Movie Festival App

Overview

Your company wanted to create an app for a movie festival and you has been assigned for backend developer for the app.

Before the festival, an admin will manually collect movie files from invited participants. The admin will then upload the movie files and add relevant movie attributes (title, etc) through CMS, and the admin is also able to update the entry if necessary.

During the festival, everyone who installed the app can search and view the uploaded movies. Every time someone sees a movie, it's counted as 1 view for the movie.

Some users can register and login to the system. These authenticated users can vote for a movie that they like, with limitation that 1 user can only vote for the same movie once. However, an authenticated user can vote for multiple movies that they like, and they also can unvote a movie if they changed their mind later.

After the festival finishes, the admin can see which movies are the most popular (have the most viewership) and which movies are the most liked (have the most votes).

Requirements

Basic Requirements

Admin APIs

- API to create and upload movies. Required information related with a movies are at least **title, description, duration, artists, genres, watch URL** (which points to the uploaded movie file)
- API to update movie
- API to see most viewed movie and most viewed genre

All Users APIs

- API to list all movies with pagination
- API to search movie by title/description/artists/genres
- API to track movie viewership

Bonus Requirements

- Vote system:
    - API to login as an authenticated user

    - API to vote a movie as an authenticated user
    - API to unvote a movie as an authenticated user
    - API to list all of the user's voted movie
    - API to see most voted movie and most viewed genre, as an admin
- User registration and authentication system:
    - API to register
    - API to login and logout
- Trace viewership based on watching duration

Development Guidelines

- You have to use **Golang** to develop the backend system
- You may use any framework (echo/gin/gorilla-mux/gorm/etc.) and any free open source libraries
- You may build using SQL
- You are free to express your opinion through your code, but try to make your code as clean as possible
- Please keep a fine grained commit with separate commits for each requirements
- Please provide API documentation in Postman collection and little description about what tools you used or anything special on your code with simple Readme file, or anything you like.
- We absolutely hate to take away your free time, so we designed the basic requirements to be achievable within **6 hours**. If you need to take more time than 6 hours, just stop and submit your work as is :)
- However, if you found the requirements to be fun, feel free to take as much time as you need and complete all the requirements, and even expand the solution beyond the original requirements!

Evaluation Criteria

The evaluation criteria is split into 3 parts with different weights:

60% for solution correctness. You don't really need to do complex optimization as we value correctness first for this test.
40% for technical aspects:

- Solution and project architecture
- SQL database design
- API route design
- Code cleanness

10% bonus points for unit and integration tests :)

Submissions

Please submit your solution as a Git repository. Don’t forget to make it publicly accessible. :)

Thank you!