กระบวนการวิศวกรรมความต้องการ และ User Stories

วัตถุประสงค์การเรียนรู้

- อธิบายกระบวนการวิศวกรรมความต้องการ
- เขียนเรื่องราวผู้ใช้ (User Stories) ที่ชัดเจน สมบูรณ์ และทดสอบได้
- กำหนดเกณฑ์การยอมรับ (Acceptance Criteria) ที่เป็นแบบ SMART (เฉพาะเจาะจง วัดผลได้ บรรลุได้ เกี่ยวข้อง มีกรอบเวลา)
- จัดลำดับความสำคัญ (Prioritize) เรื่องราวผู้ใช้ตามความสำคัญ
- เขียนเรื่องราวผู้ใช้สำหรับโครงการของตัวเอง

วิศวกรรมความต้องการ
(Requirements Engineering)

- **นิยาม:** กระบวนการในการค้นหา บันทึก วิเคราะห์ ตรวจสอบ และจัดการความต้องการของระบบ
- **ทำไมจึงสำคัญ:**
  - 70% ของการล้มเหลวของโครงการ = ความต้องการไม่ชัดเจน
  - ข้อผิดพลาดของความต้องการ (Requirements) = แพงที่สุด (ต้องทำใหม่ทั้งระบบ)
  - ความต้องการที่ดี = ประหยัดเวลา + ลดบั๊ก (bug)
- **ตัวอย่าง:**
  - ความต้องการที่ไม่ดี (Bad Requirements):
  - "ระบบต้องเร็ว" (ไม่ชัดเจน)

  - ความต้องการที่ดี (Good Requirements):
  - "หน้าเข้าสู่ระบบ (Login page) ต้องโหลดได้ภายใน < 2 วินาที โดยมีอัตราความผิดพลาด (error rate) < 5%"

ประเภทของความต้องการ
(Types of Requirements)

- **ความต้องการเชิงหน้าที่ (Functional Requirements – FR)**
  - สิ่งที่ระบบทำได้
- ตัวอย่าง (Examples):
  - ผู้ใช้สามารถเข้าสู่ระบบ (login) ด้วยอีเมล (email) และรหัสผ่าน (password)
  - ระบบสามารถส่งการแจ้งเตือนทางอีเมล (email notification)
  - ผู้ดูแลระบบ (Admin) สามารถสร้างรายงานการขาย (sales report)
  - ระบบสามารถคำนวณส่วนลด (discount) โดยอัตโนมัติ

ประเภทของความต้องการ
(Types of Requirements)

- **ความต้องการที่ไม่ใช่เชิงหน้าที่ (Non-Functional Requirements – NFR)**
  - วิธีที่ระบบควรทำงาน (คุณลักษณะด้านคุณภาพ – quality attributes)

- ประสิทธิภาพ (Performance):
  - เวลาตอบสนอง (Response time) < 2 วินาที
  - รองรับผู้ใช้พร้อมกัน (concurrent users) 1,000 คน
- ความปลอดภัย (Security):
  - รหัสผ่านเข้ารหัสด้วย SHA-256
  - เซสชัน (Session) หมดอายุหลัง 30 นาที
- ความใช้งานง่าย (Usability):
  - อินเทอร์เฟซ (Interface) ง่ายสำหรับผู้ใช้ที่ไม่มีความรู้ทางเทคนิค
  - รองรับการใช้งานบนมือถือ (Mobile responsive)
- ความน่าเชื่อถือ (Reliability):
  - เวลาทำงาน (Uptime) 99.9%
  - เวลาเฉลี่ยในการซ่อมแซม (MTTR – Mean Time To Repair) < 1 ชั่วโมง

ประเภทของความต้องการ
(Types of Requirements)

- **ความต้องการทางธุรกิจ (Business Requirements)**
  - เหตุผลที่โครงการถูกสร้างขึ้น
    - ลดต้นทุนการหาลูกค้าใหม่ (customer acquisition cost) 20%
    - เพิ่มรายได้ 30% ใน 1 ปี
    - มีผู้ใช้งานที่ใช้งานอย่างแข็งขัน (active users) 100,000 คนภายในสิ้นปี
- **ข้อจำกัด (Constraints)**
  - ข้อจำกัดที่ต้องปฏิบัติตาม
    - ด้านเทคนิค (Technical):
      - ต้องใช้ Java/Spring Boot
      - ต้องเข้ากันได้กับ MySQL 5.7 ขึ้นไป
    - ด้านธุรกิจ (Business):
      - งบประมาณ (Budget): $50,000
      - ระยะเวลา (Timeline): 6 เดือน
      - ทีม (Team): 5 คน
    - ด้านกฎหมาย (Legal):
      - ต้องปฏิบัติตามข้อกำหนด GDPR
      - ต้องมีการสำรองข้อมูล (data backup)

ประเภทของความต้องการ
(Types of Requirements)

- **ความต้องการที่ไม่ใช่เชิงหน้าที่ (Non-Functional Requirements – NFR)**
  - วิธีที่ระบบควรทำงาน (คุณลักษณะด้านคุณภาพ – quality attributes)

- ประสิทธิภาพ (Performance):
  - เวลาตอบสนอง (Response time) < 2 วินาที
  - รองรับผู้ใช้พร้อมกัน (concurrent users) 1,000 คน
- ความปลอดภัย (Security):
  - รหัสผ่านเข้ารหัสด้วย SHA-256
  - เซสชัน (Session) หมดอายุหลัง 30 นาที
- ความใช้งานง่าย (Usability):
  - อินเทอร์เฟซ (Interface) ง่ายสำหรับผู้ใช้ที่ไม่มีความรู้ทางเทคนิค
  - รองรับการใช้งานบนมือถือ (Mobile responsive)
- ความน่าเชื่อถือ (Reliability):
  - เวลาทำงาน (Uptime) 99.9%
  - เวลาเฉลี่ยในการซ่อมแซม (MTTR – Mean Time To Repair) < 1 ชั่วโมง

กระบวนการวิศวกรรมความต้องการ
(Requirements Engineering Process)

1. การรวบรวม (Elicitation - Gathering Requirements)

- สัมภาษณ์ผู้มีส่วนได้ส่วนเสีย (stakeholders)
- สังเกตขั้นตอนการทำงาน (workflows)
- สำรวจผู้ใช้ (Survey users)

→ ผลลัพธ์: รายการความต้องการ (Requirements list) ดิบ

2. การวิเคราะห์ (Analysis - Understanding Requirements)

- จัดประเภท (Classify) (FR vs NFR)
- จัดลำดับความสำคัญ (Prioritize)
- ระบุความขัดแย้ง (Identify conflicts)

→ ผลลัพธ์: ความต้องการที่จัดลำดับความสำคัญแล้ว

3. การระบุรายละเอียด (Specification - Document Requirements)

- เขียนอย่างชัดเจน
- ใช้เทมเพลต (templates) (User Stories)
- สร้างไดอะแกรม (diagrams)

→ ผลลัพธ์: เอกสารข้อกำหนดซอฟต์แวร์ (SRS Document)

4. การตรวจสอบความถูกต้อง (Validation - Verify Requirements)
- ทบทวนกับผู้มีส่วนได้ส่วนเสีย (stakeholders)
- ตรวจสอบความสมบูรณ์ (completeness)
- ตรวจสอบความสอดคล้อง (consistency)

→ ผลลัพธ์: ความต้องการที่ได้รับการอนุมัติ

5. การจัดการ (Management - Manage Changes)
- ติดตามความต้องการ (Track requirements)
- จัดการคำขอการเปลี่ยนแปลง (change requests)
- ปรับปรุงเอกสาร (Update documentation)

→ ผลลัพธ์: เมทริกซ์การติดตามความต้องการ (Requirements traceability matrix)

เทคนิคการรวบรวมความต้องการ
(Elicitation Techniques)

1. การสัมภาษณ์ (Interviews)

```
รูปแบบ (Format): แบบตัวต่อตัวหรือกลุ่ม
เวลา (Time): 30-60 นาที
เหมาะสำหรับ (Best for): ความเข้าใจโดยละเอียด

คำถามที่ควรถาม:
- ปัญหาที่คุณเผชิญในปัจจุบันคืออะไร?
- คุณต้องการฟีเจอร์ (features) อะไรมากที่สุด?
- คุณวัดความสำเร็จอย่างไร?
```

2. การสังเกต (Observation)

```
รูปแบบ (Format): ติดตามผู้ใช้ในระหว่างการทำงานประจำวัน
เวลา (Time): 2-4 ชั่วโมง
เหมาะสำหรับ (Best for): เข้าใจขั้นตอนการทำงานจริง (actual workflow)

สิ่งที่ควรสังเกต:
- วิธีที่พวกเขาทำงานจริง (เทียบกับสิ่งที่พวกเขาบอก)
- จุดที่มีปัญหา (Pain points)
- วิธีแก้ไขชั่วคราว (Workarounds)
```

เทคนิคการรวบรวมความต้องการ
(Elicitation Techniques)

3. แบบสำรวจ (Surveys)

```
รูปแบบ (Format): แบบฟอร์มออนไลน์
เวลา (Time): 5-10 นาทีต่อคน
เหมาะสำหรับ (Best for): รวบรวมข้อมูลจากหลายคน

เคล็ดลับ (Tips):
- ทำให้คำถามเรียบง่าย
- ใช้ตัวเลือกหลายทาง (multiple choice)
- เสนอสิ่งจูงใจ (incentive) (ส่วนลด คูปอง)
```

4. เวิร์กช็อป (Workshops)

```
รูปแบบ (Format): เซสชันกลุ่ม
เวลา (Time): 2-4 ชั่วโมง
เหมาะสำหรับ (Best for): สร้างฉันทามติ (consensus)

กิจกรรม:
- ระดมความคิดฟีเจอร์ (Brainstorm features)
- จัดลำดับความสำคัญร่วมกัน (Prioritize together)
- แก้ไขความขัดแย้ง (Resolve conflicts)
```

เทคนิคการรวบรวมความต้องการ
(Elicitation Techniques)

5. การสร้างต้นแบบ (Prototyping)

```
รูปแบบ (Format): สร้างโมเดลจำลอง/ต้นแบบ (mockup/prototype)
เวลา (Time): 1-2 สัปดาห์
เหมาะสำหรับ (Best for): ทำให้ความต้องการที่ไม่ชัดเจนชัดเจนขึ้น

ประโยชน์ (Benefits):
- ผู้ใช้สามารถเห็นการแสดงผลภาพ
- ได้รับคำติชมที่เป็นรูปธรรมมากขึ้น
- ระบุความต้องการที่ขาดหายไป
```

เรื่องราวผู้ใช้ (User Stories)

- **นิยาม:** คำอธิบายง่ายๆ ของฟีเจอร์จากมุมมองของผู้ใช้
- **รูปแบบ**
  - ในฐานะ [ประเภทผู้ใช้] (As a [User Type]),
  - ฉันต้องการ [การกระทำ/ฟีเจอร์] (I want [Action/Feature]),
  - เพื่อที่ [คุณค่าทางธุรกิจ/ประโยชน์] (so that [Business Value/Benefit])

เรื่องราวผู้ใช้ (User Stories)

ในฐานะลูกค้า (As a customer),
ฉันต้องการกรองสินค้าตามราคา (I want to filter products by price),
เพื่อที่ฉันจะหาสินค้าที่ราคาไม่แพงได้อย่างรวดเร็ว (so that I can find affordable items quickly)

ในฐานะผู้ดูแลระบบ (As an admin),
ฉันต้องการดูกิจกรรมของผู้ใช้ทั้งหมดในบันทึก (I want to view all user activities in a log),
เพื่อที่ฉันจะสามารถติดตามความปลอดภัยของระบบ (so that I can monitor system security)

ในฐานะนักพัฒนา (As a developer),
ฉันต้องการตั้งค่าไปป์ไลน์ CI/CD โดยอัตโนมัติ (I want to set up CI/CD pipeline automatically),
เพื่อที่ฉันจะสามารถปรับใช้ (deploy) ได้เร็วขึ้นโดยไม่ต้องทำด้วยมือ (so that I can deploy faster without manual steps)

Good vs Bad User Stories

• ไม่ดี (BAD)

"สร้างระบบเข้าสู่ระบบ (Build a login system)"

– ไม่ชัดเจนว่าต้องการอะไร

– ไม่มีคุณค่าทางธุรกิจ

• ดี (GOOD)

"ในฐานะผู้ใช้ที่ลงทะเบียนแล้ว (As a registered user), ฉันต้องการเข้าสู่ระบบด้วยอีเมล/รหัสผ่าน (I want to login with email/password),

เพื่อที่ฉันจะเข้าถึงแดชบอร์ดส่วนตัวของฉันได้อย่างปลอดภัย (so that I can access my personal dashboard securely)"

- ประเภทผู้ใช้ชัดเจน
- การกระทำชัดเจน
- คุณค่าทางธุรกิจชัดเจน

รูปแบบและรายละเอียดของเรื่องราวผู้ใช้
(User Story Format & Details)

1. หัวข้อ (Title – Judul)

- สั้นและสื่อความหมาย
- ไม่เกิน 10 คำ
  - ไม่ดี: "ทำระบบเข้าสู่ระบบ (Implement Login)"
  - ดี: "ผู้ใช้สามารถเข้าสู่ระบบด้วยอีเมลและรหัสผ่าน (User can login with email and password)"

2. ในฐานะ... ฉันต้องการ... เพื่อที่... (As a... I want... so that...)

- ใช้เทมเพลต
- 1–3 บรรทัด

  ในฐานะผู้ใช้ใหม่ (As a new user),
  ฉันต้องการลงทะเบียนด้วยข้อมูลเพียงเล็กน้อย (I want to register with minimal information),
  เพื่อที่ฉันจะสามารถเข้าร่วมแพลตฟอร์มได้อย่างรวดเร็ว (so that I can quickly join the platform)

รูปแบบและรายละเอียดของเรื่องราวผู้ใช้
(User Story Format & Details)

3. เกณฑ์การยอมรับ (Acceptance Criteria – AC)

- เงื่อนไขที่ต้องปฏิบัติตามเพื่อให้ถือว่า "เสร็จสมบูรณ์ (Done)"
- ใช้รูปแบบ GIVEN–WHEN–THEN (BDD style)
  - กำหนดให้ (GIVEN): ผู้ใช้อยู่ที่หน้าลงทะเบียน
  - เมื่อ (WHEN): ผู้ใช้ป้อนอีเมลและรหัสผ่านที่ถูกต้อง
  - แล้ว (THEN): บัญชีถูกสร้างและเข้าสู่ระบบสำเร็จ
  - และ (AND): ผู้ใช้ได้รับอีเมลยืนยัน
  - และ (AND): ผู้ใช้ถูกเปลี่ยนเส้นทางไปยังแดชบอร์ด

รูปแบบและรายละเอียดของเรื่องราวผู้ใช้
(User Story Format & Details)

4. คะแนนเรื่องราว (Story Points – Effort estimate)

- ขนาดของความซับซ้อน (1, 2, 3, 5, 8, 13, 21)
- มาตราส่วนฟีโบนัชชี (Fibonacci scale)
  - คะแนน (point): ง่ายมาก < 1 ชั่วโมง
  - 3 คะแนน (points): ง่าย 2–3 ชั่วโมง
  - 5 คะแนน (points): ปานกลาง 1 วัน
  - 8 คะแนน (points): ซับซ้อน 2–3 วัน
  - 13 คะแนน (points): ซับซ้อนมาก 1 สัปดาห์

รูปแบบและรายละเอียดของเรื่องราวผู้ใช้
(User Story Format & Details)

5. การทดสอบการยอมรับ (Acceptance Tests)

- กรณีทดสอบ (Test cases) ที่ตรวจสอบ AC

- กรณีทดสอบที่ 1 (Test Case 1): การลงทะเบียนที่ถูกต้อง
  - ข้อมูลนำเข้า (Input): อีเมลถูกต้อง รหัสผ่าน >= 8 ตัวอักษร
  - ผลลัพธ์ที่คาดหวัง (Expected): บัญชีถูกสร้าง อีเมลถูกส่ง

- กรณีทดสอบที่ 2 (Test Case 2): รูปแบบอีเมลไม่ถูกต้อง
  - ข้อมูลนำเข้า (Input): "invalid.email"
  - ผลลัพธ์ที่คาดหวัง (Expected): แสดงข้อความข้อผิดพลาด

- กรณีทดสอบที่ 3 (Test Case 3): อีเมลซ้ำ
  - ข้อมูลนำเข้า (Input): อีเมลที่มีอยู่แล้ว
  - ผลลัพธ์ที่คาดหวัง (Expected): ข้อผิดพลาด "อีเมลถูกลงทะเบียนแล้ว"

เคล็ดลับการเขียนเกณฑ์การยอมรับ
(Acceptance Criteria Writing Tips)

- **กฎทอง (Golden Rules)**

1. ใช้ Given–When–Then (BDD)

```
กำหนดให้ (GIVEN) [เงื่อนไขเบื้องต้น - precondition]
เมื่อ (WHEN) [การกระทำ - action]
แล้ว (THEN) [ผลลัพธ์ - result]

ตัวอย่าง (Example):
กำหนดให้ผู้ใช้อยู่ที่หน้าเข้าสู่ระบบ (GIVEN user is on login page)
เมื่อผู้ใช้ป้อนอีเมลและรหัสผ่านที่ถูกต้อง (WHEN user enters correct email and password)
แล้วผู้ใช้เข้าสู่ระบบและถูกเปลี่ยนเส้นทางไปยังแดชบอร์ด (THEN user is logged in and redirected to dashboard)
```

เคล็ดลับการเขียนเกณฑ์การยอมรับ
(Acceptance Criteria Writing Tips)

2. ระบุให้เฉพาะเจาะจง (ไม่คลุมเครือ) (Be Specific – not vague)

```
❌ คลุมเครือ (VAGUE): "เวลาตอบสนองเร็ว (Fast response time)"
✅ เฉพาะเจาะจง (SPECIFIC): "การตอบสนอง API ใน < 200ms ที่เปอร์เซ็นไทล์ที่ 95 (API response in < 200ms at 95th percentile)"
```

3. วัดผลได้ (Be Measurable)

```
❌ วัดผลไม่ได้ (UNMEASURABLE): "UI ที่ดี (Good UI)"
✅ วัดผลได้ (MEASURABLE): "UI โหลดใน < 2 วินาทีบนเครือข่าย 4G (UI loads in < 2 seconds on 4G network)"
```

เคล็ดลับการเขียนเกณฑ์การยอมรับ
(Acceptance Criteria Writing Tips)

4. รวมกรณีขอบเขต (Include Edge Cases)

✅ "ผู้ใช้สามารถเข้าสู่ระบบด้วยอีเมลหรือชื่อผู้ใช้ (User can login with email or username)"
✅ "รหัสผ่านคำนึงถึงตัวพิมพ์ใหญ่-เล็ก (Password case-sensitive)"
✅ "ความพยายามเข้าสู่ระบบสูงสุด 5 ครั้งก่อนล็อค (Max 5 login attempts before lock)"

5. รวมความต้องการที่ไม่ใช่เชิงหน้าที่ (Include Non-Functional Requirements)

✅ "การตอบสนอง API < 200ms"
✅ "รองรับผู้ใช้พร้อมกัน 1,000 คน"
✅ "เวลาทำงาน 99.9%"

User Story Template

- >>> ตัวอย่าง

การรวบรวมความต้องการ
(Requirements Gathering)

1. การระบุผู้มีส่วนได้ส่วนเสีย (Stakeholder Identification)

```
ผู้มีส่วนได้ส่วนเสียโดยตรง (Direct Stakeholders):
- ผู้ใช้ปลายทาง (End Users) (สำคัญที่สุด)
- เจ้าของผลิตภัณฑ์ (Product Owner)
- ทีมพัฒนา (Development Team)
- ลูกค้า (Customers)
- เจ้าของธุรกิจ (Business Owners)

ผู้มีส่วนได้ส่วนเสียโดยอ้อม (Indirect Stakeholders):
- ทีมสนับสนุน (Support team)
- ทีมปฏิบัติการ (Operations team)
- ทีมรักษาความปลอดภัย (Security team)
- ทีมกฎระเบียบ (Compliance team)
- ผู้ใช้ของคู่แข่ง (Competitors' users)
```

การรวบรวมความต้องการ
(Requirements Gathering)

2. ความต้องการจากแหล่งต่างๆ (Requirements from Different Sources)

จากผู้ใช้ (From Users):

```
"ฉันต้องการติดตามคำสั่งซื้อของฉันแบบเรียลไทม์ (I want to track my order in real-time)"
→ ความต้องการ: การติดตามคำสั่งซื้อแบบเรียลไทม์ (Real-time order tracking)

"ต้องการให้เร็วกว่าระบบเก่า (Need faster than old system)"
→ ความต้องการ: การตอบสนอง API < 2 วินาที
```

จากธุรกิจ (From Business):

```
"เราต้องการเพิ่มรายได้ 50% ในปีนี้ (We want to increase revenue by 50% this year)"
→ ความต้องการ: ปรับปรุงอัตราการแปลง (conversion rate) 30%

"คู่แข่งมีฟีเจอร์ X เราต้องการมันด้วย (Competitors have feature X, we need it too)"
→ ความต้องการ: ฟีเจอร์ X
```

จากทีมเทคนิค (From Technical Team):

```
"เซิร์ฟเวอร์เก่ารับโหลดนี้ไม่ไหว (Old server cannot handle this load)"
→ ความต้องการ: รองรับผู้ใช้พร้อมกัน 10,000 คน

"โค้ดเบสยุ่ง ดูแลรักษายาก (Code base is messy, hard to maintain)"
→ ความต้องการ: ปรับปรุงสถาปัตยกรรม (Refactor architecture)
```

การจัดทำเอกสารความต้องการ
(Requirements Documentation)

- >>> องค์ประกอบ SRS

การจัดลำดับความสำคัญของงาน
(Backlog Prioritization)

- **ทำไมการจัดลำดับความสำคัญจึงสำคัญ?**

```
ไม่มีการจัดสำดับความสำคัญ (Without prioritization):
- สร้างฟีเจอร์ที่ไม่สำคัญก่อน
- เวลาหมดสำหรับฟีเจอร์สำคัญ
- ลูกค้าไม่พอใจ

มีการจัดสำดับความสำคัญ (With prioritization):
- สร้างฟีเจอร์คุณค่าสูงก่อน
- MVP สามารถส่งมอบได้เร็วขึ้น
- ได้รับคำติชมเร็ว
```

วิธีการจัดลำดับความสำคัญ (Prioritization Methods)

- **วิธี MoSCoW**
- M = ต้องมี (MUST)
  (จำเป็น คุณค่าสูง – required, high value)
- S = ควรมี (SHOULD)
  (สำคัญ คุณค่าปานกลาง – important, medium value)
- C = อาจมี (COULD)
  (ดีที่จะมี ไม่สำคัญมาก – nice to have, not critical)
- W = จะไม่มี (WON'T)
  (ไม่อยู่ในรุ่นนี้ – not in this release)

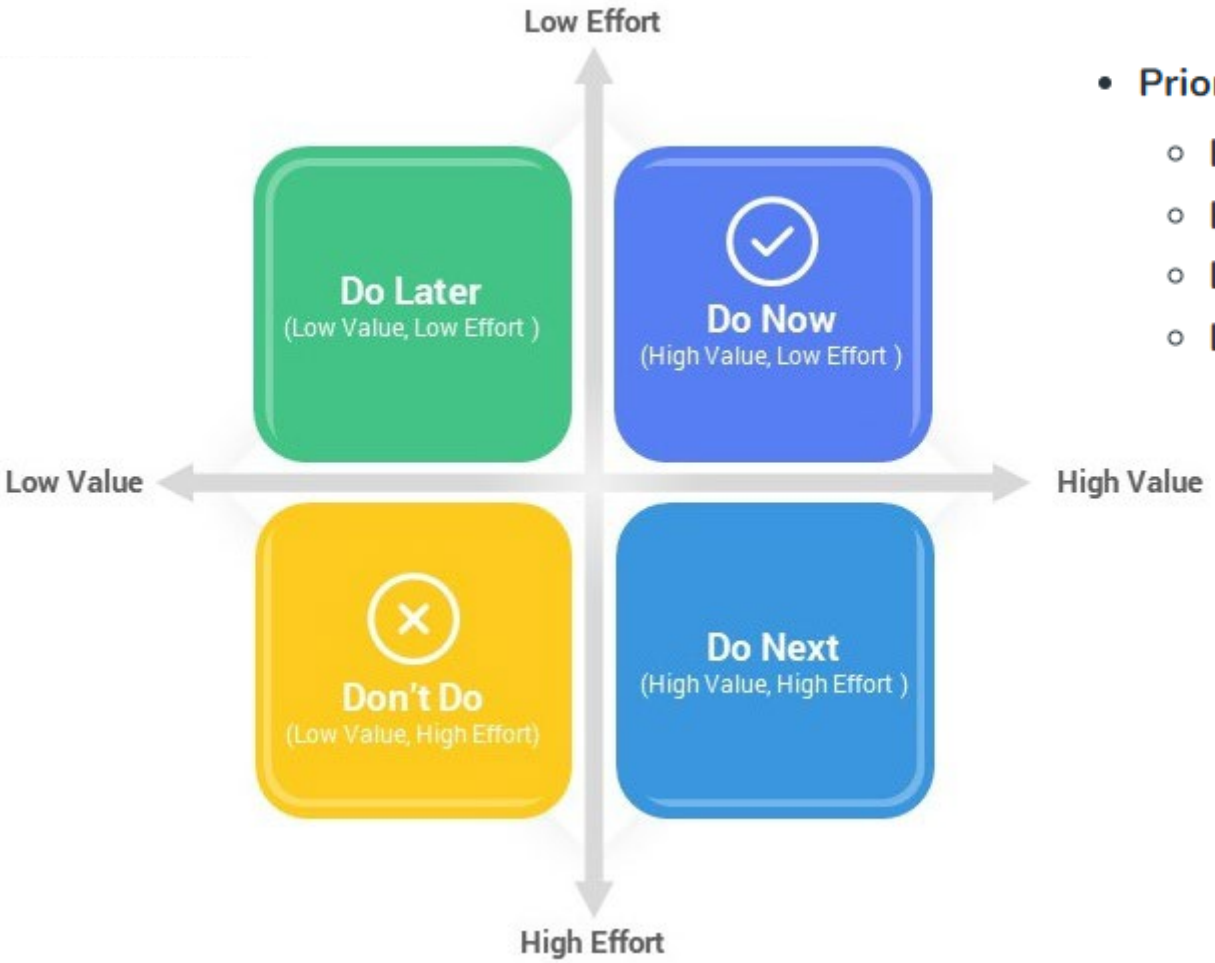
```
ตัวอย่าง (Example):
ต้องมี (MUST):
- การเข้าสู่ระบบของผู้ใช้ (User login)
- รายการสินค้า (Product listing)
- ตะกร้าสินค้า (Shopping cart)

ควรมี (SHOULD):
- รีวิวสินค้า (Product reviews)
- รายการโปรด (Wishlist)
- คำแนะนำ (Recommendation)

อาจมี (COULD):
- การแชร์โซเชียล (Social sharing)
- เกมมิฟิเคชัน (Gamification)
- ทดลองใช้ AR (AR try-on)

จะไม่มี (WON'T):
- ประสบการณ์ VR (VR experience)
- แชทบอท AI (AI chatbot)
- หลายภาษา (v2) (Multi-language)
```

เมทริกซ์คุณค่าเทียบกับความพยายาม
(Value vs Effort Matrix)



- **Prioritize Tasks:**
  - **High Value, Low Effort:** Top priority maximum impact with minimal work.
  - **High Value, High Effort:** Worth doing if benefits justify the effort.
  - **Low Value, Low Effort:** Optional; do if extra capacity exists.
  - **Low Value, High Effort:** Usually avoid; minimal return for high investment.

การให้คะแนน RICE

- RICE = การเข้าถึง × ผลกระทบ × ความมั่นใจ / ความพยายาม
  (RICE = Reach × Impact × Confidence / Effort)

R = การเข้าถึง (Reach) (กี่คนที่ได้รับผลกระทบ: มาตราส่วน 0–100 – how many users affected)
I = ผลกระทบ (Impact) (ผลกระทบต่อผู้ใช้แต่ละคนมากแค่ไหน: มาตราส่วน 1–3 – how much each user affected)
C = ความมั่นใจ (Confidence) (คุณมั่นใจแค่ไหน: 0–100% – how sure you are)
E = ความพยายาม (Effort) (ในหน่วยคน-เดือน – in person-months)

การให้คะแนน RICE

- ตัวอย่าง (Example)

```
ฟีเจอร์: เพิ่มรีวิวสินค้า (Add product review)
การเข้าถึง (Reach): 50,000 ผู้ใช้
ผลกระทบ (Impact): 2 (ปานกลาง - ดีที่จะมี - medium - nice to have)
ความมั่นใจ (Confidence): 80%
ความพยายาม (Effort): 2 เดือน

RICE = (50,000 x 2 x 0.8) / 2 = 40,000

ฟีเจอร์: ปรับปรุงประสิทธิภาพการชำระเงิน (Improve checkout performance)
การเข้าถึง (Reach): 100,000 ผู้ใช้
ผลกระทบ (Impact): 3 (สูง - ฟีเจอร์หลัก - high - core feature)
ความมั่นใจ (Confidence): 90%
ความพยายาม (Effort): 1 เดือน

RICE = (100,000 x 3 x 0.9) / 1 = 270,000
→ คะแนน RICE สูงกว่า = ความสำคัญสูงกว่า (Higher RICE score = higher priority)
```

โมเดล Kano – จัดหมวดหมู่ฟีเจอร์

- เพื่อให้รู้ว่าฟีเจอร์แบบไหนสำคัญต่อลูกค้า โมเดลแบ่งฟีเจอร์เป็น **3 ประเภท**:

- ต้องมี (Must-have – Basic): **สำคัญ:** ต้องมีให้ครบ
  - ถ้าขาด: ลูกค้าไม่พอใจ
  - ถ้ามี: ลูกค้าเฉยๆ
  - ตัวอย่าง (Example): ตะกร้าสินค้าที่ใช้งานได้

>>> การใช้งาน Kano model

- ประสิทธิภาพ (Performance -> Linear): **สำคัญ:** ลงทุนให้มากขึ้น ลูกค้ายิ่งพอใจ
  - มากขึ้น = ความพอใจมากขึ้น
  - ตัวอย่าง (Example): การชำระเงินที่รวดเร็ว ตัวเลือกการชำระเงินมากมาย

- ผู้สร้างความพึงพอใจ (Delighter – Excitement): **สำคัญ:** ถ้ามีงบประมาณให้เพิ่ม เพื่อสร้างความแตกต่าง
  - ถ้ามี: ลูกค้าพึงพอใจ
  - ถ้าขาด: ลูกค้าไม่สนใจ
  - ตัวอย่าง (Example): การทดลองใช้ AR เกมมิฟิเคชัน

- กลยุทธ์ (Strategy): สมดุลทั้งสามประเภท
  - รวมต้องมีทั้งหมด (Include ALL must-haves)
  - ลงทุนในประสิทธิภาพ (Invest in performance)
  - เพิ่มผู้สร้างความพึงพอใจบางอย่าง (ถ้ามีเวลา/งบประมาณ) (Add some delighters – if time/budget)

เทมเพลตการจัดลำดับความสำคัญของงาน
(Backlog Prioritization Template)

ความสำคัญ (Priority)	เรื่องราว (Story)	คุณค่า (Value)	ความพยายาม (Effort)	MoSCoW	RICE	เจ้าของ (Owner)
1	การเข้าสู่ระบบของผู้ใช้	สูง	8 คะแนน	ต้องมี	สูง	หัวหน้านักพัฒนา
2	รายการสินค้า	สูง	13 คะแนน	ต้องมี	สูง	นักพัฒนา
3	ตะกร้าสินค้า	สูง	8 คะแนน	ต้องมี	สูง	QA
4	การชำระเงิน	สูง	13 คะแนน	ต้องมี	สูง	หัวหน้านักพัฒนา
5	การค้นหาสินค้า	ปานกลาง	5 คะแนน	ควรมี	ปานกลาง	นักพัฒนา
6	รีวิวผู้ใช้	ปานกลาง	8 คะแนน	ควรมี	ปานกลาง	QA
7	รายการโปรด	ปานกลาง	5 คะแนน	ควรมี	ต่ำ	นักพัฒนา
8	แชร์โซเชียล	ต่ำ	3 คะแนน	อาจมี	ต่ำ	ฝึกงาน

กิจกรรม

- **กิจกรรมที่ 1: – (Write User Stories)**
  - >>> รายละเอียด
- **กิจกรรมที่ 2: จัดลำดับความสำคัญของงาน (Prioritize Backlog)**
  - >>> รายละเอียด

Homework

- >>> รายละเอียด