



โครงงาน

Order Managing System

จัดทำโดย

6704062612197 นายพาทิศ เจริญแพทย์

เสนอ

ผู้ช่วยศาสตราจารย์สฤติ ประสมพันธ์

วิชา 040613204 Object-Oriented Programming

ภาคเรียนที่ 1 ปีการศึกษา 2568

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เกี่ยวกับโครงการ

ชื่อโปรเจกต์: Order Managing System

นำเสนอโดย: นายพาทิศ เจริญแพทย์

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

Source Code: <https://github.com/PT22315/WebProjectOOP>

Website: <https://oder-managing.onrender.com/> (เว็บอาจจะใช้เวลาเปิดประมาณ 2-3 นาที)

บทที่ 1 ที่มาและความสำคัญของโครงการ

โครงการนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object-Oriented Programming โดยการนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบเว็บ โดยใช้แนวทางการเขียนโปรแกรมแบบเชิงวัตถุ และยังช่วยให้ผู้จัดทำเรียนรู้อุปกรณ์และเครื่องมือ ผู้จัดทำได้สร้างเว็บขึ้นมา

ประเภทของโครงการ

โปรแกรมเว็บแอปพลิเคชัน Full-Stack

ประโยชน์

1. เพื่อนำแนวทางการเขียนโปรแกรมแบบเชิงวัตถุมาประยุกต์ใช้
2. จำลองระบบร้านอาหารเบื้องต้นสำหรับพนักงาน

ขอบเขตของโครงการ

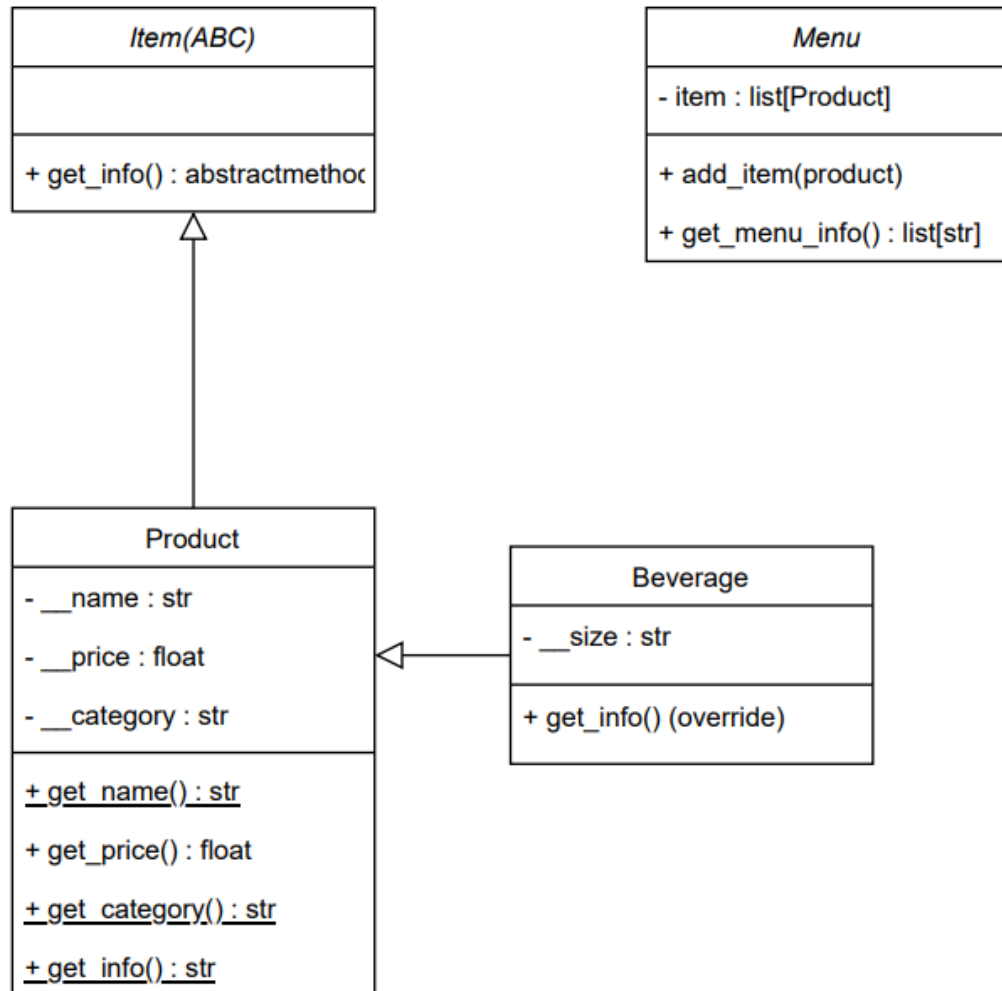
1. ทำขึ้นเพื่อให้พนักงานในร้านอาหารใช้เพื่อรับออเดอร์ได้สะดวกขึ้นซึ่งมีฟังก์ชันพื้นฐานดังนี้
 - เมนูอาหาร แสดงรายการอาหารจากฐานข้อมูล
 - เพิ่ม/แก้ไข/ลบเมนู
 - เพิ่มสินค้าในลิส
 - ลิสสินค้า ดูสรุปสินค้าของลูกค้าและราคา
 - ยืนยันออเดอร์ เพื่อเข้าสู่สถานะ active order
 - active order แสดงรายการอาหารของลูกค้าที่ยังนั่งกินอยู่ในร้านแล้วยังไม่ได้ชำระเงิน
 - ประวัติออเดอร์ แสดงออเดอร์ที่ทำการชำระเงินแล้ว
 - ล็อกอิน พนักงานล็อกอินด้วย account เดียว (admin / 1234)

2.ตารางเวลาการดำเนินโครงการ(Project Schedule)

งาน	สัปดาห์ที่ 1 (ต.ค.)	สัปดาห์ที่ 2 (ต.ค.)	สัปดาห์ที่ 3 (ต.ค.)	สัปดาห์ที่ 3 (ต.ค.)
1.ติดตั้งและออกแบบระบบ				
2.เขียนโปรแกรม				
3.นำเข้าเว็บ				
4.จัดทำเอกสาร				

บทที่ 2 การพัฒนา

แผนภาพ Class Diagram



โครงสร้างนี้จะมีคลาสหลักอยู่สามคลาสคือ

1. คลาส Product จะเก็บข้อมูลของสินค้าต่างๆเช่น

- ชื่อสินค้า
- ราคาสินค้า
- หมวดหมู่ของสินค้า

และจะมี Methods ต่างๆดังนี้

- `get_name()` ใช้เรียกชื่อสินค้า
- `get_price()` ใช้เรียกราคาสินค้า

- get_category() ใช้เรียกหมวดหมู่ของสินค้า

- get_info() คืนข้อความสรุปสรุปของสินค้า โดยจะคืนค่าเป็น name , category , price บาท

2. คลาส Beverage เป็น subclass ของ Product มีการสืบทอดข้อมูลคลาส Product มาและมี

- size เพิ่มมา เป็นขนาดของเครื่องดื่มเช่น (small, medium, large)

และมี Method get_info() ที่ override จากเมธอดเดิมโดยทำหน้าที่คืนค่าข้อความสรุปที่รวมขนาดของเครื่องดื่มเข้าไป

ด้วยเช่น โค้ก small 20 บาท

3. คลาส Menu เก็บสินค้าทั้งหมดในเมนู

- item : list [Product]

และมี Methods ดังต่อไปนี้

- add_item() เพิ่มสินค้า product ลงในเมนู

- get_menu_info() คืนรายการข้อความสรุปสินค้าในเมนู

รูปแบบการพัฒนา

- ภาษา : Python , HTML , CSS , JavaScript

- Framework : Flask

- ผู้ให้บริการโฮสติ้ง: Render

- ฐานข้อมูล: SQLite

แนวทางการเขียนโปรแกรมเชิงวัตถุ

- Constructor

```
29 class Product(Item):
30     def __init__(self, name, price, category):
31         self.__name = name
32         self.__price = price
33         self.__category = category
34
```

คลาส Product มี constructor เพื่อรับ Parameters ต่างๆ

-Encapsulation

```
29 class Product(Item):
30     def __init__(self, name, price, category):
31         self.__name = name
32         self.__price = price
33         self.__category = category
34
```

Attribute ของคลาส Product จะประกาศแบบเป็น private ซึ่งหมายความว่า จะไม่สามารถเข้าถึงโดยตรงจากภายนอก ต้องเข้าผ่านเมธอด getter เช่น `get_name()`)

-Composition

```
71 class OrderDB(db.Model):
72     __tablename__ = 'orders'
73     id = db.Column(db.Integer, primary_key=True)
74     table_number = db.Column(db.Integer, nullable=False) # โต๊ะที่ลูกค้านั่ง
75     total_price = db.Column(db.Float, default=0.0)
76     status = db.Column(db.String(20), default='active') # active = กำลังสั่ง, paid = ชำระเงินแล้ว
77
78     # Relationship (Composition): 1 order มีหลาย OrderItem
79     items = db.relationship('OrderItemDB', backref='order', cascade="all, delete")
80
```

OrderDB ประกอบด้วย OrderItemDB หลายตัว เมื่อมีการลบ Order หนึ่งอัน จะลบ OrderItem ทั้งหมดที่อยู่ภายในด้วย โดย OrderItemDB คือรายการสินค้าที่รับในออเดอร์นั้นๆ

-Polymorphism

```
59 class Beverage(Product):
60     def __init__(self, name, price, size="Medium"):
61         super().__init__(name, price, "เครื่องดื่ม")
62         self.__size = size
63
64     # Polymorphism: override get_info
65     def get_info(self):
66         return f"{self.get_name()} ({self.__size}) - {self.get_price()} บาท"
67
```

คลาส Beverage สืบทอดมาจากคลาส Product แต่มีการ Override เมธอด `get_info()` ทำให้แสดงผลต่างออกไป เป็นหลักการ Polymorphism เนื่องจากใช้ชื่อเมธอดเดียวกันแต่มีพฤติกรรมต่างกัน

-Abstract

```
25 from abc import ABC, abstractmethod
26
27 class Item(ABC):
28     @abstractmethod
29     def get_info(self):
30         pass
31
```

คลาส Item สืบทอดจาก ABC (Abstract Base Class) มีเมธอด get_info() ที่เป็น abstract method หมายความว่า คลาสที่สืบทอดจากคลาส Item ไป จะต้อง implement เมธอดนี้เอง

-Inheritance

```
31
32 class Product(Item):
33
59 class Beverage(Product):
60     def __init__(self, name, price, size="Medium"):
61         super().__init__(name, price, "เครื่องดื่ม")
62         self.__size = size
63
```

คลาส Product สืบทอดจากคลาส Item ทำให้ได้ abstract เมธอด get_info() และคลาส Beverage ก็สืบทอดจากคลาส Product ทำให้ได้ get_name(), get_price() ที่เดิมอยู่ในคลาส Product และใช้ super() เพื่อเรียก constructor จากคลาสแม่

อัลกอริทึมที่สำคัญ

```
def checkout():
    table_number = session.get("table_number")
    if not table_number:
        flash("โปรดเลือกโต๊ะก่อนสั่งอาหาร", "error")
        return redirect(url_for("index"))

    # ใ้ cart จาก session JS (POST มาจาก hidden input) หรือ session['cart']
    import json
    cart_data = request.form.get("cart_data")
    if cart_data:
        cart = json.loads(cart_data)
    else:
        cart = session.get('cart', {})

    if not cart:
        flash("ตะกร้าว่าง", "error")
        return redirect(url_for("index"))

    # เช็คว่ามี order active ของโต๊ะนี้หรือยัง
    order = OrderDB.query.filter_by(table_number=table_number, status="active").first()
    if not order:
        order = OrderDB(table_number=table_number, status="active")
        db.session.add(order)
        db.session.commit()

    total_price = order.total_price or 0

    for name, info in cart.items():
        qty = info['qty']
        product = ProductDB.query.filter_by(name=name).first()
        if product:
            # เช็คว่ามี item นี้อยู่ใน order แล้วหรือยัง
            item = OrderItemDB.query.filter_by(order_id=order.id, product_id=product.id).first()
            if item:
                item.quantity += qty
            else:
                item = OrderItemDB(order_id=order.id, product_id=product.id, quantity=qty)
                db.session.add(item)
            total_price += product.price * qty

    order.total_price = total_price
    db.session.commit()
```

อัลกอริทึมการยืนยันออเดอร์สินค้า เป็นอัลกอริทึมหลักของระบบเนื่องการเราจะต้องรับออเดอร์ของลูกค้าก่อนไม่งั้นจะดำเนินการไปจุดอื่นของระบบไม่ได้ โดยอัลกอริทึมนี้มีหลักการทำงานดังนี้

- 1.ต้องเลือกโต๊ะก่อนถึงจะสามารถรับออเดอร์จากลูกค้าได้
- 2.ตรวจสอบว่าลิสสินค้าว่างหรือไม่ ถ้าไม่มีสินค้าเลย จะไม่อนุญาตให้ checkout
- 3.ตรวจสอบว่าออเดอร์ของโต๊ะนั้นๆเคยสั่งไว้ก่อนหน้านี้หรือไม่ มีไว้สำหรับกรณีลูกค้าอยากสั่งอาหารเพิ่มทีหลัง
- 4.มีการบอกจำนวนสินค้าและราคารวมสะสมของสินค้าในออเดอร์

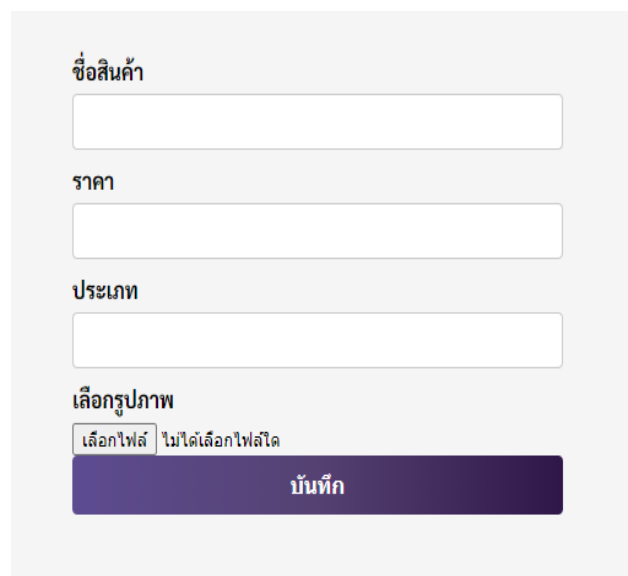
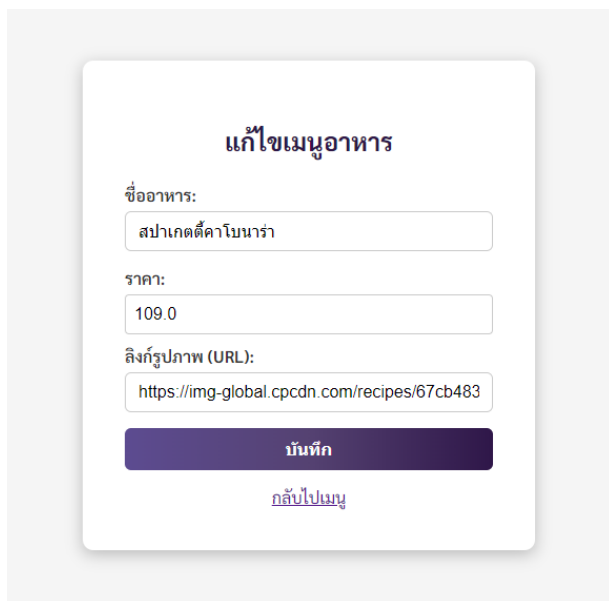
บทที่ 3 สรุป

ปัญหาที่พบระหว่างการพัฒนา

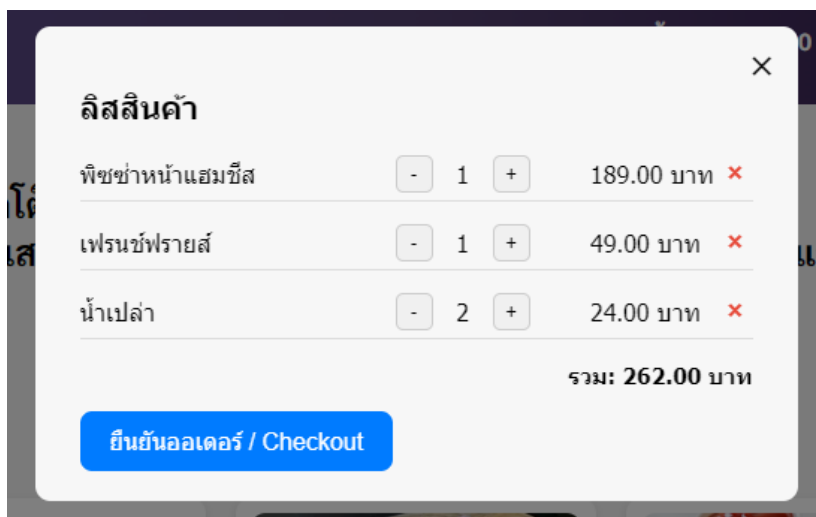
1. เนื่องจากการขาดความรู้และประสบการณ์ ทำให้การเขียนโค้ดพบerrorระหว่างทางบ่อย
2. บางฟังก์ชันยังทำงานได้ไม่ดีพอในบางจุดและอาจทำให้ระบบมีช่องโหว่

จุดเด่นของโปรแกรม

1. พนักงานสามารถเพิ่ม/แก้ไข/ลบสินค้าได้



2. เมื่อรับออเดอร์เสร็จจะมีหน้าสรุปออเดอร์ของลูกค้าให้



3.เมื่อยืนยันแล้วออเดอร์จะเข้าสู่สถานะ active order เป็นสถานะที่สั่งอาหารแล้วแต่ยังไม่ชำระเงินลูกค้ายังสามารถสั่งอาหารเพิ่มระหว่างนี้ได้

Active Orders					Home	Order History
โต๊ะ	รายการสินค้า		ราคารวม	สถานะ	จัดการ	
2	<ul style="list-style-type: none">•••	พิซซ่าหน้าแฮมชีส x1 = 189.0 บาท เฟรนช์ฟรายส์ x1 = 49.0 บาท น้ำเปล่า x2 = 24.0 บาท	262.0 บาท	active	<button>ชำระเงิน</button>	

4.มีข้อมูลออเดรีย้อนหลังให้ดู

Order History					Home	Active Orders
โต๊ะ	รายการสินค้า		ราคารวม	สถานะ		
5	<ul style="list-style-type: none">••••	เฟรนช์ฟรายส์ x4 = 196.0 บาท สปาเกตตี้คาโบนาร่า x3 = 327.0 บาท โค้ก x2 = 40.0 บาท พิซซ่าหน้าแฮมชีส x2 = 378.0 บาท	1144.0 บาท	paid		
5	<ul style="list-style-type: none">•••	เฟรนช์ฟรายส์ x3 = 147.0 บาท โค้ก x3 = 60.0 บาท สปาเกตตี้คาโบนาร่า x1 = 109.0 บาท	414.0 บาท	paid		
5	<ul style="list-style-type: none">•	สปาเกตตี้คาโบนาร่า x2 = 218.0 บาท	300.0 บาท	paid		
4	<ul style="list-style-type: none">•••	สปาเกตตี้คาโบนาร่า x2 = 218.0 บาท เฟรนช์ฟรายส์ x1 = 49.0 บาท โค้ก x1 = 20.0 บาท	388.0 บาท	paid		
5	<ul style="list-style-type: none">•••	เฟรนช์ฟรายส์ x3 = 147.0 บาท สปาเกตตี้คาโบนาร่า x1 = 109.0 บาท โค้ก x2 = 40.0 บาท	385.0 บาท	paid		
5	<ul style="list-style-type: none">•••	สปาเกตตี้คาโบนาร่า x2 = 218.0 บาท เฟรนช์ฟรายส์ x2 = 98.0 บาท โค้ก x2 = 40.0 บาท ผักโขมอบชีส x1 = 49.0 บาท	525.0 บาท	paid		

คำแนะนำสำหรับผู้สอนที่อยากให้อธิบายหรือที่เรียนแล้วไม่เข้าใจหรืออยากให้เพิ่มสำหรับน้องๆรุ่นต่อไป

อยากให้อาจารย์มีGuideLineในแลปข้อต่างๆว่าควรเริ่มประมาณไหนหรือว่าอัลกอริทึมหลักของข้อนั้นๆว่าใช้แนวคิด logicประมาณไหนและอยากให้ช่วยสอนพื้นฐาน HTML ให้ด้วยคับ;-;