IMDb Movie Rating Scraper– Post Project Report

Automated Movie Data Extraction and Reporting System

Author: PERCI MAHIBA M(Team 1),
Project Type: Individual Project

EXECUTIVE SUMMARY:

The IMDb Movie Rating Scraper project demonstrates the development of an automated web scraping system using Python and Selenium to extract movie information from the IMDb Top 250 Movies list. The system dynamically loads web content, collects essential movie attributes such as title, release year, IMDb rating, and ranking, and exports the extracted data into a structured CSV file.

PROJECT OVERVIEW:

Objective:

To develop an automated Python-based tool that:

- Scrapes movie data from IMDb Top 250 Movies.
- Extracts movie title, release year, IMDb rating, and ranking.
- Handles dynamic JavaScript-loaded web pages.
- Stores extracted data in CSV format.
- Supports headless mode for background execution.

Scope:

- **Platform Used:** IMDb (Top 250 Movies section)
- Dynamic data extraction using Selenium WebDriver
- Structured data storage using CSV files
- Execution in both normal and headless browser modes
- Designed for educational and analytical purposes

Key Features:

- **Dynamic Web Scraping:** Extracts data from JavaScript-rendered IMDb pages.
- **Automated Browser Control:** Uses Selenium to simulate real user behavior.
- **Structured Output:** Saves clean and organized movie data into CSV files.
- **Headless Mode Support:** Allows execution without opening a browser window.
- **Expandable Architecture:** Can be extended to scrape genres, cast, or reviews.

TECHNICAL IMPLEMENTATION:

Architecture:

The project follows this robust web automation pipeline:

1. Initialize Chrome WebDriver using WebDriver Manager.
2. Navigate to the IMDb Top 250 Movies webpage.
3. Wait for JavaScript content to fully load.
4. Locate and extract required movie details using DOM selectors.
5. Process and structure the extracted data.
6. Export the data into a CSV file.

Technology Stack:

- **Programming Language:** Python 3.x
- **Web Automation:** Selenium
- **Driver Management:** WebDriver Manager
- **Data Handling:** Pandas
- **Timing Control:** time module
- **Output Format:** CSV

Performance Metrics:

- **Data Extracted:** Top 250 movies per execution
- **Execution Time**: 5–10 seconds (depending on internet speed)
- **Success Rate:** High (dependent on IMDb page structure)
- **Output Format**: Structured CSV file

SOURCE CODE IMPLEMENTATION (Screenshot):

main.py imdb_top_250.csv

main.py > ...

```python
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
from webdriver_manager.chrome import ChromeDriverManager
import pandas as pd
import time


# Enable Headless Mode

options = webdriver.ChromeOptions()
options.add_argument("--headless")        # Run without UI
options.add_argument("--disable-gpu")
options.add_argument("--window-size=1920,1080")
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safa
options.add_argument("--disable-blink-features=AutomationControlled")
options.add_experimental_option("excludeSwitches", ["enable-automation"])
options.add_experimental_option('useAutomationExtension', False)


# Setup ChromeDriver automatically

driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()),
    options=options
)


# Open IMDb Top 250 page

driver.get("https://www.imdb.com/chart/top/")
```

main.py imdb_top_250.csv

main.py > ...

```python
driver.get("https://www.imdb.com/chart/top/")

time.sleep(10)

# Debug: Check page title and print available selectors
print("Page title:", driver.title)
print("Page URL:", driver.current_url)

# Try different selectors to find movies
movies = driver.find_elements(By.CSS_SELECTOR, "li.ipc-metadata-list-summary-item")
print(f"Found {len(movies)} movies with first selector")

# If first selector doesn't work, try alternative selectors
if len(movies) == 0:
    movies = driver.find_elements(By.CSS_SELECTOR, "li[data-testid='chart-list-item']")
    print(f"Trying alternative selector: Found {len(movies)} movies")

if len(movies) == 0:
    movies = driver.find_elements(By.XPATH, "//li[contains(@class, 'ipc-metadata')]")
    print(f"Trying XPATH selector: Found {len(movies)} movies")

# Extract movie data

data = []

for movie in movies:
    try:
        # Extract title
        title_elem = movie.find_element(By.CSS_SELECTOR, "h3.ipc-title__text")
        title = title_elem.text.split('\n')[0].strip()  # Get only the title part

        # Extract rating - updated selector
        rating_elem = movie.find_element(By.CSS_SELECTOR, "span.ipc-rating-star--rating")
```

main.py imdb_top_250.csv

main.py > ...

```python
    try:
        # Extract title
        title_elem = movie.find_element(By.CSS_SELECTOR, "h3.ipc-title__text")
        title = title_elem.text.split('\n')[0].strip()  # Get only the title part

        # Extract rating - updated selector
        rating_elem = movie.find_element(By.CSS_SELECTOR, "span.ipc-rating-star--rating")
        rating = rating_elem.text.split('\n')[0].strip()  # Get only the rating part

        data.append({
            "Movie Name": title,
            "IMDb Rating": rating
        })
    except Exception as e:
        print(f"Error extracting movie data: {e}")
        continue


# Save to CSV

df = pd.DataFrame(data)
df.to_csv("imdb_top_250.csv", index=False)


# Close browser

driver.quit()
print("Scraping completed successfully in headless mode!")
```

Extracted Job Result In CSV:

main.py | imdb_top_250.csv

imdb_top_250.csv > data

```
Movie Name,IMDb Rating
The Shawshank Redemption,9.3
The Godfather,9.2
The Dark Knight,9.1
The Godfather Part II,9.0
12 Angry Men,9.0
The Lord of the Rings: The Return of the King,9.0
Schindler's List,9.0
The Lord of the Rings: The Fellowship of the Ring,8.9
Pulp Fiction,8.8
"The Good, the Bad and the Ugly",8.8
The Lord of the Rings: The Two Towers,8.8
Forrest Gump,8.8
Fight Club,8.8
Inception,8.8
Star Wars: Episode V - The Empire Strikes Back,8.7
The Matrix,8.7
Goodfellas,8.7
Interstellar,8.7
One Flew Over the Cuckoo's Nest,8.6
Se7en,8.6
It's a Wonderful Life,8.6
The Silence of the Lambs,8.6
Seven Samurai,8.6
Saving Private Ryan,8.6
The Green Mile,8.6
City of God,8.6
Life Is Beautiful,8.6
Terminator 2: Judgment Day,8.6
Star Wars: Episode IV - A New Hope,8.6
Back to the Future,8.5
Spirited Away,8.6
The Pianist,8.5
```

main.py | imdb_top_250.csv

imdb_top_250.csv > data

```
Movie Name,IMDb Rating
The Third Man,8.1
Maharaja,8.3
Jai Bhim,8.6
The Terminator,8.1
The Sound of Music,8.1
The Big Lebowski,8.1
A Silent Voice: The Movie,8.2
The Best Years of Our Lives,8.1
The Seventh Seal,8.1
Room,8.1
Before Sunset,8.1
Hotel Rwanda,8.1
Platoon,8.1
Chainsaw Man - The Movie: Reze Arc,8.4
The Incredibles,8.0
The Exorcist,8.1
Hachi: A Dog's Tale,8.1
Rush,8.1
The Wizard of Oz,8.1
Stand by Me,8.1
The Iron Giant,8.1
The Passion of Joan of Arc,8.1
My Father and My Son,8.2
The Battle of Algiers,8.1
The Handmaiden,8.1
Network,8.1
Gangs of Wasseypur,8.2
Drishyam,8.2
The Grapes of Wrath,8.1
To Be or Not to Be,8.1
Demon Slayer: Kimetsu no Yaiba - The Movie: Mugen Train,8.2
Andhadhun,8.2
```

KEY ACHIEVEMENTS:

Technical Accomplishments:

- Successfully scraped dynamic IMDb web content.
- Implemented headless browser automation.
- Achieved reliable and repeatable data extraction.
- Generated structured datasets suitable for analysis.

Process Automation Benefits:

- Eliminates manual data collection.
- Saves time and effort.
- Enables quick access to updated movie ratings.

LIMITATIONS AND CONSTRAINTS:

Current Limitations:

- Dependent on IMDb's current webpage structure.
- No database integration (file-based storage only).
- Cannot scrape private or restricted content.

Resource Constraints:

  - Requires stable internet connectivity.
  - Depends on Google Chrome browser availability.
  - Script maintenance required if IMDb changes layout.

PROJECT IMPACT AND VALUE:

Immediate Benefits:

- Fast access to movie rating data.
- Simplifies data collection for analysis and reporting.
- Useful for academic projects and learning automation**.**

Long-term Potential:

- Integration with recommendation systems.
- Dashboard and visualization development.
- Database and cloud storage integration.
- Scheduled and large-scale automation.

CONCLUSION:

The IMDb Movie Rating Scraper project successfully demonstrates the use of Python and Selenium for web automation and data extraction. By converting dynamic web data into structured datasets, the project showcases essential skills in automation engineering and data handling.

With further enhancements such as database storage and advanced analytics, the system can evolve into a comprehensive movie intelligence platform.