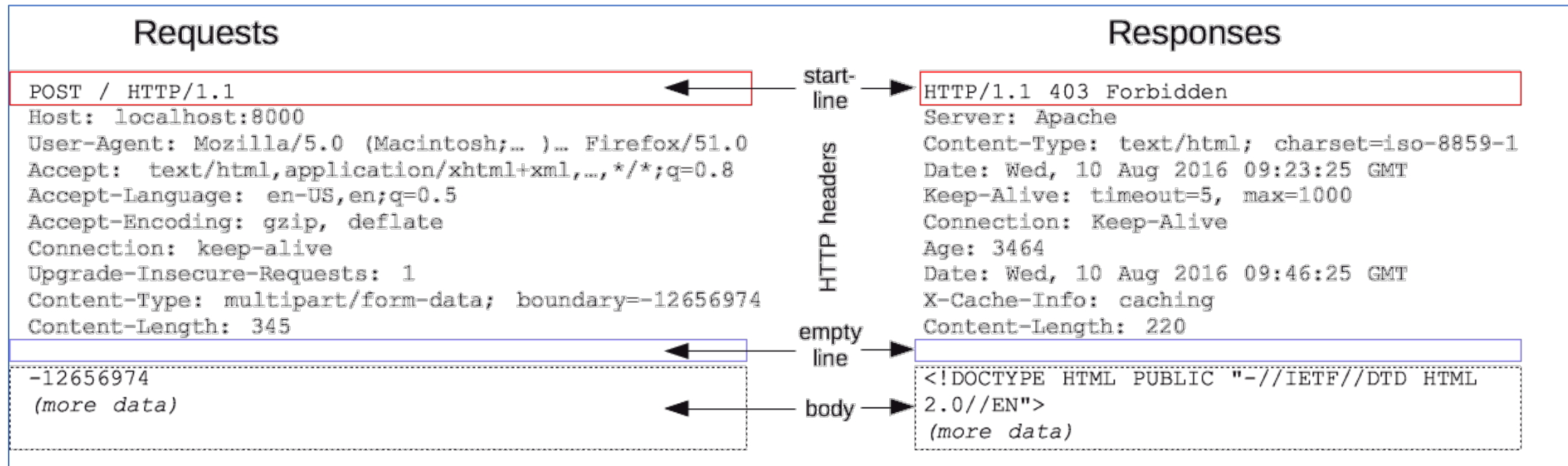


Introduction to React 02

React Back-End Integration

HTTP Protocol

- ส่วนประกอบของ HTTP Message
 - Header บอกรายละเอียดของ Message และ HTTP Method
 - Body ส่งและรับข้อมูลที่ต้องการระหว่าง Client และ Server



HTTP Method for REST API

- GET – ดึงหรืออ่านข้อมูลจากเซิร์ฟเวอร์
- POST – สร้างชุดข้อมูลใหม่บนเซิร์ฟเวอร์
- PUT – แก้ไขค่าของชุดข้อมูลบนเซิร์ฟเวอร์ (ข้อมูลทั้งหมด)
- PATCH – แก้ไขค่าของชุดข้อมูลบางส่วนบนเซิร์ฟเวอร์
- DELETE – ลบชุดข้อมูลบนเซิร์ฟเวอร์

HTTP Request

- การเชื่อมต่อระหว่าง React Application และ Web Application จะทำผ่านทาง HTTP Request
 - ข้อมูลที่รับและส่ง จะอยู่ในรูปแบบของ JSON (REST API)
- การทำ HTTP Request จาก React Application สามารถทำได้หลายช่องทางได้แก่
 - Fetch API เป็น Library มาตรฐานสำหรับส่ง HTTP Request
 - Axios เป็น 3rd party library สำหรับส่ง HTTP Request
- ในเอกสารนี้จะนำเสนอการใช้ Axios เนื่องจากมีความง่ายในการใช้งานมากกว่า

```
npm install axios --save
```

สร้าง BackEnd Server

- Clone project มาจาก <https://gitlab.psu.ac.th/suthon.s/strapi-account>
- หรือกดปุ่ม Download มาได้
- รัน back-end ด้วยคำสั่ง

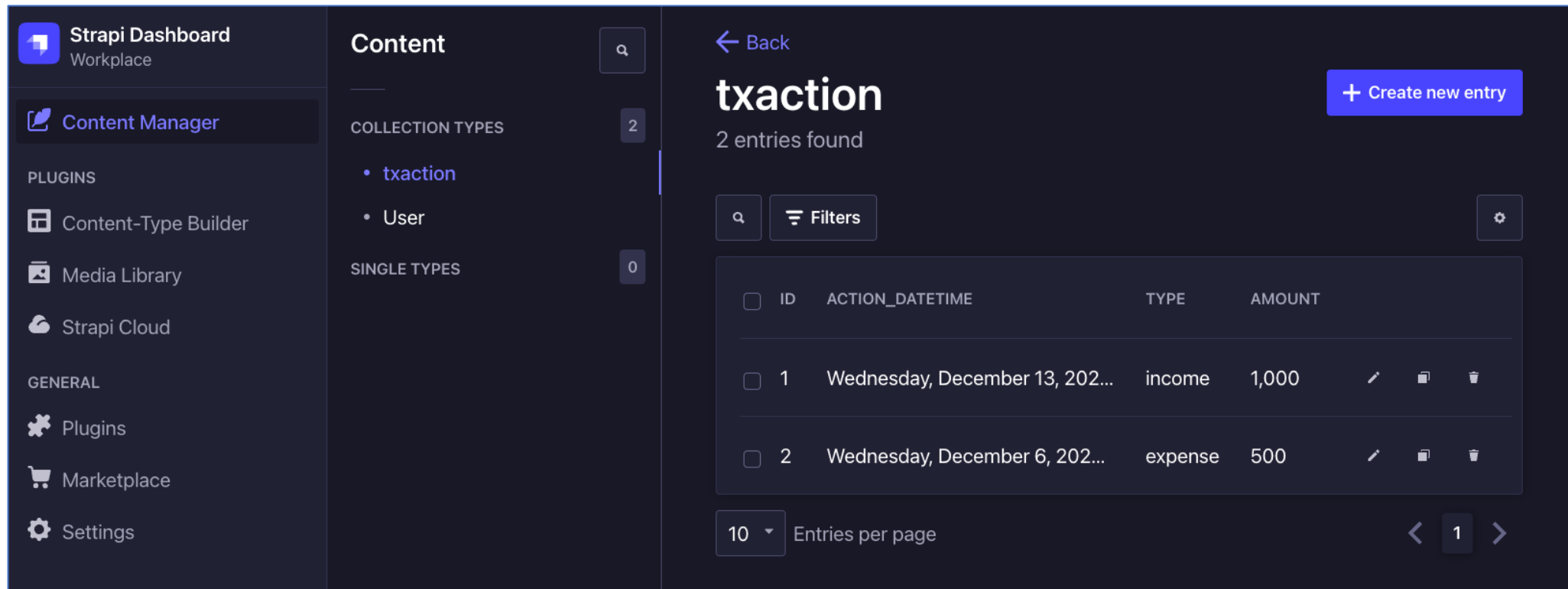
```
$cd strapi-account  
$npm install  
$npm run develop
```

- จากนั้นเข้าสู่หน้า Admin แล้วสร้าง account ใหม่ หรือ Login ด้วย

```
URL: http://localhost:1337/admin  
Username: admin@webdev.com  
Password: Wd123456
```

สร้าง Transaction ทดสอบ

- Content Manager > txaction > Create New entry



The screenshot displays the Strapi Dashboard interface. On the left, the 'Content Manager' sidebar is active, showing the 'txaction' collection type under 'COLLECTION TYPES'. The main panel shows the 'txaction' collection with 2 entries found. A table lists the entries with columns for ID, ACTION_DATETIME, TYPE, and AMOUNT. The first entry is an income of 1,000 on Wednesday, December 13, 2022. The second entry is an expense of 500 on Wednesday, December 6, 2022. A '+ Create new entry' button is visible in the top right corner of the main panel.

Strapi Dashboard
Workplace

Content Manager

PLUGINS

- Content-Type Builder
- Media Library
- Strapi Cloud

GENERAL

- Plugins
- Marketplace
- Settings

Content

COLLECTION TYPES 2







- txaction
- User

SINGLE TYPES 0

txaction
2 entries found

+ Create new entry

Filters

| <input type="checkbox"/> | ID | ACTION_DATETIME | TYPE | AMOUNT | |
|--------------------------|----|---------------------------------|---------|--------|---|
| <input type="checkbox"/> | 1 | Wednesday, December 13, 2022... | income | 1,000 |    |
| <input type="checkbox"/> | 2 | Wednesday, December 6, 2022... | expense | 500 |    |

10 Entries per page

< 1 >

ตั้งค่า

• Settings

ได้

Strapi Dash
Workplace

Content Man

PLUGINS

Content-Type

Media Library

Strapi Cloud

GENERAL

Plugins

Marketplace

Settings

Public

Default role given to unauthenticated user.

Role details

Name*

Public

Description*

Default role given to unauthenticated user.

Permissions

Only actions bound by a route are listed below.

Txaction

Define all allowed actions for the api::txaction plugin.

TXACTION

☒ create

☒ delete

☒ find

☒ findOne

☒ update

Select all

Advanced settings

Select the application's actions or the plugin's actions and click on the cog icon to display the bound route

Content types builder

Define all allowed actions for the plugin::content-type-

• Advanced settings

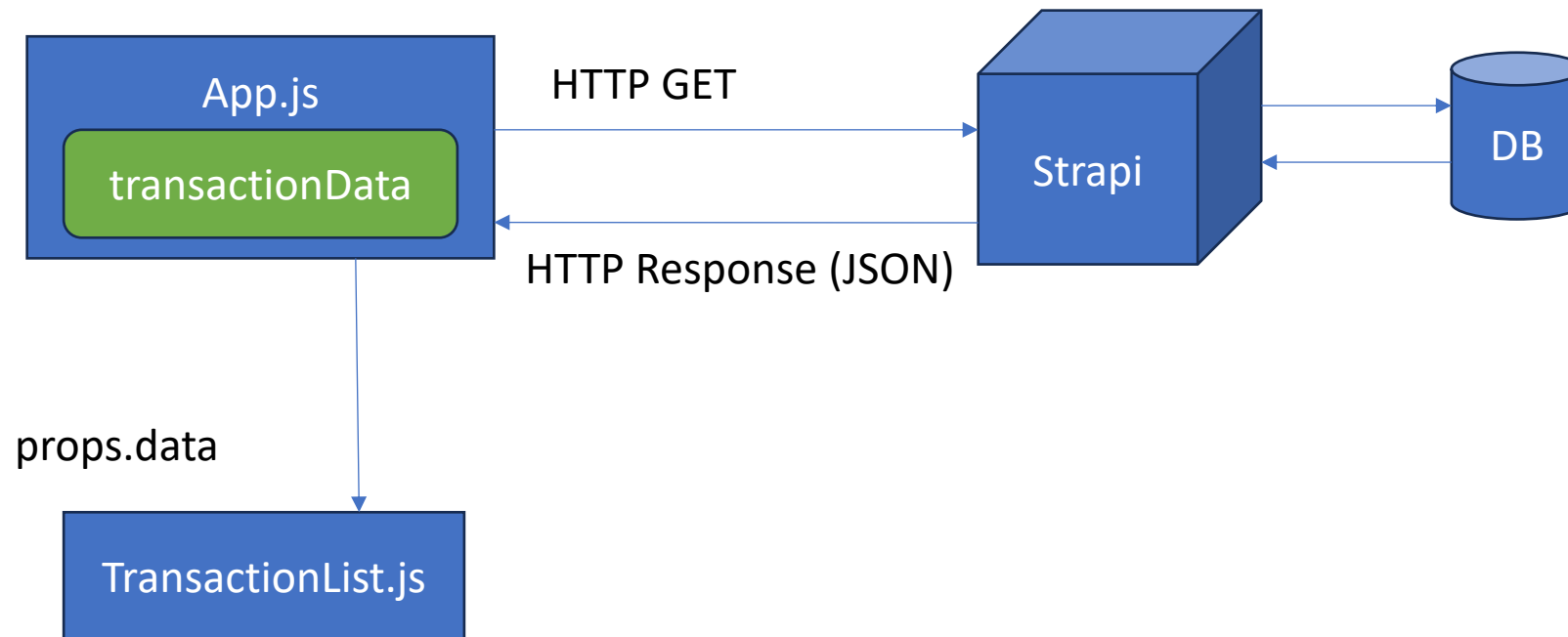
✓ Save

?

?

ทดสอบใช้งาน GET เพื่ออ่านค่า จาก Server

- ทำการ Modify TransactionList.js เพื่อดึงค่าจาก Server และแสดงผลในตาราง
 - ปรับ Code ของ App.js เพื่อให้ดึงข้อมูลจาก Server





จงแก้ไขไฟล์ App.js ให้ดึงข้อมูลจากเซิร์ฟเวอร์

src/App.js

```
import { useState, useEffect } from 'react';
import { Spin } from 'antd';
import axios from 'axios'

axios.defaults.baseURL = process.env.REACT_APP_BASE_URL || "http://localhost:1337"
const URL_TXACTIONS = '/api/txactions'

function App() {
  const [isLoading, setIsLoading] = useState(false)
  const [transactionData, setTransactionData] = useState([])

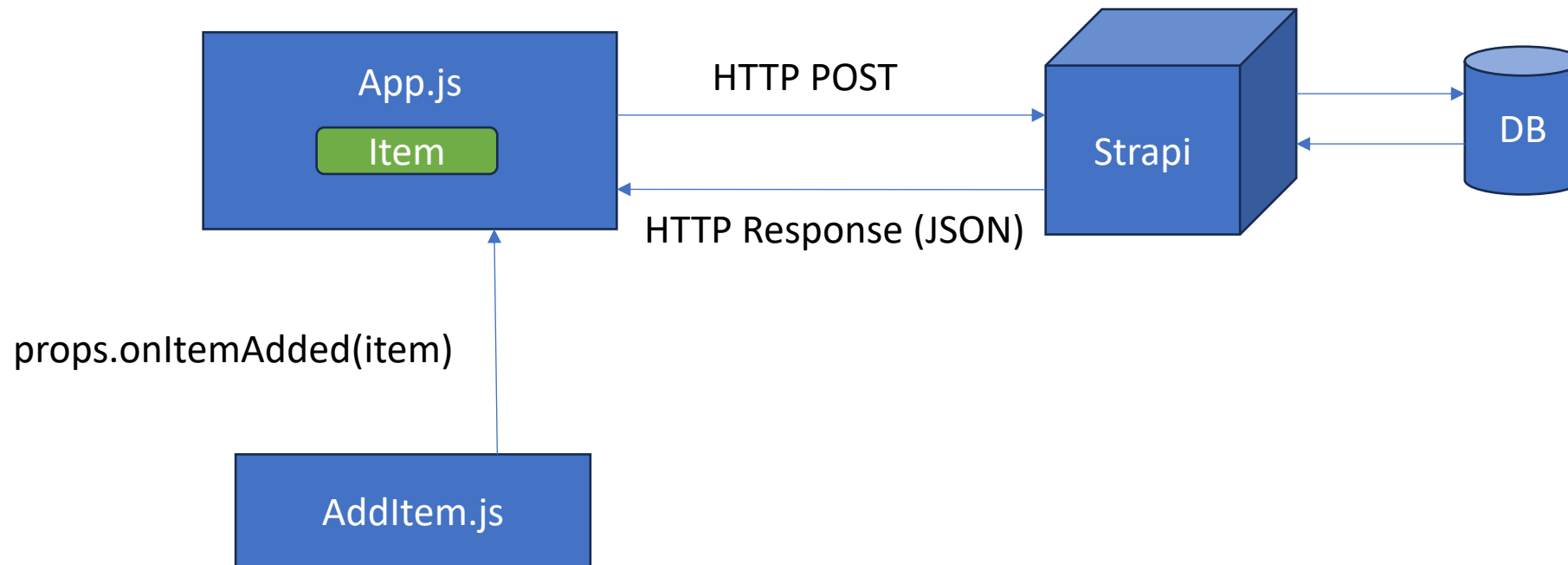
  const fetchItems = async () => {
    try {
      setIsLoading(true)
      const response = await axios.get(URL_TXACTIONS)
      //..... Add Code Here .....
    } catch (err) { console.log(err) }
    finally { setIsLoading(false) }
  }

  useEffect(() => {
    fetchItems()
  }, [])

  return (
    <div className="App">
      <header className="App-header">
        <Spin spinning={isLoading}>
          <TransactionList data={transactionData} />
        </Spin>
      </header>
    </div>
  );
}
```

ทดสอบใช้งาน POST เพื่อบันทึกค่าบน Server

- แก้ไข App.js เพื่อรับข้อมูลจาก AddItem.js แล้วนำไปสร้างข้อมูลบน Server





จงแก้ไขไฟล์ App.js เพื่อให้สามารถสร้างบันทึกรายรับ-รายจ่ายได้

src/App.js

```
import { Spin, Divider } from 'antd';
```

```
function App() {
```

```
  const addItem = async (item) => {
```

```
    try {
```

```
      setIsLoading(true)
```

```
      const params = //..... Add code here .....
```

```
      const response = await axios.post(URL_TXACTIONS, { data: params})
```

```
      const {id, attributes} = response.data.data
```

```
      setTransactionData(
```

```
        //..... Add code here .....
```

```
      )
```

```
    } catch (err) {
```

```
      console.log(err)
```

```
    } finally {
```

```
      setIsLoading(false)
```

```
    }
```

```
  }
```

```
  return (
```

```
    <div className="App">
```

```
      <header className="App-header">
```

```
        <Spin spinning={isLoading}>
```

```
          <AddItem onItemAdded={addItem} />
```

```
          <Divider>บันทึกรายรับ-รายจ่าย</Divider>
```

```
          <TransactionList
```

```
            data={transactionData} />
```

```
        </Spin>
```

```
      </header>
```

```
    </div>
```

```
  );
```



เพิ่ม จำนวนเงินปัจจุบัน

src/App.js

```
import { Spin, Divider, Typography } from 'antd';
```

```
function App() {  
  const [currentAmount, setCurrentAmount] = useState(0)  
  const [isLoading, setIsLoading] = useState(false)  
  const [transactionData, setTransactionData] = useState([])
```

```
  useEffect(() => {  
    //..... Add code here .....  
  }, [transactionData])
```

```
  return (  
    <div className="App">  
      <header className="App-header">  
        <Spin spinning={isLoading}>  
          <Typography.Title>  
            จำนวนเงินปัจจุบัน {currentAmount} บาท  
          </Typography.Title>  
          <AddItem onItemAdded={addItem} />  
          <Divider>บันทึกรายรับ-รายจ่าย</Divider>  
          <TransactionList  
            data={transactionData} />  
        </Spin>  
      </header>  
    </div>  
  );
```



จงแก้ไขไฟล์ App.js เพื่อให้ ลบ บันทึกทรายรับ-ทรายจ่ายได้

src/App.js

```
function App() {  
  const deleteItem = async (itemId) => {  
    try {  
      setIsLoading(true)  
      await axios.delete(..... Add Code Here .....)  
      fetchItems()  
    } catch (err) {  
      console.log(err)  
    } finally {  
      setIsLoading(false)  
    }  
  }  
}
```

Bound route
to api::txaction.txaction

DELETE /api/txactions/:id

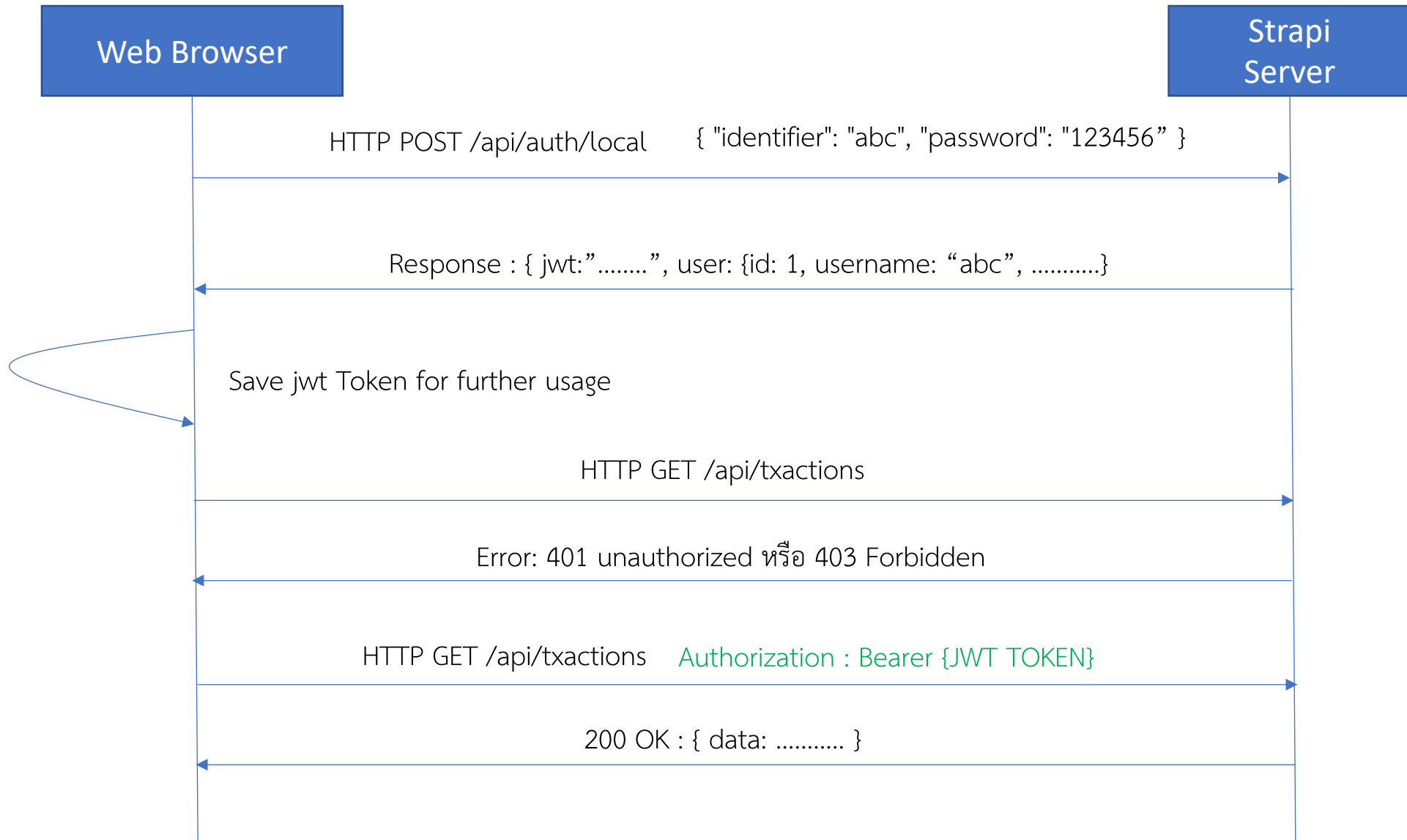
```
return (  
  <div className="App">  
    <header className="App-header">  
      <Spin spinning={isLoading}>  
        <AddItem onItemAdded={addItem} />  
        <Divider>บันทึกทรายรับ-ทรายจ่าย</Divider>  
        <TransactionList  
          data={transactionData}  
          onTransactionDeleted={deleteItem} />  
      </Spin>  
    </header>  
  </div>  
);
```

Authentication

Authentication

- การทำ Authentication ส่วนมากจะทำผ่าน API Token
- API Token จะถูกส่งไปยัง server ทุกครั้งเพื่อยืนยันว่า request ส่งมาจากผู้ใช้งานที่ลงทะเบียนแล้ว
- ข้อมูล Token จะถูกส่งผ่าน HTTP Header ชื่อ Authorization

ขั้นตอนการทำ Authentication



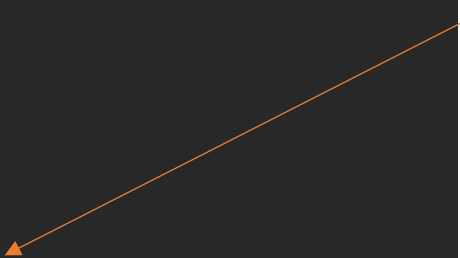
ตัวอย่าง Authorization Header

Request URL: http://localhost:1337/api/txactions
Request Method: POST
Status Code: ● 200 OK
Remote Address: 127.0.0.1:1337
Referrer Policy: strict-origin-when-cross-origin

► Response Headers (17)

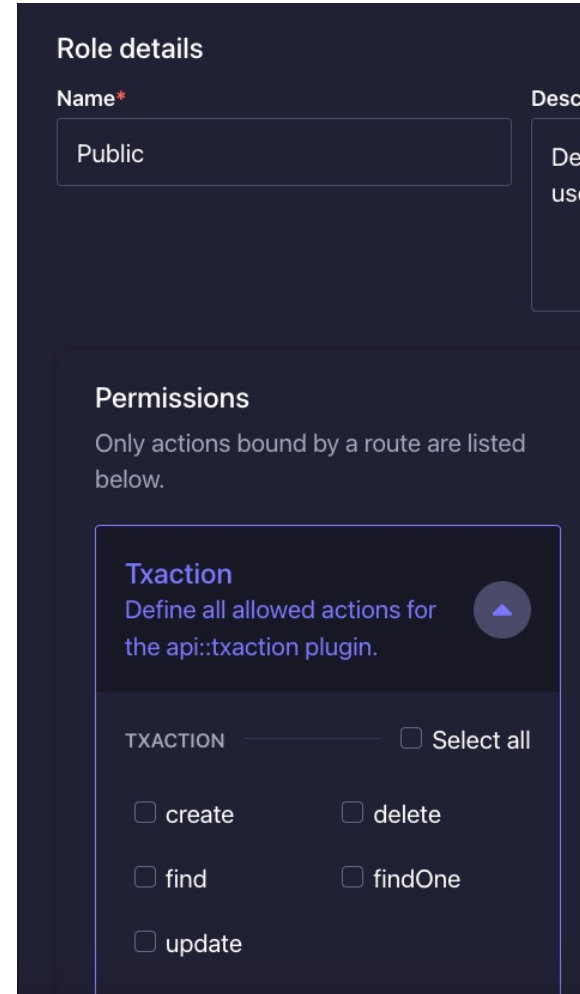
▼ Request Headers ☒ Raw

```
POST /api/txactions HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,th;q=0.8
Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNzAyNj
Connection: keep-alive
Content-Length: 107
Content-Type: application/json
Host: localhost:1337
```



ตั้งค่าสิทธิ์ใน Strapi

- สิ่งที่ต้องการ
 - public ไม่สามารถเข้าถึง API ได้
 - Authenticated สามารถเข้าถึง API ได้ทุกตัว
- Settings > Roles (Users & Permission Plugin)



This screenshot shows the 'Role details' configuration for the 'Public' role. The 'Name' field is set to 'Public'. Under the 'Permissions' section, the 'Txaction' plugin is selected, but all individual permissions (create, delete, find, findOne, update) are unchecked. The 'Select all' checkbox is also unchecked.

Role details

Name* Public

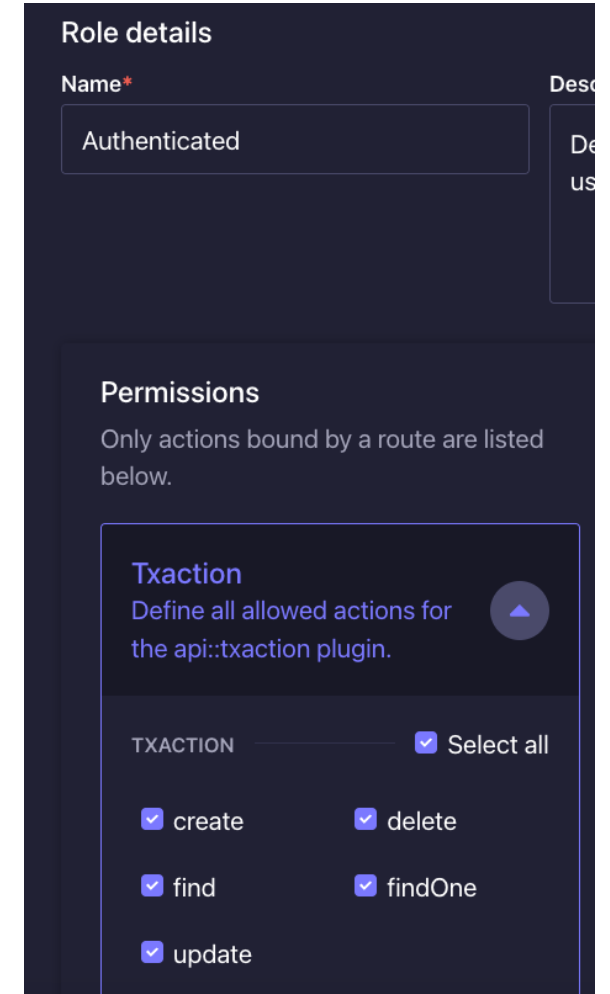
Permissions

Only actions bound by a route are listed below.

Txaction
Define all allowed actions for the api::txaction plugin.

TXACTION ☐ Select all

☐ create ☐ delete
☐ find ☐ findOne
☐ update



This screenshot shows the 'Role details' configuration for the 'Authenticated' role. The 'Name' field is set to 'Authenticated'. Under the 'Permissions' section, the 'Txaction' plugin is selected, and all individual permissions (create, delete, find, findOne, update) are checked. The 'Select all' checkbox is also checked.

Role details

Name* Authenticated

Permissions

Only actions bound by a route are listed below.

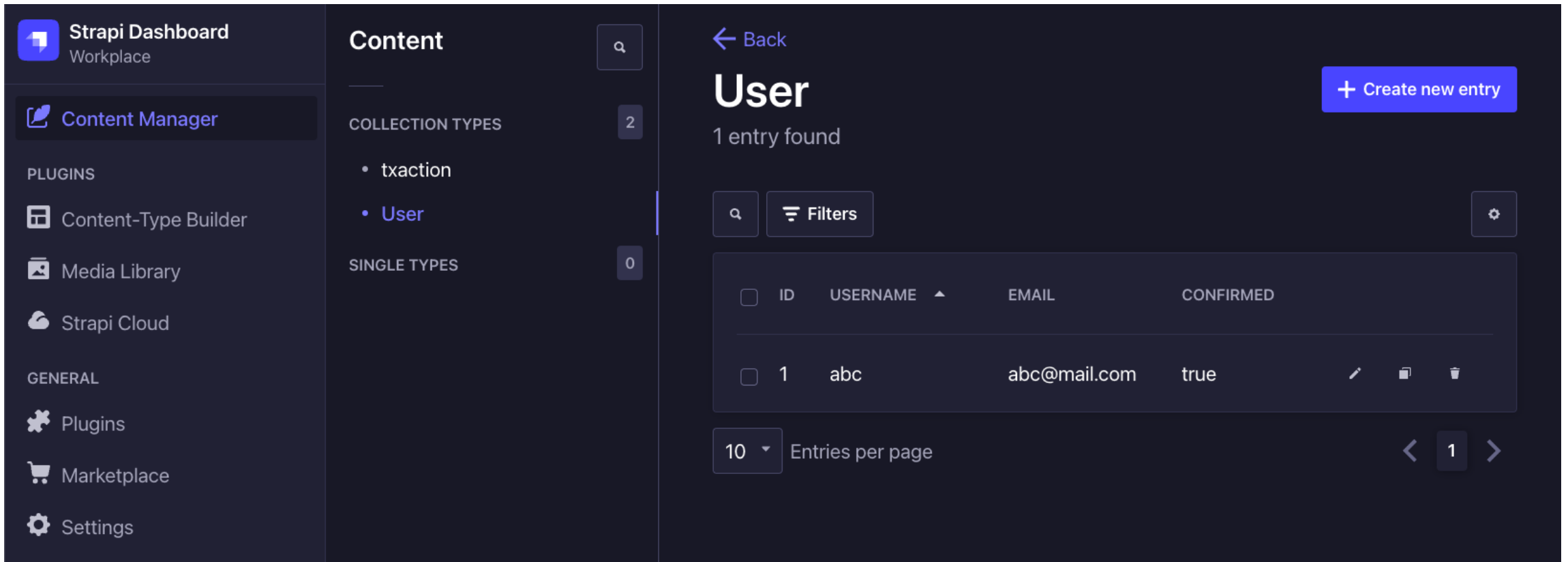
Txaction
Define all allowed actions for the api::txaction plugin.

TXACTION ☒ Select all

☒ create ☒ delete
☒ find ☒ findOne
☒ update

สร้าง User ใหม่ใน Strapi

- Content Manager > User > Create new entry



The screenshot displays the Strapi Dashboard interface. On the left, the sidebar includes the 'Strapi Dashboard' logo, 'Content Manager', 'PLUGINS' section with 'Content-Type Builder', 'Media Library', and 'Strapi Cloud', and a 'GENERAL' section with 'Plugins', 'Marketplace', and 'Settings'. The main area is titled 'Content' and shows 'COLLECTION TYPES' with 'txaction' and 'User' (selected), and 'SINGLE TYPES' with '0' items. The 'User' collection type is expanded, showing '1 entry found'. A table lists the entry with columns: ID, USERNAME, EMAIL, and CONFIRMED. The entry has ID 1, USERNAME abc, EMAIL abc@mail.com, and CONFIRMED true. At the bottom, there is a pagination control showing '10' entries per page and a single page indicator.

Strapi Dashboard
Workplace

Content Manager

PLUGINS

- Content-Type Builder
- Media Library
- Strapi Cloud

GENERAL

- Plugins
- Marketplace
- Settings

Content

COLLECTION TYPES

- txaction
- User

SINGLE TYPES

0

User

1 entry found

+ Create new entry

| ID | USERNAME | EMAIL | CONFIRMED |
|----|----------|--------------|-----------|
| 1 | abc | abc@mail.com | true |

10 Entries per page

1

ทดสอบ Code เดิม

จำนวนเงินปัจจุบัน 0 บาท


* ชนิด:

* จำนวนเงิน:

* หมายเหตุ:

Add

บันทึกรายรับ-รายจ่าย

| Date-Time | Type | Amount | Note | Action |
|--|------|--------|------|--------|
| <div> No data</div> | | | | |

Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>

✖ ▶ GET <http://localhost:1337/api/txactions> 403 (Forbidden)

▶ `AxiosError {message: 'Request failed with status code 403', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {...}, request: XMLHttpRequest, ...}`

✖ ▶ GET <http://localhost:1337/api/txactions> 403 (Forbidden)

▶ `AxiosError {message: 'Request failed with status code 403', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {...}, request: XMLHttpRequest, ...}`

>



Refactor Code โดยการแยกหน้า Login และ แสดงผล

- สร้างไฟล์ใหม่ชื่อ FinanceScreen.js
 - ย้าย Code ทั้งหมดจาก App.js ไปใส่ใน FinanceScreen.js
 - เปลี่ยนชื่อ function App() เป็น function **FinanceScreen()**
 - เปลี่ยน export default **FinanceScreen**; ที่ด้านล่างสุดของไฟล์

- แก้ไขไฟล์ App.js

src/App.js

```
import './App.css';
import axios from 'axios'

axios.defaults.baseURL = process.env.REACT_APP_BASE_URL || "http://localhost:1337"

function App() {
  return (
    <div>This is App.js</div>
  );
}

export default App;
```



สร้างไฟล์ LoginScreen.js

src/LoginScreen.js

```
import { useState } from 'react';
import { Button, Form, Input, Alert } from 'antd';
import axios from 'axios'

const URL_AUTH = "/api/auth/local"

export default function LoginScreen(props) {

  const [isLoading, setIsLoading] = useState(false)
  const [errMsg, setErrMsg] = useState(null)

  const handleLogin = async (formData) => {
    try{
      setIsLoading(true)
      setErrMsg(null)
      const response = await axios.post(..... Code Here ..... )
      const token = //..... Code Here .....
      axios.defaults.headers.common = { 'Authorization': `bearer ${token}` }
      props.onLoginSuccess();
    } catch(err) {
      console.log(err)
      setErrMsg(err.message)
    } finally { setIsLoading(false) }
  }
}
```

ต่อหน้าถัดไป



สร้างไฟล์ LoginScreen.js (ต่อ)

src/LoginScreen.js

```
return(  
  <Form  
    onFinish={handleLogin}  
    autoComplete="off">  
    {errMsg &&  
      <Form.Item>  
        <Alert message={errMsg} type="error" />  
      </Form.Item>  
    }  
  
    <Form.Item  
      label="Username"  
      name="identifier"  
      rules={[{required: true,}]}>  
      <Input />  
    </Form.Item>  
  
    <Form.Item  
      label="Password"  
      name="password"  
      rules={[{required: true,}]}>  
      <Input.Password />  
    </Form.Item>
```

```
      <Form.Item>  
        <Button  
          type="primary"  
          htmlType="submit" loading={isLoading}>  
          Submit  
        </Button>  
      </Form.Item>  
    </Form>  
  )  
}
```




แก้ไขไฟล์ App.js

src/App.js

```
import './App.css';
import axios from 'axios'
import LoginScreen from './LoginScreen';

axios.defaults.baseURL = process.env.REACT_APP_BASE_URL || "http://localhost:1337"

function App() {
  const handleLoginSuccess = () => {console.log("Login Success!!!")}
  return (
    <div className="App">
      <header className="App-header">
        <LoginScreen onLoginSuccess={handleLoginSuccess}/>
      </header>
    </div>
  );
}

export default App;
```



ปรับไฟล์ App.js ให้แสดงผล FinanceScreen.js

src/App.js

```
import './App.css';
import axios from 'axios'
import { useState } from 'react';
import LoginScreen from './LoginScreen';
import FinanceScreen from './FinanceScreen';

function App() {
  const [isAuthenticated, setIsAuthenticated] = useState(false)

  const handleLoginSuccess = () => {
    //..... Code Here .....
  }

  return (
    <div className="App">
      <header className="App-header">
        //..... Code Here .....
        //..... Code Here .....
      </header>
    </div>
  );
}
```

การ Navigation แบบซับซ้อน

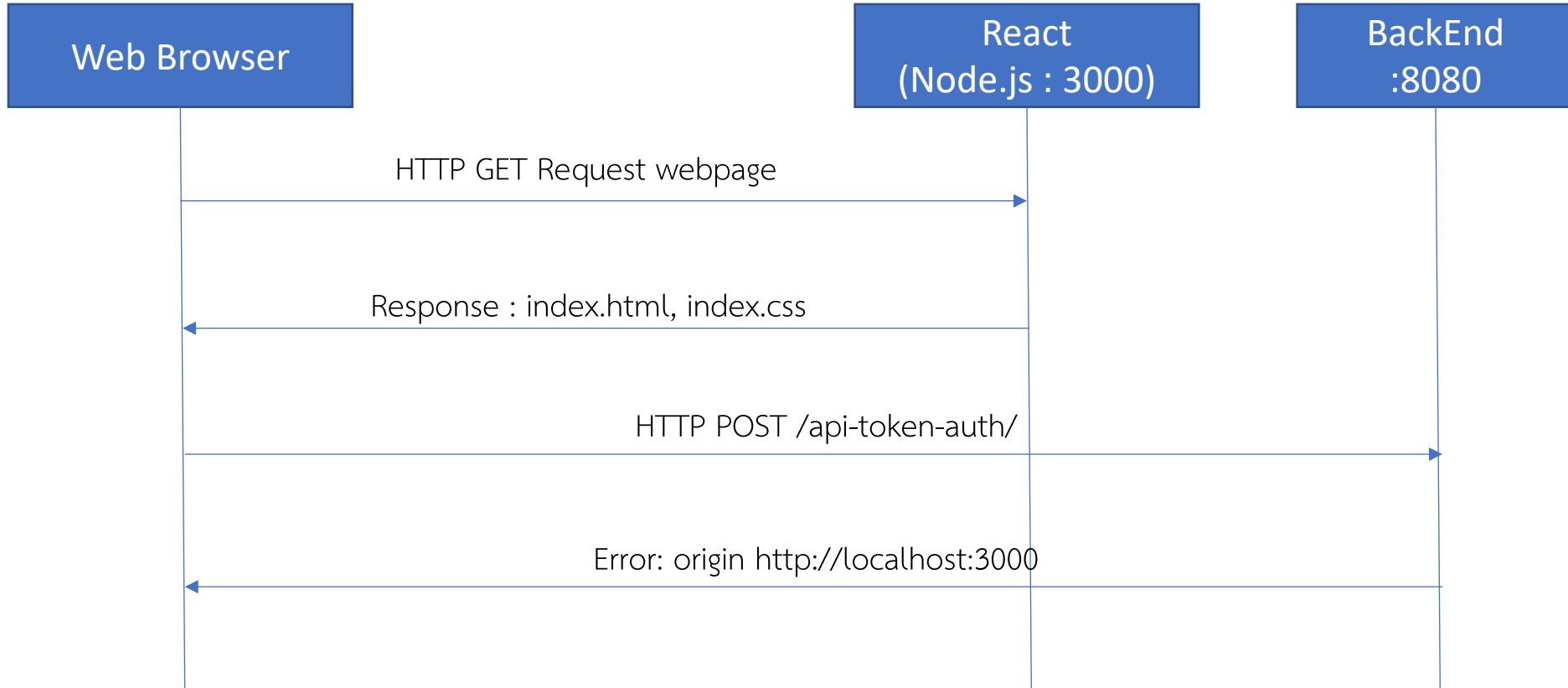
- กรณี Application มีขนาดใหญ่ขึ้น มีจำนวนหน้ามากขึ้น
- จำเป็นต้องใช้ Library อื่นๆ มาช่วย โดยตัวที่นิยมใช้กันคือ react-router-dom
 - มีความซับซ้อนสูง
 - สามารถแบ่งหน้า URL ที่ผ่านการ Authenticate แล้ว และหน้าอื่นๆ ได้
 - สามารถศึกษาได้ที่ <https://reactrouter.com/en/main>

การเก็บ Token

- กรณี Application ต้องการจดจำผู้ใช้ (Remember me)
- จำเป็นต้องมีการเก็บ Token ลงใน Persistent Storage โดยมีวิธีต่าง ๆ เช่น
 - Web Storage API (localStorage)
 - รองรับเก็บข้อมูลทั้งแบบ session และ persistent
 - Cookies (react-cookie, js-cookie)
 - มีความซับซ้อนสูงกว่าเนื่องจากใช้ backend ช่วยในการเก็บข้อมูล
 - มีความปลอดภัยสูงกว่า จึงเป็นทางเลือกในการเก็บ Token ใน Project ต่าง ๆ

Cross-Origin Resource Sharing (CORS)

```
✖ Access to XMLHttpRequest at 'http://localhost:8000/api-token-auth' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: Redirect is not allowed for a preflight request. localhost/:1
✖ ▶ POST http://localhost:8000/api-token-auth net::ERR_FAILED xhr.js:177
✖ ▶ Uncaught (in promise) Error: Network Error createError.js:16
    at createError (createError.js:16)
    at XMLHttpRequest.handleError (xhr.js:84)
```





Assignment : สร้างปุ่มสำหรับ Edit ข้อมูล

จำนวนเงินปัจจุบัน 15381 บาท

* ชนิด: * จำนวนเงิน: * เหตุผล:

บันทึกการรับ-การจ่าย

| Date-Time | Type | Amount | Note | Action |
|--------------------------|---------|--------|-------------|---|
| 2023-12-15T10:53:45.916Z | รายรับ | 15000 | เงินเดือน | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |
| 2023-12-15T10:58:01.337Z | รายรับ | 1000 | งานพิเศษ | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |
| 2023-12-15T11:03:52.855Z | รายจ่าย | 120 | อาหารเที่ยง | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |
| 2023-12-15T14:11:08.730Z | รายจ่าย | 499 | มือถือ | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |

< 1 >

Edit Transaction

* ชนิด:

* จำนวนเงิน:

* เหตุผล:

Date-Time

| | | | | |
|--------------------------|--------|-------|-----------|---|
| 2023-12-15T10:53:45.916Z | รายรับ | 15000 | เงินเดือน | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |
| 2023-12-15T10:58:01.337Z | รายรับ | 1000 | งานพิเศษ | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |



Assignment Guide (1)

- แก้ไข TransactionList.js
 - เพิ่มปุ่ม Edit ข้างๆ ปุ่ม Delete
 - แต่ callback function ต้องส่งตัวแปร record ไปทั้งหมด
 - ไม่เหมือนกับ Delete ที่ส่งไปแค่ record.id เพียงอย่างเดียว
 - ส่งไปเพื่อนำไปส่งต่อให้ component สำหรับ Edit ค่า



Assignment Guide (2)

- สร้าง Component ใหม่ ชื่อ EditItem.js
 - ภายใน Component นี้ประกอบไปด้วย [Modal](#) และ [Form](#)
 - Form สามารถลอกมาจาก AddItem.js ได้เกือบทั้งหมด
 - ใช้ useEffect แล้วตรวจสอบ props.isOpen จากนั้นใช้ form.setFieldsValue(props.item) เพื่อใส่ค่าให้กับ Form โดยอัตโนมัติ
- ผูก Event onOk ของ Modal ให้ไปเรียก function handleFormSubmit
 - สามารถเรียก form.validateFields().then(formData => { }) ภายในนั้นเพื่อ validate form และสั่งปิด Modal หรือสั่ง props.onItemEdited(formData) เพื่อใช้บอก App.js ให้ Update ค่า บน Server



Assignment Guide (3)

- แก้ไข FinanceScreen.js
 - สร้าง useState ตัวใหม่สำหรับเก็บข้อมูลของ Item ที่ต้องการ Edit
 - สามารถใช้ state ตัวนี้แทน flag สำหรับเปิด/ปิด Modal ได้ เช่น หาก item ตัวนี้เป็น null จะปิด modal และ หาก item ตัวนี้มีค่าเป็น object จะเปิด Modal ขึ้นมาแสดงผล
 - สร้าง function ใหม่ ชื่อ updateItem
 - สามารถส่ง update ได้ด้วย URL: /api/txactions/<id> และ ส่ง parameter เป็นค่าที่ถูก Edit ใหม่ในตัวแปร data เช่น
 - `await axios.put(`${URL_TXACTIONS}/${item.id}`, {data: item})`