## Introduction

This project focuses on creating a web application to provide movie recommendations and insightful data visualizations by extracting and organizing metadata such as title, year, runtime, content rating, and IMDb rating for the Top 250 movies. Users can select a movie from a dropdown list, and it will provide a list of similar movies based on predefined criteria. As a movie enthusiast, sometimes selecting a movie to watch can be a difficult task, especially with a great amount of options. This implementation project incorporates several concepts from the course, such as web crawling, metadata structuring, and faceted navigation. The project demonstrates how these theoretical ideas can be applied to solve real-world problems and improve user experience.

## Technical implementation

The project consists of three major parts: data processing, recommendation logic, and web application design.

### Data processing

The first step was to web scraping the IMDb Top 250 movies webpage by using Python's requests, BeautifulSoup and selenium to extracte title, year, runtime, content rating, and IMDb rating. After storing the scrapped data in a JSON file, the runtime field was converted into minutes for easier data calculations.

### Recommendation and visualization

The recommendation logic uses a set of conditions to identify similar movies based on:

- Content Rating: movies must share the same content rating unless marked "Not Rated" or not marked.
- IMDb Rating: include movies within a ±0.2 range of the selected movie's rating.
- Decade: movies from the same decade to reflect similar cultural and historical contexts.
- Runtime: movies with a runtime difference of ±30 minutes.

Data visualization was implemented using Python's matplotlib to generate bar charts and box plots and displayed in the web application. These visualizations provide faceted insights into the dataset for users to explore trends and patterns.

**Web application design**

The frontend was developed using HTML, CSS, JavaScript. Key features:

- A dropdown menu using Bootstrap dynamically populated with movie titles from the JSON dataset and return the recommended movies based on similarity.
- visualizations charts showcasing movie attributes such as runtime distribution, IMDb ratings, content ratings, and decade trends.

## Class concepts incorporation

This project integrates multiple concepts discussed in class:

**Structuring Information and Metadata:**

The extraction and organization of movie data into a structured JSON file demonstrate the importance of metadata design. Organizing information effectively supports retrieval and analysis; for example: The runtime field was converted from a string format (2h 22m) into minutes for consistency and easier computation. Metadata organization made it possible to apply recommendation logic directly and generate visualizations.

**Web Crawling**

The project began with a web crawler to retrieve movie data from IMDb top250 while respecting the site’s robots.txt policy. This step aligns with the course's introduction to web crawling, focusing on how to extract data from structured HTML. The crawler used Python’s requests and BeautifulSoup libraries to parse HTML content and extract metadata fields.

**Faceted Navigation:**

In data visualizations, the charts displaying insights into movies by decade, runtime, ratings, and content ratings break down the dataset into multiple facets. These facets allow users to explore and understand the dataset across various dimensions without requiring direct interaction.
The recommendation system implicitly applies faceted navigation by combining multiple criteria (facets) such as:

- Content rating (e.g., G, PG, R).
- Decade (e.g., 1990s, 2000s).
- IMDb rating range (e.g., ±0.2 from the selected movie).
- Runtime range (e.g., ±30 minutes from the selected movie).

The system dynamically filters and retrieves results based on these dimensions, providing a user-friendly exploration experience.

**Ranking and Retrieval:**

The recommendation system integrates basic ranking and retrieval concepts to determine the relevance of movies to the user-selected title. Movies that satisfy conditions are considered highly similar and are included in the recommendations. While the implementation is relatively simple, it reflects the course's principles of combining multiple filters to rank and retrieve relevant information effectively.

## Challenges and Improvements

During the development of this project, some challenges were encountered:

**Web scraping challenge:**

Initially, the project used standard HTTP requests and HTML parsing libraries to scrape data from IMDb top 250 webpage. However, due to dynamic content loading on the website, this approach failed to retrieve the complete dataset.

This issue was later resolved by incorporating Selenium, a browser automation tool, to interact with the webpage and extract the complete information. For future improvements, it could involve exploring IMDb's API or other data sources to ensure more efficient and reliable data retrieval.

**Incomplete metadata:**

Some movies had "Not Rated" as their content rating, which limited it as a filtering criterion. Some other movies lacked content rating information, making inconsistencies in the dataset.

To resolve this problem, the system currently ignores the content rating condition when it is "Not Rated" or missing(null). A potential improvement could supplement the dataset with

external sources to fill in missing metadata(content rating) to create a more complete dataset for filtering and recommendations.

**Limited attributes:**

The current implementation only includes five attributes: title, year, runtime, content rating, and IMDb rating. The lack of additional metadata (such as directors, actors, or genres) restricted the system's ability to provide more personalized recommendations. These could be integrated from external APIs to enrich the dataset and create a more sophisticated recommendation engine.

**Subjective Rule-Based Recommendation Criteria:**

The current recommendation system relies on manually defined rules, such as: IMDb rating within a ±0.2 range, runtime within a ±30-minute range, and same decade for year. These were chosen based on intuition rather than rigorous data analysis or user feedback. They may not accurately reflect user preferences or align with what users might consider "similar movies."
To improve this problem, there are some methods, such as implementing statistical analysis or collecting user feedback.

## Conclusion

This project successfully implements a web-based recommendation system and data visualization platform, leveraging IMDb's Top 250 dataset. By integrating concepts from the course, it bridges theoretical knowledge and practical application. The system not only simplifies the movie selection process but also provides users with insights into the dataset. Future iterations could further enhance its functionality and user experience, making it a more robust tool for movie exploration.