

กลุ่ม CE-DDD

ชื่อโครงการ : POS4Shop

Github repository : POS4Shop

รายชื่อสมาชิก

นายภัทรพล	เจียรเสริมพงศ์	55010923
นายศุภกร	วโรดม	55011222

โครงการนี้เป็นส่วนหนึ่งของรายวิชา

01076254 – การวิเคราะห์และออกแบบเชิงวัตถุ

(Object-oriented Analysis and Design)

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 1 ปีการศึกษา 2557

Abstract

เว็บแอปพลิเคชันจะช่วยให้การขายของในร้านค้าต่าง ๆ นั้นสะดวกมากยิ่งขึ้นและลดปัญหาความผิดพลาดและล่าช้าในการทำธุรกรรมและยังเป็นหน้าต่างของร้านค้าอีกด้วยทำให้ผู้คนเข้าถึงร้านค้าได้มากขึ้น อีกทั้งยังสามารถสรุปยอดขายของร้านค้าในรูปแบบของ วัน/เดือน โดยแสดงออกมาเป็นกราฟให้กับผู้จัดการร้านค้าและมีระบบสมาชิกของร้านค้าอีกด้วย

Introduction

ในปัจจุบันร้านค้าแต่ละร้านมักมีสินค้าในร้านอยู่หลากหลายชนิด ทำให้การที่จะสามารถจำราคาสินค้าทุกอย่างในร้านด้านนั้นเป็นไปด้านยาก ทำให้ความสามารถ(ความเร็ว)ในการขายลดลง เนื่องจากจำราคาสินค้าไม่ได้ (ในร้านที่มีคนขายหลายคนก็จะมีผู้ขายที่จำราคาได้แค่บางคน) และคิดเงินช้าเนื่องจากต้องกดเครื่องคิดเลขเองทีละรายการสินค้า ทางกลุ่มจึงคิดจะทำระบบ POS สำหรับร้านค้าขึ้นมา โดยในปัจจุบันนั้น Web technology นั้นพัฒนาไปมาก ทางกลุ่มจึงเลือกที่จะพัฒนาแอปพลิเคชันนี้เป็น Web Application เพราะง่ายต่อการพัฒนา สามารถปรับปรุง และสามารถนำไปพัฒนาต่อยอดได้ง่าย

Related works

ระบบ POS นั้นมีผู้พัฒนาอยู่หลายคนด้วยกันเช่น Easy POS (Shareware), www.citysoft.co.th, www.prosoftpos.com, www.smlsoft.com ซึ่งระบบ POS ที่ได้กล่าวมานี้มีลักษณะที่เหมือนกันคือเป็น Desktop application ซึ่งต้องใช้งานบนระบบปฏิบัติการ Windows เท่านั้น ส่วนทางกลุ่มได้พัฒนาเป็น Web application ซึ่งสามารถใช้งานได้บนทุกระบบปฏิบัติการ และยังสามารถเข้าใช้งานผ่าน Smart phone ได้อีกด้วย

Functional requirements

- เลือกดูสินค้าต่างๆในร้านได้ผ่านทางหน้าเว็บ
- ระบบล็อกอิน สำหรับผู้จัดการ
- ระบบคำนวณราคาสินค้าต่อการซื้อของลูกค้า 1 ครั้ง สำหรับผู้จัดการ
- ระบบเก็บประวัติการขายสินค้าทั้งหมด สำหรับผู้จัดการ
- ระบบเพิ่มสินค้าชนิดใหม่เข้าสู่ฐานข้อมูล สำหรับผู้จัดการ
- ระบบดูประวัติการซื้อของลูกค้า สำหรับผู้จัดการและตัวลูกค้าเอง
- ระบบค้นหาสินค้าที่มีในร้าน
- ระบบแสดงรายการสินค้าที่มียอดขายสูงสุด 10 รายการ
- มีช่องทางการเชื่อมต่อ (Rest API) สำหรับเช็คราคาสินค้าในร้าน
- มีช่องทางการเชื่อมต่อ (Rest API) สำหรับลูกค้าเพื่อดูประวัติการซื้อของตัวเอง

Non-functional requirements

- หน้าเว็บรองรับการแสดงผลทั้งบน คอมพิวเตอร์ตั้งโต๊ะ และสมาร์ตโฟน
- มีรูปแบบการแสดงรายการสินค้าเป็นตารางเพื่อง่ายแก่การเลือกดูสินค้า
- เป็นเว็บแอปพลิเคชันทำให้สามารถติดตั้งลงในระบบปฏิบัติการใดก็ได้
- มีการรักษาความปลอดภัยของข้อมูลสำคัญที่ถูกจัดเก็บหรือถูกใช้ในการสื่อสาร โดยใช้โปรโตคอลที่มีการเข้ารหัส (https)
- รองรับผู้ใช้บริการพร้อมกัน 10 คนโดยประสิทธิภาพการทำงานไม่ลดลง

Use case specifications

1. Use case name: ระบบคำนวณมูลค่าสินค้า

- **Use case purpose:** ทำการคำนวณมูลค่าสินค้าที่ลูกค้าซื้อในครั้งนั้น แล้วบันทึกลงประวัติการ ขายของร้านค้าและประวัติการซื้อสินค้าของลูกค้า
- **Preconditions:** สินค้าทุกชิ้นในร้านคามีข้อมูลอยู่ในฐานข้อมูล และมีการกำหนดราคาสินค้าไว้แล้ว
- **Postconditions:** แสดงมูลค่าสินค้าที่ลูกค้าซื้อในครั้งนั้น มีประวัติการขายบันทึกลงฐานข้อมูล มีประวัติการซื้อของลูกค้าบันทึกลงในฐานข้อมูล
-
- **Limitations:** ลูกค้าที่ไม่ได้เป็นสมาชิก จะไม่สามารถดูประวัติการซื้อย้อนหลังได้
- **Assumptions:** ลูกค้าซื้อสินค้าในร้านอย่างน้อย 1 ชิ้นขึ้นไป
- **Primary Scenario:**
 - A. ผู้จัดการร้านล็อกอินเข้าสู่ระบบ
 - B. ระบบจะถามว่าเป็น ลูกค้าในระบบ หรือไม่
 - ถ้าใช่ ระบบจะให้เลือกว่าเป็นลูกค้าคนไหน
 - C. ผู้จัดการกดเลือกสินค้า และจำนวนชิ้นที่ลูกค้า
 - D. ระบบจะเพิ่มรายการสินค้าและจำนวนที่ซื้อแสดงบนหน้าเว็บ พร้อมคำนวณมูลค่าสินค้าในรายการทั้งหมด
 - E. ทำขั้นตอน C ไปเรื่อยๆ จนกว่าจะครบทุกรายการสินค้าที่ลูกค้าซื้อ
 - F. กดสิ้นสุดการเพิ่มสินค้า
 - G. ระบบจะแสดงมูลค่าสินค้าผล และช่องให้กรอกยอดเงินที่ลูกค้าจ่ายมา
 - H. ระบบจะแสดงจำนวนเงินทอนที่ต้องทอน
 - I. ผู้จัดการกดเสร็จสิ้นการทำรายการ
- **Alternate Scenario:**
- **Condition triggering an alternate scenario:**
- **Condition 1:** เลือกสินค้าผิดหรือจำนวนผิด
 - D1. กดลบรายการสินค้าที่ต้องการจะแก้ไขรายการ
 - D2. กลับไปทำตามขั้นตอน C ใหม่

2. Use case name: *ดูรายงานการขายสินค้า*

- **Use case purpose:** ดูสรุปการขายของร้าน ในช่วงเวลาที่เลือก เป็นวัน สัปดาห์ หรือเดือน
- **Preconditions:** ร้านค้าเคยมีการขายสินค้าไปแล้ว (มีประวัติการขาย)
- **Postconditions:** ระบบจะแสดงผลสรุปการขายสินค้าออกมาตามช่วงเวลาที่เลือก
- **Limitations:** -
- **Assumptions:** ผู้จัดการต้องรู้เรื่อง กำไร ขาดทุน และอ่านกราฟได้
- **Primary Scenario:**
 - A. ผู้จัดการร้านล็อกอินเข้าสู่ระบบ
 - B. กดเลือกไปที่ รายงานการขาย
 - C. เลือกช่วงเวลาที่ต้องการจะดูสรุปผลการขาย
 - D. เลือกดูรายวันหรือรายเดือน
 - E. ระบบจะแสดงสรุปผลการขายในช่วงเวลานั้น
- **Alternate Scenario:**
- **Condition triggering an alternate scenario:**
- **Condition 1:** ใส่ วัน/เดือน/ปี ที่ไม่มีอยู่จริง
 - C1. ระบบแจ้ง error
 - C2. ใส่ข้อมูล วัน/เดือน/ปี ให้ถูกต้อง
 - C3. ไปทำตามขั้นตอน D

Problem Analysis

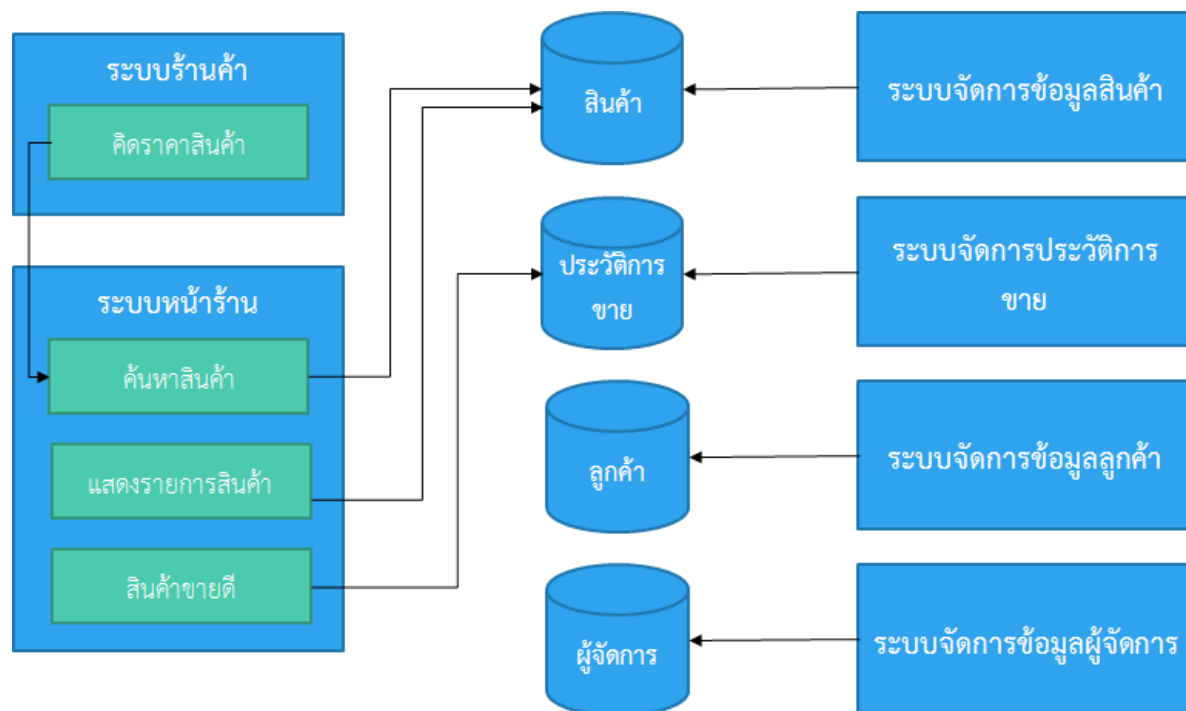
Component

- ส่วนหน้าร้าน สำหรับแสดงรายการสินค้า
- ส่วนค้นหาสินค้า
- ส่วนตั้งชื่อร้านค้า
- ส่วนคำนวณมูลค่าสินค้า
- ส่วนเพิ่มข้อมูลสินค้าชนิดใหม่
- ส่วนแก้ไขข้อมูลสินค้า
- ส่วนลบข้อมูลสินค้า
- ส่วนบันทึกข้อมูลประวัติการขายสินค้า
- ส่วนแก้ไขข้อมูลประวัติการขายสินค้า
- ส่วนลบข้อมูลประวัติการขายสินค้า
- ส่วนสมัครสมาชิกของลูกค้า
- ส่วนแสดงข้อมูลประวัติการซื้อของลูกค้า
- ส่วนแก้ไขข้อมูลลูกค้า
- ส่วนลบข้อมูลลูกค้า
- ส่วนเพิ่มผู้จัดการร้านค้า
- ส่วนแก้ไขรหัสผ่านของผู้จัดการร้านค้า
- ส่วนลบผู้จัดการร้านค้า
- ส่วนสรุปผล ยอดขายสินค้า

Abstraction

- หน้าร้าน มีรายการสินค้า ให้ผู้ใช้ได้เลือกดูข้อมูลของสินค้า
- สินค้า ซึ่งประกอบด้วยชื่อสินค้า บาร์โค้ด ราคาสินค้าและรายละเอียดอื่นๆเกี่ยวกับสินค้า
- ข้อมูลประวัติ ประกอบด้วยบันทึกการขายสินค้าของร้านค้า และข้อมูลการซื้อสินค้าของลูกค้า
- ลูกค้า ซึ่งประกอบด้วยชื่อลูกค้า และประวัติการซื้อสินค้า
- ผู้จัดการร้านค้า ซึ่งจะมีข้อมูลชื่อ, username, password ของผู้จัดการแต่ละคน
- ตัวสรุปผลการขายสินค้า เพื่อแสดงให้เห็นว่าสินค้าใดมีคนซื้อเยอะ

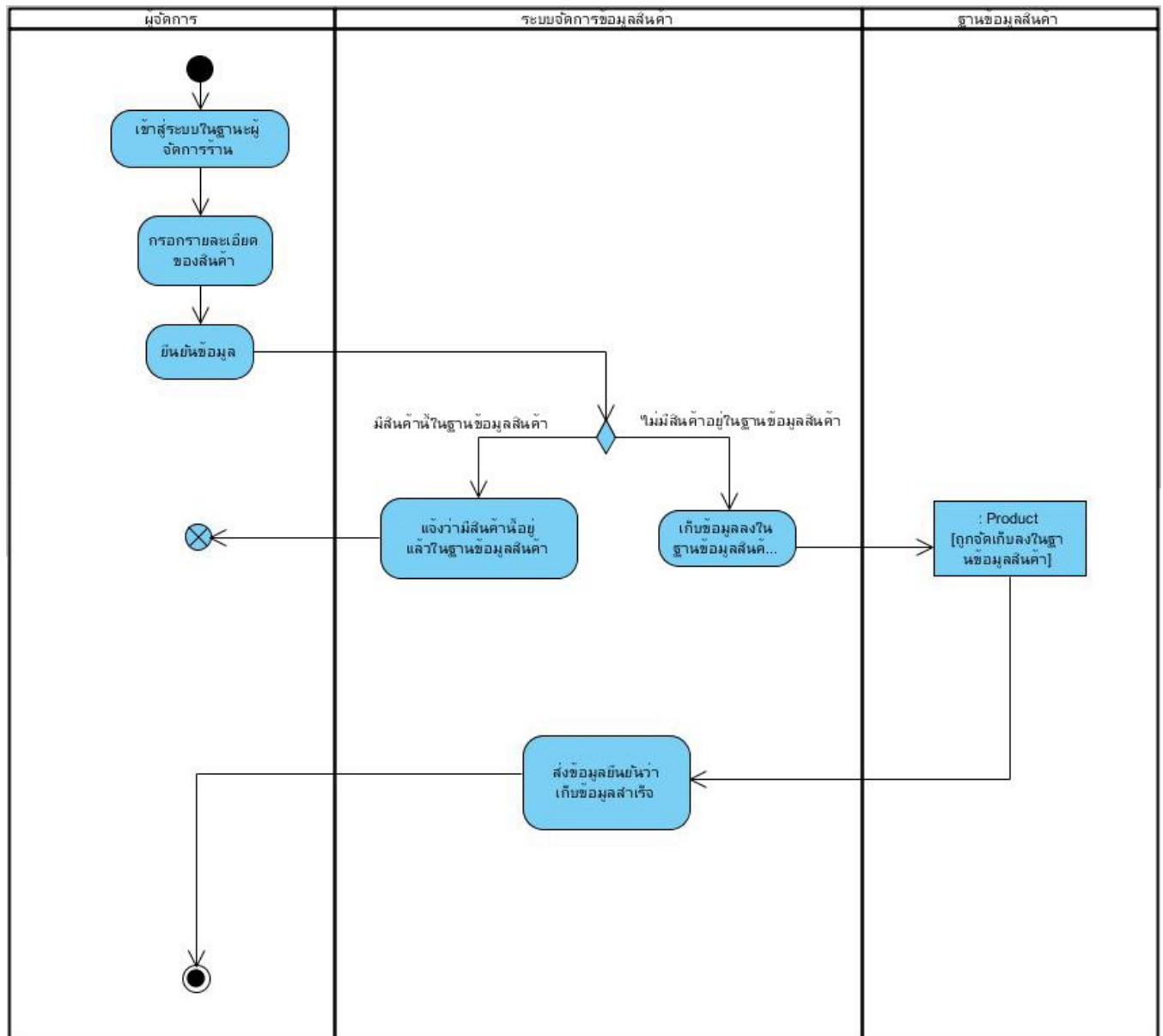
Application Architecture



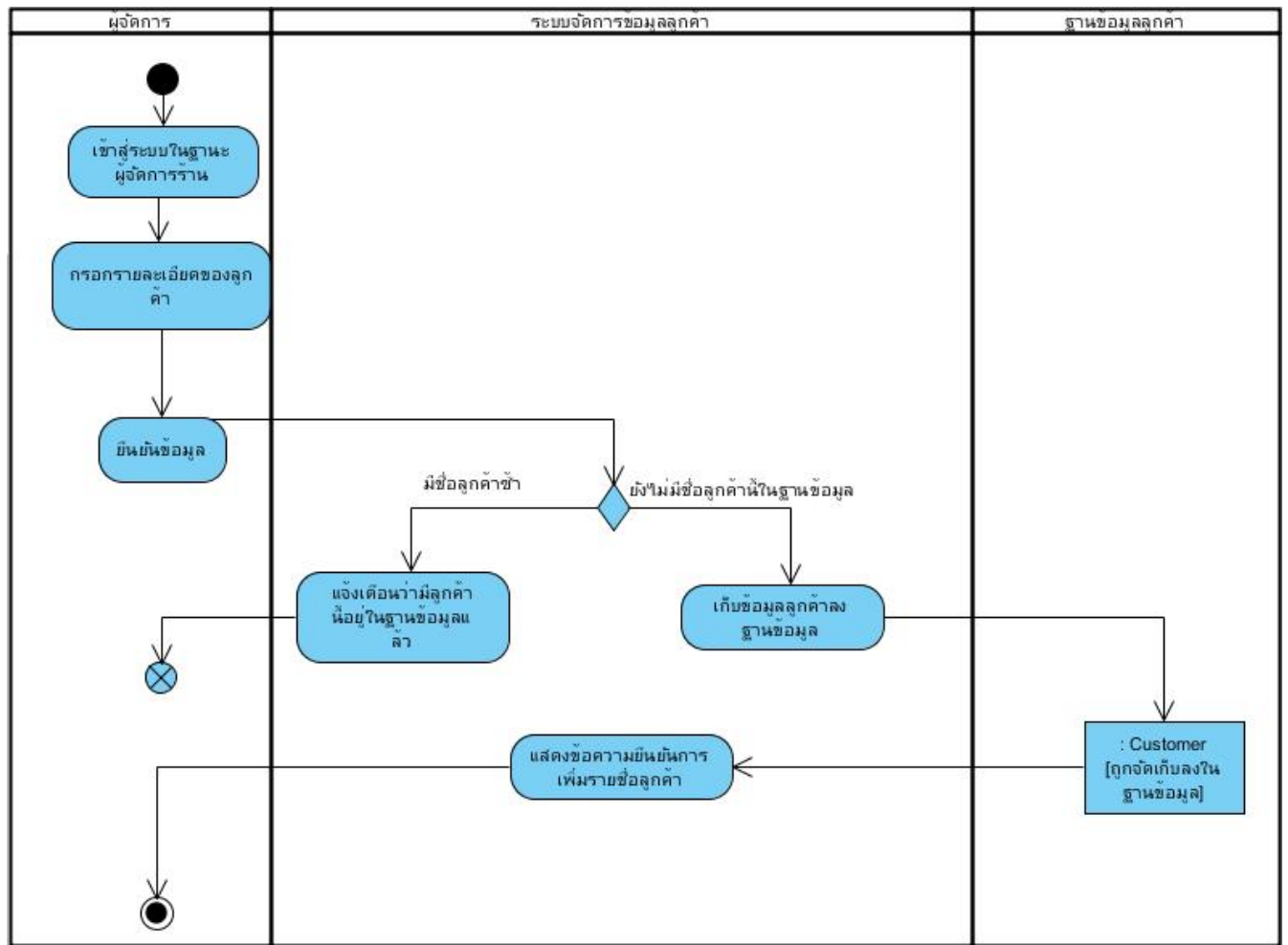
- **ระบบหน้าร้าน** - มีส่วนแสดงรายการสินค้าทั้งหมดในร้าน ส่วนแสดงรายการสินค้าที่ขายดี โดยใช้ข้อมูลจากส่วนสรุปผลยอดขายสินค้า และส่วนค้นหาสินค้า
- **ระบบร้านค้า** - มีส่วนตั้งชื่อร้านค้า และส่วนคำนวณมูลค่าสินค้า
- **ระบบจัดการข้อมูลสินค้า** - ส่วนเพิ่มข้อมูลสินค้าชนิดใหม่ แก้ไขข้อมูลสินค้า และลบข้อมูลสินค้า
- **ระบบจัดการข้อมูลประวัติการขายสินค้า** - มีส่วนบันทึกข้อมูลประวัติการขายสินค้า แก้ไข และลบข้อมูลประวัติการขายสินค้า
- **ระบบจัดการข้อมูลลูกค้า** - มีส่วนสมัครสมาชิกของลูกค้า แสดงข้อมูลประวัติการซื้อ แก้ไข และลบข้อมูลลูกค้า
- **ระบบจัดการข้อมูลผู้จัดการร้านค้า** - มีส่วนเพิ่มผู้จัดการร้านค้า ส่วนแก้ไขข้อมูลและรหัสผ่าน และส่วนลบผู้จัดการร้านค้า

Activity diagrams

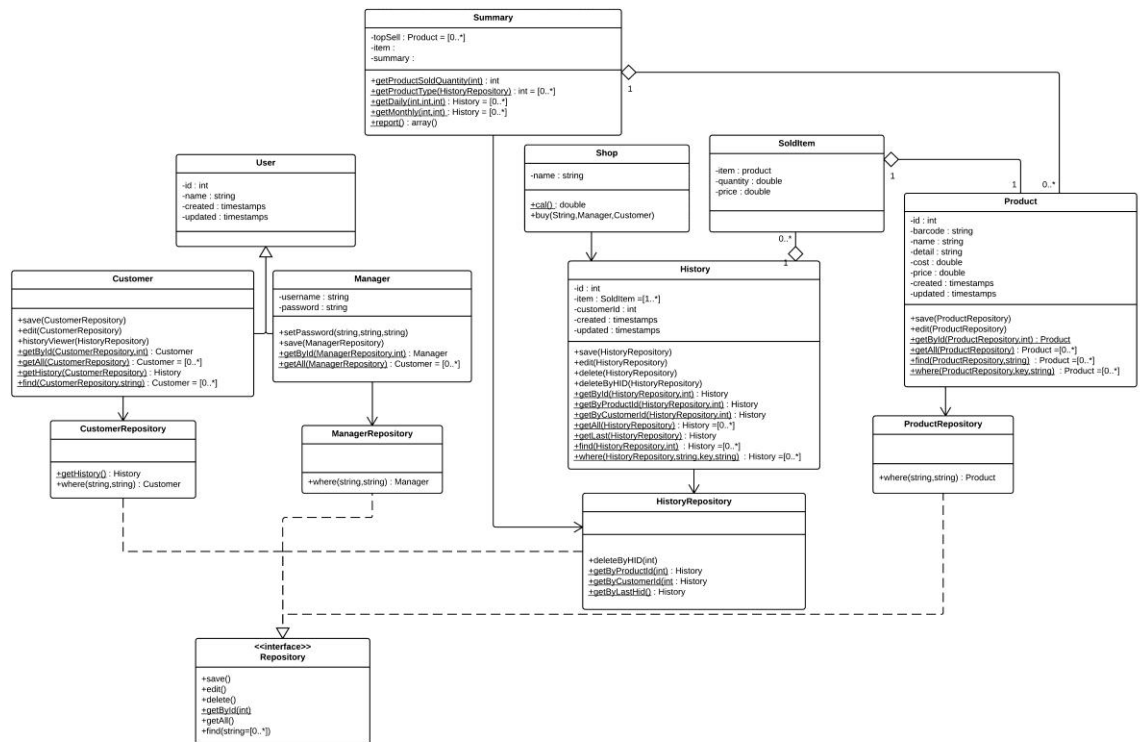
เพิ่มสินค้าในฐานข้อมูลสินค้า



เพิ่มรายชื่อลูกค้า



Domain classes



(เม็รทอดที่ขีดเส้นใต้คือ static method)

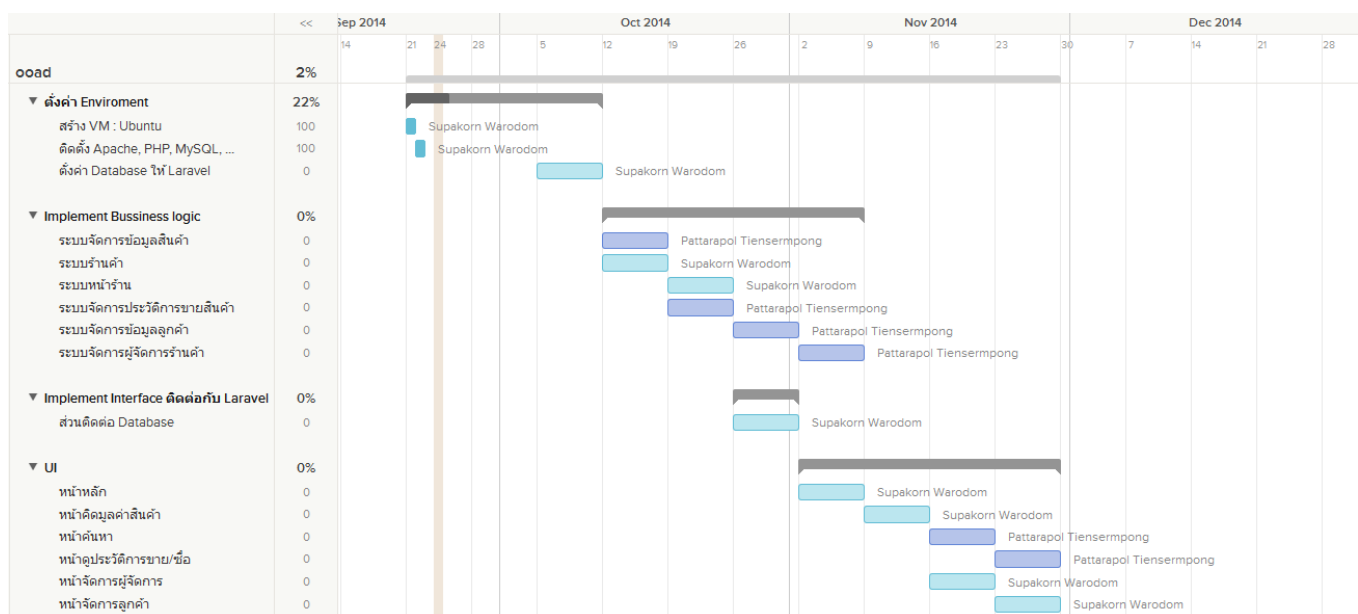
- User
 - เป็นคลาสที่มีข้อมูลพื้นฐานของผู้ใช้ ทั้งผู้จัดการ และลูกค้า
- Manager
 - เป็นคลาสที่ใช้จัดการข้อมูลของผู้จัดการร้านค้า เช่น การเปลี่ยนรหัสผ่าน, แก้ไขข้อมูลต่างๆ
- Customer
 - เป็นคลาสที่ใช้ในการจัดการข้อมูลของลูกค้า เช่น การสมัครสมาชิกใหม่, การดูประวัติการซื้อ, และดูรายชื่อลูกค้าทั้งหมด
- Product
 - เป็นคลาสที่ใช้ในการจัดการข้อมูลของสินค้า เช่น ชื่อและรายละเอียดต่างๆ, ค้นหาสินค้า, และดูรายการสินค้าที่มีทั้งหมด
- SoldItem
 - เป็นคลาสที่ใช้ในการเก็บข้อมูลว่ามีการซื้อสินค้าอะไร จำนวนกี่ชิ้น และราคาตอนซื้อนั้นเท่าไร ซึ่งจะถูกนำไปใช้ต่อไคลาส History

- History
 - เป็นคลาสที่ใช้ในการจัดการข้อมูลประวัติการขายของร้านค้า/ซื้อสินค้าของ ลูกค้า โดยภายในจะประกอบไปด้วยลิสต์อ็อบเจกต์ของคลาส SoldItem เพื่อที่จะได้เก็บข้อมูลการซื้อสินค้าว่าครั้งนั้นลูกค้าได้ซื้อสินค้าอะไร บ้างจำนวนกี่ชิ้น และราคา ณ ตอนนั้นเป็นเท่าไร และยังมีความสามารถ เช่น การบันทึกประวัติการขายสินค้า, และดึงรายการประวัติการขายสินค้าทั้งหมดออกมา (ซึ่งจะมีคลาส Summary ที่เรียกใช้เพื่อนำไปหาคำนวนเพื่อหาสินค้าที่ ขายดี - สินค้าที่ขายได้จำนวนมาก)
- Repository
 - เป็นอินเตอร์เฟซคลาส ที่มีเมธอดต่างๆไว้สำหรับใช้ติดต่อกับฐานข้อมูล
- ProductRepository
 - เป็นคลาสที่ใช้ติดต่อกับฐานข้อมูลของ Product โดยจะอิมพลีเมนต์อินเตอร์เฟซจากคลาส Repository
- HistoryRepository
 - เป็นคลาสที่ใช้ติดต่อกับฐานข้อมูลของ History โดยจะอิมพลีเมนต์อินเตอร์เฟซจากคลาส Repository
- ManagerRepository
 - เป็นคลาสที่ใช้ติดต่อกับฐานข้อมูลของ Manager โดยจะอิมพลีเมนต์อินเตอร์เฟซจากคลาส Repository
- CustomerRepository
 - เป็นคลาสที่ใช้ติดต่อกับฐานข้อมูลของ Customer โดยจะอิมพลีเมนต์อินเตอร์เฟซจากคลาส Repository
- Shop
 - เป็นคลาสที่ใช้ในการจัดการข้อมูลของร้านค้า เช่น การเปลี่ยนชื่อร้านค้า, การคิดมูลค่าของสินค้าที่ลูกค้าซื้อ
- Summary
 - เป็นคลาสที่ทำหน้าที่ในการคำนวณหาสินค้าที่ขายดี โดยการใช้ข้อมูลประวัติการขายสินค้ามาคำนวณ แล้วจะเรียงรายการสินค้าตามจำนวนชิ้นที่ขายได้มากที่สุดไปน้อยสุด เพื่อนำไปใช้ในการแสดงรายการสินค้าขายดี

Deployment

- Software และ Technology ที่ใช้ ได้แก่
 - Ubuntu 14.04
 - Apache 2
 - PHP 5.5
 - MySQL
 - Laravel - PHP framework
 - Composer - PHP Dependency manager
- โดย Software และระบบทั้งหมด จะติดตั้งลงใน VM 1 เครื่อง (128.199.138.215
<https://ooad.jigko.me/>)

Implementation Plan



UnitTest

การทดสอบระบบที่จะใช้ในการวัดว่าระบบทำงานได้ถูกต้องหรือไม่ ซึ่งทางกลุ่มเราได้มีการทดสอบระบบดังนี้

1. ทดสอบฟังก์ชันคำนวณมูลค่าสินค้า

```
class TestCase extends Illuminate\Foundation\Testing\TestCase {  
  
    /**  
     * Creates the application.  
     *  
     * @return \Symfony\Component\HttpKernel\HttpKernelInterface  
     */  
    public function createApplication()  
    {  
        $unitTesting = true;  
  
        $testEnvironment = 'testing';  
  
        return require __DIR__.'/../bootstrap/start.php';  
    }  
}
```

```
class ShopTest extends TestCase {  
  
    public function testCal()  
    {  
        // Asset  
        $totalPrice=0;  
        $arrItem=array();  
        for ($i=1 ; $i < 5; $i++) { // Mock 5 SoldItem object  
            $quantity = $i*2;  
            $price = ($i*10);  
            $totalPrice += $price*$quantity; // price * quantity  
  
            $solditem = Mockery::mock('ceddd\SoldItem');  
            $solditem->shouldReceive('get')  
                ->with('price')->andReturn($price);  
  
            $solditem->shouldReceive('get')  
                ->with('quantity')->andReturn($quantity);  
  
            $arrItem[$i] = $solditem;  
        }  
  
        // Act  
        $customer=$this->getMock('ceddd\Customer',array('__construct'),array($this->getMock('ceddd\CustomerRepository')));  
        $manager=$this->getMock('ceddd\Manager',array('__construct'),array($this->getMock('ceddd\ManagerRepository')));  
        $shop = new ceddd\Shop;  
        $result=$shop->cal($arrItem,$manager,$customer);  
  
        // Assert  
        $this->assertEquals($totalPrice, $result);  
    }  
}
```

เป็นการทดลองโดยจำลองการซื้อสินค้าซึ่งโดยจำลองให้มีการซื้อสินค้า 5 ชนิดและให้ผู้จัดการเป็นผู้ทำรายการคำนวณสินค้า ผลลัพธ์ที่คาดหวังก็คือผลการคำนวณราคาจากระบบกับผลการคำนวณราคาจาก Test code นั้นมีค่าเท่ากัน

2. ทดสอบฟังก์ชันสรุปยอดขายสินค้า

```
class TestCase extends Illuminate\Foundation\Testing\TestCase {  
  
    /**  
     * Creates the application.  
     *  
     * @return \Symfony\Component\HttpKernel\HttpKernelInterface  
     */  
    public function createApplication()  
    {  
        $unitTesting = true;  
  
        $testEnvironment = 'testing';  
  
        return require __DIR__.'/../bootstrap/start.php';  
    }  
}
```

```
class SummaryTest extends TestCase {  
  
    public function testProductSoldQuantity()  
    {  
        // Asset  
  
        $testProductId = 1;  
        $expectedQuantity = 0;  
  
        $arrOfHistory = array();  
        for ($i=0; $i < 5; $i++) { // Mock History object and data  
  
            $arrOfSoldItem=array();  
            $quantity = $i+1;  
  
            $product = Mockery::mock('ceddd\\Product');  
            $product->shouldReceive('get')->with('id')->andReturn($testProductId);  
  
            $solditem = Mockery::mock('ceddd\\SoldItem');  
            $solditem->shouldReceive('get')->with('item')->andReturn($product);  
            $solditem->shouldReceive('get')->with('quantity')->andReturn($quantity); // quantity = 1, 2, 3  
            $arrOfSoldItem[0]=$solditem;  
  
            $expectedQuantity+=$quantity;  
  
            $history = Mockery::mock('ceddd\\History');  
            $history->shouldReceive('get')->with('item')->andReturn($arrOfSoldItem);  
            $arrOfHistory[$i] = $history;  
        }  
  
        $historyRepository = Mockery::mock('ceddd\\HistoryRepository');  
        $historyRepository->shouldReceive('getByProductId')->with($testProductId)->andReturn($arrOfHistory);  
  
        // Act  
        $topSell = new ceddd\\Summary($historyRepository);  
        $result = $topSell->getProductSoldQuantity($testProductId); // $productId = 1  
  
        // Assert  
        $this->assertEquals($expectedQuantity, $result);  
    }  
}
```

เป็นการทดลองโดยจะนับว่า สินค้าแต่ละชนิดนั้นมีการขายไปกี่ชิ้นจากประวัติการขายสินค้า ผลลัพธ์ที่คาดหวังก็คือจำนวนสินค้าที่ระบบคำนวณว่าขายได้กับผลการคำนวณจำนวนสินค้าจาก Test code นั้นมีค่าเท่ากัน

และเมื่อทำการรัน Test code ทั้งสองบน <https://travis-ci.org/> จะมีผลการทดสอบคือผ่าน ดังภาพ

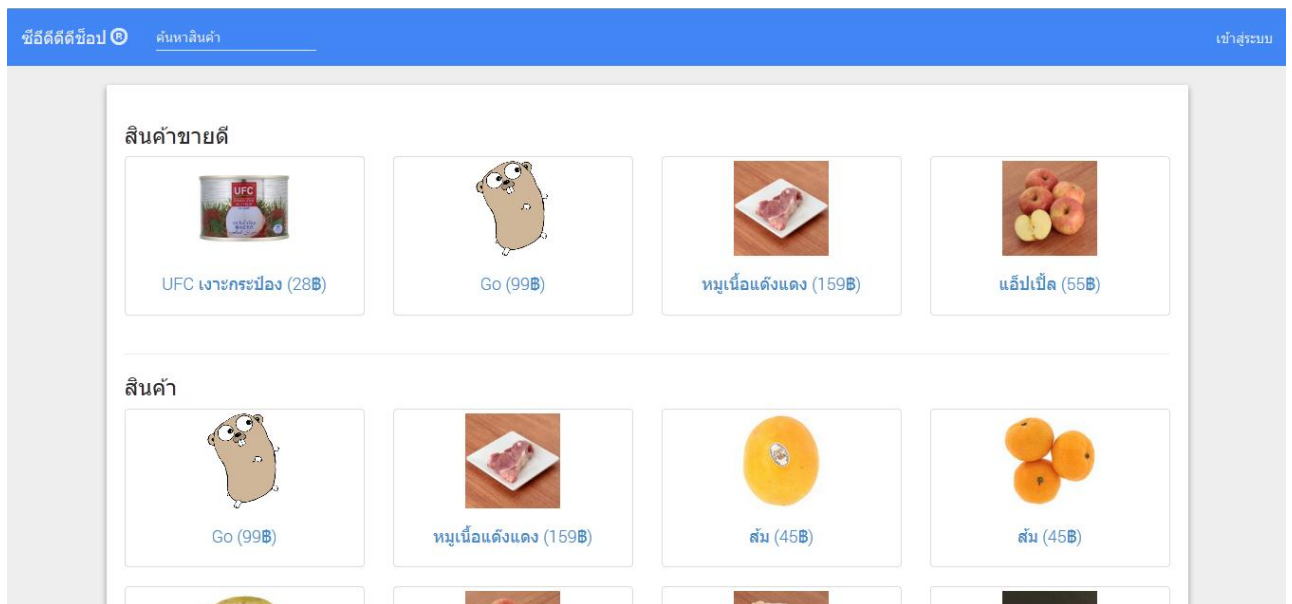
```
1 Using worker: worker-linux-7-1.bb.travis-ci.org:travis-linux-15
2
3 Build system information system_info
5
6 $ git clone --depth=50 --branch=master git://github.com/CE-KMITL-OOAD-2014/POS4Shop.git CE-KMITL- git.checkout 0.27s
16 $ phpenv global 5.5 0.03s
17 $ php --version
18 PHP 5.5.18 (cli) (built: Nov 2 2014 04:41:31)
19 Copyright (c) 1997-2014 The PHP Group
20 Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
21 with Zend OPcache v7.0.4-dev, Copyright (c) 1999-2014, by Zend Technologies
22 with Xdebug v2.2.5, Copyright (c) 2002-2014, by Derick Rethans
23 $ composer --version
24 Composer version 1.0-dev (a309e1d89ded6919935a842faeae8e888fbfe37) 2014-10-20 19:16:14
25 $ composer self-update before_script.1 2.06s
29 $ composer install --prefer-source --no-interaction --dev before_script.2 62.94s
149 $ phpunit 0.33s
150 PHPUnit 4.3.4 by Sebastian Bergmann.
151
152 Configuration read from /home/travis/build/CE-KMITL-OOAD-2014/POS4Shop/phpunit.xml
153
154 ..
155
156 Time: 216 ms, Memory: 14.00Mb
157
158 OK (2 tests, 3 assertions)
159
160 The command "phpunit" exited with 0.
161
162 Done. Your build exited with 0.
```

Evaluation

- ทดสอบระบบคำนวณมูลค่าสินค้า
 - จุดประสงค์การทดลอง
 - เพื่อทดสอบระบบคิดราคาสินค้า ว่าทำงานได้จริงและมีความถูกต้องสิ่งที่จะวัด
 - ความง่ายและ สะดวกในการใช้งาน
 - ความแม่นยำและถูกต้องในการคำนวณ
 - สิ่งที่จะวัด
 - ความถูกต้องของการคำนวณมูลค่าสินค้า
 - สิ่งที่ต้องใช้ในการทดลอง
 - ข้อมูลของสินค้าในฐานข้อมูล
 - ID,password ของ manager

- วิธีการทดลอง

- คลิกที่ “เข้าสู่ระบบ” เพื่อนำทำการ login manager



- กรอก username,password

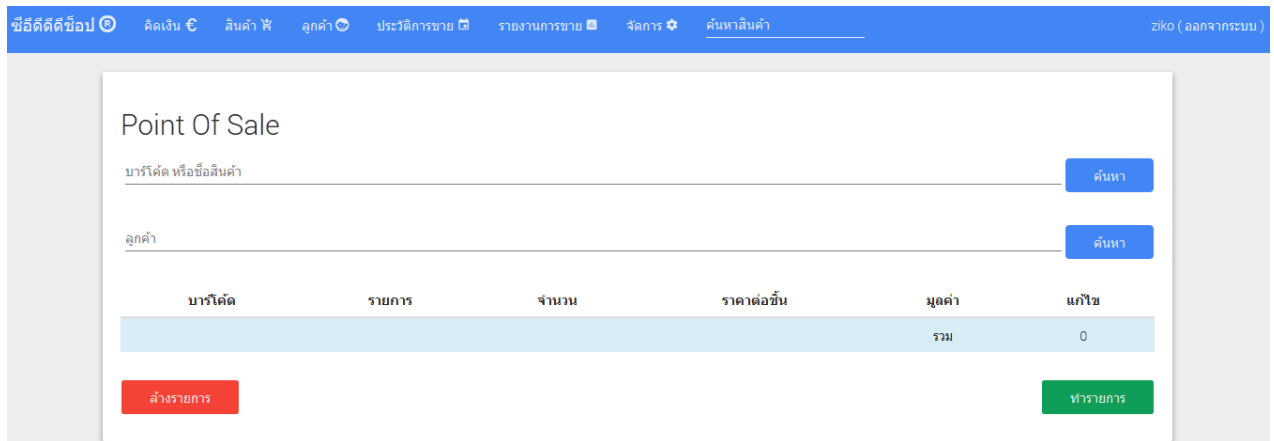
Login

Username

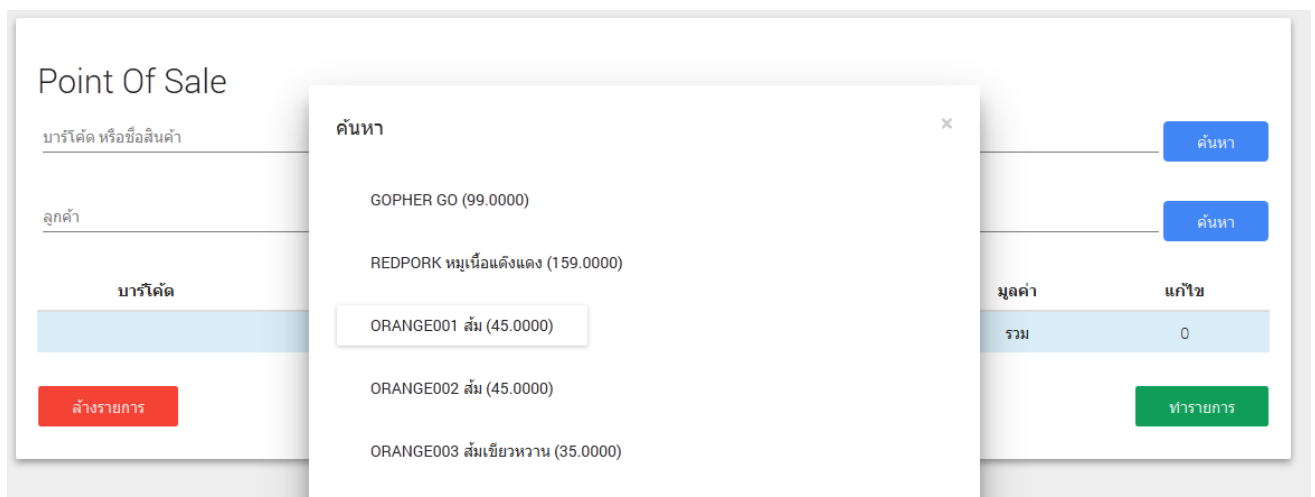
Password

SUBMIT

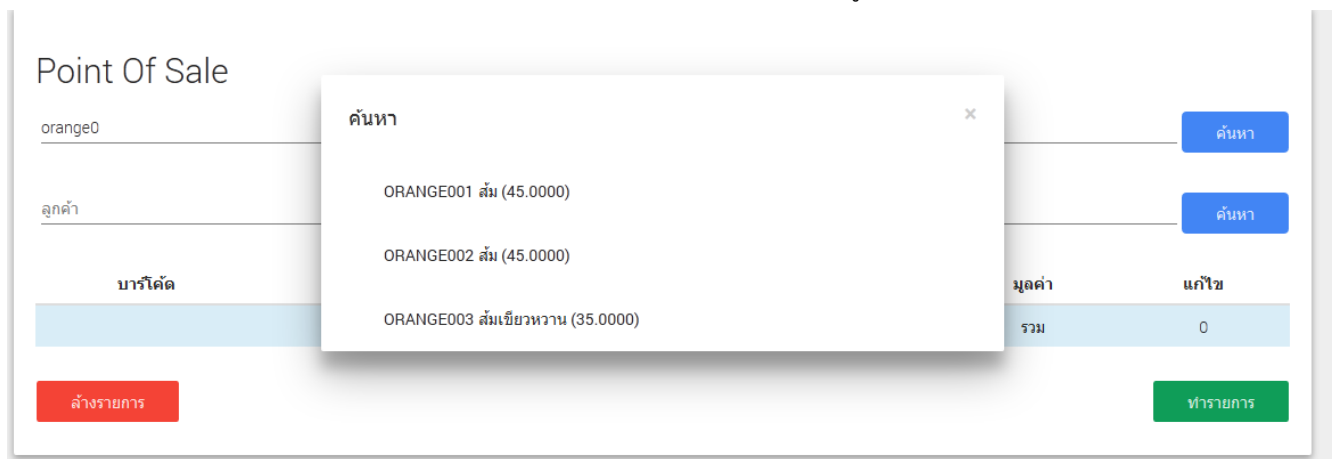
- หลังจาก login เรียบร้อยหน้าเว็บจะเป็นรูปแบบสำหรับผู้จัดการ ซึ่งเมื่อกดปุ่ม “คิดเงิน” จะมีหน้าจอคิดราคาดังรูป



- กดปุ่ม “ค้นหา” ที่บรรทัด บาร์โค้ดหรือชื่อสินค้า เพื่อเลือกสินค้า



- หรือสามารถพิมพ์หาจากชื่อสินค้า หรือบาร์โค้ดได้ดังรูป



- เมื่อเลือกสินค้ามาแล้วให้เลือกที่จะใส่ชื่อลูกค้าหรือไม่ก็ได้

Point Of Sale

บาร์โค้ด หรือชื่อสินค้า

ค้นหา

ลูกค้า

ค้นหา

บาร์โค้ด	รายการ	จำนวน	ราคาต่อชิ้น	มูลค่า	แก้ไข
orange003	ส้มเขียวหวาน	2	35.0000	70	DELETE
GOPHER	Go	2	99.0000	198	DELETE
รวม				134	

ล้างรายการ

ทำรายการ

ค้นหาลูกค้า

26 : ZIKO5555

27 : ZIKO THE CUSTOMER

29 : SUPERCUSTOMER

30 : MIDAS

Point Of Sale

บาร์โค้ด หรือชื่อสินค้า

ค้นหา

Midas

ค้นหา

บาร์โค้ด	รายการ	จำนวน	ราคาต่อชิ้น	มูลค่า	แก้ไข
orange003	ส้มเขียวหวาน	2	35.0000	70	DELETE
GOPHER	Go	2	99.0000	198	DELETE
รวม				268	

ล้างรายการ

ทำรายการ

- ระบบจะคำนวณราคาสินค้าตามที่รายการสินค้าที่ซื้อ

บาร์โค้ด	รายการ	จำนวน	ราคาต่อชิ้น	มูลค่า	แก้ไข
orange003	ส้มเขียวหวาน	2	35.0000	70	DELETE
GOPHER	Go	1	99.0000	99	DELETE
รวม				169	

- กดปุ่มทำรายการระบบจะแจ้งว่าทำรายการเรียบร้อยแล้ว

ทำรายการสำเร็จ

Point Of Sale

บาร์โค้ด หรือชื่อสินค้า

ค้นหา

ลูกค้า

ค้นหา

บาร์โค้ด	รายการ	จำนวน	ราคาต่อชิ้น	มูลค่า	แก้ไข
				รวม	0

ล้างรายการ

ทำรายการ

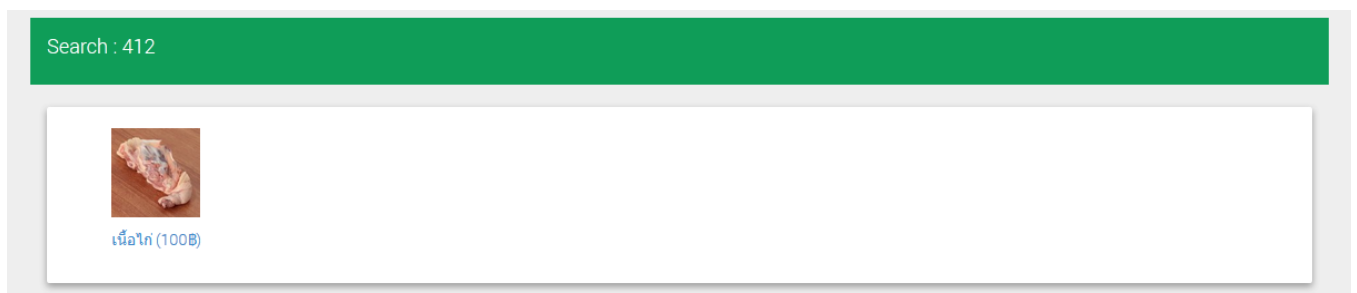
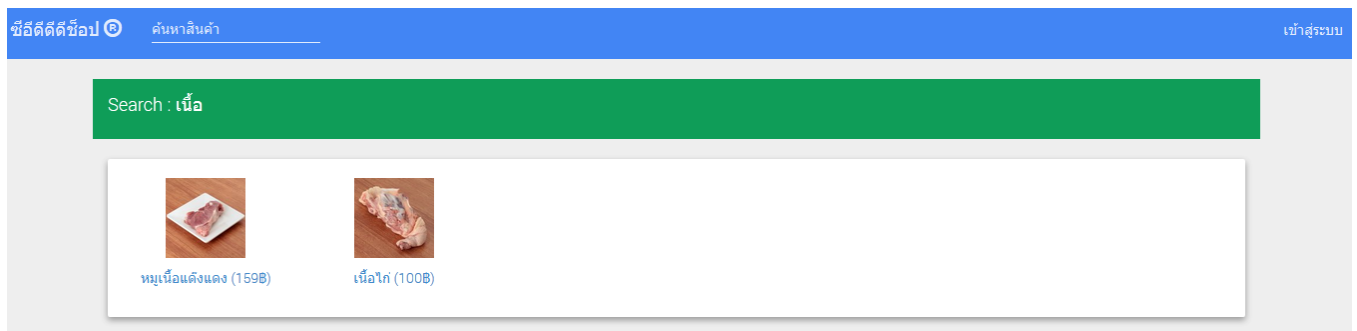
- ระบบจะบันทึกประวัติการขายซึ่งสามารถเข้าไปดูได้ที่ “ประวัติการขาย”

รายการการขายสินค้า						
ID	รายการ	จำนวน	ราคา	ลูกค้า	ผู้จัดการ	เมื่อ
1	แอปเปิ้ล	1.00	55.00	Ziko the customer	ziko	2014-11-12 20:10:18
	หมูเนื้อแดงแดง	1.00	169.00			
	Go	2.00	99.00			
2	หมูเนื้อแดงแดง	1.00	169.00	Ziko5555	ziko	2014-11-14 21:16:27
3	UFC เจาะกระป๋อง	4.00	28.00	SuperCustomer	ziko	2014-11-14 21:21:34
4	Go	1.00	99.00	Ziko the customer	ziko	2014-11-15 10:16:12
5	ส้ม	1.00	45.00	-	ziko	2014-11-15 14:44:58
6	ส้มเขียวหวาน	2.00	35.00	-	ziko	2014-11-15 14:51:25
	Go	1.00	99.00			
7	ส้มเขียวหวาน	2.00	35.00	Midas	ziko	2014-11-15 14:54:01
	Go	2.00	99.00			

- ผลที่ได้จากการทดลอง
 - ระบบสามารถคำนวณราคาสินค้าที่ซื้อได้อย่างถูกต้อง
 - เมื่อทำรายการการซื้อเสร็จแล้วระบบบันทึกประวัติการซื้อสินค้าไว้ได้
- สรุปและสิ่งที่ได้จากการทดลองนี้
 - ระบบคำนวณราคาสินค้าทำงานเชื่อมกับฐานข้อมูลได้อย่างถูกต้อง
 - ระบบคำนวณราคาสินค้ามีความสัมพันธ์กับระบบจัดการประวัติการขาย

- ทดสอบฟังก์ชันตรวจการป้อนข้อมูล

- จุดประสงค์การทดลอง
 - ทดสอบว่าระบบนี้ใช้งานได้จริงและป้องกันการเกิด error จากการป้อนข้อมูลผิด
- สิ่งที่จะวัด
 - ระบบมีการแจ้งเตือนว่าป้อนข้อมูลผิดและไม่มี error จากการป้อนข้อมูลเกิดขึ้น
- สิ่งที่ต้องใช้ในการทดลอง
 - ชุดข้อมูลที่ถูกต้อง
 - ชุดข้อมูลที่ไม่ถูกต้องที่จะทำให้ระบบตรวจการป้อนข้อมูลทำงาน
- วิธีการทดลอง
 - ทดลองกับระบบค้นหาสินค้า
 - กรอกข้อมูลเพื่อค้นหาสินค้าซึ่งสามารถใช้เป็นชื่อหรือบาร์โค้ดของสินค้า



- กรอกข้อมูลสินค้าที่ไม่มีในร้านค้า



- ทดสอบกับส่วนเพิ่มรายชื่อลูกค้า
 - ไปยังหน้าต่าง “ลูกค้า” และกด “เพิ่ม

รายชื่อลูกค้า

เพิ่ม

ID	ชื่อ
26	Ziko5555
27	Ziko the customer
29	SuperCustomer
30	Midas

ลูกค้าใหม่

ชื่อ

Input field

เพิ่ม

- ทดลองใส่ชื่อลูกค้าที่มีอยู่แล้วในฐานข้อมูล

รายชื่อลูกค้า

เพิ่ม

ID	ชื่อ
26	Ziko5555
27	Ziko the customer
29	SuperCustomer
30	Midas

ลูกค้าใหม่

ชื่อ

Midas

- จะได้รับข้อความแจ้งข้อผิดพลาดว่าชื่อสินค้านี้มีอยู่แล้ว

The name has already been taken.

×

รายชื่อลูกค้า

เพิ่ม

ID	ชื่อ	สร้างเมื่อ	แก้ไขเมื่อ
26	Ziko5555	2014-10-27 15:52:12	2014-10-27 16:12:34
27	Ziko the customer	2014-10-27 16:38:16	2014-10-27 16:38:16
29	SuperCustomer	2014-11-14 21:20:11	2014-11-14 21:20:11
30	Midas	2014-11-15 13:32:14	2014-11-15 13:32:14

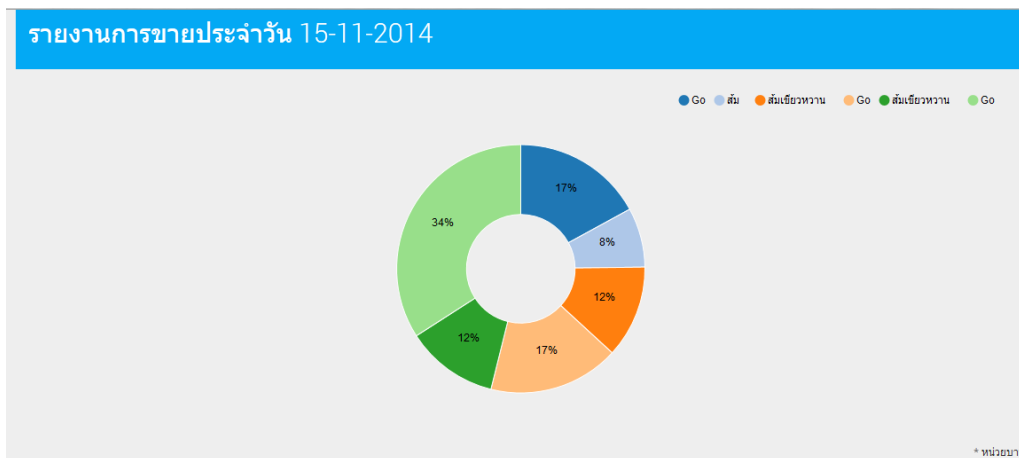
- ทดสอบกับระบบจัดการประวัติการขาย

- ไปที่หน้าต่าง “รายงานการขาย”

- ทดลองพิมพ์ วัน/เดือน/ปี ที่ไม่ถูกต้องลงไปตามรูป

- จะมีข้อความแจ้งเตือนดังรูป

- หากใส่ข้อมูล วัน/เดือน/ปี ที่ถูกต้องก็จะสามารถดูรายงานการขายได้



- ผลที่ได้จากการทดลอง
 - ระบบมีการตรวจสอบหาก user ป้อน input ที่ไม่ถูกต้อง
- สรุปและสิ่งที่ได้จากการทดลองนี้
 - application มีการตรวจสอบความผิดพลาดของ input อยู่ในหลายๆระบบ ทั้งนี้ การทดลองที่หยิบยกมานั้นไม่ได้ครอบคลุมทุกระบบอาจจะยังมีข้อผิดพลาดได้

บทสรุป

ระบบบริหารจัดการร้านค้า POS4Shop ทำให้ผู้จัดทำนั้นได้เข้าใจถึงการทำงานของร้านค้าและการขายสินค้าและยังได้ฝึกสร้างประสบการณ์การใช้ framework , Object-oriented design และรวมไปถึงการ implement web-application ทั้งนี้ผลงานที่ออกมาอาจจะยังไม่ใช่ผลงานที่เรียกว่ายอดเยี่ยมหรือสมบูรณ์แบบและยังมีข้อผิดพลาดอยู่บ้าง แต่ทางกลุ่มของเราก็ได้ความรู้จากการทำโครงงานนี้ไปมากและเชื่อว่ายังสามารถนำ application นี้ไปพัฒนาและต่อยอดได้และการทำโครงงานนี้ยังทำให้สมาชิกในกลุ่มรู้สึกสนุกไปกับการทำงานอีกด้วย

บรรณานุกรม

- Dayle Rees. Laravel:Code Bright. Leanpub,2014-06-01
- JeffreyWay. Laravel Testing Decoded. Leanpub,2013-05-28
- laravel.com/docs
- <https://www.google.co.th/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCcQjBAwAQ&url=https%3A%2F%2Fwww.digitalocean.com%2Fcommunity%2Ftutorials%2Fhow-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04&ei=nVhoVMr6DsXhuQSjtIKwCg&usg=AFQjCNFMs2tFcDvIKQSj-FG-MetmnBjf-Q&sig2=bxpmKXpmELfPB3IUOPi3gA&bvm=bv.79142246,d.c2E&cad=rja>
- <http://api.jquery.com/jquery.post/>
- <https://www.lucidchart.com/>
- <http://getbootstrap.com/>
- <https://support.cloudflare.com/hc/en-us/sections/200038156-SSL-at-CloudFlare>