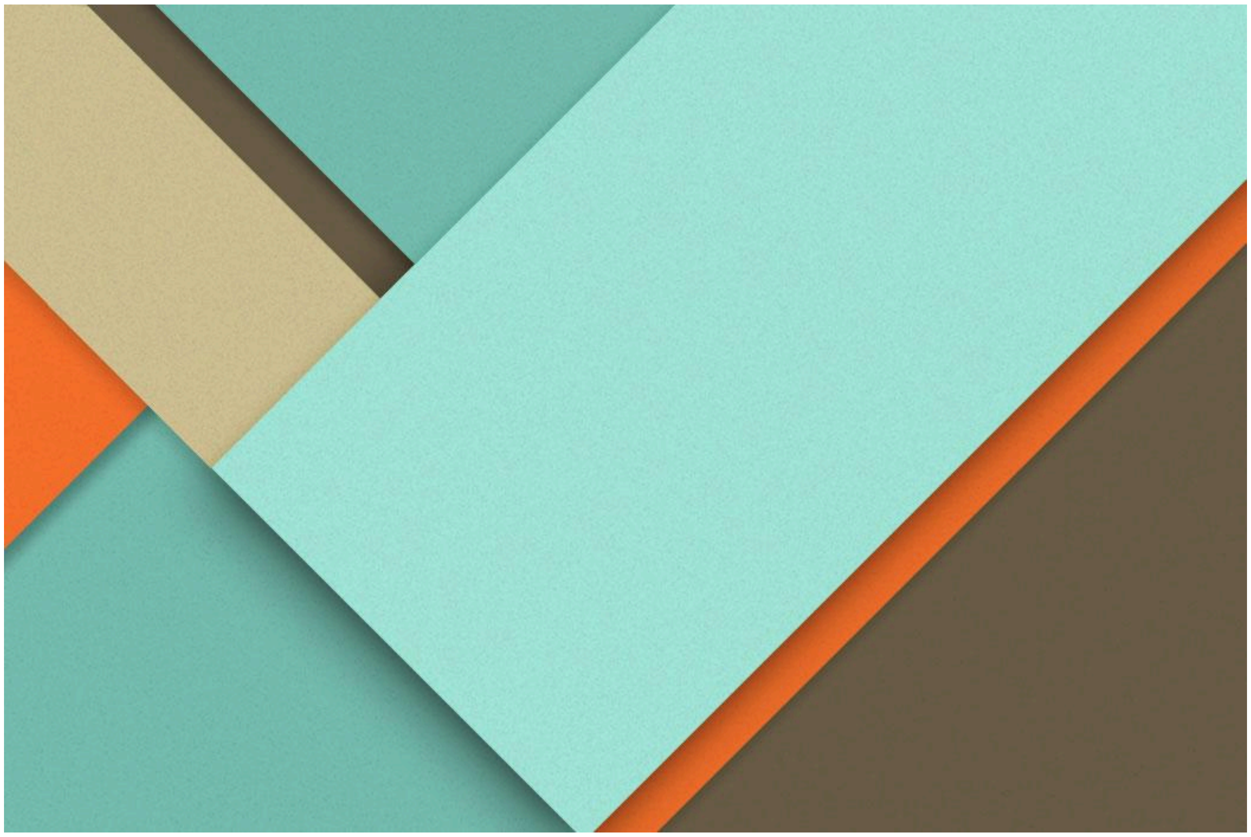

Automated CGA Data Extractor

Shalini Chaudhary
linishachaudhary@gmail.com

Overview

This project automates the extraction of financial reports from the Controller General of Accounts (CGA) website using Selenium WebDriver. It collects monthly, finance, appropriation, and accounts-at-a-glance reports and saves the data in a structured JSON file.

1. Opens the CGA website using a headless Chrome browser (runs in the background).
2. Finds and selects dropdown options (months, years) to get pdf reports for the latest years . (Bcz extraction for all possible years would have taken lot of time)
3. Clicks on links, extracts tables, and fetches pdf URLs inside the reports for direct access to embedded information.
4. Handles missing data gracefully (logs "No Data Available" instead of crashing).
5. Saves all extracted data into a JSON file for easy access and analysis.

Features

1. Automated navigation – No manual clicking needed.
2. Dropdown selection handling – Works for different years & months
3. Headless browsing – Runs faster without opening a window.
4. Error handling & logging – Avoids crashes and logs issues.
5. Optimized performance – Blocks images for speed and reduces browser loads.

Approach

I. Initialize Web Scraper

Set up Selenium WebDriver to automate browser interactions.

Use a headless browser for faster execution.

Configure settings to disable images for better performance

II. Navigate to Target Webpage

Open the CGA website.

Wait for the page to fully load before interacting.

III. Interact with Web Elements

Locate dropdown menus for selecting year and month.

Choose the latest two years and available months dynamically.

Click Go buttons to generate reports pages for each selection.

IV. Extract Data from Reports

Capture the URL of the generated report page after selection.

Identify tables inside the report page containing PDF links.

Retrieve report names, URLs for direct access to embedded information.

Convert relative URLs to absolute URLs for consistency.

V. Handle Errors & Store Data

Detects and logs missing data or unavailable reports.

Manage pop-ups, timeouts, and stale elements to prevent failures.

Save extracted data into JSON format .

Future Enhancements

1. Extract historical data for more than just the latest two years.
2. Use multi-threading or parallel processing to extract data faster.
3. Instead of JSON, store data in databases for better analysis.

Results

Repo link: https://github.com/linisha04/Automated-CGA-Data-Extractor.git

Result Json file link:
https://github.com/linisha04/Automated-CGA-Data-Extractor/blob/main/all_reports.json

How to run the code

Step1: Open the terminal and write

```
(venv) linisha@MacBook-Air web_scraping % python3 -m venv venv
(venv) linisha@MacBook-Air web_scraping % source venv/bin/activate
(venv) linisha@MacBook-Air web_scraping % pip install -r req.txt
```

This will create virtual environment and install all necessary requirements to run the code

Step2: Now write , `python3 main.py`

```
(venv) linisha@MacBook-Air web_scraping % python3 main.py
```

Step3: json files and scraper.log will be created .

all_reports.json **:** contains all the scraped data the provides direct access to embedded information.

Finance_data.json contains direct access to embedded information for Monthly accounts.

Appropriation_data.json : contains direct access to embedded information for Appropriation accounts.

account_at_glance.json : contains direct access to embedded information for account_at_glance accounts.

gfsm_data.json: contains direct access to embedded information for gfsm_data accounts.

monthly_data.json: contains direct access to embedded information for monthly_data accounts. But could not scrape this part.

Scraper.log : contains execution details of the code.