Sure, I'll rephrase the exercises to fit the context of your Movie Rating System project:

Movie Rating System

Callback Functions

Exercise 1:

Create a function named `processMovies`. This function should accept two parameters: an array of movie objects and a callback function.

The structure of a movie object in the array should be as follows:

```
{
  id: 34,
  title: "Movie Title",
  rating: 8.5
}
```

Your `processMovies` function should perform the following tasks:

1. Verify that the first argument is an array. If not, throw an error with the message 'Expected an array of movies'.
2. For each object in the array, check if it contains the properties `title` and `rating`. If any object lacks these properties, throw an error with the message `Movie with ID ${movie.id} is not correctly structured` (assume the `id` property is always present).
3. To avoid mutating the original array, clone it: `const newMovies = JSON.parse(JSON.stringify(movies));`.
4. Iterate over each object in the cloned array and pass each object to the callback function.
5. Return the new array.

Exercise 2:

Create a function named `processReviews`. This function should accept two parameters: an array of review objects and a callback function.

The structure of a review object in the array should be as follows:

```
{
  id: 45,
  movieId: 34,
  rating: 9,
  comment: "Great movie!"
}
```

Your `processReviews` function should follow the same procedures as in Exercise 1, but applied to reviews. Adjust the error messages accordingly, ensuring they're appropriate for reviews.

Exercise 3:

Create a function named `processUsers`. This function should accept two parameters: an array of user objects and a callback function.

The structure of a user object in the array should be as follows:

```
{
  id: 'a9bcf',
  name: "Username",
  age: 30,
  favoriteMovies: ["Movie Title 1", "Movie Title 2"],
  reviewsPosted: 5
}
```

Your `processUsers` function should follow the same procedures as in the previous exercises, but applied to users. The structure verification should check for `name`, `age`, `favoriteMovies`, and `reviewsPosted`. Adjust the error messages accordingly.

For each of these functions, ensure you pass a callback function that will be applied to each object in the respective array.

For example, you could test the `processUsers` function with a callback that increments each user's `reviewsPosted` by 1 like this:

```
processUsers(users, user => {
    user.reviewsPosted += 1;
    return user;
});
```

This will return a new array of users, each with their `reviewsPosted` count incremented by 1, without modifying the original `users` array.