**Mini Project 2 Report**

**IMDb Movie Rating Scraper**

**1. Abstract**

The IMDb Movie Rating Scraper is a Python-based automation tool designed to extract and organize movie data from the IMDb Top 250 Movies list. Using Selenium with Chrome WebDriver, the scraper dynamically handles JavaScript-rendered content and retrieves movie details such as title, release year, IMDb rating, and ranking. The extracted data is stored in CSV format, which can be further used for movie trend analysis, recommendation systems, or personal film databases.

**2. Objectives**

* To automate the extraction of structured movie data from IMDb.
* To handle dynamically loaded web pages using Selenium.
* To store movie data in a structured format (CSV) for further analysis.
* To create an extendable scraper that can be scaled to extract additional details such as cast, genre, and reviews.

**3. Problem Statement**

Manual collection of IMDb movie data is time-consuming, error-prone, and inefficient. Since IMDb’s content is dynamically loaded using JavaScript, traditional scraping libraries like BeautifulSoup are limited. A robust, automated tool is needed to extract real-time, accurate, and structured movie data efficiently.

**4. Methodology**

1. **Setup and Libraries**
   * Python is used as the programming language.
   * Required libraries: Selenium, pandas, time, and webdriver\_manager.
2. **Web Scraping Process**
   * Launch Chrome WebDriver using webdriver\_manager.
   * Navigate to IMDb’s Top 250 Movies page.
   * Extract movie details: title, release year, ranking, and IMDb rating.
   * Handle dynamic loading of elements using Selenium wait functions.
3. **Data Storage**
   * Store extracted movie details in a pandas DataFrame.
   * Export the DataFrame into a CSV file for easy accessibility and analysis.
4. **Optional Features**
   * Implement headless browsing for background execution.
   * Extend scraper to visit individual movie pages for more details.

**5. Features**

* Dynamic scraping of IMDb Top 250 movies.
* Extracts rank, title, year, and IMDb rating.
* Saves structured data in CSV format.
* Supports headless execution.
* Extendable for scraping more movie attributes.

**6. Technologies Used**

* **Python** – Core programming language.
* **Selenium** – For dynamic webpage scraping.
* **pandas** – For organizing and exporting data.
* **webdriver\_manager** – For automatic ChromeDriver management.
* **CSV** – Output format for structured storage.

**7. Implementation (Code)**

**Python Code: IMDb Movie Rating Scraper**

# IMDb Movie Rating Scraper

# Importing required libraries

import time

import pandas as pd

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.service import Service

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.chrome.options import Options

# Step 1: Configure Chrome options (Headless mode optional)

chrome\_options = Options()

# Uncomment below line for headless execution

# chrome\_options.add\_argument("--headless")

# Step 2: Launch the WebDriver

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

# Step 3: Open IMDb Top 250 Movies page

url = "https://www.imdb.com/chart/top/"

driver.get(url)

# Step 4: Allow page to load

time.sleep(3)

# Step 5: Extract movie details

movies = driver.find\_elements(By.CSS\_SELECTOR, "tbody.lister-list tr")

movie\_data = []

for movie in movies:

rank = movie.find\_element(By.CSS\_SELECTOR, ".titleColumn").text.split(".")[0]

title = movie.find\_element(By.CSS\_SELECTOR, ".titleColumn a").text

year = movie.find\_element(By.CSS\_SELECTOR, ".titleColumn span").text.strip("()")

rating = movie.find\_element(By.CSS\_SELECTOR, ".imdbRating strong").text

movie\_data.append([rank, title, year, rating])

# Step 6: Store data in pandas DataFrame

df = pd.DataFrame(movie\_data, columns=["Rank", "Title", "Year", "IMDb Rating"])

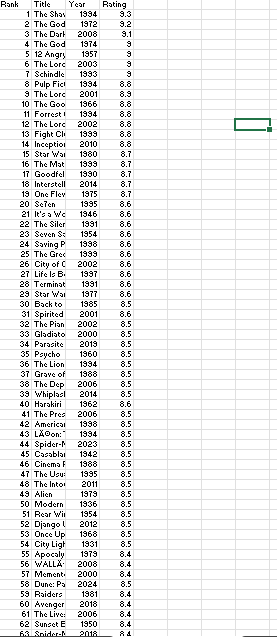
# Step 7: Save data to CSV

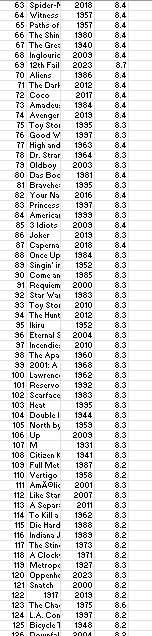
df.to\_csv("imdb\_top\_250.csv", index=False)

# Step 8: Close the driver

driver.quit()

print("✅ IMDb Top 250 movies scraped successfully! Data saved to imdb\_top\_250.csv")

**output:  
**

****

**8. Explanation of Code**

1. **Import Libraries** – Selenium for scraping, pandas for data handling, and webdriver\_manager for driver setup.
2. **Configure ChromeDriver** – Headless mode can be enabled for background execution.
3. **Access IMDb Page** – Opens IMDb’s Top 250 movies chart.
4. **Extract Data** – Uses CSS selectors to fetch rank, title, year, and IMDb rating.
5. **Store Data** – Organizes extracted data into a DataFrame.
6. **Export CSV** – Saves the data into imdb\_top\_250.csv.
7. **Automation Done** – Closes browser after execution.

**9. Outcomes**

* Extracted **IMDb Top 250 movies** with rank, title, year, and rating.
* Data saved in **CSV format** for future analysis.
* Scraper runs automatically and can be reused anytime.

**10. Applications**

* Film trend analysis.
* Recommendation system datasets.
* Visualization dashboards.
* Personal film collection tracking.

**11. Conclusion**

The IMDb Movie Rating Scraper demonstrates how Selenium can automate the extraction of real-time, dynamic web data. By exporting structured datasets into CSV, the project bridges the gap between web data and analytics. The scraper is extendable, making it a strong foundation for advanced data-driven applications in film analysis.

**12. Future Enhancements**

* Add cast, director, and genre information.
* Use database storage instead of CSV.
* Integrate data visualization for movie trends.
* Automate scheduling with CRON jobs or Task Scheduler.

**Submitted by   
Syed Ismail N**