BechaKena: Get Your Desired Books at Your Cheapest Rate!

Taneem Ahmed
Student ID: 2013102042
email: taneem.ahmed@nothsouth.edu

June 9, 2023

I. Introduction

BechaKena is your one-stop destination for hassle-free book shopping. Our revolutionary system allows you to compare prices for the same books across multiple online book vendor websites. With a user-friendly interface and comprehensive book information, finding the best deals has never been easier. Join our community of satisfied book lovers and enjoy the convenience and affordability of BechaKena today!

No longer will you need to spend endless hours navigating through numerous websites, meticulously comparing prices. With BechaKena, you can now access a wide range of book vendors all in one place. Our advanced system scours the internet, gathering prices from various online bookstores, and presents you with a comprehensive comparison. This means that you can easily find the lowest price for your desired book, saving both your time and money. Whether you're an avid reader, a student searching for textbooks, or simply looking to expand your literary collection, BechaKena is here to make your book shopping experience seamless and convenient. With our user-friendly interface, you can effortlessly search for any book and instantly see the prices offered by different vendors. We provide detailed information about each book, including reviews and ratings, so you can make an informed decision before making a purchase. Not only does BechaKena save you time and money, but it also ensures a secure and reliable shopping experience.

II. Technologies used

The system will include a ReactJS UI for submitting queries to the Django Python framework-developed API. The system will use a Django scraping tool: BeautifulSoup to make the data extraction process easier. I utilized the Selenium tool in conjunction with BeautifulSoup for handling dynamic web pages. The Scraped data stored into a MySQL Database.

III. Procedure

Initially, I developed a Django model to create the necessary database. Then, I proceeded to explore the Rokomari webpage and conducted an inspection to identify the HTML tags containing the relevant information such as book titles, authors, rokomari ids, recent prices, and book images. Based on this analysis, I created a scraper specifically designed to extract the desired data. I stored this scraped information in both JSON endpoints and a MySQL database, effectively acquiring the book data from Rokomari. Subsequently, I visited the Boibitan page and applied a similar process to extract the data from there as well. Next, I developed a JSX file for the frontend component to display the JSON responses. The file incorporated two key features: book search by title and book search by author, which dynamically presented the corresponding search results. Finally, I implemented the necessary routing to ensure a seamless user interface that showcased all the data scraped from Rokomari and Boibitan. This allowed users to conduct searches based on the aggregated data for a comprehensive book shopping experience.

IV. Problems I faced

During the data scraping process, I encountered dynamic pages where my scraper was unable to extract the images. Instead, it retrieved the default book icon. Accurately extracting the desired information from web pages becomes challenging when encountering inconsistent or irregular data formatting. Dealing with nested elements added to the difficulties faced. Moreover, the frequent updates to website design, layout, or underlying structure posed a risk of breaking existing scraping scripts. Consequently, it was crucial to consistently maintain and adapt the scraping code to ensure its compatibility and effectiveness in light of these changes.

V. How I solve the Problems

When faced with dynamic pages, I employed the Selenium tool to automate web browsing, allowing sufficient time (3 seconds) for the page to load and retrieve the desired images from the websites. Moreover, to mitigate issues caused by data changes, it is advisable to implement a scheduler that runs the scraper at specified intervals as instructed by the developer. This ensures that the data remains up-to-date and accurate over time.

VI. Important Links

GitHub Link: https://github.com/TaneemAhmed2013102/BechaKena$_{s}craper$
BechaKena API: https://taneem.winexsoft.com/bookshelf

VII. Snap Shots

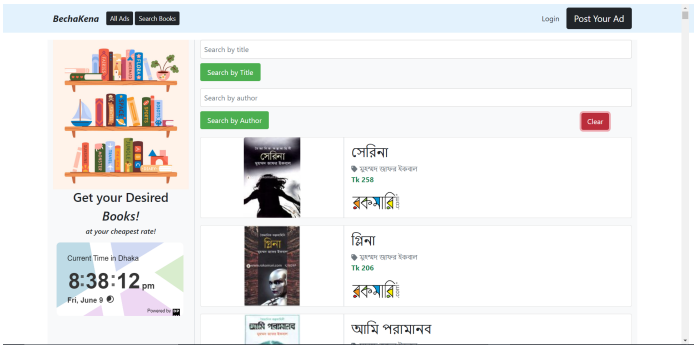

Fig. 1. WebPage

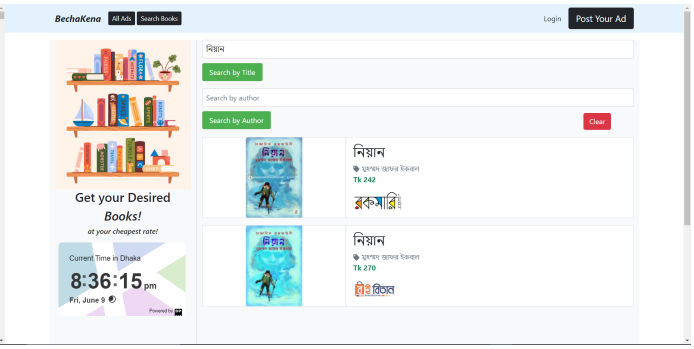

Fig. 2. Niyan Book Search

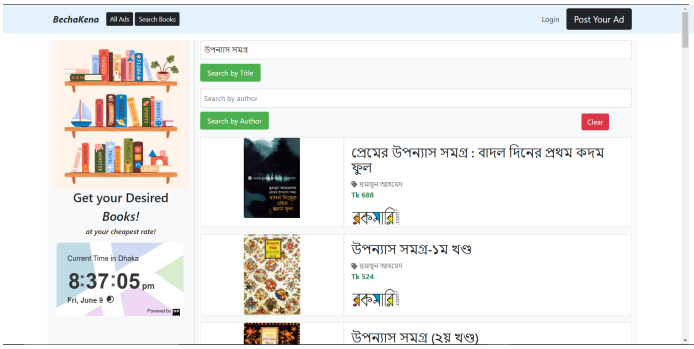

Fig. 3. Uponnash Somogro Search

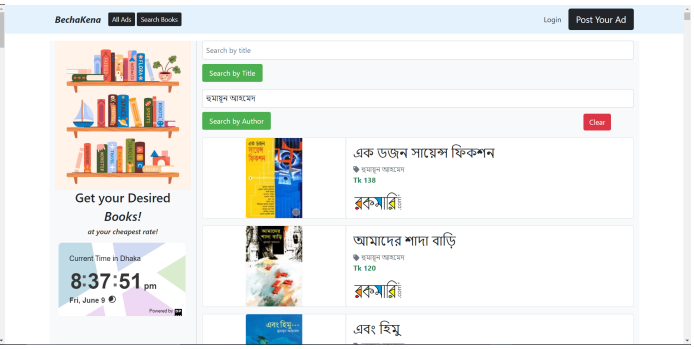

Fig. 4. Humayun Author Search

VIII. Conclusion

BechaKena is a recommended platform for those seeking affordable book purchases. With its intuitive interface and powerful price comparison system, BechaKena helps users find the best deals, saving both time and money. By utilizing advanced scraping techniques, the platform ensures access to up-to-date information from multiple book vendors. Shop with confidence and discover great books at competitive prices with BechaKena.

IX. Appendix

The code for all the above mentioned Procedure are given below.

Code for BookShelf Model:

```
class BookShelf(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=255)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    author = models.CharField(max_length=255)
    rokomari_id = models.CharField(max_length=255)
    boibitan_name = models.CharField(max_length=255)
    image_path = models.CharField(max_length=255, default='')
    company_logo = models.CharField(max_length=255, default='')

    def _str_(self):
        return self.title
```

Code for Bookshelf Request:

```
def books_shelf(request):
    books = BookShelf.objects.all()
    book_data = []

    for i in books:
        book_data.append({
            'id': i.id,
            'title': i.title,
            'price': float(i.price),
            'author': i.author,
            'rokormari_id': i.rokomari_id,
            'image_path': i.image_path,
            'company_logo': i.company_logo
        })

    return JsonResponse({'books': book_data})
```

Scrapper Code for Rokomari:

```
def scrape_rokomari(request):

    url = 'https://www.rokomari.com/book/author/930/rabindranath-tagore?ref=mm_p3&page=3'
    response = requests.get(url)

    if response.status_code == 200:

        soup = fetch_and_parse_with_selenium(url)
        book_list = soup.find_all('div', {'class': 'book-list-wrapper'})
        # print(book_list)
        # return HttpResponse(book_list)
        book_data = []

        for book in book_list:
            title = book.find('h4', {'class': 'book-title'}).text.strip()
            author = book.find('p', {'class': 'book-author'}).text.strip()
            price = book.find('p', {'class': 'book-price'}
                              ).find('span').text.strip()
            image_path = book.find(
                'div', {'class': 'book-img'}).find('img').get('src')
            rokomari_id = book.find(
                'div', {'class': 'cart-btn-area'}).find('button').get('product-id')

            book_data.append({
                'title': title,
                'price': convertStringToInteger(price),
                'author': author,
                'image_path': image_path,
                'rokomari_id': rokomari_id,
                'company_logo': 'https://www.rokomari.com/static/200/images/rokomari_logo.png'
            })
        insert_into_shelf(book_data)
```

```
        return JsonResponse({'books': book_data})
    else:
        return HttpResponse(f"Failed to fetch data. Status code: {response.
            status_code}")
```

Scrapper Code for Boibitan:

```
def scrape_boibitan(request):

    url = 'https://boibitan.com/author/gurudev-rabindranath-tagore-0ep4a?page=2'
    response = requests.get(url)

    if response.status_code == 200:

        soup = fetch_and_parse_with_selenium(url)
        book_list = soup.find_all('div', {'class': 'product-default inner-quickview
            inner-icon pl-3 pr-3'})
        book_data = []

        for book in book_list:
            title = book.find('h3', {'class': 'product-title'}).text.strip()
            author = book.find('div', {'class': 'category-list'}).text.strip()
            price = book.find('span', {'class': 'product-price'}).text.strip()
            image_path = book.find('figure').find('img').get('src')
            boibitan_name = book.find('h3', {'class': 'product-title'}).text.strip()


            book_data.append({
                'title': title,
                'price': convertStringToInteger(price),
                'author': author,
                'image_path': image_path,
                'company_logo': "https://boibitan.com/public/uploads/all/63
                    cd26644307b.jpeg",
                'boibitan_name': boibitan_name
            })
        insert_into_shelf(book_data)
        return JsonResponse({'books': book_data})
    else:
        return HttpResponse(f"Failed to fetch data. Status code: {response.
            status_code}")
```

Selenium Code for Dynamic Pages:

```
def fetch_and_parse_with_selenium(url, browser='chrome'):

    if browser.lower() == 'chrome':
        options = ChromeOptions()
        # options.add_argument('--headless')
        driver = webdriver.Chrome(options=options)
    elif browser.lower() == 'firefox':
        options = FirefoxOptions()
        options.add_argument('--headless')
        driver = webdriver.Firefox(options=options)
    else:
```

```
        raise ValueError('Invalid browser specified')

    driver.get(url)

    last_height = driver.execute_script("return document.body.scrollHeight")

    while True:
        driver.execute_script(
            "window.scrollTo(0, document.body.scrollHeight);")

        time.sleep(3)

        new_height = driver.execute_script("return document.body.scrollHeight")
        if new_height == last_height:
            break
        last_height = new_height

    html = driver.page_source

    driver.quit()

    soup = BeautifulSoup(html, 'html.parser')
    return soup
```

Code for Insertion:

```
def insert_into_shelf(books):

    for book in books:
        title = book.get('title')
        price = book.get('price')
        author = book.get('author')
        rokomari_id = book.get('rokomari_id')
        boibitan_name = book.get('boibitan_name')
        image_path = book.get('image_path', '')
        company_logo = book.get('company_logo','')

        if rokomari_id:
            BookShelf.objects.update_or_create(
                rokomari_id=rokomari_id,
                defaults={
                    'title': title,
                    'price': price,
                    'author': author,
                    'image_path': image_path,
                    'company_logo':company_logo
                }
            )
        elif boibitan_name:
            BookShelf.objects.update_or_create(
                boibitan_name=boibitan_name,
                defaults={
                    'title': title,
                    'price': price,
                    'author': author,
                    'image_path': image_path,
                    'company_logo':company_logo
```

```
32                      }
33                  )
```