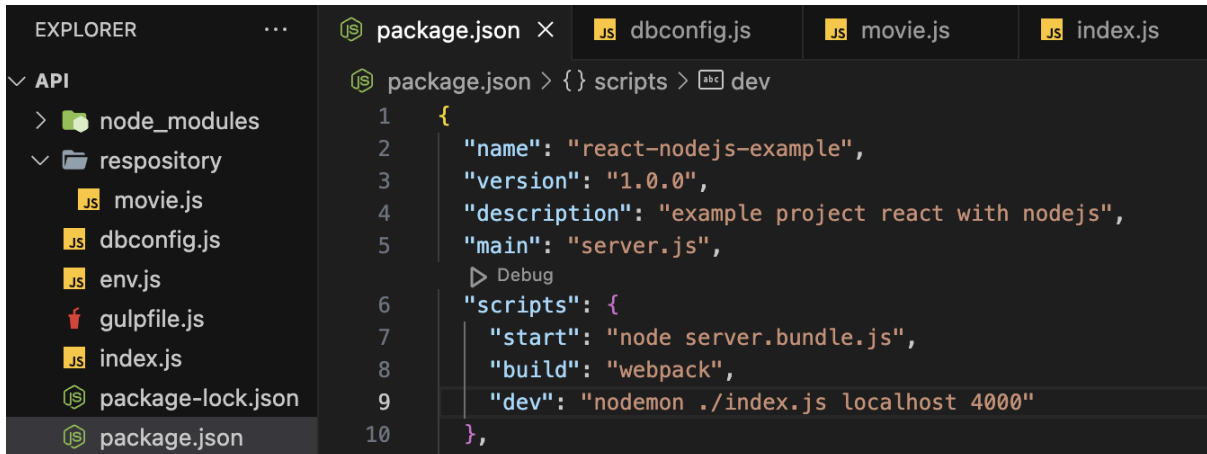


# LAB4

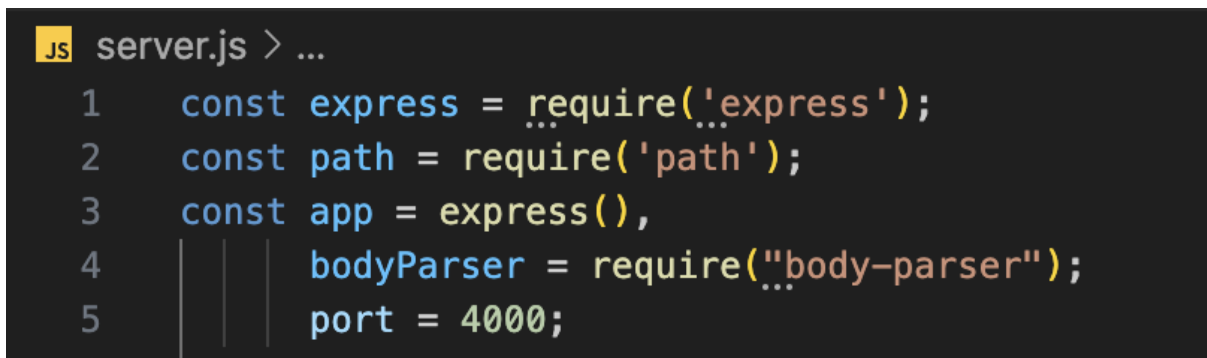


The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree with the following structure:

- API
  - node\_modules
  - respository
    - movie.js
    - dbconfig.js
    - env.js
    - gulpfile.js
    - index.js
    - package-lock.json
    - package.json

The main editor area shows the `package.json` file with the following content:

```
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 ./index.js localhost 4000"
10  },
```



The screenshot shows the `server.js` file in the VS Code editor with the following content:

```
1 const express = require('express');
2 const path = require('path');
3 const app = express(),
4     bodyParser = require('body-parser');
5 port = 4000;
```

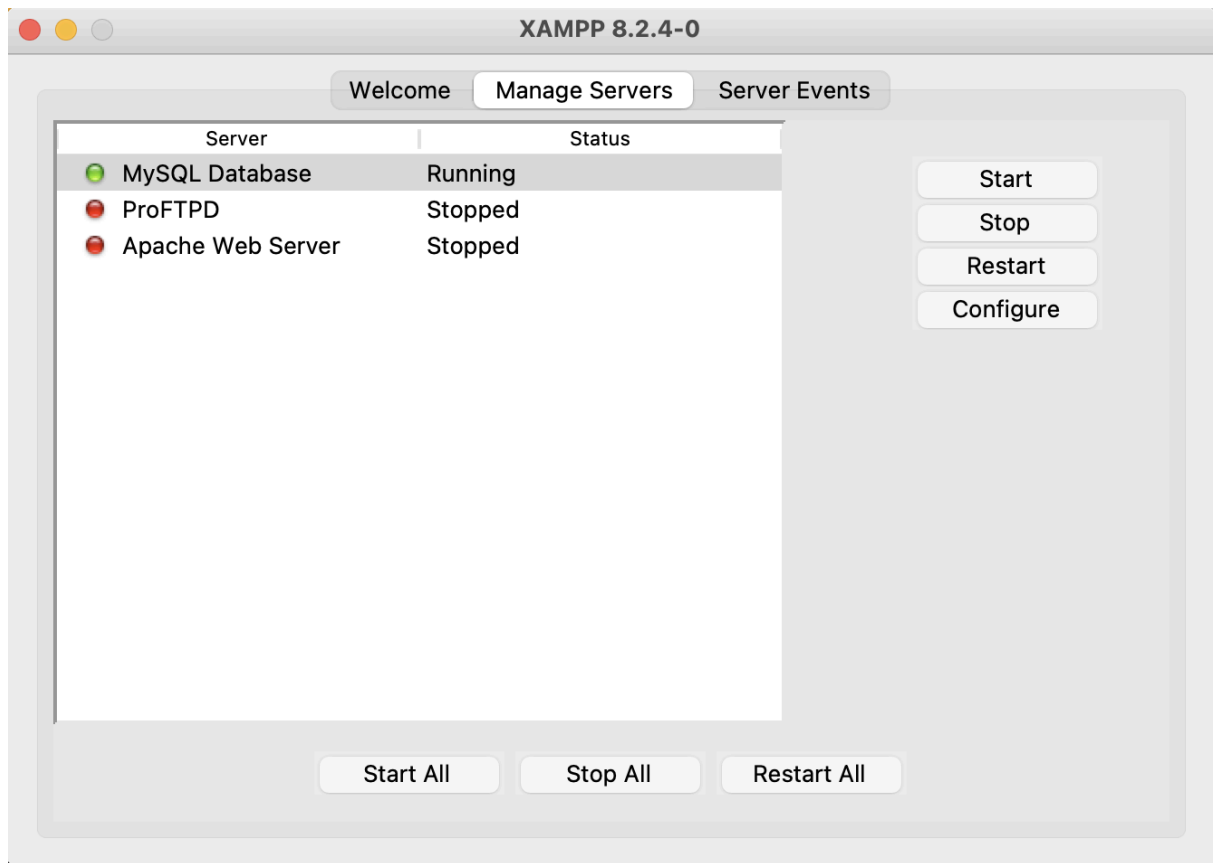
เปลี่ยน Port ของ backend เป็น 4000 หน้า package.json , server.js



The screenshot shows a web browser window with the address bar displaying `localhost:4000/api/users`. The page content shows a JSON array of three user objects:

```
[{"firstName": "first1", "lastName": "last1", "email": "abc@gmail.com"}, {"firstName": "first2", "lastName": "last2", "email": "abc@gmail.com"}, {"firstName": "first3", "lastName": "last3", "email": "abc@gmail.com"}]
```

ทดสอบ



## เปิด MYSQL ใน XAMPP

```
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(4) NOT NULL);
```

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

```
SELECT * FROM movies;
```

title	genre	director	release_year
Forest Gum	Comady	Todd Phillips	2019
Forest Gum2	Comady	Todd Phillips	2019
Joker	psychological thriller	Todd Phillips	2019
NULL	NULL	NULL	NULL

สร้าง database , เพิ่มตารางและเพิ่มข้อมูลลงในตาราง

สร้าง index.js มาใน backend และใส่โค้ด

```
js index.js > [⌘] init
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 respondedata = await Movies.MovieRepo.getMovieList();
61       if (respondedata.error) {
62         return respondedata.errMessage;
63       } else {
64         return respondedata;
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: ['*'],
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

```

```

87
88     try {
89
90         const respondedata = await Movies.MovieRepo.getMovieSearch(search_text);
91         if (respondedata.error) {
92             return respondedata.errMessage;
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.errMessage;
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();
162
```

สร้าง env.js เพื่อเก็บ config ว่าทำงานอยู่ในสภาพแวดล้อมใด

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

สร้าง 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
```

สร้างโฟลเดอร์ repository และสร้างไฟล์ movie.js

```
▼ repository
  JS movie.js
```

```
repository > JS movie.js > ...
1 var mysql = require('mysql');
2 const env = require('../env.js');
3 const config = require('../dbconfig.js')[env];
```

respository >  movie.js > ...

```
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 CO
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
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,
```



respository >  movie.js > ...

```
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 if (error) {
137     console.log('error_code_msg: ',error.code+':'+error.sqlMessage);
138     pool.end();
139     return resolve({
140         statusCode: 405,
141         returnCode: 9,
142         message: error.code+':'+error.sqlMessage,
143     });
144 } else {
145     console.log('results: ',results);
146     if (results.affectedRows > 0) {
147         pool.end();
148         return resolve({
149             statusCode: 200,
150             returnCode: 1,
151             message: 'Movie list was inserted',
152         });
153     }
154 }
155
```

แก้ได้ตามนี้

=====

```
158
159     });
160
161
162     });
163
164
165 }
166
167 module.exports.MovieRepo = {
168     getMovieList: getMovieList,
169     getMovieSearch: getMovieSearch,
170     postMovie: postMovie,
171
172 };
173
```

## ทดสอบการทำงานใน Postman

The screenshot shows the Postman interface with a collection named 'LAB4' expanded. The 'hello' endpoint is selected, which is a GET request to 'http://localhost:4000'. The 'Send' button has been clicked, and the response is displayed in the 'Body' tab. The response is a 200 OK status with a response time of 4 ms and a body size of 338 B. The response body is rendered in HTML format, showing a single line: '<h3> Welcome to API Back-end Ver. 1.0.0</h3>'. The 'Query Params' table is empty.

Key	Value	Description
Key	Value	Description

```
1 <h3> Welcome to API Back-end Ver. 1.0.0</h3>
```

http://localhost:4000

The screenshot shows the Postman interface with the 'getMovies' endpoint selected, which is a GET request to 'http://localhost:4000/api/movie/all'. The 'Send' button has been clicked, and the response is displayed in the 'Body' tab. The response is a 200 OK status with a response time of 61 ms and a body size of 400 B. The response body is rendered in JSON format, showing a single movie object: '{ "title": "Joker", "genre": "psychological thriller", "director": "Todd Phillips", "release\_year": 2019 }'. The 'Query Params' table is empty.

Key	Value	Description
Key	Value	Description

```
1 {
2   {
3     "title": "Joker",
4     "genre": "psychological thriller",
5     "director": "Todd Phillips",
6     "release_year": 2019
7   }
8 }
```

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

HTTP client interface showing a GET request to `http://localhost:4000/api/movie/search?search_text=jok`. The request is sent, and the response is displayed in the Body tab.

**Query Params**

Key	Value	Description
search_text	jok	

**Body**

```
1 {
2   "statusCode": 200,
3   "returnCode": 1,
4   "data": [
5     {
6       "title": "Joker",
7       "genre": "psychological thriller",
8       "director": "Todd Phillips",
9       "release_year": 2019
10    }
11  ]
12 }
```

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

HTTP client interface showing a POST request to `http://localhost:4000/api/movie/insert`. The request is sent, and the response is displayed in the Body tab.

**Body**

```
1 {
2   "title": "Forest Gum2",
3   "genre": "Comady",
4   "director": "Todd Phillips",
5   "release_year": 2019
6 }
```

**Body**

```
1 {
2   "statusCode": 405,
3   "returnCode": 9,
4   "message": "ER_DUP_ENTRY:Duplicate entry 'Forest Gum2' for key 'PRIMARY'"
5 }
```

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