# Develop Home React Component with MovieList Integration

## Description

Create a React component named Home to integrate the MovieList component with the getMovies service. Display this component on the route "/" and implement features to show a loader while fetching data from the API and display a notification in the case of an error.

This task ensures the development of a Home component that seamlessly integrates the MovieList component with the getMovies service, providing a user-friendly experience with loaders and error notifications. The component is also properly routed to the "/" route for user access.

## Acceptance Criteria:

1. Component Creation:
   1. Create a file named Home.tsx within the components or pages folder.
   2. Develop the Home component using functional component syntax.
2. Integration with MovieList:
   1. Integrate the MovieList component within the Home component.
   2. Utilize the getMovies service to obtain the movie data to be displayed.
3. Loader Display:
   1. Show a loader (e.g., a spinner or loading message) while fetching data from the API.
   2. Ensure that the loader is displayed to the user during the data retrieval process.
4. Error Notification:
   1. Implement a mechanism to display a notification in the case of an error while fetching data from the API.
   2. The notification should convey a clear error message to the user.
5. Routing:
   1. Ensure that the Home component is displayed when the user navigates to the route "/".
   2. Use a React Router or the routing mechanism of your choice to handle the route.

## Definition of Done:

1. The Home.tsx file is created within the components or pages folder.
2. The Home component is developed using functional component syntax.
3. The MovieList component is successfully integrated within the Home component.
4. A loader is displayed to the user while fetching data from the API.
5. A notification is displayed in the case of an error while fetching data from the API.
6. The Home component is displayed when the user navigates to the route "/".

## Tip:

Consider defining the following states using the useState hook:

const [isLoading, setIsLoading] = useState(false);

const [movies, setMovies] = useState<Movie[]>([]);

const [error, setError] = useState(false);

Use these states to manage the loading state, store fetched movie data, and handle errors during the data-fetching process. Update these states based on different stages of the component's lifecycle, such as when data fetching is initiated, when data is successfully fetched, or when an error occurs. The useEffect hook can be helpful for managing side effects, such as API requests, in functional components.