# องค์ประกอบของ Use Case Diagram

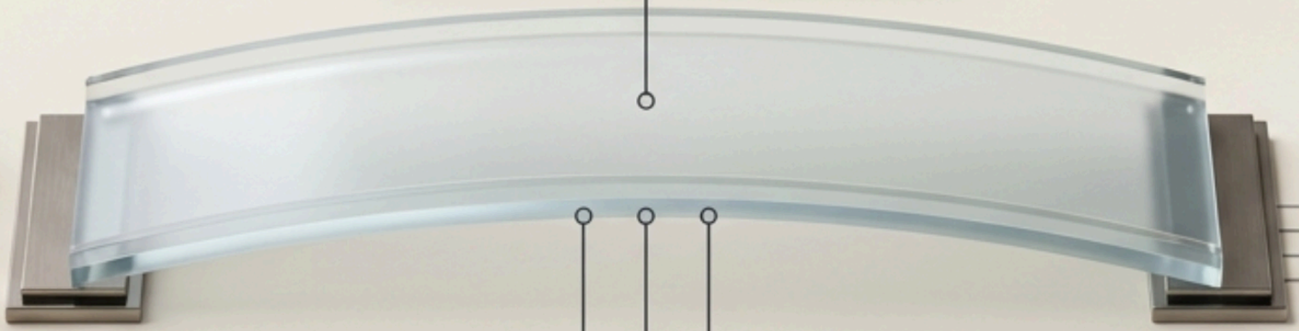
## แบบจำลองพฤติกรรมระบบเพื่อธุรกิจดิจิทัล

รายวิชา: Object-Oriented Analysis and Design (OOAD)
ผู้สอน: AJ.Nattapon Mungkala (Aj.March)
หลักสูตร: ปวส. สาขาเทคโนโลยีธุรกิจดิจิทัล

# Use Case Diagram คืออะไร?

**ความต้องการของธุรกิจ**
(Business Needs)

ภาษาสากลที่ใช้สื่อสารระหว่าง
ผู้ใช้งานกับนักพัฒนาระบบ



**ระบบคอมพิวเตอร์**
(System Functions)

**1. เห็นภาพรวม**
(Big Picture):
ระบบทำอะไรได้บ้าง
โดยไม่ต้องเจาะลึกโค้ด

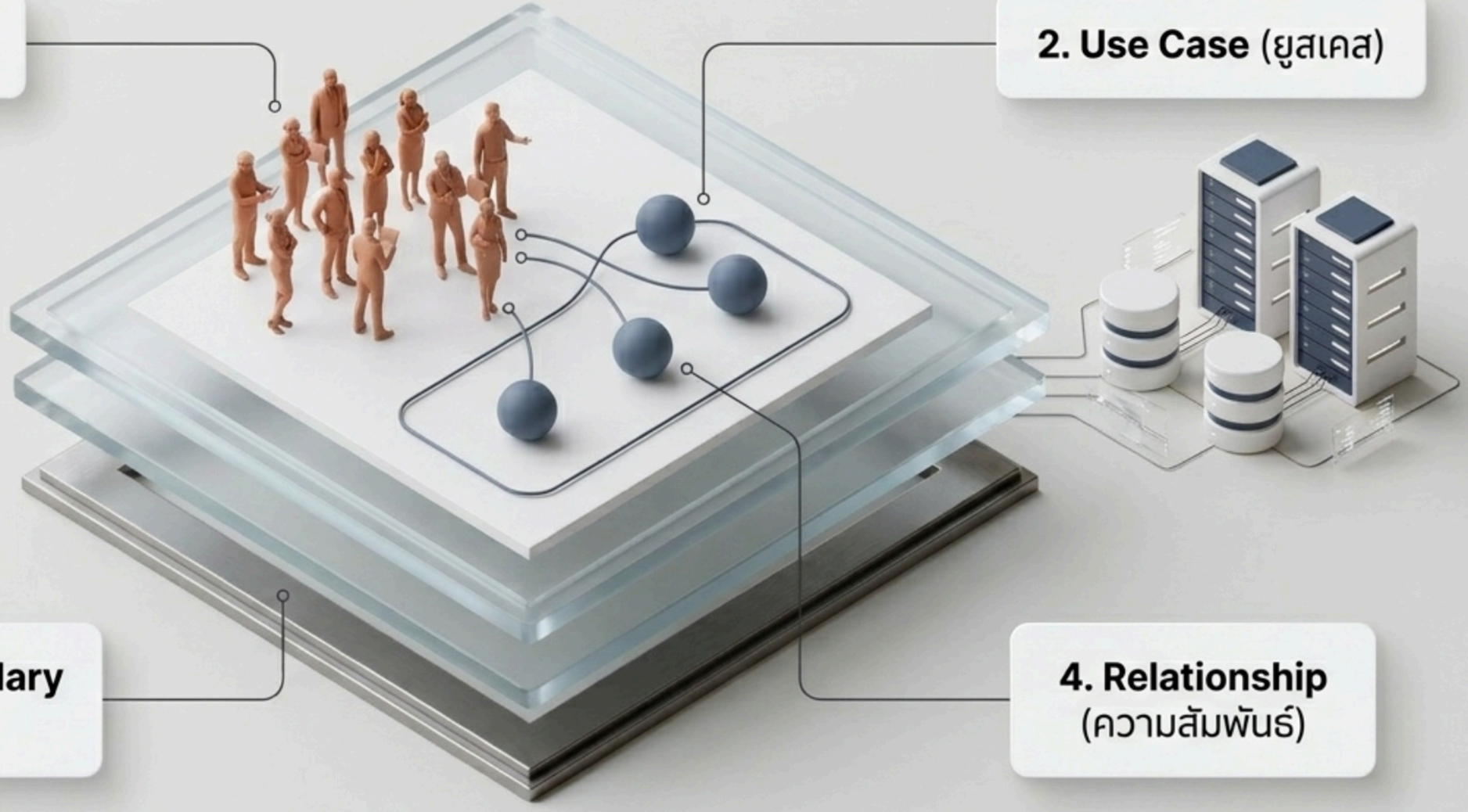
**2. ระบุผู้เกี่ยวข้อง**
(Stakeholders):
ใครคือผู้ใช้งานระบบนี้บ้าง

**3. กำหนดขอบเขต**
(Scope):
อะไรอยู่ในระบบ
อะไรอยู่นอกระบบ

# 4 องค์ประกอบหลักของ Use Case Diagram

**1. Actor** (ผู้กระทำ)

**2. Use Case** (ยูสเคส)



**3. System Boundary**
(ขอบเขตระบบ)

**4. Relationship**
(ความสัมพันธ์)

# 1. ผู้กระทำ (Actor)

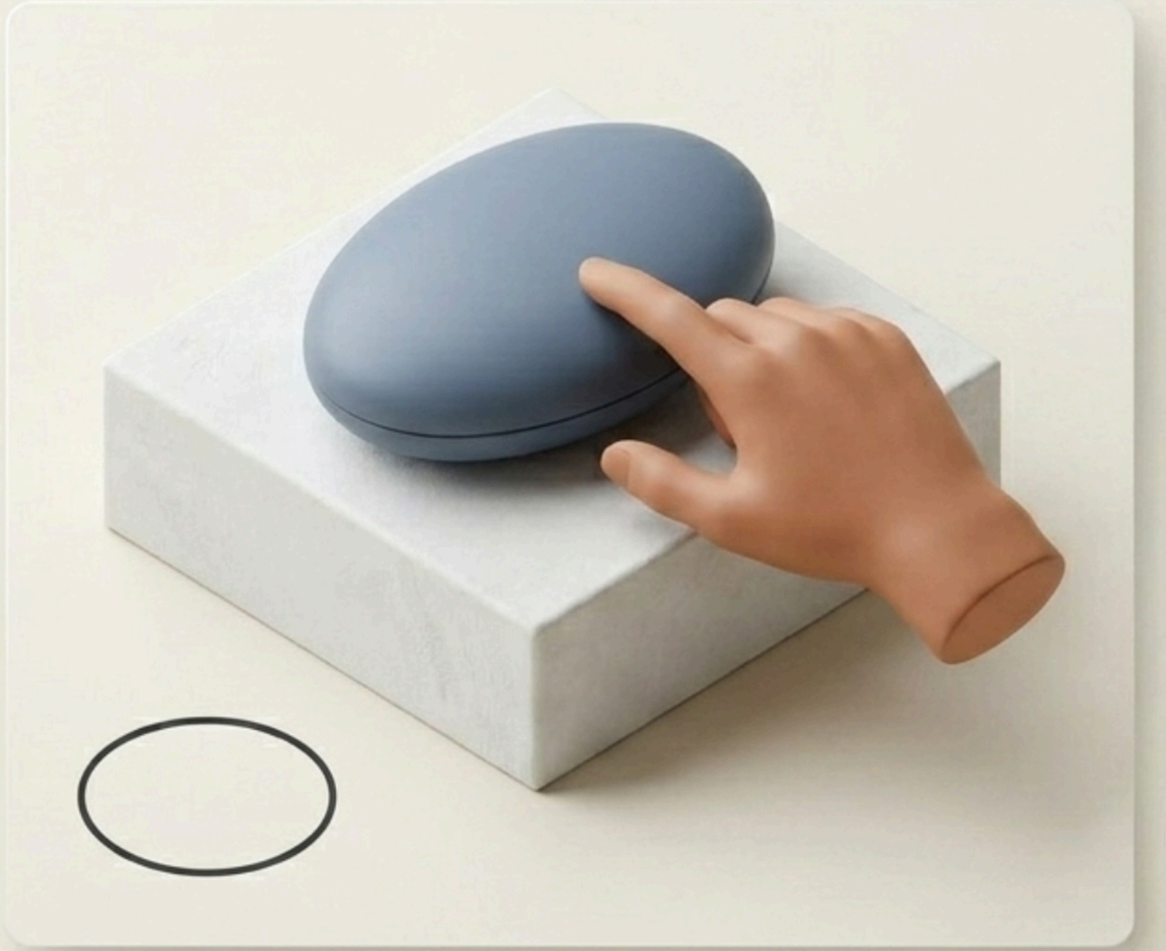
ใคร หรือ อะไร ที่เข้ามาโต้ตอบกับระบบ?


**Primary Actor (ผู้ใช้งานหลัก)**
คนที่เริ่มใช้งานระบบ เช่น ลูกค้า (Customer), ผู้ดูแลระบบ (Admin)

**Secondary Actor** (ระบบภายนอก)
ระบบอื่นที่ถูกเรียกใช้
เช่น ระบบตัดบัตรเครดิต (Payment Gateway)

# 2. ยูสเคส (Use Case)

ระบบมีฟังก์ชั่นหลักอะไรบ้าง?



**กฎเหล็ก: ต้องใช้ คำกริยา (Verb) นำหน้าเสมอ**

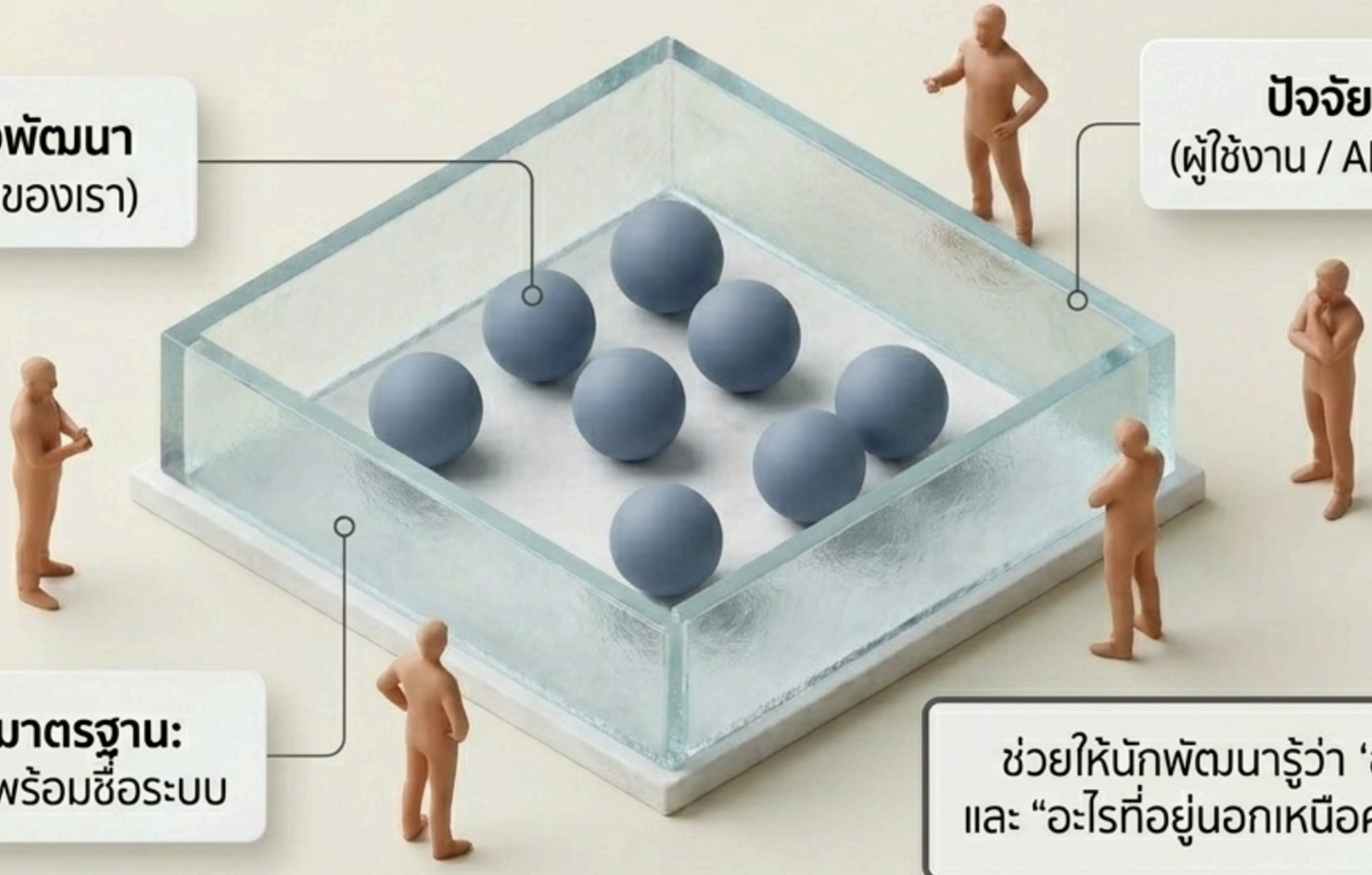
| ✅ **ถูกต้อง** (Correct) | ❌ **ผิด** (Incorrect) |
| --- | --- |
| สมัครสมาชิก | ข้อมูลลูกค้า |
| ค้นหาสินค้า | ตะกร้าสินค้า |
| ชำระเงิน | |

# 3. ขอบเขตระบบ (System Boundary)

เส้นแบ่งระหว่างแอปพลิเคชันกับโลกภายนอก

**สิ่งที่เราต้องพัฒนา**
(แอปพลิเคชันของเรา)

**ปัจจัยภายนอก**
(ผู้ใช้งาน / API ของบริษัทอื่น)

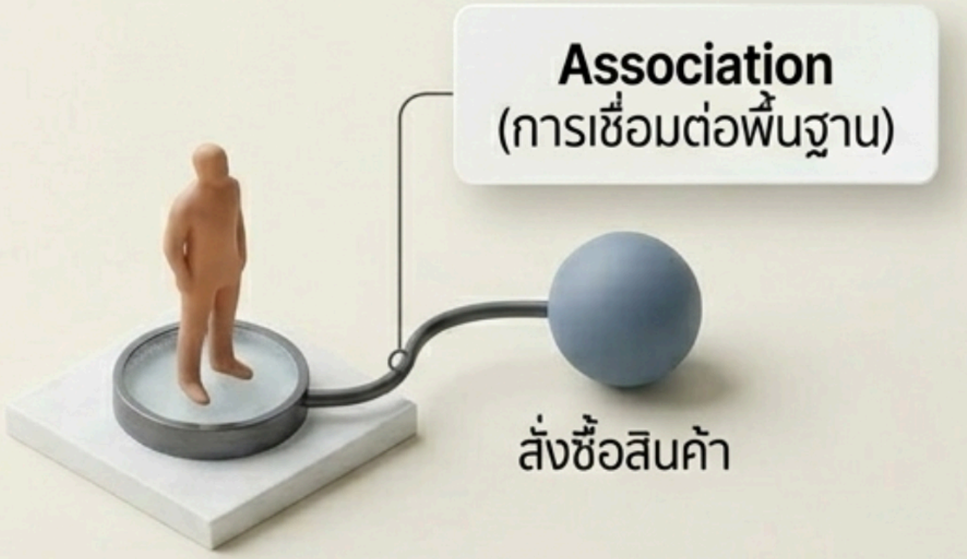


**สัญลักษณ์มาตรฐาน:**
กรอบสี่เหลี่ยมพร้อมชื่อระบบ

ช่วยให้นักพัฒนารู้ว่า ‘อะไรที่ต้องทำ”
และ “อะไรที่อยู่นอกเหนือความรับผิดชอบ’

# 4. ความสัมพันธ์ (Relationships)

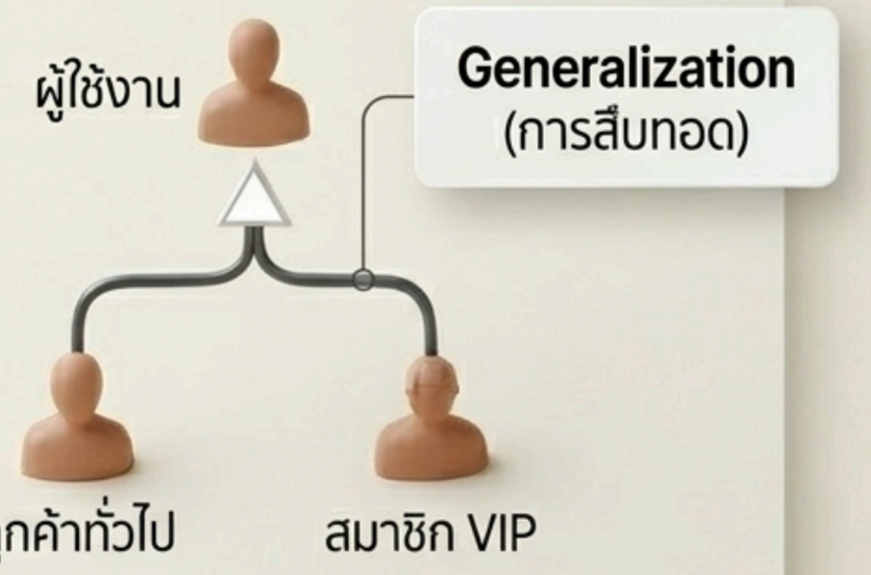
## การเชื่อมต่อข้อมูลและการสืบทอดคุณสมบัติ




### Association (การเชื่อมต่อพื้นฐาน)

แสดงให้เห็นว่าใครสามารถทำฟังก์ชันไหนได้บ้าง

**สัญลักษณ์มาตรฐาน:** เส้นตรงไม่มีหัวลูกศร

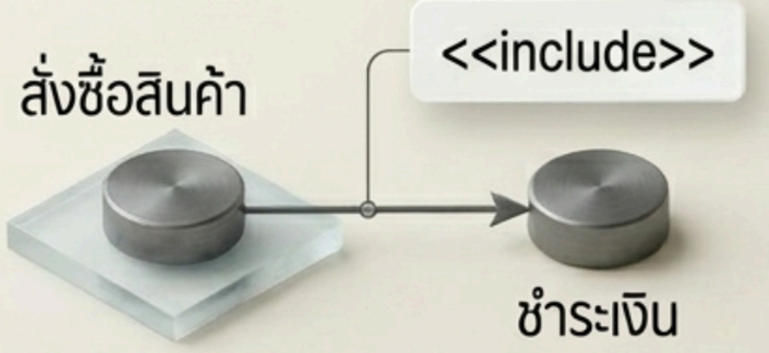
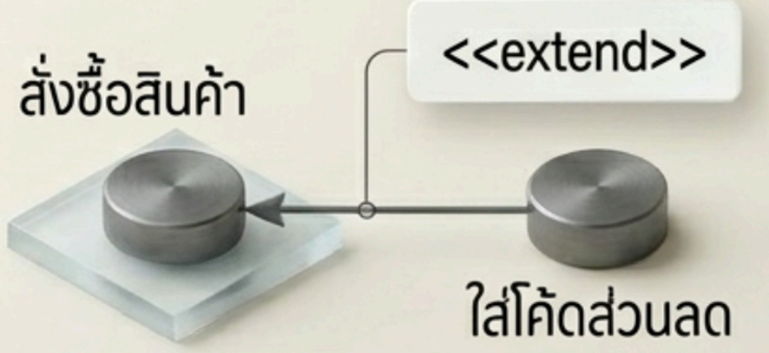



### Generalization (การสืบทอด)

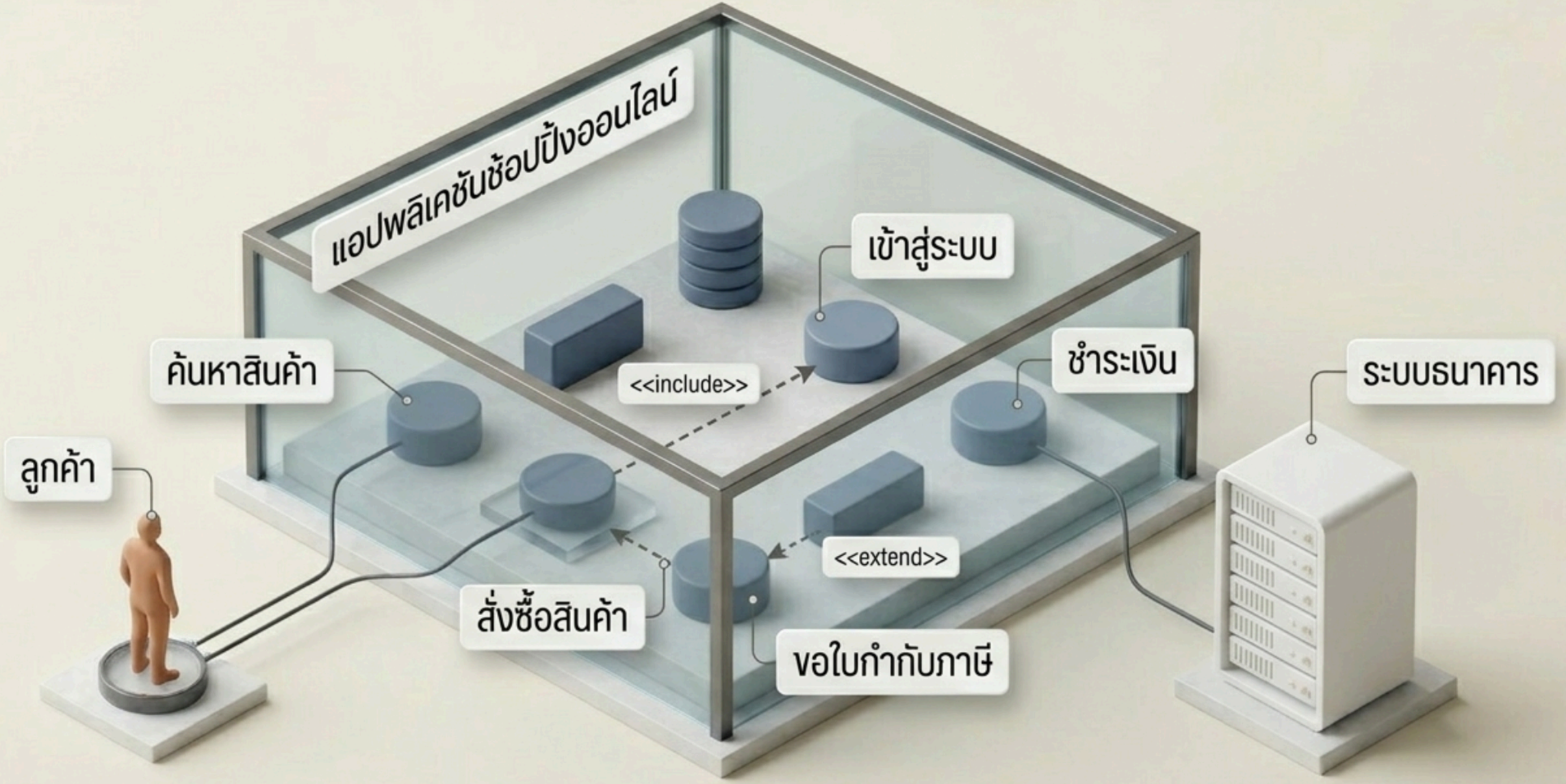
จัดกลุ่มผู้ใช้งานที่มีสิทธิพื้นฐานเหมือนกัน
เพื่อลดความซ้ำซ้อนในแผนภาพ

**สัญลักษณ์มาตรฐาน:** เส้นทึบพร้อมหัวลูกศรสามเหลี่ยมกลวง

# เจาะลึกความแตกต่าง: <<include>> vs <<extend>>

| | <<include>> | <<extend>> |
|---|---|---|
| **ลักษณะการทำงาน** (Behavior) | ต้องทำเสมอ (Mandatory)<br>- ระบบหลักจะทำงานไม่เสร็จถ้าขาดสิ่งนี้ | ทางเลือก (Optional)<br>- เป็นฟังก์ชันเสริม ทำหรือไม่ทำก็ได้ |
| **ทิศทางลูกศร** (Arrow Direction) | ชี้ ออกจาก Use Case หลัก<br>(ชี้ไปหาฟังก์ชันที่ถูกเรียกใช้) | ชี้ กลับเข้าหา Use Case หลัก<br>(ฟังก์ชันเสริมวิ่งเข้ามาหา) |
| **ตัวอย่าง** E-Commerce |  <br>สั่งซื้อสินค้า --> <<include>> --> ชำระเงิน |  <br>สั่งซื้อสินค้า <-- <<extend>> <-- ใส่โค้ดส่วนลด |

# ภาพรวมระบบ: ตัวอย่าง E-Commerce Platform



# Checklist: ตรวจสอบความถูกต้องก่อนนำไปใช้งาน

## 5 ข้อที่ต้องเช็คก่อนวาด Use Case Diagram ให้สมบูรณ์

- ✅ **หา Actor ให้ครบ:** มีทั้งผู้ใช้งานที่เป็นคน และระบบภายนอก (API) หรือยัง?
- ✅ **ใช้คำกริยา:** ชื่อ Use Case ทุกตัวเป็น ‘การกระทำ’ (เช่น สมัคร, ล็อกอิน) ใช่ไหม?
- ✅ **มีกรอบเขตชัดเจน:** วาดสี่เหลี่ยม System Boundary คลุมฟังก์ชันทั้งหมดแล้วหรือยัง?
- ✅ **ทิศทางลูกศรถูกต้อง:** <<include>> ชี้ออกไปฟังก์ชั่นย่อย / <<extend>> ชี้เข้าหาฟังก์ชันหลัก
- ✅ **ไม่ซับซ้อนเกินไป:** มุ่งเน้นที่ ‘ระบบทำอะไร’ ไม่ใช่ ‘ระบบทำอย่างไร’ (What, not How).

**โลโก้รายวิชา OOAD** | พร้อมสร้างแบบจำลองธุรกิจดิจิทัลของคุณแล้วหรือยัง?