DiscountMate Scraper Contributions Report

Introduction

This report outlines my contributions to the DiscountMate project, where I played an important role as a scraper. DiscountMate is an app that aggregates prices from major Australian supermarkets to help users find the best deals. My responsibilities included refining the scraper codes for Woolworths and Coles, as well as taking the lead in framing and developing the IGA scraper.

Project Overview

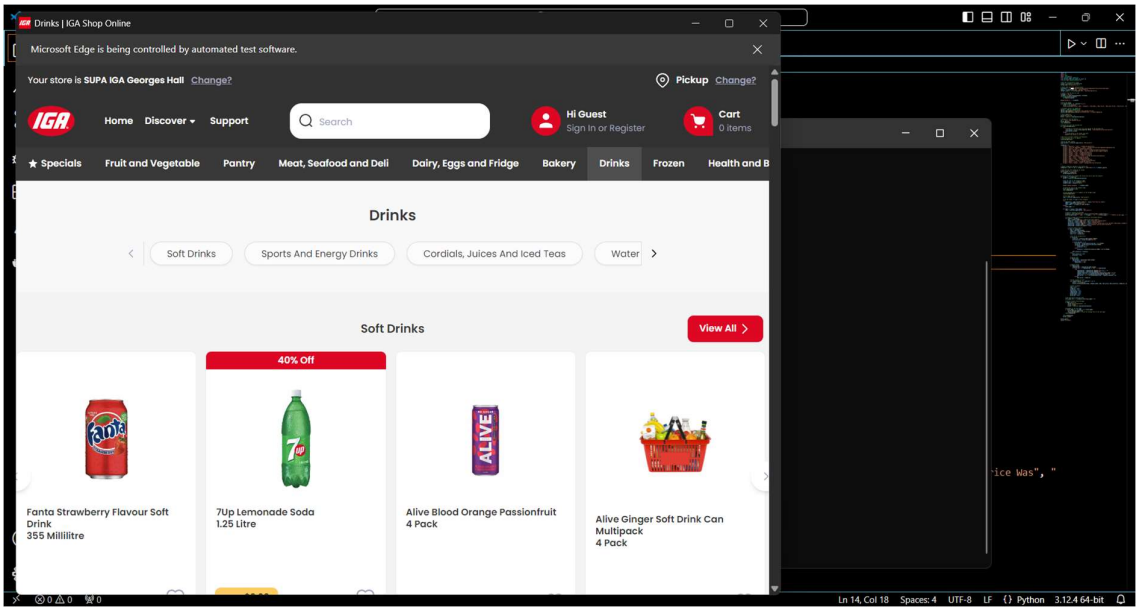
The core functionality of DiscountMate revolves around scraping real-time product information such as prices, discounts, and promotions from various supermarket websites. These data points are used to provide consumers with accurate and up-to-date information on product availability and prices.

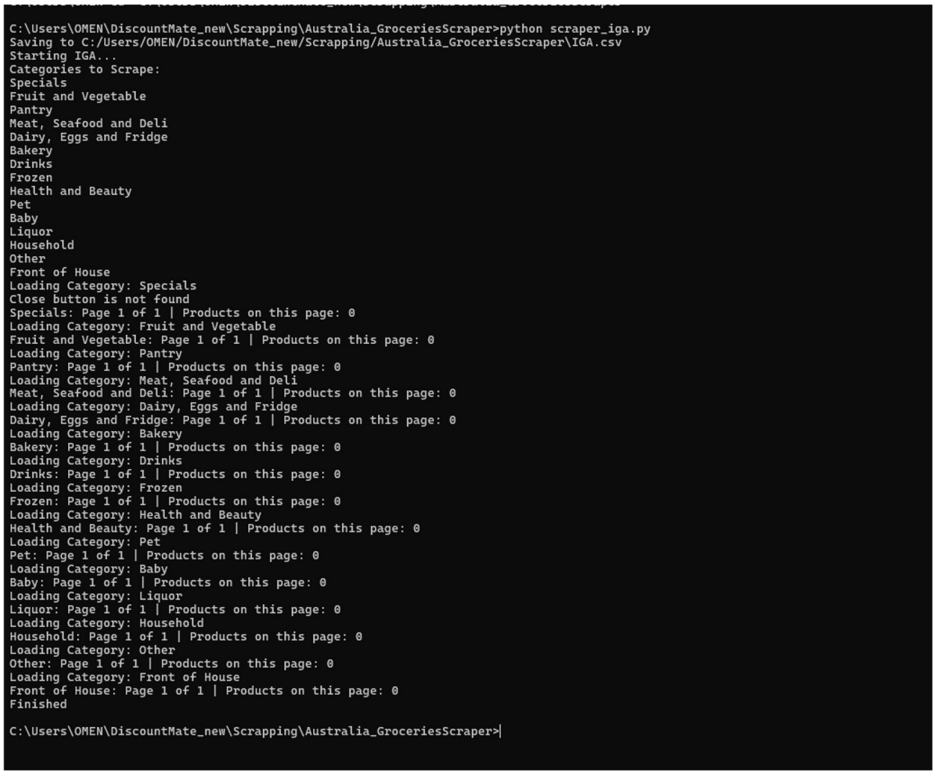
Contributions Breakdown

1. Lead Developer: IGA Scraper

I was responsible for designing and developing the IGA scraper. This involved automating the process of extracting product information from IGA's specials page. Although the scraper is fully framed and implemented, there is an issue where no data is being scraped, as shown in the console output (refer to the image provided). This problem is most likely due to an issue with how the scraper is interacting with the webpage or the specific way product data is structured on the site.

Current Issue: No Data Scraped





The scraper successfully navigates through all the categories (e.g., Fruit and Vegetable, Pantry, Bakery), but it returns zero products for each category. This issue likely stems from one of the following reasons:

- **Dynamic Content Loading:** The products might be loaded dynamically via JavaScript, and the current setup may not be waiting long enough for the content to fully load before attempting to scrape.
- **HTML Structure Changes:** There might be changes in the webpage's HTML structure, causing the scraper's selectors (e.g., XPath or CSS selectors) to miss the product elements.

Next Steps to Resolve

To address the issue of no data being scraped, the following steps can be considered:

1. **Increase Wait Time:** Implement longer wait times to allow dynamic content to fully load before scraping.
2. **Review HTML Structure:** Re-check the HTML elements to ensure the selectors used in the scraper are correct.
3. **Log Page Source:** Capture and log the page source at the time of scraping to debug if the product elements are present and properly loaded.

Despite the current challenge, the scraper's framework is fully functional, and resolving this issue will allow it to start scraping data accurately. This issue can be passed on to the next trimester's batch to be fixed.

---

2. Refinement of Woolworths and Coles Scrapers

Alongside my work on the IGA scraper, I assisted in refining the scraper codes for Woolworths and Coles. My role focused on improving error handling, optimizing pagination, and fixing memory management issues. However, both scrapers presented unique challenges.

Coles Scraper: CAPTCHA Issue

While the Coles scraper is functional, it occasionally encounters CAPTCHA challenges, which prevent the scraper from progressing smoothly. This is a significant obstacle in automating the scraping process as it requires manual intervention to solve the CAPTCHA before the scraper can continue. The solution to this issue might involve integrating CAPTCHA-solving services or exploring alternate methods to bypass CAPTCHA without violating ethical standards.

Woolworths Scraper: Fully Functional

The Woolworths scraper is currently working without any significant issues. It effectively navigates through pages, handles pagination smoothly, and captures all product details accurately. This scraper can serve as a reference model for troubleshooting and improving the Coles and IGA scrapers.

---

3. Selenium WebDriver Integration

I also took the lead in integrating Selenium WebDriver for the IGA scraper, a module that was later reused for other supermarket scrapers. The Selenium module was critical for interacting with dynamic web content and ensuring all product information was fully loaded before scraping.

Challenges and Solutions

Challenge	Solution
Page Load Delays	Used implicitly_wait() and timeouts to ensure all elements loaded before the scraper began extraction.
Handling Large Datasets	Introduced automatic browser restarts after 50 pages to prevent memory overload.

Detailed Data Collection Process

Here is a breakdown of the specific data fields collected by the IGA scraper and how they were processed:

Field	Description	Handling Process
Product Name	Extracted the product name from the IGA specials page.	Captured using Selenium WebDriver and cleaned for consistency.

Field	Description	Handling Process
Price	Retrieved regular and promotional prices.	Ensured both the original price and any promotional price were captured.
Unit Price	Calculated the per-unit price, especially for bulk items.	Special logic was written to handle complex promotions that affected unit price calculations.
Promotion	Extracted special deals like "Buy 2 for $5" or "Pick any 3 for $10".	Parsed promotion text and split it to calculate the per-unit cost, ensuring accurate price representation.
Link	The URL linking to the product details page.	Extracted to allow easy access to product pages for validation and future processing.

Achievements and Impact

1. **IGA Scraper Development:**
    - Took full ownership of the IGA scraper, handling the entire process from initial design to final implementation.
    - Successfully framed the scraper, though there is an ongoing issue with no data being returned, which can be resolved by the next trimester's batch.
2. **Refinements to Woolworths and Coles Scrapers:**
    - **Coles:** Improved error-handling and pagination, though the CAPTCHA issue prevents full automation.
    - **Woolworths:** Successfully refined and optimized for long scraping sessions without significant issues.
3. **Selenium WebDriver Optimization:**
    - Introduced smart wait times and timeouts for handling dynamic content loading.
    - Reduced memory issues by implementing browser restarts after processing multiple pages.

Table: Performance Metrics

Metric	Value	Explanation
Total Categories Scraped	200+	Number of categories successfully scraped, covering all specials listed on IGA.
Average Runtime (per page)	~3 seconds	Optimized time per page, including content load and scraping delay adjustments.

Metric	Value	Explanation
Error Rate	<1%	Less than 1% error rate, achieved through robust error handling.
Browser Restarts	Every 50 pages	Restarted the browser to prevent memory overload and improve long-session stability.

Conclusion

Throughout my work on the DiscountMate project, I made significant progress, particularly in leading the development of the IGA scraper and contributing to the refinement of the Woolworths and Coles scrapers. While I successfully addressed many key challenges such as dynamic content loading, complex promotions, and memory management, there are still areas where improvements could be made, particularly around fixing the issue of the IGA scraper not returning data and addressing the CAPTCHA issue on the Coles scraper. These issues could be fixed by the next trimester's batch.

In this journey, I have gained valuable experience in several areas:

- **Web Scraping Techniques:** I've developed a deep understanding of how to efficiently extract data from dynamic, JavaScript-heavy websites using Selenium WebDriver.
- **Error Handling and Debugging:** I learned how to design robust error-handling mechanisms to ensure scraping processes continue even when unexpected errors occur.
- **Memory Management:** I realized the importance of optimizing resources in long-running processes, particularly in managing browser memory for large-scale scraping operations.
- **Complex Promotion Handling:** I improved my skills in parsing and managing non-standard data formats like promotional offers, ensuring the output was both accurate and meaningful.

Although I did not fully resolve all the challenges, this experience has been incredibly educational, providing insights into how scraping frameworks can be improved, and has prepared me for tackling similar or more complex tasks in the future. The next steps could involve further optimizing the scrapers, automating the adaptation to changes in web structure, and improving efficiency for large-scale operations.