

CS 2567

โครงการคอมพิวเตอร์

ระบบ POS ร้านขายของตกแต่งบ้าน

POS system home decoration store

โดย

| | | |
|-------------|--------------------------|-------|
| 653380015-9 | นางสาวจิตาพร เติมสุข | sec.1 |
| 653380120-2 | นายกัมแพงเพชร สิงห์ชรรณ | sec.1 |
| 653380130-9 | นางสาวทักษณันท์ แก้วกลม | sec.1 |
| 653380265-6 | นายณรงค์ฤทธิ์ พาอามาศ | sec.1 |
| 653380279-5 | นางสาวมุตธิญา จันดาววงศ์ | sec.1 |

เสนอ

รศ. ดร.ปัญญาพล หอระตะ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา CP353002 หลักการพัฒนาซอฟต์แวร์
ภาคเรียนที่ 1 ปีการศึกษา 2567

ภาควิชาวิทยาการคอมพิวเตอร์ วิทยาลัยการคอมพิวเตอร์

มหาวิทยาลัยขอนแก่น

เดือน กันยายน พ.ศ. 2567

วิทยาลัยการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น

ชื่อหัวข้อโครงการภาษาไทย : ระบบ POS ร้านขายของตกแต่งบ้าน
ชื่อหัวข้อโครงการภาษาอังกฤษ : POS system home decoration store
ชื่อผู้จัดทำโครงการ : นางสาวฐิตาพร เติมสุข รหัสนักศึกษา 653380015-9 sec.1
 นายกัมแพงเพชร สิงห์รรณ รหัสนักศึกษา 653380120-2 sec.1
 นางสาวทักษันนท์ แก้วกลม รหัสนักศึกษา 653380130-9 sec.1
 นายณรงค์ฤทธิ์ พาอามาศ รหัสนักศึกษา 653380265-6 sec.1
 นางสาวมุตธิชา จันดาววงศ์ รหัสนักศึกษา 653380279-5 sec.1
ที่ปรึกษาโครงการ : รศ. ดร.ปัญญาพล หอระตะ
สาขาวิชา : วิทยาการคอมพิวเตอร์ คณะวิทยาลัยการคอมพิวเตอร์
ปีการศึกษา : 2567

บทคัดย่อ

ระบบ POS ร้านขายของตกแต่งบ้านถูกพัฒนาขึ้นเพื่อเพิ่มประสิทธิภาพในการจัดการการขายสินค้าและบริการภายในร้าน โดยระบบนี้ออกแบบมาเพื่อตอบสนองความต้องการของผู้ใช้งานหลักสองกลุ่มคือ พนักงานขาย (Cashier) และผู้ดูแลระบบ (Administrator) สำหรับพนักงานขาย ระบบจะช่วยในการสร้างคำสั่งซื้อ การคำนวนราคาสินค้า และในขณะที่ผู้ดูแลระบบสามารถติดตามข้อมูลสินค้า ผู้ใช้งาน รวมถึงรายงานการขายและสต็อกสินค้าได้อย่างมีประสิทธิภาพ

นอกจากนี้ ระบบยังถูกออกแบบภายใต้หลักการ SOLID ซึ่งช่วยเพิ่มความยืดหยุ่นในการบำรุงรักษาและขยายระบบในอนาคต ด้วยการรองรับการเชื่อมต่อกับฐานข้อมูลผ่าน Spring Boot และการจัดเก็บข้อมูลที่เป็นมาตรฐาน ระบบ POS นี้ยังสามารถรองรับการพิมพ์ใบเสร็จ เพื่อให้การทำงานของร้านขายของตกแต่งบ้านมีความราบรื่นและมีประสิทธิภาพมากยิ่งขึ้น

สารบัญ

| เรื่อง | หน้า |
|--|------|
| บทคัดย่อ | ๑ |
| สารบัญ | ๒ |
| สารบัญตาราง | ๓ |
| สารบัญรูปภาพ | ๔ |
| บทที่ 1 บทนำ | |
| 1.1 ที่มาและความสำคัญ | 1 |
| 1.2 วัตถุประสงค์ | 1 |
| 1.3 ขอบเขตของงาน | 1 |
| 1.4 ผลที่คาดว่าจะได้รับ | 2 |
| 1.5 กระบวนการดำเนินงาน | 2 |
| บทที่ 2 ทฤษฎีและเครื่องมือที่เกี่ยวข้อง | |
| 2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง | 4 |
| 2.1.1 แนวคิดและทฤษฎีเกี่ยวกับการพัฒนาระบบ | 4 |
| 2.1.2 แนวคิดและทฤษฎีเกี่ยวกับระบบฐานข้อมูล | 6 |
| 2.2 เครื่องมือที่เกี่ยวข้อง | 9 |
| 2.2.1 Spring Boot | 9 |
| 2.2.2 Maven | 9 |
| 2.2.3 Hibernate | 10 |
| 2.2.4 MySQL | 10 |
| 2.2.5 Thymeleaf | 11 |
| 2.2.6 Git | 11 |
| 2.2.7 Visual Studio Code / IntelliJ IDEA | 11 |
| 2.2.8 Postman | 11 |
| 2.3 ภาษาที่ใช้เขียน | 12 |
| 2.3.1 JAVA | 12 |
| 2.3.2 HTML | 12 |
| 2.3.3 CSS | 13 |
| 2.3.4 JavaScript | 13 |

สารบัญ (ต่อ)

| เรื่อง | หน้า |
|---|------|
| บทที่ 3 การออกแบบและการใช้งาน | |
| 3.1 การออกแบบการพัฒนาระบบ | 15 |
| 3.1.1 วิเคราะห์ความต้องการของผู้ใช้งาน | 15 |
| 3.1.2 การนำหลัก SOLID ใน Spring ส่วนใหญ่มีมาประยุกต์กับระบบ | 15 |
| 3.2 Unified Modeling Language (UML) | 16 |
| 3.2.1 Use Case Diagram | 16 |
| 3.2.2 Use Case Text | 17 |
| 3.2.3 Sequence Diagram | 18 |
| 3.2.4 Class Diagram | 19 |
| 3.3 การออกแบบฐานข้อมูลของระบบ | 20 |
| 3.3.1 ER Diagram | 20 |
| บทที่ 4 ผลการพัฒนาระบบ | |
| 4.1 ผลการพัฒนาโปรแกรม | 21 |
| 4.2 ผลการทดสอบโปรแกรม | 100 |
| บทที่ 5 อภิปรายและเสนอแนะ | |
| 5.1 สรุปผลการดำเนินงาน | 126 |
| 5.2 การทำงานในการพัฒนาระบบ | 126 |
| 5.3 ปัญหาและข้อจำกัด | 130 |
| 5.4 ข้อเสนอแนะ | 130 |
| เอกสารอ้างอิง | ณ |

สารบัญตาราง

| | หน้า |
|--------------------------------|------|
| ตารางที่ 1 ตารางการดำเนินงาน | 2 |
| ตารางที่ 2 ตาราง Use Case Text | 17 |

สารบัญรูปภาพ

| | หน้า |
|--|------|
| ภาพที่ 1 การอธิบายการ normalization | 7 |
| ภาพที่ 2 ภาพการอธิบายการ normalization | 7 |
| ภาพที่ 3 ภาพการอธิบายการ normalization | 8 |
| ภาพที่ 4 ภาพการอธิบายการ normalization | 8 |
| ภาพที่ 5 ภาพโครงสร้างภาษา html | 12 |
| ภาพที่ 6 ภาพ Use Case Diagram | 16 |
| ภาพที่ 7 ภาพ Sequence Diagram | 18 |
| ภาพที่ 8 ภาพ Class Diagram | 19 |
| ภาพที่ 9 ภาพ ER Diagram | 20 |
| ภาพที่ 10 ภาพฐานข้อมูล | 21 |
| ภาพที่ 11 ภาพ add-category-form.html | 22 |
| ภาพที่ 12 ภาพ add-product-form.html | 23 |
| ภาพที่ 13 ภาพ add-room-form.html | 25 |
| ภาพที่ 14 ภาพ category-list.html | 26 |
| ภาพที่ 15 ภาพ edit-category-form.html | 29 |
| ภาพที่ 16 ภาพ edit-product-form.html | 30 |
| ภาพที่ 17 ภาพ edit-room-form.html | 32 |
| ภาพที่ 18 ภาพ index.html | 33 |
| ภาพที่ 19 ภาพ login.html | 34 |
| ภาพที่ 20 ภาพ product-list.html | 35 |
| ภาพที่ 21 ภาพ productDetail.html | 38 |
| ภาพที่ 22 ภาพ productDetailUser.html | 40 |
| ภาพที่ 23 ภาพ register.html | 41 |
| ภาพที่ 24 ภาพ room-list.html | 42 |
| ภาพที่ 25 ภาพ user-product.html | 45 |
| ภาพที่ 26 ภาพ AuthController.java | 48 |
| ภาพที่ 27 ภาพ HomeController.java | 51 |
| ภาพที่ 28 ภาพ ProductController.java | 52 |
| ภาพที่ 29 ภาพ RoomController.java | 54 |
| ภาพที่ 30 ภาพ UserController.java | 56 |
| ภาพที่ 31 ภาพ SecurityConfig.java | 57 |

สารบัญรูปภาพ (ต่อ)

| | หน้า |
|--|------|
| ภาพที่ 32 ภาพ UserNotFoundException.java | 59 |
| ภาพที่ 33 ภาพ DataLoader.java | 60 |
| ภาพที่ 34 ภาพ Category.java | 61 |
| ภาพที่ 35 ภาพ Product.java | 62 |
| ภาพที่ 36 ภาพ Room.java | 62 |
| ภาพที่ 37 ภาพ User.java | 63 |
| ภาพที่ 38 ภาพ UserRepository.java | 64 |
| ภาพที่ 39 ภาพ CategoryService.java | 65 |
| ภาพที่ 40 ภาพ CustomUserDetailsService.java | 66 |
| ภาพที่ 41 ภาพ ProductService.java | 67 |
| ภาพที่ 42 ภาพ RoomService.java | 68 |
| ภาพที่ 43 ภาพ UserService.java | 70 |
| ภาพที่ 44 ภาพ WebConfig.java | 70 |
| ภาพที่ 45 ภาพ WebsiteHouselyApplication.java | 71 |
| ภาพที่ 46 ภาพ CategoryController.java | 72 |
| ภาพที่ 47 ภาพ OrderController.java | 73 |
| ภาพที่ 48 ภาพ ProductController.java | 75 |
| ภาพที่ 49 ภาพ RoomController.java | 76 |
| ภาพที่ 50 ภาพ Category.java | 78 |
| ภาพที่ 51 ภาพ CustomerOrder.java | 79 |
| ภาพที่ 52 ภาพ OrderItem.java | 80 |
| ภาพที่ 53 ภาพ OrderItemKey.java | 81 |
| ภาพที่ 54 ภาพ Product.java | 82 |
| ภาพที่ 55 ภาพ Room.java | 83 |
| ภาพที่ 56 ภาพ CategoryConfig.java | 84 |
| ภาพที่ 57 ภาพ CategoryRepository.java | 85 |
| ภาพที่ 58 ภาพ OrderItemRepository.java | 85 |
| ภาพที่ 59 ภาพ OrderRepository.java | 86 |
| ภาพที่ 60 ภาพ ProductConfig.java | 87 |
| ภาพที่ 61 ภาพ ProductRepository.java | 87 |
| ภาพที่ 62 ภาพ RoomConfig.java | 88 |

สารบัญรูปภาพ (ต่อ)

| | หน้า |
|--|------|
| ภาพที่ 63 ภาพ RoomRepository.java | 89 |
| ภาพที่ 64 ภาพ CategoryService.java | 90 |
| ภาพที่ 65 ภาพ OrderItemService.java | 91 |
| ภาพที่ 66 ภาพ OrderService.java | 92 |
| ภาพที่ 67 ภาพ ProductService.java | 93 |
| ภาพที่ 68 ภาพ RoomService.java | 94 |
| ภาพที่ 69 ภาพ ApiHouselyApplication.java | 95 |
| ภาพที่ 70 ภาพหน้าลงทะเบียน (Register) | 95 |
| ภาพที่ 71 ภาพหน้าการเข้าสู่ระบบ (Login) | 96 |
| ภาพที่ 72 ภาพหน้าแรก | 96 |
| ภาพที่ 73 ภาพหน้าจัดการสินค้า | 97 |
| ภาพที่ 74 ภาพหน้ารายละเอียดสินค้า | 97 |
| ภาพที่ 75 ภาพหน้าเพิ่มสินค้าใหม่ | 98 |
| ภาพที่ 76 ภาพตัวอย่างการกรอกข้อมูลเพิ่มสินค้าใหม่ | 99 |
| ภาพที่ 77 ภาพໂປຣ໌ທີ່หน้าแสดงสินค้าເມື່ອບັນທຶກຂໍ້ມູນສິນຄ້າລະຮູານຂໍ້ມູນເຮືອບຮ້ອຍ | 100 |
| ภาพที่ 78 ภาพหน้า Edit Product | 100 |
| ภาพที่ 79 ภาพตัวอย่างการໃຫ້ຝຶກ໌ພັນຄັນຫາ | 100 |
| ภาพที่ 80 ภาพตัวอย่างເນື່ອເມື່ອກົດປຸ່ມລັບການຄັນຫາ | 101 |
| ภาพที่ 81 ภาพตัวอย่างເນື່ອເມື່ອກົດປຸ່ມ delete | 101 |
| ภาพที่ 82 ภาพหน้า Categories | 103 |
| ภาพที่ 83 ภาพหน้าเพิ่มหมวดหมู่ใหม่ | 104 |
| ภาพที่ 84 ภาพตัวอย่างการเพิ่มหมวดหมู่ใหม่ | 104 |
| ภาพที่ 85 ภาพหน้าແກ້ໄຂຂໍ້ມູນหมวดหมู่ | 105 |
| ภาพที่ 86 ภาพตัวอย่างการลบหมวดหมู่ການຝຶກ໌ນີລົບສໍາເຮົ້າ | 106 |
| ภาพที่ 87 ภาพตัวอย่างการລົບหมวดหมู่ການຝຶກ໌ນີລົບໄມ່ສໍາເຮົ້າ | 107 |
| ภาพที่ 88 ภาพตัวอย่างກົງຝຶກ໌ພັນຄັນຫາ | 108 |
| ภาพที่ 89 ภาพหน้าเพิ่ມห้องใหม่ | 109 |
| ภาพที่ 90 ภาพหน้าຕົວຢ່າງເພີ່ມຫ້ອງໃໝ່ | 109 |
| ภาพที่ 91 ภาพหน้าຕົວຢ່າງແກ້ໄຂຫ້ອງ | 110 |
| ภาพที่ 92 ภาพตัวอย่างຝຶກ໌ພັນຄັນການລົບຂໍ້ມູນຫ້ອງ | 112 |
| ภาพที่ 93 ภาพຕົວຢ່າງຝຶກ໌ພັນຄັນການລົບຂໍ້ມູນຫ້ອງການຝຶກ໌ນີລົບໄມ່ສໍາເຮົ້າ | 113 |

สารบัญรูปภาพ (ต่อ)

| | หน้า |
|---|------|
| ภาพที่ 94 ภาพหน้าการซื้อสินค้า | 114 |
| ภาพที่ 95 ภาพตัวอย่างการเลือกสินค้าจากหมวดหมู่ | 115 |
| ภาพที่ 96 ภาพตัวอย่างการเลือกสินค้าจากห้อง | 123 |
| ภาพที่ 97 ภาพตัวอย่างการเพิ่มสินค้าลงตระกร้า | 124 |
| ภาพที่ 98 ภาพแสดงปุ่มล็อกເຂົ້າໃນหน้า product | 125 |
| ภาพที่ 99 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ housely-website | 127 |
| ภาพที่ 100 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ api-housely | 128 |
| ภาพที่ 101 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ HouselyWebsite | 128 |
| ภาพที่ 102 ภาพแสดงประวัติการทำงานของทีมผ่าน github ชื่อ IKEA_Project101 | 129 |
| ภาพที่ 103 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ Furniture-selling-website | 129 |
| ภาพที่ 104 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ HouselyServerAPI | 130 |

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันที่เทคโนโลยีมีบทบาทสำคัญต่อธุรกิจร้านขายของตกแต่งบ้านจึงต้องเพิ่มความท้าทายในการจัดการการขายและการบริการลูกค้าให้มีประสิทธิภาพมากขึ้น ระบบ POS (Point of Sale) ถูกพัฒนาขึ้นเพื่อตอบสนองความต้องการในการทำงานของร้านค้าในด้านนี้ โดยเฉพาะการช่วยในการจัดการคำสั่งซื้อ การคำนวณราคาสินค้า และการจัดการข้อมูลสินค้าตกแต่งบ้าน

ระบบ POS มีความสำคัญในการทำให้การขายสินค้าตกแต่งบ้านเป็นไปอย่างราบรื่น และลดระยะเวลาในการทำงาน ช่วยให้พนักงานขายสามารถให้บริการลูกค้าได้อย่างรวดเร็ว นอกจากนี้ ระบบยังสามารถช่วยผู้ดูแลระบบในการจัดการข้อมูลสินค้าคงคลังได้อย่างมีประสิทธิภาพ ลดปัญหาการขาดแคลนหรือมีสินค้าคงเหลือมากเกินไป

การใช้ระบบ POS ยังช่วยให้ร้านขายของตกแต่งบ้านสามารถติดตามการขายและวิเคราะห์แนวโน้มต่าง ๆ ได้อย่างแม่นยำ ทำให้สามารถปรับกลยุทธ์การตลาดและการบริหารจัดการสินค้าที่ตอบสนองต่อความต้องการของลูกค้า ได้ดียิ่งขึ้น โดยรวมแล้ว ระบบ POS ไม่เพียงแต่ช่วยในการจัดการการขายเท่านั้นแต่ยังช่วยสร้างความพึงพอใจให้กับลูกค้า และสนับสนุนการเติบโตของธุรกิจอีกด้วย

1.2 วัตถุประสงค์

1.2.1 เพื่อช่วยให้พนักงานขายสามารถดำเนินการขายสินค้าได้อย่างรวดเร็วและแม่นยำ มีระบบที่สามารถคำนวณราคาได้

1.2.2 เพื่อให้ร้านค้าสามารถตรวจสอบสินค้าได้แบบเรียลไทม์ และสามารถจัดการเพิ่ม แก้ไข หรือลบสินค้าจากระบบได้便捷

1.2.5 เพื่อป้องกันการเข้าถึงข้อมูลที่ไม่พึงประสงค์และควบคุมการใช้งานระบบ โดยใช้การยืนยันตัวตนและการกำหนดสิทธิ์ของผู้ใช้งาน

1.3 ขอบเขตของงาน

1.3.1 ด้านผู้ใช้งานระบบ

- ผู้ดูแลระบบ (Administrator) ผู้ที่มีสิทธิสูงสุดในการจัดการระบบทั้งหมด เช่น การเพิ่มผู้ใช้งาน และการตั้งค่าระบบ

- พนักงานขาย (Cashier) ผู้ที่ใช้ระบบ POS เพื่อดำเนินการขายสินค้าตกแต่งบ้าน ตรวจสอบสต็อกสินค้า

1.3.2 ด้านข้อมูล

- ข้อมูลสินค้า (Product Data) ข้อมูลสินค้าตกแต่งบ้าน เช่น ชื่อสินค้า ราคา หมวดหมู่ สถานะสต็อก รายละเอียดสินค้า และรูปภาพ

- ข้อมูลคำสั่งซื้อ (Order Data) ข้อมูลคำสั่งซื้อ เช่น หมายเลขคำสั่งซื้อ รายการสินค้า ยอดรวมการสั่งซื้อ

- ข้อมูลสต็อกสินค้า (Inventory Data) ข้อมูลเกี่ยวกับจำนวนสินค้าที่มีอยู่ในสต็อก พร้อมทั้งการอัปเดตเมื่อมีการขายหรือเติมสินค้า

1.3.3 ด้านระยะเวลา เดือนมิถุนายน พ.ศ. 2567 – กันยายน พ.ศ. 2567

ตารางที่ 1 ตารางการดำเนินงาน

| การดำเนินงาน | มิถุนายน | | | | | | กรกฎาคม | | | | | สิงหาคม | | | | | กันยายน | | | |
|-------------------------|----------|---|---|---|---|---|---------|---|---|---|---|---------|---|---|---|---|---------|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 |
| 1. คิดหัวข้อโครงงาน | | | | | | | | | | | | | | | | | | | | |
| 2. วิเคราะห์ความต้องการ | | | | | | | | | | | | | | | | | | | | |
| 3. ออกแบบระบบ | | | | | | | | | | | | | | | | | | | | |
| 4. เขียนโปรแกรม | | | | | | | | | | | | | | | | | | | | |
| 5. การพัฒนาและทดสอบ | | | | | | | | | | | | | | | | | | | | |
| 6. จัดทำรายงาน | | | | | | | | | | | | | | | | | | | | |
| 7. นำเสนอโครงงาน | | | | | | | | | | | | | | | | | | | | |

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 ระบบ POS จะช่วยให้พนักงานขายสามารถทำธุรกรรมได้รวดเร็วและมีประสิทธิภาพมากขึ้น ซึ่งจะช่วยลดเวลาที่ใช้ในการขาย

1.4.2 ระบบจะช่วยให้สามารถตรวจสอบและจัดการสต็อกสินค้าได้แบบเรียลไทม์ ทำให้ลดการขาดสต็อกและเพิ่มความสามารถในการจัดการสินค้าในร้าน

1.4.3 การบันทึกข้อมูลการขายและสต็อกจะมีความถูกต้องมากขึ้น ลดข้อผิดพลาดที่อาจเกิดจากการทำงานด้วยมือ

1.4.4 ลูกค้าจะได้รับบริการที่รวดเร็วและมีคุณภาพ เช่น การชำระเงินที่สะดวกสบาย ความรวดเร็วในการคิดเงิน

1.4.5 ระบบจะมีการควบคุมการเข้าถึงข้อมูลที่มีความปลอดภัย ช่วยป้องข้อมูลสำคัญของร้านค้าและลูกค้า

1.4.6 มีความสามารถในการขยายระบบในอนาคต มีความยืดหยุ่นในการพัฒนาฟีเจอร์ใหม่ ๆ

1.4.7 ระบบที่ใช้งานง่ายจะช่วยให้พนักงานสามารถทำงานได้อย่างมีความสุข และลดความเครียดจากการทำงาน

1.5 กระบวนการดำเนินงาน

1.5.1 วิเคราะห์ความต้องการ

รวบรวมและวิเคราะห์ความต้องการของผู้ใช้งาน คือ พนักงานขาย ผู้จัดการร้าน ลูกค้า รวมถึงพั้งก์ชันหลักที่ระบบต้องรองรับ เช่น การจัดการสินค้า การจัดการสต็อก และการรายงานยอดขาย

1.5.2 ออกแบบระบบ

เลือกใช้สถาปัตยกรรม MVC (Model-View-Controller) โดยแยกส่วนการทำงานเป็น Model Controller Service และ Repository เพื่อให้ระบบมีความยืดหยุ่นและง่ายต่อการบำรุงรักษา

การออกแบบฐานข้อมูล ออกแบบโครงสร้างฐานข้อมูลสำหรับจัดการสินค้า คำสั่งซื้อ สต็อก และข้อมูลลูกค้า เช่น ตารางสำหรับเก็บข้อมูลสินค้า ตารางหมวดหมู่ของสินค้า ตารางห้องของสินค้า

การออกแบบ API โดยวางแผนการสร้าง RESTful API สำหรับการจัดการสินค้า คำสั่งซื้อ สต็อก

การออกแบบ UX/UI ออกแบบหน้าจอให้มีความง่ายต่อการใช้งาน ทั้งในส่วนพนักงานขาย ได้แก่ หน้าจอสำหรับการจัดการสินค้าและคำสั่งซื้อ รวมถึงการชำระเงิน

1.5.4 การพัฒนา

พัฒนา API และบริการต่าง ๆ สำหรับการจัดการสินค้า คำสั่งซื้อ และสต็อก โดยใช้ Spring Boot พร้อมเชื่อมต่อกับฐานข้อมูลผ่าน Spring Data JPA

พัฒนาโมดูลสำหรับตรวจสอบและอัปเดตสต็อกสินค้าอัตโนมัติเมื่อมีการขายสินค้า

1.5.5 การทดสอบ

ทดสอบแต่ละโมดูลในระบบเพื่อให้แน่ใจว่าทำงานได้ถูกต้อง

1.5.6 การวิเคราะห์ผลลัพธ์และปรับปรุง

วิเคราะห์ข้อมูลการขายและประสิทธิภาพของระบบ เพื่อนำไปพัฒนาระบบการทำงานและการบริการลูกค้าในอนาคต

บทที่ 2

ทฤษฎีและเครื่องมือที่เกี่ยวข้อง

2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง

2.1.1 แนวคิดและทฤษฎีเกี่ยวกับการพัฒนาระบบ

(1) SOLID Principle

S.O.L.I.D เป็นหลักการออกแบบที่จะช่วยให้รู้ว่าควรจัดการอย่างไรกับ function และโครงสร้างข้อมูลใน class รวมถึงการส่งข้อมูลไปมาระหว่าง class ด้วย โดยมีเป้าหมายสำคัญ 3 ข้อคือ

1. ให้มีความคงทน ต่อการเปลี่ยนแปลงในอนาคต
2. ให้เข้าใจง่าย
3. ให้มันเป็นเรื่องพื้นฐานที่ใช้ได้กับทุกระบบ

หลักการของ SOLID มีมาอย่างยาวนาน โดยเริ่มต้นช่วงทศวรรษที่ 80 โดยเริ่มมีการพูดคุยกันเกี่ยวกับหลักการออกแบบ software บน USENET (Facebook สมัยบรรพบุรุษ) ในช่วงหลายปีที่ผ่านมาจากวันนั้น หลักการต่าง ๆ ได้เปลี่ยนแปลงไป บางส่วนถูกกลบไป บางส่วนถูกรวบเข้าด้วยกัน จนกระทั่งช่วงปี 2000 ได้มีการคิดค้นคำว่า SOLID ซึ่งได้รวมหลักการออกแบบเอาไว้โดยย่อมาจากคำว่า

S: SRP — The Single Responsibility Principle

โครงสร้างระบบที่ดีที่สุด คือ 1 module ทำ 1 อย่าง และ มี 1 เหตุผลที่จะเปลี่ยนแปลงเท่านั้น เพราะการที่ทำหลายหน้าที่จะทำให้ module มีความใหญ่และซับซ้อนเกินไป รวมทั้งการเปลี่ยนแปลงจะทำได้ยากขึ้นอีกด้วย

O: OCP — The Open-Closed Principle

หลักการนี้จะอนุญาตให้มีการเพิ่ม function การทำงานใหม่ได้โดยต้องไม่ไปแตะต้องของเดิม เพราะการแก้ไขของเดิมจะต้องไป concern เกี่ยวกับการระบบทรัพการทำงานดังเดิมด้วย แต่การออกแบบให้สามารถขยายได้ตลอดจะทำให้มีต้องกังวลในเรื่องนี้

L: LSP — The Liskov Substitution Principle

Class ลูกทุกตัวจะต้องสามารถทำหน้าที่แทน class แม้ได้ทั้งหมด ถ้ามี class B ที่สืบทอดมาจาก class A แล้วเราสามารถใช้ function ใน class B แทนการเรียกจาก class A ได้หมด ถ้าเกิดยังมี function บางอย่างที่เราต้องไปเรียกใช้จาก super class อยู่แสดงว่าผิดหลักในข้อนี้

I: ISP — The Interface Segregation Principle

Code ของเรา ไม่ควรประกาศอะไรที่ไม่ได้ใช้เอามาก โดยจะทำข้อนี้ได้เราต้องระลึกเสมอว่า module ที่เราเขียนมานี้ทำเพื่อใคร และเข้าต้องการอะไรบ้าง หลายครั้งที่เราสร้างโปรแกรมโดยรวมทุก function ไว้ใน module เดียวกันหมด และ user ทุกคนมาเรียกใช้ที่เดียวกันหมด ซึ่ง function บางอย่าง บางคนก็ไม่ได้ใช้ ดังนั้นจะต้องแยก module สำหรับ user แต่ละกลุ่ม และให้ใช้งานได้เพียง function ที่ตัวเองใช้งานจริงเท่านั้น

D: DIP — The Dependency Inversion Principle

High level module ไม่ควรยึดกับ low level module และ high level ไม่ควรรู้รายละเอียดการทำงานของ low level เพราะถ้าวันหนึ่งเราจะมีการเปลี่ยนแปลง low level module ขึ้นมาแปลงว่า high level module ก็จะต้องเปลี่ยนแปลงด้วย อีกทั้งเราจะไม่สามารถ reuse function ตั้งกล่าวได้อีกด้วย [1]

(2) Model-View-Controller (MVC) Architecture

MVC (Model, View, Controller) คือ Software Design Pattern อีกรูปแบบหนึ่งที่ได้รับความยอดนิยมในโลกของการพัฒนาซอฟต์แวร์ ยึดหลักของการทำงานที่แยกส่วนหน้าที่ต่อ กันชัดเจน ทำให้มีความสามารถในการจัดการและบำรุงรักษาแอปพลิเคชันได้อย่างดีเยี่ยม [2]

Model หมายถึง ส่วนที่ทำหน้าที่จัดการเกี่ยวกับข้อมูลทุกรูปแบบ เช่น การรับข้อมูลจากอินพุตต่างๆ การบันทึกข้อมูล หรือ การจัดการฐานข้อมูล โดย Model จะทำหน้าที่ในการแปลงข้อมูลไม่ว่าจะมาจากที่ไหนก็ตาม ให้อยู่ในรูปแบบที่ระบบเข้าใจและสามารถนำไปใช้งานต่อได้ ยกตัวอย่าง เช่น เราจะใช้ Model ใน การเชื่อมต่อกับฐานข้อมูล MySQL และ MongoDB ซึ่งเป็นฐานข้อมูลคนละประเภทกัน โดยใช้ Model คนละตัว เชื่อมต่อกับฐานข้อมูลแต่ละประเภทแล้วแปลงผลลัพธ์ออกมาให้เป็นรูปแบบเดียวกัน (เช่น JSON) ซึ่งผลลัพธ์ที่ได้จะออกมากเหมือนกันทำให้เราสามารถเชื่อมต่อกับฐานข้อมูลได้หากได้เพียงแค่เปลี่ยน Model

View หมายถึง ส่วนที่ทำหน้าที่จัดการแสดงผล เช่น Template (HTML) หรือแม้กระทั่ง Template Engine โดยหลักการที่สำคัญของ MVC คือการแยกส่วนการแสดงผลออกจากโค้ดส่วนประมวลผล เพื่อให้โค้ดเรียบง่าย และ บำรุงรักษาง่ายยั่งยืน

Controller เปรียบเสมือนส่วนสมองของโปรแกรม ทำหน้าที่เป็นแกนของระบบในการเชื่อมต่อกับ Model และ View ซึ่งโดยปกติแล้ว เมื่อมีการเรียกหน้าเว็บ ก็จะมีการเรียกไปยัง Controller เมื่อ Controller ต้องการข้อมูลก็จะไปร้องขอมาจาก Model และเมื่อต้องการแสดงผล ก็จะส่งไปยัง View เพื่อจัดการแสดงผล [3]

ตัวอย่างการทำงาน ของตัวอย่างหน้าฟอร์ม ตัวฟอร์มที่เราเห็นก็คือ User Interface ที่จะมีช่องกรอกฟอร์ม ซึ่งก็ประกอบไปด้วยฟิลด์ต่าง ๆ และมีปุ่มสำหรับกด submit ฟอร์ม ส่วนนี้จะเป็นหน้าที่ของ View (V) และ ซึ่งก็เปรียบเสมือน entry point

หลังจากที่เราได้กด submit แล้ว ก็จะมีการส่ง request (POST) ของฟอร์มไปที่ผู้ให้บริการ server โดยถัดไปก็จะเป็นหน้าที่ของ Controller (C) ที่รับ request มาจาก View และนำมาระบบผลลัพธ์ที่ผู้ให้ร้องขอมาซึ่งแน่นอนว่าเป็น form submission จากนั้น Controller จะทำการ validate ฟอร์มที่ถูกส่งมา เมื่อไม่มีปัญหาอะไร ก็จัดการ save เข้าไปในฐานข้อมูล

ส่วนที่ save เข้าไปในฐานข้อมูลก็จะมีความเกี่ยวข้องกับ Model (M) เพราะว่าฟิลด์ต่าง ๆ ในฟอร์มที่ถูก submit มาล้วนแล้วแต่ต้องถูกนำไปบันทึกในฐานข้อมูล เมื่อบันทึกสำเร็จแล้วก็จะจบหน้าที่ของ Model จากนั้นผู้ให้บริการจะรีเทิร์นข้อมูลของรีบาร์งอย่างอุ่นๆ ไปที่ผู้ให้บริการ (View) เช่น ส่งข้อความแจ้งเตือนหรือแสดงผลว่าบันทึกฟอร์มสำเร็จแล้ว เป็นต้น [2]

2.1.2 แนวคิดและทฤษฎีเกี่ยวกับระบบฐานข้อมูล

(1) ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS)

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) คือระบบจัดการฐานข้อมูลที่จัดระเบียบและจัดเก็บข้อมูลในตารางที่มีແຄາ (บันทึก) และคอลัมน์ (ช่อง) คิดค้นโดย Edgar F. Codd ในปี 1970 โดยเดลเชิงสัมพันธ์ได้กล่าวเป็นแนวทางหลักในการจัดการข้อมูลที่มีโครงสร้างในแอปพลิเคชันซอฟต์แวร์สมัยใหม่ RDBMS อนุญาตให้แอปพลิเคชันซอฟต์แวร์ดำเนินการต่างๆ กับข้อมูล เช่น สร้าง อ่าน อัปเดต และลบ โดยใช้ภาษาคิวอาร์ที่เรียกว่า Structured Query Language (SQL) โดยทั่วไป RDBMS ที่ได้รับความนิยมมากที่สุดในตลาดปัจจุบัน ได้แก่ Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL และ IBM Db2 แต่ละรายการมีคุณสมบัติและความสามารถเฉพาะตัว และการเลือก RDBMS ที่เหมาะสมกับความต้องการของคุณนั้นขึ้นอยู่กับปัจจัยต่างๆ เช่น ประสิทธิภาพ ความสามารถในการปรับขนาด ความปลอดภัย และความย่ายในการใช้งาน องค์ประกอบหลักและแนวคิดของ RDBMS ได้แก่

1. ตาราง ตารางคือการจัดเรียงข้อมูลอย่างมีโครงสร้าง ประกอบด้วยແຄາและคอลัมน์ ตารางแสดงถึงเอนทิตี้ (เช่น ลูกค้า คำสั่งซื้อ พนักงาน) โดยแต่ละคอลัมน์ในตารางสอดคล้องกับคุณลักษณะ (เช่น ชื่อลูกค้า วันที่สั่งซื้อ รหัสพนักงาน) และแต่ละແຄาแสดงถึงบันทึกแต่ละรายการ (เช่น ลูกค้ารายได้รายหนึ่ง คำสั่งพนักงาน)

2. พิลด์ พิลด์หรือที่เรียกว่าแอ็ตทริบิวต์เป็นองค์ประกอบข้อมูลแต่ละรายการภายในตาราง อธิบายคุณสมบัติของเอนทิตี้ที่ตารางเป็นตัวแทน (เช่น ชื่อลูกค้า วันที่สั่งซื้อ รหัสพนักงาน)

3. บันทึก บันทึกหรือແຄาเป็นอินสแตนซ์เดียวของข้อมูลภายในตาราง ซึ่งสอดคล้องกับเอนทิตี้เฉพาะ (เช่น John Doe ในฐานลูกค้า สั่งซื้อเมื่อ 01/01/2023 พนักงาน #123)

4. คีย์หลัก คีย์หลักคือตัวระบุเฉพาะสำหรับบันทึกในตาราง ช่วยให้มั่นใจได้ว่าแต่ละระเบียนสามารถระบุได้ไม่ซ้ำกันภายในตาราง ป้องกันไม่ให้มีรายการซ้ำกัน คีย์หลักอาจเป็นแบบธรรมดा (แอ็ตทริบิวต์เดียว) หรือแบบผสม (แอ็ตทริบิวต์รวมกัน)

5. คีย์ต่างประเทศ Foreign Key คือเขตข้อมูล (หรือชุดของเขตข้อมูล) ในตารางที่ลิงก์ไปยังคีย์หลักของตารางอื่น Foreign Key ใช้เพื่อรับรองความสัมพันธ์ระหว่างตารางและบังคับใช้ Referential Integrity (ความสอดคล้องระหว่างตารางที่เกี่ยวข้อง)

6. ดัชนี ดัชนีเป็นโครงสร้างข้อมูลเพิ่มเติมที่ใช้เพื่อปรับปรุงประสิทธิภาพของการดำเนินการดึงข้อมูลใน RDBMS ทำหน้าที่เป็นระบบการค้นหา ช่วยให้ค้นหาบันทึกเฉพาะในตารางได้รวดเร็วขึ้น คุณสามารถสร้างดัชนีได้ตั้งแต่หนึ่งคอลัมน์ขึ้นไปในตาราง ขึ้นอยู่กับข้อกำหนดของคิวอาร์ [4]

(2) Normalization

การทำ Normalization (นอร์มัลไลเซชัน) เป็นวิธีการในการกำหนด Attribute (แอ็ตทริบิวต์) ให้กับแต่ละ Entity (เอนทิตี้) เพื่อให้ได้โครงสร้างของตารางที่ดี สามารถควบคุมความซ้ำซ้อนของข้อมูลหลีกเลี่ยงความผิดปกติของข้อมูล

การทำนอร์มัลไลเซชัน จะประกอบด้วย Normal Form (นอร์มัลฟอร์ม) แบบต่าง ๆ ที่มีเงื่อนไขของ การทำให้อยู่ในรูปของนอร์มัลฟอร์มที่แตกต่างกันไป ขึ้นอยู่กับผู้ออกแบบฐานข้อมูลว่า ต้องการลดความซ้ำซ้อนในฐานข้อมูลให้อยู่ในระดับใด [5] โดยการทำางแต่ละขั้นตอนจะเป็นในลักษณะดังนี้

1NF จัดระเบียบข้อมูล อัพเดท แยก แก้ไข เป็นการจัดระเบียบข้อมูลทั่วไป โดยแต่ละ Column ต้องมี Attribute เพียง 1 ประเภท และ 1 Value เท่านั้น

2NF กำจัด Partial Dependency แยก Table ออกมาด้วยที่จะให้ Table แต่ละอันมี Primary key (PK) เพียง 1 ตัว ที่ซึ่งค่าไปยัง Attribute อื่นๆภายในตารางเท่านั้น

| Serial Number | Order Number | Name | QTY |
|---------------|--------------|---------|-----|
| C1 | 0235 | Chicken | 20 |
| C1 | 1354 | Pork | 10 |
| C2 | 1589 | Chilli | 500 |
| C3 | 5689 | Lime | 320 |

ภาพที่ 1 การอธิบายการ normalization

PK = Serial Number + Order Number

Partial Dependency = Order Number (ON สามารถซึ่งไปยัง Name และ QTY)

3NF กำจัด Transitive Dependency

| Order | Dish | Cost | Depart | Chef |
|-------|------------|------|---------|--------|
| 1 | Pork Steak | 20\$ | Grilled | J.Chun |
| 2 | Eggs | 5\$ | Boild | K.Cin |
| 3 | Beef Steak | 30\$ | Grilled | J.Chun |
| 4 | Chicken | 10\$ | Fried | T.Tom |
| 5 | Vetgetable | 8\$ | Stir | K.Wong |

ภาพที่ 2 ภาพการอธิบายการ normalization

จะเห็นว่า Depart สามารถระบุ Chef ที่ทำให้ในตอนนั้น จึงเป็น Transitive Dependency

| Order | Dish | Cost | Depart | Chef |
|-------|------------|------|---------|--------|
| 1 | Pork Steak | 20\$ | Grilled | J.Chun |
| 2 | Eggs | 5\$ | Boild | K.Cin |
| 3 | Beef Steak | 30\$ | Grilled | J.Chun |
| 4 | Chicken | 10\$ | Fried | T.Tom |
| 5 | Vetgetable | 8\$ | Stir | K.Wong |

ภาพที่ 3 ภาพการอธิบายการ normalization

4NF กำจัด Multivalued Dependency

| Serial Number | Order Number | Name |
|---------------|--------------|---------|
| C1 | 0235 | Chicken |
| C1 | 0235 | Pork |
| C2 | 1589 | Chilli |
| C2 | 1589 | Lime |

| Serial Number | Order Number | Serial Number | Name |
|---------------|--------------|---------------|---------|
| C1 | 0235 | C1 | Chicken |
| C1 | 0235 | C1 | Pork |
| C2 | 1589 | C2 | Chilli |
| C2 | 1589 | C2 | Lime |

ภาพที่ 4 ภาพการอธิบายการ normalization

5NF Join Dependency แล้ว ได้ตารางที่เหมือนเดิมคือขั้นตอนสุดท้ายการรวมข้อมูลกลับมา Join แล้วต้องได้ในลักษณะที่เหมือนทำขั้นตอน 1NF เสร็จแล้ว

การทำ Database normalization ไม่ได้มีแต่ข้อดี ข้อเสียก็มีเช่นกัน คือ ความซับซ้อนมากขึ้น ตารางมากขึ้น บางกรณีทำให้ query ซ้ำลงกว่าเดิม ทำให้พื้นที่จัดเก็บมากขึ้น เป็นอย่างที่พยากรณ์มากขึ้น [6]

(3) Structured Query Language (SQL)

SQL เป็นภาษาการเขียนโปรแกรมมาตรฐานสำหรับจัดการฐานข้อมูลเชิงสัมพันธ์ สร้างฐานข้อมูล SQL และจัดการข้อมูลในฐานข้อมูลโดยการทำงานที่แตกต่างกัน SQL ถูกนำมาใช้ในปี 1970 SQL เป็นวิธีที่ดีที่สุดในการจัดการกับฐานข้อมูลเชิงสัมพันธ์ ฐานข้อมูลเชิงสัมพันธ์จะจัดระเบียบข้อมูลเป็นตาราง [7]

สำหรับภาษา SQL นั้น จะประกอบไปด้วยภาษาใน 4 รูปแบบ ซึ่งแต่ละแบบจะมีหน้าที่เฉพาะแตกต่างกันไป

1. ภาษาสำหรับการนิยามข้อมูล (Data Definition Language: DDL)

ภาษาสำหรับการนิยามข้อมูล คือภาษาโปรแกรมที่ถูกเขียนขึ้น สำหรับใช้ในการสร้างฐานข้อมูล และกำหนดโครงสร้างของข้อมูลว่าต้องการให้เป็นลักษณะใด หรือมีชนิดของข้อมูลเป็นรูปแบบใด รวมทั้งยังเป็นคำสั่งที่ใช้ในการเปลี่ยนแปลงตารางและการสร้างด้วย โดยมีคำสั่งในการนิยามข้อมูล เช่น Create, Drop, Alter

2. ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language: DML)

ภาษาสำหรับการจัดการข้อมูล เป็นคำสั่งสำหรับการเรียกใช้ข้อมูล เพิ่ม ลบ หรือเปลี่ยนแปลงข้อมูล โดยมีคำสั่งในการจัดการข้อมูล เช่น Select, Insert, Update, Delete

3. ภาษาสำหรับการควบคุมข้อมูล (Data Control Language: DCL)

ภาษาสำหรับการควบคุมข้อมูล คือภาษาโปรแกรมที่ถูกเขียนขึ้น เพื่อใช้ในการกำหนดสิทธิ์การเข้าถึงฐานข้อมูลต่าง ๆ โดยสามารถให้การอนุญาต หรือยกเลิกการเข้าถึงฐานข้อมูลของผู้ใช้งานรายนั้น ๆ ได้ เพื่อป้องกันความปลอดภัยของฐานข้อมูล โดยมีคำสั่งในการควบคุมข้อมูล เช่น Grant, Revoke

4. ภาษาสำหรับควบคุมการทำธุรกรรม (Transaction Control Language: TCL)

ภาษาสำหรับควบคุมการทำธุรกรรม คือคำสั่งที่ถูกเขียนขึ้น เพื่อทำการควบคุมการทำธุรกรรมต่าง ๆ ไม่ว่าจะเป็นการเริ่มการทำธุรกรรม การยืนยันธุรกรรม รวมไปถึงการยกเลือกการทำธุรกรรม ช่วยให้สามารถเปลี่ยนแปลงข้อมูลในฐานข้อมูลได้แบบอัตโนมัติ เช่น Rollback [8]

2.2 เครื่องมือที่เกี่ยวข้อง

2.2.1 Spring Boot

Spring boot เป็น framework package ที่ต่อยอดมาจาก spring framework ช่วยให้เราสามารถสร้างแอ��เพล็กชัน (application) แบบ stand-alone แบบรวดเร็ว และง่ายดาย [9]

2.2.2 Maven

Apache Maven อาจเรียกสั้น ๆ ว่า Maven เป็นเครื่องมือตัวหนึ่งที่ช่วยให้นักพัฒนา (Developer) เขียน Java Application ได้ง่ายขึ้น เป็น Project Management Tools ที่อยู่อำนวยความสะดวกในการสร้าง Java Project โดยมีความสามารถที่

- ช่วยในเรื่องการจัดการ Dependencies ต่าง ๆ ให้กับ Java Project (ทำหน้าที่เป็น Dependency Management)
- ช่วยในเรื่องการ Compile Source Code ทั้งหมดภายในโปรเจ็ค จาก .java เป็น .class ให้โดยอัตโนมัติ
- ช่วยในเรื่องการ Run Code ทดสอบ เช่น การ Run Unit Test/Integration Test ต่าง ๆ ภายในโปรเจ็ค
- ช่วยในเรื่องการ Build Project ไปเป็นรูปแบบต่าง ๆ ตามที่ต้องการจะนำไปใช้งาน เช่น .jar .war .ear .bundle
- ช่วยในเรื่องการ Deploy Source Code แบบอัตโนมัติ
- ช่วยในเรื่องการแบ่งโปรเจ็คออกเป็นระบบโมดูล (การทำ Multiple Modules)
- มีโครงสร้างโปรเจ็คที่เป็นมาตรฐาน (Standard)
- ช่วยในเรื่องการทำprofile (Profile) คือ สามารถ Config ค่าต่าง ๆ สำหรับโปรเจ็ค เพื่อนำไปใช้งานในแต่ละ Environment ที่แตกต่างกันได้
 - และอื่น ๆ อีกมาก many ตามที่ระบบ Plugins ของ Maven มีให้ใช้
 - นักพัฒนา สามารถเขียน Maven Plugin ขึ้นมาใช้งานเองได้ [10]

2.2.3 Hibernate

Hibernate เป็น Framework Java ใช้ในการจัดการข้อมูลแบบ ORM (Object/Relation Mapping คือ การ mapping Java Object กับ ข้อมูลจากฐานข้อมูลแบบอัตโนมัติกลับไปกลับมาได้) เพื่อความสะดวกในการทำงาน ต่างๆ เช่น การเข้าถึงข้อมูล การเรียกค้นข้อมูล ซึ่งจะช่วยให้เราทำงานได้อย่างมีประสิทธิภาพ นอกจากนี้แล้ว Hibernate ยังเป็นโอเพนซอร์สอีกด้วย จึงสามารถดาวน์โหลดมาใช้งานกันได้ฟรีๆ ไม่มีค่าใช้จ่ายใดๆ ทั้งสิ้น

ประโยชน์ของ Hibernate คือ

1. มีเวลาในการ focus business logic ของโปรแกรมอย่างเต็มที่
2. ทำให้การบำรุงรักษาง่ายขึ้นเนื่องจากโคดที่เขียนน้อยลง สะอาด เข้าใจง่าย
3. ไม่ยืดติดกับฐานข้อมูล เราสามารถใช้ฐานข้อมูลได้หลากหลาย
4. มี APIs ที่เรียบง่ายที่ใช้ในการบันทึกและอ่านข้อมูลจากฐานข้อมูลได้โดยตรงผ่าน Java objects.
5. หากมีการเปลี่ยนแปลงโครงสร้างฐานข้อมูลหรือตารางต่างๆ สามารถแก้ไขได้ง่ายๆเพียงแค่แก้ที่ไฟล์ XML เท่านั้น
6. Hibernate ไม่ต้องเพิ่ม application server เพื่อใช้งานแต่อย่างใด
7. สามารถจัดการความสมดุลที่ซับซ้อนของตารางต่างๆ ได้อย่างง่ายดาย
8. จัดการการเข้าถึงฐานข้อมูลได้อย่างมีประสิทธิภาพ ทำให้ลดภาระของ Database Server ซึ่งส่งผลให้การเชื่อมต่อเร็วขึ้น [11]

2.2.4 MySQL

MySQL คือ ระบบจัดการฐานข้อมูล หรือ Database Management System (DBMS) แบบข้อมูลเชิงสัมพันธ์ หรือ Relational Database Management System (RDBMS) ซึ่งเป็นระบบฐานข้อมูลที่จัดเก็บรวมข้อมูลในรูปแบบตาราง โดยมีการแบ่งข้อมูลออกเป็นแถว (Row) และในแต่ละแถวแบ่งออกเป็นคอลัมน์ (Column) เพื่อเข้ามายังระหว่างข้อมูลในตารางกับข้อมูลในคอลัมน์ที่กำหนด แทนการเก็บข้อมูลที่แยกออกจากกัน โดยไม่มีความเชื่อมโยงกัน ซึ่งประกอบด้วยข้อมูล (Attribute) ที่มีความสัมพันธ์กัน (Relation) โดยใช้ RDBMS Tools สำหรับการควบคุมและจัดเก็บฐานข้อมูลที่จำเป็น ทำให้นำไปประยุกต์ใช้งานได้ง่าย ช่วยเพิ่มประสิทธิภาพในการทำงานให้มีความยืดหยุ่นและรวดเร็วได้มากยิ่งขึ้น รวมถึงเชื่อมโยงข้อมูล ที่จัดแบ่งกลุ่มข้อมูลแต่ละประเภทได้ตามต้องการ จึงทำให้ MySQL เป็นโปรแกรมระบบจัดการฐานข้อมูลที่ได้รับความนิยมสูง [12]

2.2.5 Thymeleaf

เป็น Template Engine โดยหลักการเขียน view ด้วย HTML แต่มีบาง Tag ช่วยให้จัดการกับข้อมูลได้ง่ายขึ้น [13]

2.2.6 Git

Git คือ Version Control ตัวหนึ่งซึ่งจะเป็นระบบที่มีหน้าที่ทำการจัดเก็บการเปลี่ยนแปลงของไฟล์ใน Project และมีการ backup ให้เราสามารถที่จะเรียกดูหรือทำการย้อนกลับไปคู่เวอร์ชันต่างๆของ Project ที่ได้ เวลาใดก็ได้ ดังนั้น Version Control ก็หมายความว่าสามารถติดตามการเปลี่ยนแปลงที่เกิดขึ้นในรูปแบบเดียวหรือกลุ่มก็ตาม และนอกจากนั้นก็ยังสามารถเรียกดูได้ว่าใครเป็นคนเขียนหรือใครเป็นคนแก้ไข Project ในส่วนต่างๆ [14]

2.2.7 Visual Studio Code/ IntelliJ IDEA

เป็น IDE (Integrated Development Environment) ที่นิยมใช้ในการพัฒนาซอฟต์แวร์ โดยมีเครื่องมือที่ช่วยในการเขียนโค้ด การจัดการโปรเจกต์ และการดีบัก (Debugging)

2.2.8 Postman

Postman คือเครื่องมือที่ใช้สำหรับพัฒนาและทดสอบ API (Application Programming Interfaces) เช่น การส่ง Parameter ไปใน header หรือ body เพื่อให้เราทดสอบว่าระบบสามารถตอบกลับมาได้ถูกต้องหรือไม่ ซึ่งตัว Postman สามารถสร้างและแบ่งปัน request และ collection API กับผู้อื่นได้อย่างสะดวก สามารถสร้าง API mock และตรวจสอบประสิทธิภาพได้

นอกจากนี้ Postman ยังมีเครื่องมือในการสร้างเอกสาร API ช่วยให้การทำงานร่วมกันภายในทีม ทำได้อย่างราบรื่น มีประสิทธิภาพมากขึ้น มี UI ที่ใช้งานง่าย สำหรับการส่ง request ดู response ที่ตอบกลับมา และยังสามารถ (debug) ปัญหา สร้าง Mocking API และยังไม่หมดเพียงเท่านี้ ตัว Postman เองยังสามารถบันทึกทำเป็นเอกสาร Document API เพื่อใช้แบ่งปันข้อมูลให้กับผู้อื่นได้ [15]

2.3 ภาษาที่ใช้เขียน

2.3.1 JAVA

Java เป็นภาษาการเขียนโปรแกรมที่ใช้กันอย่างแพร่หลายสำหรับการเขียนโค้ดเว็บแอปพลิเคชัน ซึ่งเป็นตัวเลือกที่ได้รับความนิยมในหมู่นักพัฒนามากกว่าสองศตวรรษ โดยมีการใช้งานแอปพลิเคชัน Java หลายล้านรายการในปัจจุบัน Java เป็นภาษาแบบหลายแพลตฟอร์ม เชิงวัตถุ และใช้เครื่อข่ายเป็นศูนย์กลางที่สามารถใช้เป็นแพลตฟอร์มได้ในตัวเอง ซึ่งเป็นภาษาการเขียนโปรแกรมที่รวดเร็ว ปลอดภัย และเชื่อถือได้สำหรับการเขียนโค้ดทุกประเภทตั้งแต่แอปมือถือและซอฟต์แวร์ระดับองค์กร ไปจนถึงแอปพลิเคชัน Big Data และเทคโนโลยีฝั่งเซิร์ฟเวอร์ [16]

2.3.2 HTML

HTML ย่อมาจาก Hyper Text Markup Language คือภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบน website หรือที่เราเรียกว่าเว็บเพจ ถูกพัฒนาและกำหนดมาตรฐานโดยองค์กร World Wide Web Consortium (W3C) และจากการพัฒนาทางด้าน Software ของ Microsoft ทำให้ภาษา HTML เป็นอีกภาษาหนึ่งที่ใช้เขียนโปรแกรมได้ หรือที่เรียกว่า HTML Application HTML เป็นภาษาประเภท Markup สำหรับการสร้างเว็บเพจ โดยใช้ภาษา HTML สามารถทำโดยใช้โปรแกรม Text Editor ต่างๆ เช่น Notepad, Editplus หรือจะอาศัยโปรแกรมที่เป็นเครื่องมือช่วยสร้างเว็บเพจ เช่น Microsoft FrontPage, Dream Weaver ซึ่งอำนวยความสะดวกในการสร้างหน้า HTML ส่วนการเรียกใช้งานหรือทดสอบการทำงานของเอกสาร HTML จะใช้โปรแกรม web browser เช่น IE Microsoft Internet Explorer (IE), Mozilla Firefox, Safari, Opera, และ Netscape Navigator เป็นต้น [17]

โครงสร้างของ HTML จะประกอบไปด้วยส่วนของคำสั่ง 2 ส่วน คือ ส่วนที่เป็น ส่วนหัว (Head) และส่วนที่เป็นเนื้อหา (Body) โดยมีรูปแบบคำสั่งดังนี้ [18]



ภาพที่ 5 ภาพโครงสร้างภาษา html

2.3.3 CSS

CSS คือ ภาษาที่ใช้สำหรับตกแต่งเอกสาร HTML/XHTML ให้มีหน้าตา สีสัน ระยะห่าง พื้นหลัง เส้นขอบ และอื่นๆ ตามที่ต้องการ CSS ย่อมาจาก Cascading Style Sheets มีลักษณะเป็นภาษาที่มีรูปแบบในการเขียน Syntax แบบเฉพาะและได้ถูกกำหนดมาตรฐานโดย W3C เป็นภาษาหนึ่งในการตกแต่งเว็บไซต์ ได้รับความนิยมอย่างแพร่หลาย ประযุชน์ของ CSS

CSS มีประโยชน์อย่างหลากหลาย ซึ่งได้แก่

- ช่วยให้เนื้อหาภายในเอกสาร HTML มีความเข้าใจได้ง่ายขึ้นและในการแก้ไขเอกสารก็สามารถทำได้ง่ายกว่าเดิม เพราะการใช้ CSS จะช่วยลดการใช้ภาษา HTML ลงได้ในระดับหนึ่ง และแยกระหว่างเนื้อหา กับรูปแบบในการแสดงผลได้อย่างชัดเจน

- ทำให้สามารถดาวน์โหลดไฟล์ได้เร็ว เนื่องจาก code ในเอกสาร HTML ลดลง จึงทำให้ไฟล์มีขนาดเล็กลง

- สามารถกำหนดรูปแบบการแสดงผลจากคำสั่ง style sheet ชุดเดียวกัน ให้มีการแสดงผลในเอกสารแบบเดียวกันทั้งหน้าหรือในทุกๆ หน้าได้ ช่วยลดเวลาในการปรับปรุงและทำให้การสร้างเอกสารบนเว็บมีความรวดเร็วขึ้น นอกจากนี้ยังสามารถควบคุมการแสดงผล ให้คล้ายหรือเหมือนกันได้ในหลาย Web Browser

- ช่วยในการกำหนดการแสดงผลในรูปแบบที่มีความเหมาะสมกับสื่อต่างๆ ได้เป็นอย่างดี

- ทำให้เว็บไซต์มีความเป็นมาตรฐานมากขึ้น และมีความทันสมัย สามารถรองรับการใช้งานในอนาคตได้ดี

[19]

2.3.4 JavaScript

ภาษา JavaScript คือภาษาสคริปต์ทางคอมพิวเตอร์ที่ได้รับความนิยมสูงสุดในหมู่นักพัฒนาโปรแกรมทั่วโลก ด้วยคุณสมบัติที่สามารถนำมาสร้างได้ทั้ง Website และ Web Application รวมถึงสามารถนำไปใช้ประโยชน์ได้ทั้งในฝั่ง Client และ Server ผลิตภัณฑ์ที่ทำให้ JavaScript เป็นภาษาที่ได้รับความนิยมในหมู่นักพัฒนาซอฟต์แวร์ สามารถสรุปได้ ดังนี้

เขียนโปรแกรมได้ครบวงจรทั้งฝั่งหน้าบ้านและฝั่งหลังบ้าน

Node.js ทำให้การพัฒนาเว็บไซต์และแอปฯ ของฝั่ง Back-end ด้วย JavaScript กลายเป็นเรื่องง่าย ทุกวันนี้ เราสามารถเขียนโปรแกรมด้วยภาษา Java Script แล้วนำไปรันบนระบบปฏิบัติการที่สนับสนุนโดย Node.js ได้เลย ทันที ภาษา JavaScript จึงกลายเป็นภาษาเดียวที่หยอดมาใช้งานได้อย่างครอบคลุมทั้ง Client Side และ Server Side
ไลบรารีและเฟรมเวิร์ค

แหล่งเฟรมเวิร์ค (Pre-Written Code) และไลบรารีที่มีอยู่มากมายสามารถนำไปใช้กับโค้ดของプロジェクトที่เหล่านักพัฒนาต้องการสร้างได้ตามสะดวก สิ่งนี้ทำให้การสร้างเว็บและแอปฯ ทำได้อย่างรวดเร็วขึ้นและช่วยลดโอกาสการเกิดความผิดพลาดในการเขียนโค้ดจากภาษา JavaScript ได้

ติดตั้งง่ายและมีการใช้งานที่แพร่หลายมากกว่าภาษาสคริปต์อื่น

ทุกวันนี้ไม่ว่าจะบราузர์ไหนก็รองรับการทำงานของภาษา JavaScript หันนั้น หากเป็นภาษาอื่นๆ เราอาจจะต้องดาวน์โหลดภาษาสคริปต์เหล่านั้นลงในเครื่องเพื่อให้สามารถเข้าถึงซอฟต์แวร์พัฒนาเว็บได้ แต่ JavaScript ไม่จำเป็น ข้อดีตรงนี้ช่วยลดความยุ่งยากในขั้นตอนการโค้ดไปได้มาก

JavaScript ถูกนำไปใช้งานหลัก ๆ ดังนี้

- **ใช้เพิ่มความ Interactive (มีปฏิสัมพันธ์กับผู้ใช้งาน) ให้กับหน้าเว็บไซต์**

JavaScript สามารถใช้ในการสร้างปฏิสัมพันธ์กับผู้ใช้งานในลักษณะต่างๆ เพื่อเพิ่มลูกเล่นในการใช้งานได้ เช่น ใช้เพิ่มการเคลื่อนไหวให้กับ Element มากมายบนหน้าใช้งาน การเล่นเสียงหรือวิดีโอ หรือแม้แต่การตั้งค่าให้ปุ่มต่างๆ เปลี่ยนรูปแบบเมื่อเลื่อนเมาส์ไปโดน

- **พัฒนาเว็บไซต์หรือแอปพลิเคชันบนมือถือ**

ทุกวันนี้นักพัฒนาโปรแกรมสามารถเลือกใช้เฟรมเวิร์ค Pre-Written Code บน JavaScript Code Libraries ในการสร้างเว็บไซต์หรือแอปพลิเคชันได้เลย ไม่ต้องนั่งโค้ดโครงสร้างใหม่เองแบบนับจากศูนย์ สิ่งนี้ที่มาพร้อมกับการเขียน JavaScript เพิ่มความสะดวกให้กับนักพัฒนาได้มากที่เดียว ตัวอย่างของ Front-end Framework ที่คนนิยมใช้ในการเขียนเว็บและโมบายแอปกัน ได้แก่ React, React Native, Angular และ Vue

- **สร้างและพัฒนาเว็บไซต์เซิร์ฟเวอร์ รวมถึงแอปเซิร์ฟเวอร์**

JavaScript คือภาษาที่สามารถใช้ทำงานได้ทั้งฝั่งหน้าบ้าน (Front End) และหลังบ้าน (Back End) ดังนั้น แล้วนอกเหนือจากเว็บไซต์และแอปฯ นักพัฒนาสามารถใช้ JavaScript เพื่อสร้างเซิร์ฟเวอร์อย่างง่ายและพัฒนาโครงสร้างพื้นฐานเบื้องตนโดยใช้ Node.js ได้ด้วย

- **พัฒนาเกม**

รู้ไหมว่าเกมดังอย่าง TowerBuilding และ Polycraft เองก็ใช้ JavaScript ใน การเขียนขึ้นมาเพื่อสนับสนุนกัน เนื่องจาก JavaScript สามารถสร้างอนิเมชันรวมถึงการตอบสนองกับผู้ใช้บนหน้าเว็บได้ ดังนั้น นักพัฒนาเกมหลายคนจึงเลือกใช้ JavaScript ในการสร้างเกมของพวกรเข้า นอกจากนี้ เพราะเป็นภาษาที่มีพัฒนาชั้นค่าสั่งครบถ้วนและง่ายกว่าภาษาอื่น การสร้างเกมจาก JavaScript จึงเหมาะสมเป็นอีกทางเลือกสำหรับการเรียนรู้วิธีเขียนโค้ดแบบเบื้องต้นสำหรับเด็กและผู้ใหญ่บางคนที่พื้นฐานยังไม่แข็งมาก

บทที่ 3

การออกแบบและการใช้งาน

3.1 การออกแบบการพัฒนาระบบ

3.1.1 วิเคราะห์ความต้องการของผู้ใช้งาน

ระบบขายเพอร์นิเจอร์ต้องตอบสนองต่อความต้องการของผู้ใช้งานหลัก 2 กลุ่มคือ

(1) พนักงานขาย (Cashier)

- การจัดการคำสั่งซื้อ พนักงานขายต้องการระบบที่สามารถสร้างคำสั่งซื้อได้อย่างรวดเร็วและมีความสะดวก รวมถึงการคำนวณยอดรวมและการจัดการส่วนลด
- การชำระเงิน ระบบต้องรองรับหลายช่องทางการชำระเงิน เช่น เงินสด, บัตรเครดิต, และระบบชำระเงินออนไลน์ ซึ่งต้องมีความสะดวกในการใช้งาน
- การคืนทรัพย์สินค้า ต้องการฟังก์ชันการคืนทรัพย์สินค้าได้อย่างรวดเร็ว โดยสามารถกรอกตามหมวดหมู่, ราคา, หรือชื่อสินค้า
- การจัดการสต็อกสินค้า ต้องการดูข้อมูลสต็อกสินค้าปัจจุบัน เพื่อให้สามารถแนะนำลูกค้าได้อย่างถูกต้อง

(2) ผู้ดูแลระบบ (Administrator)

- การจัดการผู้ใช้ ต้องการฟังก์ชันในการเพิ่ม แก้ไข และลบผู้ใช้ในระบบ รวมถึงการทำหน้าที่ต่างๆ เช่น จัดการสิทธิ์การเข้าถึงของแต่ละผู้ใช้
- การจัดการข้อมูลสินค้า ต้องการระบบที่สามารถเพิ่ม แก้ไข และลบข้อมูลสินค้าได้ รวมถึงการจัดการหมวดหมู่สินค้า
- การติดตามการขาย ต้องการดูข้อมูลการขายทั้งหมดและรายงานเพื่อวิเคราะห์แนวโน้มการขาย และการตอบสนองต่อความต้องการของตลาด
- การจัดการการคืนสินค้า ต้องการฟังก์ชันในการจัดการการคืนสินค้าจากลูกค้าอย่างมีประสิทธิภาพ

- การจัดการสต็อก ต้องการตรวจสอบและจัดการระดับสต็อกสินค้าเพื่อป้องกันการขาดแคลนหรือการเก็บสินค้ามากเกินไป

3.1.2 การนำหลัก SOLID ใน Spring ส่วนใหญ่มีประยุกต์ระบบ

ในการพัฒนาระบบด้วย Spring Framework ได้มีการนำหลักการ SOLID มาใช้เพื่อทำให้โค้ดมีความยึดหยุ่นและดูแลรักษาง่าย ดังนี้

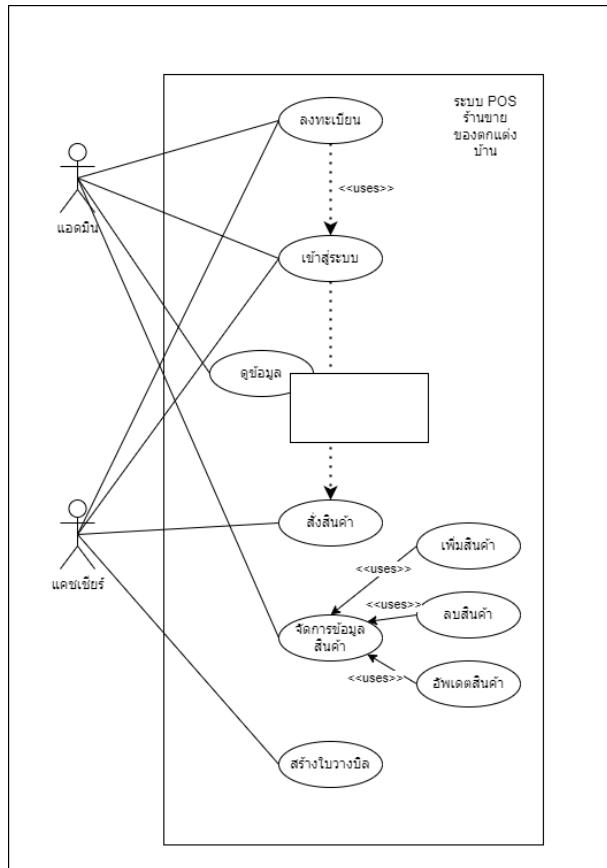
- **Single Responsibility Principle (SRP)** แยกคลาสต่าง ๆ เช่น ProductService OrderService และ CategoryService เพื่อให้แต่ละคลาสทำหน้าที่เฉพาะเจาะจง

- Open/Closed Principle (OCP) มีการแบ่งสินค้าออกเป็นหลายประเภท เช่น เก้าอี้ โต๊ะ โซฟา เราสามารถถอดออกแบบระบบให้รองรับการเพิ่มประเภทสินค้าใหม่ ๆ ได้โดยไม่ต้องแก้ไขคลาสสินค้าเดิม
- Liskov Substitution Principle (LSP) ระบบมีการใช้ interface เพื่อให้สามารถแทนที่กันได้ตามหลักของ Liskov Substitution Principle
- Interface Segregation Principle (ISP) แยก interface สำหรับบริการต่าง ๆ เช่น การจัดการสินค้า และการจัดการคำสั่งซื้อ เพื่อให้ client ใช้เฉพาะ interface ที่จำเป็น
- Dependency Inversion Principle (DIP) ใช้การ Inject Dependencies ผ่าน @Autowired

3.2 Unified Modeling Language (UML)

3.2.1 Use Case Diagram

Use Case Diagram แสดงกรณีการใช้งานของระบบ เช่น การสั่งซื้อสินค้า การเพิ่มสินค้า การจัดการคำสั่งซื้อ โดยแบ่งออกเป็นกรณีการใช้งานสำหรับผู้ใช้งานทั่วไปและผู้ดูแลระบบ



ภาพที่ 6 ภาพ Use Case Diagram

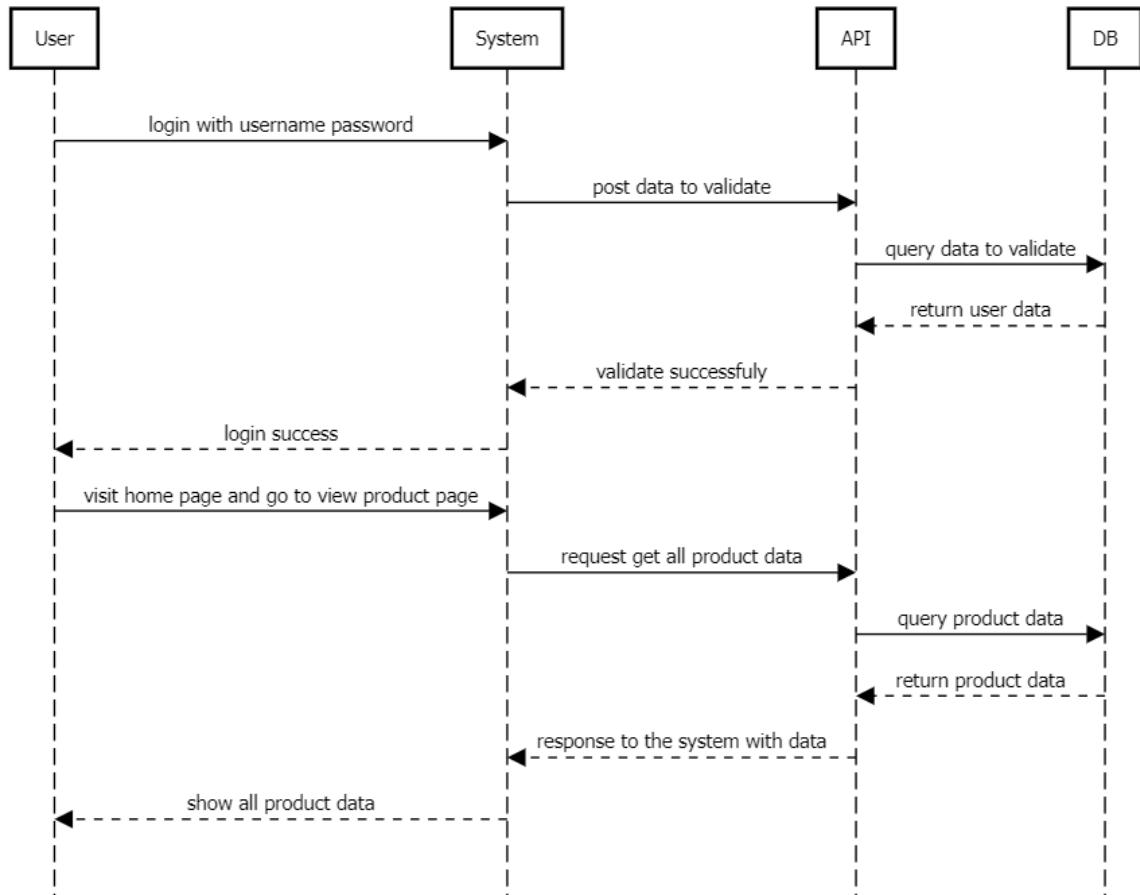
3.2.2 Use Case Text

ตารางที่ 2 ตาราง Use Case Text

| Users | Use Case | Descriptions |
|--------------------|-----------|--|
| ลงทะเบียน | แอดมิน | แอดมินทำการลงทะเบียนในระบบให้กับแคชเชียร์ |
| เข้าสู่ระบบ | แอดมิน | แคชเชียร์เข้าสู่ระบบเพื่อเข้าหน้าสั่งสินค้า/แคชเชียร์เข้าสู่ระบบเพื่อเข้าสู่ส่วนจัดการสินค้า |
| สั่งสินค้า | แคชเชียร์ | แคชเชียร์สั่งสินค้าจากระบบ |
| จัดการข้อมูลสินค้า | แอดมิน | แอดมินสามารถจัดการสินค้าในฐานข้อมูล |
| เพิ่มสินค้า | แอดมิน | แอดมินเพิ่มสินค้าลงฐานข้อมูล |
| ลบสินค้า | แอดมิน | แอดมินลบสินค้าออกจากฐานข้อมูล |
| อัพเดตสินค้า | แอดมิน | แอดมินอัพเดตสินค้าลงฐานข้อมูล |

3.2.3 3 Sequence Diagram

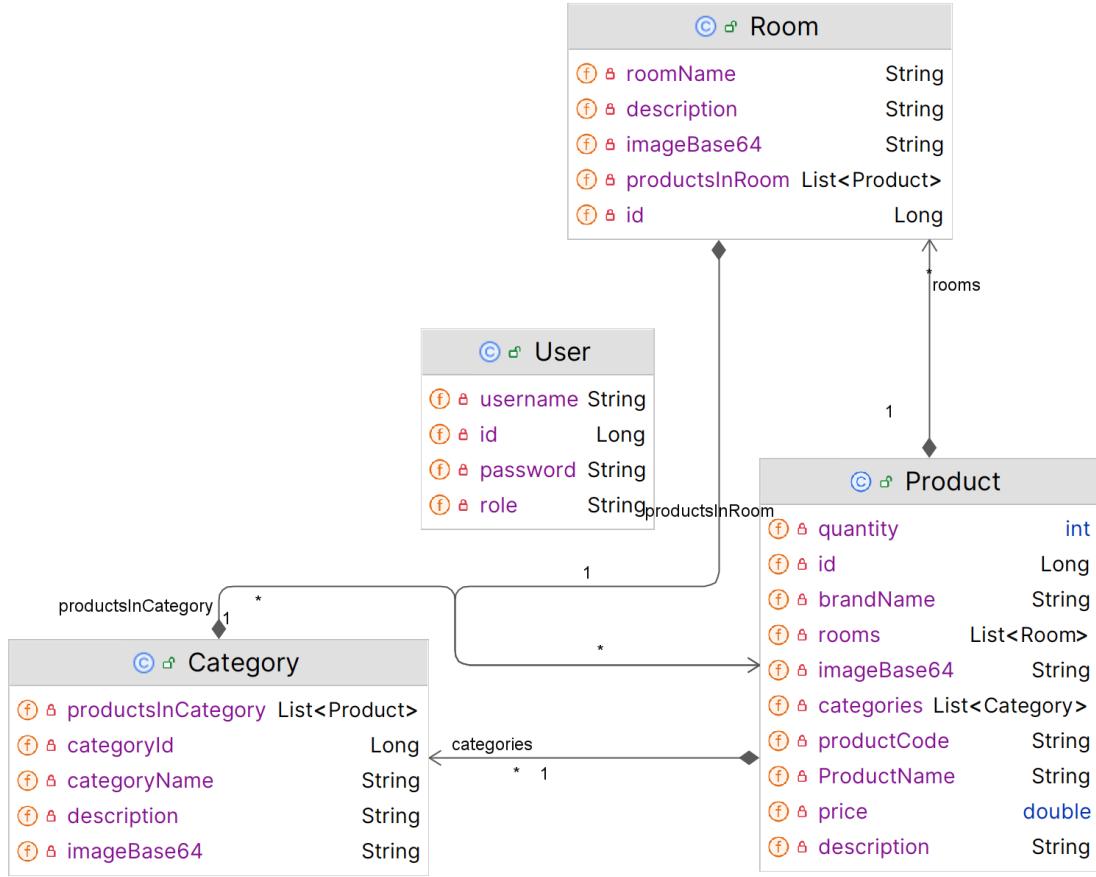
User log in and view all product information pages



ภาพที่ 7 ภาพ Sequence Diagram

3.2.4 Class Diagram

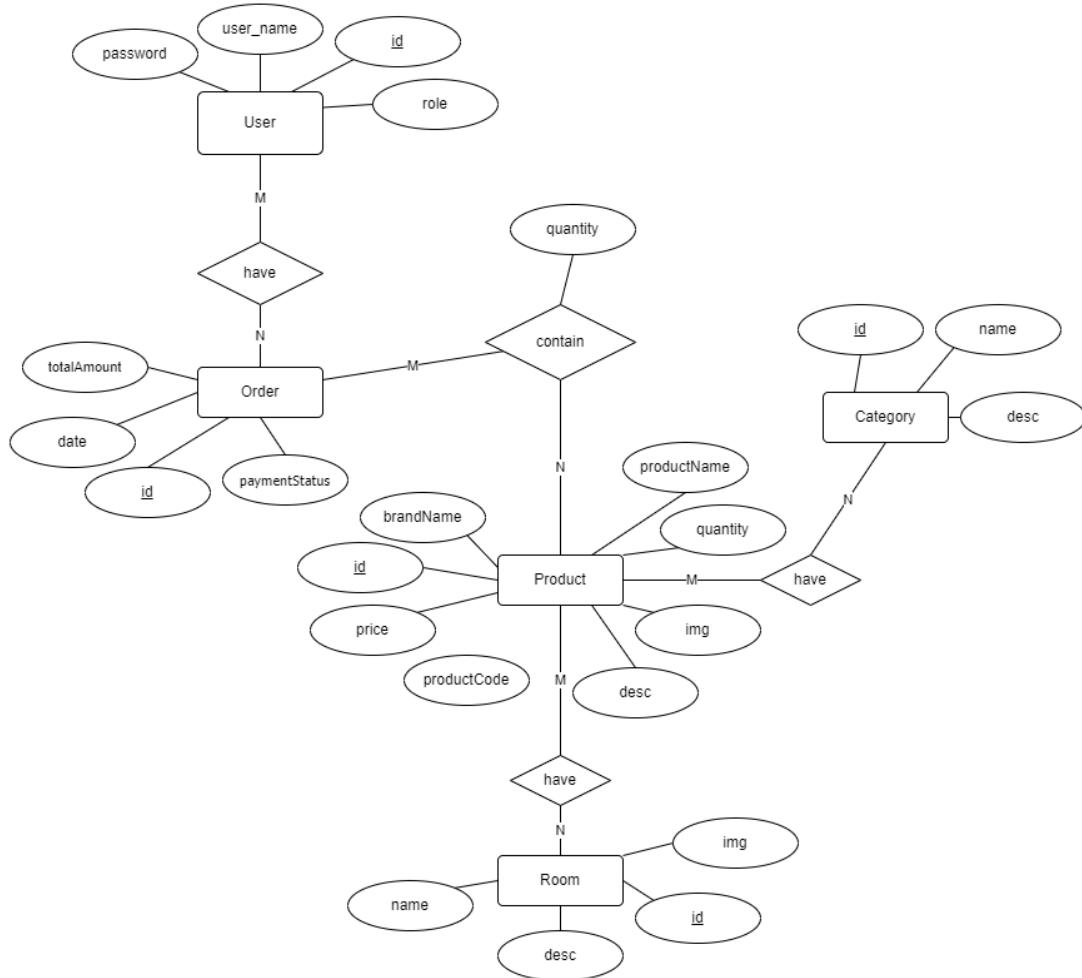
แสดงโครงสร้างของคลาสหลักในระบบ เช่น Product, User, Order, และความสัมพันธ์ระหว่างคลาส รวมถึงข้อมูลที่แต่ละคลาสจัดเก็บ



ภาพที่ 8 ภาพ Class Diagram

3.3 การออกแบบฐานข้อมูลของระบบ

3.3.1 ER Diagram



ภาพที่ 9 ภาพ ER Diagram

Class Diagram

ER Diagram แสดงความสัมพันธ์ระหว่างตารางในฐานข้อมูล เช่น ตาราง Product, Category, Order, และ User โดยมีความสัมพันธ์แบบ one-to-many ระหว่างตาราง Category กับ Product และ one-to-many ระหว่าง Order กับ Product

บทที่ 4

ผลการพัฒนาระบบ

4.1 ผลการพัฒนาโปรแกรม

ผลการพัฒนาโปรแกรมสามารถแบ่งออกได้ดังนี้

- 4.1.1. การออกแบบโครงสร้างฐานข้อมูลและการสร้าง Entity Relationship Diagram (ERD) เพื่อแสดงความสัมพันธ์ของข้อมูล

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** MyDB
- Tables:** A list of tables including category, customer_order, order_item, Orderitem, product, product_category, product_room, room, users.
- Query Editor:** A SQL query window containing: `1 • SELECT * FROM houselyDB.product;`
- Result Grid:** A table displaying the results of the query. The columns are: id, brand_name, description, image_base64, price, product_code, product_name, quantity, image, imageBase64. The data includes three rows of furniture products with their respective details.
- Action Output:** A log of the last query execution: `1 11:58:20 SELECT * FROM houselyDB.product LIMIT 0, 1000`. It shows the response was 3 rows returned in 0.0012 sec / 0.0006...
- Object Info:** A table showing the schema definition for the 'product' table, including columns: id (bigint AI PK), brand_name (varchar(255)), description (varchar(1000)), image_base64 (tinytext), price (double), product_code (varchar(255)), product_name (varchar(255)), quantity (int), image (tinytext), imageBase64 (tinytext).

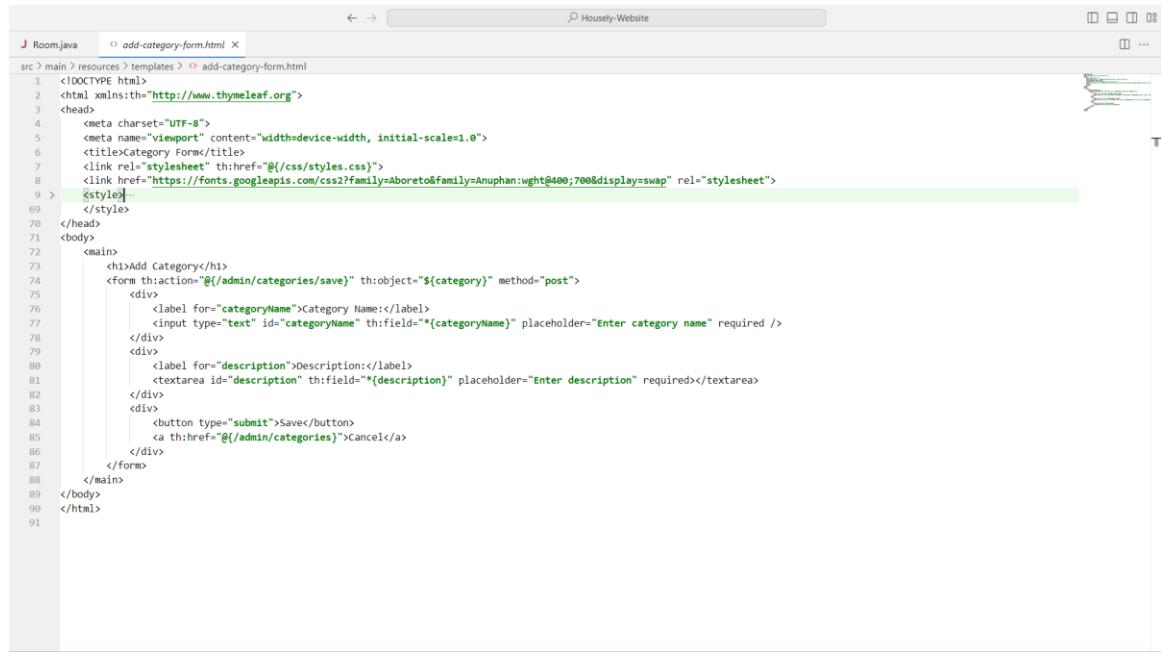
ภาพที่ 10 ภาพฐานข้อมูล

- 4.1.2. การพัฒนา Front-end โดยใช้ภาษา HTML, CSS และ JavaScript เพื่อสร้างส่วนติดต่อผู้ใช้งาน (UI) Resource

- add-category-form.html

ฟอร์มนี้ทำหน้าที่ให้ผู้ใช้กรอกข้อมูลชื่อหมวดหมู่และคำอธิบาย

แล้วส่งข้อมูลไปบันทึกลงในฐานข้อมูลผ่านเชิร์ฟเวอร์ โดยใช้ Thymeleaf สำหรับการผูกข้อมูลจากโมเดล Category กับฟิลด์ในฟอร์ม



```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Category Form</title>
7   <link rel="stylesheet" th:href="@{/css/styles.css}">
8   <link href="https://fonts.googleapis.com/css?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9 > <style>
10 </style>
11 </head>
12 <body>
13   <main>
14     <h3>Add Category</h3>
15     <form th:action="@{/admin/categories/save}" th:object="${category}" method="post">
16       <div>
17         <label for="categoryName">Category Name:</label>
18         <input type="text" id="categoryName" th:field="#{categoryName}" placeholder="Enter category name" required />
19       </div>
20       <div>
21         <label for="description">Description:</label>
22         <textarea id="description" th:field="#{description}" placeholder="Enter description" required></textarea>
23       </div>
24       <div>
25         <button type="submit">Save</button>
26         <a th:href="@{/admin/categories}">Cancel</a>
27       </div>
28     </form>
29   </main>
30 </body>
31 </html>
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

```

ภาพที่ 11 ภาพ add-category-form.html

- add-product-form.html

ฟอร์มนี้ออกแบบมาเพื่อเพิ่มข้อมูลผลิตภัณฑ์ โดยมีการเลือกหมวดหมู่และห้องคล้ายรายการ การอัปโหลดรูปภาพผลิตภัณฑ์ และปุ่มสำหรับบันทึกหรือยกเลิกการทำรายการ

ฟอร์ม: มีฟิลเตอร์สำหรับรหัสผลิตภัณฑ์, ชื่อแบรนด์, ชื่อผลิตภัณฑ์, ราคา, จำนวน, คำอธิบาย และอัปโหลดรูปภาพเลือกหมวดหมู่และห้อง: ใช้ Select2 สำหรับเลือกหมวดหมู่และห้องแบบคล้ายตัวเลือก

JavaScript: ใช้ในการแสดงชื่อไฟล์ที่เลือกและจัดการค่าภายในฟิลด์ input

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Product Form</title>
7   <link rel="stylesheet" th:href="@{/css/styles.css}">
8   <link href="https://cdn.jsdelivr.net/npm/select2@4.0.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
9   <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Amuphan:wght@400;700&display=swap" rel="stylesheet">
10 > <style>...
11 </style>
12 </head>
13 <body>
14   <main>
15     <h1>Add Product</h1>
16     <form th:action="@{/admin/products/save}" th:object="${product}" method="post" enctype="multipart/form-data">
17       <input type="hidden" th:field="*{id}">
18       <div>
19         <label for="productCode">Product Code:</label>
20         <input type="text" id="productCode" th:field="*(productCode)" placeholder="Enter product code" required />
21       </div>
22       <div>
23         <label for="brandName">Brand Name:</label>
24         <input type="text" id="brandName" th:field="*(brandName)" placeholder="Enter brand name" required />
25       </div>
26       <div>
27         <label for="productName">Product Name:</label>
28         <input type="text" id="productName" th:field="*(productName)" placeholder="Enter product name" required />
29       </div>
30       <div>
31         <label for="price">Price:</label>
32         <input type="number" id="price" th:field="*(price)" placeholder="Enter price" required
33           onfocus="clearValue(this)" onblur="resetValue(this, 0)" />
34       </div>
35       <div>
36         <label for="quantity">Quantity:</label>
37         <input type="number" id="quantity" th:field="*(quantity)" placeholder="Enter quantity" required
38           onfocus="clearValue(this)" onblur="resetValue(this, 0)" />
39       </div>
40       <div>
41         <label for="description">Description:</label>
42         <textarea id="description" th:field="*(description)" placeholder="Enter description" required></textarea>
43       </div>
44       <!-- Categories with Select2 -->
45       <div class="form-section">
46         <label>Categories:</label>
47         <select id="categorySelect" multiple name="categoryIds" required>
48           <option th:each="category : ${categories}"
49             th:value="${category.categoryId}"
50             th:text="${category.categoryName}"></option>
51         </select>
52       </div>
53       <!-- Rooms with Select2 -->
54       <div class="form-section">
55         <label>Rooms:</label>
56         <select id="roomSelect" multiple name="roomIds" required>
57           <option th:each="room : ${rooms}"
58             th:value="${room.id}"
59             th:text="${room.roomName}"></option>
59         </select>
60       </div>
61       <div>
62         <label for="image">Image:</label>
63         <label class="custom-file-upload" for="image">
64           choose file
65         </label>
66         <input type="file" id="image" name="image" accept="image/*" required onchange="displayFileName()"/>
67         <p id="fileName" style="margin-top: 5px; color: #333;"></p>
68       </div>
69       <div>
70         <button type="submit">Submit</button>
71         <a th:href="@{/admin/products}">Cancel</a>
72       </div>
73     </form>
74   </main>
75 </body>
76 </html>

```

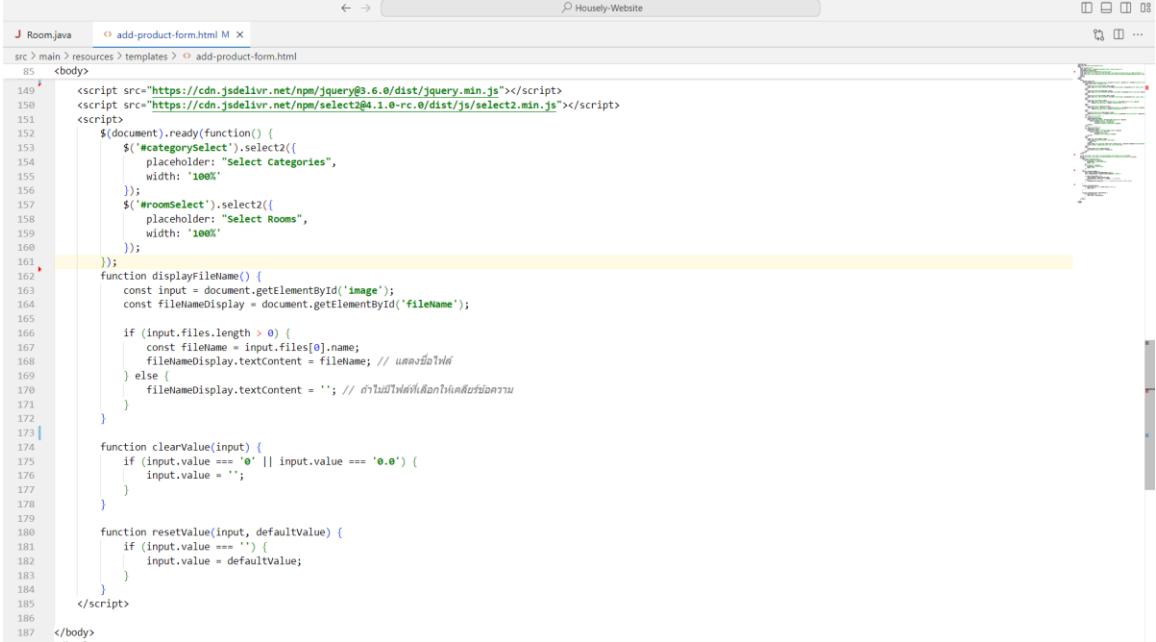
ภาพที่ 12 ภาพ add-product-form.html

```

85 <body>
86   <main>
87     <form th:action="@{/admin/products/save}" th:object="${product}" method="post" enctype="multipart/form-data">
88       <div>
89         <label for="description">Description:</label>
90         <textarea id="description" th:field="*(description)" placeholder="Enter description" required></textarea>
91       </div>
92       <!-- Categories with Select2 -->
93       <div class="form-section">
94         <label>Categories:</label>
95         <select id="categorySelect" multiple name="categoryIds" required>
96           <option th:each="category : ${categories}"
97             th:value="${category.categoryId}"
98             th:text="${category.categoryName}"></option>
99         </select>
100       </div>
101       <!-- Rooms with Select2 -->
102       <div class="form-section">
103         <label>Rooms:</label>
104         <select id="roomSelect" multiple name="roomIds" required>
105           <option th:each="room : ${rooms}"
106             th:value="${room.id}"
107             th:text="${room.roomName}"></option>
108         </select>
109       </div>
110       <div>
111         <label for="image">Image:</label>
112         <label class="custom-file-upload" for="image">
113           choose file
114         </label>
115         <input type="file" id="image" name="image" accept="image/*" required onchange="displayFileName()"/>
116         <p id="fileName" style="margin-top: 5px; color: #333;"></p>
117       </div>
118       <div>
119         <button type="submit">Submit</button>
120         <a th:href="@{/admin/products}">Cancel</a>
121       </div>
122     </form>
123   </main>
124 </body>
125 </html>

```

ภาพที่ 12 ภาพ add-product-form.html (ต่อ)



```

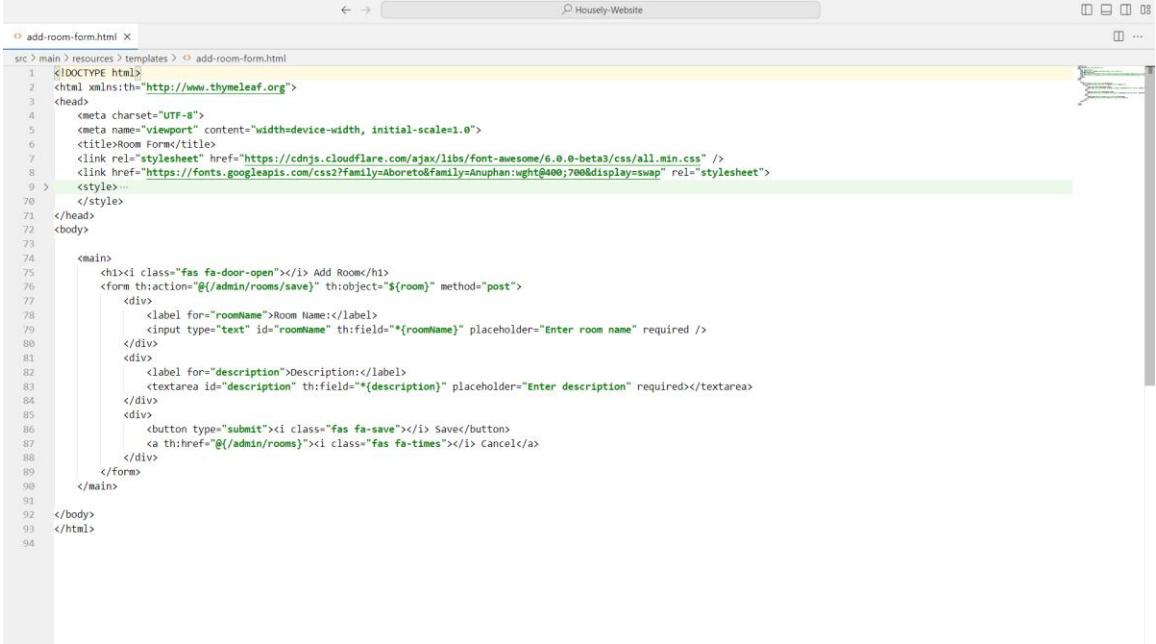
Room.java
src > main > resources > templates > add-product-form.html
85   <body>
149   <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.min.js"></script>
150   <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
151   <script>
152     $(document).ready(function() {
153       $('#categorySelect').select2({
154         placeholder: "Select Categories",
155         width: '100%'
156       });
157       $('#roomSelect').select2({
158         placeholder: "Select Rooms",
159         width: '100%'
160       });
161     });
162     function displayName() {
163       const input = document.getElementById('image');
164       const fileNameDisplay = document.getElementById('fileName');
165
166       if (input.files.length > 0) {
167         const fileName = input.files[0].name;
168         fileNameDisplay.textContent = fileName; // นามสกุลไฟล์
169       } else {
170         fileNameDisplay.textContent = ''; // กรณีไม่มีไฟล์ที่เลือกไว้จะแสดงว่าง
171       }
172     }
173
174     function clearValue(input) {
175       if (input.value === '0' || input.value === '0.0') {
176         input.value = '';
177       }
178     }
179
180     function resetValue(input, defaultValue) {
181       if (input.value === '') {
182         input.value = defaultValue;
183       }
184     }
185   </script>
186
187 </body>

```

ภาพที่ 12 ภาพ add-product-form.html (ต่อ)

- add-room-form.html

ฟอร์มนี้ถูกออกแบบมาให้เรียบง่ายและใช้งานง่ายสำหรับผู้ใช้ เพื่อกรอกข้อมูลห้องใหม่ โดยมีการใช้ไอคอนเพื่อเพิ่มความสวยงามให้กับปุ่มและหัวข้อ



```

src > main > resources > templates > add-room-form.html
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Room Form</title>
7      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.0.0-beta3/css/all.min.css" />
8      <link href="https://fonts.googleapis.com/css?family=Aboreto&family=Amuphan:wght@400;700&display=swap" rel="stylesheet">
9  >
10     </style>...
11 </head>
12 <body>
13
14     <main>
15         <i class="fas fa-door-open"></i> Add Room</h1>
16         <form th:action="@{/admin/rooms/save}" th:object="${room}" method="post">
17             <div>
18                 <label for="roomName">Room Name:</label>
19                 <input type="text" id="roomName" th:field="*{roomName}" placeholder="Enter room name" required />
20             </div>
21             <div>
22                 <label for="description">Description:</label>
23                 <textarea id="description" th:field="*{description}" placeholder="Enter description" required></textarea>
24             </div>
25             <div>
26                 <button type="submit"><i class="fas fa-save"></i> Save</button>
27                 <a th:href="@{/admin/rooms}"><i class="fas fa-times"></i> Cancel</a>
28             </div>
29         </form>
30     </main>
31
32 </body>
33 </html>
34

```

ภาพที่ 13 ภาพ add-room-form.html

- category-list.html

เมื่อผู้ใช้เข้าไปที่หน้าเว็บ จะสามารถดูรายการห้องวัดหมู่ทั้งหมดที่มีในฐานข้อมูล
 ผู้ใช้สามารถค้นหาห้องวัดหมู่ที่ต้องการได้โดยใช้คำค้นและประเภทการค้นหา
 ผู้ใช้สามารถเพิ่ม แก้ไข หรือ ลบห้องวัดหมู่ได้จากตัวเลือกที่มีในตาราง
 หลังจากดำเนินการเสร็จสิ้น ผู้ใช้สามารถออกจากระบบได้โดยคลิกที่ปุ่ม Logout

```

src > main > resources > templates > category-list.html
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org" lang="th">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>ຮ່ວມມືນາງົມ</title>
7      <script src="https://cdn.tailwindcss.com"></script>
8      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />
9      <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@100..700&display=swap" />
10     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/tailwindcss@2.0.3/dist/tailwind.min.css" />
11  </head>
12  <body>
13      <div class="container mx-auto p-4">
14          <div class="header-container">
15              <h1 class="text-3xl font-bold mb-8" style="margin: 20px 0px 30px 0px;">ຮ່ວມມືນາງົມ

```

ກາພທີ 14 ກາພ category-list.html

```

src > main > resources > templates > category-list.html
222 <body>
223     <div class="container mx-auto p-4">
224         <div class="mb-4">
225             <div class="flex space-x-4 items-center">
226                 <div class="search-input">
227                     <label for="searchInput" class="search-inp">ຄົນທ້າວ</label>
228                     <input type="text" id="searchInput" class="search-text" placeholder="ຫຼັບເຄີຍທ້າວ..." />
229                 </div>
230
231                 <div class="search-button-container">
232                     <button id="searchButton" class="search-button">
233                         <i class="fas fa-search h-5 w-5 mr-2"></i> ຕ່າງ
234                     </button>
235                 </div>
236
237                 <div class="search-button-container">
238                     <button id="clearButton" class="clear-button">
239                         <i class="fas fa-times h-5 w-5 mr-2"></i> ດ້ວຍເກີດ
240                     </button>
241                 </div>
242             </div>
243         </div>
244
245         <table id="categoryTable">
246             <thead>
247                 <tr>
248                     <th>ID</th>
249                     <th>ລັບມືນາງົມ</th>
250                     <th>ອາກສິນ</th>
251                     <th>ວິຊາຂອງ</th>
252                 </tr>
253             </thead>
254             <tbody>
255                 <tr th:each="category : ${categories}" class="table-row">
256                     <td th:text="${category.categoryId}"></td>
257                     <td th:text="${category.categoryName}"></td>
258                     <td th:text="${category.description}"></td>
259                     <td class="action-buttons">
260                         <a th:href="@{/admin/categories/edit/{id}(id:${category.categoryId})}" class="edit-button">Edit</a>
261                         <a th:href="@{/admin/categories/delete/{id}(id:${category.categoryId})}" class="delete-button" onclick="return confirmDelete()">Delete</a>
262                     </td>
263                 </tr>
264             </tbody>
265         </table>
266     </div>
267 
```

ກາພທີ 14 ກາພ category-list.html (ຕອ)

The screenshot shows the browser's developer tools with the "Sources" tab selected. The left sidebar lists the file structure: src > main > resources > templates > category-list.html. The main pane displays the code for category-list.html. The code includes a body section with a container and a table (#categoryTable). A tbody section contains rows for each category. An error message is displayed if there is a message or error. Below the table, a script block handles a search button click. It retrieves the search type and input values, then loops through the table rows to change the display style of each cell based on whether its value matches the search input. A comment indicates this is for finding categories by name. A function getRowIndex is defined to return the index of a cell based on its type. The right side of the interface shows the DOM tree and a detailed view of the current element.

```
src > main > resources > templates > category-list.html
222 <body>
223     <div class="container mx-auto p-4">
224         <table id="categoryTable">
225             <tbody>
226                 <tr th:each="category : ${categories}" class="table-row">
227                     <td>
228                         </td>
229                     </tr>
230                 </tbody>
231             </table>
232             <div id="message" th:if="${message}" th:text="${message}" style="color: green;"></div>
233             <div id="error" th:if="${error}" th:text="${error}" style="color: red;"></div>
234         </div>
235     </body>
236     <script>
237         document.getElementById("searchButton").addEventListener("click", function () {
238             var searchType = document.getElementById("searchType").value;
239             var searchInput = document.getElementById("searchInput").value.toLowerCase();
240             var tableRows = document.querySelectorAll("#categoryTable .table-row");
241
242             tableRows.forEach(function (row) {
243                 var cellIndex = getColumnIndex(searchType);
244                 var cell = row.querySelector("td:nth-child(" + cellIndex + ")");
245                 if (cell) {
246                     var cellValue = cell.textContent.toLowerCase();
247                     if (cellValue.includes(searchInput)) {
248                         row.style.display = "";
249                     } else {
250                         row.style.display = "none";
251                     }
252                 }
253             });
254         });
255
256         // ฟังก์ชันที่ใช้ในการคืนค่าอินดีกัมเมอร์ของตารางตามหัวข้อ
257         function getColumnIndex(type) {
258             switch (type) {
259                 case " categoryName":
260                     return 2;
261                 default:
262                     return 1;
263             }
264         }
265     </script>

```

ภาพที่ 14 ภาพ category-list.html (ต่อ)

The screenshot shows a browser window with the title "Housely-Website". The address bar contains the URL. The left sidebar shows the file structure: "src > main > resources > templates > category-list.html". The main area displays the HTML and JavaScript code for "category-list.html". The code includes a table for categories, message and error divs, and a search functionality. A yellow highlight covers the entire code block.

```
src > main > resources > templates > category-list.html
222 <body>
223   <div class="container mx-auto p-4">
224     <table id="categoryTable">
225       <tbody>
226         <tr th:each="category : ${categories}" class="table-row">
227           <td>${category.name}</td>
228         </tr>
229       </tbody>
230     </table>
231     <div id="message" th:if="${message}" th:text="${message}" style="color: green;"></div>
232     <div id="error" th:if="${error}" th:text="${error}" style="color: red;"></div>
233   </div>
234 </body>
235 <script>
236   document.getElementById("searchButton").addEventListener("click", function () {
237     var searchType = document.getElementById("searchType").value;
238     var searchInput = document.getElementById("searchInput").value.toLowerCase();
239     var tableRows = document.querySelectorAll("#categoryTable .table-row");
240
241     tableRows.forEach(function (row) {
242       var columnIndex = getColumnIndex(searchType);
243       var cell = row.querySelector(`td:nth-child(${columnIndex + 1})`);
244       if (cell) {
245         var cellValue = cell.textContent.toLowerCase();
246         if (cellValue.includes(searchInput)) {
247           row.style.display = "";
248         } else {
249           row.style.display = "none";
250         }
251       }
252     });
253   });
254
255   // ท่านอ่านที่อยู่ในการตั้งค่า columnIndex ที่ต้องการค้นหา
256   function getColumnIndex(type) {
257     switch (type) {
258       case "categoryName":
259         return 2;
260       default:
261         return 1;
262     }
263   }
264 </script>
```

ภาพที่ 14 ภาพ category-list.html (ต่อ)

```

category-list.html M X
src > main > resources > templates > category-list.html
294 <script>
295
296 // ฟังก์ชันที่ใช้ในการคิดค่าอินเด็กซ์เมื่อการค้นหา
297 function getColumnIndex(type) {
298     switch (type) {
299         case "categoryName":
300             return 2;
301         default:
302             return 1;
303     }
304 }
305
306 document.addEventListener('DOMContentLoaded', function() {
307     var messageElement = document.getElementById('message');
308     var errorElement = document.getElementById('error');
309
310     function hideMessage(element) {
311         if (element) {
312             setTimeout(function() {
313                 element.style.transition = 'opacity 1s';
314                 element.style.opacity = '0';
315                 setTimeout(function() {
316                     element.style.display = 'none';
317                 }, 1000); // Delay before hiding the element
318             }, 3000); // change this to 3000 milliseconds (3 seconds)
319         }
320     }
321
322     hideMessage(messageElement);
323     hideMessage(errorElement);
324 });
325
326
327 function confirmDelete() {
328     return confirm("ຕ້ອນນີ້ໃຫຍ່ຈະບໍ່ສາມາດລົບອອນ (Category) ດັ່ງນີ້?");
329 }
330
331 document.getElementById("clearButton").addEventListener("click", function () {
332     // ລົບການໃຫຍ່ອອນ
333     document.getElementById("searchInput").value = "";
334     var tableRows = document.querySelectorAll("#categoryTable .table-row");
335
336     tableRows.forEach(function (row) {
337         row.style.display = "";
338     });
339 });
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361

```

ກາພທີ 14 ກາພ category-list.html (ຕົວ)

```

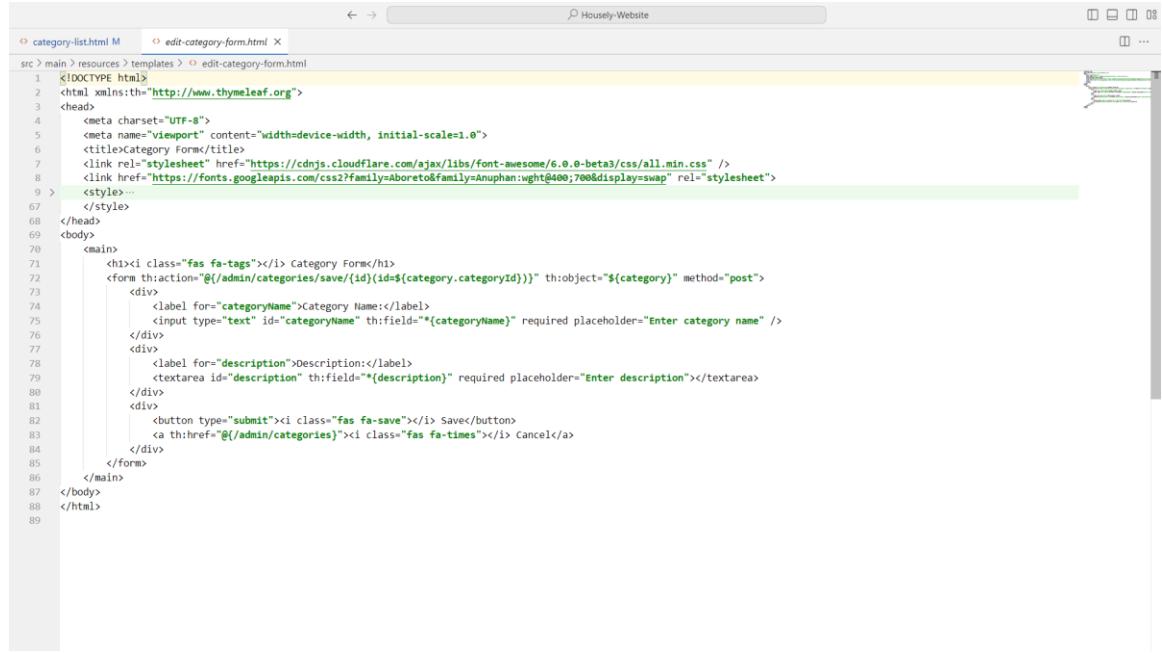
category-list.html M X
src > main > resources > templates > category-list.html
294 <script>
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361

```

ກາພທີ 14 ກາພ category-list.html (ຕົວ)

- edit-category-form.html

ฟอร์มนี้ถูกออกแบบมาเพื่อให้ผู้ใช้สามารถสร้างหรือแก้ไขข้อมูลหมวดหมู่ได้ โดยเมื่อผู้ใช้กรอกข้อมูลเสร็จแล้ว คลิกปุ่มบันทึก ฟอร์มจะส่งข้อมูลไปยังเซิร์ฟเวอร์เพื่อทำการบันทึกข้อมูลนั้นในฐานข้อมูล



```

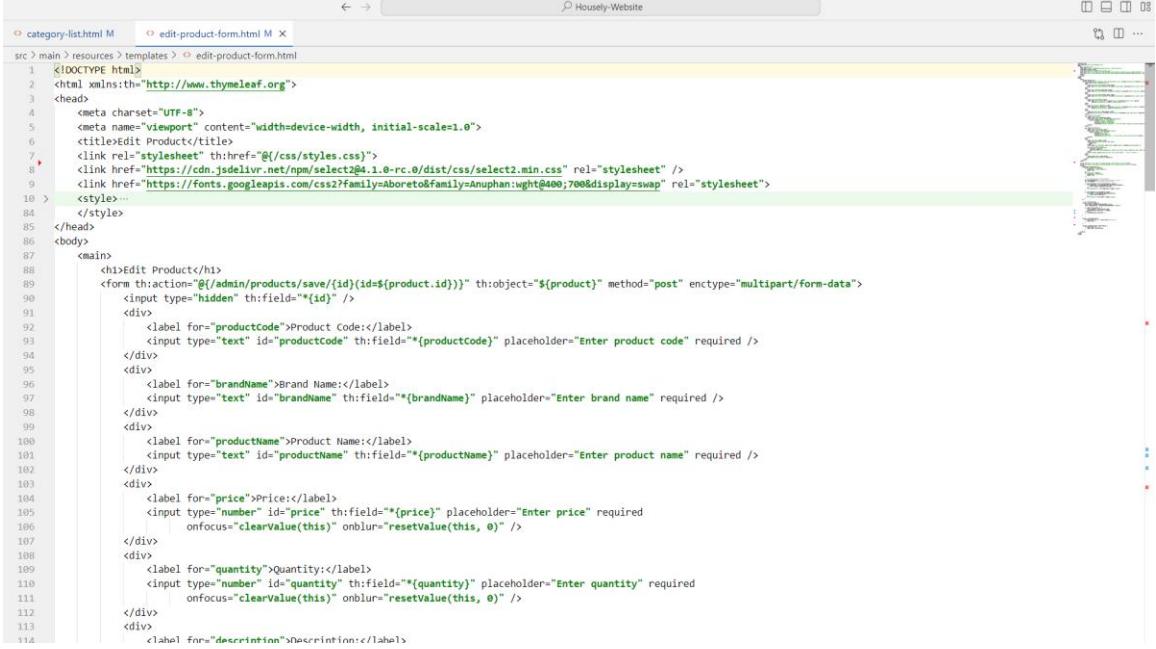
category-list.html M edit-category-form.html X
src > main > resources > templates > edit-category-form.html
1 <!DOCTYPE html>
2 <html xmlns="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Category Form</title>
7   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" />
8   <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9 > <style>...
67 </style>
68 </head>
69 <body>
70   <main>
71     <i class="fas fa-tags"></i> Category Form</h1>
72     <form th:action="@{/admin/categories/save/{id}}({category.categoryId})" th:object="${category}" method="post">
73       <div>
74         <label for="categoryName">Category Name:</label>
75         <input type="text" id="categoryName" th:field="*{categoryName}" required placeholder="Enter category name" />
76       </div>
77       <div>
78         <label for="description">Description:</label>
79         <textarea id="description" th:field="*{description}" required placeholder="Enter description"></textarea>
80       </div>
81       <div>
82         <button type="submit"><i class="fas fa-save"></i> Save</button>
83         <a th:href="@{/admin/categories}"><i class="fas fa-times"></i> Cancel</a>
84       </div>
85     </form>
86   </main>
87 </body>
88 </html>
89

```

ภาพที่ 15 ภาพ edit-category-form.html

- edit-product-form.html

ฟอร์มนี้ถูกออกแบบมาเพื่อให้ผู้ใช้สามารถสร้างหรือแก้ไขข้อมูลห้องได้ โดยเมื่อผู้ใช้กรอกข้อมูลเสร็จแล้วคลิกปุ่มบันทึก ฟอร์มจะส่งข้อมูลไปยังเซิร์ฟเวอร์เพื่อทำการบันทึกข้อมูลนั้นในฐานข้อมูล

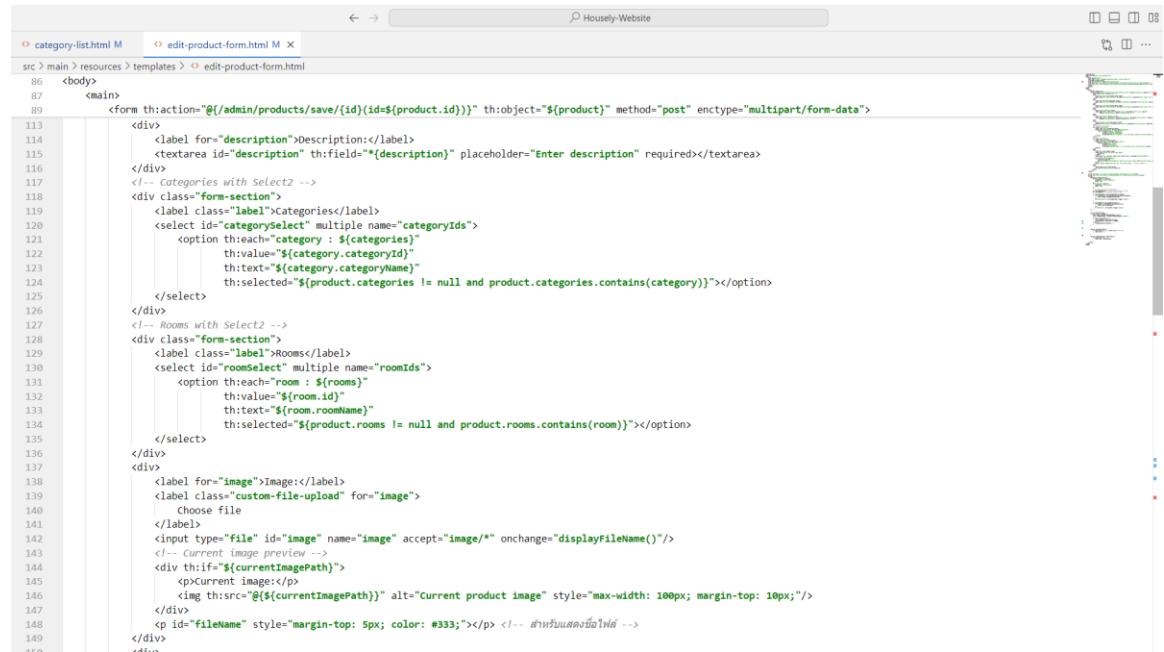


```

category-list.html M edit-product-form.html M
src > main > resources > templates > edit-product-form.html
1 <!DOCTYPE html>
2 <html xmlns="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Edit Product</title>
7   <link rel="stylesheet" th:href="@{/css/styles.css}">
8   <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
9   <link href="https://fonts.googleapis.com/css2?family=aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
10  <style>...
11  </style>
12  </head>
13  <body>
14    <main>
15      <h1>Edit Product</h1>
16      <form th:action="@{/admin/products/save/{id}{?product.id}}" th:object="${product}" method="post" enctype="multipart/form-data">
17        <input type="hidden" th:field="*{id}" />
18        <div>
19          <label for="productCode">Product Code:</label>
20          <input type="text" id="productCode" th:field="*{productCode}" placeholder="Enter product code" required />
21        </div>
22        <div>
23          <label for="brandName">Brand Name:</label>
24          <input type="text" id="brandName" th:field="*{brandName}" placeholder="Enter brand name" required />
25        </div>
26        <div>
27          <label for="productName">Product Name:</label>
28          <input type="text" id="productName" th:field="*{productName}" placeholder="Enter product name" required />
29        </div>
30        <div>
31          <label for="price">Price:</label>
32          <input type="number" id="price" th:field="*{price}" placeholder="Enter price" required
33            onfocus="clearValue(this)" onblur="resetValue(this, 0)" />
34        </div>
35        <div>
36          <label for="quantity">Quantity:</label>
37          <input type="number" id="quantity" th:field="*{quantity}" placeholder="Enter quantity" required
38            onfocus="clearValue(this)" onblur="resetValue(this, 0)" />
39        </div>
40        <div>
41          <label for="description">Description:</label>
42        </div>
43      </form>
44    </main>
45  </body>
46</html>

```

ภาพที่ 16 ภาพ edit-product-form.html

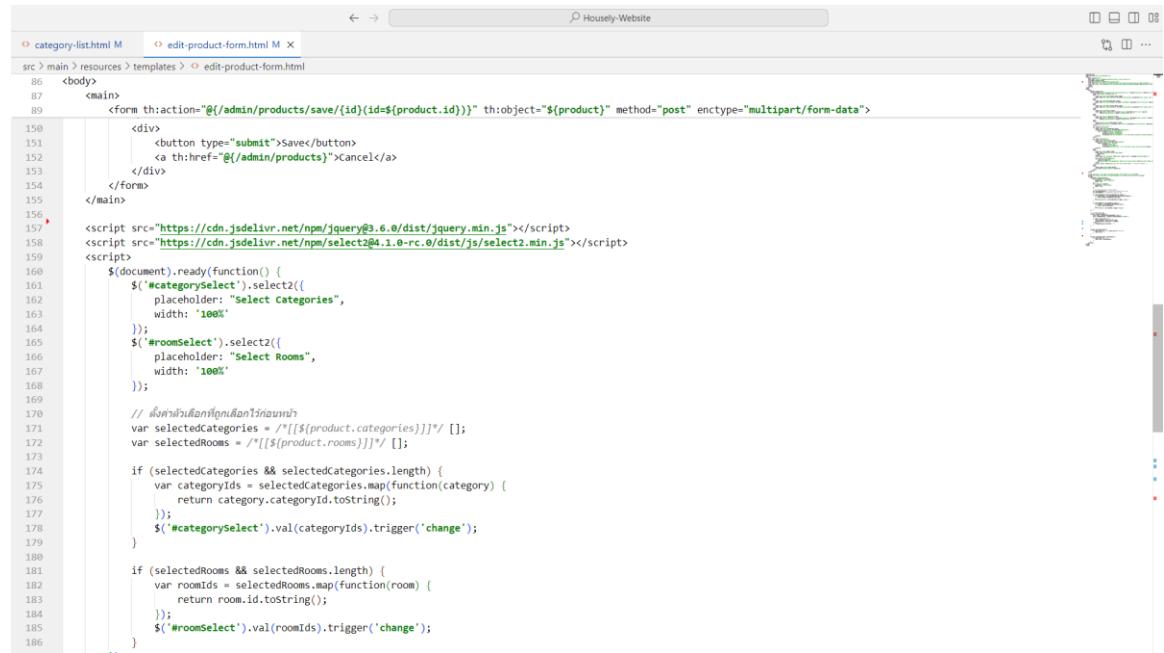


```

src > main > resources > templates > edit-product-form.html
86 <body>
87   <main>
88     <form th:action="@{/admin/products/save/{id ${product.id}}}" th:object="${product}" method="post" enctype="multipart/form-data">
89       <div>
113       <label for="description">Description:</label>
114       <textarea id="description" th:field="*{description}" placeholder="Enter description" required></textarea>
115     </div>
116     <!-- Categories with Select2 -->
117     <div class="form-section">
118       <label class="label">Categories</label>
119       <select id="categorySelect" multiple name="categoryIds">
120         <option th:each="category : ${categories}"
121             th:value="${category.categoryId}"
122             th:text="${category.categoryName}"
123             th:selected="${product.categories != null & product.categories.contains(category)}"></option>
124       </select>
125     </div>
126     <!-- Rooms with Select2 -->
127     <div class="form-section">
128       <label class="label">Rooms</label>
129       <select id="roomSelect" multiple name="roomIds">
130         <option th:each="room : ${rooms}"
131             th:value="${room.id}"
132             th:text="${room.roomName}"
133             th:selected="${product.rooms != null & product.rooms.contains(room)}"></option>
134       </select>
135     </div>
136     <div>
137       <label for="image">Image:</label>
138       <label class="custom-file-upload" for="image">
139         Choose file
140       </label>
141       <input type="file" id="image" name="image" accept="image/*" onchange="displayFileName()"/>
142       <!-- Current image preview -->
143       <div th:if="${currentImagePath}">
144         <p>Current image:</p>
145         
146       </div>
147       <p id="fileName" style="margin-top: 5px; color: #333;"></p> <!-- สำหรับแสดงชื่อไฟล์ -->
148     </div>
149   </form>

```

ภาพที่ 16 ภาพ edit-product-form.html (ต่อ)

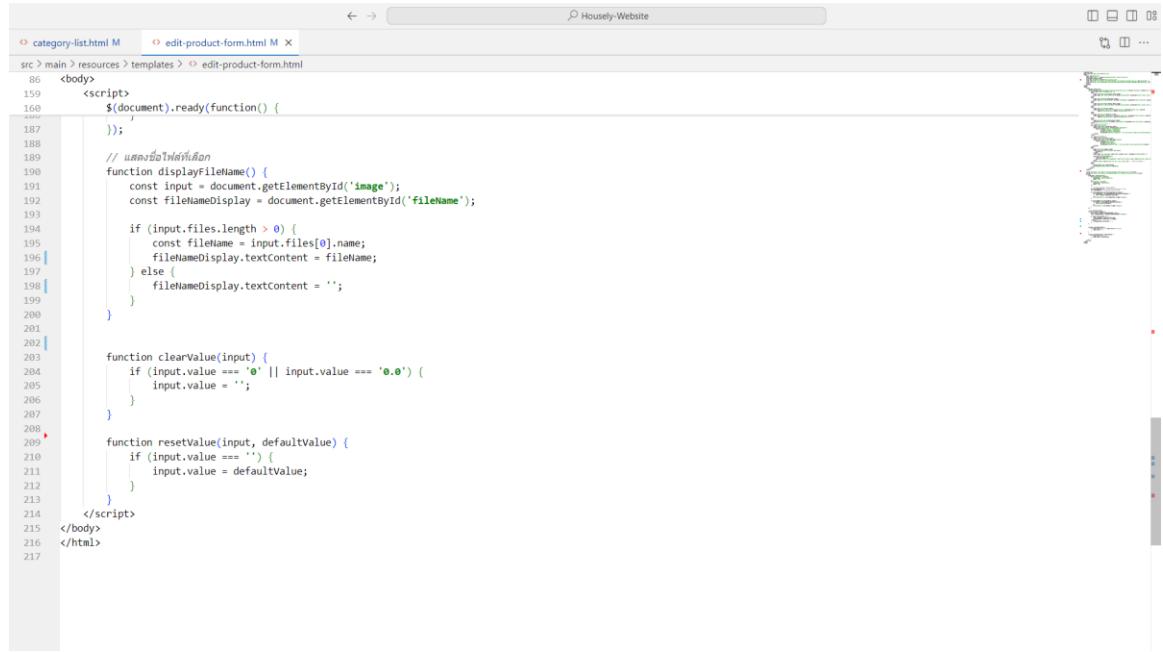


```

src > main > resources > templates > edit-product-form.html
86 <body>
87   <main>
88     <form th:action="@{/admin/products/save/{id ${product.id}}}" th:object="${product}" method="post" enctype="multipart/form-data">
89       <div>
150       <button type="submit">Save</button>
151       <a th:href="@{/admin/products}">Cancel</a>
152     </div>
153   </form>
154 </main>
155
156 <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.min.js"></script>
157 <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
158 <script>
159   $(document).ready(function() {
160     $('#categorySelect').select2({
161       placeholder: "Select Categories",
162       width: '100%'
163     });
164     $('#roomSelect').select2({
165       placeholder: "Select Rooms",
166       width: '100%'
167     });
168   });
169
170   // เมื่อกดเลือกที่กล่องเลือกจะเก็บมา
171   var selectedCategories = /*[[${product.categories}]]*/ [];
172   var selectedRooms = /*[[${product.rooms}]]*/ [];
173
174   if (selectedCategories && selectedCategories.length) {
175     var categoryId = selectedCategories.map(function(category) {
176       return category.categoryId.toString();
177     });
178     $('#categorySelect').val(categoryId).trigger('change');
179   }
180
181   if (selectedRooms && selectedRooms.length) {
182     var roomIds = selectedRooms.map(function(room) {
183       return room.id.toString();
184     });
185     $('#roomSelect').val(roomIds).trigger('change');
186   }

```

ภาพที่ 16 ภาพ edit-product-form.html (ต่อ)

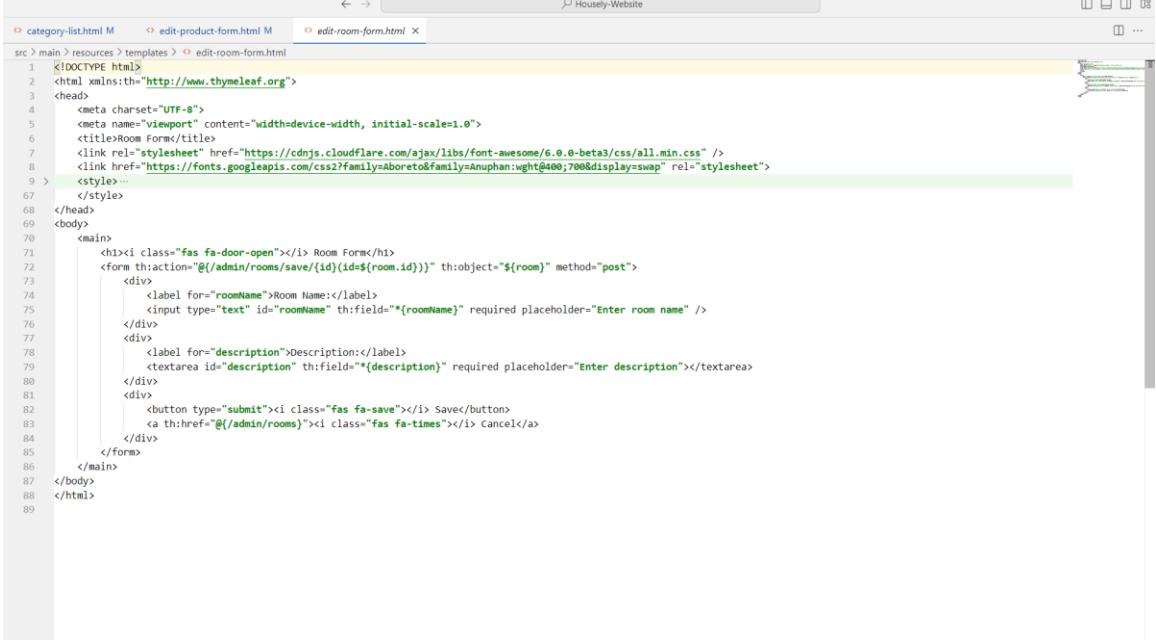


The screenshot shows a browser developer tools interface with the title bar "Housely-Website". The left pane displays the file structure: "category-list.html M" and "edit-product-form.html M X". The right pane shows the code editor with the file "edit-product-form.html". The code is as follows:

```
category-list.html M
edit-product-form.html M X
src > main > resources > templates > edit-product-form.html
86     <body>
87         <script>
88             $(document).ready(function() {
89                 );
90
91                 // မေတ္တာပုံစံအမြန်ထိန်းချွခဲ့ခြင်း
92                 function displayFileName() {
93                     const input = document.getElementById('image');
94                     const fileNameDisplay = document.getElementById('fileName');
95
96                     if (input.files.length > 0) {
97                         const fileName = input.files[0].name;
98                         fileNameDisplay.textContent = fileName;
99                     } else {
100                         fileNameDisplay.textContent = '';
101                     }
102
103                 function clearValue(input) {
104                     if (input.value === '0' || input.value === '0.0') {
105                         input.value = '';
106                     }
107
108                 function resetValue(input, defaultValue) {
109                     if (input.value === '') {
110                         input.value = defaultValue;
111                     }
112                 }
113             </script>
114         </body>
115     </html>
116
117
```

ภาพที่ 16 ภาพ edit-product-form.html (ต่อ)

- edit-room-form.html

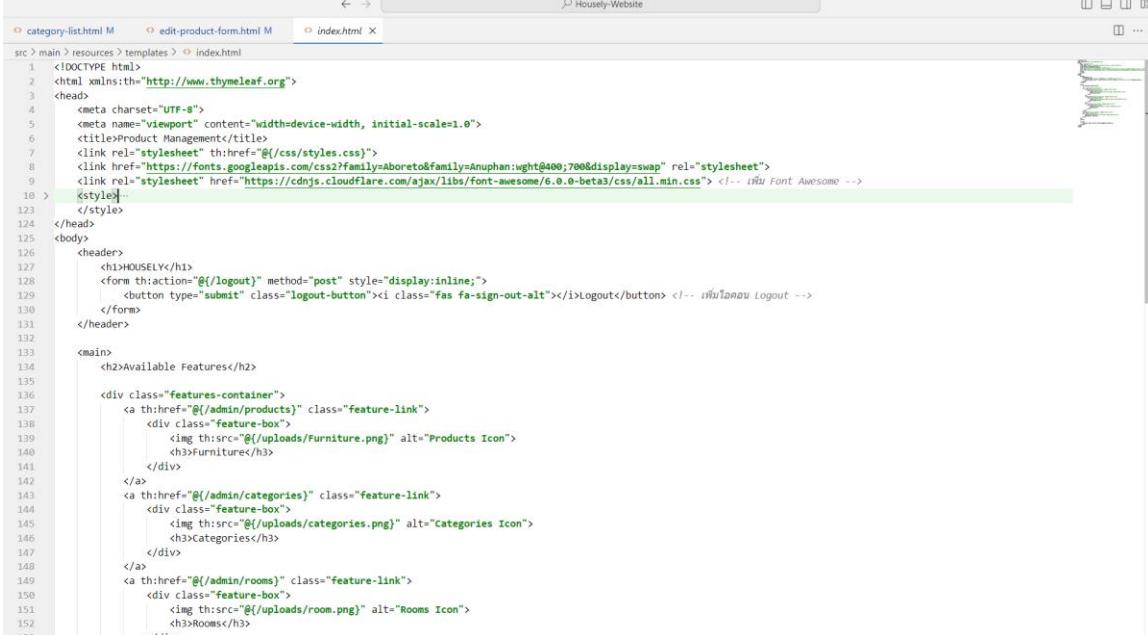


```
1 <!DOCTYPE html>
2 <html xmlns="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Room Form</title>
7   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" />
8   <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9 >   <style>...
10  </style>
11 </head>
12 <body>
13   <main>
14     <i class="fas fa-door-open"></i> Room Form</h1>
15     <form th:action="@{/admin/rooms/save/{id}(#{room.id})" th:object="${room}" method="post">
16       <div>
17         <label for="roomName">Room Name:</label>
18         <input type="text" id="roomName" th:field="*{roomName}" required placeholder="Enter room name" />
19       </div>
20       <div>
21         <label for="description">Description:</label>
22         <textarea id="description" th:field="*{description}" required placeholder="Enter description"></textarea>
23       </div>
24       <div>
25         <button type="submit"><i class="fas fa-save"></i> Save</button>
26         <a th:href="@{/admin/rooms}"><i class="fas fa-times"></i> Cancel</a>
27       </div>
28     </form>
29   </main>
30 </body>
31 </html>
```

ภาพที่ 17 ภาพ edit-room-form.html

- index.html

หน้าเว็บนี้ถูกออกแบบมาเพื่อให้ผู้ใช้งานสามารถเข้าถึงฟีเจอร์ต่างๆ ในระบบจัดการผลิตภัณฑ์ได้อย่างสะดวก โดยมีการนำเสนอด้วยภาพและข้อความที่ชัดเจน รวมถึงมีปุ่มสำหรับล็อกอิน/ออกจากระบบ

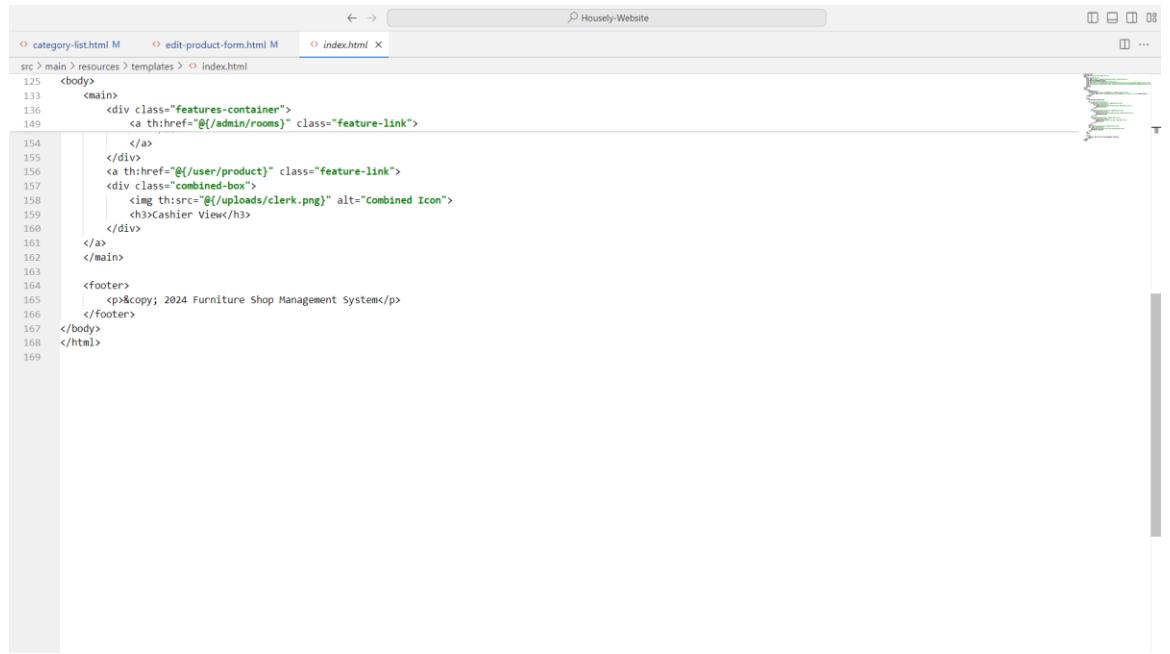


```

1 <!DOCTYPE html>
2 <html xmlns="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Product Management</title>
7   <link rel="stylesheet" href="@{/css/styles.css}">
8   <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.0.0-beta3/css/all.min.css"> <!-- ใช้ Font Awesome -->
10  <style>...
11  </style>
12 </head>
13 <body>
14   <header>
15     <h1>HOUSELY</h1>
16     <form th:action="@{/logout}" method="post" style="display:inline;">
17       <button type="submit" class="logout-button"><i class="fas fa-sign-out-alt"></i>Logout</button> <!-- ปุ่มLogout -->
18     </form>
19   </header>
20   <main>
21     <h2>Available Features</h2>
22
23     <div class="features-container">
24       <a th:href="@{/admin/products}" class="feature-link">
25         <div class="feature-box">
26           
27           <h3>Furniture</h3>
28         </div>
29       </a>
30       <a th:href="@{/admin/categories}" class="feature-link">
31         <div class="feature-box">
32           
33           <h3>Categories</h3>
34         </div>
35       </a>
36       <a th:href="@{/admin/rooms}" class="feature-link">
37         <div class="feature-box">
38           
39           <h3>Rooms</h3>
40         </div>
41     </div>
42   </main>
43   <footer>
44     <p>Housely - Product Management System</p>
45   </footer>
46 </body>
47 </html>

```

ภาพที่ 18 ภาพ index.html

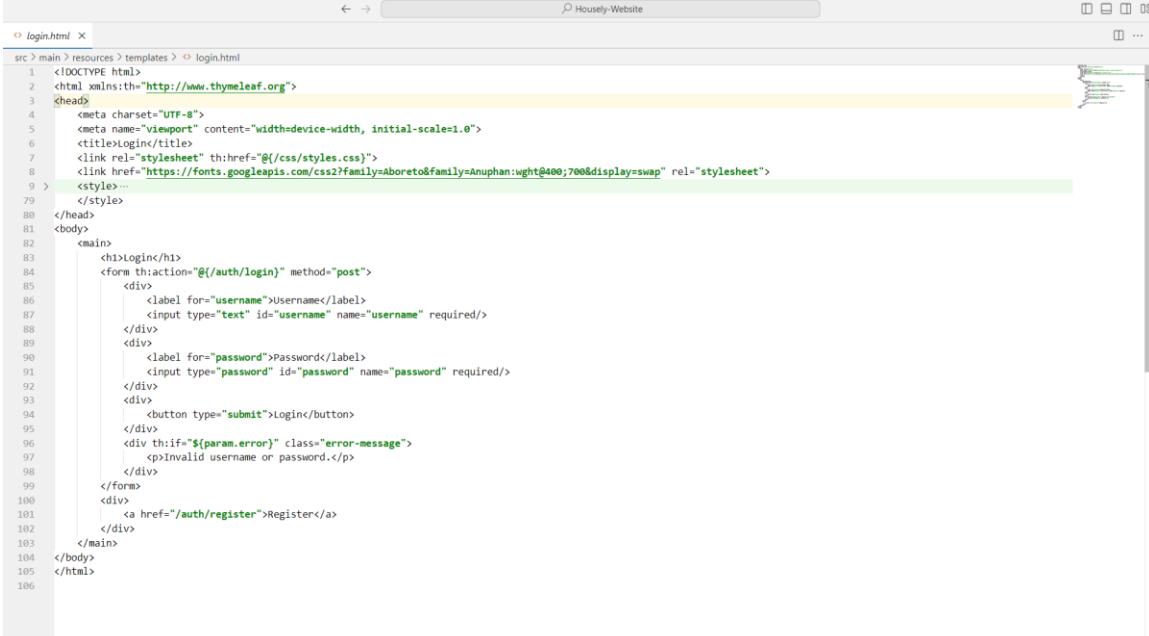


```
category-list.html M edit-product-form.html M index.html X
src > main > resources > templates > index.html
125   <body>
126     <main>
127       <div class="features-container">
128         <a th:href="@{/admin/rooms}" class="feature-link">
129           </a>
130         </div>
131         <a th:href="@{/user/product}" class="feature-link">
132           <div class="combined-box">
133             
134             <h3>Cashier View</h3>
135           </div>
136         </a>
137       </main>
138     <footer>
139       <p>&copy; 2024 Furniture Shop Management System</p>
140     </footer>
141   </body>
142 </html>
143
144
145
146
147
148
149
```

ภาพที่ 18 ภาพ index.html (ต่อ)

- login.html

หน้าเว็บนี้ถูกออกแบบมาเพื่อให้ผู้ใช้งานสามารถเข้าสู่ระบบได้ โดยมีการกรอกชื่อผู้ใช้และรหัสผ่าน และมีปุ่มสำหรับส่งข้อมูลเข้าสู่ระบบ นอกจากนี้ยังมีลิงก์สำหรับการลงทะเบียนผู้ใช้ใหม่อีกด้วย



```

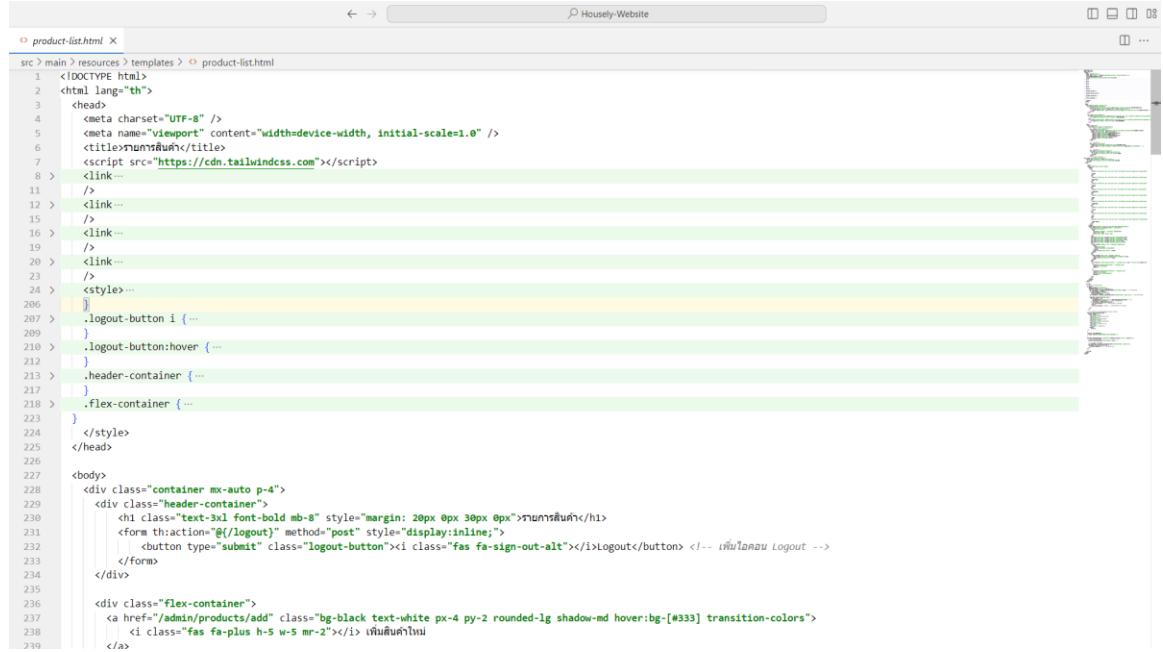
src > main > resources > templates > login.html
1  <!DOCTYPE html>
2  <html xmlns="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Login</title>
7      <link rel="stylesheet" th:href="@{/css/styles.css}">
8      <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9  >
10     <style>...
11     </style>
12  </head>
13  <body>
14      <main>
15          <h1>Login</h1>
16          <form th:action="@{/auth/login}" method="post">
17              <div>
18                  <label for="username">Username</label>
19                  <input type="text" id="username" name="username" required/>
20              </div>
21              <div>
22                  <label for="password">Password</label>
23                  <input type="password" id="password" name="password" required/>
24              </div>
25              <div>
26                  <button type="submit">Login</button>
27              </div>
28              <div th:if="${param.error}" class="error-message">
29                  <p>Invalid username or password.</p>
30              </div>
31          </form>
32          <div>
33              <a href="/auth/register">Register</a>
34          </div>
35      </main>
36  </body>
37  </html>
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106

```

ภาพที่ 19 ภาพ login.html

- product-list.html

หน้านี้คือหน้า "รายการสินค้า" สำหรับการจัดการสินค้าในระบบ โดยมีฟังก์ชันการค้นหาและแสดงรายละเอียดสินค้า รวมถึงปุ่มสำหรับเพิ่ม แก้ไข และลบสินค้า มีตารางแสดงข้อมูลต่าง ๆ เช่น รูปภาพ รหัสสินค้า ยี่ห้อ ชื่อสินค้า ราคา จำนวนสต็อก ประเภท และห้อง นอกจากนี้ยังมีปุ่มออกจากระบบและปุ่มล้างการค้นหาเพื่อคืนค่าตารางกลับเป็นปกติ

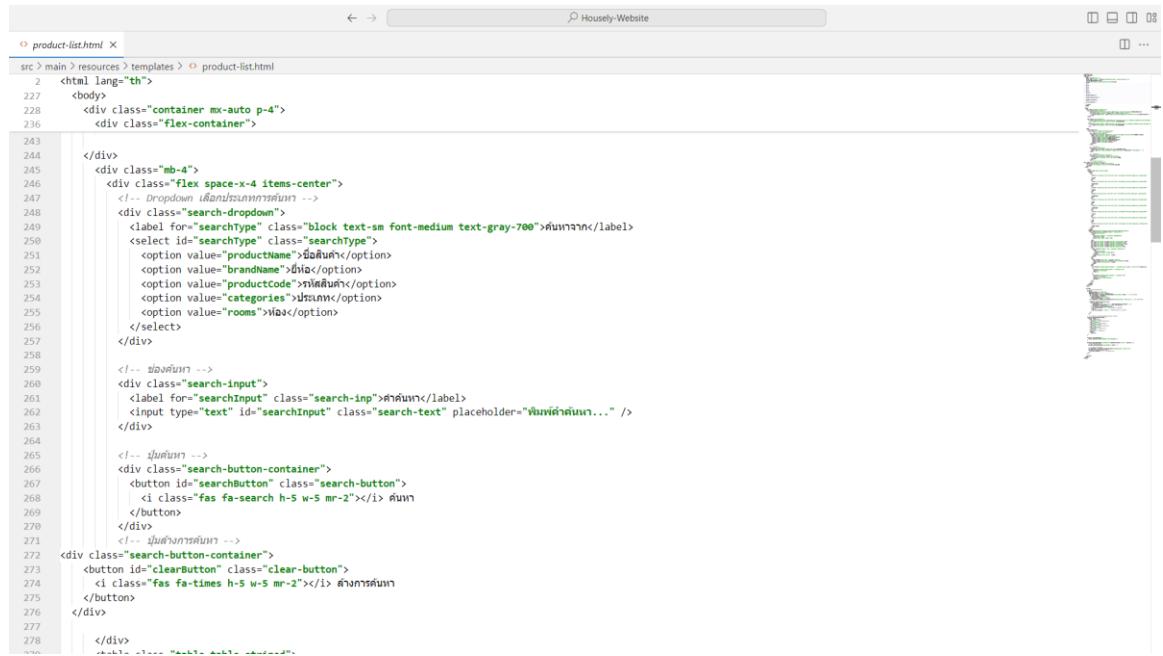


```

src > main > resources > templates > product-list.html
1  <!DOCTYPE html>
2  <html lang="th">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Housey-Website</title>
7      <script src="https://cdn.tailwindcss.com"></script>
8  >     <link ...>
9  >     <link ...>
10 >    <link ...>
11 >    <link ...>
12 >    <link ...>
13 >    <link ...>
14 >    <link ...>
15 >    <link ...>
16 >    <link ...>
17 >    <link ...>
18 >    <link ...>
19 >    <link ...>
20 >    <link ...>
21 >    <link ...>
22 >    <link ...>
23 >    <link ...>
24 >    <style>...
25 >    ...
26 >    .logout-button i ...
27 >    ...
28 >    .logout-button:hover ...
29 >    ...
30 >    .header-container ...
31 >    ...
32 >    ...
33 >    ...
34 >    ...
35 >    ...
36 >    ...
37 >    ...
38 >    ...
39 >    ...

```

ภาพที่ 20 ภาพ product-list.html



```

src > main > resources > templates > product-list.html
1  <html lang="th">
2  <body>
3  <div class="container mx-auto p-4">
4  <div class="flex-container">
5      <div class="header-container">
6          <h1 class="text-3xl font-bold mb-8" style="margin: 20px 0px 30px 0px">รายการสินค้า</h1>
7          <form th:action="@{/logout}" method="post" style="display:inline;">
8              <button type="submit" class="logout-button">i class="fas fa-sign-out-alt"</i>Logout</button> <!-- เมื่อไปคลิก Logout -->
9          </form>
10 </div>
11 </div>
12 <div class="flex-container">
13     <a href="/admin/products/add" class="bg-black text-white px-4 py-2 rounded-lg shadow-md hover:bg-[#333] transition-colors">
14         <i class="fas fa-plus h-5 w-5 mr-2"></i> เพิ่มสินค้าใหม่
15     </a>
16 </div>
17 </div>
18 </div>
19 <div class="mb-4">
20     <div class="flex space-x-4 items-center">
21         <!-- dropdown เมื่อป้อนลงมาแล้วพิมพ์ -->
22         <div class="search-dropdown">
23             <label for="searchType" class="block text-sm font-medium text-gray-700">ค้นหาด้วย</label>
24             <select id="searchType" class="searchType">
25                 <option value="productName">ชื่อสินค้า</option>
26                 <option value="brandName">ยี่ห้อ</option>
27                 <option value="productCode">รหัสสินค้า</option>
28                 <option value="categories">ประเภท</option>
29                 <option value="rooms">ห้อง</option>
30             </select>
31         </div>
32         <!-- ปุ่มค้นหา -->
33         <div class="search-input">
34             <label for="searchInput" class="search-inp">ค้นหา</label>
35             <input type="text" id="searchInput" class="search-text" placeholder="พิมพ์คำค้นหา..." />
36         </div>
37         <!-- ปุ่มต่อการค้นหา -->
38         <div class="search-button-container">
39             <button id="searchButton" class="search-button">
40                 <i class="fas fa-search h-5 w-5 mr-2"></i> ค้นหา
41             </button>
42         </div>
43         <!-- ปุ่มลบการค้นหา -->
44         <div class="search-button-container">
45             <button id="clearButton" class="clear-button">
46                 <i class="fas fa-times h-5 w-5 mr-2"></i> ล้างค้นหา
47             </button>
48         </div>
49     </div>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </div>
102 </div>
103 </div>
104 </div>
105 </div>
106 </div>
107 </div>
108 </div>
109 </div>
110 </div>
111 </div>
112 </div>
113 </div>
114 </div>
115 </div>
116 </div>
117 </div>
118 </div>
119 </div>
120 </div>
121 </div>
122 </div>
123 </div>
124 </div>
125 </div>
126 </div>
127 </div>
128 </div>
129 </div>
130 </div>
131 </div>
132 </div>
133 </div>
134 </div>
135 </div>
136 </div>
137 </div>
138 </div>
139 </div>
140 </div>
141 </div>
142 </div>
143 </div>
144 </div>
145 </div>
146 </div>
147 </div>
148 </div>
149 </div>
150 </div>
151 </div>
152 </div>
153 </div>
154 </div>
155 </div>
156 </div>
157 </div>
158 </div>
159 </div>
160 </div>
161 </div>
162 </div>
163 </div>
164 </div>
165 </div>
166 </div>
167 </div>
168 </div>
169 </div>
170 </div>
171 </div>
172 </div>
173 </div>
174 </div>
175 </div>
176 </div>
177 </div>
178 </div>
179 </div>
180 </div>
181 </div>
182 </div>
183 </div>
184 </div>
185 </div>
186 </div>
187 </div>
188 </div>
189 </div>
190 </div>
191 </div>
192 </div>
193 </div>
194 </div>
195 </div>
196 </div>
197 </div>
198 </div>
199 </div>
200 </div>
201 </div>
202 </div>
203 </div>
204 </div>
205 </div>
206 </div>
207 </div>
208 </div>
209 </div>
210 </div>
211 </div>
212 </div>
213 </div>
214 </div>
215 </div>
216 </div>
217 </div>
218 </div>
219 </div>
220 </div>
221 </div>
222 </div>
223 </div>
224 </div>
225 </div>
226 </div>
227 </div>
228 </div>
229 </div>
230 </div>
231 </div>
232 </div>
233 </div>
234 </div>
235 </div>
236 </div>
237 </div>
238 </div>
239 </div>

```

ภาพที่ 20 ภาพ product-list.html (ต่อ)

```

278     </div>
279     <table class="table table-striped">
280       <thead>
281         <tr>
282           <th>
283             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
284             ឈ្មោះ
285           </th>
286           <th>
287             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
288             នាយកដំណើរ
289           </th>
290           <th>
291             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
292             បូត្រ
293           </th>
294           <th>
295             class="col-name px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
296             ឯកសារ
297           </th>
298           <th>
299             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
300             ចានមនីតុល
301           </th>
302           <th>
303             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
304             ភាគ
305           </th>
306           <th>
307             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
308             ចានមនីតុល
309           </th>
310           <th>
311             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
312             មេរោគ
313           </th>
314           <th>
315             class="col-fixed px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
316             មេរោគ
317           </th>
318           <th>
319             class="col-action px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
320             ឯកតាមបញ្ជាក់
321           </th>
322           <th>
323             class="col-action px-6 py-3 text-left text-1 font-medium text-white uppercase tracking-wider"
324             ការតាមបញ្ជាក់
325           </th>
326           </tr>
327         </thead>
328         <tbody class="bg-white divide-y divide-gray-200" id="productTable">
329           <tr class="table-row" th:each="product : ${products}">
330             <td class="word-break">
331               
336             </td>
337             <td class="word-break" th:text="${product.productCode}"></td>
338             <td class="word-break" th:text="${product.brandName}"></td>
339             <td class="word-break" th:text="${product.productName}"></td>
340             <td class="word-break" th:text="${product.price}"></td>
341             <td class="word-break" th:text="${product.quantity}"></td>
342             <td>
343               <span th:each="category, stat : ${product.categories}">
344                 <span
345                   class="word-break"
346                   th:text="${category.categoryName}"
347                 ></span>
348                 <span th:if="${!stat.last}">, </span>
349               </span>
350             </td>

```

រាយទី 20 រាយ product-list.html (ពីរ)

```

351             </span>
352           </td>
353         </tr>
354       </tbody>
355     </table>
356   </div>
357 
```

រាយទី 20 រាយ product-list.html (ពីរ)

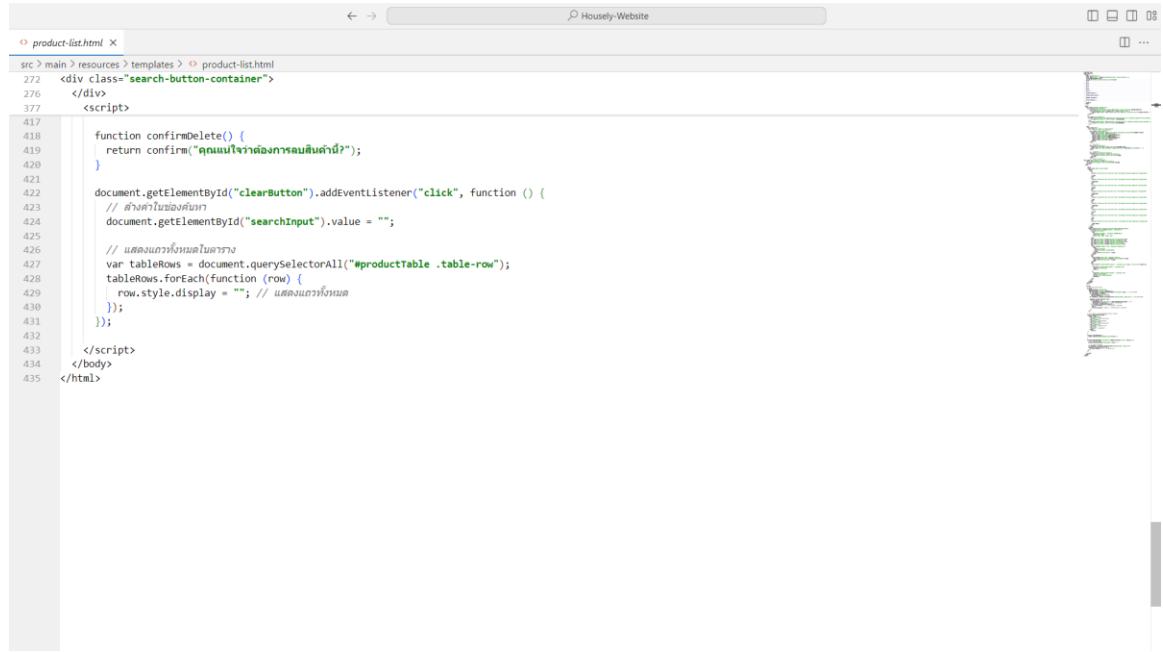
```
src > main > resources > templates > product-list.html
272 <div class="search-button-container">
273   </div>
274   <table class="table table-striped">
275     <tbody class="bg-white divide-y divide-gray-200" id="productTable">
276       <tr class="table-row" th:each="product : ${products}">
277         <td>
278           <span>
279             <span th:each="room, stat : ${product.rooms}">
280               <span class="word-break" th:text="${room.roomName}"></span>
281               <span th:if="${!stat.last}">, </span>
282             </span>
283           </span>
284         </td>
285         <td>
286           <a th:href="@('/admin/products/detail/' + ${product.id})" class = "btn btn-info">Detail</a>
287           <a th:href="@('/admin/products/edit/' + ${product.id})"
288             class="btn btn-warning">Edit</a>
289           <a th:href="@('/admin/products/delete/' + ${product.id})"
290             class="btn btn-danger"
291             onclick="return confirmDelete()">Delete</a>
292         </td>
293       </tr>
294     </tbody>
295   </table>
296 </div>

297 <script>
298 // ฟังก์ชันการค้นหาข้อมูล
299 document
300   .getElementById("searchButton")
301   .addEventListener("click", function () {
302     var searchType = document.getElementById("searchType").value; // ประเภทการค้นหา
303     var searchInput = document
304       .getElementById("searchInput")
```

ภาพที่ 20 ภาพ product-list.html (ต่อ)

```
product-list.html X
src > main > resources > templates > product-list.html
272 <div class="search-button-container">
273   </div>
274   <script>
275     document
276       .addEventListener("click", function () {
277         var searchInput = document
278           .getElementById("searchInput")
279           .value.toLowerCase(); // ຄຳນັ້ນທາງ
280         var tableRows = document.querySelectorAll("#productTable .table-row"); // ແກ້ວຂອບໃຈໃນຕາຮາງ
281
282         tableRows.forEach(function (row) {
283           var cellValue = row
284             .querySelector(`td:nth-child(${getColumnIndex(searchType)} + ${})`)
285             .textContent.toLowerCase(); // ຄຳນັ້ນຂອບໃຈທີ່ຄຳນັ້ນທາງ
286           if (cellValue.includes(searchInput)) {
287             row.style.display = ""; // ແກ້ວຂອບໃຈກິດຄຳນັ້ນທາງ
288           } else {
289             row.style.display = "none"; // ແກ້ວຂອບໃຈທີ່ໄປຮອກກິດຄຳນັ້ນທາງ
290           }
291         });
292       });
293     });
294
295     // ພຶກອົບນີ້ໃຊ້ໄວ້ການເຫັນຄວາມຄືນທີ່ຜົດການຄຳນັ້ນທາງ
296     function getColumnIndex(type) {
297       switch (type) {
298         case "productCode":
299           return 2; // ຂອບໃຈກິດຄຳນັ້ນທາງ
300         case "brandName":
301           return 3; // ຂອບໃຈເປົ້າໂອັດ
302         case "productName":
303           return 4; // ຂອບໃຈເປົ້າເລີ່ມຕົ້ນຄໍາ
304         case "categories":
305           return 8; // ຂອບໃຈນິຍາກາ
306         case "rooms":
307           return 9; // ຂອບໃຈນິຍານຸ່ມ
308         default:
309           return 0;
310       }
311     }
312
313     function configDelete() {
314       ...
315     }
316   </script>
317 
```

ภาพที่ 20 ภาพ product-list.html (ต่อ)

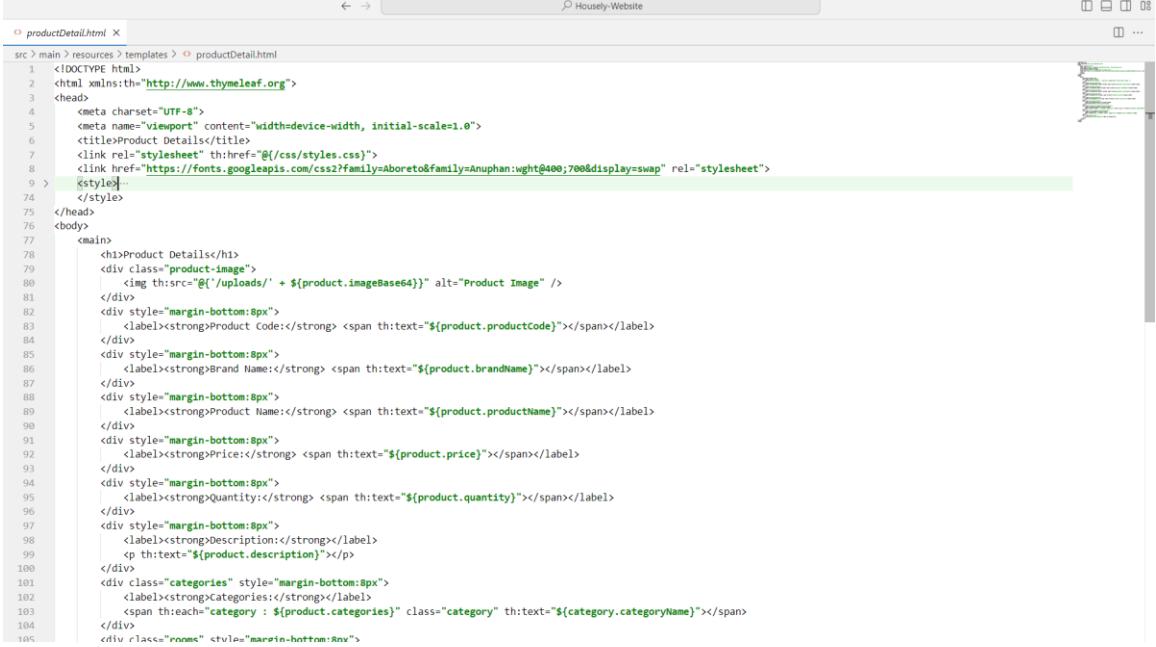


The screenshot shows a browser window with the title "Housely-Website". The address bar contains the URL "http://localhost:3000/product-list". The main content area displays the source code of the file "product-list.html". The code is written in JavaScript and includes comments in Thai. The code handles a confirmation dialog for deleting items, clears the search input field, and hides table rows. The browser's developer tools sidebar is visible on the right, showing the DOM tree and other developer information.

```
src > main > resources > templates > product-list.html
272   <div class="search-button-container">
273     </div>
274   <script>
275
276     function confirmDelete() {
277       return confirm("คุณแนใจว่าต้องการลบข้อมูลนี้?");
278     }
279
280     document.getElementById("clearButton").addEventListener("click", function () {
281       // ล้างค่าในฟิลด์ค้นหา
282       document.getElementById("searchInput").value = "";
283
284       // แสดงผลลัพธ์ใหม่ๆ
285       var tableRows = document.querySelectorAll("#productTable .table-row");
286       tableRows.forEach(function (row) {
287         row.style.display = ""; // รีบูตผลลัพธ์ใหม่
288       });
289     });
290
291   </script>
292 </body>
293 </html>
```

ภาพที่ 20 ภาพ product-list.html (ต่อ)

- productDetail.html

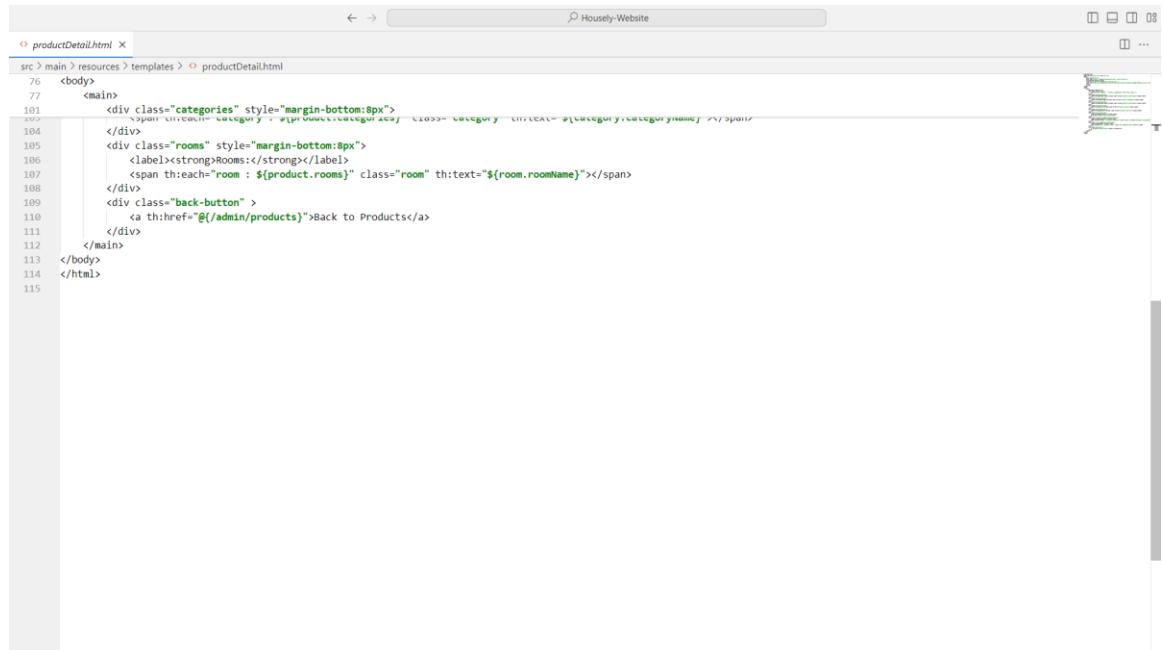


```

productDetail.html
src > main > resources > templates > productDetail.html
1  <!DOCTYPE html>
2  <html xmlns="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Product Details</title>
7      <link rel="stylesheet" th:href="@{/css/styles.css}">
8      <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9  >     <style> ...
10 </style>
11 </head>
12 <body>
13     <main>
14         <h1>Product Details</h1>
15         <div class="product-image">
16             
17         </div>
18         <div style="margin-bottom:8px">
19             <label><strong>Product Code:</strong> <span th:text="${product.productCode}"></span></label>
20         </div>
21         <div style="margin-bottom:8px">
22             <label><strong>Brand Name:</strong> <span th:text="${product.brandName}"></span></label>
23         </div>
24         <div style="margin-bottom:8px">
25             <label><strong>Product Name:</strong> <span th:text="${product.productName}"></span></label>
26         </div>
27         <div style="margin-bottom:8px">
28             <label><strong>Price:</strong> <span th:text="${product.price}"></span></label>
29         </div>
30         <div style="margin-bottom:8px">
31             <label><strong>Quantity:</strong> <span th:text="${product.quantity}"></span></label>
32         </div>
33         <div style="margin-bottom:8px">
34             <label><strong>Description:</strong></label>
35             <p th:text="${product.description}"></p>
36         </div>
37         <div class="categories" style="margin-bottom:8px">
38             <label><strong>Categories:</strong></label>
39             <span th:each="category : ${product.categories}" class="category" th:text="${category.categoryName}"></span>
40         </div>
41         <div class="rname" style="margin-bottom:8px">

```

ภาพที่ 21 ภาพ productDetail.html



```

productDetail.html
src > main > resources > templates > productDetail.html
76   <body>
77     <main>
101       <div class="categories" style="margin-bottom:8px">
102         <span><!--category : ${product.categories}--><span>${category}</span><span>${category.categoryName}</span></span>
103       </div>
104       <div class="rooms" style="margin-bottom:8px">
105         <label><strong>Rooms:</strong></label>
106         <span th:each="room : ${product.rooms}" class="room" th:text="${room.roomName}"></span>
107       </div>
108       <div class="back-button" >
109         <a th:href="@{/admin/products}">Back to Products</a>
110       </div>
111     </main>
112   </body>
113 </html>
114
115

```

ภาพที่ 21 ภาพ productDetail.html (ต่อ)

- productDetailUser.html

นี่คือโค้ด HTML สำหรับหน้า "รายละเอียดสินค้า" ซึ่งแสดงข้อมูลสินค้า รวมถึงภาพสินค้า รหัสสินค้า ยี่ห้อ ชื่อ ราคา จำนวน และคำอธิบาย นอกจากนี้ยังมีหมวดหมู่และห้องที่เกี่ยวข้องกับสินค้า และปุ่มสำหรับกลับไปยังหน้า รายการสินค้า มีการใช้สтиล์ CSS เพื่อให้หน้าเว็บมีความสวยงามและอ่านง่าย

```
src > main > resources > templates > productDetailUser.html
1  <!DOCTYPE html>
2  
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Product Details</title>
7      <link rel="stylesheet" th:href="@{/css/styles.css}">
8      <link href="https://fonts.googleapis.com/css?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9  >
10     <style>...
11     </style>
12 </head>
13 <body>
14     <main>
15         <h1>Product Details</h1>
16         <div class="product-image">
17             
18         </div>
19         <div style="margin-bottom:8px">
20             <label><strong>Product Code:</strong> <span th:text="${product.productCode}"></span></label>
21         </div>
22         <div style="margin-bottom:8px">
23             <label><strong>Brand Name:</strong> <span th:text="${product.brandName}"></span></label>
24         </div>
25         <div style="margin-bottom:8px">
26             <label><strong>Product Name:</strong> <span th:text="${product.productName}"></span></label>
27         </div>
28         <div style="margin-bottom:8px">
29             <label><strong>Price:</strong> <span th:text="${product.price}"></span></label>
30         </div>
31         <div style="margin-bottom:8px">
32             <label><strong>Quantity:</strong> <span th:text="${product.quantity}"></span></label>
33         </div>
34         <div style="margin-bottom:8px">
35             <label><strong>Description:</strong></label>
36             <p th:text="${product.description}"></p>
37         </div>
38         <div class="categories" style="margin-bottom:8px">
39             <label><strong>Categories:</strong></label>
40             <span th:each="category : ${product.categories}" class="category" th:text="${category.categoryName}"></span>
41         </div>
42         <div class="rnames" style="margin-bottom:8px">
43             <table>
44                 <thead>
45                     <tr>
46                         <th>Category Name</th>
47                         <th>Category Description</th>
48                     </tr>
49                 </thead>
50                 <tbody>
51                     <tr>
52                         <td>${category.categoryName}</td>
53                         <td>${category.categoryDescription}</td>
54                     </tr>
55                 </tbody>
56             </table>
57         </div>
58     </main>
59 </body>
60 </html>
```

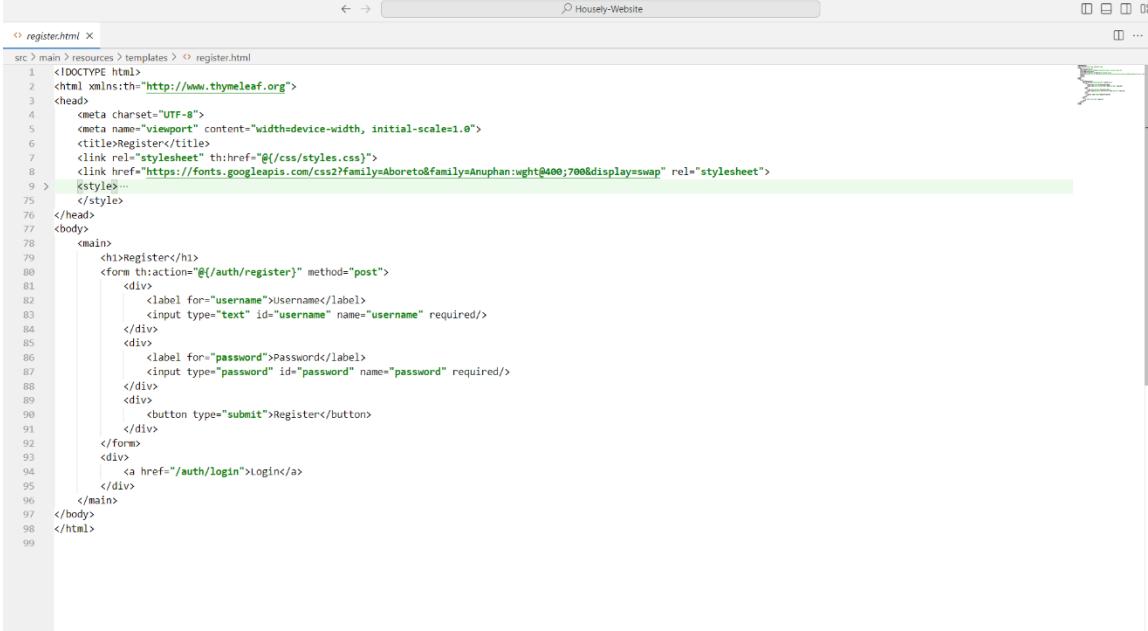
ภาพที่ 22 ภาพ productDetailUser.html

```
productDetailUser.html ×  
src > main > resources > templates > productDetailUser.html  
76 <body>  
77   <main>  
78     </div>  
79   <div class="rooms" style="margin-bottom:8px">  
80     <label><strong>Rooms:</strong></label>  
81     <span th:each="room : ${product.rooms}" class="room" th:text="${room.roomName}"></span>  
82   </div>  
83   <div class="back-button" >  
84     <a th:href="@{/user/product}">Back to Products</a>  
85   </div>  
86 </main>  
87 </body>  
88 </html>  
89
```

ภาพที่ 22 ภาพ productDetailUser.html (ต่อ)

- register.html

สำหรับหน้า "ลงทะเบียน" ที่มีฟอร์มสำหรับกรอกชื่อผู้ใช้และรหัสผ่าน โดยมีปุ่มสำหรับส่งข้อมูลไปยังเซิร์ฟเวอร์เพื่อทำการลงทะเบียน นอกจากนี้ยังมีลิงก์เพื่อไปยังหน้าเข้าสู่ระบบ มีการใช้สైట్‌CSS เพื่อให้หน้ามีความสวยงามและใช้งานง่าย



```

src > main > resources > templates > register.html
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Register</title>
7      <link rel="stylesheet" th:href="@{/css/styles.css}">
8      <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@400;700&display=swap" rel="stylesheet">
9      <style>...
10     </style>
11  </head>
12  <body>
13      <main>
14          <h1>Register</h1>
15          <form th:action="@{/auth/register}" method="post">
16              <div>
17                  <label for="username">Username</label>
18                  <input type="text" id="username" name="username" required/>
19              </div>
20              <div>
21                  <label for="password">Password</label>
22                  <input type="password" id="password" name="password" required/>
23              </div>
24              <div>
25                  <button type="submit">Register</button>
26              </div>
27          </form>
28          <div>
29              <a href="/auth/login">Login</a>
30          </div>
31      </main>
32  </body>
33 </html>
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

ภาพที่ 23 ภาพ register.html

- room-list.html

โครงสร้างหลัก: เป็นไฟล์ HTML ที่ใช้ Thymeleaf สำหรับการสร้างหน้าเว็บ "รายการห้อง" โดยมีการกำหนดภาษาเป็นภาษาไทย (lang="th")

การจัดการสైట్‌CSS: ใช้ Tailwind CSS และฟอนต์จาก Google Fonts รวมถึง CSS แบบกำหนดเองเพื่อปรับแต่งการแสดงผลของหน้าเว็บ เช่น สีพื้นหลัง, สีตัวอักษร, และการอักษรแบบตาราง

ส่วนหัว: มีการแสดงชื่อหน้าว่า "รายการห้อง" และปุ่มออกจากระบบ

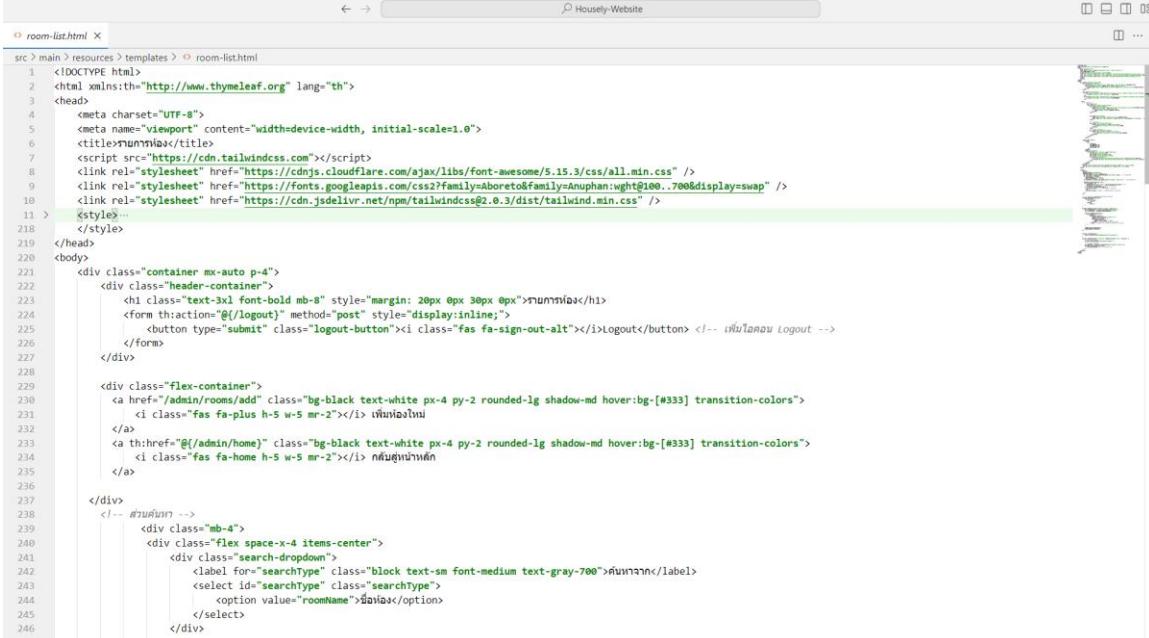
การค้นหา: มีฟังก์ชันการค้นหาห้อง โดยสามารถเลือกประเภทการค้นหา (ชื่อห้อง) และกรอกคำค้นหา พร้อมปุ่มค้นหาและล้างการค้นหา

ตารางข้อมูล: แสดงข้อมูลห้องในรูปแบบตาราง ซึ่งมีคอลัมน์สำหรับ ID, ชื่อห้อง, คำอธิบาย และตัวเลือก (ปุ่มแก้ไขและลบ)

การยืนยันการลบ: มีการยืนยันเมื่อผู้ใช้คลิกปุ่มลบห้องเพื่อป้องกันการลบโดยไม่ตั้งใจ

ข้อความแจ้งเตือน: มีฟังก์ชันที่แสดงข้อความแจ้งเตือนและข้อผิดพลาด โดยจะซ่อนข้อความหลังจาก 3 วินาที.

JavaScript: ใช้ JavaScript สำหรับการค้นหาในตารางและการจัดการการแสดงผลของข้อความแจ้งเตือน โดยรวมแล้ว เป็นหน้าเว็บที่มีการจัดระเบียบดี ใช้งานง่าย และมีฟังก์ชันการค้นหาที่มีประสิทธิภาพ



```

src > main > resources > templates > room-list.html
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org" lang="th">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>numnab</title>
7      <script src="https://cdn.tailwindcss.com"></script>
8      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />
9      <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Aboreto&family=Abuphan:wght@100..700&display=swap" />
10     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/tailwindcss@2.0.3/dist/tailwind.min.css" />
11     <style>...
12     </style>
13 </head>
14 <body>
15     <div class="container mx-auto p-4">
16         <div class="header-container">
17             <h1 class="text-3xl font-bold mb-8" style="margin: 20px 0px 30px 0px">ระบบบ้าน</h1>
18             <form th:action="@{/logout}" method="post" style="display:inline;">
19                 <button type="submit" class="logout-button"><i class="fas fa-sign-out-alt"></i>Logout</button> <!-- ปุ่มออก Logout -->
20             </form>
21         </div>
22         <div class="flex-container">
23             <a href="/admin/rooms/add" class="bg-black text-white px-4 py-2 rounded-lg shadow-md hover:bg-[#333] transition-colors">
24                 <i class="fas fa-plus h-5 w-5 mr-2"></i> เพิ่มบ้านใหม่
25             </a>
26             <a th:href="@{/admin/home}" class="bg-black text-white px-4 py-2 rounded-lg shadow-md hover:bg-[#333] transition-colors">
27                 <i class="fas fa-home h-5 w-5 mr-2"></i> หน้าหลักบ้านเดิม
28             </a>
29         </div>
30         <!-- ค้นหาน้ำหน้า -->
31         <div class="mb-4">
32             <div class="flex space-x-4 items-center">
33                 <div class="search-dropdown">
34                     <label for="searchType" class="block text-sm font-medium text-gray-700">ค้นหาจาก</label>
35                     <select id="searchType" class="searchType" value="roomName">
36                         <option value="roomName">ห้อง</option>
37                     </select>
38                 </div>
39             </div>
40         </div>
41     </div>
42 
```

ภาพที่ 24 ภาพ room-list.html

```

src > main > resources > templates > room-list.html
220 <body>
221   <div class="container mx-auto p-4">
237     <!-- សោរពិន្ទាន -->
238     <div class="mb-4">
239       <div class="search-input">
240         <label for="searchInput" class="search-input">សោរពិន្ទាន</label>
241         <input type="text" id="searchInput" class="search-text" placeholder="ស្ថាបន្ទាន់បាន..." />
242       </div>
243       <!-- ចូលពិន្ទាន -->
244       <div class="search-button-container">
245         <button id="searchButton" class="search-button">
246           <i class="fas fa-search h-5 w-5 mr-2"></i> ចូល
247         </button>
248       </div>
249       <!-- ឲ្យមើលការសែនា -->
250       <div class="search-button-container">
251         <button id="clearButton" class="clear-button">
252           <i class="fas fa-times h-5 w-5 mr-2"></i> តាមរាល់ខ្លាត
253         </button>
254       </div>
255     </div>
256   </div>
257   <table class="table">
258     <thead>
259       <tr>
260         <th>ID</th>
261         <th>ជំនាញ</th>
262         <th>ឈ្មោះបន្ទាន់</th>
263         <th>អ្នកបង់បាន</th>
264       </tr>
265     </thead>
266     <tbody id="roomTable">
267       <tr th:each="room : $rooms" class="table-row">
268         <td th:text="#{room.id}"></td>
269         <td th:text="#{room.roomName}"></td>
270         <td th:text="#{room.description}"></td>
271         <td class="action-buttons">
272           <a href="#" class="edit-button">Edit</a>
273           <a href="#" class="delete-button" onclick="return confirmDelete()">Delete</a>
274         </td>
275       </tr>
276     </tbody>
277   </table>
278   <div id="message" th:if="${message}" th:text="${message}" style="color: green;"></div>
279   <div id="error" th:if="${error}" th:text="${error}" style="color: red;"></div>
280 </div>
281 <script>
282   document
283     .getElementsByName("searchType")
284     .addEventListener("click", function () {
285       var searchType = document.getElementsByName("searchType").value; // ប្រភេទការសែនា
286       var searchInput = document
287         .getElementById("searchInput")
288         .value.toLowerCase(); // សំគាល់
289       var tableRows = document.querySelectorAll("#roomTable .table-row"); // លេខខ្លួនការងារ
290
291       tableRows.forEach(function (row) {
292         var cellValue = row
293           .querySelector("td:nth-child(" + getColumnIndex(searchType) + ")")
294             .textContent.toLowerCase(); // គ្រប់គ្រងការសែនាដែលត្រូវការសែនា
295         if (cellValue.includes(searchInput)) {
296           row.style.display = ""; // ដោយការសែនាដែលត្រូវការសែនា
297         } else {
298           row.style.display = "none"; // ដោយការសែនាដែលត្រូវការសែនា
299         }
300       });
301     });
302   </script>
303   // ផ្លូវការការពារការសែនាដែលត្រូវការសែនា
304   function getColumnIndex(type) {
305     switch (type) {
306       case "Room Type": return 1;
307       case "Room Name": return 2;
308       case "Description": return 3;
309       default: return 0;
310     }
311   }
312 </div>
313 </div>
314 </div>
315 </div>
316 </div>
317 </div>
318 </div>

```

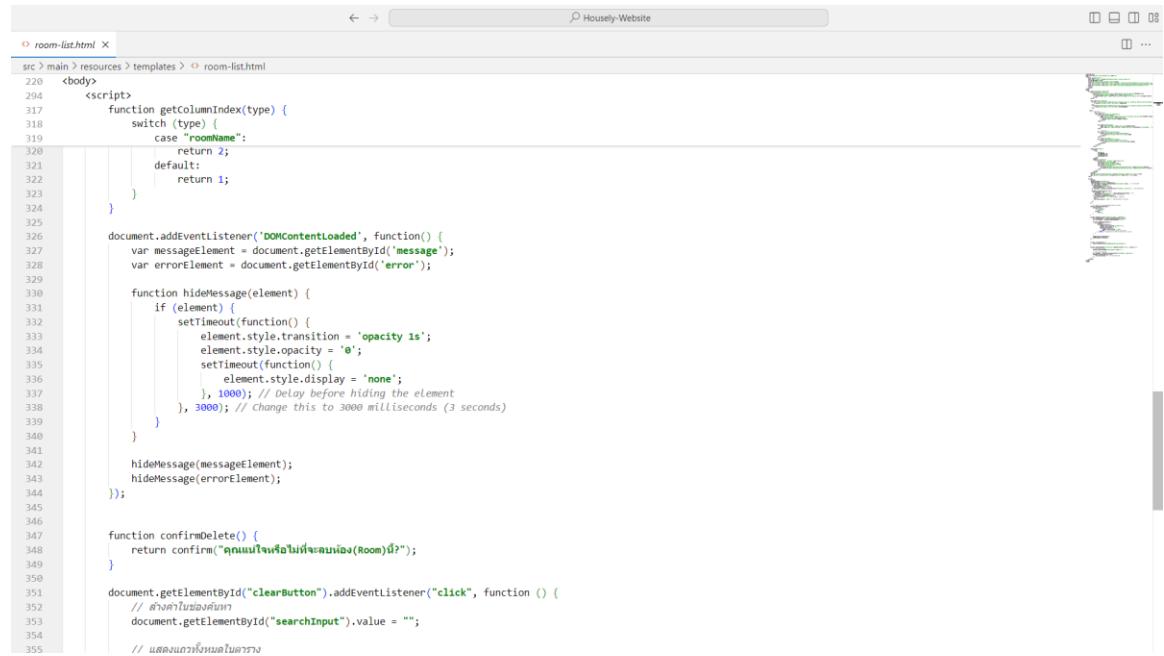
រាយទី 24 រាយ room-list.html (ពីរ)

```

src > main > resources > templates > room-list.html
220 <body>
221   <div class="container mx-auto p-4">
237     <!-- សោរពិន្ទាន -->
238     <div class="mb-4">
239       <div class="search-input">
240         <label for="searchInput" class="search-input">សោរពិន្ទាន</label>
241         <input type="text" id="searchInput" class="search-text" placeholder="ស្ថាបន្ទាន់បាន..." />
242       </div>
243       <!-- ចូលពិន្ទាន -->
244       <div class="search-button-container">
245         <button id="searchButton" class="search-button">
246           <i class="fas fa-search h-5 w-5 mr-2"></i> ចូល
247         </button>
248       </div>
249       <!-- ឲ្យមើលការសែនា -->
250       <div class="search-button-container">
251         <button id="clearButton" class="clear-button">
252           <i class="fas fa-times h-5 w-5 mr-2"></i> តាមរាល់ខ្លាត
253         </button>
254       </div>
255     </div>
256   </div>
257   <table class="table">
258     <tbody id="roomTable">
259       <tr th:each="room : $rooms" class="table-row">
260         <td th:text="#{room.id}"></td>
261         <td th:text="#{room.roomName}"></td>
262         <td th:text="#{room.description}"></td>
263         <td class="action-buttons">
264           <a href="#" class="edit-button">Edit</a>
265           <a href="#" class="delete-button" onclick="return confirmDelete()">Delete</a>
266         </td>
267       </tr>
268     </tbody>
269   </table>
270   <div id="message" th:if="${message}" th:text="${message}" style="color: green;"></div>
271   <div id="error" th:if="${error}" th:text="${error}" style="color: red;"></div>
272 </div>
273 <script>
274   document
275     .getElementsByName("searchType")
276     .addEventListener("click", function () {
277       var searchType = document.getElementsByName("searchType").value; // ប្រភេទការសែនា
278       var searchInput = document
279         .getElementById("searchInput")
280         .value.toLowerCase(); // សំគាល់
281       var tableRows = document.querySelectorAll("#roomTable .table-row"); // លេខខ្លួនការងារ
282
283       tableRows.forEach(function (row) {
284         var cellValue = row
285           .querySelector("td:nth-child(" + getColumnIndex(searchType) + ")")
286             .textContent.toLowerCase(); // គ្រប់គ្រងការសែនាដែលត្រូវការសែនា
287         if (cellValue.includes(searchInput)) {
288           row.style.display = ""; // ដោយការសែនាដែលត្រូវការសែនា
289         } else {
290           row.style.display = "none"; // ដោយការសែនាដែលត្រូវការសែនា
291         }
292       });
293     });
294   </script>
295   // ផ្លូវការការពារការសែនាដែលត្រូវការសែនា
296   function getColumnIndex(type) {
297     switch (type) {
298       case "Room Type": return 1;
299       case "Room Name": return 2;
300       case "Description": return 3;
301       default: return 0;
302     }
303   }
304 </div>
305 </div>
306 </div>
307 </div>
308 </div>
309 </div>
310 </div>
311 </div>
312 </div>
313 </div>
314 </div>
315 </div>
316 </div>
317 </div>
318 </div>

```

រាយទី 24 រាយ room-list.html (ពីរ)

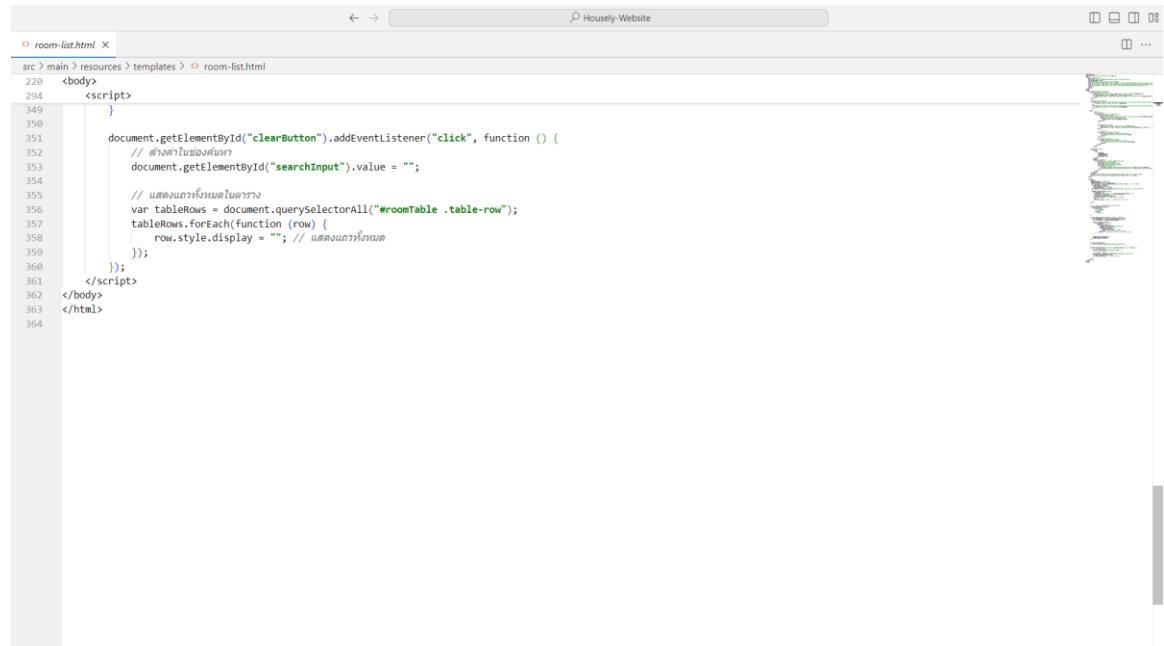


```

src > main > resources > templates > room-list.html
220  <body>
221      <script>
222          function getColumnIndex(type) {
223              switch (type) {
224                  case "roomname":
225                      return 2;
226                  default:
227                      return 1;
228              }
229          }
230
231          document.addEventListener('DOMContentLoaded', function() {
232              var messageElement = document.getElementById('message');
233              var errorElement = document.getElementById('error');
234
235              function hideMessage(element) {
236                  if (element) {
237                      setTimeout(function() {
238                          element.style.opacity = '0';
239                      }, 1000); // Delay before hiding the element
240                      setTimeout(function() {
241                          element.style.display = 'none';
242                      }, 3000); // Change this to 3000 milliseconds (3 seconds)
243                  }
244              }
245
246              hideMessage(messageElement);
247              hideMessage(errorElement);
248          });
249
250
251          function confirmDelete() {
252              return confirm("คุณต้องการลบห้องน้ำ (Room) นี้?");
253          }
254
255          document.getElementById("clearButton").addEventListener("click", function () {
256              // ล้างค่าในช่องค้นหา
257              document.getElementById("searchInput").value = "";
258
259              // และลบรายการทั้งหมดในตาราง
260              var tableRows = document.querySelectorAll("#roomTable .table-row");
261              tableRows.forEach(function (row) {
262                  row.style.display = ""; // และลบรายการ
263              });
264          });
265
266      </script>
267  </body>
268 </html>
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364

```

ภาพที่ 24 ภาพ room-list.html (ต่อ)



```

src > main > resources > templates > room-list.html
220  <body>
221      <script>
222          }
223
224          document.getElementById("clearButton").addEventListener("click", function () {
225              // ล้างค่าในช่องค้นหา
226              document.getElementById("searchInput").value = "";
227
228              // และลบรายการทั้งหมดในตาราง
229              var tableRows = document.querySelectorAll("#roomTable .table-row");
230              tableRows.forEach(function (row) {
231                  row.style.display = ""; // และลบรายการ
232              });
233          });
234
235      </script>
236  </body>
237 </html>
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364

```

ภาพที่ 24 ภาพ room-list.html (ต่อ)

- user-product.html

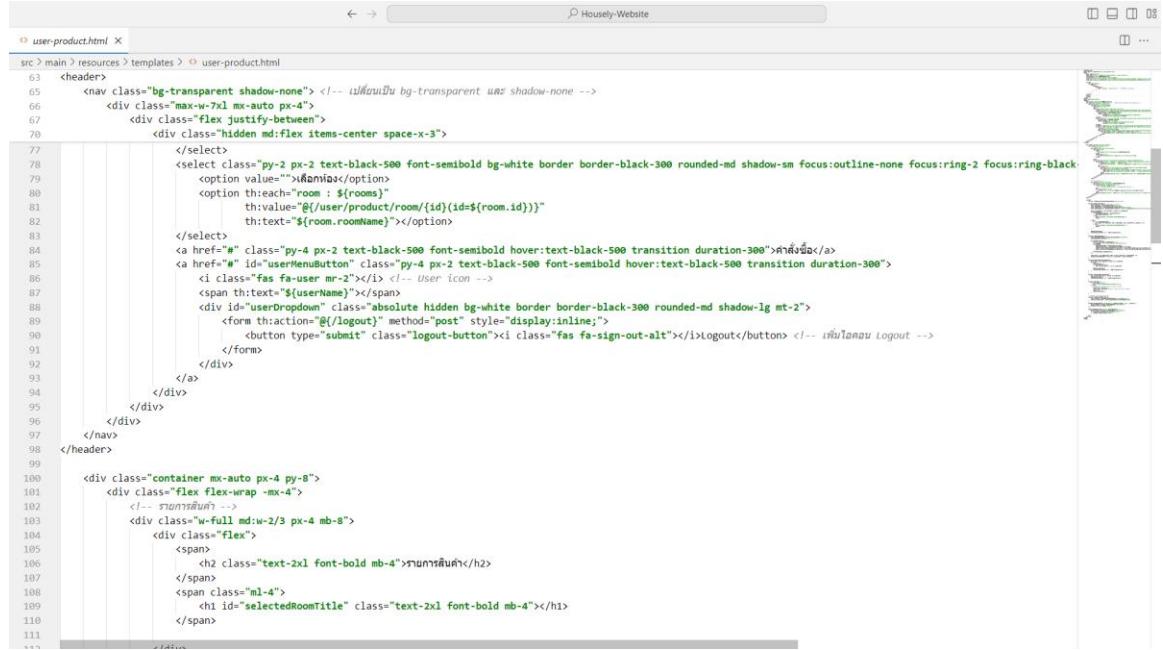
หน้านี้เป็นหน้าหลักของระบบขายหน้าร้าน "Housely" ที่ให้ผู้ใช้เรียกดูสินค้าพร้อมปุ่ม "เพิ่มลงตะกร้า" และจัดการตะกร้าสินค้า รวมถึงเลือกหมวดหมู่และห้อง โดยมีฟังก์ชันการเพิ่ม ลด และลบสินค้าในตะกร้า และปุ่มออกจากระบบเพื่อให้ผู้ใช้สามารถออกจากระบบได้ง่าย ๆ

```

src > main > resources > templates > user-product.html
1  <!DOCTYPE html>
2  <html lang="th" xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Housely - ระบบขายหน้าร้าน</title>
7      <script src="https://cdn.tailwindcss.com"></script>
8      <link href="https://fonts.googleapis.com/css2?family=Aboreto&family=Anuphan:wght@100..700&display=swap" rel="stylesheet">
9      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
10
11     <script>
12         tailwind.config = {
13             theme: {
14                 extend: {
15                     fontFamily: {
16                         sans: ['Anuphan', 'sans-serif'], // မူခံစွမ်းပြီး Anuphan
17                     },
18                 }
19             }
20         }
21     </script>
22     <style>...
23     </style>
24 </head>
25 <body class="bg-gray-100">
26 <header>
27     <h1><a href="/admin/home">HOUSELY</a></h1>
28     <nav class="bg-transparent shadow-none" > <!-- အသေးစိတ် bg-transparent သူ့ shadow-none -->
29         <div class="max-w-7xl mx-auto px-4">
30             <div class="flex justify-between">
31                 <div class="flex space-x-7">
32                     <div>
33                         <div class="hidden md:flex items-center space-x-3">
34                             <a href="/user/product" class="py-4 px-2 text-black-500 font-semibold hover:text-black-500 transition duration-300">ສິນຄ້າ</a>
35                             <select class="py-2 px-2 text-black-500 font-semibold bg-white border border-black-300 rounded-md shadow-sm focus:outline-none focus:ring-black-500 focus:ring-2 focus:ring-black-300">
36                                 <option value="">အລັດວຽກ</option>
37                                 <option value="@{/user/product/category/{id}}(id=${cate.categoryId})">${cate.categoryName}</option>
38                             </select>
39                         <select class="py-2 px-2 text-black-500 font-semibold bg-white border border-black-300 rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-black-300">
40                             <option value="">ຫຼາຍ</option>
41                             <option value="@{/user/product/room/{id}}(id=${room.id})">${room.name}</option>
42                         </select>
43                     </div>
44                 </div>
45             </div>
46         </div>
47     </nav>
48 </body>

```

ภาพที่ 25 ภาพ user-product.html

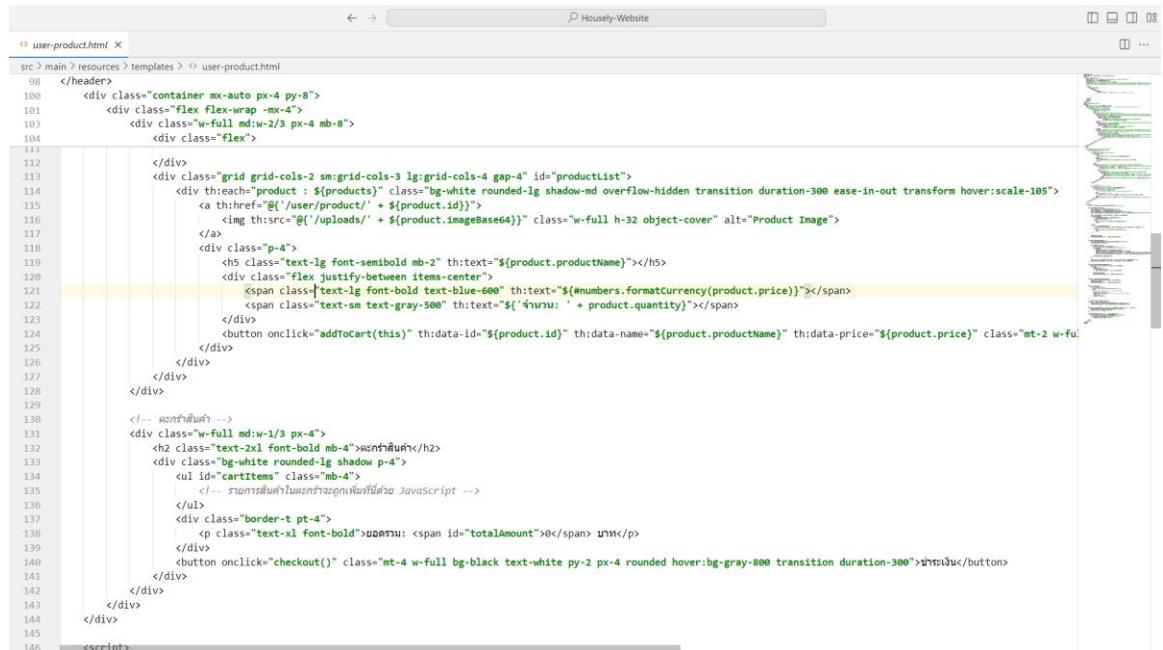


```

src > main > resources > templates > user-product.html
53   <header>
54     <nav class="bg-transparent shadow-none"> <!-- အမှုပါးပို့ bg-transparent user shadow-none -->
55       <div class="max-w-7xl mx-auto px-4">
56         <div class="flex justify-between">
57           <div class="hidden md:flex items-center space-x-3">
58             </select>
59             <select class="py-2 px-2 text-black-500 font-semibold bg-white border border-black-300 rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-black">
60               <option value="">&nbsp;

```

ภาพที่ 25 ภาพ user-product.html (ต่อ)



```

src > main > resources > templates > user-product.html
98   <header>
99     <div class="container mx-auto px-4 py-8">
100       <div class="flex flex-wrap -mx-4">
101         <div class="w-full md:w-2/3 px-4 mb-8">
102           <div class="flex">
103             <div class="grid grid-cols-2 sm:grid-cols-3 lg:grid-cols-4 gap-4" id="productList">
104               <div th:each="product : ${products}" class="bg-white rounded-lg shadow-md overflow-hidden transition duration-300 ease-in-out transform hover:scale-105">
105                 <a href="#" th:href="@{/user/product/ + ${product.id}}">
106                   
107                 </a>
108                 <div class="p-4">
109                   <h5 class="text-lg font-semibold mb-2" th:text="${product.productName}"></h5>
110                   <div class="flex justify-between items-center">
111                     <span class="text-lg font-bold text-blue-500" th:text="#${numbers.formatCurrency(product.price)}"></span>
112                     <span class="text-sm text-gray-500" th:text="${'မြန်း + ' + product.quantity}"></span>
113                   </div>
114                   <button onclick="addToCart(this)" th:data-id="${product.id}" th:data-name="${product.productName}" th:data-price="${product.price}" class="mt-2 w-fu...
115                 </div>
116               </div>
117             </div>
118           </div>
119         </div>
120       </div>
121     </div>
122   </div>
123   </div>
124   </div>
125   </div>
126   </div>
127   </div>
128   </div>
129   </div>
130   </div>
131   </div>
132   </div>
133   </div>
134   </div>
135   </div>
136   </div>
137   </div>
138   </div>
139   </div>
140   </div>
141   </div>
142   </div>
143   </div>
144   </div>
145   </div>
146   <script>

```

ภาพที่ 25 ภาพ user-product.html (ต่อ)

```

src > main > resources > templates > user-product.html
98 </header>
145 <script>
146   let cart = JSON.parse(localStorage.getItem('cart')) || [];
147
148   function addToCart(button) {
149     const productId = button.getAttribute('data-id');
150     const productName = button.getAttribute('data-name');
151     const productPrice = parseFloat(button.getAttribute('data-price'));
152     const productQuantity = parseInt(button.closest('.bg-white').querySelector('span.text-sm.text-gray-500').textContent.split(':')[1]);
153
154     const existingItem = cart.find(item => item.id === productId);
155     if (existingItem) {
156       if (existingItem.quantity < productQuantity) {
157         existingItem.quantity += 1;
158       } else {
159         alert("ເພື່ອນການທີ່ບໍ່ສິ້ນຕ່ານກວດກຳທີ່ມີຂູ້ໃນຄະລຸງ");
160         return;
161       }
162     } else {
163       if (1 <= productQuantity) {
164         cart.push({ id: productId, name: productName, price: productPrice, quantity: 1 });
165       } else {
166         alert("ເພື່ອນການທີ່ບໍ່ສິ້ນຕ່ານກວດກຳທີ່ມີຂູ້ໃນຄະລຸງ");
167         return;
168       }
169     }
170   }
171
172   updateCartDisplay();
173   localStorage.setItem('cart', JSON.stringify(cart));
174 }
175
176 function updateCartDisplay() {
177   const cartItemsElement = document.getElementById('cartItems');
178   cartItemsElement.innerHTML = '';
179
180   cart.forEach(item => {
181     const li = document.createElement('li');
182     li.className = 'flex justify-between items-center mb-2';
183     li.innerHTML =
184       `<span>${item.name} x ${item.quantity}</span>
185       <span>${(item.price * item.quantity).toFixed(2)} ນາມ</span>
186       <button onclick="removeFromCart('${item.id}')">ລົບ</button>
187       <button onclick="decreaseQuantity('${item.id}')">ຈອຍຂັນ</button>
188     `;
189     cartItemsElement.appendChild(li);
190   });
191
192   const total = cart.reduce((sum, item) => sum + item.price * item.quantity, 0);
193   document.getElementById('totalAmount').textContent = total.toFixed(2);
194 }
195
196 function decreaseQuantity(productId) {
197   const item = cart.find(item => item.id === productId);
198   if (item) {
199     item.quantity -= 1;
200     if (item.quantity <= 0) {
201       cart = cart.filter(i => i.id !== productId);
202     }
203     updateCartDisplay();
204     localStorage.setItem('cart', JSON.stringify(cart));
205   }
206 }
207
208 function removeFromCart(productId) {
209   cart = cart.filter(item => item.id !== productId);
210   updateCartDisplay();
211   localStorage.setItem('cart', JSON.stringify(cart));
212 }
213
214 function checkout() {
215   if (cart.length === 0) {
216     alert("ຍຸດຍັນເພີ້ມສິ້ນໃນກະຕົວການນີ້ແລ້ວລົບ");
217   }
218 }

```

ກາພທີ 25 ກາພ user-product.html (ຕ່ອ)

```

src > main > resources > templates > user-product.html
98 </header>
146 <script>
147   function updateCartDisplay() {
148
149     cart.forEach(item => {
150       const li = document.createElement('li');
151       li.className = 'flex justify-between items-center mb-2';
152       li.innerHTML =
153         `<span>${item.name} x ${item.quantity}</span>
154         <span>${(item.price * item.quantity).toFixed(2)} ນາມ</span>
155         <button onclick="removeFromCart('${item.id}')">ລົບ</button>
156         <button onclick="decreaseQuantity('${item.id}')">ຈອຍຂັນ</button>
157       `;
158       cartItemsElement.appendChild(li);
159     });
160
161     const total = cart.reduce((sum, item) => sum + item.price * item.quantity, 0);
162     document.getElementById('totalAmount').textContent = total.toFixed(2);
163   }
164
165   function decreaseQuantity(productId) {
166     const item = cart.find(item => item.id === productId);
167     if (item) {
168       item.quantity -= 1;
169       if (item.quantity <= 0) {
170         cart = cart.filter(i => i.id !== productId);
171       }
172       updateCartDisplay();
173       localStorage.setItem('cart', JSON.stringify(cart));
174     }
175   }
176
177   function removeFromCart(productId) {
178     cart = cart.filter(item => item.id !== productId);
179     updateCartDisplay();
180     localStorage.setItem('cart', JSON.stringify(cart));
181   }
182
183   function checkout() {
184     if (cart.length === 0) {
185       alert("ຍຸດຍັນເພີ້ມສິ້ນໃນກະຕົວການນີ້ແລ້ວລົບ");
186     }
187   }
188 }
189
190 
```

ກາພທີ 25 ກາພ user-product.html (ຕ່ອ)

```

src > main > resources > templates > user-product.html
98  </header>
146  <script>
147    // Checkout function
148    function checkout() {
149      if (cart.length === 0) {
150        alert('ກະຕົນບໍ່ມີຄວາມກຳລັງທີ່ຈ່າຍ');
151      } else {
152        // Redirect to checkout page or process checkout
153        alert('ກະຕົນບໍ່ມີຄວາມກຳລັງທີ່ຈ່າຍ');
154        localStorage.removeItem('cart'); // Clear cart after checkout
155        cart = [];
156        updateCartDisplay();
157      }
158    }
159
160    // ຕິດຕາການເລືອດອົງທຶນທີ່ເລີດ
161    function showSelectedRoom(select) {
162      const roomTitle = document.getElementById('selectedRoomTitle');
163      roomTitle.textContent = select.options[select.selectedIndex].text;
164    }
165
166    document.getElementById('userMenuButton').addEventListener('click', function() {
167      const dropdown = document.getElementById('userDropdown');
168      dropdown.classList.toggle('hidden');
169    });
170
171    window.addEventListener('click', function(event) {
172      const dropdown = document.getElementById('userDropdown');
173      if (!event.target.matches('#userMenuButton')) {
174        dropdown.classList.add('hidden');
175      }
176    });
177  </script>
178 </body>
179 </html>

```

ภาพที่ 25 ภาพ user-product.html (ต่อ)

Controller

- AuthController.java

คลาส AuthController เป็นคอนโทรลเลอร์ในแอปพลิเคชัน Spring ที่จัดการการเข้าสู่ระบบและการลงทะเบียนผู้ใช้ โดยมีเมธอดดังนี้:

login(): แสดงหน้าเข้าสู่ระบบ (login.html)

registerForm(): แสดงแบบฟอร์มการลงทะเบียน (register.html)

register(User user, Model model): รับข้อมูลผู้ใช้จากแบบฟอร์มลงทะเบียนและเรียกใช้บริการ userService เพื่อทำการลงทะเบียน จากนั้นเปลี่ยนสันทางไปยังหน้าเข้าสู่ระบบเมื่อการลงทะเบียนเสร็จสมบูรณ์

```

src > main > java > com > cp > kku > housely > controller > J AuthController.java > Language Support for Java(TM) by Red Hat > AuthController > register(User, Model)
1 package com.cp.kku.housely.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PostMapping;
8 import org.springframework.web.bind.annotation.RequestMapping;
9
10 import com.cp.kku.housely.model.User;
11 import com.cp.kku.housely.service.UserService;
12
13 @Controller
14 @RequestMapping("/auth")
15 public class AuthController {
16
17     @Autowired
18     private UserService userService;
19
20     @GetMapping("/login")
21     public String login() {
22         return "login";
23     }
24
25     @GetMapping("/register")
26     public String registerForm() {
27         return "register";
28     }
29
30     @PostMapping("/register")
31     public String register(User user, Model model) {
32         userService.register(user);
33         return "redirect:/auth/login";
34     }
35 }
36

```

ภาพที่ 26 ภาพ AuthController.java

- CategoryController.java

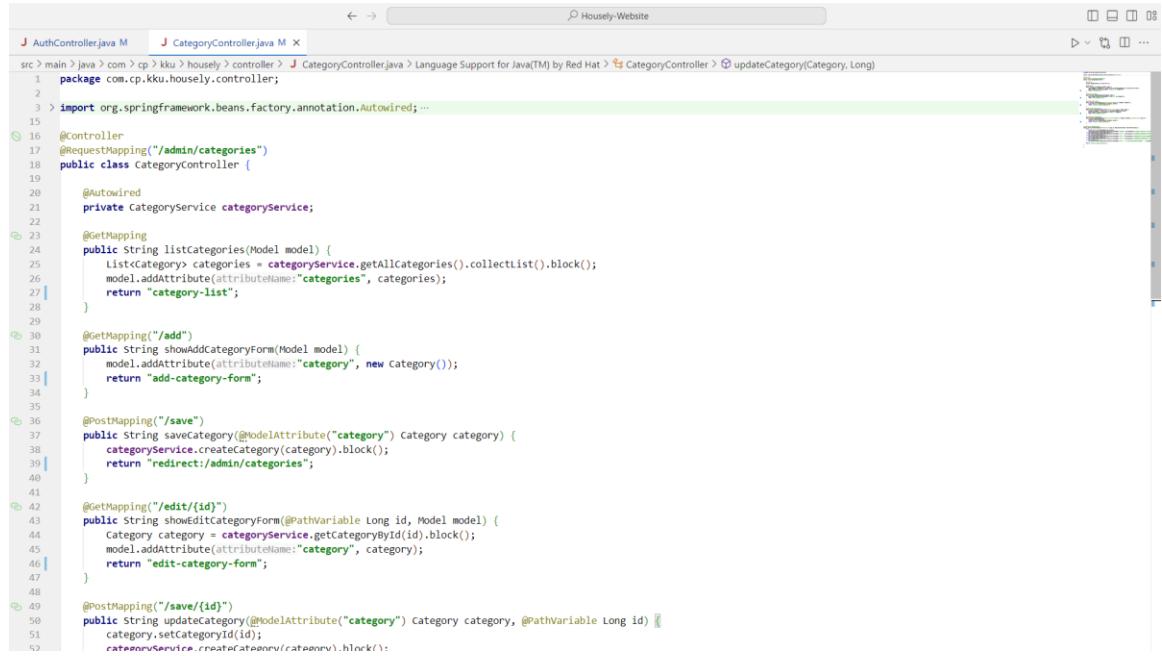
คลาส CategoryController จัดการการทำงานเกี่ยวกับหมวดหมู่ในแอปพลิเคชัน Spring มีเมธอดหลัก ๆ ดังนี้

- listCategories(Model model): แสดงรายการหมวดหมู่ทั้งหมดในแบบฟอร์ม (category-list.html)
- showAddCategoryForm(Model model): แสดงแบบฟอร์มสำหรับเพิ่มหมวดหมู่ใหม่ (add-category-form.html)
- saveCategory(Category category):

บันทึกหมวดหมู่ใหม่ในฐานข้อมูลและเปลี่ยนเส้นทางไปยังรายการหมวดหมู่
- showEditCategoryForm(Long id, Model model): แสดงแบบฟอร์มสำหรับแก้ไขหมวดหมู่ที่เลือก (edit-category-form.html)
- updateCategory(Category category, Long id):

อัปเดตข้อมูลหมวดหมู่และเปลี่ยนเส้นทางไปยังรายการหมวดหมู่
- deleteCategory(Long id, RedirectAttributes redirectAttributes): ลบหมวดหมู่ตาม ID

โดยจัดการข้อผิดพลาดที่อาจเกิดขึ้น เช่น ข้อมูลที่เกี่ยวข้องหรือการไม่อนุญาต
เมื่อต้องลบหมวดหมู่จะใช้ categoryService ในการเรียกใช้งานบริการที่เกี่ยวข้องกับหมวดหมู่

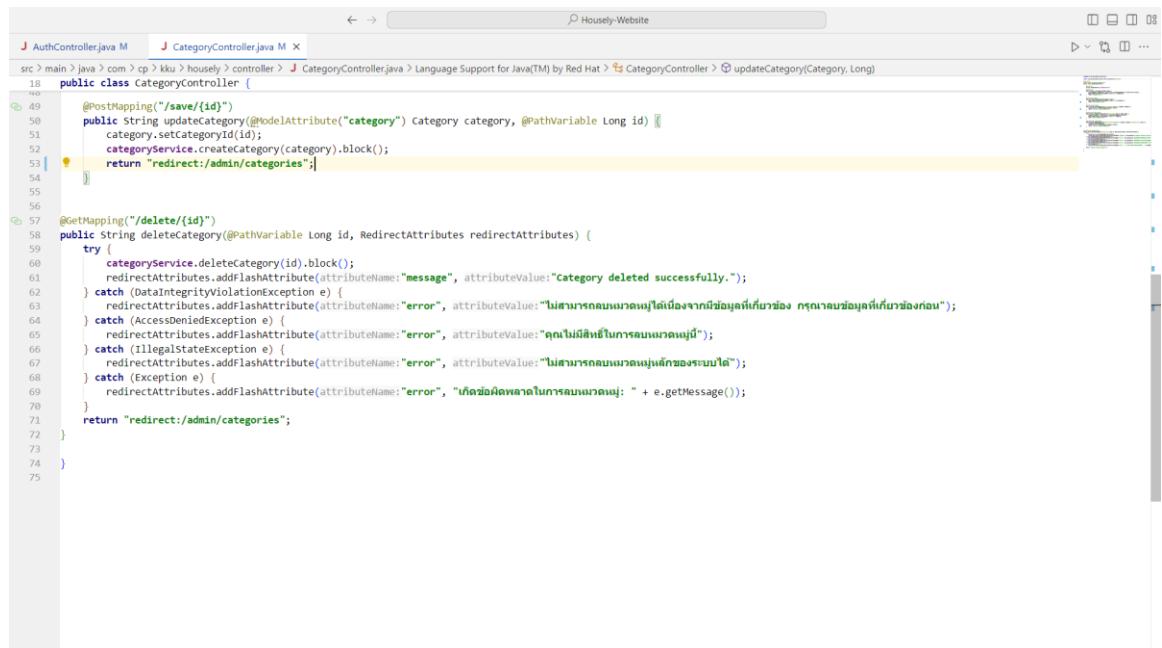


```

J AuthController.java M J CategoryController.java M X
src > main > java > com > cp > kku > housely > controller > J CategoryController.java > Language Support for Java(TM) by Red Hat > CategoryController > updateCategory(Category, Long)
1 package com.cp.kku.housely.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired; ...
4
5 @Controller
6 @RequestMapping("/admin/categories")
7 public class CategoryController {
8
9     @Autowired
10    private CategoryService categoryService;
11
12    @GetMapping
13    public String listCategories(Model model) {
14        List<Category> categories = categoryService.getAllCategories().collectList().block();
15        model.addAttribute("categories", categories);
16        return "category-list";
17    }
18
19    @GetMapping("/add")
20    public String showAddCategoryForm(Model model) {
21        model.addAttribute("category", new Category());
22        return "add-category-form";
23    }
24
25    @PostMapping("/save")
26    public String saveCategory(@ModelAttribute("category") Category category) {
27        categoryService.createCategory(category).block();
28        return "redirect:/admin/categories";
29    }
30
31    @GetMapping("/edit/{id}")
32    public String showEditCategoryForm(@PathVariable Long id, Model model) {
33        Category category = categoryService.getCategoryById(id).block();
34        model.addAttribute("category", category);
35        return "edit-category-form";
36    }
37
38    @PostMapping("/save/{id}")
39    public String updateCategory(@ModelAttribute("category") Category category, @PathVariable Long id) {
40        category.setCategoryId(id);
41        categoryService.createCategory(category).block();
42        return "redirect:/admin/categories";
43    }
44
45    @GetMapping("/delete/{id}")
46    public String deleteCategory(@PathVariable Long id, RedirectAttributes redirectAttributes) {
47        try {
48            categoryService.deleteCategory(id).block();
49            redirectAttributes.addFlashAttribute("message", "Category deleted successfully.");
50        } catch (DataIntegrityViolationException e) {
51            redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบรายการได้เนื่องจากมีข้อมูลที่เกี่ยวข้อง กรุณารายงานข้อมูลที่เกี่ยวข้องก่อน");
52        } catch (AccessDeniedException e) {
53            redirectAttributes.addFlashAttribute("error", "คุณไม่มีสิทธิ์ในการลบรายการนี้");
54        } catch (IllegalStateException e) {
55            redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบรายการนี้ได้");
56        } catch (Exception e) {
57            redirectAttributes.addFlashAttribute("error", "เกิดข้อผิดพลาดในการลบรายการ: " + e.getMessage());
58        }
59    }
60    return "redirect:/admin/categories";
61 }
62
63
64
65
66
67
68
69
70
71
72
73
74
75

```

ภาพที่ 26 ภาพ CategoryController.java



```

J AuthController.java M J CategoryController.java M X
src > main > java > com > cp > kku > housely > controller > J CategoryController.java > Language Support for Java(TM) by Red Hat > CategoryController > updateCategory(Category, Long)
1 package com.cp.kku.housely.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired; ...
4
5 @Controller
6 @RequestMapping("/admin/categories")
7 public class CategoryController {
8
9     @Autowired
10    private CategoryService categoryService;
11
12    @GetMapping("/add")
13    public String showAddCategoryForm(Model model) {
14        model.addAttribute("category", new Category());
15        return "add-category-form";
16    }
17
18    @PostMapping("/save/{id}")
19    public String updateCategory(@ModelAttribute("category") Category category, @PathVariable Long id) {
20        category.setCategoryId(id);
21        categoryService.createCategory(category).block();
22        return "redirect:/admin/categories";
23    }
24
25    @GetMapping("/edit/{id}")
26    public String showEditCategoryForm(@PathVariable Long id, Model model) {
27        Category category = categoryService.getCategoryById(id).block();
28        model.addAttribute("category", category);
29        return "edit-category-form";
30    }
31
32    @PostMapping("/save/{id}")
33    public String updateCategory(@ModelAttribute("category") Category category, @PathVariable Long id) {
34        category.setCategoryId(id);
35        categoryService.createCategory(category).block();
36        return "redirect:/admin/categories";
37    }
38
39    @GetMapping("/delete/{id}")
40    public String deleteCategory(@PathVariable Long id, RedirectAttributes redirectAttributes) {
41        try {
42            categoryService.deleteCategory(id).block();
43            redirectAttributes.addFlashAttribute("message", "Category deleted successfully.");
44        } catch (DataIntegrityViolationException e) {
45            redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบรายการได้เนื่องจากมีข้อมูลที่เกี่ยวข้อง กรุณารายงานข้อมูลที่เกี่ยวข้องก่อน");
46        } catch (AccessDeniedException e) {
47            redirectAttributes.addFlashAttribute("error", "คุณไม่มีสิทธิ์ในการลบรายการนี้");
48        } catch (IllegalStateException e) {
49            redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบรายการนี้ได้");
50        } catch (Exception e) {
51            redirectAttributes.addFlashAttribute("error", "เกิดข้อผิดพลาดในการลบรายการ: " + e.getMessage());
52        }
53    }
54    return "redirect:/admin/categories";
55 }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75

```

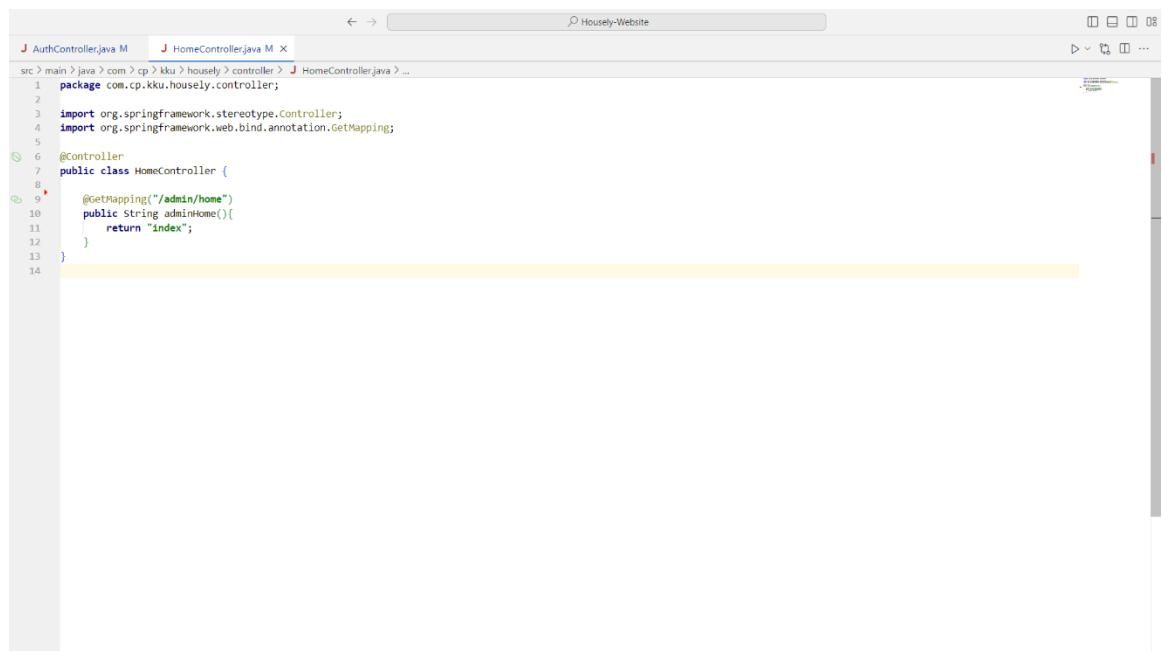
ภาพที่ 26 ภาพ CategoryController.java (ต่อ)

- HomeController.java

คลาส HomeController มีหน้าที่หลักในการจัดการสั่นทางสำหรับหน้าแรกของแอปพลิเคชัน โดยมีเมธอดหลัก ๆ ดังนี้

- **home():** เมื่อมีการเข้าถึงเส้นทาง /home จะคืนค่าชื่อไฟล์ home.html ซึ่งใช้แสดงหน้าแรกของแอปฯทั่วไป
- **adminHome():** เมื่อมีการเข้าถึงเส้นทาง /admin/home จะคืนค่าชื่อไฟล์ index ซึ่งใช้แสดงหน้าแรกสำหรับผู้ดูแลระบบ

ทั้งสองเมธอดนี้ถูกใช้เพื่อแสดงผล HTML ตามเส้นทางที่กำหนดในแอปพลิเคชัน Spring



```

J AuthController.java M J HomeController.java M ...
src > main > java > com > cp > kku > housely > controller > J HomeController.java > ...
1 package com.cp.kku.housely.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @Controller
7 public class HomeController {
8
9     @GetMapping("/admin/home")
10    public String adminHome(){
11        return "index";
12    }
13 }
14

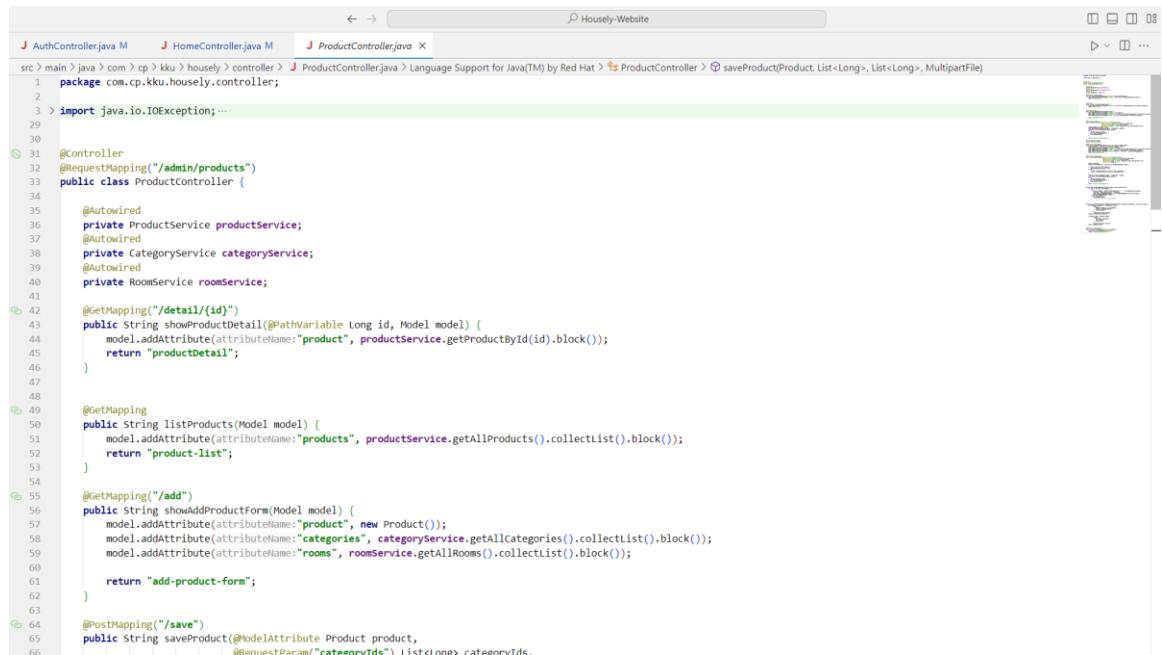
```

ภาพที่ 27 ภาพ HomeController.java

- ProductController.java

- 1. showProductDetail: แสดงรายละเอียดของผลิตภัณฑ์ตาม id ที่ระบุ
- 2. listProducts: แสดงรายการผลิตภัณฑ์ทั้งหมด
- 3. showAddProductForm: แสดงฟอร์มสำหรับเพิ่มผลิตภัณฑ์ใหม่ โดยโหลดหมวดหมู่และห้องที่มีอยู่
- 4. saveProduct: บันทึกผลิตภัณฑ์ใหม่ โดยจัดการการอัปโหลดรูปภาพและกำหนดหมวดหมู่และห้อง
- 5. showEditProductForm: แสดงฟอร์มแก้ไขผลิตภัณฑ์ตาม id ที่ระบุ

- 6. updateProduct: อัปเดตผลิตภัณฑ์ที่มีอยู่ โดยสามารถรักษาภาพเดิมได้หากไม่มีการอัปโหลดภาพใหม่
 - 7. handleImageUpload: จัดการการอัปโหลดภาพและบันทึกเส้นทางของภาพในผลิตภัณฑ์
 - 8. setProductCategoriesAndRooms: ตั้งค่าหมวดหมู่และห้องให้กับผลิตภัณฑ์
 - 9. deleteProduct: ลบผลิตภัณฑ์ตาม id ที่ระบุ
- คลาสนี้มีการใช้งานบริการ ProductService, CategoryService, และ RoomService เพื่อเข้าถึงข้อมูลเกี่ยวกับผลิตภัณฑ์ หมวดหมู่ และห้องในระบบ



```

src > main > java > cp > kku > hously > controller > J ProductController.java > Language Support for Java(TM) by Red Hat > saveProduct(Product, List<Long>, MultipartFile)
1 package com.cp.kku.hously.controller;
2
3 > import java.io.IOException;
4
5
6 @Controller
7 @RequestMapping("/admin/products")
8 public class ProductController {
9
10
11     @Autowired
12     private ProductService productService;
13
14     @Autowired
15     private CategoryService categoryService;
16
17     @Autowired
18     private RoomService roomService;
19
20
21     @GetMapping("/detail/{id}")
22     public String showProductDetail(@PathVariable Long id, Model model) {
23         model.addAttribute("product", productService.getProductById(id).block());
24         return "productDetail";
25     }
26
27
28     @GetMapping
29     public String listProducts(Model model) {
30         model.addAttribute("products", productService.getAllProducts().collectList().block());
31         return "product-list";
32     }
33
34     @GetMapping("/add")
35     public String showAddProductForm(Model model) {
36         model.addAttribute("product", new Product());
37         model.addAttribute("categories", categoryService.getAllCategories().collectList().block());
38         model.addAttribute("rooms", roomService.getAllRooms().collectList().block());
39
40         return "add-product-form";
41     }
42
43
44     @PostMapping("/save")
45     public String saveProduct(@ModelAttribute Product product,
46                               @RequestParam("categoryIds") List<Long> categoryIds,
47                               @RequestParam("roomIds") List<Long> roomIds);
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

ภาพที่ 28 ไฟล์ ProductController.java

```

33  public class ProductController {
34
35      ...
36
37      @PostMapping("/save")
38      public String saveProduct(@ModelAttribute Product product,
39          @RequestParam("categoryIds") List<Long> categoryIds,
40          @RequestParam("roomIds") List<Long> roomIds,
41          @RequestParam(value = "image", required = false) MultipartFile file) {
42
43          handleImageUpload(product, file);
44          setProductCategoriesAndRooms(product, categoryIds, roomIds);
45          productService.createProduct(product).block();
46
47          try {
48              Thread.sleep(500);
49          } catch (InterruptedException e) {
50              e.printStackTrace();
51          }
52
53          return "redirect:/admin/products";
54      }
55
56      @Value("${upload.path}")
57      private String uploadPath;
58
59      @GetMapping("/edit/{id}")
60      public String showEditProductForm(@PathVariable Long id, Model model) {
61          Product product = productService.getProductById(id).block();
62          model.addAttribute("product", product);
63          model.addAttribute("attributeName": "categories", categoryService.getAllCategories().collectList().block());
64          model.addAttribute("attributeName": "rooms", roomService.getAllRooms().collectList().block());
65          model.addAttribute("attributeName": "currentImagePath", "/uploads/" + product.getImageBase64());
66
67          return "edit-product-form";
68      }
69
70      @PostMapping("/update/{id}")
71      public String updateProduct(@ModelAttribute Product product,
72          @RequestParam("categoryIds") List<Long> categoryIds,
73          @RequestParam("roomIds") List<Long> roomIds,
74          @RequestParam(value = "image", required = false) MultipartFile file,
75          @PathVariable Long id) {
76
77          product.setId(id);
78          Product existingProduct = productService.getProductById(id).block();
79
80          ...
81
82          if (file != null && !file.isEmpty()) {
83              handleImageUpload(product, file);
84          } else {
85              // keep the existing image if no new image is uploaded
86              product.setImageBase64(existingProduct.getImageBase64());
87          }
88
89          setProductCategoriesAndRooms(product, categoryIds, roomIds);
90          productService.createProduct(product).block();
91
92          try {
93              Thread.sleep(500);
94          } catch (InterruptedException e) {
95              e.printStackTrace();
96          }
97
98          return "redirect:/admin/products";
99      }
100
101
102      private void handleImageUpload(Product product, MultipartFile file) {
103          if (file != null && !file.isEmpty()) {
104              try {
105                  String fileName = System.currentTimeMillis() + "_" + file.getOriginalFilename();
106                  Path path = Paths.get(uploadPath + fileName);
107                  Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE_EXISTING);
108                  product.setImageBase64(fileName);
109              } catch (IOException e) {
110                  e.printStackTrace();
111                  // Consider proper error handling here
112              }
113          }
114      }
115
116      private void setProductCategoriesAndRooms(Product product, List<Long> categoryIds, List<Long> roomIds) {
117          List<Category> categories = categoryIds.stream()
118              .map(id -> {
119                  Category category = new Category();
120                  category.setCategoryId(id);
121                  return category;
122              })
123      }
124
125      ...
126
127      ...
128
129
130
131
132
133
134
135
136
137
138
139

```

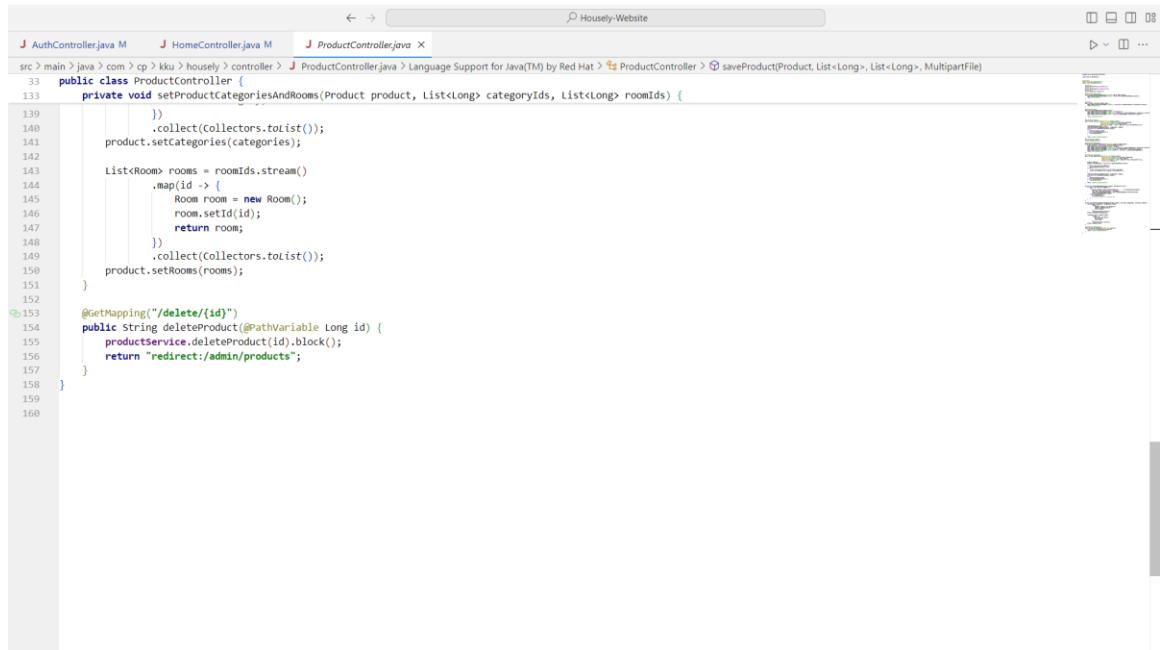
ภาพที่ 28 ภาพ ProductController.java (ต่อ)

```

33  public class ProductController {
34
35      ...
36
37      @PostMapping("/update/{id}")
38      public String updateProduct(@ModelAttribute Product product,
39          @RequestParam("categoryIds") List<Long> categoryIds,
40          @RequestParam("roomIds") List<Long> roomIds,
41          @RequestParam(value = "image", required = false) MultipartFile file,
42          @PathVariable Long id) {
43
44          product.setId(id);
45          Product existingProduct = productService.getProductById(id).block();
46
47          if (file != null && !file.isEmpty()) {
48              handleImageUpload(product, file);
49          } else {
50              // keep the existing image if no new image is uploaded
51              product.setImageBase64(existingProduct.getImageBase64());
52          }
53
54          setProductCategoriesAndRooms(product, categoryIds, roomIds);
55          productService.createProduct(product).block();
56
57          try {
58              Thread.sleep(500);
59          } catch (InterruptedException e) {
60              e.printStackTrace();
61          }
62
63          return "redirect:/admin/products";
64      }
65
66
67      private void handleImageUpload(Product product, MultipartFile file) {
68          if (file != null && !file.isEmpty()) {
69              try {
70                  String fileName = System.currentTimeMillis() + "_" + file.getOriginalFilename();
71                  Path path = Paths.get(uploadPath + fileName);
72                  Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE_EXISTING);
73                  product.setImageBase64(fileName);
74              } catch (IOException e) {
75                  e.printStackTrace();
76                  // Consider proper error handling here
77              }
78          }
79      }
80
81      private void setProductCategoriesAndRooms(Product product, List<Long> categoryIds, List<Long> roomIds) {
82          List<Category> categories = categoryIds.stream()
83              .map(id -> {
84                  Category category = new Category();
85                  category.setCategoryId(id);
86                  return category;
87              })
88      }
89
90      ...
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139

```

ภาพที่ 28 ภาพ ProductController.java (ต่อ)



```

J AuthController.java M J HomeController.java M J ProductController.java X
src > main > java > com > cp > kku > housely > controller > J ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > saveProduct(Product, List<Long>, List<Long>, MultipartFile)
33 public class ProductController {
34     private void setProductCategoriesAndRooms(Product product, List<Long> categoryIds, List<Long> roomIds) {
35         categoryIds.stream()
36             .collect(Collectors.toList());
37         product.setCategories(categoryIds);
38
39         List<Room> rooms = roomIds.stream()
40             .map(id -> {
41                 Room room = new Room();
42                 room.setId(id);
43                 return room;
44             })
45             .collect(Collectors.toList());
46         product.setRooms(rooms);
47     }
48
49     @GetMapping("/delete/{id}")
50     public String deleteProduct(@PathVariable Long id) {
51         productService.deleteProduct(id).block();
52         return "redirect:/admin/products";
53     }
54
55 }
56
57
58
59
60

```

ภาพที่ 28 ภาพ ProductController.java (ต่อ)

- RoomController.java

listRooms: แสดงรายการห้องทั้งหมดในระบบ โดยดึงข้อมูลจาก RoomService และส่งไปยังเทมเพลต room-list เพื่อแสดง

showAddRoomForm: แสดงฟอร์มสำหรับเพิ่มห้องใหม่ โดยเตรียมออบเจกต์ Room เป็นสำหรับการกรอกข้อมูล

saveRoom (POST): บันทึกห้องใหม่หรืออัปเดตห้องที่มีอยู่ โดยเรียกใช้บริการ createRoom จาก RoomService และเปลี่ยนเส้นทางกลับไปยังหน้ารายการห้อง

showEditRoomForm: แสดงฟอร์มแก้ไขห้องตาม id ที่ระบุ โดยดึงข้อมูลห้องที่ต้องการแก้ไขจาก RoomService และส่งไปยังเทมเพลต edit-room-form

deleteRoom: ลบห้องตาม id ที่ระบุ โดยใช้ RedirectAttributes เพื่อส่งข้อความแจ้งผลการลบห้องไปยังหน้าแสดงรายการห้อง หากมีข้อผิดพลาดระหว่างการลบ เช่น ข้อมูลที่เกี่ยวข้องอยู่ในระบบ จะมีการจัดการข้อผิดพลาด และแสดงข้อความที่เหมาะสม

คลาสนี้ทำงานร่วมกับ RoomService เพื่อจัดการข้อมูลห้องในฐานข้อมูล และใช้เทมเพลต HTML ที่เหมาะสมในการแสดงผลให้ผู้ใช้เห็น

```

1 package com.cp.kku.housely.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired; ...
4
5
6 @Controller
7 @RequestMapping("/admin/rooms")
8 public class RoomController {
9
10     @Autowired
11     private RoomService roomService;
12
13     @GetMapping
14     public String listRooms(Model model) {
15         List<Room> rooms = roomService.getAllRooms().collectList().block();
16         model.addAttribute("rooms", rooms);
17         return "room-list";
18     }
19
20     @GetMapping("/add")
21     public String showAddRoomForm(Model model) {
22         model.addAttribute("room", new Room());
23         return "add-room-form";
24     }
25
26     @PostMapping("/save")
27     public String saveRoom(@ModelAttribute("room") Room room) {
28         System.out.println("here");
29         roomService.createRoom(room).block();
30         return "redirect:/admin/rooms";
31     }
32
33     @GetMapping("/edit/{id}")
34     public String showEditRoomForm(@PathVariable Long id, Model model) {
35         Room room = roomService.getRoomById(id).block();
36         model.addAttribute("room", room);
37         return "edit-room-form";
38     }
39
40     @PostMapping("/save/{id}")
41     public String saveRoom(@ModelAttribute("room") Room room, @PathVariable Long id) {
42         room.setId(id);
43         roomService.createRoom(room).block();
44         return "redirect:/admin/rooms";
45     }
46
47     @DeleteMapping("/delete/{id}")
48     public String deleteRoom(@PathVariable Long id, RedirectAttributes redirectAttributes) {
49         try {
50             roomService.deleteRoom(id).block();
51             redirectAttributes.addFlashAttribute("message", "Room deleted successfully.");
52         } catch (DataIntegrityViolationException e) {
53             redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบห้องได้เนื่องจากมีผู้บุคคลที่เก็บข้อมูล  กรุณาลบผู้บุคคลก่อน");
54         } catch (AccessDeniedException e) {
55             redirectAttributes.addFlashAttribute("error", "คุณไม่มีสิทธิ์ในการลบห้องนี้");
56         } catch (IllegalStateException e) {
57             redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบห้องได้เนื่องจากระบบไม่ดี");
58         } catch (Exception e) {
59             redirectAttributes.addFlashAttribute("error", "เกิดข้อผิดพลาดในการลบห้อง: " + e.getMessage());
60         }
61         return "redirect:/admin/rooms";
62     }
63 }
64
65
66
67
68
69
70
71
72
73
74

```

ภาพที่ 29 ภาพ RoomController.java

```

18 public class RoomController {
19
20     public String showEditRoomForm(@PathVariable Long id, Model model) {
21         model.addAttribute("room", room);
22         return "edit-room-form";
23     }
24
25     @PostMapping("/save/{id}")
26     public String saveRoom(@ModelAttribute("room") Room room, @PathVariable Long id) {
27         room.setId(id);
28         roomService.createRoom(room).block();
29         return "redirect:/admin/rooms";
30     }
31
32     @DeleteMapping("/delete/{id}")
33     public String deleteRoom(@PathVariable Long id, RedirectAttributes redirectAttributes) {
34         try {
35             roomService.deleteRoom(id).block();
36             redirectAttributes.addFlashAttribute("message", "Room deleted successfully.");
37         } catch (DataIntegrityViolationException e) {
38             redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบห้องได้เนื่องจากมีผู้บุคคลที่เก็บข้อมูล  กรุณาลบผู้บุคคลก่อน");
39         } catch (AccessDeniedException e) {
40             redirectAttributes.addFlashAttribute("error", "คุณไม่มีสิทธิ์ในการลบห้องนี้");
41         } catch (IllegalStateException e) {
42             redirectAttributes.addFlashAttribute("error", "ไม่สามารถลบห้องได้เนื่องจากระบบไม่ดี");
43         } catch (Exception e) {
44             redirectAttributes.addFlashAttribute("error", "เกิดข้อผิดพลาดในการลบห้อง: " + e.getMessage());
45         }
46         return "redirect:/admin/rooms";
47     }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

ภาพที่ 29 ภาพ RoomController.java (ต่อ)

- UserController.java

- 1. showProducts: แสดงรายการผลิตภัณฑ์ทั้งหมด โดยดึงข้อมูลจาก ProductService, CategoryService, และ RoomService เพื่อส่งข้อมูลไปยังเทมเพลต user-product รวมถึงชื่อผู้ใช้ปัจจุบัน
 - 2. getUserId: ตรวจสอบและส่งคืนชื่อผู้ใช้ที่ล็อกอินอยู่ โดยใช้ SecurityContextHolder เพื่อเข้าถึงข้อมูลผู้ใช้
 - 3. getViewProductById: แสดงรายละเอียดผลิตภัณฑ์ตาม id ที่ระบุ โดยดึงข้อมูลผลิตภัณฑ์จาก ProductService และส่งไปยังเทมเพลต productDetailUser
 - 4. getViewProductByCategory: แสดงผลิตภัณฑ์ที่จัดกลุ่มตามหมวดหมู่ โดยดึงผลิตภัณฑ์ที่เกี่ยวข้องจาก ProductService ตาม categoryId ที่ระบุ และส่งข้อมูลไปยังเทมเพลต user-product
 - 5. getViewProductByRoom: แสดงผลิตภัณฑ์ตามห้องที่กำหนด โดยดึงข้อมูลจาก ProductService ตาม roomId ที่ระบุ และส่งข้อมูลไปยังเทมเพลต user-product
 - 6. decreaseProductQuantity: ลดจำนวนผลิตภัณฑ์ตาม id ที่ระบุ โดยใช้ PUT สำหรับการลดจำนวน และส่งคืน ResponseEntity ของผลิตภัณฑ์ที่อัปเดตแล้วหรือข้อผิดพลาดถ้าไม่สำเร็จ
- คลาสนี้ทำงานร่วมกับหลายบริการเพื่อจัดการและแสดงข้อมูลผลิตภัณฑ์ รวมถึงจัดการการเปลี่ยนแปลงจำนวนผลิตภัณฑ์ในระบบ

```

src > main > java > cp > kku > housely > controller > UserController.java > Spring Boot Tools > UserController (@Controller <: @Component) UserController
1 package com.cp.kku.housely.controller;
2
3 > import org.springframework.beans.factory.annotation.Autowired; ...
22
23     @Controller
24     @RequestMapping("/user")
25     public class UserController {
26
27         @Autowired
28         private ProductService productService;
29
30         @Autowired
31         private CategoryService categoryService;
32
33         @Autowired
34         private RoomService roomService;
35
36         @GetMapping("/product")
37         public String showProducts(Model model) {
38             model.addAttribute("products", productService.getAllProducts().collectList().block());
39             model.addAttribute("categorys", categoryService.getAllCategories().collectList().block());
40             model.addAttribute("rooms", roomService.getAllRooms().collectList().block());
41             model.addAttribute("userName", getCurrentUserId());
42             return "user-product";
43         }
44
45         public String getCurrentUserId() {
46             Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
47             if (authentication != null && authentication.getPrincipal() instanceof UserDetails) {
48                 UserDetails userDetails = (UserDetails) authentication.getPrincipal();
49                 return userDetails.getUsername();
50             }
51             return null;
52         }
53
54         @GetMapping("/product/{id}")
55         public String getViewProductById(@PathVariable Long id, Model model) {
56             model.addAttribute("product", productService.getProductById(id).block());
57         }
58
59         @GetMapping("/product/category/{id}")

```

ภาพที่ 30 ภาพ UserController.java

```

 UserController.java M ×
src > main > java > com > cp > kku > housely > controller > UserController.java > Spring Boot Tools > UserController (@Controller <: @Component) UserController
25 public class UserController {
26
27     @GetMapping("/product/{id}")
28     public String getViewProductById(@PathVariable Long id, Model model) {
29         model.addAttribute("product", productService.getProductById(id).block());
30         return "productDetailUser";
31     }
32
33     @GetMapping("/product/category/{id}")
34     public String getViewProductByCategory(@PathVariable Long id, Model model) {
35         model.addAttribute("products", productService.getProductsByCategoryId(id).collectList().block());
36         model.addAttribute("categorys", categoryService.getAllCategories().collectList().block());
37         model.addAttribute("rooms", roomService.getAllRooms().collectList().block());
38
39         return "user-product";
40     }
41
42     @GetMapping("/product/room/{id}")
43     public String getViewProductByRoom(@PathVariable Long id, Model model) {
44         model.addAttribute("products", productService.getProductsByRoomId(id).collectList().block());
45         model.addAttribute("categorys", categoryService.getAllCategories().collectList().block());
46         model.addAttribute("rooms", roomService.getAllRooms().collectList().block());
47
48         return "user-product";
49     }
50
51     @PutMapping("/decreaseQuantity/{id}")
52     public Mono

```

ภาพที่ 30 ภาพ UserController.java (ต่อ)

Configuration

- SecurityConfig.java

คลาส SecurityConfig กำหนดการตั้งค่าความปลอดภัยของแอปพลิเคชัน โดย:

- กำหนดหน้าและไฟล์ที่เข้าถึงได้โดยไม่ต้องล็อกอิน
- จำกัดการเข้าถึงหน้า /admin/** เฉพาะผู้ใช้ที่มีบทบาท "ADMIN"
- ใช้ BCryptPasswordEncoder เพื่อเข้ารหัสรหัสผ่าน
- กำหนดการเปลี่ยนเส้นทางหลังจากล็อกอินสำเร็จไปยังหน้าที่เหมาะสมตามบทบาท (ADMIN หรือ USER)

```

src > main > java > com > cp > kku > housely > configuration > J SecurityConfig.java > Language Support for Java(TM) by Red Hat > SecurityConfig > passwordEncoder()
1 package com.cp.kku.housely.configuration;
2
3 > import org.springframework.context.annotation.Bean; ...
4
5 @Configuration
6 @EnableWebSecurity
7 public class SecurityConfig {
8
9     @Bean
10    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
11        http
12            .authorizeHttpRequests(auth -> auth
13                .requestMatchers(..patterns:"/", "/home", "/auth/register", "/auth/login").permitAll()
14                .requestMatchers(..patterns:"/static/**", "/uploads/**", "/css/**", "/js/**").permitAll()
15                .requestMatchers(..patterns:"/admin/**").hasRole(role:"ADMIN")
16                .anyRequest().authenticated()
17            )
18            .formLogin(form -> form
19                .loginPage(loginPage:"/auth/login")
20                .successHandler(customSuccessHandler()) // If custom Success Handler
21                .permitAll()
22            )
23            .logout(logout -> logout
24                .logoutSuccessUrl(logoutSuccessUrl:"/auth/login?logout")
25                .permitAll()
26            );
27
28        return http.build();
29    }
30
31    @Bean
32    public PasswordEncoder passwordEncoder() {
33        return new BCryptPasswordEncoder();
34    }
35
36    @Bean
37    public AuthenticationSuccessHandler customSuccessHandler() {
38        return (HttpServletRequest request, HttpServletResponse response, Authentication authentication) -> {
39            if (authentication.getAuthorities().stream()
40                .anyMatch(grantedAuthority -> grantedAuthority.getAuthority().equals(anObject:"ROLE_ADMIN")))
41            {
42                response.sendRedirect(location:"/admin/home");
43            }
44            else {
45                response.sendRedirect(location:"/user/product");
46            }
47        };
48    }
49
50
51
52
53
54
55
56
57
58
59
60
}

```

ภาพที่ 31 ภาพ SecurityConfig.java

```

src > main > java > com > cp > kku > housely > configuration > J SecurityConfig.java > ...
18 public class SecurityConfig {
19
20     @Bean
21     public AuthenticationSuccessHandler customSuccessHandler() {
22         return (HttpServletRequest request, HttpServletResponse response, Authentication authentication) -> {
23             if (authentication.getAuthorities().stream()
24                 .anyMatch(grantedAuthority -> grantedAuthority.getAuthority().equals(anObject:"ROLE_ADMIN")))
25             {
26                 response.sendRedirect(location:"/admin/home");
27             }
28             else {
29                 response.sendRedirect(location:"/user/product");
30             }
31         };
32     }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
}

```

ภาพที่ 31 ภาพ SecurityConfig.java (ต่อ)

Exception

- UserNotFoundException.java

- ขยายจาก RuntimeException: คลาสที่สืบทอดคุณสมบัติจาก RuntimeException ทำให้สามารถใช้ในสถานการณ์ที่ไม่คาดคิดได้
- อนุญาตให้ตั้งสถานะ HTTP: ใช้คำ annotations @ResponseStatus(HttpStatus.NOT_FOUND) เพื่อกำหนดให้มีอเกิดข้อผิดพลาดนี้ จะตอบกลับด้วยรหัสสถานะ HTTP 404 (ไม่พบ)
- มีคอนสตรัคเตอร์: คอนสตรัคเตอร์ที่รับข้อความข้อผิดพลาด ซึ่งจะถูกส่งไปยังคอนสตรัคเตอร์ของ RuntimeException

```

UserNotFoundException.java
src > main > java > com > cp > kku > housely > exception > J UserNotFoundException.java > Language Support for Java(TM) by Red Hat > com.cp.kku.housely.exception
1 package com.cp.kku.housely.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 @ResponseStatus(HttpStatus.NOT_FOUND)
7 public class UserNotFoundException extends RuntimeException {
8     public UserNotFoundException(String message) {
9         super(message);
10    }
11 }
12

```

ภาพที่ 32 ภาพ UserNotFoundException.java

Loader

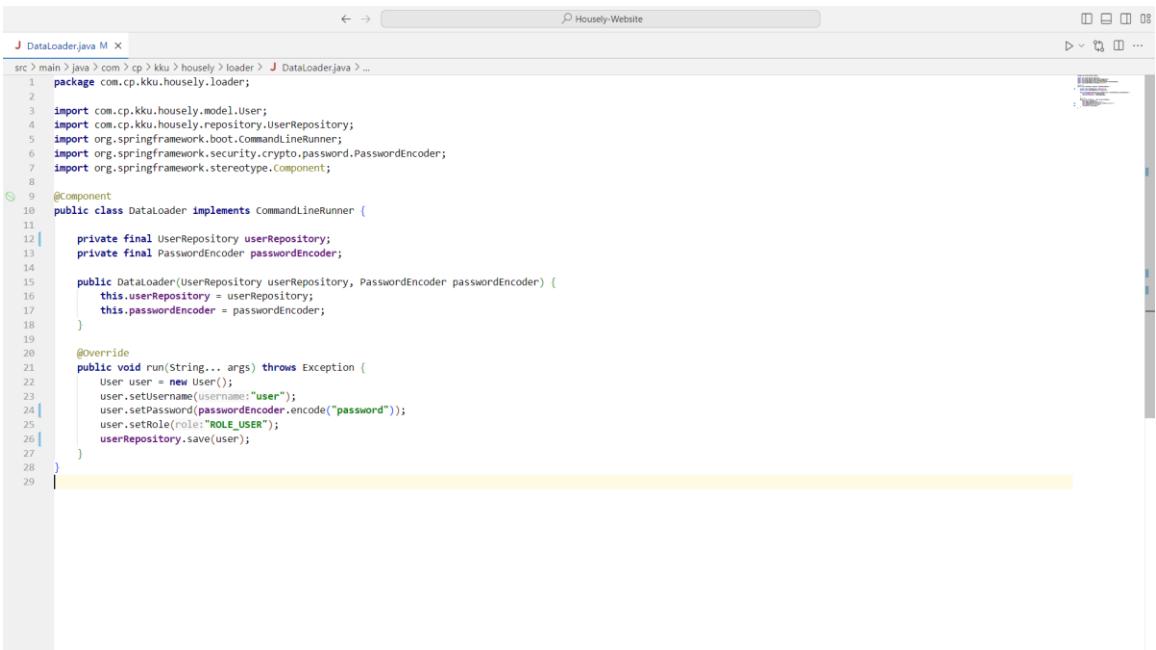
- DataLoader.java

คลาส DataLoader โหลดข้อมูลเริ่มต้นเข้าสู่ฐานข้อมูลเมื่อแอปพลิเคชันเริ่มต้น โดย

- implements CommandLineRunner: ทำให้รันเมื่อแอปพลิเคชันเริ่ม
- สร้างผู้ใช้: สร้าง User และเข้ารหัสผ่าน

- บันทึก: ใช้ UserRepository เพื่อบันทึกผู้ใช้ในฐานข้อมูล

ช่วยเพิ่มผู้ใช้เริ่มต้นอัตโนมัติเมื่อเริ่มแอปพลิเคชัน



```

src > main > java > com > cp > kku > housely > loader > J DataLoader.java > ...
1 package com.cp.kku.housely.loader;
2
3 import com.cp.kku.housely.model.User;
4 import com.cp.kku.housely.repository.UserRepository;
5 import org.springframework.boot.CommandLineRunner;
6 import org.springframework.security.crypto.password.PasswordEncoder;
7 import org.springframework.stereotype.Component;
8
9 @Component
10 public class DataLoader implements CommandLineRunner {
11     private final UserRepository userRepository;
12     private final PasswordEncoder passwordEncoder;
13
14     public DataLoader(UserRepository userRepository, PasswordEncoder passwordEncoder) {
15         this.userRepository = userRepository;
16         this.passwordEncoder = passwordEncoder;
17     }
18
19     @Override
20     public void run(String... args) throws Exception {
21         User user = new User();
22         user.setUsername("user");
23         user.setPassword(passwordEncoder.encode("password"));
24         user.setRole(role:"ROLE_USER");
25         userRepository.save(user);
26     }
27 }
28
29

```

ภาพที่ 33 ภาพ DataLoader.java

Model

- Category.java

- **@Data:** เป็น annotation จาก Lombok ที่จะสร้าง getter, setter, toString(), equals(), และ hashCode() โดยอัตโนมัติสำหรับทุก field ในคลาสนี้
- **Field categoryId:** เป็นตัวแปรประเภท Long ใช้เก็บหมายเลข ID ของหมวดหมู่
- **Field categoryName:** เป็นตัวแปรประเภท String ใช้เก็บชื่อของหมวดหมู่
- **Field description:** เป็นตัวแปรประเภท String ใช้เก็บรายละเอียดของหมวดหมู่
- **Field imageBase64:** เป็นตัวแปรประเภท String ใช้เก็บภาพในรูปแบบ Base64 (ซึ่งคุณสามารถเปลี่ยนรูปแบบการเก็บได้ตามที่ต้องการ ถ้าต้องการเก็บในรูปแบบไฟล์)
- **Field productsInCategory:** เป็น List<Product> ใช้เก็บรายการสินค้า (Product) ที่อยู่ในหมวดหมู่นั้น โดยเริ่มต้นเป็น ArrayList ว่าง ๆ

```

J DataLoader.java M J Category.java X
src > main > java > com > cp > kku > housely > model > J Category.java > {} com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import lombok.Data;
7
8 @Data
9 public class Category {
10
11     private Long categoryId;
12     private String categoryName;
13     private String description;
14     private String imageBase64;
15     private List<Product> productsInCategory = new ArrayList<>();
16
17 }
18

```

ภาพที่ 34 ภาพ Category.java

- Product.java

Field id: เป็นตัวแปรประเภท Long ใช้เก็บหมายเลข ID ของสินค้าแต่ละชิ้น

Field productCode: เป็นตัวแปรประเภท String ใช้เก็บรหัสสินค้า (อาจเป็นรหัสเฉพาะหรือ SKU ของสินค้า)

Field brandName: เป็นตัวแปรประเภท String ใช้เก็บชื่อแบรนด์ของสินค้า

Field ProductName: เป็นตัวแปรประเภท String ใช้เก็บชื่อของสินค้า (ชื่อพิเศษนี้ควรเริ่มต้นด้วยตัวพิมพ์เล็ก: productName)

Field price: เป็นตัวแปรประเภท double ใช้เก็บราคาของสินค้า

Field quantity: เป็นตัวแปรประเภท int ใช้เก็บจำนวนสินค้าที่มีในสต็อก

Field imageBase64: เป็นตัวแปรประเภท String ใช้เก็บรูปภาพของสินค้าในรูปแบบ Base64 (คุณสามารถเปลี่ยนรูปแบบการเก็บได้หากต้องการเก็บเป็นไฟล์)

Field description: เป็นตัวแปรประเภท String ใช้เก็บคำอธิบายรายละเอียดของสินค้า

Field categories: เป็น List<Category> ใช้เก็บรายการหมวดหมู่ (Category) ที่สินค้านี้อยู่ในหลายหมวดหมู่ได้

Field rooms: เป็น List<Room> ใช้เก็บรายการห้อง (Room) ที่สินค้านี้เกี่ยวข้องด้วย (เช่น สินค้าอาจถูกใช้ในห้องต่าง ๆ)

```

J DataLoader.java M J Product.java X
src > main > java > com > cp > kku > housely > model > J Product.java > Product
1 package com.cp.kku.housely.model;
2
3
4 import java.util.ArrayList;
5 import java.util.List;
6
7 import lombok.Data;
8
9 @Data
10 public class Product {
11     private Long id;
12     private String productCode;
13     private String brandName;
14     private String productName;
15     private double price;
16     private int quantity;
17     private String imageBase64;
18     private String description;
19     private List<Category> categories = new ArrayList<>();
20     private List<Room> rooms = new ArrayList<>();
21 }
22
23

```

ภาพที่ 35 ภาพ Product.java

- Room.java

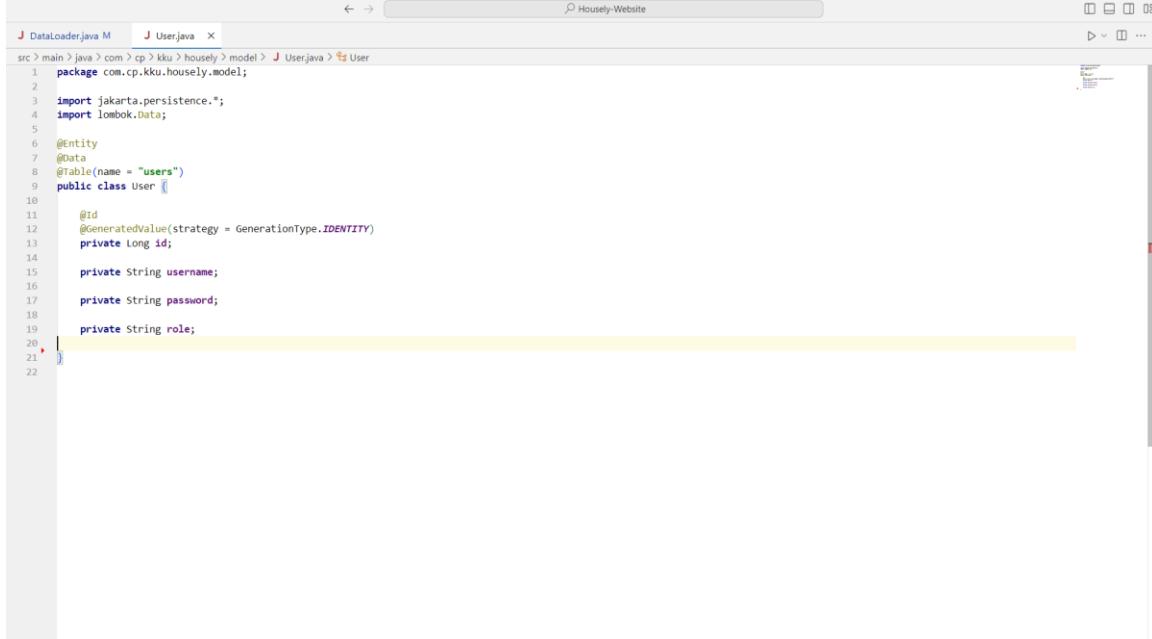
```

J DataLoader.java M J Room.java X
src > main > java > com > cp > kku > housely > model > J Room.java > Room com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import lombok.Data;
7
8 @Data
9 public class Room {
10     private Long id;
11     private String description;
12     private String imageBase64;
13     private String roomName;
14     private List<Product> productsInRoom = new ArrayList<>();
15 }
16
17

```

ภาพที่ 36 ภาพ Room.java

- User.java



```

J DataLoader.java M J User.java X Housely-Website
src > main > java > com > cp > kku > housely > model > J User.java > User
1 package com.cp.kku.housely.model;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5
6 @Entity
7 @Data
8 @Table(name = "users")
9 public class User {
10
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     private String username;
16
17     private String password;
18
19     private String role;
20
21 }
22

```

ภาพที่ 37 ภาพ User.java

Repository

- UserRepository.java

คลาส UserRepository เป็น Repository สำหรับจัดการข้อมูลของผู้ใช้:

- **extends JpaRepository:** ใช้สำหรับดำเนินการ CRUD กับ Entity User

Method:

- **findByUsername(String username):** ค้นหาผู้ใช้ตามชื่อผู้ใช้ (username)
และส่งกลับผลลัพธ์ในรูปแบบ Optional<User>

```

J DataLoader.java M J User.java M J UserRepository.java X
src > main > java > com > cp > kku > housely > repository > J UserRepository.java > ...
1 package com.cp.kku.housely.repository;
2
3 import com.cp.kku.housely.model.user;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 import java.util.Optional;
7
8 public interface UserRepository extends JpaRepository<User, Long> {
9     Optional<User> findByUsername(String username);
10 }
11

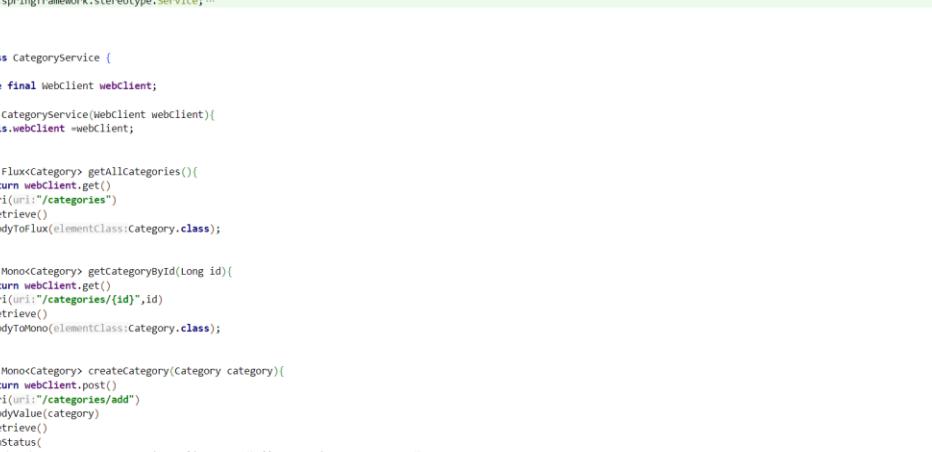
```

ภาพที่ 38 ภาพ UserRepository.java

Service

- CategoryService.java

- getAllCategories(): ดึงรายการหมวดหมู่ทั้งหมด (ส่งคืน Flux<Category>)
- getCategoryById(Long id): ดึงหมวดหมู่ตาม ID (ส่งคืน Mono<Category>)
- createCategory(Category category): สร้างหมวดหมู่ใหม่ (ส่งคืน Mono<Category>)
- updateCategory(Long id, Category category): อัปเดตหมวดหมู่ที่มีอยู่ (ส่งคืน Mono<Category>)
- deleteCategory(Long id): ลบหมวดหมู่ตาม ID (ส่งคืน Mono<Void>)



The screenshot shows a Java IDE interface with the following details:

- Title Bar:** Shows the title "Hously-Website".
- Project Explorer:** On the right side, it displays a tree view of the project structure, including "src", "com", "cp", "kku", "hously", "service", and "CategoryService.java".
- Code Editor:** The main area contains the code for `CategoryService.java`. The code uses `Flux` and `Mono` from the `org.springframework.webflux.core` package.
- Code Snippets:** Several code snippets are visible in the editor, such as `getCategoryById`, `createCategory`, and `updateCategory`.

```
src > main > java > com > cp > kku > hously > service > J CategoryService.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.hously.service
1 package com.cp.kku.hously.service;
2
3 import org.springframework.stereotype.Service;...
4
5
6 @Service
7 public class CategoryService {
8
9     private final WebClient webClient;
10
11     public CategoryService(WebClient webClient){
12         this.webClient = webClient;
13     }
14
15     public Flux<Category> getAllCategories(){
16         return webClient.get()
17             .uri(uri:"/categories")
18             .retrieve()
19             .bodyToFlux(elementClass:Category.class);
20     }
21
22     public Mono<Category> getCategoryById(Long id){
23         return webClient.get()
24             .uri(uri:"/categories/{id}",id)
25             .retrieve()
26             .bodyToMono(elementClass:Category.class);
27     }
28
29     public Mono<Category> createCategory(Category category){
30         return webClient.post()
31             .uri(uri:"/categories/add")
32             .bodyValue(category)
33             .retrieve()
34             .onStatus(
35                 status -> status.is4xxClientError() || status.is5xxServerError(),
36                 response -> Mono.error(new RuntimeException(message:"Error creating category"))
37             )
38             .bodyToMono(elementClass:Category.class);
39     }
40
41     public Mono<Category> updateCategory(Long id,Category category){
42
43
44
45
46
47 }
```

ภาพที่ 39 ภาพ CategoryService.java

The screenshot shows a Java IDE interface with the following details:

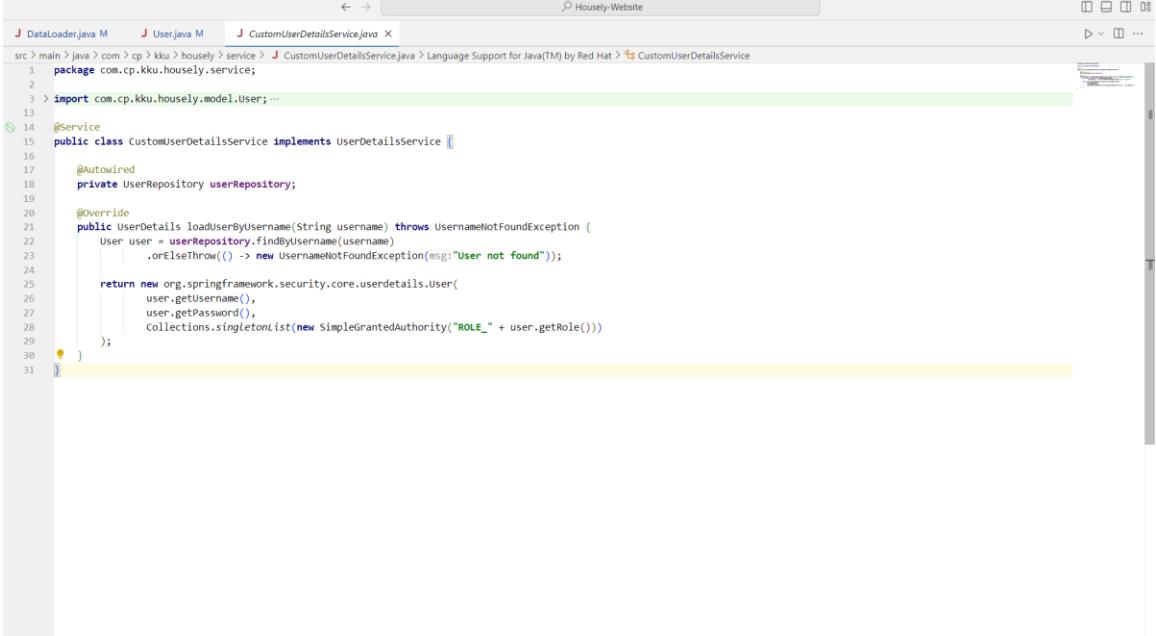
- Title Bar:** Shows the title "Housely-Website".
- Project Structure:** The path is "src > main > java > com > cp > kku > housely > service > CategoryService.java".
- Code Editor:** The file "CategoryService.java" is open, displaying Java code for a REST client using WebClient.
- Code Content:**

```
13 public class CategoryService {
14
15     public Mono<Category> updateCategory(Long id,Category category){
16         return webClient.put()
17             .uri(uri->"categories/update/{id}",id)
18             .bodyValue(category)
19             .retrieve()
20             .onStatus(
21                 status -> status.is4xxClientError() || status.is5xxServerError(),
22                 response -> Mono.error(new RuntimeException(message:"Error updating category")))
23         )
24         .bodyToMono(elementClass:Category.class);
25     }
26
27     public Mono<Void> deleteCategory(Long id){
28         return webClient.delete()
29             .uri(uri->"categories/delete/{id}",id)
30             .retrieve()
31             .bodyToMono(elementClass:Void.class);
32     }
33 }
```
- Right Panel:** Shows various toolbars and panels typical of an IDE, including a file tree, search, and help sections.

ภาพที่ 39 ภาพ CategoryService.java (ต่อ)

- CustomUserDetailsService.java

- loadUserByUsername(String username): ดึงผู้ใช้ตามชื่อผู้ใช้:
- หากพบผู้ใช้ จะคืนค่า UserDetails ที่ประกอบด้วยชื่อผู้ใช้ รหัสผ่าน และสิทธิ์ (ROLE_)
- หากไม่พบ จะโยน UsernameNotFoundException



```

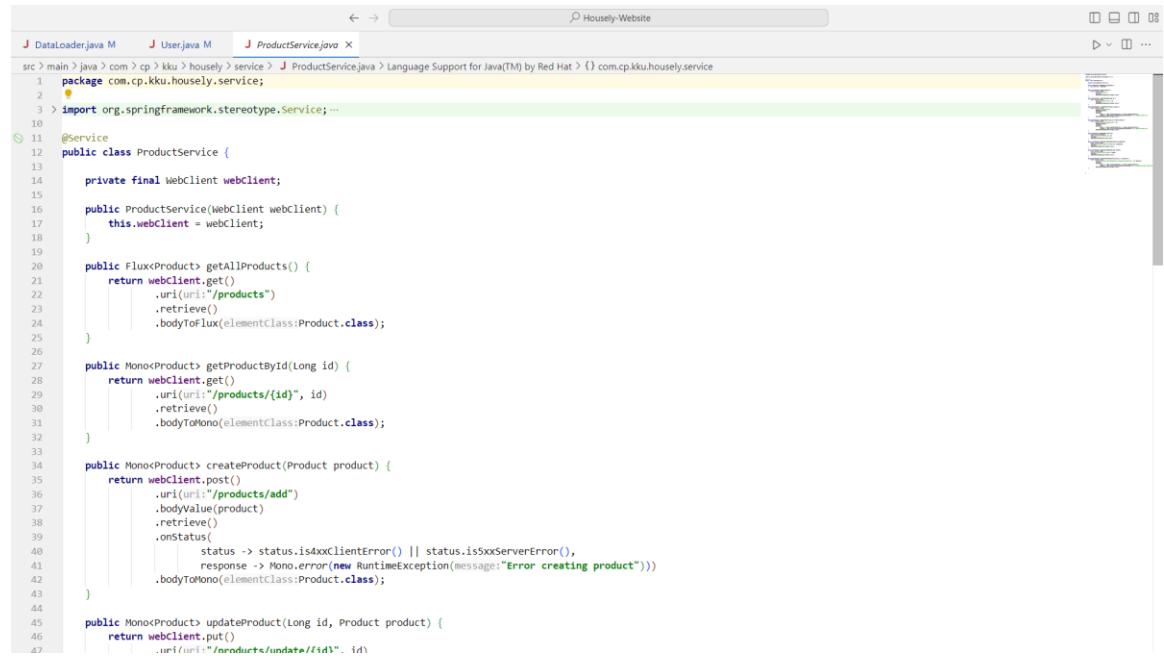
1 package com.cp.kku.housely.service;
2
3 > import com.cp.kku.housely.model.User; ...
4
5 @Service
6 public class CustomUserDetailsService implements UserDetailsService {
7
8     @Autowired
9     private UserRepository userRepository;
10
11     @Override
12     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
13         User user = userRepository.findByUsername(username)
14             .orElseThrow(() -> new UsernameNotFoundException("User not found"));
15
16         return new org.springframework.security.core.userdetails.User(
17             user.getUsername(),
18             user.getPassword(),
19             Collections.singletonList(new SimpleGrantedAuthority("ROLE_" + user.getRole())));
20     }
21 }
22
23
24
25
26
27
28
29
30
31

```

ภาพที่ 40 ภาพ CustomUserDetailsService.java

- ProductService.java

- **CRUD Operations:** พังก์ชันสำหรับจัดการผลิตภัณฑ์ (สร้าง, อ่าน, อัปเดต, ลบ)
- **Get Products by Category/Room:** ดึงผลิตภัณฑ์ตามหมวดหมู่หรือห้อง
- **Decrease Product Quantity:** ลดจำนวนผลิตภัณฑ์

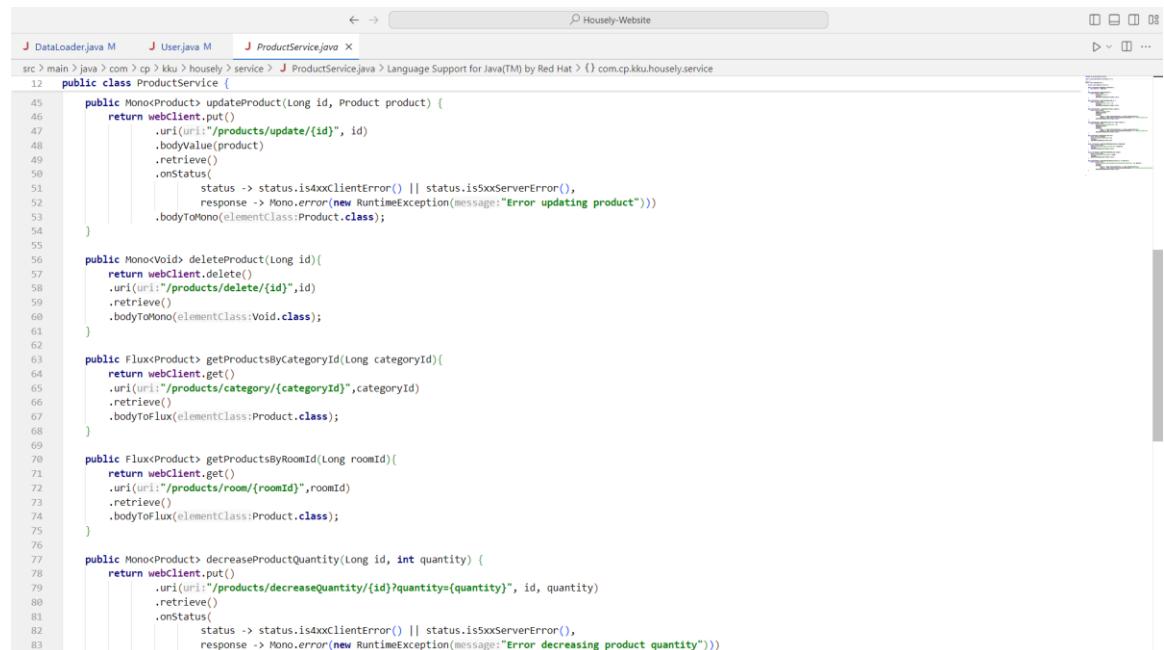


```

J DataLoader.java M J User.java M J ProductService.java X
src > main > java > com > cp > kku > housely > service > J ProductService.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.service
1 package com.cp.kku.housely.service;
2
3 > import org.springframework.stereotype.Service; ...
4
5 @Service
6 public class ProductService {
7
8     private final WebClient webClient;
9
10    public ProductService(WebClient webClient) {
11        this.webClient = webClient;
12    }
13
14    public Flux<Product> getAllProducts() {
15        return webClient.get()
16            .uri("/products")
17            .retrieve()
18            .bodyToFlux(elementClass:Product.class);
19    }
20
21    public Mono<Product> getProductById(Long id) {
22        return webClient.get()
23            .uri("/products/{id}", id)
24            .retrieve()
25            .bodyToMono(elementClass:Product.class);
26    }
27
28    public Mono<Product> createProduct(Product product) {
29        return webClient.post()
30            .uri("/products/add")
31            .bodyValue(product)
32            .retrieve()
33            .onStatus(
34                status -> status.is4xxClientError() || status.is5xxServerError(),
35                response -> Mono.error(new RuntimeException(message:"Error creating product")))
36            .bodyToMono(elementClass:Product.class);
37    }
38
39    public Mono<Product> updateProduct(Long id, Product product) {
40        return webClient.put()
41            .uri("/products/update/{id}", id)
42            .bodyValue(product)
43            .retrieve()
44            .onStatus(
45                status -> status.is4xxClientError() || status.is5xxServerError(),
46                response -> Mono.error(new RuntimeException(message:"Error updating product")))
47            .bodyToMono(elementClass:Product.class);
48    }
49
50    public Mono<Void> deleteProduct(Long id){
51        return webClient.delete()
52            .uri("/products/delete/{id}", id)
53            .retrieve()
54            .bodyToMono(elementClass:Void.class);
55    }
56
57    public Flux<Product> getProductsByCategoryId(Long categoryId){
58        return webClient.get()
59            .uri("/products/category/{categoryId}", categoryId)
60            .retrieve()
61            .bodyToFlux(elementClass:Product.class);
62    }
63
64    public Flux<Product> getProductsByRoomId(Long roomId){
65        return webClient.get()
66            .uri("/products/room/{roomId}", roomId)
67            .retrieve()
68            .bodyToFlux(elementClass:Product.class);
69    }
70
71    public Mono<Product> decreaseProductQuantity(Long id, int quantity) {
72        return webClient.put()
73            .uri("/products/decreaseQuantity/{id}?quantity={quantity}", id, quantity)
74            .retrieve()
75            .onStatus(
76                status -> status.is4xxClientError() || status.is5xxServerError(),
77                response -> Mono.error(new RuntimeException(message:"Error decreasing product quantity")))
78    }
79
80
81
82
83

```

ภาพที่ 41 ภาพ ProductService.java



```

J DataLoader.java M J User.java M J ProductService.java X
src > main > java > com > cp > kku > housely > service > J ProductService.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.service
12 public class ProductService {
13
14    public Mono<Product> updateProduct(Long id, Product product) {
15        return webClient.put()
16            .uri("/products/update/{id}", id)
17            .bodyValue(product)
18            .retrieve()
19            .onStatus(
20                status -> status.is4xxClientError() || status.is5xxServerError(),
21                response -> Mono.error(new RuntimeException(message:"Error updating product")))
22            .bodyToMono(elementClass:Product.class);
23    }
24
25
26    public Mono<Void> deleteProduct(Long id){
27        return webClient.delete()
28            .uri("/products/delete/{id}", id)
29            .retrieve()
30            .bodyToMono(elementClass:Void.class);
31    }
32
33    public Flux<Product> getProductsByCategoryId(Long categoryId){
34        return webClient.get()
35            .uri("/products/category/{categoryId}", categoryId)
36            .retrieve()
37            .bodyToFlux(elementClass:Product.class);
38    }
39
40    public Flux<Product> getProductsByRoomId(Long roomId){
41        return webClient.get()
42            .uri("/products/room/{roomId}", roomId)
43            .retrieve()
44            .bodyToFlux(elementClass:Product.class);
45    }
46
47    public Mono<Product> decreaseProductQuantity(Long id, int quantity) {
48        return webClient.put()
49            .uri("/products/decreaseQuantity/{id}?quantity={quantity}", id, quantity)
50            .retrieve()
51            .onStatus(
52                status -> status.is4xxClientError() || status.is5xxServerError(),
53                response -> Mono.error(new RuntimeException(message:"Error decreasing product quantity")))
54    }
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

```

ภาพที่ 41 ภาพ ProductService.java (ต่อ)

```

src > main > java > com > cp > kku > housely > service > J ProductService.java > Language Support for Java(TM) by Red Hat > () com.cp.kku.housely.service
12 public class ProductService {
13
14     public Mono<Product> decreaseProductQuantity(Long id, int quantity) {
15         return webClient.put()
16             .uri(uri("products/decreaseQuantity/{id}?quantity={quantity}", id, quantity)
17                 .retrieve()
18                 .onStatus(
19                     status -> status.is4xxClientError() || status.is5xxServerError(),
20                     response -> Mono.error(new RuntimeException(message:"Error decreasing product quantity")))
21             .bodyToMono(elementClass:Product.class);
22     }
23
24 }

```

ภาพที่ 41 ภาพ ProductService.java (ต่อ)

- RoomService.java

- CRUD Operations: ฟังก์ชันสำหรับจัดการห้อง (สร้าง, อ่าน, อัปเดต, ลบ)
- WebClient: ใช้ WebClient ในการเรียก API สำหรับห้อง

```

src > main > java > com > cp > kku > housely > service > J RoomService.java > Language Support for Java(TM) by Red Hat > () com.cp.kku.housely.service
1 package com.cp.kku.housely.service;
2
3 > import org.springframework.stereotype.Service;
4
5 @Service
6 public class RoomService {
7     private final WebClient webClient;
8
9     public RoomService(WebClient webclient) {
10         this.webClient = webclient;
11     }
12
13     public Flux<Room> getAllRooms() {
14         return webClient.get()
15             .uri(uri("rooms"))
16             .retrieve()
17             .bodyToFlux(elementClass:Room.class);
18     }
19
20     public Mono<Room> getRoomById(Long id) {
21         return webClient.get()
22             .uri(uri("rooms/{id}", id)
23                 .retrieve()
24                 .bodyToMono(elementClass:Room.class));
25     }
26
27     public Mono<Room> createRoom(Room room) {
28         return webClient.post()
29             .uri(uri("rooms/add"))
30             .bodyValue(room)
31             .retrieve()
32             .onStatus(
33                 status -> status.is4xxClientError() || status.is5xxServerError(),
34                 response -> Mono.error(new RuntimeException(message:"Error creating room"))
35             )
36             .bodyToMono(elementClass:Room.class);
37     }
38
39     public Mono<Room> updateRoom(Long id, Room room) {
40         return webClient.put()
41             .uri(uri("rooms/update/{id}", id)
42                 .bodyValue(room)
43                 .retrieve()
44                 .onStatus(
45                     status -> status.is4xxClientError() || status.is5xxServerError(),
46                     response -> Mono.error(new RuntimeException(message:"Error updating room"))
47             )
48             .bodyToMono(elementClass:Room.class);
49     }
50 }

```

ภาพที่ 42 ภาพ RoomService.java



The screenshot shows a Java IDE interface with the following details:

- Title Bar:** Shows the title "Housely-Website".
- Project Structure:** The project tree shows "src > main > java > com > cp > kku > housely > service > RoomService.java".
- Code Editor:** The code for `RoomService.java` is displayed. It contains two methods: `updateRoom` and `deleteRoom`. Both methods use a WebClient to interact with a REST API endpoint for rooms.

```
12 public class RoomService {  
13     ...  
14     public Mono<Room> updateRoom(Long id, Room room){  
15         return webclient.put()  
16             .uri(uri:"/rooms/update/{id}",id)  
17             .bodyValue(room)  
18             .retrieve()  
19             .onStatus(  
20                 status -> status.is4xxClientError() || status.is5xxServerError(),  
21                 response -> Mono.error(new RuntimeException(message:"Error updating room"))  
22             )  
23             .bodyToMono(elementClass:Room.class);  
24     }  
25     ...  
26     public Mono<Void> deleteRoom(Long id){  
27         return webclient.delete()  
28             .uri(uri:"/rooms/delete/{id}",id)  
29             .retrieve()  
30             .bodyToMono(elementClass:Void.class);  
31     }  
32     ...  
33 }
```

ภาพที่ 42 ภาพ RoomService.java (ต่อ)

- UserService.java

- Register User: เข้ารหัสรหัสผ่านและบันทึกผู้ใช้ในฐานข้อมูล
 - Find User: ค้นหาผู้ใช้ตามชื่อผู้ใช้

```

J DataLoader.java M J User.java M J UserService.java X
src > main > java > com > cp > kku > housely > service > J UserService.java > Language Support for Java(TM) by Red Hat > () com.cp.kku.housely.service
1 package com.cp.kku.housely.service;
2
3 import com.cp.kku.housely.model.User;
4 import com.cp.kku.housely.repository.UserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.security.crypto.password.PasswordEncoder;
7 import org.springframework.stereotype.Service;
8
9 @Service
10 public class UserService {
11
12     @Autowired
13     private UserRepository userRepository;
14
15     @Autowired
16     private PasswordEncoder passwordEncoder;
17
18     public void register(User user) {
19         user.setPassword(passwordEncoder.encode(user.getPassword())); // เมื่อ註冊ผู้ใช้งาน
20         userRepository.save(user);
21     }
22
23     public User findbyUsername(String username) {
24         return userRepository.findByUsername(username).orElse(null);
25     }
26 }

```

ภาพที่ 43 ภาพ UserService.java

- WebConfig.java

WebClient Bean: สร้าง WebClient ที่ตั้งค่า base URL สำหรับ API

```

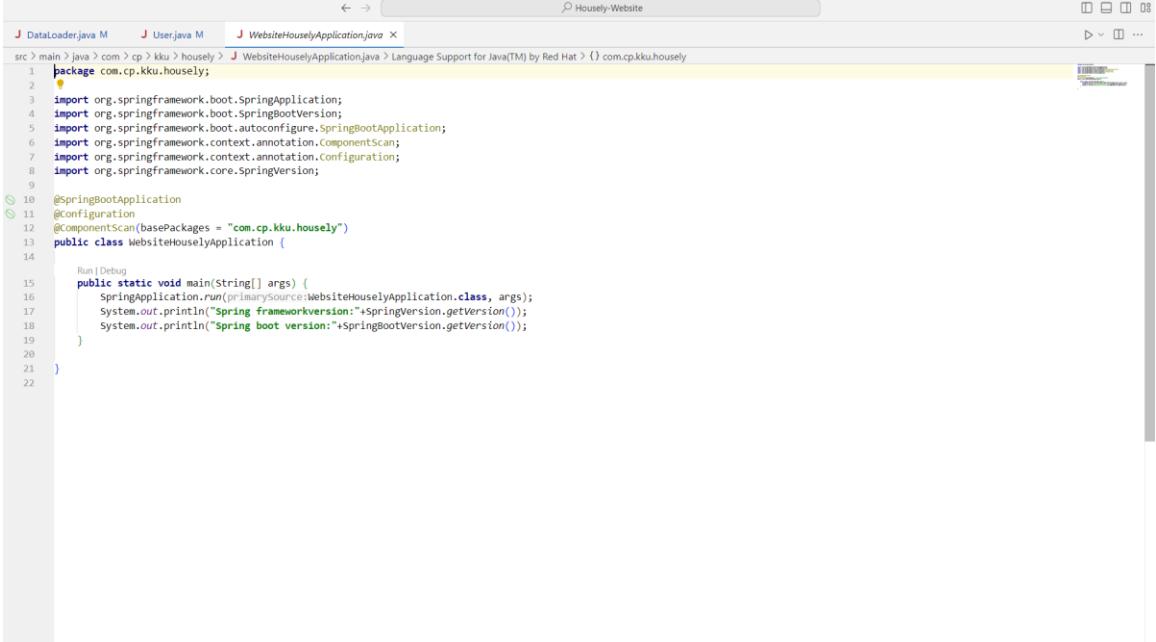
J DataLoader.java M J User.java M J WebConfig.java X
src > main > java > com > cp > kku > housely > service > J WebConfig.java > Language Support for Java(TM) by Red Hat > () com.cp.kku.housely.service
1 package com.cp.kku.housely.service;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.web.reactive.function.client.WebClient;
6
7 @Configuration
8 public class WebConfig {
9
10     @Bean
11     public WebClient webClient(WebClient.Builder builder) {
12         return builder.baseUrl("http://localhost:8089/api").build();
13     }
14 }

```

ภาพที่ 44 ภาพ WebConfig.java

WebsiteHouselyApplication.java

- **Main Class:** เริ่มต้นแอปพลิเคชัน Spring Boot
- **Print Versions:** แสดงเวอร์ชันของ Spring Framework และ Spring Boot



```

1 package com.cp.kku.housely;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.SpringBootVersion;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6 import org.springframework.context.annotation.ComponentScan;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.core.SpringVersion;
9
10 @SpringBootApplication
11 @Configuration
12 @ComponentScan(basePackages = "com.cp.kku.housely")
13 public class WebsiteHouselyApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(WebsiteHouselyApplication.class, args);
17         System.out.println("Spring framework version:" + SpringVersion.getVersion());
18         System.out.println("Spring boot version:" + SpringBootVersion.getVersion());
19     }
20 }
21
22

```

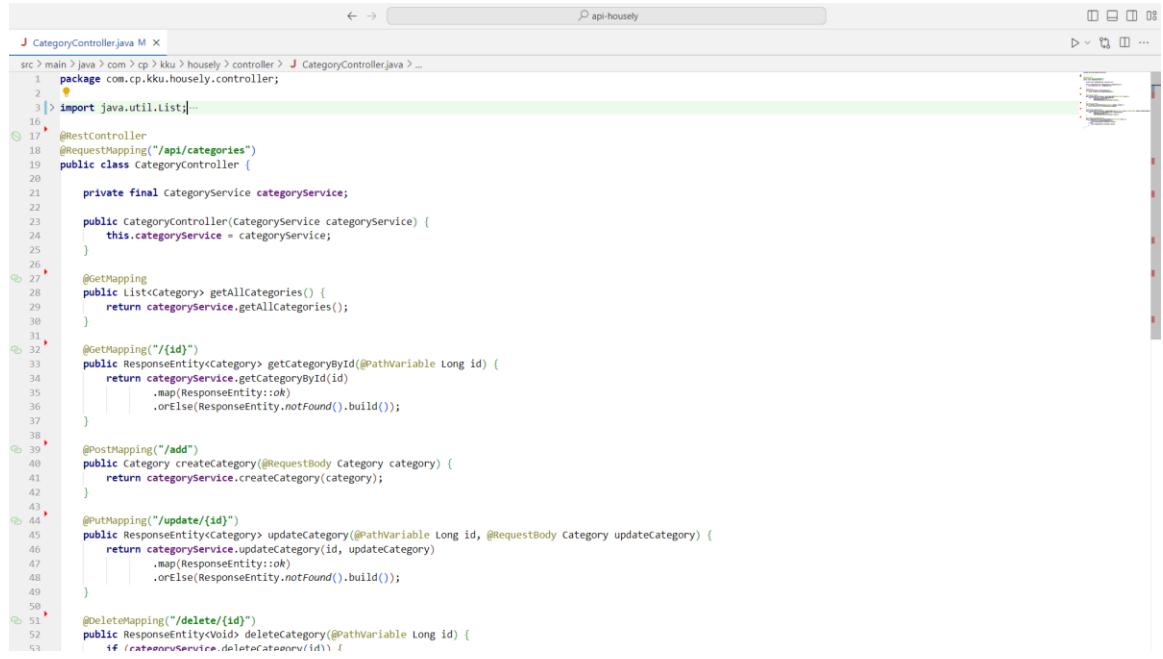
ภาพที่ 45 ภาพ WebsiteHouselyApplication.java

4.1.3 การเชื่อมต่อระบบกับฐานข้อมูลโดยใช้ JPA เพื่อดำเนินการ CRUD Operation และการพัฒนา Backend โดยใช้ Spring Boot Framework และ RESTful API ในการจัดการกับข้อมูล

Controller

- CategoryController.java

จัดการ CRUD สำหรับ Category คล้ายกับ controller อื่น ๆ

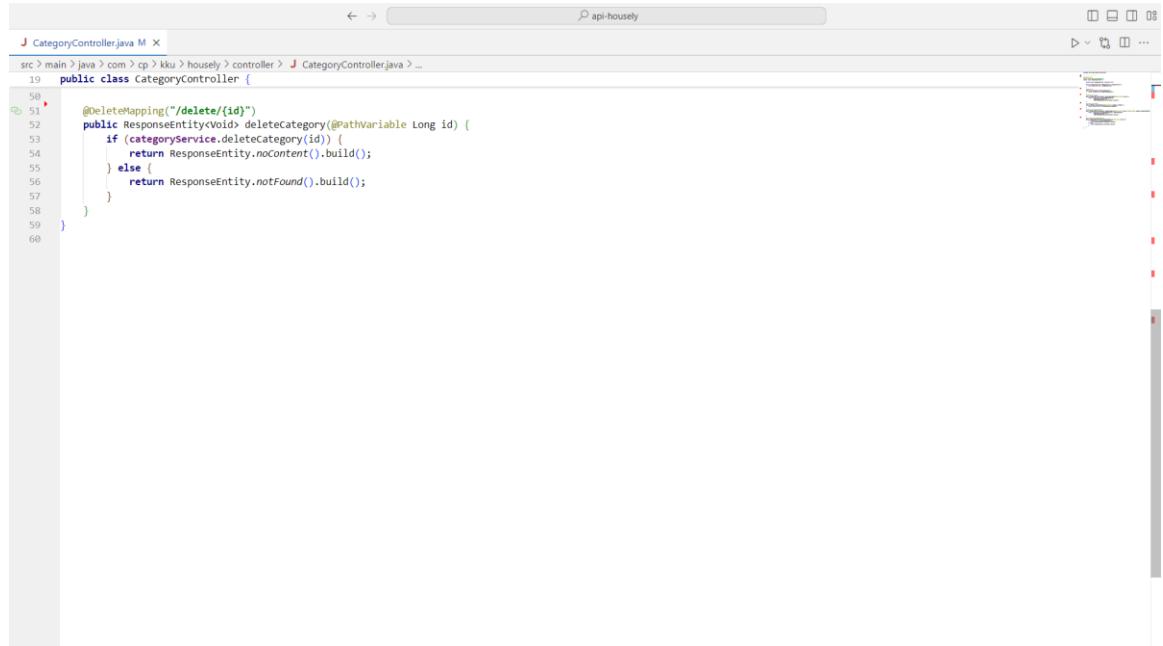


```

src > main > java > com > cp > kku > housely > controller > CategoryController.java ...
1 package com.cp.kku.housely.controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.*;
8
9 public class CategoryController {
10
11     private final CategoryService categoryService;
12
13     @Autowired
14     public CategoryController(CategoryService categoryService) {
15         this.categoryService = categoryService;
16     }
17
18     @GetMapping("/categories")
19     public List<Category> getAllCategories() {
20         return categoryService.getAllCategories();
21     }
22
23     @GetMapping("/{id}")
24     public ResponseEntity<Category> getCategoryById(@PathVariable Long id) {
25         return categoryService.getCategoryById(id)
26             .map(ResponseEntity::ok)
27             .orElse(ResponseEntity.notFound().build());
28     }
29
30     @PostMapping("/add")
31     public Category createCategory(@RequestBody Category category) {
32         return categoryService.createCategory(category);
33     }
34
35     @PutMapping("/update/{id}")
36     public ResponseEntity<Category> updateCategory(@PathVariable Long id, @RequestBody Category updateCategory) {
37         return categoryService.updateCategory(id, updateCategory)
38             .map(ResponseEntity::ok)
39             .orElse(ResponseEntity.notFound().build());
40     }
41
42     @DeleteMapping("/delete/{id}")
43     public ResponseEntity<Void> deleteCategory(@PathVariable Long id) {
44         if (categoryService.deleteCategory(id)) {
45             return ResponseEntity.noContent().build();
46         } else {
47             return ResponseEntity.notFound().build();
48         }
49     }
50
51 }
52
53
54
55
56
57
58
59
60

```

ภาพที่ 46 ภาพ CategoryController.java



```

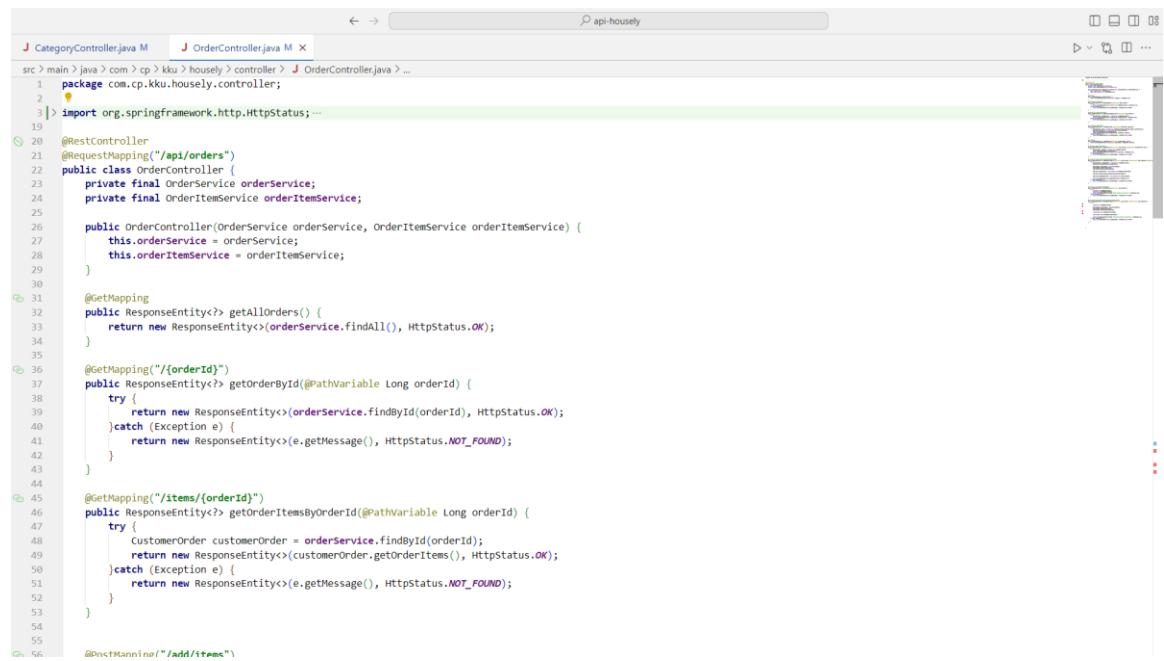
src > main > java > com > cp > kku > housely > controller > CategoryController.java ...
19 public class CategoryController {
20
21     @DeleteMapping("/delete/{id}")
22     public ResponseEntity<Void> deleteCategory(@PathVariable Long id) {
23         if (categoryService.deleteCategory(id)) {
24             return ResponseEntity.noContent().build();
25         } else {
26             return ResponseEntity.notFound().build();
27         }
28     }
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

ภาพที่ 46 ภาพ CategoryController.java (ต่อ)

- OrderController.java

จัดการ CustomerOrder และ OrderItem รวมถึงการสร้าง, ดึง, อัปเดต,
และลบออร์เดอร์และการสินค้า โดยมีการจัดการข้อมูลพลาด

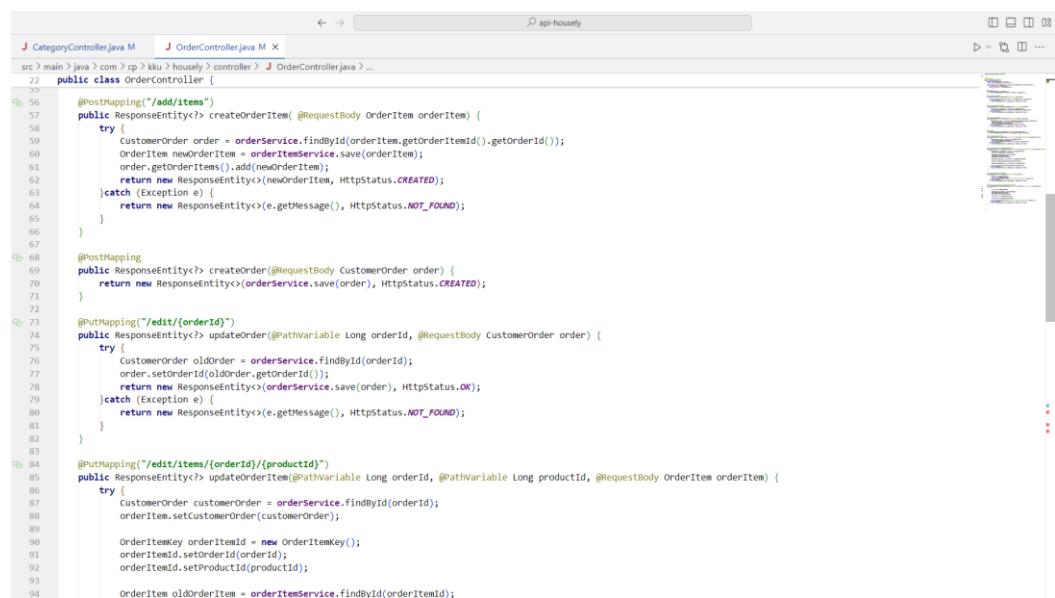


```

J CategoryController.java M J OrderController.java M X
src > main > java > com > cp > kku > housely > controller > J OrderController.java > ...
1 package com.cp.kku.housely.controller;
2
3 import org.springframework.http.HttpStatus;
4
5 @RestController
6 @RequestMapping("/api/orders")
7 public class OrderController {
8     private final OrderService orderService;
9     private final OrderItemService orderItemService;
10
11     public OrderController(OrderService orderService, OrderItemService orderItemService) {
12         this.orderService = orderService;
13         this.orderItemService = orderItemService;
14     }
15
16     @GetMapping
17     public ResponseEntity<> getAllOrders() {
18         return new ResponseEntity<>(orderService.findAll(), HttpStatus.OK);
19     }
20
21     @GetMapping("/{orderId}")
22     public ResponseEntity<> getOrderById(@PathVariable Long orderId) {
23         try {
24             return new ResponseEntity<>(orderService.findById(orderId), HttpStatus.OK);
25         } catch (Exception e) {
26             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
27         }
28     }
29
30     @GetMapping("/{items/{orderId}}")
31     public ResponseEntity<> getOrderItemsByOrderId(@PathVariable Long orderId) {
32         try {
33             CustomerOrder customerOrder = orderService.findById(orderId);
34             return new ResponseEntity<>(customerOrder.getOrderedItems(), HttpStatus.OK);
35         } catch (Exception e) {
36             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
37         }
38     }
39
40     @PostMapping("/add/items")
41
42     public ResponseEntity<> createOrderItem( @RequestBody OrderItem orderItem) {
43         try {
44             CustomerOrder order = orderService.findById(orderItem.getOrderId());
45             OrderItem newItem = orderItemService.save(orderItem);
46             order.getOrderedItems().add(newItem);
47             return new ResponseEntity<>(newItem, HttpStatus.CREATED);
48         } catch (Exception e) {
49             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
50         }
51     }
52
53 }
54
55
56     @PostMapping("/add/items")
57
58     public ResponseEntity<> createOrder(@RequestBody CustomerOrder order) {
59         return new ResponseEntity<>(orderService.save(order), HttpStatus.CREATED);
60     }
61
62     @PutMapping("/edit/{orderId}")
63     public ResponseEntity<> updateOrder(@PathVariable Long orderId, @RequestBody CustomerOrder order) {
64         try {
65             CustomerOrder oldOrder = orderService.findById(orderId);
66             oldOrder.setOrder(order);
67             return new ResponseEntity<>(orderService.save(order), HttpStatus.OK);
68         } catch (Exception e) {
69             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
70         }
71     }
72
73     @PutMapping("/edit/items/{orderId}/{productId}")
74     public ResponseEntity<> updateOrderItem(@PathVariable Long orderId, @PathVariable Long productId, @RequestBody OrderItem orderItem) {
75         try {
76             CustomerOrder customerOrder = orderService.findById(orderId);
77             orderItem.setCustomerOrder(customerOrder);
78             OrderItemKey orderItemId = new OrderItemKey();
79             orderItemId.setOrderId(orderId);
80             orderItemId.setProductId(productId);
81             orderItem.setId(orderItemId);
82             OrderItem oldOrderItem = orderItemService.findById(orderItemId);
83             oldOrderItem.setOrderItem(orderItem);
84             orderItemService.update(oldOrderItem);
85         } catch (Exception e) {
86             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
87         }
88     }
89
90 }
91
92
93
94

```

ภาพที่ 47 ภาพ OrderController.java



```

J CategoryController.java M J OrderController.java M X
src > main > java > com > cp > kku > housely > controller > J OrderController.java > ...
22
23     public class OrderController {
24
25         @PostMapping("/add/items")
26         public ResponseEntity<> createOrderItem( @RequestBody OrderItem orderItem) {
27             try {
28                 CustomerOrder order = orderService.findById(orderItem.getOrderId());
29                 OrderItem newItem = orderItemService.save(orderItem);
30                 order.getOrderedItems().add(newItem);
31                 return new ResponseEntity<>(newItem, HttpStatus.CREATED);
32             } catch (Exception e) {
33                 return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
34             }
35         }
36
37         @PostMapping("/add/items")
38
39         public ResponseEntity<> createOrder(@RequestBody CustomerOrder order) {
40             return new ResponseEntity<>(orderService.save(order), HttpStatus.CREATED);
41         }
42
43         @PutMapping("/edit/{orderId}")
44         public ResponseEntity<> updateOrder(@PathVariable Long orderId, @RequestBody CustomerOrder order) {
45             try {
46                 CustomerOrder oldOrder = orderService.findById(orderId);
47                 oldOrder.setOrder(order);
48                 return new ResponseEntity<>(orderService.save(order), HttpStatus.OK);
49             } catch (Exception e) {
50                 return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
51             }
52         }
53
54         @PutMapping("/edit/items/{orderId}/{productId}")
55         public ResponseEntity<> updateOrderItem(@PathVariable Long orderId, @PathVariable Long productId, @RequestBody OrderItem orderItem) {
56             try {
57                 CustomerOrder customerOrder = orderService.findById(orderId);
58                 orderItem.setCustomerOrder(customerOrder);
59                 OrderItemKey orderItemId = new OrderItemKey();
60                 orderItemId.setOrderId(orderId);
61                 orderItemId.setProductId(productId);
62                 orderItem.setId(orderItemId);
63                 OrderItem oldOrderItem = orderItemService.findById(orderItemId);
64                 oldOrderItem.setOrderItem(orderItem);
65                 orderItemService.update(oldOrderItem);
66             } catch (Exception e) {
67                 return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
68             }
69         }
70
71     }
72
73 }
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

ภาพที่ 47 ภาพ OrderController.java (ต่อ)

```

J CategoryController.java M J OrderController.java M X
src > main > java > com > cp > kku > housely > controller > J OrderController.java > Language Support for Java(TM) by Red Hat > OrderController > updateOrderItem(Long, Long, OrderItem)
22 public class OrderController {
23
24     @PutMapping("/edit/items/{orderId}/{productId}")
25     public ResponseEntity<> updateOrderItem(@PathVariable Long orderId, @PathVariable Long productId, @RequestBody OrderItem orderItem) {
26         try {
27             CustomerOrder customerOrder = orderService.findById(orderId);
28             orderItem.setCustomerOrder(customerOrder);
29
30             OrderItemKey orderItemId = new OrderItemKey();
31             orderItemId.setOrderId(orderId);
32             orderItemId.setProductId(productId);
33
34             OrderItem oldOrderItem = orderItemService.findById(orderItemId);
35
36             orderItem.setOrderItemId(oldOrderItem.getOrderId());
37
38             OrderItem updatedOrderItem = orderItemService.save(orderItem);
39
40             return new ResponseEntity<>(updatedOrderItem, HttpStatus.OK);
41         } catch (Exception e) {
42             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
43         }
44     }
45
46
47     @DeleteMapping("/delete/{orderId}")
48     public ResponseEntity<> deleteOrder(@PathVariable Long orderId) {
49         try {
50             orderService.deleteById(orderId);
51             orderService.deleteById(orderId);
52             return new ResponseEntity<>("Order deleted successfully", HttpStatus.OK);
53         } catch (Exception e) {
54             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
55         }
56     }
57
58     @DeleteMapping("/delete/items/{orderId}/{productId}")
59     public ResponseEntity<> deleteOrderItem(@PathVariable Long orderId, @PathVariable Long productId) {
60         try {
61             orderService.deleteById(orderId);
62
63             OrderItemKey orderItemId = new OrderItemKey();
64             orderItemId.setOrderId(orderId);
65             orderItemId.setProductId(productId);
66
67             orderItemService.findById(orderItemId);
68
69             orderItemService.deleteById(orderItemId);
70
71             return new ResponseEntity<>("OrderItem deleted successfully", HttpStatus.OK);
72         } catch (Exception e) {
73             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
74         }
75     }
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121

```

ภาพที่ 47 ภาพ OrderController.java (ต่อ)

```

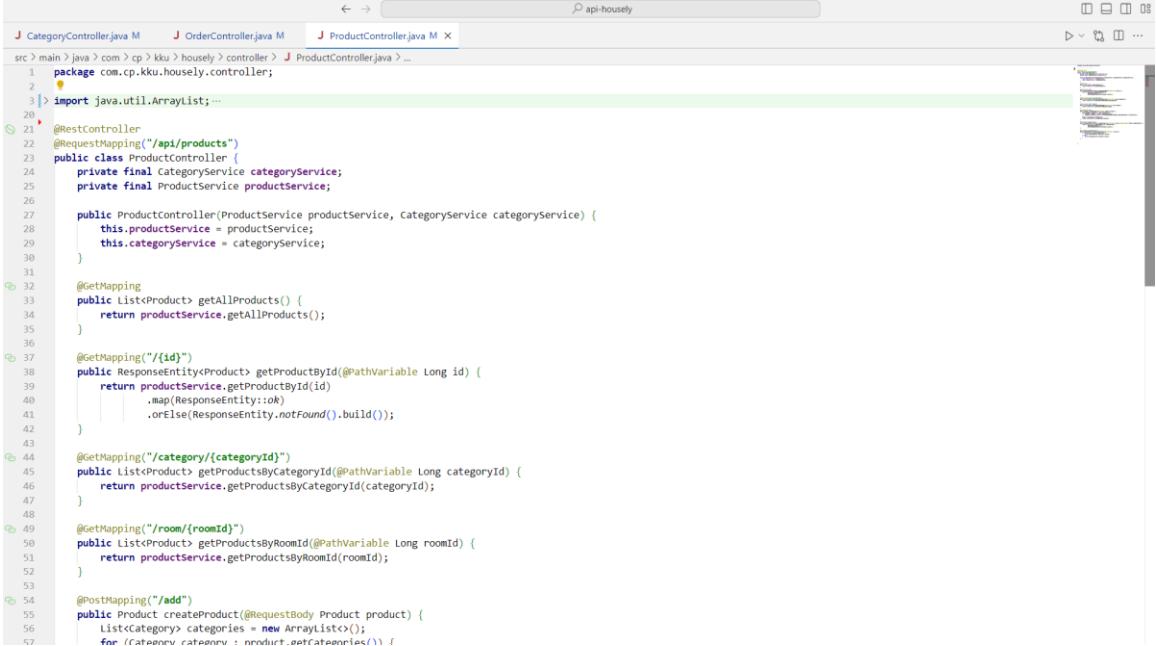
J CategoryController.java M J OrderController.java M X
src > main > java > com > cp > kku > housely > controller > J OrderController.java > Language Support for Java(TM) by Red Hat > OrderController > updateOrderItem(Long, Long, OrderItem)
22 public class OrderController {
23
24     public ResponseEntity<> deleteOrderItem(@PathVariable Long orderId, @PathVariable Long productId) {
25         try {
26             orderService.deleteById(orderId);
27
28             OrderItemKey orderItemId = new OrderItemKey();
29             orderItemId.setOrderId(orderId);
30             orderItemId.setProductId(productId);
31
32             orderItemService.findById(orderItemId);
33
34             orderItemService.deleteById(orderItemId);
35
36             return new ResponseEntity<>("OrderItem deleted successfully", HttpStatus.OK);
37         } catch (Exception e) {
38             return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
39         }
40     }
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121

```

ภาพที่ 47 ภาพ OrderController.java (ต่อ)

- ProductController.java

จัดการ CRUD สำหรับ Product รวมถึงการกรองสินค้าตามหมวดหมู่หรือห้องพัก
พร้อมตรวจสอบหมวดหมู่ก่อนการสร้างสินค้า

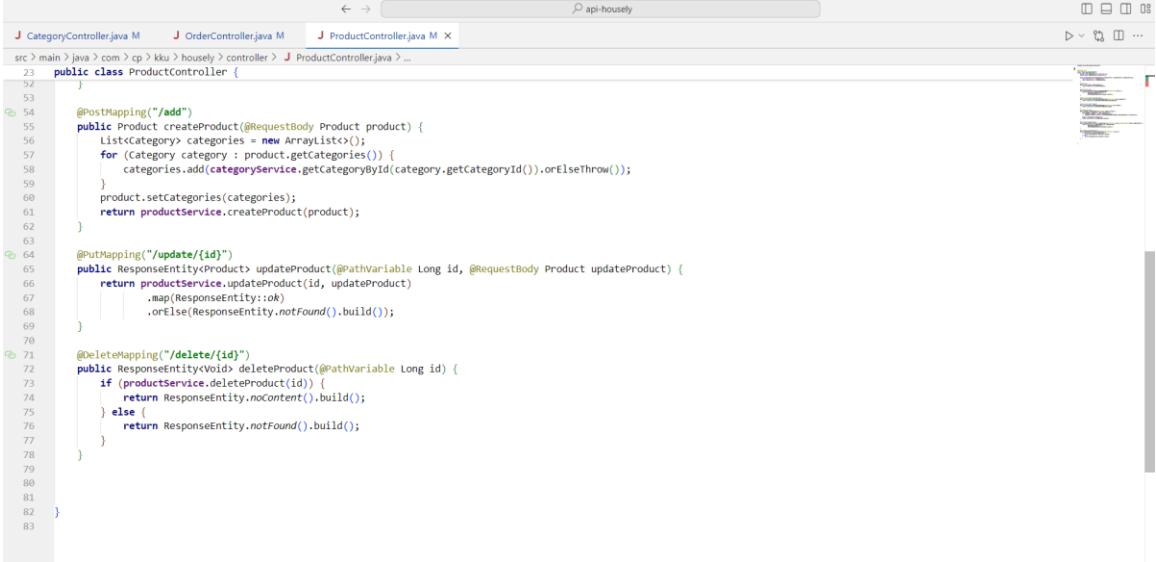


```

J CategoryController.java M J OrderController.java M J ProductController.java M X
src > main > java > com > cp > kku > housely > controller > J ProductController.java ...
1 package com.cp.kku.housely.controller;
2
3 import java.util.ArrayList;
4
5 import org.springframework.web.bind.annotation.*;
6
7 @RestController
8 @RequestMapping("/api/products")
9 public class ProductController {
10     private final CategoryService categoryService;
11     private final ProductService productService;
12
13     public ProductController(ProductService productService, CategoryService categoryService) {
14         this.productService = productService;
15         this.categoryService = categoryService;
16     }
17
18     @GetMapping
19     public List<Product> getAllProducts() {
20         return productService.getAllProducts();
21     }
22
23     @GetMapping("/{id}")
24     public ResponseEntity<Product> getProductById(@PathVariable Long id) {
25         return productService.getProductById(id)
26             .map(ResponseEntity::ok)
27             .orElse(ResponseEntity.notFound().build());
28     }
29
30     @GetMapping("/category/{categoryId}")
31     public List<Product> getProductsByCategoryId(@PathVariable Long categoryId) {
32         return productService.getProductsByCategoryId(categoryId);
33     }
34
35     @GetMapping("/room/{roomId}")
36     public List<Product> getProductsByRoomId(@PathVariable Long roomId) {
37         return productService.getProductsByRoomId(roomId);
38     }
39
40     @PostMapping("/add")
41     public Product createProduct(@RequestBody Product product) {
42         List<Category> categories = new ArrayList<>();
43         for (Category category : product.getCategories()) {
44             categories.add(category);
45         }
46         product.setCategories(categories);
47         return productService.createProduct(product);
48     }
49
50     @PutMapping("/{id}")
51     public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product product) {
52         return productService.updateProduct(id, product)
53             .map(ResponseEntity::ok)
54             .orElse(ResponseEntity.notFound().build());
55     }
56
57     @DeleteMapping("/{id}")
58     public void deleteProduct(@PathVariable Long id) {
59         productService.deleteProduct(id);
60     }
61
62     @GetMapping("/search")
63     public List<Product> searchProducts(@RequestParam String query) {
64         return productService.searchProducts(query);
65     }
66
67     @GetMapping("/category/{categoryId}/search")
68     public List<Product> searchProductsByCategory(@PathVariable Long categoryId, @RequestParam String query) {
69         return productService.searchProductsByCategory(categoryId, query);
70     }
71
72     @GetMapping("/room/{roomId}/search")
73     public List<Product> searchProductsByRoom(@PathVariable Long roomId, @RequestParam String query) {
74         return productService.searchProductsByRoom(roomId, query);
75     }
76
77     @GetMapping("/category/{categoryId}/room/{roomId}/search")
78     public List<Product> searchProductsByCategoryAndRoom(@PathVariable Long categoryId, @PathVariable Long roomId, @RequestParam String query) {
79         return productService.searchProductsByCategoryAndRoom(categoryId, roomId, query);
80     }
81
82     @GetMapping("/category/{categoryId}/room/{roomId}/order/{orderId}/search")
83     public List<Product> searchProductsByCategoryAndRoomAndOrder(@PathVariable Long categoryId, @PathVariable Long roomId, @PathVariable Long orderId, @RequestParam String query) {
84         return productService.searchProductsByCategoryAndRoomAndOrder(categoryId, roomId, orderId, query);
85     }
86
87     @GetMapping("/category/{categoryId}/room/{roomId}/order/{orderId}/category/{categoryId}/search")
88     public List<Product> searchProductsByCategoryAndRoomAndOrderAndCategory(@PathVariable Long categoryId, @PathVariable Long roomId, @PathVariable Long orderId, @PathVariable Long categoryId2, @RequestParam String query) {
89         return productService.searchProductsByCategoryAndRoomAndOrderAndCategory(categoryId, roomId, orderId, categoryId2, query);
90     }
91
92     @GetMapping("/category/{categoryId}/room/{roomId}/order/{orderId}/category/{categoryId}/room/{roomId}/search")
93     public List<Product> searchProductsByCategoryAndRoomAndOrderAndCategoryAndRoom(@PathVariable Long categoryId, @PathVariable Long roomId, @PathVariable Long orderId, @PathVariable Long categoryId2, @PathVariable Long roomId2, @RequestParam String query) {
94         return productService.searchProductsByCategoryAndRoomAndOrderAndCategoryAndRoom(categoryId, roomId, orderId, categoryId2, roomId2, query);
95     }
96
97     @GetMapping("/category/{categoryId}/room/{roomId}/order/{orderId}/category/{categoryId}/room/{roomId}/order/{orderId}/search")
98     public List<Product> searchProductsByCategoryAndRoomAndOrderAndCategoryAndRoomAndOrder(@PathVariable Long categoryId, @PathVariable Long roomId, @PathVariable Long orderId, @PathVariable Long categoryId2, @PathVariable Long roomId2, @PathVariable Long orderId2, @RequestParam String query) {
99         return productService.searchProductsByCategoryAndRoomAndOrderAndCategoryAndRoomAndOrder(categoryId, roomId, orderId, categoryId2, roomId2, orderId2, query);
100    }
101}

```

ภาพที่ 48 ภาพ ProductController.java



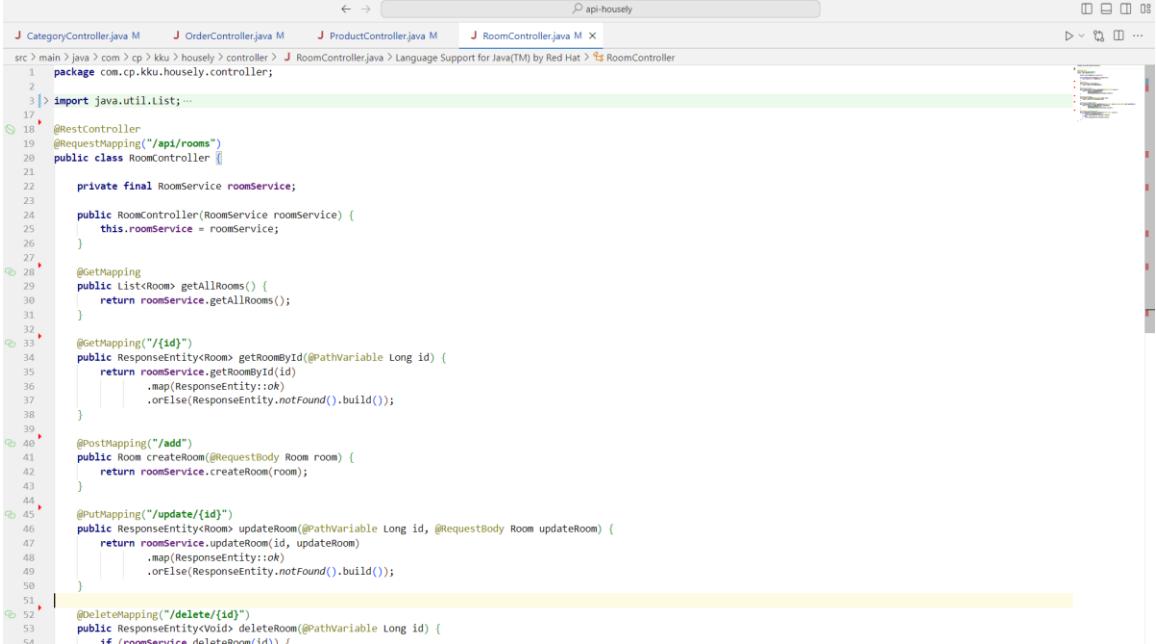
```

src > main > java > com > cp > kku > housely > controller > J ProductController.java > ...
23 public class ProductController {
24     ...
25     @PostMapping("/add")
26     public Product createProduct(@RequestBody Product product) {
27         List<Category> categories = new ArrayList<>();
28         for (Category category : product.getCategories()) {
29             categories.add(categoryService.getCategoryById(category.getId()).orElseThrow());
30         }
31         product.setCategories(categories);
32         return productService.createProduct(product);
33     }
34     ...
35     @PutMapping("/update/{id}")
36     public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product updateProduct) {
37         return productService.updateProduct(id, updateProduct)
38             .map(ResponseEntity::ok)
39             .orElse(ResponseEntity.notFound().build());
40     }
41     ...
42     @DeleteMapping("/delete/{id}")
43     public ResponseEntity<Void> deleteProduct(@PathVariable Long id) {
44         if (productService.deleteProduct(id)) {
45             return ResponseEntity.noContent().build();
46         } else {
47             return ResponseEntity.notFound().build();
48         }
49     }
50     ...
51 }

```

ภาพที่ 48 ภาพ ProductController.java (ต่อ)

- RoomController.java



```

src > main > java > com > cp > kku > housely > controller > J RoomController.java > Language Support for Java(TM) by Red Hat > RoomController
1 package com.cp.kku.housely.controller;
2 ...
3 import java.util.List;
4 ...
5 @RestController
6 @RequestMapping("/api/rooms")
7 public class RoomController {
8     ...
9     private final RoomService roomService;
10    ...
11    public RoomController(RoomService roomService) {
12        this.roomService = roomService;
13    }
14    ...
15    @GetMapping
16    public List<Room> getAllRooms() {
17        return roomService.getAllRooms();
18    }
19    ...
20    @GetMapping("/{id}")
21    public ResponseEntity<Room> getRoomById(@PathVariable Long id) {
22        return roomService.getRoomById(id)
23            .map(ResponseEntity::ok)
24            .orElse(ResponseEntity.notFound().build());
25    }
26    ...
27    @PostMapping("/add")
28    public Room createRoom(@RequestBody Room room) {
29        return roomService.createRoom(room);
30    }
31    ...
32    @PutMapping("/update/{id}")
33    public ResponseEntity<Room> updateRoom(@PathVariable Long id, @RequestBody Room updateRoom) {
34        return roomService.updateRoom(id, updateRoom)
35            .map(ResponseEntity::ok)
36            .orElse(ResponseEntity.notFound().build());
37    }
38    ...
39    @DeleteMapping("/delete/{id}")
40    public ResponseEntity<Void> deleteRoom(@PathVariable Long id) {
41        if (roomService.deleteRoom(id)) {
42            return ResponseEntity.noContent().build();
43        } else {
44            return ResponseEntity.notFound().build();
45        }
46    }
47 }

```

ภาพที่ 49 ภาพ RoomController.java



```

20  public class RoomController {
21      @DeleteMapping("/delete/{id}")
22      public ResponseEntity<Void> deleteRoom(@PathVariable Long id) {
23          if (roomService.deleteRoom(id)) {
24              return ResponseEntity.noContent().build();
25          } else {
26              return ResponseEntity.notFound().build();
27          }
28      }
29  }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61

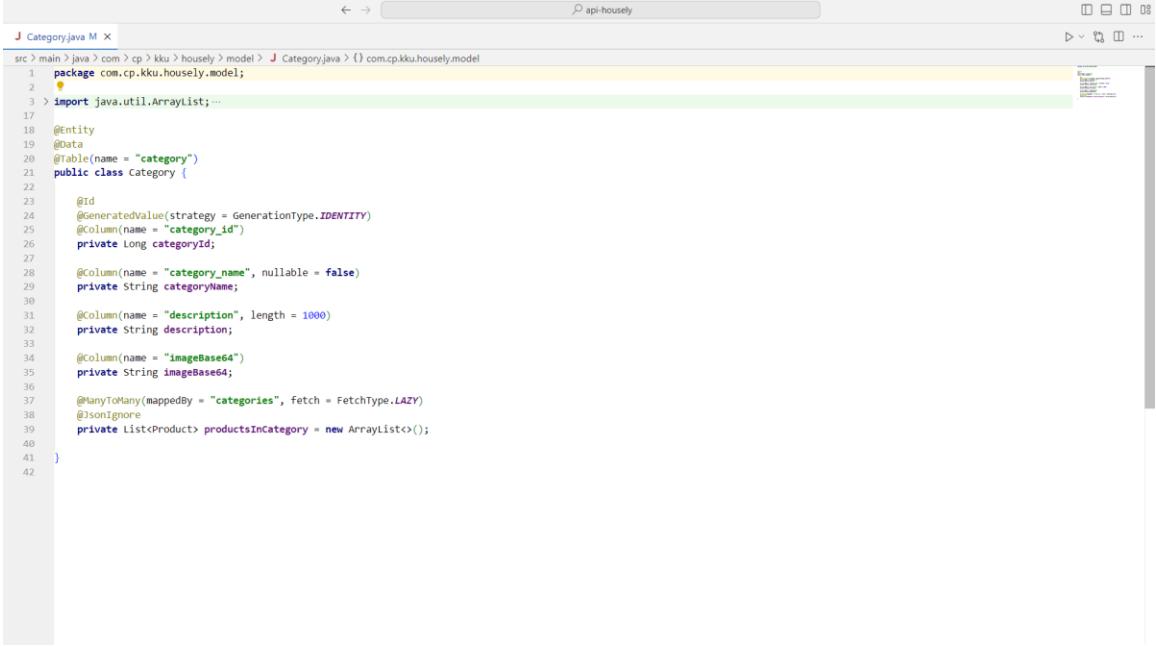
```

ภาพที่ 49 ภาพ RoomController.java (ต่อ)

Model

- Category.java

- เก็บข้อมูลหมวดหมู่สินค้า เช่น ID, ชื่อหมวดหมู่, รายละเอียด, และรูปภาพ
- มีความสัมพันธ์แบบ Many-to-Many กับ Product



```

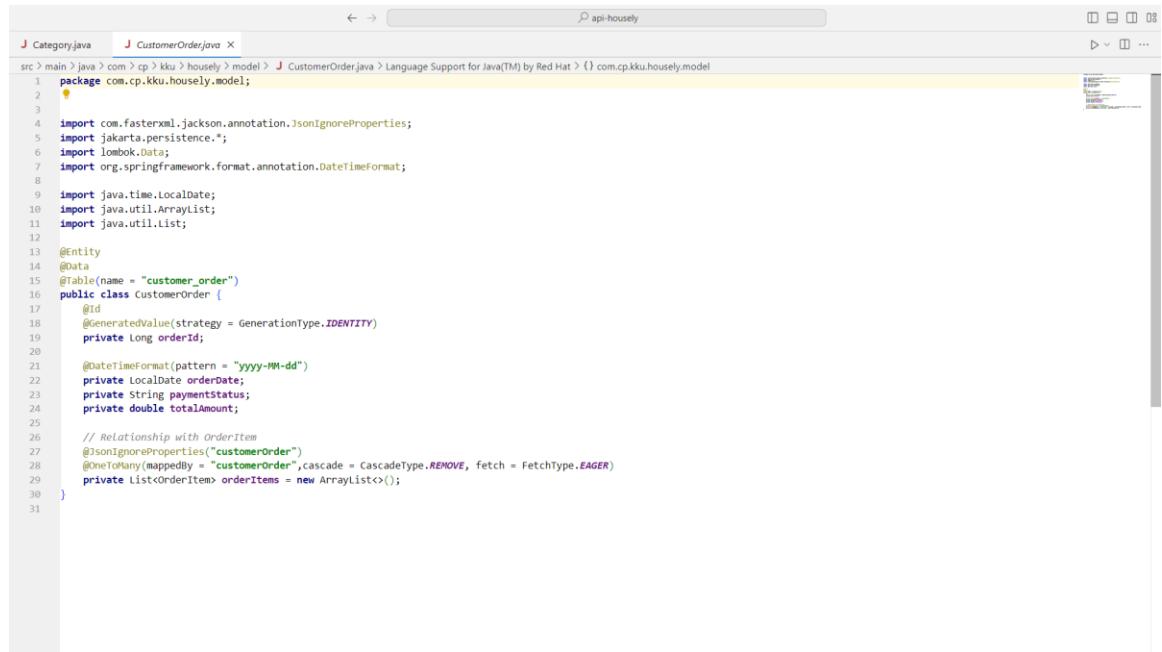
J Category.java M ×
src > main > java > com > cp > kku > housely > model > J Category.java > {} com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 > import java.util.ArrayList;
4
5 @Entity
6 @Data
7 @Table(name = "category")
8 public class Category {
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     @Column(name = "category_id")
13     private Long categoryId;
14
15     @Column(name = "category_name", nullable = false)
16     private String categoryName;
17
18     @Column(name = "description", length = 1000)
19     private String description;
20
21     @Column(name = "imageBase64")
22     private String imageBase64;
23
24     @ManyToMany(mappedBy = "categories", fetch = FetchType.LAZY)
25     @JsonIgnore
26     private List<Product> productsInCategory = new ArrayList<>();
27
28 }
29
30
31
32
33
34
35
36
37
38
39
40
41
42

```

ภาพที่ 50 ภาพ Category.java

- CustomerOrder.java

- เก็บข้อมูลคำสั่งซื้อ เช่น ID, วันที่สั่งซื้อ, สถานะการชำระเงิน, และยอดรวม
- มีความสัมพันธ์แบบ One-to-Many กับ OrderItem



```

J Category.java   J CustomerOrder.java X
src > main > java > com > cp > kku > housely > model > J CustomerOrder.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3
4 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
5 import jakarta.persistence.*;
6 import lombok.Data;
7 import org.springframework.format.annotation.DateTimeFormat;
8
9 import java.time.LocalDate;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 @Entity
14 @Data
15 @Table(name = "customer_order")
16 public class CustomerOrder {
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long orderId;
20
21     @DateTimeFormat(pattern = "yyyy-MM-dd")
22     private LocalDate orderDate;
23     private String paymentStatus;
24     private double totalAmount;
25
26     // Relationship with OrderItem
27     @JsonIgnoreProperties("customerOrder")
28     @OneToMany(mappedBy = "customerOrder", cascade = CascadeType.REMOVE, fetch = FetchType.EAGER)
29     private List<OrderItem> orderItems = new ArrayList<>();
30 }
31

```

ภาพที่ 51 ภาพ CustomerOrder.java

- OrderItem.java

- เก็บข้อมูลเกี่ยวกับรายการสั่งซื้อ เช่น ID ออเดอร์และ ID สินค้า
- มีความสัมพันธ์แบบ Many-to-One กับ CustomerOrder และ Product

```

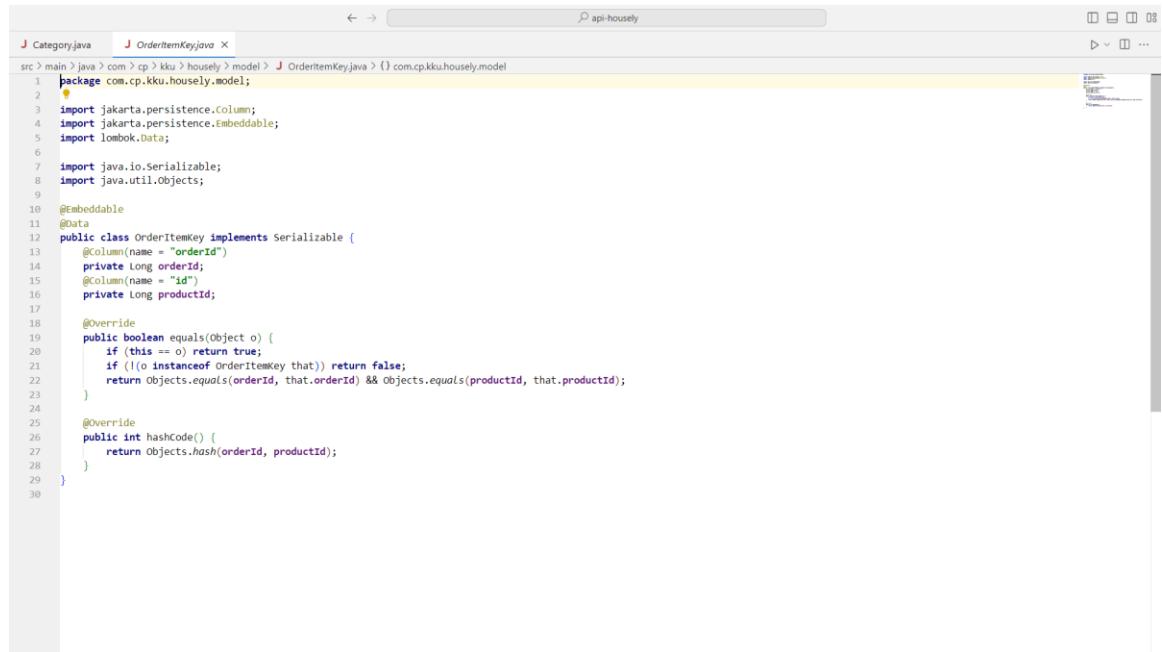
J Category.java   J OrderItem.java X
src > main > java > com > cp > kku > housely > model > J OrderItem.java > {} com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4 import jakarta.persistence.*;
5 import lombok.Data;
6
7 @Entity
8 @Data
9 @Table(name = "order_item")
10 @JsonIgnoreProperties("customerOrder")
11 public class OrderItem {
12     @EmbeddedId
13     private OrderItemKey orderItemId;
14     private int quantity;
15
16     // Relationship with CustomerOrder
17     @ManyToOne(fetch = FetchType.LAZY)
18     @MapId("orderId")
19     @JoinColumn(name = "order_id")
20     private CustomerOrder customerOrder;
21
22     // Relationship with Product
23     @ManyToOne(fetch = FetchType.EAGER)
24     @MapId("productId")
25     @JoinColumn(name = "product_id")
26     @JsonIgnoreProperties("orderItems")
27     private Product product;
28 }
29

```

ภาพที่ 52 ภาพ OrderItem.java

- OrderItemKey.java

OrderItemKey เป็นคลาสที่ใช้สร้าง **Composite Key** ใน JPA สำหรับ OrderItem โดยมีพิล็อต orderId และ productId เพื่อรับความเป็นเอกลักษณ์ มีเมธอด equals() และ hashCode() เพื่อเปรียบเทียบและสร้างค่า哈希สำหรับกุญแจหลักในฐานข้อมูล



```

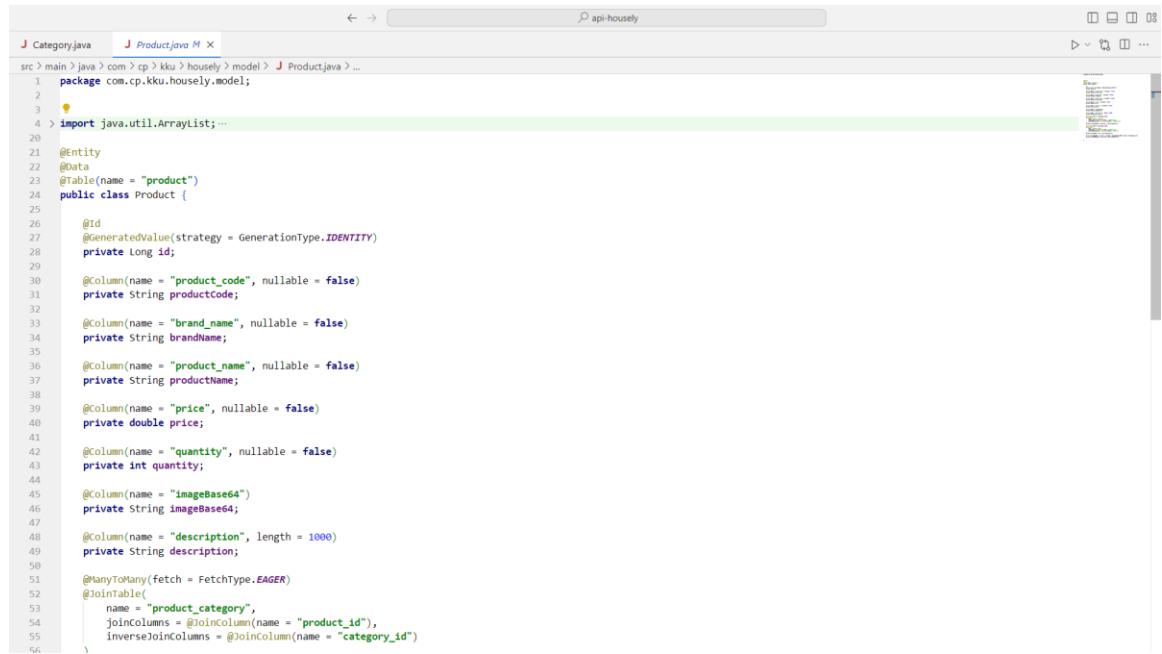
J Category.java J OrderItemKey.java X
src > main > java > com > cp > kku > housely > model > J OrderItemKey.java < com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Embeddable;
5 import lombok.Data;
6
7 import java.io.Serializable;
8 import java.util.Objects;
9
10 @Embeddable
11 @Data
12 public class OrderItemKey implements Serializable {
13     @Column(name = "orderId")
14     private Long orderId;
15     @Column(name = "id")
16     private Long productId;
17
18     @Override
19     public boolean equals(Object o) {
20         if (this == o) return true;
21         if (!(o instanceof OrderItemKey)) return false;
22         return Objects.equals(orderId, that.orderId) && Objects.equals(productId, that.productId);
23     }
24
25     @Override
26     public int hashCode() {
27         return Objects.hash(orderId, productId);
28     }
29 }
30

```

ภาพที่ 53 ภาพ OrderItemKey.java

- Product.java

- เก็บข้อมูลสินค้า เช่น ID, รหัสสินค้า, ชื่อแบรนด์, ชื่อสินค้า, ราคา, จำนวน, และรูปภาพ
- มีความสัมพันธ์แบบ Many-to-Many กับ Category และ Room
- มีความสัมพันธ์แบบ One-to-Many กับ OrderItem

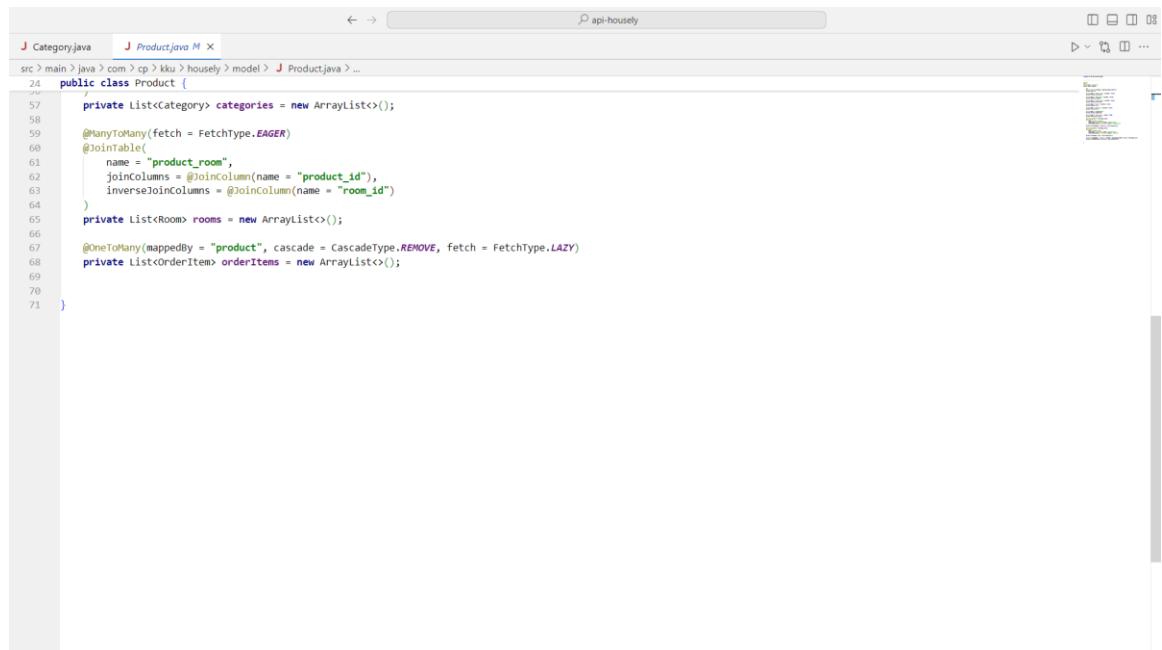


```

J Category.java   J Product.java M X
src > main > java > com > cp > kku > housely > model > J Product.java > ...
1 package com.cp.kku.housely.model;
2
3
4 > import java.util.ArrayList;...
5
6
7 @Entity
8 @Data
9 @Table(name = "product")
10 public class Product {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private long id;
15
16     @Column(name = "product_code", nullable = false)
17     private String productCode;
18
19     @Column(name = "brand_name", nullable = false)
20     private String brandName;
21
22     @Column(name = "product_name", nullable = false)
23     private String productName;
24
25     @Column(name = "price", nullable = false)
26     private double price;
27
28     @Column(name = "quantity", nullable = false)
29     private int quantity;
30
31     @Column(name = "imageBase64")
32     private String imageBase64;
33
34     @Column(name = "description", length = 1000)
35     private String description;
36
37     @ManyToMany(fetch = FetchType.EAGER)
38     @JoinTable(
39         name = "product_category",
40         joinColumns = @JoinColumn(name = "product_id"),
41         inverseJoinColumns = @JoinColumn(name = "category_id")
42     )
43 }

```

ภาพที่ 54 ภาพ Product.java



```

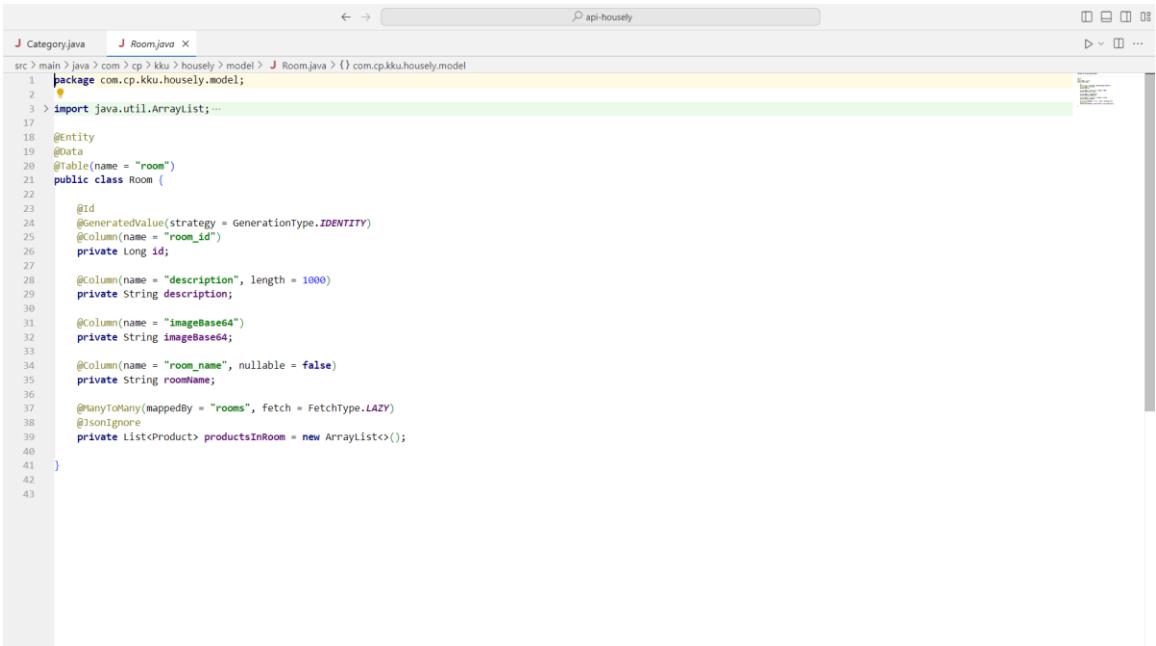
J Category.java   J Product.java M X
src > main > java > com > cp > kku > housely > model > J Product.java > ...
24 public class Product {
25
26     private List<Category> categories = new ArrayList<>();
27
28     @ManyToMany(fetch = FetchType.EAGER)
29     @JoinTable(
30         name = "product_room",
31         joinColumns = @JoinColumn(name = "product_id"),
32         inverseJoinColumns = @JoinColumn(name = "room_id")
33     )
34     private List<Room> rooms = new ArrayList<>();
35
36     @OneToMany(mappedBy = "product", cascade = CascadeType.REMOVE, fetch = FetchType.LAZY)
37     private List<OrderItem> orderItems = new ArrayList<>();
38
39
40 }

```

ภาพที่ 54 ภาพ Product.java (ต่อ)

- Room.java

- เก็บข้อมูลห้อง เช่น ID, รายละเอียด, ชื่อห้อง, และรูปภาพ
- มีความสัมพันธ์แบบ Many-to-Many กับ Product



```

J Category.java J Room.java X
src > main > java > com > cp > kku > housely > model > J Room.java < com.cp.kku.housely.model
1 package com.cp.kku.housely.model;
2
3 > import java.util.ArrayList;...
17
18 @Entity
19 @Data
20 @Table(name = "room")
21 public class Room {
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.IDENTITY)
25     @Column(name = "room_id")
26     private Long id;
27
28     @Column(name = "description", length = 1000)
29     private String description;
30
31     @Column(name = "imageBase64")
32     private String imageBase64;
33
34     @Column(name = "room_name", nullable = false)
35     private String roomName;
36
37     @ManyToMany(mappedBy = "rooms", fetch = FetchType.LAZY)
38     @JsonIgnore
39     private List<Product> productsInRoom = new ArrayList<>();
40
41 }
42
43

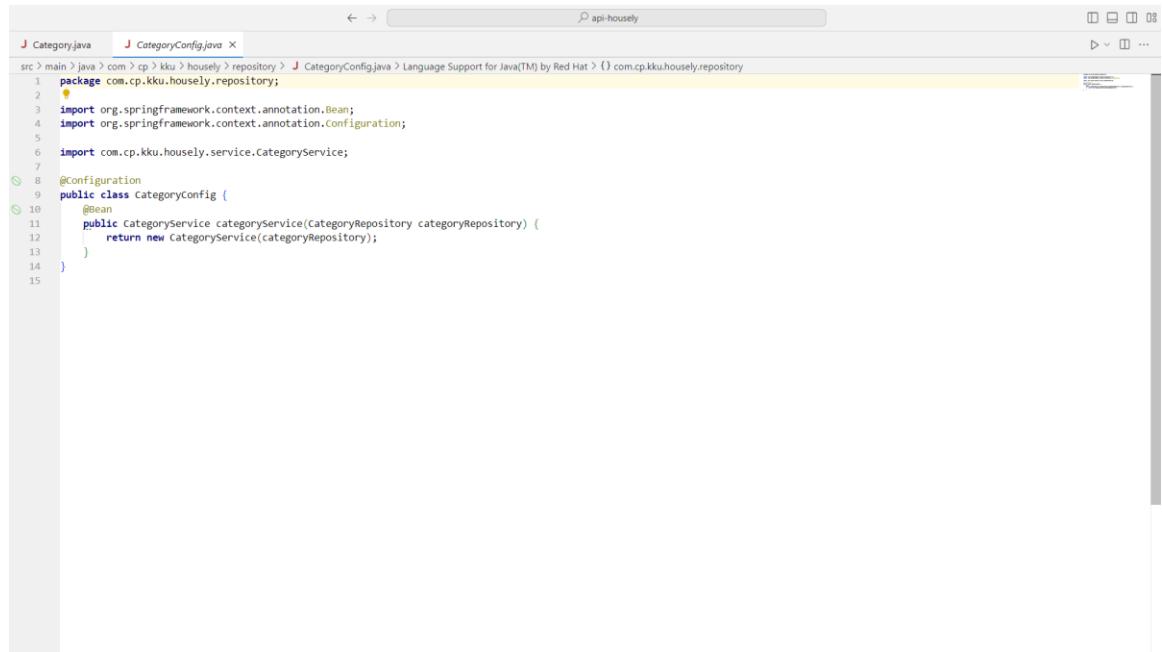
```

ภาพที่ 55 ภาพ Room.java

Repository

- CategoryConfig.java

คลาสคอนฟิกที่สร้างบริการ (Service) สำหรับโพซิทอรีต่างๆ โดยใช้ @Bean.



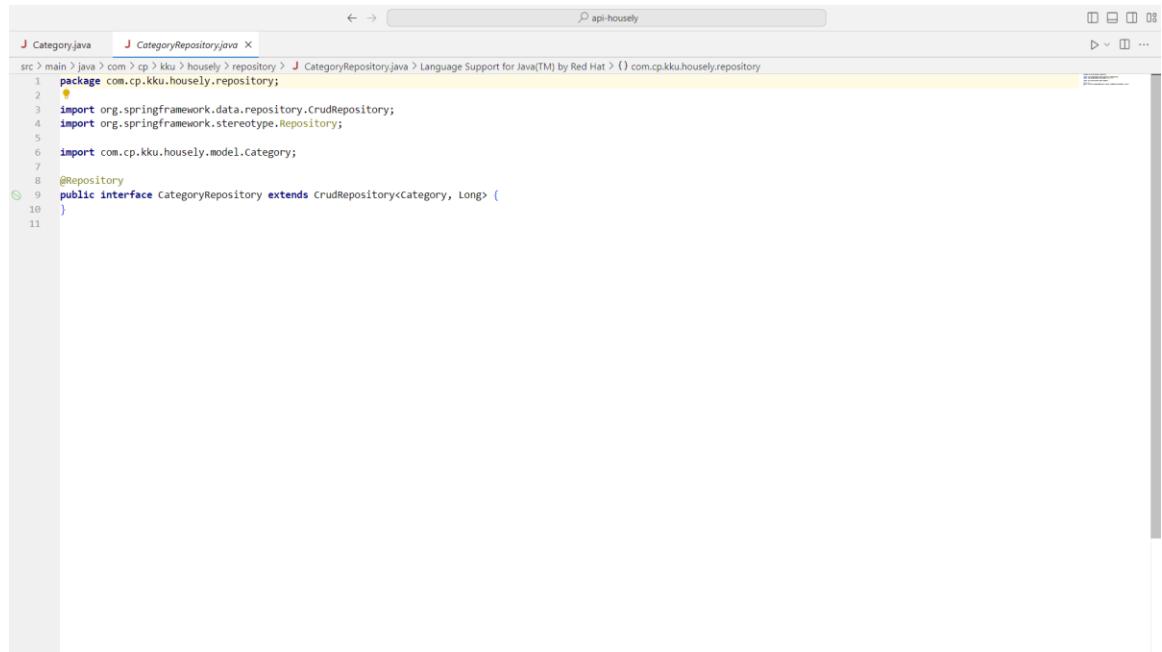
The screenshot shows a Java code editor window titled "api-housely". The current file is "CategoryConfig.java" located at "src > main > java > com > cp > kku > housely > repository". The code defines a configuration class for a CategoryService:

```
1 package com.cp.kku.housely.repository;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 import com.cp.kku.housely.service.CategoryService;
7
8 @Configuration
9 public class CategoryConfig {
10     @Bean
11     public CategoryService categoryService(CategoryRepository categoryRepository) {
12         return new CategoryService(categoryRepository);
13     }
14 }
15
```

ภาพที่ 56 ภาพ CategoryConfig.java

- CategoryRepository.java

เป็นอินเทอร์เฟซที่สืบทอดจาก CrudRepository สำหรับจัดการข้อมูลหมวดหมู่ (Category) ในฐานข้อมูล โดยมีฟังก์ชันพื้นฐานสำหรับการสร้าง อ่าน ปรับปรุง และลบ (CRUD) หมวดหมู่ในโปรเจกต์ JPA



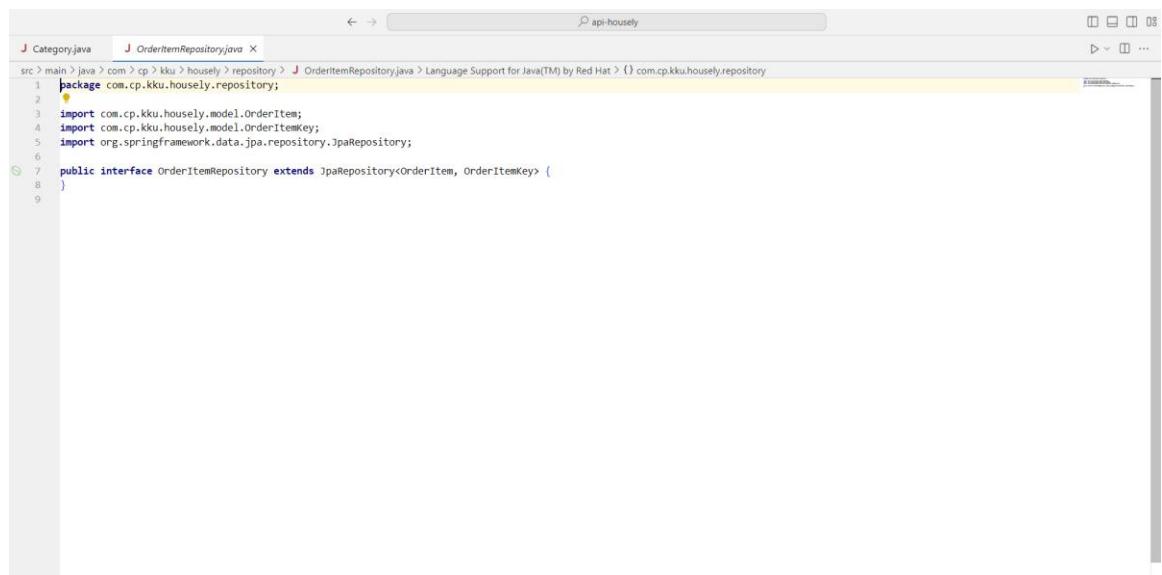
```

J Category.java   J CategoryRepository.java X
src > main > java > com > cp > kku > housely > repository > J CategoryRepository.java > Language Support for Java(TM) by Red Hat > (} com.cp.kku.housely.repository
1 package com.cp.kku.housely.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.cp.kku.housely.model.Category;
7
8 @Repository
9 public interface CategoryRepository extends CrudRepository<Category, Long> {
10 }
11

```

ภาพที่ 57 ภาพ CategoryRepository.java

- OrderItemRepository.java
- สำหรับจัดการรายการในคำสั่งซื้อ โดยใช้คีย์ที่ประกอบกัน (OrderItemKey)



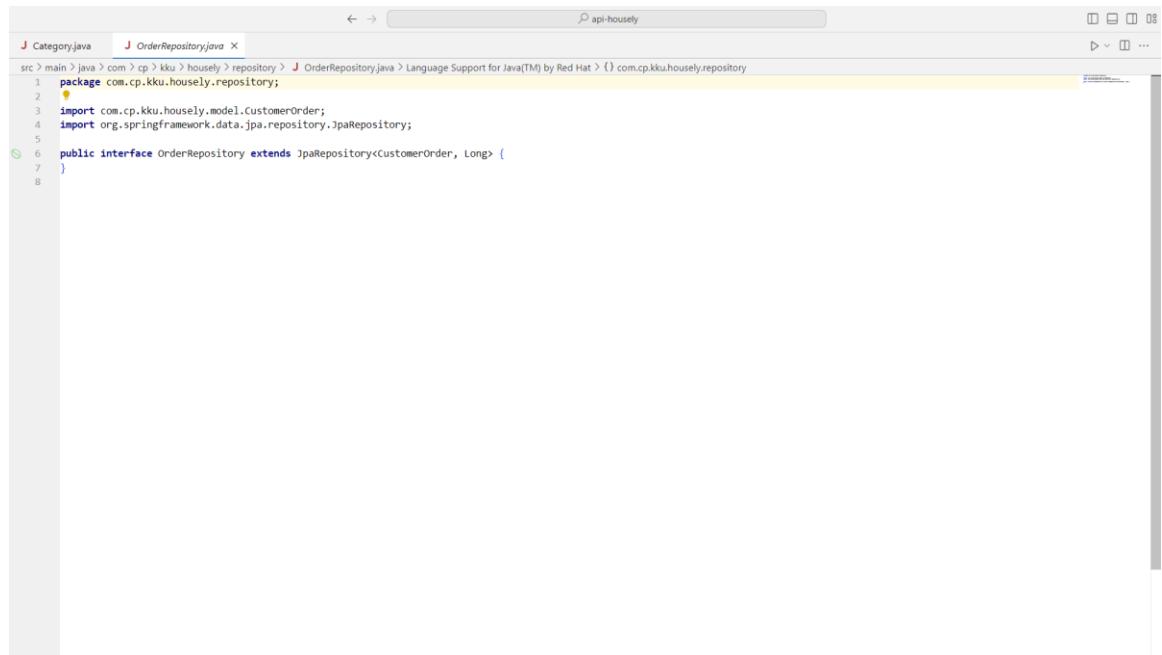
```

J Category.java   J OrderItemRepository.java X
src > main > java > com > cp > kku > housely > repository > J OrderItemRepository.java > Language Support for Java(TM) by Red Hat > (} com.cp.kku.housely.repository
1 package com.cp.kku.housely.repository;
2
3 import com.cp.kku.housely.model.OrderItem;
4 import com.cp.kku.housely.model.OrderItemKey;
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 public interface OrderItemRepository extends JpaRepository<OrderItem, OrderItemKey> {
8 }
9

```

ภาพที่ 58 ภาพ OrderItemRepository.java

- OrderRepository.java
สำหรับจัดการคำสั่งซื้อ (CustomerOrder)

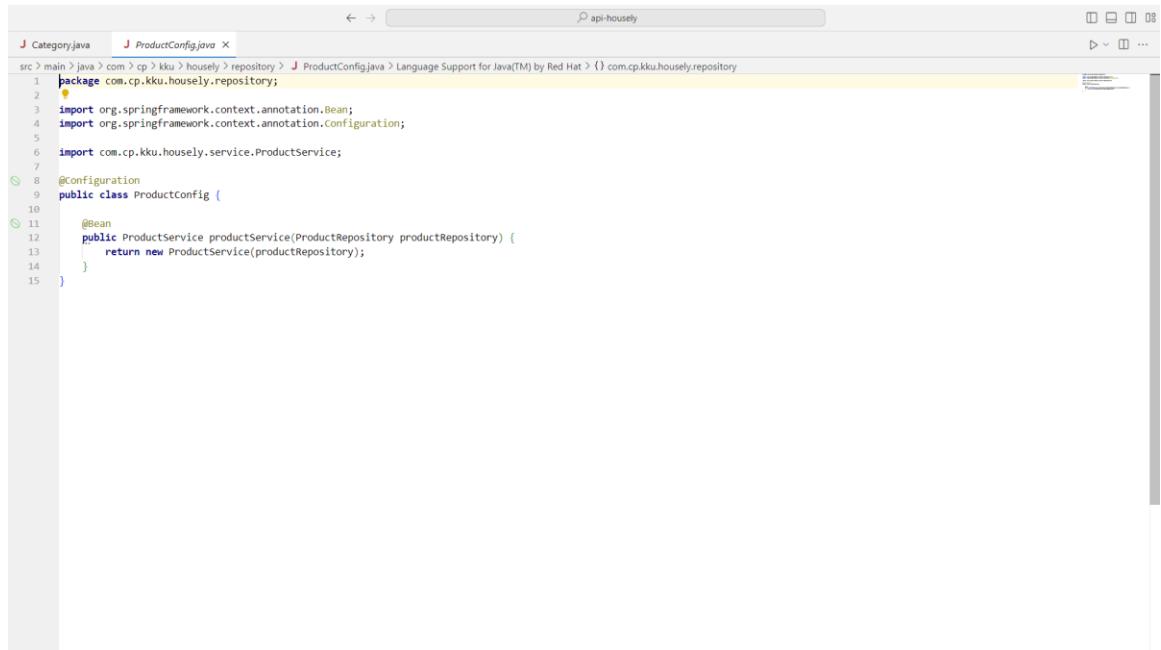


The screenshot shows a Java code editor window with the title bar "api-housely". The left sidebar lists two files: "Category.java" and "OrderRepository.java". The "OrderRepository.java" file is currently open and displayed in the main editor area. The code is as follows:

```
1 package com.cp.kku.housely.repository;
2
3 import com.cp.kku.housely.model.CustomerOrder;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface OrderRepository extends JpaRepository<CustomerOrder, Long> {
7 }
```

ภาพที่ 59 ภาพ OrderRepository.java

- ProductConfig.java
คลาสคอนฟิกที่สร้างบริการ (Service) สำหรับรีโพซิทอรีต่างๆ โดยใช้ @Bean.



```

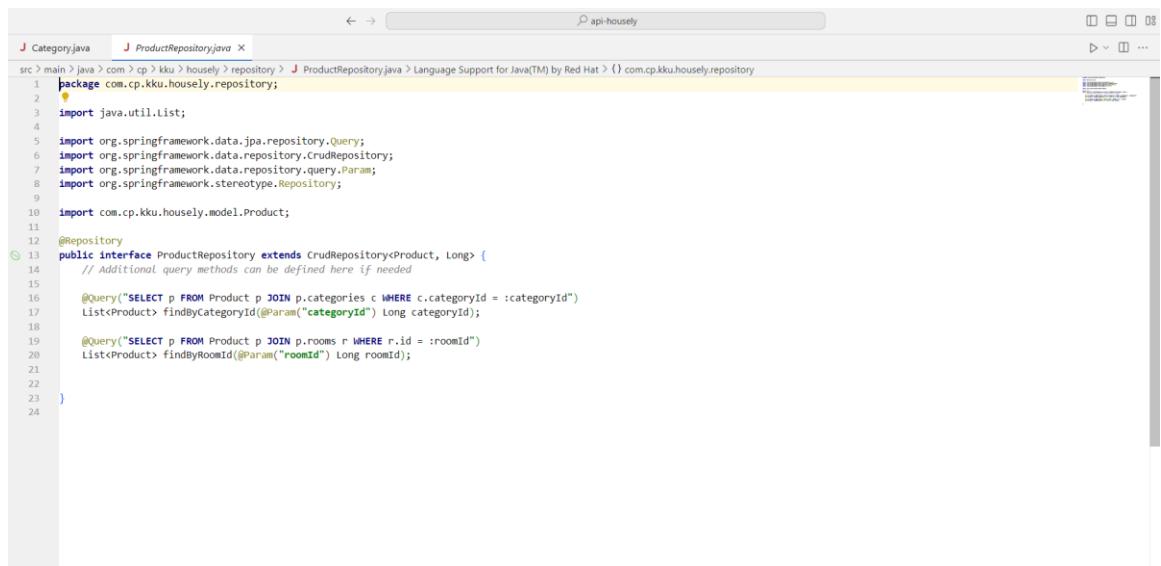
J Category.java J ProductConfig.java ×
src > main > java > com > cp > kku > housely > repository > J ProductConfig.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.repository
1 package com.cp.kku.housely.repository;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 import com.cp.kku.housely.service.ProductService;
7
8 @Configuration
9 public class ProductConfig {
10
11     @Bean
12     public ProductService productService(ProductRepository productRepository) {
13         return new ProductService(productRepository);
14     }
15 }

```

ภาพที่ 60 ภาพ ProductConfig.java

- ProductRepository.java

จัดการข้อมูลผลิตภัณฑ์ (Product) พร้อมวิธีค้นหาตาม categoryId และ roomId.



```

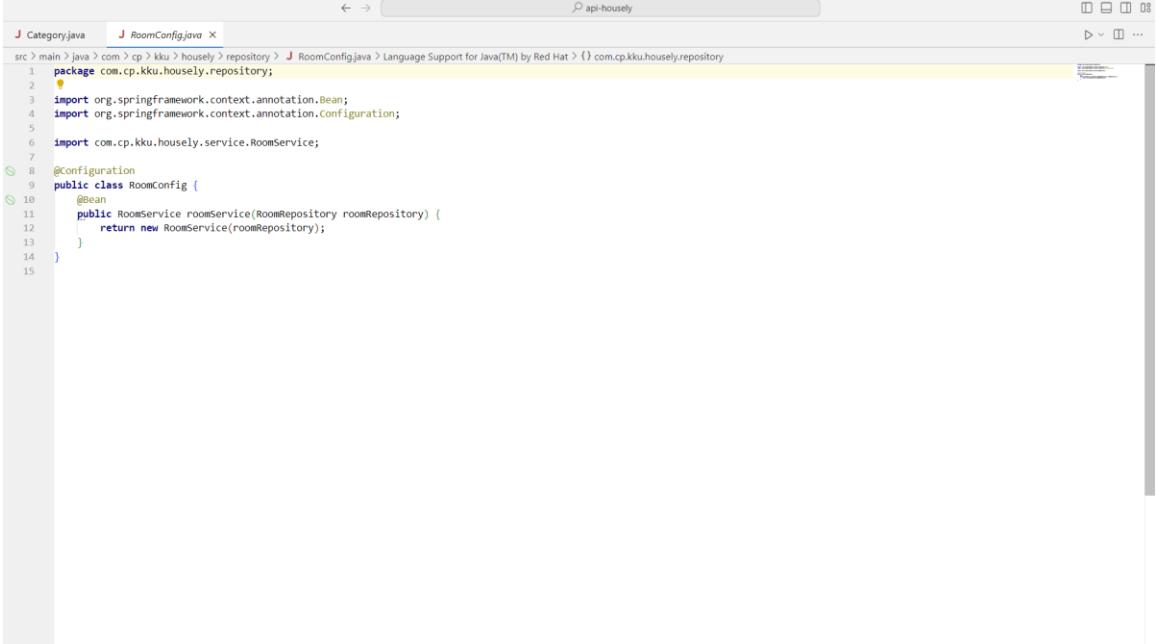
J Category.java J ProductRepository.java ×
src > main > java > com > cp > kku > housely > repository > J ProductRepository.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.repository
1 package com.cp.kku.housely.repository;
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.data.repository.CrudRepository;
7 import org.springframework.data.repository.query.Param;
8 import org.springframework.stereotype.Repository;
9
10 import com.cp.kku.housely.model.Product;
11
12 @Repository
13 public interface ProductRepository extends CrudRepository<Product, Long> {
14     // Additional query methods can be defined here if needed
15
16     @Query("SELECT p FROM Product p JOIN p.categories c WHERE c.categoryId = :categoryId")
17     List<Product> findByCategoryId(@Param("categoryId") Long categoryId);
18
19     @Query("SELECT p FROM Product p JOIN p.rooms r WHERE r.id = :roomId")
20     List<Product> findByRoomId(@Param("roomId") Long roomId);
21
22
23 }

```

ภาพที่ 61 ภาพ ProductRepository.java

- RoomConfig.java

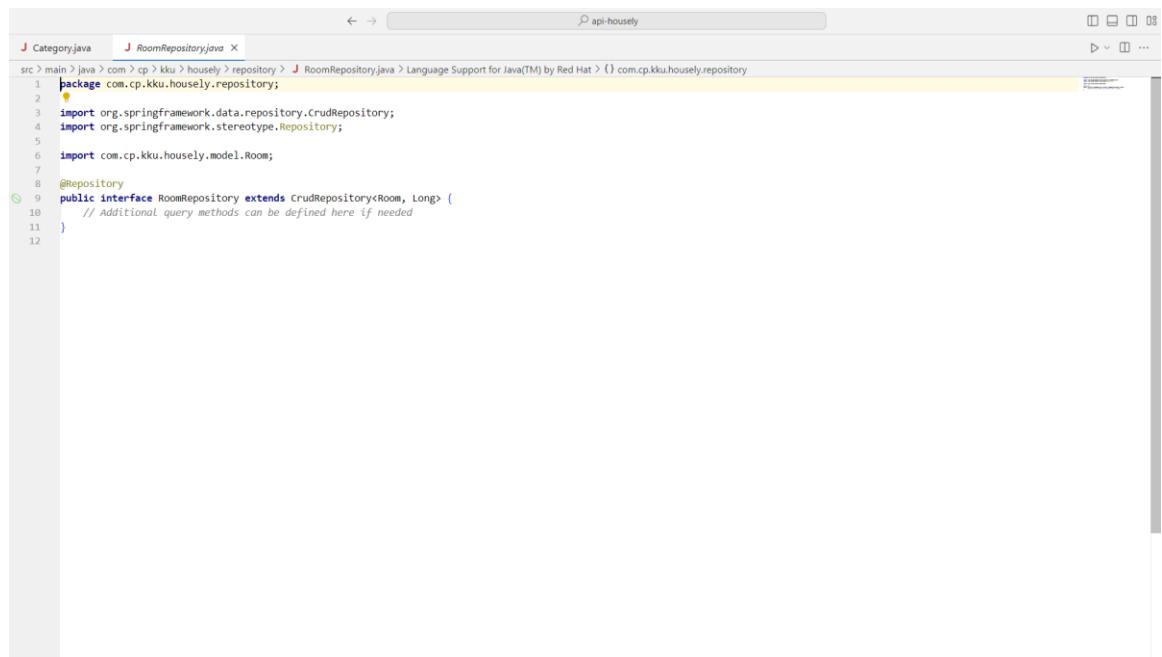
คลาสคอนฟิกที่สร้างบริการ (Service) สำหรับเพชิทธิ์ต่างๆ โดยใช้ @Bean.



```
src > main > java > com > cp > kku > housely > repository > RoomConfig.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.repository
1 package com.cp.kku.housely.repository;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 import com.cp.kku.housely.service.RoomService;
7
8 @Configuration
9 public class RoomConfig {
10     @Bean
11     public RoomService roomService(RoomRepository roomRepository) {
12         return new RoomService(roomRepository);
13     }
14 }
15
```

ภาพที่ 62 ภาพ RoomConfig.java

- RoomRepository.java
- ใช้สำหรับจัดการข้อมูลห้อง (Room)



The screenshot shows a Java code editor with the tab 'RoomRepository.java' selected. The code defines a Spring Data repository interface for managing rooms. The interface extends the 'CrudRepository' class from the 'org.springframework.data.repository' package, specifically for the 'Room' entity. It includes methods for saving, finding by ID, and deleting rooms.

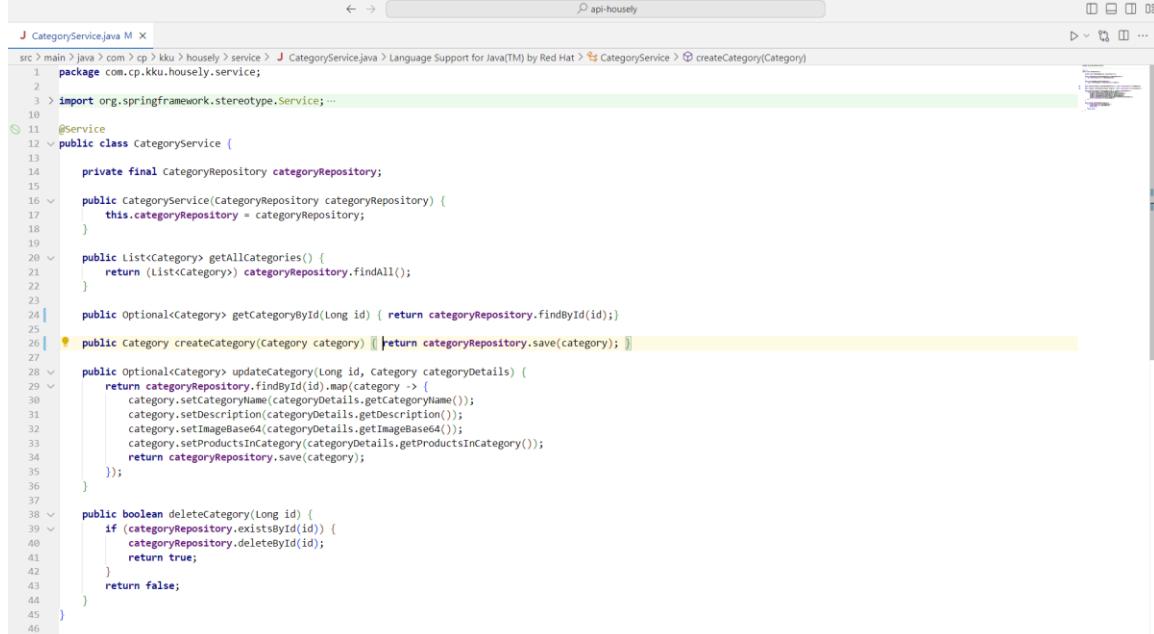
```
src > main > java > com > cp > kku > housely > repository > RoomRepository.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.repository  
1 package com.cp.kku.housely.repository;  
2  
3 import org.springframework.data.repository.CrudRepository;  
4 import org.springframework.stereotype.Repository;  
5  
6 import com.cp.kku.housely.model.Room;  
7  
8 @Repository  
9 public interface RoomRepository extends CrudRepository<Room, Long> {  
10     // Additional query methods can be defined here if needed  
11 }  
12
```

ภาพที่ 63 ภาพ RoomRepository.java

Service

- CategoryService.java

จัดการหมวดหมู่ มีฟังก์ชันเพื่อดึงหมวดหมู่ทั้งหมด, ค้นหาหมวดหมู่ตาม ID, สร้าง, อัปเดต,
และลบหมวดหมู่



```

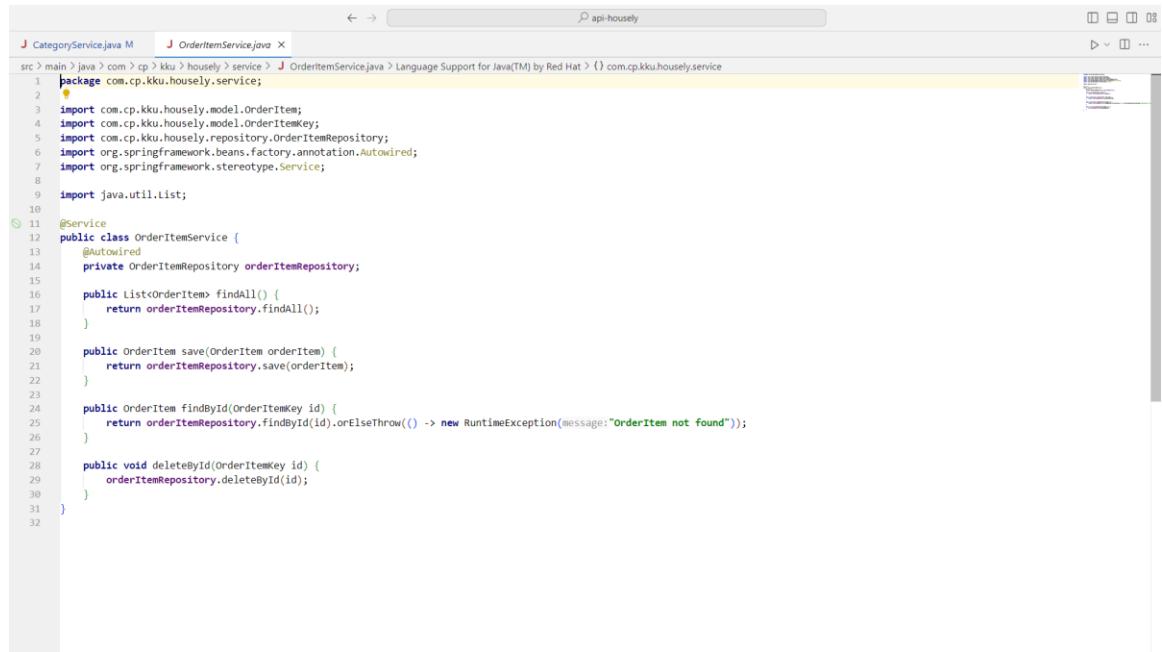
J CategoryService.java M ×
src > main > java > com > cp > kku > housesly > service > J CategoryService.java > Language Support for Java(TM) by Red Hat > CategoryService > createCategory(Category)
1 package com.cp.kku.housesly.service;
2
3 > import org.springframework.stereotype.Service;
4
5 @Service
6 public class CategoryService {
7
8     private final CategoryRepository categoryRepository;
9
10    public CategoryService(CategoryRepository categoryRepository) {
11        this.categoryRepository = categoryRepository;
12    }
13
14    public List<Category> getAllCategories() {
15        return (List<Category>) categoryRepository.findAll();
16    }
17
18    public Optional<Category> getCategoryById(Long id) { return categoryRepository.findById(id); }
19
20    public Category createCategory(Category category) { return categoryRepository.save(category); }
21
22    public Optional<Category> updateCategory(Long id, Category categoryDetails) {
23        return categoryRepository.findById(id).map(category -> {
24            category.setCategoryName(categoryDetails.getCategoryName());
25            category.setDescription(categoryDetails.getDescription());
26            category.setImageBase64(categoryDetails.getImageBase64());
27            category.setProductsInCategory(categoryDetails.getProductsInCategory());
28            return categoryRepository.save(category);
29        });
30    }
31
32    public boolean deleteCategory(Long id) {
33        if (categoryRepository.existsById(id)) {
34            categoryRepository.deleteById(id);
35            return true;
36        }
37        return false;
38    }
39}
40
41
42
43
44
45
46

```

ภาพที่ 64 ภาพ CategoryService.java

- OrderItemService.java

จัดการรายการในคำสั่งซื้อ มีฟังก์ชันเพื่อดึงรายการทั้งหมด, บันทึกรายการ, ค้นหารายการตาม ID,
และลบรายการ



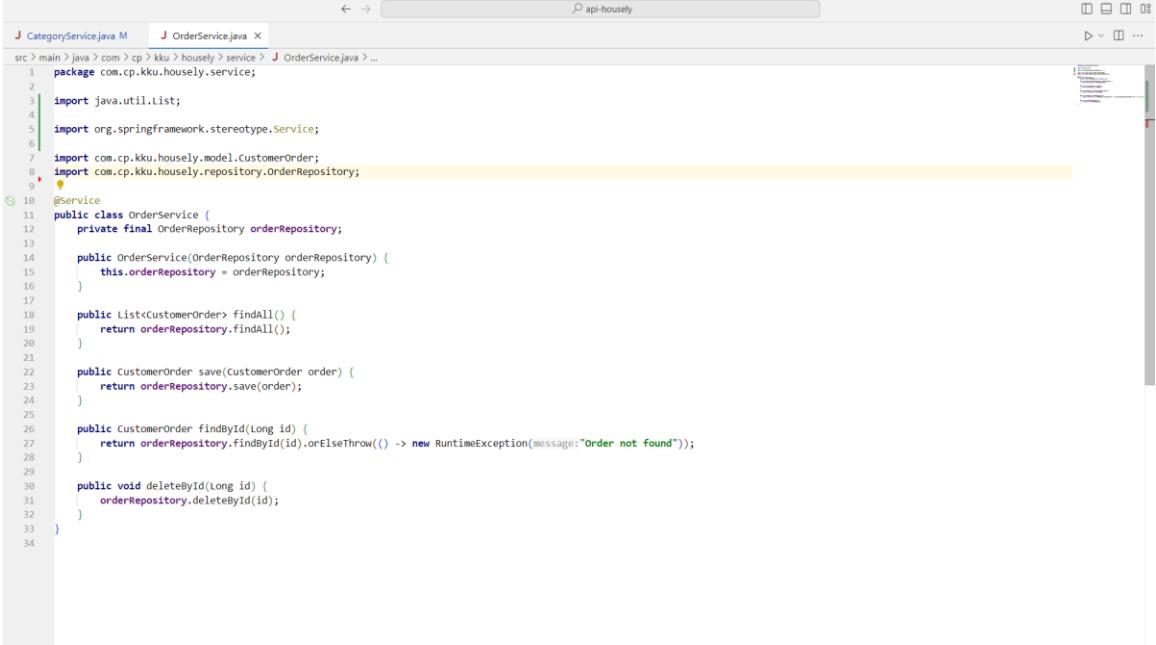
The screenshot shows a Java code editor within an IDE. The title bar indicates the project is named 'api-housely'. The code editor displays the 'OrderItemService.java' file. The code is as follows:

```
1 package com.cp.kku.housely.service;
2
3 import com.cp.kku.housely.model.OrderItem;
4 import com.cp.kku.housely.model.OrderItemKey;
5 import com.cp.kku.housely.repository.OrderItemRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import java.util.List;
10
11 @Service
12 public class OrderItemService {
13     @Autowired
14     private OrderItemRepository orderItemRepository;
15
16     public List<OrderItem> findAll() {
17         return orderItemRepository.findAll();
18     }
19
20     public OrderItem save(OrderItem orderItem) {
21         return orderItemRepository.save(orderItem);
22     }
23
24     public OrderItem findById(OrderItemKey id) {
25         return orderItemRepository.findById(id).orElseThrow(() -> new RuntimeException(message:"OrderItem not found"));
26     }
27
28     public void deleteById(OrderItemKey id) {
29         orderItemRepository.deleteById(id);
30     }
31 }
32
```

ภาพที่ 65 ภาพ OrderItemService.java

- OrderService.java

จัดการคำสั่งซื้อ มีฟังก์ชันเพื่อดึงคำสั่งซื้อทั้งหมด, บันทึกคำสั่งซื้อ, ค้นหาคำสั่งซื้อตาม ID, และลบคำสั่งซื้อ



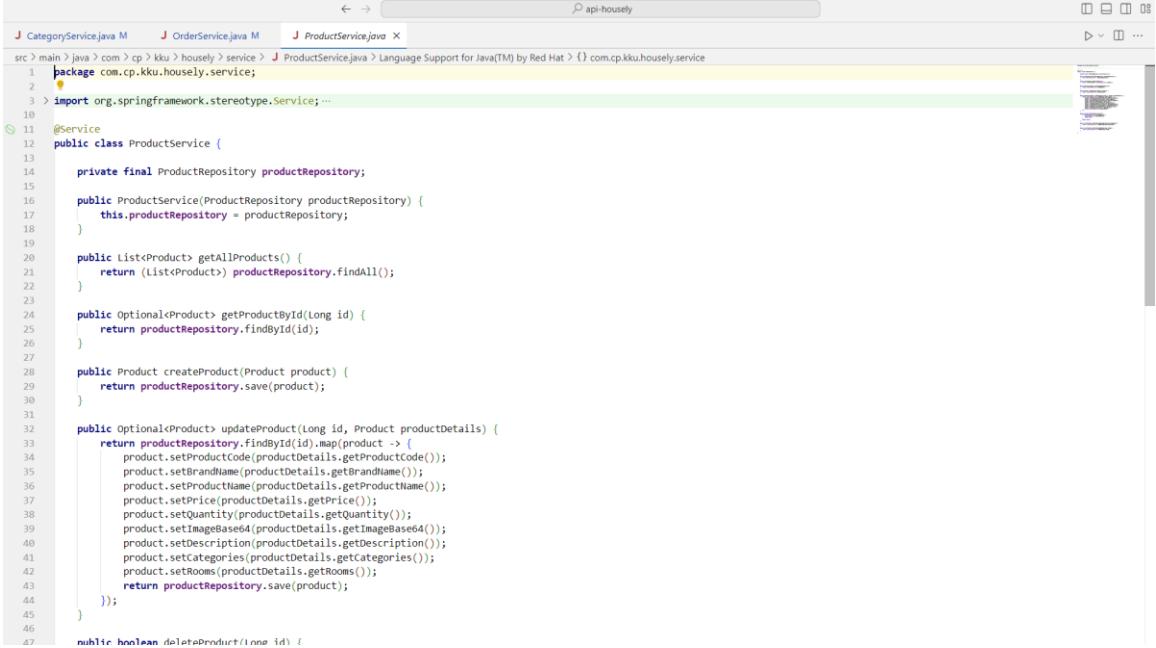
The screenshot shows a Java code editor with the file `OrderService.java` open. The code implements a service layer for managing orders. It includes imports for `List`, `CustomerOrder`, and `OrderRepository`. The class is annotated with `@Service`. It contains methods for finding all orders, saving a new order, finding an order by ID, and deleting an order by ID. The code uses `orElseThrow` to handle cases where an order might not be found.

```
1 package com.cp.kku.houseley.service;
2
3 import java.util.List;
4
5 import org.springframework.stereotype.Service;
6
7 import com.cp.kku.houseley.model.CustomerOrder;
8 import com.cp.kku.houseley.repository.OrderRepository;
9
10 @Service
11 public class OrderService {
12     private final OrderRepository orderRepository;
13
14     public OrderService(OrderRepository orderRepository) {
15         this.orderRepository = orderRepository;
16     }
17
18     public List<CustomerOrder> findAll() {
19         return orderRepository.findAll();
20     }
21
22     public CustomerOrder save(CustomerOrder order) {
23         return orderRepository.save(order);
24     }
25
26     public CustomerOrder findById(Long id) {
27         return orderRepository.findById(id).orElseThrow(() -> new RuntimeException(message:"Order not found"));
28     }
29
30     public void deleteById(Long id) {
31         orderRepository.deleteById(id);
32     }
33 }
34
```

ภาพที่ 66 ภาพ OrderService.java

- ProductService.java

จัดการผลิตภัณฑ์ มีฟังก์ชันเพื่อดึงผลิตภัณฑ์ทั้งหมด, ค้นหาผลิตภัณฑ์ตาม ID, สร้าง, อัปเดต, ลบผลิตภัณฑ์ และค้นหาผลิตภัณฑ์ตามหมวดหมู่หรือห้อง

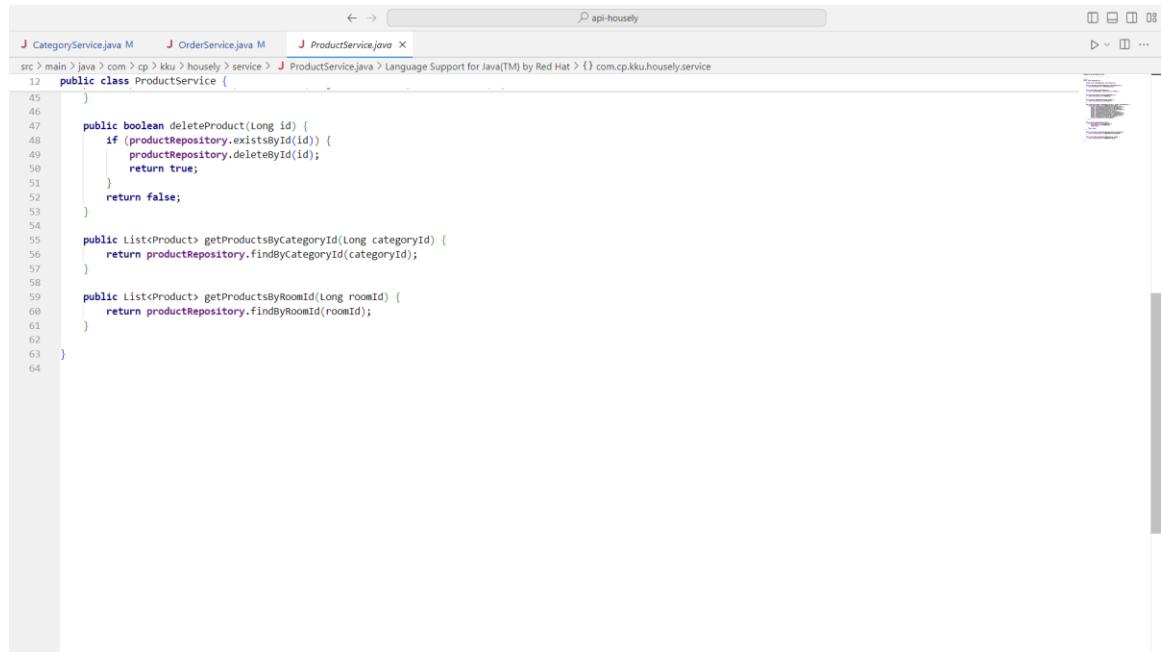


```

J CategoryService.java M J OrderService.java M J ProductService.java X
src > main > java > com > cp > kku > housely > service > J ProductService.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.service
1 package com.cp.kku.housely.service;
2
3 > import org.springframework.stereotype.Service;
4
5 @Service
6 public class ProductService {
7
8     private final ProductRepository productRepository;
9
10    public ProductService(ProductRepository productRepository) {
11        this.productRepository = productRepository;
12    }
13
14    public List<Product> getAllProducts() {
15        return (List<Product>) productRepository.findAll();
16    }
17
18    public Optional<Product> getProductById(Long id) {
19        return productRepository.findById(id);
20    }
21
22    public Product createProduct(Product product) {
23        return productRepository.save(product);
24    }
25
26    public Optional<Product> updateProduct(Long id, Product productDetails) {
27        return productRepository.findById(id).map(product -> {
28            product.setProductCode(productDetails.getProductCode());
29            product.setBrandName(productDetails.getBrandName());
30            product.setProductName(productDetails.getProductName());
31            product.setPrice(productDetails.getPrice());
32            product.setQuantity(productDetails.getQuantity());
33            product.setImageBase64(productDetails.getImageBase64());
34            product.setDescription(productDetails.getDescription());
35            product.setCategories(productDetails.getCategories());
36            product.setRooms(productDetails.getRooms());
37            return productRepository.save(product);
38        });
39    }
40
41    public boolean deleteProduct(Long id) {
42
43    }
44}
45
46
47

```

ภาพที่ 67 ภาพ ProductService.java

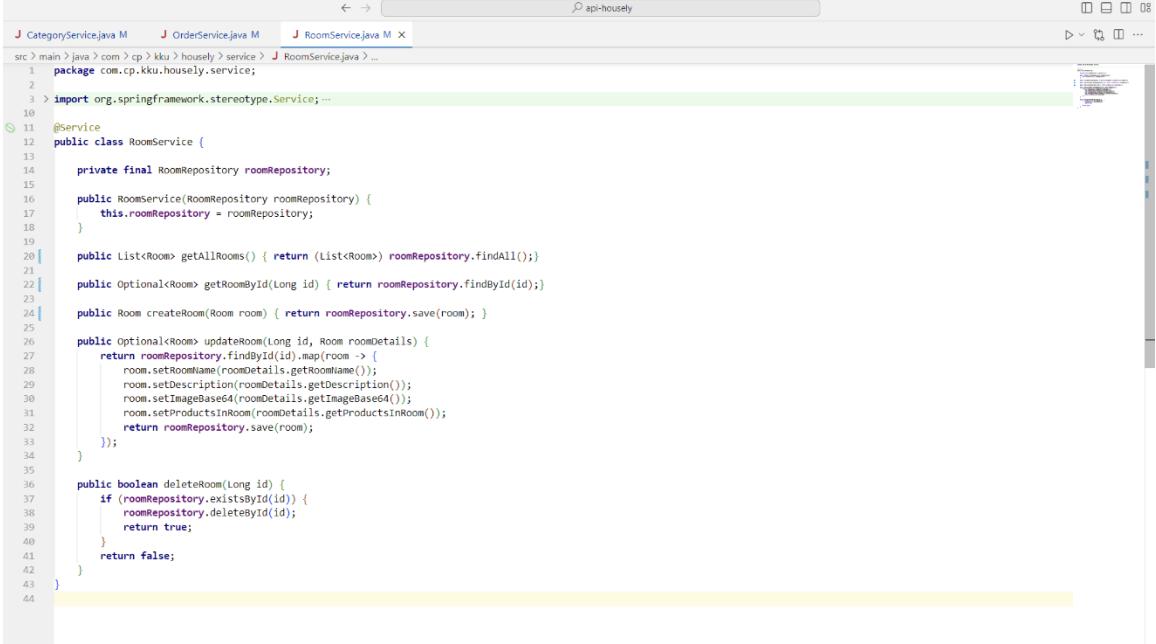


```
CategoryService.java M OrderService.java M ProductService.java X
src > main > java > com > cp > kku > housely > service > ProductService.java > Language Support for Java(TM) by Red Hat > {} com.cp.kku.housely.service
12 public class ProductService {
13
14     ...
15
16     public boolean deleteProduct(Long id) {
17         if (productRepository.existsById(id)) {
18             productRepository.deleteById(id);
19             return true;
20         }
21         return false;
22     }
23
24     public List<Product> getProductsByCategoryId(Long categoryId) {
25         return productRepository.findByCategoryId(categoryId);
26     }
27
28     public List<Product> getProductsByRoomId(Long roomId) {
29         return productRepository.findByRoomId(roomId);
30     }
31
32 }
```

ภาพที่ 67 ภาพ ProductService.java (ต่อ)

- RoomService.java

จัดการห้องพัก มีฟังก์ชันเพื่อดึงห้องทั้งหมด, ค้นหาห้องตาม ID, สร้าง, อัปเดต และลบห้อง



```

J CategoryService.java M J OrderService.java M J RoomService.java M X
src > main > java > com > cp > kku > housely > service > J RoomService.java ...
1 package com.cp.kku.housely.service;
2
3 > import org.springframework.stereotype.Service;
4
5 @Service
6 public class RoomService {
7
8     private final RoomRepository roomRepository;
9
10    public RoomService(RoomRepository roomRepository) {
11        this.roomRepository = roomRepository;
12    }
13
14    public List<Room> getAllRooms() { return (List<Room>) roomRepository.findAll(); }
15
16    public Optional<Room> getRoomById(Long id) { return roomRepository.findById(id); }
17
18    public Room createRoom(Room room) { return roomRepository.save(room); }
19
20    public Optional<Room> updateRoom(Long id, Room roomDetails) {
21        return roomRepository.findById(id).map(room -> {
22            room.setRoomName(roomDetails.getRoomName());
23            room.setDescription(roomDetails.getDescription());
24            room.setImageBase64(roomDetails.getImageBase64());
25            room.setProductsInRoom(roomDetails.getProductsInRoom());
26            return roomRepository.save(room);
27        });
28    }
29
30    public boolean deleteRoom(Long id) {
31        if (roomRepository.existsById(id)) {
32            roomRepository.deleteById(id);
33            return true;
34        }
35        return false;
36    }
37
38}
39
40
41
42
43
44

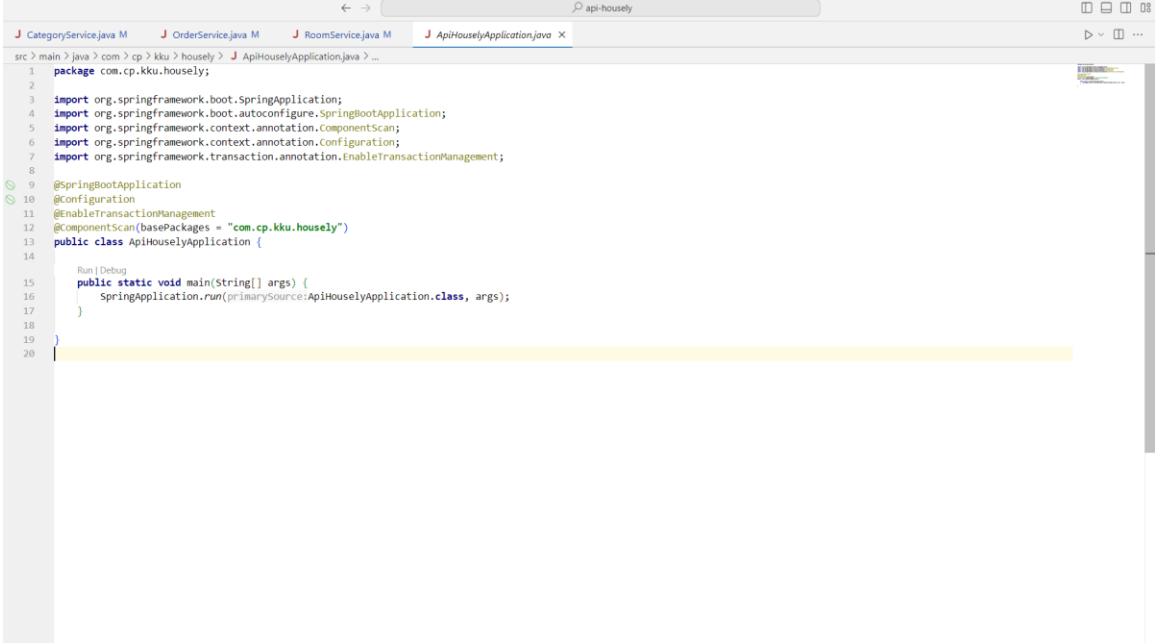
```

ภาพที่ 68 ภาพ RoomService.java

ApiHouselyApplication.java

คลาสหลักสำหรับแอปพลิเคชัน Spring Boot:

- **@SpringBootApplication:** กำหนดว่าเป็นแอปพลิเคชัน Spring Boot
- **@Configuration:** ระบุว่าเป็นคลาสคอนฟิก
- **@EnableTransactionManagement:** เปิดใช้งานการจัดการธุรกรรม
- **@ComponentScan:** ระบุแพ็คเกจที่ Spring จะค้นหาคอมโพเนนต์
- ฟังก์ชัน main จะเริ่มต้นแอปพลิเคชัน



The screenshot shows a Java code editor with the file `ApiHouselyApplication.java` open. The code is a Spring Boot application configuration. It includes imports for `SpringApplication`, `SpringBootConfiguration`, `EnableTransactionManagement`, and `ComponentScan`. The class `ApiHouselyApplication` is annotated with `@SpringBootApplication`, `@Configuration`, and `@EnableTransactionManagement`. It has a constructor with `@ComponentScan` and a static `main` method that runs the application.

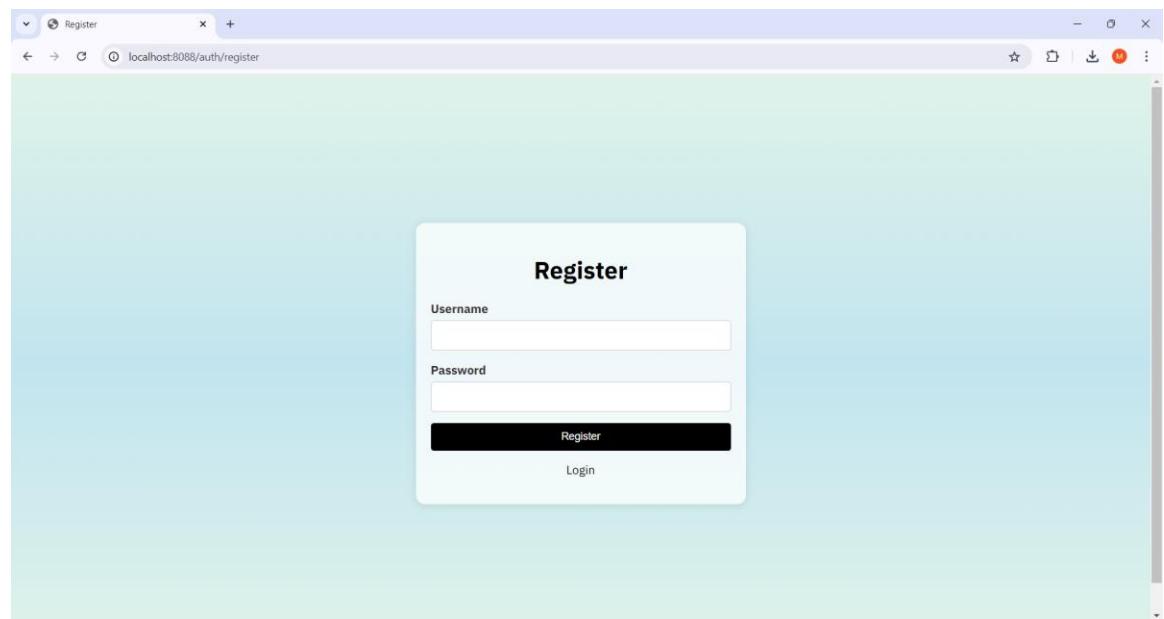
```
1 package com.cp.kku.housely;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.ComponentScan;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.transaction.annotation.EnableTransactionManagement;
8
9 @SpringBootApplication
10 @Configuration
11 @EnableTransactionManagement
12 @ComponentScan(basePackages = "com.cp.kku.housely")
13 public class ApiHouselyApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(ApiHouselyApplication.class, args);
17     }
18
19 }
20
```

ภาพที่ 69 ไฟล์ ApiHouselyApplication.java

4.2 ผลการทดสอบโปรแกรม

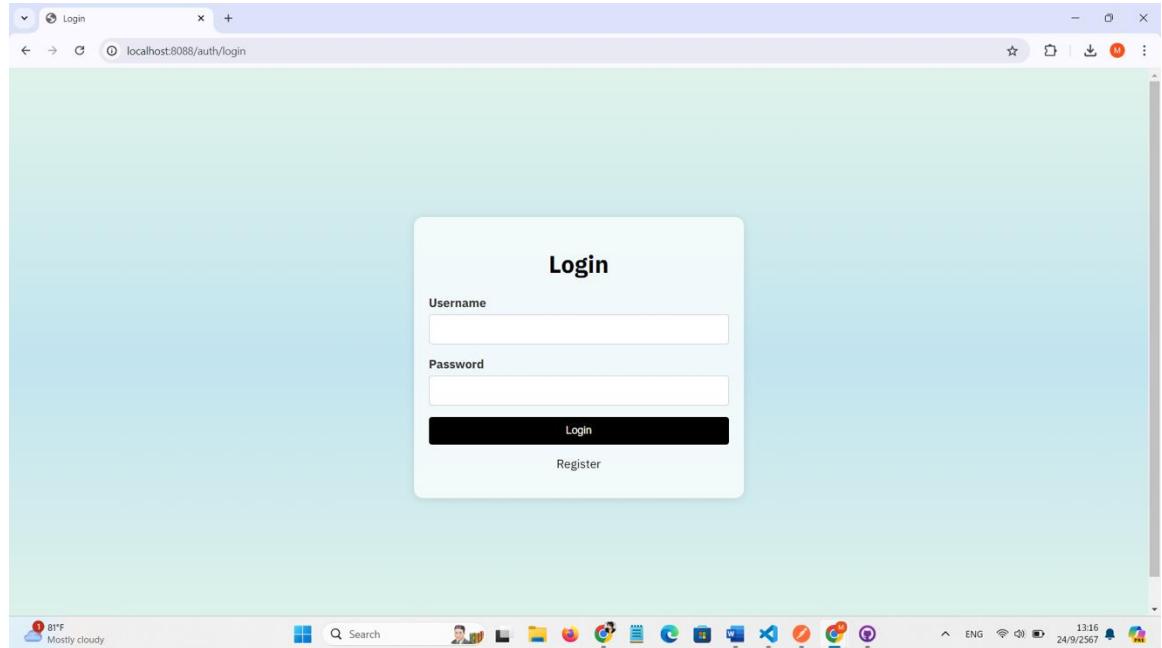
ผลการทดสอบโปรแกรม เป็นดังนี้

หน้าลงทะเบียน (Register)



ภาพที่ 70 ภาพหน้าลงทะเบียน (Register)

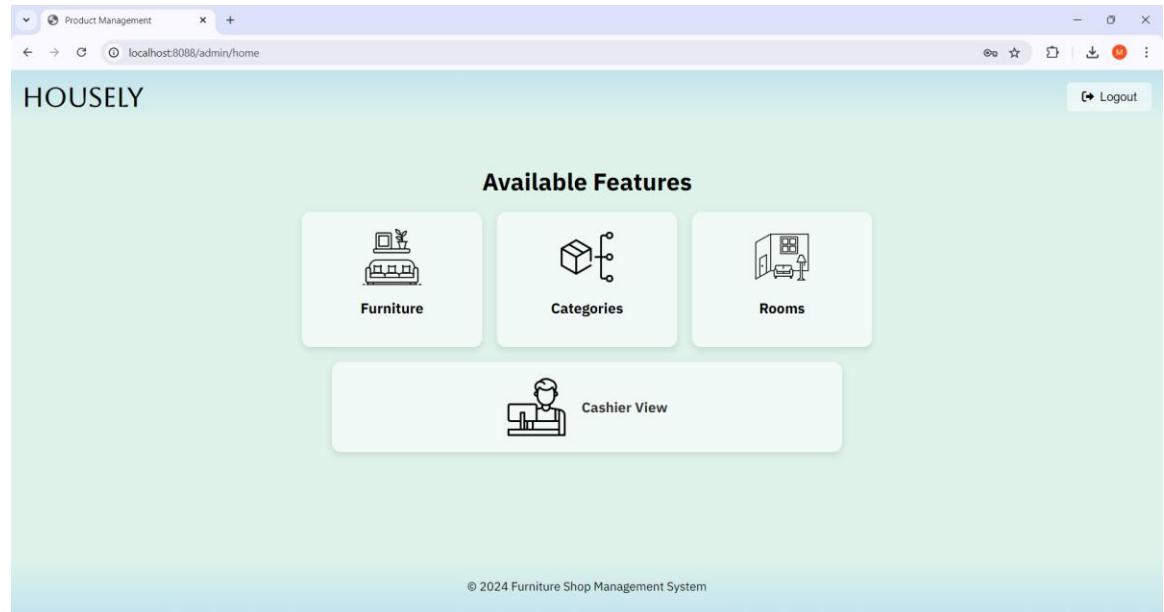
หน้าการเข้าสู่ระบบ (Login)



ภาพที่ 71 ภาพหน้าการเข้าสู่ระบบ (Login)

หน้าเว็บและฟังก์ชันที่บบทบทแอดมิน (Admin) สามารถเข้าถึงได้

- หน้าหลักของแอดมิน (/admin/home) ใช้ในการจัดการข้อมูลในฐานข้อมูล



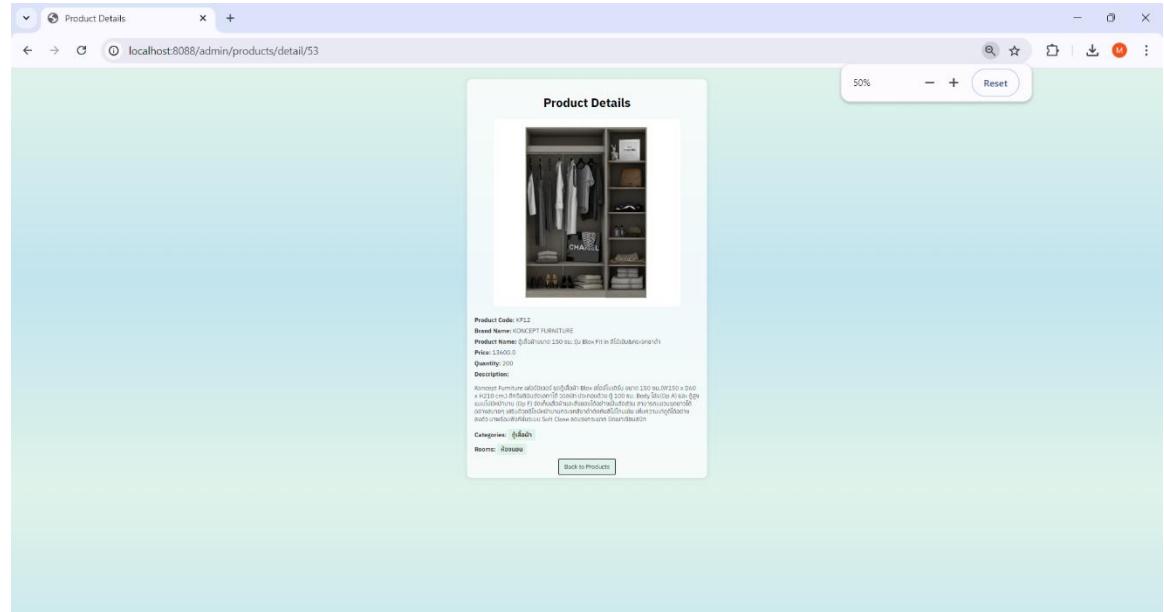
ภาพที่ 72 ภาพหน้าแรก

- หน้าจัดการสินค้า/เฟอร์นิเจอร์ (/admin/products): เข้าถึงได้เมื่อกดเมนู Furniture

| รูปภาพ | รหัสสินค้า | ยี่ห้อ | รีวิวสินค้า | ราคา | จำนวนสต็อก | ประเภท | ผู้ดูแล | การดำเนินการ |
|--------|------------|-------------------|---------------------------------------|---------|------------|------------|---------|---|
| | KF12 | KONCEPT FURNITURE | โต๊ะรับแขกขาตัน 150 ซม. รุ่น Blox Fit | 13600.0 | 200 | โต๊ะรับแขก | ก้องบอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | LS1 | LIVING STORY | LS โคมไฟแขวน รุ่น #CS-PD113 | 3590.0 | 200 | โคมไฟ | ก้องบอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB1 | SB FURNITURE | โซฟาเข้าบานเข้ารุ่น Geezer สีเทา | 38900.0 | 200 | โซฟา | ก้องบอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 73 ภาพหน้าจัดการสินค้า

- หน้ารายละเอียดสินค้า (/admin/products/{id}): เข้าถึงได้เมื่อกดปุ่ม Detail



ภาพที่ 74 ภาพหน้ารายละเอียดสินค้า

- หน้าเพิ่มสินค้าใหม่ (/admin/products/add): เข้าสู่ได้เมื่อกดปุ่มเพิ่มสินค้าใหม่

The screenshot shows a Windows desktop environment. A browser window titled "Product Form" is open, displaying a form titled "Add Product". The form includes fields for Product Code, Brand Name, Product Name, Price, Quantity, Description, and Categories. The "Categories" field is currently set to "Select Categories". The taskbar at the bottom shows various application icons.

ภาพที่ 75 ภาพหน้าเพิ่มสินค้าใหม่

The screenshot shows a Windows desktop environment. A browser window titled "Product Form" is open, displaying a form titled "Add Product". The form includes fields for Product Name, Price, Quantity, Description, Categories, Rooms, and Image. The "Image" field has a "Choose file" button. The "Rooms" field is currently set to "Select Rooms". The "Categories" field is also present. At the bottom of the form are "Submit" and "Cancel" buttons.

ภาพที่ 75 ภาพหน้าเพิ่มสินค้าใหม่ (ต่อ)

- ตัวอย่างการกรอกข้อมูลเพิ่มสินค้าใหม่

ภาพที่ 76 ภาพตัวอย่างการกรอกข้อมูลเพิ่มสินค้าใหม่

ภาพที่ 76 ภาพตัวอย่างการกรอกข้อมูลเพิ่มสินค้าใหม่ (ต่อ)

- เมื่อกด submit จะทำการบันทึกข้อมูลสินค้าลงฐานข้อมูล และจะมาโชว์ที่หน้าแสดงสินค้า

| รูปภาพ | รหัสสินค้า | ชื่อห้อง | ชื่อสินค้า | ราคา | จำนวนสต็อก | ประเภท | ห้อง | การดำเนินการ |
|--------|------------|-------------------|---|---------|------------|-------------|---------------------|---|
| | KF12 | KONCEPT FURNITURE | ตู้เสื้อผ้าบานบาน 150 ซม. รุ่น Blox Fit ในสีبنيเข้ม&กรุงชาดำ | 13600.0 | 200 | ตู้เสื้อผ้า | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | LS1 | LIVING STORY | LS โคมไฟแขวน รุ่น #CS-PD113 เงินก้าวตาม | 3590.0 | 200 | โคมไฟ | ห้องน้ำ… ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB1 | SB FURNITURE | โซฟาบ้านบานเข้า รุ่น Geezer สีเทา | 38900.0 | 200 | โซฟ่า | ห้องน้ำ… | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB2 | KONCEPT FURNITURE | เตียง 3.5 ฟุต KC-PLAY รุ่น EMBRACE สีสันเบร์เก-ขาว | 3150.0 | 20 | เตียงนอน | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 77 ภาพโชว์ที่หน้าแสดงสินค้าเมื่อบันทึกข้อมูลสินค้าลงฐานข้อมูลเรียบร้อย

- หน้า edit (/admin/products/edit/{id}): เข้าถึงได้จากการกดปุ่ม Edit

ກາພີ້ 78 ກາພໜ້າ Edit Product

ກາພີ້ 78 ກາພໜ້າ Edit Product (ຕ້ອ)

- ตัวอย่างการใช้ฟังก์ชันการค้นหา: เลือกคันหาจากรหัสสินค้า SB จะแสดงสินค้าที่มีรหัส SB

The screenshot shows a browser window titled "รายการสินค้า" (Product List) from the URL "localhost:8088/admin/products". The interface includes a search bar with the term "SB" and a table displaying two product entries:

| รูปภาพ | รหัสสินค้า | ผู้ผลิต | ชื่อสินค้า | ราคา | จำนวนเดิม | ประเภท | ห้อง | การดำเนินการ |
|--------|------------|-------------------|---|---------|-----------|---------|------------|---|
| | SB1 | SB FURNITURE | โซฟาเขามบเขาย รุ่น Geezer สีเทา | 38900.0 | 200 | โซฟ่า | ห้องน้ำเสบ | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB2 | KONCEPT FURNITURE | เก้าอี้ 3.5 ผู้ด KC-PLAY รุ่น EMBRACE สีลมเบอร์นา | 3150.0 | 20 | เก้าอี้ | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 79 ภาพตัวอย่างการใช้ฟังก์ชันค้นหา

- เมื่อ กดปุ่มล้างการค้นหา ข้อมูลทั้งหมดจะกลับมาแสดงดังเดิม

| รูปภาพ | รหัสสินค้า | ชื่อห้อง | ชื่อสินค้า | ราคา | จำนวนสต็อก | ประเภท | ห้อง | การดำเนินการ |
|--------|------------|-------------------|--|---------|------------|-------------|--------------------------|---|
| | KF12 | KONCEPT FURNITURE | ตู้เสื้อผ้าบาน联动 150 ซม. รุ่น Blox Fit ในสีبنيเข้ม&กรุงชาดำ | 13600.0 | 200 | ตู้เสื้อผ้า | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | LS1 | LIVING STORY | LS โคมไฟเบนวุ รุ่น #CS-PD113 เงื่อนก้าวตาม | 3590.0 | 200 | โคมไฟ | ห้องน้ำ&สปา , ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB1 | SB FURNITURE | โซฟาบันนับเขาย รุ่น Geever สีเทา | 38900.0 | 200 | โซฟ่า | ห้องน้ำ&สปา | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB2 | KONCEPT FURNITURE | เตียง 3.5 ฟุต KC-PLAY รุ่น EMBRACE สีสันเบร์เก-ขาว | 3150.0 | 20 | เตียงนอน | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 80 ภาพตัวอย่างเมื่อเมื่อ กดปุ่มล้างการค้นหา

- กดปุ่ม Delete เพื่อลบข้อมูลออกจากฐานข้อมูล

The screenshot shows a web browser window with the URL `localhost:8088/admin/products`. The main content is a table titled "รายการสินค้า" (Product List) displaying four items. A confirmation dialog box is overlaid on the page, asking "localhost:8088 says คุณแน่ใจว่าต้องการลบสินค้าี้นี้?" (Are you sure you want to delete this item?). The dialog has "OK" and "Cancel" buttons.

| รูปภาพ | รหัสสินค้า | ยี่ห้อ | ชื่อสินค้า | ราคา | จำนวนสต็อก | ประเภท | ห้อง | การดำเนินการ |
|--------|------------|-------------------|---|---------|------------|-------------|-----------------|---|
| | KF12 | KONCEPT FURNITURE | ตู้เสื้อผ้าบานบาน 150 ซม. รุ่น Blox Fit ในสีبنيเข้ม&กรุงชาดำ | 13600.0 | 200 | ตู้เสื้อผ้า | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | LS1 | LIVING STORY | LS โคมไฟเบนวุ รุ่น #CS-PD113 เงินก้าวตาม | 3590.0 | 200 | โคมไฟ | ห้องน้ำ&ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB1 | SB FURNITURE | โซฟาเข้าบันเข้า รุ่น Geezer สีเทา | 38900.0 | 200 | โซฟา | ห้องน้ำ&ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB2 | KONCEPT FURNITURE | เตียง 3.5 ฟุต KC-PLAY รุ่น EMBRACE สีสันเบร์ลีน-ขาว | 3150.0 | 20 | เตียงนอน | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 81 ภาพตัวอย่างเมื่อเมื่อกดปุ่ม delete

The screenshot shows the same web browser window and product list table. However, the fourth row (SB2) is now missing from the table, indicating it has been successfully deleted.

| รูปภาพ | รหัสสินค้า | ยี่ห้อ | ชื่อสินค้า | ราคา | จำนวนสต็อก | ประเภท | ห้อง | การดำเนินการ |
|--------|------------|-------------------|---|---------|------------|-------------|-----------------|---|
| | KF12 | KONCEPT FURNITURE | ตู้เสื้อผ้าบานบาน 150 ซม. รุ่น Blox Fit ในสีبنيเข้ม&กรุงชาดำ | 13600.0 | 200 | ตู้เสื้อผ้า | ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | LS1 | LIVING STORY | LS โคมไฟเบนวุ รุ่น #CS-PD113 เงินก้าวตาม | 3590.0 | 200 | โคมไฟ | ห้องน้ำ&ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |
| | SB1 | SB FURNITURE | โซฟาเข้าบันเข้า รุ่น Geezer สีเทา | 38900.0 | 200 | โซฟา | ห้องน้ำ&ห้องนอน | <button>Detail</button> <button>Edit</button> <button>Delete</button> |

ภาพที่ 81 ภาพตัวอย่างเมื่อเมื่อกดปุ่ม delete (ต่อ)

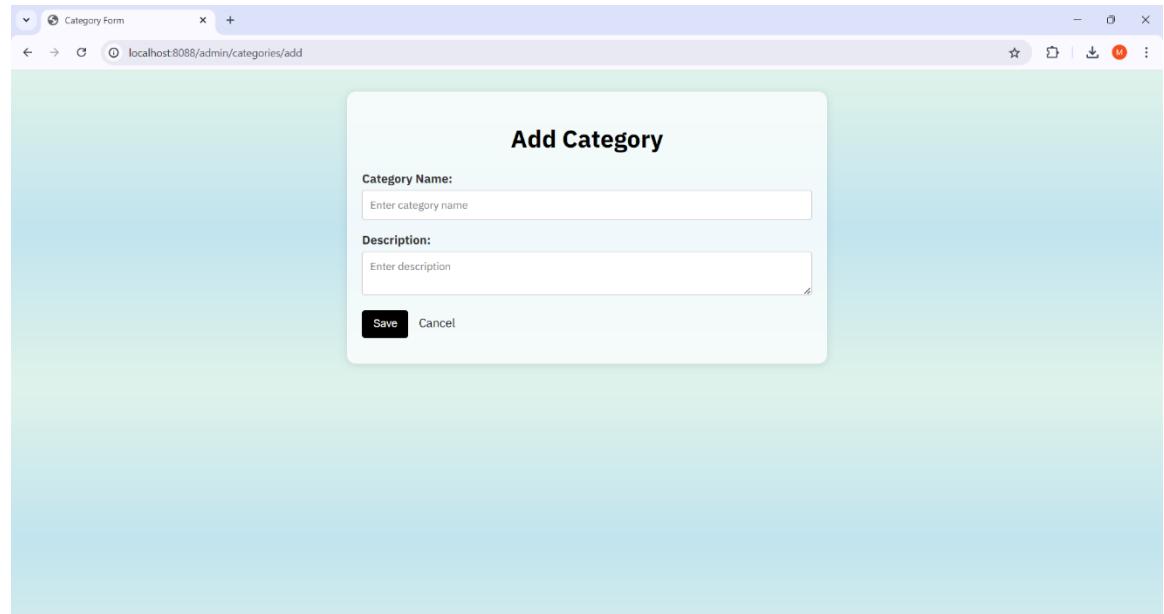
- กดปุ่มกลับสู่หน้าหลักเพื่อกลับไปหน้าหลักของแอ็อดมิน (/admin/home) หรือกดปุ่มออกจากระบบ (Logout) เพื่อไปสู่หน้าเข้าสู่ระบบ (/Login)
- หน้า Categories (/admin/categories): เข้าถึงได้จากการกดปุ่ม Category ที่หน้าหลักของแอ็อดมิน (/admin/home)

รายการหมวดหมู่

| ID | ชื่อหมวดหมู่ | คำอธิบาย | ตัวเลือก |
|----|--------------|----------------|---|
| 1 | โชฟ่า | บุน สนาย | <button>Edit</button> <button>Delete</button> |
| 2 | โคงไฟ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 3 | เก้าอี้ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 6 | เตี๊ยะ | เตี๊ยะ | <button>Edit</button> <button>Delete</button> |
| 11 | ตู้เสื้อผ้า | ตู้ใส่เสื้อผ้า | <button>Edit</button> <button>Delete</button> |
| 12 | เตียงนอน | เตียงใบนอน | <button>Edit</button> <button>Delete</button> |

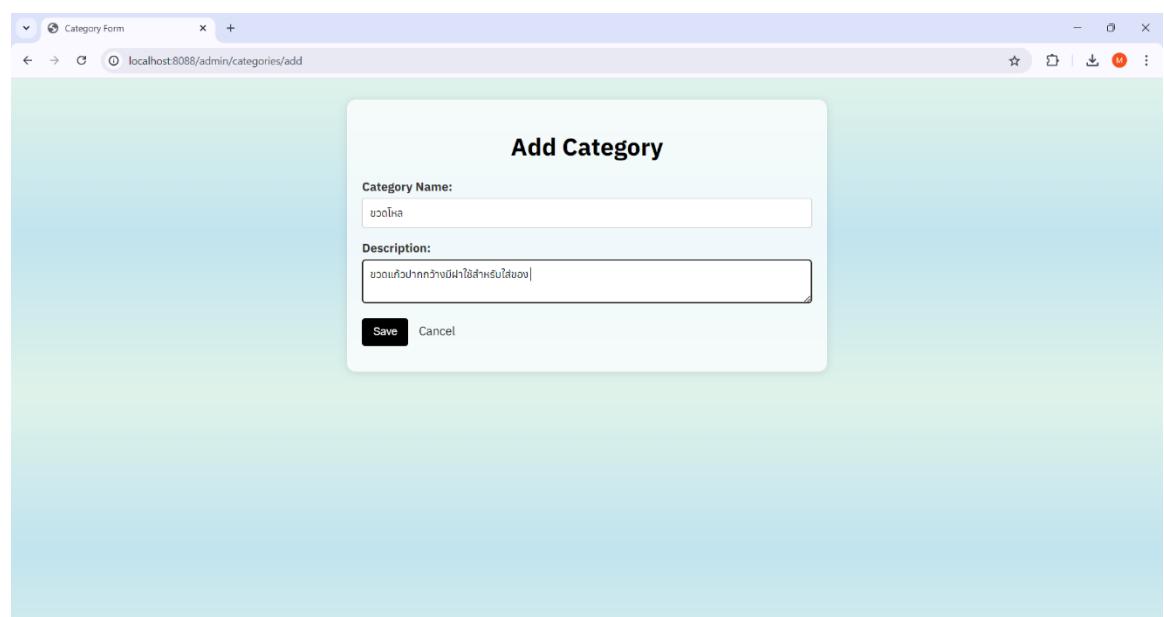
ภาพที่ 82 ภาพหน้า Categories

- หน้าเพิ่มหมวดหมู่ใหม่ (/admin/categories/add): เข้าถึงได้จากการกดปุ่มเพิ่มหมวดหมู่ใหม่



ภาพที่ 83 ภาพหน้าเพิ่มหมวดหมู่ใหม่

- ตัวอย่างการเพิ่มหมวดหมู่ใหม่



ภาพที่ 84 ภาพตัวอย่างการเพิ่มหมวดหมู่ใหม่

| ID | ชื่อหมวดหมู่ | คำอธิบาย | ตัวเลือก |
|----|--------------|------------------------------------|---|
| 1 | ไข่พ่า | บุบ สาย | <button>Edit</button> <button>Delete</button> |
| 2 | โคโนฟิ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 3 | เก้าอี้ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 6 | ได- | ได- | <button>Edit</button> <button>Delete</button> |
| 11 | ถุงเสื่อม | ถุงเสื่อม | <button>Edit</button> <button>Delete</button> |
| 12 | เด็กบอน | เด็กบอน | <button>Edit</button> <button>Delete</button> |
| 13 | ขวดโพล | ขวดแก้วปากกว้างมีฝาใช้สำหรับใส่ของ | <button>Edit</button> <button>Delete</button> |

ภาพที่ 84 ภาพตัวอย่างการเพิ่มหมวดหมู่ใหม่ (ต่อ)

- หน้าแก้ไขข้อมูลหมวดหมู่ (/admin/categories/edit/{id}): เข้าถึงได้จากการกดปุ่ม Edit

ภาพที่ 85 ภาพหน้าแก้ไขข้อมูลหมวดหมู่

- การลบ (/admin/categories/delete/{id}): เข้าถึงได้จากการกดปุ่ม Delete โดยมีการแจ้งเตือนหากลบสำเร็จ และไม่สำเร็จ
- กรณีลับสำเร็จ

| ID | ឈឺអងគេង | ការបំពេញ | ចំណាំ |
|----|------------|------------|---|
| 1 | ឯកតា | បុរិ សាយ | <button>Edit</button> <button>Delete</button> |
| 2 | គុណភុទ | ធម្មុន | <button>Edit</button> <button>Delete</button> |
| 3 | កោវី | ធម្មុន | <button>Edit</button> <button>Delete</button> |
| 6 | ទីតាំង | ទីតាំង | <button>Edit</button> <button>Delete</button> |
| 11 | ចូលរួមជាតិ | ចូលរួមជាតិ | <button>Edit</button> <button>Delete</button> |
| 12 | ផែិកប៊ូន | ផែិកប៊ូន | <button>Edit</button> <button>Delete</button> |
| 13 | ឯកតាអិល | ឯកតាអិល | <button>Edit</button> <button>Delete</button> |

រាយការណាមុខ រាយការណាមុខ

| ID | ឈឺអងគេង | ការបំពេញ | ចំណាំ |
|----|------------|------------|---|
| 1 | ឯកតា | បុរិ សាយ | <button>Edit</button> <button>Delete</button> |
| 2 | គុណភុទ | ធម្មុន | <button>Edit</button> <button>Delete</button> |
| 3 | កោវី | ធម្មុន | <button>Edit</button> <button>Delete</button> |
| 6 | ទីតាំង | ទីតាំង | <button>Edit</button> <button>Delete</button> |
| 11 | ចូលរួមជាតិ | ចូលរួមជាតិ | <button>Edit</button> <button>Delete</button> |
| 12 | ផែិកប៊ូន | ផែិកប៊ូន | <button>Edit</button> <button>Delete</button> |

រាយការណាមុខ រាយការណាមុខ

-กรณีลบไม่สำเร็จ

| ID | ชื่อหมวดหมู่ | คำอธิบาย | ดำเนินการ |
|----|--------------|----------------|---|
| 1 | ใบฟ้า | บุน สาย | <button>Edit</button> <button>Delete</button> |
| 2 | โคบฟ์ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 3 | เก้าอี้ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 6 | โต๊ะ | โต๊ะ | <button>Edit</button> <button>Delete</button> |
| 11 | ตู้เสื้อผ้า | ตู้ใส่เสื้อผ้า | <button>Edit</button> <button>Delete</button> |

ภาพที่ 87 ภาพตัวอย่างการลบหมวดหมู่กรณีลบไม่สำเร็จ

| ID | ชื่อหมวดหมู่ | คำอธิบาย | ดำเนินการ |
|----|--------------|----------------|---|
| 1 | ใบฟ้า | บุน สาย | <button>Edit</button> <button>Delete</button> |
| 2 | โคบฟ์ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 3 | เก้าอี้ | อบอุ่น | <button>Edit</button> <button>Delete</button> |
| 6 | โต๊ะ | โต๊ะ | <button>Edit</button> <button>Delete</button> |
| 11 | ตู้เสื้อผ้า | ตู้ใส่เสื้อผ้า | <button>Edit</button> <button>Delete</button> |

ภาพที่ 87 ภาพตัวอย่างการลบหมวดหมู่กรณีลบไม่สำเร็จ (ต่อ)

- พัฟฟ์ชันการค้นหา สามารถค้นจากชื่อหมวดหมู่ได้เพียงอย่างเดียว และมีพัฟฟ์ชันกลับสู่หน้าแรก และล็อกเอาต์
- หน้าข้อมูลห้อง (/admin/rooms): สามารถเข้าถึงได้จากการกดเมนู Room บนหน้าหลักของแอปมิน ใช้ในการจัดการกับข้อมูลของห้องต่าง ๆ

| ID | ชื่อห้อง | คำอธิบาย | ตัวเลือก |
|----|---------------|----------|---|
| 1 | Outdoor | dd | <button>Edit</button> <button>Delete</button> |
| 3 | ห้องเบื้องลับ | บ้านน้ำ | <button>Edit</button> <button>Delete</button> |
| 10 | ห้องนอน | ห้องนอน | <button>Edit</button> <button>Delete</button> |

ภาพที่ 88 ภาพตัวอย่างพัฟฟ์ชันการค้นหา

- หน้าเพิ่มห้องใหม่ (/admin/rooms/add): สามารถเข้าถึงได้จากการกดปุ่มเพิ่มห้องใหม่

ภาพที่ 89 ภาพหน้าเพิ่มห้องใหม่

- ตัวอย่างการเพิ่มห้องใหม่

ภาพที่ 90 ภาพหน้าตัวอย่างเพิ่มห้องใหม่

| ID | ชื่อห้อง | คำอธิบาย | ดำเนินการ |
|----|---------------|---------------------------------------|---|
| 1 | Outdoor | dd | <button>Edit</button> <button>Delete</button> |
| 3 | ห้องเบื้องลับ | บานเป็น | <button>Edit</button> <button>Delete</button> |
| 10 | ห้องนอน | ห้องไว็บอน | <button>Edit</button> <button>Delete</button> |
| 11 | ห้องครัว | ห้องกายนในบ้าน ที่ใช้ในการประกอบอาหาร | <button>Edit</button> <button>Delete</button> |

ภาพที่ 90 ภาพหน้าตัวอย่างเพิ่มห้องใหม่ (ต่อ)

- หน้าแก้ไขข้อมูลห้อง (/admin/rooms/edit/{id}): สามารถเข้าถึงได้จากปุ่ม Edit

ภาพที่ 91 ภาพหน้าตัวอย่างแก้ไขห้อง

| ID | ชื่อห้อง | คำอธิบาย | ดูเพิ่ม |
|----|------------|--|---|
| 1 | Outdoor | dd | <button>Edit</button> <button>Delete</button> |
| 3 | ห้องบึงเสบ | บ้านีง | <button>Edit</button> <button>Delete</button> |
| 10 | ห้องนอน | เป็นห้องส่วนตัวที่เชื่อมโยงกับห้องน้ำในบ้านกลางคืน หรือใช้พื้นที่ในบ้านกลางวัน | <button>Edit</button> <button>Delete</button> |
| 11 | ห้องครัว | ห้องภายในบ้าน ใช้ในการประกอบอาหาร | <button>Edit</button> <button>Delete</button> |

ภาพที่ 91 ภาพหน้าตัวอย่างแก้ไขห้อง (ต่อ)

- พังก์ชันการลบข้อมูลห้อง (/admin/rooms/delete/{id}): เข้าถึงได้จากการกดปุ่ม Delete โดยมีการแจ้งเตือนหากลบสำเร็จ และไม่สำเร็จ
- กรณีลับสำเร็จ

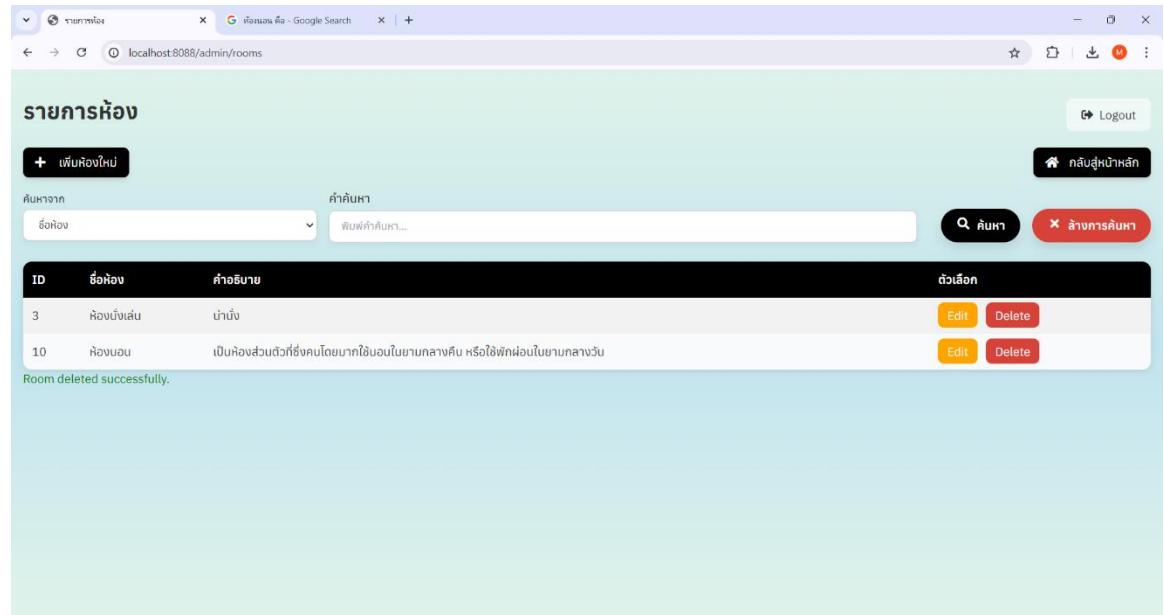
localhost:8088 says
คุณแนใจว่าต้องลบห้อง (Room) นี้?

OK Cancel

| ID | ชื่อห้อง | คำอธิบาย | ดูเพิ่ม |
|----|------------|--|---|
| 3 | ห้องบึงเสบ | บ้านีง | <button>Edit</button> <button>Delete</button> |
| 10 | ห้องนอน | เป็นห้องส่วนตัวที่เชื่อมโยงกับห้องน้ำในบ้านกลางคืน หรือใช้พื้นที่ในบ้านกลางวัน | <button>Edit</button> <button>Delete</button> |
| 11 | ห้องครัว | ห้องภายในบ้าน ใช้ในการประกอบอาหาร | <button>Edit</button> <button>Delete</button> |

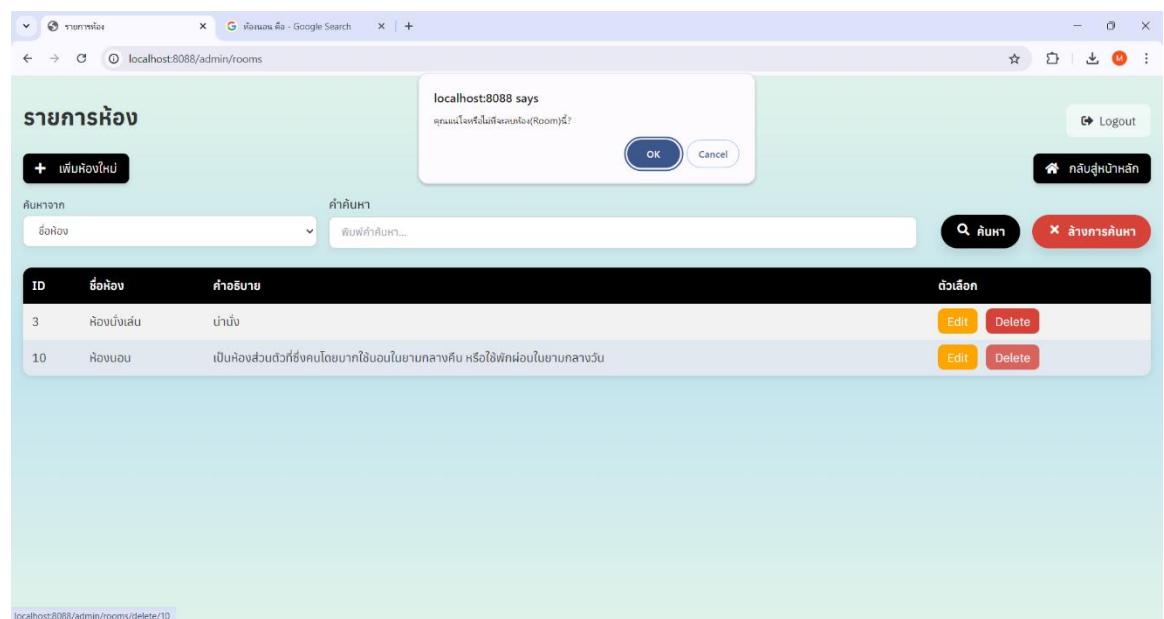
localhost:8088/admin/rooms/delete/11

ภาพที่ 92 ภาพตัวอย่างฟังก์ชันการลบข้อมูลห้อง

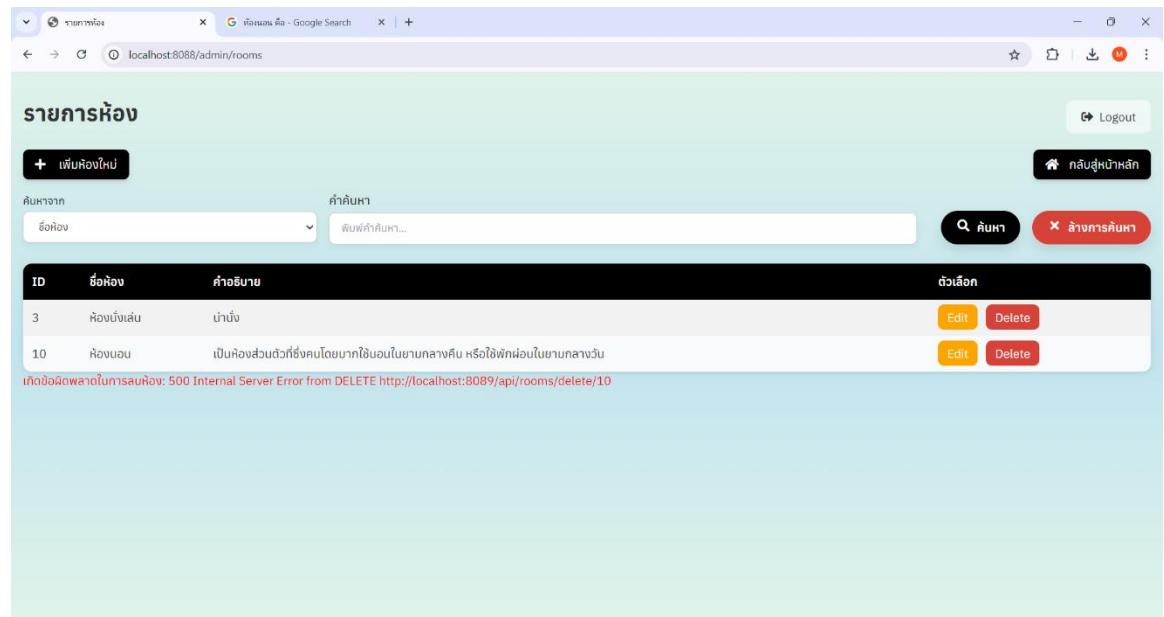


ภาพที่ 92 ภาพตัวอย่างฟังก์ชันการลบข้อมูลห้อง (ต่อ)

- กรณีลบไม่สำเร็จ



ภาพที่ 93 ภาพตัวอย่างฟังก์ชันการลบข้อมูลห้องกรณีลบไม่สำเร็จ



ภาพที่ 93 ภาพตัวอย่างฟังก์ชันการลบข้อมูลห้องกรณีลับไม่สำเร็จ (ต่อ)

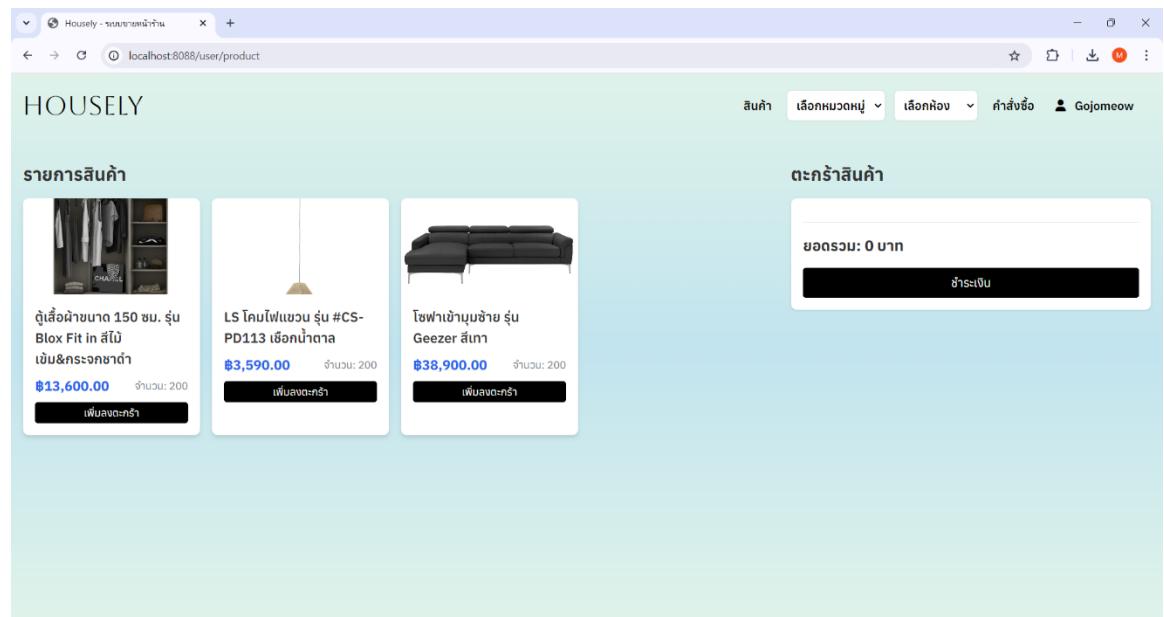
- ฟังก์ชันการค้นหา สามารถค้นจากชื่อหมวดหมู่ได้เพียงอย่างเดียว และมีฟังก์ชันกลับสู่หน้าแรก และล็อกเอาต์

หน้าเว็บและฟังก์ชันที่แออดมิน (Admin) และผู้ใช้ (User) สามารถเข้าถึงได้

- หน้าการซื้อสินค้า (/user/product)

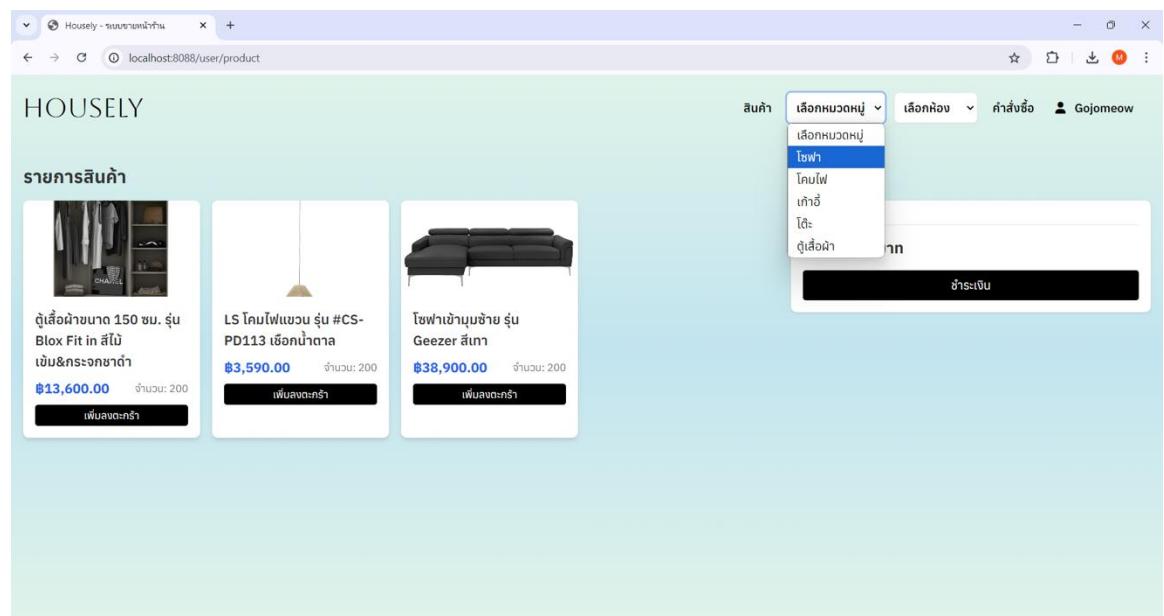
- แออดมินสามารถเข้าถึงได้จากการกดเมนู Cashier View ที่หน้าหลักของแออดมิน

- ผู้ใช้จะสามารถเข้าถึงได้ทันทีเมื่อเข้าสู่ระบบ

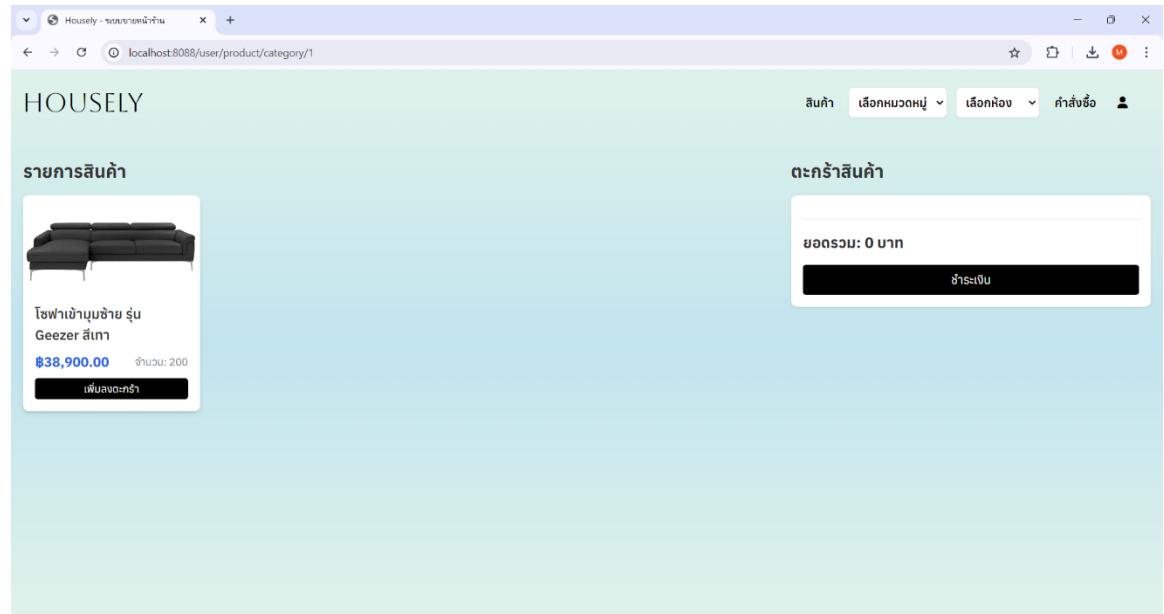


ภาพที่ 94 ภาพหน้าการซื้อสินค้า

- สามารถเลือกสินค้าจากหมวดหมู่ได้

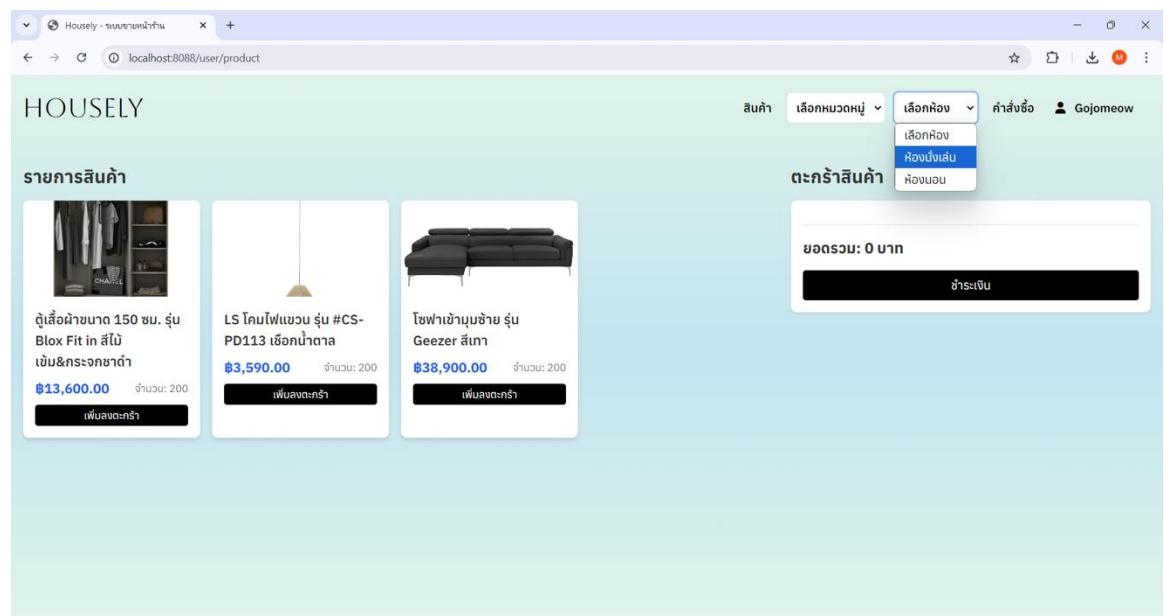


ภาพที่ 95 ภาพตัวอย่างการเลือกสินค้าจากหมวดหมู่

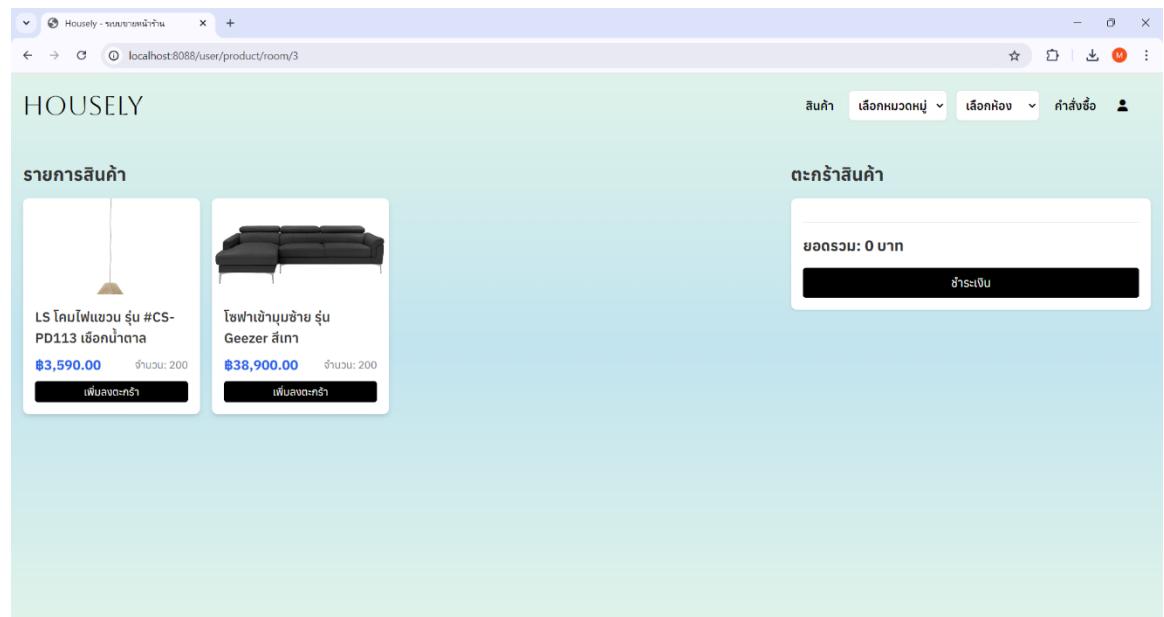


ภาพที่ 95 ภาพตัวอย่างการเลือกสินค้าจากหมวดหมู่ (ต่อ)

- สามารถเลือกสินค้าจากห้องได้

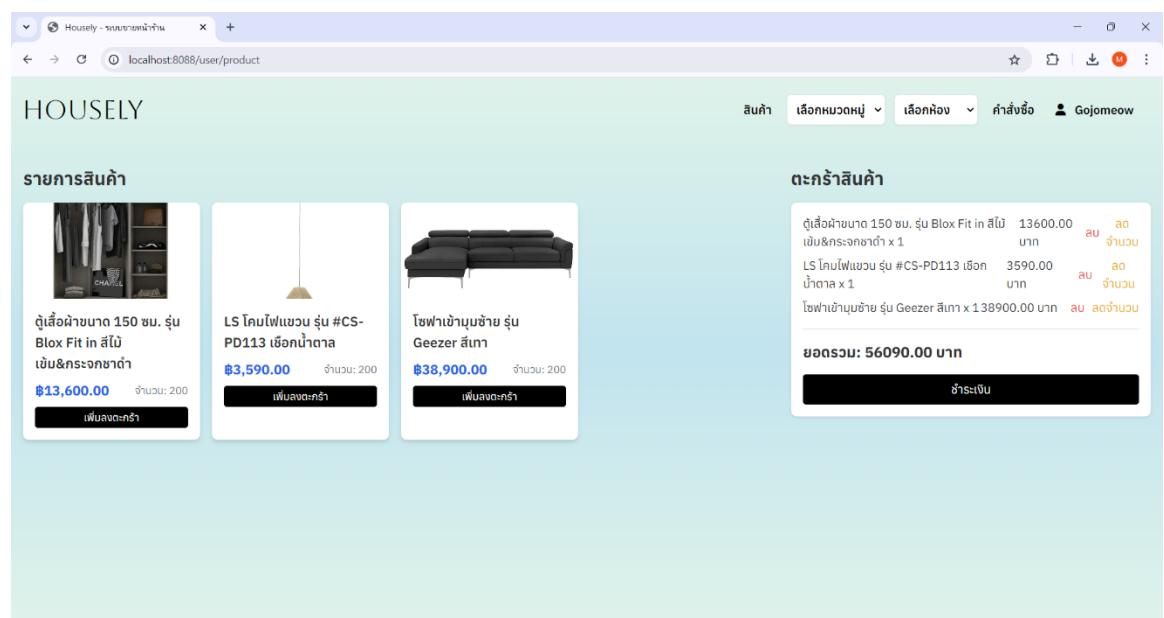


ภาพที่ 96 ภาพตัวอย่างการเลือกสินค้าจากห้อง



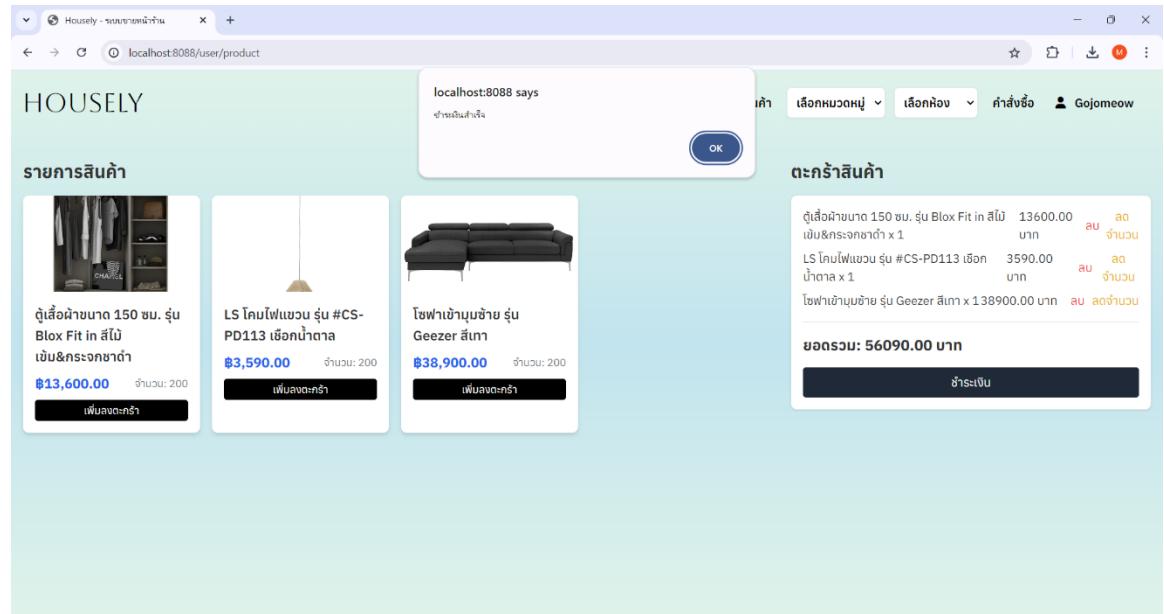
ภาพที่ 96 ภาพตัวอย่างการเลือกสินค้าจากห้อง (ต่อ)

- เพิ่มสินค้าลงในกระรักสินค้าได้โดยการกดปุ่มเพิ่มลงในกระรัก



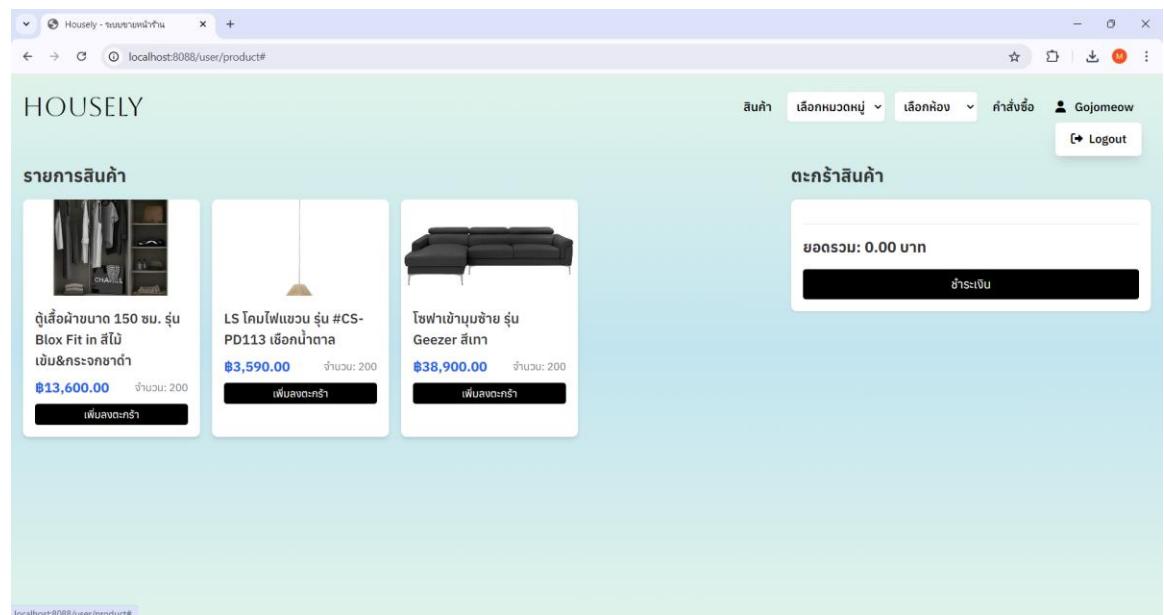
ภาพที่ 97 ภาพตัวอย่างการเพิ่มสินค้าลงในกระรัก

- เมื่อกดชำระบัญชีนั้นแจ้งเตือนว่าชำระบัญชีสำเร็จแล้ว



ภาพที่ 97 ภาพตัวอย่างการเพิ่มสินค้าลงตะกร้า (ต่อ)

- กดที่ช่องผู้ใช้จะมีปุ่มล็อกເອົາຕໍ່



ภาพที่ 98 ภาพแสดงปุ่มล็อกເອົາຕໍ່ໃນหน้า product

บทที่ 5

อภิปรายและข้อเสนอแนะ

ในบทนี้จะสรุปผลการวิจัยที่ได้จากการพัฒนาระบบขายเฟอร์นิเจอร์ออนไลน์ ซึ่งได้ดำเนินการในส่วนต่างๆ ของระบบ รวมถึงการทดสอบการทำงานและประสิทธิภาพ โดยรายละเอียดสำคัญสามารถสรุปได้ดังนี้

5.1 ผลการพัฒนาระบบ

ระบบจัดการเฟอร์นิเจอร์ที่พัฒนาขึ้นนี้ได้ตอบสนองตามวัตถุประสงค์ที่ตั้งไว้ ซึ่งประกอบด้วยฟังก์ชันการขาย การจัดการสินค้า การเพิ่ม ลบ และแก้ไขข้อมูลสินค้า และการเชื่อมต่อกับฐานข้อมูลเพื่อบันทึกข้อมูลสินค้า ข้อมูลลูกค้า และการขาย ระบบเนี้ยบสามารถรองรับการใช้งานผ่านอินเตอร์เฟชที่ใช้งานง่ายและตอบสนองได้ดีบนเว็บ

จากการทดสอบระบบ พนักงานขายสามารถดำเนินการสั่งซื้อสินค้าผ่านระบบได้ ผู้ดูแลระบบสามารถจัดการสินค้าได้ ระบบสามารถแสดงข้อมูลได้ถูกต้องและตอบสนองการค้นหาและเรียกดูสินค้าตามหมวดหมู่ได้ดี มี spring security ที่ใช้ในการ login และ register มี role ที่สามารถกำหนด path ให้แต่ละ role ได้ ทำให้สามารถจำกัดสิทธิ์ได้ เพิ่มความปลอดภัยให้แก่ระบบ

5.2 การทำงานในการพัฒนาระบบ

5.2.1 ในการทำงานเพื่อพัฒนาระบบได้มีการใช้ github desktop เพื่อดำเนินการดังนี้

- จัดการกับเวอร์ชันของโค้ด: ทำให้สามารถติดตามและควบคุมการเปลี่ยนแปลงของโค้ดได้อย่างมีประสิทธิภาพ
- ประสานงานกับทีม: อัปโหลดและดึงโค้ดจาก repository เพื่อให้ทุกคนในทีมใช้โค้ดเดียวกันล่าสุดและทำงานร่วมกันได้อย่างราบรื่น
- จัดการ branch: สร้างและสลับ branch ได้ง่าย เพื่อการพัฒนาในส่วนต่างๆ โดยไม่กระทบกับโค้ดหลัก
- ลดความซับซ้อนในการ merge: ทำการรวม branch โดยใช้ GUI ลดข้อผิดพลาดจากการใช้คำสั่ง manual
- ติดตามการเปลี่ยนแปลงโค้ด: เปรียบเทียบการเปลี่ยนแปลงและตรวจสอบความแตกต่างในแต่ละ commit

5.2.2 ประวัติการทำงานของทีมผ่าน github



ภาพที่ 99 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ housely-website



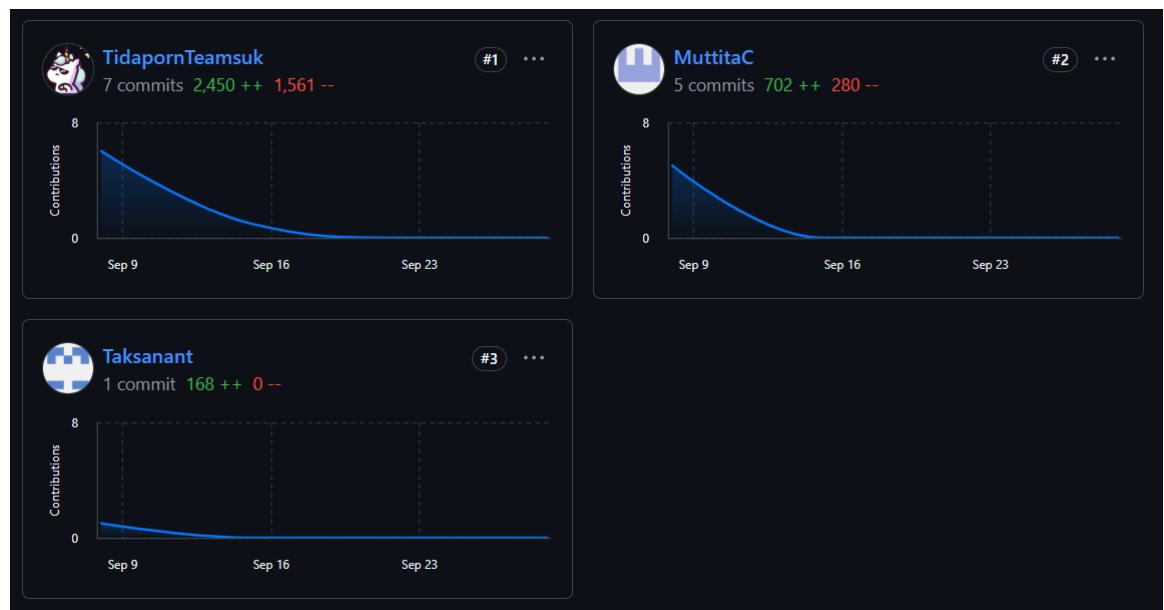
ภาพที่ 100 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ api-housely



ภาพที่ 101 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ HouselyWebsite



ภาพที่ 102 ภาพแสดงประวัติการทำงานของทีมผ่าน github ชื่อ IKEA_Project101



ภาพที่ 103 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ Furniture-selling-website



ภาพที่ 104 ภาพแสดงประวัติการทำงานของทีมผ่าน github ใน repository ชื่อ HouselyServerAPI

5.3 ปัญหาและข้อจำกัด

แม้ระบบจะตอบสนองต่อความต้องการของผู้ใช้เดียว แต่ยังมีข้อเสนอแนะบางประการที่อาจพิจารณาปรับปรุงในอนาคต ได้แก่

- การพัฒนาระบบให้รองรับการชำระเงินออนไลน์
- การเพิ่มฟีเจอร์สำหรับการแนะนำสินค้าที่เหมาะสมกับลูกค้าแต่ละราย
- การพัฒนาแอปพลิเคชันมือถือเพื่อเพิ่มความสะดวกในการใช้งาน

จากการพัฒนาระบบและการทดสอบผลการใช้งานพบว่าระบบขายเฟอร์นิเจอร์ออนไลน์ที่พัฒนาขึ้นนี้สามารถตอบสนองความต้องการของผู้ใช้และกลุ่มเป้าหมายได้เป็นอย่างดี และสามารถใช้เป็นต้นแบบในการพัฒนาระบบที่คอมเมิร์ซอื่น ๆ ได้ต่อไป

5.4 ข้อเสนอแนะ

ในการพัฒนาระบบขายเฟอร์นิเจอร์ออนไลน์ มีข้อเสนอแนะที่ควรพิจารณาสำหรับการปรับปรุงและพัฒนาระบบในอนาคตเพื่อเพิ่มประสิทธิภาพและความพึงพอใจของผู้ใช้งาน ดังนี้

- การเพิ่มช่องทางการชำระเงินออนไลน์

ควรเพิ่มตัวเลือกการชำระเงินออนไลน์ให้หลากหลายขึ้น เช่น การชำระผ่านบัตรเครดิต, การโอนเงินผ่านธนาคาร, การใช้บริการ E-Wallet ต่างๆ เช่น TrueMoney, PromptPay, หรือ PayPal เพื่อรับความต้องการที่หลากหลายของผู้ใช้งาน

2. ฟีเจอร์แนะนำสินค้า

การพัฒนาระบบแนะนำสินค้าที่สอดคล้องกับความสนใจและความต้องการของลูกค้าโดยอิงจากประวัติการสั่งซื้อหรือการเรียกดูสินค้า สามารถช่วยเพิ่มโอกาสในการขายและความพึงพอใจของลูกค้าได้

3. การพัฒนาระบบสมาชิกและโปรแกรมสะสมแต้ม

เพิ่มระบบสมาชิกเพื่อให้ลูกค้าสามารถติดตามคำสั่งซื้อของตนเองและประวัติการซื้อสินค้าได้ รวมถึงการสร้างโปรแกรมสะสมแต้มที่สามารถแลกของรางวัลหรือรับส่วนลดในครั้งถัดไป ซึ่งจะช่วยสร้างความภักดีต่อแบรนด์และกระตุ้นให้ลูกค้ากลับมาซื้อซ้ำ

4. การพัฒนาแอปพลิเคชันมือถือ

เนื่องจากผู้ใช้งานส่วนใหญ่นิยมใช้งานผ่านโทรศัพท์มือถือ การพัฒนาแอปพลิเคชันเฉพาะสำหรับการขายสินค้าจะช่วยเพิ่มประสบการณ์การใช้งานที่สะดวกยิ่งขึ้น รวมถึงสามารถใช้ประโยชน์จากฟังก์ชันพิเศษของมือถือ เช่น การแจ้งเตือนแบบพุช (push notification) เพื่อกระตุ้นยอดขาย

5. การรองรับหลายภาษา

เพื่อขยายตลาดไปยังต่างประเทศหรือกลุ่มลูกค้าที่ใช้ภาษาต่างจากภาษาไทย ควรพิจารณาเพิ่มฟังก์ชันการรองรับหลายภาษาในระบบ เช่น ภาษาอังกฤษ หรือภาษาจีน เพื่อเพิ่มโอกาสทางธุรกิจ

6. การบูรณาการกับระบบขนส่ง

ควรพัฒนาระบบที่สามารถเชื่อมต่อกับบริการขนส่งได้โดยตรง เพื่อให้ลูกค้าสามารถติดตามสถานะการจัดส่งสินค้าได้อย่างเรียลไทม์ เพิ่มความมั่นใจและความลงทะเบียนให้กับลูกค้า

7. เพิ่มโมเดลการสร้างใบวางบิล

ระบบควรมีฟังก์ชันในการสร้างใบวางบิลอัตโนมัติหลังจากการสั่งซื้อเสร็จสิ้น เพื่อให้ลูกค้าได้รับเอกสารที่แสดงรายละเอียดการซื้อขายทั้งหมดและเก็บประวัติการสั่งซื้อ ข้อเสนอแนะเหล่านี้จะช่วยให้ระบบขายเฟอร์นิเจอร์ออนไลน์มีความครบถ้วนและมีศักยภาพสูงขึ้นในการรองรับความต้องการของผู้ใช้ในปัจจุบันและอนาคต

เอกสารอ้างอิง

1. Mark Poramest. (2565). Clean Architecture Ep.7: หลักออกแบบ SOLID ที่คุ้นเคย. Medium. ค้นเมื่อ 15 กันยายน 2567, จาก <https://developers.ascendcorp.com/clean-architecture-ep-7-หลักออกแบบ-solid-ที่คุ้นเคย-8d96968e4859>
2. DH Team. (2566). แนวคิดของ MVC Design Pattern. ค้นเมื่อ 15 กันยายน 2567, <https://devhub.in.th/blog/mvc-design-pattern>
3. กรกฎ วิริยะ. (2561). MVC และ MMVC คืออะไร ทำความรู้จักกับสถาปัตยกรรม MVC และ MMVC ของ Somtum. Goragod.com. ค้นเมื่อ 15 กันยายน 2567, จาก <https://somtum.kotchasan.com/mvc>
4. ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) คืออะไร. (2566). ค้นเมื่อ 15 กันยายน 2567, จาก <https://appmaster.io/th/blog/rabbkaarcchadkaarthaanmuulechingsamphanth-rdbms>
5. Normalization (นอร์มัลไลเซชัน) ลดปัญหาการซ้ำซ้อนของข้อมูลในฐานข้อมูล. (2560). ค้นเมื่อ 15 กันยายน 2567, จาก <https://www.mindphp.com/developer/21-sql-mysql/4103-normalization.html>
6. J.Dechavee. (2567). Data normalization. ค้นเมื่อ 16 กันยายน 2567, จาก <https://medium.com/@dechavee.j/data-normalization-7e71d49ed171>
7. SQL คืออะไร. (2565). ค้นเมื่อ 15 กันยายน 2567, จาก <https://appmaster.io/th/blog/sql-khuue-aair>
8. Cloud HM – MKT. (2566) รู้จัก Query Database คืออะไร มีประโยชน์อย่างไร?. ค้นเมื่อ 16 กันยายน 2567, จาก <https://blog.cloudhm.co.th/know-sql-and-benefits/>
9. Spring Boot ทำความรู้จัก เริ่มติดตั้งและเริ่มใช้งานเบื้องต้น EP1. (2563). ค้นเมื่อ 18 กันยายน 2567, จาก <https://www.poolsawat.com/spring-boot-ทำความรู้จัก-เริ่มติด/>
10. จิตกร พิทักษ์เมธากุล. (2563). รู้จักกับ Apache Maven. ค้นเมื่อ 18 กันยายน 2567, จาก <https://www.jittagornp.me/blog/what-is-apache-maven/>
11. PROGRAMMING HUNTER. (2558). Hibernate คืออะไร. ค้นเมื่อ 18 กันยายน 2567, จาก <http://programminghunter.blogspot.com/2015/07/what-is-hibernate-framework.html>
12. openlandscape. (2566). MySQL คืออะไร ? โปรแกรมจัดการฐานข้อมูล Open Source ยอดนิยม !. ค้นเมื่อ 15 กันยายน 2567, จาก <https://blog.openlandscape.cloud/mysql>
13. Chatri Ngambenchawong. (2561). [SPRING] ทำความรู้จักกับ Thymeleaf. ค้นเมื่อ 20 กันยายน 2567, จาก <https://naiwaen.debuggingsoft.com/2018/09/spring-ทำความรู้จักกับ-thymeleaf/>
14. Medium. (2563). ทำความรู้จัก Git & GitHub พร้อมการใช้งานร่วมกับ VS Code เป็นต้น. ค้นเมื่อ 15 กันยายน 2567, จาก <https://stackpython.medium.com/ทำความรู้จัก-git-github-พร้อมการใช้งานร่วมกับ-vs-code-เป็นต้น-f848f41a39e9>

15. Rusfirdao. (2567). Postman : มาเร็จกับ postman กันเถอะ. ค้นเมื่อ 20 กันยายน 2567, จาก <https://medium.com/lotuss-it/postman-มาเร็จกับ-postman-กันเถอะ-157193f5a215>
16. Amazon Web Services. (2566). Java คืออะไร. ค้นเมื่อ 20 กันยายน 2567, จาก <https://aws.amazon.com/th/what-is/java/>
17. HTML คืออะไร เอชทีเอ็มแอล ภาษาคอมพิวเตอร์ที่ใช้ในการสร้างเว็บเพจ ใช้เขียนโปรแกรม ย่อมาจากอะไร. (2565). ค้นเมื่อ 15 กันยายน 2567, จาก <https://www.mindphp.com/คืออะไร/2026-html-คืออะไร.html>
18. บริษัท วันบลิฟ จำกัด. (2560). โครงสร้างพื้นฐานของ HTML. ค้นเมื่อ 21 กันยายน 2567, จาก <https://www.1belief.com/article/html/>
19. บริษัท โปรซอฟท์ เว็บ จำกัด. (ม.ป.ป.). CSS คืออะไร มีประโยชน์อย่างไร. ค้นเมื่อ 21 กันยายน 2567, จาก <https://blog.sogoodweb.com/Article/Detail/79237/CSS-คืออะไร-มีประโยชน์อย่างไร>