

BÀI THỰC HÀNH 6

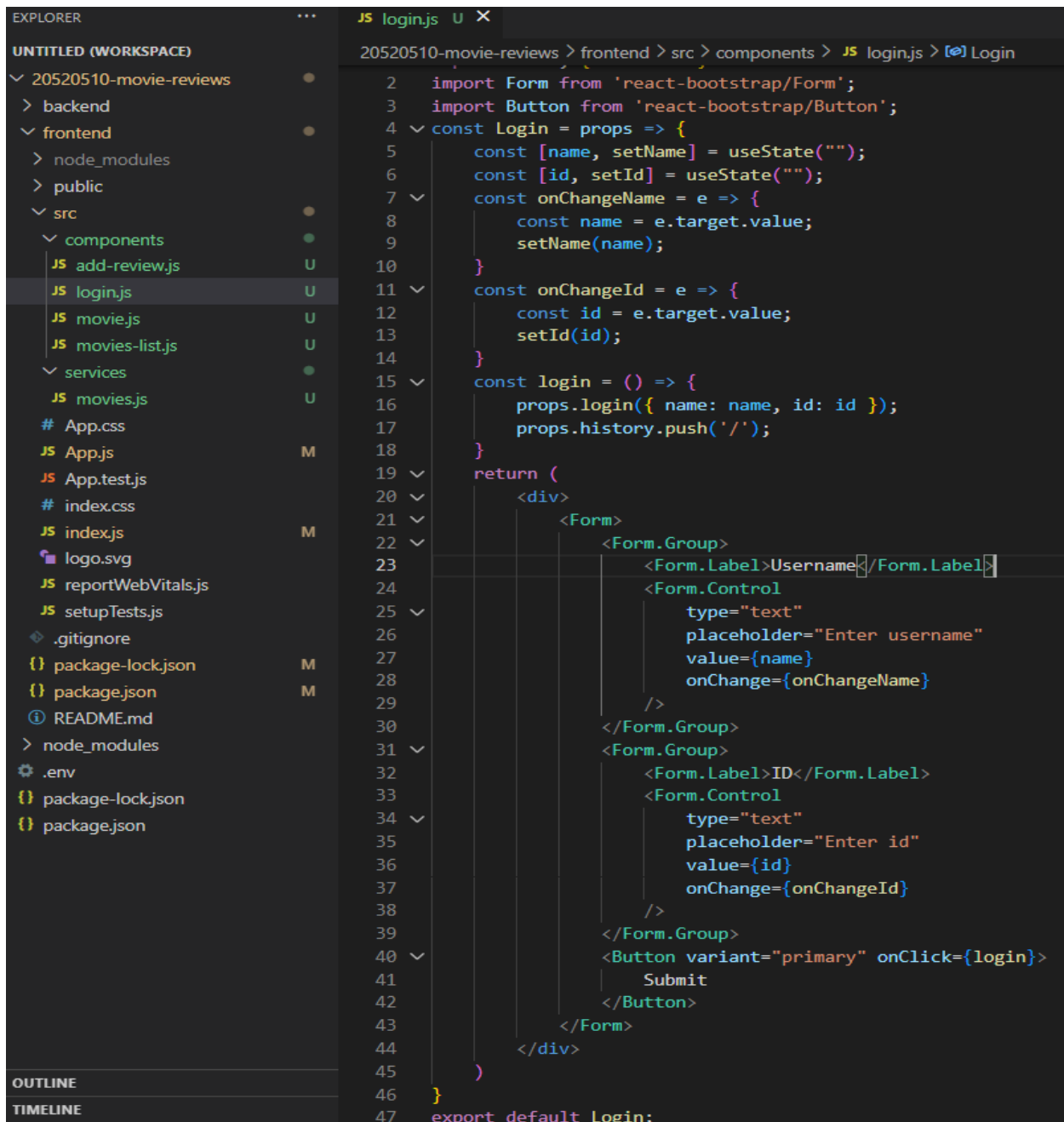
XÂY DỰNG FRONTEND VỚI REACTJS (tt)

Bài 1: Thêm và Sửa Review.

1.1 Tạo login component.

Yêu cầu khi thiết lập login component thì khi người dùng đăng nhập thành công, họ sẽ thấy được các chức năng như Edit và Delete review của chính họ.

Sau khi login thành công, người dùng sẽ được redirect về lại trang Home.



```
EXPLORER
...
20520510-movie-reviews
  backend
  frontend
    node_modules
    public
    src
      components
        JS add-review.js
        JS login.js
        JS movie.js
        JS movies-list.js
      services
        JS movies.js
      # App.css
      JS App.js
      JS App.test.js
      # index.css
      JS index.js
      logo.svg
      JS reportWebVitals.js
      JS setupTests.js
      .gitignore
      {} package-lock.json
      {} package.json
      ① README.md
    node_modules
    .env
    {} package-lock.json
    {} package.json

UNTITLED (WORKSPACE)
JS login.js
20520510-movie-reviews > frontend > src > components > JS login.js > [e] Login
2 import Form from 'react-bootstrap/Form';
3 import Button from 'react-bootstrap/Button';
4 const Login = props => {
5   const [name, setName] = useState("");
6   const [id, setId] = useState("");
7   const onChangeName = e => {
8     const name = e.target.value;
9     setName(name);
10  }
11  const onChangeId = e => {
12    const id = e.target.value;
13    setId(id);
14  }
15  const login = () => {
16    props.login({ name: name, id: id });
17    props.history.push('/');
18  }
19  return (
20    <div>
21      <Form>
22        <Form.Group>
23          <Form.Label>Username</Form.Label>
24          <Form.Control
25            type="text"
26            placeholder="Enter username"
27            value={name}
28            onChange={onChangeName}
29          />
30        </Form.Group>
31        <Form.Group>
32          <Form.Label>ID</Form.Label>
33          <Form.Control
34            type="text"
35            placeholder="Enter id"
36            value={id}
37            onChange={onChangeId}
38          />
39        </Form.Group>
40        <Button variant="primary" onClick={login}>
41          Submit
42        </Button>
43      </Form>
44    </div>
45  )
46 }
47 export default Login;
```

1.2 Thêm review

Tạo các biến như hướng dẫn sau:

- Biến `editing` sẽ có giá trị `true` khi component đang ở chế độ `Editing`.
- Ngược lại là chế độ thêm review.
- Biến trạng thái review được thiết lập thông qua biến `initialReviewState`.
- Trong chế độ `editing`, `initialReviewState` sẽ được thiết lập có nội dung `text`.
- Biến trạng thái `submitted` để theo dấu nếu như có một review được thêm mới.

Tạo các hàm phù hợp:

- Hàm `onChangeReview()` theo vết khi người dùng thêm giá trị review dưới form.
- Hàm `saveReview()` được gọi khi nút `submit` được nhấn.
- Trong hàm này, đầu tiên ta tạo một object tên `data` chứa các giá trị thuộc tính của review.
- 2 giá trị `name` và `user_id` sẽ nhận từ `props` được gửi từ `App.js`.
- Lấy `movie_id` trực tiếp từ `url` (xem lại nội dung `movie.js`).
- Sau đó gọi hàm `createReview(data)` trong `MovieDataService`.
- Định tuyến này gọi tới `ReviewsController` trong `backend` và gọi `apiPostReview()`, trích xuất data từ `request's body params`.

Phần `return()` chứa nội dung `JSX` giúp hiển thị và xử lý tính năng thêm review:

```
EXPLORER 20520510-movie-reviews > frontend > src > components > JS add-review.js > [e] AddReview > [e] saveReview
20520510-movie-reviews
  > backend
  > frontend
    > node_modules
    > public
    > src
      > components
        JS add-review.js U
        JS login.js U
        JS movie.js U
        JS movies-list.js U
      > services
        JS movies.js U
      # App.css
      JS App.js M
      JS App.test.js
      # index.css
      JS index.js M
      logo.svg
      JS reportWebVitals.js
      JS setupTests.js
      .gitignore
      {} package-lock.json M
      {} package.json M
      README.md
      > node_modules
      .env

1 import React, { useState } from 'react';
2 import MovieDataService from '../services/movies';
3 import { Link } from 'react-router-dom';
4 import Form from 'react-bootstrap/Form';
5 import Button from 'react-bootstrap/Button';
6 const AddReview = props => {
7   let editing = false;
8   let initialReviewState = '';
9   const [review, setReview] = useState(initialReviewState);
10   // keeps track if review is submitted
11   const [submitted, setSubmitted] = useState(false);
12   const onChangeReview = e => {
13     const review = e.target.value;
14     setReview(review);
15   }
16   const saveReview = () => {
17     var data = {
18       review: review,
19       name: props.user.name,
20       user_id: props.user.id,
21       movie_id: props.match.params.id // get movie id direct from url
22     };
23     MovieDataService.createReview(data)
24       .then(response => {
25         setSubmitted(true);
26       })
27       .catch(e => {
28         console.log(e);
29       })
30   }
31 }
```

```
EXPLORER 20520510-movie-reviews > frontend > src > components > JS add-review.js > [e] AddReview > [e] saveReview
20520510-movie-reviews
  > backend
  > frontend
    > node_modules
    > public
    > src
      > components
        JS add-review.js U
        JS login.js U
        JS movie.js U
        JS movies-list.js U
      > services
        JS movies.js U
      # App.css
      JS App.js M
      JS App.test.js
      # index.css
      JS index.js M
      logo.svg
      JS reportWebVitals.js
      JS setupTests.js
      .gitignore
      {} package-lock.json M
      {} package.json M
      README.md
      > node_modules
      .env

31 }
32 return (
33   <div>
34     {submitted ? (
35       <div>
36         <h4>Review submitted successfully</h4>
37         <Link to={"/movies/" + props.match.params.id}>
38           Back to Movie
39         </Link>
40       </div>
41     ) : (
42       <Form>
43         <Form.Group>
44           <Form.Label>{editing ? "Edit" : "Create"}
45             Review</Form.Label>
46           <Form.Control
47             type="text"
48             required
49             value={review}
50             onChange={onChangeReview}
51           />
52         </Form.Group>
53         <Button variant="primary" onClick={saveReview}>
54           Submit
55         </Button>
56       </Form>
57     )
58   </div>
59 );
60 }
61 export default AddReview;
```

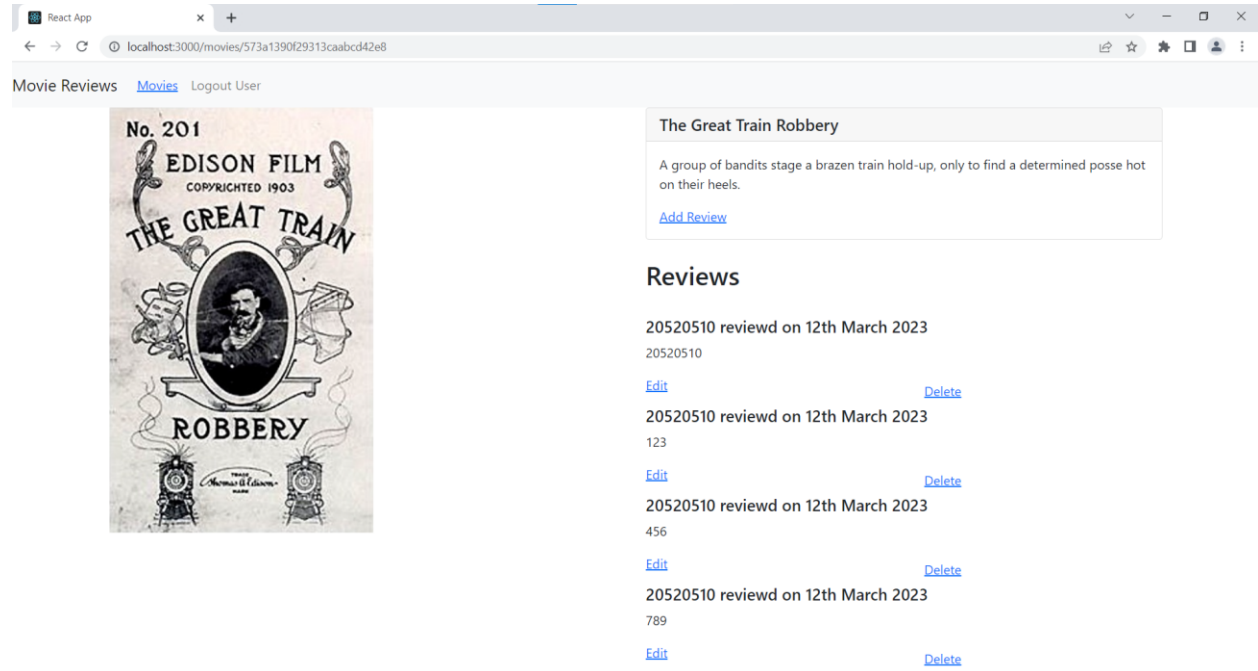
1.3 Sửa review

Viết mã nguồn thực hiện các việc sau:

- Đầu tiên, kiểm tra trạng thái truyền vào cho AddReview (xem lại tệp tin movie.js sẽ thấy prop state).
- Nếu state được truyền vào chứa thuộc tính currentReview thì chuyển editing thành true và initialReviewState thành currentReview.review.
- Nếu editing là true thì gọi updateReview() trong MovieDataService.
- Phương thức apiUpdateReview() trong ReviewsController ở backend sẽ được gọi, tương tự apiPostReview.

```
6 const AddReview = props => {
7   let editing = false;
8   let initialReviewState = "";
9   const [review, setReview] = useState(initialReviewState);
10
11   if (props.location.state && props.location.state.currentReview) {
12     editing = true;
13     initialReviewState = props.location.state.currentReview.review;
14   }
15
16   // keeps track if review is submitted
17   const [submitted, setSubmitted] = useState(false);
18   const onChangeReview = e => {
19     const review = e.target.value;
20     setReview(review);
21   }
22   const saveReview = () => {
23     var data = {
24       review: review,
25       name: props.user.name,
26       user_id: props.user.id,
27       movie_id: props.match.params.id // get movie id direct from url
28     }
29     if (editing) {
30       // get existing review id
31       data.review_id = props.location.state.currentReview._id;
32       MovieDataService.updateReview(data)
33         .then(response => {
34           setSubmitted(true);
35           console.log(response.data);
36         })
37         .catch(e => {
38           console.log(e);
39         });
40     } else {
41       MovieDataService.createReview(data)
42         .then(response => {
43           setSubmitted(true);
44         })
45         .catch(e => {
46           console.log(e);
47         });
48     }
49   }
50 }
51
```

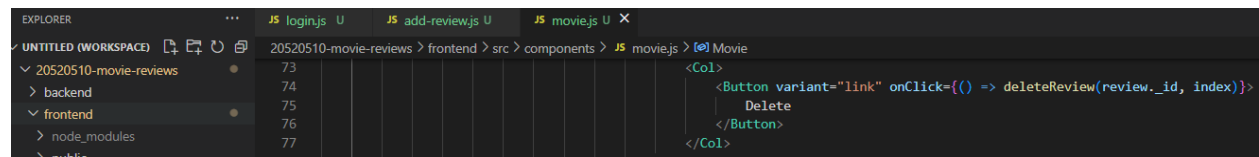
Kiểm tra chạy ứng dụng sẽ cho phép cập nhật lại review:



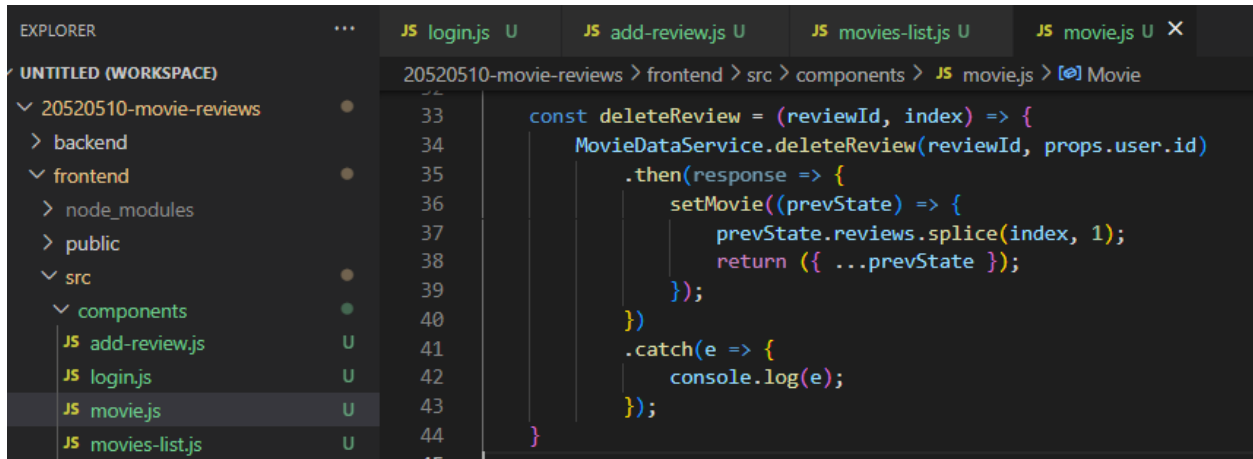
Bài 2: Xoá review

Thêm mã nguồn vào movie.js để xử lý phần xoá review:

- Trong nút delete, ta truyền review id và index chúng ta nhận được từ `movie.reviews.map()` vào phương thức `deleteReview()`.

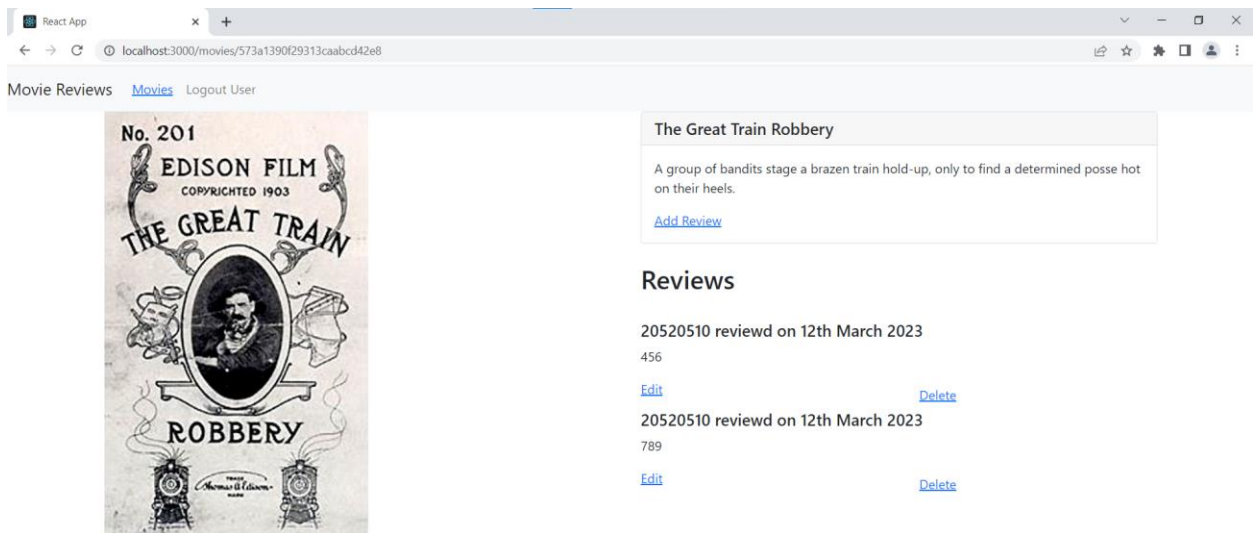


- Trong phương thức `deleteReview()`, ta gọi hàm `deleteReview()` trong `MovieDataService`.
- Sau đó, tạo một callback `.then()` sẽ được gọi sau khi `deleteReview()` được gọi xong.
- Trong callback này, ta lấy mảng các reviews trong trạng thái hiện tại. Sau đó cung cấp index của review sẽ bị xoá cho phương thức `splice()` để xoá nó đi.
- Cuối cùng là cập nhật lại mảng reviews như là trạng thái mới.



```
const deleteReview = (reviewId, index) => {
  MovieDataService.deleteReview(reviewId, props.user.id)
    .then(response => {
      setMovie((prevState) => {
        prevState.reviews.splice(index, 1);
        return ({ ...prevState });
      });
    })
    .catch(e => {
      console.log(e);
    });
}
```

Chạy lại ứng dụng -> log in -> chọn một movie cụ thể có review mà mình đã đăng.



Bài 3: Lấy dữ liệu cho trang tiếp theo

3.1 getAll()

Trong component movies-list.js thêm mã nguồn để thực hiện yêu cầu.

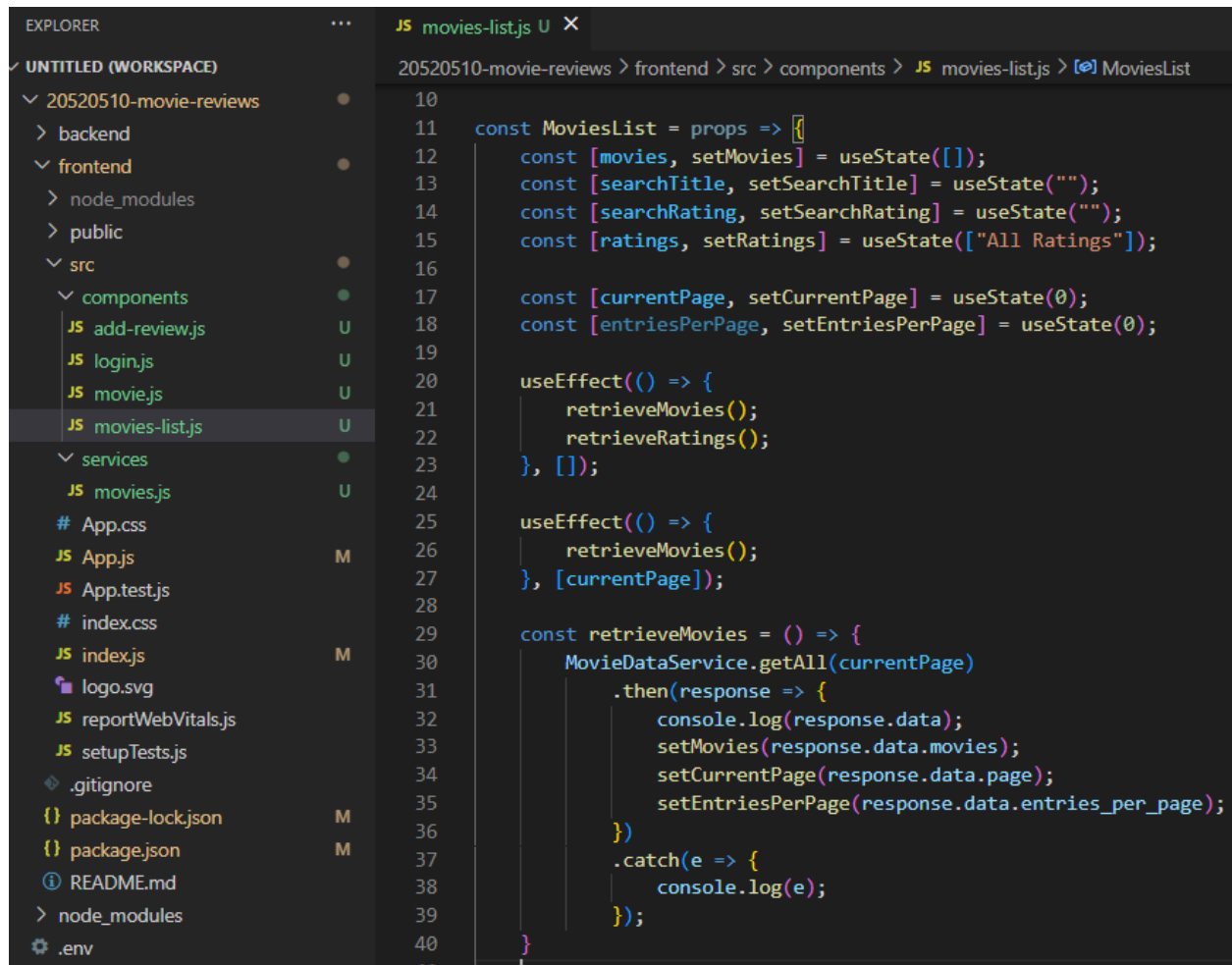
- Thêm 2 biến trạng thái currentPage và entriesPerPage.
- Thiết lập 2 biến trạng thái này trong phương thức retrieveMovies().
- Thêm một useEffect hook:

```
useEffect(() => {
  retrieveMovies();
}, []);
```

}, [currentPage]);

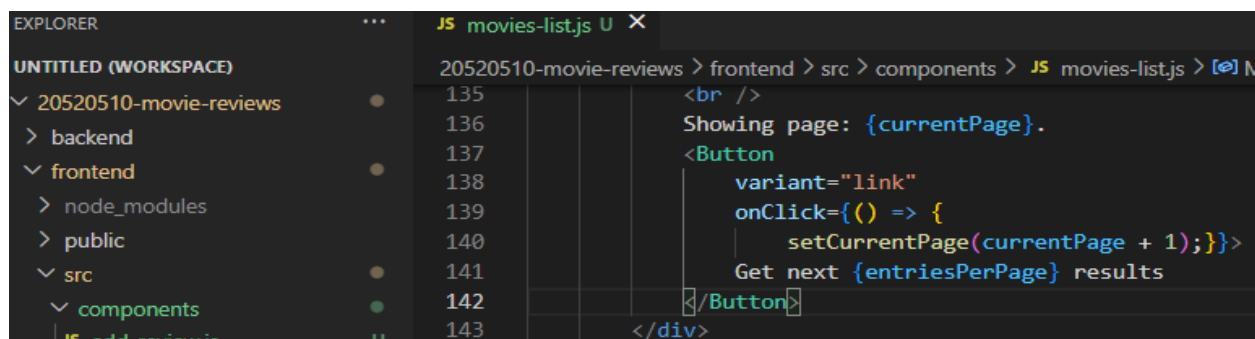
- Biến trạng thái currentPage được truyền trong tham số thứ 2 (trong mảng) nhằm mục đích khi giá trị của nó thay đổi thì hàm retrieveMovies() sẽ được gọi.

- Quan trọng: nhớ truyền currentPage vào lời gọi MovieDataService.get().



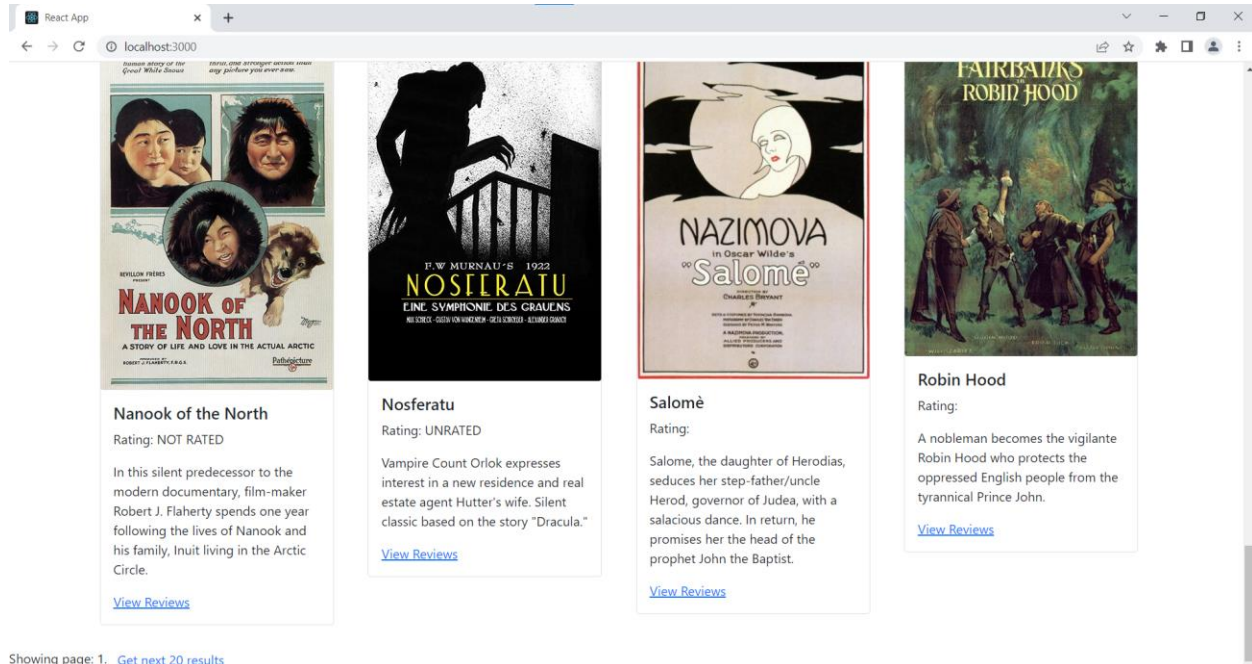
```
10
11 const MoviesList = props => {
12   const [movies, setMovies] = useState([]);
13   const [searchTitle, setSearchTitle] = useState("");
14   const [searchRating, setSearchRating] = useState("");
15   const [ratings, setRatings] = useState(["All Ratings"]);
16
17   const [currentPage, setCurrentPage] = useState(0);
18   const [entriesPerPage, setEntriesPerPage] = useState(0);
19
20   useEffect(() => {
21     retrieveMovies();
22     retrieveRatings();
23   }, []);
24
25   useEffect(() => {
26     retrieveMovies();
27   }, [currentPage]);
28
29   const retrieveMovies = () => {
30     MovieDataService.getAll(currentPage)
31       .then(response => {
32         console.log(response.data);
33         setMovies(response.data.movies);
34         setCurrentPage(response.data.page);
35         setEntriesPerPage(response.data.entries_per_page);
36       })
37       .catch(e => {
38         console.log(e);
39       });
40   }
41 }
```

- Đồng thời, thêm đoạn mã xử lý vào hàm return().



```
135 <br />
136 Showing page: {currentPage}.
137 <Button
138   variant="link"
139   onClick={() => {
140     setCurrentPage(currentPage + 1);}}>
141   Get next {entriesPerPage} results
142 </Button>
143 </div>
```

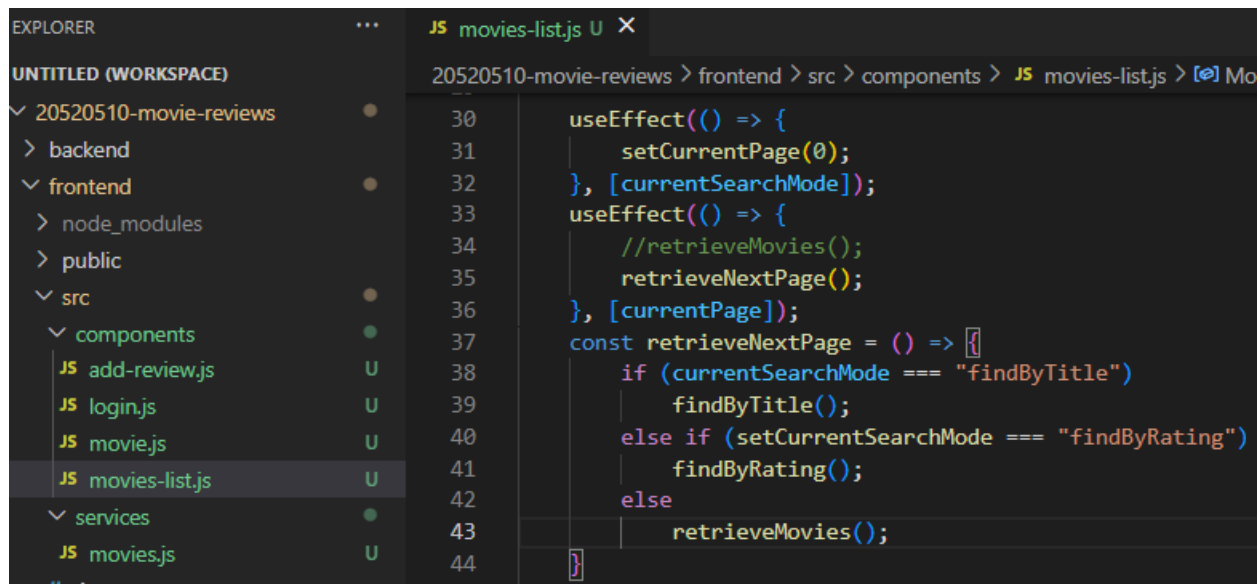
Kết quả:



3.2 find()

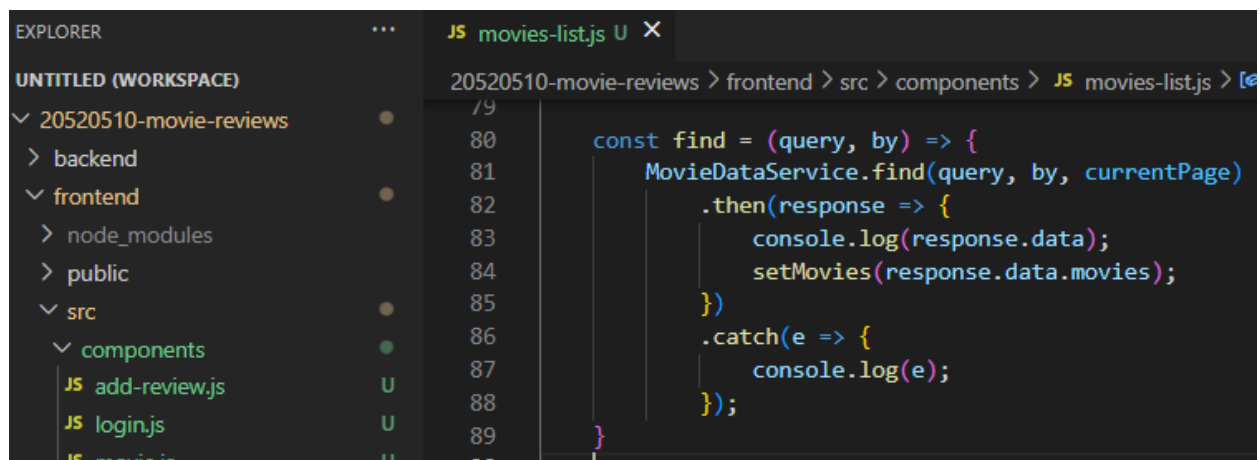
Trong movies-list.js ta điều chỉnh mã nguồn theo các yêu cầu sau:

- Đầu tiên, tạo biến trạng thái `currentSearchMode` để nhận 2 giá trị “`findByTitle`” hoặc “`findByRating`”.
- Sử dụng một `useEffect()` để khi nào `currentSearchMode` thay đổi thì thiết lập lại `currentPage` về 0.
- Tạo phương thức mới `retrieveNextPage()` -> dựa vào `currentSearchMode` để gọi các hàm tương ứng.



```
30  useEffect(() => {
31      setCurrentPage(0);
32  }, [currentSearchMode]);
33  useEffect(() => {
34      //retrieveMovies();
35      retrieveNextPage();
36  }, [currentPage]);
37  const retrieveNextPage = () => {
38      if (currentSearchMode === "findByTitle")
39          findByTitle();
40      else if (setCurrentSearchMode === "findByRating")
41          findByRating();
42      else
43          retrieveMovies();
44  }
```

- Thêm tham số `currentPage` vào trong lời gọi `MovieDataService.find()`.



```
79
80
81  const find = (query, by) => {
82      MovieDataService.find(query, by, currentPage)
83      .then(response => {
84          console.log(response.data);
85          setMovies(response.data.movies);
86      })
87      .catch(e => {
88          console.log(e);
89      });
90  }
```

- Lần lượt thêm các dòng mã lệnh `setCurrentSearchMode()` tương ứng vào 3 phương thức điều khiển:

- `retrieveMovies()`;

- `findByTitle()`;

- `findByRating()`;

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar shows the project structure: 20520510-movie-reviews > frontend > src > components. The file 'movies-list.js' is selected. The main editor shows the code for the 'retrieveMovies' function. The function is an arrow function that takes no arguments. It calls 'setCurrentSearchMode' with an empty string, then 'MovieDataService.getAll' with 'currentPage'. It then uses '.then' to handle the response, logging it, setting 'movies', 'currentPage', and 'entries_per_page'. Finally, it uses '.catch' to log any errors.

```
46 const retrieveMovies = () => {  
47   setCurrentSearchMode("");  
48   MovieDataService.getAll(currentPage)  
49     .then(response => {  
50     console.log(response.data);  
51     setMovies(response.data.movies);  
52     setCurrentPage(response.data.page);  
53     setEntriesPerPage(response.data.entries_per_page);  
54   })  
55   .catch(e => {  
56     console.log(e);  
57   });  
58 }
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar shows the project structure: 20520510-movie-reviews > frontend > src > components. The file 'movies-list.js' is selected. The main editor shows the code for the 'findByTitle' and 'findByRating' functions. The 'findByTitle' function is an arrow function that takes no arguments. It calls 'setCurrentSearchMode' with 'findByTitle' and 'find' with 'searchTitle' and 'title'. The 'findByRating' function is an arrow function that takes no arguments. It calls 'setCurrentSearchMode' with 'findByRating'. It then checks if 'searchRating' is 'All Ratings'. If so, it calls 'retrieveMovies'. Otherwise, it calls 'find' with 'searchRating' and 'rated'.

```
92 const findByTitle = () => {  
93   setCurrentSearchMode("findByTitle");  
94   find(searchTitle, "title");  
95 }  
96  
97 const findByRating = () => {  
98   setCurrentSearchMode("findByRating");  
99   if (searchRating === "All Ratings") {  
100     retrieveMovies();  
101   }  
102   else {  
103     find(searchRating, "rated");  
104   }  
105 }
```

Kết quả:

React App

localhost:3000

Movie Reviews [Movies](#) [Login](#)

Search by title


Search

Where Are My Children?

Rating: APPROVED

A District Attorney's outspoken stand on abortion gets him in trouble with the local community.

[View Reviews](#)



The Flying Carpet

Douglas Fairbanks

The THIEF BAGDAD

The Thief of Bagdad

Rating: APPROVED

A recalcitrant thief vies with a duplicitous Mongol ruler for the hand of a beautiful princess.

[View Reviews](#)

APPROVED

Search

Lady Windermere's Fan

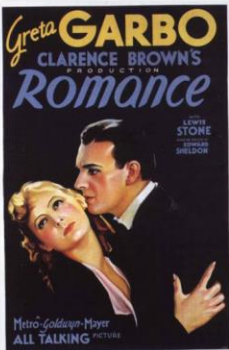
ERNEST LUBITSCH PRESENTS EDWINA COLEMAN MAY McAVOY

LOST SILENT CLASSICS COLLECTION

Lady Windermere's Fan

Rating: APPROVED

Mrs Erlynne, the mother of Lady Windermere - her daughter does not know about her - wants to be introduced in society, so that she can marry Lord Augustus Lorton.



Greta GARBO

CLARENCE BROWN'S PRODUCTION

Romance

LEWIS STONE

Metro-Goldwyn-Mayer ALL TALKING PICTURE

Romance

Rating: APPROVED

Young Harry is in love and wants to marry an actress, much to the displeasure of his family. Harry thinks that Bishop Armstrong