

C:\node\mysql 폴더를 생성

cd mysql

npm init -yes

/* express 와 mysql 모듈을 설치 함 */

```
$ npm install express mysql
```

package.json 을 다음과 같이 수정함



다음은 루트 디렉터리에 index.js를 생성한다.

JAVASCRIPT

```
// index.js  
console.log('Hello world')
```

애플리케이션을 실행하여 콘솔에 'Hello world'가 출력되는 것을 확인한다.

BASH

```
$ npm start
```

root 권한으로 접속

#

3. 테이블 생성 및 테스트용 데이터 삽입

MySQL에 접속하여 테이블을 생성하고 테스트용 데이터를 insert해 놓자.

SQL

```
CREATE DATABASE IF NOT EXISTS my_db;

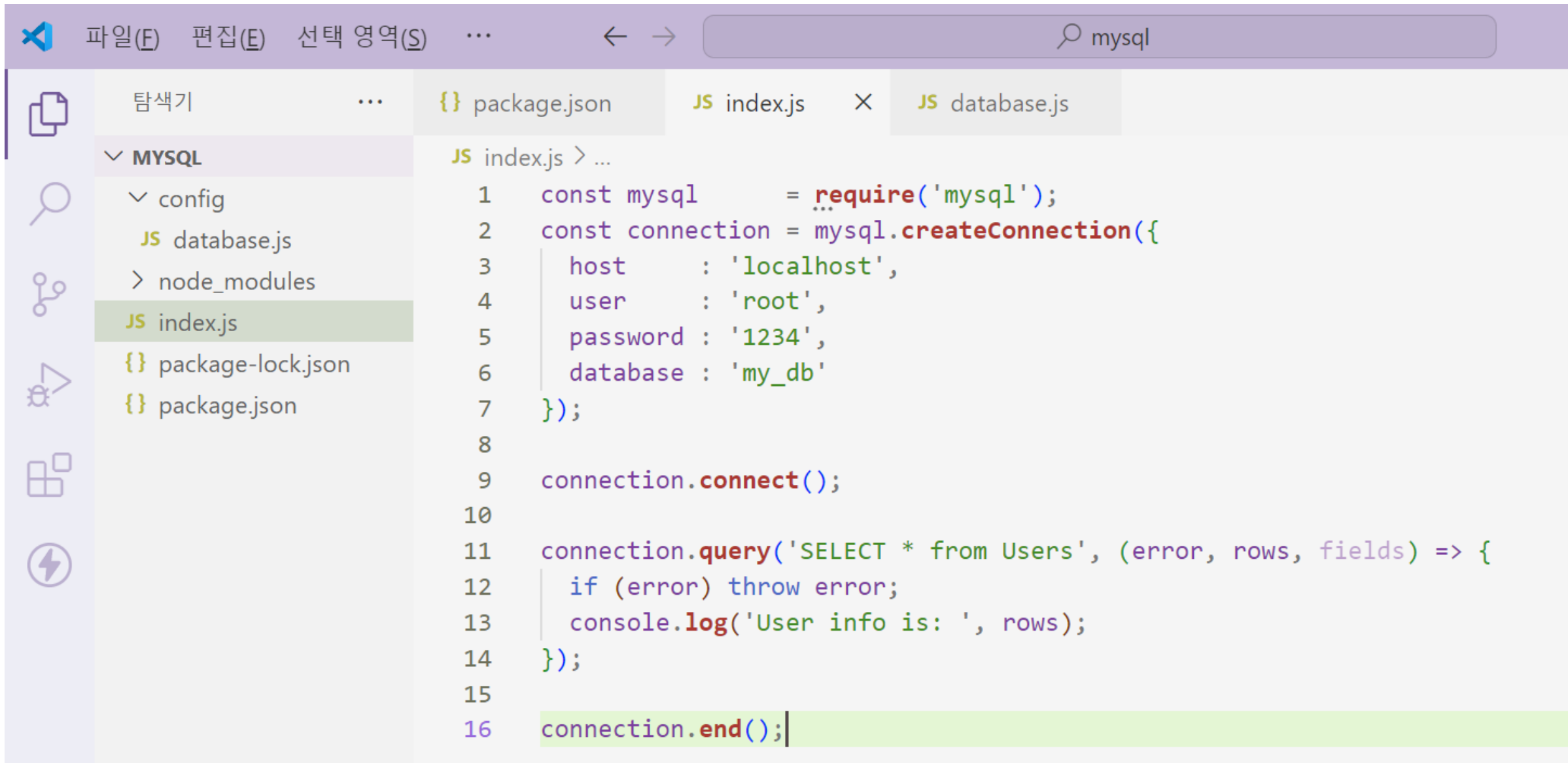
USE my_db;

CREATE TABLE IF NOT EXISTS Users (
  id VARCHAR(45) NOT NULL,
  password VARCHAR(45) NOT NULL,
  PRIMARY KEY (id));

INSERT INTO Users (id, password) VALUES ('ungmo2', '1234');

SELECT password FROM Users WHERE id='ungmo2';
```

Node.js 와 MySQL 연동 – index.js를 다음과 같이 변경 함



The screenshot shows the Visual Studio Code editor interface. The top bar includes the menu (File, Edit, Select Area, etc.), navigation arrows, and a search bar containing 'mysql'. The Explorer sidebar on the left shows the project structure under the 'MYSQL' folder, with 'index.js' selected. The main editor area displays the content of 'index.js'.

```
JS index.js > ...  
1  const mysql      = require('mysql');  
2  const connection = mysql.createConnection({  
3      host      : 'localhost',  
4      user      : 'root',  
5      password  : '1234',  
6      database  : 'my_db'  
7  });  
8  
9  connection.connect();  
10  
11 connection.query('SELECT * from Users', (error, rows, fields) => {  
12     if (error) throw error;  
13     console.log('User info is: ', rows);  
14 });  
15  
16 connection.end();
```

콘솔에 이하의 결과가 출력되면 성공이다.

CODE

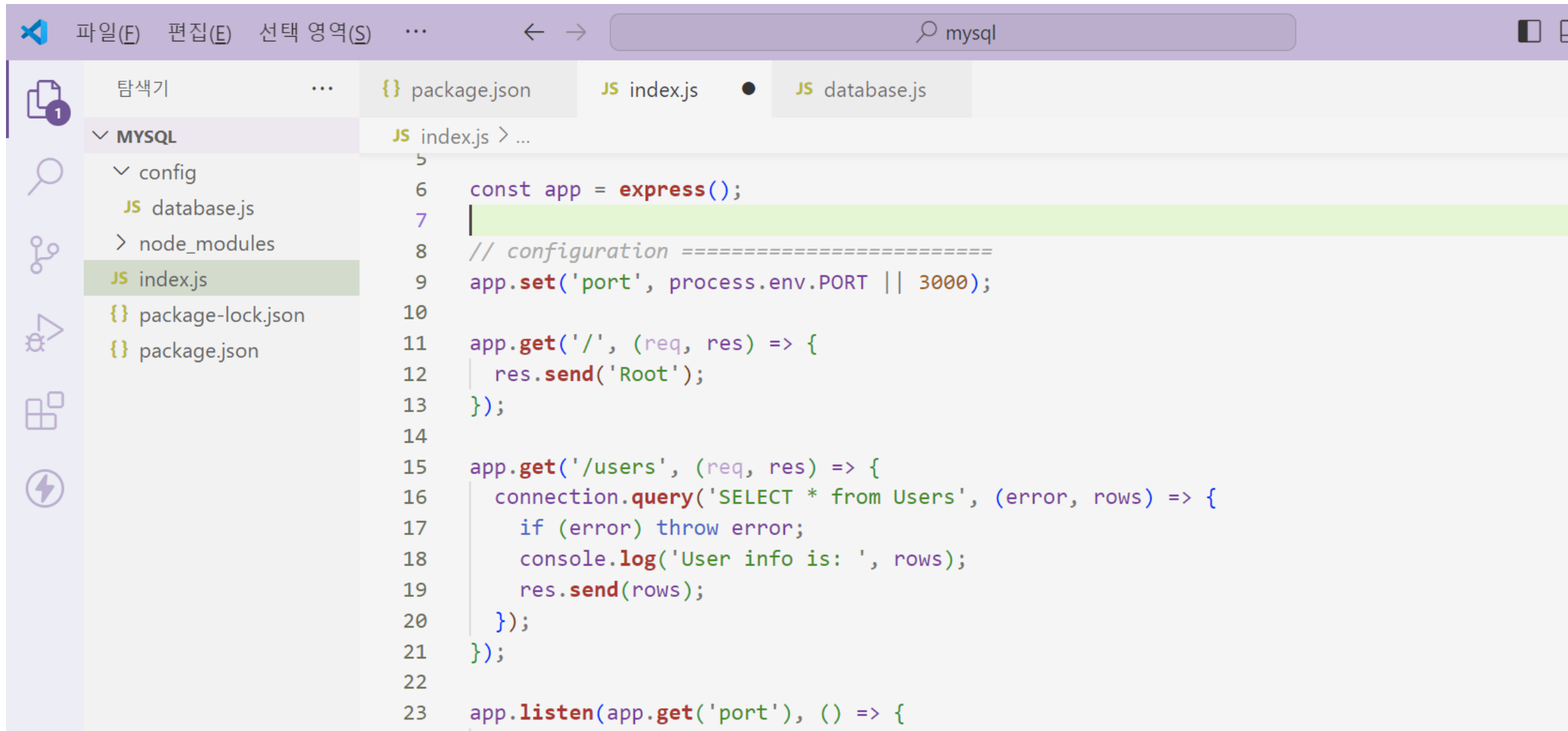
```
User info is: [ RowDataPacket { id: 'ungmo2', password: '1234' } ]
```

만약 'ER_NOT_SUPPORTED_AUTH_MODE' 에러가 발생하면 아래 sql을 실행하고 다시 접속해보자.

SQL

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '< MySQL password >';  
FLUSH PRIVILEGES;
```

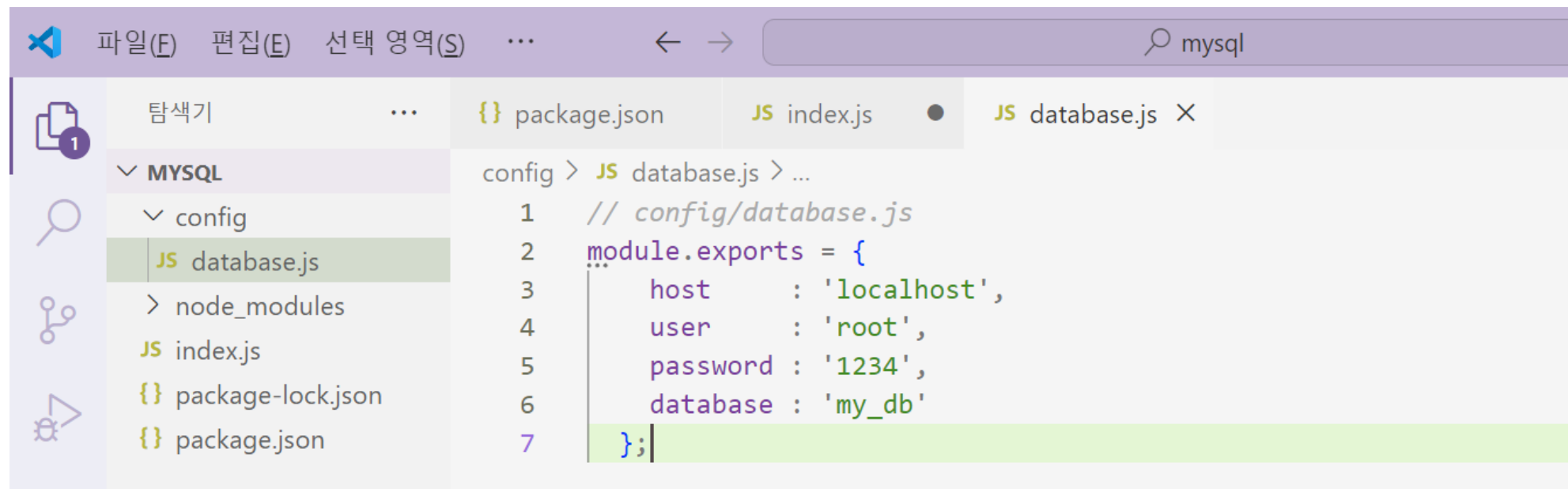
Route 작성 - index.js 수정



The screenshot shows the Visual Studio Code editor interface. The top bar includes the menu bar (파일(E), 편집(E), 선택 영역(S), ...) and a search bar containing 'mysql'. The left sidebar displays the file explorer with a search icon and a list of files under the 'MYSQL' folder: config, database.js, node_modules, index.js (selected), package-lock.json, and package.json. The main editor area shows the content of 'index.js' with line numbers 5 through 23. The code defines an Express.js application with two routes: a root route that sends 'Root' and a '/users' route that queries a database for user information.

```
5
6  const app = express();
7
8  // configuration =====
9  app.set('port', process.env.PORT || 3000);
10
11 app.get('/', (req, res) => {
12   res.send('Root');
13 });
14
15 app.get('/users', (req, res) => {
16   connection.query('SELECT * from Users', (error, rows) => {
17     if (error) throw error;
18     console.log('User info is: ', rows);
19     res.send(rows);
20   });
21 });
22
23 app.listen(app.get('port'), () => {
```

데이터베이스 설정 정보를 아래와 같이 작성한 후 루트 디렉터리 아래 config 디렉터리에 저장 – database.js



The screenshot shows the Visual Studio Code editor interface. The top menu bar includes '파일(F)', '편집(E)', '선택 영역(S)', and a search bar with 'mysql' entered. The Explorer sidebar on the left shows a project structure with a 'MYSQL' folder containing a 'config' subfolder. Inside 'config', the file 'database.js' is selected. The Editor pane shows the content of 'database.js' with the following code:

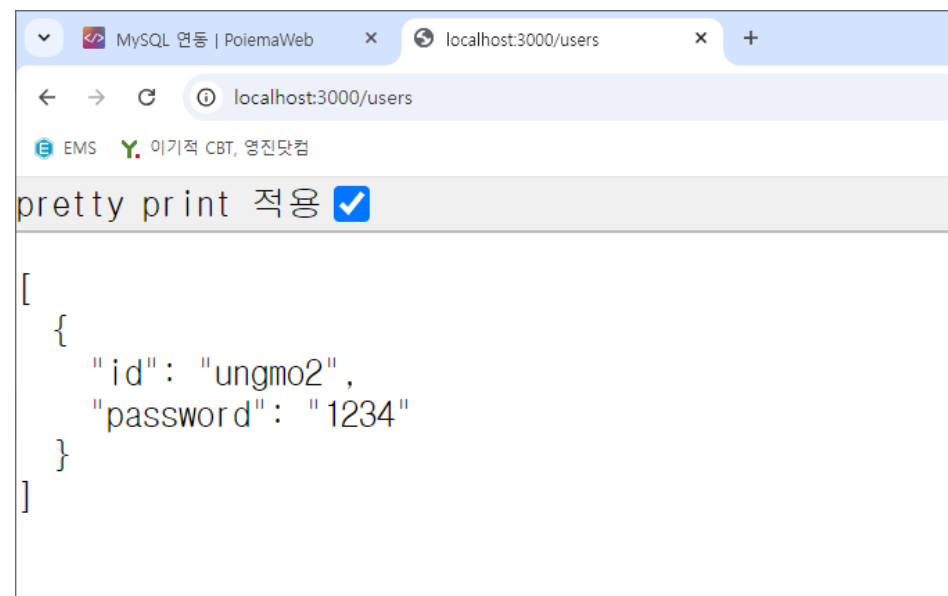
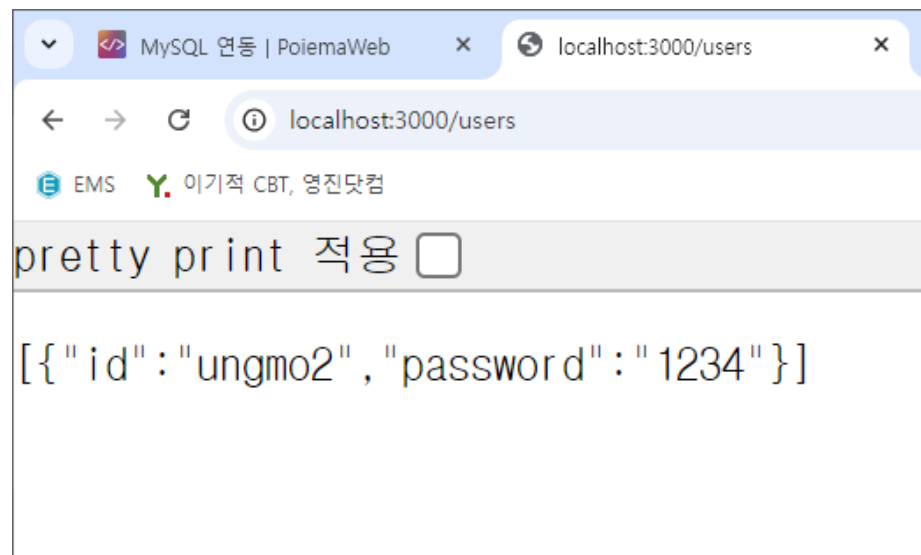
```
config > JS database.js > ...
1 // config/database.js
2 module.exports = {
3     host      : 'localhost',
4     user      : 'root',
5     password  : '1234',
6     database  : 'my_db'
7 };
```

서버를 실행한다.

BASH

```
$ npm start
```

localhost:3000/users에 접속하여 결과를 확인한다.



Node.js 와 MySQL을 이용한 로그인/회원가입 예제

필요 모듈

1. mysql2
2. express
3. express-session
4. session-file-store
5. body-parser

npm을 이용하여 다운로드하시면 됩니다.

mysql2 대신 mysql 모듈을 사용하셔도 무방하며,
session-file-store 모듈은 세션을 다른 방식으로 저장하실 경우 사용하지 않으셔도 됩니다.
body-parser 역시 편의를 위한 것으로, 소스코드에서 request.body 부분만 수정하시고 사용하셔도 됩니다.

MySQL 세팅

사용을 원하시는 database에서 아래와 같이 테이블을 만들어 주시면 됩니다.
MySQL이 익숙하지 않으시면 [링크](#) 를 참고해주세요

```
CREATE TABLE userTable (  
  id int(12) NOT NULL AUTO_INCREMENT,  
  username varchar(50) NOT NULL,  
  password varchar(255) NOT NULL,  
  PRIMARY KEY(id)  
) charset=utf8;
```

Main.js

```
const express = require('express')
const session = require('express-session')
const bodyParser = require('body-parser');
const FileStore = require('session-file-store')(session)

var authRouter = require('./lib_login/auth');
var authCheck = require('./lib_login/authCheck.js');
var template = require('./lib_login/template.js');

const app = express()
const port = 3000

app.use(bodyParser.urlencoded({ extended: false }));
app.use(session({
  secret: '~~~',          // 원하는 문자 입력
  resave: false,
  saveUninitialized: true,
  store: new FileStore(),
})))

app.get('/', (req, res) => {
  if (!authCheck.isOwner(req, res)) { // 로그인 안되어있으면 로그인 페이지로 이동시킴
    res.redirect('/auth/login');
    return false;
  } else {                          // 로그인 되어있으면 메인 페이지로 이동시킴
    res.redirect('/main');
    return false;
  }
})

// 인증 라우터
app.use('/auth', authRouter);

// 메인 페이지
```

```
// 메인 페이지
app.get('/main', (req, res) => {
  if (!authCheck.isOwner(req, res)) { // 로그인 안되어있으면 로그인 페이지로 이동시킴
    res.redirect('/auth/login');
    return false;
  }
  var html = template.HTML('Welcome',
    `<hr>
      <h2>메인 페이지에 오신 것을 환영합니다</h2>
      <p>로그인에 성공하셨습니다.</p>`,
    authCheck.statusUI(req, res)
  );
  res.send(html);
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

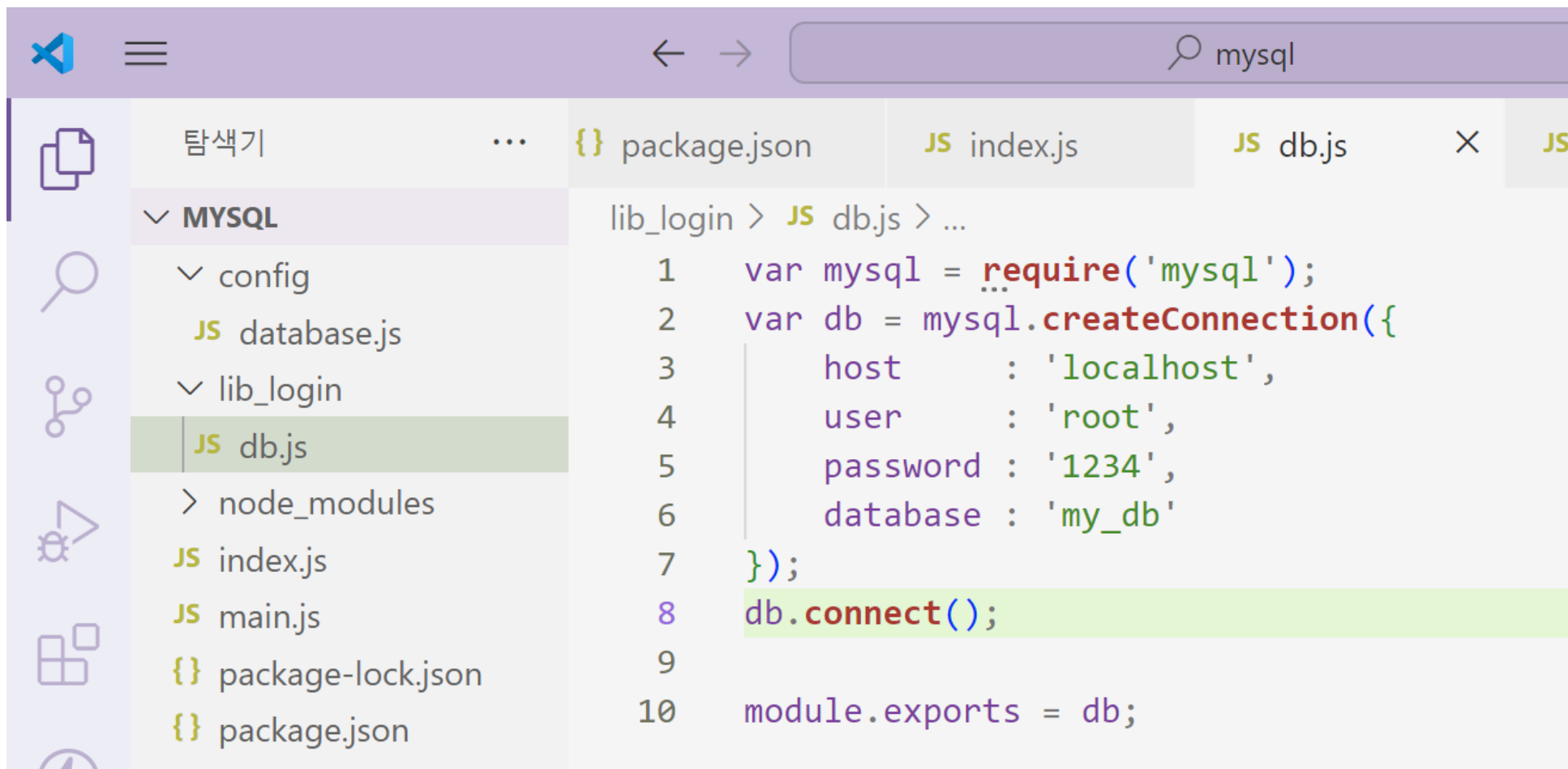
express로 서버를 실행시키는 main.js 파일입니다.

각 페이지의 세부 구현은 lib_login 폴더에 있는 파일에 저장되어 있으며, 경로의 수정을 원하는 경우 바뀌어서 사용하시면 됩니다.

페이지의 root 에 접속했을 때 세션을 확인해서 로그인 되어있으면 /main 페이지로, 아니면 /auth/login 페이지로 리다이렉션 시킵니다.

/main 페이지 내용은 간단하게 해당 파일에 작성되어있으며, 각종 인증관련 페이지는 /auth/ 경로를 통해 접속합니다.

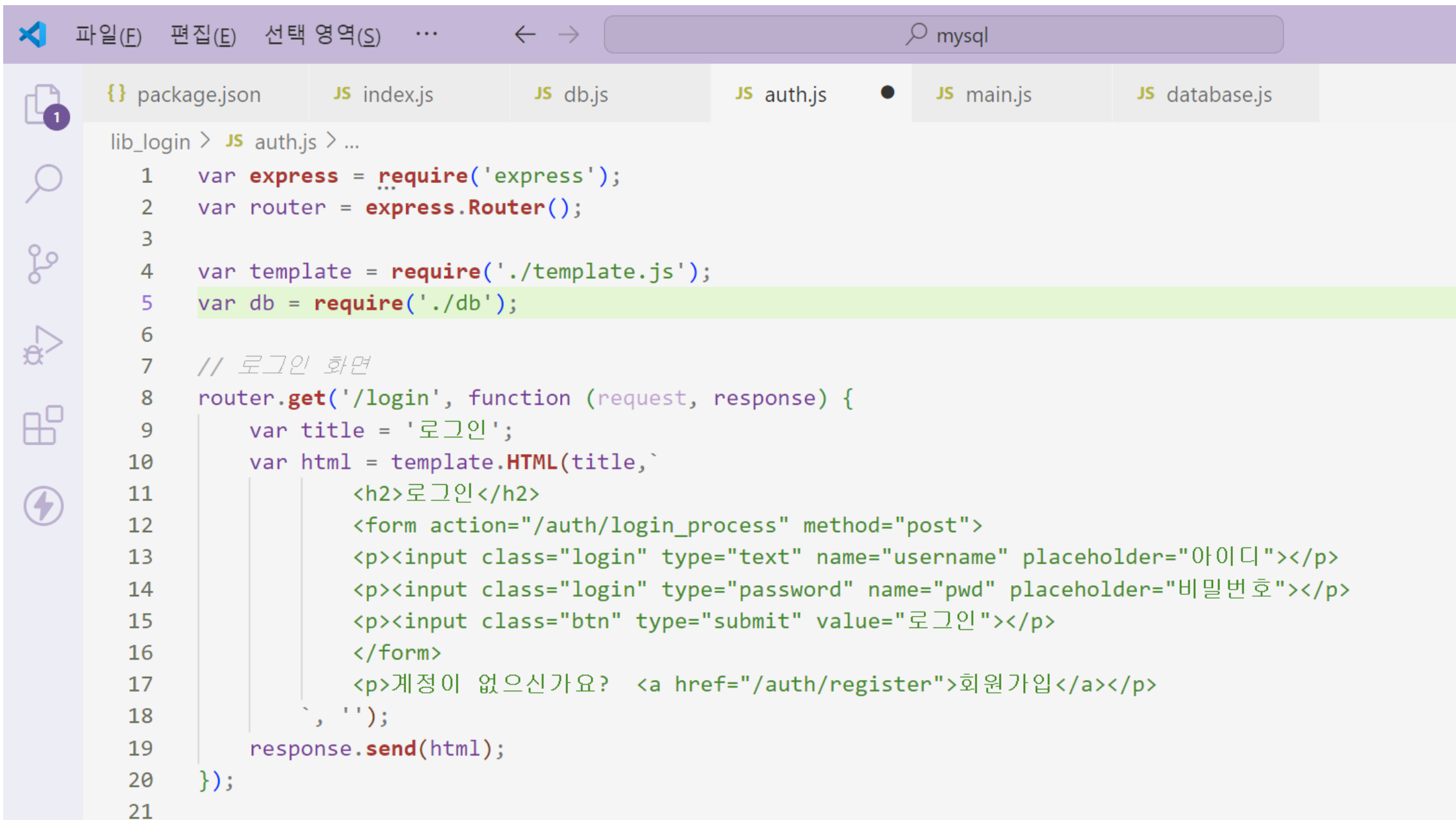
lib_login 폴더에 db.js 파일 생성



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure. Under the 'lib_login' folder, a new file 'db.js' has been created and is highlighted. The main editor area shows the content of 'db.js', which uses the 'mysql' module to create a database connection. The code is as follows:

```
lib_login > JS db.js > ...
1  var mysql = require('mysql');
2  var db = mysql.createConnection({
3      host      : 'localhost',
4      user      : 'root',
5      password  : '1234',
6      database  : 'my_db'
7  });
8  db.connect();
9
10 module.exports = db;
```

lib_login 폴더에 auth.js 파일 생성



```
lib_login > JS auth.js > ...
1  var express = require('express');
2  var router = express.Router();
3
4  var template = require('./template.js');
5  var db = require('./db');
6
7  // 로그인 화면
8  router.get('/login', function (request, response) {
9      var title = '로그인';
10     var html = template.HTML(title, `
11         <h2>로그인</h2>
12         <form action="/auth/login_process" method="post">
13             <p><input class="login" type="text" name="username" placeholder="아이디"></p>
14             <p><input class="login" type="password" name="pwd" placeholder="비밀번호"></p>
15             <p><input class="btn" type="submit" value="로그인"></p>
16         </form>
17         <p>계정이 없으신가요? <a href="/auth/register">회원가입</a></p>
18     `, '');
19     response.send(html);
20 });
21
```



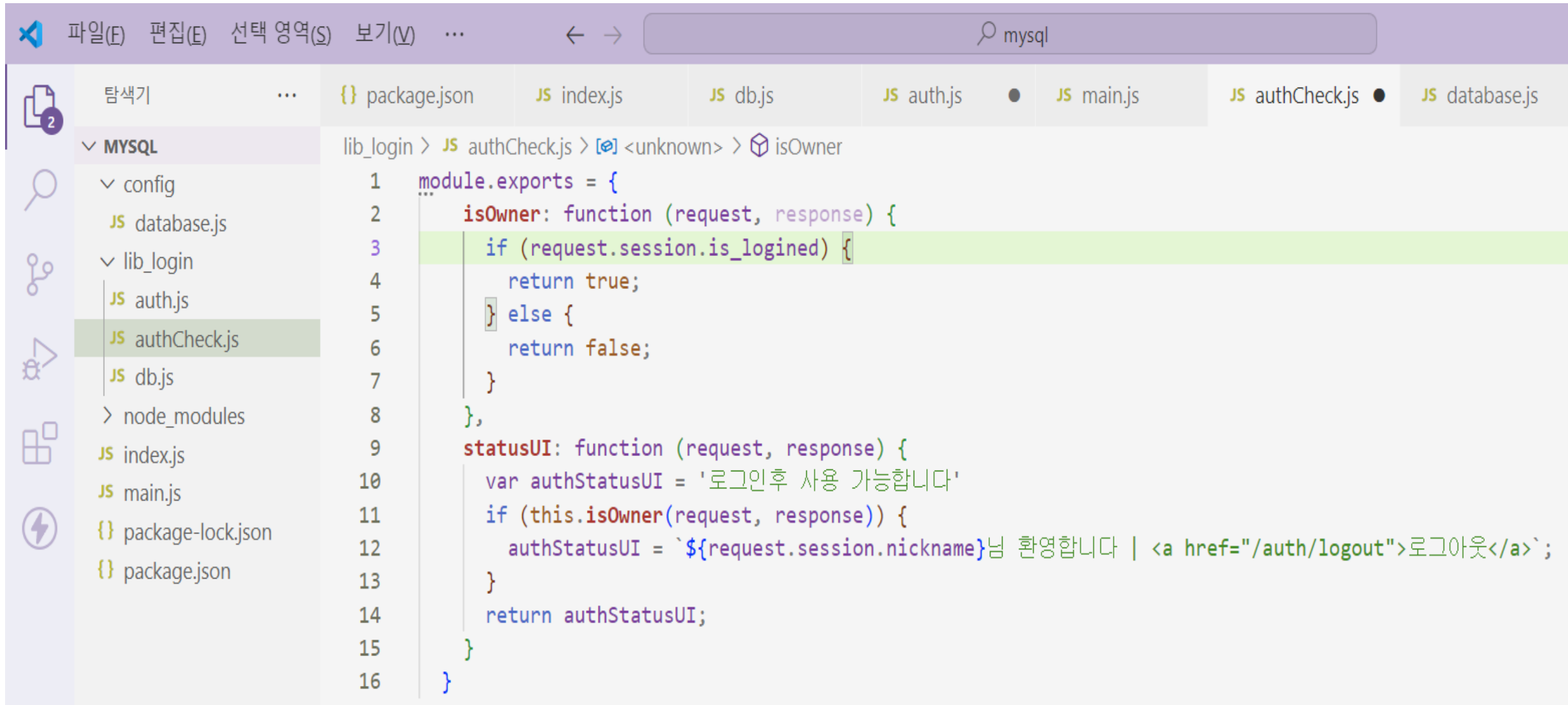
```
22 // 로그인 프로세스
23 router.post('/login_process', function (request, response) {
24     var username = request.body.username;
25     var password = request.body.pwd;
26     if (username && password) {          // id와 pw가 입력되었는지 확인
27
28         db.query('SELECT * FROM usertable WHERE username = ? AND password = ?', [username, password], function(error, results, fields) {
29             if (error) throw error;
30             if (results.length > 0) {    // db에서의 반환값이 있으면 로그인 성공
31                 request.session.is_logged = true;    // 세션 정보 갱신
32                 request.session.nickname = username;
33                 request.session.save(function () {
34                     response.redirect(`/`);
35                 });
36             } else {
37                 response.send(`
```



```
48 // 로그아웃
49 router.get('/logout', function (request, response) {
50     request.session.destroy(function (err) {
51         response.redirect('/');
52     });
53 });
54
55
56 // 회원가입 화면
57 router.get('/register', function(request, response) {
58     var title = '회원가입';
59     var html = template.HTML(title, `
60     <h2>회원가입</h2>
61     <form action="/auth/register_process" method="post">
62     <p><input class="login" type="text" name="username" placeholder="아이디"></p>
63     <p><input class="login" type="password" name="pwd" placeholder="비밀번호"></p>
64     <p><input class="login" type="password" name="pwd2" placeholder="비밀번호 재확인"></p>
65     <p><input class="btn" type="submit" value="제출"></p>
66     </form>
67     <p><a href="/auth/login">로그인화면으로 돌아가기</a></p>
68     `, '');
69     response.send(html);
70 });
71
```

```
72 // 회원가입 프로세스
73 router.post('/register_process', function(request, response) {
74     var username = request.body.username;
75     var password = request.body.pwd;
76     var password2 = request.body.pwd2;
77     if (username && password && password2) {
78         db.query('SELECT * FROM usertable WHERE username = ?', [username], function(error, results, fields) { // DB에 같은 이름의 회원아이디가 있는
79             if (error) throw error;
80             if (results.length <= 0 && password == password2) { // DB에 같은 이름의 회원아이디가 없고, 비밀번호가 올바르게 입력된 경우
81                 db.query('INSERT INTO usertable (username, password) VALUES(?,?)', [username, password], function (error, data) {
82                     if (error) throw error2;
83                     response.send('<script type="text/javascript">alert("회원가입이 완료되었습니다!");');
84                     document.location.href="/";</script>`);
85                 });
86             } else if (password != password2) { // 비밀번호가 올바르게 입력되지 않은 경우
87                 response.send('<script type="text/javascript">alert("입력된 비밀번호가 서로 다릅니다.");');
88                 document.location.href="/auth/register";</script>`);
89             }
90             else { // DB에 같은 이름의 회원아이디가 있는 경우
91                 response.send('<script type="text/javascript">alert("이미 존재하는 아이디 입니다.");');
92                 document.location.href="/auth/register";</script>`);
93             }
94         });
95     } else { // 입력되지 않은 정보가 있는 경우
96         response.send('<script type="text/javascript">alert("입력되지 않은 정보가 있습니다.");');
97         document.location.href="/auth/register";</script>`);
98     }
99 }
100 });
101 module.exports = router;
```


lib_login 폴더에 authCheck.js 파일 생성



The screenshot shows the Visual Studio Code interface. The top menu bar includes '파일(E)', '편집(E)', '선택 영역(S)', '보기(V)', and a search bar containing 'mysql'. The Explorer sidebar on the left shows a project structure with folders 'MYSQL' and 'lib_login'. The 'lib_login' folder is expanded, showing files 'auth.js', 'authCheck.js' (selected), 'db.js', 'index.js', 'main.js', 'package-lock.json', and 'package.json'. The Editor pane displays the content of 'authCheck.js' with the following code:

```
lib_login > JS authCheck.js > [?] <unknown> > isOwner
1  module.exports = {
2      isOwner: function (request, response) {
3          if (request.session.is_loggedin) {
4              return true;
5          } else {
6              return false;
7          }
8      },
9      statusUI: function (request, response) {
10         var authStatusUI = '로그인후 사용 가능합니다'
11         if (this.isOwner(request, response)) {
12             authStatusUI = `${request.session.nickname}님 환영합니다 | <a href="/auth/logout">로그아웃</a>`;
13         }
14         return authStatusUI;
15     }
16 }
```

lib_login 폴더에 template.js 파일 생성



```
1 module.exports = {
2   HTML: function (title, body, authStatusUI) {
3     return `
4     <!doctype html>
5     <html>
6     <head>
7       <title>Login TEST - ${title}</title>
8       <meta charset="utf-8">
9       <style>
10        @import url(http://fonts.googleapis.com/earlyaccess/notosanskr.css);
11
12        body {
13          font-family: 'Noto Sans KR', sans-serif;
14          background-color: #AAA2C2;
15          margin: 50px;
16
17        }
18
19        .background {
20          background-color: white;
21          height: auto;
22          width: 90%;
23          max-width: 450px;
24          padding: 10px;
25          margin: 0 auto;
26          border-radius: 5px;
27          box-shadow: 0px 40px 30px -20px rgba(0, 0, 0, 0.3);
28          text-align: center;
29        }
30      </style>
31    `
32  }
33 }
```



lib_login

JS auth.js

JS authCheck.js

JS db.js

JS template.js

> node_modules

JS index.js

JS main.js

{ } package-lock.json

{ } package.json



form {

display: flex;

padding: 30px;

flex-direction: column;

}

.login {

border: none;

border-bottom: 2px solid #D1D1D4;

background: none;

padding: 10px;

font-weight: 700;

transition: .2s;

width: 75%;

}

.login:active,

.login:focus,

.login:hover {

outline: none;

border-bottom-color: #6A679E;

}

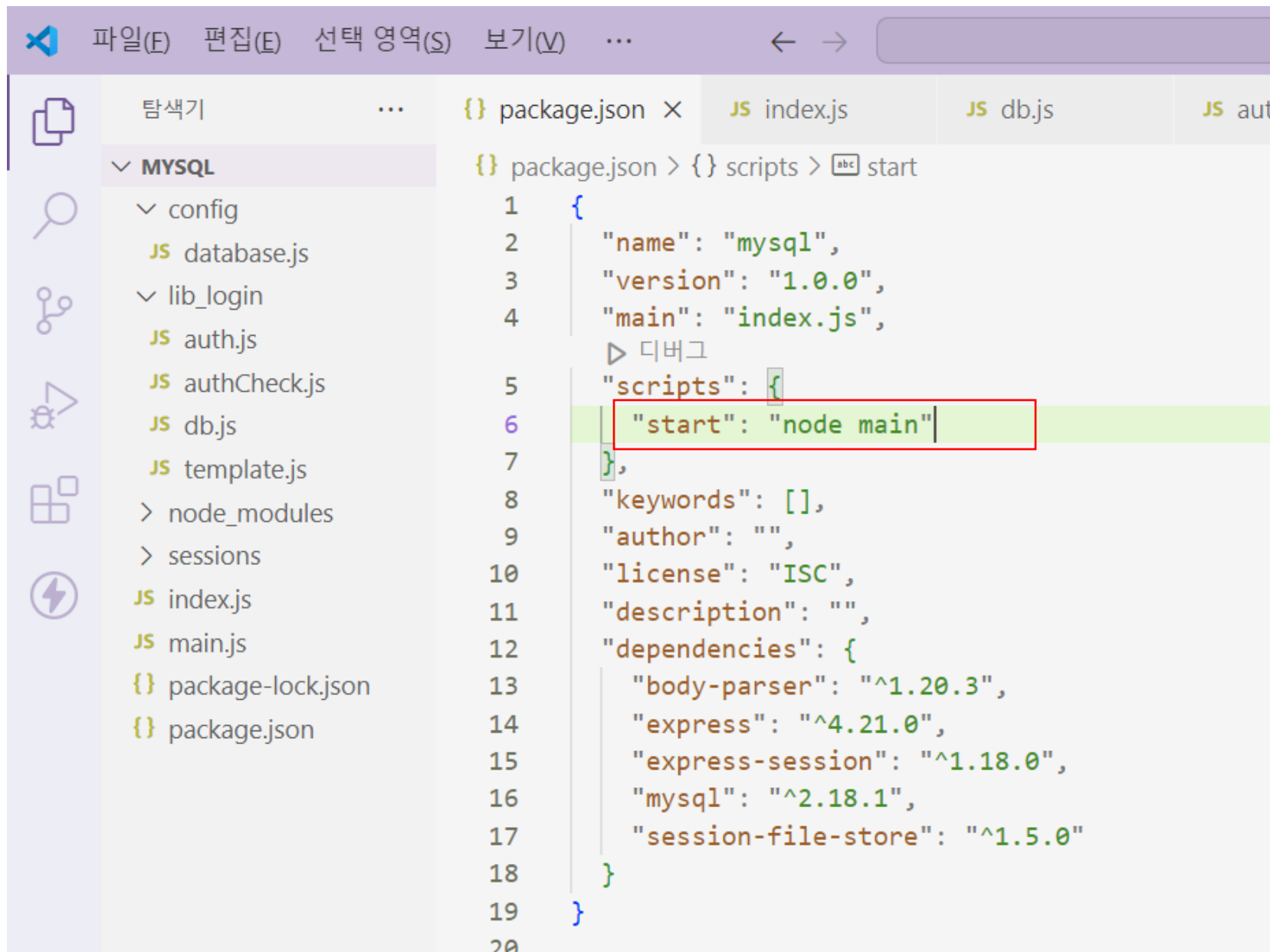


JS auth.js
JS authCheck.js
JS db.js
JS template.js
> node_modules
JS index.js
JS main.js
{ } package-lock.json
{ } package.json

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78

```
.btn {  
  border: none;  
  width: 75%;  
  background-color: #6A679E;  
  color: white;  
  padding: 15px 0;  
  font-weight: 600;  
  border-radius: 5px;  
  cursor: pointer;  
  transition: .2s;  
}  
.btn:hover {  
  background-color: #595787;  
}  
</style>  
</head>  
<body>  
  <div class="background">  
    ${authStatusUI}  
    ${body}  
  </div>  
</body>  
</html>  
`;  
}  
}
```

package.json 파일 수정 후



The screenshot shows the Visual Studio Code editor interface. The left sidebar displays the file explorer with a project structure including a 'MYSQL' folder and various JavaScript files. The main editor area shows the 'package.json' file, which is a JSON configuration file for a Node.js project. The 'scripts' section is expanded, and the 'start' script is highlighted with a red box. The 'start' script is set to 'node main'.

```
{
  "name": "mysql",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node main"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "body-parser": "^1.20.3",
    "express": "^4.21.0",
    "express-session": "^1.18.0",
    "mysql": "^2.18.1",
    "session-file-store": "^1.5.0"
  }
}
```

npm start 실행

js와 MySQL을 이용한 토 x Login TEST - 로그인 x +

localhost:3000/auth/login

기적 CBT, 영진닷컴

로그인

아이디

비밀번호

로그인

계정이 없으신가요? [회원가입](#)

토 x Login TEST - 회원가입 x +

3000/auth/register

회원가입

아이디

비밀번호

비밀번호 재확인

제출

[로그인화면으로 돌아가기](#)

회원가입

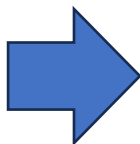
lbi5320

.....

.....|

제출

[로그인화면으로 돌아가기](#)



localhost:3000 내용:
회원가입이 완료되었습니다!

확인

PERFORMA

47



Dash

48



Perf

49



Perf

50

51



Result Grid



Filter Rows:

Edit:



Export/Import:



Wrap Cell Content:



	id	username	password
▶	1	lbi5320	lbi#1017
•	NULL	NULL	NULL

Ad



lbi5320님 환영합니다 | [로그아웃](#)

메인 페이지에 오신 것을 환영합니다

로그인에 성공하셨습니다.