**บทที่ 3**

**วิธีการดำเนินงาน**

จากการศึกษาข้อมูลเกี่ยวกับการทำระบบบริหารจัดการร้านถวิลผ่านเว็บแอปพลิเคชัน ผู้จัดทำได้รวบรวมข้อมูลต่าง ๆ ซึ่งมีขั้นตอนการดำเนินงานดังต่อไปนี้

3.1 วิเคราะห์ระบบบริหารจัดการร้านถวิลผ่านเว็บแอปพลิเคชัน

3.2 การออกแบบแผนภาพกระแสการไหลของข้อมูล (Data Flow Diagrams)

3.3 พจนานุกรมข้อมูล (Data Dictionary)

3.4 ตัวอย่างขั้นตอนการทำงานของระบบ

**3.1 วิเคราะห์และออกแบบระบบฐานข้อมูล**

การวิเคราะห์และออกแบบโครงสร้างการดำเนินงานภายในระบบนั้นเป็นการวิเคราะห์ขั้นตอนเกี่ยวกับการรับรายการอาหาร และการจัดการสต๊อก รวมทั้งการวิเคราะห์ขั้นตอนต่าง ๆ ทั้งหมดที่เกิดขึ้นภายในระบบแสดงโดยใช้ผังงาน (Flow Chart) ได้ดังนี้

Diagram

Description automatically generated

**รูปที่ 3.1** ผังงานเบื้องต้นของระบบบริหารจัดการร้านถวิลผ่านเว็บแอปพลิเคชัน

จากรูปที่ 3.1 ผู้ใช้งานสามารถเข้าสู่ระบบโดยตรงผ่านหน้าเว็บแอปพลิเคชัน โดยระบบมีสิทธิ์ผู้ใช้ทั้งหมด 2 สิทธิ์ คือ สิทธิ์ของพนักงานรับOrder และพนักงานCashier

1. สิทธิ์ของพนักงานรับ Order

- รับรายการสั่งอาหาร

- ตรวจสอบสถานะรายการอาหารของแต่ละโต๊ะ

- ยกเลิกรายการอาหารที่สั่ง

- คำนวณค่าอาหาร

- เรียกดูข้อมูลรายการอาหาร

- เรียกดูข้อมูลสต๊อก

- เรียกดูข้อมูลโปรโมชั่น

- จัดการการย้ายโต๊ะ

- จัดการการรวมโต๊ะ

2. สิทธิ์ของพนักงาน Cashier

- จัดการข้อมูลรายการอาหาร และเครื่องดื่ม

- จัดการข้อมูลโปรโมชั่น

- จัดการข้อมูลวัตถุดิบ และต้นทุน

- ดูรายงานรายได้ย้อนหลัง รายวัน รายเดือน รายปี

- ดูรายงานวัตถุดิบคงเหลือ และต้นทุน ย้อนหลัง รายวัน รายเดือน รายปี

- ดูรายงานรายรับ - รายจ่าย รายวัน รายเดือน รายปี

- ตรวจสอบสถานะรายการอาหารของแต่ละโต๊ะ

- สรุปยอดรายการอาหารขายดี

- สามารถแสดงเมนูที่ขายดี และเมนูแนะนำ

- ออกใบเสร็จให้กับลูกค้า

- บันทึกการชำระค่าอาหาร

- ยกเลิกรายการอาหารที่สั่ง

- คำนวณค่าอาหาร

- จัดการการย้ายโต๊ะ

- จัดการการรวมโต๊ะ

3 สิทธิ์ของครัว

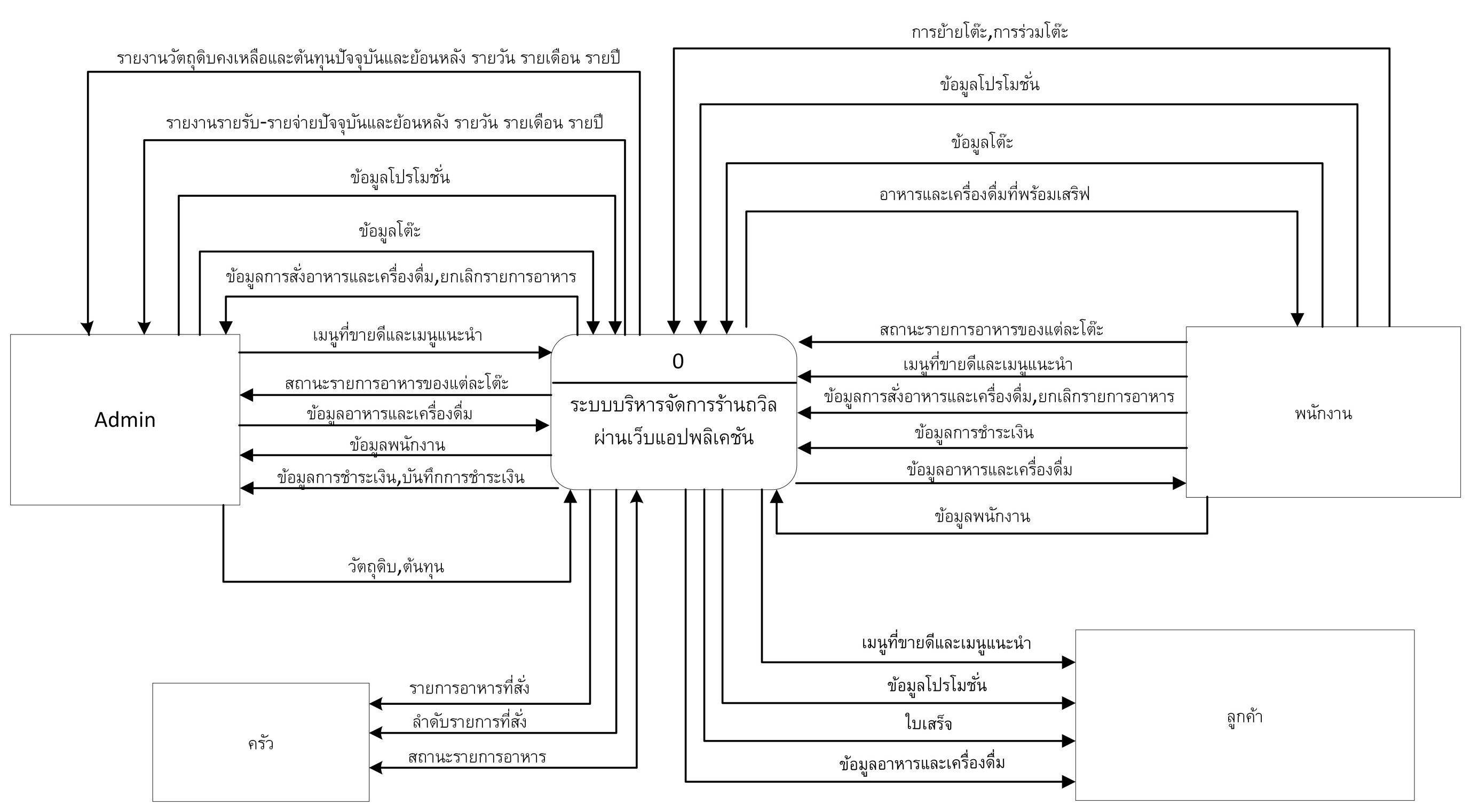
- ดูรายการอาหารที่สั่ง

- ดูลำดับรายการที่สั่ง

- รายงานอาหารที่ทำเสร็จ

**3.2 การออกแบบแผนภาพกระแสไหลของข้อมูล (Data Flow Diagrams)**

แผนภาพกระแสการไหลของข้อมูลระดับสูงสุดได้นําข้อมูลจากการวิเคราะห์ และออกแบบโครงสร้างระบบงาน มาใช้ในการออกแบบแผนภาพกระแสการไหลของข้อมูลระดับสูงสุดโดยอธิบายกระบวนการทํางาน ดังรูปที่ 3.2



**รูปที่ 3.2** แผนภาพกระแสการไหลของข้อมูลระดับสูงสุด (Context Diagram)

3.2.1 การออกแบบแผนภาพกระแสการไหลของข้อมูลระดับที่ 1

แผนภาพกระแสการไหลข้อมูล ได้นําข้อมูลการออกแบบแผนภาพกระแสการไหลของข้อมูลระดับสูงสุดมาใช้ในการออกแบบแผนภาพกระแสการไหลของข้อมูลระดับที่ 1 โดยอธิบายกระบวนการทํางานกระแสการไหลของข้อมูลแยกเป็นกระบวนการทํางานต่าง ๆ แสดงดังรูป 3.3 ดังนี้

กระบวนการที่ 1.0 กระบวนการข้อมูลพนักงาน ทําหน้าที่ในการเรียกดูเพิ่มลบแก้ไขข้อมูลพนักงาน

กระบวนการที่ 2.0 กระบวนการจัดการสต๊อก ทําหน้าที่ในการเรียกดูเพิ่มลบแก้ไขข้อมูลสต๊อก

กระบวนการที่ 3.0 กระบวนการเรียกดูข้อมูลในสต๊อก ทําหน้าที่ แสดงสถานะของสต๊อก

กระบวนการที่ 4.0 กระบวนการรับอาหารและเครื่องดื่ม ทําหน้าที่ในรับอาหารและเครื่องดื่มกับลูกค้า

กระบวนการที่ 5.0 กระบวนการรายงาน ทําหน้าที่ในการเรียกดู รายรับรายจ่าย ออกรายงานใบชําระเงิน สรุปยอดรายเดือน รายปี

Diagram

Description automatically generated

**รูปที่ 3.3** แผนภาพกระแสการไหลข้อมูลระดับที่ 1

**3.3 พจนานุกรมข้อมูล**

3.3.1 แผนภาพแสดงความสัมพันธ์เอนทิตีของระบบ

Diagram

Description automatically generated

**รูปที่ 3.4** แผนภาพความสัมพันธ์เอนทิตีของระบบ

3.3.2 พจนานุกรมโครงสร้างข้อมูล

จากแผนภาพความสัมพันธ์เอนทิตีในรูปที่ 3.4 นั้นได้แสดงแผนภาพความสัมพันธ์เอนทิตีออกเป็นตารางที่ใช้ในระบบ สามารถอธิบายลักษณะข้อมูลในแต่ละตาราง ดังต่อไปนี้

**ตารางที่ 3.1** พนักงาน (employee)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| emp\_ID | รหัสพนักงาน | Int(11) | No | PK |  |
| emp\_Name | ชื่อพนักงาน | Varchar(100) | No |  |  |
| emp\_Phone | เบอร์โทรพนักงาน | Varchar(10) | No |  |  |
| emp\_Username | ชื่อผู้ใช้งานพนักงาน | Varchar(45) | No |  |  |
| emp\_Password | รหัสเข้าใช้งานพนักงาน | Varchar(45) | No |  |  |
| emp\_Address | ที่อยู่พนักงาน | Varchar(255) | No |  |  |
| emp\_Slip | สลิปโอนเงิน | Varchar(255) | Yes |  |  |
| emp\_Line | ไอดีไลน์พนักงาน | Varchar(45) | No |  |  |

ตารางที่ 3.1 ตารางพนักงาน ใช้เก็บข้อมูลพนักงานภายในร้าน คอลัมน์ emp\_ID ใช้เก็บรหัสพนักงาน ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ emp\_Name ใช้เก็บชื่อของพนักงานร้าน ประเภทข้อมูลเป็น Varchar ไม่อนุญาติให้มีค่าว่าง, คอลัมน์ emp\_Phone ใช้เก็บเบอร์โทรของพนักงาน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ emp\_Username ใช้เก็บชื่อผู้ใช้งานของพนักงาน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ emp\_Password ใช้เก็บรหัสเข้าใช้งานของพนักงาน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ emp\_Address ใช้เก็บที่อยู่ของพนักงาน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ emp\_Slip ใช้เก็บสลิปโอนเงิน ประเภทข้อมูลเป็น Varchar อนุญาตให้มีค่าว่าง, คอลัมน์ emp\_Line ใช้เก็บไอดีไลน์พนักงาน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง

**ตารางที่ 3.2** สิทธิ์ของพนักงาน (employee\_role)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| empRole\_ID | รหัสสถานะพนักงาน | Int(1) | No | PK |  |
| emp\_ID | รหัสพนักงาน | Int(11) | No | FK | emp\_ID  (emp\_ID) |

ตารางที่ 3.2 ตารางสิทธิ์ของพนักงาน ใช้เก็บสิทธิ์ของพนักงาน คอลัมน์ empRole\_ID ใช้เก็บรหัสสิทธิ์ของพนักงาน ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ emp\_ID ใช้เก็บรหัสพนักงาน ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจาก ตารางที่ 3.1 ตารางพนักงาน คอลัมน์ emp\_ID

**ตารางที่ 3.3** โต๊ะภายในร้าน (table)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| table\_ID | รหัสโต๊ะ | Int(11) | No | PK |  |
| table\_Zone | พื้นที่โต๊ะ | Varchar(45) | No |  |  |

ตารางที่ 3.3 ตารางโต๊ะภายในร้าน ใช้เก็บข้อมูลของโต๊ะภายในร้าน คอลัมน์ Table\_ID ใช้เก็บรหัสโต๊ะ ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ Table\_Zone ใช้เก็บพื้นที่โต๊ะ หรือตำแหน่งที่อยู่ของโต๊ะ ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง

**ตารางที่ 3.4** แคชเชียร์ (cashier)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| cashier\_ID | รหัสแคชเชียร์ | Int(11) | No | PK |  |

ตารางที่ 3.4 ตารางแคชเชียร์ ใช้เก็บรหัสแคชเชียร์ คอลัมน์ cashier\_ID ใช้เก็บรหัสแคชเชียร์ ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก

**ตารางที่ 3.5** ออเดอร์รวม (total\_order)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| totalOrder\_ID | รหัสออเดอร์รวม | Int(11) | No | PK |  |
| totalPrice | จำนวนเงินรวม | Int(11) | No |  |  |
| totalOrder\_TimeStamp | เวลาที่คิดเงิน | DateTime | No |  |  |
| totalOrder\_Status | สถานะการคิดเงิน | Varchar (45) | No |  |  |

**ตารางที่ 3.5** ออเดอร์รวม (total\_order) (ต่อ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| cashier\_ID | รหัสแคชเชียร์ | Int(11) | No | FK | cashier\_ID (cashier\_ID) |
| table\_ID | รหัสโต๊ะ | Int(11) | No | FK | table\_ID (table\_ID) |

ตาราง 3.5 ตารางออเดอร์รวม ใช้เก็บรายการสั่งอาหารต่าง ๆ ที่พนักงานรับเข้ามา คอลัมน์ totalOrder\_ID ใช้เก็บรหัสออเดอร์รวม ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ totalPrice ใช้เก็บจำนวนเงินรวม ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ totalOrder\_TimeStamp ใช้เก็บเวลาที่คิดเงิน ประเภทข้อมูลเป็น DateTime ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ totalOrder\_Status ใช้เก็บสถานการณ์คิดเงิน ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ cashier\_ID ใช้เก็บรหัสแคชเชียร์ ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจาก ตารางที่ 3.4 ตารางแคชเชียร์ คอลัมน์ cashier\_ID, คอลัมน์ table\_ID ใช้เก็บรหัสโต๊ะ ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจาก ตารางที่ 3.3 ตารางโต๊ะ คอลัมน์ table\_ID

**ตารางที่ 3.6** สต๊อกสินค้า (stock)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| stock\_ID | รหัสสต๊อก | Int(11) | No | PK |  |
| stock\_Name | ชื่อสินค้า | Varchar(50) | No |  |  |
| stock\_Qty | จำนวนสินค้า | Int(10) | No |  |  |
| stock\_Cost | ต้นทุนสินค้า | Int(10) | No |  |  |
| stock\_Min | จำนวนแจ้งเตือน | Int(5) | No |  |  |
| stock\_Status | สถานะสินค้า | Varchar(50) | No |  |  |
| stock\_TimeStamp | เวลาที่ตัดสินค้า | DateTime | No |  |  |

ตารางที่ 3.6 ตารางสต๊อกสินค้า ใช้เก็บข้อมูลวัตถุดิบต่าง ๆ ที่อยู่ในร้าน คอลัมน์ stock\_ID ใช้เก็บรหัสสต๊อกสินค้า ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ stock\_Name ใช้เก็บชื่อสินค้า ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stock\_Qty ใช้เก็บจำนวนสินค้า ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stock\_Cost ใช้เก็บต้นทุนของสินค้า ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stock\_Min ใช้เก็บค่าของสต๊อกที่ต่ำที่สุดที่ต้องการให้แจ้งเตือน ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง , คอลัมน์ stock\_Status ใช้เก็บสถานะของสินค้า ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, stock\_TimeStamp ใช้เก็บเวลาของสินค้า ประเภทข้อมูลเป็น DateTime ไม่อนุญาตให้มีค่าว่าง

**ตารางที่ 3.7** เมนูสินค้า และอาหาร (menu)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| menu\_ID | รหัสเมนู | Int(11) | No | PK |  |
| menu\_Name | ชื่อเมนู | Varchar(50) | No |  |  |
| menu\_Price | ราคาเมนู | Int(11) | No |  |  |
| menu\_Pic | รูปภาพเมนู | Varchar(255) | No |  |  |
| menu\_Qty | จำนวนเมนู | Int(10) | No |  |  |
| menu\_Cost | ต้นทุนเมนู | Int(10) | No |  |  |
| menu\_Status | สถานะเมนู | Varchar(45) | No |  |  |
| menu\_TimeStamp | เวลาที่สั่งเมนู | DateTime | No |  |  |
| typeMenu\_ID | ชนิดของเมนู | Int(11) | No | FK | typeMenu\_ID  (typeMenu\_ID) |

ตารางที่ 3.7 ตารางเมนูสินค้า และอาหาร ใช้เก็บข้อมูลเมนูต่าง ๆ คอลัมน์ menu\_ID ใช้เก็บรหัสของเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ menu\_Name ใช้เก็บชื่อเมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_Price ใช้เก็บราคาเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_Pic ใช้เก็บรูปภาพเมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_Qty ใช้เก็บจำนวนของเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_Cost ใช้เก็บต้นทุนของเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_Status ใช้เก็บสถานะของเมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_TimeStamp ใช้เก็บเวลาที่สั่งเมนู ประเภทข้อมูลเป็น DateTime ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ typeMenu\_ID ใช้เก็บชนิดของเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจากตารางที่ 3.8 ตารางชนิดเมนู คอลัมน์ typeMenu\_ID

**ตารางที่ 3.8** ชนิดเมนู (type\_Menu)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| typeMenu\_ID | รหัสชนิดเมนู | Int(11) | No | PK |  |
| typeMenu\_Name | ชื่อชนิดเมนู | Varchar(45) | No |  |  |

ตารางที่ 3.8 ตารางชนิดเมนู ใช้เก็บข้อมูลของชนิดเมนูภายในร้าน คอลัมน์ typeMenu\_ID ใช้เก็บรหัสชนิดเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ typeMenu\_Name ใช้เก็บชื่อชนิดเมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง

**ตารางที่ 3.9** สต๊อกสินค้า และเมนู (stock\_Menu)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| stockMenu\_ID | รหัสสต๊อกเมนู | Int(11) | No | PK |  |
| stockMenu\_Qty | จำนวนสต๊อกเมนู | Int(10) | No |  |  |
| stockMenu\_Status | สถานะสต๊อกเมนู | Varchar(45) | No |  |  |
| stockMenu\_TimeStamp | เวลาที่เข้าถึงสต๊อก | DateTime | No |  |  |
| stock\_ID | รหัสสต๊อก | Int(11) | No | FK | stock\_ID (stock\_ID) |
| menu\_ID | รหัสเมนู | Int(11) | No | FK | menu\_ID (menu\_ID) |

ตารางที่ 3.9 ตารางสต๊อกสินค้า และเมนู ใช้ในการเรียกดูข้อมูลวัตถุดิบจากตารางสต๊อกสินค้า เพื่อมารวมเป็น หนึ่งเมนู ในตารางของเมนู คอลัมน์ stockMenu\_ID ใช้เก็บรหัสสต๊อกเมนู ประเภทข้อมูล Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ stockMenu\_Qty ใช้เก็บจำนวนของสต๊อกเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stockMenu\_Status ใช้แสดงสถานะของสต๊อกเมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stockMenu\_TimeStamp ใช้เก็บเวลาที่เข้าถึงสต๊อกเมนู ประเภทข้อมูลเป็น DateTime ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ stock\_ID ใช้เก็บข้อมูล รหัสสต๊อก ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจาก ตารางที่ 3.6 ตารางสต๊อกสินค้า คอลัมน์ stock\_ID, คอลัมน์ Menu\_ID ใช้เก็บข้อมูลรหัสเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจากตารางที่ 3.7 ตารางเมนูสินค้า และอาหาร คอลัมน์ menu\_ID

**ตารางที่ 3.10** ออเดอร์เมนู (order\_menu)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ชื่อคอลัมน์ | คำอธิบาย | ประเภทข้อมูล | ค่าว่าง | คีย์ | หมายเหตุ |
| order\_ID | รหัสออเดอร์เมนู | Int(11) | No | PK |  |
| order\_Qty | จำนวนออเดอร์เมนู | Int(10) | No |  |  |
| orderMenu\_Status | สถานะออเดอร์เมนู | Varchar(45) | No |  |  |
| orderMenu\_TimeStamp | เวลาที่สั่งเมนู | DateTime | No |  |  |
| menu\_ID | รหัสเมนู | Int(11) | No | FK | menu\_ID (menu\_ID) |
| emp\_ID | รหัสพนักงาน | Int(11) | No | FK | emp\_ID (emp\_ID) |
| totalOrder\_ID | รหัสออเดอร์รวม | Int(11) | No | FK | totalOrder\_ID  (totalOrder\_ID) |

ตารางที่ 3.10 ออเดอร์เมนู ใช้เก็บข้อมูลพนักงาน แต่ละคนสั่งเมนูอะไร คอลัมน์ order\_ID ใช้เก็บรหัสออเดอร์เมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง ให้เป็นคีย์หลัก, คอลัมน์ order\_Qty ใช้เก็บจำนวนออเดอร์เมนูที่สั่ง ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ orderMenu\_Status ใช้เก็บสถานะออเดอร์เมนู ประเภทข้อมูลเป็น Varchar ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ orderMenu\_TimeStamp ใช้เก็บเวลาที่สั่งเมนู ประเภทข้อมูลเป็น DateTime ไม่อนุญาตให้มีค่าว่าง, คอลัมน์ menu\_ID ใช้เก็บข้อมูลรหัสเมนู ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจากตารางที่ 3.7 ตารางเมนูสินค้า และอาหาร คอลัมน์ menu\_ID, คอลัมน์ emp\_ID ใช้เก็บรหัสพนักงาน ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจากตารางที่ 3.1 ตารางพนักงาน คอลัมน์ emp\_ID, คอลัมน์ totalOrder\_ID ใช้เก็บรหัสออเดอร์รวม ประเภทข้อมูลเป็น Int ไม่อนุญาตให้มีค่าว่าง เป็นคีย์นอกจาก ตารางที่ 3.5 ตารางออเดอร์รวม คอลัมน์ totalOrder\_ID

**3.4 ตัวอย่างขั้นตอนการทำงานของระบบ**

การออกแบบหน้าจัดการพนักงานและการจัดการสต๊อกของระบบ คือหน้าของ Admin

ประกอบด้วยหน้าจอเมนูย่อย ๆ ดังนี้

3.4.1หน้าจอจัดการพนักงาน

ส่วนหน้าจอที่ใช้ในการจัดการพนักงาน โดยสามารถ เพิ่ม,ลบ,แก้ไข,ค้นหา ได้

โดยสิทธิ์ที่จะเข้ามาหน้าจัดการพนักงานได้จะต้องเป็น Admin ของระบบเท่านั้น

Graphical user interface

Description automatically generated

**รูปที่ 3.5 หน้าจอส่วนจัดการพนักงาน**

3.4.2 หน้าจอส่วนของการเพิ่มพนักงาน

ส่วนหน้าจอแสดงการเพิ่มพนักงาน เป็นส่วนที่เพิ่มข้อมูลพนักงานต่างๆ เช่น สิทธิ์ของพนักงาน ชื่อ Username Password เบอร์โทร ที่อยู่ Line ของพนักงาน

Table

Description automatically generated

**รูปที่ 3.6 หน้าจอส่วนของการเพิ่มพนักงาน**

3.4.3 หน้าจอส่วนของการแก้ไขพนักงาน

ส่วนหน้าจอแสดงการแก้ไขพนักงาน เป็นส่วนที่แก้ไขข้อมูลพนักงานต่างๆ เช่น สิทธิ์ของพนักงาน ชื่อ Username Password เบอร์โทร ที่อยู่ Line ของพนักงาน โดยที่ดึงข้อมูลที่เคยเพิ่มมาแล้วมาแก้ไข

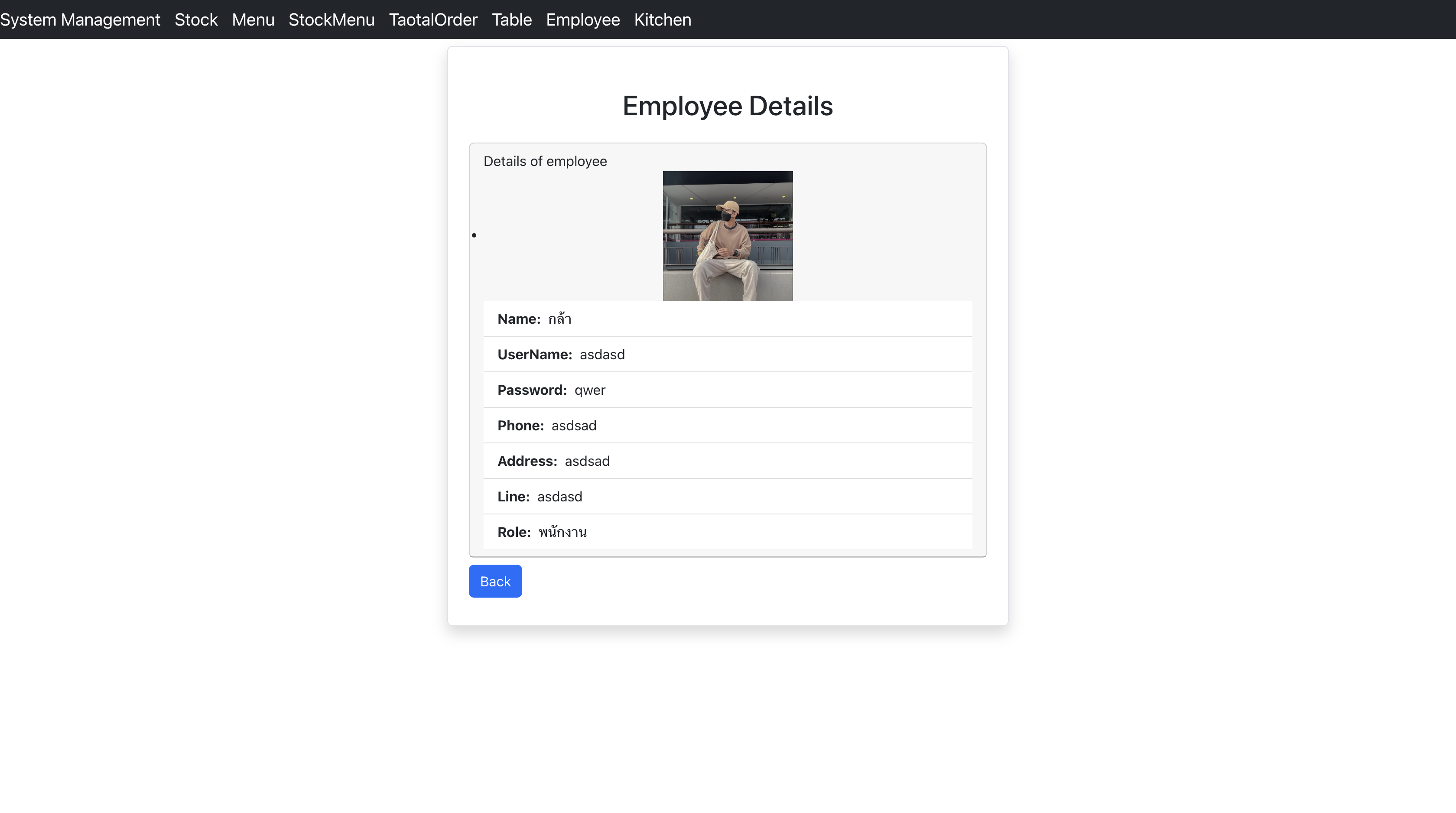
Graphical user interface, application

Description automatically generated

**รูปที่ 3.7 หน้าจอส่วนของการแก้ไขพนักงาน**

3.4.4 หน้าจอส่วนของการแสดงข้อมูลพนักงาน

ส่วนหน้าจอแสดงข้อมูลพนักงาน เป็นส่วนที่แสดงข้อมูลพนักงานต่างๆ เช่น สิทธิ์ของพนักงาน ชื่อ Username Password เบอร์โทร ที่อยู่ Line ของพนักงาน



**รูปที่ 3.8 หน้าจอส่วนของการแสดงข้อมูลพนักงาน**

3.4.5 หน้าจอส่วนของการแสดงสต๊อก

ส่วนหน้าจอแสดงสต๊อก โดยสามารถ เพิ่ม,ลบ,แก้ไข,ค้นหา ได้ และแสดงข้อมูลสต๊อก ต่างๆ เช่น ชื่อประเภท จำนวน ต้นทุน จำนวนการแจ้งเตือนสต๊อกเหลือน้อย วันและเวลา

Graphical user interface, application

Description automatically generated

**รูปที่ 3.9 หน้าจอส่วนของการแสดงสต๊อก**

3.4.6 หน้าจอส่วนของการเพิ่มสต๊อก

ส่วนหน้าจอการเพิ่มสต๊อก เป็นในส่วนการเพิ่มสต๊อกต่างๆ เช่น ชื่อประเภท จำนวน ต้นทุน จำนวนการแจ้งเตือนสต๊อกเหลือน้อย วันและเวลา

**Graphical user interface, application

Description automatically generated**

**รูปที่ 3.10 หน้าจอส่วนของการเพิ่มสต๊อก**

3.4.7 หน้าจอส่วนของการจัดการเมนู

ส่วนหน้าจอแสดงการจัดการเมนู โดยสามารถ เพิ่ม,ลบ,แก้ไข,ค้นหา ได้ และแสดงข้อมูลเมนูต่างๆ เช่น ชื่อเมนู ราคา จำนวน ต้นทุน สถานะ ประเภทเมนู วันและเวลา

**Graphical user interface, application, table

Description automatically generated**

**รูปที่ 3.11 หน้าจอส่วนของการจัดการเมนู**

3.4.8 หน้าจอส่วนของการเพิ่มเมนู

ส่วนหน้าจอการเพิ่มเมนู เป็นในส่วนการเพิ่มเมนูต่างๆ เช่น ชื่อเมนู ราคา จำนวน ต้นทุน สถานะ ประเภทเมนู วันและเวลา

Graphical user interface

Description automatically generated

**รูปที่ 3.12 หน้าจอส่วนของการเพิ่มเมนู**

3.4.9 หน้าจอส่วนของการจัดการสต๊อกเมนู

ส่วนหน้าจอแสดงการจัดการสต๊อกเมนู สามารถดูส่วนผสมของเมนูได้ โดยสามารถ เพิ่ม,ลบ,แก้ไข

ค้นหา ได้ และแสดงข้อมูลสต๊อกเมนูต่างๆ เช่น จำนวน สถานะ วันและเวลา ชื่อเมนู ชื่อสต๊อก

Graphical user interface, application

Description automatically generated

**รูปที่ 3.13 หน้าจอส่วนของการจัดการสต๊อกเมนู**

3.4.10 หน้าจอส่วนของการเพิ่มสต๊อกเมนู

ส่วนหน้าจอการเพิ่มสต๊อกเมนู เป็นในส่วนการเพิ่มสต๊อกเมนูต่างๆ เช่น จำนวน สถานะ วันและเวลา ชื่อเมนู ชื่อสต๊อก

**Graphical user interface, application

Description automatically generated**

**รูปที่ 3.14 หน้าจอส่วนของการเพิ่มสต๊อกเมนู**

3.4.11 หน้าจอส่วนของโต๊ะ

ส่วนหน้าจอแสดงการจัดการโต๊ะ สามารถดูโต๊ะโดยสามารถ เพิ่ม,ลบ,ย้ายโต๊ะ,รวมโต๊ะ

,เปิดโต๊ะ,ค้นหา ได้ และสามารถสั่งอาหารได้

Diagram

Description automatically generated

**รูปที่ 3.15 หน้าจอส่วนของโต๊ะ**

3.4.12 หน้าจอส่วนหน้าแคชเชียร์

ส่วนหน้าจอแสดงการจัดการโต๊ะ สามารถดูโต๊ะโดยสามารถ เพิ่ม,ลบ,ย้ายโต๊ะ,รวมโต๊ะ

,เปิดโต๊ะ,ค้นหาได้ และสามารถดูสถานะของแต่ละโต๊ะว่าชำระเงินหรือไม่

**Graphical user interface, application

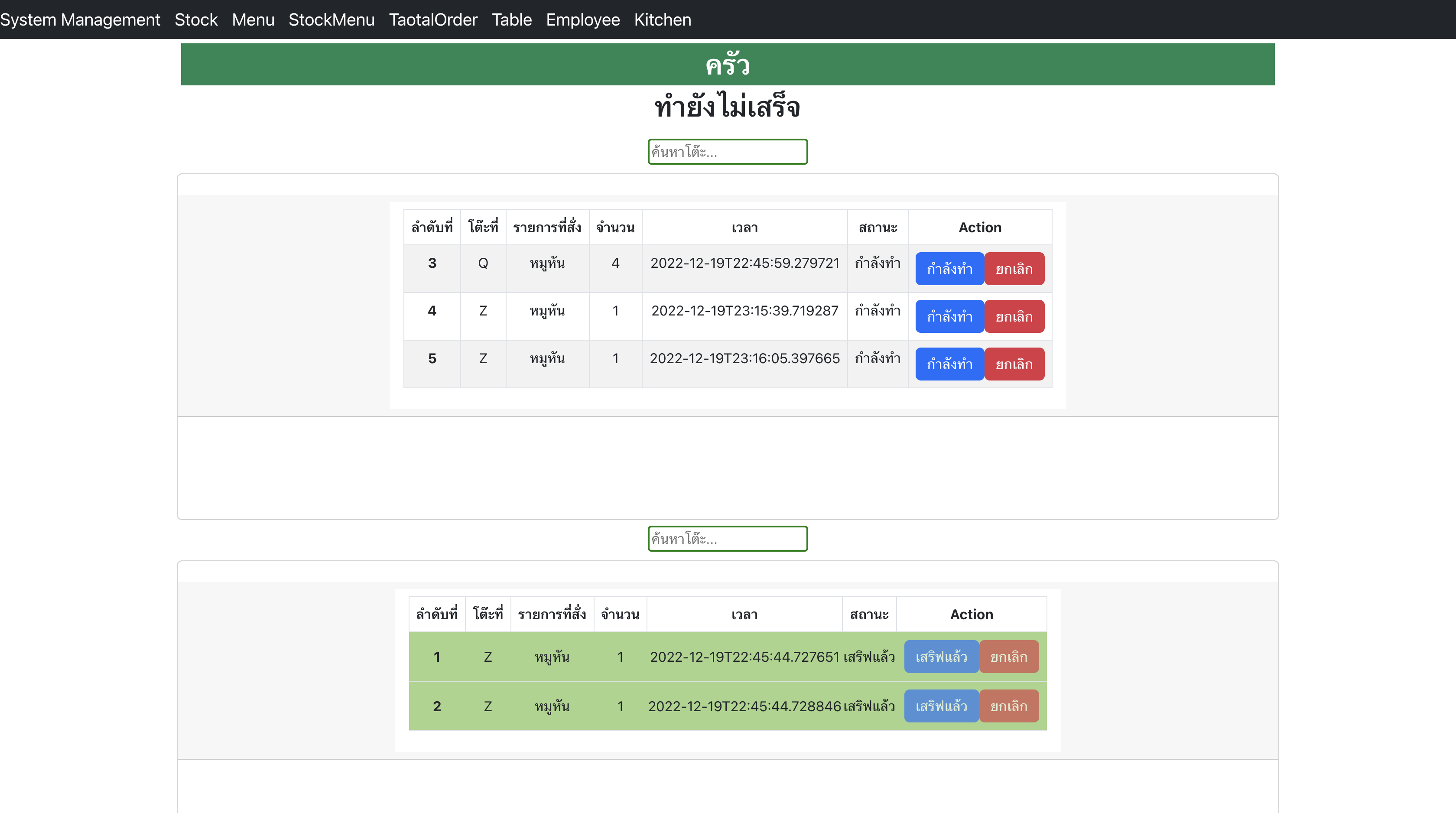
Description automatically generated**

**รูปที่ 3.16 หน้าจอส่วนของแคชเชียร์**

3.4.13 หน้าจอส่วนหน้าพนักงานครัว

ส่วนหน้าจอแสดงการจัดการโต๊ะ สามารถดูโต๊ะโดยสามารถ เพิ่ม,ลบ,ย้ายโต๊ะ,รวมโต๊ะ

,เปิดโต๊ะ,ค้นหา ได้ และสามารถดูสถานะของแต่ละโต๊ะว่าชำระเงินหรือไม่



**รูปที่ 3.17 หน้าจอส่วนของแคชเชียร์**

**โปรแกรมที่ 3.1** การสร้างเทเบิลดาต้าเบสด้วย Spring Boot

1. @Entity

2. @Table(name = "employee")

3. public class Employee {

4. @Id

5. @GeneratedValue(strategy = GenerationType.IDENTITY)

6. private long id;

7. @Column(name = "name\_Emp")

8. private String name\_Emp;

9. @Column(name = "username")

10. private String username;

11. @Column(name = "password")

12. private String password;

13. @Column(name = "phone")

14. private String phone;

15. @Column(name = "address")

16. private String address;

17. @Column(name = "line")

18. private String line;

19. public Employee() { }

20. @ManyToOne

21. @JoinColumn(name = "idRole")

22. private Role idRole;

**โปรแกรมที่ 3.2** การสร้างเทเบิลดาต้าเบสด้วย Spring Boot

1. @RequestMapping("/api/v1/")

2. public class EmployeeController {

3. @Autowired

4. private EmployeeRepository employeeRepository;

5. @GetMapping("/employees")

6. public List<Employee> getAllEmployees(){

7. return employeeRepository.findAll(); }

8. @PostMapping("/employees")

9. public Employee createEmployee(@RequestBody Employee employee) {

10. return employeeRepository.save(employee); }

11. @GetMapping("/employees/{id}")

12. public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

13. Employee employee = employeeRepository.findById(id)

14. .orElseThrow(() -> new ResourceNotFoundException("Employee not 15exist with 15. id :" + id));

16. return ResponseEntity.ok(employee);

**โปรแกรมที่ 3.2** การสร้าง API ด้วย Spring Boot

(ต่อ)

17. @PutMapping("/employees/{id}")

18. public ResponseEntity<Employee> updateEmployee(@PathVariable Long id,

19. @RequestBody Employee employeeDetails){

20. Employee employee = employeeRepository.findById(id)

21. .orElseThrow(() -> new ResourceNotFoundException("Employee not

22. exist with id :" + id));

23. employee.setName\_Emp(employeeDetails.getName\_Emp());

24. employee.setUsername(employeeDetails.getUsername());

25. employee.setPassword(employeeDetails.getPassword());

26. employee.setPhone(employeeDetails.getPhone());

27. employee.setAddress(employeeDetails.getAddress());

28. employee.setLine(employeeDetails.getLine());

29. employee.setIdRole(employeeDetails.getIdRole());

30. Employee updatedEmployee = employeeRepository.save(employee);

31. return ResponseEntity.ok(updatedEmployee); }

32. @DeleteMapping("/employees/{id}")

33. public ResponseEntity<Map<String, Boolean>> deleteEmployee(@PathVariable 34. Long id){

35. Employee employee = employeeRepository.findById(id)

36. .orElseThrow(() -> new ResourceNotFoundException("Employee not

37. exist with id :" + id));

38. employeeRepository.delete(employee);

39. Map<String, Boolean> response = new HashMap<>();

40. response.put("deleted", Boolean.TRUE);

41. return ResponseEntity.ok(response); }}

**โปรแกรมที่ 3.3** การสร้างหน้าเรียกใช้ API ด้วย React

1. import axios from 'axios'

2. const EMPLOYEE\_BASE\_REST\_API\_URL = 'http://localhost:8080/api/v1/employees';

3. class EmployeeService{

4. getAllEmployees(){

5. return axios.get(EMPLOYEE\_BASE\_REST\_API\_URL)

6. }

7. createEmployee(employee){

8. return axios.post(EMPLOYEE\_BASE\_REST\_API\_URL, employee)

9. }

10. getEmployeeById(employeeId){

11. return axios.get(EMPLOYEE\_BASE\_REST\_API\_URL + '/' + employeeId);

12. }

13. updateEmployee(employeeId, employee ){

14. return axios.put(EMPLOYEE\_BASE\_REST\_API\_URL + '/' +employeeId,

15. employee );

16. }

17. deleteEmployee(employeeId){

18. return axios.delete(EMPLOYEE\_BASE\_REST\_API\_URL + '/' + employeeId);

19. }

20. }

21. export default new EmployeeService();

**โปรแกรมที่ 3.3** การเรียกใช้ API แสดงข้อมูลพนักงาน ด้วย React

1.const getEmployeeId = (employeeId) => {

2. EmployeeService.getEmployeeById(employeeId).then(response => {

3. setEmployee(response.data);

4. }).catch(error => {

5. console.log('Something went wrong', error);

6. });

7. }

8. EmployeeService.createEmployee(employee).then((response) => {

9. console.log(response.data)

10. console.log(response.data.role)

11. history('/employees');

12. }).catch(error => {

13. console.log(employee)

14. console.log(employee.idRole)

15. console.log(error)

16. })

17. EmployeeService.updateEmployee(id,employee).then((response) => {

18. history('/employees')

19. console.table(employee)

20. }).catch(error => {

21. console.log(employee.idRole)

22. console.log(error)

23. })

24. const deleteEmployee = (employeeId) => {

25. if (window.confirm('Are you sure you want to delete this employee?')) {

26. EmployeeService.deleteEmployee(employeeId).then((response) => {

27. getAllEmployees();

28. }).catch(error => {

29. console.log(error); })}

}

**โปรแกรมที่ 3.4** การสร้างการย้ายโต๊ (หลังบ้าน)

1. @Query(value = "SELECT \* from table\_tawin A left join total\_order B on
2. A.table\_id = B.table\_id where B.table\_id is null", nativeQuery = true)
3. List<TableTawin> GetmoveTableTawins();

4. @Modifying

5. @Transactional

6. @Query(value = "UPDATE total\_order SET table\_id = :table\_id WHERE

7. total\_order\_id = :total\_order\_ID ", nativeQuery = true)

8. public int moveTableTawins(@Param("table\_id") Integer table\_id,

9. @Param("total\_order\_ID") Integer total\_order\_ID);

10. @GetMapping("/moveTable/{table\_id}/{total\_order\_ID}")

11. public int mergeTable(@PathVariable int table\_id,@PathVariable int

12. total\_order\_ID) {

13. return (int) tableRepo.moveTableTawins(table\_id,total\_order\_ID);

14. }

15. @GetMapping("/getmoveTable")

16. public List<TableTawin> getmoveTable(){

17. return (List<TableTawin>) tableRepo.GetmoveTableTawins();

18. }

**โปรแกรมที่ 3.5** การสร้างการย้ายโต๊ (หน้าบ้าน)

1.const Movetable = (pointTable, totalOrder\_ID) => {

2. if (pointTable == null) {

3. window.alert("กรุณาเลือกโต๊ะที่ต้องการย้าย!!")

4. } else {

5. TableService.MoveTable(pointTable, totalOrder\_ID).then(() => {

6. })

7. OrderMenuService.mergeTable(totalOrder\_ID, pointTable).then(() => {

8. setShowMove(false);

9. getAllTotalOrder();

10. GetMoveTable();

11. })}}

12.const MoveTable = (table,totalOrder\_ID) =>{

13. return axios.get("http://localhost:8080/table/moveTable/" + table + "/" +

14. totalOrder\_ID);

15. }

16. const mergeTable = (totalOrder\_ID,pointTable) =>{

17. return

18. axios.get

19. ("http://localhost:8080/OrderMenu/mergeTable/"+totalOrder\_ID+"/"+point.Table);

20. }

**โปรแกรมที่ 3.6** การสร้างการรวมโต๊ (หลังบ้าน)

1. @Modifying

2. @Transactional

3. @Query(value = "UPDATE total\_order\n"

4. + "SET total\_order.compo\_site = :pointTable\n"

5. + "WHERE total\_order.total\_order\_id = :totalOrder\_ID", nativeQuery = true)

6. public int mergeTable(@Param("totalOrder\_ID") int totalOrder\_ID,

7. @Param("pointTable") int pointTable);

8. @Query(value = "SELECT \* "+ "FROM total\_order "

9. + "WHERE table\_id NOT IN (SELECT table\_id FROM total\_order WHERE table\_id = 10. :table\_ID) AND table\_id = compo\_site", nativeQuery = true)

11. List<Total\_Order> findByMixTable\_ID(@Param("table\_ID") Integer table\_ID);

12. @GetMapping("/getMixTable/{table\_ID}")

13. public List<Total\_Order> getMixtable(@PathVariable Integer table\_ID) {

14. return (List<Total\_Order>) totalOrderRepo.findByMixTable\_ID(table\_ID);}

15. @GetMapping("/mergeTable/{totalOrder\_ID}/{pointTable}")

16. public int mergeTable(@PathVariable int totalOrder\_ID,

17. @PathVariable int pointTable) {

18. orderMenuRepo.mergeTable(totalOrder\_ID,pointTable);

19.return pointTable;

20. }

**โปรแกรมที่ 3.6** การสร้างการรวมโต๊ (หน้าบ้าน)

1. const getMixTebles = () => {

2. TableService.MixTable(data?.table\_ID?.table\_ID).then(response => {

3. setTabel(response.data);

4. console.log("mix", response.data);

5. }).catch(error => {

6. console.log('Something went wrong', error);});}

7. const mixTable = (totalOrder\_ID) => {

8. if (pointTable == null) {

9. window.alert("กรุณาเลือกโต๊ะที่ต้องการรวม!!")

10. } else {

11. OrderMenuService.mergeTable(totalOrder\_ID, pointTable).then(() => {

12. setShowMix(false);

13. getAllTotalOrder();

14.GetMoveTable(); })}}

15. const MixTable = (table\_ID) => {

16. return axios.get("http://localhost:8080/totalOrder/getMixTable/"+ table\_ID);

17. };

18. const mergeTable = (totalOrder\_ID,pointTable) =>{

19. return

20. axios.get

21. ("http://localhost:8080/OrderMenu/mergeTable/"+totalOrder\_ID+"/"+pointTable);

22. }

**จากโปรแกรมที่ 3.1** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1 : สร้างตาราง

บรรทัดที่ 2 : กำหนดชื่อตาราง

บรรทัดที่ 3 : สร้างคลาส Employee

บรรทัดที่ 4-18 : สร้าง Attribute ในตาราง

บรรทัดที่ 19 : สร้าง Constructor

บรรทัดที่ 20 : สร้าง Relationship ManyToOne

บรรทัดที่ 21 : Join กับ Column ชื่อ idRole

บรรทัดที่ 22 : สร้างตัวแปร

บรรทัดที่ 27 : Attribute ชื่อ idRole

**จากโปรแกรมที่ 3.2** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1 : กำหนดการเข้าถึง API

บรรทัดที่ 2 : สร้างคลาส EmployeeController

บรรทัดที่ 3-4 : การกำหนดให้ Attribute ของ Class EmployeeRepository

บรรทัดที่ 5 : กำหนดการเข้าถึง API แบบ GetMapping เพื่อนเรียกดูข้อมูลทั้งหมดใน Data base

บรรทัดที่ 6-7 : เรียกดูข้อมูลทั้งหมด

บรรทัดที่ 8-10 : เพิ่มข้อมูลลงใน Data base

บรรทัดที่ 11-16 : เรียกดูข้อมูลเฉพาะ id

บรรทัดที่ 17-31 : เป็นการ Update ข้อมูลลงใน Data base

บรรทัดที่ 32-41 : ลบข้อมูลใน Data base

**จากโปรแกรมที่ 3.3** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1 : import axios

บรรทัดที่ 2 : สร้างตัวแปรเพื่อเก็บ url ของ API

บรรทัดที่ 3: สร้างคลาส EmployeeService

บรรทัดที่ 4-6 : สร้างฟังก์ชัน getAllEmployees เพื่อแสดงข้อมูลทั้งหมด

บรรทัดที่ 7-9 : สร้างฟังก์ชัน createEmployee เพื่อเพิ่มข้อมูลพนักงาน

บรรทัดที่ 10-12 : เสร้างฟังก์ชัน getEmployeeById เพื่อแสดงข้อมูลพนักงาน

บรรทัดที่ 13-16 : สร้างฟังก์ชัน updateEmployee เพื่อแก้ไขข้อมูล

บรรทัดที่ 17-20 : สร้างฟังก์ชัน deleteEmployee เพื่อลบข้อมูล

**จากโปรแกรมที่ 3.3** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1-7 : สร้างฟังก์ชันเพื่อเรียกดูข้อมูลพนักงาน โดยเรียกใช้ฟังก์ชัน getEmployeeById เพื่อแสดงที่หน้าเว็บ

บรรทัดที่ 8-16 : สร้างฟังก์ชันเพื่อเพิ่มข้อมูลพนักงาน โดยเรียกใช้ฟังก์ชัน createEmployee

บรรทัดที่ 17-23: สร้างฟังก์ชันเพื่อแก้ไขข้อมูลพนักงาน โดยเรียกใช้ฟังก์ชัน updateEmployee

บรรทัดที่ 24-29 : สร้างฟังก์ชันเพื่อลบข้อมูลพนักงาน โดยเรียกใช้ฟังก์ชัน deleteEmployee

**จากโปรแกรมที่ 3.4** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1-3 : คิวรี่หาโต๊ะที่สามารถที่ยังไม่ถูกเปิด โดยจะใช้ left join total\_order B on 2. A.table\_id = B.table\_id where B.table\_id is null เพื่อแสดงว่าโต๊ะไหนที่ยังไม่ถูกเปิด

บรรทัดที่ 4-9 : คิวรี่อัปเดตการย้ายโต๊ะ

บรรทัดที่ 10-14 : สร้าง API เส้นย้ายโต๊ะ โดยจะไปเรียกใช้คิวรี่อัปเดตการย้ายโต๊ะ

บรรทัดที่ 15-18 : สร้าง API เส้นแสดงว่าโต๊ะไหนที่สามารถย้ายได้

**จากโปรแกรมที่ 3.5** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1-6 : เป็นการส่งข้อมูลทั้งหมดไปหลังบ้าน โดยจะมีการเช็คก่อนว่า มีการเลือกโต๊ะไปหรือยังบรรทัดที่ 7-11 : เป็นการส่งข้อมูลทั้งหมดไปหลังบ้าน ว่ามีการอัปเดตการย้ายโต๊ะไปแล้ว บรรทัดที่ 12-15 : เรียกใช้ API เส้นย้ายโต๊ะ จากหลังบ้าน บรรทัดที่ 16-20 : เรียกใช้ API เส้นแสดงว่าโต๊ะไหนที่สามารถย้ายได้

**จากโปรแกรมที่ 3.6** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1-5 : คิวรี่อัปเดตการรวมโต๊ะ บรรทัดที่ 6-11 : เป็นการคิวรี่เพื่อแสดงว่าโต๊ะไหนสามารถรวมได้ โดยโต๊ะที่สามารถรสมได้ต้องเป็นโต๊ะที่เปิดแล้วและต้องไม่ใช่โต๊ะตัวเอง บรรทัดที่ 12-14 : สร้าง API เส้นแสดงโต๊ะที่รวมได้ บรรทัดที่ 16-20 : สร้าง API เส้นการรวมโต๊ะ

**จากโปรแกรมที่ 3.7** สามารถอธิบายการทำงานได้ดังนี้

บรรทัดที่ 1-6 : เป็นการเรียกใช้ API เส้น แสดงว่าโต๊ะไหนสามารถรวมได้ยกเว้นโต๊ะตัวเอง บรรทัดที่ 7-14 : เป็นการส่งข้อมูลทั้งหมดไปหลังบ้าน ว่ามีการรวมโต๊ะ บรรทัดที่ 15-17 : เรียกใช้ API เส้นรวมโต๊ะ จากหลังบ้าน บรรทัดที่ 18-22 : เรียกใช้ API การรวมโต๊ะได้