Project Overview

PropertyGuruScraper is a Python-based web scraping project designed to extract real estate listings from PropertyGuru Singapore. The scraper automates browser interaction using Selenium to handle dynamically loaded JavaScript content. It collects structured information such as title, price, location, and property features across multiple pages, and stores the results in a CSV file.

This scraper was developed as part of a technical take-home challenge, with the goal of demonstrating proficiency in browser automation, data extraction, and handling web anti-bot measures.

Link

www.github.com/stevenpyae/propertyguru_scraper

Technology Stack

Component	Description
Python 3.x	Core programming language
Selenium	Browser automation (used with ChromeDriver)
ChromeDriver	Automates interaction with the Chrome browser
csv	Writing structured data to CSV files
random, time	Human-like delays to avoid detection

Design & Approach

1. Session Setup
The scraper creates a fresh browser session for each page to mimic a new user and minimize anti-scraping detection. A rotating list of user-agent strings is used to spoof different browsers.

2. Web Driver Configuration
The `navigator.webdriver` property is overridden to hide automation flags from the browser environment.

3. Data Extraction
For each property listing, the following fields are extracted:

- Title
- Listing URL
- Price
- Location
- Bedrooms
- Bathrooms
- Floor Area
- Price per sqft

4. Pagination
The scraper uses the "Next" pagination button to navigate across pages until the target number of listings is reached (e.g., 200 records).

5. Record-Based Limit
Instead of fixed page scraping, the loop terminates once the total number of listings reaches the defined threshold, improving flexibility.

6. Output Format
All results are written to `output/propertyguru_listings.csv` with UTF-8 encoding.

✅ Output

- Over 200 property listings collected
- Output file: `output/propertyguru_listings.csv`
- Format:

```
Title,URL,Price,Location,Bedrooms,Bathrooms,Floor Area,Price per sqft
"640 Bedok Reservoir Road", https://..., "$980,000", "Bedok", "4",
"2", "1550 sqft", "$632.26"
```

🧪 Challenges & Solutions

Challenge	Solution
Site uses JavaScript to load content	Used Selenium to render dynamic HTML and wait for element loading
Pages don't expose a "last" button	Used loop to follow "Next" links until `None` is returned
Some listings had missing fields	Applied default values (e.g., "N/A") with defensive parsing
Risk of IP bans and blocks	Random delays and rotating user-agent headers used per session

Key Takeaways

- Selenium is effective for scraping JavaScript-rendered content.
- Proper use of user-agents and delays can mitigate anti-bot detection.
- It’s critical to implement fault tolerance when scraping from real-world sites.
- Modular script design helps adapt scrapers to similar page structures quickly.

Potential for Improvements

- Brute-force way will take too much time scraping to get a large dataset. There is a need to discover more ethical ways to bypass bot-scraping protections such as exploring Scrapy with Splash
- Adding python test scripts to avoid manual testing of the results fetched, validate selectors and data formats