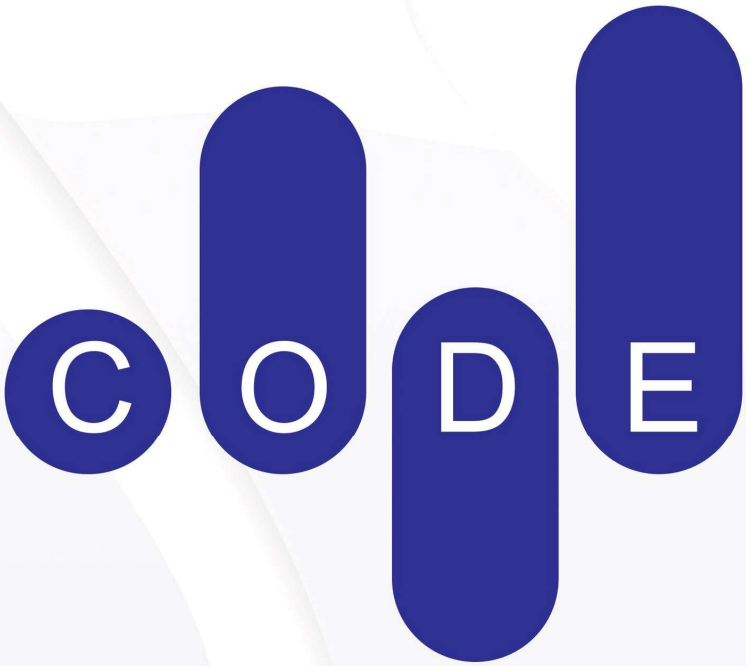
IOT Connect




# Coding Standard

Software Engineering
Burapha University

Document change control

table 0-1 Change document control

| Document name | | Coding standard Document | | |
|---|---|---|---|---|
| Current version | | 2.1.0 | | |
| Date last version modified | | 15 October 2566 | | |
| **Version** | **Date** | **The content/reasons of change** | **Author** | **Note** |
| 1.0.0 | 9 August 2566 | - create a template | Nattawut Pisit | - |
| 1.1.0 | 15 August 2566 | - Edit settings page<br>- Fixed typos<br>- Edit the details of each per page<br>- Edit table of contents<br>- Added control table Change document<br>- Add table of contents | Nattawut Pisit | - |
| 2.0.0 | 8 August 2566 | - Fixed typos<br>- Edit the details of each per page<br>- Edit table of contents<br>- Organize spacing | Nattawut Pisit | - |
| 2.1.0 | 15 October 2566 | - Database name changed<br>- Delete Json file comment<br>- Add UI Standards<br>- Add table of Pictures<br>- Add Sequelize Standards | Nattawut Pisit | - |

## Table of Contents

## รายการตารางประกอบ

## รายการภาพประกอบ

# บทนำ

ในการพัฒนาซอฟต์แวร์นั้นแต่ละคนนั้นก็มีการเขียนในรูปแบบที่ไม่เหมือนกัน ซึ่งซอฟต์แวร์นั้นจำเป็นจะต้องแก้ไขหรือปรับปรุงได้ง่าย จนอาจเป็นทำให้เกิดปัญหาได้ในภายหลังเมื่อสมาชิกคนอื่นในทีมต้องทำการแก้ไขหรือปรับปรุง เพราะฉะนั้นในการทำงานเป็นทีมทุก ๆ คน จึงจำเป็นต้องให้ทุกคนมีความเข้าใจที่ตรงกัน ซึ่งเอกสารนี้เป็นการทำข้อตกลงในการเขียนโปรแกรมร่วมกัน ตั้งแต่การกำหนดตัวแปร การตั้งชื่อฟังก์ชันต่าง ๆ การตั้งชื่อไฟล์ รวมไปถึงการเขียนคอมเมนต์

ทางทีมผู้พัฒนาจึงจัดทำเอกสารมาตรฐานการเขียนโปรแกรมขึ้นมา เพื่อความเป็นระเบียบช่วยลดโอกาสในการเกิดปัญหาและข้อผิดพลาดต่าง ๆ ให้เหลือน้อยที่สุด

## ส่วนที่ 1 สมาชิกภายในทีม

โครงการพัฒนาระบบ IoT Connect พัฒนามาตรฐานการเขียนโปรแกรมขึ้นมา โดยมีวัตถุประสงค์ เพื่อสร้างคำแนะนำและมาตรฐานสำหรับการเขียนโปรแกรมภายในทีม ช่วยให้สมาชิกในทีมมีความเข้าใจที่ตรงกันในการเขียนโปรแกรม ลดข้อผิดพลาดในการเขียนโปรแกรม ลดการใช้ทรัพยากรในการแก้ไขข้อผิดพลาดในโปรแกรม เพิ่มประสิทธิภาพในการพัฒนาโครงการ IoT Connect ประหยัดเวลาในการพัฒนาโครงการ IoT Connect

ตาราง 1-1 สมาชิกในทีม

| ลำดับ | รหัสนิสิต | ชื่อ-สกุล | ตำแหน่ง |
|---|---|---|---|
| 1 | 64160280 | นายพชร อุ่นกิตติ | Team Leader |
| 2 | 64160156 | นายกลวัชร เปรมจิตต์ | Plan Manager |
| 3 | 64160066 | นายณัฐวุฒิ สมดุลยกนก | Development Manager |
| 4 | 64160165 | นางสาวเบญจมาภรณ์ วงศ์วิริยะ | Quality Manager |
| 5 | 64160154 | นายกรชนก รักษาภัย | Support Manager |
| 6 | 64160293 | นายอวยชัย แซ่พ่าน | Development Engineer |
| 7 | 64160168 | นายพิสิษฐ์ นามศิริ | Development Engineer |
| 8 | 64160073 | นางสาววิไลวรรณ รุณบุตร | Quality Engineer |
| 9 | 64160068 | นายพชรพล ผลประเสริฐ | Support Engineer |

## ส่วนที่ 2 มาตรฐานการเขียนโปรแกรม (CODING STANDARD)

มาตรฐานการเขียนโปรแกรมนี้เป็นมาตรฐานที่กำหนดขึ้นในบริบทของการพัฒนาระบบ โดยใช้ Node.js Express.js และ React.js ซึ่งสามารถประยุกต์นำมาสร้างสถาปัตยกรรมซอฟต์แวร์แบบ Three-Tier Architecture (TTA) เป็นการออกแบบสถาปัตยกรรมของระบบ (System Architecture) โดยมีแนวคิดพื้นฐาน คือการแบ่งแยกหน้าที่ความรับผิดชอบของแต่ละ Tier ประกอบด้วย Presentation Tier รับผิดชอบในการแสดงผลด้าน UI Business Logic Tier รับผิดชอบในการประมวลผลด้าน Business Logic และ Data Tier รับผิดชอบในส่วนการจัดการฐานข้อมูล โดยส่วนที่มีการกำหนดมาตรฐานนั้น ประกอบไปด้วยไฟล์และโฟลเดอร์ที่ใช้ ดังนี้

IOT_CONNECT_SERVICE คือ โฟลเดอร์ Project ส่วนของ Business Logic Tier และ Data Tier

- .env คือ ไฟล์สำหรับเก็บค่า Config ต่าง ๆ
- app.js คือ ไฟล์สำหรับกำหนดเส้นทางของ Routes
- routes คือ โฟลเดอร์สำหรับเก็บ Route ที่สร้างไว้ ยกตัวอย่างเช่น dashboard.js เก็บ Route ที่จะถูกเรียกใช้งานในหน้า Dashboard ผ่าน app.js ฯลฯ
- service คือ โฟลเดอร์สำหรับเก็บ Service ที่สร้างไว้ ยกตัวอย่างเช่น dashboardService.js เก็บ Service ต่าง ๆ ที่เกี่ยวข้องกับหน้า Dashboard โดยจะถูกเรียกใช้งานผ่าน Function ในไฟล์ Route ฯลฯ
- query คือ โฟลเดอร์สำหรับเก็บ Query string ในรูปแบบของ ไฟล์ json ยกตัวอย่างเช่น queryDashboard.json เก็บ Query string ต่าง ๆ ที่เกี่ยวข้องกับการ Query database โดยจะถูกเรียกใช้งานผ่าน Function ในไฟล์ Service ฯลฯ
- models คือ โฟลเดอร์สำหรับเก็บ ไฟล์ Sequelize ที่สร้างไว้ ยกตัวอย่างเช่น ic_monitors.js เก็บการ define ของตาราง ic_monitors ใน Database ฯลฯ

IOT_CONNECT_CLIENT คือ โฟลเดอร์ Project ส่วนของ Presentation Tier

- Index.js คือ ไฟล์ที่ทำให้ Component หลักของแอปพลิเคชันถูกเรียกใช้และแสดงผลบนหน้าเว็บ
- App.js คือ ไฟล์ที่คอยจัดการ Route เพื่อให้เรียกใช้ Component ได้ถูกต้อง
- components คือ โฟลเดอร์สำหรับเก็บ Component ที่สร้างไว้ ยกตัวอย่างเช่น energyChart.js ที่จะเก็บ Component ที่ใช้งาน
- view คือ โฟลเดอร์สำหรับเก็บหน้าเว็บที่สร้างไว้ ยกตัวอย่างเช่น landingPage.js ที่จะเก็บ ไฟล์ที่รวบรวม Component จาก Components เพื่อมาเรียกใช้ เมื่อเข้า Route ที่ได้ทำการติดต่อเอาไว้
- css คือ โฟลเดอร์สำหรับเก็บไฟล์ CSS ที่จะนำมาใช้ในไฟล์ Component ต่าง ๆ

ดังนั้นจึงมีการกำหนดมาตรฐานการเขียนโปรแกรมแบ่งตามหัวข้อเรื่อง รวมถึง TTA และมีมาตรฐานเกี่ยวกับข้อมูล ดังนี้

**1. การตั้งชื่อไฟล์**

**1.1 การตั้งชื่อไฟล์ Route (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อไฟล์ Route

1. ใช้รูปแบบ camelCase สำหรับชื่อไฟล์ เช่น dashboard.js ฯลฯ
2. หากชื่อไฟล์มีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อให้สื่อความหมาย

**1.2 การตั้งชื่อไฟล์ Service (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อไฟล์ Service

1. ใช้รูปแบบ camelCase สำหรับชื่อไฟล์ เช่น dashboardService.js ฯลฯ
2. หากชื่อไฟล์มีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อให้สื่อความหมาย

**1.3 การตั้งชื่อไฟล์ Query (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อไฟล์ Query

1. ใช้รูปแบบ camelCase สำหรับชื่อไฟล์ เช่น queryDashboard.js ฯลฯ
2. หากชื่อไฟล์มีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
4. ต้องขึ้นต้นด้วยคำว่า query เสมอ
5. ตั้งชื่อให้สื่อความหมาย

**1.4 การตั้งชื่อไฟล์ Sequelize (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อไฟล์ Sequelize

1. ใช้รูปแบบ snake_case สำหรับชื่อไฟล์ เช่น ic_monitors ฯลฯ
2. ตั้งชื่อให้เหมือนตารางในฐานของมูล เช่น ic_transactions ฯลฯ

**1.5 การตั้งชื่อไฟล์ CSS (IOT_CONNECT_CLIENT)**

หลักการตั้งชื่อไฟล์ CSS

1. ใช้รูปแบบ camelCase สำหรับชื่อไฟล์ เช่น landingPage.css ฯลฯ
2. หากชื่อไฟล์มีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อให้สื่อความหมาย

**1.6 การตั้งชื่อไฟล์ View (IOT_CONNECT_CLIENT)**

หลักการตั้งชื่อไฟล์ View

1. ใช้รูปแบบ camelCase สำหรับชื่อไฟล์ เช่น landingPage.js ฯลฯ
2. หากชื่อไฟล์มีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อไฟล์ให้สอดคล้องกับ Route

**2. การตั้งชื่อฟังก์ชัน**

**2.1 การตั้งชื่อฟังก์ชันใน Route (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อฟังก์ชันที่เป็น path

1. ใช้รูปแบบ camelCase สำหรับชื่อฟังก์ชัน เช่น /getMonitorHistoryById ฯลฯ
2. หากชื่อฟังก์ชันมีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อฟังก์ชันให้สื่อความหมาย

**2.2 การตั้งชื่อฟังก์ชันใน Service (IOT_CONNECT_SERVICE)**

หลักการตั้งชื่อฟังก์ชัน

1. ใช้รูปแบบ camelCase สำหรับชื่อฟังก์ชัน เช่น addMonitor ฯลฯ
2. หากชื่อฟังก์ชันมีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อฟังก์ชันให้สื่อความหมาย

**2.3 การตั้งชื่อฟังก์ชันใน Components (IOT_CONNECT_CLIENT)**

หลักการตั้งชื่อฟังก์ชัน

1. ใช้รูปแบบ camelCase สำหรับชื่อฟังก์ชัน เช่น energyChart ฯลฯ
2. หากชื่อฟังก์ชันมีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อฟังก์ชันให้ตรงกับชื่อไฟล์ Component ที่สร้าง

**3. การนำเข้าโมดูล**

**3.1 การนำเข้าโมดูล (IOT_CONNECT_SERVICE)**

1. ใช้รูปแบบ camelCase สำหรับชื่อ
2. หากชื่อฟังก์ชันมีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ตั้งชื่อโมดูลให้สื่อความหมาย เช่น companyService ฯลฯ

**3.2 การสร้าง Instance (IOT_CONNECT_SERVICE)**

1. ใช้รูปแบบ camelCase สำหรับชื่อ

2. หากชื่อฟังก์ชันมีมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น
3. ใช้เครื่องหมายขีดล่าง (_) ก่อนใส่ชื่อที่ตั้งให้กับโมดูลไว้ เช่น _companyService ฯลฯ
4. ตั้งชื่อให้สื่อความหมาย

**4. การตั้งชื่อตัวแปร**

**4.1 ตัวแปรทั่วไป**

1. ขึ้นต้นด้วยตัวอักษรพิมพ์เล็ก เช่น monitor
2. หากมีคำมากกว่า 1 คำ ให้ใช้ตัวอักษรภาษาอังกฤษตัวใหญ่ขึ้นต้น เช่น allMonitors
3. ตั้งชื่อให้สื่อความหมาย

**4.2 ตัวแปรนับรอบ**

1. ให้ใช้คำที่เกี่ยวข้องกับการนับ เช่น round index
2. ห้ามใช้ตัวย่อหรืออักษรตัวเดียว เช่น i j หรือ k
3. ตั้งชื่อให้สื่อความหมาย

**5. การจัดทำมาตรฐานเกี่ยวกับฐานข้อมูล**

**5.1 การตั้งชื่อฐานข้อมูล**

ให้ตั้งชื่อฐานข้อมูลเป็นตัวอักษรพิมพ์เล็กทั้งหมดตามด้วยเครื่องหมายขีดล่าง (_) ยกตัวอย่างเช่น 66M_IOTConnect ฯลฯ

**5.2 การตั้งชื่อตาราง**

ให้ตั้งชื่อตารางเป็นตัวอักษรพิมพ์เล็กทั้งหมด โดยชื่อตารางขึ้นต้นด้วยตัวย่อชื่อของระบบ และให้ใช้ขีดล่าง (_) คั่นระหว่างชื่อตารางแต่ละคำให้ตั้งชื่อตารางโดยใช้ชื่อข้อมูลเป็นพหูพจน์ ยกตัวอย่างเช่น ic_monitors ฯลฯ

**5.3 การตั้งชื่อฟิลด์**

ให้ตั้งชื่อฟิลด์โดยใช้ตัวอักษรภาษาอังกฤษพิมพ์เล็กทั้งหมด โดยชื่อฟิลด์ขึ้นต้นด้วยตัวย่อของชื่อตาราง โดยตัวย่อต้องเป็นตัวอักษรตัวแรกของแต่ละคำของชื่อตารางข้อมูลคั่นด้วยเครื่องหมายขีดล่าง (_) ตามด้วยชื่อฟิลด์ ยกตัวอย่างเช่น ims_name ims มาจาก ic_monitors ฯลฯ

**5.4 การเขียนคอมเมนต์ของตารางและฟิลด์**

ทุกตารางและทุกฟิลด์ต้องมีการคอมเมนต์หรือนิยามความหมายกำกับไว้ให้ครบถ้วน ไม่มีข้อยกเว้น

หลักการเขียนคอมเมนต์

1. ตาราง ให้นิยามความหมายว่า ตารางเก็บข้อมูลอะไรหรือใช้สำหรับทำอะไร ตัวอย่างเช่น ตาราง ic_monitors คือ ตารางเก็บข้อมูล monitor ทั้งหมด เป็นต้น

2. ฟิลด์ ให้นิยามความหมายว่า ใช้เก็บข้อมูลอะไร ตัวอย่างเช่น ims_id คือ รหัสประจำ monitor ims_name คือ ชื่อ monitor ims_in_use คือ สถานะการถูกใช้งาน เป็นต้น

3. ฟิลด์ที่อ้างอิงจากฟิลด์อื่น (FK) ให้อธิบายความหมายเดียวกันกับตารางตั้งต้น (PK) และต่อท้ายด้วยว่ามาจากตรงไหนหรือฐานข้อมูลไหน (ระบุชื่อฐานข้อมูลด้วย หากอยู่คนละฐานข้อมูล) ยกตัวอย่างเช่น ihs_monitor_id คือ รหัสประจำ monitor (ตาราง ic_histories) เป็นต้น

**6. การตั้งชื่อตัวแปรของ Config (.env)**

ข้อบังคับ

1. ต้องเป็นตัวพิมพ์ใหญ่ทั้งหมด
2. ใช้รูปแบบ snake_case ต้องขั้นด้วยเครื่องหมายขีดล่าง (_)

หลักการตั้งชื่อ

1. ใช้ชื่อย่อของหัวข้อหลัก ตามด้วยเครื่องหมายขีดล่าง (_)
2. ต่อท้ายด้วยชื่อของข้อมูลที่ต้องการ ยกตัวอย่างเช่น DB_USERNAME และ DB_PASSWORD ฯลฯ

ข้อห้าม

1. ห้ามตั้งชื่อ Config ซ้ำกับชื่อที่มีอยู่แล้ว
2. ห้ามแก้ไขหรือลบ Config โดยพลการต้องทำการปรึกษากับหัวหน้าทีมพัฒนาและสมาชิกนักพัฒนาคนอื่นก่อนเสมอ

**7. การเขียนคอมเมนต์**

**7.1 คอมเมนต์ของไฟล์ Route (IOT_CONNECT_SERVICE)**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. เขียนคอมเมนต์ไฟล์ Route ไว้บรรทัดแรกของไฟล์
3. เขียนคอมเมนต์ไฟล์ Route ด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของไฟล์ Route

ไฟล์ Route ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อไฟล์ Route
2. คำอธิบายรายละเอียดไฟล์ Route
3. ชื่อผู้เขียน/แก้ไข

4. วันที่จัดทำ/แก้ไข

หมายเหตุ

แต่ละบรรทัดให้ใส่เครื่องหมาย * และเว้นวรรค 1 ครั้งก่อนเขียนข้อความเสมอ

(ยกเว้นบรรทัดที่ 1 และบรรทัดที่ 6)

ตัวอย่างการคอมเมนต์ไฟล์ Route

```
/**
 * ชื่อไฟล์: dashboard.js
 * คำอธิบาย: สำหรับเก็บ Route ที่สร้างไว้สำหรับใช้กับหน้า Dashboard โดยจะเก็บเป็น Path
 *                กลางสำหรับเรียกใช้งาน Service ต่าง ๆ
 * ชื่อผู้เขียน/แก้ไข: John Doe
 * วันที่จัดทำ/แก้ไข: 1 กุมภาพันธ์ 2566
 */
```

**7.2 คอมเมนต์ฟังก์ชันของ Route (IOT_CONNECT_SERVICE)**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. ให้เขียนคอมเมนต์ฟังก์ชันกำกับทุกฟังก์ชัน ไม่มีข้อยกเว้น
3. เขียนคอมเมนต์ฟังก์ชันไว้ก่อนเริ่มลงมือเขียนฟังก์ชันนั้น ๆ
4. เขียนคอมเมนต์ฟังก์ชันด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของฟังก์ชันของ Route

ฟังก์ชันของ Route ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อฟังก์ชัน (Path)
2. Http request
3. คำอธิบายฟังก์ชัน
4. Input
5. Output
6. ชื่อผู้เขียน/แก้ไข
7. วันที่จัดทำ/แก้ไข

หมายเหตุ

แต่ละบรรทัดให้ใส่เครื่องหมาย * และเว้นวรรค 1 ครั้งก่อนเขียนข้อความเสมอ

(ยกเว้นบรรทัดที่ 1 และบรรทัดที่ 9)

ตัวอย่างการคอมเมนต์ฟังก์ชันของ Route

```
/**
 * ชื่อฟังก์ชัน: getAllMoniter
 * Http request: get
 * คำอธิบาย: ฟังก์ชันนี้ใช้สำหรับดึงข้อมูล Monitor ทั้งหมดจากฐานข้อมูล
 * Input: ไม่มี
 * Output: ข้อมูล Monitor ทั้งหมด (Map)
 * ชื่อผู้เขียน/แก้ไข: John Doe
 * วันที่จัดทำ/แก้ไข: 1 กุมภาพันธ์ 2566
 */
Router.get('/getAllMoniter', function (req, res, next) {
      .....
});
```

### 7.3 คอมเมนต์ของไฟล์ Service (IOT_CONNECT_SERVICE)

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. เขียนคอมเมนต์ไฟล์ Service ไว้บรรทัดแรกของไฟล์
3. เขียนคอมเมนต์ไฟล์ Service ด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของไฟล์ Service

ไฟล์ Service ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อไฟล์ Service
2. คำอธิบายราบละเอียดไฟล์ Service
3. ชื่อผู้เขียน/แก้ไข
4. วันที่จัดทำ/แก้ไข

หมายเหตุ

แต่ละบรรทัดให้ใส่เครื่องหมาย * และเว้นวรรค 1 ครั้งก่อนเขียนข้อความเสมอ (ยกเว้นบรรทัดที่ 1 และบรรทัดที่ 6)

ตัวอย่างการคอมเมนต์ไฟล์ Service

```
/**
 * ชื่อไฟล์: dashboardService.js
```

* คำอธิบาย: สำหรับเก็บ Service ที่สร้างไว้สำหรับใช้ในหน้า Dashboard โดยเก็บวิธีการทำงานต่าง ๆ

* ชื่อผู้เขียน/แก้ไข: John Doe

* วันที่จัดทำ/แก้ไข: 1 กุมภาพันธ์ 2566

*/

**7.4 คอมเมนต์ฟังก์ชันของ Service (IOT_CONNECT_SERVICE)**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. ให้เขียนคอมเมนต์ฟังก์ชันกำกับทุกฟังก์ชัน ไม่มีข้อยกเว้น
3. เขียนคอมเมนต์ฟังก์ชันไว้ก่อนเริ่มลงมือเขียนฟังก์ชันนั้น ๆ
4. เขียนคอมเมนต์ฟังก์ชันด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของฟังก์ชันของ Service

ฟังก์ชันของ Service ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อฟังก์ชัน
2. คำอธิบายฟังก์ชัน
3. Input
4. Output
5. ชื่อผู้เขียน/แก้ไข
6. วันที่จัดทำ/แก้ไข

หมายเหตุ

แต่ละบรรทัดให้ใส่เครื่องหมาย * และเว้นวรรค 1 ครั้งก่อนเขียนข้อความเสมอ

(ยกเว้นบรรทัดที่ 1 และบรรทัดที่ 8)

ตัวอย่างการคอมเมนต์ฟังก์ชันของ Service

/**

* ชื่อฟังก์ชัน: addNewMonitor

* คำอธิบาย: ฟังก์ชันนี้ใช้สำหรับเพิ่มข้อมูล Monitor ลงฐานข้อมูล

* Input: ข้อมูลของ Monitor

* Output: ไม่มี

* ชื่อผู้เขียน/แก้ไข: John Doe

* วันที่จัดทำ/แก้ไข: 1 กุมภาพันธ์ 2566

*/

```
async addNewMonitor(model) {
    .....
}
```

**7.5 คอมเมนต์ของไฟล์ Sequelize (IOT_CONNECT_SERVICE)**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. เขียนคอมเมนต์ไฟล์ Sequelize ไว้บรรทัดแรกของไฟล์
3. เขียนคอมเมนต์ไฟล์ Sequelize ด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของไฟล์ Sequelize

ไฟล์ Sequelize ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อไฟล์ Sequelize
2. คำอธิบายราบละเอียดไฟล์ Sequelize
3. ชื่อผู้เขียน/แก้ไข
4. วันที่จัดทำ/แก้ไข

หมายเหตุ

แต่ละบรรทัดให้ใส่เครื่องหมาย * และเว้นวรรค 1 ครั้งก่อนเขียนข้อความเสมอ

(ยกเว้นบรรทัดที่ 1 และบรรทัดที่ 6)

ตัวอย่างการคอมเมนต์ไฟล์ Sequelize

```
/**
 * ชื่อไฟล์: ic_monitors.js
 * คำอธิบาย: สำหรับเก็บ Model ที่ใช้อ้างอิงถึงตาราง ic_monitors ใน Database
 * ชื่อผู้เขียน/แก้ไข: John Doe
 * วันที่จัดทำ/แก้ไข: 1 กุมภาพันธ์ 2566
 */
```

**7.6 การเขียนคอมเมนต์บรรทัดเดียวหรือตัวแปรต่าง ๆ**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. เขียนคอมเมนต์ฟังก์ชันด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์

บ่งบอกความหมายหรือวิธีการทำงานที่ชัดเจน

ตัวอย่างการคอมเมนต์บรรทัดเดียวหรือตัวแปรต่าง ๆ (IOT_CONNECT_SERVICE)

```
// Query string สำหรับดึงค่า Monitor ทั้งหมดที่มีสถานะการใช้งานเป็น True หรือยังไม่โดนลบ
"getAll":"select * from \"ic_monitors\" where is_use=true",
// นำเข้าโมดูลในไฟล์ dashboardService.js มาใช้งานจาก path  '../Service/dashboardService'
var dashboardService = require('../Service/dashboardService');
```

**7.7 การเขียนคอมเมนต์ของไฟล์ component (IOT_CONNECT_CLIENT)**

ข้อบังคับ

1. ให้เขียนคอมเมนต์ทุกไฟล์ ไม่มีข้อยกเว้น
2. เขียนคอมเมนต์ส่วนของ Component ไว้บรรทัดแรกของไฟล์
3. เขียนคอมเมนต์คลาสด้วยภาษาอังกฤษหรือภาษาไทยเท่านั้น

หลักการเขียนคอมเมนต์ส่วนของไฟล์ Component

คอมเมนต์ส่วนของ Component ให้เขียนคอมเมนต์อยู่ในรูปแบบเดียวกัน

1. ชื่อไฟล์
2. คำอธิบายการทำงาน
3. Input
4. Output
5. ชื่อผู้เขียน/แก้ไข
6. วันที่จัดทำ/แก้ไข

ตัวอย่างการคอมเมนต์ส่วนของ Component

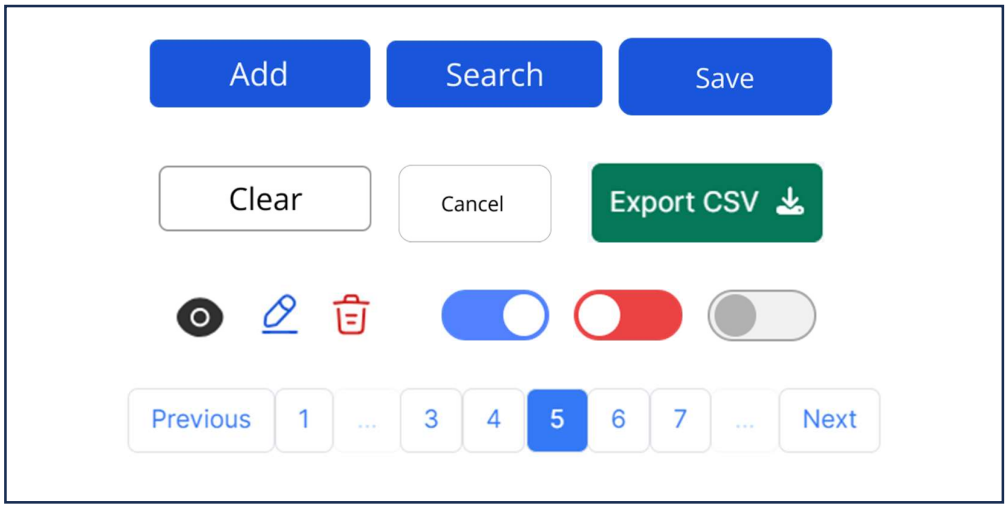
```
/**
 * ชื่อไฟล์: dashboard.js
 * คำอธิบาย: ไฟล์นี้แสดง Dashboard
 * Input: -
 * Output: Dashboard
 * ชื่อผู้เขียน/แก้ไข: Jane Smith
 * วันที่จัดทำ/แก้ไข: 15 กุมภาพันธ์ 2566
*/
<head>
    <title>User List</title>
</head>
```

```
<body>
    /* Dashboard จะถูกแสดงที่นี่ */
</body>
```

## ส่วนที่ 3 มาตรฐานส่วนติดต่อผู้ใช้งาน (UI Standards)

มาตรฐานส่วนติดต่อผู้ใช้นี้ใช้เทมเพลต Bootstrap และ Tailwind เป็นต้นแบบในการกำหนดมาตรฐานที่ใช้ในการพัฒนากันอย่างแพร่หลายและมีลักษณะที่เหมือนกัน ดังนั้นจึงจัดทำมาตรฐานนี้ขึ้นเพื่อให้การแสดงผลส่วนติดต่อผู้ใช้งาน (User Interface) เป็นไปตามมาตรฐานเดียวกัน

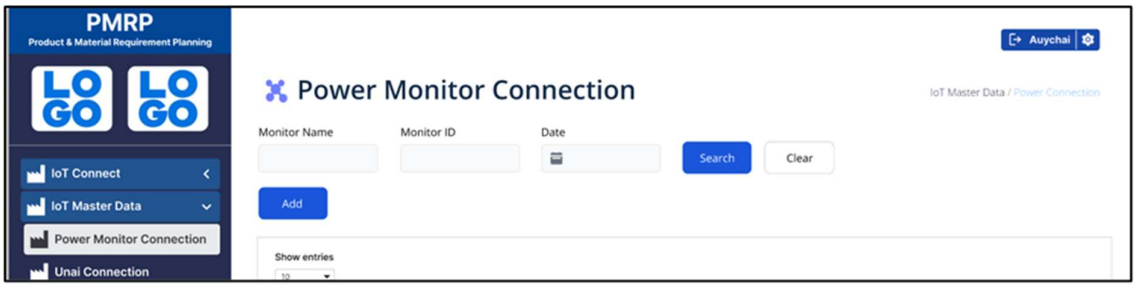
### 1. การแสดงสีปุ่ม (Button Color)




รูปที่ 1-1 แสดงรายการปุ่ม

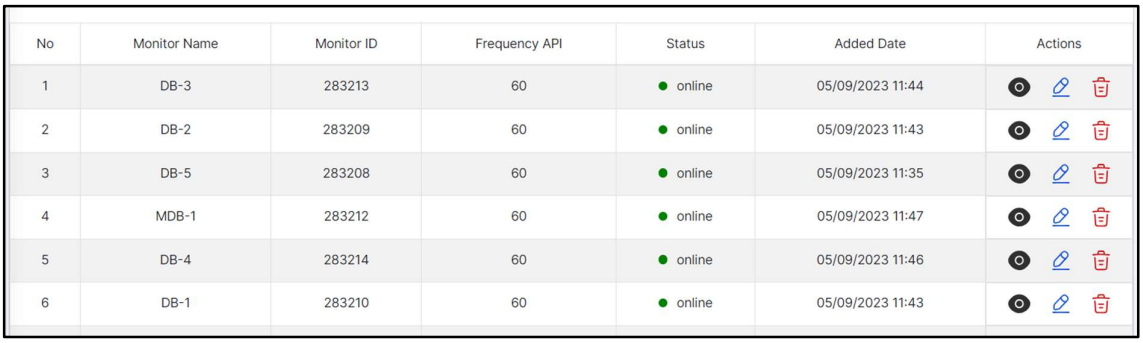
- ปุ่มเพิ่ม ปุ่มค้นหา ปุ่มบันทึก แสดงเป็น สีน้ำเงิน
- ปุ่มล้างฟิลด์ ปุ่มยกเลิก แสดงเป็น สีขาว
- ปุ่มดูรายละเอียด แสดงเป็น สีดำ
- ปุ่มแก้ไข แสดงเป็น สีน้ำเงิน
- ปุ่มลบ แสดงเป็น สีแดง
- ปุ่ม Export CSV แสดงเป็น สีเขียว
- ปุ่มก่อนหน้า ปุ่มถัดไป แสดงเป็น สีขาว
- ปุ่ม Toggle (Active / เลือกทางขวา) แสดงเป็น สีน้ำเงิน
- ปุ่ม Toggle (Inactive) แสดงเป็น สีแดง
- ปุ่ม Toggle (เลือกทางซ้าย) แสดงเป็น สีเทา
- ปุ่มเลขหน้า แสดงเป็น สีขาว ส่วนหน้าที่อยู่เป็นสีน้ำเงิน

## 2. การจัดวางตำแหน่งปุ่ม (Button Position)




รูปที่ 2-1 แสดงการจัดวางตำแหน่งปุ่มค้นหา ปุ่มล้างฟิลด์ ปุ่มเพิ่ม

- ปุ่มค้นหา จัดวางตำแหน่งด้านขวาของช่องฟิลเตอร์ค้นหาวันที่
- ปุ่มล้างฟิลด์ จัดวางตำแหน่งด้านขวาของปุ่มค้นหา
- ปุ่มเพิ่ม จัดวางตำแหน่งด้านล่างของช่องฟิลเตอร์ค้นหาชื่อมอนิเตอร์



| No | Monitor Name | Monitor ID | Frequency API | Status | Added Date | Actions |
|---|---|---|---|---|---|---|
| 1 | DB-3 | 283213 | 60 | online | 05/09/2023 11:44 | |
| 2 | DB-2 | 283209 | 60 | online | 05/09/2023 11:43 | |
| 3 | DB-5 | 283208 | 60 | online | 05/09/2023 11:35 | |
| 4 | MDB-1 | 283212 | 60 | online | 05/09/2023 11:47 | |
| 5 | DB-4 | 283214 | 60 | online | 05/09/2023 11:46 | |
| 6 | DB-1 | 283210 | 60 | online | 05/09/2023 11:43 | |

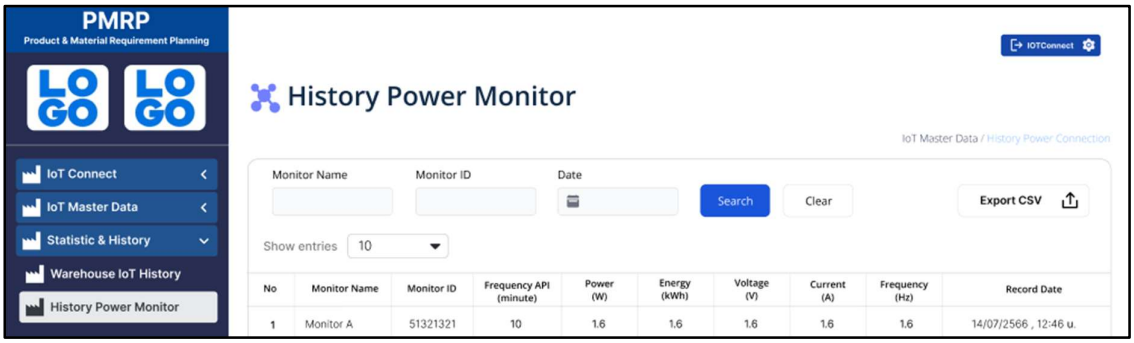
รูปที่ 2-2 แสดงการจัดวางตำแหน่งปุ่มดูรายละเอียด ปุ่มแก้ไข ปุ่มลบ

- ปุ่มดูรายละเอียด จัดวางตำแหน่งด้านซ้ายสุดในคอลัมน์การดำเนินการ (Actions)
- ปุ่มแก้ไข จัดวางตำแหน่งด้านขวาถัดจากปุ่มดูรายละเอียด
- ปุ่มลบ จัดวางตำแหน่งด้านขวาถัดจากปุ่มแก้ไข


รูปที่ 2-3 แสดงการจัดวางตำแหน่งปุ่มก่อนหน้า ปุ่มเลขหน้า ปุ่มถัดไป

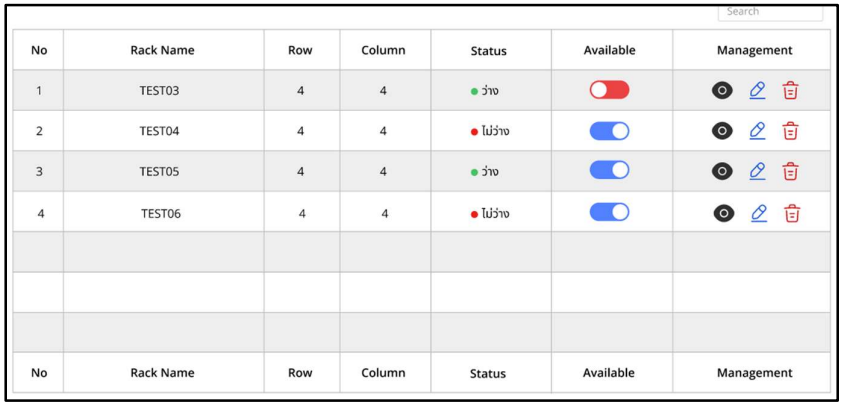
- ปุ่มก่อนหน้า จัดวางตำแหน่งด้านล่างของตาราง
- ปุ่มเลขหน้า จัดวางตำแหน่งด้านขวาถัดจากปุ่มก่อนหน้า

- ปุ่มถัดไป จัดวางตำแหน่งด้านขวาถัดจากปุ่มเลขหน้า



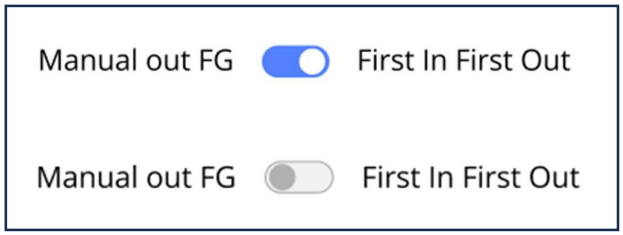

รูปที่ 2-4 แสดงการจัดวางตำแหน่งปุ่ม Export CSV

- ปุ่ม Export CSV จัดวางตำแหน่งด้านขวาของช่องฟิลเตอร์ค้นหาวันที่




รูปที่ 2-5 แสดงการจัดวางตำแหน่งปุ่ม Toggle (Active) ปุ่ม Toggle (Inactive)

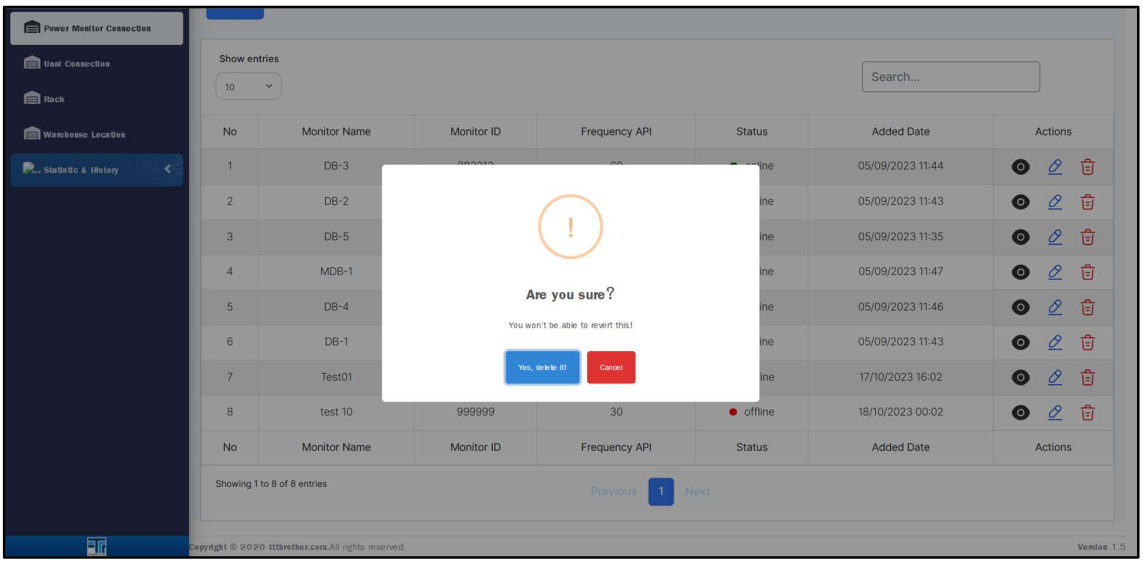
- ปุ่ม Toggle (Active) จัดวางตำแหน่งกึ่งกลางในคอลัมน์พร้อมใช้งาน (Available)
- ปุ่ม Toggle (Inactive) จัดวางตำแหน่งกึ่งกลางในคอลัมน์พร้อมใช้งาน (Available)




รูปที่ 2-6 แสดงการจัดวางตำแหน่งปุ่ม Toggle (แบบตัวเลือก)

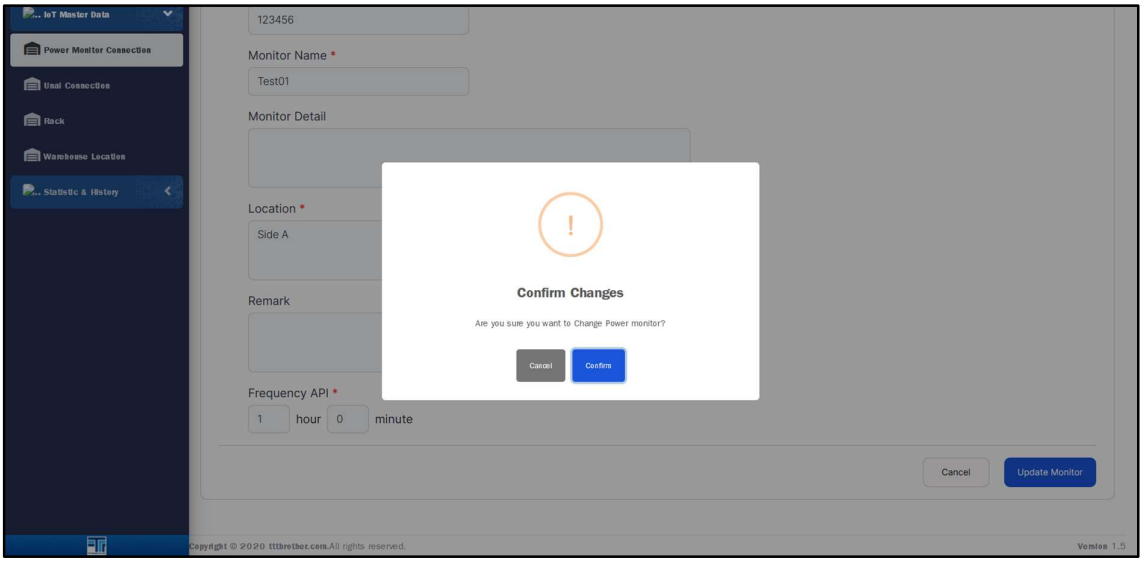
- ปุ่ม Toggle (เลือกทางขวา) จัดวางตำแหน่งกึ่งกลางระหว่างตัวเลือก
- ปุ่ม Toggle (เลือกทางซ้าย) จัดวางตำแหน่งกึ่งกลางระหว่างตัวเลือก

### 3. การแสดงกล่องข้อความยืนยัน (Confirm Box)



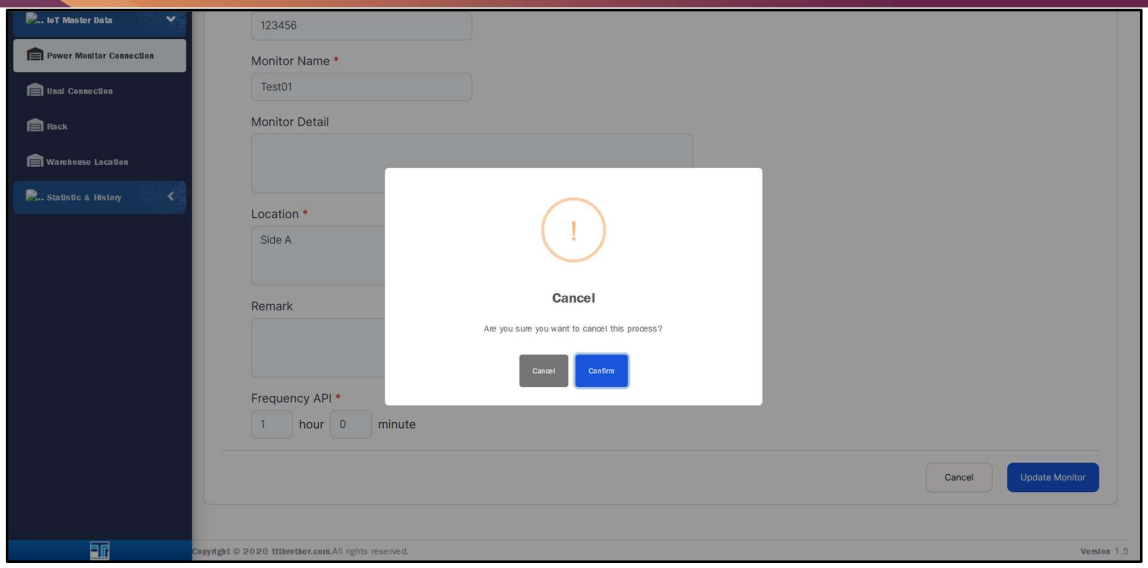

รูปที่ 3-1 ตำแหน่งการแสดงกล่องข้อความยืนยันการลบ

- ปุ่มยืนยัน แสดงเป็นสีน้ำเงิน จัดวางตำแหน่งซ้ายล่าง
- ปุ่มยกเลิก แสดงเป็นสีแดง จัดวางตำแหน่งด้านขวาถัดจากปุ่มยืนยัน




รูปที่ 3-2 ตำแหน่งการแสดงกล่องข้อความยืนยันการแก้ไข

- ปุ่มยกเลิก แสดงเป็นสีเทา จัดวางตำแหน่งซ้ายล่าง
- ปุ่มยืนยัน แสดงเป็นสีน้ำเงิน จัดวางตำแหน่งด้านขวาถัดจากปุ่มยืนยัน




รูปที่ 3-3 ตำแหน่งการแสดงกล่องข้อความยกเลิกการแก้ไข

- ปุ่มยกเลิก แสดงเป็นสีเทา จัดวางตำแหน่งซ้ายล่าง
- ปุ่มยืนยัน แสดงเป็นสีน้ำเงิน จัดวางตำแหน่งด้านขวาถัดจากปุ่มยืนยัน

## 4. การแสดงผลอื่น ๆ

กรณีเกี่ยวกับมาตรฐานในส่วนติดต่อกับผู้ใช้ที่กำหนดรูปแบบกันในทีมพัฒนาให้เป็น รูปแบบเดียวกันภายในระบบ มีดังนี้

**การแสดงข้อความแจ้งเตือน Form Validation** รูปแบบการแสดงผลจะเป็นในลักษณะของ Alert Box ทั้งระบบ

**การแสดง Tooltip** จะมีคำอธิบายเพิ่มเติมในส่วนเงื่อนไขต่าง ๆ ที่จำเป็นในการกรอกข้อมูลลงแบบฟอร์มซึ่งจะเป็นในลักษณะเดียวกันทั้งระบบ

**การแสดง Placeholder** ใช้ในการแสดงตัวอย่างของข้อความที่จำเป็นต้องกรอกภายในฟิลด์นั้น ๆ ซึ่งข้อความที่แสดงขึ้นมานั้นจะเป็นในลักษณะข้อความพื้นหลังของฟิลด์

**การแสดงไอคอนหรือรูปภาพแทนการดำเนินการ** เลือกใช้ในสิ่งที่เหมาะสมกับเทมเพลตของระบบ ซึ่งจะมีลักษณะเดียวกันทั้งระบบ

**การแสดงวันที่ (Date Format)** จะเป็นการแสดง วัน เดือน ปี และเวลา ซึ่งจะมีรูปแบบเดียวกันทั้งระบบ