Introduction(web scraping in Data engineering)

Why web scraping

Use-cases of web scraping

Pros-cons of web scraping

Challenges Limitations in web scraping

Tools and Technologies(OLD VS New)

Project overview

Project Introduction

Technologies Used in this project

Use case of this project

Conclusion and future work

Project Demo

📘 Web Scraping Project Guide

Table of Contents

1. Project Overview
2. Technologies Used
3. Project Structure
4. Detailed Code Explanation
5. Scraping Process
6. Learning Objectives
7. Challenges and Solutions

1. Project Overview 🌐

**What is Web Scraping?**  
Web scraping is the process of extracting data from websites automatically using code. In this project, we'll scrape movie information from IMDb's Top 250 Movies list.**Project Goal:**

* Extract top 250 movies data
* Store information in a MySQL database
* Create a reusable web scraping script

2. Technologies Used 🛠️

**Programming Language:**

* Python 3.8+

**Libraries:**

* requests: Send HTTP requests
* BeautifulSoup: Parse HTML content
* mysql-connector-python: Connect to MySQL database
* lxml: HTML parsing

**Database:**

* MySQL

3. Project Structure 📂

text

imdb\_scraper/

│

├── config.py # Configuration settings

├── database.py # Database interaction

├── scraper.py # Main scraping logic

├── utils.py # Data parsing utilities

└── requirements.txt # Project dependencies

4. Detailed Code Explanation 🔍

4.1 config.py

**Purpose:** Store configuration settings

python

*# Database connection details*

DB\_CONFIG = {

'host': 'localhost',

'user': 'root',

'password': '1234',

'database': 'imdb\_movies'

}

*# Scraping configuration*

SCRAPER\_CONFIG = {

'url': 'https://www.imdb.com/chart/top/',

'headers': {

*# Simulate browser request*

'User-Agent': 'Mozilla/5.0...'

}

}

4.2 utils.py

**Purpose:** Data cleaning and parsing functions

python

**def** parse\_year(year\_text):

*# Extract year from text*

*# Example: "1994" → 1994*

**def** parse\_duration(duration\_text):

*# Convert duration to minutes*

*# Example: "2h 22m" → 142 minutes*

**def** parse\_rating(rating\_text):

*# Extract numeric rating*

*# Example: "9.3" → 9.3*

4.3 database.py

**Purpose:** Manage database operations

python

**class** DatabaseManager:

**def** create\_connection():

*# Establish MySQL connection*

**def** create\_movies\_table():

*# Create table to store movie data*

**def** insert\_movies():

*# Insert scraped movie data into database*

4.4 scraper.py

**Purpose:** Main scraping logic

python

**class** IMDbScraper:

**def** extract\_movie\_details(movie\_html):

*# Extract individual movie details*

*# 1. Movie title*

*# 2. Ranking*

*# 3. Release year*

*# 4. Duration*

*# 5. IMDb rating*

**def** scrape\_top\_250():

*# Main scraping method*

*# 1. Send request to IMDb*

*# 2. Parse HTML*

*# 3. Extract movie containers*

*# 4. Process each movie*

5. Scraping Process Step-by-Step 🕵️

1. **Send Request**
   * Use requests to get IMDb Top 250 page
   * Include headers to mimic browser request
2. **Parse HTML**
   * Use BeautifulSoup to parse HTML content
   * Find movie containers
3. **Extract Movie Details**
   * Title
   * Ranking
   * Release Year
   * Duration
   * IMDb Rating
   * Total Ratings
4. **Clean and Validate Data**
   * Remove unwanted characters
   * Convert to appropriate data types
5. **Store in Database**
   * Connect to MySQL
   * Create table
   * Insert movie data

6. Learning Objectives 🎓

**Technical Skills:**

* Web scraping techniques
* HTML parsing
* Data cleaning
* Database interaction
* Error handling
* Python programming

**Soft Skills:**

* Problem-solving
* Code organization
* Understanding web structures

7. Challenges and Solutions 🧩

**Challenge:** Websites change HTML structure  
**Solution:**

* Use flexible parsing
* Regular expression matching
* Robust error handling

**Challenge:** Ethical web scraping  
**Solution:**

* Respect robots.txt
* Add delays between requests
* Use official APIs when possible

8. Best Practices 📌

1. Always get permission before scraping
2. Add request delays
3. Handle errors gracefully
4. Clean and validate data
5. Use virtual environments
6. Secure database credentials

9. Potential Improvements 🚀

* Add logging
* Implement proxy rotation
* Create command-line interface
* Add more robust error handling

10. Recommended Learning Path

1. Python basics
2. HTML/CSS understanding
3. HTTP requests
4. Database basics
5. Web scraping ethics

Conclusion

This project demonstrates a complete web scraping workflow, from sending requests to storing data, while teaching valuable programming skills.