WORKSHOP DAY 3

# AI Workflow Automation สำหรับโรงงานอาหารมูลค่าสูง

Workflow Automation Architecture • Prototype Plan • Executive Pitch

**เปลี่ยน Hero Use Case ให้เป็นระบบตัดสินใจที่ตรวจสอบได้ ไม่ใช่แค่ workflow demo**

**Trigger → Risk → Decision → Evidence Log**

# วันนี้เราจะทำอะไร

จาก Hero Use Case ไปสู่ระบบตัดสินใจที่ตรวจสอบได้

**ผลลัพธ์ที่ต้องได้**

- เข้าใจ workflow automation ในบริบทโรงงานอาหาร
- ออกแบบ trigger, data contract, risk logic และ decision boundary
- สร้าง prototype plan ที่ยังไม่แตะระบบ production จริง
- เตรียม executive pitch ที่ผูกกับ KPI และ ROI

**เป้าหมาย: ระบบเล็ก แต่คิดครบเรื่องความเสี่ยง**

เข้าใจ → ออกแบบ → ตรวจสอบ → Pitch

ระบบตัดสินใจแบบมีหลักฐาน

ลด false alarm ด้วย trigger ที่วัดได้

คุม decision boundary ในจุดเสี่ยง

พิสูจน์ KPI / ROI จาก baseline จริง

# จาก Day 2 มาสู่ Day 3

ต่อยอดจากการเลือก Hero Use Case ให้กลายเป็นระบบจริง

**ความต่างของวันนี้**

- Day 2 เน้นเลือก use case และประเมิน feasibility
- Day 3 เน้น architecture, control และ evidence
- ไม่เริ่มจาก model แต่เริ่มจาก decision point
- ต้องตอบได้ว่า AI ห้ามตัดสินใจเองตรงไหน

**Use Case ที่ดีต้องเข้า workflow จริง**

Use Case → Blueprint → Prototype → Validation → Pitch

Day 3 Output

- Automation Blueprint
- Exception Pack
- Prototype Plan
- Business Case

# Workflow Automation คืออะไร

ระบบที่เปลี่ยนเหตุการณ์หน้างานให้เป็น action ที่มีหลักฐาน

**นิยามใช้งานง่าย**

- รับสัญญาณจาก process, quality, traceability หรือ logistics
- ตรวจคุณภาพข้อมูลก่อนใช้ตัดสินใจ
- คำนวณ risk หรือเลือก rule ที่เหมาะกับสถานการณ์
- ส่งให้คนที่มีอำนาจอนุมัติและเก็บหลักฐานทุกครั้ง

**จุดจบไม่ใช่ dashboard แต่คือ action + evidence**

Trigger → Data → Risk → Decision → Action → Evidence

**หลักคิด**

เริ่มจากเหตุการณ์หน้างาน แล้วจบที่หลักฐานตรวจย้อนหลังได้

**คำถามหลัก**

ถ้าถูก audit อีก 6 เดือน จะอธิบายได้หรือไม่

# ทำไมโรงงานอาหารต้องระวังมากกว่า

AI ในโรงงานอาหารกระทบ safety, quality, compliance และ brand trust

**ความเสี่ยงสำคัญ**

- Food safety decision มีต้นทุนความผิดพลาดสูง
- Hold / release / recall ต้องมีผู้รับผิดชอบชัดเจน
- ข้อมูลต้องย้อนกลับถึง lot, batch, route และ parameter ได้
- ระบบต้องอธิบายได้ต่อ auditor และผู้บริหาร

**Productivity ต้องไม่มาก่อน food safety**

**Food Safety**

false negative แพงมาก: สินค้าเสี่ยงหลุดไปถึงลูกค้า

**Quality**

ระบบต้องเชื่อม defect, lot, operator, shift และ action

**Audit**

ต้องบอกได้ว่าใช้ข้อมูลชุดใด เวอร์ชันใด ใครอนุมัติ

**Business**

ต้องเห็นผลจริงจาก KPI ไม่ใช่แค่ demo ดูทันสมัย

# Automation ที่ดีไม่ใช่ให้ AI ตัดสินใจแทนคน

AI ควรช่วยเตรียมหลักฐาน จัดลำดับความเสี่ยง และเสนอทางเลือก

**หลักคิดสำหรับ supervisor**

- แยก “recommendation” ออกจาก “approval” ให้ชัด
- กำหนดสิทธิ์ override ตาม role, severity และ shift
- เก็บเหตุผลของมนุษย์ ไม่ใช่แค่ปุ่ม approve/reject
- มี fallback manual เมื่อระบบหรือข้อมูลล้มเหลว

**มนุษย์ยังเป็นเจ้าของ decision ที่มีความรับผิดชอบ**

Alarm → Triage → Approval → Controlled Action

Controlled Action

ต้องมี owner + SLA

ต้องมี fallback เมื่อข้อมูลขาด

ต้องมี evidence log

# Reference Architecture ของ Food Automation

แยกข้อมูล ตรรกะการตัดสินใจ และหลักฐานตรวจย้อนหลัง

## โครงสร้างหลัก

- Layer 1: process data จาก sensor, batch sheet, line speed
- Layer 2: quality & food safety records เช่น lab, inspection, complaint
- Layer 3: traceability context เช่น raw lot, packaging, warehouse, route
- Control plane: validation → scoring → approval → evidence archive

**เปลี่ยน model ได้โดยไม่ทำ traceability พัง**

**Layer 1: Process Data**
sensor • batch sheet • line speed

**Layer 2: Quality & Safety Records**
lab • inspection • complaint • release

**Layer 3: Traceability Context**
raw lot • packaging • warehouse • route

Validate → Score → Gate → Approve → Archive

Part 1

# Trigger Engineering

เปลี่ยนเหตุการณ์หน้างานให้เป็น trigger ที่วัดได้

# Trigger คืออะไร

Trigger คือเหตุการณ์ที่ทำให้ workflow เริ่มทำงาน

Trigger → Data → Risk → Decision → Action → Evidence

## Trigger ที่ดีต้องมี

- นิยามเหตุการณ์ชัด เช่น temp excursion, defect spike, lab result late
- มี threshold หรือ pattern ที่วัดได้
- มี time window เช่น 10 นาที, 30 นาที, 1 shift
- มี owner และ action ถัดไป

## Trigger ที่คลุมเครือจะสร้าง false alarm

**หลักคิด**

เริ่มจากเหตุการณ์หน้างาน แล้วจบที่หลักฐานตรวจย้อนหลังได้

**คำถามหลัก**

ถ้าถูก audit อีก 6 เดือน จะอธิบายได้หรือไม่

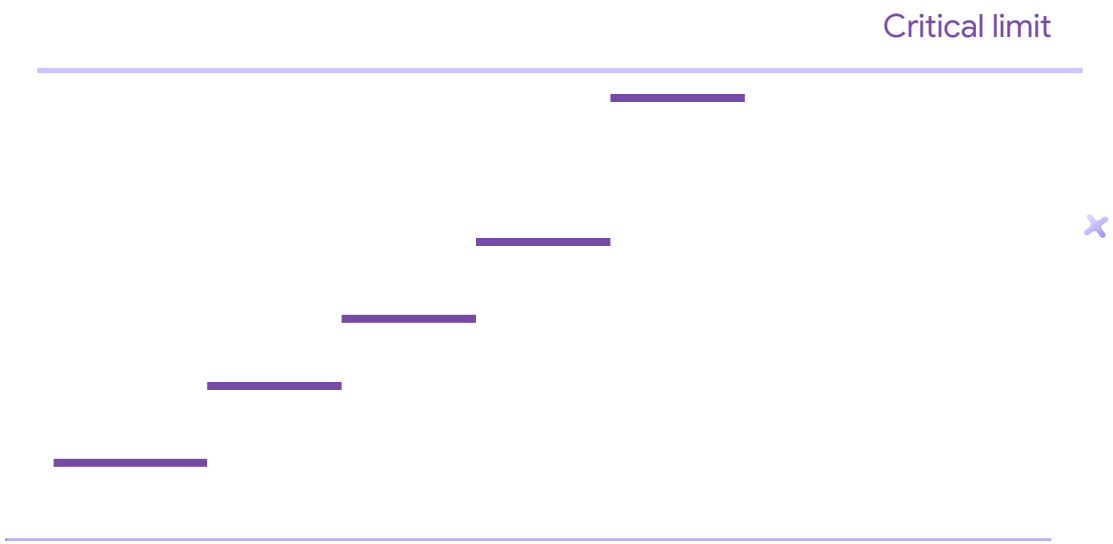
# Trigger แบบ Threshold

ใช้เมื่อมี spec, limit หรือ critical limit ที่ชัดเจน

**ตัวอย่าง**

- อุณหภูมิห้องเย็น > 8°C ต่อเนื่อง 12 นาที
- pH หลุดช่วงที่กำหนดใน batch fermentation
- fill level ต่ำกว่า minimum legal requirement
- ผล microbiology ไม่ผ่านเกณฑ์ release

**เหมาะกับ rule gate และ safety control**




**Threshold Trigger**

ค่าเกิน spec หรือ critical limit เป็นเวลาที่กำหนด

# Trigger แบบ Pattern

ใช้เมื่อปัญหาเกิดซ้ำตามเงื่อนไขบางอย่าง

**ตัวอย่าง**

- defect เพิ่มเฉพาะ film lot เดียว
- complaint สูงเฉพาะ shift กลางคืน
- cold-chain excursion เกิดซ้ำใน route เดิม
- sensor gap เกิดซ้ำหลัง maintenance บางประเภท

| | | |
|---|---|---|
| Shift A | Shift B | Shift C |
| Film Lot X | Line 2 | Line 3 |

**Pattern บอกว่าเราควรตรวจที่ไหนก่อน**

| Pattern Trigger |
|---|
| พบ pattern ซ้ำตาม shift, line หรือ material lot |

# Trigger แบบ Compound Risk

ใช้เมื่อความเสี่ยงเกิดจากหลายสัญญาณพร้อมกัน

**ตัวอย่าง**

- temperature exposure + shelf-life เหลือน้อย + route delay
- defect spike + raw material lot ใหม่ + operator change
- pH drift + culture lot ใหม่ + sensory score ต่ำลง
- sensor gap + high-risk SKU + shipment ใกล้ถึงลูกค้า

**เหมาะกับ risk scoring มากกว่า threshold เดี่ยว**

| Temp exposure | Shelf-life เหลือน้อย | Route delay |
|---|---|---|
| 12 นาที > limit | เหลือ 5 วัน | ดีเลย์ 6 ชม. |

**Risk Score สูงขึ้นเมื่อหลายสัญญาณเกิดพร้อมกัน**

| Compound Risk |
|---|
| เหมาะกับ decision ที่ต้องรวมหลายหลักฐาน |

# ลด False Alarm อย่างไร

Trigger ที่เตือนเยอะเกินไปจะทำให้คนเลิกเชื่อระบบ

**วิธีลด false alarm**

- ใช้ dwell time แทนการดูค่าเกินครั้งเดียว
- แยก product risk profile ตาม SKU หรือ process
- ตั้ง triage queue สำหรับ alert ที่ยังไม่ต้องหยุดทันที
- วัด precision: alert แล้วเกิด action จริงกี่เปอร์เซ็นต์

**Alert ที่ดีต้องนำไปสู่ action ที่มีคุณค่า**

Signal → Dwell time → Product risk → Triage

แยก alarm ที่หยุดทันที

แยก alarm ที่เข้า triage queue

วัด precision: แจ้งแล้วต้องมี action กี่ %

# แบบฝึก: เขียน Trigger ของ Use Case

ให้ทีมแปลง pain point เป็นเหตุการณ์ที่วัดได้

**ให้ตอบ 5 ข้อ**

- เหตุการณ์เริ่มต้นคืออะไร
- ใช้ข้อมูลใดวัดเหตุการณ์นี้
- threshold / pattern / time window คืออะไร
- ใครเป็น owner ของ trigger
- ถ้าเกิด trigger แล้ว action แรกคืออะไร

**ถ้ายังเขียน trigger ไม่ได้ use case ยังไม่พร้อมทำ prototype**

1. Automation Blueprint
2. Exception Pack
3. Prototype Plan
4. Business Case
5. Executive Pitch Draft

Part 2

# Data Contract

ข้อมูลต้องมีคุณภาพพอ ก่อนให้ AI ช่วยตัดสินใจ

# Data Contract คืออะไร

ข้อตกลงขั้นต่ำของข้อมูลก่อนเข้าสู่ workflow

**องค์ประกอบสำคัญ**

- Schema: field, format, unit และ expected range
- Owner: ใครรับผิดชอบความถูกต้องของข้อมูล
- Freshness: ข้อมูลล่าสุดพอสำหรับ decision window หรือไม่
- Lineage: รู้ต้นทางและการแปลงข้อมูลหรือไม่
- Missing-data rule: เมื่อข้อมูลขาดต้องทำอย่างไร

**เขียน data contract ก่อนเขียน automation**

**Schema**
ชื่อ field / format / unit

**Owner**
ใครรับผิดเมื่อข้อมูลผิด

**Freshness**
ล่าสุดพอสำหรับ decision window หรือไม่

**Lineage**
รู้ต้นทางและ transformation หรือไม่

**Missing-data rule**
เมื่อข้อมูลขาดต้องทำอย่างไร

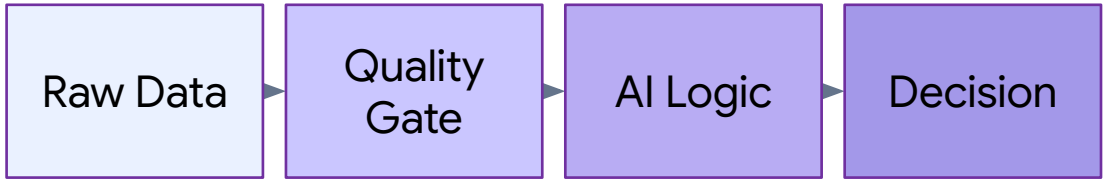
# Data Quality Gate

ห้ามให้ข้อมูลผิดผ่านเข้าสู่ decision logic

**Quality gate ควรถามว่า**

- ข้อมูลครบช่วงเวลาที่ต้องใช้หรือไม่
- หน่วยถูกต้องหรือไม่ เช่น °C, นาที, kg, CFU
- ข้อมูลมาจาก source ที่ได้รับอนุมัติหรือไม่
- ถ้า data fail ต้องหยุด workflow หรือใช้ fallback rule

**ข้อมูลผิด + automation เร็ว = ความเสี่ยงเร็วขึ้น**




**Quality Gate**

ไม่ผ่าน = หยุด workflow หรือใช้ fallback rule

# Freshness: ข้อมูลต้องใหม่แค่ไหน

ความสดของข้อมูลขึ้นกับประเภท decision

**ตัวอย่าง decision window**

- CCP หรือ line stop: ต้อง near-real-time
- defect trend: อาจใช้ทุก 15–30 นาที
- lab result review: ใช้ตามรอบการ upload
- complaint / supplier review: รายวันหรือรายสัปดาห์พอได้

**อย่าใช้ข้อมูลเมื่อวานตัดสินใจเรื่องที่ต้องรู้ตอนนี้**

| Real-time | Hourly |
|---|---|
| เช่น CCP, line stop, temperature excursion | เช่น defect trend, lab proxy, maintenance signal |

| Daily / Weekly |
|---|
| เช่น cost, complaint, supplier review |

**Decision window ต้องเป็นตัวกำหนด freshness**

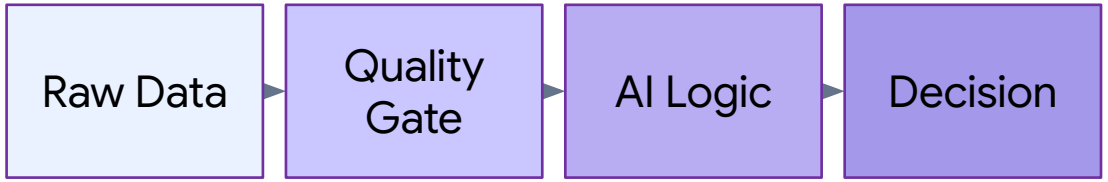
# Data Quality Gate

ห้ามให้ข้อมูลผิดผ่านเข้าสู่ decision logic

**Quality gate ควรถามว่า**

- ข้อมูลครบช่วงเวลาที่ต้องใช้หรือไม่
- หน่วยถูกต้องหรือไม่ เช่น °C, นาที, kg, CFU
- ข้อมูลมาจาก source ที่ได้รับอนุมัติหรือไม่
- ถ้า data fail ต้องหยุด workflow หรือใช้ fallback rule

**ข้อมูลผิด + automation เร็ว = ความเสี่ยงเร็วขึ้น**




**Quality Gate**

ไม่ผ่าน = หยุด workflow หรือใช้ fallback rule

# Missing Data Policy

ข้อมูลขาดต้องมี policy ไม่ใช่แก้เฉพาะหน้า

**ตัวอย่าง policy**

- sensor gap < 5 นาที: ใช้ interpolation ได้เฉพาะ non-safety trend
- sensor gap > 10 นาที: ส่ง QA review
- lab result late: ไม่ให้ release อัตโนมัติ
- missing lot mapping: hold evidence package จนกว่าจะ reconcile

**บางช่องว่างข้อมูลเล็ก แต่ผลต่อ safety ใหญ่มาก**

**Sensor gap 17 นาที**

ห้ามเฉลี่ยอัตโนมัติถ้าเป็น food safety decision

Detect → Hold / Fallback → Owner Review → Evidence

**Policy**

ต้องเขียนก่อนทำ automation

# ตัวอย่าง Data Contract: Cold Chain

ใช้กับ workflow ตอบสนอง temperature excursion

**Field ที่ควรกำหนด**

- shipment ID, route ID, vehicle ID, pallet/lot mapping
- temperature logger frequency และหน่วย °C
- door-open event, dwell time, delivery timestamp
- shelf-life requirement และ product risk profile
- missing rule เมื่อ temp log ขาดช่วง

**Data contract ทำให้ cold-chain decision ป้องกัน audit ได้**

Route ID + Shipment ID

Temperature log: unit °C / frequency 5 นาที

Dwell time + door-open event

Lot / pallet / SKU mapping

Missing rule: gap > 10 นาที → QA review

Part 3

# AI Logic และ Risk Logic

เลือกตรรกะให้เหมาะกับความเสี่ยง ไม่ใช้โมเดลเพราะดูทันสมัย

# เลือก Logic ให้เหมาะกับปัญหา

AI logic มีหลายแบบและเหมาะกับความเสี่ยงต่างกัน

**ตัวเลือกหลัก**

- Rule gate: ใช้กับ limit ที่ไม่ต่อรองได้
- SPC: ใช้กับ drift และ process stability
- Anomaly detection: ใช้กับ pattern ผิดปกติที่ไม่รู้ล่วงหน้า
- Risk score: ใช้รวมหลายหลักฐานและ uncertainty

**Logic ที่ดีต้องอธิบายได้และควบคุมได้**

**Rule gate**
critical limit / regulation

**SPC**
drift / process stability

**Anomaly detection**
pattern ที่มนุษย์เห็นช้า

**Risk score**
รวมหลายหลักฐานและ uncertainty

# Rule-Based Logic

เหมาะกับเกณฑ์ที่ชัดและต้อง audit ง่าย

**ควรใช้เมื่อ**

- มี Regulatory limit หรือ Critical limit
- Decision boundary ต้องชัดและไม่ต่อรอง
- ทีมต้องอธิบายเหตุผลแบบตรงไปตรงมา
- ข้อมูลยังไม่พอสำหรับ model ที่ซับซ้อน

**IF**
อุณหภูมิ > limit นานกว่า 12 นาที

**THEN**
ส่งเข้า QA Review / Hold

**เหมาะเมื่อ**
เกณฑ์ไม่ต่อรองได้และต้อง audit ได้ง่าย

**Simple rule ที่ดี อาจปลอดภัยกว่า AI model ที่อธิบายยาก**

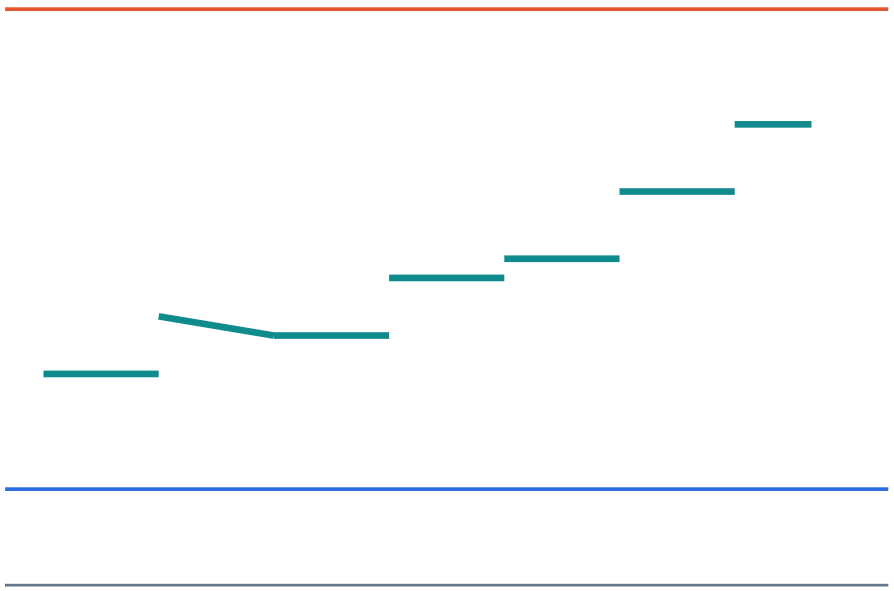
# Statistical Process Control

เหมาะกับการจับ drift และความนิ่งของ process

**ใช้เพื่อ**

- ตรวจว่ากระบวนการเริ่มแกว่งผิดปกติหรือไม่
- แยก special cause ออกจาก common cause
- จับ process ที่ยังไม่หลุด spec แต่เริ่มไม่นิ่ง
- เชื่อมกับ corrective action ก่อนเกิด defect จำนวนมาก

**SPC ช่วยให้เห็นปัญหาก่อน KPI เสีย**



**SPC**
จับ drift ก่อนหลุด control limit

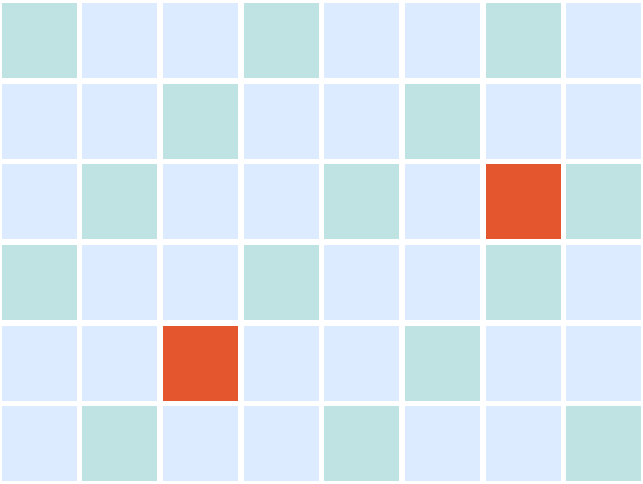
# Anomaly Detection

เหมาะกับ pattern ใหม่หรือซ่อนอยู่ในข้อมูลจำนวนมาก

**เหมาะเมื่อ**

- ไม่รู้ threshold ที่ดีล่วงหน้า
- ข้อมูลเป็นภาพ sensor time-series หรือหลาย feature
- ต้องการหาเหตุการณ์ผิดปกติเร็วขึ้น
- ยังต้องมี human review เพราะ anomaly ไม่เท่ากับ defect เสมอ

**Anomaly เป็นสัญญาณให้ตรวจ ไม่ใช่คำตัดสินสุดท้าย**



**Anomaly Detection**
เหมาะกับ pattern ที่ไม่รู้ threshold ล่วงหน้า

# Risk Scoring

เหมาะกับ decision ที่ต้องรวมหลายหลักฐาน

**Risk score ที่ดีต้องมี**

- Feature ที่อธิบายได้ เช่น exposure, shelf-life, route delay
- Calibration กับ outcome จริง
- Threshold และ uncertainty handling
- Reject option เมื่อข้อมูลไม่พอหรือความมั่นใจต่ำ

**สีแดง-เหลือง-เขียวต้องมีหลักฐานรองรับ**

**Evidence 1**
Temp exposure

**Evidence 2**
Shelf-life

**Evidence 3**
Route delay

**Risk Score = 82/100**

**ต้องมี Calibration**
คะแนนต้องสัมพันธ์กับ outcome จริง

# เมื่อไรไม่ควรใช้ AI Model

บาง use case ควรเริ่มจาก rule, SOP หรือ data readiness ก่อน

**สัญญาณเตือน**

- ข้อมูลน้อยหรือ label ไม่น่าเชื่อถือ
- เกณฑ์ business rule ชัดอยู่แล้ว
- Decision เสี่ยงสูงแต่ model อธิบายไม่ได้
- ปัญหาแท้จริงคือ SOP ไม่ถูกทำตาม ไม่ใช่ไม่มี model

**อย่าใช้ AI เพื่อแก้ปัญหาที่ governance ยังไม่พร้อม**

กฎชัดเจนอยู่แล้ว → ใช้ rule gate

ข้อมูลน้อยเกินไป → เริ่มเก็บข้อมูลก่อน

Decision เสี่ยงสูงและอธิบายไม่ได้ → ใช้ Human review

ปัญหาเป็น process discipline → แก้ SOP ก่อน model

# False Positive vs False Negative

ต้องคุย trade-off ก่อนออกแบบ threshold

**ความหมาย**

- False positive: ระบบเตือนทั้งที่ไม่เสี่ยงจริง
- False negative: ระบบไม่เตือนทั้งที่มีความเสี่ยงจริง
- ใน food safety หลายกรณี false negative แพงกว่า
- ต้องกำหนด risk appetite ตามประเภท decision

**ความแม่นรวมไม่พอ ต้องดูความผิดพลาดที่อันตรายที่สุด**

**False Positive**

แจ้งเตือนผิด → เสียเวลา / hold เกินจำเป็น

**False Negative**

ไม่เตือน case เสี่ยง → สินค้าเสี่ยงหลุด

ใน food safety หลายกรณี false negative มีต้นทุนสูงกว่า

Part 4

# Decision Boundary

AI แนะนำได้ แต่ decision สำคัญต้องมีเจ้าของ

# Decision Boundary คืออะไร

เส้นแบ่งว่า AI ทำอะไรได้ ต้องให้ใครอนุมัติ และอะไรห้ามทำ

**ต้องนิยามให้ชัด**

- **AI ทำได้:** เตรียมหลักฐาน ให้ score จัดลำดับงาน
- **มนุษย์ต้องอนุมัติ:** hold, release, quarantine, deviation
- **AI ห้ามทำ:** override food safety rule หรือลบ evidence log
- ทุก boundary ต้องสัมพันธ์กับ role และ severity

**Boundary ที่ไม่ชัด = Accountability ไม่ชัด**

**AI ทำได้**
รวบรวมหลักฐาน / ให้ risk score / draft narrative

**มนุษย์ต้องอนุมัติ**
hold / release / quarantine / deviation

**AI ห้ามทำอัตโนมัติ**
release lot ที่เกี่ยวกับ CCP / override rule / ลบ log

# AI ทำอะไรได้บ้าง

ให้ AI เป็นผู้ช่วยตัดสินใจ ไม่ใช่ผู้มีอำนาจสุดท้าย

**AI เหมาะกับงานเหล่านี้**

- รวบรวมหลักฐานจากหลายระบบ
- ให้ risk score พร้อม confidence
- จัดลำดับความเร่งด่วนของ ticket
- ร่าง Non-Conformance narrative
- เสนอ corrective action alternatives

**AI ทำงานเตรียมหลักฐานได้เร็วมาก**

รวบรวมหลักฐานจากหลายระบบ

ให้ Risk score + Confidence

จัดลำดับความเร่งด่วน

ร่าง NC narrative

เสนอทางเลือก Corrective action

# งานที่ต้องให้มนุษย์อนุมัติ

Decision ที่กระทบสินค้า กระบวนการ หรือ compliance ต้องมีผู้รับผิดชอบ

**ตัวอย่าง**

- ปล่อยหรือกักสินค้า
- เปลี่ยน process window
- ยืนยัน root cause
- อนุมัติ deviation
- สื่อสารกับลูกค้าหรือ regulator

**Approval ต้องมี Rationale ไม่ใช่แค่คลิกผ่าน**

ปล่อยหรือกักสินค้า

เปลี่ยน Process window

ยืนยัน Root cause

อนุมัติ Deviation

สื่อสารกับลูกค้าหรือ Regulator

# งานที่ห้ามให้ AI ตัดสินใจอัตโนมัติ

ต้องล็อกไว้ใน policy และ system permission

**ห้ามทำโดยอัตโนมัติ**

- Release lot ที่เกี่ยวกับ Critical Control Point
- ยกเลิก quarantine โดยไม่มี QA approval
- สรุป recall scope เอง
- Override food safety rule
- ลบหรือแก้ evidence log หลังเกิดเหตุ

**AI ไม่ควรเป็นเจ้าของ food safety disposition**

Release lot ที่เกี่ยวกับ Critical Control Point

ยกเลิก Quarantine เอง

สรุป Recall scope เอง

Override food safety rule

ลบหรือแก้ evidence log

# Override ต้องบันทึกอะไรบ้าง

ทุก override เป็นหลักฐานสำคัญเมื่อเกิด deviation หรือ audit

**ต้องเก็บอย่างน้อย**

- ชื่อและ role ของผู้อนุมัติ
- timestamp ของการตัดสินใจ
- เหตุผลและ evidence ที่ใช้ประกอบ
- Action ที่ทำจริง
- Outcome หลัง action และ post-review

**Override ที่ไม่มีเหตุผลคือช่องโหว่ของ governance**

Approver
ชื่อและ role

Timestamp
เวลาที่ตัดสินใจ

Rationale
เหตุผลและ evidence ที่ใช้

Outcome
ผลหลัง action

Post-review
เรียนรู้อะไรเพื่อปรับ rule

# ตัวอย่าง Decision Boundary: Lot Release

ใช้ AI ช่วยรวบรวมหลักฐาน แต่ final disposition ต้องมีผู้อนุมัติ

**แนวทาง**

- AI สรุป risk score และ evidence package
- QA ตรวจ data completeness และ rule violation
- Supervisor ให้ข้อมูล operation context
- ผู้มีอำนาจอนุมัติ hold/release/rework
- ระบบเก็บ rationale และ outcome แบบ append-only

**AI ควรช่วยให้การตัดสินใจดีขึ้น ไม่ใช่ลัดขั้นตอน**

Risk Score → QA Review → Hold/ Release → Evidence Log

**Boundary**
AI ให้ recommendation ได้
แต่ disposition ต้องมีผู้มีอำนาจอนุมัติ

Part 5

# Evidence Log และ Auditability

ถ้าอธิบายย้อนหลังไม่ได้ ระบบยังไม่พร้อมใช้จริง

# Evidence Log คืออะไร

บันทึกหลักฐานของทุก workflow run ตั้งแต่ input ถึง outcome

**ต้องตอบได้ว่า**

- เหตุการณ์เริ่มเมื่อไรและด้วย trigger อะไร
- ใช้ข้อมูลชุดใด เวอร์ชันใด timestamp ใด
- ใช้ rule, model หรือ prompt เวอร์ชันใด
- AI แนะนำอะไร confidence เท่าไร
- ใครอนุมัติ ทำอะไร และผลหลัง action คืออะไร

**ไม่มี evidence log = workflow ใช้ตรวจสอบจริงไม่ได้**

Workflow Run ID

Input data version + timestamp

Rule / model / prompt version

Output + confidence + uncertainty

Approver + rationale + outcome

# Auditor อยากเห็นอะไร

คิดแบบผู้ตรวจตั้งแต่ตอนออกแบบ prototype

**คำถามจาก auditor**

- ใครตัดสินใจและมีอำนาจตาม matrix หรือไม่
- ตัดสินจากหลักฐานชุดใด
- ข้อมูลและ model version ตามย้อนกลับได้หรือไม่
- มี override หรือ exception หรือไม่
- ผลหลัง action ตรงกับสิ่งที่บันทึกหรือไม่

**ออกแบบให้ตอบคำถาม auditor ได้ใน 5 นาที**

ใครตัดสินใจ?

ตัดสินจากหลักฐานชุดใด?

ใช้ rule / model version ใด?

มี override หรือไม่?

ผลหลัง action เป็นอย่างไร?

# Evidence Log ต้องเก็บอะไรบ้าง

## Specification ขั้นต่ำสำหรับระบบ audit-ready

**รายการหลัก**

- Workflow run identifier และ timestamp
- Input data version และ source lineage
- Rule/model/prompt version
- Output, risk score, confidence และ uncertainty
- Approver, rationale, corrective action และ outcome

**เก็บเฉพาะ output ไม่พอ ต้องเก็บ path ของ decision**

Input data version

Rule/model/prompt version

Output + risk score + confidence

Human approval + rationale

Corrective action + outcome

# Versioning: Prompt / Model / Rule

เมื่อผลลัพธ์เปลี่ยน ต้องรู้ว่าอะไรเปลี่ยน

**แนวทาง**

- Prompt template version สำหรับ generative AI
- Rule version สำหรับ threshold และ gate
- Model version สำหรับ prediction หรือ risk scoring
- Workflow configuration version เช่น threshold และ escalation
- Validation dataset version

**เวอร์ชันคือภาษาของ Auditability**

Prompt v1.2 → Rule v3.0 → Model v0.9 → Config v4

**Version Control**
หากผลลัพธ์เปลี่ยน ต้องรู้ว่าเปลี่ยน
เพราะข้อมูล กฎ โมเดล หรือ prompt

# Append-only Log

หลักฐานต้องป้องกันการแก้ย้อนหลังหลังเกิด deviation

**หลักการ**

- เพิ่ม record ใหม่ได้ แต่ไม่ควรเขียนทับ record เดิม
- จำกัดสิทธิ์แก้ไข และเก็บ change history
- แยก correction note ออกจาก original event
- เก็บ retention policy ตามข้อกำหนดองค์กรและกฎหมาย

**Log ที่แก้ย้อนหลังง่ายจะไม่น่าเชื่อถือ**

Append-only log

สิทธิ์แก้ไขจำกัด

ห้ามลบ log หลังเกิด deviation

Log ที่ดีต้องป้องกันการแก้ย้อนหลัง

# แบบฝึก: ออกแบบ Evidence Log

ให้ทีมกำหนด evidence ที่ต้องเก็บใน use case ของตน

**ให้ตอบ**

- Workflow run ID จะสร้างจากอะไร
- Input data source และ version คืออะไร
- AI output จะเก็บ confidence หรือ rationale อย่างไร
- ใครเป็น approver และต้องกรอกเหตุผลใด
- Outcome หลัง action จะติดตามเมื่อไร

**อย่ารอทำ log ตอน go-live ให้คิดตั้งแต่ prototype**

Workflow Run ID

Input data version + timestamp

Rule / model / prompt version

Output + confidence + uncertainty

Approver + rationale + outcome

Part 6

# Prototype Plan

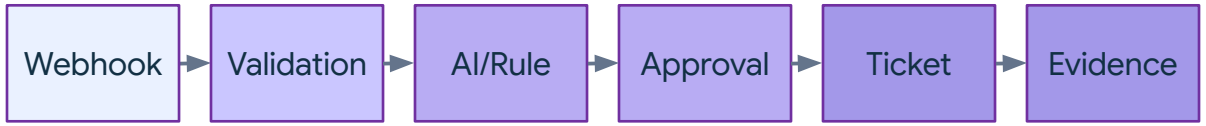
ต้นแบบต้องเล็ก แต่ต้องมี control พอสำหรับโรงงานจริง

# Prototype ที่เล็กแต่ใช้จริงได้

Minimum viable prototype ต้องไม่ใช่แค่ demo หน้าจอสวย

**Prototype ต้องมี**

- Input sample 30–50 เหตุการณ์ย้อนหลังพร้อม outcome จริง
- Data validation step ก่อน AI หรือ rule logic
- Approval queue ที่บังคับกรอก rationale
- Evidence log ที่ export ได้
- Validation sheet เทียบกับ decision จริงของทีม




**MVP Prototype**
เล็กพอทดลองได้ แต่ต้องมี control ครบ

**เล็กได้ แต่ห้ามข้าม control**

# Input Sample สำหรับ Prototype

ใช้ข้อมูลย้อนหลังเพื่อทดสอบว่า workflow คิดถูกหรือไม่

**ข้อมูลขั้นต่ำ**

- เหตุการณ์จริง 30–50 cases จาก process เดียวกัน
- มี input ครบพอสำหรับ trigger และ risk logic
- มี human decision เดิมเพื่อใช้เทียบ
- มี actual outcome เช่น release, reject, rework, complaint
- ระบุ case ที่ข้อมูลขาดหรือ exception ด้วย

**Prototype ที่ไม่มี outcome จริงจะวัด value ยาก**

30–50 เหตุการณ์ย้อนหลัง
ต้องมี input + outcome จริง เพื่อเทียบระบบกับการตัดสินใจเดิม

Incident data

Human decision

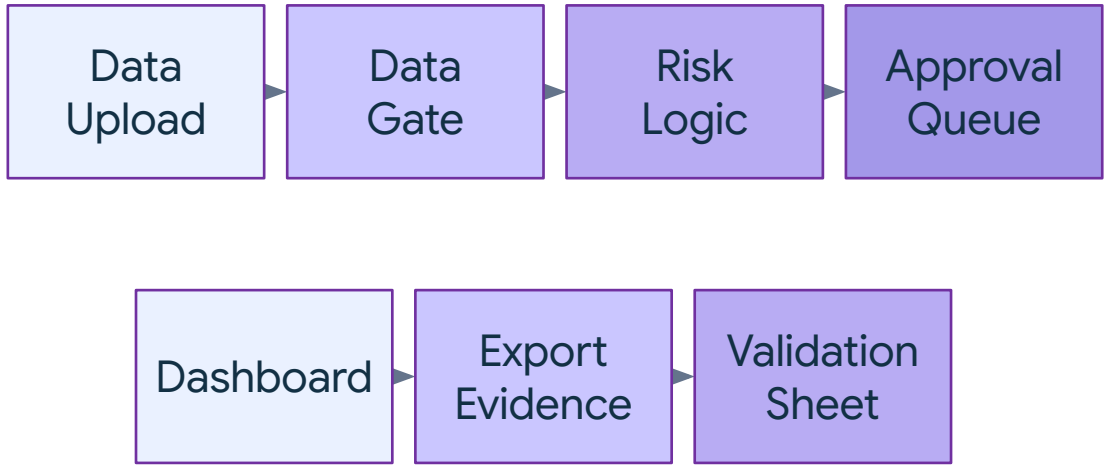
Actual outcome

# Prototype Architecture แบบง่าย

เริ่มจาก sandbox ไม่เขียนทับระบบ production

**โครงสร้างแนะนำ**

- Data upload หรือ schedule ดึงข้อมูลแบบ read-only
- Data gate ตรวจ completeness, freshness และ unit
- AI/rule step สร้าง risk score หรือ recommendation
- Approval queue ให้ supervisor/QA review
- Dashboard + evidence export + validation sheet




**เริ่มแบบ read-only ก่อนเสมอ**

# Exception Handling

ออกแบบกรณีผิดปกติก่อนทำ demo ที่ดูดี

**ต้องเตรียมอย่างน้อย**

- Missing record เช่น checklist late หรือ sensor gap
- Conflicting evidence เช่น operator note ปกติ
- แต่ temp profile ผิดปกติ
- Model uncertainty เช่น score ใกล้ threshold
- Escalation failure เช่น approver ไม่อยู่กะกลางคืน
- System failure เช่น dashboard หรือ ticket system ล่ม

**Happy path ไม่พอสำหรับโรงงานจริง**

Missing record
late checklist / sensor gap

Conflicting evidence
operator note normal but temp abnormal

Model uncertainty
risk score ใกล้ threshold

Escalation failure
ผู้อนุมัติไม่อยู่กะกลางคืน

# Validation Plan

อย่าวัดแค่ความแม่นของ model ให้วัดผลต่อการตัดสินใจจริง

**มิติการวัด**

- Analytical validity: precision, recall, calibration, uncertainty
- Operational validity: time-to-decision, action completion, latency
- Safety validity: false negative on critical cases, boundary breach
- Audit validity: evidence completeness, version traceability
- ตั้ง go/no-go criteria ก่อนทดสอบ

**ถ้าไม่ผ่าน go/no-go ต้อง revise ไม่ใช่ฝืน scale**

**Analytical**
precision / recall / calibration

**Operational**
time-to-decision / action completion

**Safety**
false negative / boundary breach

**Audit**
evidence completeness / traceability

**ตั้ง go/no-go ก่อนทดสอบ**

Part 7

# Business Case และ Pitch

ผู้บริหารต้องเห็นปัญหา กลไกผลลัพธ์ ความเสี่ยง และสิ่งที่ต้องอนุมัติ

# Executive Pitch: Problem → Insight → Recommendation → Impact → Ask

ทำไมเราถึงต้องการระบบที่ฉลาดขึ้น? นี่คือปัญหาที่เราพบเจอกันเป็นประจำ:

- **Data Pipelines เปราะบาง:**
  ระบบพังบ่อยเมื่อแหล่งข้อมูลต้นทางเปลี่ยนโครงสร้างต้องคอยซ่อมแซมแบบ Manual
- **ข้อมูลขยะ (Dirty Data) มหาศาล:** หมดเวลาไปกับการเขียน Rulesets หรือ Regex ยืดยาวเพื่อทำความสะอาดข้อมูลที่ไม่ได้มาตรฐาน
- **คอขวด (Bottleneck):**
  Business ต้องการ Report หรือข้อมูลด่วน แต่ทีม Data ทำงานให้ไม่ทัน

*"เรากำลังใช้คนระดับผู้เชี่ยวชาญทำงานที่ซ้ำซากเหมือนเครื่องจักร"*

# Solution (Example)

**Google Docs Link:**
https://docs.google.com/document/d/1d64LyiTDgKgSwWRmiB9WtbxiTOJdxybF/edit?usp=sharing&ouid=114216695598597482325&rtpof=true&sd=true

# Thank You