MOVIE HUB

Nour El-Huda Mohamed Salama

FRONTEND

NAVBAR

The component is responsible for rendering the navigation bar on the top of the page.

- The Navbar component is implemented as a functional component using React hooks. It imports the required dependencies such as React, useState, useEffect, and useNavigate from the React Router DOM library.

- The component renders a navigation bar with a logo on the left side and a menu icon that toggles a collapsible menu on the right side. The menu items are defined in an unordered list that contains links to the Home, About, Categories, Movies, and Community pages.

- The Navbar component also contains some conditional logic to display different content based on whether the user is logged in or not. If the user is logged in, it displays a greeting message with the user's name and a logout button. Otherwise, it displays sign-in and sign-up buttons.

HOME

This is a React component for the Home page of a Movie Hub project.

- The component imports several dependencies, including React, Navbar, and useNavigate from the react-router-dom library. It also imports some image files and a CSS file for styling.

- The component defines a function that returns JSX code. It uses the useEffect hook to implement two functionalities. First, it generates a random number to display in the "Reviews" and "Watching" sections of the page every two seconds. Second, it shows a loading spinner for four seconds when the component mounts, and then it renders the rest of the content.

- The JSX code returned by the function includes a Navbar component, a header section with a welcome message, an "About" section with some text and three cards, a "Categories" section with two images side by side, and a "Review" section with three columns that display random numbers and titles.

- Overall, this Home component sets up the layout and basic content for the home page of a movie hub, and it also adds some dynamic features to enhance user experience.

SIGN UP

the component renders the signup form with the input fields and a signup button

- This is a React functional component that renders a signup form with inputs for name, email, and password. It uses useState hook to manage the state of these inputs and an axios post request to send the data to a backend server.

- The component also includes validation functions for name and email fields, which use regular expressions to ensure the input format is correct. If the input is invalid, an error message is displayed to the user.

- The handleSubmit function is called when the form is submitted. It prevents the default form submission behavior, makes an axios post request to send the data to the server and if validationFirstName and validationEmail functions return true, saves the user data to localStorage and navigates the user to the login page.

LOGIN

- The component function returns a JSX code block, which contains a form for logging in with a username, email, and password. The form uses React state hooks to track the user input values and an onSubmit event handler to process the login request.

- The handleAlertClick function is used to display a SweetAlert message when the user clicks on the "forgot password" link. If there is an error during the login process, an error message will be displayed using the error state hook.

- The code also includes a fetch call to the backend API at http://127.0.0.1:8000/login/ to authenticate the user's login credentials. If the login is successful, the user data is stored in local storage and the user is redirected to the home page. If the login fails, an error message is displayed

MOVIES

this component renders a responsive grid of movie posters with their corresponding vote averages overlaid on top of the posters

- This component is responsible for rendering the list of movies fetched from the API. The useEffect hook is used to fetch the data from the API when the component is mounted. The fetched movies are then stored in the movies state using the setMovies method.

- The component then returns a JSX template which iterates over the movies state using the map method to generate a div for each movie. The key attribute is set to the id of each movie to help React keep track of each element in the list.

- Each div element contains a div with a span that displays the vote_average of the movie, and an img tag that displays the poster_path of the movie.

COMMUNITY

community component render posts and comments

- The Community component in the Movie Hub project is responsible for rendering the posts and comments. It fetches the data from the backend API using Axios and sets the state of the posts with comments. The useState hook is used to manage the state of the comment form and posts array.

- The useEffect hook is used to fetch data from the backend API when the component mounts. It also fetches comments for each post using Promise.all() and sets the state of the posts with comments.

- The JSX code in the render function is responsible for rendering the posts and comments. It maps over the posts array and renders a Post component for each post. It also renders a CommentForm component for each post if the showCommentForm state is set to true for that post. Finally, it maps over the comments for each post and renders a Comment component for each comment.

POST

the CreatePost component is responsible for allowing users to create new posts

- This is the frontend code for the CreatePost component in the Movie Hub project. The component has three states - title, content, and userid - which are set using the useState hook. The component also uses the useEffect hook to retrieve the userid from local storage.

- The handleSubmit function is called when the form is submitted. It creates a new post by sending a POST request to the server using the axios library. The request includes the title, content, and userid in the request body, as well as an authorization header with a JWT access token. If the request is successful, the onPostCreated function is called to notify the parent component of the new post.

- The JSX code for the component includes an HTML form with input fields for the title and content, and a button to submit the form. The component also includes some CSS styles to format the input fields and button.

COMMENT

simple form for users to add comments to posts and sends the data to the server for processing.

- The component has two states defined using the useState hook: content and userid. content stores the content of the comment input field and userid stores the id of the user who is currently logged in.

- The useEffect hook is used to retrieve the user data from local storage and update the userid state with the user's id.

- The handleSubmit function is an asynchronous function that is called when the form is submitted. It prevents the default form submission behavior and sends a POST request to the server to create a new comment. The request includes the comment content, the post id, and the user id. The Authorization header includes a bearer token retrieved from local storage. If the request is successful, the onSubmit function passed down from the parent component is called with the new comment data and the content state is reset to an empty string.

BACKEND

MODELS

There are four models defined - Users, Movies, Post, and Comment.

- The Users model is used to store user information such as name, email, password, bio, and profile picture. It has a one-to-many relationship with the Movies model, as each user can create multiple movies.
- The Movies model is used to store information about movies, including the title, description, image, and whether it was added by an admin user. It has a many-to-one relationship with the Users model, as each movie belongs to one user.
- The Post model is used to store posts created by users, including the title, content, and timestamps for creation and update. It has a many-to-one relationship with the Users model, as each post is created by one user. It also has a one-to-many relationship with the Comment model, as each post can have multiple comments.
- The Comment model is used to store comments made by users on posts, including the content and timestamps for creation and update. It has a many-to-one relationship with both the Users model and the Post model, as each comment is made by one user on one post.

VIEWS

API to manage users, posts, comments, and movies.

- he views are implemented using Django's class-based views and Django Rest Framework's generic views. There are four main models: Users, Post, Comment, and Movies. Each model is associated with a serializer, which converts the model's data to and from JSON.
- The SignupView view handles user sign-up. The login_view view handles user logins. Both views accept POST requests with JSON data in the request body.
- The UserListView, UserCreateView, UserUpdateView, and UserDeleteView views handle user management. The PostListView, PostCreateView, PostUpdateView, and PostDeleteView views handle post management. The CommentListView, CommentCreateView, CommentUpdateView, CommentDeleteView, and CommentdetailView views handle comment management. The MovieListGet, MovieListCreate, MovieDetailGet, MovieDetailUpdate, and MovieDetailDelete views handle movie management.
- Each view is associated with a URL, which is defined in the urls.py file. The URLs are grouped by model and use the appropriate view for each operation (e.g., GET, POST, PUT, DELETE).

SERIALIZER

API to manage users, posts, comments, and movies.

- these serializers define how data is converted to and from JSON format for each of the corresponding Django models. They play a critical role in the communication between the frontend and backend of the Movie Hub project.

URL'S

urls.py file helps to map the appropriate URLs to the appropriate views.

ADMIN

The code registers the models Users, Movies, Post, and Comment to the Django admin site

- allows the admin to view and modify the data stored in these models. The models will be displayed on the admin dashboard and can be accessed by users with administrative permissions. The admin.site.register() function is used to register each model.