Objective

Build a Movie Explorer Platform that allows film enthusiasts to explore movies, actors, directors, and genres.
There is no need for user authentication or authorization in this assignment.

Backend Requirements

Implement the backend using Python.
Use Flask, FastAPI, or any other Python web framework of your choice.
Use any database (e.g., SQLite, PostgreSQL, MongoDB).
Document the API using Swagger with OpenAPI Specs.

Design and implement the following core entities and API resources:
- Movies
- Actors
- Directors
- Genres

Ensure relationships between models are meaningful:
- Movies can belong to multiple genres.
- Movies can have multiple actors and a single director.

Enable filtering in APIs:
- Movies by genre, director, release year, or actor.
- Actors by movies or genres they acted in.

Frontend Requirements

Use ReactJS or VueJS only (TypeScript is recommended, but JavaScript is acceptable).
Use a CSS framework of your choice (e.g., Tailwind CSS, Bootstrap, Material UI).

Implement a user interface that allows users to:
- Browse a list of movies with key details (title, release year, genres, director).
- Filter/search movies by genre, actor, or director.
- View a detailed page for each movie with cast, director, and genre info.
- View a profile for each actor/director with movies they've worked on.

Expectations

A working full-stack application with a clear and concise README.md file.
The project should be Dockerized, with commands to build and run both frontend and backend.
Use Vite as the frontend bundler (recommended).

Include linting as part of the build step.
Include unit tests for both frontend and backend, integrated into the build step.
API filtering should be handled on the backend — do not filter data in the frontend.
Provide inline code documentation where necessary.

General Guidelines

Push your project to a public GitHub repository and share the link for evaluation.
Add movie ratings or reviews (even as dummy/mock data).
Evaluators will clone your repo and follow the instructions in the README.md file.
Ensure the UI adheres to good design standards and extensible patterns.
Handle and document edge case scenarios (e.g., no movies available, invalid filters, etc.).
Keep the code modular, testable, and maintainable.

Bonus (Optional, not mandatory)

Add a "favorites" or "watch later" section (no need for user accounts; use local storage).