Please create a .Net 6 Web Api for managing a list of movies, which allows you to view and edit information about Movies, Actors, and Movie Ratings.

Backend Requirements:

- Have endpoints that allow you to create, retrieve, update, and delete records
- Use Entity Framework Core (in-memory db, sqlite, whatever you want), with a code first approach, for **CRUD** (***C**reate **R**etrieve **U**pdate **D**elete)* operations and the database should be seeded with some dummy data

- Have at least 3 entity types:
    - Movies
    - Actors
    - MovieRatings

  and they should have the appropriate relationships with each other.
  For example, a Movie could have a list of actors or a list of ratings or no ratings.

- Must expose a swagger endpoint, with full documentation for each API endpoint

- Must allow you to do partial searches for Movies or Actors, based on name.

- Must be able to view all Movies an Actor has been in, and list all Actors in a given Movie.

- Must require an API secret/token and validate it, when trying to Create/Update/Delete entities via corresponding API endpoints. Searching/listing shouldn't check for the API token/secret. **You can hard code the secret.**

- Properly validate requests and return appropriate HTTP responses and response codes, based on the invocation result *(ie: if an error happened or if trying to access a restricted endpoint, without the API secret, the right HTTP code is returned)*

Frontend Requirements:

- Create a basic React (version 16 or higher) front end application, using functional components (not class based components: https://www.twilio.com/blog/react-choose-functional-components) that loads and displays Movies and Actors, from the endpoints you create for the backend.

Requirements for completion:

The code should be uploaded to a public GitHub repository and a link given to us, so that we can clone and run the project(s) locally.

- You can use any nuget/npm package/starter project/library desired.
- Code must be appropriately commented
- **The project shouldn't crash or throw any errors; you should anticipate where/when errors can happen, and handle them, accordingly.**
- The project should be runnable in either Windows, OSX, or Linux
- Bonus points for Docker usage
- You will be evaluated on the usage of SOLID principles (see https://en.wikipedia.org/wiki/SOLID)
- Bonus points if you setup any GitHub Action pipelines in your repo (such as running tests on commits)

Please be prepared to explain why you made the coding decisions you did, when building this application.