

รายงาน  
ระบบสมาร์ทฟาร์มขนาดเล็ก

จัดทำโดย

นายวีรวิชญ์ พิชิตวงศ์ศรี	รหัส	603040109-0
นางสาวสุธิมา วิเชียรทวี	รหัส	613040412-0
นายกิตติพัฒน์ แดงดี	รหัส	613040438-2
นางสาวสุพัชรี ไชยยา	รหัส	613040582-5

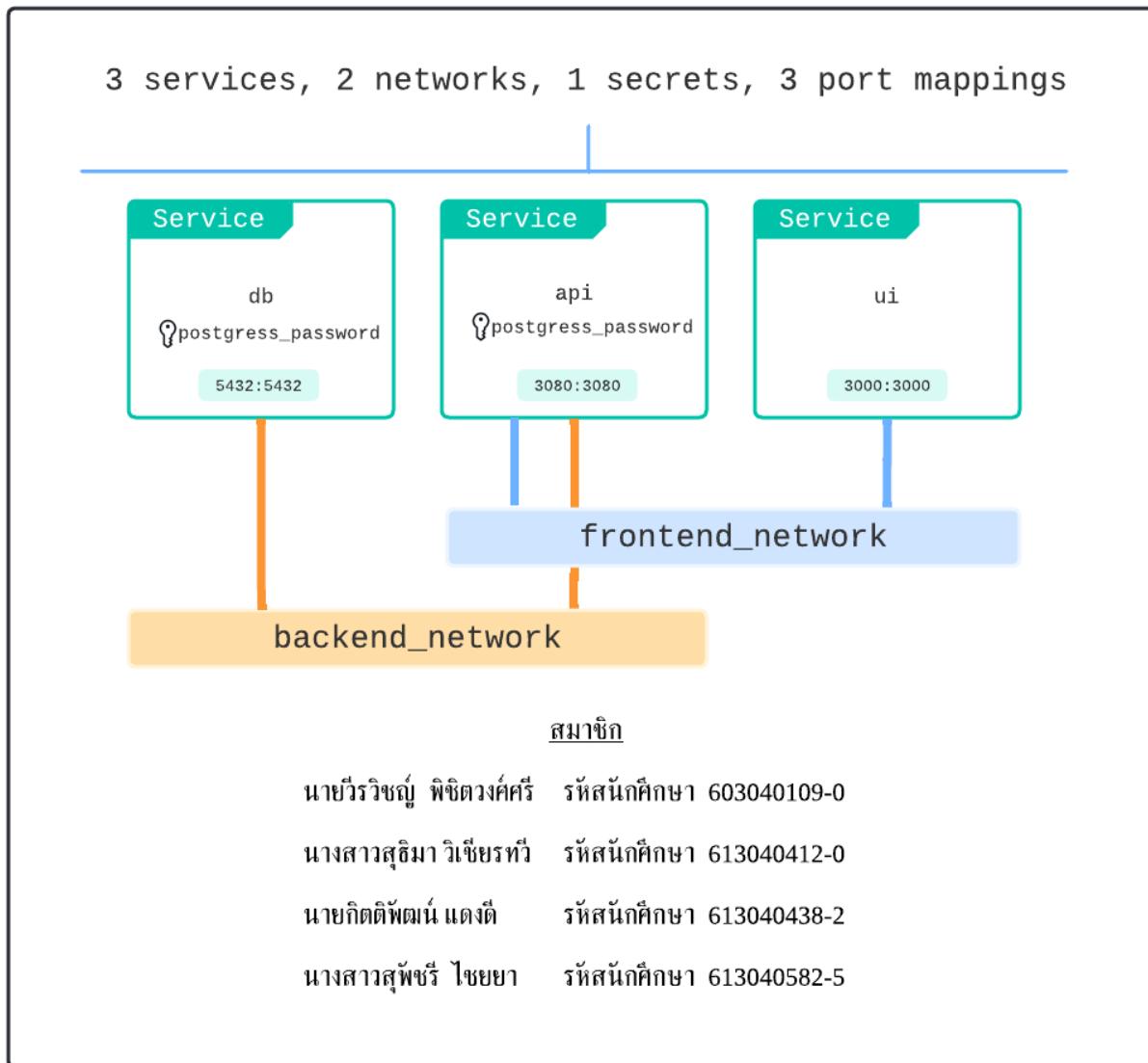
เสนอ

ผศ. ดร. ชัชชัย คุณบัว

รายงานนี้เป็นส่วนหนึ่งของรายวิชา EN814774 หัวข้อพิเศษทางคอมพิวเตอร์ซอฟต์แวร์  
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น  
ภาคการศึกษาปลายปีการศึกษา 2564

## บทที่ 1 วิธีการดำเนินงาน

### 1.1 ออกแบบระบบการทำงาน



รูปที่ 1.1 โครงสร้างการออกแบบระบบการทำงาน

## 1.2 โครงสร้างไฟล์ YAML

คณผู้จัดทำเขียนไฟล์ docker-compose.yaml ในรูปแบบไฟล์ YAML ดังนี้

```
1  version: "3"
2
3  services:
4    # Database
5    db:
6      image: postgres
7      ports:
8        - 5432:5432
9      container_name: database
10     environment:
11       POSTGRES_DB: database
12       POSTGRES_USER: user
13       POSTGRES_PASSWORD: root
14     networks:
15       - backend_network
16     volumes:
17       - postgres:/data/db
18   # Backend server
19   api:
20     build:
21       context: ./api
22       dockerfile: Dockerfile.dev
23     ports:
24       - 3080:3080
25     container_name: api
26     networks:
27       - backend_network
28       - frontend_network
29     environment:
30       POSTGRES_DB: database
31       POSTGRES_USER: user
32       POSTGRES_PASSWORD: root
33       POSTGRES_HOST: db
```

รูปที่ 1.2 ไฟล์ docker-compose.yaml (1)

```
34      volumes:
35        - ./api/src:/usr/app/src
36      depends_on:
37        - db
38    # Frontend server
39    ui:
40      build:
41        context: ./ui
42        dockerfile: Dockerfile.dev
43      ports:
44        - 3000:3000
45      container_name: ui
46      networks:
47        - frontend_network
48      volumes:
49        - ./ui/src:/usr/app/src
50        - ./ui/public:/usr/app/public
51      depends_on:
52        - api
53      stdin_open: true
54      tty: true
55
56    networks:
57      backend_network:
58        driver: bridge
59      frontend_network:
60        driver: bridge
61
62    volumes:
63      postgres:
64        driver: local
```

รูปที่ 1.3 ไฟล์ docker-compose.yaml (2)

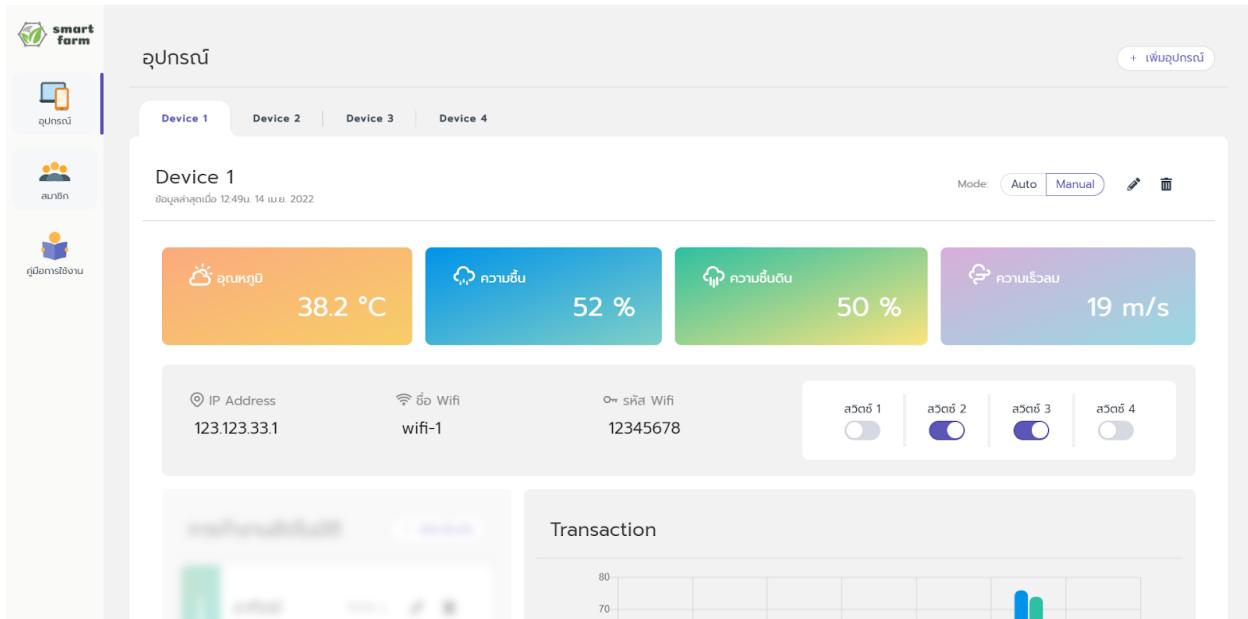
## บทที่ 2

### ผลการดำเนินงาน

#### 2.1 ระบบสมาร์ทฟาร์มขนาดเล็ก

องค์ประกอบบนระบบสมาร์ทฟาร์มขนาดเล็ก เมื่อทำการเปิดเว็บแอปพลิเคชันระบบสมาร์ทฟาร์มขนาดเล็กจะพบเมนูดังนี้

1. เมนูอุปกรณ์
2. สมาชิก
3. คู่มือการใช้งาน



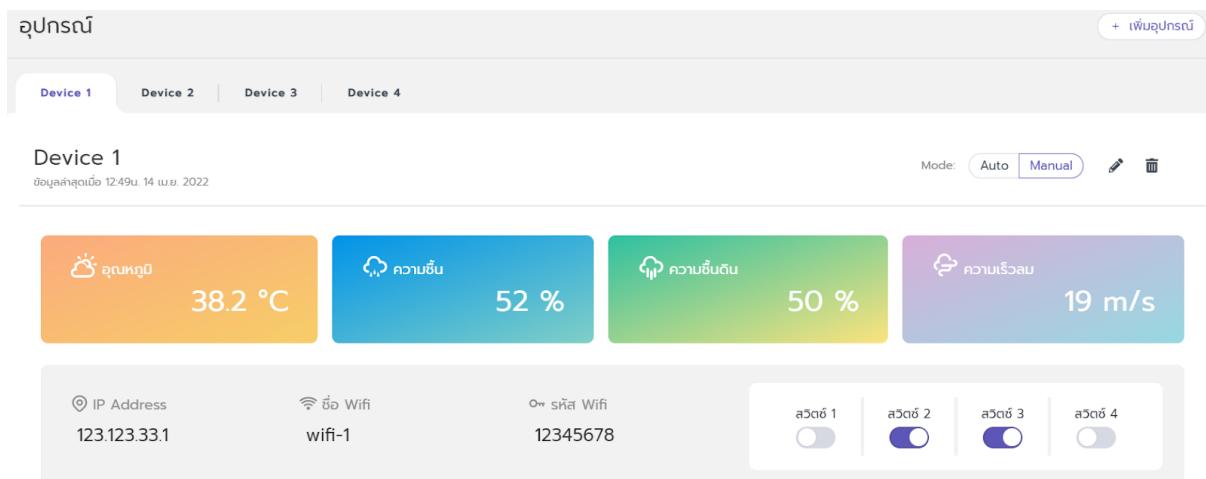
รูปที่ 2.1 เว็บแอปพลิเคชันระบบสมาร์ทฟาร์มขนาดเล็ก

## 2.2. เมนูอุปกรณ์

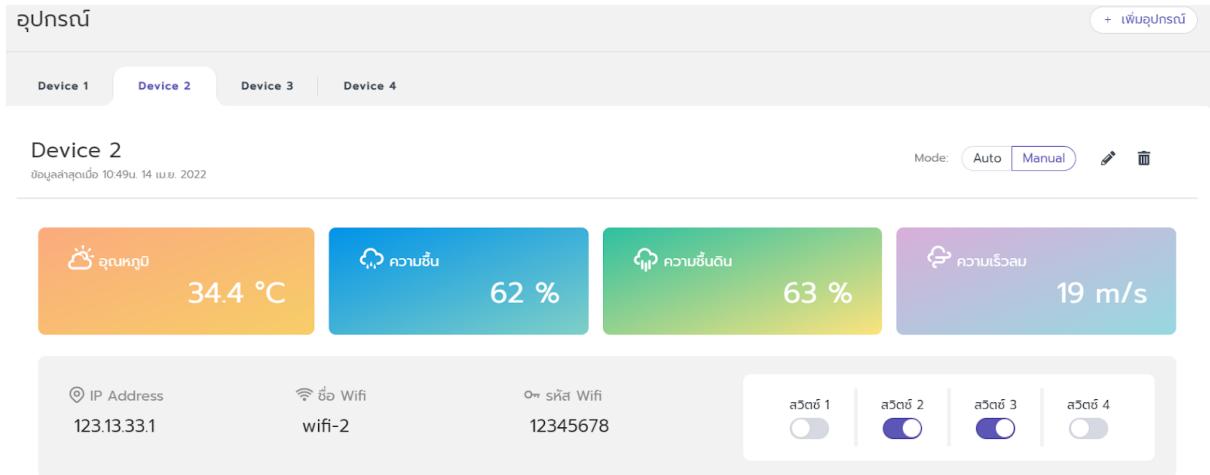
เมนูอุปกรณ์ จากรูปที่ 2.1 เมื่อเลือกเมนูอุปกรณ์จะแสดงหน้าต่างเว็บไซต์ดังนี้

1. แสดงรายการของอุปกรณ์ ซึ่งเริ่มต้นจะมี 4 อุปกรณ์
2. ในแต่ละอุปกรณ์จะมี Dashboard แสดงข้อมูลต่างๆ ได้แก่ ข้อมูลค่าอุณหภูมิ, ความชื้น, ความชื้นดิน และความเร็วลม ของอุปกรณ์นั้นๆ
3. มีฟังก์ชันการทำงาน ได้แก่ เพิ่มอุปกรณ์, แก้ไขอุปกรณ์ และลบอุปกรณ์
4. สามารถเลือก Mode การทำงานได้เป็นแบบ Auto/Manual
5. แต่ละอุปกรณ์ จะสามารถสั่งเปิด/ปิด Relay ได้ 4 ตัว
6. แสดงผลการตั้งค่าการทำงานอัตโนมัติ ได้แก่ ทำงานอัตโนมัติรายสัปดาห์ และทำงานอัตโนมัติแบบเซ็นเซอร์
7. มีฟังก์ชันการทำงานอัตโนมัติของอุปกรณ์ ได้แก่ เพิ่มการทำงานอัตโนมัติ, แก้ไขการทำงานอัตโนมัติ และลบการทำงานอัตโนมัติ
8. แสดงผล Transaction ของเวลาในรูปแบบกราฟแท่ง ซึ่งมีข้อมูลค่าอุณหภูมิ, ความชื้น, ความชื้นดิน และความเร็วลม

แสดงผลข้อมูลค่าอุณหภูมิ, ความชื้น, ความชื้นดิน และความเร็วลม ของอุปกรณ์แต่ละตัว

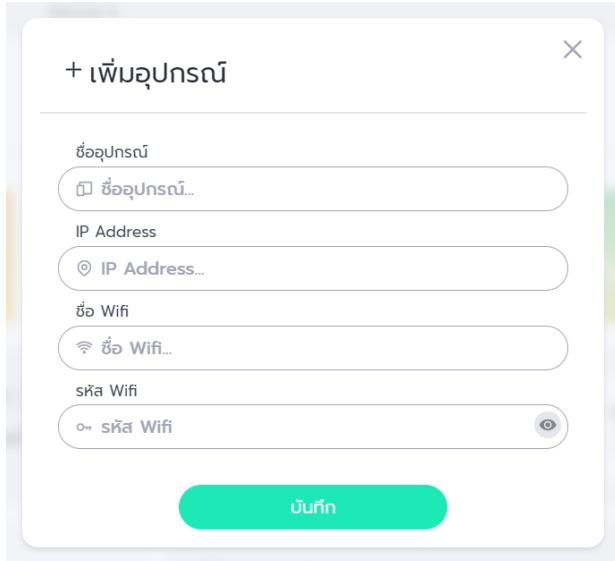


รูปที่ 2.2 แสดง Dashboard ของอุปกรณ์ชื่อ Device 1



รูปที่ 2.3 แสดง Dashboard ของอุปกรณ์ชื่อ Device 2

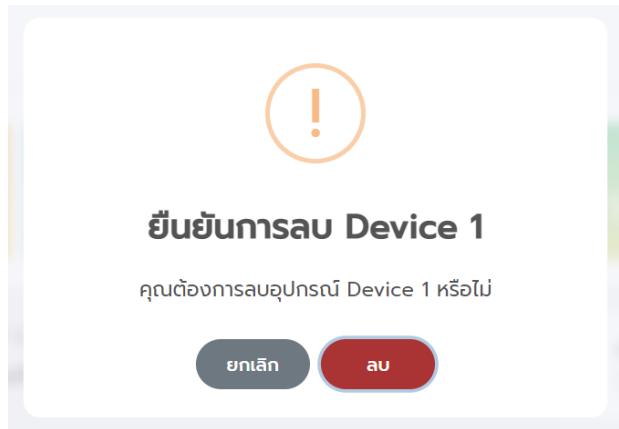
### พัฒนาการทำงานเพิ่มอุปกรณ์, แก้ไขอุปกรณ์ และลบอุปกรณ์



รูปที่ 2.4 ฟอร์มพัฒนาการเพิ่มอุปกรณ์



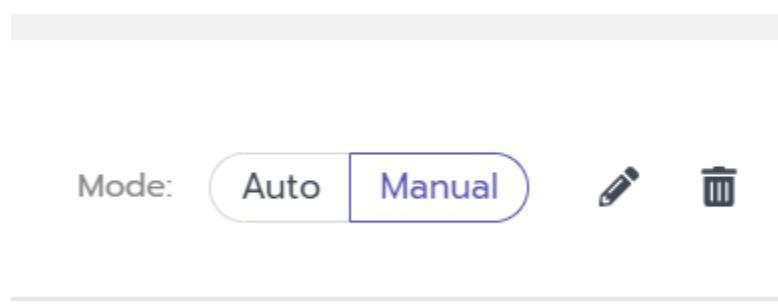
รูปที่ 2.5 ฟอร์มฟังก์ชันการแก้ไขอุปกรณ์



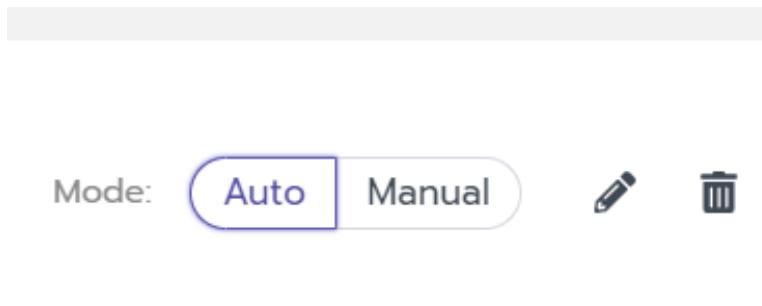
รูปที่ 2.6 ฟังก์ชัน Modal แสดงการลบอุปกรณ์

### แสดงการเลือก Mode การทำงานแบบ Auto/Manual

- หากทำการเลือก mode เป็นแบบ Auto จะสามารถทำการเพิ่มเงื่อนไขการทำงานอัตโนมัติได้ ซึ่งการเปิด/ปิดสวิตช์ Relay จะถูกตั้งแต่ให้ทำงานเป็นแบบ Auto ไม่สามารถตั้งค่าได้
- แต่หากทำการเลือก mode เป็นแบบ Manual จะทำตั้งค่าการเปิด/ปิดสวิตช์ Relay ได้ ซึ่งการเพิ่มเงื่อนไขการทำงานอัตโนมัติจะถูกตั้งแต่ให้ทำงานเป็นแบบ Auto ไม่สามารถตั้งค่าได้
- โดยค่าที่ถูกตั้งเป็นค่าเริ่มต้นจะเป็นแบบ Manual

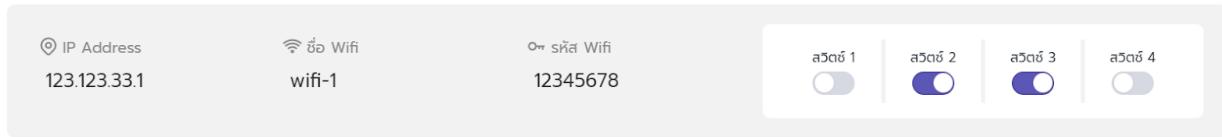


รูปที่ 2.7 แสดงการเลือก Mode การทำงาน Manual หรือค่าเริ่มต้น



รูปที่ 2.8 แสดงการเลือก Mode การทำงาน Auto

## แสดงการเปิด/ปิดสวิตช์ Relay ของแต่ละอุปกรณ์ในการทำงานแบบ Manual



รูปที่ 2.9 แสดงการตั้งค่าเปิด/ปิดสวิตช์ Relay แบบ Manual

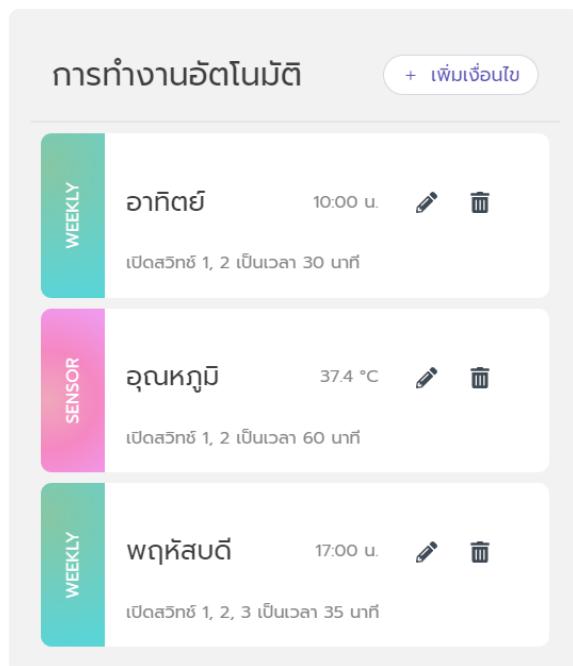
## แสดงการเปิด/ปิดสวิตช์ Relay ของแต่ละอุปกรณ์ในการทำงานแบบ Auto



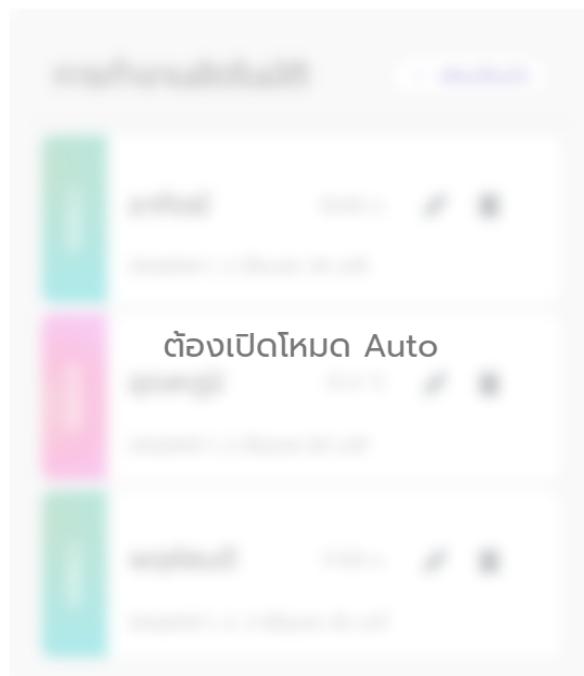
รูปที่ 2.10 แสดงการตั้งค่าเปิด/ปิดสวิตช์ Relay แบบ Auto

## แสดงการทำงานแบบอัตโนมัติ

- หากทำการเลือก mode เป็นแบบ Auto จะสามารถทำการเพิ่มเงื่อนไขการทำงานอัตโนมัติได้โดยจะแสดงรายการที่ได้ทำการเพิ่มเงื่อนไขการทำงานอัตโนมัติเป็นรายการที่ได้ตั้งค่าเอาไว้
- แต่หากทำการเลือก mode เป็นแบบ Manual จะถูกตั้งแต่ให้ทำงานเป็นแบบ Auto โดยจะไม่สามารถตั้งค่าได้
- โดยค่าที่ถูกตั้งเป็นค่าเริ่มต้นจะเป็นแบบ Manual



รูปที่ 2.11 แสดงการทำงานอัตโนมัติแบบ Auto



รูปที่ 2.12 แสดงการทำงานอัตโนมัติแบบ Manual

ฟังก์ชันเพิ่มการทำงานอัตโนมัติ, แก้ไขการทำงานอัตโนมัติ และลบการทำงานอัตโนมัติ

+ เพิ่มการทำงานอัตโนมัติของอุปกรณ์ 1

กรุณาเลือกประเภท Schedule

รายสัปดาห์

Sensor

กำหนดวัน

จันทร์

เวลา

—

ตั้งเวลาการเปิด Relay (นาที)

— ตั้งเวลา... +

สวิตซ์ 1 สวิตซ์ 2 สวิตซ์ 3 สวิตซ์ 4

บันทึก

รูปที่ 2.13 ฟอร์มฟังก์ชันการเพิ่มการทำงานอัตโนมัติ

แก้ไขการทำงานอัตโนมัติของอุปกรณ์ 1

กรุณาเลือกประเภท Schedule

รายสัปดาห์

Sensor

กำหนดวัน

อาทิตย์

เวลา

10:00

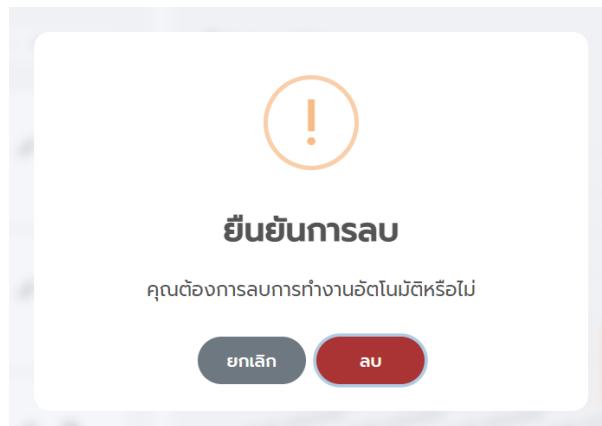
ตั้งเวลาการเปิด Relay (นาที)

— 30 +

สวิตซ์ 1 สวิตซ์ 2 สวิตซ์ 3 สวิตซ์ 4

บันทึก

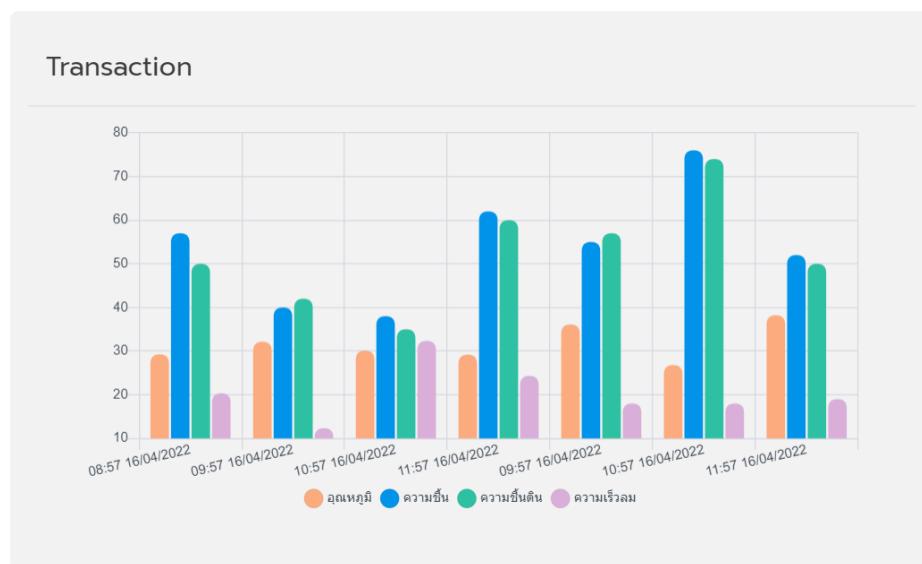
รูปที่ 2.14 ฟอร์มฟังก์ชันการแก้ไขการทำงานอัตโนมัติ



รูปที่ 2.15 พอร์มฟังก์ชันการลบการทำงานอัตโนมัติ

### แสดงผล Transaction ของเวลาในรูปแบบกราฟแท่ง

- กราฟแสดงผลของ Transaction จะเป็นกราฟแท่งที่ตรวจสอบและทำการบันทึก Transaction จากการทำงานที่ถูกตั้งค่าไว้ทั้ง Auto หรือ Manual ของแต่ละอุปกรณ์
- โดยจะมีค่าที่แสดง ได้แก่ ข้อมูลค่าอุณหภูมิ, ค่าความชื้น, ค่าความชื้นดิน และค่าความเร็วลม ของแต่ละอุปกรณ์



รูปที่ 2.16 กราฟแท่งแสดงผลการบันทึก Transaction ข้อมูลค่าต่างๆ

## 2.3 เมนูสมาชิก

เมนูสมาชิก จากรูปที่ 1 เมื่อเลือกเมนูสมาชิกจะแสดงหน้าต่างเว็บไซต์ ดังนี้

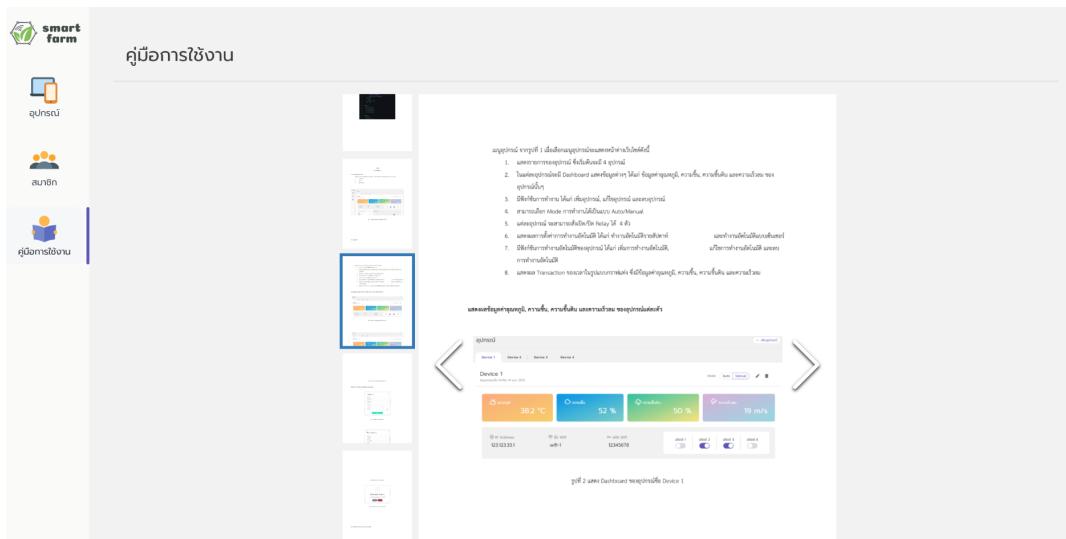
- หน้าสมาชิกเป็นการแสดงข้อมูลผู้จัดทำโครงการนี้ โดยมีข้อมูล ชื่อ รหัสนักศึกษา มหาวิทยาลัยขอนแก่น อีเมล และโซเชียลมีเดียต่างๆ สำหรับการติดต่อ

序號	ชื่อ	อีเมล	บัตรประชาชน	โซเชียลมีเดีย
1	VP	viravich.phi@kkumail.com	603040109-0	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">LinkedIn</a> <a href="#">YouTube</a>
2	SV	sutima_eci@kkumail.com	613040412-0	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">LinkedIn</a> <a href="#">YouTube</a>
3	KD	kittipat_dd@kkumail.com	613040438-2	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">LinkedIn</a> <a href="#">YouTube</a>
4	SC	supatcharee_chaiya@kkumail.com	613040582-5	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">LinkedIn</a> <a href="#">YouTube</a>

รูปที่ 2.17 แสดงรายชื่อสมาชิกผู้จัดทำ

## 2.4 เมนูคู่มือการใช้งาน

- หน้าคู่มือการใช้งานเป็นหน้าแสดงรายละเอียดการใช้งานของเว็บแอปพลิเคชันระบบสมาร์ทฟาร์มขนาดเล็ก



รูปที่ 2.18 แสดงคู่มือการใช้งาน

## 2.5 API เชื่อมต่อระหว่าง โมดูล และระบบฐานข้อมูล

- API เชื่อมต่อระหว่าง module และระบบฐานข้อมูล ของ device

```
route.get("/", async (_, res) => {
  try {
    const devices = await Devices.findAll({
      order: [[ "id", "ASC" ]],
    });
    res.json(devices);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.19 พังก์ชันการแสดงรายการ device ทั้งหมด

```
route.get("/:id", async (req, res) => {
  const { id } = req.params;
  try {
    const devices = await Devices.findAll({
      where: {
        id,
      },
    });
    res.json(devices);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.20 พิมพ์ชันการแสดงข้อมูลเฉพาะ device ที่ต้องการ

```
route.patch("/:id", async (req, res) => {
  const { id } = req.params;
  const data = req.body as { [key: string]: any };
  Object.keys(data).forEach(
    (key) => data[key] === undefined && delete data[key]
  );
  try {
    const device = await Devices.update(data, {
      where: {
        id: Number(id),
      },
    });
    res.json(device);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.21 พิมพ์ชันการอัปเดตข้อมูลของ device

```
route.post("/", async (req, res) => {
  const data = req.body;
  try {
    const device = await Devices.create(data);
    res.json(device);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.22 พังก์ชันการสร้างข้อมูลของ device

```
route.delete("/:id", async (req, res) => {
  const { id } = req.params;
  try {
    const device = await Devices.destroy({
      where: {
        id,
      },
    });
    res.json(device);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.23 พังก์ชันการลบข้อมูลเฉพาะ device ที่ต้องการ

- API เชื่อมต่อระหว่าง module และระบบฐานข้อมูล ของ transaction แต่ละตัว

```
route.get("/", async (_, res) => {
  try {
    const transactions = await Transactions.findAll();
    res.json(transactions);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.24 พิ้งก์ชันการแสดงรายการ transaction ทั้งหมด

```
route.get("/:id/latest", async (req, res) => {
  const { id } = req.params;
  try {
    const transactions = await Transactions.findAll({
      where: {
        deviceId: id,
      },
      order: [["id", "DESC"]],
      limit: 1,
    });
    if (!transactions.length) return res.json(null);
    res.json(transactions[0]);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.25 พิ้งก์ชันการแสดงข้อมูล transaction ล่าสุด เฉพาะ device ที่ต้องการ

```

route.get("/:id", async (req, res) => {
  const { id } = req.params;
  try {
    const transactions = await Transactions.findAll({
      where: {
        deviceId: id,
      },
    });
    res.json(transactions);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});

```

รูปที่ 2.26 พัฟก์ชันการแสดงรายการ transaction ทั้งหมด เนื่องจาก device ที่ต้องการ

- API เชื่อมต่อระหว่าง module และระบบฐานข้อมูล ของ schedule แต่ละตัว

```

route.get("/", async (_, res) => {
  try {
    const schedules = await Schedules.findAll();
    res.json(schedules);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});

```

รูปที่ 2.27 พัฟก์ชันการแสดงรายการ schedule ทั้งหมด

```
route.get("/:id", async (req, res) => {
  const { id } = req.params;
  try {
    const schedules = await Schedules.findAll({
      where: {
        deviceId: id,
      },
    });
    res.json(schedules);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.28 พังก์ชันการแสดงข้อมูลเฉพาะ schedule ที่ต้องการ

```
route.post("/", async (req, res) => {
  const data = req.body;
  try {
    const schedules = await Schedules.create(data);
    res.json(schedules);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.29 พังก์ชันการสร้างข้อมูลของ schedule

```
route.patch("/:id", async (req, res) => {
  const { id } = req.params;
  const data = req.body as { [key: string]: any };
  Object.keys(data).forEach(
    (key) => data[key] === undefined && delete data[key]
  );
  try {
    const schedules = await Schedules.update(data, {
      where: {
        id: Number(id),
      },
    });
    res.json(schedules);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.30 ฟังก์ชันการอัปเดตข้อมูลของ schedule

```
route.delete("/:id", async (req, res) => {
  const { id } = req.params;
  try {
    const schedules = await Schedules.destroy({
      where: {
        id,
      },
    });
    res.json(schedules);
  } catch (error) {
    res.status(400).json({
      message: error.message,
    });
  }
});
```

รูปที่ 2.31 ฟังก์ชันการลบข้อมูลเฉพาะ schedule ที่ต้องการ

## 2.6 แสดง database table ข้อมูล

The screenshot shows a PostgreSQL client interface with the 'transactions' table selected. The table has columns: id, device\_id, temperature, moisture, soil\_moisture, wind\_speed, timestamp, createdAt, and updatedAt. The data consists of 19 rows of transaction records. Below the table, there is a SQL history pane with several log entries related to the database schema and data.

id	device_id	temperature	moisture	soil_moisture	wind_speed	timestamp	createdAt	updatedAt
1	1	29.26	57	50	20.34	2022-04-16 10:50:54.995+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
2	1	32.14	40	42	12.34	2022-04-16 11:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
3	1	30.11	38	35	32.34	2022-04-16 12:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
4	1	29.19	62	60	24.32	2022-04-16 13:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
5	2	28.5	64	52	30.32	2022-04-16 14:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
6	2	37.1	35	33	28.32	2022-04-16 13:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
7	2	28.5	54	50	33.54	2022-04-16 12:50:55+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
8	2	35.4	28	30	8.2	2022-04-16 11:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
9	3	27.6	63	60	13	2022-04-16 10:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
10	3	36.3	56	52	23	2022-04-16 11:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
11	3	28.6	75	73	18	2022-04-16 12:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
12	3	34.6	63	60	24	2022-04-16 13:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
13	3	29.5	78	76	16	2022-04-16 14:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
14	4	35.3	61	59	20	2022-04-16 13:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
15	4	27.5	78	76	21	2022-04-16 12:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
16	4	35.4	61	59	22	2022-04-16 11:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
17	4	27.6	78	79	20	2022-04-16 10:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
18	1	36.1	55	57	18	2022-04-16 11:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00
19	1	26.8	76	74	18	2022-04-16 12:50:55.001+00	2022-04-16 15:50:56.099+00	2022-04-16 15:50:56.099+00

รูปที่ 2.32 ระบบฐานข้อมูล โครงสร้างข้อมูลของ transaction

The screenshot shows a PostgreSQL client interface with the 'schedule' table selected. The table has columns: id, type, condition, value, period, active\_relay, deviceld, createdAt, and updatedAt. The data consists of 12 rows of schedule records. Below the table, there is a SQL history pane with several log entries related to the database schema and data.

id	type	condition	value	period	active_relay	deviceld	createdAt	updatedAt
1	WEEKLY	sunday	10:00	30	1, 2	1	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
2	SENSOR	temperat...	37.4	60	1, 2	1	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
3	WEEKLY	monday	11:00	30	2, 3	2	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
4	SENSOR	moisture	33.6	60	2, 3	2	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
5	WEEKLY	tuesday	09:00	30	3, 4	3	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
6	SENSOR	temperat...	35.4	60	3, 4	3	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
7	WEEKLY	wednesday	15:00	30	4, 1	4	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
8	SENSOR	temperat...	34.2	60	4, 1	4	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
9	WEEKLY	thursday	17:00	35	1, 2, 3	1	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
10	SENSOR	temperat...	34.2	60	1, 2, 3	1	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
11	WEEKLY	friday	14:07	15	1, 2, 3, 4	2	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00
12	SENSOR	moisture	37.3	60	1, 2, 3, 4	2	2022-04-16 15:50:56.158+00	2022-04-16 15:50:56.158+00

รูปที่ 2.33 ระบบฐานข้อมูล โครงสร้างข้อมูลของ schedule

PostgreSQL 14.2 : Spetial : database : public.devices

	id	name	ip_address	wifi_name	wifi_password	mode	status_relay_1	status_relay_2	status_relay_3	status_relay_4	createdAt	updatedAt
1	Device 1	123.123.33.1	wifi-1	12345678	MAN...	FALSE	◊	TRUE	◊	TRUE	◊	2022-04-16 15:50:56.166+00
2	Device 2	123.13.33.1	wifi-2	12345678	MAN...	◊	FALSE	◊	TRUE	◊	FALSE	◊ 2022-04-16 15:50:56.166+00
3	Device 3	123.13.33.1	wifi-3	12345678	MAN...	◊	FALSE	◊	TRUE	◊	FALSE	◊ 2022-04-16 15:50:56.166+00
4	Device 4	123.13.33.1	wifi-	12345678	MAN...	◊	FALSE	◊	TRUE	◊	TRUE	◊ 2022-04-16 15:50:56.166+00

Data Structure + Row 1-4 of 4 rows Columns Filters

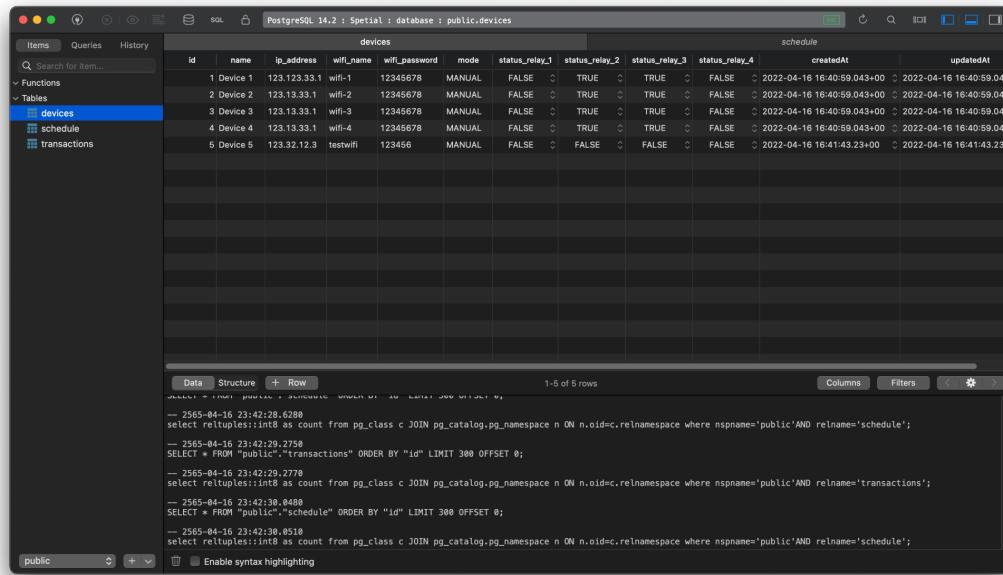
```
-- 2565-04-16 22:51:10.6480
SELECT * FROM "public"."devices" ORDER BY "id" LIMIT 300 OFFSET 0;
-- 2565-04-16 22:51:10.6530
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.relnamespace where nspname='public'AND relname='devices';
```

public Enable syntax highlighting

รูปที่ 2.34 ระบบฐานข้อมูล โครงสร้างข้อมูลของ devices

## 2.7 สามารถจัดเก็บข้อมูลไปยัง database แบบ Persistent data

- จากราบข้อมูลเดิมจะมีข้อมูลเดิมแสดงอยู่ ขณะที่ยังเปิดการใช้งาน Service database อยู่



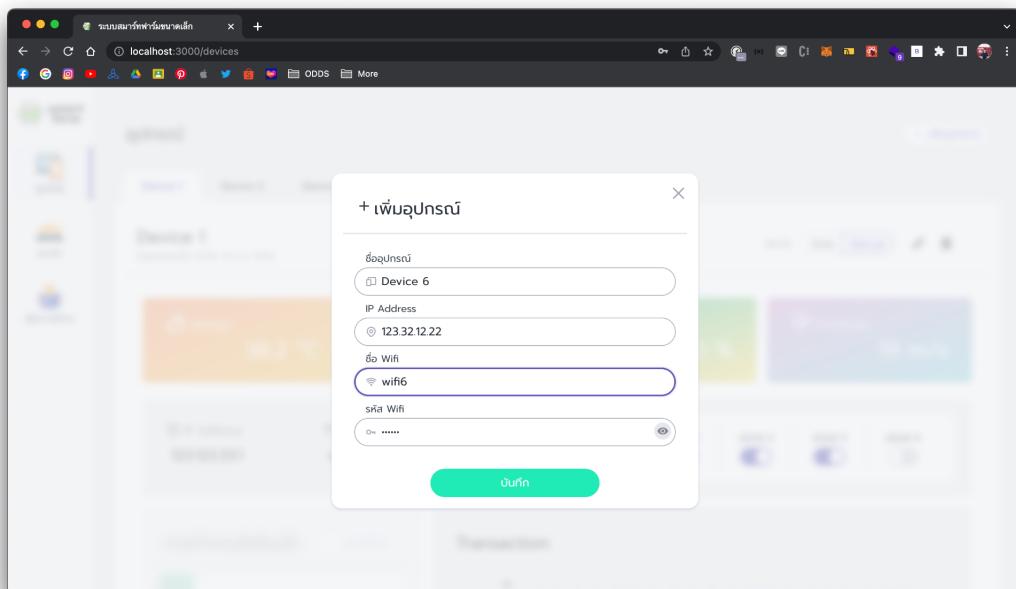
The screenshot shows the PostgreSQL 14.2 interface with the 'public' schema selected. The 'devices' table is open, displaying the following data:

	devices										schedule	
	<b>id</b>	<b>name</b>	<b>ip_address</b>	<b>wifi_name</b>	<b>wifi_password</b>	<b>mode</b>	<b>status_relay_1</b>	<b>status_relay_2</b>	<b>status_relay_3</b>	<b>status_relay_4</b>	<b>createdAt</b>	<b>updatedAt</b>
1	Device 1	123.123.33.1	wifi-1	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	TRUE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043+00
2	Device 2	123.13.33.1	wifi-2	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	TRUE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043+00
3	Device 3	123.13.33.1	wifi-3	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	TRUE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043+00
4	Device 4	123.13.33.1	wifi-4	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	TRUE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043+00
5	Device 5	123.32.12.3	testwifi	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:41:43.23+00	2022-04-16 16:41:43.23+00

Below the table, the SQL history shows several queries related to the 'public' schema, including counts for 'transactions' and 'schedule' tables.

รูปที่ 2.35 ฐานข้อมูลก่อนเพิ่มอุปกรณ์

- ทำการเพิ่มอุปกรณ์



The screenshot shows a web browser window with the URL [localhost:3000/devices](http://localhost:3000/devices). A modal dialog box is open, titled '+ เพิ่มอุปกรณ์'. The form contains the following fields:

- ชื่ออุปกรณ์: Device 6
- IP Address: 123.32.12.22
- ชื่อ WiFi: wifi6
- รหัส WiFi: \*\*\*\*\*

A green 'บันทึก' (Save) button is located at the bottom right of the modal.

รูปที่ 2.36 เพิ่มอุปกรณ์ใหม่

- จะมี Device 6 เพิ่มเข้ามาใน Database

PostgreSQL 14.2 : Spelt : database : public.devices

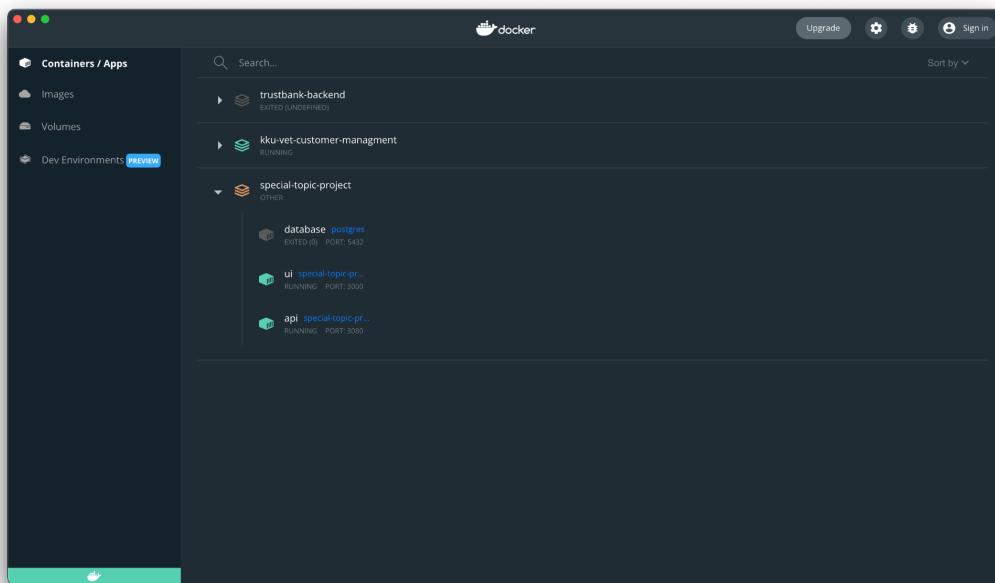
devices									schedule		
id	name	ip_address	wifi_name	wifi_password	mode	status_relay_1	status_relay_2	status_relay_3	status_relay_4	createdAt	updatedAt
1	Device 1	123.123.33.1	wifi-1	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
2	Device 2	123.13.33.1	wifi-2	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
3	Device 3	123.13.33.1	wifi-3	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
4	Device 4	123.13.33.1	wifi-4	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
5	Device 5	123.32.12.3	testwifii	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:41:43.23+00	2022-04-16 16:41:43.23+00
6	Device 6	123.32.12.22	wifif6	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:43:13.707+00	2022-04-16 16:43:13.707

```
-- 2565-04-16 23:42:30.8510
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.relnamespace where nsname='public' AND relname='schedule';
-- 2565-04-16 23:43:15.7758
SELECT * FROM "public"."devices" ORDER BY "id" LIMIT 300 OFFSET 0;
-- 2565-04-16 23:43:15.7770
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.relnamespace where nsname='public' AND relname='devices';
-- 2565-04-16 23:43:15.7788
SELECT * FROM "public"."schedule" ORDER BY "id" LIMIT 300 OFFSET 0;
-- 2565-04-16 23:43:15.7930
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.relnamespace where nsname='public' AND relname='schedule';

```

รูปที่ 2.37 ฐานข้อมูลหลังเพิ่มอุปกรณ์

- ทำการปิด Service Database เพื่อทดสอบว่าดาต้าที่เพิ่มเข้ามาจะไม่หายไป



รูปที่ 2.38 ทำการปิด Service Database

- ทำการเปิด Service Database และเชื่อมต่อ กับฐานข้อมูลใหม่อีกรัง

The screenshot shows the pgAdmin 4 interface with the 'Database 'database' is disconnected! Tap to reconnect...' message at the top. The left sidebar shows 'Functions' and 'Tables' with 'devices' selected. The main area displays the 'devices' table with the following data:

	devices										schedule	
	<b>id</b>	<b>name</b>	<b>ip_address</b>	<b>wifi_name</b>	<b>wifi_password</b>	<b>mode</b>	<b>status_relay_1</b>	<b>status_relay_2</b>	<b>status_relay_3</b>	<b>status_relay_4</b>	<b>createdAt</b>	<b>updatedAt</b>
1	Device 1	123.123.33.1	wifi-1	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
2	Device 2	123.13.33.1	wifi-2	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
3	Device 3	123.13.33.1	wifi-3	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
4	Device 4	123.13.33.1	wifi-4	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
5	Device 5	123.32.12.3	testwifi	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:41:43.23+00	2022-04-16 16:41:43.23+00
6	Device 6	123.32.12.22	wifi6	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:43:13.707+00	2022-04-16 16:43:13.707+00

Below the table, the SQL query is visible:

```

SELECT * FROM public.schedule ORDER BY "id" LIMIT 300 OFFSET 0;
-- 2565-04-16 23:42:39.0510
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='schedule';
-- 2565-04-16 23:43:15.7750
SELECT * FROM "public"."devices" ORDER BY "id" LIMIT 300;
-- 2565-04-16 23:43:15.7770
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='devices';
-- 2565-04-16 23:43:15.7890
SELECT * FROM "public"."schedule" ORDER BY "id" LIMIT 300;
-- 2565-04-16 23:43:15.7930
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='schedule';

```

At the bottom, there are buttons for Data, Structure, + Row, Columns, Filters, and a syntax highlighting checkbox.

รูปที่ 2.39 ฐานข้อมูลขณะเชื่อมต่อใหม่อีกรัง

- จะพบว่าข้อมูลยังอยู่แม้เราปิด Database เปิดใหม่

The screenshot shows the pgAdmin 4 interface with the 'Database 'database' is disconnected! Tap to reconnect...' message at the top. The left sidebar shows 'Functions' and 'Tables' with 'devices' selected. The main area displays the 'devices' table with the same data as in the previous screenshot:

	devices										schedule	
	<b>id</b>	<b>name</b>	<b>ip_address</b>	<b>wifi_name</b>	<b>wifi_password</b>	<b>mode</b>	<b>status_relay_1</b>	<b>status_relay_2</b>	<b>status_relay_3</b>	<b>status_relay_4</b>	<b>createdAt</b>	<b>updatedAt</b>
1	Device 1	123.123.33.1	wifi-1	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
2	Device 2	123.13.33.1	wifi-2	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
3	Device 3	123.13.33.1	wifi-3	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
4	Device 4	123.13.33.1	wifi-4	12345678	MANUAL	FALSE	TRUE	TRUE	FALSE	FALSE	2022-04-16 16:40:59.043+00	2022-04-16 16:40:59.043
5	Device 5	123.32.12.3	testwifi	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:41:43.23+00	2022-04-16 16:41:43.23+00
6	Device 6	123.32.12.22	wifi6	123456	MANUAL	FALSE	FALSE	FALSE	FALSE	FALSE	2022-04-16 16:43:13.707+00	2022-04-16 16:43:13.707+00

Below the table, the SQL query is visible:

```

SELECT * FROM public.schedule ORDER BY "id" LIMIT 300 OFFSET 0;
-- 2565-04-16 23:43:41.4280
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='schedule';
-- 2565-04-16 23:43:43.4030
SELECT * FROM "public"."devices" ORDER BY "id" LIMIT 300;
-- 2565-04-16 23:43:43.4050
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='devices';
-- 2565-04-16 23:43:43.4150
SELECT * FROM "public"."schedule" ORDER BY "id" LIMIT 300;
-- 2565-04-16 23:43:43.4170
select reltuples::int8 as count from pg_class c JOIN pg_catalog.pg_namespace n ON n.oid=c.renamespace where nspname='public'AND relname='schedule';

```

At the bottom, there are buttons for Data, Structure, + Row, Columns, Filters, and a syntax highlighting checkbox.

รูปที่ 2.40 ฐานข้อมูลหลังทำการปิด Service Database และเปิดใหม่