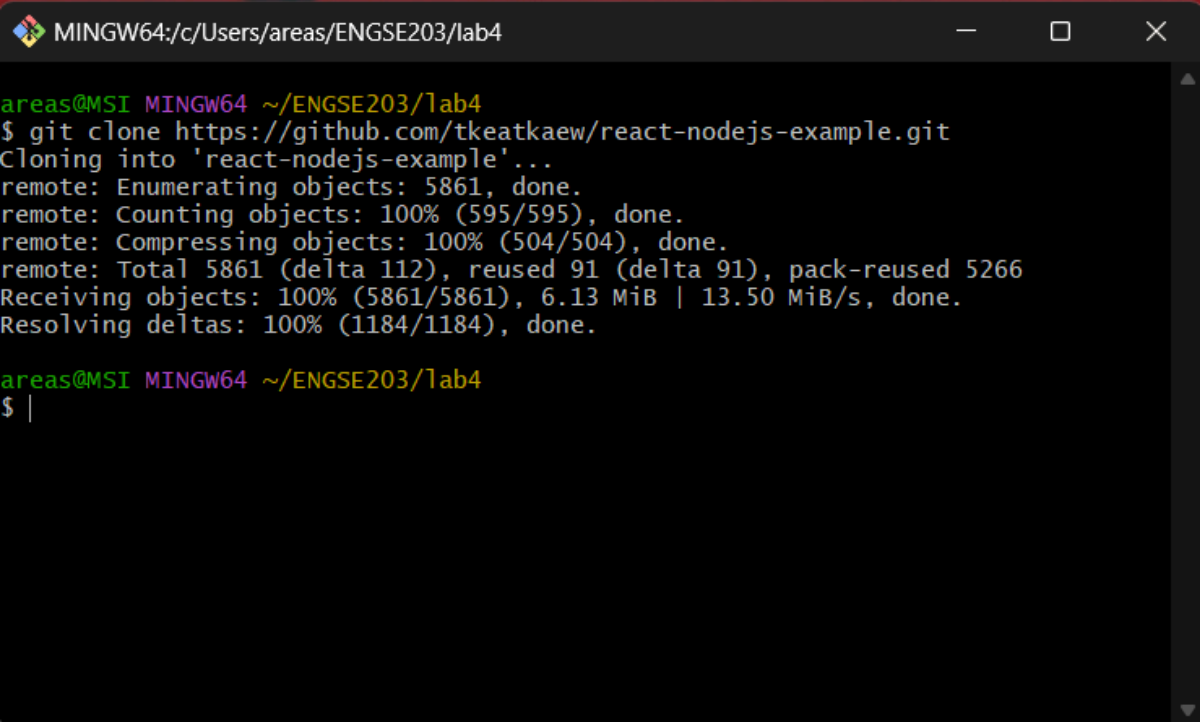


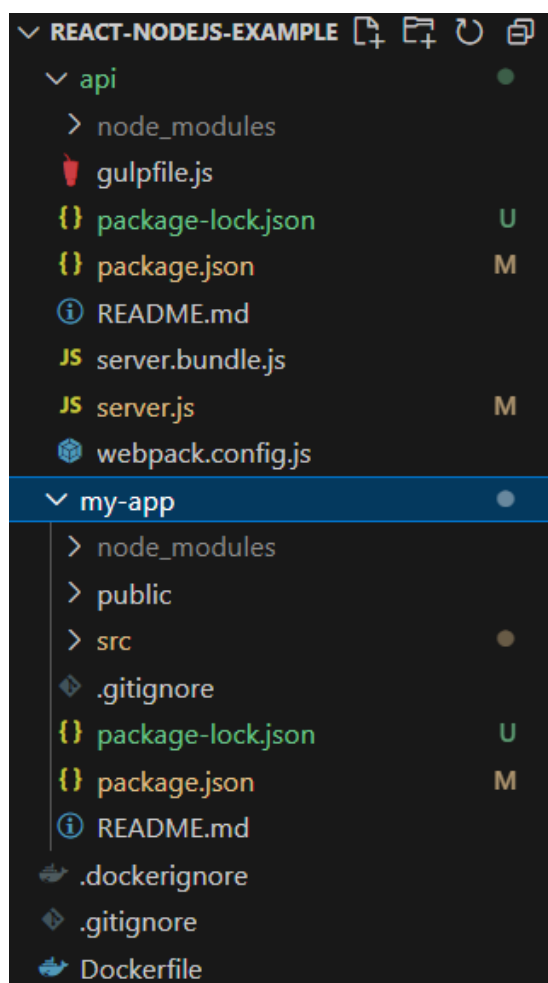
clone react-nodejs-example

- git clone <https://github.com/tkeatkaew/react-nodejs-example.git>



```
MINGW64:/c/Users/areas/ENGSE203/lab4
areas@MSI MINGW64 ~/ENGSE203/lab4
$ git clone https://github.com/tkeatkaew/react-nodejs-example.git
Cloning into 'react-nodejs-example'...
remote: Enumerating objects: 5861, done.
remote: Counting objects: 100% (595/595), done.
remote: Compressing objects: 100% (504/504), done.
remote: Total 5861 (delta 112), reused 91 (delta 91), pack-reused 5266
Receiving objects: 100% (5861/5861), 6.13 MiB | 13.50 MiB/s, done.
Resolving deltas: 100% (1184/1184), done.

areas@MSI MINGW64 ~/ENGSE203/lab4
$ |
```

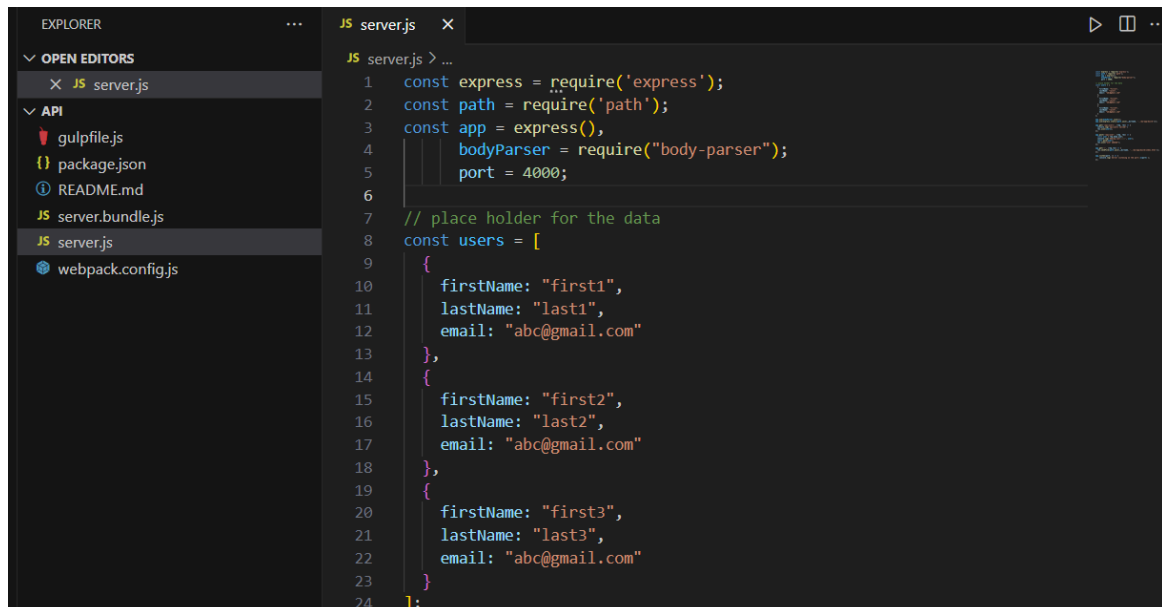


ติดตั้ง npm ที่ folder my-app , api

```
PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example\my-app> npm install  
[.....] / idealTree:my-app: sill idealTree buildDeps
```

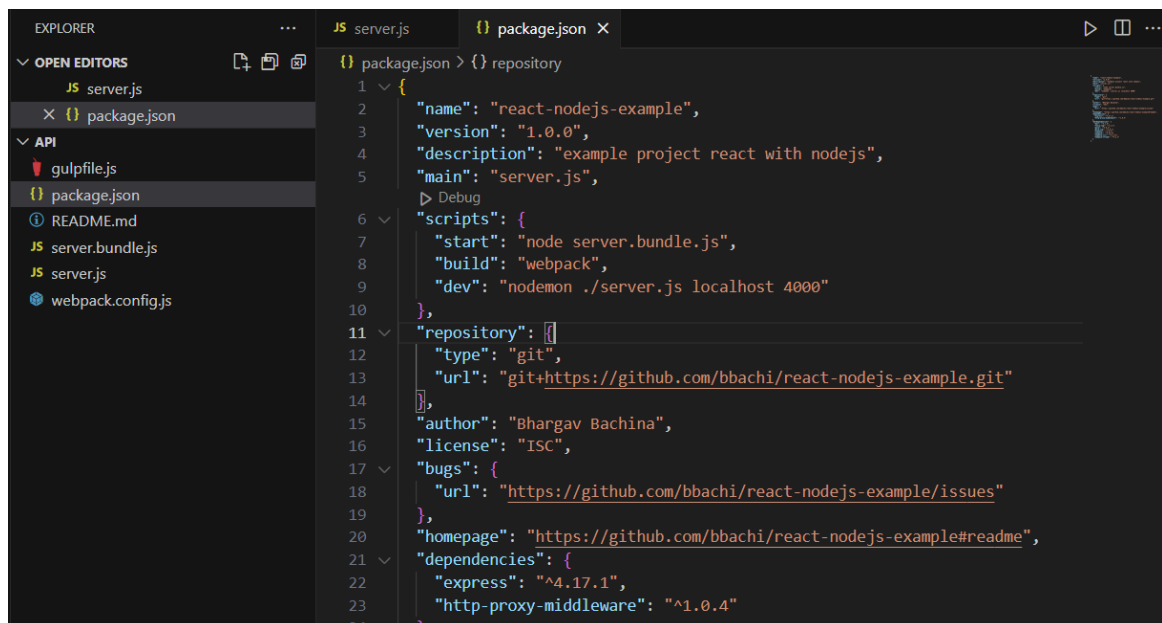
```
PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example\api> npm install  
[.....] - idealTree:api: sill idealTree buildDeps
```

แก้ไข port ที่ api



The screenshot shows the Visual Studio Code editor with the 'EXPLORER' sidebar on the left displaying the file structure of the 'API' folder, including 'server.js'. The main editor window shows the content of 'server.js'. The code defines an Express application, sets the port to 4000, and creates an array of user objects.

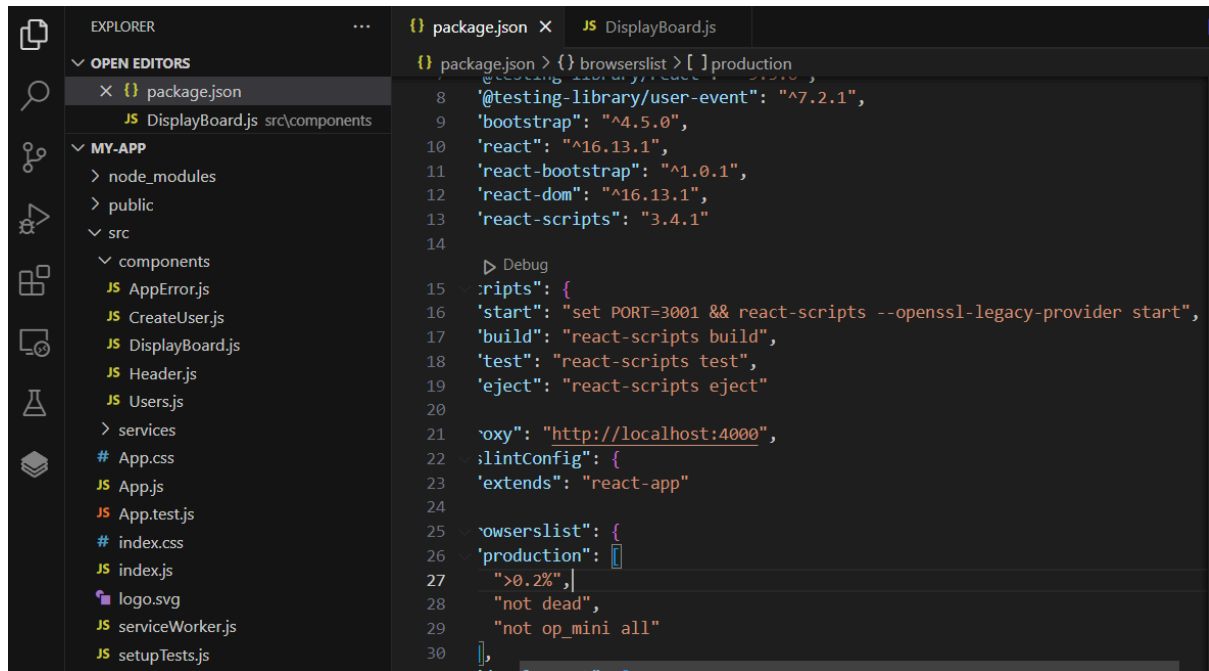
```
1 const express = require('express');  
2 const path = require('path');  
3 const app = express(),  
4     bodyParser = require("body-parser");  
5     port = 4000;  
6  
7 // place holder for the data  
8 const users = [  
9     {  
10        firstName: "first1",  
11        lastName: "last1",  
12        email: "abc@gmail.com"  
13    },  
14    {  
15        firstName: "first2",  
16        lastName: "last2",  
17        email: "abc@gmail.com"  
18    },  
19    {  
20        firstName: "first3",  
21        lastName: "last3",  
22        email: "abc@gmail.com"  
23    }  
24 ];
```



The screenshot shows the Visual Studio Code editor with the 'EXPLORER' sidebar on the left displaying the file structure of the 'API' folder, including 'package.json'. The main editor window shows the content of 'package.json'. The configuration includes project metadata, scripts for start, build, and dev, repository information, author, license, bugs, homepage, and dependencies for express and http-proxy-middleware.

```
1 {  
2   "name": "react-nodejs-example",  
3   "version": "1.0.0",  
4   "description": "example project react with nodejs",  
5   "main": "server.js",  
6   "scripts": {  
7     "start": "node server.bundle.js",  
8     "build": "webpack",  
9     "dev": "nodemon ./server.js localhost 4000"  
10  },  
11  "repository": {  
12    "type": "git",  
13    "url": "git+https://github.com/bbachi/react-nodejs-example.git"  
14  },  
15  "author": "Bhargav Bachina",  
16  "license": "ISC",  
17  "bugs": {  
18    "url": "https://github.com/bbachi/react-nodejs-example/issues"  
19  },  
20  "homepage": "https://github.com/bbachi/react-nodejs-example#readme",  
21  "dependencies": {  
22    "express": "^4.17.1",  
23    "http-proxy-middleware": "^1.0.4"  
24  }  
}
```

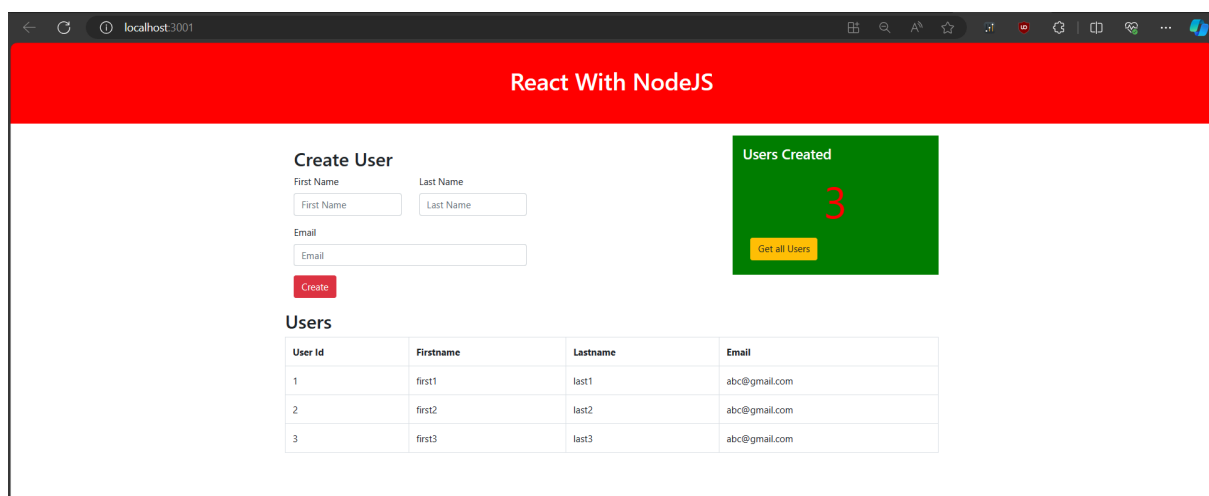
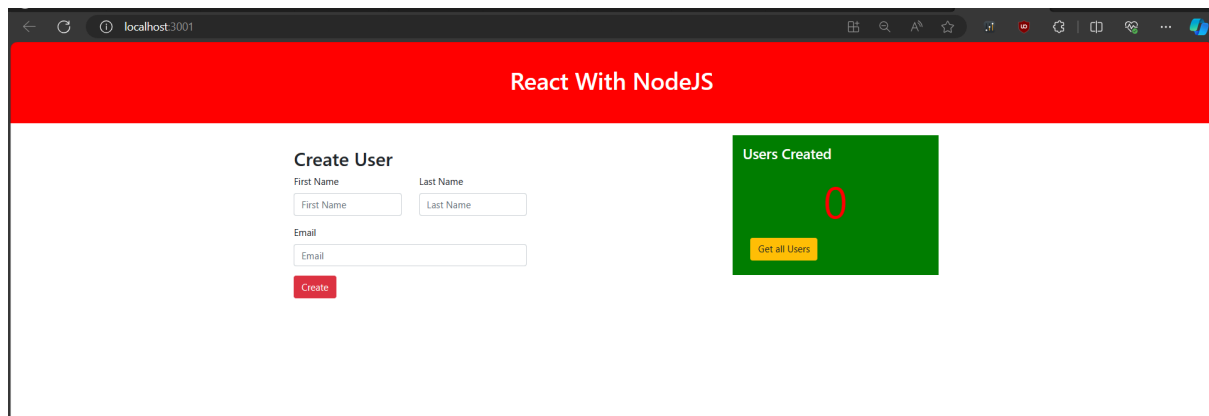
แก้ไข port ที่ my-app

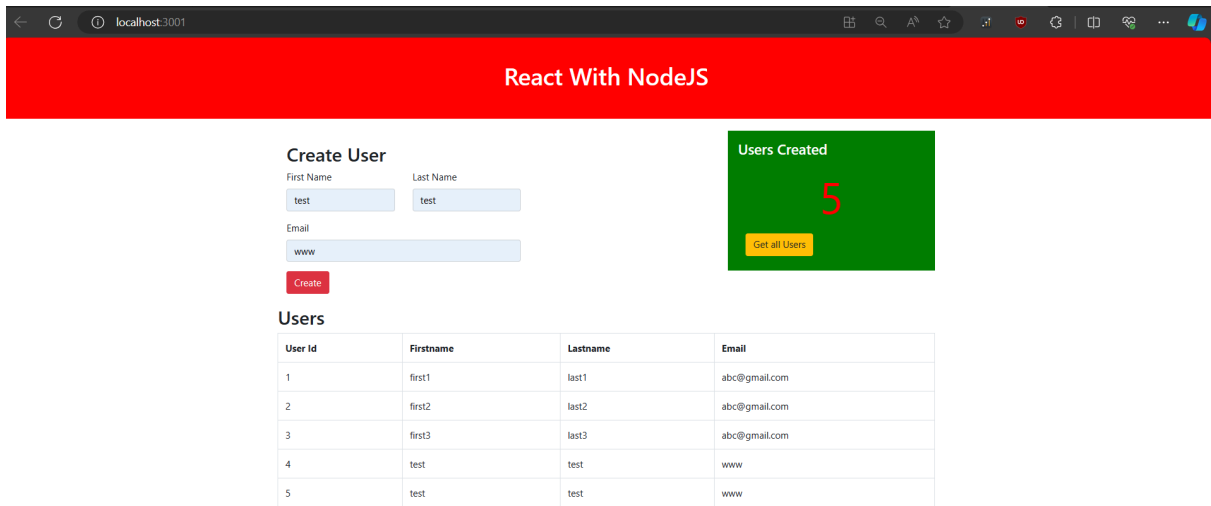


The screenshot shows the VS Code editor interface. On the left, the Explorer sidebar shows the project structure with folders like 'node_modules', 'public', and 'src'. The 'src' folder is expanded, showing subfolders like 'components' and 'services'. The 'components' folder is selected, showing files like 'AppError.js', 'CreateUser.js', 'DisplayBoard.js', 'Header.js', 'Users.js', and 'App.js'. The 'DisplayBoard.js' file is open in the editor. The main editor area shows the 'package.json' file with the following content:

```
{
  "name": "my-app",
  "version": "0.0.0",
  "private": true,
  "dependencies": {
    "@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": "set PORT=3001 && react-scripts --openssl-legacy-provider 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"
    ]
  }
}
```

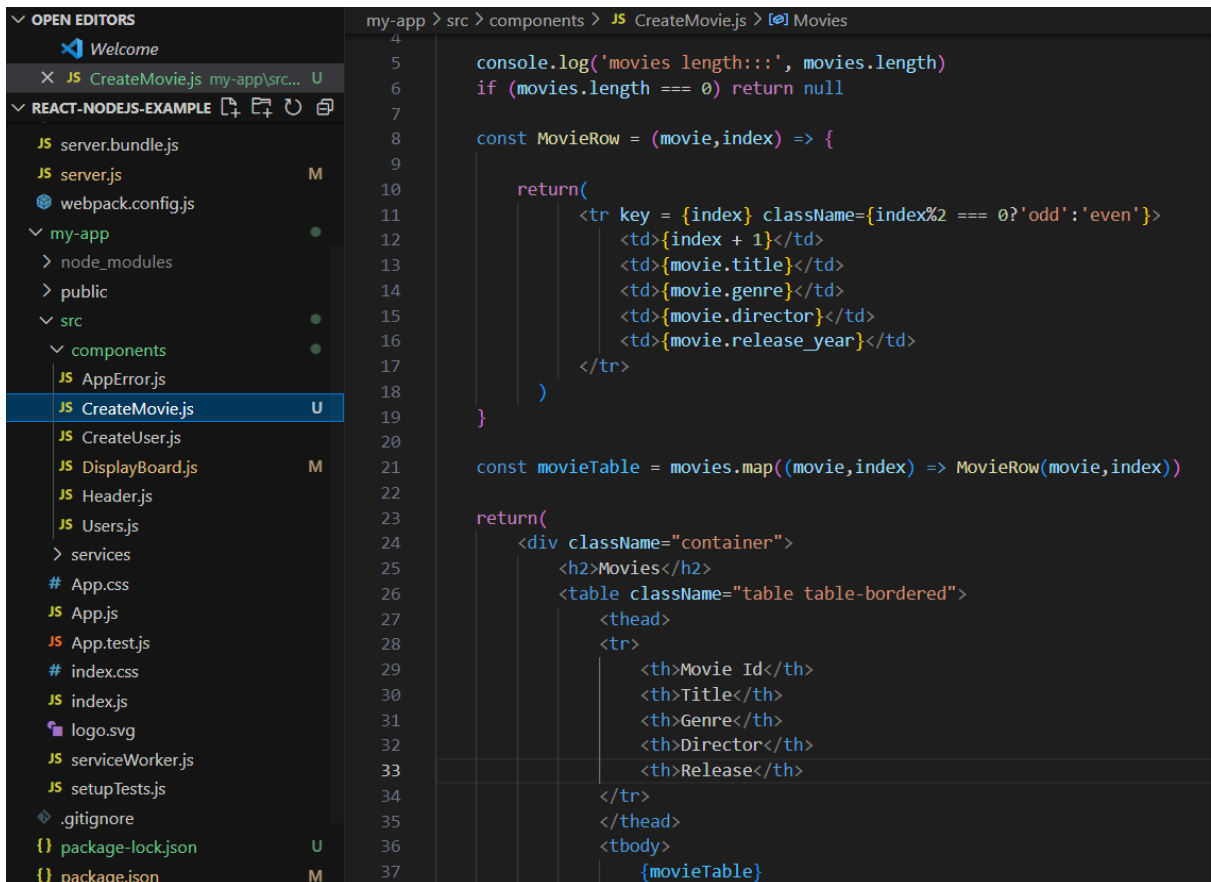
test





STEP3 (ฝั่ง Front-end เพิ่ม Movies Component)

สร้างไฟล์ **Movies.js** และ copy ข้อมูลไปใส่ในไฟล์



สร้างไฟล์ **MovieService.js** และ copy ข้อมูลไปใส่ในไฟล์

OPEN EDITORS

JS CreateMovie.js my-app\src... U

JS UserService.js my-app\src\services

X JS MovieService.js my-app\src... U

REACT-NODEJS-EXAMPLE

JS server.js M

webpack.config.js

my-app

node_modules

public

src

components

JS AppError.js

JS CreateMovie.js U

JS CreateUser.js

JS DisplayBoard.js M

JS Header.js

JS Users.js

services

JS MovieService.js U

JS UserService.js

App.css

JS App.js

JS App.test.js

index.css

my-app > src > services > JS MovieService.js > getAllMovies

1 export async function getAllMovies() {

2

3

4 try{

5 //const response = await fetch('/api/users');

6 const response = await fetch('http://localhost:3001/api/movie/all');

7 //const response = await fetch('/api/movie/all');

8 return await response.json();

9 }catch(error) {

10 return [];

11 }

12 }

13

14 export async function createMovie(data) {

15 const response = await fetch('/api/user', {

16 method: 'POST',

17 headers: {'Content-Type': 'application/json'},

18 body: JSON.stringify({user: data})

19 })

20 return await response.json();

21 }

22

app.js

OPEN EDITORS

JS CreateMovie.js my-app\src... U

JS MovieService.js my-app\src... U

X JS App.js my-app\src M

REACT-NODEJS-EXAMPLE

JS server.js M

webpack.config.js

my-app

node_modules

public

src

components

JS AppError.js

JS CreateMovie.js U

JS CreateUser.js

JS DisplayBoard.js M

JS Header.js

JS Users.js

services

JS MovieService.js U

JS UserService.js

App.css

JS App.js M

JS App.test.js

index.css

JS index.js

logo.svg

JS serviceWorker.js

JS setupTests.js

.gitignore

package-lock.json

my-app > src > JS App.js > ...

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 { Users } from './components/Users'

6 import { DisplayBoard } from './components/DisplayBoard'

7 import CreateUser from './components/CreateUser'

8 import { getAllUsers, createUser } from './services/UserService'

9 //-----

10 import { Movies } from './components/Movies'

11 import { getAllMovies, createMovie } from './services/MovieService'

12

13 function App() {

14

15 const [user, setUser] = useState({})

16 const [users, setUsers] = useState([])

17 const [numberOfUsers, setNumberOfUsers] = useState(0)

18

19 //-----

20

21 const [movies, setMovies] = useState([])

22

23

24 const userCreate = (e) => {

25

26 createUser(user)

27 .then(response => {

28 console.log(response);

29 setNumberOfUsers(numberOfUsers+1)

30 });

31 }

32

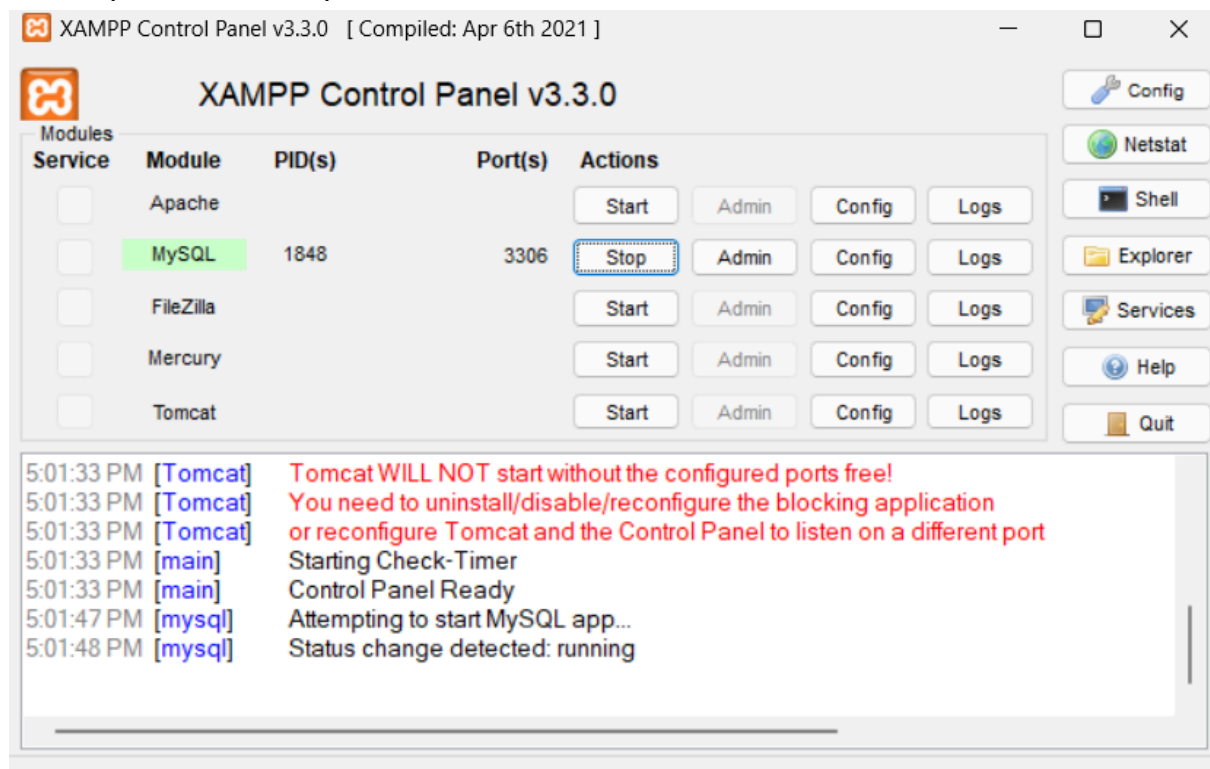
33 const fetchAllUsers = () => {

34 getAllUsers()

The screenshot shows a VS Code editor with a project named 'my-app'. The file explorer on the left shows the project structure, including 'src' and 'components' folders. The main editor displays the 'App.js' file, which contains the following code:

```
41
42
43 useEffect(() => {
44
45   getAllUsers()
46     .then(users => {
47       console.log(users)
48       setUsers(users);
49       setNumberOfUsers(users.length)
50     });
51
52   getAllMovies()
53     .then(movies => {
54       console.log(movies)
55       setMovies(movies);
56       //setNumberOfUsers(users.length)
57     });
58 }, []);
59
60 const onChangeForm = (e) => {
61   if (e.target.name === 'firstname') {
62     user.firstName = e.target.value;
63   } else if (e.target.name === 'lastname') {
64     user.lastName = e.target.value;
65   } else if (e.target.name === 'email') {
66     user.email = e.target.value;
67   }
68   setUser(user)
69 }
```

STEP4 (เตรียม Database)



The screenshot shows a database management interface. On the left, a 'SCHEMAS' panel lists 'moviedb' with sub-items: Tables, Views, Stored Procedures, Functions, phpmyadmin, and test. The main area displays a SQL query in 'Query 1':

```

1 • CREATE DATABASE moviedb;
2
3 • USE moviedb;
4
5 • CREATE TABLE movies(title VARCHAR(50) NOT NULL,genre VARCHAR(30) NOT NULL,director VARCHAR(60) NOT NULL,release_year
6
7 • INSERT INTO movies VALUE ("Joker", "psychological thriller", "Todd Phillips", 2019);
8
9 • SELECT * FROM movies;
10

```

Below the query, the 'Result Grid' shows the following data:

title	genre	director	release_year
Joker	psychological thriller	Todd Phillips	2019

At the bottom, the 'Output' panel shows the execution log:

#	Time	Action	Message
1	17:02:10	SELECT * FROM movies LIMIT 0, 1000	Error Code: 1046. No database selected Select the default DB to use
2	17:02:12	USE moviedb	0 row(s) affected
3	17:02:15	SELECT * FROM movies LIMIT 0, 1000	1 row(s) returned

STEP5 (ฝั่ง Back-end ปรับปรุง API) ติดตั้ง hapi API package

```

PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example> cd .\node_modules\hapi\api
>> C:\Users\areas\ENGSE203\lab4\react-nodejs-example\api>

```

added 29 packages, and audited 740 packages in 43s

36 packages are looking for funding
run `npm fund` for details

13 vulnerabilities (3 moderate, 10 high)

To address issues that do not require attention, run:
npm audit fix

To address all issues (including breaking changes), run:
npm audit fix --force

Run `npm audit` for details.

```

PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example\api> npm install @hapi/inert

```

added 2 packages, and audited 742 packages in 6s

36 packages are looking for funding
run `npm fund` for details

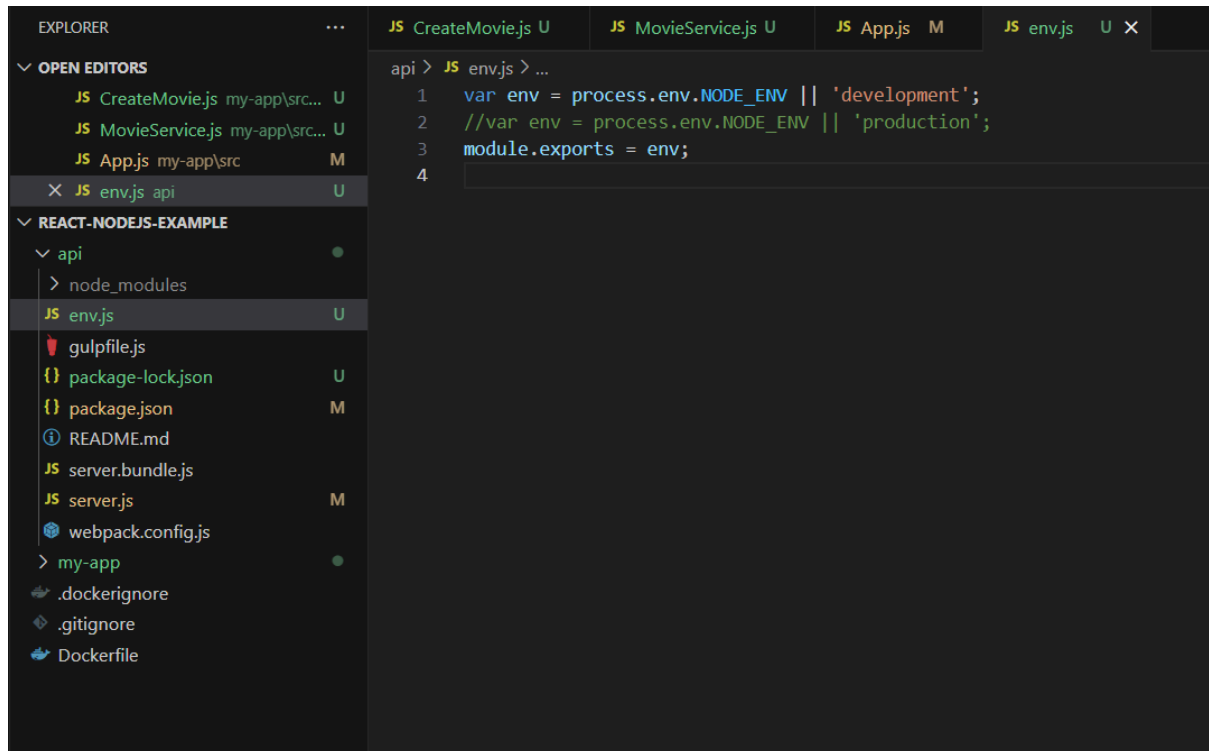
13 vulnerabilities (3 moderate, 10 high)

To address issues that do not require attention, run:
npm audit fix

To address all issues (including breaking changes), run:
npm audit fix --force

Run `npm audit` for details.

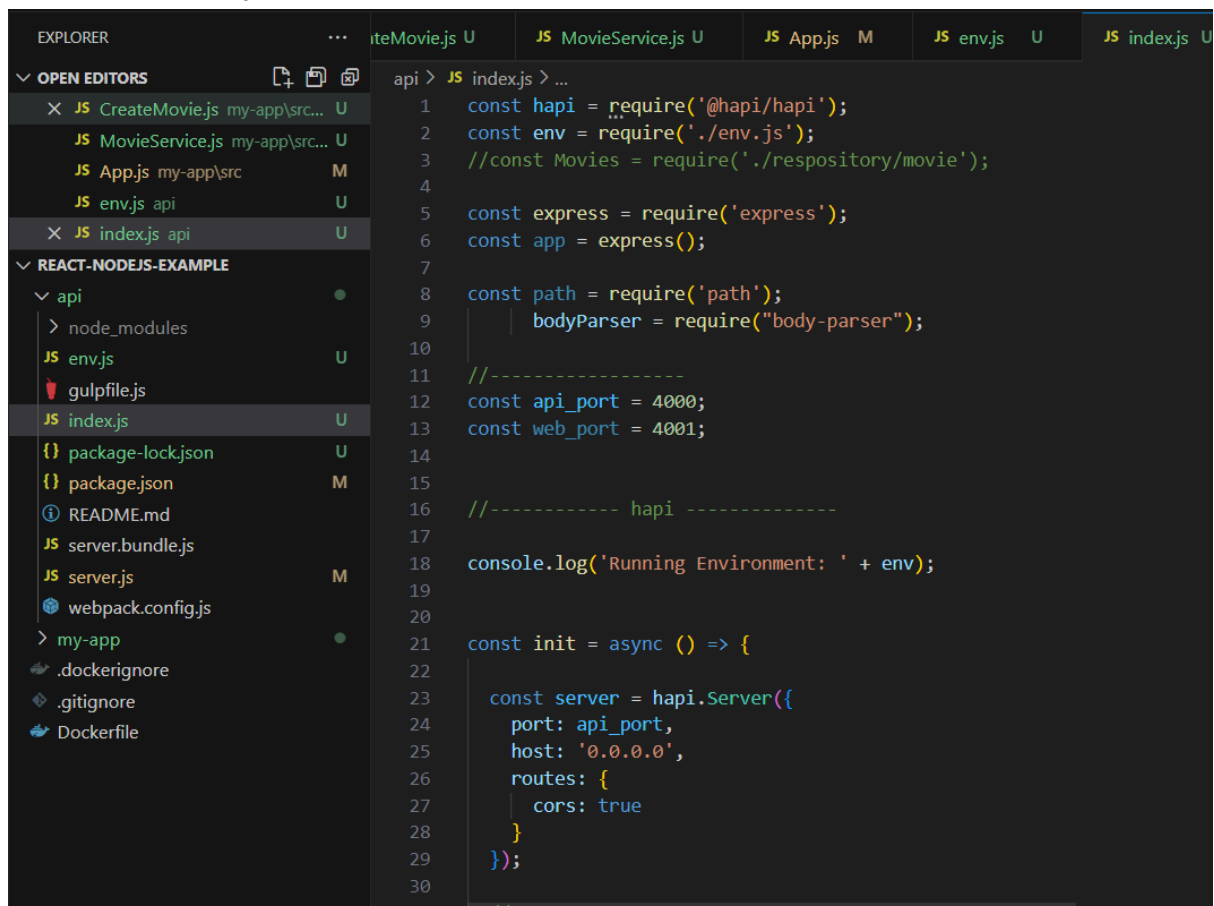
สร้างไฟล์ env.js เพื่อเก็บ config ว่าทำงานอยู่ในสภาพแวดล้อมใด



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project structure with folders 'api' and 'my-app', and various files. The Editor shows the 'env.js' file being created in the 'api' folder. The code in 'env.js' is as follows:

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

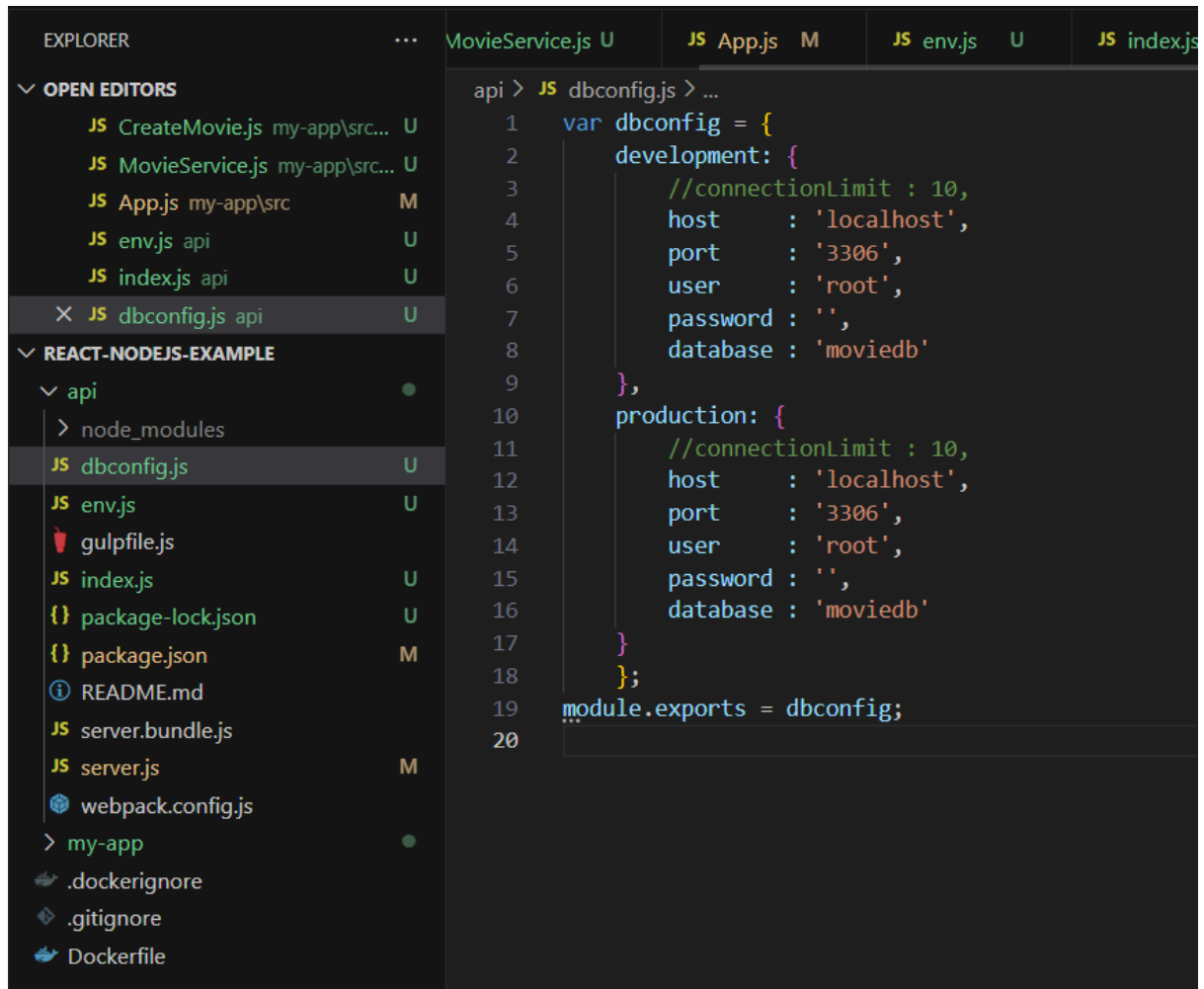
สร้าง index.js



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the project structure with 'index.js' added to the 'api' folder. The Editor shows the 'index.js' file being created in the 'api' folder. The code in 'index.js' is as follows:

```
1 const hapi = require('@hapi/hapi');
2 const env = require('./env.js');
3 //const Movies = require('./respository/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 //----- hapi -----
16
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   //
```


สร้างไฟล์ dbconfig.js เพื่อเก็บ config ใช้สำหรับติดต่อกับ db server (MySQL)



```
api > 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
```

ติดตั้ง mysql package for nodejs

```
Run `npm audit` for details.
PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example\api> npm install mysql

added 5 packages, and audited 747 packages in 2s

36 packages are looking for funding
  run `npm fund` for details

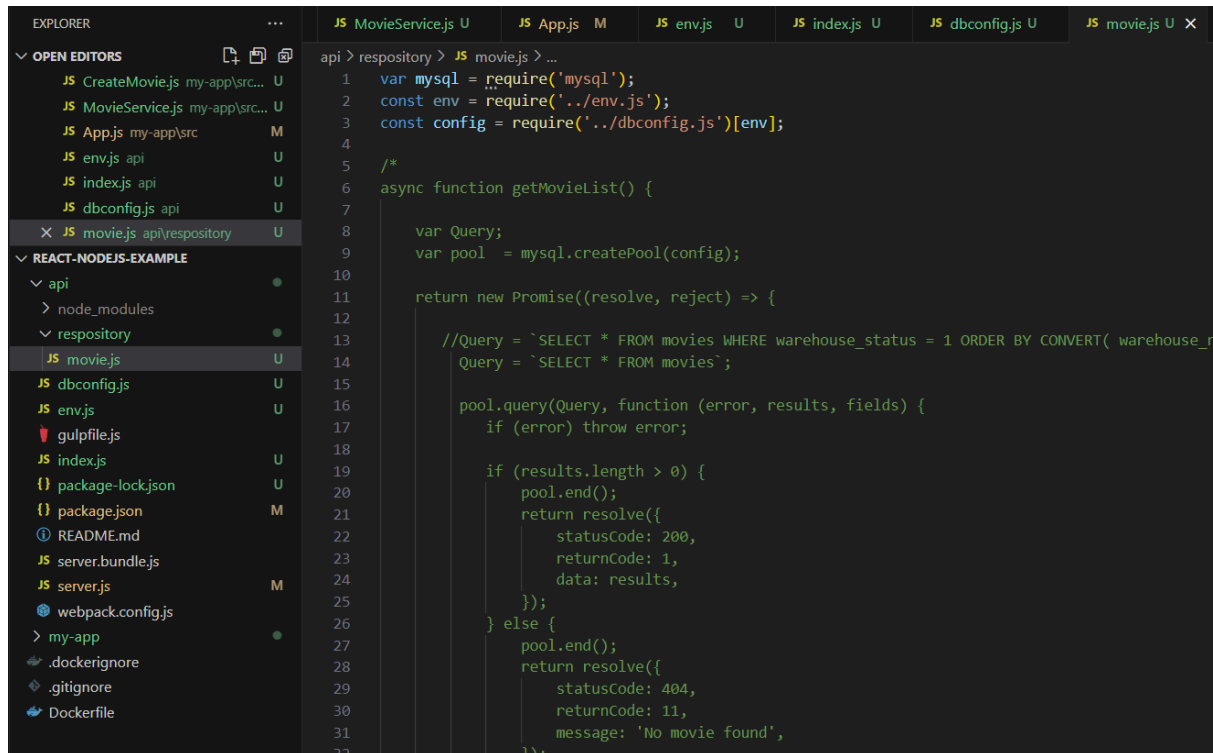
13 vulnerabilities (3 moderate, 10 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\areas\ENGSE203\lab4\react-nodejs-example\api> █
```

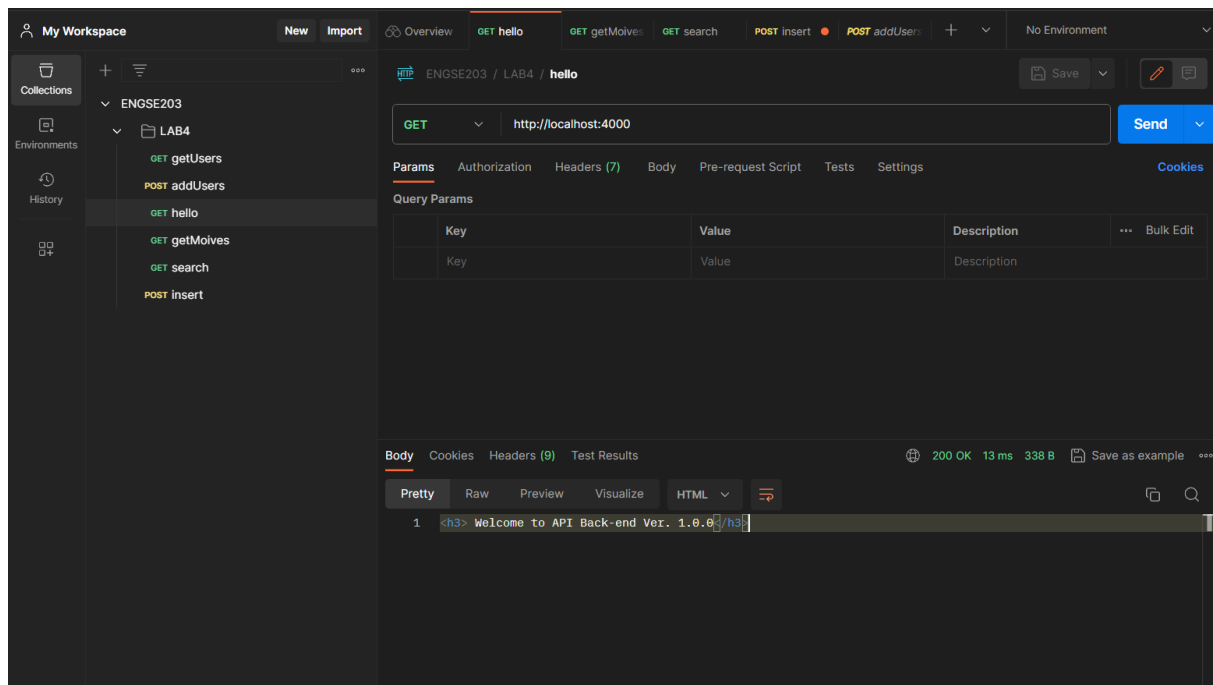
สร้างไฟล์ movie.js ใน folder ชื่อ repository



The screenshot shows the VS Code interface with the Explorer sidebar on the left. The 'repository' folder is selected under the 'api' directory. The 'movie.js' file is being created. The main editor shows the following code:

```
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_r
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  }
```

ให้ทดสอบ API endpoint ด้วยโปรแกรม Postman



My Workspace New Import Overview GET hello GET getMoives GET search POST insert POST addUsers + No Environment

ENGSE203 / LAB4 / insert

POST http://localhost:4000/api/movie/insert Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON

```
1 {
2   "title": "Joker3",
3   "genre": "psychological thriller",
4   "director": "Todd Phillips",
5   "release_year": 2026
6 }
```

Body Cookies Headers (8) Test Results 200 OK 21 ms 349 B Save as example

Pretty Raw Preview Visualize JSON ↗

```
1 {
2   "statusCode": 200,
3   "returnCode": 1,
4   "message": "Movie list was inserted"
5 }
```

My Workspace New Import Overview GET hello GET getMoives GET search POST insert POST addUsers + No Environment

ENGSE203 / LAB4 / getMoives

GET http://localhost:4000/api/movie/all Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results 200 OK 10 ms 599 B Save as example

Pretty Raw Preview Visualize JSON ↗

```
1 [
2   {
3     "title": "Joker",
4     "genre": "psychological thriller",
5     "director": "Todd Phillips",
6     "release_year": 2019
7   },
8   {
9     "title": "Joker2",
10    "genre": "psychological thriller",
11    "director": "Todd Phillips",
12    "release_year": 2022
13  },
14  {
15    "title": "Joker3",
16    "genre": "psychological thriller",
17    "director": "Todd Phillips",
18    "release_year": 2026
19  }
20 ]
```

The screenshot displays the VS Code interface with the REST Client extension. The left sidebar shows the 'My Workspace' with a file explorer. The main editor shows a REST client file named 'ENGSE203 / LAB4 / search'. The 'GET' method is selected, and the URL is 'http://localhost:4000/api/movie/search?search_text=jo'. The 'Send' button is visible. Below the URL bar, the 'Query Params' section is expanded, showing a table with 'search_text' set to 'jo'. The 'Body' tab is selected, showing a JSON response with movie details for 'Joker2' and 'Joker3'.

```
{
  "genre": "psychological thriller",
  "director": "Todd Phillips",
  "release_year": 2019
},
{
  "title": "Joker2",
  "genre": "psychological thriller",
  "director": "Todd Phillips",
  "release_year": 2022
},
{
  "title": "Joker3",
  "genre": "psychological thriller",
  "director": "Todd Phillips",
  "release_year": 2026
}
```