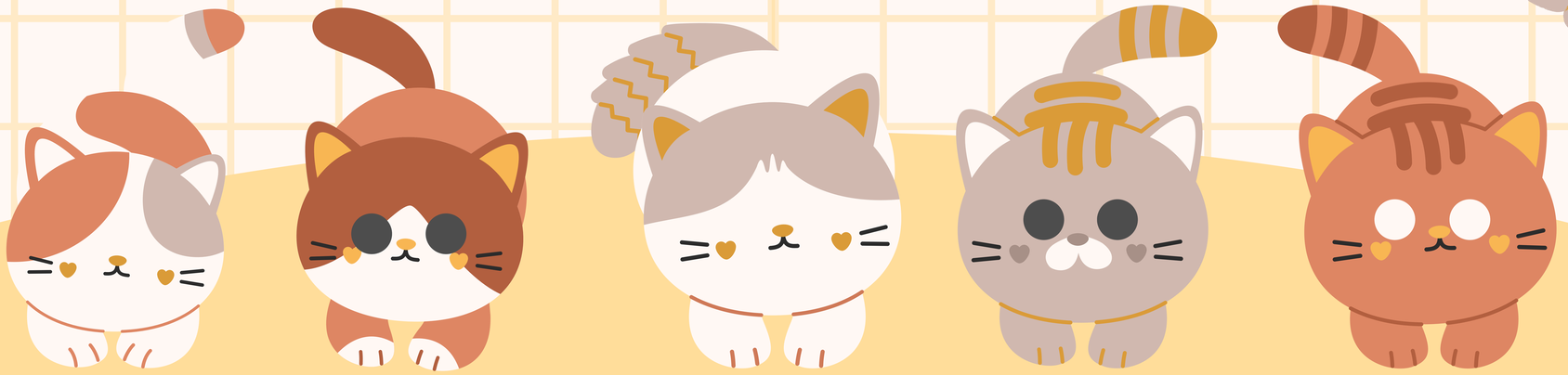
PETSHOP

PET SHOP ONLINE



+66123-456-7890

KhonKaen,Thailand

www.myshop.com

Member

นายธนพนธ์ รวิภาสกร
643020610-1

นายธนากร ผาเป้า
643020613-5

นางสาวสุพัตรา แพงจันทร์
643021343-3

ที่มาและความสำคัญ

ในปัจจุบัน ผู้คนนิยมเลี้ยงสัตว์มากขึ้น ส่งผลให้ความต้องการอาหารสัตว์ก็เพิ่มสูงขึ้นตามไป ด้วยเว็ปไซต์ขายอาหารสัตว์จึงเป็นช่องทางที่สะดวกในการเลือกดูและซื้อสินค้าผ่านทางเว็ปไซต์ขายอาหารโดยเว็ปไซต์โดยสามารถเลือกดูและเลือกซื้อ

ทฤษฎีที่เกี่ยวข้อง

1

Spring Boot

Spring Boot เป็น Open-Source Framework ที่ช่วยในการออกแบบแอพพลิเคชันและ Microservices อย่างรวดเร็วและมีความสะดวก ด้วย Auto Configuration และความเสถียรที่มีการพัฒนาต่อเนื่อง เป็นเครื่องมือที่ทันสมัยสำหรับการพัฒนาซอฟต์แวร์ในปี 2022 ได้มีการอัพเดทเป็น Spring Boot 3.0 ด้วยความเพิ่มเติมในความสะดวกสบายและประสิทธิภาพ

2

SOLID: The First 5 Principles of Object-Oriented Design

S – Single-Responsibility principle

O – Open/Closed principle

L – Liskov Substitution principle

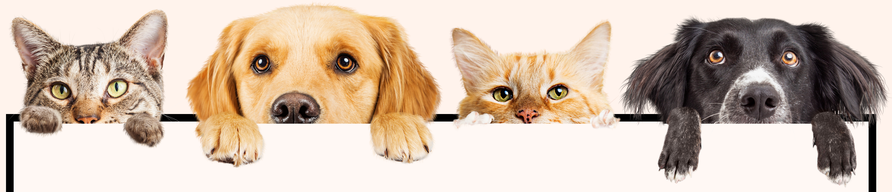
I – Interface Segregation principle

D – Dependency Inversion principle

Pattern in Spring boot

รูปแบบการออกแบบสำคัญใน Spring Boot ประกอบด้วย Singleton pattern (สร้างอินสแตนซ์เดียว), Factory method pattern (สร้างวัตถุโดยไม่เปิดเผยตรรกะ), Proxy pattern (ควบคุมการเรียกวัตถุ), Template method pattern (กำหนดโครงร่างของอัลกอริทึม), และ Dependency injection (DI) ที่ช่วยในการจัดการ Dependencies และทำให้รหัสมีประสิทธิภาพและเวิร์คส่งและทดสอบได้ง่ายขึ้น

เครื่องมือเกี่ยวข้อง



eclipse

MySQL™

ตัวอย่างการนำหลัก SOLID และ pattern ใน Spring มาประยุกต์ใช้

Single Responsibility Principle (SRP) แต่ละคลาสควรมีหน้าที่รับผิดชอบเพียงอย่างเดียว

- คลาส Product มีหน้าที่เก็บข้อมูลสินค้า
- คลาส Customer มีหน้าที่เก็บข้อมูลลูกค้า
- คลาส Category มีหน้าที่เก็บประเภทของสินค้า
- คลาส Repository มีหน้าที่เข้าถึงข้อมูลจากฐานข้อมูล
- คลาส Service มีหน้าที่ประมวลผลข้อมูล
- คลาส Controller มีหน้าที่จัดการคำขอ HTTP

ตัวอย่างการนำหลัก SOLID และ pattern ใน Spring มาประยุกต์ใช้

Interface Segregation Principle (ISP) อินเตอร์เฟซไม่ควรมีเมธอดมากเกินไป

- คลาส Repository อิมพลีเมนท์อินเทอร์เฟซ IRepository ซึ่งแยกหน้าที่ในการเข้าถึงข้อมูลออกจากโมเดลข้อมูล

ตัวอย่างการนำหลัก SOLID และ pattern ใน Spring มาประยุกต์ใช้

หลัก SOLID ช่วยให้ระบบมีโครงสร้างที่ชัดเจนและแยกหน้าที่ของแต่ละส่วนออกจากกัน

- ช่วยให้ระบบสามารถขยายหรือปรับเปลี่ยนได้ง่าย
- ช่วยให้ระบบมีความปลอดภัยมากขึ้น
- ช่วยให้ระบบมีความสามารถในการบำรุงรักษาที่ดีขึ้น

ฐานข้อมูล MySQL

Table Name: category

Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expressio
id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

ตาราง category

Table Name: customer

Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_user	INT	☑	☑	☐	☐	☐	☐	☐	☐	
name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
telephone	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
email	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
id_address	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

ตาราง customer

ฐานข้อมูล MySQL

Table Name: product

Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
productId	INT	☑	☑	☐	☐	☐	☐	☐	☐	
productName	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
price	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
category_id	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
quantity	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
image	TINYBLOB	☐	☐	☐	☐	☐	☐	☐	☐	NULL
image_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
product_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
product_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

ตาราง product

```java

@Entity
@Table(name="category")
public class Category {
    @Id
    //@GeneratedValue(strategy=GenerationType.AUTO)

    @Column(name="id")
    private int id;
    @Column(length=45, nullable=false, unique=true)
    private String name;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @OneToMany(targetEntity=Product.class,mappedBy="category",
            cascade=CascadeType.ALL,fetch = FetchType.LAZY)
    private List<Product> products;
    public List<Product> getProducts() {
        return products;
    }
    public void setProducts(List<Product> products) {
        this.products = products;
    }
```

คลาส Category

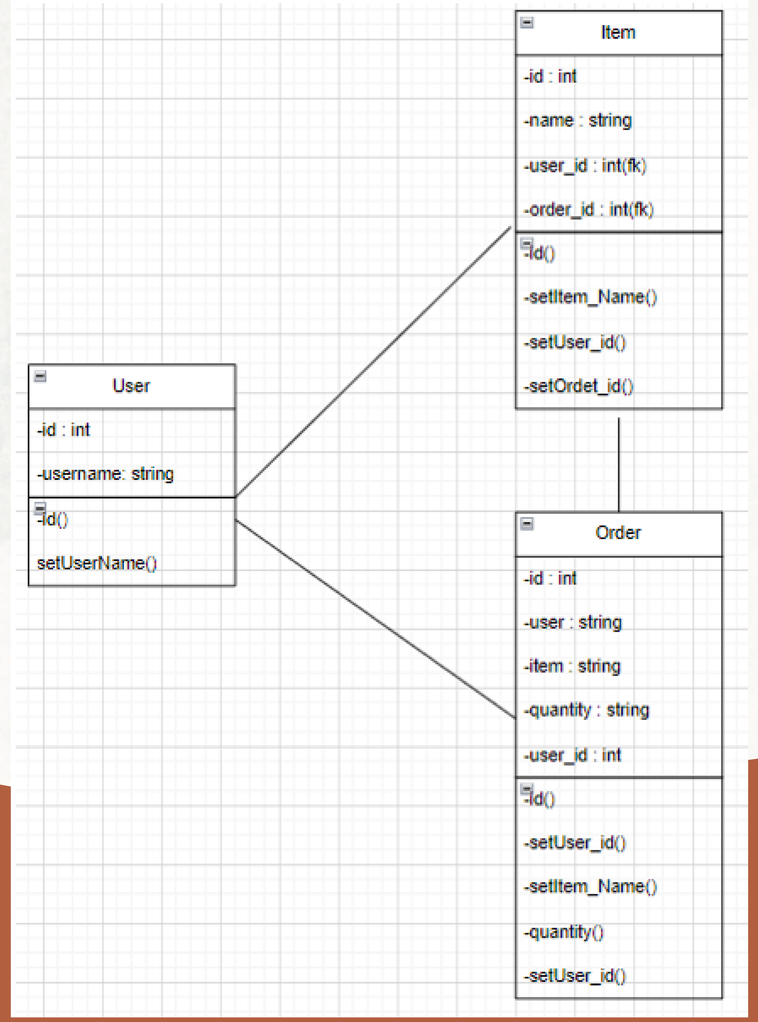
คลาส InventoryService

```java
package com.example.demo;

import java.util.List;

@Service
public class InventoryService {
    @Autowired
    private CategoryRepository catRepo;
    @Autowired
    private ProductRepository productRepo;
    @Autowired
    private CustomerRepository customRepo;
    @Autowired
    public InventoryService() {}
    //@Autowired //Constructor dependency injection
    public InventoryService(CategoryRepository catRepo, ProductRepository productRepo, CustomerRepository customRepo) {
        this.catRepo = catRepo;
        this.productRepo = productRepo;
        this.customRepo  = customRepo;
    }

    public List<Category> getAllCategory(){
        List<Category> cats = (List<Category>) this.catRepo.findAll();
        return cats;
    }
    public List<Product> getAllProduct(){
        List<Product> Product = (List<Product>) this.productRepo.findAll();
        return Product;
    }
    public List<customer> getAllCustomer(){
        List<customer> customer = (List<customer>) this.customRepo.findAll();
        return customer;
    }
}
```

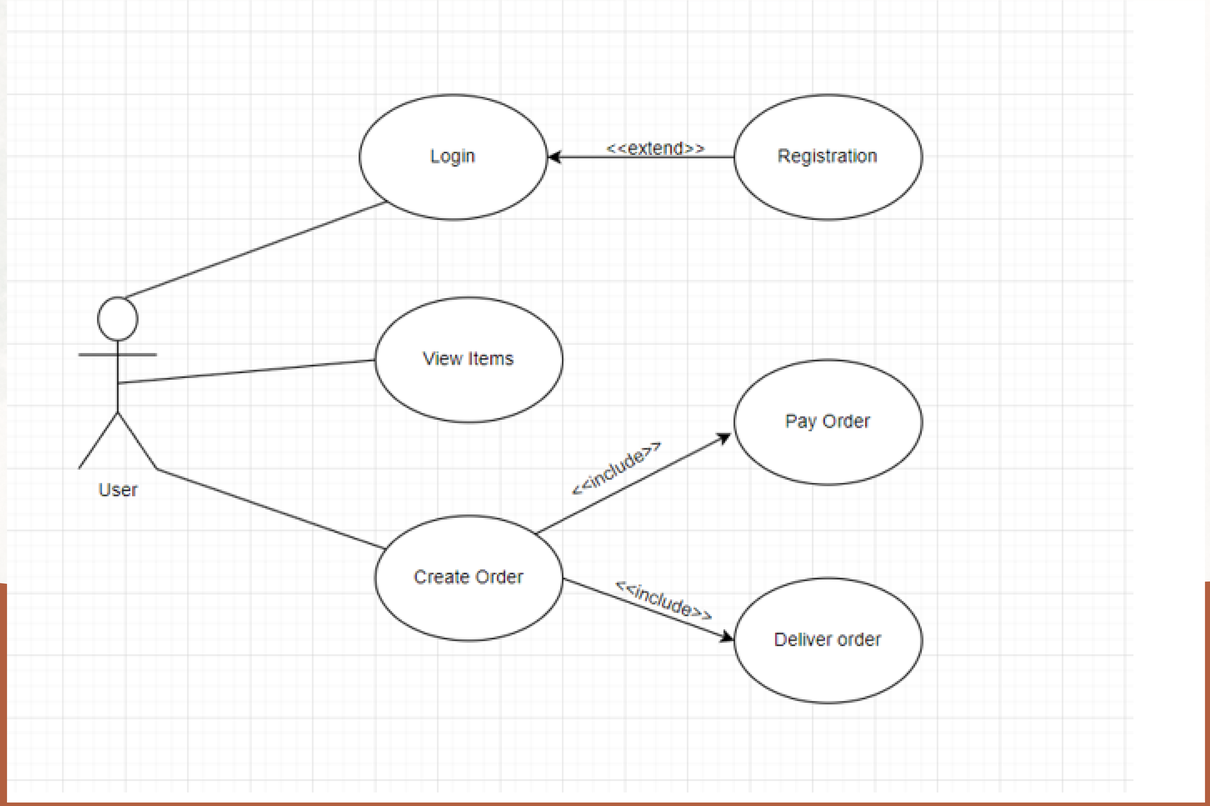
คลาส Controller

```
40              return "admin";
41          }
42
43      @GetMapping("/categoryList")
44          public String getListOfCateryList (Model model) {
45          List<Category> categorys= invSer.getAllCategory();
46          model.addAttribute("categorys",categorys);
47              return "categoryList";
48          }
49
50
51      @GetMapping("/list_product")
52          public String  getproduct(Model model) {
53           List<Product> Product = invSer.getAllProduct();
54           model.addAttribute("products",Product);
55              return "productList";
56          }
57
58
59      @GetMapping("/customer")
60          public String getcustomer(Model model) {
61          List<customer> customer= invSer.getAllCustomer();
62          model.addAttribute("customer",customer);
63              return "customerList";
64          }
65      @GetMapping("/new_category")
66          public String showForm(Model model) {
67              Category category = new Category();
68               model.addAttribute("category", category);
69           return "new_category";
```

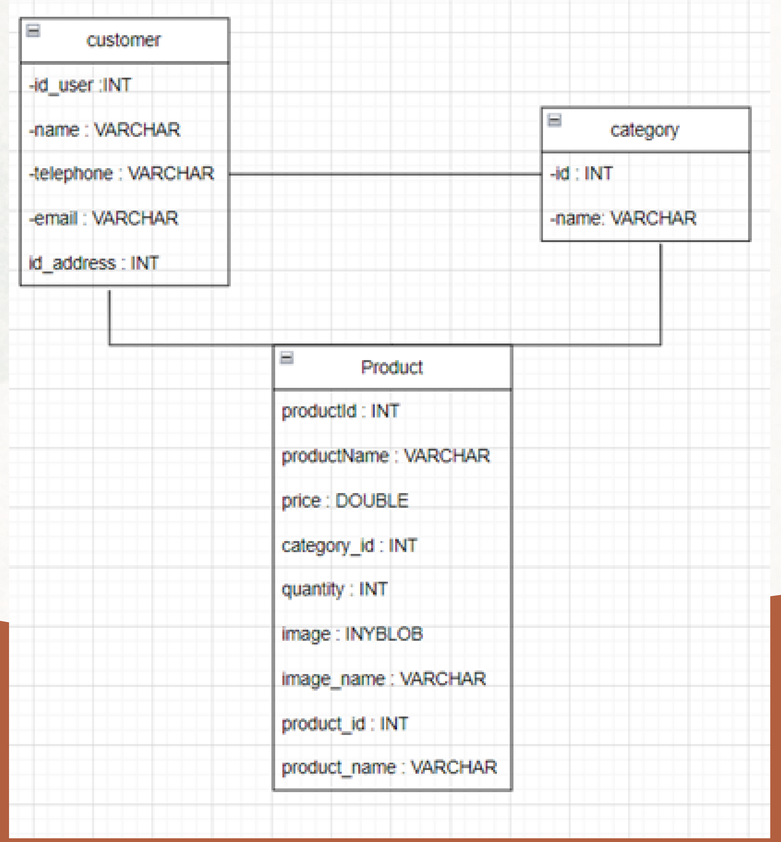
แผนภาพ UML


Item
-id : int
-name : string
-user_id : int(fk)
-order_id : int(fk)
-Id()
-setItem_Name()
-setUser_id()
-setOrdet_id()
User
-id : int
-username: string
-Id()
setUserName()
Order
-id : int
-user : string
-item : string
-quantity : string
-user_id : int
-Id()
-setUser_id()
-setItem_Name()
-quantity()
-setUser_id()

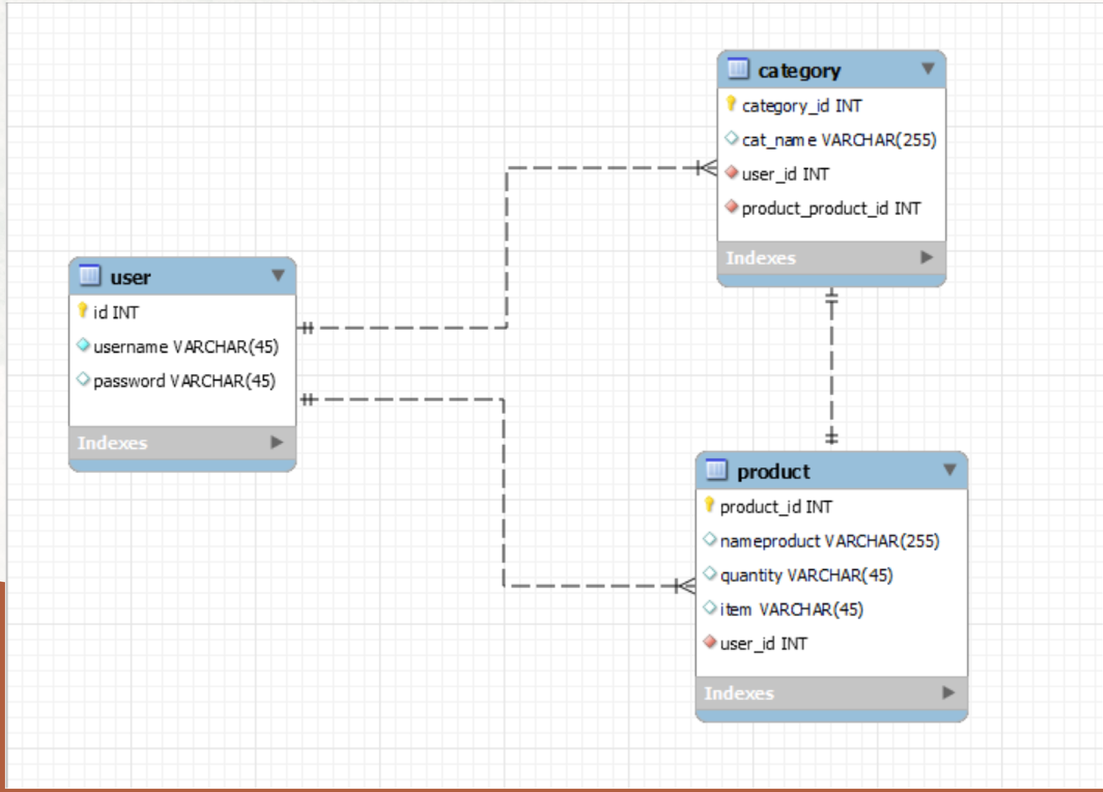
แผนภาพ
USE CASE DIAGRAM



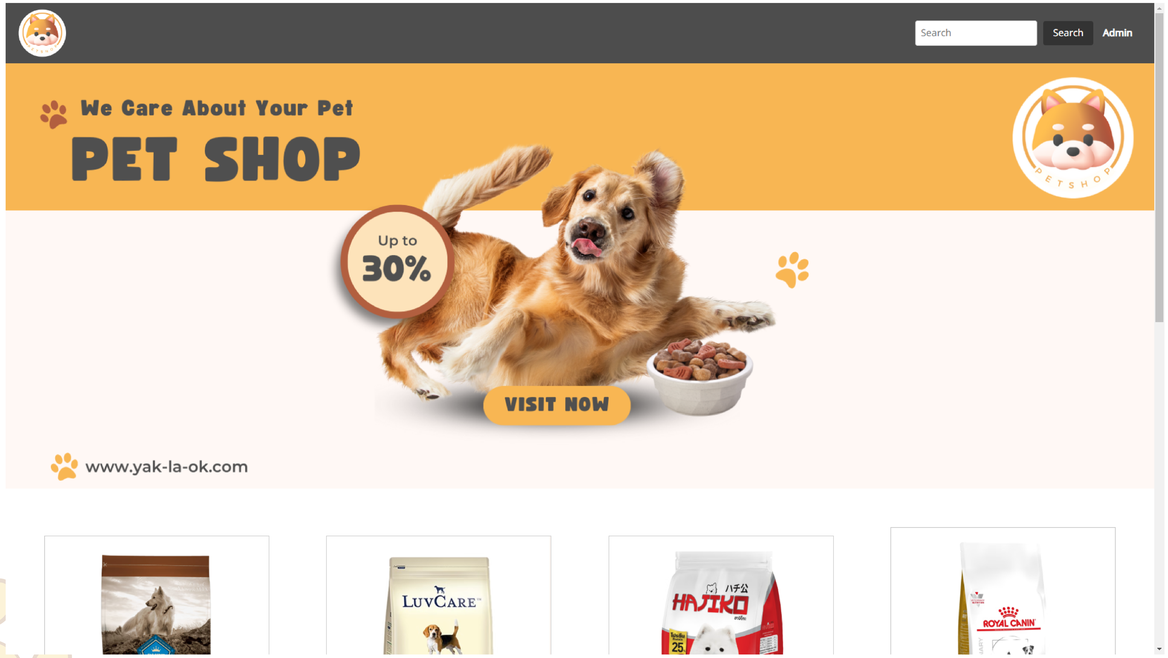
แผนภาพ
CLASS DIAGRAM

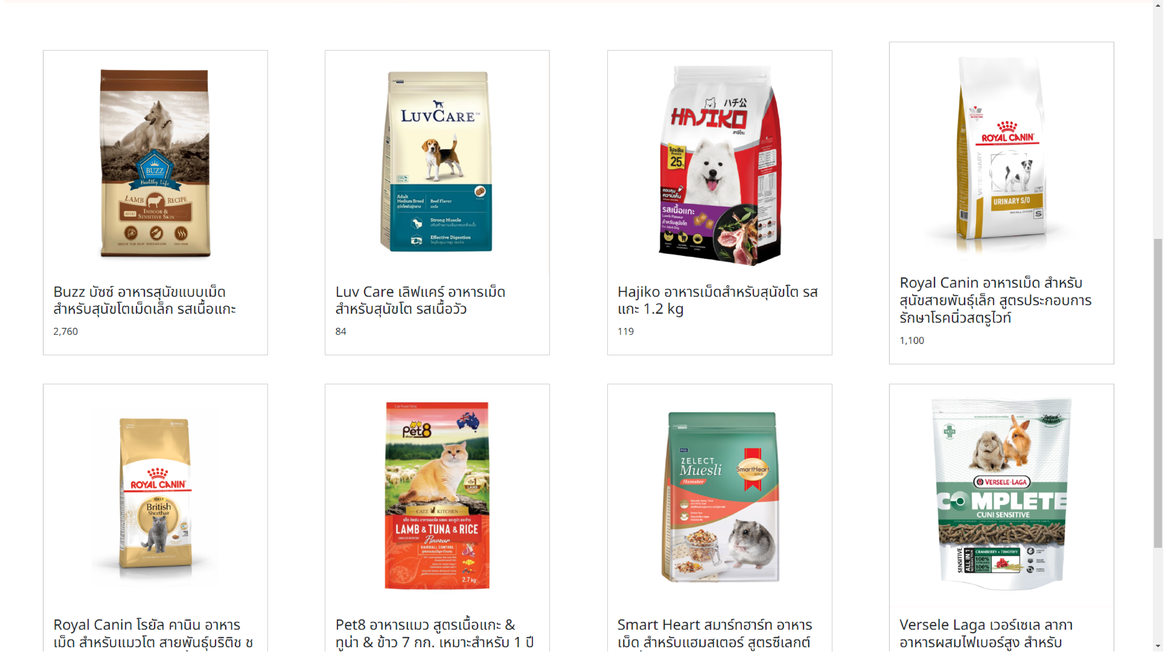


แผนภาพ
ER DIAGRAM

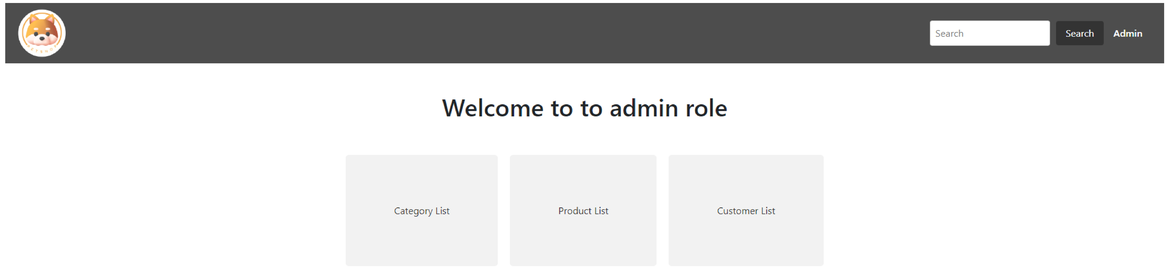


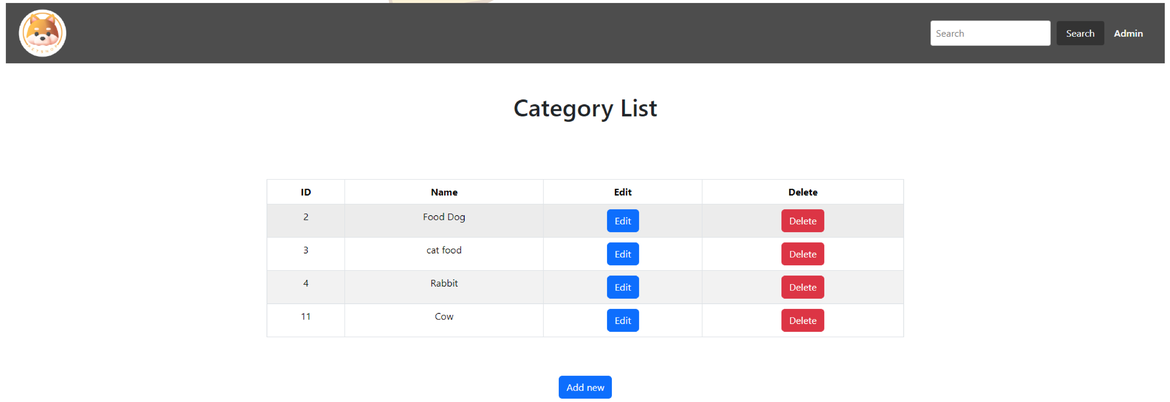
USER INTERFACE



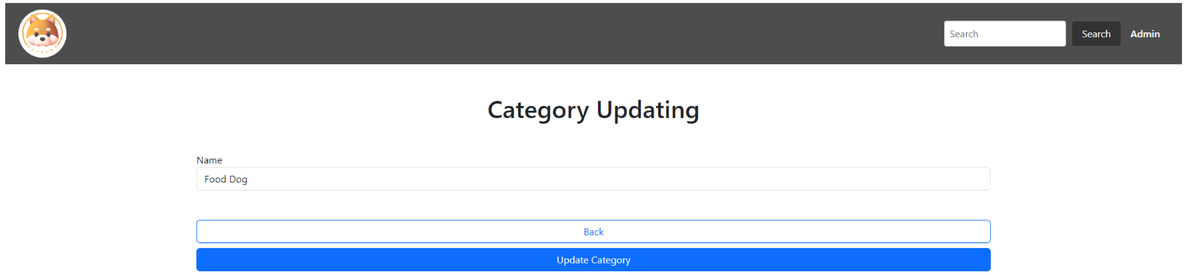


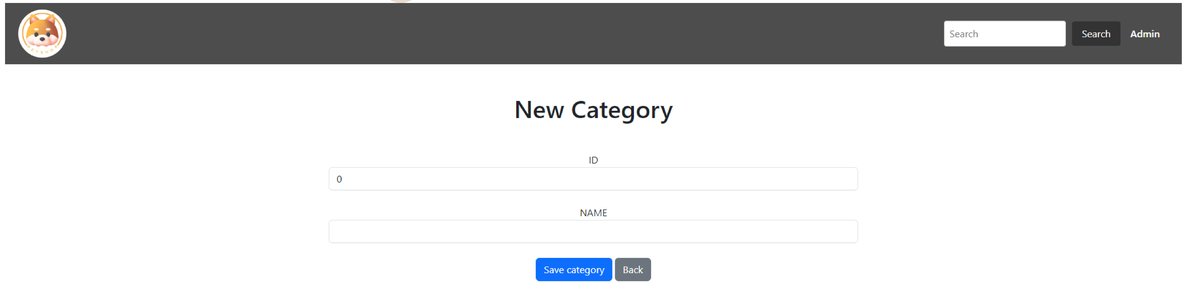
USER INTERFACE



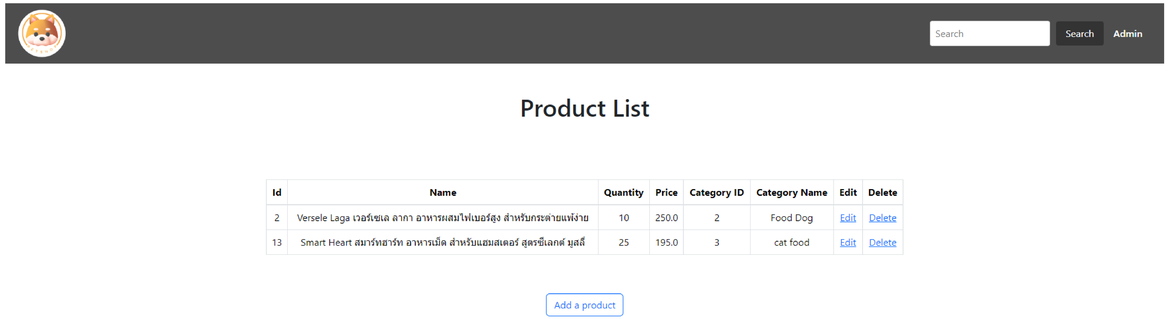


USER INTERFACE

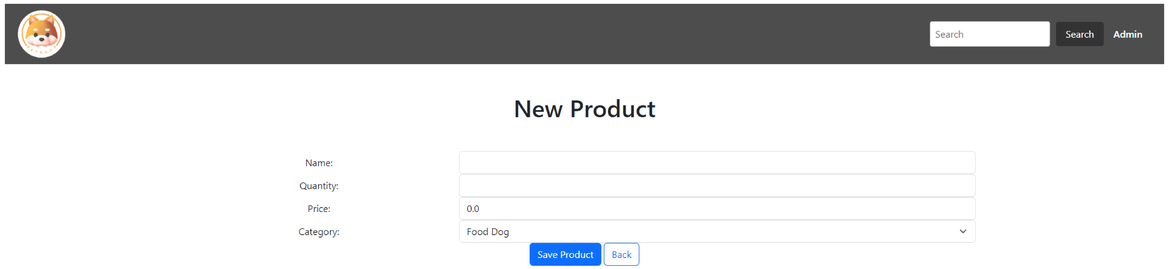


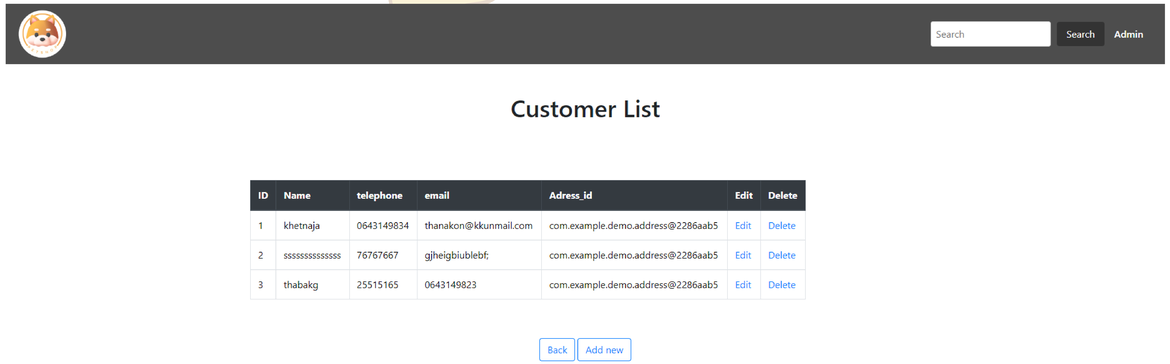


USER INTERFACE

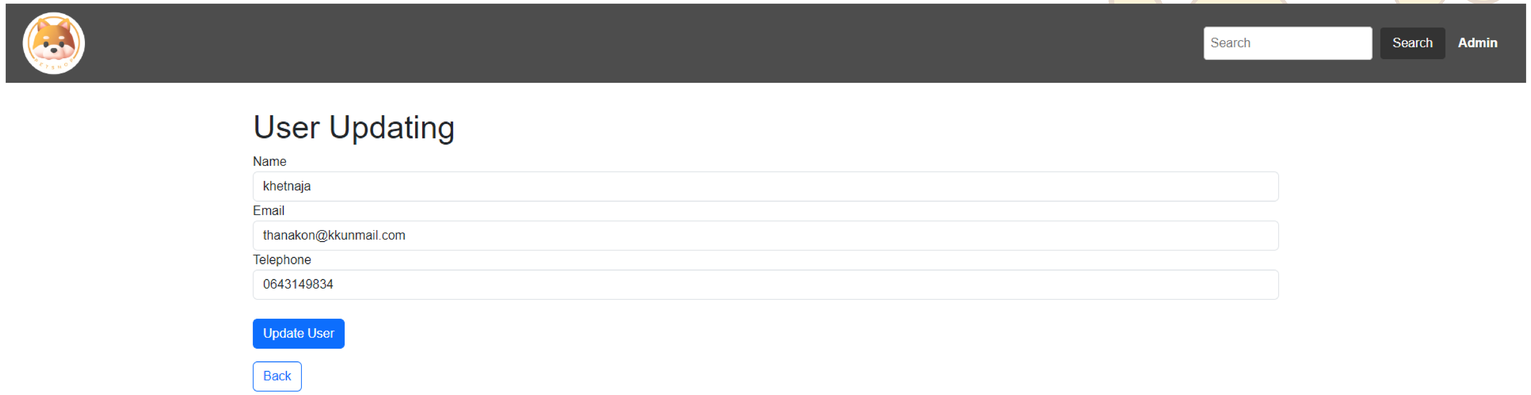


USER INTERFACE

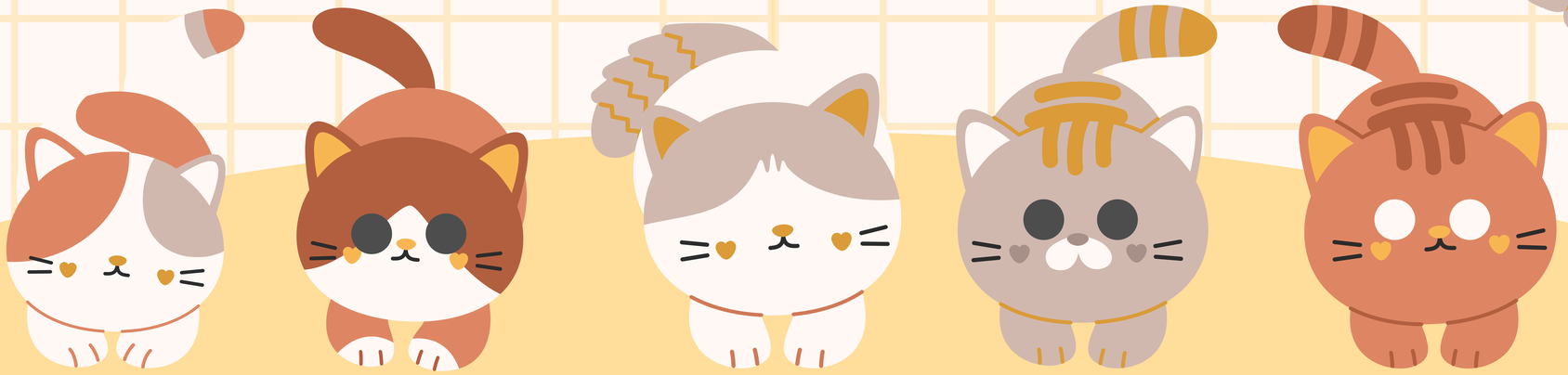




USER INTERFACE



THANK YOU



+66123-456-7890

KhonKaen,Thailand

www.myshop.com