

ENGSE203 Mid-term (Take-home) Reactjs (Front-end) - งานเพิ่ม Movies

นำ้งานจาก LAB4 มาแก้ไข โดยเริ่มจากการสร้างไฟล์ CreateMovies.js ใน Folder Components

```
src > components > JS CreateMovie.js > (e) CreateMovies
1  import React from 'react';
2
3
4  const CreateMovies = ({onChangeForm,createMovie}) => {
5
6
7    return(
8      <div className="container">
9        <div className="row">
10          <div className="col-md-7 mrgnbtm">
11            <h2>Create Movies</h2>
12            <form>
13              <div className="row">
14                <div className="form-group col-md-8">
15                  <label htmlFor="exampleInputEmail1">Title:</label>
16                  <input type="text" onChange={(e) => onChangeForm(e)} className="form-control" name="title" id="title" aria-describedby="emailHelp" placeholder="Title" />
17                </div>
18                <div className="form-group col-md-8">
19                  <label htmlFor="exampleInputPassword1">Genre:</label>
20                  <input type="text" onChange={(e) => onChangeForm(e)} className="form-control" name="genre" id="genre" placeholder="Genre" />
21                </div>
22              </div>
23              <div className="row">
24                <div className="form-group col-md-12">
25                  <label htmlFor="exampleInputEmail1">Director:</label>
26                  <input type="text" onChange={(e) => onChangeForm(e)} className="form-control" name="director" id="director" aria-describedby="emailHelp" placeholder="Director" />
27                </div>
28              </div>
29              <div className="row">
30                <div className="form-group col-md-12">
31                  <label htmlFor="exampleInputEmail1">Release:</label>
32                  <input type="text" onChange={(e) => onChangeForm(e)} className="form-control" name="release_year" id="release_year" aria-describedby="emailHelp" placeholder="Release" />
33                </div>
34              </div>
35              <button type="button" onClick={() => createMovie()} className="btn btn-danger">Create</button>
36            </form>
37          </div>
38        </div>
39      </div>
40    )
41  }
42
43  export default CreateMovies;
```

อธิบายโค้ด

โค้ดในส่วน CreateMovies.js จะเป็นในส่วนของการสร้างแบบฟอร์ม ของหน้าเว็บ
เป็นส่วนการเพิ่มข้อมูล

Create Movies

Title

Genre

Director

Release

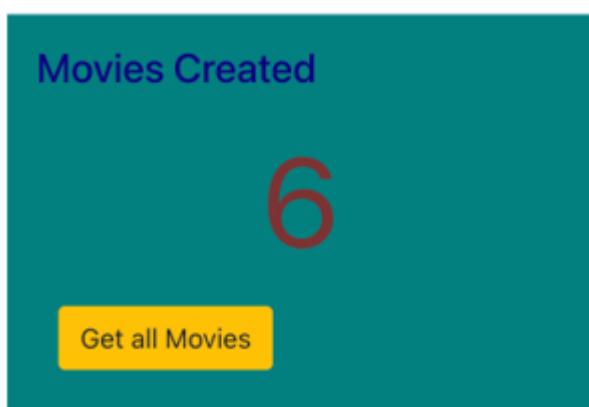
Create

ต่อมาในส่วนไฟล์ DisplayBoard.js ก็ทำการแก้ไข ตามรูป

```
src > components > JS DisplayBoard.js > DisplayBoard
1 import React from 'react'
2
3 export const DisplayBoard = ({numberOfMovies, getAllMovies}) => [
4
5   const headerStyle = {
6     width: '100%',
7     padding: '2%',
8     backgroundColor: "red",
9     color: 'white',
10    textAlign: 'center'
11  }
12
13  return(
14    <div style={{backgroundColor:'green'}} className="display-board">
15      <h4 style={{color: 'white'}}>Movies Created</h4>
16      <div className="number">
17        {numberOfMovies}
18      </div>
19      <div className="btn">
20        <button type="button" onClick={(e) => getAllMovies()} className="btn btn-warning">Get all Movies</button>
21      </div>
22    </div>
23  )
24 ]
```

อธิบายโค้ด

เป็นส่วนที่แสดงจำนวนข้อมูลที่เราทำการเพิ่มและมีปุ่ม Get all Movies เพื่อดึงข้อมูล หนังทั้งหมดให้แสดง.



ต่อไปจะเป็นส่วน Folder service

ไฟล์ MovieService.js

```
src > services > JS MovieService.js > createMovie
1  export async function getAllMovies() {
2
3      try{
4          const response = await fetch('http://localhost:3001/api/movie/all');
5
6          return await response.json();
7      }catch(error) {
8          return [];
9      }
10 }
11
12
13 export async function createMovie(data) {
14     const response = await fetch('http://localhost:4000/api/movie/insert', {
15         method: 'POST',
16         headers: {'Content-Type':'application/json'},
17         body: JSON.stringify({...data})
18     })
19     return await response.json();
20 }
```

อธิบายโค้ด

โค้ดมีสองฟังก์ชัน: getAllMovies และ createMovie ซึ่งให้บริการการเรียก API เพื่อดึงข้อมูลหนังทั้งหมดและสร้างหนังใหม่ ดังนี้:

1. ฟังก์ชัน getAllMovies:

- `export async function getAllMovies():` ประกาศฟังก์ชัน getAllMovies เพื่อทำให้สามารถนำไปใช้ในไฟล์อื่น ๆ ได้.
- `try {...}:` ในบล็อก try จะทำการเรียก API ที่ `http://localhost:3001/api/movie/all` โดยใช้ `fetch`. `fetch` เป็นฟังก์ชันที่ใช้สำหรับการทำ HTTP request และคืน Promise ที่ resolve เป็น Response object.
- `const response = await fetch('http://localhost:3001/api/movie/all');`: เรียก API ด้วย URL ที่กำหนด และรอให้ Request เสร็จสมบูรณ์โดยใช้ `await`.
- `return await response.json();`: นำ Response object ที่ได้มาแปลงเป็น JSON โดยใช้ `response.json()` และคืนค่า Promise ของ JSON นั้น.
- `catch(error) {...}:` ในการที่มีข้อผิดพลาดในการดึงข้อมูลหนัง จะคืนค่า error เรย์ว่าง.

2. ฟังก์ชัน createMovie:

- `export async function createMovie(data):` ประกาศฟังก์ชัน createMovie เพื่อทำให้สามารถนำไปใช้ในไฟล์อื่น ๆ ได้.
- `const response = await fetch('http://localhost:4000/api/movie/insert', {...})`: เรียก API สำหรับการสร้างหนังใหม่ โดยใช้ URL `http://localhost:4000/api/movie/insert` และใช้ method POST.
- `headers: {'Content-Type':'application/json'}`: กำหนด header ให้เป็น JSON.
- `body: JSON.stringify({...data})`: แปลงข้อมูลที่ถูกส่งเข้ามาเป็น JSON string

และเป็นส่วนหนึ่งของ request body.

- return await response.json();: นำ Response object ที่ได้มาแปลงเป็น JSON โดยใช้ response.json() และคืนค่า Promise ของ JSON นั้น.

Movies.js

```
src > components > JS Movies.js > [x] Movies > [x] handleSearch
1  import React, { useState } from "react";
2  import "../App.css";
3
4
5
6  export const Movies = ({ movies }) => {
7    const [searchText, setSearchText] = useState("");
8    const [searchResults, setSearchResults] = useState([]);
9    const [isSearching, setIsSearching] = useState(false);
10
11  const handleSearch = async () => {
12    try {
13      const response = await fetch(`http://localhost:3001/api/movie/search?search_text=${searchText}`);
14      const data = await response.json();
15
16      if (data.returnCode === 1) {
17        setSearchResults(data.data);
18        setIsSearching(true);
19      } else {
20        setSearchResults([]);
21        setIsSearching(false);
22      }
23    } catch (error) {
24      console.error('Error searching for movies:', error);
25    }
26  };
27
28
29  if (movies.length === 0) return null;
30
31  const MovieRow = (movie, index) => {
32    return (
33      <tr key={index} className={index % 2 === 0 ? "odd" : "even"}>
34        <td>{index + 1}</td>
35        <td>{movie.title}</td>
36        <td>{movie.genre}</td>
37        <td>{movie.director}</td>
38        <td>{movie.release_year}</td>
39      </tr>
40    );
41  };
42
43  const movieTable = isSearching ?
44    searchResults.map((movie, index) => MovieRow(movie, index)) :
45    movies.map((movie, index) => MovieRow(movie, index));
46
47  return (
48    <div className="container">
49      <div style={{ display: "flex", justifyContent: "flex-start", alignItems: "center" }}>
50        <div>
51          <h2>Movies</h2>
52          <div style={{ display: "flex", alignItems: "center" }}>
53            <input
54              type="text"
55              className="Search_input"
56              style={{
57                padding: "10px",
58                fontSize: "16px",
59                border: "1px solid #ccc",
60                borderRadius: "5px",
61                width: "300px",
62                marginRight: "10px",
63              }}
64              placeholder="Enter The Title Of Movie"
65            />
```

```

51      |
52      |      <input type="text" value={searchText} onChange={(e) => setSearchText(e.target.value)} />
53      |      <button className="btn btn-warning" style={{ padding: "10px", borderRadius: "5px" }} onClick={handleSearch}>
54      |          Search
55      |      </button>
56      |  </div>
57      |</div>
58      |<hr />
59
60      <table className="table table-bordered table-hover">
61          <thead className="table-info">
62              <tr>
63                  <th>Movie Id</th>
64                  <th>Title</th>
65                  <th>Genre</th>
66                  <th>Director</th>
67                  <th>Release</th>
68              </tr>
69          </thead>
70          <tbody>
71              {movieTable}
72              {isSearching && searchResults.length === 0 && <tr><td colSpan="5">No movies found</td></tr>}
73          </tbody>
74      </table>
75  </div>
76 );
77 };
78

```

อธิบายโค้ด

โค้ดนี้เป็นส่วนหนึ่งของ React component ที่ใช้ในการแสดงรายการหนังและระบบค้นหาหนังบนหน้าเว็บ

1. State Hooks:

- useState ใช้ในการเก็บ state ของ component นี้.
- searchText: เก็บข้อความที่ผู้ใช้ป้อนในช่องค้นหา.
- searchResults: เก็บผลลัพธ์การค้นหาหนัง.
- isSearching: เก็บค่า boolean ที่บ่งว่ากำลังมีการค้นหาหรือไม่.

2. handleSearch Function:

- ใช้ fetch เพื่อทำการค้นหาหนังจาก API โดยใช้ searchText ที่ผู้ใช้ป้อน.
- จัดการกับข้อมูลที่ได้รับ โดยตรวจสอบ returnType ถ้าเท่ากับ 1 แสดงว่าค้นหาสำเร็จ และจะแสดงผลลัพธ์.
- ในกรณีที่ returnType ไม่เท่ากับ 1 จะแสดงข้อความ "No movies found."

3. รูปแบบการแสดงผล:

- แสดง input field และ button สำหรับค้นหา.
- ใช้ Bootstrap classes เพื่อทำให้ตารางดูสวยงาม.
- แสดงข้อมูลหนังในตารางโดยใช้ฟังก์ชัน MovieRow.

4. การจัดการ State และผลลัพธ์ค้นหา:

- ใช้ตัวแปร isSearching เพื่อตรวจสอบว่ากำลังมีการค้นหาหรือไม่.
- ถ้ากำลังค้นหา จะแสดงผลลัพธ์จาก searchResults แต่ถ้าไม่กำลังค้นหาจะแสดงข้อมูลจาก movies.
- ถ้าไม่มีผลลัพธ์จากการค้นหา จะแสดงข้อความ "No movies found."

5. Event Handlers:

- onChange: เมื่อมีการเปลี่ยนแปลงใน input field จะทำการอัปเดต searchText.
- onClick: เมื่อปุ่มค้นหาถูกคลิก จะเรียกฟังก์ชัน handleSearch.

เปรียบเทียบ Movies หลังแก้กับก่อนแก้

หลังแก้

```
src> Components > R/MoviesCopy.js ...
1- import React, { useState } from "react";
2- import "../App.css";
3-
4-
5- export const Movies = ({ movies }) => {
6-   const [searchText, setSearchText] = useState("");
7-   const [searchResults, setSearchResults] = useState([]);
8-   const [isSearching, setIsSearching] = useState(false);
9-
10-   const handleSearch = async () => {
11-     try {
12-       const response = await fetch(`http://localhost:3001/api/movie/search?search_text=${searchText}`);
13-
14-       if (data.returnCode === 1) {
15-         setSearchResults(data.data);
16-         setIsSearching(true);
17-       } else {
18-         setSearchResults([]);
19-         setIsSearching(false);
20-       }
21-     } catch (error) {
22-       console.error("Error searching for movies!", error);
23-     }
24-   };
25-
26-   if (movies.length === 0) return null;
27-
28-   const MovieRow = (movie, index) => {
29-
30-     return (
31-       <tr key={index} className={index % 2 === 0 ? "odd" : "even"}>
32-         <td>{index}</td>
33-         <td>{movie.title}</td>
34-         <td>{movie.genre}</td>
35-         <td>{movie.director}</td>
36-         <td>{movie.release_year}</td>
37-       </tr>
38-     );
39-   };
40-
41-   const movieTable = isSearching ?
42-     searchResults.map((movie, index) => MovieRow(movie, index)) :
43-     movies.map((movie, index) => MovieRow(movie, index));
44-
45-   return (
46-     <div className="container">
47-       <div style={{ display: "flex", justifyContent: "flex-start", alignItems: "center" }}>
48-         <h2>Movies</h2>
49-         <div style={{ display: "flex", alignItems: "center" }}>
50-           <input type="text" placeholder="Enter The Title Of Movie"
51-             value={searchText}
52-             onChange={(e) => setSearchText(e.target.value)}
53-           />
54-           <button
55-             className="btn btn-warning"
56-             style={{ padding: "10px", borderRadius: "5px" }}
57-             onClick={handleSearch}
58-           >
59-             Search
60-           </button>
61-         </div>
62-       </div>
63-     </div>
64-
65-     <table className="table table-bordered table-hover">
66-       <thead className="table-info">
67-         <tr>
68-           <th>Movie Id</th>
69-           <th>Title</th>
70-           <th>Genre</th>
71-           <th>Director</th>
72-           <th>Release Year</th>
73-         </tr>
74-       </thead>
75-       <tbody>
76-         {movieTable}
77-       </tbody>
78-     </table>
79-   );
80- }
```

ก่อนแก้

```
→ 1- import React from "react";
2-
3- export const Movies = ({ movies }) => {
4-   console.log(`movies length ${movies.length}`);
5-
6-   if (movies.length === 0) return null;
7-   const MovieRow = (movie, index) => {
8-     return (
9-       <tr key={index} className={index % 2 === 0 ? "odd" : "even"}>
10-         <td>{index}</td>
11-         <td>{movie.title}</td>
12-         <td>{movie.genre}</td>
13-         <td>{movie.director}</td>
14-         <td>{movie.release_year}</td>
15-       </tr>
16-     );
17-   };
18-
19-   const movieTable = movies.map((movie, index) => MovieRow(movie, index));
20-
21-   return(
22-     <div className="container">
23-       <h2>Movies</h2>
24-       <table className="table table-bordered">
25-         <thead>
26-           <tr>
27-             <th>Movie Id</th>
28-             <th>Title</th>
29-             <th>Genre</th>
30-             <th>Director</th>
31-             <th>Release Year</th>
32-           </tr>
33-         </thead>
34-         <tbody>
35-           {movieTable}
36-         </tbody>
37-       </table>
38-     </div>
39-   );
40- }
```

App.js

```
src > JS App.js > App
1  import React, { useState, useEffect } from "react";
2  import "bootstrap/dist/css/bootstrap.min.css";
3  import "./App.css";
4  import { Header } from "./components/Header";
5  import { DisplayBoard } from "./components/DisplayBoard";
6  //-----
7  import { Movies } from "./components/Movies";
8  import CreateMovie from "./components/CreateMovie";
9  import { getAllMovies, createMovie } from "./services/MovieService";
10
11 function App() {
12   /*
13   const [user, setUser] = useState({})
14   const [users, setUsers] = useState([])
15   const [numberOfUsers, setNumberOfUsers] = useState(0)
16   */
17   //-----
18   const [movie, setMovie] = useState({});
19   const [movies, setMovies] = useState([]);
20   const [numberOfMovies, setNumberOfMovies] = useState(0);
21
22   /*
23   const userCreate = (e) => {
24
25     createUser(user)
26       .then(response => {
27         console.log(response);
28         setNumberOfUsers(numberOfUsers+1)
29       });
30   }
31   */
32   const movieCreate = (e) => {
33     createMovie(movie).then(response => {
34       console.log(response);
35       setNumberOfMovies(numberOfMovies + 1);
36     });
37   };
38
39   /*
40   const fetchAllUsers = () => {
41     getAllUsers()
42       .then(users => {
43         console.log(users)
44         setUsers(users);
45         setNumberOfUsers(users.length)
46       });
47   }
48   */

```

```
49  const fetchAllMovies = () => {
50    getAllMovies().then(movies) => {
51      console.log(movies);
52      setMovies(movies);
53      setNumberOfMovies(movies.length);
54    );
55  };
56
57  useEffect(() => {
58    /*
59     * getAllUsers()
60     * .then(users => {
61     *   | console.log(users)
62     *   | setUsers(users);
63     *   | setNumberOfUsers(users.length)
64     * );

```

```
src > JS App.js > ⚡ App
65  */
66  |     getAllMovies().then((movies) => {
67  |         console.log(movies);
68  |         setMovies(movies);
69  |         setNumberOfMovies(movies.length);
70  |     });
71  |     [], []
72  |
73  /*
74  const onChangeForm = (e) => {
75  |     if (e.target.name === 'firstname') {
76  |         user.firstName = e.target.value;
77  |     } else if (e.target.name === 'lastname') {
78  |         user.lastName = e.target.value;
79  |     } else if (e.target.name === 'email') {
80  |         user.email = e.target.value;
81  |     }
82  |     setUser(user)
83  }
84  */
85
86
87  const onChangeForm = (e) => {
88  |     if (e.target.name === "title") {
89  |         movie.title = e.target.value;
90  |     } else if (e.target.name === "genre") {
91  |         movie.genre = e.target.value;
92  |     } else if (e.target.name === "director") {
93  |         movie.director = e.target.value;
94  |     } else if (e.target.name === "release_year") {
95  |         movie.release_year = e.target.value;
96  |     }
97  |     setMovie(movie);
98  };
99
100 return (
101     <div className="App">
102         <Header></Header>
103         <div className="container mrgnbm">
104             <div className="row">
105                 {/* <div className="col-md-8">
106                     <CreateUser
107                         user={user}
108                         onChangeForm={onChangeForm}
109                         createUser={userCreate}
110                     >
111                     </CreateUser>
112                 </div> */}
113                 <div className="col-md-8">
114                     <CreateMovie
115                         movie={movie}
116                         onChangeForm={onChangeForm}
117                         createMovie={movieCreate}
118                     ></CreateMovie>
119                 </div>
120                 <div className="col-md-4">
121                     <DisplayBoard
122                         numberOfMovies={numberOfMovies}
123                         getAllMovies={fetchAllMovies}
124                     ></DisplayBoard>
125                 </div>
126             </div>{" "}
127             {/* <div className="row mrgnbm">
128                 <Users users={users}></Users>
```

```

129         </div> *)
130     <div className="row mrgnbtm">
131       <Movies movies={movies}></Movies>
132     </div>
133   </div>
134 );
135 ]
136
137
138 export default App;

```

อธิบายโค้ด

โค้ดนี้เป็นส่วนหนึ่งของ React ที่ใช้ในการจัดการข้อมูลหนัง โดยมีหน้าตาหน้าเว็บที่ประกอบไปด้วยการสร้างหนังใหม่, แสดงจำนวนหนังทั้งหมด, และแสดงรายการหนังโดยใช้ Component ต่าง ๆ เช่น Header, Users, DisplayBoard, CreateUser, Movies, และ CreateMovie มีการเลือกใช้

State Hooks:

- useState ใช้ในการเก็บ state ของ component นี้.
- movie: เก็บข้อมูลของหนังที่ใช้ในการสร้าง.
- movies: เก็บข้อมูลของหนังทั้งหมด.
- numberOfMovies: เก็บจำนวนหนังทั้งหมด.

Effect Hook:

- ใช้ useEffect เพื่อดึงข้อมูลหนังทั้งหมดทันทีหลังจากที่ component ถูกโหลด.

การใช้ Effect Hook เพื่อดึงข้อมูลเมื่อ Component ถูกโหลด:

- ใน useEffect, มีการเรียก fetchAllMovies เพื่อดึงข้อมูลหนังทั้งหมดทันทีหลังจาก component ถูกโหลด.

Event Handlers:

- onChangeForm: เมื่อมีการเปลี่ยนแปลงใน input field ใน form การสร้างหนัง, จะทำการอัปเดต state movie.

มีการใช้ Bootstrap classes เพื่อปรับแต่ง UI ให้ดูน่าสนใจ และมีการเรียกใช้ Component อื่น ๆ เพื่อแสดงและสร้างหนังใหม่, เพื่อแสดงจำนวนหนังทั้งหมด, เพื่อแสดงรายการหนัง

App.css

```
src > # App.css > .Search_input::placeholder
1
2  .header {
3    width: 100%;
4    padding: 2%;
5    background-color: burlywood;
6    color: white;
7    text-align: center;
8  }
9
10 .display-board {
11   width: 100%;
12   background-color: rgb(555, 200, 789);
13   padding: 5%;
14 }
15
16 .number {
17   color: rgb(123, 51, 51);
18   font-size: 75px;
19   text-align: center;
20 }
21
22 .mrgnbtm {
23   margin-top: 20px;
24 }
25 /* Search bar */
26 /* External Stylesheet or in your component's styles */
27 .Searchbar {
28   display: flex;
29   justify-content: center;
30   align-items: left;
31 }
32
33 .Search_input {
34   padding: 10px;
35   font-size: 16px;
36   border: 1px solid #ccc;
37   border-radius: 5px;
38   width: 300px;
39   margin-right: 10px; /* Optional: add some space to the right of the input */
40 }
41
42 /* Optional: add some styling for the placeholder text */
43 .Search_input::placeholder {
44   color: #999;
45 }
```

ອົບປາຍໂຄດ
ໃໝ່ໃນກາຣຕົກແຕ່ງ Code

package.json

```
{} package.json > {} browserslist
1  {
2    "name": "my-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^4.2.4",
7      "@testing-library/react": "^9.5.0",
8      "@testing-library/user-event": "^7.2.1",
9      "bootstrap": "^4.5.0",
10     "react": "^16.13.1",
11     "react-bootstrap": "^1.0.1",
12     "react-dom": "^16.13.1",
13     "react-scripts": "3.4.1"
14   },
15   > Debug
16   "scripts": {
17     "start": "PORT=3001 react-scripts start",
18     "build": "react-scripts build",
19     "test": "react-scripts test",
20     "eject": "react-scripts eject"
21   },
22   "proxy": "http://localhost:4000",
23   "eslintConfig": {
24     "extends": "react-app"
25   },
26   "browserslist": [
27     "production": [
28       ">0.2%",
29       "not dead",
30       "not op_mini all"
31     ],
32     "development": [
33       "last 1 chrome version",
34       "last 1 firefox version",
35       "last 1 safari version"
36     ]
37   }
38 }
```

อธิบายโค้ด

ใช้ตั้งค่า PONT สคริปต์สำหรับเริ่มต้นโปรเจกต์ และ กำหนด proxy ที่ใช้ในการสื่อสารกับ backend

env.js

```
JS env.js > ...
1  var env = process.env.NODE_ENV || 'development';
2  //var env = process.env.NODE_ENV || 'production';
3  module.exports = env;
4
```

อธิบายโค้ด

โค้ดนี้กำหนดค่าตัวแปร env ซึ่งเป็น environment ที่ใช้เพื่อบรุ่งว่าแอปพลิเคชันนั้นกำลังทำงานในสภาพแวดล้อมใด

1. `process.env.NODE_ENV`: นี้เป็นการอ่านค่าจาก environment variable `NODE_ENV` ที่ถูกตั้งค่าไว้ในระบบ. ส่วนใหญ่ใน Node.js, ค่านี้จะถูกใช้เพื่อบรุ่งว่าแอปพลิเคชันนั้นกำลังทำงานในสภาพแวดล้อมไหน.
2. `'development'`: ถ้า `NODE_ENV` ไม่ได้ถูกตั้งค่าหรือไม่มีค่า, ค่าเริ่มต้นของ env จะเป็น `'development'`. นั่นคือ, ถ้าไม่มีการตั้งค่า `NODE_ENV`, แอปพลิเคชันจะถือว่ากำลังทำงานในโหมดพัฒนา
3. `module.exports = env;`: ทำให้ env สามารถใช้งานได้จากภายนอก module นี้.

dbconfig.js

```
JS dbconfig.js > ...
1  var dbconfig = {
2      development: {
3          //connectionLimit : 10,
4          host      : 'localhost',
5          port      : '3306',
6          user      : 'root',
7          password  : '',
8          database  : 'moviedb'
9      },
10     production: {
11         //connectionLimit : 10,
12         host      : 'localhost',
13         port      : '3306',
14         user      : 'root',
15         password  : '',
16         database  : 'moviedb'
17     }
18 };
19 module.exports = dbconfig;
20
```

อธิบายโค้ด

โค้ดนี้เป็นการกำหนดค่าของฐานข้อมูล MySQL สำหรับสภาพแวดล้อมการพัฒนา และสภาพแวดล้อมการให้บริการในแอปพลิเคชัน

Movies.js ของ API

```
repository > js movie.js > ...
1  var mysql = require('mysql');
2  const env = require('../env.js');
3  const config = require('../dbconfig.js')[env];
4
5  /*
6  * async function getMovieList() {
7
8      var Query;
9      var pool  = mysql.createPool(config);
10
11     return new Promise((resolve, reject) => {
12
13         //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT( warehouse_name USING tis620 ) ASC `;
14         Query = `SELECT * FROM movies`;
15
16         pool.query(Query, function (error, results, fields) {
17             if (error) throw error;
18
19             if (results.length > 0) {
20                 pool.end();
21                 return resolve({
22                     statusCode: 200,
23                     returnCode: 1,
24                     data: results,
25                 });
26             } else {
27                 pool.end();
28                 return resolve({
29                     statusCode: 404,
30                     returnCode: 11,
31                     message: 'No movie found',
32                 });
33             }
34         });
35     });
36 };
37
38 });
39 */
40
41 */
42
43 async function getMovieList() {
44
45     var Query;
46     var pool  = mysql.createPool(config);
47
48     return new Promise((resolve, reject) => {
49
50         //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT( warehouse_name USING tis620 ) ASC `;
51         Query = `SELECT * FROM movies`;
52
53         pool.query(Query, function (error, results, fields) {
54             if (error) throw error;
55
56             if (results.length > 0) {
57                 pool.end();
58                 return resolve(results);
59             } else {
60                 pool.end();
61                 return resolve({
62                     statusCode: 404,
63                     returnCode: 11,
64                     message: 'No movie found',
65                 });
66             }
67         });
68     });
69 }
```

```

69      });
70
71
72  }
73
74
75
76  async function getMovieSearch(search_text) {
77
78    var Query;
79    var pool = mysql.createPool(config);
80
81    return new Promise((resolve, reject) => {
82
83      Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;
84
85      pool.query(Query, function (error, results, fields) {
86        if (error) throw error;
87
88        if (results.length > 0) {
89          pool.end();
90          return resolve({
91            statusCode: 200,
92            returnCode: 1,
93            data: results,
94          });
95        } else {
96          pool.end();
97          return resolve({
98            statusCode: 404,
99            returnCode: 11,
100           message: 'No movie found',
101         });
102       }
103     });
104   });
105
106 });
107
108
109 }
110
111 async function postMovie(p_title,p_genre,p_director,p_release_year) {
112
113   var Query;
114   var pool = mysql.createPool(config);
115
116   return new Promise((resolve, reject) => {
117
118     //Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}'`;
119
120     var post = {
121       title: p_title,
122       genre: p_genre,
123       director: p_director,
124       release_year: p_release_year
125     };
126
127     console.log('post is: ', post);
128
129
130     Query = 'INSERT INTO movies SET ?';
131     pool.query(Query, post, function (error, results, fields) {
132       //pool.query(Query, function (error, results, fields) {
133
134         // if (error) throw error;
135
136         if (error) {
137

```

```
138     console.log('error_code_msg: ', error.code+' '+error.sqlMessage);
139     pool.end();
140     return resolve({
141         statusCode: 405,
142         returnCode: 9,
143         messsage: error.code+' '+error.sqlMessage
144     });
145 }
146 else{
147     console.log('results: ', results);
148     if (results.affectedRows > 0) {
149         pool.end();
150         return resolve({
151             statusCode: 200,
152             returnCode: 1,
153             messsage: 'Movie list was inserted',
154         });
155     }
156 }
157
158
159
160 });
161
162
163 });
164
165
166 }
167
168 module.exports.MovieRepo = {
169     getMovieList,
170     getMovieSearch,
171     postMovie,
172 };
173
174
```

อธิบายโค้ด

เป็นการใช้ MySQL queries เพื่อดึง, ค้นหา, และเพิ่มข้อมูลหนังในฐานข้อมูล MySQL โดยใช้การเชื่อมต่อแบบ connection pool ที่สร้างขึ้นตอนเริ่มต้นการทำงานของแอปพลิเคชัน.

index.js

```
js index.js > [o] init
1  const hapi = require('@hapi/hapi');
2  const env = require('./env.js');
3  const Movies = require('./repository/movie');
4
5  const express = require('express');
6  const app = express();
7
8  const path = require('path');
9  ||| bodyParser = require("body-parser");
10
11 //-----
12 const api_port = 4000;
13 const web_port = 4001;
14
15
16 //----- hapi -----
17
18 console.log('Running Environment: ' + env);
19
20
21 const init = async () => {
22
23     const server = hapi.Server({
24         port: api_port,
25         host: '0.0.0.0',
26         routes: {
27             cors: true
28         }
29     });
30
31 //-----
32
33 await server.register(require('@hapi/inert'));
34
35 server.route({
36     method: "GET",
37     path: "/",
38     handler: () => {
39         return '<h3> Welcome to API Back-end Ver. 1.0.0</h3>';
40     }
41 });
42
43
44 //API: http://localhost:3001/api/movie/all
45 server.route([
46     {
47         method: 'GET',
48         path: '/api/movie/all',
49         config: {
50             cors: {
51                 origin: ['*'],
52                 additionalHeaders: ['cache-control', 'x-requested-width']
53             }
54         },
55         handler: async function (request, reply) {
56             //var param = request.query;
57             //const category_code = param.category_code;
58
59             try {
60
61                 const respondedata = await Movies.MovieRepo.getMovieList();
62                 if (respondedata.error) {
63                     return respondedata.errorMessage;
64                 } else {
65                     return respondedata;
66                 }
67             } catch (err) {
68                 return { error: err.message };
69             }
70         }
71     }
72 ]);
73
74
75 //----- express -----
76
77 app.use(bodyParser.json());
78
79 app.get('/api/movie/all', (req, res) => {
80     res.send('Hello World');
81 });
82
83
84 //----- web -----
85
86 app.listen(web_port, () => {
87     console.log(`Server is running on port ${web_port}`);
88 });
89
90
91 //----- inert -----
92
93 server.start();
94
```

```
JS index.js > [o] init > ⚡ handler
65      }
66    } catch (err) {
67      server.log(["error", "home"], err);
68      return err;
69    }
70  }
71}
72);
73
74  server.route({
75    method: 'GET',
76    path: '/api/movie/search',
77    config: {
78      cors: {
79        origin: ['http://localhost:3001'],
80        additionalHeaders: ['cache-control', 'x-requested-width']
81      }
82    },
83    handler: async function (request, reply) {
84      var param = request.query;
85      const search_text = param.search_text;
86      //const title = param.title;
87
88      try {
89
90        const respondedata = await Movies.MovieRepo.getMovieSearch(search_text);
91        if (respondedata.error) {
92          return respondedata.errorMessage;
93        } else {
94          return respondedata;
95        }
96      } catch (err) {
97        server.log(["error", "home"], err);
98        return err;
99      }
100    }
101  });
102}
103
104
105  server.route({
106    method: 'POST',
107    path: '/api/movie/insert',
108    config: {
109      payload: {
110        multipart: true,
111      },
112      cors: {
113        origin: ['*'],
114        additionalHeaders: ['cache-control', 'x-requested-width']
115      }
116    },
117    handler: async function (request, reply) {
118
119      const {
120        title,
121        genre,
122        director,
123        release_year
124      } = request.payload;
125
126      //const title = request.payload.title;
127      //const genre = request.payload.genre;
128
```

```
129      try {
130
131         const respondedata = await Movies.MovieRepo.postMovie(title, genre, director, release_year);
132         if (respondedata.error) {
133             return respondedata.errorMessage;
134         } else {
135             return respondedata;
136         }
137     } catch (err) {
138         server.log(["error", "home"], err);
139         return err;
140     }
141   }
142 );
143
144
145
146
147
148   await server.start();
149   console.log('API Server running on %s', server.info.uri);
150
151 //-----
152 };
153
154
155 process.on('unhandledRejection', (err) => {
156
157   console.log(err);
158   process.exit(1);
159 });
160
161 init();
```

อธิบายโค้ด

โค้ดนี้เป็นส่วนหนึ่งของ Backend ที่ใช้ Hapi.js และ Express.js เพื่อสร้าง API สำหรับการจัดการข้อมูลหนัง (Movies). สรุปโค้ดโดยรวมคือ API server ที่ให้บริการการจัดการข้อมูลหนังผ่าน RESTful API ที่ถูกสร้างขึ้นโดยใช้ Hapi.js และ Express.js ในบางส่วน.

รูปเว็บ

The screenshot shows a web application interface. At the top, a dark blue header bar displays the text "React With NodeJS". Below the header, there are two main sections: "Create Movies" on the left and "Movies Created" on the right.

Create Movies: This section contains input fields for Title, Genre, Director, and Release, followed by a red "Create" button.

Movies: This section includes a search bar with placeholder "Enter The Title Of Movie" and a yellow "Search" button. Below the search bar is a table with columns: Movie Id, Title, Genre, Director, and Release. The table contains 6 rows of movie data.

Movies Created: A teal-colored box on the right side of the screen displays the number "6" and a yellow "Get all Movies" button.

อธิบายการทำงาน

ทำการเพิ่มข้อมูลแล้วทำการกดที่ปุ่ม **Create** จะทำการเพิ่มข้อมูลลงใน Database
แล้วทำการกดปุ่ม **Get all Movies**

The screenshot shows the same web application interface after a new movie has been added. The "Create Movies" and "Movies" sections remain the same, but the "Movies Created" box now displays the number "7" and a yellow "Get all Movies" button.

Create Movies: The input fields show the newly added movie's details: Title "asasas22", Genre "Thanaphooowssaaaa", Director "Thanaphon3sfafaa", and Release "2020".

Movies: The search bar and table are identical to the previous screenshot, showing 7 movies in the database.

แล้วจะทำการแสดงข้อมูลที่เราทำการเพิ่มลงในตาราง จากนั้นลองพิมพ์ชื่อของหนังที่
ต้องการหาในช่องค้นหาดู แล้วกด แล้วจะ
แสดงชื่อที่เราต้องการหาขึ้นมา

Movies

Movie Id	Title	Genre	Director	Release
1	Note0009	Thanaphoo001	Thanaphoncs9	2023
2	Note055	Thanaphon90	Thanaphon3012	2019
3	note09	Thanaphon222	ssssdsds	2019