Movie Review API

Objective:
Develop a RESTful API for a Movie Review System with user authentication. This system should allow users to manage movies, reviews, directors, and genres. The project will involve database relationships, including one-to-many and many-to-many, and should include a search feature using query parameters. Additionally, it will include file uploads for movie thumbnails and a scenes gallery, along with authentication functionality.

Requirements:

1. **Entities:**
    - **User:** `id`, `username`, `email`, `password`
        - A user can have multiple reviews (one-to-many).
    - **Director:** `id`, `name`, `bio`
        - A director can direct multiple movies (one-to-many).
    - **Movie:** `id`, `title`, `description`, `releaseDate`, `directorId`, `genreId`, `thumbnail`, `scenesGallery`
        - A movie belongs to one director (many-to-one).
        - A movie can belong to multiple genres (many-to-many).
        - A movie has a thumbnail image and a scenes gallery (array of images).
    - **Genre:** `id`, `name`
        - A genre can have multiple movies (many-to-many).
    - **Review:** `id`, `rating`, `comment`, `movieId`, `userId`
        - A review belongs to one movie (many-to-one).
        - A review belongs to one user (many-to-one).
2. **Authentication:**

- **Register:**
  - `POST /auth/register` - Register a new user with `username`, `email`, and `password`.
- **Login:**
  - `POST /auth/login` - Authenticate a user with `email` and `password`, returning a JWT token.
- **Protected Routes:**
  - Use JWT middleware to protect certain routes. Only authenticated users can access these routes.
  - Users should only be able to create, update, or delete their own reviews.

3. **API Endpoints:**
   - **Users:**
     - `GET /users/:id` - Get details of a specific user (protected).
     - `PUT /users/:id` - Update user details (protected).
     - `DELETE /users/:id` - Delete a user (protected).
   - **Directors:**
     - `GET /directors` - Get a list of all directors.
     - `GET /directors/:id` - Get details of a specific director.
     - `POST /directors` - Create a new director (protected).
     - `PUT /directors/:id` - Update a director (protected).
     - `DELETE /directors/:id` - Delete a director (protected).
   - **Movies:**
     - `GET /movies` - Get a list of all movies with optional search filters (`title`, `director`, `genre`).
     - `GET /movies/:id` - Get details of a specific movie.

    - `POST /movies` - Create a new movie, including uploading a thumbnail and scenes gallery (protected).
    - `PUT /movies/:id` - Update a movie, including replacing the thumbnail and scenes gallery (protected).
    - `DELETE /movies/:id` - Delete a movie (protected).
  - **Genres:**
    - `GET /genres` - Get a list of all genres.
    - `GET /genres/:id` - Get details of a specific genre.
    - `POST /genres` - Create a new genre (protected).
    - `PUT /genres/:id` - Update a genre (protected).
    - `DELETE /genres/:id` - Delete a genre (protected).
  - **Reviews:**
    - `GET /reviews` - Get a list of all reviews with optional search filters (`rating`, `movie`, `user`).
    - `GET /reviews/:id` - Get details of a specific review.
    - `POST /reviews` - Create a new review (protected).
    - `PUT /reviews/:id` - Update a review (protected, only by the review owner).
    - `DELETE /reviews/:id` - Delete a review (protected, only by the review owner).
4. **File Uploads:**
  - Implement file upload functionality using Multer.
  - Allow uploading a single image for the movie thumbnail.
  - Allow uploading multiple images for the scenes gallery.
5. **Search Functionality:**
  - Implement search functionality for movies with title , director , genre.
  - Example: `GET /movies?title=Inception&director=Christopher%20Nolan&genre=Sci-Fi`.

6. **Validation:**
    - Implement basic validation for all entities, ensuring required fields are provided and are of the correct type.
    - Validate image files for the correct format and size.
7. **Deliverables:**
    - **Postman Collection:** A collection of all API endpoints with sample requests for testing, including authentication tokens.
