US $0.99

Figure 2: Result of our system

After selecting the product with the lowest price, the user can go to the corresponding product page directly from the link.



People who viewed this item also viewed

Men's Jordan 11 Retro Hig... HKD 58.31 + HKD 351.10 shipping Sponsored

Size 9.5 2022 Air Jordan 6... HKD 17.49 + HKD 348.06 shipping

Size 9.5 - Jordan 13 Retro Bre... HKD 50.53 + HKD 364.94 shipping

Air Jordan 13 Retro 2013 He... HKD 58.31 + HKD 365.17 shipping

Size 10 - Air Jordan 8 Retro SP... HKD 202.14 + HKD 433.89 shipping

Size 9.5 2021 Air Jordan 5 Retro Shattered Backboard used with box Nike

doctkessu (493)
100% positive · Seller's other items · Contact seller

US $0.99
Approximately HKD 7.70
1 bid · Ends in 5d 16h · Sunday, 08:49 AM

Condition: Pre-owned

Figure 3 : Product detail page after jumping from the link

Future

Extending and enhancing the current eBay Price Comparison Tool, it could be developed in the future into a more comprehensive e-commerce solution that not only simplifies the buying decision process for users, but also automates routine tasks and provides deeper data insights. The following is a specific description of the system's future outlook:

First, considering the need to automate the buying process, an auto-buy function can be integrated, allowing users to set auto-buy conditions, such as automatically executing a purchase when a price threshold is reached. In addition, an automated bidding system would be a valuable addition to increase the chances of obtaining low-priced items by automatically placing bids in auctions based on preset rules.

Secondly, enhanced product comparison features will greatly improve user experience. By adding more detailed filtering criteria, such as the new or old status of the item, shipping options, seller ratings and return policies, users will be able to make more detailed choices. Meanwhile, support for multi-brand or multi-product simultaneous comparisons will also enable users to see a wide range of options and comparisons at a glance.

Price trend analysis is also an important direction for future development. The tool tracks the price history of a specific product, analyzes price trends, and predicts the best time to buy. The price alert feature can notify users when the price of an item drops to a level set by the user.

Optimizing the efficiency and speed of web crawling is necessary from a performance and scalability point of view, which not only reduces waiting time but also lightens the computational load. In addition, migrating data processing and storage to a cloud platform allows for faster processing and better data management capabilities.

Improvements in user experience are equally critical. Designing a more intuitive user interface will simplify the process and make it easy for non-technical users. And, optimizing mobile device compatibility ensures that users will enjoy the same functionality on mobile devices as they do on the desktop version.

Finally, ensuring data security is a must when expanding functionality. It's especially important to implement strong security measures to protect user data and credentials, especially in features that involve transactions and personal information. At the same time, ensure that the tool is compliant with data protection regulations such as GDPR and CCPA.

By integrating these extensions into the eBay Price Comparison tool, the online shopping experience can be greatly enhanced, providing users with not only practical help with their purchases, but also strategic shopping advice.

Conclusion

In this report, we have implemented a Python script and Selenium WebDriver automation to crawl and compare product prices by brand name from the eBay website. The script simulates the process of a user browsing a web page, and is able to navigate to eBay's search function, extract product prices, and determine the lowest-priced item for a given brand. This process not only demonstrates the practical application of automated web crawlers in e-commerce, but also highlights the importance of data crawling techniques in the modern web environment.

By dynamically constructing search URLs and extracting price data with regular expressions, this script is able to flexibly respond to different query needs of users and accurately perform price comparisons. In addition, the script also properly manages the opening and closing of browser sessions to ensure the efficient utilization of computing resources and prevent memory leakage or resource hanging problems. The rational use of these operations greatly enhances the user interactivity and usefulness of the script.

In conclusion, this study utilizes Python in conjunction with Selenium to provide users with a powerful tool for monitoring and comparing the prices of different brands of products on eBay. This not only provides users with value insights to help them make informed purchasing decisions, but also provides an effective method for price monitoring and market analysis in the e-commerce space. Through this study, we can see the potential of automation technologies in processing and analyzing complex data, foreshadowing the key role of automation tools in future data-driven decision-making processes.

Code Reference

```
from flask import Flask, request, render_template
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import re
import os

app = Flask(__name__)

def get_lowest_price_product(brand):
   # Setup Chrome WebDriver
   chrome_driver_path = os.getenv('CHROME_DRIVER_PATH',
'/opt/homebrew/bin/chromedriver')
   s = Service(chrome_driver_path)
   options = webdriver.ChromeOptions()
   options.headless = True  # Enable headless mode for performance
   driver = webdriver.Chrome(service=s, options=options)

   # Prepare the URL
   url = f"https://www.ebay.com/sch/i.html?_nkw={brand.replace(' ',
'+')}&_sop=15"
   driver.get(url)

   # Wait until the item titles are loaded
   try:
       WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.CSS_SELECTOR,
'.s-item__title')))
   except:
       driver.quit()
```

```
        return "No products found", float('inf'), None

    # Scrape the products
    products = driver.find_elements(By.CSS_SELECTOR, '.s-item__wrapper')
    lowest_price = float('inf')
    lowest_price_product = "No products found"
    product_url = None

    # Iterate through products to find the lowest priced one
    for product in products:
        try:
            price_text = product.find_element(By.CSS_SELECTOR,
'.s-item__price').text
            match = re.search(r'\d+[\.,]?\d*', price_text.replace('US $',
'').replace(',', ''))
            if match:
                price = float(match.group().replace(',', ''))
                title = product.find_element(By.CSS_SELECTOR,
'.s-item__title').text
                link = product.find_element(By.CSS_SELECTOR,
'.s-item__link').get_attribute('href')
                if price < lowest_price:
                    lowest_price = price
                    lowest_price_product = title
                    product_url = link
        except:
            continue  # if an error occurs, skip this product

    driver.quit()
    return lowest_price_product, lowest_price, product_url

@app.route('/')
def index():
    return render_template('app.html')

@app.route('/compare', methods=['POST'])
def compare():
    brand1 = request.form['brand1']
    brand2 = request.form['brand2']

    # Fetch product info
    product1, price1, link1 = get_lowest_price_product(brand1)
    product2, price2, link2 = get_lowest_price_product(brand2)

    # Determine the overall lowest price
    if price1 == float('inf') and price2 == float('inf'):
        result = "No products found for either brand."
    elif price1 <= price2:
        result = f'Overall lowest priced product is {product1} from {brand1},
priced at US ${price1:.2f}'
        best_link = link1
    else:
        result = f'Overall lowest priced product is {product2} from {brand2},
priced at US ${price2:.2f}'
        best_link = link2

    return render_template('results.html', brand1=brand1, product1=product1,
price1=price1, link1=link1,
                          brand2=brand2, product2=product2, price2=price2,
link2=link2, result=result, best_link=best_link)

if __name__ == '__main__':
    app.run(debug=True)
```

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>eBay Price Comparison</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            background-color: #f4f4f4;
        }
        h1, h2 {
            text-align: center;
            color: #333;
        }
        form {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }
        label {
            margin-bottom: 10px;
            font-size: 16px;
        }
        input[type="text"] {
            width: calc(100% - 22px);
            padding: 10px;
            margin-top: 5px;
            margin-bottom: 20px;
            border: 1px solid #ddd;
            border-radius: 4px;
        }
        button {
            background-color: #007BFF;
            color: white;
            border: none;
            padding: 10px 20px;
            font-size: 16px;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h2>Price Compare System</h2>
    <h1>eBay Price Comparison Tool</h1>
    <form action="/compare" method="post">
        <label for="brand1">Enter the first brand name:</label>
        <input type="text" id="brand1" name="brand1" required>
        <br><br>
        <label for="brand2">Enter the second brand name:</label>
        <input type="text" id="brand2" name="brand2" required>
        <br><br>
        <button type="submit">Compare Prices</button>
    </form>
</body>
</html>
```

```
<h3>Results for {{ brand1 }}:</h3>
<p>Product link: <a href="{{ link1 }}">{{ product1 }}</a></p>
<p>Price: ${{ price1 }}</p>

<h3>Results for {{ brand2 }}:</h3>
<p>Product link: <a href="{{ link2 }}">{{ product2 }}</a></p>
<p>Price: ${{ price2 }}</p>

<p><strong>{{ result }}</strong></p>
```