Overview for Movie-Hub Website's Code

ALaa Mostafa Hafez

Amit Learning

Agenda

Frontend

1. Home
2. Signup
3. Login
4. Navbar
5. Footer
6. Movies
7. Community
8. Comment Form
9. Post

Backend

1. Models
2. Views
3. Serializers
4. URLs

1- FRONTEND

1- Home

This code is a React functional component that renders a Home page . It imports some images and CSS files, as well as two other React components (Navbar and Footer). The component has three useEffect hooks, the first one animates some text by changing it to a random number every two seconds, the second one sets a loading state and removes a loading screen after two seconds, and the third one sets the loading state to true and false after four seconds. Inside the return statement, there are several HTML sections such as a hero section, an "about" section with some text, and a "category" section with some images. Lastly, there is a "review" section that uses the random number state from the first useEffect hook.

2- Sign up

This code defines a functional component called Signup that renders a form to sign up a user. The component uses React hooks such as useState to manage the form state and useNavigate to handle the routing.

The form allows the user to input their name, email, and password, and it includes validation functions for each field. If the form passes validation, the form data is stored in localStorage and sent to a backend server using the axios library. If there is an error, an error message is displayed on the form.

When the form is submitted, the handleSubmit function is called, which prevents the default form submission behavior and checks if the form input is valid. If it is valid, the function stores the form data in localStorage and redirects the user to the login page. If it is not valid, the function displays an error message.

The form includes an HTML template with CSS styling imported from an external signup.css file.

The component is exported as a default export to be used in other parts of the application.

3- Login

This code defines a React component for a Login form that sends a POST request to a backend API endpoint for authentication. It uses useState hooks to manage state variables for name, email, password, and error messages. The handleSubmit function is called when the form is submitted, and it validates the name and email fields using regular expressions. If the validation passes and the API response indicates a successful login, the user data is stored in local storage and the user is redirected to the home page. If there is an error, an error message is displayed. The code also includes Bootstrap and SweetAlert libraries for styling and displaying alerts.

4- Navbar

This is a React component that represents a Navbar. It includes functionality such as a scroll -to-top button, navigation links, and login/logout functionality. The component also uses local storage to store and retrieve user data. Overall, the code provides a user interface for navigating a website and managing user authentication.

5- Footer

This is a React component called " Footer" that renders a footer section of a website. It contains HTML markup and uses Bootstrap classes for styling. The component also includes social media icons for Facebook, LinkedIn, and GitHub. The first line of the code is a comment that disables a particular eslint rule for this file.

6- Movies

This code defines a function called "Movies" using the React library. It imports a CSS file, a "Navbar" component, the "useNavigate" hook, and the "useState" and "useEffect" hooks from React.

The component sets up a state variable called "movies" using the "useState" hook, initializes it as an empty array, and then uses the "useEffect" hook to fetch data from an external API endpoint. The fetched data is then stored in the "movies" state variable using the "setMovies" function.

The component then returns a JSX element that includes the Navbar component and a list of movie cards, each displaying information about a movie fetched from the API endpoint. The movie data is mapped over using the "map" function and displayed using JSX. Finally, the component is exported as the default export for use in other parts of the application.

There is also a commented out code block at the end of the file that includes a default export for a function component that returns an empty JSX fragment.

7- Community

This function displays a Community page with posts and their associated comments. The component fetches data from a local API server using axios, and displays the posts and comments using a map function.

The component has functions to handle creating, deleting, and updating posts and comments. The UI for each post includes a button to show the comment form, and a delete button for the post and comments. The component also includes a Navbar and a Footer component.

8- Comment Form

This is a React component that creates a Comment form with a textarea input and a submit button. The component takes two props, postId and onSubmit, and manages the state of the content and userid variables using the useState hook. The userid is retrieved from local storage using the useEffect hook.

When the form is submitted, the handleSubmit function creates a new comment object, sends a POST request to the backend API using axios, calls the onSubmit prop with the response data, and resets the content variable to an empty string.

9- Post

This is a React component called "CreatePost" that allows users to create a new post. It imports React, useState and useEffect from the React library, and axios for making HTTP requests. The component creates three state variables using the useState hook: title, content and userid. The useEffect hook is used to fetch the user id from localStorage and set it to the userid state variable.

The handleSubmit function is used to handle the form submission. It constructs an HTTP POST request to a local URL using the userid state variable and sends the title and content data entered by the user. It also sets the headers for the request, including an access token from localStorage. If the request is successful, the response data is logged to the console and the onPostCreated function is called, which notifies the parent component of the new post. Finally, the title and content inputs are cleared.

The component renders a form with two inputs for title and content, and a "Create post" button.
When the form is submitted, the handleSubmit function is called.

2- BACKEND

1- Models

This code defines Django Models for a social media web application. There are four models defined: Users, Movies, Post, and Comment. The Users model has fields for name, email, password, bio, and profile -pic. The Movies model has fields for title, description, image, admin_post, and user (a foreign key to the Users model). The Post model has fields for user (a foreign key to the Users model), title, content, created_at, and updated_at. The Comment model has fields for user (a foreign key to the Users model), post (a foreign key to the Post model), content, created_at, and updated_at. Each model inherits from Django's models.Model class and defines its own set of fields.

2- Views

-This code is a Python script that contains several Django Views and serializers for implementing an API. It includes views and serializers for managing users, posts, comments, and movies.

-The code defines different class-based views for handling HTTP requests such as ListAPIView, CreateAPIView, UpdateAPIView, RetrieveAPIView, and DestroyAPIView. These views have a specific purpose such as listing, creating, updating, retrieving, and deleting objects.

-Additionally, the code includes a SignupView and login_view functions that handle user registration and authentication using JSON responses. The SignupView includes a post method that saves user information to the database and a get method that returns an error message.

-Moreover, this code implements CRUD (Create, Read, Update, Delete) operations for each model (Users, Post, Comment, Movies) with the necessary permissions. For example, CommentDeleteView checks if the user attempting to delete the comment is the author of the comment, otherwise, it returns an error message.

-Finally, the code includes a UserCommentCreateView, which creates a new comment for a specific user. The view expects a user ID to associate the comment with that user.

3- Serializers

This code defines four serializers (UserSerializer , PostSerializer , CommentSerializer , and MovieSerializer) using the Django REST Framework . The UserSerializer serializes the Users model, while the PostSerializer and CommentSerializer serialize the Post and Comment models respectively, with a nested UserSerializer to serialize the User foreign key relationship . The MovieSerializer serializes the Movies model, with a StringRelatedField to display the username of the user who created the movie . The read_only_fields attribute is set to ensure that the 'id' and 'user' fields are read-only.

4- URLs

This is a Django URL configuration code that defines different paths and views for different functionalities of a web application. It includes paths for user authentication (login and signup), user management (user list, create, update, and delete), post management (post list, create, update, and delete), comment management (comment list, create, update, and delete), movie management (movie list, create, update, and delete), and admin movie management (admin movie list, and create). The views used for each path are imported from different modules.


Thanks