Basic Login

308-232 FRONTEND APPLICATION DEVELOPMENT

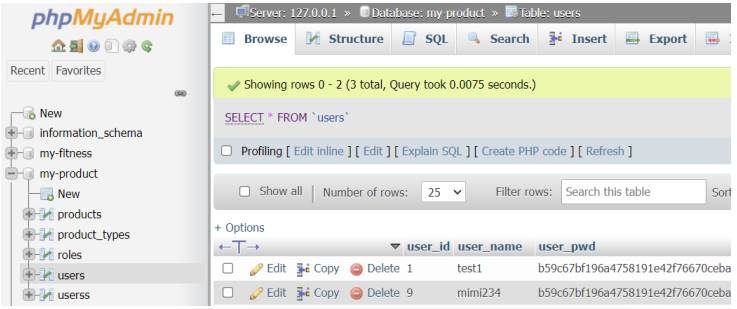
Login

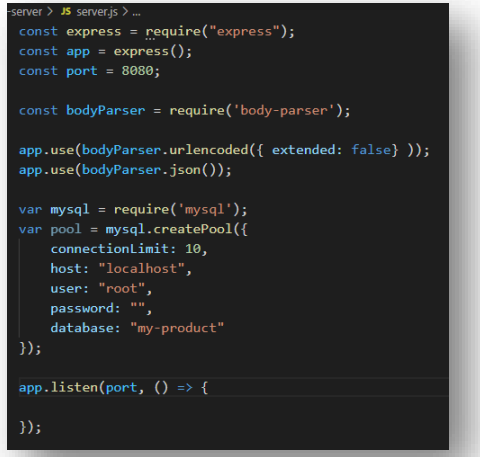
Front-end
port: 3000
file: Login.js
folder: myreact

npm install react-bootstrap bootstrap@5.1.3

npm install react-router-dom@6

Back-end
port:8080
file: server.js
folder: my-app-server





```
const express = require("express");
const app = express();
const port = 8080;

const bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: false} ));
app.use(bodyParser.json());

var mysql = require('mysql');
var pool = mysql.createPool({
    connectionLimit: 10,
    host: "localhost",
    user: "root",
    password: "",
    database: "my-product"
});

app.listen(port, () => {

});
```

npm install express

npm install cors

Login
สร้าง Back-end

เตรียม server.js สำหรับ
การดำเนินการเข้าสู่ระบบ

Express เป็น Package ที่ทำให้เราสามารถตอบสนองต่อการร้องขอผ่าน URL ที่แตกต่างกัน
แต่ละ URL จะได้รับผลลัพธ์ที่ไม่เหมือนกัน เช่น

ถ้า Client ส่ง URL เป็น http://localhost:3000/users → เราจะส่งข้อมูลผู้ใช้ทั้งหมดไปให้ Front-end

http://localhost:3000/add_user → เราจะรับข้อมูลจากผู้ใช้เพื่อเพิ่มข้อมูลลงในตาราง users

http://localhost:3000/user/1 → เราจะส่งข้อมูลผู้ใช้ที่มี user_id = 1 ไปให้

server > JS server.js > ...

```
const express = require("express");
const app = express();
const port = 8080;

const bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: false} ));
app.use(bodyParser.json());

var mysql = require('mysql');
var pool = mysql.createPool({
    connectionLimit: 10,
    host: "localhost",
    user: "root",
    password: "",
    database: "my-product"
});

app.listen(port, () => {

});
```

Login
สร้าง Back-end

ใช้ express สำหรับให้บริการข้อมูลการเข้าสู่ระบบ

```
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

Login
สร้าง Back-end


```
app.post("/login", (req, res) => {
    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

รับค่าข้อมูล username และ password จากฝั่ง front-end

Login
สร้าง Back-end

```
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query('SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)', [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

ตรวจสอบว่าในตาราง users มี username และ password ที่ตรงกันหรือไม่

Login
สร้าง Back-end

```
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

ถ้าเกิด error ขึ้นในฐานข้อมูลให้ส่ง object ที่ประกอบด้วยตัวแปร 2 ตัว คือ result และ message

result => ผลลัพธ์การเข้าสู่ระบบ
message => ข้อความแจ้งการเกิด error

Login
สร้าง Back-end

ตัวแปร
- results เก็บผลลัพธ์ที่ได้จากคำสั่ง select
- result เก็บค่า true/false

```
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

results.length คือตัวแปรที่บอกว่าผลลัพธ์จากคำสั่ง query ของเรามีอยู่กี่แถว

ในกรณีที่พบข้อมูลที่ตรงกัน result.length จะมีค่า >= 1 ทำให้เงื่อนไขใน if เป็นจริง

เราจะส่งข้อมูลกลับเป็น object ที่มีสมาชิก 1 ตัวคือ result = true เพื่อบอกว่าฝั่ง front-end ว่า login สำเร็จ

ข้อสังเกต
ตัวแปร
- results
- result
เป็นคนละตัวกัน

Login
สร้าง Back-end

```
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password], function(error, results, fields){
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

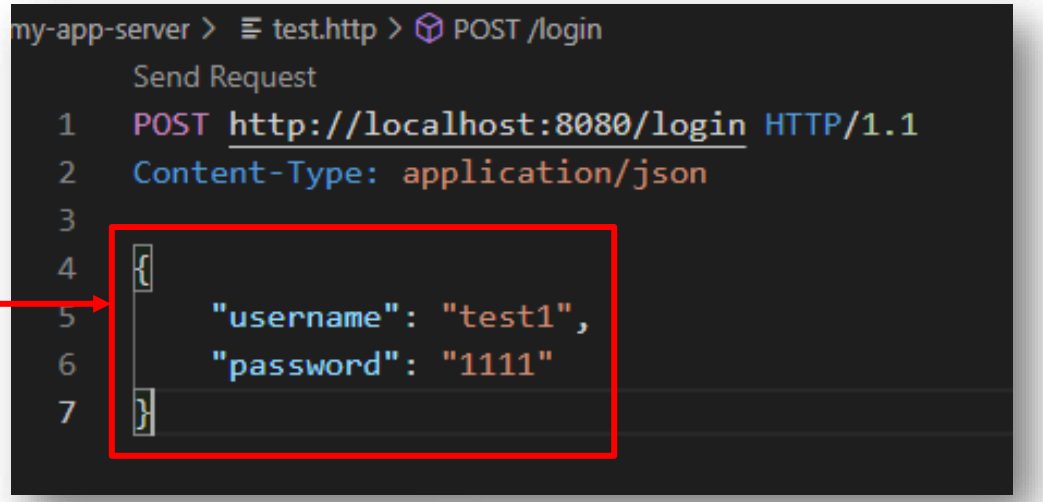
กรณีนี้จะเกิดขึ้นเมื่อ results.length = 0 นั่นแปลว่า เราไม่พบข้อมูล username และ password ไม่ตรงกัน
ดังนั้นเราจะส่ง object ที่มีสมาชิก 2 ตัวคือ
result = false เพื่อบอกว่าการ login ไม่สำเร็จ
message = แจ้ง front-end ถึงเหตุผลที่ทำให้ login ไม่สำเร็จ

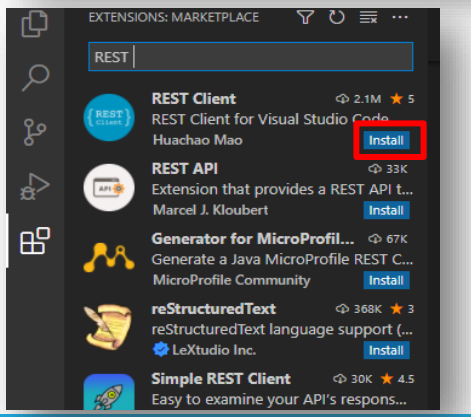
Login

สร้าง Back-end > ทดสอบการทำงานของ Back-end

แก้ไขไฟล์ test.http เพื่อทดสอบการทำงานของ /login

ต้องแน่ใจได้ว่าตัวแปรที่เราส่งไป server มีชื่อที่ตรงกัน





เพิ่ม Rest API

React-bootstrap

เป็น library สำหรับ react

เป็น open source

สร้าง react component โดยนำ bootstrap มาใช้งาน

ทำไมต้อง react-bootstrap

- เพราะมีกลไกการทำงานที่รองรับการใช้งานบน react
- เมื่อเรา install ตัว react-component เราก็จะได้ bootstrap มาพร้อมกันด้วย

นักศึกษาสามารถศึกษาข้อมูลเพิ่มเติมได้ที่ https://react-bootstrap.github.io/getting-started/introduction

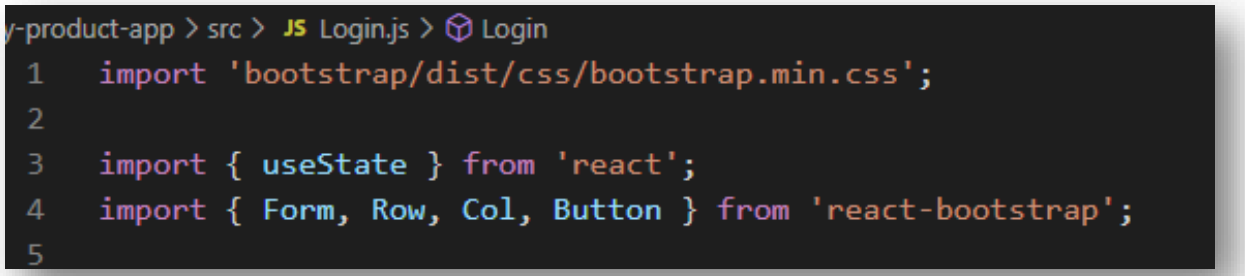
```
npm install react-bootstrap bootstrap@5.1.3
```

Install ตัว react-bootstrap ใน react project ของเรา

Login
สร้าง Front-end

สร้างไฟล์ชื่อ Login.js

Import ตัว library ต่าง ๆ ที่ต้องการใช้งาน



Login
สร้าง Front-end

สร้าง state สำหรับ

validated ➔ ตรวจสอบว่าผู้ใช้กรอกข้อมูลครบหรือไม่

username ➔ เก็บข้อมูล username ที่ผู้ใช้ป้อน

Password ➔ เก็บข้อมูล password ที่ผู้ใช้ป้อน

```
export default function Login() {

    const [validated, setValidated] = useState(false);
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
```

Login
สร้าง Front-end

สร้าง User interface


```jsx
return (
    <div className='container m-auto'>
        <Form noValidate validated={validated} onSubmit={onLogin}>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validateUsername">
                    <Form.Label>Username</Form.Label>
                    <Form.Control
                        required
                        type="text"
                        placeholder="Username"
                        onChange={(e) => setUsername(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Username
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validatePassword">
                    <Form.Label>Password</Form.Label>
                    <Form.Control
                        required
                        type="password"
                        placeholder="Password"
                        onChange={(e) => setPassword(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Password
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row>
                <Col md={3}>
                    <Button type="submit">Login</Button>
                </Col>
            </Row>
        </Form>
    </div>
);
```

Login
สร้าง Front-end

ใช้สำหรับบันทึกว่ามีการตรวจสอบการกรอกข้อมูลของผู้ใช้ไว้หรือยัง

```
return (
    <div className='container m-auto'>
        <Form noValidate validated={validated} onSubmit={onLogin}>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validateUsername">
                    <Form.Label>Username</Form.Label>
                    <Form.Control
                        required
                        type="text"
                        placeholder="Username"
                        onChange={(e) => setUsername(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Username
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validatePassword">
                    <Form.Label>Password</Form.Label>
                    <Form.Control
                        required
                        type="password"
                        placeholder="Password"
                        onChange={(e) => setPassword(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Password
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row>
                <Col md={3}>
                    <Button type="submit">Login</Button>
                </Col>
            </Row>
        </Form>
    </div>
);
```

Login
สร้าง Front-end

สร้าง Event handler เพื่อให้ทำงานในกรณีที่ผู้ใช้กดปุ่ม Login

```
return (
    <div className='container m-auto'>
        <Form noValidate validated={validated} onSubmit={onLogin}>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validateUsername">
                    <Form.Label>Username</Form.Label>
                    <Form.Control
                        required
                        type="text"
                        placeholder="Username"
                        onChange={(e) => setUsername(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Username
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validatePassword">
                    <Form.Label>Password</Form.Label>
                    <Form.Control
                        required
                        type="password"
                        placeholder="Password"
                        onChange={(e) => setPassword(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Password
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row>
                <Col md={3}>
                    <Button type="submit">Login</Button>
                </Col>
            </Row>
        </Form>
    </div>
);
```

Login
สร้าง Front-end

บังคับให้ผู้ใช้ต้องกรอกข้อมูลในส่วนนี้

```
return (
    <div className='container m-auto'>
        <Form noValidate validated={validated} onSubmit={onLogin}>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validateUsername">
                    <Form.Label>Username</Form.Label>
                    <Form.Control
                        required
                        type="text"
                        placeholder="Username"
                        onChange={(e) => setUsername(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Username
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validatePassword">
                    <Form.Label>Password</Form.Label>
                    <Form.Control
                        required
                        type="password"
                        placeholder="Password"
                        onChange={(e) => setPassword(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Password
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row>
                <Col md={3}>
                    <Button type="submit">Login</Button>
                </Col>
            </Row>
        </Form>
    </div>
);
```

Login
สร้าง Front-end

สร้าง Feedback เพื่อให้แสดงในกรณีที่ผู้ใช้งานยังกรอกข้อมูลไม่ครบถ้วน

```
return (
    <div className='container m-auto'>
        <Form noValidate validated={validated} onSubmit={onLogin}>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validateUsername">
                    <Form.Label>Username</Form.Label>
                    <Form.Control
                        required
                        type="text"
                        placeholder="Username"
                        onChange={(e) => setUsername(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Username
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row className="mb-3">
                <Form.Group as={Col} controlId="validatePassword">
                    <Form.Label>Password</Form.Label>
                    <Form.Control
                        required
                        type="password"
                        placeholder="Password"
                        onChange={(e) => setPassword(e.target.value)}
                    />
                    <Form.Control.Feedback type="invalid">
                        กรุณากรอก Password
                    </Form.Control.Feedback>
                </Form.Group>
            </Row>
            <Row>
                <Col md={3}>
                    <Button type="submit">Login</Button>
                </Col>
            </Row>
        </Form>
    </div>
);
```

Login
สร้าง Front-end

สร้าง function onLogin สำหรับตรวจสอบการกรอกข้อมูลของผู้ใช้

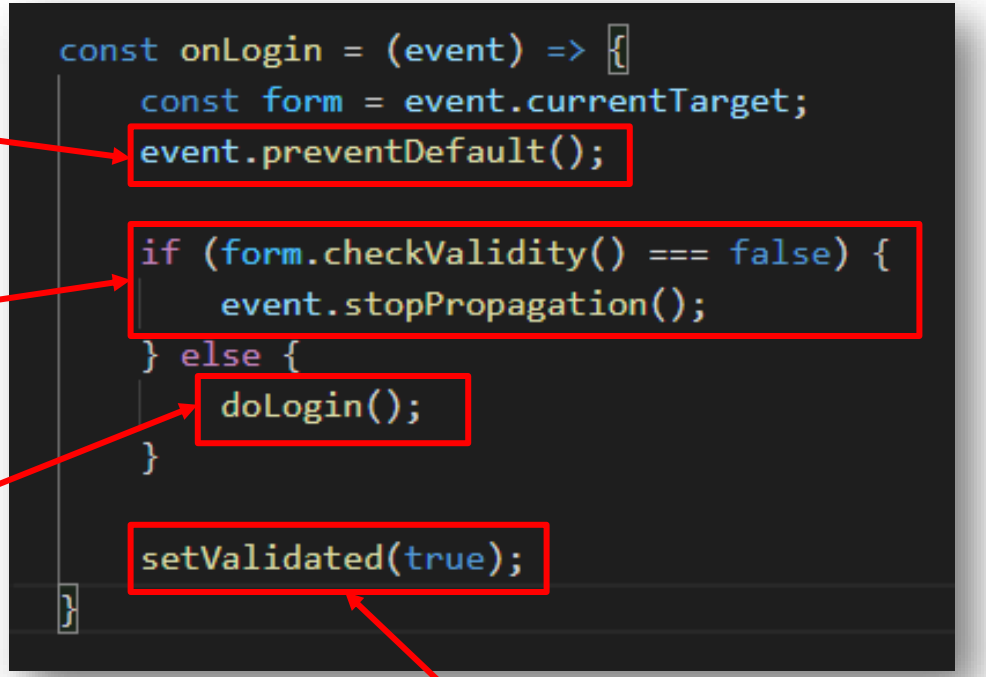
```
const onLogin = (event) => {
    const form = event.currentTarget;
    event.preventDefault();

    if (form.checkValidity() === false) {
        event.stopPropagation();
    } else {
        doLogin();
    }

    setValidated(true);
}
```

Login
สร้าง Front-end

ป้องกันไม่ให้เกิดการเปลี่ยนหน้า Page จากการ submit form

ในกรณีที่ผู้ใช้กรอกข้อมูลไม่ครบ เราจะไม่ให้เกิด event ใด ๆ ขึ้นโดยใช้คำสั่ง stopPropagation

ในกรณีที่ผู้ใช้กรอกข้อมูลครบถ้วน เราจะเรียกใช้ฟังก์ชันชื่อ doLogin เพื่อส่งข้อมูลไปยัง Back-end



เปลี่ยนแค่ตัวแปร validated เป็น true เพื่อบอกว่าเราได้ทำงานตรวจสอบข้อมูลแล้ว

Login
สร้าง Front-end

สร้างฟังก์ชันชื่อ doLogin เพื่อติดต่อไปยัง Back-end

เมื่อทดสอบ run program แล้วจะพบว่าไม่สามารถส่งข้อมูลไปได้

เหตุผลคือ กฎ CORS ไม่อนุญาตให้เราส่ง request ไปยัง URL ที่อยู่ภายนอกตัว Web

ถึงแม้จะเป็น localhost เหมือนกัน แต่อยู่ใช้กันคนละ port

```
const doLogin = async () => {
    const response = await fetch(
        "http://localhost:8080/login",
        {
            method: "POST",
            headers: {
                Accept: "application/json",
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                username: username,
                password: password
            })
        }
    );

    const data = await response.json();

    console.log(data);
}
```

```jsx
export default function Login() {

    const [validated, setValidated] = useState(false);
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
```

```jsx
<Form.Group as={Col} controlId="validateUsername">
    <Form.Label>Username</Form.Label>
    <Form.Control
        required
        type="text"
        placeholder="Username"
        onChange={(e) => setUsername(e.target.value)}
    />
    <Form.Control.Feedback type="invalid">
        กรุณากรอก Username
    </Form.Control.Feedback>
</Form.Group>
```

```jsx
<Form.Group as={Col} controlId="validatePassword">
    <Form.Label>Password</Form.Label>
    <Form.Control
        required
        type="password"
        placeholder="Password"
        onChange={(e) => setPassword(e.target.value)}
    />
    <Form.Control.Feedback type="invalid">
        กรุณากรอก Password
    </Form.Control.Feedback>
</Form.Group>
```

Front-end

```jsx
const doLogin = async () => {
    const response = await fetch(
        "http://localhost:8080/login",
        {
            method: "POST",
            headers: {
                Accept: "application/json",
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                username: username,
                password: password
            })
        }
    );

    const data = await response.json();

    console.log(data);
}
```

Back-end

```jsx
app.post("/login", (req, res) => {

    const username = req.body.username;
    const password = req.body.password;

    pool.query("SELECT * FROM users WHERE user_name = ? AND user_pwd = MD5(?)", [username, password],
        if (error) {
            res.json({
                result: false,
                message: error.message
            });
        }

        if (results.length) {
            res.json({
                result: true
            });
        } else {
            res.json({
                result: false,
                message: "ไม่พบ Username หรือ Password ไม่ถูกต้อง"
            });
        }
    });
});++
```

Login
วิธีแก้ไขปัญหาของ CORS

Install ตัว package ชื่อ cors ใน server ของเราโดยใช้คำสั่ง

npm install cors

แก้ไข server.js โดยเรียกใช้ cors เพื่อให้สามารถรับ request จาก domain อื่น ๆ ได้

ทดลอง run อีกครั้ง

p-server > JS server.js > app.post("/login") callback

```
const express = require("express");
const app = express();
const port = 8080;

const bodyParser = require('body-parser');

const cors = require("cors");

app.use(cors());
app.use(bodyParser.urlencoded({ extended: false} ));
app.use(bodyParser.json());
```

React-router

React ใช้สำหรับสร้าง Single-page web application
- ไม่มีการเปลี่ยนหน้าหรือ Refresh หน้า
- ไม่มีการเปลี่ยน URL
- จะใช้การเปลี่ยน Component ที่แสดงแทน

ทำให้ยากต่อการสร้าง web application ที่มีขนาดใหญ่และซับซ้อน

React-router ทำให้ react application สามารถเปลี่ยน URL ได้เหมือน web application ทั่วไป
- ไม่มีการเปลี่ยนหน้าหรือ Refresh
- เปลี่ยน Component ที่ใช้ในการแสดงผล

สามารถศึกษาเพิ่มเติมได้ที่ https://reactrouter.com/docs/en/v6/getting-started/tutorial

React-router

Installation

เปิด command line ใน react project แล้วพิมพ์คำสั่งต่อไปนี้

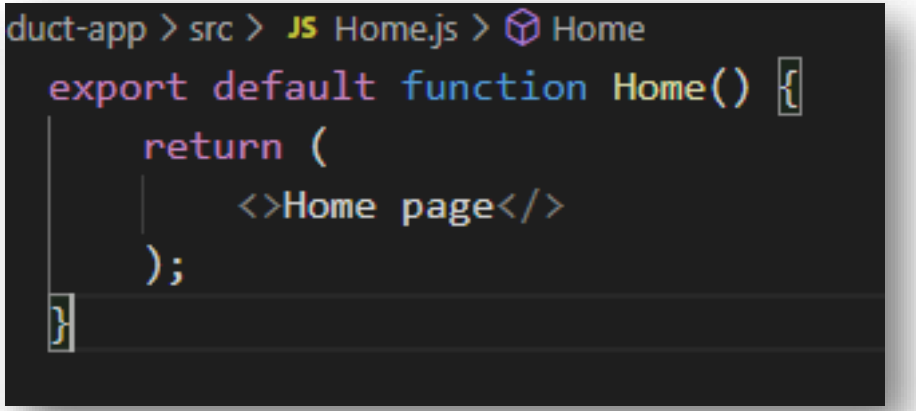
```
npm install react-router-dom@6
```

React-router
Home page

สร้าง Homepage สำหรับ Web application (Home.js)

ใน Workshop นี้เรายังไม่ใส่รายละเอียด

ใช้สำหรับทดสอบ react-router



React-router
กำหนดให้ Login.js เป็นหน้าแรกของ web application

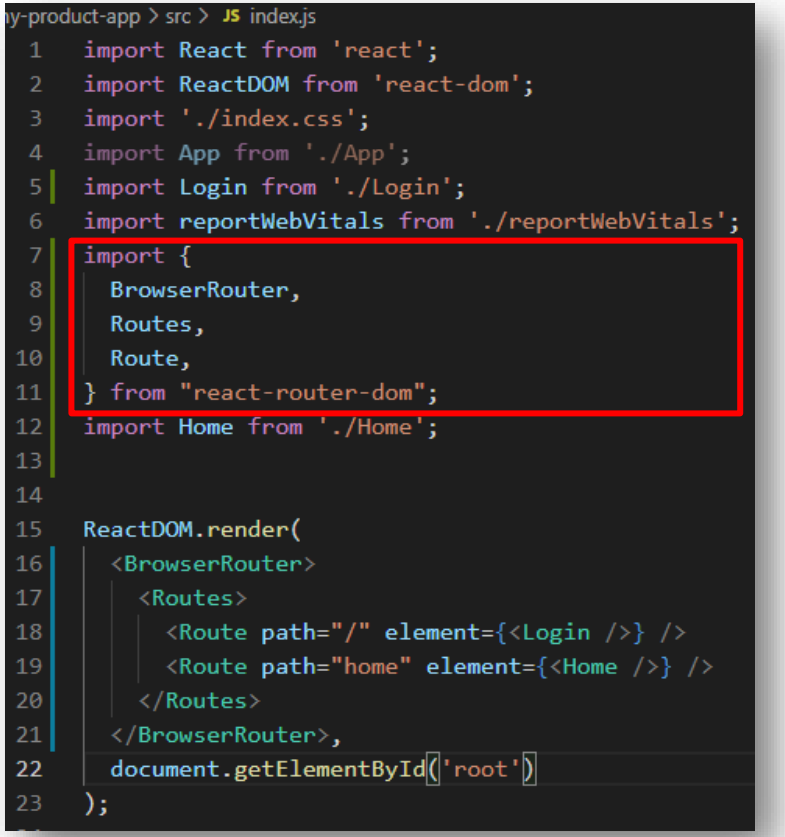
แก้ไขไฟล์ index.js ตามตัวอย่าง

ny-product-app > src > JS index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import './index.css';
4   import App from './App';
5   import Login from './Login';
6   import reportWebVitals from './reportWebVitals';
7   import {
8     BrowserRouter,
9     Routes,
10    Route,
11  } from "react-router-dom";
12  import Home from './Home';
13
14
15  ReactDOM.render(
16    <BrowserRouter>
17      <Routes>
18        <Route path="/" element={<Login />} />
19        <Route path="home" element={<Home />} />
20      </Routes>
21    </BrowserRouter>,
22    document.getElementById('root')
23  );
```

React-router
กำหนดให้ Login.js เป็นหน้าแรกของ web application

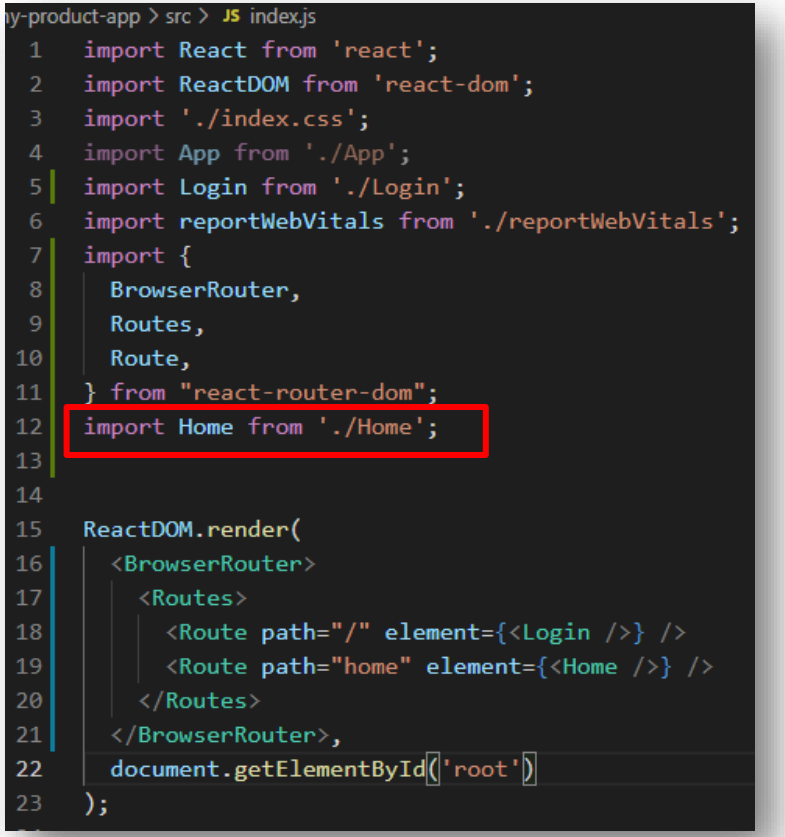
Import ตัว library ต่าง ๆ ที่จำเป็น
จาก react-router



React-router

กำหนดให้ Login.js เป็นหน้าแรกของ web application

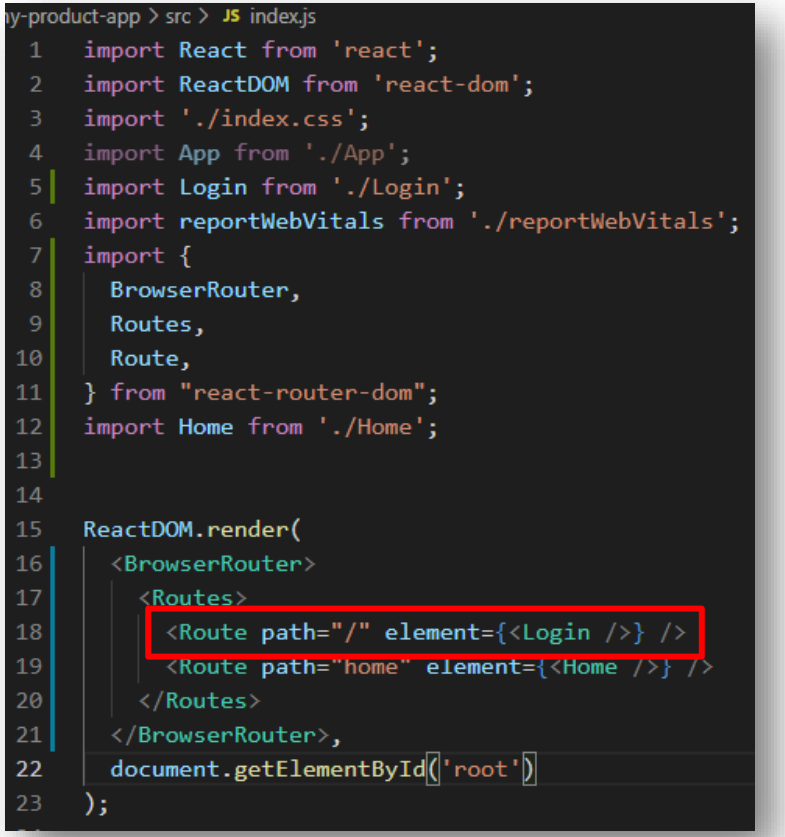
Import หน้า Home ที่เราสร้างขึ้น



React-router
กำหนดให้ Login.js เป็นหน้าแรกของ web application

ตั้งค่า root ให้แสดงหน้า Login

(/) หมายถึง หน้าแรกของ web application



React-router

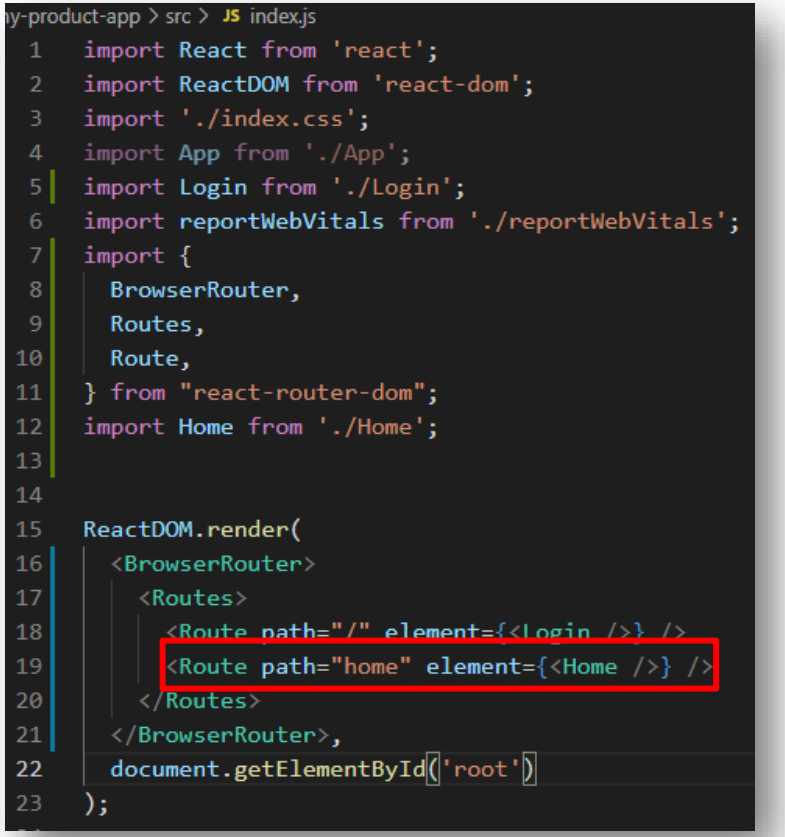
กำหนดให้ Login.js เป็นหน้าแรกของ web application

ถ้าผู้ใช้กรอก URL เป็น

localhost:3000/home

จะแสดงหน้า Home

ทดสอบ run โปรแกรม



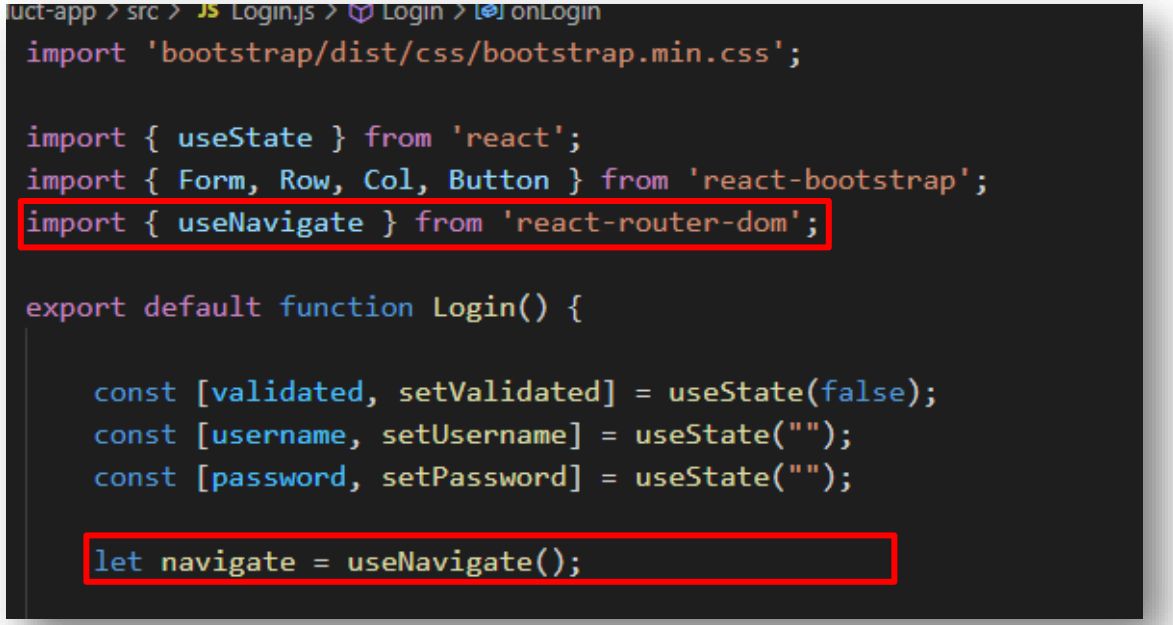
React-router

แก้ไขให้เมื่อ Login ผ่านแล้วให้นำผู้ใช้ไปยังหน้า Home

แก้ไขไฟล์ Login.js

Import ตัว library ชื่อ useNavigate ของ react-router

สร้างตัวแปรเพื่อเรียกใช้ useNavigate ในที่นี้กำหนดให้ตัวแปรมีชื่อว่า navigate



React-router
แก้ไขให้เมื่อ Login ผ่านแล้วให้นำผู้ใช้ไปยังหน้า Home

แก้ไขฟังก์ชัน doLogin

ถ้าผู้ใช้ login เข้าสู่ระบบผ่านแล้ว จะพบว่าตัวแปร result ที่ส่งมาจาก server มีค่าเป็น true

ใช้ตัวแปร navigate เพื่อเปลี่ยน URL เป็น /home

ทดลอง run program

```
const doLogin = async () => {
    const response = await fetch(
        "http://localhost:8080/login",
        {
            method: "POST",
            headers: {
                Accept: "application/json",
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                username: username,
                password: password
            })
        }
    );

    const data = await response.json();

    console.log(data);

    if (data.result) {
        navigate("home", { replace: false });
    }
}
```

(property) NavigateOptions.replace?: boolean