zenity x

AI AGENT CO-WORKING SYSTEM · DAY 1

First Vibe Coding

สร้างระบบ CRM สโมสรแบบพื้นฐาน

จากไอเดียสู่ระบบ CRM ที่ล็อกอินได้จริง — เข้าใจ Frontend · Backend · Database และทำงานกับทีม AI Agent อย่างถูกวิธี

1 วัน · 10:00–17:00 น.

Claude Code / Codex · Supabase · Cloudflare



หลังเรียนจบวันนี้

ผลลัพธ์หลังเรียน 5 ข้อ

01

อธิบายได้ว่า 1 ระบบประกอบด้วย Frontend · Backend · Database และข้อมูลไหลอย่างไร

02

วาง Use Case ได้ว่าระบบมีกี่บทบาท แต่ละบทบาททำอะไรได้ + เขียนตารางสิทธิ์ (Permission Matrix)

03

เข้าใจการทำงานจริงของ AI Agent / Subagent เลือกใช้ให้ถูกงาน และเขียน SPEC.md + ไฟล์กติกาทีมให้ Agent อ่าน

04

ได้ระบบ CRM ต้นแบบ: ฟอร์มสนใจสมัคร → ลีดเข้าฐานข้อมูล → หลังบ้านจัดการลีด + ล็อกอิน 3 บทบาท

05

ขึ้นเว็บจริงบน Cloudflare Pages เปิดบนมือถือได้

ผู้เรียน: ทีมสโมสรสุขภาพ ~3 คน (เจ้าของ/ผู้จัดการ · แอดมินและดูแลสมาชิก · ฝ่ายขายและการตลาด) ไม่ต้องมีพื้นฐานเขียนโปรแกรม · ข้อมูลทั้งหมดในคลาสเป็นข้อมูลจำลอง

RUN OF SHOW

ตารางสอน 1 วัน

เวลาเรียน 10:00–17:00 | พักกลางวัน 12:30–13:30 | เบรก 15:00–15:15

ช่วงเช้า · 10:00–12:30

10:00–10:30	**Session 1 · Vibe Coding คืออะไร + เปิดโจทย์สโมสร** แผนที่โจทย์: วันนี้ทำอะไร อะไรต่อยอด
10:30–11:00	**Session 2 · ระบบ 1 ระบบประกอบด้วยอะไร** Frontend / Backend / Database + Authentication / Authorization → อธิบายระบบสโมสรเป็น 3 ชั้นได้
11:00–11:30	**Session 3 · AI Agent ทำงานจริงอย่างไร** Main Agent · Subagent · ไฟล์กติกา → เข้าใจว่าทำไมต้องเขียน SPEC และไฟล์กติกาให้ดี
11:30–12:30	**Workshop 1 · ออกแบบก่อนสร้าง** Use Case Diagram · Permission Matrix · SPEC.md · CLAUDE.md / AGENTS.md

ช่วงบ่าย · 13:30–17:00

13:30–15:00	**Workshop 2 · สร้าง CRM ด้วย Main Agent ทีละชั้น** **ด่าน 1** กรอกฟอร์มแล้วลีดโผล่ใน CRM · เปลี่ยนสถานะแล้วค่าในตารางเปลี่ยน
15:00–15:15	**เบรก**
15:15–16:30	**Workshop 3 · ระบบล็อกอินและบทบาท + ล็อกสิทธิ์ที่ฐานข้อมูล** **ด่าน 2** สลับล็อกอิน 3 บทบาท เห็น/ทำได้ตรง Permission Matrix
16:30–17:00	**Workshop 4 · ขึ้นเว็บจริง + สรุป + แผนต่อยอด** **ด่าน 3** เปิด URL บนมือถือแล้วล็อกอินได้

ตารางเวลาเป็นแนวทาง ผู้สอนอาจปรับเวลาแต่ละช่วงให้เหมาะกับความคืบหน้าของผู้เรียน โดยเนื้อหาและผลลัพธ์หลักยังครบถ้วน

ก่อนวันเรียน · เตรียมเครื่อง

เตรียมให้พร้อม ก่อนเข้าคลาส

คอมพิวเตอร์ + Chrome

บัญชี Claude (Pro ขึ้นไป) หรือ ChatGPT (Plus ขึ้นไป) · ติดตั้ง Claude Code หรือ Codex

สมัคร Supabase + Cloudflare

ฟรีทั้งคู่ · ใช้อีเมลที่ใช้ประจำ

ไม่ต้องติดตั้งอะไรเพิ่มเอง

Setup 1 ให้ Agent เช็คและติดตั้ง Git / Node.js ให้

เรื่อง	Claude Code	Codex
ไฟล์บันทึกโปรเจกต์ (คู่มือกติกาทีม)	`CLAUDE.md`	`AGENTS.md`
สร้างระบบในคลาส	Main Agent ตัวเดียว ทำทีละชั้น	เหมือนกัน — ใช้ Prompt เดียวกัน
การถามคำถาม	ตัวเลือกให้กดเลือก	ฟอร์มข้อ ๆ ในแชท
แนบไฟล์/รูป	ลากไฟล์เข้าแชท หรือพิมพ์ path	ลากไฟล์เข้าแชท หรือพิมพ์ path

ใช้ Claude Code หรือ Codex ก็ได้ · Prompt ทุกตัวใช้คำกลาง เช่น "ไฟล์บันทึกโปรเจกต์ (CLAUDE.md / AGENTS.md)"

S1

SESSION 1 · 10:00-10:30

Vibe Coding คืออะไร + เปิดโจทย์สโมสร

แผนที่โจทย์: วันนี้ทำอะไร อะไรต่อยอด · ทำ Setup 1–3 ในช่วงนี้



บอกสิ่งที่ต้องการเป็นภาษาคน ให้ AI Agent เขียนโค้ด — เราทดสอบ ตรวจ และตัดสินใจ

10 ความต้องการ เริ่มที่แกนกลางเดียว



SESSION 1 · ตารางความครอบคลุม

วันนี้ทำแกนกลาง — ที่เหลือต่อยอดด้วยวิธีเดียวกัน

ระดับ	ความต้องการของสโมสร
ทำวันนี้	**Sales & CRM**
ทำบางส่วน	**Central Management** (ฐานข้อมูลกลาง สิทธิ์ตามบทบาท) · **Website & Join Now** (เฉพาะฟอร์มลงทะเบียน)
ต่อยอด	Membership & Waiting List · Booking · Document & Contract · Chatbot / Customer Service
นอกขอบเขต	Marketing / Content Creator (งานผลิตสื่อ)

งานต่อยอดทุกข้อมี Prompt ตั้งต้นให้ในช่วงท้ายวัน

SESSION 1 · กติกาของห้อง

กติกา 3 ข้อ ตลอดทั้งวัน

01

ใช้ข้อมูลจำลองในห้อง

สโมสรตัวอย่างชื่อสมมติ Harmony Club · ข้อมูลทั้งหมดในคลาสเป็นข้อมูลจำลอง

02

คีย์ลับไม่อยู่บนหน้าเว็บ

คีย์ลับอยู่ Backend เท่านั้น — Frontend ใครก็เปิดดูโค้ดได้

03

คนอนุมัติเรื่องสำคัญ

AI ร่างได้ คนตัดสินใจ

กติกาเหล่านี้จะถูกเขียนลงไฟล์กติกาทีม CLAUDE.md / AGENTS.md ใน Workshop 1.6 ให้ Agent ทุกตัวอ่านก่อนทำงาน

SETUP 1 · เช็คเครื่องมือในเครื่อง

ให้ Agent เช็คและติดตั้งเครื่องมือที่ขาด

- **Git**
  ตรวจว่ามีแล้วหรือยัง
- **Node.js**
  เวอร์ชัน 20 ขึ้นไป
- **npx**
  ใช้รันเว็บบนเครื่องและ Deploy

ต้องใช้รหัสผ่านเครื่อง → Agent ต้องถามก่อนทุกครั้ง

SETUP 1

```
ช่วยเช็คเครื่องของฉันว่าพร้อมทำเว็บหรือยัง:
- ตรวจว่ามี Git, Node.js (เวอร์ชัน 20 ขึ้นไป) และ npx หรือยัง
- ตัวไหนยังไม่มี ให้บอกวิธีติดตั้งที่ง่ายที่สุดสำหรับเครื่องนี้ แล้วช่วยติดตั้งให้ (ถามก่อนทุกครั้งที่ต้องใช้
รหัสผ่านเครื่อง)
สรุปผลเป็นตาราง: เครื่องมือ · เวอร์ชัน · พร้อมใช้ / ต้องติดตั้ง
```

SETUP 2 · สร้างโปรเจกต์ SUPABASE (ทำเองบนเว็บ)

ฐานข้อมูล + ระบบล็อกอิน: Supabase

1. **New project**
   ตั้งชื่อ `club-crm`
2. **ตั้งรหัสผ่านฐานข้อมูล**
   แล้วจดเก็บไว้
3. **Region Singapore**
   รอ 1–2 นาทีให้โปรเจกต์พร้อม
4. **Project Settings → API**
   จด **Project URL** และ **anon / publishable key**

anon / publishable key

ใส่หน้าเว็บได้

service_role / secret key

= กุญแจห้องเก็บของ **ห้ามใส่หน้าเว็บเด็ดขาด**

SETUP 3 · ตั้งโฟลเดอร์โปรเจกต์ + GIT

โฟลเดอร์ แบ่งตามชั้นของระบบ

public/
หน้าเว็บ · Frontend

functions/api/
Backend บน Cloudflare Pages Functions

supabase/
ไฟล์ SQL · Database

docs/
เอกสารออกแบบ

`.gitignore` กันไฟล์ลับ (.dev.vars, .env) และ node_modules ไม่ให้หลุดขึ้น Git

SETUP 3

```
ช่วยตั้งโปรเจกต์ใหม่ชื่อ club-crm ให้ฉัน (ระบบ CRM ของสโมสรสุขภาพ ใช้ข้อมูลจำลองเท่านั้น):
- โครงโฟลเดอร์: public/ (หน้าเว็บ) · functions/api/ (Backend บน Cloudflare Pages Functions) · supabase/ (ไฟล์ SQL) · docs/ (เอกสารออกแบบ)
- เริ่ม Git และสร้าง .gitignore กันไฟล์ลับ (.dev.vars, .env) และ node_modules ไม่ให้หลุดขึ้น Git
- ยังไม่ต้องเขียนหน้าเว็บ แค่โครงพร้อมใช้
อธิบายสั้น ๆ ว่าแต่ละโฟลเดอร์คือชั้นไหนของระบบ (Frontend / Backend / Database)
```

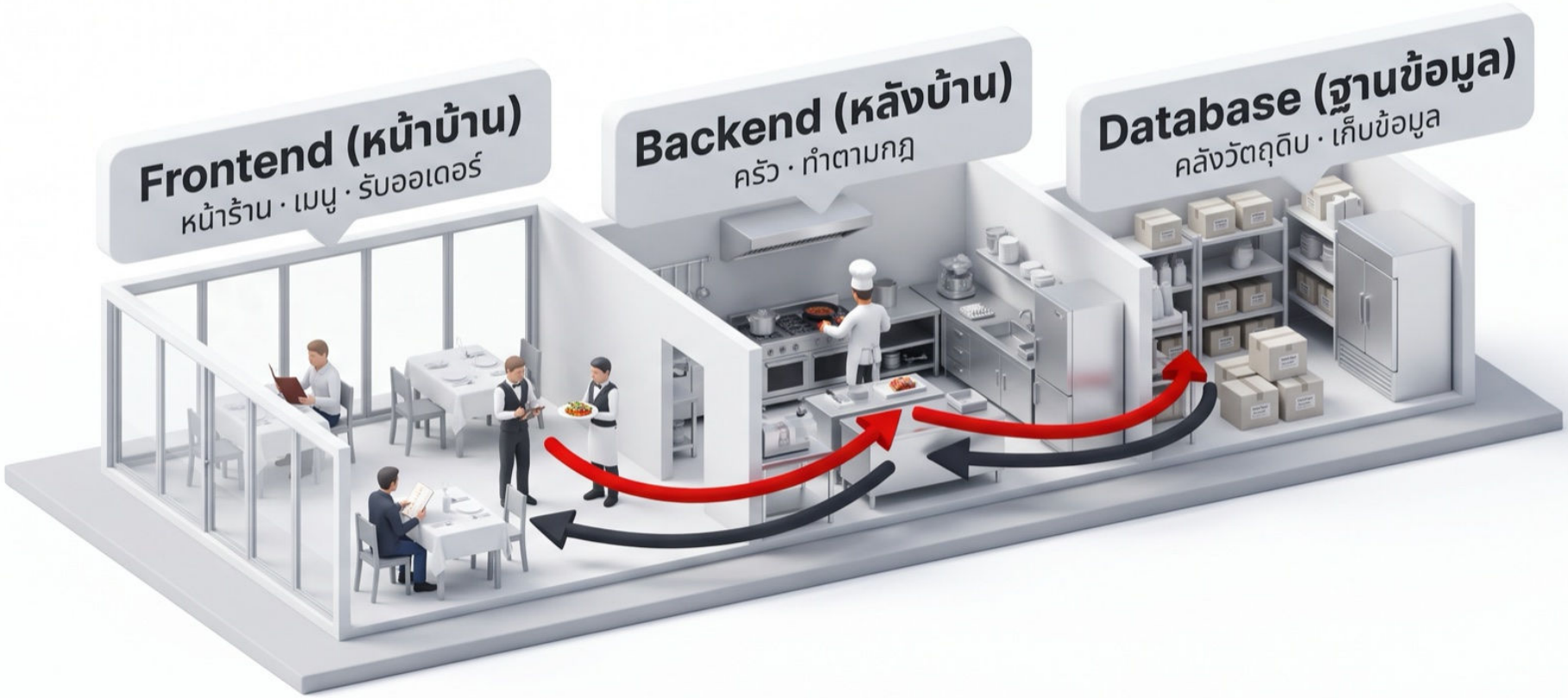
SESSION 2 · 10:30-11:00

S2

ระบบ 1 ระบบ ประกอบด้วยอะไร

Frontend / Backend / Database + Authentication / Authorization · ผลลัพธ์: อธิบายระบบสโมสรเป็น 3 ชั้นได้

ระบบ 1 ระบบ = ร้านอาหาร 3 ส่วน



ตามรอยคำขอ: ลูกค้ากดสมัคร



1 ลูกค้ากรอกฟอร์ม

Frontend

2 /api/join ตรวจข้อมูล

Backend

3 บันทึกลีดใหม่

Database

4 ตอบกลับ: ได้รับข้อมูลแล้ว

5 ทีมงานเห็นลีดใหม่ใน CRM

CRM

ล็อกอิน = บัตรพนักงาน





แอดมิน

พนักงานขาย

ผู้บริหาร

SESSION 2 · คีย์ลับอยู่ชั้นไหน

คีย์ลับอยู่ Backend เท่านั้น

Frontend (หน้าบ้าน)
ใครก็เปิดดูโค้ดได้ — ใส่ได้แค่ anon / publishable key

Backend (หลังบ้าน)
รับข้อมูลฟอร์ม ตรวจความถูกต้อง สร้างลีด เก็บคีย์ลับ

Database (ฐานข้อมูล)
เก็บข้อมูลถาวรเป็นตาราง: ลีด ประวัติการติดต่อ ทีมงาน

MINI-ACTIVITY · ให้ Agent อธิบาย 3 ชั้น

ฉันกำลังจะทำระบบ CRM ให้สโมสรสุขภาพ: ลูกค้ากรอกฟอร์มสนใจสมัคร → ทีมขายติดตาม → ผู้บริหารดูภาพรวม
อธิบายให้คนที่ไม่เคยเขียนโปรแกรมเข้าใจว่า ระบบนี้แบ่งเป็น Frontend / Backend / Database อย่างไร
- แต่ละชั้นทำหน้าที่อะไร เปรียบกับร้านอาหาร
- ตามรอย 1 คำขอ "ลูกค้ากดส่งฟอร์มสมัคร" ว่าผ่านชั้นไหนบ้างตามลำดับ
- คีย์ลับของฐานข้อมูลควรอยู่ชั้นไหน เพราะอะไร
ตอบสั้น กระชับ เป็นข้อ ๆ ยังไม่ต้องเขียนโค้ด

S3

SESSION 3 · 11:00-11:30

AI Agent ทำงานจริงอย่างไร

Main Agent · Subagent · ไฟล์กติกา — เข้าใจว่าทำไมต้องเขียน SPEC และไฟล์กติกาให้ดี

SESSION 3 · ความเชื่อผิด VS ความจริง

Agent หลายตัว คุยกันเองไหม?

ความเชื่อผิด

- ✕ **Agent หลายตัวคุยกันเอง**
- ✕ **นัดแบ่งงานกันเอง**
- ✕ **ทำงานอัตโนมัติไปเรื่อย ๆ**

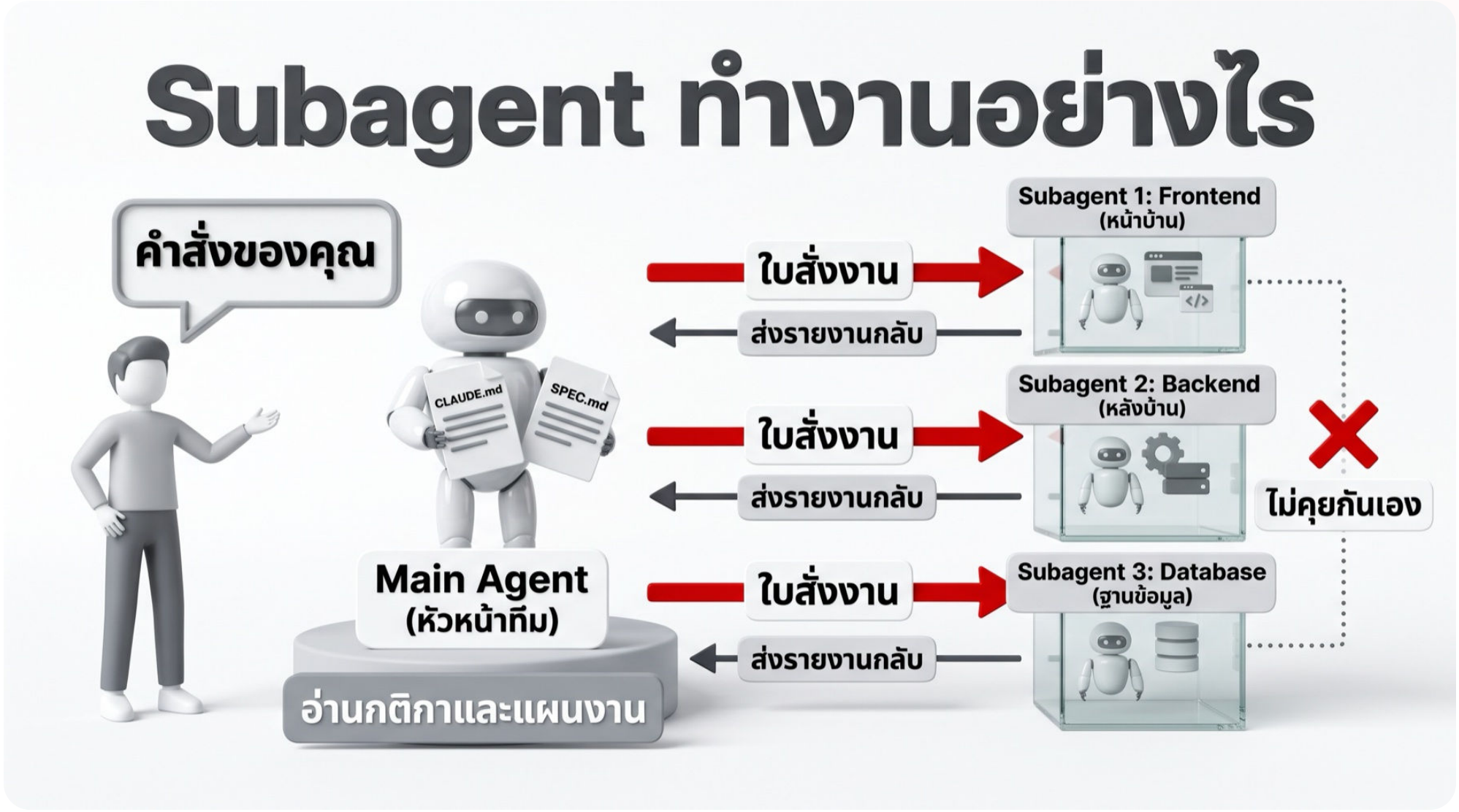
ความจริง · ทำงานผ่าน 3 อย่าง

- ✓ **ใบสั่งงาน (brief)**
  หัวหน้าเขียนให้ Subagent แต่ละตัว
- ✓ **กติกากลาง**
  ไฟล์กติกา CLAUDE.md / AGENTS.md
- ✓ **ข้อมูลที่ใช้ร่วมกัน**
  ไฟล์ในโปรเจกต์ เช่น SPEC.md และโค้ด

"Agent ไม่ได้คุยกันเอง แต่ทำงานผ่านใบสั่งงาน กติกากลาง และข้อมูลที่ใช้ร่วมกัน"

Main Agent = หัวหน้าทีม เขียนใบสั่งงานให้แต่ละคน





Subagent เริ่มจากความจำว่าง รู้แค่ใบสั่งงาน + ไฟล์กติกา + ไฟล์ในโปรเจกต์ → SPEC.md ที่ดีจึงสำคัญ

SESSION 3 · งานเดฟ ใช้แบบไหนดี

คลาสนี้สร้างด้วย Main Agent ตัวเดียว — ทำไม?

ทุกชั้นพึ่งกัน

Frontend เรียก API ของ Backend · Backend ใช้ตารางของ Database — แยกกันทำ ต่างคนต่างตัดสินใจ รวมแล้วไม่เข้ากัน

โทเค็นแพงขึ้น

หลาย Agent ใช้โทเค็นราว **15 เท่า**ของแชตปกติ (Agent ตัวเดียวราว 4 เท่า) เพราะทุกตัวอ่านบริบทซ้ำ

ที่แพงจริงคือรอบแก้

ทำพร้อมกันเสร็จเร็ว แต่ไม่ถูกลง ถ้าไม่เข้ากันต้องแก้หลายรอบ ทุกรอบอ่านโค้ดใหม่ทั้งก้อน

เริ่มจากแบบที่ง่ายที่สุด

เพิ่มความซับซ้อนเมื่อจำเป็นเท่านั้น — ทำทีละชั้น ทดสอบทีละชั้น

> **"Subagent ช่วยให้เร็ว ไม่ได้ช่วยให้ถูก — งานเดฟที่ทุกชั้นพึ่งกัน ให้ Main Agent ตัวเดียวทำทีละชั้น"**

อ้างอิง

- Anthropic · Multi-agent research system
- Anthropic · Building effective agents
- Cognition · Don't Build Multi-Agents

SESSION 3 · SUBAGENT ยังมีประโยชน์

ใช้ Subagent ช่วย อ่าน · ค้น · ตรวจ ไม่ใช่ช่วยเขียนพร้อมกัน

เหมาะกับ

- **ค้นหา / อ่านโค้ดแล้วสรุป**
  งานข้างเคียงที่ทำให้แชตหลักรก ส่งกลับมาแค่สรุป
- **รีวิว / ตรวจทาน**
  อ่านอย่างเดียว ไม่แก้ไฟล์
- **งานที่แยกขาดกันจริง**
  ไม่ต้องรู้ผลของกันและกัน

ไม่เหมาะกับ

- **เขียนหลายชั้นของระบบเดียวพร้อมกัน**
- **งานที่ต้องคุยโต้ตอบกับเราบ่อย**
- **งานเล็ก แก้จุดเดียว**

อ้างอิง

- Claude Code Docs · Subagents
- Anthropic · Multi-agent research system

SESSION 3 · AGENT 2 แบบ

Agent ช่วยสร้างระบบ vs Agent ที่ทำงานในระบบ

	Agent ช่วยสร้างระบบ	Agent ที่ทำงานในระบบ
ตัวอย่าง	Claude Code / Codex	แชตบอตตอบลูกค้า, ผู้ช่วยสรุปลีด
ทำงานเมื่อไหร่	เมื่อคนสั่งในเครื่อง	ตลอด 24 ชม. บนเว็บที่ขึ้นแล้ว
อะไรสั่งให้เริ่ม	คนพิมพ์คำสั่ง	เหตุการณ์ (trigger) เช่น ลีดใหม่เข้า ลูกค้าทักแชต ถึงเวลาที่ตั้งไว้
ทำงานร่วมกันผ่าน	ใบสั่งงาน + CLAUDE.md / AGENTS.md + ไฟล์	ฐานข้อมูลกลาง + กฎ Workflow

"อัตโนมัติ" ในระบบจริง · งานต่อยอด

1
เกิดเหตุการณ์
เช่น ลีดใหม่เข้า

→

2
ระบบเรียก AI ตามกฎ
AI สรุปความสนใจ + ร่างข้อความ

→

3
เขียนผลลงฐานข้อมูลกลาง

→

4
ส่วนอื่นอ่านไปทำต่อ
แจกลีดให้ฝ่ายขาย

→

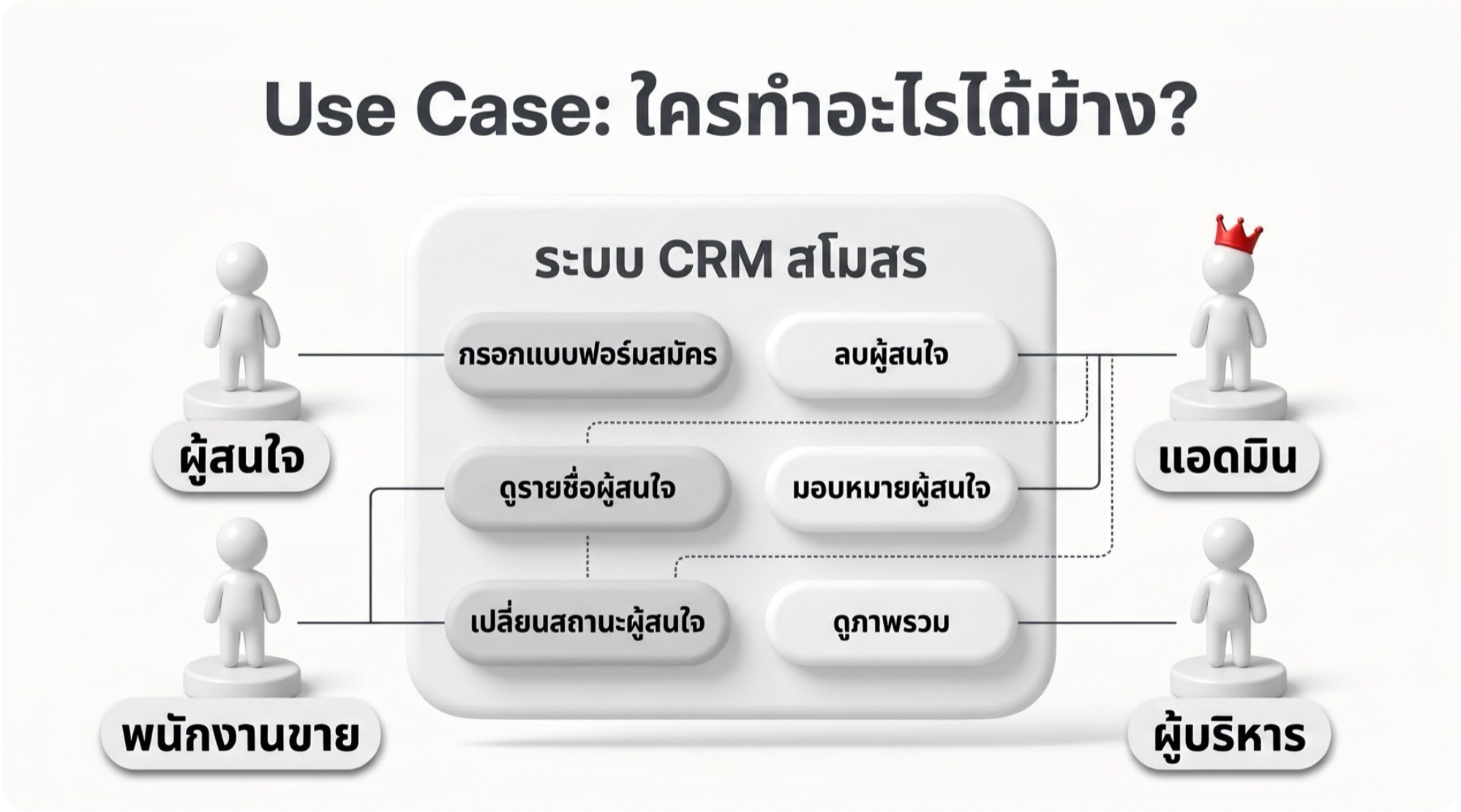
5
เรื่องสำคัญรอคนกดอนุมัติ
ฝ่ายขายกดส่งเอง

W1

WORKSHOP 1 · 11:30–12:30

ออกแบบก่อนสร้าง

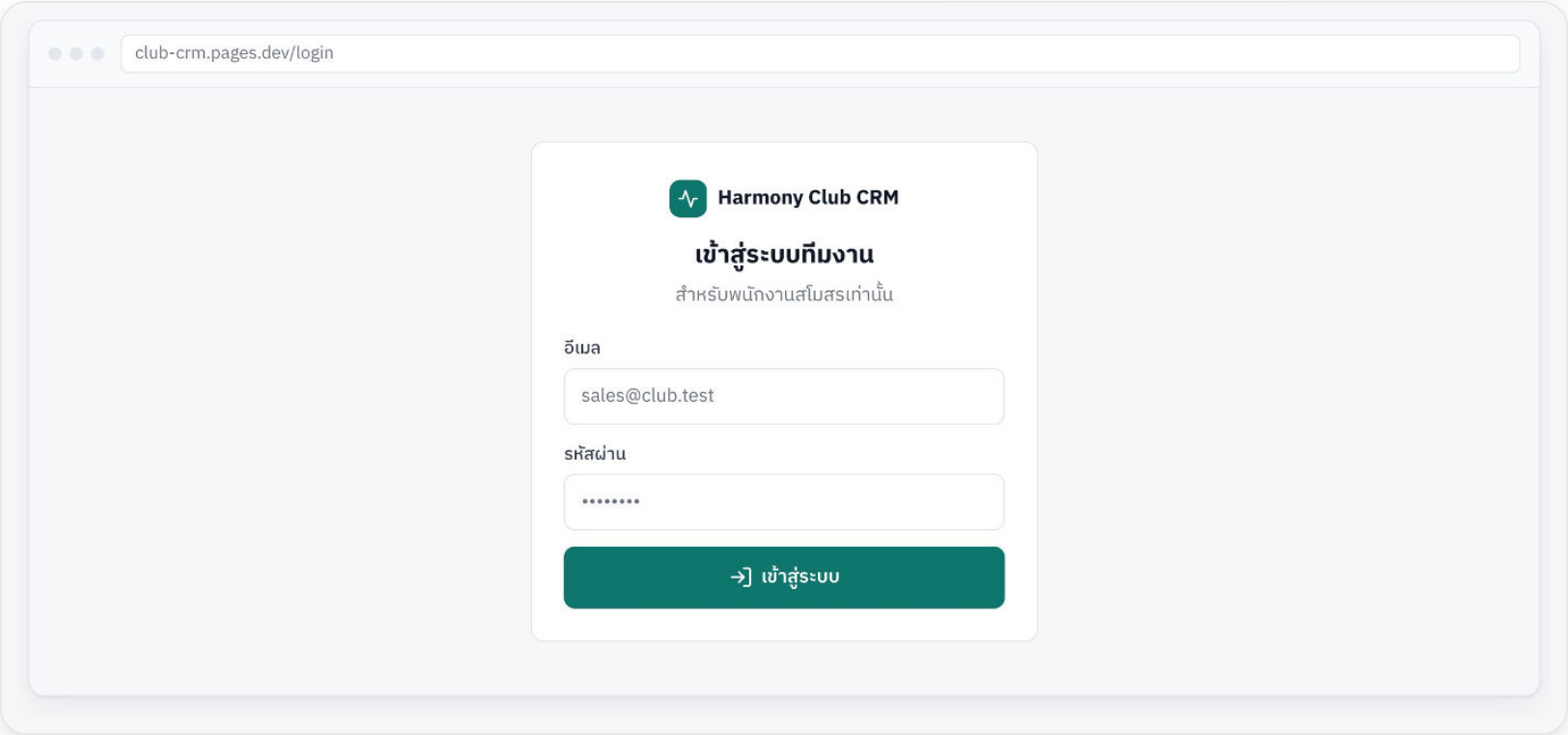
Use Case · หน้าจอตามบทบาท · Permission Matrix · ข้อมูล 3 ตาราง · SPEC.md · ไฟล์กติกา — ได้แบบระบบครบก่อนสั่งสร้าง



4 บทบาท — แต่ละบทบาททำอะไรได้ ดูหน้าจอจริงในสไลด์ถัดไป

1.2 · หน้าจอตามบทบาท · หน้าล็อกอิน

ล็อกอินหน้าเดียว → เปิดเมนูตามสิทธิ์



01
กรอกอีเมล + รหัสผ่าน
หน้า /login

→

02
ระบบตรวจว่าเป็นใคร
Authentication

→

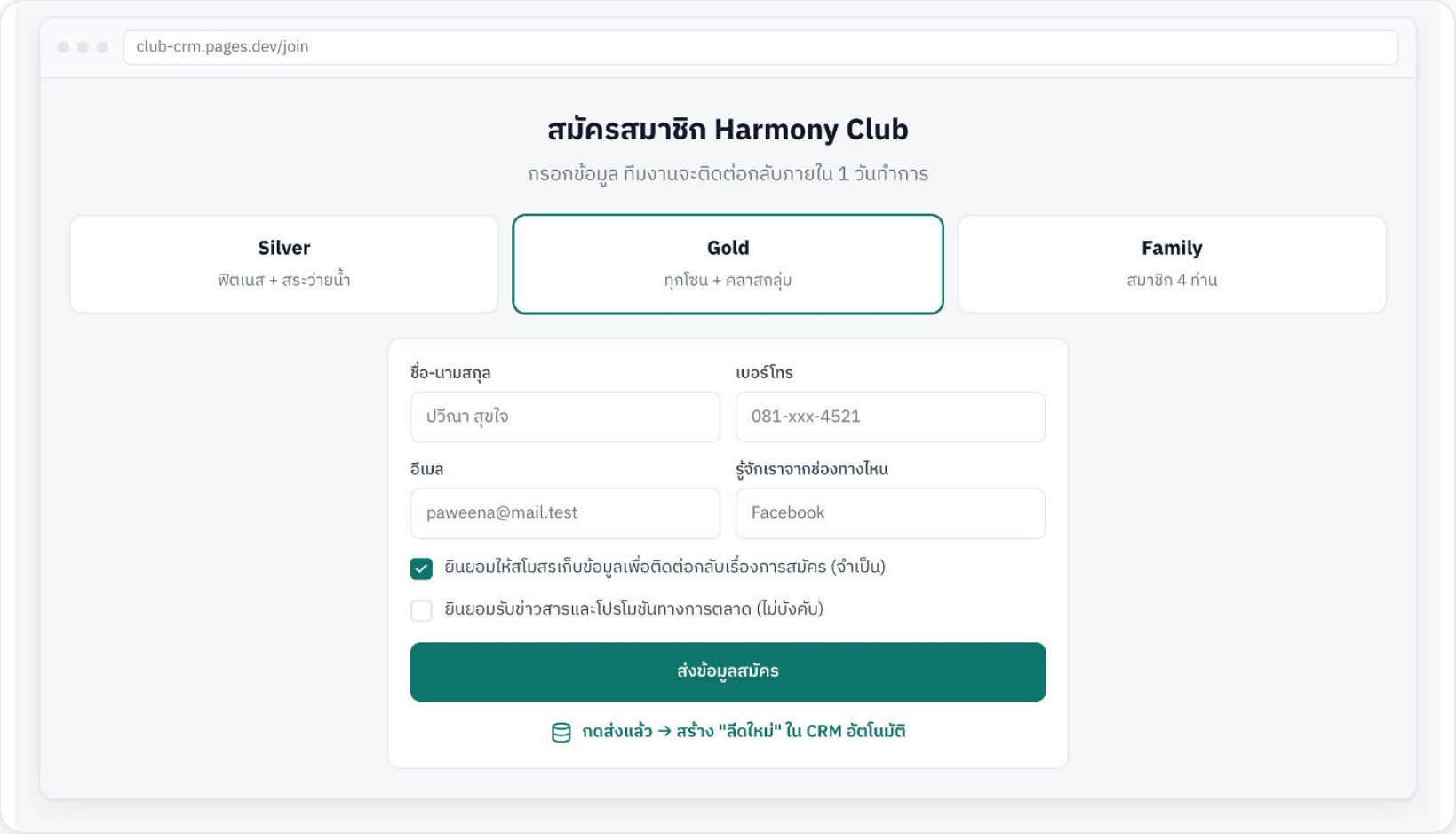
03
อ่านบทบาทจากตาราง staff
admin / sales / manager

→

04
เปิดเมนูตามสิทธิ์
Authorization

1.2 · หน้าจอตามบทบาท · ผู้สนใจ

ผู้สนใจ: ไม่ต้องล็อกอิน กรอกฟอร์มสนใจสมัคร



ทำได้

- ดูประเภทสมาชิกและสิทธิ์
- กรอกฟอร์มสนใจสมัคร
- เลือกให้/ไม่ให้ความยินยอมการตลาด

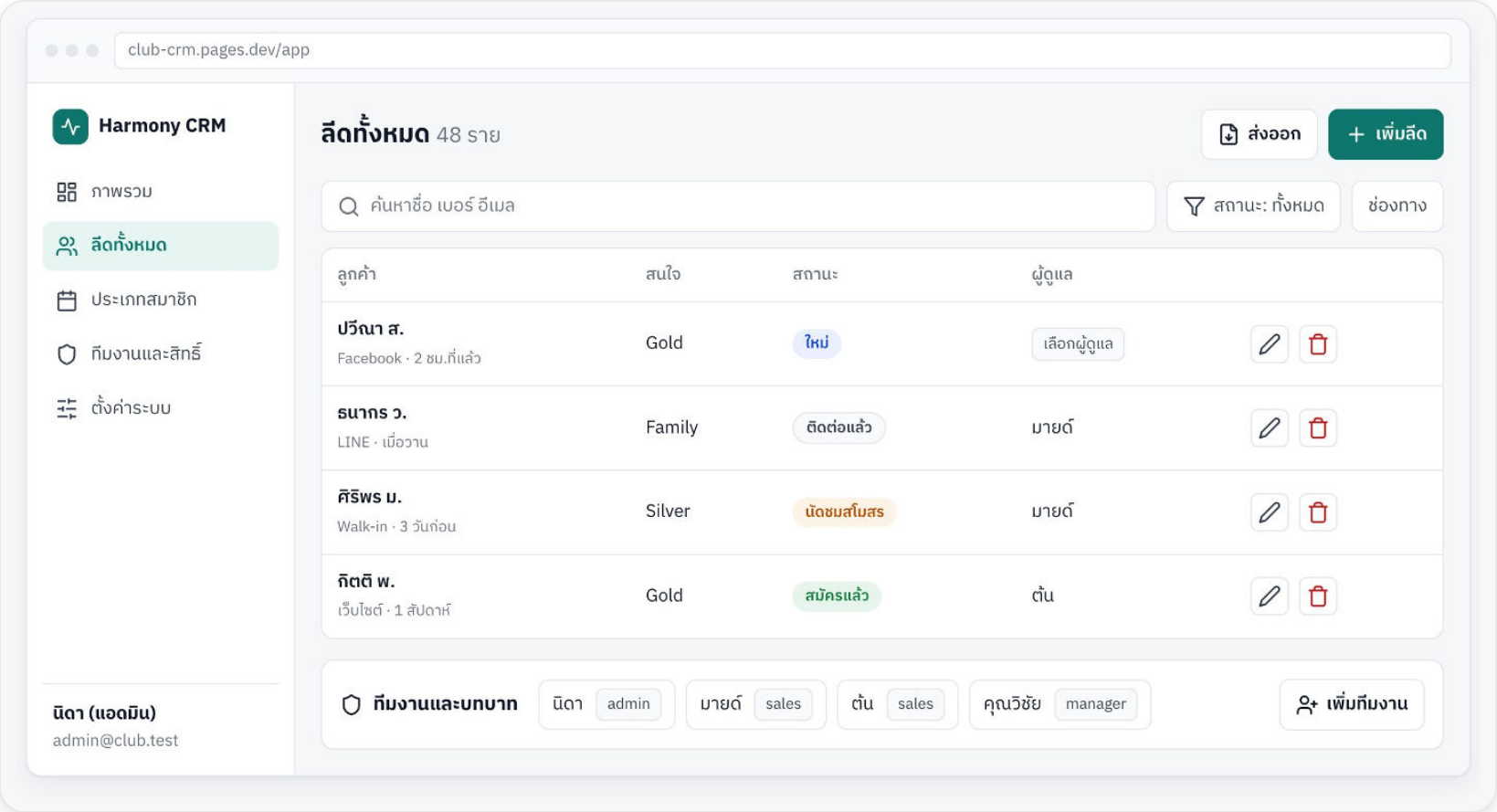
ทำไม่ได้

- เห็นข้อมูลลูกค้าคนอื่น
- เข้าหลังบ้าน

หน้าจอตัวอย่างจาก mockups/roles.html · ชื่อและตัวเลขเป็นข้อมูลจำลอง · กดที่ภาพเพื่อดูเต็มจอ

1.2 · หน้าจอตามบทบาท · แอดมิน

แอดมิน: เห็นลีดทุกคน จัดการได้ครบ



ทำได้

- เห็นลีดทุกคน ค้นหา กรอง
- เพิ่ม/แก้/ลบลีด
- มอบหมายลีดให้พนักงานขาย
- เพิ่มทีมงานและกำหนดบทบาท
- จัดการประเภทสมาชิก

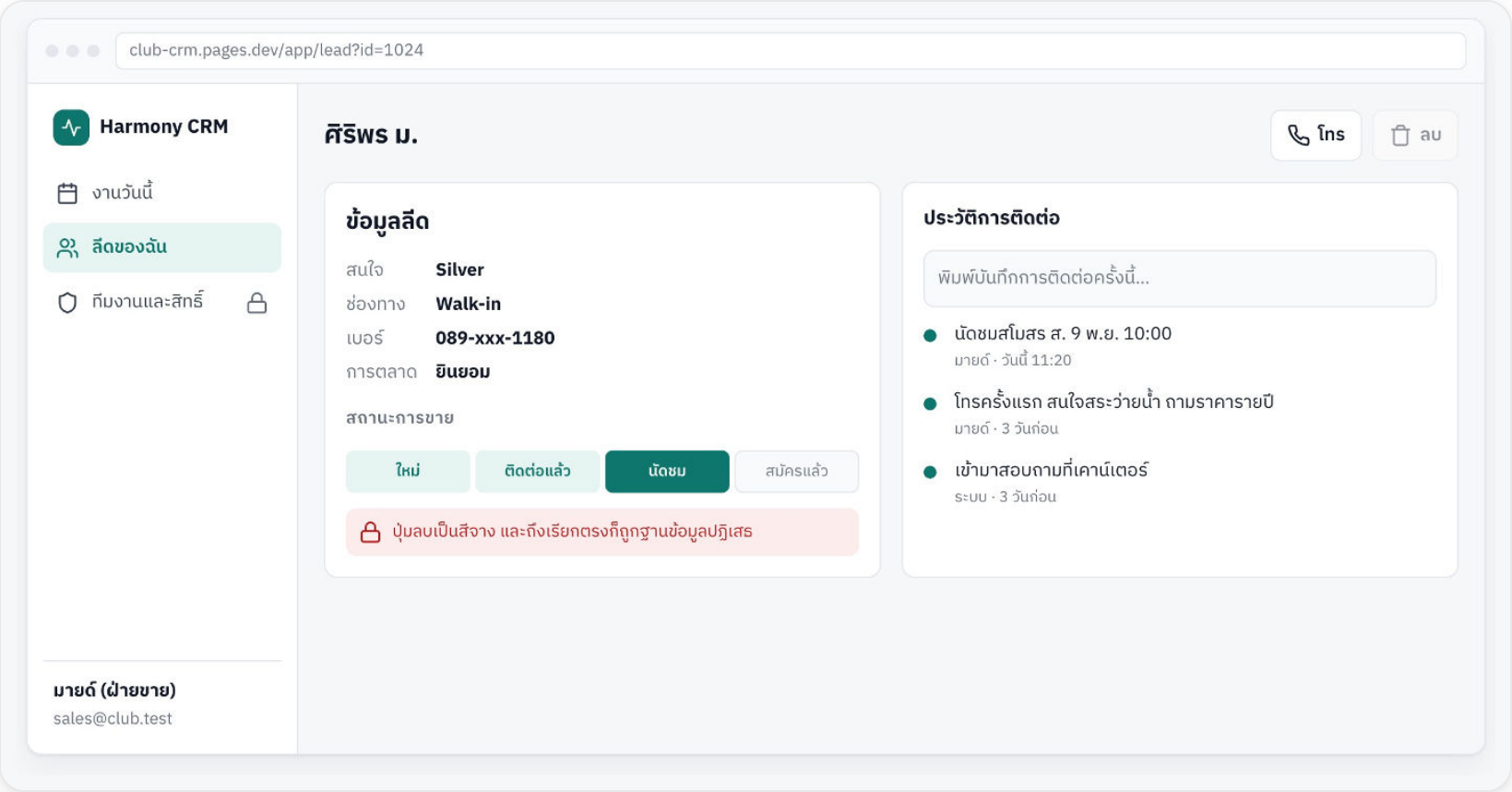
ทำไม่ได้

— แอดมินทำได้ทุกการกระทำในตารางสิทธิ์

หน้าจอตัวอย่างจาก mockups/roles.html · ชื่อและตัวเลขเป็นข้อมูลจำลอง · กดที่ภาพเพื่อดูเต็มจอ

1.2 · หน้าจอตามบทบาท · พนักงานขาย

พนักงานขาย: เฉพาะลีดที่ได้รับมอบหมาย



หน้าจอตัวอย่างจาก mockups/roles.html · ชื่อและตัวเลขเป็นข้อมูลจำลอง · กดที่ภาพเพื่อดูเต็มจอ

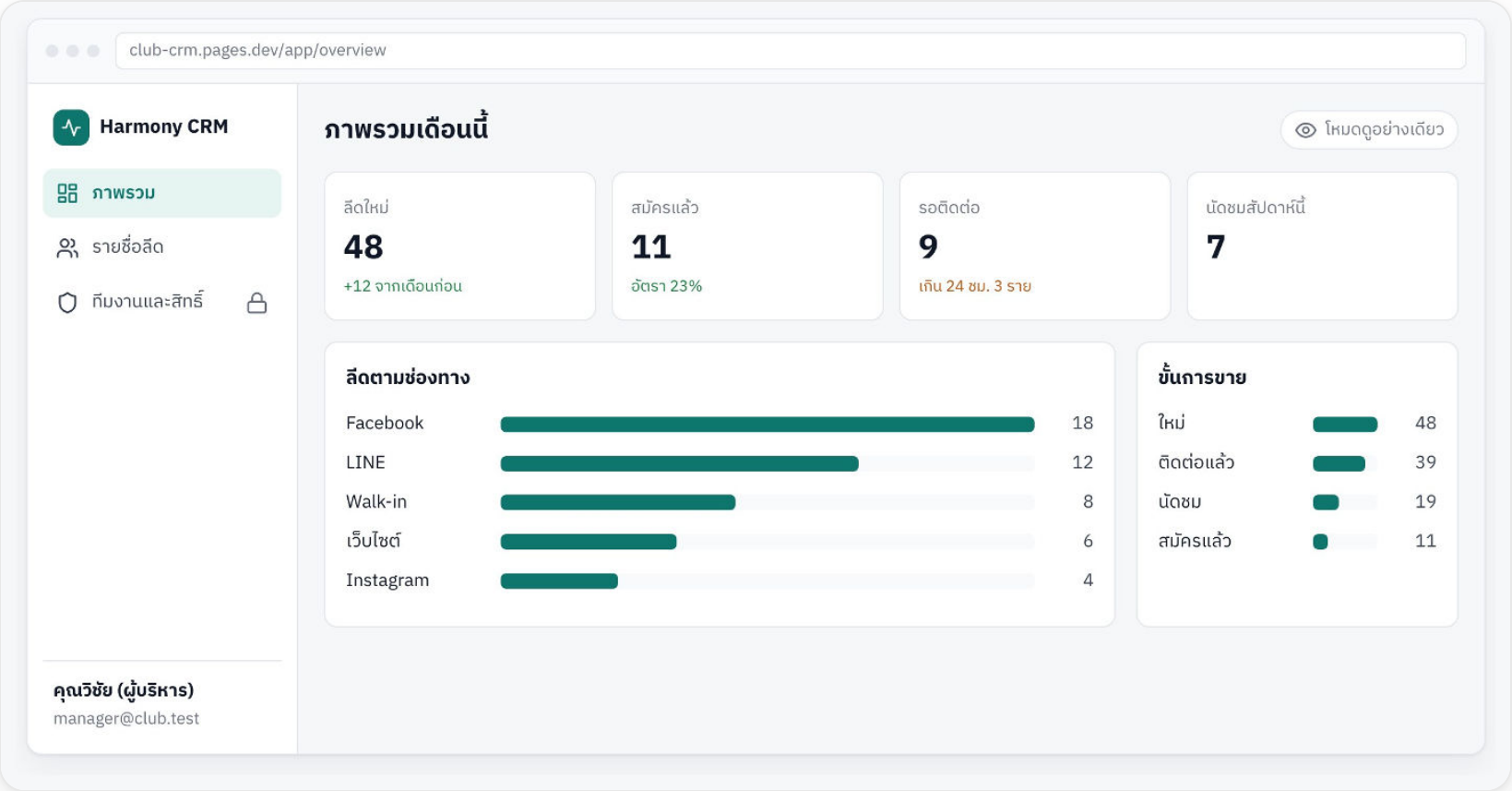
ทำได้

- ดูลีดที่ได้รับมอบหมาย
- เปลี่ยนสถานะตามขั้นการขาย
- บันทึกการโทร/แชต/นัดหมาย

ทำไม่ได้

- ลบลีด
- เพิ่มทีมงานหรือเปลี่ยนบทบาท

ผู้บริหาร: ดูภาพรวมอย่างเดียว



หน้าจอตัวอย่างจาก mockups/roles.html · ชื่อและตัวเลขเป็นข้อมูลจำลอง · กดที่ภาพเพื่อดูเต็มจอ

ทำได้

- ดูตัวเลขสรุป (ลีด อัตราสมัคร งานค้าง)
- ดูลีดตามช่องทางและสถานะ
- ดูรายชื่อลีดทั้งหมด

ทำไม่ได้

- แก้ไขหรือลบข้อมูล (ดูอย่างเดียว)

1.3 · PERMISSION MATRIX

ตารางสิทธิ์ = ด่านทดสอบตอนท้าย Workshop 3

การกระทำ	ผู้สนใจ	แอดมิน	พนักงานขาย	ผู้บริหาร
กรอกฟอร์มสนใจสมัคร	ได้	ได้	ได้	ได้
ดูรายชื่อลีด	ไม่ได้	ได้	เฉพาะของตัวเอง	ได้
เปลี่ยนสถานะ / บันทึกการติดต่อ	ไม่ได้	ได้	ได้	ไม่ได้
ลบลีด	ไม่ได้	ได้	ไม่ได้	ไม่ได้
มอบหมายลีดให้พนักงาน	ไม่ได้	ได้	ไม่ได้	ไม่ได้
ดูภาพรวมตัวเลข	ไม่ได้	ได้	ไม่ได้	ได้
เพิ่มทีมงาน / กำหนดบทบาท	ไม่ได้	ได้	ไม่ได้	ไม่ได้

1.1 + 1.3 · PROMPT USE CASE + ตารางสิทธิ์

ให้ Agent วาด Use Case แล้วทำตารางสิทธิ์ต่อ

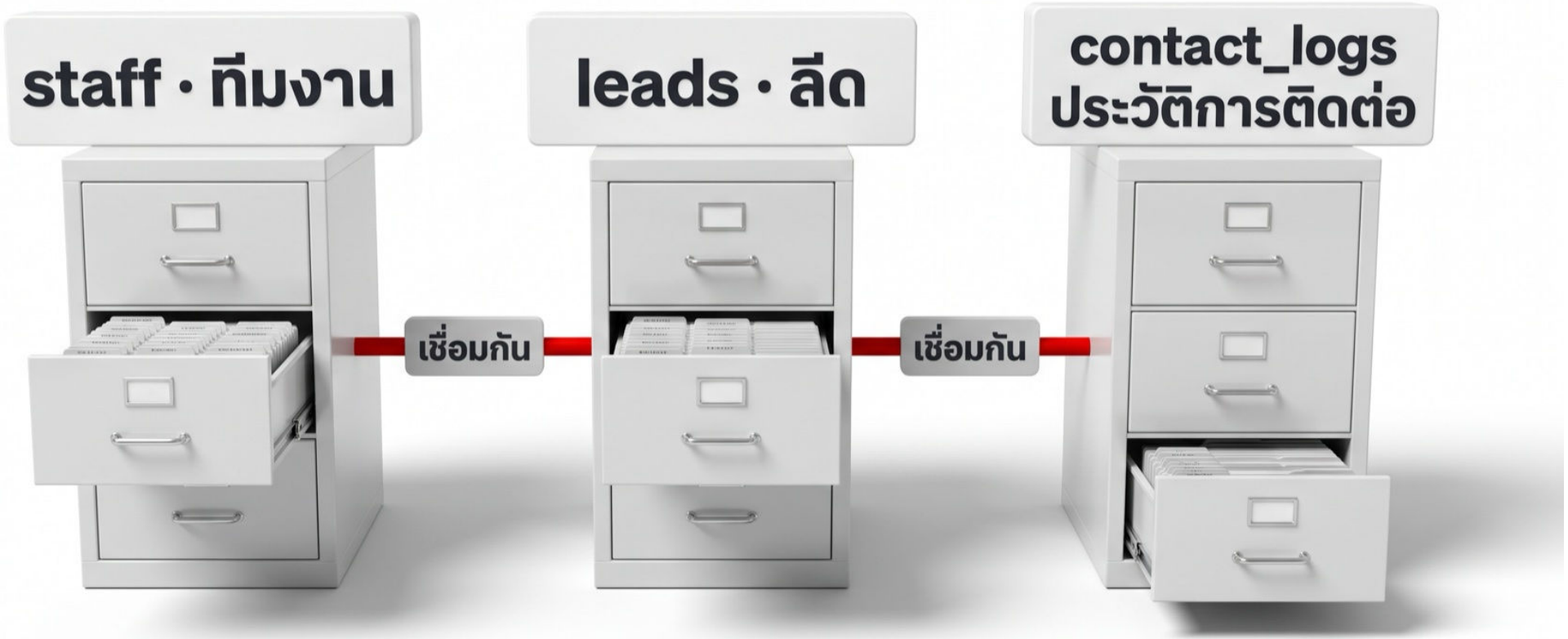
1.1 · USE CASE

ระบบ CRM ของสโมสรสุขภาพ Harmony Club (ชื่อสมมติ) มีผู้ใช้ 4 บทบาท:
1. ผู้สนใจ (ไม่ต้องล็อกอิน): ดูประเภทสมาชิก · กรอกฟอร์มสนใจสมัคร · เลือกให้/ไม่ให้ความยินยอมด้านการตลาด
2. แอดมิน: เห็นลีดทุกคน ค้นหา กรอง · เพิ่ม/แก้/ลบลีด · มอบหมายลีดให้พนักงานขาย · เพิ่มทีมงานและกำหนดบทบาท · จัดการประเภทสมาชิก
3. พนักงานขาย: ดูเฉพาะลีดที่ได้รับมอบหมาย · เปลี่ยนสถานะ · บันทึกการโทร/แชต/นัดหมาย · ลบไม่ได้
4. ผู้บริหาร: ดูตัวเลขสรุปและรายชื่อลีดทั้งหมด · แก้ไขไม่ได้
วาด Use Case Diagram ด้วย Mermaid แล้วบันทึกที่ docs/use-case.md
ใต้แผนภาพ ทำตาราง "บทบาท · ทำได้ · ทำไม่ได้" และบอกว่ามีบทบาทหรือการกระทำไหนที่ฉันลืมคิด (ยังไม่ต้องเพิ่ม ให้ถามก่อน)

1.3 · PERMISSION MATRIX

จาก docs/use-case.md ทำตารางสิทธิ์ (Permission Matrix) บันทึกที่ docs/permissions.md
- แถว = การกระทำ (กรอกฟอร์มสนใจสมัคร · ดูรายชื่อลีด · เปลี่ยนสถานะ/บันทึกการติดต่อ · ลบลีด · มอบหมายลีด · ดูภาพรวมตัวเลข · เพิ่มทีมงาน/กำหนดบทบาท)
- คอลัมน์ = ผู้สนใจ · แอดมิน · พนักงานขาย · ผู้บริหาร
- ช่อง = ได้ / ไม่ได้ / ได้บางส่วน (ระบุเงื่อนไข เช่น "เฉพาะลีดของตัวเอง")
ตารางนี้จะใช้เป็นด่านทดสอบตอนท้าย Workshop 3 ให้เขียนแบบที่ทดสอบได้จริง

ฐานข้อมูล = ตู้เก็บของ 3 ตู้



1.4 · PROMPT ออกแบบข้อมูล (ยังไม่สร้าง)

ให้ AI เสนอ แล้วคนตรวจก่อนอนุมัติ

staff · ทีมงาน

id (ผูกกับบัญชีล็อกอิน) · full_name · role (admin / sales / manager)

leads · ลีด

id · full_name · phone · email · interest_plan (Silver / Gold / Family) · channel (Facebook / LINE / Walk-in / เว็บไซต์ / Instagram) · status · assigned_to (→ staff) · consent_contact · consent_marketing · created_at · updated_at

contact_logs · ประวัติการติดต่อ

id · lead_id (→ leads) · staff_id (→ staff) · note · created_at

1.4 · DATA

ออกแบบฐานข้อมูลสำหรับ CRM นี้บน Supabase (Postgres) — เสนอให้ดูก่อน ยังไม่ต้องสร้างจริง
ใช้ 3 ตาราง:
- staff: ทีมงาน ผูกกับบัญชีล็อกอินของ Supabase Auth · ชื่อ · role (admin / sales / manager)
- leads: ลีด · ชื่อ · เบอร์ · อีเมล · ประเภทที่สนใจ (Silver / Gold / Family) · ช่องทางที่รู้จัก · สถานะ (new / contacted / tour / member / lost) · ผู้ดูแล (→ staff) · ยินยอมให้ติดต่อ · ยินยอมการตลาด · เวลาสร้าง/แก้ไข
- contact_logs: ประวัติการติดต่อ · ลีด (→ leads) · พนักงาน (→ staff) · บันทึก · เวลา
แสดงเป็นตาราง: ชื่อคอลัมน์ · ชนิดข้อมูล · ตัวอย่างค่า · ความหมายภาษาไทย
แล้ววาดความสัมพันธ์ด้วย Mermaid (erDiagram) บันทึกที่ docs/data.md
ถ้าเห็นว่าควรเพิ่มหรือตัดอะไร ให้ถามฉันเป็นตัวเลือกให้กดเลือก หรือให้ตอบสั้น ๆ ทีละชุด แล้วรอคำตอบ



ค่าที่เก็บในฐานข้อมูล: `new → contacted → tour → member / lost`

1.5 + 1.6 · SPEC.MD + ไฟล์กติกาทีม

เขียน ใบสั่งงานที่ดี ให้ Agent ทุกตัวอ่าน

1.5 · SPEC.md

รวมทุกอย่างที่เราตกลงกันเขียนเป็น SPEC.md ที่ root ของโปรเจกต์ ให้ Agent ตัวไหนมาอ่านก็สร้างตามได้:
1. เป้าหมายระบบ (CRM สโมสร เวอร์ชันพื้นฐาน · ข้อมูลจำลองเท่านั้น)
2. Stack: HTML/CSS/JavaScript ธรรมดา ไม่มี build step · supabase-js จาก CDN · Backend = Cloudflare Pages Functions · Database = Supabase (Postgres + Auth + Row Level Security)
3. บทบาทและตารางสิทธิ์ (อ้าง docs/use-case.md และ docs/permissions.md)
4. ข้อมูล 3 ตาราง (อ้าง docs/data.md)
5. หน้าจอ: /join ฟอร์มสนใจสมัคร · /login · /app รายการลีด · /app/lead หน้ารายละเอียดลีด · /app/overview ภาพรวมผู้บริหาร · /app/team จัดการทีมงาน (แอดมิน)
6. แบ่งงานตาม 3 ชั้น: Database (supabase/) · Backend (functions/api/) · Frontend (public/) — ระบุไฟล์ของแต่ละชั้นให้ชัด และสร้างทีละชั้นตามลำดับนี้ ทดสอบทุกชั้นก่อนไปต่อ
7. ด่านทดสอบ: ด่าน 1 ฟอร์ม → ลีดเข้า CRM · ด่าน 2 สลับล็อกอิน 3 บทบาทตรงตารางสิทธิ์ · ด่าน 3 เปิดบนมือถือได้
8. สิ่งที่ยังไม่ทำ (ต่อยอดภายหลัง): แบบประเมิน % · Audit Log · Waiting List · ระบบจอง · สัญญา · แชตบอต

1.6 · CLAUDE.md / AGENTS.md

สร้างไฟล์บันทึกโปรเจกต์ (CLAUDE.md / AGENTS.md) ให้เป็น "คู่มือกติกาทีม" ที่ Agent และ Subagent ทุกตัวต้องอ่านก่อนทำงาน ใส่:
- โปรเจกต์นี้คืออะไร + ให้อ่าน SPEC.md ก่อนเริ่มงานทุกครั้ง
- กติกาข้อมูล: ใช้ข้อมูลจำลองเท่านั้น · ห้ามใส่ service_role key หรือคีย์ลับใด ๆ ในโฟลเดอร์ public/ · คีย์ลับอยู่ใน .dev.vars และ Cloudflare secret เท่านั้น
- กติกาสิทธิ์: ทุกตารางเปิด Row Level Security · สิทธิ์ต้องตรงกับ docs/permissions.md
- กติกาการทำงาน: ทำทีละชั้น แก้เฉพาะไฟล์ของชั้นที่กำลังทำ · ทำเสร็จให้รายงานว่าแก้ไฟล์อะไร ทดสอบอะไร แล้วรอให้ฉันทดสอบ · ไม่แน่ใจให้ถาม ห้ามเดา
- ภาษา: อธิบายผู้ใช้เป็นภาษาไทย ข้อความบนหน้าเว็บเป็นภาษาไทย
เขียนให้สั้น อ่านจบใน 1 นาที

12:30-13:30 · พักรับประทานอาหารกลางวัน

พักเที่ยง

ช่วงบ่ายเริ่มสร้าง CRM จริงด้วย Main Agent ทีละชั้น

W2

WORKSHOP 2 · 13:30–15:00

สร้าง CRM ทีละชั้น

Main Agent ตัวเดียว · Database → Backend → Frontend · ด่าน 1: กรอกฟอร์มแล้วลีดโผล่ใน CRM · เปลี่ยนสถานะแล้วค่าในตารางเปลี่ยน

2.1 · เชื่อมโปรเจกต์กับ SUPABASE

คีย์แต่ละตัว อยู่ไฟล์ไหน

public/config.js

SUPABASE_URL + SUPABASE_ANON_KEY

เปิดเผยได้ (หน้าเว็บ)

.dev.vars

SUPABASE_URL + SUPABASE_SERVICE_ROLE_KEY

ไฟล์ลับ ใช้เฉพาะ Backend · ไม่ขึ้น Git

2.1 · CONNECT

ช่วยตั้งค่าเชื่อมโปรเจกต์กับ Supabase:
- สร้าง public/config.js เก็บ SUPABASE_URL และ SUPABASE_ANON_KEY (คีย์นี้เปิดเผยได้)
- สร้าง .dev.vars เก็บ SUPABASE_URL และ SUPABASE_SERVICE_ROLE_KEY (คีย์ลับ ใช้เฉพาะ Backend) และยืนยันว่า .dev.vars อยู่ใน .gitignore
- ทำช่องว่างไว้ แล้วบอกฉันว่าต้องไปคัดลอกค่าไหนจากหน้าไหนใน Supabase มาวางเอง (ฉันจะวางคีย์เองในไฟล์ ไม่ส่งในแชต)

วางคีย์เองในไฟล์ — ไม่ส่งคีย์ในแชต

สร้างทีละชั้น ทดสอบทุกชั้นก่อนไปต่อ

1 · Database `supabase/`

3 ตาราง + ข้อมูลจำลอง 20 ราย · **ทดสอบ:** เห็นตารางใน Table Editor

2 · Backend `functions/api/`

รับฟอร์ม ตรวจ แล้วสร้างลีดด้วยคีย์ลับ · **ทดสอบ:** ส่งข้อมูลตัวอย่างแล้วลีดเข้า

3 · Frontend `public/`

/join ฟอร์ม · /app รายการลีด · รายละเอียดลีด · **ทดสอบ:** ด่าน 1

2.2 · BUILD LAYER BY LAYER

อ่าน SPEC.md และไฟล์บันทึกโปรเจกต์ แล้วสร้าง CRM เวอร์ชันแรกด้วยตัวเองทีละชั้นตามลำดับ (ไม่ต้องแตก subagent)
ทำเสร็จแต่ละชั้นให้หยุด สรุปสั้น ๆ ว่าแก้ไฟล์อะไร แล้วบอกวิธีให้ฉันทดสอบ รอฉันยืนยันก่อนไปชั้นถัดไป
1. Database (ฐานข้อมูล) — แก้เฉพาะ supabase/
   schema.sql สร้าง 3 ตาราง staff / leads / contact_logs ตาม docs/data.md และเปิด Row Level Security ทุกตาราง (ช่วงนี้อนุญาตให้ผู้ที่ล็อกอินอ่าน/แก้ได้ก่อน จะล็อกตามบทบาทใน Workshop 3)
   seed.sql ข้อมูลจำลอง 20 ลีด ชื่อไทยสมมติ กระจายสถานะและช่องทาง
   → บอกให้ฉันไปวางใน Supabase SQL Editor แล้วเปิด Table Editor ดูว่ามีตารางและข้อมูลครบ
2. Backend (หลังบ้าน) — แก้เฉพาะ functions/api/
   join.js รับข้อมูลฟอร์ม ตรวจความถูกต้อง (ชื่อ เบอร์ ต้องมี · ต้องยินยอมให้ติดต่อ) แล้วสร้างลีดสถานะ new ด้วย service_role key จาก env
   → ทดสอบส่งข้อมูลตัวอย่างเข้า /api/join แล้วรายงานผล
3. Frontend (หน้าบ้าน) — แก้เฉพาะ public/
   หน้า /join ฟอร์มสนใจสมัคร (ชื่อ เบอร์ อีเมล ประเภทที่สนใจ ช่องทาง ช่องยินยอมให้ติดต่อ [จำเป็น] แยกกับช่องยินยอมการตลาด [ไม่บังคับ]) ส่งไป /api/join
   หน้า /app รายการลีด ค้นหา กรองตามสถานะ/ช่องทาง/ประเภท · หน้ารายละเอียดลีด เปลี่ยนสถานะ + ไทม์ไลน์ประวัติการติดต่อ + ช่องเพิ่มบันทึก
ใช้ชื่อตาราง คอลัมน์ และรูปแบบข้อมูลเดียวกันทั้ง 3 ชั้นตาม SPEC.md

2.3-2.4 · สร้างตารางจริง + รันบนเครื่อง

เห็น Database เป็นรูปธรรมใน Table Editor

1. **Supabase → SQL Editor**
   ทำเองบนเว็บ
2. **วาง schema.sql → Run**
   สร้าง 3 ตาราง staff / leads / contact_logs
3. **วาง seed.sql → Run**
   ข้อมูลจำลอง 20 ราย
4. **เปิด Table Editor**
   เห็นตาราง = เห็น Database เป็นรูปธรรม

2.4 · RUN + TEST

ฉันรัน schema.sql และ seed.sql ใน Supabase SQL Editor แล้ว
ช่วยรันเว็บบนเครื่องด้วย npx wrangler pages dev (ให้ Pages Functions ทำงานด้วย) แล้วบอกลิงก์ที่ต้องเปิด
จากนั้นบอกขั้นตอนทดสอบด่าน 1 ทีละข้อ:
1. กรอกฟอร์ม /join ด้วยข้อมูลทดสอบ → ลีดใหม่ต้องโผล่ในหน้า /app และใน Table Editor
2. เปลี่ยนสถานะลีดในหน้า CRM → ค่าใน Table Editor ต้องเปลี่ยนตาม
ถ้ามี error ให้อ่าน log แล้วแก้ให้ อธิบายสาเหตุสั้น ๆ

ด่าน 1 · 2.5 ตั้งจุดย้อนกลับ

ด่าน 1 ฟอร์ม → ลีดเข้า CRM

ด่าน 1

- **กรอกฟอร์ม /join ด้วยข้อมูลทดสอบ**
  ลีดโผล่ในหน้า CRM + ใน Table Editor
- **เปลี่ยนสถานะลีดในหน้า CRM**
  ค่าใน Table Editor เปลี่ยนตาม

2.5 · GIT

ผ่านด่าน 1 แล้ว ช่วยบันทึกจุดย้อนกลับด้วย Git ข้อความ "CRM v1: ฟอร์มสนใจสมัคร + หลังบ้านจัดการลีด"
ตรวจก่อนว่าไม่มีไฟล์ลับ (.dev.vars) ถูกบันทึกไปด้วย

> ผ่านด่าน 1 แล้วค่อยบันทึกจุดย้อนกลับ — ตรวจว่าไม่มีไฟล์ลับ (.dev.vars) ถูกบันทึกไปด้วย

15:00–15:15 · พักเบรก

เบรก

ต่อด้วย Workshop 3 · ระบบล็อกอินและบทบาท

W3

WORKSHOP 3 · 15:15–16:30

ระบบล็อกอินและบทบาท

แอดมิน / พนักงานขาย / ผู้บริหาร + ล็อกสิทธิ์ที่ฐานข้อมูล · ด่าน 2: สลับล็อกอิน 3 บทบาท เห็น/ทำได้ตรง Permission Matrix

3.1-3.2 · บัญชีทดสอบ + ผูกบทบาท

สร้าง 3 บัญชี แล้วผูกกับบทบาท

1. **Supabase → Authentication → Users**
   ทำเองบนเว็บ
2. **Add user → Create new user**
   ติ๊ก Auto Confirm
3. **สร้าง 3 บัญชี**
   `admin@club.test · sales@club.test · manager@club.test`
4. **วาง SQL ที่ได้ใน SQL Editor → Run**
   ผูกบัญชีกับบทบาทในตาราง staff

3.2 · ROLES

ฉันสร้างบัญชีทดสอบใน Supabase Authentication แล้ว 3 บัญชี:
admin@club.test · sales@club.test · manager@club.test
เขียน SQL ไฟล์ supabase/staff.sql ที่เพิ่มทั้ง 3 บัญชีลงตาราง staff พร้อมบทบาท admin / sales / manager (หา id จาก auth.users ด้วยอีเมล ไม่ต้องให้ฉันคัดลอก id เอง)
และมอบหมายลีดจำลองครึ่งหนึ่งให้ sales@club.test
บอกฉันว่าต้องไปวางที่ SQL Editor แล้วกด Run

3.3 · หน้าล็อกอิน + หน้าจอตามบทบาท

แนบหน้าจอตัวอย่าง ให้ Agent ทำตามแบบ

แอดมิน

ลีดทั้งหมด (เพิ่ม/แก้/ลบ มอบหมายผู้ดูแล) · ทีมงานและสิทธิ์

พนักงานขาย

ลีดของฉัน (เปลี่ยนสถานะ บันทึกการติดต่อ) · ไม่มีปุ่มลบ

ผู้บริหาร

ภาพรวม (ตัวเลขสรุป + ลีดตามช่องทางและขั้นการขาย) · รายชื่อลีดแบบดูอย่างเดียว

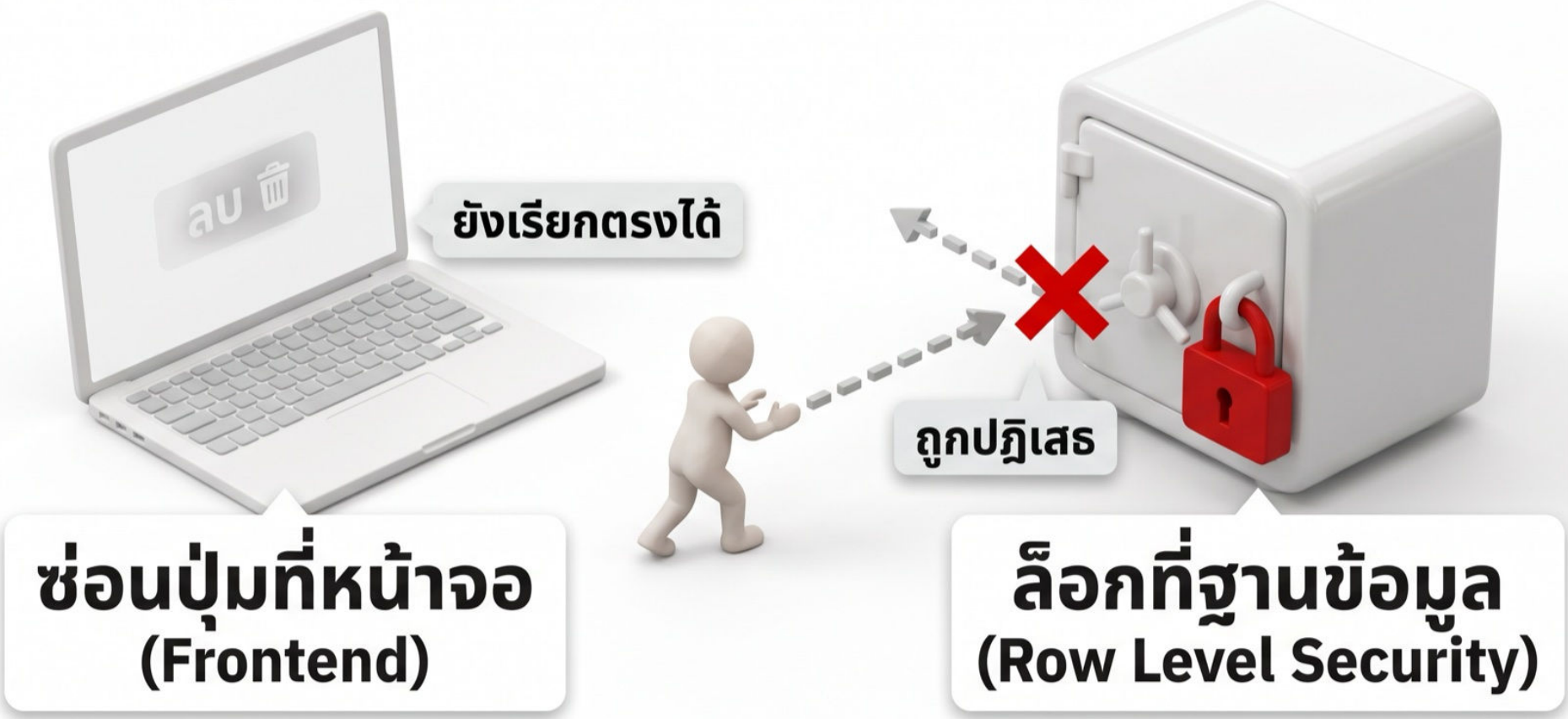
3.3 · LOGIN

ทำระบบล็อกอินด้วย Supabase Auth (อีเมล + รหัสผ่าน) และหน้าจอตามบทบาท ให้หน้าตาใกล้กับหน้าจอตัวอย่างที่แนบ [แนบภาพหน้าจอจาก mockups/roles.html]:

- /login หน้าเดียวสำหรับทุกบทบาท · ทุกหน้าใน /app ต้องล็อกอินก่อน ไม่งั้นเด้งกลับ /login
- หลังล็อกอิน อ่าน role จากตาราง staff แล้วเปิดเมนูตามบทบาท:
  แอดมิน — ลีดทั้งหมด (เพิ่ม/แก้/ลบ มอบหมายผู้ดูแล) · ทีมงานและสิทธิ์
  พนักงานขาย — ลีดของฉัน (เปลี่ยนสถานะ บันทึกการติดต่อ) · ไม่มีปุ่มลบ
  ผู้บริหาร — ภาพรวม (ตัวเลขสรุป + ลีดตามช่องทางและขั้นการขาย) · รายชื่อลีดแบบดูอย่างเดียว
- มุมล่างของเมนูแสดงชื่อและบทบาทของคนที่ล็อกอิน + ปุ่มออกจากระบบ

ทำทีละส่วน: ระบบล็อกอินก่อน แล้วหน้าจอทีละบทบาท หยุดให้ฉันทดสอบทุกส่วน

ซ่อนปุ่มไม่พอ ต้องล็อกที่ตู้



3.4 · PROMPT ROW LEVEL SECURITY

ล็อกที่ ฐานข้อมูล ให้ตรงตารางสิทธิ์

- **ทุกตารางเปิด Row Level Security**
  สิทธิ์ต้องตรงกับ docs/permissions.md
- **อธิบายทีละ policy เป็นภาษาคน**
  แล้วไปวางที่ SQL Editor

3.4 · RLS

```
ตอนนี้ซ่อนปุ่มที่หน้าจอแล้ว แต่ยังต้องล็อกที่ฐานข้อมูลด้วย
เขียน supabase/policies.sql ตั้ง Row Level Security ให้ตรงกับ
docs/permissions.md:
- leads: แอดมินและผู้บริหารอ่านได้ทุกแถว · พนักงานขายอ่าน/แก้ได้เฉพาะแถวที่
assigned_to เป็นตัวเอง · ลบได้เฉพาะแอดมิน · ผู้บริหารแก้ไม่ได้
- contact_logs: เพิ่มได้เฉพาะแอดมินและพนักงานขาย (เฉพาะลีดของตัวเอง) · อ่านได้ตาม
สิทธิ์ของลีด
- staff: ทุกคนที่ล็อกอินอ่านได้ · แก้ได้เฉพาะแอดมิน
อธิบายทีละ policy เป็นภาษาคนสั้น ๆ แล้วบอกให้ฉันไปวางที่ SQL Editor
```

3.5 · ทดสอบสิทธิ์ · ด่าน 2

ด่าน 2 สลับล็อกอิน 3 บทบาท

3.5 · TEST

ทำเช็คลิสต์ทดสอบด่าน 2 จาก docs/permissions.md ให้ฉันกดทดสอบเองทีละบทบาท (admin / sales / manager)
และทดสอบที่ฐานข้อมูลโดยตรงด้วย: ล็อกอินเป็น sales แล้วลองสั่งลบลีดผ่าน supabase-js ใน console ของเบราว์เซอร์ — ต้องถูกปฏิเสธ
บอกโค้ด 1 บรรทัดที่ต้องวางใน console และผลที่ควรเห็น

ด่าน 2

- **เห็น/ทำได้ตรงกับ Permission Matrix**
  สลับล็อกอินทั้ง 3 บทบาท
- **พนักงานขายลบไม่ได้**
  แม้เรียกฐานข้อมูลตรง
- **ผู้บริหารแก้ไม่ได้**
  ดูอย่างเดียว

W4

WORKSHOP 4 · 16:30–17:00

ขึ้นเว็บจริง + สรุป + แผนต่อยอด

ด่าน 3: เปิด URL บนมือถือแล้วล็อกอินได้

4.1 · DEPLOY · ด่าน 3

ขึ้นเว็บจริงบน Cloudflare Pages

1 **Deploy ขึ้น Cloudflare Pages**
หน้าเว็บ + Backend ผ่าน Pages Functions

2 **ตั้ง secret ฝั่ง Cloudflare**
SUPABASE_SERVICE_ROLE_KEY (ห้ามใส่ในโค้ด)

3 **ใส่ URL เว็บใน Supabase**
Authentication → URL Configuration

ด่าน 3

เปิด URL บนมือถือ
ล็อกอินได้ → กรอกฟอร์มได้

4.1 · DEPLOY

```
ช่วย Deploy โปรเจกต์นี้ขึ้น Cloudflare Pages:
- ชื่อโปรเจกต์ club-crm-[ชื่อเล่นของฉัน] · โฟลเดอร์หน้าเว็บ public/ · Functions อยู่ใน functions/
- ตั้งค่า SUPABASE_URL และ SUPABASE_SERVICE_ROLE_KEY เป็น secret ฝั่ง Cloudflare (ห้ามใส่ในโค้ด)
- บอกให้ฉันไปใส่ URL ของเว็บใน Supabase → Authentication → URL Configuration
- ให้ลิงก์เว็บจริง แล้วบอกขั้นตอนทดสอบด่าน 3 บนมือถือ
ถ้าต้องล็อกอิน Cloudflare ให้บอกฉันก่อน ฉันจะล็อกอินเอง
```

4.2 · ปิดวัน + ทบทวน

ทบทวน 4 เรื่อง แล้วบันทึกว่าทำถึงไหน

3 ชั้น

Frontend หน้าบ้าน · Backend หลังบ้าน · Database ฐานข้อมูล

Use Case

4 บทบาท + Permission Matrix ก่อนสั่งสร้าง

Main Agent ทีละชั้น

งานเดฟสร้างด้วย Main Agent ตัวเดียว · Subagent ใช้ช่วยอ่าน ค้น ตรวจ

กติกา 3 ข้อ

ข้อมูลจำลอง · คีย์ลับไม่อยู่บนหน้าเว็บ · คนอนุมัติเรื่องสำคัญ

4.2 · WRAP-UP

อัปเดตไฟล์บันทึกโปรเจกต์ (CLAUDE.md / AGENTS.md) ว่า:
- ตอนนี้ระบบทำอะไรได้แล้ว ผ่านด่านไหนแล้ว ลิงก์เว็บจริงคืออะไร
- ไฟล์สำคัญอยู่ที่ไหน
- งานต่อยอดที่ยังไม่ได้ทำ (จาก SPEC.md ข้อ 8)

แล้วบันทึกจุดย้อนกลับด้วย Git ข้อความ "Day 1 complete"

เช็คลิสต์ก่อนใช้จริง (อ่านอย่างเดียว ไม่ทำในคลาส): เปลี่ยนจากข้อมูลจำลองเป็นข้อมูลจริงต้องผ่านเช็คลิสต์ความปลอดภัย · นโยบาย PDPA · สำรองข้อมูล · รีเซ็ตรหัสผ่าน



เพื่อให้ผู้เรียนเข้าใจพื้นฐานอย่างแท้จริงภายใน 1 วัน คลาสนี้สอนเฉพาะแกนกลางของระบบ ส่วนที่เหลือเป็นงานต่อยอดที่ผู้เรียนทำต่อเองได้ด้วยวิธีเดียวกัน

แผนต่อยอด · PROMPT ตั้งต้นทุกข้อ

ต่อยอดจาก ฐานข้อมูลกลาง + CRM เดิม ด้วยวิธีเดียวกัน

#	งานต่อยอด	ต่อจากของวันนี้อย่างไร	Prompt
1	**Join Now ขั้น 1–2: หน้า Explore + Ask AI + แบบประเมินความเหมาะสม %**	เพิ่มหน้าก่อนฟอร์ม /join ผลประเมินเก็บลงลีด	
2	**ขั้นอนุมัติแบบเต็ม + Audit Log**	เพิ่มสถานะ อนุมัติ/ปฏิเสธ/รอคิว + ตาราง audit_logs	
3	**Waiting List + จับคู่ซื้อ/ขาย/โอนสิทธิ์**	ตาราง waitlist / listings เชื่อมกับลีดและสมาชิก	
4	**ระบบจองห้อง/บริการ**	ตาราง bookings เชื่อมกับสมาชิก	
5	**ร่างสัญญาจากข้อมูลสมาชิก**	ดึงข้อมูลจาก leads/members ไปเติมแม่แบบ + ขั้นอนุมัติ	
6	**Chatbot / FAQ / LINE OA**	ตาราง faqs + แชตส่งลีดเข้า CRM เดิม	
7	**Agent ที่ทำงานในระบบ + แจ้งเตือนทีมขาย**	trigger ลีดใหม่ → AI สรุป/ร่างข้อความ → คนกดอนุมัติ	
8	**Dashboard ผู้บริหารแบบละเอียด + ล็อกอินขั้นสูง**	ต่อจากหน้าผู้บริหาร · ลืมรหัสผ่าน / Google / 2FA	

นอกขอบเขตหลักสูตร: งานผลิตสื่อ · การตั้งเวลาโพสต์ · ค้นหา/เข้ากลุ่มออนไลน์อัตโนมัติ · ระบบชำระเงินออนไลน์ · ลายเซ็นอิเล็กทรอนิกส์

zenityx

สรุปเนื้อหา · ถาม-ตอบ

Q&A

ขอบคุณที่ร่วมเรียนวันนี้

ทุกคำถามยินดีตอบ — ทั้งเรื่องระบบที่สร้างวันนี้ และการต่อยอดในงานจริงของสโมสร

โทร
062-262-9644

อีเมล
admin@zenityxai.com

