# Workshop: Movie Magic – Part 3

Your team at work is tasked with creating a website that will attract movie enthusiasts. The site is called "Movie Magic". Its purpose is to create an attractive and interactive platform for movie fans that offers a unique and enriching experience.

If you missed the first two parts of this workshop, make sure you complete them before you continue because all parts of this workshop are related to each other.

## Main Task

Now it's time to implement **user service** in your app, so people can **register**, **log in** and **log out**, as well as be able to **edit** or **delet** their movies. Some of the functionalities must **require** **authentication** (such as create a movie, create a cast) and **authorization** (such as edit and delete). Also, all **routes** should be **protected**!

## Installing Dependencies

You should install a few more packages which you will use. They are:

1. [**jsonwebtoken**](https://www.npmjs.com/package/jsonwebtoken) – allows you to decode, verify and generate JWT
2. [**bcrypt**](https://www.npmjs.com/package/bcrypt) – a library to help you hash passwords
3. [**cookie-parser**](https://www.npmjs.com/package/cookie-parser) – parses the **cookie header** and populates **req.cookies** with an object keyed by the cookie names (if you choose to store the **jwt** as а **cookie**)

## Model

The **User** model structure:

* **email** – String
* **password** – String (**hashed**): Use **bcrypt** to **hash** and **compare** the password.

When you successfully create a new user into the database, make sure you generate a **jsonwebtoken** and use it later for **authentication** and **authorization**.

Also, you have to add a property to the **Movie** **model,** which is **creatorId** (type: **String** and it's **required**), so you can keep tracking every movie's creator.

## Routes Protection

Make sure the **anonymous** (guest) users **can't** **reach** the functionality which requires authentication, such as creating a movie view. **Already logged-in** users must have generated and stored **jwt**, **can** see the correct navigation, and **can't reach** the login and register form. If some of these scenarios happen, make sure the current user is redirected to the home page.

### Authentication

**Guest** users can **see** and **access** the following URLs**:**



* **Home Page**
* **About Page**
* **Search Page**
* **Login Page**
* **Register Page**

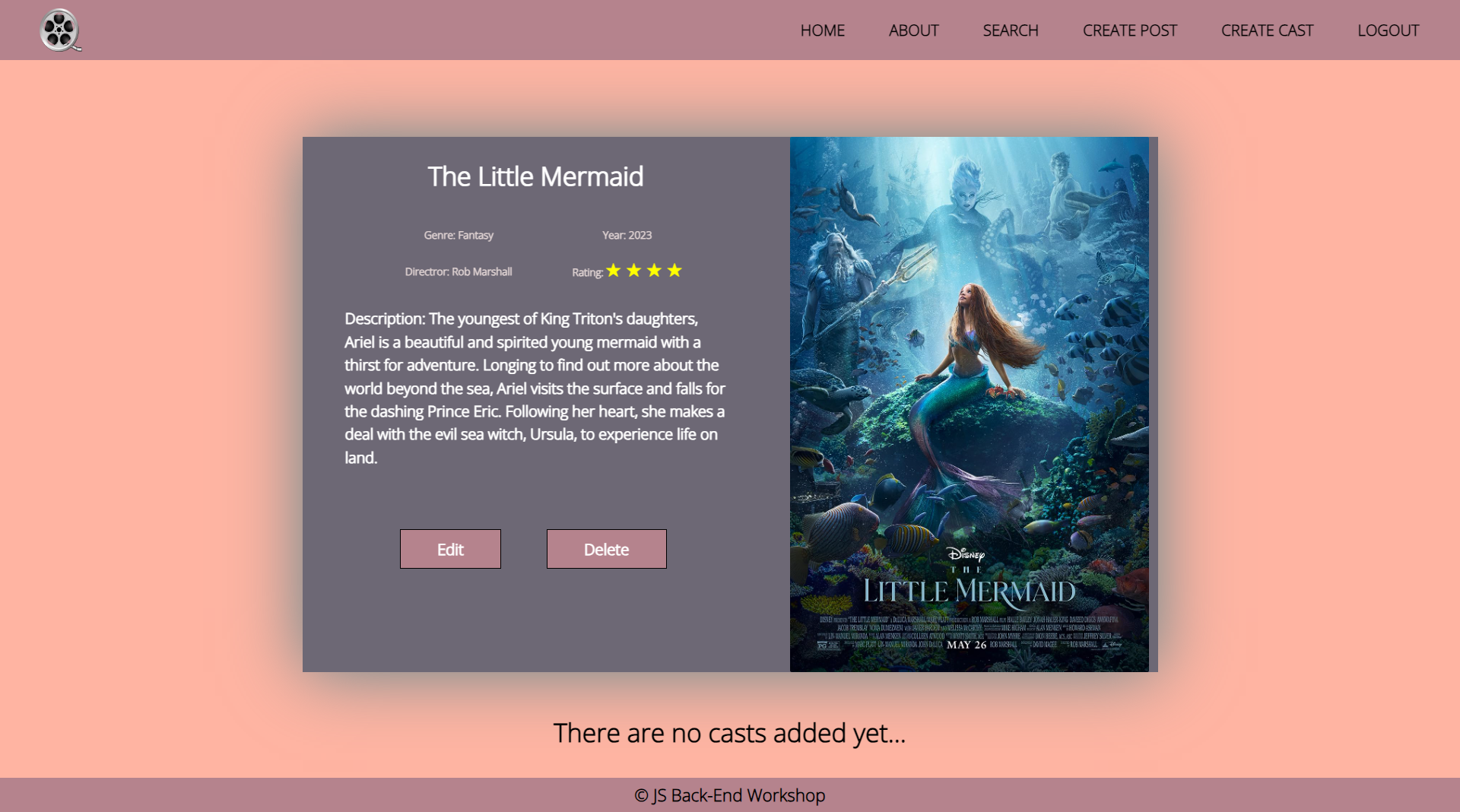
They **can't access** anything else.

**Logged-in** users can **see** and **access** the following URLs:



* **Home Page**
* **About Page**
* **Search Page**
* **Details Page**
* **Create Post**
* **Create Cast**
* **Attach Cast**
* **Logout**

### Authorization

Only **the creator of the current movie** should be able to see the [**Edit**] and [**Delete**] buttons. Otherwise, the buttons should be **hidden**. 

## Additional Pages

You should implement **4** new routes:

* **/login** – should render the login form
* **/register** – should render the register form
* **/edit/:id** – should render the edit form

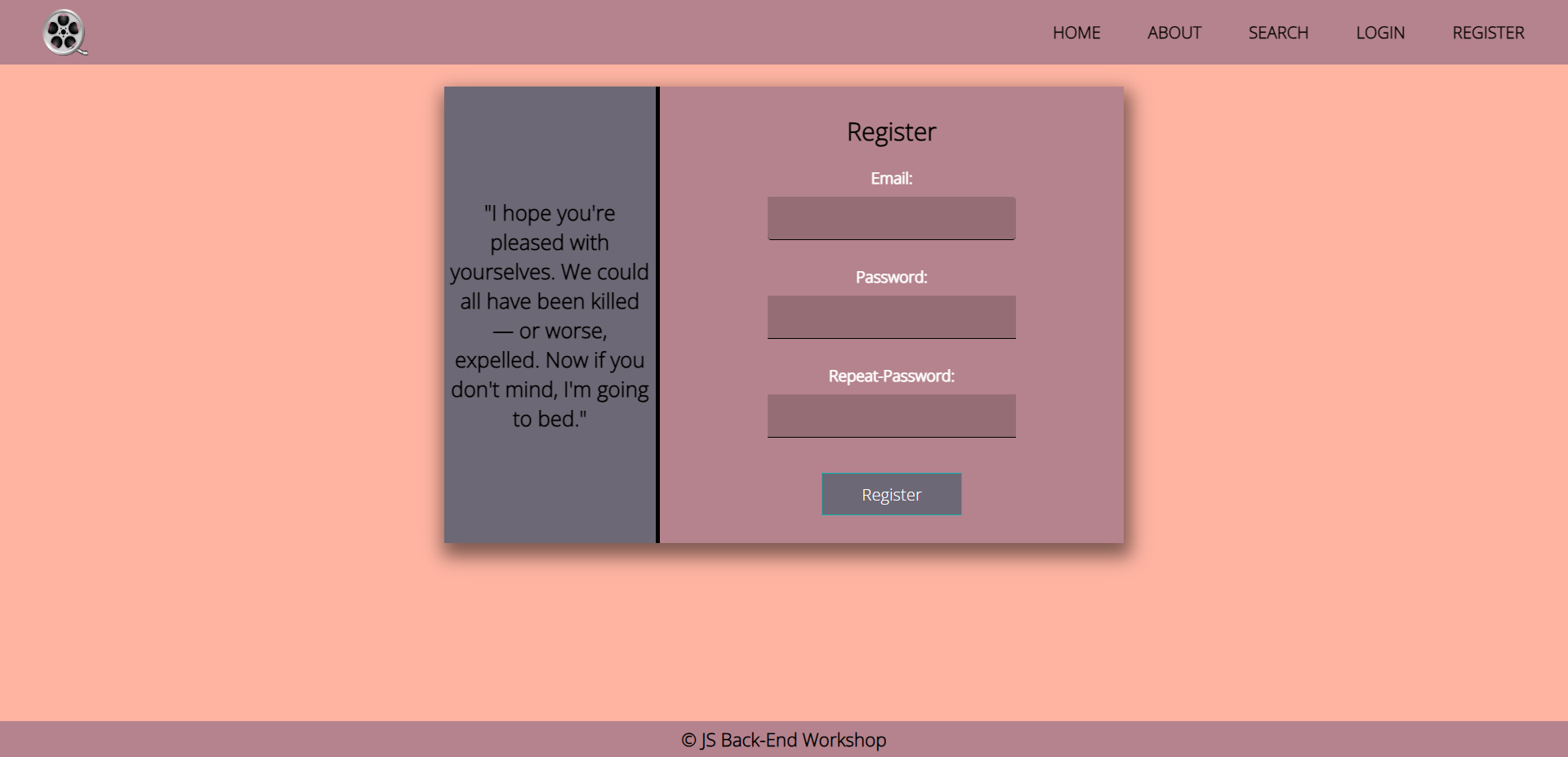
Make sure that when you access **edit page,** it shows the current movie information.

### Login Page

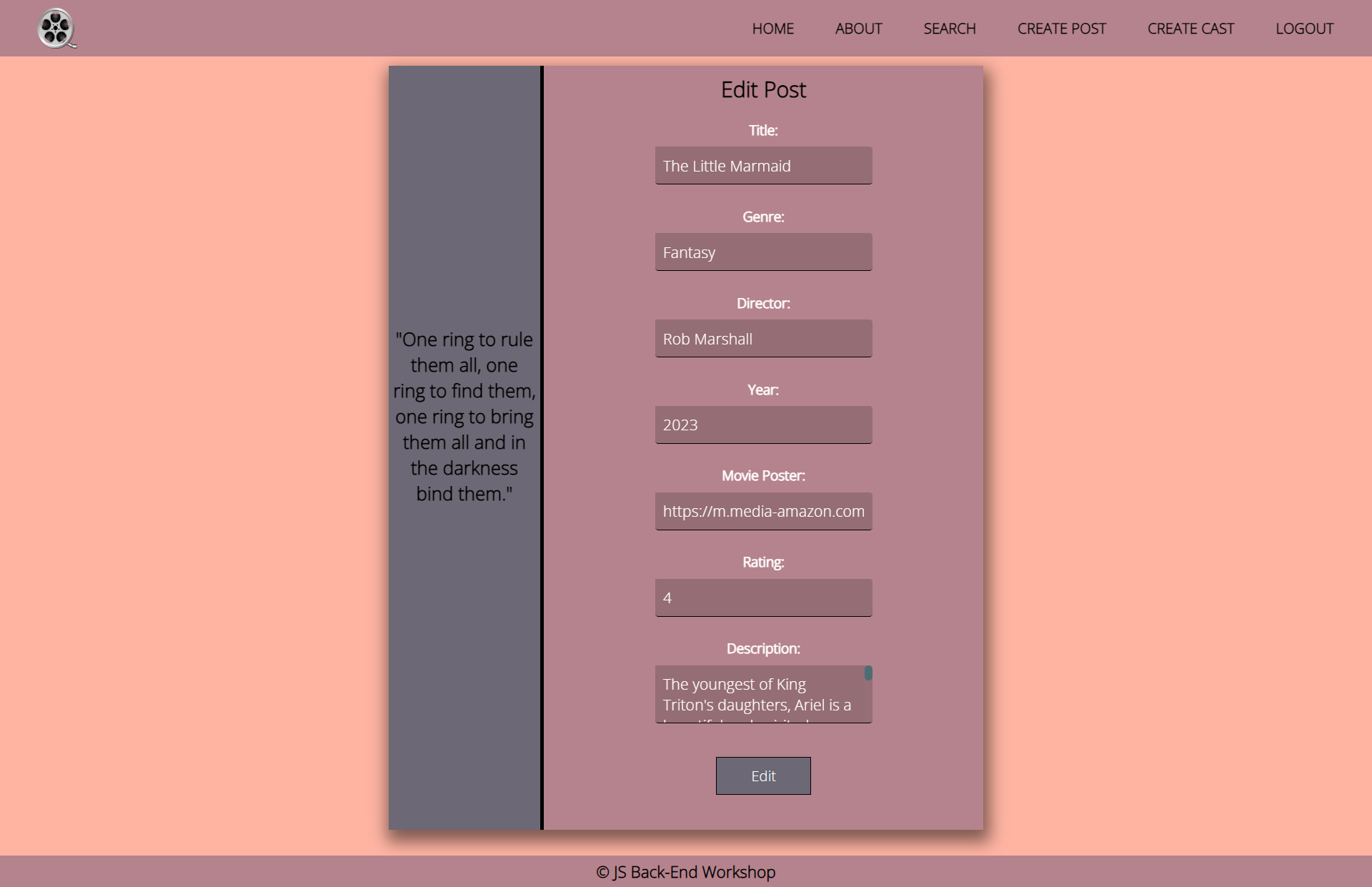
Картина, която съдържа текст, екранна снимка

Описанието е генерирано автоматично

Register Page



### Edit Movie Page



**Good Luck! 😊**