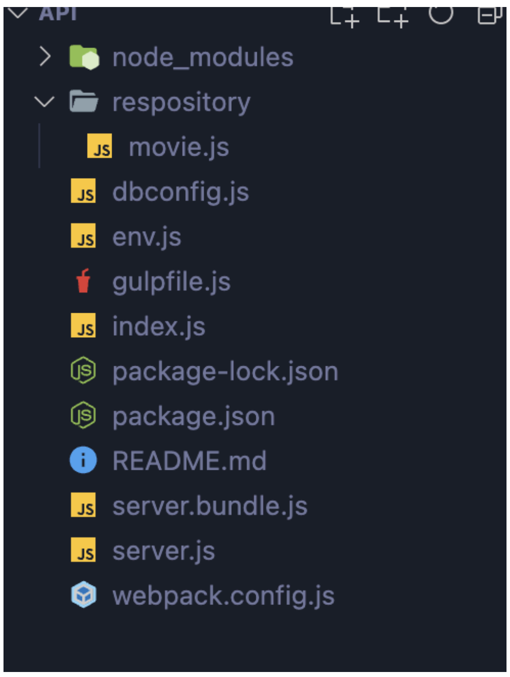
VSCODE API



```
API
  node_modules
  respository
    movie.js
  dbconfig.js
  env.js
  gulpfile.js
  index.js
  package-lock.json
  package.json
  README.md
  server.bundle.js
  server.js
  webpack.config.js
```

ไฟล์ทั้งหมดที่ใช้

รูปไฟล์ที่จำเป็น

-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;
//------------ hapi --------------
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>';
  }
});
  //API: http://localhost:3001/api/movie/all
  server.route({
    method: 'GET',
    path: '/api/movie/all',
    config: {
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        //var param = request.query;
        //const category_code = param.category_code;

        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: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        var param = request.query;
        const search_text = param.search_text;
        //const title = param.title;
        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;

       //const title = request.payload.title;
       //const genre = request.payload.genre;

       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();
```

-movie.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 WHERE warehouse_status = 1 ORDER BY CONVERT(
warehouse_name USING tis620 ) ASC `;
        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) => {

       //Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;

       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) {
       //pool.query(Query, function (error, results, fields) {


        // if (error) throw error;
       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,

};
```

-server.js

```
const express = require('express');
const path = require('path');
const app = express(),
     bodyParser = require("body-parser");
     port = 4000;

// place holder for the data
const users = [
 {
   firstName: "first1",
   lastName: "last1",
   email: "abc@gmail.com"
 },
 {
   firstName: "first2",
   lastName: "last2",
   email: "abc@gmail.com"
 },
 {
   firstName: "first3",
   lastName: "last3",
   email: "abc@gmail.com"
 }
];

app.use(bodyParser.json());
//app.use(express.static(path.join(__dirname, '../my-app/build')));

app.get('/api/users', (req, res) => {    //เรียกใช้ฟังชั่น
 console.log('api/users called!') //คำสั่ง ปริ้นค่าตัวแปร หรือ ดีบัค
 res.json(users); //  ให้รีเทนแบบ เจสัน
});

app.post('/api/user', (req, res) => {
 const user = req.body.user;
 console.log('Adding user:::::', user);
 users.push(user);
 res.json("user addedd");
});

app.get('/', (req,res) => {
 //res.sendFile(path.join(__dirname, '../my-app/build/index.html'));
});

app.listen(port, () => {
   console.log(`Server listening on the port::${port}`);
});
```

-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;
```

VSCODE MY-APP

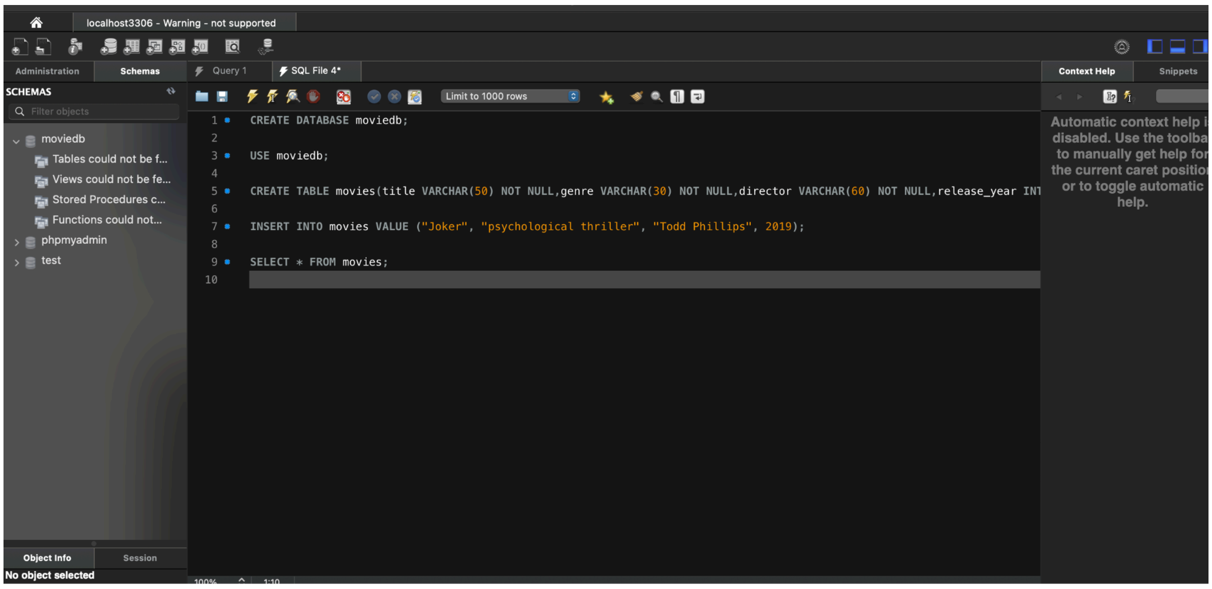
-App.js

```
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 { getAllMovies, createMovie } from './services/MovieService'

function App() {

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

 //--------------

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


 const userCreate = (e) => {

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

 const fetchAllUsers = () => {
   getAllUsers()
     .then(users => {
       console.log(users)
       setUsers(users);
```

```
      setNumberOfUsers(users.length)
    });
}

useEffect(() => {

  getAllUsers()
    .then(users => {
       console.log(users)
      setUsers(users);
       setNumberOfUsers(users.length)
    });

    getAllMovies()
    .then(movies => {
      console.log(movies)
      setMovies(movies);
      //setNumberOfUsers(users.length)
    });

}, [])

const onChangeForm = (e) => {
    if (e.target.name === 'firstname') {
        user.firstName = e.target.value;
    } else if (e.target.name === 'lastname') {
        user.lastName = e.target.value;
    } else if (e.target.name === 'email') {
        user.email = e.target.value;
    }
    setUser(user)
}
  return (
      <div className="App">
        <Header></Header>
         <div className="container mrgnbtm">
          <div className="row">
            <div className="col-md-8">
                <CreateUser
                  user={user}
                  onChangeForm={onChangeForm}
                  createUser={userCreate}
                  >
                </CreateUser>
            </div>
```

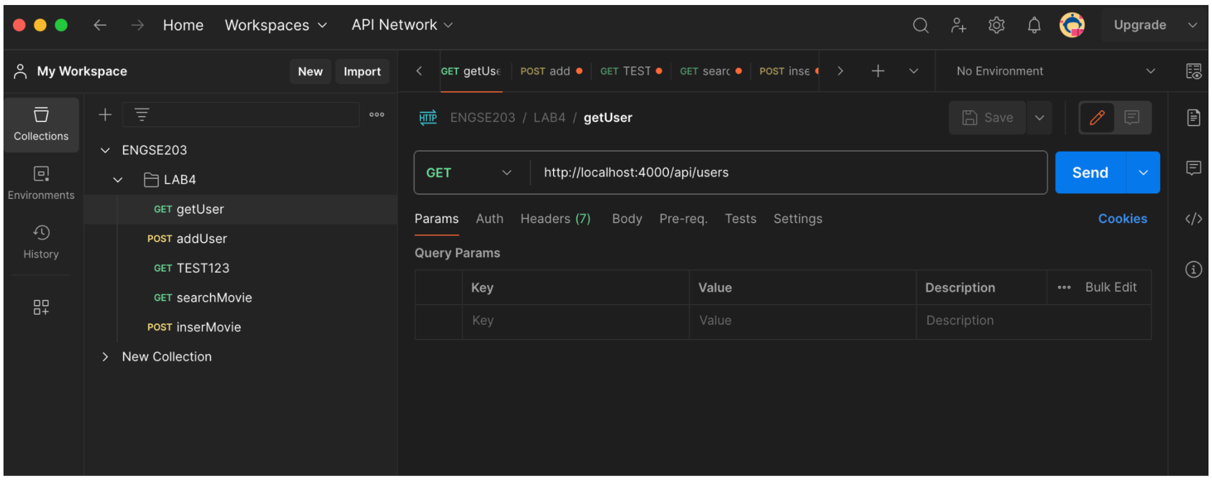
```
             <div className="col-md-4">
                 <DisplayBoard
                   numberOfUsers={numberOfUsers}
                   getAllUsers={fetchAllUsers}
                 >
                 </DisplayBoard>
             </div>
           </div>
         </div>
         <div className="row mrgnbtm">
           <Users users={users}></Users>
         </div>
         <div className="row mrgnbtm">
           <Movies movies={movies}></Movies>
         </div>
       </div>
   );
}
export default App;
```

MySQL

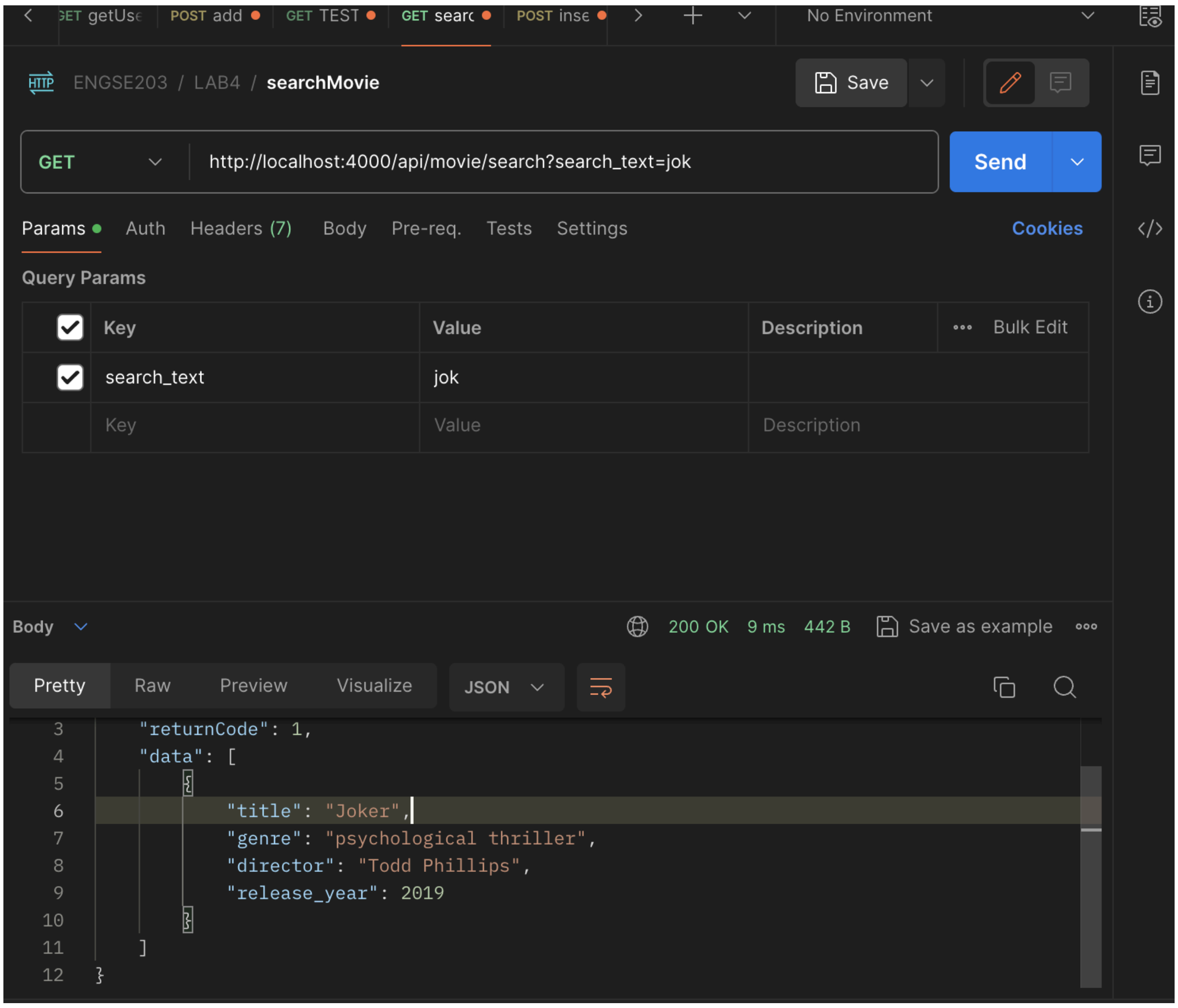


```
CREATE DATABASE moviedb;

USE moviedb;

CREATE TABLE movies(title VARCHAR(50) NOT NULL,genre VARCHAR(30) NOT NULL,director VARCHAR(60) NOT NULL,release_year INT

INSERT INTO movies VALUE ("Joker", "psychological thriller", "Todd Phillips", 2019);

SELECT * FROM movies;
```

POSTMAN



```
GET http://localhost:4000/api/users
```

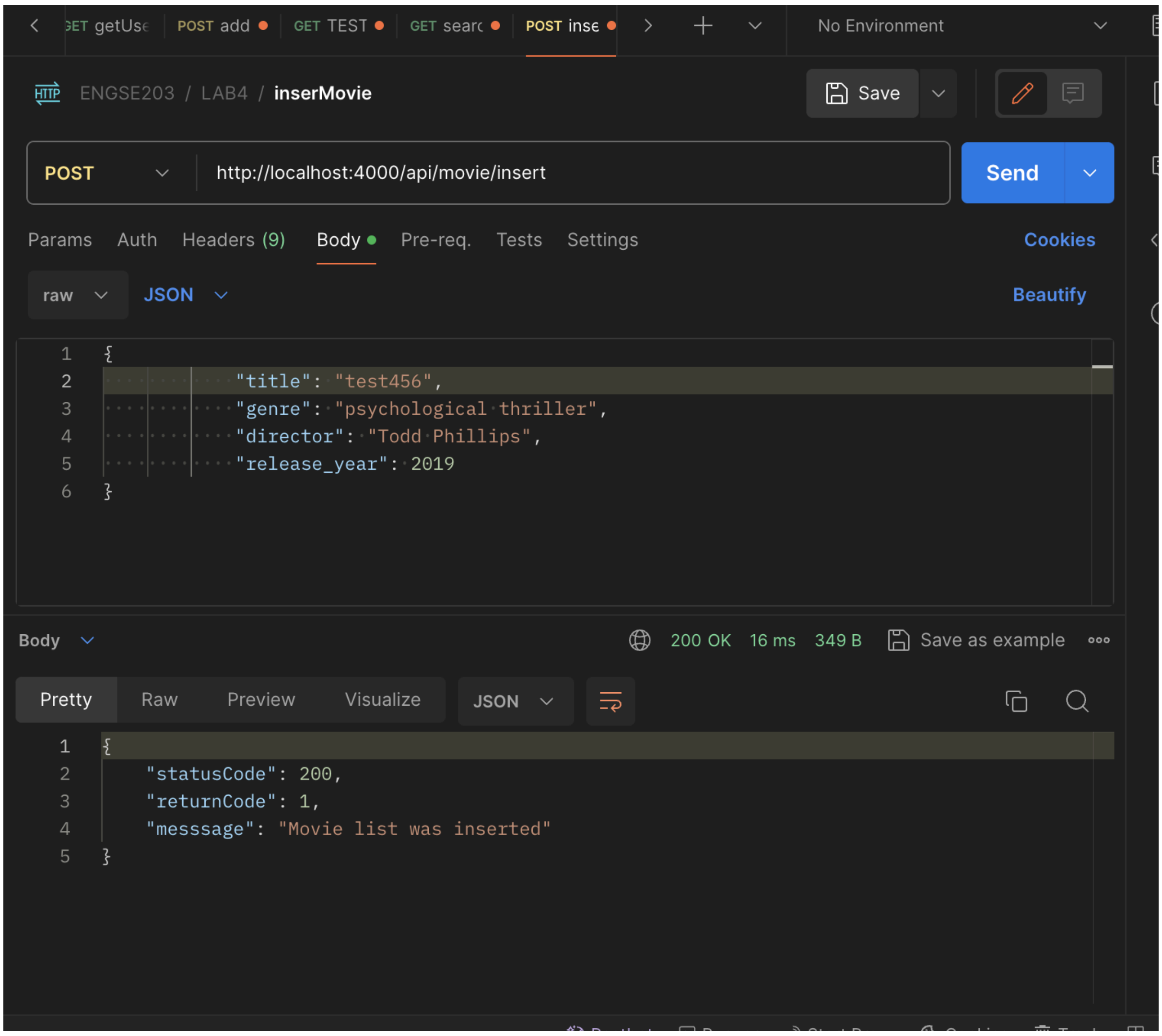


```
GET http://localhost:4000/api/movie/search?search_text=jok
```

Key	Value	Description
search_text	jok	

200 OK 9 ms 442 B

```
    "returnCode": 1,
    "data": [
        {
            "title": "Joker",
            "genre": "psychological thriller",
            "director": "Todd Phillips",
            "release_year": 2019
        }
    ]
}
```

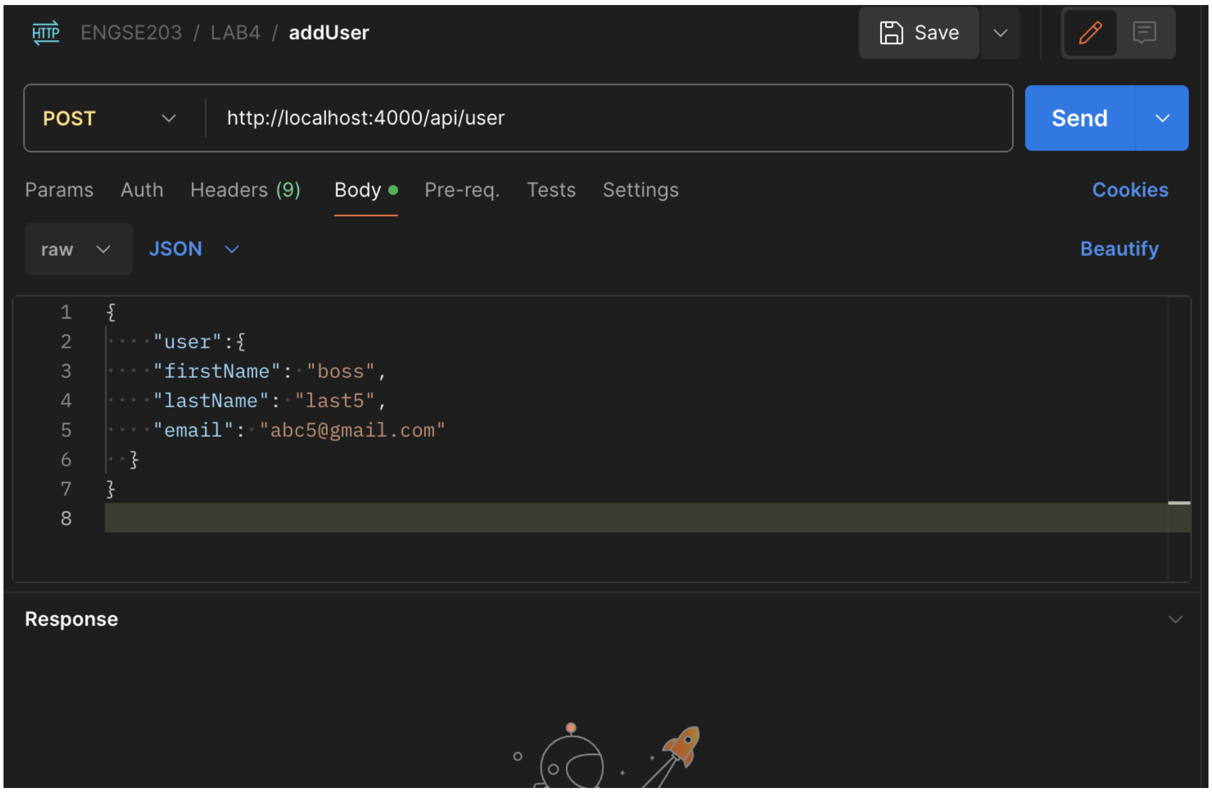


POST http://localhost:4000/api/movie/insert

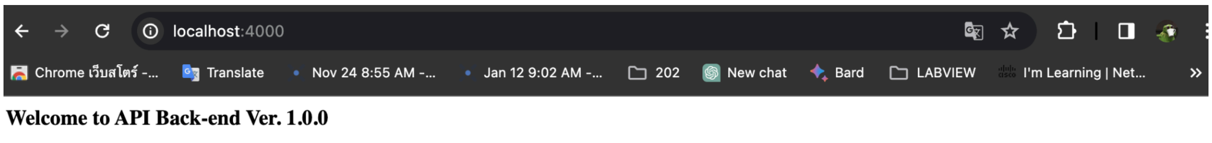
```
{
        "title": "test456",
        "genre": "psychological thriller",
        "director": "Todd Phillips",
        "release_year": 2019
}
```

200 OK 16 ms 349 B

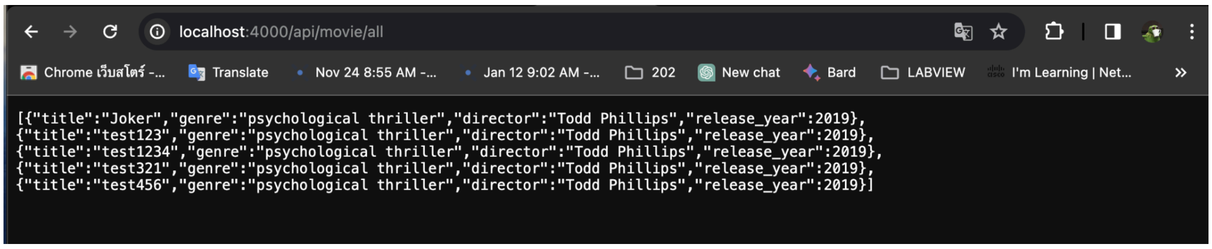
```
{
    "statusCode": 200,
    "returnCode": 1,
    "messsage": "Movie list was inserted"
}
```


ENGSE203 / LAB4 / addUser
Save
POST
http://localhost:4000/api/user
Send
Params Auth Headers (9) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
Response

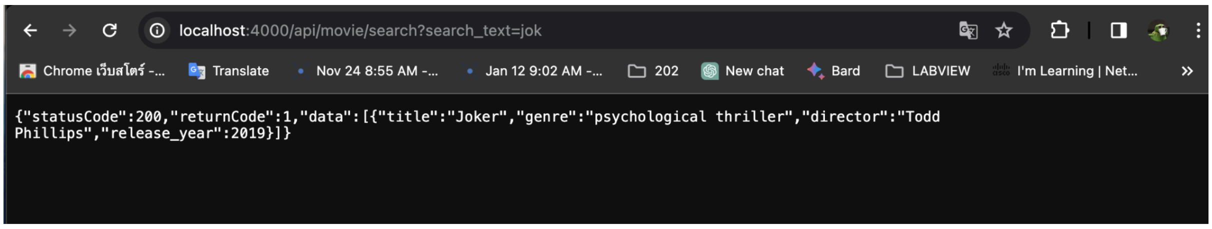
```
{
    "user":{
    "firstName": "boss",
    "lastName": "last5",
    "email": "abc5@gmail.com"
  }
}
```



Welcome to API Back-end Ver. 1.0.0



```
[{"title":"Joker","genre":"psychological thriller","director":"Todd Phillips","release_year":2019},
{"title":"test123","genre":"psychological thriller","director":"Todd Phillips","release_year":2019},
{"title":"test1234","genre":"psychological thriller","director":"Todd Phillips","release_year":2019},
{"title":"test321","genre":"psychological thriller","director":"Todd Phillips","release_year":2019},
{"title":"test456","genre":"psychological thriller","director":"Todd Phillips","release_year":2019}]
```



```
{"statusCode":200,"returnCode":1,"data":[{"title":"Joker","genre":"psychological thriller","director":"Todd
Phillips","release_year":2019}]}
```