CONTENTWISE

Recommendation System

Objective

Create a Java-based RESTful service that simulates a recommendation system for video contents. The service has to manage a catalog of movies, user interactions (i.e. events), and allow for the ingestion of new interactions. Particularly, the service has to manage two different types of event:

- **Rating:** an event that provides a rating for a specific movie as an integer value ranging from 1 to 5, meaning 1 lowest rating (disliked very much) and 5 highest rating (liked very much);
- **View:** an event that provides a view percentage for a specific movie ranging from 0% to 100%. Moreover, this type of event has to generate an implicit rating value based on the given view percentage. E.g.
    - if “View” between 60% and 80% => Rating 4;
    - if “View” greater than 80% => Rating 5;

Note: explicit ratings have to be considered with a higher priority than implicit ones. E.g.

- User Alice rated "Toy Story" with a 4 and viewed more than 80% of its content generating an implicit rating of 5, the final rating to be considered is 4.

Finally, the application must include a persistence layer to store and retrieve data.

Functional requirements

You are asked to implement the following RESTful APIs:

1. Retrieve a list of all movies

Retrieve a list of all movies with the possibility to specify optional query parameters:

- **genre**: filter movies by genre;
- **minRating**: filter movies with an average rating above a certain value;
- **maxRating**: filter movies with an average rating below a certain value;

2. Retrieve a user's interaction history

Retrieve a user's interaction history. The API has to provide an optional query parameter to retrieve ratings only, views only or both.

3. Add an event for a movie by the user

Ingest a new event (rating or view) for a movie by updating the user's interaction history in the database and adjusting recommendation results accordingly.

4. Recommend movies for a user

Retrieve a list of recommended movies similar to those that the user has rated highly (e.g. 4 and 5) and hasn't rated yet. For sake of simplicity, similarity between movies has to be computed on “Genre” metadata only. Optionally, prioritize movies with the higher number of events.

Hint

- User Alice rated "Toy Story" (genres: Adventure, Animation, Children, Comedy, Fantasy) with a 4 and "Grumpier Old Men" (genres: Comedy, Romance) with a 5;
- Preferred genres are Adventure, Animation, Children, Comedy, Fantasy and Romance;
- Recommend movies matching these genres that Alice hasn't rated;

Provided Data Files

You are given the following CSV files to use as your initial data source:

- **movies.csv**: contains movie information.
- **users.csv**: contains user information.
- **ratings.csv**: contains user ratings for movies.

Samples can be found in the appendix section of this document.

Technologies

- Java 11 or higher. Kotlin is also a valid alternative.
- Free of choice to use a SQL or NoSQL database;
- Any Java frameworks or libraries of your choice (e.g., Spring Boot);

Bonus Points (Optional)

Search Capability

You are asked to implement a search functionality, exposing a new REST API, to allow searching for movies based on their metadata(e.g. “Title” and “Genre”). The search should:

- match an exact title;
- match 1 or more genres;
- match 1 or more words within the title;

Deliverables

- **Source Code**: the complete source code of your application;
- **Data Files**: include the provided CSV files (movies.csv, users.csv, ratings.css) in your project;
- **Docker Compose:** to allow deploying and running the application;
- **README.md:** instructions for building and running the application;

Bonus Points (Optional)

- **API Spec:** to understand the capabilities of the service without access to source code;
- **DB Schema:** design choice for the selected DB;

Evaluation Criteria

- **Functionality**: how well the application meets the specified requirements;
- **Code Quality**: cleanliness, organization, and readability of your code.
- **Design Principles**: effective use of design patterns and adherence to best practices;
- **Testing:** cleanliness and effective designing of unit tests;
- **Documentation**: clarity and comprehensiveness of the documentation;

Bonus Points (Optional)

- **Monitoring & Observability:** comprehensiveness and reliability of the service by exposing service metrics and implementing logging capabilities.

CONTENTWISE

Guidelines

- **Frameworks/Libraries:** you may use any Java frameworks or libraries you prefer for building RESTful services and interacting with the database (e.g. Spring Boot);
- **Focus:** prioritize creating a functional prototype over a production-ready application.
- **Simplicity:** keep the recommendation algorithm and optional search implementation straightforward.

Submission

Please provide a link to a Git repository containing your project files, including the data files and documentation.

Conclusion

We are aware that some specifications and requirements may appear vague, and it is up to you to make reasonable assumptions and motivate your choices accordingly. There are many possible ways to implement this challenge, but we are more interested in hearing all your insights about how you would design this simple application.

Contacts

If you find the text of this challenge not clear enough please don't hesitate to reach out to riccardo.biondi@contentwise.com.

Appendix

1. movies.csv

This file contains movie information, including `movie_id`, `title`, and `genres`.

```
Unset
movie_id,title,genres
1,Toy Story,Adventure|Animation|Children|Comedy|Fantasy
2,Grumpier Old Men,Comedy|Romance
3,Die Hard,Action|Thriller
4,Star Wars: Return of the Jedi,Action|Adventure|Fantasy|Sci-Fi
5,The Lion King,Adventure|Animation|Children|Drama|Musical
6,Pulp Fiction,Crime|Drama|Thriller
7,Forrest Gump,Comedy|Drama|Romance
8,The Matrix,Action|Sci-Fi
9,Goodfellas,Biography|Crime|Drama
10,Jurassic Park,Adventure|Sci-Fi|Thriller
```

2. users.csv

This file contains user information, including `user_id` and `user_name`.

```
Unset
user_id,username
1,Alice
2,Bob
3,Charlie
```

3. ratings.csv

This file has columns for `user_id`, `movie_id`, `rating` (explicit ratings from 1 to 5), and `view_percentage` (indicating the view percentage for generating implicit ratings if an explicit rating is missing).

```
Unset
user_id,movie_id,rating,view_percentage
1,1,4,85
1,2,5,
2,1,,90
2,3,3,
3,4,,70
3,2,2,
```