Lovable.dev ซึ่งเป็นแพลตฟอร์มสร้างเว็บไซต์และเว็บแอปพลิเคชันด้วย AI โดยผู้ใช้บรรยายสิ่งที่ต้องการด้วยภาษาธรรมชาติ แล้วระบบจะช่วยสร้างหน้าจอ เขียนโค้ด เชื่อมฐานข้อมูล ทดสอบ และเผยแพร่เว็บให้ใช้งานได้จริง โค้ดของแต่ละโปรเจกต์สามารถเชื่อมต่อกับ GitHub เพื่อสำรอง แก้ไขต่อ หรือส่งให้นักพัฒนาดูแลได้

คู่มือใช้ Lovable.dev สำหรับผู้เริ่มต้น

1. Lovable.dev เหมาะกับงานแบบใด

Lovable เหมาะกับการสร้างงาน เช่น

- เว็บไซต์องค์กรหรือเว็บไซต์แนะนำบริการ
- ระบบลงทะเบียนกิจกรรม
- ระบบสมาชิก
- แบบประเมินออนไลน์
- Dashboard แสดงผลข้อมูล
- ระบบจองห้องหรือจองกิจกรรม
- ระบบจัดการรายชื่อผู้เข้าร่วม
- เว็บแอปสำหรับบันทึกข้อมูล
- Prototype สำหรับทดลองแนวคิด
- Minimum Viable Product หรือ MVP

จุดสำคัญคือ **Lovable ไม่ได้สร้างเพียงภาพตัวอย่าง แต่สร้างเป็นเว็บแอปที่มีโค้ดจริง และสามารถเชื่อมฐานข้อมูล ระบบเข้าสู่ระบบ API และบริการภายนอกได้**

อย่างไรก็ตาม ผู้เริ่มต้นควรมอง Lovable เป็น **ผู้ช่วยพัฒนาแอป** ไม่ใช่ผู้รับผิดชอบทุกอย่างแทนเรา ***คุณยังต้องกำหนดความต้องการ ตรวจสอบผลลัพธ์ ทดลองระบบ และพิจารณาความปลอดภัยของข้อมูลด้วย***

---

2. ก่อนเริ่มใช้ ต้องมีความรู้อะไรบ้าง

คุณไม่จำเป็นต้องเขียนโปรแกรมเก่ง แต่ควรรู้แนวคิดพื้นฐานต่อไปนี้

2.1 เข้าใจส่วนประกอบของเว็บแอป

Frontend

ส่วนที่ผู้ใช้มองเห็นและกดใช้งาน เช่น

- หน้าแรก
- เมนู
- แบบฟอร์ม
- ปุ่ม
- ตาราง
- Dashboard
- หน้าสมัครสมาชิก

Backend

ระบบที่ทำงานเบื้องหลัง เช่น

- ตรวจสอบชื่อผู้ใช้และรหัสผ่าน
- บันทึกข้อมูล
- คำนวณคะแนน
- ส่งอีเมล
- กำหนดสิทธิ์ผู้ใช้
- เชื่อมต่อบริการอื่น

Database

พื้นที่เก็บข้อมูลอย่างเป็นระบบ เช่น

- รายชื่อสมาชิก

- กิจกรรม
- คะแนน
- การลงทะเบียน
- คำตอบแบบประเมิน
- ประวัติการใช้งาน

Lovable สามารถสร้างส่วนติดต่อผู้ใช้ได้โดยตรง ส่วนระบบฐานข้อมูลและการเข้าสู่ระบบสามารถเชื่อมกับ Lovable Cloud หรือ Supabase ได้ โดย Supabase รองรับฐานข้อมูล PostgreSQL ระบบ Authentication การเก็บไฟล์ และการทำงานฝั่งเซิร์ฟเวอร์

2.2 เข้าใจคำศัพท์ที่ควรรู้

คำศัพท์	ความหมายอย่างง่าย
UI	หน้าตาและองค์ประกอบของแอป
UX	ประสบการณ์และความสะดวกในการใช้งาน
Responsive	แสดงผลได้ดีบนคอมพิวเตอร์ แท็บเล็ต และมือถือ
Authentication	ระบบสมัครสมาชิกและเข้าสู่ระบบ
Authorization	การกำหนดว่าใครมีสิทธิ์ทำอะไร
Database	ฐานข้อมูล
Table	ตารางเก็บข้อมูล
Field หรือ Column	หัวข้อข้อมูล เช่น ชื่อ อีเมล คะแนน
Record หรือ Row	ข้อมูลหนึ่งรายการ
API	ช่องทางให้ระบบต่าง ๆ แลกเปลี่ยนข้อมูล
Deploy หรือ Publish	นำระบบขึ้นออนไลน์
Domain	ชื่อเว็บไซต์
GitHub	ระบบเก็บโค้ดและประวัติการแก้ไข
Bug	ข้อผิดพลาดของระบบ
Component	ชิ้นส่วนของหน้าเว็บ เช่น ปุ่ม การ์ด หรือเมนู

Lovable มีอภิธานศัพท์อย่างเป็นทางการที่รวมคำศัพท์ด้าน Frontend, Backend, Supabase และการออกแบบไว้ด้วย

2.3 ทักษะที่มีประโยชน์มากที่สุด

สำหรับผู้เริ่มต้น ทักษะสำคัญที่สุดไม่ใช่การเขียนโค้ด แต่คือ

1. การอธิบายปัญหาให้ชัด
2. การแยกงานใหญ่เป็นงานย่อย
3. การเขียน Prompt แบบมีโครงสร้าง
4. การทดลองใช้งานเหมือนผู้ใช้จริง
5. การสังเกตว่าอะไรทำงานผิด
6. การบอก AI ว่าให้แก้ตรงไหนโดยไม่กระทบส่วนอื่น

---

3. สิ่งที่ควรเตรียมก่อนสร้างโปรเจกต์

อย่าเริ่มด้วยคำสั่งสั้น ๆ เช่น สร้างระบบลงทะเบียนให้หน่อย เพราะ AI ต้องเดาหลายเรื่อง และมีโอกาสสร้างระบบที่ไม่ตรงความต้องการ

ควรเตรียมข้อมูล 7 ส่วนนี้ก่อน

3.1 ปัญหาที่ต้องการแก้

ตัวอย่าง:

ปัจจุบันผู้สมัครกิจกรรมกรอก Google Form แต่เจ้าหน้าที่ต้องนำข้อมูลมาจัดกลุ่มและตรวจสอบการชำระเงินด้วยตนเอง ทำให้ใช้เวลามาก

3.2 กลุ่มผู้ใช้งาน

ตัวอย่าง:

- ผู้สมัครกิจกรรม
- เจ้าหน้าที่
- ผู้ดูแลระบบ
- วิทยากร
- ผู้บริหารที่ดูรายงาน

3.3 งานที่ผู้ใช้แต่ละกลุ่มต้องทำ

ตัวอย่าง:

ผู้สมัคร:

- ดูรายละเอียดกิจกรรม
- กรอกใบสมัคร
- อัปโหลดหลักฐาน
- ตรวจสอบสถานะ

เจ้าหน้าที่:

- ดูรายชื่อ
- เปลี่ยนสถานะ
- ค้นหาและกรองข้อมูล
- ดาวน์โหลดรายงาน

3.4 หน้าจอที่ต้องมี

ตัวอย่าง:

1. หน้าแรก
2. หน้ารายละเอียดกิจกรรม

3. หน้าลงทะเบียน
4. หน้าเข้าสู่ระบบ
5. หน้าตรวจสอบสถานะ
6. Dashboard เจ้าหน้าที่
7. หน้าจัดการกิจกรรม

3.5 ข้อมูลที่ต้องเก็บ

ตัวอย่างตาราง participants

- id
- full_name
- email
- phone
- organization
- position
- activity_id
- payment_status
- registration_status
- created_at

3.6 กฎการทำงาน

ตัวอย่าง:

- อีเมลหนึ่งรายการลงทะเบียนกิจกรรมเดียวกันได้หนึ่งครั้ง
- รับผู้สมัครไม่เกิน 60 คน
- เมื่อเต็มให้เปลี่ยนเป็นรายชื่อสำรอง
- เจ้าหน้าที่เท่านั้นที่เปลี่ยนสถานะได้
- ผู้สมัครดูได้เฉพาะข้อมูลของตนเอง

3.7 รูปแบบการออกแบบ

ระบุให้ชัด เช่น

- เรียบง่าย เป็นมิตร
- ธีมสิ่งแวดล้อม
- ใช้สีเขียวอ่อน ขาว และน้ำตาล
- ตัวอักษรภาษาไทยอ่านง่าย
- รองรับมือถือเป็นหลัก
- ปุ่มสำคัญต้องเห็นเด่นชัด

---

4. วิธีเริ่มใช้งาน Lovable.dev

ขั้นที่ 1 สมัครบัญชี

เข้า Lovable แล้วสมัครด้วยอีเมล หรือบัญชีที่แพลตฟอร์มรองรับ จากนั้นจะเข้าสู่ Workspace สำหรับจัดเก็บโปรเจกต์ ไม่ต้องติดตั้งโปรแกรมลงในเครื่องก่อนเริ่มใช้งาน

แพ็กเกจและจำนวนเครดิตอาจเปลี่ยนแปลงได้ ควรตรวจสอบหน้าแผนบริการในวันที่เริ่มใช้งาน ปัจจุบันเอกสาร Quick Start ระบุว่ามีแผนฟรีพร้อมโควตาการสร้างแบบจำกัด

ขั้นที่ 2 สร้างโปรเจกต์ใหม่

ในช่อง Prompt ให้บรรยายสิ่งที่ต้องการสร้าง อย่าใส่ทุกฟังก์ชันในครั้งแรก ให้เริ่มจากโครงสร้างหลักหรือ MVP ก่อน

ตัวอย่าง Prompt เริ่มต้น:

สร้างเว็บแอปพลิเคชันสำหรับลงทะเบียนเข้าร่วมกิจกรรมอบรมด้านสิ่งแวดล้อม

กลุ่มผู้ใช้มี 2 กลุ่ม:
1. ผู้สมัคร

2. เจ้าหน้าที่

เวอร์ชันแรกให้สร้างเฉพาะ Frontend และใช้ข้อมูลตัวอย่างก่อน

หน้าที่ต้องมี:
- หน้าแรกแสดงชื่อกิจกรรม รายละเอียด วัน เวลา สถานที่ และจำนวนที่นั่ง
- หน้ารายการกิจกรรม
- หน้ารายละเอียดกิจกรรม
- แบบฟอร์มลงทะเบียน
- หน้าสำเร็จหลังลงทะเบียน
- Dashboard เจ้าหน้าที่แบบข้อมูลตัวอย่าง

การออกแบบ:
- ภาษาไทย
- เรียบง่าย เป็นมิตร และดูเป็นมืออาชีพ
- ธีมสีเขียวธรรมชาติ
- รองรับมือถือ แท็บเล็ต และคอมพิวเตอร์
- ใช้การ์ด มุมโค้ง และพื้นที่ว่างอ่านง่าย

ยังไม่ต้องเชื่อมฐานข้อมูลหรือระบบเข้าสู่ระบบ

Lovable รองรับการบรรยายแอปด้วยภาษาธรรมชาติ และเอกสารแนะนำว่าการให้รายละเอียดชัดเจนจะช่วยให้ได้ผลลัพธ์ตรงขึ้น

ขั้นที่ 3 ตรวจสอบโครงสร้างแรก

หลังระบบสร้างเสร็จ อย่ารีบเพิ่มฐานข้อมูลทันที ให้ตรวจสอบก่อนว่า

- มีหน้าครบหรือไม่
- เมนูไปถูกหน้าหรือไม่
- ปุ่มทำงานหรือไม่
- ข้อความภาษาไทยถูกต้องหรือไม่

- แบบฟอร์มมีช่องครบหรือไม่
- มือถือแสดงผลดีหรือไม่
- สีและตัวอักษรอ่านง่ายหรือไม่

ขั้นที่ 4 แก้ทีละเรื่อง

หลีกเลี่ยง Prompt แบบนี้:

ช่วยปรับให้สวยขึ้นและเพิ่มระบบสมาชิก ฐานข้อมูล รายงาน อีเมล และแก้ปุ่มทั้งหมด

ควรแยกเป็นคำสั่งย่อย เช่น

ปรับเฉพาะหน้าแรกให้มีลำดับข้อมูลดังนี้:
1. Hero section
2. กิจกรรมที่กำลังเปิดรับ
3. จุดเด่นของกิจกรรม
4. ขั้นตอนการสมัคร
5. คำถามที่พบบ่อย
6. Footer

อย่าแก้ไขหน้าอื่นและอย่าเปลี่ยนโครงสร้างฐานข้อมูล

จากนั้นจึงสั่งต่อ:

ปรับแบบฟอร์มลงทะเบียนให้เหมาะกับมือถือ
ให้แต่ละช่องเรียงหนึ่งคอลัมน์บนหน้าจอขนาดเล็ก
เพิ่มข้อความแจ้งเตือนใต้ช่องที่กรอกไม่ครบ
อย่าแก้ Dashboard

หลักสำคัญคือ **หนึ่ง Prompt ต่อหนึ่งเป้าหมายหลัก**

---

5. สูตรเขียน Prompt ที่ใช้ได้ดี

ใช้โครงสร้างนี้:

Context + Goal + Users + Features + Rules + Design + Constraints

ตัวอย่าง:

บริบท:
ระบบนี้ใช้สำหรับโรงเรียนที่เข้าร่วมโครงการอนุรักษ์พลังงานและสิ่งแวดล้อม

เป้าหมาย:
ช่วยให้นักเรียนแกนนำบันทึกกิจกรรมและติดตามผลการดำเนินงานของโรงเรียน

ผู้ใช้งาน:
- นักเรียนแกนนำ
- ครูที่ปรึกษา
- ผู้ดูแลโครงการ

ฟังก์ชัน:
- เข้าสู่ระบบ
- สร้างกิจกรรม
- อัปโหลดภาพ
- บันทึกผลลัพธ์
- แสดงคะแนนรายหมวด
- Dashboard สรุปผล

กฎ:
- นักเรียนแก้ไขได้เฉพาะข้อมูลของโรงเรียนตนเอง
- ครูอนุมัติกิจกรรมได้
- ผู้ดูแลเห็นทุกโรงเรียน
- กิจกรรมต้องมีชื่อ วันที่ หมวด และคำอธิบาย

การออกแบบ:
- ภาษาไทย
- ทันสมัย เหมาะกับนักเรียนมัธยม
- ใช้สีเขียว น้ำเงิน และพื้นหลังสีอ่อน
- รองรับมือถือ

ข้อจำกัด:
เริ่มจาก Frontend และข้อมูลจำลองก่อน
ยังไม่ต้องเชื่อมฐานข้อมูล

ใช้ Prompt แบบ Acceptance Criteria

Acceptance Criteria คือเกณฑ์ที่ใช้ตัดสินว่างานเสร็จหรือยัง

ตัวอย่าง:

ปรับหน้ารายการกิจกรรม โดยถือว่างานสำเร็จเมื่อ:

- ผู้ใช้ค้นหาจากชื่อกิจกรรมได้
- กรองตามเดือนและสถานะได้
- แสดงผลเป็นการ์ดบนมือถือ
- แสดงผลเป็นตารางบนหน้าจอใหญ่
- เมื่อไม่มีข้อมูลให้แสดง Empty State
- ปุ่มดูรายละเอียดเชื่อมไปหน้ารายละเอียดได้
- ห้ามเปลี่ยนส่วน Header และ Footer

Prompt แบบนี้ช่วยลดการตีความคลาดเคลื่อนได้มาก

---

6. Plan Mode, การทำทีละขั้น และการใช้เครดิต

สำหรับงานที่มีหลายส่วน ควรขอให้ Lovable วางแผนก่อนลงมือ เช่น

ยังไม่ต้องแก้โค้ด

ช่วยวิเคราะห์โปรเจกต์นี้และเสนอแผนเพิ่มระบบสมาชิก โดยอธิบาย:
1. หน้าที่ต้องเพิ่ม
2. ตารางฐานข้อมูลที่ต้องมี
3. สิทธิ์ของผู้ใช้แต่ละประเภท
4. ความเสี่ยงด้านความปลอดภัย
5. ลำดับการพัฒนา

รอให้ฉันตรวจแผนก่อนจึงค่อยลงมือ

เอกสารแนวปฏิบัติของ Lovable แนะนำให้ตั้งรากฐานของโปรเจกต์ ใช้ Knowledge หรือ Context, Plan mode, Visual edits และระบบควบคุมเวอร์ชันอย่างเหมาะสม

แนวทางประหยัดเครดิต:

- วางแผนก่อนสั่งสร้าง
- รวมการแก้ไขเล็ก ๆ ที่เกี่ยวข้องกัน
- หลีกเลี่ยงคำว่า “ทำให้ดีขึ้น” โดยไม่ระบุจุด
- ใช้ Visual Edit เมื่อแก้สี ขนาด ระยะห่าง หรือข้อความ
- ตรวจ Preview ก่อนส่ง Prompt รอบใหม่
- เก็บเวอร์ชันที่ใช้งานได้ก่อนเพิ่มฟังก์ชันใหญ่

---

7. การเชื่อมฐานข้อมูล

เมื่อหน้าจอและ User flow เริ่มนิ่ง จึงค่อยเชื่อม Backend

ทางเลือกที่ 1: Lovable Cloud

เหมาะกับผู้เริ่มต้นที่ต้องการให้ระบบจัดการโครงสร้างพื้นฐานโดยไม่ต้องตั้งค่าหลายบริการเอง

ทางเลือกที่ 2: Supabase

เหมาะเมื่อคุณต้องการ

- ฐานข้อมูล
- สมัครสมาชิกและเข้าสู่ระบบ
- อัปโหลดไฟล์
- กำหนดสิทธิ์ข้อมูล
- เชื่อม API
- ดูและบริหารข้อมูลจาก Dashboard ของ Supabase

ขั้นตอนทั่วไปคือสร้างบัญชี Supabase สร้าง Project แล้วเชื่อมจากส่วน Integrations ของ Lovable หลังเชื่อมแล้วสามารถสั่งเพิ่มระบบ Login หรือให้ Lovable สร้าง Schema ของฐานข้อมูลได้

ตัวอย่าง Prompt:

เชื่อมโปรเจกต์นี้กับ Supabase

สร้างตารางดังนี้:

1. profiles
- id เชื่อมกับ auth.users
- full_name
- role มีค่า participant หรือ admin
- organization
- created_at

2. activities
- id
- title
- description
- activity_date
- location
- capacity
- status
- created_at

3. registrations
- id
- user_id
- activity_id
- registration_status
- payment_status
- created_at

กำหนดความสัมพันธ์ของตารางให้ถูกต้อง ยังไม่ต้องสร้าง Dashboard เพิ่ม หลังทำเสร็จให้อธิบายสิ่งที่สร้าง

จากนั้นเพิ่ม Authentication:

เพิ่มระบบสมัครสมาชิกและเข้าสู่ระบบด้วยอีเมลและรหัสผ่าน

เมื่อสมัครสำเร็จ:
- สร้างข้อมูลในตาราง profiles อัตโนมัติ
- กำหนด role เริ่มต้นเป็น participant
- พาไปหน้า Dashboard ของผู้ใช้

เพิ่มหน้า:

- Login
- Sign up
- Forgot password
- Reset password

ห้ามให้ผู้ใช้เลือก role เป็น admin ตอนสมัคร

---

8. ความปลอดภัยที่ผู้เริ่มต้นต้องระวัง

การที่แอปทำงานได้ ไม่ได้หมายความว่าแอปปลอดภัยแล้ว

8.1 อย่าเก็บ Secret ใน Frontend

ข้อมูลต่อไปนี้ไม่ควรเขียนไว้ในโค้ดหน้าเว็บที่ผู้ใช้ดาวน์โหลดได้:

- Secret API key
- Service role key
- รหัสผ่านฐานข้อมูล
- Private token
- Credential ของบริการชำระเงิน

ควรใช้ระบบ Secrets หรือฟังก์ชันฝั่งเซิร์ฟเวอร์

8.2 กำหนดสิทธิ์ที่ฐานข้อมูล

ห้ามพึ่งเพียงการซ่อนปุ่มบนหน้าจอ เพราะผู้ใช้ที่มีความรู้ทางเทคนิคอาจเรียกฐานข้อมูลโดยตรงได้

ตัวอย่างกฎ:

- ผู้สมัครอ่านข้อมูลของตัวเอง
- ผู้สมัครแก้ไขข้อมูลของตัวเอง
- ผู้สมัครห้ามอ่านข้อมูลผู้อื่น

- Admin อ่านและแก้ไขได้ทั้งหมด

Prompt ตัวอย่าง:

ตรวจสอบความปลอดภัยของ Supabase

สร้างหรือปรับ Row Level Security ให้:
- ผู้ใช้ที่เข้าสู่ระบบดูและแก้ไข profile ของตัวเองได้เท่านั้น
- ผู้ใช้ดู registrations ของตัวเองได้เท่านั้น
- ผู้ใช้สร้าง registration ในนามตัวเองได้เท่านั้น
- admin สามารถดูและจัดการข้อมูลทั้งหมด
- ผู้ที่ยังไม่เข้าสู่ระบบห้ามอ่าน registrations

อธิบายแต่ละ policy ที่สร้าง และตรวจสอบว่าไม่มีตารางสำคัญเปิดเป็น public

8.3 ข้อมูลส่วนบุคคล

เก็บเฉพาะข้อมูลที่จำเป็น และพิจารณาเรื่อง

- วัตถุประสงค์ในการเก็บ
- ความยินยอม
- ระยะเวลาการเก็บ
- ผู้มีสิทธิ์เข้าถึง
- การลบข้อมูล
- การดาวน์โหลดข้อมูล
- การแจ้งเหตุข้อมูลรั่วไหล

โดยเฉพาะระบบที่เก็บข้อมูลนักเรียน สุขภาพ เลขประจำตัวประชาชน หรือเอกสารสำคัญ ควรให้ผู้เชี่ยวชาญตรวจสอบก่อนเปิดใช้จริง

---

9. วิธีทดสอบเว็บแอป

สร้าง Test checklist ก่อน Publish

9.1 ทดสอบหน้าจอ

- Desktop
- Tablet
- Mobile
- หน้าจอแนวตั้งและแนวนอน
- ข้อความยาว
- รูปภาพหลายขนาด
- ข้อมูลจำนวนมาก
- ไม่มีข้อมูล

9.2 ทดสอบแบบฟอร์ม

ลองกรณีต่อไปนี้:

- ไม่กรอกข้อมูล
- อีเมลไม่ถูกต้อง
- กรอกข้อมูลซ้ำ
- กรอกข้อความยาวมาก
- อัปโหลดไฟล์ผิดประเภท
- ไฟล์ใหญ่เกินกำหนด
- กดปุ่มส่งซ้ำ
- อินเทอร์เน็ตขาดระหว่างส่ง

9.3 ทดสอบสิทธิ์

สร้างบัญชีทดลองอย่างน้อย 3 แบบ:

1. ผู้ใช้ทั่วไป A

2. ผู้ใช้ทั่วไป B
3. Admin

ตรวจสอบว่า A มองไม่เห็นข้อมูลของ B และผู้ใช้ทั่วไปเข้าหน้า Admin ไม่ได้

9.4 ทดสอบ User journey

ตัวอย่างเส้นทาง:

เข้าเว็บไซต์ ⟶ ดูกิจกรรม ⟶ สมัครสมาชิก ⟶ ลงทะเบียน ⟶ ได้รับการยืนยัน ⟶ ดูสถานะ ⟶ ยกเลิกการลงทะเบียน

ควรทำทุกขั้นตอนจริง ไม่ใช่แค่เปิดดูแต่ละหน้าแยกกัน

---

10. วิธีแก้ปัญหาเมื่อระบบผิดพลาด

ลำดับการแก้ที่แนะนำ

ขั้นที่ 1 อ่านข้อความ Error

อย่าสั่งเพียงว่า “แก้ให้หน่อย”

ให้คัดลอก Error และบอกเหตุการณ์ที่ทำให้เกิด เช่น

เกิดข้อผิดพลาดเมื่อกดปุ่มบันทึกกิจกรรม

ขั้นตอนที่ทำ:
1. Login ด้วยบัญชี admin
2. เปิดหน้า /admin/activities/new
3. กรอกข้อมูลครบ
4. กด Save

ผลที่คาดหวัง:
บันทึกข้อมูลและกลับไปหน้ารายการกิจกรรม

ผลที่เกิดขึ้น:
หน้าจอค้างและ Console แสดงข้อความ:
[วางข้อความ error]

ช่วยวิเคราะห์สาเหตุก่อน
อย่าเปลี่ยน UI และอย่าแก้ส่วนอื่นที่ไม่เกี่ยวข้อง

ขั้นที่ 2 ใช้ Try to Fix

Lovable มีปุ่ม **Try to Fix** สำหรับข้อผิดพลาดในการ Build เอกสาร Troubleshooting แนะนำให้ลองใช้ปุ่มนี้ก่อน แล้วตรวจว่าปัญหาหายจริงหรือไม่

ขั้นที่ 3 ตรวจ Console และ Logs

ปัญหา Frontend มักดูได้จาก Browser console ส่วนปัญหา Backend หรือฐานข้อมูลควรตรวจ Logs ของ Supabase หรือบริการที่เชื่อมต่อ แล้วนำข้อความ Error ที่แท้จริงกลับมาให้ Lovable วิเคราะห์

ขั้นที่ 4 แยกประเภทปัญหา

อาการ	จุดที่ควรตรวจ
หน้าเว็บขาว	Build error, import หรือ JavaScript
ปุ่มกดไม่ได้	Event handler, element ซ้อนทับ
บันทึกไม่ได้	Validation, database, permission
Login ไม่ได้	Authentication configuration
เห็นข้อมูลคนอื่น	RLS หรือ authorization
รูปไม่ขึ้น	URL, storage policy, file path
หน้าโหลดช้า	รูปใหญ่, query มาก, component render ซ้ำ

Publish แล้วต่างจาก Preview environment, build หรือ cache

ขั้นที่ 5 ย้อนกลับเวอร์ชัน

หาก AI แก้แล้วระบบเสียมากกว่าเดิม ให้ย้อนกลับไปยังเวอร์ชันที่ทำงานได้ เอกสารทางการแนะนำการใช้ Revert เมื่อการแก้ไขทำให้โปรเจกต์ติดขัด

Prompt ที่เหมาะสม:

การแก้ไขล่าสุดทำให้หน้า Dashboard ใช้งานไม่ได้

กรุณา:
1. เปรียบเทียบกับเวอร์ชันก่อนหน้า
2. ระบุไฟล์ที่ถูกเปลี่ยน
3. ระบุสาเหตุที่น่าจะทำให้เกิดปัญหา
4. แก้เฉพาะส่วนที่เกี่ยวข้อง
5. ห้ามเปลี่ยนฐานข้อมูลและระบบ Login

---

11. การเชื่อม GitHub

แม้จะไม่ใช่นักพัฒนา ก็ควรเชื่อม GitHub เมื่อโปรเจกต์เริ่มสำคัญ เพราะช่วยเรื่อง

- สำรองโค้ด
- เก็บประวัติการแก้ไข
- ส่งให้นักพัฒนาตรวจ
- แก้โค้ดจากเครื่อง
- ทำงานร่วมกัน
- นำไป Deploy ที่อื่น
- ลดการผูกติดกับแพลตฟอร์มเดียว

Lovable รองรับการซิงก์สองทางกับ GitHub และ GitLab โค้ดสามารถนำไปแก้ไขหรือ Deploy ภายนอกได้

ข้อควรระวังคือการแก้โค้ดจากหลายทางพร้อมกันอาจเกิด Conflict จึงควรกำหนดว่าใครแก้ส่วนใด และ Commit งานเป็นระยะ

---

12. การเผยแพร่เว็บไซต์

เมื่อระบบพร้อม ให้กด Publish และตั้ง URL ของโปรเจกต์

Lovable สามารถเผยแพร่ผ่านโดเมน lovable.app ได้ และสามารถเชื่อม Custom domain ในแผนที่รองรับ หลัง Publish แล้วสามารถอัปเดตเวอร์ชันใหม่ เปลี่ยน URL หรือกำหนดผู้ที่เข้าถึงได้ตามตัวเลือกของแผน

ขั้นตอนก่อน Publish:

1. ตรวจข้อความและรูปภาพ
2. ลบข้อมูลทดลอง
3. ตรวจสิทธิ์ฐานข้อมูล
4. ทดสอบมือถือ
5. ทดสอบ Login และ Logout
6. ทดสอบลืมรหัสผ่าน
7. ตรวจ Privacy policy
8. ตรวจ Contact information
9. ตรวจ URL และ Metadata
10. สำรองโค้ดไว้ใน GitHub

Custom domain ต้องตั้งค่าผ่านส่วน Domains และโปรเจกต์ต้อง Publish ก่อนโดเมนจึงเริ่มแสดงเว็บไซต์ได้

---

13. ตัวอย่างโปรเจกต์สำหรับฝึก

โปรเจกต์ที่ 1: เว็บไซต์แนะนำศูนย์การเรียนรู้

เหมาะสำหรับฝึก:

- Layout
- Responsive design

- เมนู
- การ์ด
- แกลเลอรี
- Contact form

Prompt:

สร้างเว็บไซต์ภาษาไทยสำหรับศูนย์การเรียนรู้ด้านพลังงานและสิ่งแวดล้อม

กลุ่มเป้าหมาย:
- โรงเรียน
- ครู
- ผู้บริหารสถานศึกษา
- HR บริษัท
- องค์กรปกครองส่วนท้องถิ่น

หน้าที่ต้องมี:
- หน้าแรก
- เกี่ยวกับเรา
- หลักสูตรและกิจกรรม
- ค่ายสำหรับโรงเรียน
- กิจกรรมสำหรับองค์กร
- บทความ
- ติดต่อเรา

หน้าแรกประกอบด้วย:
- Hero พร้อมปุ่มดูหลักสูตรและติดต่อจัดกิจกรรม
- จุดเด่นของศูนย์
- กลุ่มหลักสูตร
- ผลลัพธ์ที่ผู้เข้าร่วมจะได้รับ
- กิจกรรมล่าสุด

- ความคิดเห็นจากผู้ใช้บริการ
- Call to action
- Footer

ออกแบบแนวธรรมชาติร่วมสมัย
ดูเป็นมืออาชีพและเป็นมิตร
รองรับมือถือ
ใช้ข้อมูลตัวอย่างและภาพ placeholder
ยังไม่ต้องมีฐานข้อมูล

โปรเจกต์ที่ 2: แบบประเมินความสัมพันธ์ 30 ข้อ

เหมาะสำหรับฝึก:

- Form
- State
- การคำนวณ
- Progress bar
- Result page
- การแสดงคำแนะนำตามคะแนน

Prompt:

สร้างเว็บแอปแบบประเมินความสัมพันธ์คู่รักจำนวน 30 ข้อ

แต่ละข้อมีคำตอบ 5 ระดับ:
1 = ไม่เป็นจริงเลย
2 = ค่อนข้างไม่จริง
3 = ปานกลาง
4 = ค่อนข้างจริง
5 = เป็นจริงมาก

ฟังก์ชัน:
- แสดงทีละข้อ
- มี progress bar
- ย้อนกลับไปแก้คำตอบได้
- ป้องกันการข้ามข้อ
- คำนวณคะแนนเมื่อทำครบ
- แสดงคะแนนรวม
- แสดงผลแยกเป็น 5 มิติ
- แสดงคำอธิบายและคำแนะนำ
- มีปุ่มเริ่มใหม่
- ไม่เก็บข้อมูลส่วนบุคคล

ใช้ข้อมูลคำถามตัวอย่างก่อน
ออกแบบให้สงบ น่าเชื่อถือ และอ่านง่ายบนมือถือ
เพิ่มข้อความว่าแบบประเมินนี้ไม่ใช่เครื่องมือวินิจฉัยทางการแพทย์

โปรเจกต์ที่ 3: ระบบลงทะเบียนกิจกรรม

เหมาะสำหรับฝึก Full-stack:

- Database
- Authentication
- CRUD
- Admin dashboard
- File upload
- การกำหนดสิทธิ์

ลำดับสร้างที่เหมาะสม:

1. สร้างหน้าจอด้วยข้อมูลจำลอง
2. ตรวจ User flow
3. เชื่อม Supabase

4. สร้างตาราง
5. เพิ่ม Login
6. เชื่อมแบบฟอร์มกับฐานข้อมูล
7. สร้าง Admin dashboard
8. เพิ่ม RLS
9. ทดสอบสิทธิ์
10. Publish

โปรเจกต์ที่ 4: Green School Activity Tracker

เหมาะกับงานด้านสิ่งแวดล้อมและการศึกษา

ฟังก์ชัน:

- โรงเรียนสมัครเข้าระบบ
- บันทึกกิจกรรม
- เลือกหมวดพลังงาน น้ำ ขยะ พื้นที่สีเขียว
- อัปโหลดหลักฐาน
- ครูตรวจสอบ
- คำนวณคะแนน
- Dashboard รายโรงเรียน
- Leaderboard
- รายงานรายเดือน
- Export CSV

Prompt เริ่มต้น:

สร้างต้นแบบเว็บแอป Green School Activity Tracker

วัตถุประสงค์:
ให้โรงเรียนบันทึกและติดตามกิจกรรมด้านการอนุรักษ์พลังงานและสิ่งแวดล้อม

ผู้ใช้:
- นักเรียนแกนนำ
- ครูที่ปรึกษา
- ผู้ดูแลโครงการ

สร้าง Frontend ด้วยข้อมูลตัวอย่างก่อน

หน้าที่ต้องมี:
- Login
- Dashboard โรงเรียน
- เพิ่มกิจกรรม
- รายการกิจกรรม
- รายละเอียดกิจกรรม
- คะแนนรายหมวด
- Leaderboard
- Dashboard ผู้ดูแลระบบ

หมวดกิจกรรม:
- พลังงาน
- น้ำ
- ขยะ
- การเดินทาง
- อาหาร
- พื้นที่สีเขียว
- การสื่อสารและรณรงค์

Dashboard แสดง:
- คะแนนรวม
- จำนวนกิจกรรม
- กิจกรรมล่าสุด

- คะแนนแยกหมวด
- ความคืบหน้ารายเดือน

ออกแบบให้เหมาะกับนักเรียนมัธยม
ทันสมัย สนุก แต่ยังดูน่าเชื่อถือ
รองรับภาษาไทยและมือถือ

---

14. แผนฝึกใช้ Lovable สำหรับผู้เริ่มต้น 14 วัน

วันที่ 1–2: เรียนรู้โครงสร้างเว็บ

ศึกษา:

- หน้าเว็บ
- Component
- Frontend
- Backend
- Database
- Authentication

เป้าหมายคืออธิบายแอปหนึ่งระบบได้ว่า “มีใครใช้ ใช้ทำอะไร และมีหน้าอะไรบ้าง”

วันที่ 3–4: สร้างเว็บไซต์แบบไม่มีฐานข้อมูล

สร้างเว็บไซต์ 4–5 หน้า ฝึก:

- Prompt
- Layout
- Responsive
- Visual edits
- เมนู

วันที่ 5–6: สร้างแบบฟอร์มและระบบคำนวณ

เช่น แบบประเมิน เครื่องคิดคาร์บอน หรือแบบสำรวจ

วันที่ 7: ฝึกแก้ Bug

จงใจทดสอบ:

- ช่องว่าง
- ข้อมูลผิด
- ปุ่มซ้ำ
- มือถือ
- URL ผิด

วันที่ 8–9: เชื่อม Supabase

ฝึกสร้าง:

- ตาราง
- เพิ่มข้อมูล
- อ่านข้อมูล
- แก้ไขข้อมูล
- ลบข้อมูล

วันที่ 10: เพิ่ม Login

สร้างบัญชีผู้ใช้และหน้า Protected route

วันที่ 11: เรียนรู้ Role และ RLS

แยก User กับ Admin และทดสอบการเข้าถึงข้อมูล

วันที่ 12: เชื่อม GitHub

เรียนรู้คำพื้นฐาน:

- Repository
- Commit
- Branch
- Sync
- Revert

วันที่ 13: ทดสอบระบบครบวงจร

ใช้ Test checklist และให้คนอื่นทดลอง

วันที่ 14: Publish

เผยแพร่ เก็บ Feedback และวางรายการพัฒนาเวอร์ชันต่อไป

---

15. ข้อผิดพลาดที่ผู้เริ่มต้นพบบ่อย

เริ่มจากระบบใหญ่เกินไป

แทนที่จะสร้าง “ระบบบริหารโรงเรียนครบวงจร” ให้เริ่มจาก “ระบบบันทึกกิจกรรมของนักเรียนแกนนำ”

เปลี่ยนความต้องการระหว่างทางบ่อย

ก่อนสร้าง ให้เขียนขอบเขตของ Version 1 ให้ชัด

สั่งหลายเรื่องใน Prompt เดียว

ทำให้ AI แก้ส่วนที่ไม่เกี่ยวข้องและหา Bug ยาก

เชื่อว่า AI ทำถูกทั้งหมด

ต้องตรวจโค้ด พฤติกรรม สิทธิ์ และข้อมูลทุกครั้ง

เชื่อมฐานข้อมูลเร็วเกินไป

ควรทำ UI และ User flow ให้ค่อนข้างนิ่งก่อน

ไม่มีข้อมูลตัวอย่าง

ควรเตรียม Mock data ที่ใกล้เคียงข้อมูลจริง เพื่อดูว่าหน้าจอรองรับข้อความยาวและข้อมูลจำนวนมากหรือไม่

ไม่สำรองเวอร์ชัน

เชื่อม GitHub หรือใช้ระบบประวัติเวอร์ชันก่อนเปลี่ยนแปลงครั้งใหญ่

เปิดใช้จริงโดยไม่ตรวจ Security

ระบบที่มี Login ไม่ได้แปลว่าข้อมูลทุกอย่างถูกป้องกันแล้ว

สรุปกระบวนการทำงานที่แนะนำ

ใช้วงจรนี้ทุกโปรเจกต์:

คิดปัญหา ⟶ กำหนดผู้ใช้ ⟶ เขียน User flow ⟶ ระบุหน้าจอ ⟶ สร้าง Frontend ⟶ ทดสอบ ⟶ เชื่อมฐานข้อมูล ⟶ กำหนดสิทธิ์ ⟶ ทดสอบความปลอดภัย ⟶ เชื่อม GitHub ⟶ Publish ⟶ เก็บ Feedback ⟶ ปรับปรุง

สำหรับโปรเจกต์แรก ผมแนะนำให้เริ่มจาก **ระบบลงทะเบียนกิจกรรมขนาดเล็ก** หรือ **แบบประเมินออนไลน์** เพราะมีขอบเขตชัด ได้ฝึกทั้ง UI, Form, Logic, Database และ Dashboard โดยไม่ซับซ้อนเกินไปครับ