**A red and white circular logo

Description automatically generated**

**คู่มือการพัฒนาระบบ ( Code Manual )**

**ระบบสั่งซื้อสินค้าออนไลน์  
(Resonance)**

**จัดทำโดย**

6604062630021 นายกฤษกร ขามวิเศษ

6604062630218 นายทองบุรินทร์ ศิริภิวัฒน์

6604062630293 นายนิธิกร หีบทอง

6604062630315 นายปฏิพน พลยุทธภูมิ

6604062630366 นายพีรพล วิสระพันธุ์

**เสนอ**

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์ **วิชา 040613306 Software Engineering**ภาคเรียนที่ 2 ปีการศึกษา 2567

**ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

**สารบัญ**

**คู่มืออธิบายการทำงานของระบบ**

ระบบ Resonance Sound Shop ใช้สถาปัตยกรรมระบบแบบ Client-Server เพื่อให้ง่ายต่อการจัดการโครงสร้างของระบบเนื่องจากมีการแบ่งการทำงานชัดเจน (Frontend , Backend) เพื่อสร้างตัวระบบการใช้งาน โดยพัฒนาด้วย Nuxt3 (Vue.js) + Django (API – driven ) + PostgresSQL

โครงสร้างโปรเจคสำหรับ Client Server Architecture

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Design Pattern**

มีการเรียบเรียงตัวโครงสร้างระบบ จัดระเบียบด้วย Design Pattern เพื่อให้ง่ายต่อการพัฒนา

1. Singleton Pattern (ตัวอย่างจากไฟล์ frontend/store/useAuth.ts*)*

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

มีการใช้ Pinia ซึ่งเป็น Store Management ทำงานแบบ Singleton ด้วยการสร้าง instance ของผู้ใช้ขึ้นมาในระบบและสามารถทำการเรียกใช้เพื่อทำงานข้ามส่วนต่างๆได้

ตัวอย่างการใช้ Store ที่เก็บไว้ในระบบ (frontend/pages/login.vue)

เรียก login จาก store

รูปภาพประกอบด้วย ข้อความ, แสดง, ภาพหน้าจอ, คอมพิวเตอร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

เช็คสิทธิ์การเข้าถึงผ่าน Store และ redirect ตามสิทธิ์

1. Proxy Pattern

มีการใช้ Proxy Pattern ในระบบในส่วนของของระบบหลังบ้านเพื่อจำกัดการ Query ของ User เนื่องจาก User มี Customer และ Admin เป็น child component สิทธ์การเข้าถึงไม่เหมือนกัน

ตัวอย่างการประกาศ Proxy ใน (backend/models/user\_model.py)

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้องรูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**เมื่อ User ถูกสร้างจะถูกกำหนดว่าสามารถเข้าถึง Query แบบใดได้**

**ตัวอย่างด้านล่างเป็นการ Query Admin**

1. Observer Pattern

มีการใช้ Observer Pattern ในการเปลี่ยนแปลง component ของ UI โดยใช้ v-model ซึ่งเป็นคุณสมบัติของ Vue3 ในการตรวจจับตัวแปรหลักและเปลี่ยนค่าตาม

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, สี่เหลี่ยมผืนผ้า

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้องตัวอย่างการใช้งาน Observer Pattern จากไฟล์ (frontend/admin/product.vue)

1. Decorator Pattern

มีการใช้ Decorator Pattern ในการตรวจสอบสิทธิ์ของ backend controller เพื่อใช้งานโดยในที่นี้จะเป็นการใช้ decorator ของ python Django เอง

ตัวอย่างจากไฟล์ (backend/cart\_controller.py)

รูปภาพประกอบด้วย ข้อความ, นามบัตร, ภาพหน้าจอ, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้องรูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Nuxt Dynamic Routing**

Nuxt.js มีระบบ routing ที่สร้างขึ้นอัตโนมัติตามโครงสร้างไฟล์ในโฟลเดอร์ pages/ ซึ่งรวมถึงความสามารถในการสร้าง dynamic routes ที่สามารถรับพารามิเตอร์ได้

* วิธีการสร้าง Dynamic Routes ใน Nuxt.js

Nuxt.js ใช้วิธีการตั้งชื่อไฟล์พิเศษเพื่อกำหนด dynamic routes:

* การใช้วงเล็บเพื่อสร้าง Dynamic Segments

เมื่อสร้างไฟล์ที่มีชื่อในวงเล็บ เช่น [id].vue หรือ [slug].vue Nuxt จะตีความส่วนนั้นเป็น dynamic parameter

ตัวอย่างการใช้ dynamic routing ในโปรเจค

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้องรูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

การใช้ dynamic routing สามารถนำไปปรับใช้กับการเรียกใช้ API และ Filter ได้

**Developing Environment**

มีการใช้ env ในการพัฒนา ป้องกันการรั่วไหลของ key หรือข้อมูล สำคัญต่างๆ อีกทั้งยังสามารถปรับเปลี่ยนไปใช้ได้กับ environment อื่นเช่น การ deploy บน vercel หรือ docker

ตัวอย่างการใช้งาน env ใน (backend/setting.py) สำหรับการตั้งค่าเชื่อมต่อ database

รูปภาพประกอบด้วย ข้อความ, แสดง, ภาพหน้าจอ, คอมพิวเตอร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

ตัวอย่าง env สำหรับการทดสอบ local

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Middleware**

ระบบมีการใช้ middleware เพื่อควบคุมสิทธิ์ในการเข้าถึงหน้าต่างๆของระบบ

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

ควบคุมการเข้าถึงหน้า admin, setting,order และcheckout โดยเช็คว่าได้ยืนยันตัวตนก่อนเข้าถึงหรือไม่

ควบคุมการเข้าถึง path reset password โดยการเช็คว่ามี token ใน URL ที่ query มาหรือไม่

ตัวอย่างการใช้งาน Middle ในไฟล์ (frontend/admin/order.vue)

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ออกแบบ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Authentication System**

ระบบมีการลงทะเบียนและยืนยันสิทธิ์โดยสามารถ ลงทะเบียนได้ทั้งในส่วนของ ลูกค้า และ พนักงานแต่หากจะลงทะเบียนให้พนักงานจำเป็นต้องมีสิทธิ์ superuser

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, เมนู

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

ตรวจสอบว่าเป็น superuser ในการสร้างบัญชีพนักงาน



**Admin / Staff System**

ระบบจะแบ่งเป็น Admin ( เจ้าของร้าน ) และ Staff (พนักงานในร้าน) โดย Admin จะมีสิทธิ์ Superuser และสามารถจัดการทุกอย่างในร้านรวมถึงพนักงานในร้านได้

ตัวอย่างฟังก์ชันสำหรับการ เรียกดูข้อมูล Dashboard

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

จากภาพระบบจะทำการตรวจสอบว่าเป็น admin หรือ superuser หรือไม่และทำการ query Order จากModel ของ Order และคืนค่าเป็น JSON Response

ตัวอย่างการเรียกใช้งาน API ของ get\_admin\_stat

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

จะทำการเรียกใช้ API โดยใช้ token ของ authStore ซึ่งเป็น Singleton ที่อธิบายไว้ก่อนหน้าในการ authentication

**Order System**

ระบบมีการจัดการ Order โดยแบ่งออกเป็น 4 Schemas ได้แก่ Order OrderItem Cart และ

CartItem ใช้สำหรับการสร้าง Order เมื่อเวลากด Place Order

Schmemas

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

ฟังก์ชันสำหรับการสร้าง Order ลง database



ส่วนของ Frontend ในการเรียก API 

**Order History System**

ระบบเพื่อให้ผู้ใช้งานสามารถเรียกดูคำสั่งซื้อที่เคยสั่งไปได้

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

การเรียกใช้งานในฝั่ง frontend

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

มีการใช้ orderService ช่วยในการเรียก API เพื่อให้ง่ายต่อการ debug ระหว่างพัฒนา

รูปภาพประกอบด้วย ข้อความ, อิเล็กทรอนิกส์, แสดง, ภาพหน้าจอ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Cart System**

ระบบที่ให้ผู้ใช้สามารถเลือกของใส่ตะกร้าได้

ตัวอย่าง ฟังก์ชันในการเรียกข้อมูลของตะกร้า และ เลือกตะกร้าของ User นั้นๆ

รูปภาพประกอบด้วย ข้อความ, อิเล็กทรอนิกส์, ภาพหน้าจอ, แสดง

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

การเรียกใช้งานฟังก์ชันของฝั่ง frontend

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**Password Recovery System**

ระบบสามารถให้ผู้ใช้กู้คืนรหัสผ่านได้โดยใช้ลิงค์ผ่านทางอีเมล

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, คอมพิวเตอร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

จากภาพเป็นเพียงบางส่วนโดยระบบจะทำการสร้าง reset token ให้และแนปลิ้งที่เป็น reset URL และส่งเมลไปด้วย Gmail SMTP service ของ django

(ต่อ)

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**API Endpoints**

ระบบยังมีฟังก์ชันอีกมากสำหรับในแต่ละหัวข้อดังนั้นจึงจะขอสรุปมาไว้ตรงนี้ครับ

**สรุป API Endpoints ของ Resonance Sound Shop**

**API Endpoints การยืนยันตัวตน (Authentication)**

* POST /api/auth/register/ - ลงทะเบียนผู้ใช้ใหม่
* POST /api/auth/login/ - เข้าสู่ระบบ
* POST /api/auth/logout/ - ออกจากระบบ
* GET /api/auth/user/ - ดึงข้อมูลผู้ใช้ปัจจุบัน
* POST /api/auth/forgot-password/ - ขอรีเซ็ตรหัสผ่าน
* POST /api/auth/reset-password/ - รีเซ็ตรหัสผ่านด้วยโทเคน
* POST /api/auth/validate-password/ - ตรวจสอบความถูกต้องของรหัสผ่าน
* POST /api/auth/validate-reset-token/ - ตรวจสอบความถูกต้องของโทเคนรีเซ็ตรหัสผ่าน
* PUT /api/auth/update-profile/ - อัปเดตโปรไฟล์ผู้ใช้
* PUT /api/auth/update-address/ - อัปเดตที่อยู่ผู้ใช้

**API Endpoints สินค้า (Products)**

* GET /api/products/ - ดึงรายการสินค้าทั้งหมด
* GET /api/products/recommendations/ - ดึงสินค้าแนะนำ
* GET /api/products/filters/ - ดึงตัวเลือกการกรองสินค้า
* GET /api/products/<category>/ - ดึงสินค้าตามหมวดหมู่
* GET /api/products/<category>/<id>/ - ดึงรายละเอียดสินค้าตามหมวดหมู่และไอดี
* GET /api/products/<id>/ - ดึงรายละเอียดสินค้าตามไอดี
* GET /api/products/check-stock/<id>/ - ตรวจสอบสถานะสต็อกสินค้า

**API Endpoints คำสั่งซื้อ (Orders)**

* POST /api/orders/create/ - สร้างคำสั่งซื้อใหม่
* GET /api/orders/ - ดึงรายการคำสั่งซื้อของผู้ใช้
* GET /api/orders/<id>/ - ดึงรายละเอียดคำสั่งซื้อ
* POST /api/orders/<id>/payment/ - ประมวลผลการชำระเงินสำหรับคำสั่งซื้อ

**API Endpoints ผู้ดูแลระบบ (Admin)**

* GET /api/staff/stats/ - ดึงสถิติสำหรับแดชบอร์ดผู้ดูแลระบบ
* GET /api/staff/products/ - ดึงรายการสินค้าทั้งหมด (สำหรับผู้ดูแลระบบ)
* POST /api/staff/products/ - สร้างสินค้าใหม่
* PUT /api/staff/products/<id>/ - อัปเดตสินค้า
* DELETE /api/staff/products/<id>/ - ลบสินค้า
* GET /api/staff/orders/ - ดึงรายการคำสั่งซื้อทั้งหมด (สำหรับผู้ดูแลระบบ)
* PUT /api/staff/orders/<id>/ - อัปเดตสถานะคำสั่งซื้อ
* POST /api/staff/products/<id>/upload-image/ - อัปโหลดรูปภาพสินค้า
* DELETE /api/staff/products/<id>/delete-image/ - ลบรูปภาพสินค้า

**API Endpoints การจัดการพนักงาน**

* GET /api/admin/staff/ - ดึงรายชื่อพนักงานทั้งหมด
* GET /api/admin/staff/<id>/ - ดึงข้อมูลพนักงาน
* PUT /api/admin/staff/<id>/ - อัปเดตข้อมูลพนักงาน
* DELETE /api/admin/staff/<id>/ - ลบพนักงาน

**API Endpoints อื่นๆ**

* GET /api/categories/ - ดึงหมวดหมู่ทั้งหมด
* GET /api/user/get\_user\_by\_id/<id>/ - ดึงข้อมูลผู้ใช้ตามไอดี
* GET /keepalive/ - เช็คการเชื่อมต่อกับเซิร์ฟเวอร์

**การจัดการ User Interface (UI)**

โปรเจคนี้ทำการจัดการ UI ด้วย DaisyUI และ Tailwind ในการตกแต่งส่วนต่างๆ โดย DaisyUI เปรียบเสมือน Tailwind สำเร็จรูป ช่วยลดการที่ต้องเขียนคลาส Tailwind ยาวๆได้

ตัวอย่างจากไฟล์ (frontend/component/hero\_section.vue)

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, คอมพิวเตอร์

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

โดยในรูปตัวอย่าง กรอบสีแดงจะเป็นการใช้ DaisyUI สำหรับการทำ banner ที่มีภาพพื้นหลังซ้อนส่วนกรอบสีเหลืองจะเป็นการใช้ Tailwind CSS ในการกำหนดตกแต่งตามปกติ