LAB 4

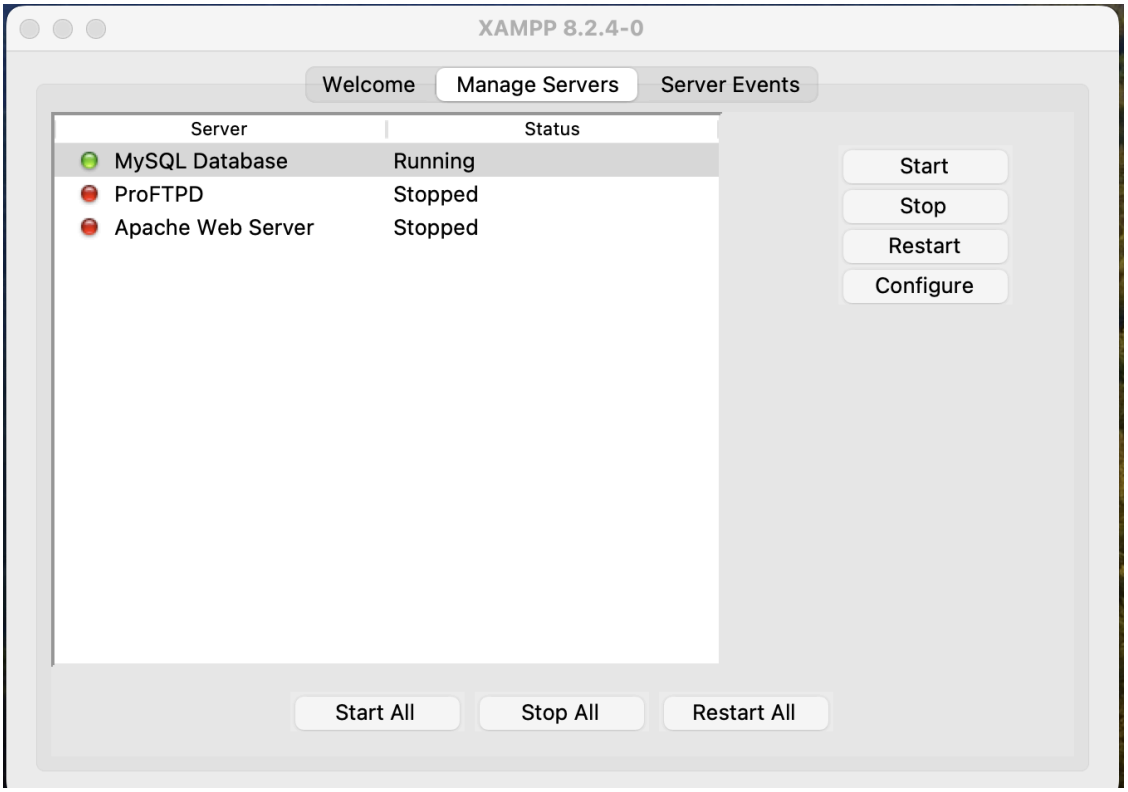
เปลี่ยน Port ของ backend(API) เป็น 4000 หน้า package.json และ server.js

```
{} package.json > { } scripts
1   {
2     "name": "react-nodejs-example",
3     "version": "1.0.0",
4     "description": "example project react with nodejs",
5     "main": "server.js",
        ▷ Debug
6     "scripts": {
7       "start": "node server.bundle.js",
8       "build": "webpack",
9       "dev": "nodemon ./index.js localhost 4000"
10    },
```

```
JS server.js > ...
1   const express = require('express');
2   const path = require('path');
3   const app = express(),
4         bodyParser = require("body-parser");
5         port = 4000;
6
```

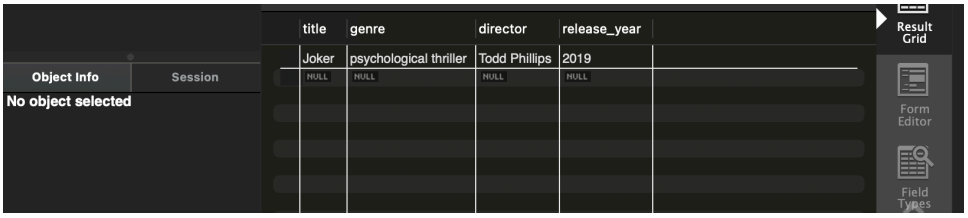
แล้วรัน http://localhost:4000/api/users

เปิด MySQL Database ใน Xammp



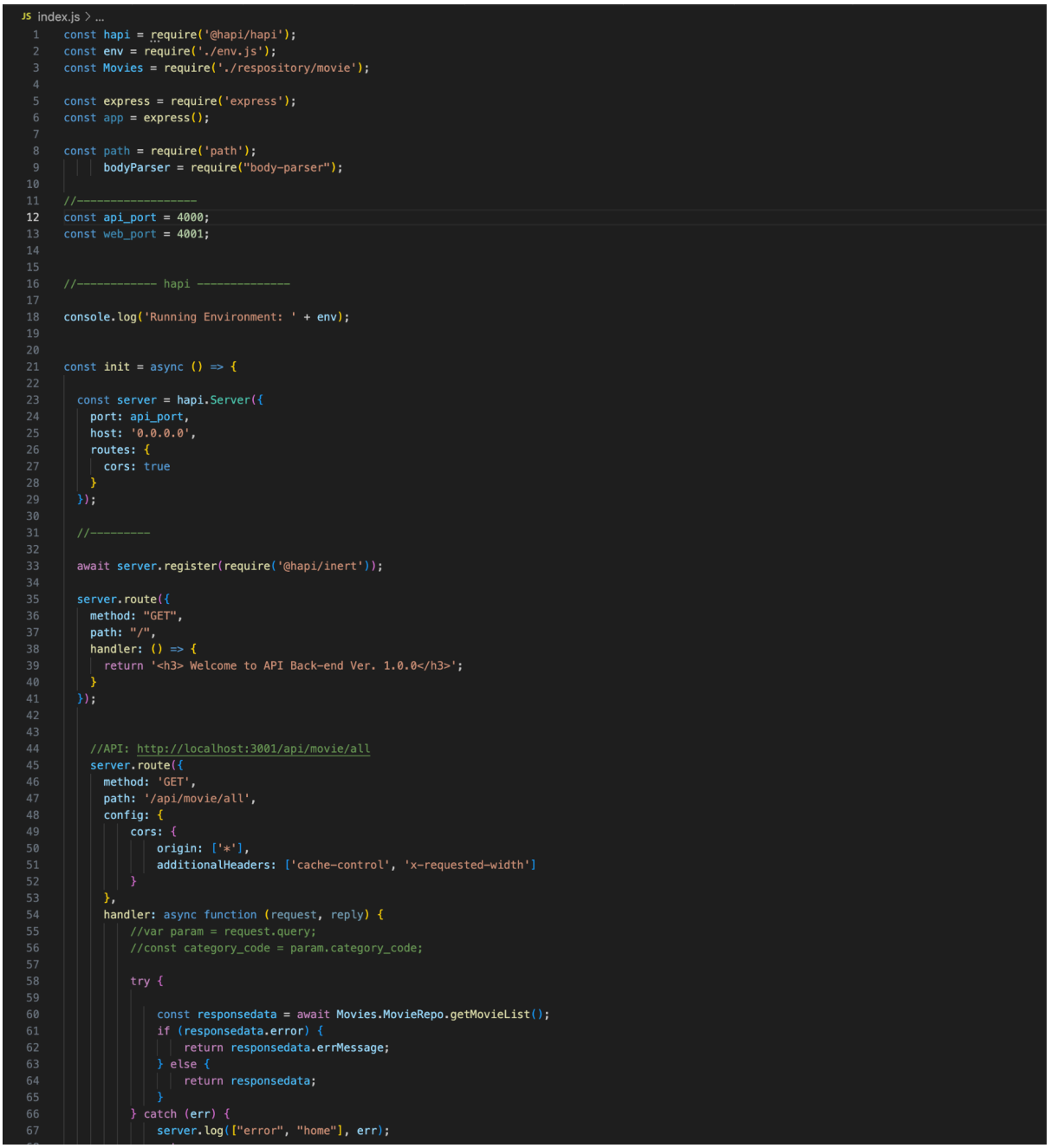
สร้าง database

แสดงข้อมูล



สร้างหน้า index.js มาใน backend และใส่โค้ด

index.js



```
index.js > ...
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
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        method: 'GET',
47        path: '/api/movie/all',
48        config: {
49            cors: {
50                origin: ['*'],
51                additionalHeaders: ['cache-control', 'x-requested-width']
52            }
53        },
54        handler: async function (request, reply) {
55            //var param = request.query;
56            //const category_code = param.category_code;
57
58            try {
59
60                const responsedata = await Movies.MovieRepo.getMovieList();
61                if (responsedata.error) {
62                    return responsedata.errMessage;
63                } else {
64                    return responsedata;
65                }
66            } catch (err) {
67                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();
```

สร้างหน้า env.js และ dbconfig.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
```

JS 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;
```

สร้างโฟลเดอร์ respository และสร้างไฟล์ movie.js

โค้ดหน้า movie.js

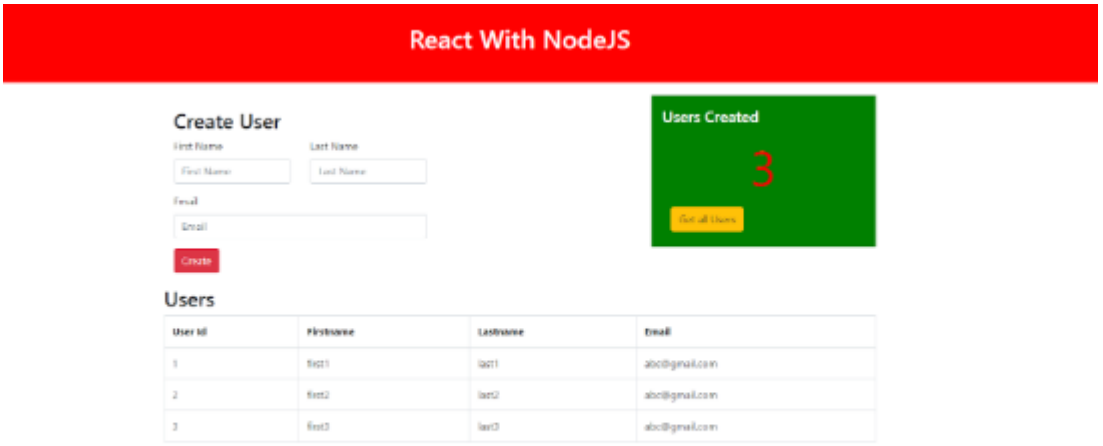
respository > JS movie.js > ...

```
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({
                   statusCode: 200,
                   returnCode: 1,
                   data: results,
               });
           } else {
               pool.end();
               return resolve({
                   statusCode: 404,
                   returnCode: 11,
                   message: 'No movie found',
               });
           }

        });

    });


}
*/

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',
               });
           }

        });
```

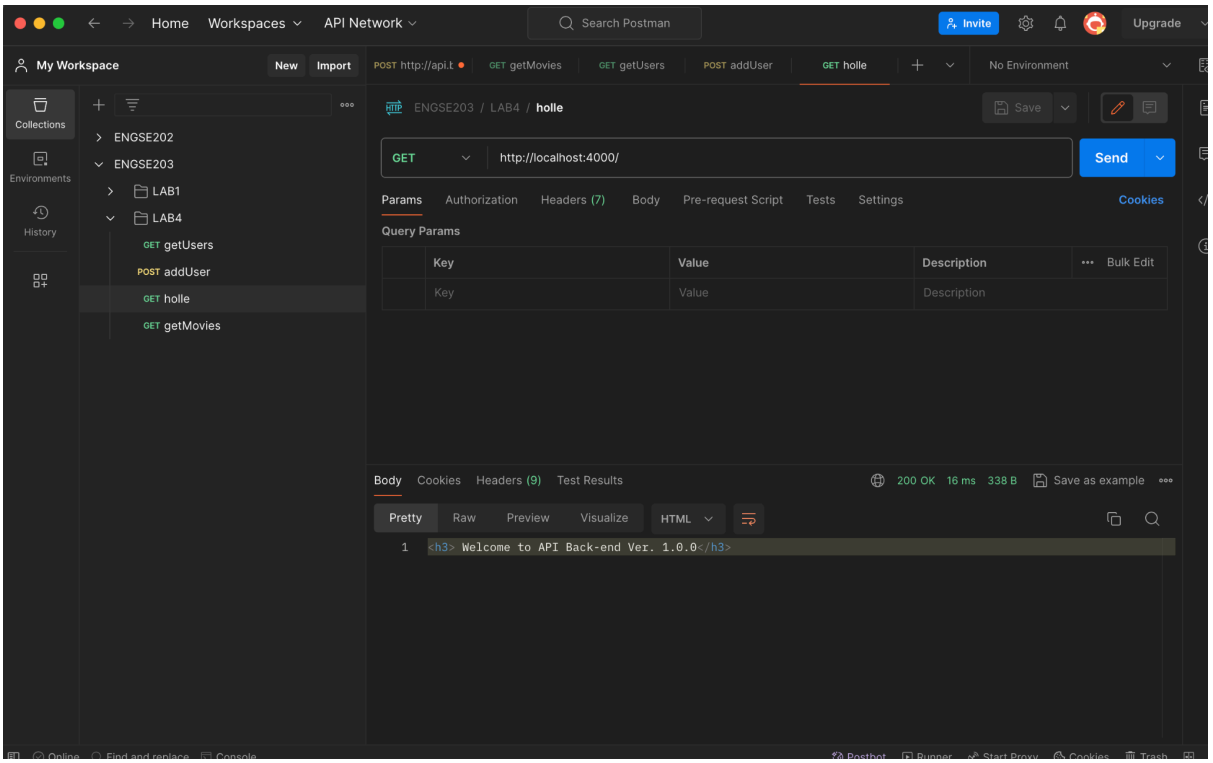
```
69
70      });
71
72
73  }
74
75
76  async function getMovieSearch(search_text) {
77
78      var Query;
79      var pool  = mysql.createPool(config);
80
81      return new Promise((resolve, reject) => {
82
83          Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;
84
85     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

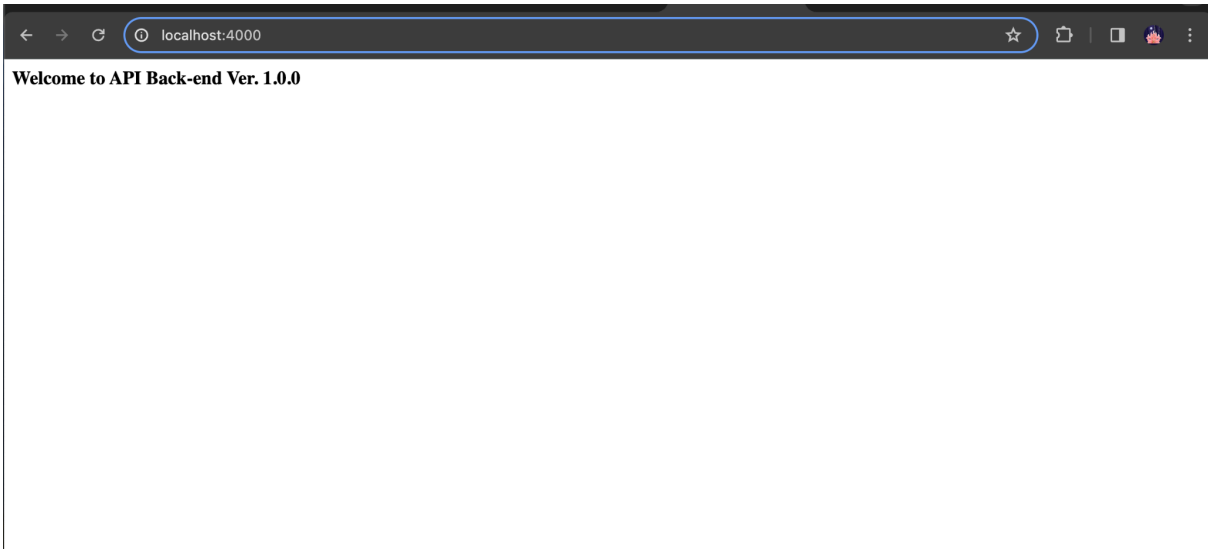
```
                console.log('error_code_msg: ', error.code+':'+error.sqlMessage);
                pool.end();
                return resolve({
                    statusCode: 405,
                    returnCode: 9,
                    messsage: error.code+': '+error.sqlMessage
                });
            }
            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,

};

```

ผลรัน

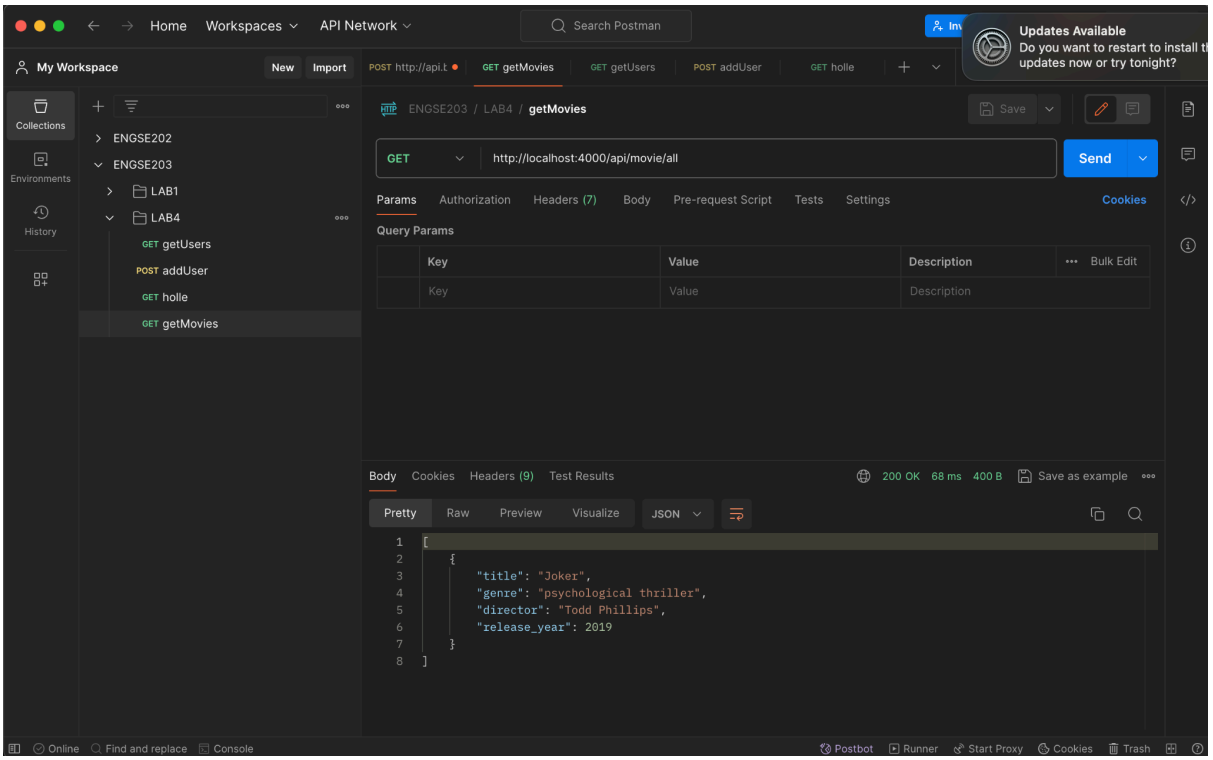


Postman
http://localhost:4000/

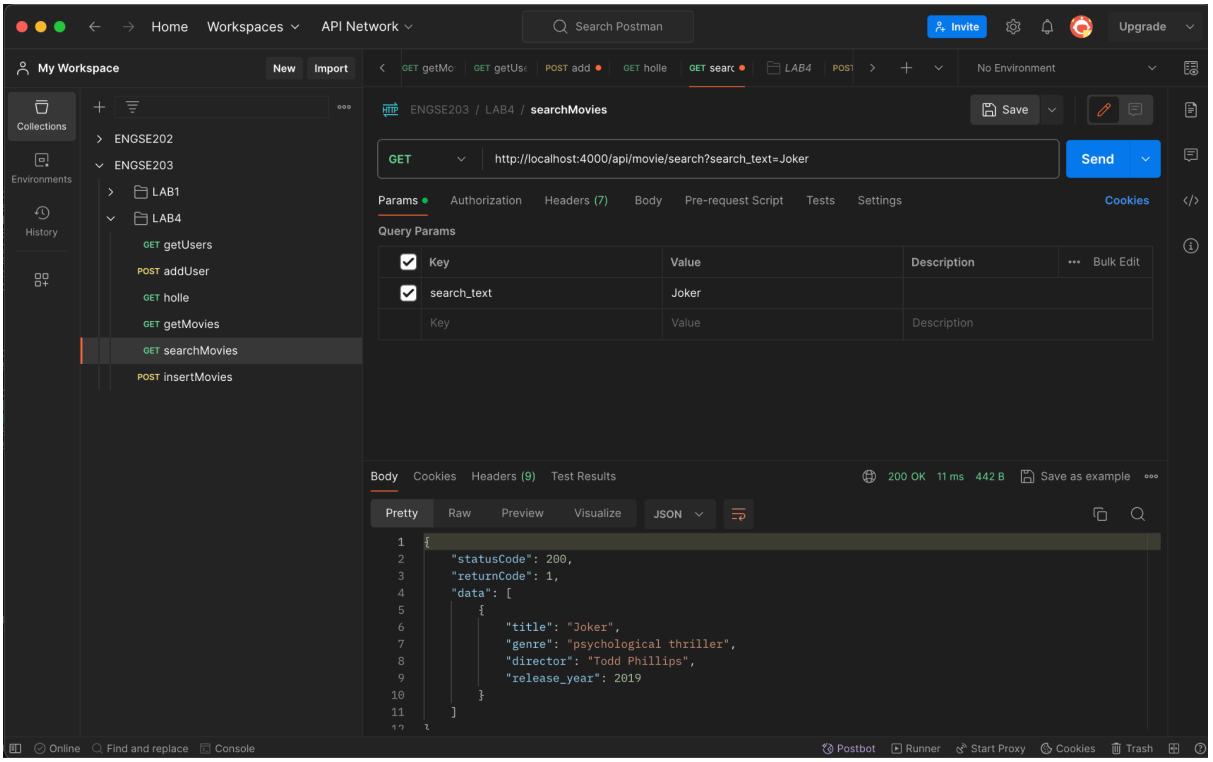




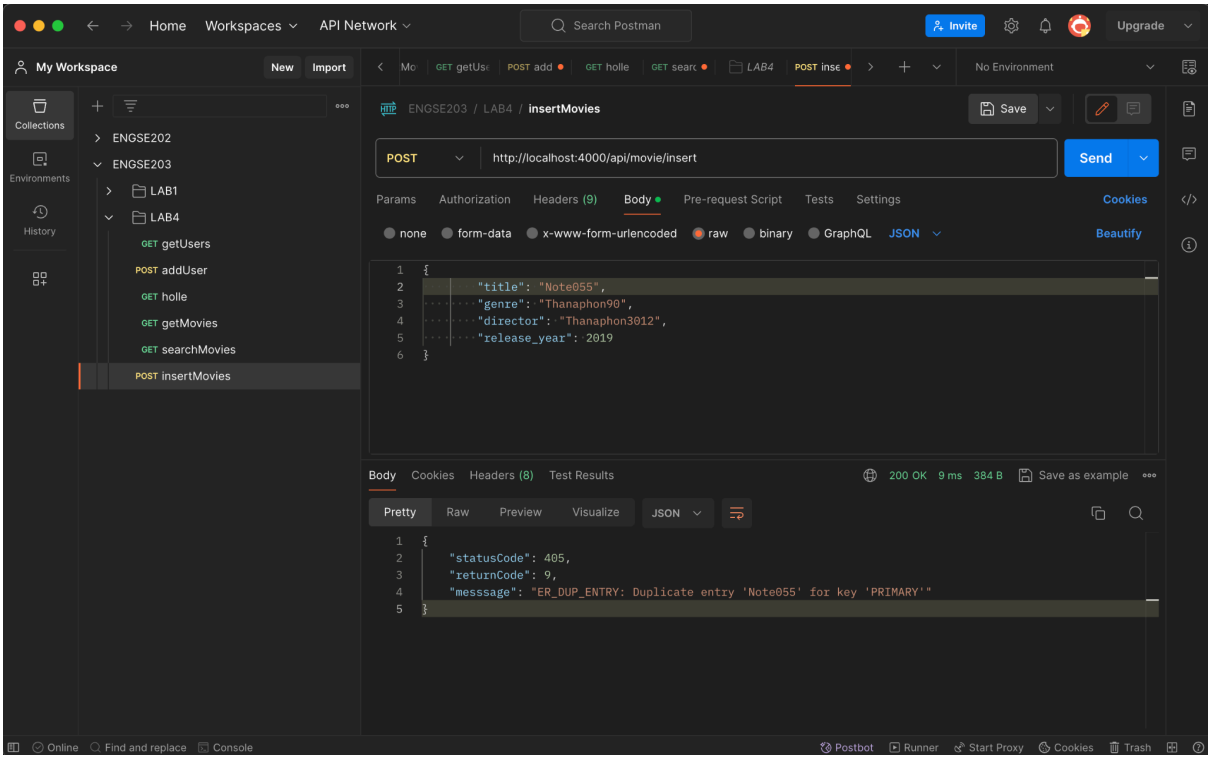
http://localhost:4000/api/movie/all



http://localhost:4000/api/movie/search?search_text=Joker



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



```
post is:  {
  title: 'Note055',
  genre: 'Thanaphon90',
  director: 'Thanaphon3012',
  release_year: 2019
}
results:  OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

กรณี insert ข้อมูลซ้ำ

```
post is:  {
  title: 'Note055',
  genre: 'Thanaphon90',
  director: 'Thanaphon3012',
  release_year: 2019
}
error_code_msg:  ER_DUP_ENTRY:Duplicate entry 'Note055' for key 'PRIMARY'
```