Movis Cards API – Extended functionality

Searching and Filtering:

- **Objective**: Allow users to search and filter movies through query string parameters.
- **Tasks**:
    - Implement search on the GET request for all Movies
    - Add query string parameters to search movies by Title, Genre, (**Extra**: ActorName, DirectorName and ReleaseDate)
    - Enable combining multiple search criteria in a single request.
    - Add optional query string parameters to include related entities (e.g., include all actors for a movie).

Sorting:

- **Objective**: Provide sorting functionality for movie listings.
- **Tasks**:
    - Implement sorting options on the GET /api/movies endpoint.
    - Allow sorting by:
        - `Title` (ascending/descending)
        - `ReleaseDate` (newest/oldest)
        - `Rating` (highest/lowest)
    - Support multiple sorting criteria in a single request.
    - Combine sorting with filtering and searching
    - Validate and handle invalid sorting parameters gracefully.

Custom Validation:

- **Objective**: Enhance validation to ensure data integrity.
- **Tasks**:
    - Prevent adding duplicate actors with the same Name and DateOfBirth.
    - Add validation to ensure that a Movie's ReleaseDate is not set in the future.
    - Ensure that Title is unique across all movies.
    - Validate that Rating falls within an acceptable range (0 to 10).

API Functionality: Add Actors to Existing Movie:

- **Objective**: Provide an endpoint to add actors to an existing movie.
- **Tasks**:
  - Implement POST /api/movies/{id}/actors to add one or multiple actors to a specific movie.
  - Accept a list of actor IDs or detailed actor information in the request body.
  - Validate that actors exist before association; create new actors if detailed information is provided.
  - Prevent adding duplicate actor associations to the same movie.
  - Return updated movie details upon successful addition.
  - Handle and return appropriate error responses for scenarios like invalid movie ID or actor IDs.

PATCH Operations for Partial Updates:

- **Objective:** Enable partial updates to resources using HTTP PATCH method.
- **Tasks:**
  - Implement PATCH /api/movies/{id} to allow partial updates to a movie's details.
  - Utilize JsonPatchDocument<MovieUpdateDto> to apply JSON Patch operations.
  - Ensure that partial updates adhere to all validation rules.
  - Handle scenarios where invalid properties are provided in the patch document.

AutoMapper Integration:

- **Objective**: Automate mapping between your entities and DTOs using AutoMapper.
- **Tasks**:
  - Configure AutoMapper profiles for mapping entities to DTOs and vice versa.
  - Implement mappings for Movie, Director, Actor, Genre, and their corresponding DTOs.

Repository Pattern:

- **Objective**: Abstract the data access layer using the repository pattern.
- **Tasks**:

  Create repository interfaces and concrete implementations for:

  - IMovieRepository
  - IDirectorRepository
  - IActorRepository

  Implement unit of work pattern to coordinate repository operations.

  Inject repositories into services or controllers using dependency injection.

  Ensure that all data access operations are performed through these repositories.

Status Codes and Error Handling:

- **Objective**: Ensure that the API returns appropriate and consistent HTTP status codes along with meaningful error messages.
- **Tasks**:
  - (**Extra:** Define and implement a global error handling mechanism (using middleware) to catch and process exceptions uniformly.)
  - Return standard HTTP status codes for various outcomes:
    - 200 OK for successful GET, PATCH, and POST operations.
    - 201 Created for successful resource creation.
    - 400 Bad Request for validation errors and malformed requests.
    - 404 Not Found when requested resources do not exist.
    - 409 Conflict for concurrency and uniqueness conflicts.
    - 500 Internal Server Error for unhandled exceptions.
  - (**Extra**: Log errors appropriately for monitoring and diagnostics).

Extra Tasks:

Integration with React App:

- **Objective**: Enhance the existing React application to utilize the new API features effectively.

- **Tasks**:
  - Update the frontend to support searching, filtering, and sorting of movies using the implemented API endpoints.
  - Implement forms and UI components to perform partial updates (PATCH operations) on movies, directors, and actors.
  - Add functionality to associate actors with movies through the UI, leveraging the POST /api/movies/{id}/actors endpoint.
  - Handle and display validation errors and status messages received from the API.
  - Ensure a responsive and user-friendly interface by providing loading states and error notifications.