มหาวิทยาลัยอุบลราชธานี

คณะรัฐศาสตร์

รายงานการถอดบทเรียนและถ่ายทอดองค์ความรู้

(Community of Practice: CoP)

ชื่อองค์ความรู้ เรื่อง

การพัฒนาบุคลากรสายสนับสนุนวิชาการให้มีทักษะการใช้ AI
ในการสร้าง WebApp สำนักงานด้วย ChatGPT (Codex)
และเชื่อมฐานข้อมูลจริงด้วย Supabase

กลุ่ม CoP

Future Office POLUBU

1. ประเด็นความรู้และการขับเคลื่อน (Knowledge Themes & Drivers)

1.1 ความสอดคล้องกับยุทธศาสตร์มหาวิทยาลัย: [/] ด้านการบริหารจัดการองค์กร
(การเปลี่ยนผ่านสู่มหาวิทยาลัยดิจิทัลและลดขั้นตอนการทำงาน)

1.2 ความสอดคล้องกับยุทธศาสตร์คณะ: [/] ด้านการบริหารจัดการองค์กรที่เป็นเลิศ

2. โจทย์ความรู้ / ปัญหาที่พบ (Pain Point)

การพัฒนาบุคลากรสายสนับสนุนวิชาการให้มีทักษะการใช้ AI ในการสร้าง WebApp สำนักงานด้วย ChatGPT (Codex) และเชื่อมฐานข้อมูลจริงด้วย Supabase

ที่มาของโจทย์ งานสำนักงานของคณะจำนวนมากยังทำงานบนไฟล์ Excel ที่ส่งต่อกันไปมา เช่น ทะเบียนลูกหนี้ ทะเบียนคำขอจัดซื้อจัดจ้าง และทะเบียนพัสดุ ซึ่งทำให้เกิดปัญหาซ้ำ ๆ คือข้อมูลไม่เป็นปัจจุบัน มีหลายเวอร์ชัน ค้นหายาก ไม่มีการควบคุมสิทธิ์การเข้าถึง และทำรายงานได้ช้า ขณะเดียวกันการพัฒนาระบบสารสนเทศแบบเดิมต้องพึ่งนักพัฒนา ใช้เวลาและงบประมาณสูง จนหลายงานเลือกอยู่กับ Excel ต่อไป เครื่องมือ AI ช่วยเขียนโค้ดในปัจจุบันเปลี่ยนสมการนี้ ทำให้เจ้าหน้าที่สำนักงานที่ไม่มีพื้นฐานการเขียนโปรแกรม สามารถสร้างเว็บแอปพลิเคชันที่ใช้งานได้จริงและเชื่อมฐานข้อมูลจริงได้ด้วยตนเอง โจทย์จึงเปลี่ยนจาก “จะจ้างใครทำ” มาเป็น “จะทำให้เจ้าหน้าที่ของเราทำเองได้อย่างไร”

สภาพปัญหาที่พบจากการปฏิบัติงานจริง

ประเด็นปัญหา	ลักษณะที่เกิดขึ้น	ผลกระทบ
1. ไม่รู้ว่าต้องเริ่มจากอะไร	ติดตั้งเครื่องมือไม่ครบ หรือเปิดโปรแกรมแล้วไม่รู้จะพิมพ์อะไรลงช่องแชท	ล้มเลิกตั้งแต่ชั่วโมงแรก
2. กลัวว่าต้องเขียนโค้ดเป็นก่อน	คิดว่าเรื่องนี้เป็นงานของสายไอทีเท่านั้น	ไม่กล้าเริ่ม ทั้งที่งานของตัวเองรู้ดีที่สุด
3. เขียน Requirement ไม่เป็น	บอก AI ว่า “ทำเว็บให้หน่อย” โดยไม่ระบุว่าเว็บเกี่ยวกับอะไร มี role อะไร ใครเข้าถึงหน้าไหนได้	ได้ผลลัพธ์ไม่ตรงความต้องการ ต้องรื้อใหม่หลายรอบ
4. รีบเชื่อมฐานข้อมูลตั้งแต่แรก	สร้าง UI กับฐานข้อมูลพร้อมกัน	เกิด error พร้อมกันหลายจุด แก้ไม่ถูกจุด
5. ไม่เข้าใจว่า API Keys คืออะไร	สับสนระหว่าง Project URL, Anon public และ Service Role หยิบผิดตัว	เว็บเชื่อมฐานข้อมูลไม่ได้
6. เจอ error แล้วหยุด	เห็นข้อความสีแดงแล้วคิดว่าทำผิดจนแก้ไม่ได้	เลิกกลางคัน ทั้งที่วิธีแก้คือคัดลอก error ไปวางในช่องแชทให้ AI แก้ให้
7. ทำเสร็จแล้วใช้ได้คนเดียว	เว็บรันอยู่ที่ localhost:3000 บนเครื่องตัวเอง	เพื่อนร่วมงานใช้ไม่ได้ ไม่เกิดประโยชน์กับสำนักงาน

ประเด็นปัญหา	ลักษณะที่เกิดขึ้น	ผลกระทบ
8. ไม่มีที่เก็บโค้ด	แก้ไขทับไปเรื่อย ๆ ไม่มีสำเนา	เครื่องเสียแล้วงานหาย ย้อนกลับไม่ได้
9. ความเสี่ยงด้านความปลอดภัยและข้อมูลส่วนบุคคล	นำ Service Role Key หรือข้อมูลจริงของบุคลากรขึ้นพื้นที่สาธารณะโดยไม่ตั้งใจ	ข้อมูลรั่วไหล และอาจขัดต่อกฎหมายคุ้มครองข้อมูลส่วนบุคคล
10. ความรู้อยู่กับคนเดียว	ผู้ที่ไปอบรมกลับมาทำคนเดียว ไม่มีการถ่ายทอด	เมื่อผู้นั้นลาหรือย้ายงาน ระบบไม่มีคนดูแลต่อ

ทักษะนี้ไม่ใช่การเปลี่ยนเจ้าหน้าที่สำนักงานให้เป็นโปรแกรมเมอร์ แต่เป็นการทำให้ คนที่รู้กระบวนงานดีที่สุดสามารถสร้างเครื่องมือของตัวเองได้ ซึ่งตรงกับทิศทางการเปลี่ยนผ่านสู่มหาวิทยาลัยดิจิทัลและการลดขั้นตอนการทำงาน โดยใช้ทรัพยากรน้อยกว่าการพัฒนาระบบแบบเดิมมาก

3. กลุ่มเป้าหมายและวิธีการถอดบทเรียน

กลุ่มหลัก เจ้าหน้าที่สำนักงาน คณะรัฐศาสตร์ (บุคลากรสายสนับสนุนวิชาการทุกงาน)
กลุ่มร่วมแลกเปลี่ยน งานการเงิน พัสดุ บุคคล แผนและงบประมาณ
ซึ่งเป็นเจ้าของข้อมูลและกระบวนงาน
กลุ่มผู้ให้ความรู้ วิทยากรผู้สอน และเจ้าหน้าที่ที่ผ่านการอบรมรุ่นแรก
กลุ่มผู้ใช้ประโยชน์ต่อ ผู้บริหารคณะ คณาจารย์

วิธีการถอดบทเรียน

1. เรียนรู้จากการลงมือทำจริง (Learning by Doing) อบรมเชิงปฏิบัติการ 2 ครั้ง ครั้งละประมาณ 2 ชั่วโมงครึ่ง ผ่านระบบประชุมออนไลน์ พร้อมบันทึกวิดีโอย้อนหลังให้ทบทวน
2. ใช้โจทย์จริงของสำนักงานเป็นโปรเจกต์ ไม่ใช้ตัวอย่างสมมติ แต่ให้ผู้เรียนนำงานที่ตนรับผิดชอบมาเป็นระบบที่จะสร้าง เช่น ระบบบริหารจัดการคำขอการจัดซื้อจัดจ้างและการบริหารพัสดุ หรือระบบทะเบียนลูกหนี้ที่แปลงจากไฟล์ Excel เดิม
3. ตอบแบบสอบถามเตรียมความพร้อมก่อนเรียน กำหนดให้ผู้เรียนตอบคำถาม 4 ข้อล่วงหน้า เพื่อให้มี Requirement ตั้งต้นก่อนเข้าชั้นเรียน
4. ถอดบทเรียนจากอุปสรรคระหว่างทำ บันทึกทุกจุดที่ผู้เรียนติดขัด พร้อมวิธีแก้ที่ได้ผล แล้วรวบรวมเป็นคลัง Troubleshooting
5. รวบรวมคลัง Prompt ที่ใช้ได้ผล เก็บ Prompt ที่ให้ผลลัพธ์ดีไว้เป็นชุดมาตรฐาน เพื่อให้คนต่อไปไม่ต้องลองผิดลองถูกใหม่
6. สรุปเป็นคู่มือทีละขั้น เรียบเรียงเป็นคู่มือที่คนไม่มีพื้นฐานเขียนโค้ดอ่านแล้วทำตามได้เอง
7. ขยายผลสู่เพื่อนร่วมงาน ผู้ผ่านการอบรมเป็นพี่เลี้ยงให้เจ้าหน้าที่คนอื่นในรุ่นถัดไป

ระยะเวลา ดำเนินการในปีงบประมาณ พ.ศ. 2569

4. องค์ความรู้ที่ได้จากการแลกเปลี่ยนเรียนรู้

(1) ภาพรวมเส้นทางการสร้างเว็บแอปพลิเคชัน 8 ขั้น

> 1. เตรียมเครื่องมือ (Node.js + VS Code + Codex + skill impeccable)
> ↓
> 2. เขียน Requirement ตั้งต้น (เว็บเกี่ยวกับอะไร / role / สิทธิ์ / โทนสี)
> ↓
> 3. ให้ AI แปลง Requirement เป็นเอกสาร PRD.md
> ↓
> 4. สร้างโปรเจกต์และหน้าเว็บ (UI) ก่อน — ยังไม่เชื่อมฐานข้อมูล
> ↓
> 5. ให้ AI ออกแบบฐานข้อมูลเป็นเอกสาร database.md แล้วตรวจก่อน
> ↓
> 6. เชื่อมเว็บกับ Supabase + นำไฟล์ .sql ไป run สร้างตารางจริง
> ↓
> 7. เก็บโค้ดขึ้น GitHub
> ↓
> 8. นำขึ้นระบบ (Deploy) ที่ Vercel ⟶ ได้ลิงก์ที่ส่งต่อให้คนอื่นใช้ได้

หลักคิดสำคัญที่สุดของทั้งกระบวนการ ทำทีละชั้น อย่าลัดขั้นตอน — ออกแบบก่อน ⟶ สร้างหน้าเว็บก่อน ⟶ ค่อยต่อฐานข้อมูล ⟶ ค่อยนำขึ้นระบบ การข้ามขั้นทำให้เกิดปัญหาพร้อมกันหลายจุดจนหาสาเหตุไม่เจอ

(2) เครื่องมือที่ต้องเตรียม

เครื่องมือ	หน้าที่	หมายเหตุการติดตั้ง
Node.js	ตัวรันโปรแกรมเบื้องหลัง	ดาวน์โหลดจาก nodejs.org กด next จนเสร็จ
VS Code	โปรแกรมที่ใช้ทำงานและคุยกับ AI	ดาวน์โหลดจาก code.visualstudio.com กด next จนเสร็จ
ส่วนขยาย Codex	ตัว AI ที่จะเขียนโค้ดให้	เปิด VS Code ⟶ เมนู Extension (ไอคอนสี่เหลี่ยมด้านซ้าย) ⟶ ค้นหา Codex ⟶ เลือกอันแรก ⟶ Install ⟶ Trust Publisher and Install

เครื่องมือ	หน้าที่	หมายเหตุการติดตั้ง
skill impeccable	ตัวช่วยออกแบบหน้าตาเว็บให้สวยและเป็นระบบ	สั่งผ่านช่องแชท: *"ช่วยติดตั้ง skill impeccable ให้หน่อย"* พร้อมแนบลิงก์เอกสารของ impeccable
บัญชี ChatGPT	บัญชีที่ใช้กับ Codex	ผู้เรียนในรอบนี้ใช้แบบเสียเงิน
Supabase	ฐานข้อมูลจริงบนคลาวด์	สมัครและสร้าง Project ไว้ล่วงหน้า
GitHub	ที่เก็บโค้ดและประวัติการแก้ไข	สมัครบัญชีไว้ล่วงหน้า
Vercel	ที่นำเว็บขึ้นระบบให้คนอื่นเข้าใช้ได้	สมัครด้วยบัญชี Google แล้วเชื่อมกับ GitHub

ขั้นเตรียมพื้นที่ทำงาน สร้างโฟลเดอร์สำหรับเก็บไฟล์โค้ด (เช่น ที่หน้า Desktop) ⟶ เปิด VS Code เลือก Open Folder แล้วเลือกโฟลเดอร์นั้น ⟶ คลิกไอคอน ChatGPT ด้านบนซ้ายเพื่อเปิดช่องแชท

(3) การเขียน Requirement ตั้งต้น 4 คำถามที่ต้องตอบให้ได้ก่อน

ก่อนพิมพ์อะไรให้ AI ต้องตอบคำถาม 4 ข้อนี้ให้ชัดก่อน เพราะเป็นวัตถุดิบของทุกขั้นตอนถัดไป

คำถาม	ตัวอย่างคำตอบจากผู้เรียนจริง
1. เว็บเกี่ยวกับอะไร	ระบบบริหารจัดการคำขอการจัดซื้อจัดจ้างและการบริหารพัสดุ
2. อยากมี role อะไรบ้าง	admin, user, executive, staff
3. แต่ละ role เข้าถึงหน้าไหนได้บ้าง	admin เข้าได้ทุกหน้า / user บางหน้า / executive บางหน้า / staff เฉพาะข้อมูลของตนเอง
4. อยากได้เว็บโทนสีแบบไหน	โทนสีส้ม

บทเรียน คำถามข้อ 2 และ 3 คือหัวใจของงานสำนักงาน เพราะเป็นเรื่อง "ใครเห็นอะไรได้บ้าง" ถ้าตอบข้อนี้ไม่ชัดตั้งแต่ต้น ระบบที่ได้จะควบคุมสิทธิ์ไม่ได้ และต้องรื้อทำใหม่ทั้งหมด

(4) ขั้นออกแบบ: จาก Requirement สู่ PRD.md

Prompt ที่ใช้

> วิเคราะห์จาก requirement เบื้องต้นออกมาให้เป็น PRD.md อย่างละเอียด รายละเอียด requirement เบื้องต้น: <คำตอบ 4 ข้อข้างต้น>

PRD (Product Requirement Document) คือเอกสารสรุปว่าระบบต้องทำอะไรได้บ้าง เปรียบได้กับ "แบบแปลนบ้าน" ก่อนลงมือก่อสร้าง ให้อ่านตรวจก่อนไปขั้นถัดไป ถ้ามีอะไรไม่ตรง ให้พิมพ์บอก AI แก้ในช่องแชทได้เลย

(5) ขั้นสร้างหน้าเว็บ (UI) — ยังไม่เชื่อมฐานข้อมูล

Prompt ที่ใช้

> ช่วยสร้างโปรเจคตามเอกสาร PRD.md ด้วย tech stack ดังนี้ - Next.js 16 app router - typescript - tailwind - supabase
>
> ซึ่งเริ่มต้นในฝั่งของ UI ก่อน ยังไม่ต้องทำการเชื่อมกับ supabase การดีไซน์หน้าเว็บหรือ ui ทั้งหมดให้ใช้ skill impeccable เข้ามาช่วยดีไซน์

เมื่อ AI ทำงานเสร็จ จะได้ลิงก์ **http://localhost:3000** ซึ่งเป็นเว็บที่รันอยู่บนเครื่องของเราเอง

ถ้าเปิดไม่ได้: พิมพ์ในช่องแชทว่า *"รันโปรเจคให้หน่อย"*

ศัพท์ที่ควรรู้แบบง่าย ๆ

ศัพท์	อธิบายอย่างง่าย
Next.js	โครงสร้างสำเร็จรูปสำหรับสร้างเว็บ
TypeScript	ภาษาเขียนโค้ดที่ช่วยกันความผิดพลาดเรื่องชนิดข้อมูล
Tailwind	ชุดคำสั่งจัดหน้าตาเว็บให้สวยเร็ว
UI	หน้าตาเว็บที่ผู้ใช้เห็นและกดใช้งาน
localhost:3000	เว็บที่เปิดอยู่บนเครื่องเราเท่านั้น คนอื่นยังเข้าไม่ได้

(6) ขั้นออกแบบฐานข้อมูล: database.md

Prompt พื้นฐาน

> ช่วยออกแบบฐานข้อมูล โดยวิเคราะห์จากฟีเจอร์และ requirement ทั้งหมดของเว็บ เป็นไฟล์ database.md

กรณีมีไฟล์ Excel เดิมอยู่แล้ว (สำคัญมากสำหรับงานสำนักงาน) — ให้นำไฟล์ Excel ไปวางในโฟลเดอร์โปรเจกต์ก่อน แล้วจึงใช้ Prompt

> วิเคราะห์ฟีเจอร์และ Requirement ทั้งหมดของเว็บไซต์ โดยใช้ไฟล์ Excel ชื่อ <ชื่อไฟล์> เป็นข้อมูลอ้างอิงหลัก ให้ทำงานตามลำดับนี้: 1. วิเคราะห์ข้อมูลจากไฟล์ Excel และฟีเจอร์ของเว็บไซต์ 2. ออกแบบฐานข้อมูลให้รองรับระบบจริง 3. สร้างไฟล์ database.md เพื่อสรุปโครงสร้างฐานข้อมูล ได้แก่ ตาราง ฟิลด์ ชนิดข้อมูล และความสัมพันธ์ 4. ตรวจสอบว่าข้อมูลใน Excel สอดคล้องกับฐานข้อมูลและหน้าเว็บหรือไม่ 5. หากข้อมูลไม่ตรงหรือไม่เพียงพอ ให้เสนอรายการข้อมูลที่ควรแก้ไข เพิ่มเติม หรือตัดออก 6. หลังจากออกแบบฐานข้อมูลและสร้าง database.md เสร็จแล้ว ค่อยแก้ไขหน้าเว็บให้สอดคล้องกับข้อมูลจริงและโครงสร้างฐานข้อมูลใหม่
>
> หมายเหตุ: ให้ยึดข้อมูลเดิมใน Excel และข้อมูลที่เกี่ยวข้องเป็นหลัก

นี่คือจุดที่ทรงคุณค่าที่สุดสำหรับสำนักงาน เพราะเป็นการยกทะเบียนที่ทำใน Excel มาหลายปี ขึ้นเป็นระบบจริง โดยไม่ทิ้งข้อมูลเดิม

ต้องอ่าน database.md ก่อนไปต่อเสมอ ถ้ายังขาดอะไรให้พิมพ์บอกเพิ่ม เช่น *"ฉันต้องการเก็บข้อมูลของ user เพิ่มเติม เก็บเบอร์โทรศัพท์ ที่อยู่ เพิ่มเติมด้วย"*

(7) ขั้นเชื่อมฐานข้อมูลจริงด้วย Supabase

7.1 สั่งให้ AI เริ่มเชื่อม

> เริ่มเชื่อมต่อหน้าเว็บกับ supabase โดยอ้างอิงการออกแบบฐานข้อมูลจากไฟล์ database.md ฉันจะนำ API Key ทั้งหมดมาใส่ภายหลัง

7.2 ระหว่างรอ AI ทำงาน ให้ไปเตรียม API Keys

API Keys เปรียบเสมือนรหัสผ่านหรือบัตรประจำตัวของฐานข้อมูล ที่ทำให้เว็บของเราเข้าถึงข้อมูลได้
ต้องใช้ทั้งหมด **3 ค่า**

ลำดับ	ค่าที่ต้องใช้	หาได้จากที่ไหนใน Supabase
1	**Project URL**	หน้า Dashboard ⟶ กดปุ่ม copy ⟶ เลือกอันแรก Project URL
2	**Anon public**	Project Settings (ไอคอนฟันเฟืองล่างสุด) ⟶ เมนู API Keys ⟶ เลือกแท็บ Legacy anon ⟶ กด copy ที่ Anon public
3	**Service Role**	หน้าเดียวกัน ⟶ กด reveal ที่ Service Role ⟶ กด copy

วิธีส่งให้ AI พิมพ์ในช่องแชทของ Codex ทีละบรรทัด โดยกด **Shift + Enter**
เพื่อขึ้นบรรทัดใหม่ระหว่างค่าแต่ละตัว แล้วกด Enter ครั้งเดียวเมื่อครบทั้ง 3 ค่า

7.3 นำไฟล์ .sql ไปสร้างตารางจริง

เมื่อ AI ทำงานเสร็จ จะได้ไฟล์นามสกุล **.sql** (มักอยู่ในโฟลเดอร์ supabase หรือ database) ให้ทำตามนี้

1. คัดลอกโค้ดในไฟล์ .sql จาก VS Code
2. เปิด Supabase เลือกเมนู **SQL Editor**
3. วางโค้ดลงในช่องว่าง
4. กดปุ่ม **Run** ด้านล่าง
5. ลองเข้า http://localhost:3000 แล้วทดสอบเพิ่ม ลบ แก้ไขข้อมูลจริงในทุกหน้า

กรณีไม่ได้ไฟล์ .sql มา ให้ใช้ Prompt

> ตรวจสอบโค้ดให้ทำการเชื่อมต่อ supabase ให้เรียบร้อย ลบ mock data ทั้งหมด
> รวมถึงการสร้างตารางหรือการเก็บข้อมูลเชื่อมกับ supabase เป็นไฟล์ .sql เพื่อให้ฉันนำไปเชื่อมกับ supabase ได้

(8) ขั้นเก็บโค้ดขึ้น GitHub

1. เข้า github.com ⟶ กดปุ่ม **New** ⟶ กรอกชื่อในช่อง Repository name
2. เลือกความเป็นส่วนตัว: **Public** = สาธารณะ ใครก็เข้าดูได้ / **Private** = เห็นเฉพาะเราเท่านั้น
3. กดสร้าง แล้วกดไอคอน copy ด้านขวาเพื่อคัดลอก URL
4. กลับมาที่ VS Code พิมพ์ในช่องแชท

> ช่วยล็อกอิน github ให้หน่อย แบบ auth ผ่าน browser username: Email: <email ที่ล็อกอิน github> และช่วยนำโค้ดขึ้น github ให้หน่อย

ครั้งแรกจะมีหน้าต่างเล็ก ๆ ให้กด login ในเบราว์เซอร์ กด confirm หรือ authorize จนเสร็จ แล้วรีเฟรชหน้า GitHub จะเห็นไฟล์โค้ดทั้งหมด

(9) ขั้นนำขึ้นระบบ (Deploy) ที่ Vercel

สมัครครั้งแรก Sign up with Google ⟶ เข้าหน้า Dashboard ⟶ กดปุ่มดำ Continue with GitHub ⟶ เลื่อนลงมากด Install

นำโปรเจกต์ขึ้นระบบ

1. เลือกโปรเจกต์จากรายการ GitHub แล้วกด **Import**
2. ตั้งชื่อในช่อง Project name
3. เลื่อนลงมาที่เมนู **Environment Variable** แล้วคลิกเปิด
4. กลับไปที่ VS Code หาไฟล์ **.env** หรือ **.env.local** คัดลอกโค้ดทั้งหมด
5. นำมาวางที่ช่อง Key ช่องแรก ระบบจะแตกเป็นหลายค่าให้เอง
6. กด **Deploy** ⟶ รอจนขึ้น Congratulation ⟶ กด Continue to dashboard
7. ในหน้า Dashboard จะมีคำว่า **domain** พร้อมลิงก์ด้านล่าง —
   นั่นคือลิงก์เว็บของเราที่ส่งต่อให้คนอื่นใช้งานได้ทันที

เมื่อมีการแก้ไขในอนาคต

- ถ้ามีไฟล์ .sql เพิ่ม ให้นำไป run ที่ Supabase ⟶ SQL Editor อีกครั้ง
- บอก AI ใน VS Code ให้นำโค้ดขึ้น GitHub
- Vercel จะ deploy ให้อัตโนมัติ รอประมาณ 2–3 นาที แล้วลองเข้าเว็บใหม่

(10) คลัง Prompt ที่ใช้ได้ผล

วัตถุประสงค์	Prompt
ตรวจคุณภาพโครงสร้างโค้ด	ช่วยตรวจสอบโครงสร้างและคุณภาพโค้ดของโปรเจกต์นี้ตาม best practice ของเทคโนโลยีที่ใช้ ตรวจเรื่องโครงสร้างไฟล์/โฟลเดอร์ การแยก routes, components, logic, data, types, hooks, utils, styles / ไฟล์ที่ใหญ่เกินไป / การใช้ any และ hardcoded data / การตั้งชื่อ / ความอ่านง่ายและขยายต่อได้ / การทำตาม convention — ให้ตอบเป็น 1. สรุปภาพรวม 2. ปัญหาที่พบ 3. แนวทางที่แนะนำ 4. แผนปรับปรุงแบบเป็นขั้นตอน โดยอธิบายแบบเข้าใจง่ายสำหรับผู้ไม่มีความรู้ด้านโค้ด **ห้ามแก้ไฟล์ก่อน ให้เสนอแผนและรออนุมัติ**
หาและสร้างหน้าที่ยังขาด	ช่วยตรวจจากดีไซน์และ flow ของเว็บทั้งหมด ว่ายังขาดหน้าไหนหรือฟังก์ชันไหนบ้าง แล้วช่วยสร้างให้ครบ โดยให้ดีไซน์เข้ากับของเดิม และอย่าแก้ส่วนที่ทำเสร็จแล้วถ้าไม่จำเป็น
สร้างหน้าต่างเพิ่ม–แก้ไข–ลบข้อมูล	สร้าง popup สำหรับ crud ข้อมูลในทุกหน้า / สร้าง page ใหม่สำหรับ crud ข้อมูลของปุ่ม... ในหน้า...
เปลี่ยนแบบอักษร	ช่วยเปลี่ยน font เป็น ... ของ google font ทั้งโปรเจค

ข้อสังเกตจากวง CoP Prompt ตรวจคุณภาพโค้ดข้างต้นมีสองประโยคที่ควรติดเป็นนิสัย คือ *“อธิบายแบบเข้าใจง่าย สำหรับผู้ที่ไม่มีความรู้ในการเขียนโค้ด”* และ *“ห้ามแก้ไฟล์ก่อนให้เสนอแผนและรออนุมัติก่อน”* ประโยคหลังทำให้เรายังเป็นผู้ตัดสินใจ ไม่ใช่ปล่อยให้ AI แก้ทั้งโปรเจกต์โดยเราไม่รู้ว่าเปลี่ยนอะไรไปบ้าง

(11) คลัง Troubleshooting เจอปัญหาแล้วทำอย่างไร

อาการ	วิธีแก้
เปิด localhost:3000 ไม่ได้	พิมพ์ในช่องแชท: *“รันโปรเจคให้หน่อย”* แล้วรอจนขึ้นลิงก์ใหม่
นำโค้ด .sql ไป run ที่ Supabase แล้วขึ้น error	คัดลอก error ให้หมด ⟶ วางในช่องแชท Codex ⟶ กด Enter ⟶ รอ AI แก้ ⟶ คัดลอกโค้ดในไฟล์ .sql ใหม่ทั้งหมด ⟶ ลบโค้ดเก่าใน SQL Editor ⟶ วางใหม่ ⟶ กด Run
ไม่ได้ไฟล์ .sql	ใช้ Prompt ตรวจสอบการเชื่อมต่อและสั่งสร้างไฟล์ .sql (ดูข้อ 7.3)
ผลลัพธ์ไม่ตรงกับที่ต้องการ	พิมพ์บอกสิ่งที่อยากแก้เป็นภาษาคนธรรมดาลงในช่องแชทได้เลย ไม่ต้องแก้โค้ดเอง

อาการ	วิธีแก้
เว็บยังใช้ข้อมูลตัวอย่าง (mock data)	สั่งให้ลบ mock data ทั้งหมดและเชื่อมกับข้อมูลจริง

หลักที่ได้จากวง CoP ข้อความ error สีแดงไม่ใช่สัญญาณว่าทำผิดจนต้องเริ่มใหม่ แต่เป็น “ข้อมูลนำเข้า” ที่ดีที่สุดสำหรับ AI คัดลอกทั้งก้อนไปวางในช่องแชทคือวิธีแก้ที่ได้ผลที่สุด

5. แนวปฏิบัติที่ดี (Good Practice) และขั้นตอนการปฏิบัติงาน

แนวปฏิบัติที่ดีที่ 1 เตรียม Requirement ให้เสร็จก่อนเปิดคอมพิวเตอร์

ใช้แบบฟอร์มสั้น ๆ ให้เจ้าหน้าที่กรอกก่อนเริ่มโปรเจกต์ทุกครั้ง

หัวข้อ	คำตอบ
ระบบนี้แก้ปัญหาอะไรของสำนักงาน	
ปัจจุบันทำงานนี้ด้วยอะไร (Excel / กระดาษ / ระบบเดิม)	
มี role อะไรบ้าง	
แต่ละ role เห็นและทำอะไรได้บ้าง	
ข้อมูลที่ต้องเก็บมีอะไรบ้าง (ดูจากหัวคอลัมน์ในไฟล์เดิม)	
รายงานที่ต้องออกมีอะไรบ้าง	
โทนสีและรูปแบบที่ต้องการ	

เคล็ดลับ หัวคอลัมน์ในไฟล์ Excel เดิมคือ Requirement ที่ดีที่สุดที่มีอยู่แล้ว เพราะสะท้อนข้อมูลที่หน่วยงานใช้จริงมาหลายปี

แนวปฏิบัติที่ดีที่ 2 ทำทีละชั้น และตรวจก่อนไปต่อทุกครั้ง (Checkpoint)

ชั้นงาน	สิ่งที่ต้องตรวจก่อนไปขั้นถัดไป
PRD.md	ฟีเจอร์ครบไหม / role และสิทธิ์ตรงกับที่ต้องการไหม
หน้าเว็บ (UI)	กดใช้งานได้ทุกหน้าไหม / หน้าตาใช้งานง่ายไหม
database.md	เก็บข้อมูลครบไหม / ตรงกับไฟล์ Excel เดิมไหม
เชื่อม Supabase	เพิ่ม ลบ แก้ไขข้อมูลได้จริงทุกหน้าไหม
GitHub	โค้ดขึ้นครบไหม / ตั้งค่า Private ถูกต้องไหม
Vercel	เปิดจากเครื่องอื่นหรือมือถือได้ไหม

การมี Checkpoint ทำให้เมื่อเกิดปัญหา เรารู้ทันทีว่าปัญหาอยู่ชั้นไหน ไม่ต้องรื้อทั้งระบบ

แนวปฏิบัติที่ดีที่ 3 มาตรฐานความปลอดภัยขั้นต่ำของสำนักงาน

เรื่อง	สิ่งที่ต้องทำ
Service Role Key	ถือเป็นกุญแจหลักของฐานข้อมูล ห้ามเผยแพร่ ห้ามใส่ในหน้าเว็บฝั่งผู้ใช้ และห้ามส่งในกลุ่มแชททั่วไป
ไฟล์ .env / .env.local	ต้องไม่ถูกอัปขึ้น GitHub ให้ตรวจสอบทุกครั้งก่อนนำโค้ดขึ้น และใส่ค่าเหล่านี้ผ่านเมนู Environment Variable ของ Vercel แทน
การตั้งค่า Repository	ระบบที่มีข้อมูลของหน่วยงานควรตั้งเป็น **Private** เสมอ
ข้อมูลทดสอบ	ช่วงพัฒนาให้ใช้ข้อมูลสมมติ อย่านำข้อมูลจริงของบุคลากรหรือนักศึกษามาทดลอง
ข้อมูลส่วนบุคคล	ก่อนนำระบบขึ้นใช้จริง ต้องพิจารณาความสอดคล้องกับกฎหมายคุ้มครองข้อมูลส่วนบุคคล และหารือผู้รับผิดชอบด้านข้อมูลของคณะ
การควบคุมสิทธิ์	ตรวจว่า role แต่ละระดับเห็นข้อมูลเท่าที่ควรเห็นจริง โดยเฉพาะ staff ที่ควรเห็นเฉพาะข้อมูลของตนเอง

แนวปฏิบัติที่ดีที่ 4 ทำงานกับ AI อย่างเป็นผู้ควบคุม ไม่ใช่ผู้ตาม

1. สั่งให้เสนอแผนก่อนลงมือแก้ ใช้ประโยค *“ห้ามแก้ไฟล์ก่อน ให้เสนอแผนและรออนุมัติก่อน”* ในงานที่กระทบหลายส่วน
2. สั่งให้อธิบายเป็นภาษาคนธรรมดา เติมท้าย Prompt ว่าให้อธิบายแบบเข้าใจง่าย พร้อมขยายความศัพท์เทคนิค
3. แก้ทีละเรื่อง พิมพ์สิ่งที่อยากแก้ทีละประเด็น ดีกว่ารวมสิบเรื่องในข้อความเดียว
4. ทดสอบทุกครั้งหลัง AI แก้ โดยเฉพาะหน้าที่ไม่ได้สั่งให้แก้ เพราะอาจกระทบกันได้
5. เก็บ Prompt ที่ได้ผลไว้ สะสมเป็นคลังของคณะ เพื่อไม่ต้องคิดใหม่ทุกครั้ง

แนวปฏิบัติที่ดีที่ 5 เกณฑ์เลือกงานที่เหมาะจะทำเป็นระบบก่อน

ไม่ใช่ทุกงานควรทำเป็นเว็บแอป ให้เลือกงานที่เข้าเกณฑ์ต่อไปนี้ก่อน

- ทำซ้ำเป็นประจำและมีขั้นตอนแน่นอน
- ปัจจุบันใช้ Excel หลายไฟล์หรือหลายเวอร์ชัน
- มีคนหลายคนต้องเข้าถึงข้อมูลชุดเดียวกัน
- ต้องออกรายงานหรือสรุปยอดเป็นประจำ
- มีเรื่องสิทธิ์การเข้าถึงที่ต่างกันตามบทบาท

ตัวอย่างงานที่เข้าเกณฑ์ในคณะ ทะเบียนคำขอจัดซื้อจัดจ้างและพัสดุ ทะเบียนลูกหนี้เงินยืม ทะเบียนคำสั่งมอบอำนาจ ระบบขออนุมัติเดินทางไปปฏิบัติงาน และระบบจองห้องประชุม

แนวปฏิบัติที่ดีที่ 6 เอกสารประกอบระบบที่ต้องมีทุกโปรเจกต์

เอกสาร	ใช้ทำอะไร
PRD.md	บอกว่าระบบทำอะไรได้บ้าง (AI สร้างให้)
database.md	บอกว่าเก็บข้อมูลอะไรบ้างและสัมพันธ์กันอย่างไร (AI สร้างให้)
คู่มือผู้ใช้อย่างย่อ	ให้เพื่อนร่วมงานใช้เป็น โดยไม่ต้องถามผู้สร้าง
บันทึกผู้ดูแลระบบ	ระบุผู้ดูแลหลัก ผู้ดูแลสำรอง บัญชี Supabase/GitHub/Vercel ที่ใช้ และวิธีเข้าถึง

เอกสารสองรายการหลังคือสิ่งที่ป้องกันไม่ให้ระบบกลายเป็นของคนเดียว

6. ปัจจัยความสำเร็จและข้อควรระวัง

ปัจจัยความสำเร็จ (Key Success Factors)

1. ใช้โจทย์งานจริงของตัวเองเป็นโปรเจกต์ ผู้เรียนมีแรงจูงใจสูงกว่าการทำตัวอย่างสมมติมาก และได้ผลงานที่ใช้ได้จริงตั้งแต่วันแรก
2. ผู้บริหารคณะสนับสนุนเวลาและอุปกรณ์ จัดสรรเวลาให้ฝึกได้จริง และสนับสนุนค่าบริการที่จำเป็น เช่น บัญชี ChatGPT แบบเสียเงิน
3. เรียนแบบลงมือทำพร้อมกันและมีวิดีโอย้อนหลัง คนที่ตามไม่ทันในห้องเรียนกลับไปทบทวนเองได้
4. เตรียม Requirement มาก่อนเข้าเรียน ทำให้เวลาในชั้นเรียนใช้กับการลงมือทำ ไม่ใช่การคิดโจทย์
5. ทำทีละชั้นและตรวจก่อนไปต่อ ลดปัญหาสะสมจนแก้ไม่ไหว
6. มีคลัง Prompt และคลัง Troubleshooting ของคณะ คนรุ่นถัดไปไม่ต้องเริ่มจากศูนย์
7. มีระบบพี่เลี้ยง ผู้ผ่านการอบรมรุ่นแรกช่วยดูแลรุ่นถัดไป ทำให้ความรู้ขยายตัวเอง

ข้อควรระวัง (Risk & Caution)

1. อย่าลัดขั้นตอน การเชื่อมฐานข้อมูลตั้งแต่ยังไม่เสร็จ UI
   ทำให้เกิดปัญหาพร้อมกันหลายจุดจนหาสาเหตุไม่เจอ
2. อย่าเผยแพร่ Service Role Key และไฟล์ .env
   เป็นความเสี่ยงด้านความปลอดภัยที่ร้ายแรงที่สุดของกระบวนการนี้
3. อย่าใช้ข้อมูลจริงของบุคคลในการทดสอบ
   ให้ใช้ข้อมูลสมมติจนกว่าระบบจะพร้อมและผ่านการพิจารณาเรื่องข้อมูลส่วนบุคคล
4. อย่าเชื่อผลลัพธ์ของ AI โดยไม่ตรวจ โดยเฉพาะการออกแบบฐานข้อมูล ต้องอ่าน database.md ให้เข้าใจก่อนสร้างตารางจริง เพราะแก้ทีหลังมีต้นทุนสูง
5. ระวังระบบกลายเป็นของคนเดียว ต้องมีผู้ดูแลสำรองและเอกสารส่งมอบตั้งแต่ต้น
6. ระวังการทำระบบซ้ำซ้อนกับระบบกลางของมหาวิทยาลัย ก่อนเริ่ม
   ควรตรวจสอบว่ามหาวิทยาลัยมีระบบกลางที่รองรับงานนั้นอยู่แล้วหรือไม่
   และหารือกับหน่วยงานที่รับผิดชอบด้านเทคโนโลยีสารสนเทศ

7. ระวังเรื่องความต่อเนื่องของบริการภายนอก Supabase, Vercel และ GitHub
   เป็นบริการภายนอกที่มีข้อจำกัดของแพ็กเกจฟรี
   ควรวางแผนสำรองข้อมูลและงบประมาณหากระบบขยายตัว
8. ระวังการนำระบบขึ้นใช้จริงโดยไม่ผ่านการเห็นชอบ
   ระบบที่เก็บข้อมูลของหน่วยงานควรได้รับความเห็นชอบจากผู้บริหารคณะก่อนเปิดใช้จริง
9. ระวังการติดกับดักความสมบูรณ์แบบ ระบบที่ใช้งานได้ 80%
   วันนี้มีประโยชน์กว่าระบบสมบูรณ์แบบที่ไม่เสร็จสักที ให้ทยอยปรับปรุงจากการใช้งานจริง
10. เวอร์ชันของเครื่องมือเปลี่ยนเร็ว ชื่อเมนู ตำแหน่งปุ่ม และขั้นตอนบางอย่างของ Supabase, Vercel หรือ VS Code อาจเปลี่ยนไปจากคู่มือ ต้องปรับปรุงคู่มือเป็นระยะ

7. การถ่ายทอดองค์ความรู้

กิจกรรม	รูปแบบ	กลุ่มเป้าหมาย	ผลผลิต
1. เวที CoP งานสำนักงาน	ประชุมแลกเปลี่ยนเรียนรู้ ครั้งละ 2–3 ชั่วโมง	เจ้าหน้าที่สำนักงานทุกงาน	สรุปประเด็นและบทเรียนร่วม
2. อบรมเชิงปฏิบัติการรุ่นต่อไป	ลงมือทำจริง 2 ครั้ง ครั้งละ 2 ชั่วโมงครึ่ง	เจ้าหน้าที่ที่ยังไม่เคยอบรม	เว็บแอปคนละ 1 ระบบ
3. คู่มือทีละขั้นฉบับคนไม่เขียนโค้ด	เอกสาร/ไฟล์ PDF พร้อมภาพประกอบ	เจ้าหน้าที่คณะ	คู่มือ 8 ขั้นตอน
4. คลัง Prompt ของคณะ	ไฟล์กลางที่เพิ่มเติมได้ตลอด	ผู้ใช้งานทุกคน	Prompt ที่ทดสอบแล้วว่าได้ผล
5. คลัง Troubleshooting	ตาราง “อาการ–วิธีแก้”	ผู้ใช้งานทุกคน	ลดเวลาติดขัด
6. วิดีโอบันทึกการสอนย้อนหลัง	ไฟล์วิดีโอในคลังความรู้ของคณะ	ทุกคน	ทบทวนได้ตลอดเวลา
7. คลินิก AI ประจำสัปดาห์	ให้คำปรึกษารายเรื่องตามนัดหมาย	ผู้ที่กำลังทำโปรเจกต์	แก้ปัญหาเฉพาะหน้าและเก็บเป็น FAQ
8. เวทีนำเสนอผลงาน (Demo Day)	นำเสนอระบบที่สร้างเสร็จต่อผู้บริหารและเพื่อนร่วมงาน	ทั้งคณะ	สร้างแรงจูงใจและขยายผล
9. พี่เลี้ยงสอนงาน (Coaching)	ผู้ผ่านการอบรมรุ่นแรกเป็นพี่เลี้ยง	ผู้เรียนรุ่นถัดไป	ขยายผลโดยไม่ต้องพึ่งวิทยากรภายนอก

8. ผลลัพธ์และประโยชน์ที่ได้รับ

ผลลัพธ์เชิงองค์ความรู้

-เจ้าหน้าที่สำนักงานอธิบายเส้นทางการสร้างเว็บแอปทั้ง 8 ขั้นได้ และรู้ว่าแต่ละขั้นต้องตรวจอะไร
-เขียน Requirement และสั่งงาน AI ได้อย่างมีโครงสร้าง ไม่ใช่การพิมพ์ลอย ๆ
-เข้าใจบทบาทของ API Keys ทั้ง 3 ค่า และรู้ว่าค่าใดห้ามเผยแพร่
-แก้ปัญหาเบื้องต้นได้เอง โดยใช้ error เป็นข้อมูลป้อนกลับให้ AI

ผลลัพธ์เชิงกระบวนการ

-มีเครื่องมือมาตรฐานใช้ร่วมกัน ได้แก่ แบบฟอร์ม Requirement คู่มือ 8 ขั้น คลัง Prompt และคลัง Troubleshooting
-งานที่เคยอยู่ในไฟล์ Excel หลายเวอร์ชัน ถูกยกขึ้นเป็นระบบเดียวที่ทุกคนเห็นข้อมูลชุดเดียวกัน
-ลดเวลาค้นหาข้อมูลและจัดทำรายงานประจำเดือน
-ระบบเปิดใช้งานผ่านลิงก์ได้ทันที ไม่จำกัดว่าต้องอยู่ที่เครื่องผู้สร้าง

ผลลัพธ์เชิงการควบคุมภายในและความเสี่ยง

-ข้อมูลมีการควบคุมสิทธิ์ตามบทบาท แทนการส่งไฟล์ให้กันซึ่งควบคุมไม่ได้
-มีประวัติการแก้ไขข้อมูลและสำเนาโค้ดบน GitHub ลดความเสี่ยงข้อมูลสูญหาย
-มีมาตรฐานความปลอดภัยขั้นต่ำที่ทุกโปรเจกต์ต้องปฏิบัติ

ผลลัพธ์เชิงบุคลากรและองค์กร

-บุคลากรสายสนับสนุนวิชาการมีทักษะดิจิทัลเพิ่มขึ้นอย่างเป็นรูปธรรม และมีผลงานที่จับต้องได้
-ลดการพึ่งพาการจ้างพัฒนาระบบภายนอกสำหรับงานขนาดเล็กถึงกลาง
-เกิดวัฒนธรรมการแก้ปัญหาด้วยเครื่องมือใหม่ แทนการยอมรับข้อจำกัดเดิม

ตัวชี้วัดที่เสนอให้ใช้ติดตามผล

ตัวชี้วัด	เป้าหมาย
จำนวนเจ้าหน้าที่ที่ผ่านการอบรมและสร้างเว็บแอปได้สำเร็จอย่างน้อย 1 ระบบ	ไม่น้อยกว่า 3 คนต่อปี
จำนวนระบบที่นำขึ้นใช้งานจริง (Deploy แล้วและมีผู้ใช้จริง)	ไม่น้อยกว่า 2 ระบบต่อปี
ร้อยละของระบบที่มีเอกสารคู่มือผู้ใช้และบันทึกผู้ดูแลระบบครบ	ร้อยละ 100
ร้อยละของระบบที่ผ่านการตรวจมาตรฐานความปลอดภัยขั้นต่ำ	ร้อยละ 100
จำนวน Prompt และรายการ Troubleshooting ที่เพิ่มเข้าคลังความรู้	ไม่น้อยกว่า 20 รายการต่อปี
ความพึงพอใจของผู้ใช้ระบบที่พัฒนาขึ้น	ไม่น้อยกว่าร้อยละ 80
เวลาที่ใช้จัดทำรายงานประจำเดือนของงานที่เปลี่ยนมาใช้ระบบ	ลดลงไม่น้อยกว่าร้อยละ 50

9. ข้อเสนอแนะและแผนพัฒนาต่อไป

ข้อเสนอแนะระดับคณะ

1. จัดทำคู่มือ 8 ขั้นตอนฉบับคณะพร้อมภาพประกอบ และปรับปรุงทุกครั้งที่เครื่องมือเปลี่ยนหน้าตา
2. กำหนดมาตรฐานความปลอดภัยขั้นต่ำเป็นข้อปฏิบัติของคณะ โดยเฉพาะเรื่องไฟล์. env, Service Role Key และการตั้ง Repository เป็น Private
3. จัดทำทะเบียนระบบที่พัฒนาขึ้น ระบุผู้ดูแลหลักและผู้ดูแลสำรองทุกระบบ เพื่อไม่ให้ระบบผูกกับคนเดียว
4. คัดเลือกงานนำร่องปีละ 2–3 งาน ตามเกณฑ์ในแนวปฏิบัติที่ดีที่ 5 แล้วผลักดันให้เสร็จจริง แทนการเริ่มหลายงานพร้อมกันแล้วไม่เสร็จสักงาน
5. จัด Demo Day ปีละครั้ง ให้ผู้พัฒนานำเสนอผลงานต่อผู้บริหารและเพื่อนร่วมงาน เพื่อสร้างแรงจูงใจและขยายผล
6. สนับสนุนบัญชีเครื่องมือที่จำเป็น และพิจารณาตั้งงบประมาณรองรับเมื่อระบบขยายเกินขีดจำกัดของแพ็กเกจฟรี
7. กำหนดให้ระบบที่จะเปิดใช้จริงต้องผ่านการเห็นชอบของคณบดี และตรวจสอบความสอดคล้องกับกฎหมายคุ้มครองข้อมูลส่วนบุคคลก่อนนำข้อมูลจริงเข้าระบบ

ข้อเสนอแนะต่อมหาวิทยาลัย

1. จัดอบรมหลักสูตรนี้ในระดับมหาวิทยาลัย เพื่อขยายผลไปยังส่วนงานอื่น โดยใช้รูปแบบเรียน 2 ครั้งพร้อมโจทย์งานจริง
2. กำหนดแนวปฏิบัติกลางด้านความปลอดภัยและข้อมูลส่วนบุคคล สำหรับระบบที่ส่วนงานพัฒนาขึ้นเอง
3. จัดให้มีที่ปรึกษาด้านเทคนิคจากหน่วยงานเทคโนโลยีสารสนเทศ เพื่อตรวจทานระบบก่อนเปิดใช้จริง และช่วยประเมินว่าซ้ำซ้อนกับระบบกลางหรือไม่
4. พิจารณาจัดหาบัญชีเครื่องมือ AI แบบองค์กร ซึ่งคุ้มค่ากว่าการให้แต่ละส่วนงานจัดหาเอง และควบคุมความปลอดภัยได้ดีกว่า
5. สร้างพื้นที่กลางแลกเปลี่ยนผลงานและ Prompt ระหว่างส่วนงาน เพื่อไม่ให้แต่ละคณะพัฒนาสิ่งเดียวกันซ้ำ
6. นับผลงานการพัฒนาระบบเป็นผลงานเชิงพัฒนางาน ของบุคลากรสายสนับสนุนวิชาการ เพื่อเป็นแรงจูงใจในความก้าวหน้าทางสายอาชีพ

หมายเหตุ เนื้อหาในรายงานฉบับนี้ถอดจากเอกสารประกอบการอบรมเชิงปฏิบัติการ “สร้างเว็บด้วย ChatGPT (Codex)” รอบที่ 1 ซึ่งจัดการเรียนการสอนออนไลน์ 2 ครั้ง พร้อมวิดีโอบันทึกย้อนหลัง ขั้นตอน ชื่อเมนู และรูปแบบหน้าจอของเครื่องมือภายนอก (VS Code, Supabase, GitHub, Vercel) อาจเปลี่ยนแปลงตามเวอร์ชันของผู้ให้บริการ จึงควรตรวจสอบกับหน้าจอจริงและปรับปรุงคู่มือเป็นระยะ