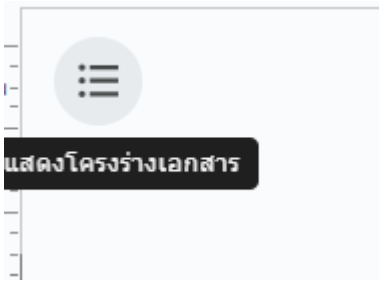
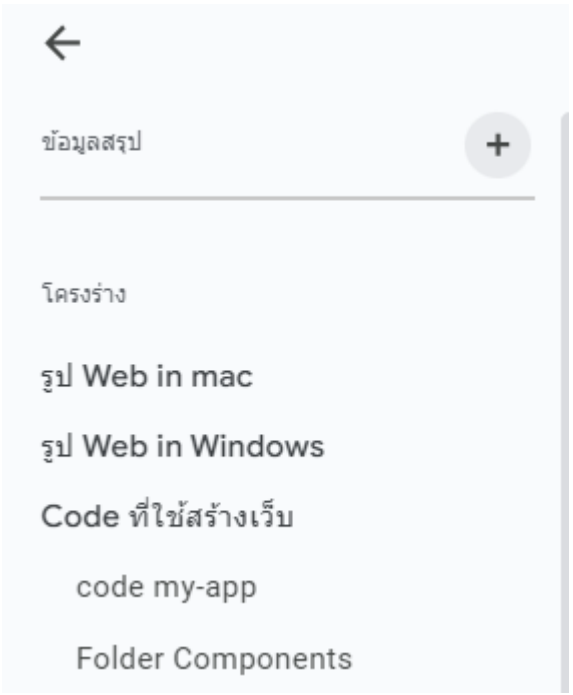
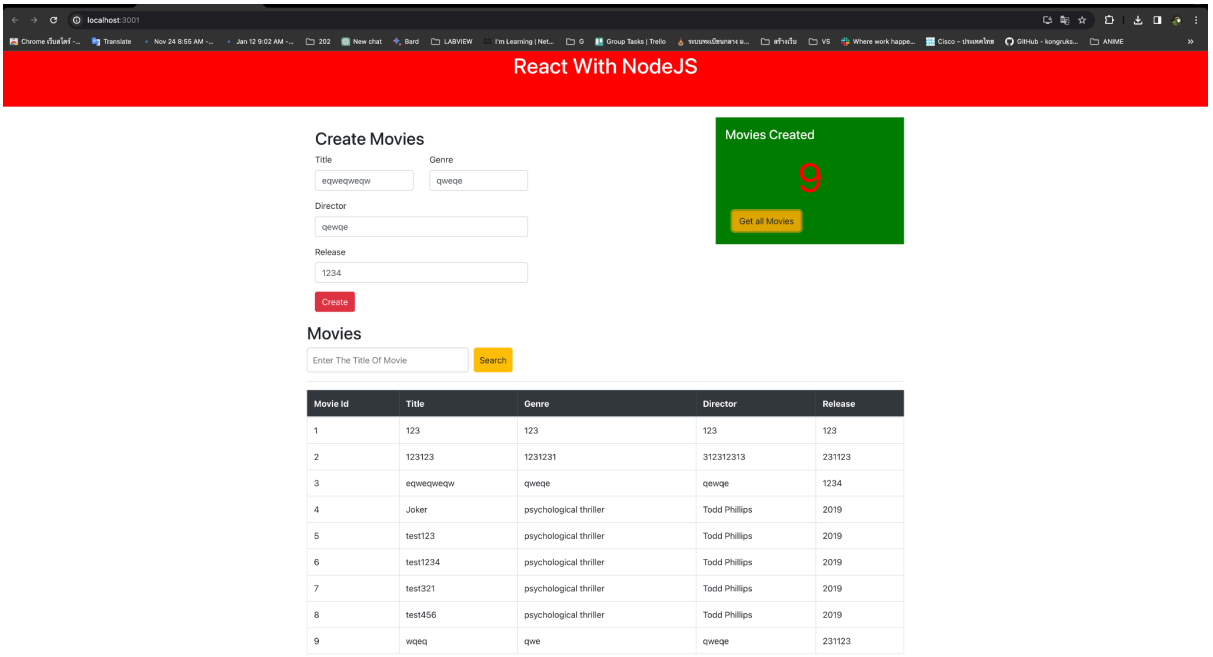
ENGSE203 - Mid-term (Take-home)

****อย่าลืมกดเปิดนะครับเพราะเอกสารมันเยอะ*****

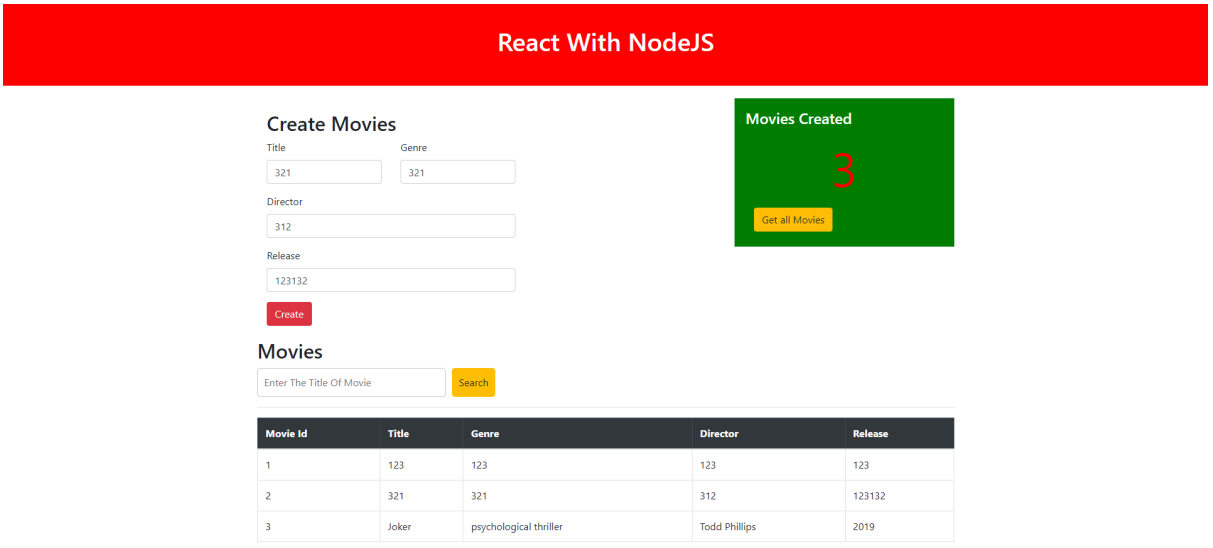


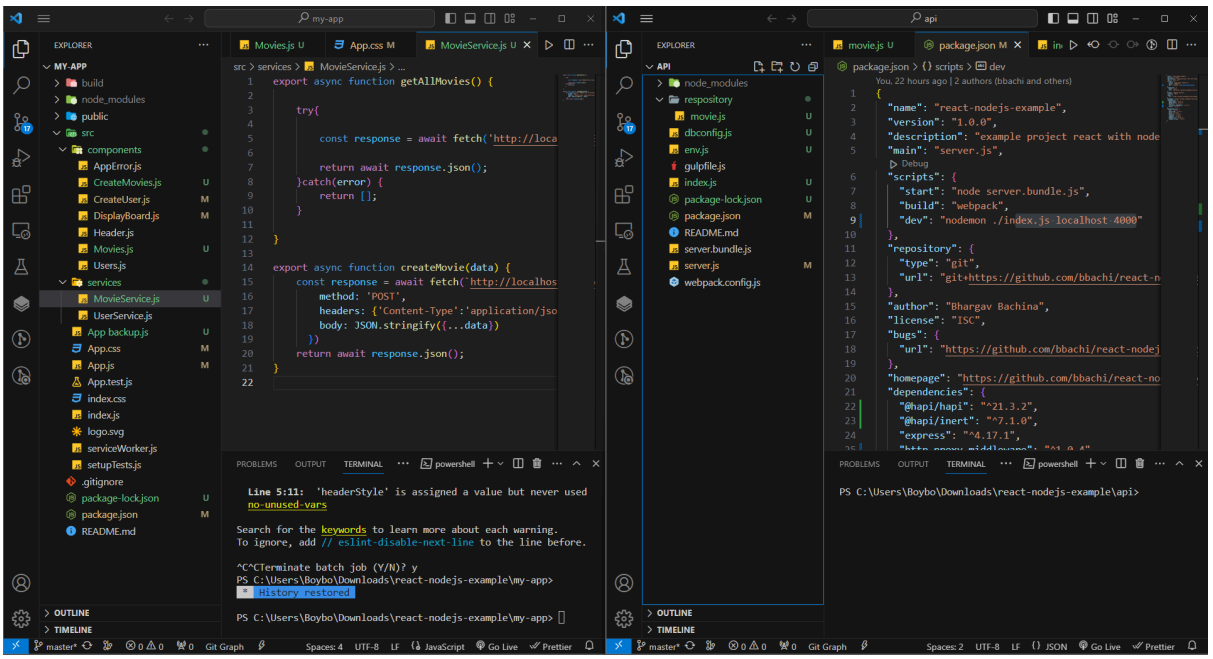


รูป Web in mac

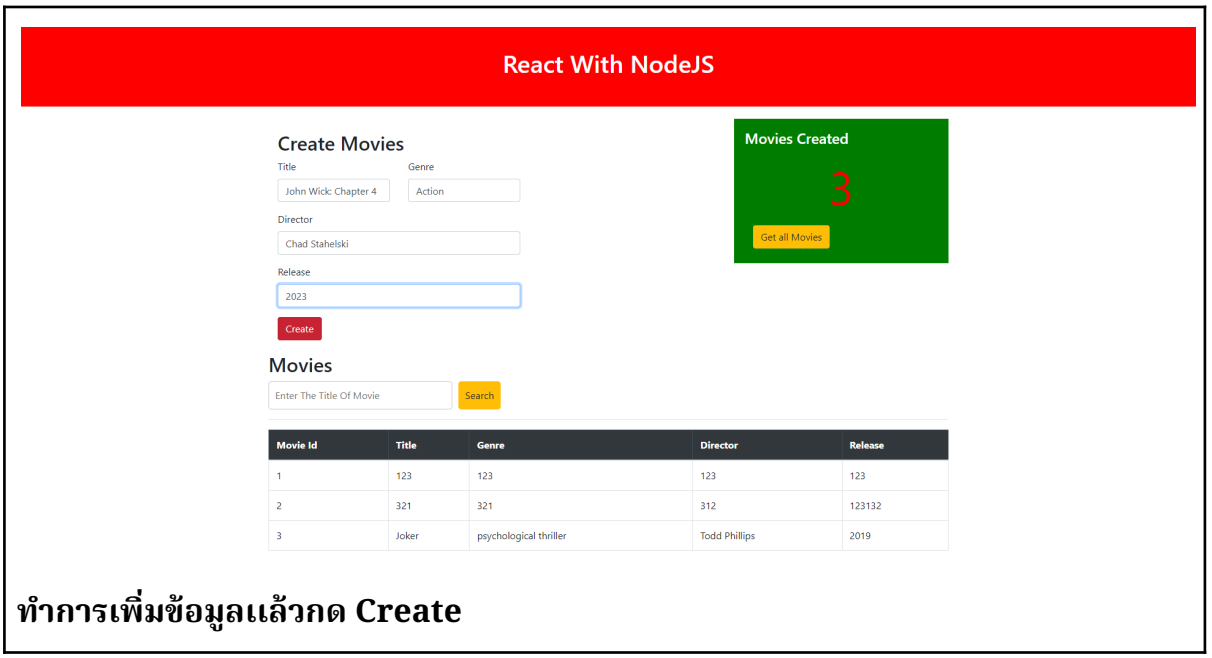


รูป Web in Windows

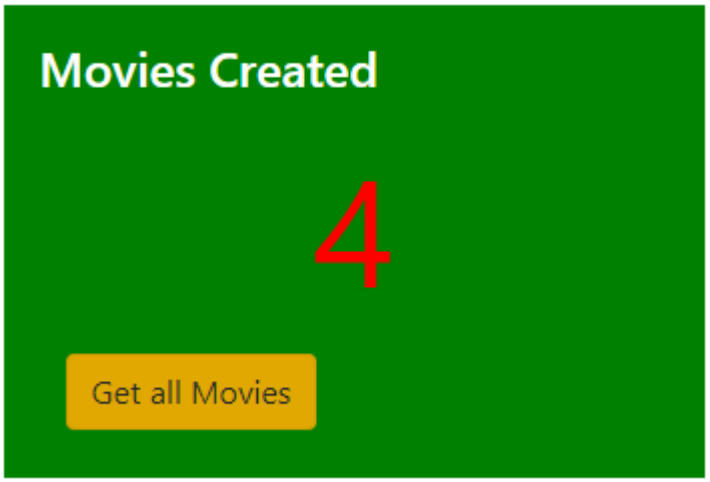




การทำงานของ Web

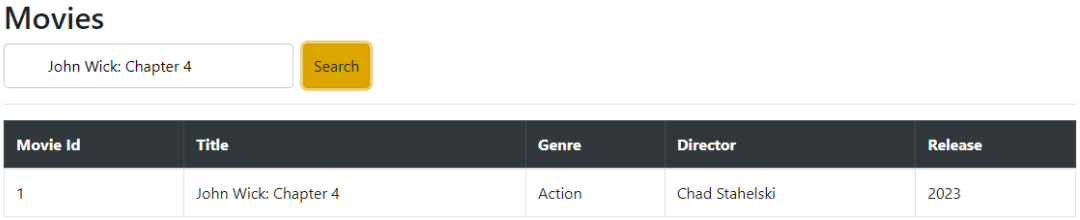


ทำการเพิ่มข้อมูลแล้วกด Create



ทำการกด Get all Movies

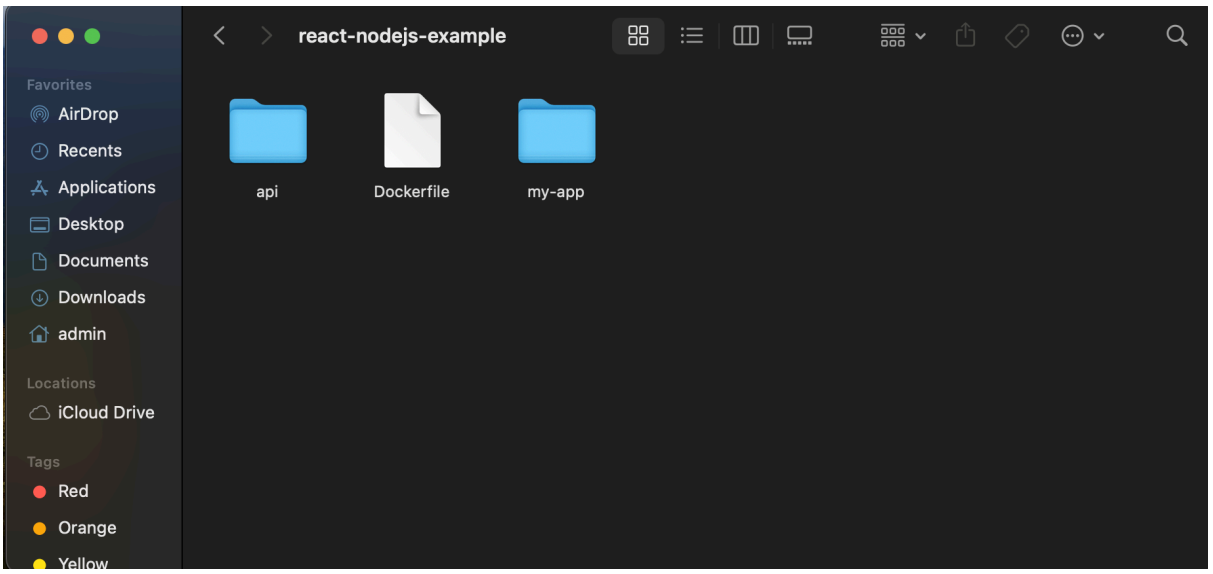
มันจะทำการอัพเดจ ข้อมูลลงไป



กดช่องค้นหาแล้ว พิมพ์ชื่อหนัง แล้วกด Search จะทำการแสดงชื่อหนัง

Code ที่ใช้สร้างเว็บ

ทำการสร้าง Folder เก็บไฟล์ api กับ my-app



นำ code จาก LAB4 มาแก้ไข

code my-app

Folder Components

CreateMovies.js

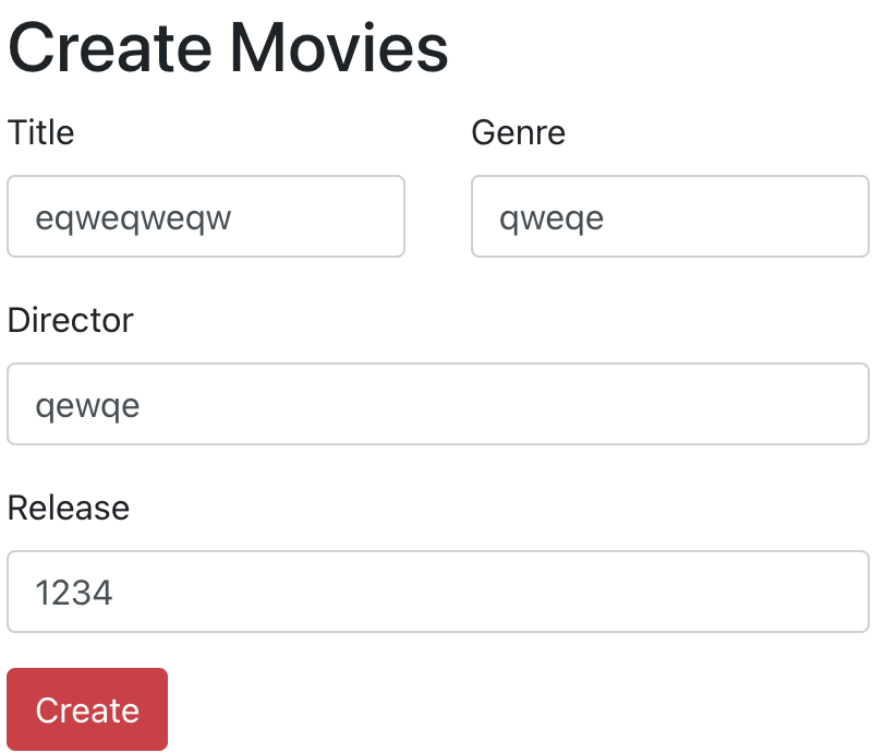
ทำการสร้าง CreateMovies.js

```
const CreateMovies = ({onChangeForm,createMovie}) => {



 return(
    <div className="container">
      <div className="row">
        <div className="col-md-7 mrgnbtm">
        <h2>Create Movies</h2>
        <form>
          <div className="row">
            <div className="form-group col-md-6">
```

```
                    <label htmlFor="exampleInputEmail1">Title</label>
                    <input type="text" onChange={(e) => onChangeForm(e)}  className="form-control"
name="title" id="title" aria-describedby="emailHelp" placeholder="Title" />
                  </div>
                  <div className="form-group col-md-6">
                    <label htmlFor="exampleInputPassword1">Genre</label>
                    <input type="text" onChange={(e) => onChangeForm(e)} className="form-control"
name="genre" id="genre" placeholder="Genre" />
                  </div>
                </div>
                <div className="row">
                  <div className="form-group col-md-12">
                    <label htmlFor="exampleInputEmail1">Director</label>
                    <input type="text" onChange={(e) => onChangeForm(e)} className="form-control"
name="director" id="director" aria-describedby="emailHelp" placeholder="Director" />
                  </div>
                </div>
                <div className="row">
                  <div className="form-group col-md-12">
                    <label htmlFor="exampleInputEmail1">Release</label>
                    <input type="text" onChange={(e) => onChangeForm(e)} className="form-control"
name="release_year" id="release_year" aria-describedby="emailHelp" placeholder="Release" />
                  </div>
                </div>
                <button type="button" onClick= {(e) => createMovie()} className="btn
btn-danger">Create</button>
              </form>
              </div>
            </div>
          </div>
    )
}
export default CreateMovies;
```

อธิบายโค้ด

โค้ดนี้คือการสร้างแบบ ฟอร์ม หน้าเว็บ



DisplayBoard.js

```
import React from 'react'


export const DisplayBoard = ({numberOfMovies, getAllMovies}) => {


  const headerStyle = {
    width: '100%',
    padding: '2%',
    backgroundColor: "red",
    color: 'white',
    textAlign: 'center'
```

```
  }


  return(
    <div style={{backgroundColor:'green'}} className="display-board">
      <h4 style={{color: 'white'}}>Movies Created</h4>
      <div className="number">
      {numberOfMovies}
      </div>
      <div className="btn">
        <button type="button" onClick={(e) => getAllMovies()} className="btn btn-warning">Get all
Movies</button>
      </div>
    </div>
  )
}
```

อธิบายโค้ด

โค้ดนี้คือ Component ของ React ชื่อ DisplayBoard ที่แสดงจำนวนหนัง และมีปุ่ม Get all Movies เพื่อดึงข้อมูลหนังทั้งหมด โดยมีสไตล์ UI ที่กำหนดไว้ด้วยสีแดงและเขียว.

Movies.js

```
import React, { useState } from "react";
import "../App.css";
export const Movies = ({ movies }) => {
 const [searchText, setSearchText] = useState('');
 const [searchResults, setSearchResults] = useState([]);
 const [isSearching, setIsSearching] = useState(false);
 const handleSearch = async () => {
```

```
  try {
    const response = await fetch(`http://localhost:3001/api/movie/search?search_text=${searchText}`);
    const data = await response.json();
    if (data.returnCode === 1) {
      setSearchResults(data.data);
      setIsSearching(true);
    } else {
      setSearchResults([]);
      setIsSearching(false);
    }
  } catch (error) {
    console.error('Error searching for movies:', error);
  }
};
if (movies.length === 0) return null;
const MovieRow = (movie, index) => {
  return (
    <tr key={index} className={index % 2 === 0 ? "odd" : "even"}>
      <td>{index + 1}</td>
      <td>{movie.title}</td>
      <td>{movie.genre}</td>
      <td>{movie.director}</td>
      <td>{movie.release_year}</td>
    </tr>
  );
};
const movieTable = isSearching ?
  searchResults.map((movie, index) => MovieRow(movie, index)) :
  movies.map((movie, index) => MovieRow(movie, index));
return (
  <div className="container">
    <div style={{ display: "flex", justifyContent: "flex-start", alignItems: "center" }}>
      <div>
```

```
    <h2>Movies</h2>
    <div style={{ display: "flex", alignItems: "center" }}>
     <input
      type="text"
      className="Search_input"
      style={{
       padding: "10px",
       fontSize: "16px",
       border: "1px solid #ccc",
       borderRadius: "5px",
       width: "300px",
       marginRight: "10px",
      }}
      placeholder="Enter The Title Of Movie"
      value={searchText}
      onChange={(e) => setSearchText(e.target.value)}
     />
     <button
      className="btn btn-warning"
      style={{ padding: "10px", borderRadius: "5px" }}
      onClick={handleSearch}
     >
      Search
     </button>
    </div>
   </div>
  </div>
  <hr />

  <table className="table table-bordered table-hover">
   <thead className="table-dark">
    <tr>
     <th>Movie Id</th>
```

```
        <th>Title</th>
        <th>Genre</th>
        <th>Director</th>
        <th>Release</th>
      </tr>
    </thead>
    <tbody>
      {movieTable}
      {isSearching && searchResults.length === 0 && <tr><td colSpan="5">No movies found</td></tr>}
    </tbody>
   </table>
  </div>
 );
};
```

อธิบายโค้ด

โค้ดนี้คือ Component **ของ** React **ชื่อ** Movies **ที่ให้บริการการแสดงข้อมูลหนังและระบบค้นหา โดยใช้** State **เพื่อเก็บข้อมูลการค้นหาและสถานะการค้นหา มีการใช้** Fetch API **เพื่อดึงข้อมูลหนังจาก** API **และมี** UI **ที่มีช่องค้นหา, ตารางข้อมูลหนัง, และการแสดงผลลัพธ์ของการค้นหา**

Folder service

MovieService.js

```
export async function getAllMovies() {
  try{
    const response = await fetch('http://localhost:3001/api/movie/all');

    return await response.json();
  }catch(error) {
    return [];
```

```
  }
}
export async function createMovie(data) {
  const response = await fetch(`http://localhost:4000/api/movie/insert`, {
     method: 'POST',
     headers: {'Content-Type':'application/json'},
     body: JSON.stringify({...data})
    })
  return await response.json();
}
```

อธิบายโค้ด

โค้ดนี้เป็นฟังก์ชันสองฟังก์ชันที่ใช้สื่อสารกับ API เพื่อดึงข้อมูลหรือสร้างหนัง

1. `export async function getAllMovies() {...}`: ฟังก์ชันนี้ใช้เพื่อดึงข้อมูลหนังทั้งหมด โดยทำการส่งคำขอไปยัง API ที่ `http://localhost:3001/api/movie/all` และรอการตอบกลับ หากสำเร็จจะคืนข้อมูลที่ได้จากการแปลง JSON จาก response นั้น แต่หากมีข้อผิดพลาดจะคืนค่าเป็น array ว่าง.
2. `export async function createMovie(data) {...}`: ฟังก์ชันนี้ใช้สำหรับการสร้างหนังใหม่ โดยทำการส่งคำขอไปยัง API ที่ `http://localhost:4000/api/movie/insert` โดยใช้วิธี POST และส่งข้อมูลในรูปแบบ JSON ที่ถูกทำการสร้างจากตัวแปร `data` ที่รับมา หลังจากนั้นรอรับการตอบกลับจาก API และคืนข้อมูลที่ได้จากการแปลง JSON จาก response นั้น.

App.js

```
import React, { useState, useEffect } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import './App.css';
import { Header } from './components/Header'
```

```
import { Users } from './components/Users'
import { DisplayBoard } from './components/DisplayBoard'
import CreateUser from './components/CreateUser'
import { getAllUsers, createUser } from './services/UserService'
//----------------
import { Movies } from './components/Movies'
import CreateMovie from './components/CreateMovies'
import { getAllMovies, createMovie } from './services/MovieService'

function App() {

 const [user, setUser] = useState({})
 const [users, setUsers] = useState([])
 const [numberOfUsers, setNumberOfUsers] = useState(0)

 //--------------
 const [movie, setMovie] = useState({})
 const [movies, setMovies] = useState([])
 const [numberOfMovies, setNumberOfMovies] = useState(0)


 const userCreate = (e) => {

    createUser(user)
      .then(response => {
        console.log(response);
        setNumberOfUsers(numberOfUsers+1)
    });
 }

 const movieCreate = (e) => {

   createMovie(movie)
    .then(response => {
      console.log(response);
      setNumberOfMovies(numberOfMovies+1)
   });
}
 const fetchAllMovies = () => {
   getAllMovies()
    .then(movies => {
      console.log(movies)
```

```
      setMovies(movies);
      setNumberOfMovies(movies.length)
    });
}

useEffect(() => {
    getAllMovies()
    .then(movies => {
      console.log(movies)
      setMovies(movies);
      setNumberOfMovies(movies.length)
    });

}, [])
const onChangeForm = (e) => {
  if (e.target.name === 'title') {
      movie.title = e.target.value;
  } else if (e.target.name === 'genre') {
      movie.genre = e.target.value;
  } else if (e.target.name === 'director') {
      movie.director = e.target.value;
  } else if (e.target.name === 'release_year') {
      movie.release_year = e.target.value;
  }
  setMovie(movie)
}
  return (
      <div className="App">
        <Header></Header>
        <div className="container mrgnbtm">
         <div className="row">
          <div className="col-md-8">
              <CreateMovie
                movie={movie}
                onChangeForm={onChangeForm}
                createMovie={movieCreate}
                >
              </CreateMovie>
          </div>
          <div className="col-md-4">
              <DisplayBoard
                numberOfMovies={numberOfMovies}
```

```
              getAllMovies={fetchAllMovies}
            >
            </DisplayBoard>
          </div>
        </div>
        <div className="row mrgnbtm">
         <Movies movies={movies}></Movies>
        </div>
        </div>
      </div>
  );
}

export default App;
```

อธิบายโค้ด

โค้ดนี้เป็นแอปพลิเคชัน React ที่ให้บริการการจัดการข้อมูลผู้ใช้และหนัง โดยใช้ Component ต่าง ๆ เช่น Header, Users, DisplayBoard, CreateUser, Movies, และ CreateMovie. มีการใช้ State และ Effect Hook เพื่อการจัดการข้อมูลและทำงานต่าง ๆ ที่เกี่ยวข้องกับการสร้างและดึงข้อมูลผู้ใช้และหนัง ผู้ใช้สามารถสร้างหนังใหม่, ดึงข้อมูลหนังทั้งหมด, และดูรายการหนังที่มีอยู่ได้ผ่าน UI ที่ถูกออกแบบเรียบง่าย.

App.css

```css
.header {
  width: 100%;
  padding: 2%;
  background-color: burlywood;
  color: white;
  text-align: center;
}

.display-board {
  width: 100%;
  background-color: rgb(555, 200, 789);
  padding: 5%;
}

.number {
  color: red;
  font-size: 75px;
  text-align: center;
}

.mrgnbtm {
  margin-top: 20px;
}

/*  Search bar  */
/* External Stylesheet or in your component's styles */
.Seachbar {
  display: flex;
  justify-content: center;
  align-items: left;
}

.Seach_input {
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 5px;
  width: 300px;
  margin-right: 10px; /* Optional: add some space to the right of t
}

/* Optional: add some styling for the placeholder text */
.Seach_input::placeholder {
  color: #999;
```

อธิบายโค้ด
ใช้ตกแต่ง code

package.json

```
{
"name": "my-app",
"version": "0.1.0",
"private": true,
"dependencies": {
  "@testing-library/jest-dom": "^4.2.4",
  "@testing-library/react": "^9.5.0",
  "@testing-library/user-event": "^7.2.1",
  "bootstrap": "^4.5.0",
  "react": "^16.13.1",
  "react-bootstrap": "^1.0.1",
  "react-dom": "^16.13.1",
  "react-scripts": "3.4.1"
},
"scripts": {
  "start": "PORT=3001 react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"proxy": "http://localhost:4000",
"eslintConfig": {
  "extends": "react-app"
},
"browserslist": {
```

```
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
}
}
```

อธิบายโค้ด

ใช้ตั้งค่า PONT สคริปต์สำหรับเริ่มต้นโปรเจ็กต์ และ กำหนด proxy ที่ใช้ในการสื่อสารกับ backend

code api

Folder respository

Movies.js

```
var mysql = require('mysql');
const env = require('../env.js');
const config = require('../dbconfig.js')[env];
async function getMovieList() {
  var Query;
  var pool  = mysql.createPool(config);
    return new Promise((resolve, reject) => {

      Query = `SELECT * FROM movies`;
```

```
      pool.query(Query, function (error, results, fields) {
        if (error) throw error;

        if (results.length > 0) {
          pool.end();
          return resolve(results);
        } else {
          pool.end();
          return resolve({
            statusCode: 404,
            returnCode: 11,
            message: 'No movie found',
          });
        }
      });
  });
}
async function getMovieSearch(search_text) {
  var Query;
  var pool  = mysql.createPool(config);

  return new Promise((resolve, reject) => {
     Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;
      pool.query(Query, function (error, results, fields) {
        if (error) throw error;

        if (results.length > 0) {
          pool.end();
          return resolve({
            statusCode: 200,
            returnCode: 1,
            data: results,
          });
```

```
        } else {
            pool.end();
            return resolve({
                statusCode: 404,
                returnCode: 11,
                message: 'No movie found',
            });
        }
    });
  });
}

async function postMovie(p_title,p_genre,p_director,p_release_year) {
  var Query;
  var pool  = mysql.createPool(config);

  return new Promise((resolve, reject) => {
    var post  = {
        title: p_title,
        genre: p_genre,
        director: p_director,
        release_year: p_release_year
    };

    console.log('post is: ', post);



    Query = 'INSERT INTO movies SET ?';
    pool.query(Query, post, function (error, results, fields) {

    if(error){
        console.log('error_code_msg: ', error.code+':'+error.sqlMessage);
        pool.end();
```

```
        return resolve({
          statusCode: 405,
          returnCode: 9,
          messsage: error.code+':'+error.sqlMessage
        });
      }
      else{
        console.log('results: ', results);
        if (results.affectedRows > 0) {
          pool.end();
          return resolve({
            statusCode: 200,
            returnCode: 1,
            messsage: 'Movie list was inserted',
          });
        }
      }
      });
  });
}


module.exports.MovieRepo = {
  getMovieList: getMovieList,
  getMovieSearch: getMovieSearch,
  postMovie: postMovie,
};
```

อธิบายโค้ด

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

dbconfig.js

```
var dbconfig = {
  development: {
    //connectionLimit : 10,
    host     : 'localhost',
    port     : '3306',
    user     : 'root',
    password : '',
    database : 'moviedb'
  },
  production: {
    //connectionLimit : 10,
    host     : 'localhost',
    port     : '3306',
    user     : 'root',
    password : '',
    database : 'moviedb'
  }
  };
module.exports = dbconfig;
```

อธิบายโค้ด

โค้ดนี้เป็นการกำหนดค่าของฐานข้อมูล MySQL สำหรับสภาพแวดล้อมการพัฒนา (development) และสภาพแวดล้อมการให้บริการ (production) ในแอปพลิเคชัน. ข้อมูลการเชื่อมต่อถูกกำหนดใน Object dbconfig

env.js

```
var env = process.env.NODE_ENV || 'development';
//var env = process.env.NODE_ENV || 'production';
module.exports = env;
```

อธิบายโค้ด

โค้ดนี้ใช้ในการกำหนดค่าของ environment ของแอปพลิเคชัน โดยใช้ process.env.NODE_ENV ถ้ามีการกำหนดค่า, ถ้าไม่ได้ระบุค่า จะใช้ค่า 'development' เป็นค่า default. ค่า environment นี้สามารถนำไปใช้กำหนดพฤติกรรมของแอปพลิเคชันในสภาพแวดล้อมที่กำหนด.

index.js

```
const hapi = require('@hapi/hapi');
const env = require('./env.js');
const Movies = require('./respository/movie');

const express = require('express');
const app = express();

const path = require('path');
    bodyParser = require("body-parser");
const api_port = 4000;
const web_port = 4001;
console.log('Running Environment: ' + env);
const init = async () => {
 const server = hapi.Server({
   port: api_port,
   host: '0.0.0.0',
   routes: {
     cors: true
```

```
  }
});
await server.register(require('@hapi/inert'));
server.route({
  method: "GET",
  path: "/",
  handler: () => {
    return '<h3> Welcome to API Back-end Ver. 1.0.0</h3>';
  }
});

  server.route({
    method: 'GET',
    path: '/api/movie/all',
    config: {
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        try {
            const responsedata = await Movies.MovieRepo.getMovieList();
            if (responsedata.error) {
                return responsedata.errMessage;
            } else {
                return responsedata;
            }
        } catch (err) {
            server.log(["error", "home"], err);
            return err;
        }
```

```
    }
});


  server.route({
    method: 'GET',
    path: '/api/movie/search',
    config: {
        cors: {
            origin: ['http://localhost:3001'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        var param = request.query;
        const search_text = param.search_text;
        try {
          const responsedata = await Movies.MovieRepo.getMovieSearch(search_text);
          if (responsedata.error) {
              return responsedata.errMessage;
          } else {
              return responsedata;
          }
      } catch (err) {
          server.log(["error", "home"], err);
          return err;
      }
    }
});
server.route({
  method: 'POST',
  path: '/api/movie/insert',
  config: {
      payload: {
```

```
        multipart: true,
      },
      cors: {
        origin: ['*'],
        additionalHeaders: ['cache-control', 'x-requested-width']
      }
    },
    handler: async function (request, reply) {

      const {
        title,
        genre,
        director,
        release_year
      } = request.payload;
      try {
        const responsedata = await Movies.MovieRepo.postMovie(title, genre, director,release_year);
        if (responsedata.error) {
          return responsedata.errMessage;
        } else {
          return responsedata;
        }
      } catch (err) {
        server.log(["error", "home"], err);
        return err;
      }

    }
});

await server.start();
console.log('API Server running on %s', server.info.uri);
```

```
};
process.on('unhandledRejection', (err) => {


 console.log(err);
 process.exit(1);
});


init();
```

อธิบายโค้ด

โค้ดนี้ใช้ Hapi.js เป็นเฟรมเวิร์คของ API และ Express.js เพื่อให้บริการหน้าเว็บ. มีการจัดการ route ต่าง ๆ สำหรับ API Endpoint และการให้บริการ static files ใน Express.js.