Vibe Coding for Everyone

สร้างแอปของตัวเองด้วย AI — โดยไม่ต้องเขียนโค้ดสักบรรทัด

คอร์สสดออนไลน์ · Bot Please (botplease.me)

จบคลาส = มีแอปใช้งานได้จริง 1 ตัว

OPENING

Roadmap วันนี้ — 3 ก้อนใหญ่

ก้อนที่ 1 · เข้าใจ

- Vibe Coding คืออะไร
- องค์ประกอบของเว็บ/แอป 8 ส่วน
- เครื่องมือแยกตามชั้น 5 หมวด
- ส่วนประกอบเว็บไซต์ vs แอป
- Framework สั่งงาน AI
- AI Builder vs Coding Agent

ก้อนที่ 2 · เห็นของจริง

- Live Demo 1: AI Builder (Google AI Studio)
- Live Demo 2: Coding Agent (Google Antigravity)

ก้อนที่ 3 · ลงมือทำ

- สร้างแอปของตัวเอง (Guided Hands-on)
- Wrap-up + Homework 7 วัน

ถามใน Chat ได้ตลอดคลาส

OPENING

ตั้งความคาดหวังให้ตรงกันก่อน

✓ วันนี้คุณจะได้

- แอปของตัวเอง 1 ตัว กดใช้งานได้จริง
- แชร์ลิงก์ preview ให้เพื่อนกดเล่นได้
- มองทะลุว่าแอปประกอบด้วยอะไร
- Framework สั่งงาน AI แบบมืออาชีพ
- แผนที่เครื่องมือ 20+ ตัว (Handout)

คลาสนี้ไม่สอน

- Deploy on Play store / App store
- เขียนโค้ด / อ่านโค้ด
- ประกอบเครื่องมือแยกชั้นเองละเอียด

(ทั้งหมดนี้คือ "ก้าวถัดไป" — ชี้ทางให้ตอนท้ายคลาส)

💡 เปิด **App Idea Worksheet** แล้วจดไอเดียแอปของคุณไว้เลย — เดี๋ยวช่วง **Hands-on** ได้สร้างจริง

01

Vibe Coding คืออะไร

ที่มาของคำ · ความหมายที่ใช้จริงวันนี้ · ศัพท์จำเป็น 6 คำ

จุดกำเนิด: โพสต์เดียวที่เปลี่ยนวงการ

“ปล่อยให้ AI เขียนโค้ดแทนทั้งหมด
จนไม่ต้องดูโค้ดเลย — แค่คุยกับมันตาม vibe”

— Andrej Karpathy (อดีตทีม AI ของ OpenAI / Tesla)
โพสต์ใน X · 2 ก.พ. 2025

ความหมายวันนี้ขยายกว้างขึ้น

“การสร้างแอป/เว็บด้วยภาษาพูด
คุยกับ AI — ไม่ว่าจะอ่านโค้ด
เป็นหรือไม่ก็ตาม”

⚠️ **ซื่อสัตย์ไว้ก่อน: Vibe Coding ไม่ใช่ยาวิเศษ — แม้แต่ Karpathy บางงานก็ยังกลับไปเขียนโค้ดเอง จบคลาสนี้คุณจะแยกออกว่างานไหน "ใช่"**

ไม่ใช่ hype ชั่วคราว — โลกรับรองแล้ว

“Vibe Coding”

Collins Dictionary — Word of the Year 2025

คำแห่งปีจากพจนานุกรมระดับโลก สะท้อนว่ากระแส "คนธรรมดาสร้างแอปเองด้วย AI" กลายเป็นเรื่องปกติใหม่ ไม่ใช่ของเล่นสายเทคอีกต่อไป

แปลว่าอะไรสำหรับคุณ?

- ทักษะนี้เรียนได้เลย ไม่ต้องรอ
- เครื่องมือฟรีมีให้ครบทุกชั้น
- คนที่เริ่มก่อน ได้เปรียบก่อน

ศัพท์จำเป็น 6 คำ (เจอตลอดวันนี้)

Prompt

คำสั่งที่พิมพ์บอก AI ว่าต้องการอะไร

เว็บไซต์ vs เว็บแอป

หน้าโชว์ข้อมูล vs ของที่กดโต้ตอบ/ใช้งานซ้ำได้

Repo

กล่องเก็บโค้ดทั้งหมดของโปรเจกต์

API key

กุญแจส่วนตัวไว้เรียกใช้บริการภายนอก — ห้ามแชร์ใคร

Credit

เหรียญ/โควตาที่เครื่องมือให้ใช้ หมดแล้วรอ/จ่ายเพิ่ม

Deploy

เอาแอปขึ้นออนไลน์จริงให้คนอื่นเข้าใช้

จำไม่หมดไม่เป็นไร — ครบทุกคำ ใน Handout "คำศัพท์ Vibe Coding ฉบับกระเป๋า"

รู้จักคำว่า Vibe Coding แล้ว —
ที่นี้มาผ่าดูว่าแอป 1 ตัว จริงๆ ประกอบด้วยอะไรบ้าง

02

องค์ประกอบสำคัญของเว็บ/แอป

ผ่าแอป 1 ตัวออกเป็น 8 ส่วน — เทียบกับ "ร้านอาหาร" 1 ร้าน

Frontend / UI

คืออะไร

ส่วนที่ผู้ใช้เห็นและกดในเบราว์เซอร์/แอป — ปุ่ม ฟอร์ม หน้าตา สี

จำง่ายๆ

ลูกค้าเห็นและสัมผัสแค่ส่วนนี้ แต่ร้านไม่ได้มีแค่นี้

เทียบร้านอาหาร

ห้องอาหาร + เมนู

Backend

คืออะไร

โค้ดที่รันบนเซิร์ฟเวอร์ คอยประมวลผล ตรวจเงื่อนไข คำนวณ เชื่อมฐานข้อมูล

จำง่ายๆ

สั่ง AI ว่า "ให้ผู้ใช้กดรีวิวได้" = กำลังสั่งห้องครัว ไม่ใช่หน้าร้าน

เทียบร้านอาหาร

ห้องครัว

Database

คืออะไร

ที่เก็บข้อมูลถาวร — บัญชีผู้ใช้ โพสต์ ออเดอร์ สินค้า

จำง่ายๆ

ปิดแอปแล้วเปิดใหม่ข้อมูลยังอยู่ เพราะส่วนนี้

เทียบร้านอาหาร

คลังวัตถุดิบ / สต็อก

Authentication (Auth)

คืออะไร

ระบบยืนยันตัวตน — ล็อกอิน/สมัคร "Sign in with Google" + คุมสิทธิ์ว่าใครทำอะไรได้

จำง่ายๆ

เช็คบัตรก่อนเข้าร้าน แยกโซน VIP

เทียบร้านอาหาร

พนักงานต้อนรับเช็คชื่อ

API / Integrations

คืออะไร

ตัวกลางส่งคำขอระหว่าง Frontend ↔ Backend หรือเชื่อมบริการภายนอก (จ่ายเงิน แผนที่ อีเมล)

จำง่ายๆ

วิ่งรับออเดอร์เข้าครัว แล้วเสิร์ฟกลับมา

เทียบร้านอาหาร

บ๋อยรับ-ส่งออเดอร์

Cloud

คืออะไร

บริการ "เช่าใช้" ฐานข้อมูล/ระบบ auth/โครงสร้างพื้นฐาน โดยไม่ต้องดูแลเซิร์ฟเวอร์เอง

จำง่ายๆ

ไม่ต้องสร้างโรงงานเอง — เช่าเอา

เทียบร้านอาหาร

ซัพพลายเออร์ส่งของสำเร็จรูป

Hosting

คืออะไร

เครื่องที่ทำให้แอปออนไลน์ตลอด 24 ชม. — ที่ "วาง" ให้แอปรันอยู่จริง

จำง่ายๆ

ไม่มีตึก = ไม่มีที่เสิร์ฟลูกค้า ต่อให้อาหารอร่อยแค่ไหน

เทียบร้านอาหาร

ตัวอาคารร้าน

Domain / DNS

คืออะไร

ชื่อที่มนุษย์อ่านได้ (botplease.com) + ระบบแปลงชื่อเป็นที่อยู่จริงของเซิร์ฟเวอร์

จำง่ายๆ

บอกทาง ให้คนหาร้านเจอ

เทียบร้านอาหาร

ป้ายชื่อร้าน + ที่อยู่

จุดที่มักพลาด #0.5

"เว็บสวย" ≠ "แอปเสร็จ"

- ✗ เว็บสวยคือแค่ Frontend — ห้องอาหารส่วนเดียว
- ✗ ถ้า Backend/Database ยังไม่เชื่อม กดปุ่มไปก็ไม่มีอะไรเกิดขึ้นจริง
- ✗ ทางแก้: "กดทดสอบจริง" ทุกครั้งก่อนเชื่อว่าเสร็จ (= ขั้นสั่งตรวจสอบ)

หมายเหตุสำคัญ

Cloud กับ Hosting ทับซ้อนกันได้ — จำ "หน้าที่" ไม่ใช่ "ยี่ห้อ"

Cloud	ระบบหลังบ้าน/โครงสร้างพื้นฐานที่เช่าใช้ (เช่น Supabase, AWS)
Hosting	จุดที่วางแอปให้ออนไลน์จริง (เช่น Vercel, Netlify, Cloudflare)
ในชีวิตจริง	บางเจ้าทำทั้งสองอย่าง (Google Cloud, Cloudflare, Supabase) — ไม่ต้องเถียงหมวด ให้เข้าใจหน้าที่พอ

ถัดไป → แต่ละส่วนมีเครื่องมืออะไรทำหน้าที่นั้นบ้าง (และตัวไหนฟรีจริง)

03

เครื่องมือแยกตามชั้น

Frontend → Backend → Fullstack → Cloud → Hosting · โชว์ไฮไลต์ 2-3 ตัว/หมวด — ตารางเต็ม 20+ ตัวอยู่ใน Handout

กฎเหล็กก่อนดูตาราง: ตัวเลข "หมดอายุ" ได้เสมอ

1

ราคา/โควตาฟรี เปลี่ยนบ่อยมาก

ตัวเลขบนสไลด์ = ณ วันที่เช็คล่าสุด (ก.ค. 2026) เปิดหน้า pricing ก่อนใช้เสมอ

2

สิ่งที่ควรจำไม่ใช่ตัวเลข แต่คือ "หมวด"

เครื่องมือไหนทำหน้าที่ชั้นไหน — แผนที่นี้ใช้ได้อีกหลายปี

3

รายละเอียดเต็มอยู่ใน Handout

Tool Stack Cheat Sheet — 20+ เครื่องมือ พร้อมลิงก์หน้าราคาทางการ

Frontend Tools — ออกแบบหน้าตาด้วย AI

เน้นสร้าง/ออกแบบ UI — เหมาะกับงานที่ "หน้าตา" คือพระเอก

v0 (Vercel)

ฟรีได้แค่ไหน

$0 — เครดิต $5/เดือน + จำกัด 7 ข้อความ/วัน ชนลิมิตเร็วตอนฝึก

จ่ายเงินเริ่มที่

Team $30/user/เดือน (เครดิต $30 + เครดิตรายวัน $2)

Framer (Framer)

ฟรีได้แค่ไหน

ฟรี — AI credits ~500/วัน (เพดานรวม ~1,000/เดือน) ใช้ subdomain ฟรี ผูกโดเมนเองไม่ได้

จ่ายเงินเริ่มที่

Basic ~$10/เดือน (รายปี) ได้ custom domain

ใช้ในคลาสนี้

Google Stitch (Google Labs)

ฟรีได้แค่ไหน

ฟรีทั้งหมด (สถานะ Beta) — พิมพ์/วาด/พูด ได้ดีไซน์ UI + export โค้ด

จ่ายเงินเริ่มที่

ยังไม่มีแผนจ่ายเงิน (เช็คแล้วไม่มีหน้า pricing)

Claude Design (Anthropic)

ฟรีได้แค่ไหน

ไม่มีในแผนฟรี — ใช้ได้เฉพาะ Pro/Max/Team (สถานะ Beta)

จ่ายเงินเริ่มที่

รวมในแผน Claude Pro ปกติ ไม่คิดเงินแยก

ตัวเลขตรวจสอบจากหน้าทางการ ณ ก.ค. 2026 — เช็คซ้ำก่อนสอนเสมอ

Backend-as-a-Service — ครัวสำเร็จรูป

ฐานข้อมูล + auth + server logic โดยไม่ต้องดูแลเซิร์ฟเวอร์เอง — จำ 2 ชื่อนี้พอสำหรับวันนี้

Supabase (Supabase)

ฟรีได้แค่ไหน

ฟรี — Postgres 500MB · auth 50,000 MAU · 2 โปรเจกต์ · ⚠️ ไม่ใช้งาน 1 สัปดาห์โดน pause

จ่ายเงินเริ่มที่

Pro $25/เดือน — ไม่ pause, disk 8GB, backup รายวัน

ใช้ในคลาสนี้

Firebase (Google)

ฟรีได้แค่ไหน

แผน Spark ฟรี — Auth 50,000 MAU · Firestore 1GiB + 50k reads / 20k writes ต่อวัน · ⚠️ เกินโควตา ระบบส่วนนั้นหยุดถึงสิ้นวัน

จ่ายเงินเริ่มที่

แผน Blaze จ่ายตามใช้จริง ไม่มีค่าคงที่รายเดือน

ตัวเลขตรวจสอบจากหน้าทางการ ณ ก.ค. 2026 — เช็คซ้ำก่อนสอนเสมอ

Fullstack — ทำครบทุกชั้นในที่เดียว (รวม AI Builder)

"AI Builder" = Fullstack แบบอัตโนมัติเต็มรูปแบบ: คุยแล้วได้แอปเลย ไม่ต้องแตะ terminal

Lovable (Lovable Labs)

ฟรี 5 เครดิต/วัน (สูงสุด 30/เดือน) ไม่ต้องผูกบัตร

Pro $25/เดือน (100 เครดิต)

Base44 (Wix)

ฟรี 25 ข้อความ/เดือน สร้างได้ 5 แอป ไม่ต้องผูกบัตร

Starter $16/เดือน (รายปี)

Google AI Studio (Google)

ใช้ในคลาสนี้

Build mode ฟรีทั้งหมด ใช้โควตา Gemini API ฟรี ไม่ต้องผูกบัตร

ไม่มีแผนรายเดือน — เกินโควตาจ่ายตามใช้

Replit (Replit)

สายประกอบเอง — Agent ช่วยเขียน มีฐานข้อมูลในตัว

ฟรี: เครดิต Agent รายวัน (ไม่ประกาศตัวเลข) + publish ได้ 1 โปรเจกต์ · Core $20/เดือน (รายปี)

Google Sheets + Apps Script (Google)

คนละผลิตภัณฑ์กัน: Sheets = สเปรดชีตเก็บข้อมูล · Apps Script = ตัวเขียนออโตเมชัน/โค้ดบน Workspace — ใช้คู่กันเป็นระบบบันทึกข้อมูลเบาสุด

ฟรี 100% — Apps Script ลิมิตรัน 6 นาที/ครั้ง · trigger รวม 90 นาที/วัน (บัญชีฟรี)

ตัวเลขตรวจสอบจากหน้าทางการ ณ ก.ค. 2026 — เช็คซ้ำก่อนสอนเสมอ

Cloud — โครงสร้างพื้นฐานที่ "เช่าใช้"

ระบบหลังบ้านระดับกว้าง — Firebase คือตัวที่คลาสนี้ใช้จริง (เชื่อมกับ Google AI Studio)

Supabase

ฟรีได้แค่ไหน

(ตัวเดียวกับหมวด Backend — ทำหน้าที่ cloud infrastructure ด้วย)

จ่ายเงินเริ่มที่

จุดเริ่มที่เหมาะกับมือใหม่สุดในหมวดนี้

ใช้ในคลาสนี้

Firebase (Google)

ฟรีได้แค่ไหน

แผน Spark ฟรี — Auth 50,000 MAU · Firestore 1GiB + 50k reads/20k writes ต่อวัน

จ่ายเงินเริ่มที่

แผน Blaze จ่ายตามใช้จริง ไม่มีค่าคงที่รายเดือน

Amazon AWS

ฟรีได้แค่ไหน

บัญชีใหม่ได้เครดิตทดลองสูงสุด $200 / 6 เดือน

จ่ายเงินเริ่มที่

⚠ ซับซ้อน + ต้องผูกบัตร — ไว้โตแล้วค่อยไป

คำแนะนำตรงๆ: มือใหม่เริ่มจาก Supabase/Firebase ก่อน — AWS/Google Cloud/Azure คือด่านของสาย Developer

Hosting — ที่วางแอปให้ออนไลน์จริง

วันนี้เราไม่ deploy กัน — แต่รู้จักชื่อไว้ ถึงวันที่พร้อมค่อยกลับมาเปิดตารางนี้

Vercel

ฟรีได้แค่ไหน

Hobby ฟรี — deploy อัตโนมัติ + CDN ทั่วโลก

จ่ายเงินเริ่มที่

ตัวแรกที่ควรลองเมื่อพร้อม deploy

Netlify

ฟรีได้แค่ไหน

ฟรี 300 เครดิต/เดือน (build + bandwidth + forms)

จ่ายเงินเริ่มที่

Personal $9/เดือน

Railway

ฟรีได้แค่ไหน

เครดิตทดลอง $5 (30 วัน) ไม่ต้องผูกบัตร

จ่ายเงินเริ่มที่

Hobby ขั้นต่ำ $5/เดือน

Cloudflare Pages

ฟรีได้แค่ไหน

bandwidth ฟรีไม่จำกัด + build 500 ครั้ง/เดือน

จ่ายเงินเริ่มที่

Workers ฟรี 100k requests/วัน

ตัวเลข ณ ก.ค. 2026 — เช็คหน้า pricing ก่อนใช้เสมอ

มีแผนที่เครื่องมือครบแล้ว —
แต่จะสั่งงานให้แม่น ต้องรู้ "ศัพท์เรียกชิ้นส่วน" ของหน้าเว็บ/แอปก่อน

04

ส่วนประกอบ "เว็บไซต์" vs "แอป"

คำศัพท์ชี้จุดตอนสั่ง AI — สองประเภทนี้โครงไม่เหมือนกัน ห้ามปนกัน

จุดต่างสำคัญ

เว็บไซต์ ≠ แอป — เป้าหมายการใช้งานคนละแบบ

เว็บไซต์ (Website)

ออกแบบให้ "ดูจบใน 1 ครั้ง แล้วตัดสินใจ"

- เป้าหมาย: โน้มน้าวให้ซื้อ/สมัคร/ติดต่อ
- โครงตายตัว: Hero → Feature → CTA
- ตัวอย่าง: เว็บแบรนด์ หน้าร้านขายของ
- ศัพท์เรียกหน้า: Page > Section

📱 แอป (App)

ออกแบบให้ "กลับมาใช้ซ้ำเป็น loop"

- เป้าหมาย: ให้เปิดใช้ซ้ำทุกวัน
- หัวใจ: Home → Core Screen → Notification
- ตัวอย่าง: แอปบันทึกรายจ่าย นับก้าว เกม
- ศัพท์เรียกหน้า: Screen > Block

⚠️ **สั่ง AI สร้าง "แอป" แต่ใช้ศัพท์ของ "เว็บไซต์" (หรือกลับกัน) = ได้โครงผิดประเภทตั้งแต่ต้น**

A) เว็บไซต์

โครงมาตรฐาน 8 ส่วน — ไล่จากบนลงล่าง

- Header / Navbar
- **Hero Section**
- Feature / Detail
- Testimonial
- Pricing
- **CTA**
- FAQ
- Footer

Hero	บล็อกแรกเต็มจอ — บอกใน 3 วินาทีว่าคุณขาย/ทำอะไร
Feature	การ์ดจุดเด่น 3 ใบ — ทำไมต้องเลือกคุณ
Testimonial	รีวิวลูกค้าจริง — สร้างความเชื่อก่อนตัดสินใจ
CTA	ปุ่มกระตุ้นตัดสินใจ — "สมัครฟรีวันนี้"
FAQ	ตอบคำถามค้างใจ — ลดเหตุผลที่จะไม่ซื้อ

เปิดเว็บโปรดของคุณตอนนี้ — จะเห็นโครงนี้เกือบทุกเว็บ

B) แอป

โครงมาตรฐาน 10 ส่วน — ออกแบบให้กลับมาใช้ซ้ำ

Splash / Onboarding	สไลด์แนะนำ 3 หน้าแรก — แอปนี้ทำอะไร ใช้ยังไง
Feed / List + Detail	รายการข้อมูลย้อนหลัง + หน้าเจาะดูทีละรายการ
Login (Auth)	ผูกตัวตนผู้ใช้กับข้อมูล — "เข้าสู่ระบบด้วย Google"
Notification	แจ้งเตือนดึงกลับมาใช้ — "อย่าลืมบันทึกวันนี้"
Home / Dashboard	หน้าหลักหลังล็อกอิน — สรุปภาพรวม/สถานะล่าสุด
Profile / Settings	จัดการข้อมูลส่วนตัว ตั้งค่าการใช้งาน
Bottom Navigation	แถบสลับเมนูด้านล่าง (ต่างจากเว็บที่เมนูอยู่บน)
Empty State	หน้าตอนยังไม่มีข้อมูล — มือใหม่ลืมทำบ่อยสุด!

ยังขาดอีก 1 ส่วนที่สำคัญที่สุด → หน้าถัดไป

หัวใจของทุกแอป

Core Interactive Screen

หน้าที่ผู้ใช้ "ลงมือทำ" ฟีเจอร์หลักของแอปจริงๆ — เหตุผลเดียวที่ทำให้คนเปิดแอปซ้ำ

แอปรายจ่าย	แอป Habit	แอปเกม
หน้ากดบันทึกรายจ่าย	หน้ากดเช็คอินวันนี้	หน้าเล่นเกม

ถ้าไม่มีเหตุผลให้เปิดหน้านี้ซ้ำ = แอปตาย · ตอน Hands-on ให้ถามตัวเองก่อน: "Core Screen ของแอปฉันคือหน้าไหน?"

ได้ "พิกัด" เรียกชิ้นส่วนครบแล้ว —
ต่อไปคือช่วงสำคัญที่สุดของวันนี้: สูตรการสั่งงาน **AI**

05

Framework การสั่งงาน AI

Loop 3 ขั้น · RCTC · P-S-C (Frontend) · E-F-R-A (Backend) — ช่วงที่ห้ามหลุด

FRAMEWORK หลัก

Loop 3 ขั้น: ทั้งชีวิต Vibe Coding วนอยู่ในนี้

1. สั่งสร้าง (Build)

จัดโครงคำสั่งด้วย RCTC
ให้ AI สร้างเวอร์ชันแรก

→

2. สั่งแก้ไข (Refine)

แก้ทีละจุด ระบุพิกัด
P-S-C หรือ E-F-R-A

→

3. สั่งตรวจสอบ (Verify)

กดใช้งานจริงทุกปุ่ม
ก่อนเชื่อว่าเสร็จ

วนซ้ำจนพอใจ = Growth Loop: ทดลอง → ดูผล → ปรับ → ดีขึ้น

ขั้นที่คนข้ามบ่อยสุดคือขั้น 3 — และนั่นคือที่มาของคำว่า "AI ทำพัง" ทั้งที่จริงคือ "ไม่เคยตรวจ"

RCTC — โครงคำสั่งประจำแบรนด์ Bot Please

R — Role	บอก AI ว่าให้สวมบทเป็นใคร	*"คุณคือนักออกแบบแอปมือถือ"*
C — Context	บริบทธุรกิจ/ผู้ใช้ของเรา	*"ผู้ใช้คือเจ้าของร้านกาแฟ ไม่ถนัดเทคโนโลยี"*
T — Task	สิ่งที่ต้องทำ ชัดๆ วัดได้	*"สร้างแอปบันทึกออเดอร์ มี 3 หน้าจอ"*
C — Criteria	เกณฑ์ตัดสินว่างานเสร็จสมบูรณ์	*"ใช้ภาษาไทยทั้งแอป โทนสีเขียว ไม่ต้องมีล็อกอิน"*

Template เปล่า + ตัวอย่างเต็มอยู่ใน Handout "RCTC + 3-ขั้น Prompt Sheet"

P-S-C — ระบุพิกัดเหมือนบอก "ตึก → ชั้น → ห้อง"

Page / Screen

อยู่หน้าไหน

"หน้าแรก" / "Home Screen"

Section / Block

บล็อกใหญ่ในหน้านั้น

"Hero Section" / "การ์ดสรุปยอด"

Component

ชิ้นส่วนย่อยที่จะแก้

"ปุ่ม CTA" / "ช่องกรอกอีเมล"

เว็บไซต์ใช้คำว่า Page > Section · แอปใช้ Screen > Block — เลือกชุดศัพท์ให้ตรงประเภทที่กำลังสร้าง

เทคนิคเสริม: แนบ screenshot วงกลม/ลูกศรชี้จุดที่จะแก้ ประกอบคำสั่งเสมอ

P-S-C ของจริง: สั่งแบบไหน AI ถึงไม่งง

สั่งลอยๆ

"เปลี่ยนสีปุ่ม ให้หน่อย"

ปุ่มไหน? หน้าไหน? — AI เดา แล้วแก้ผิดที่

สั่งแบบ P-S-C

"หน้าแรก > Hero Section > ปุ่ม CTA
→ เปลี่ยนเป็นสีเขียว"

ตรงจุดทันที ไม่มีแก้ผิดที่

เวอร์ชันแอป (Screen > Block > Component):

"Home Screen > การ์ดสรุปยอดเดือนนี้ > ปุ่มบันทึก → เปลี่ยนเป็นสีเขียว"

💡 ลองฝึก: แปลง "อยากให้รีวิวลูกค้าเด่นขึ้น" เป็นคำสั่งแบบ P-S-C ดูสิครับ

E-F-R-A — ออกแบบโครงข้อมูลก่อนสั่ง AI

E — Entity	"สิ่งที่ต้องเก็บ"	*ลูกค้า · ออเดอร์ · สินค้า*
F — Field	ข้อมูลย่อยของ Entity	*ชื่อ · เบอร์โทร · วันที่ · ราคา*
R — Relation	เชื่อมกับ Entity ไหน	*ลูกค้า 1 คน → มีหลายออเดอร์*
A — Action	ทำอะไรกับมันได้บ้าง	*เพิ่ม · ดู · แก้ · ลบ (CRUD)*

ตัวอย่างคำสั่งจริง

"Entity ลูกค้า > Field เบอร์โทร → บังคับกรอก ห้ามซ้ำ"

ไม่ใช่ "แก้ระบบลูกค้าหน่อย"

💡 **ก่อนสั่ง AI สร้างระบบ: เขียน E-F-R-A ลงกระดาษ 2 นาที ประหยัดเวลาแก้ครึ่งชั่วโมง (Template ใน Handout)**

ก่อนสั่งทุกครั้ง ถามตัวเอง: "กำลังสั่งชั้นไหน?"

Frontend | **Backend** | **Database** | **Auth** | **API** | **Cloud** | **Hosting**

"เพิ่มระบบล็อกอิน"	→ กำลังสั่งชั้น Auth (+ Frontend ฟอร์ม)
"ทำให้โหลดเร็วขึ้น"	→ อาจเป็น Frontend, Backend หรือ Hosting — ถาม AI ให้วิเคราะห์ก่อน
"เก็บประวัติการสั่งซื้อ"	→ Database (Entity ออเดอร์) + Backend (logic บันทึก)

รู้ว่าสั่งชั้นไหน = prompt แม่นขึ้น + debug ง่ายขึ้น

จุดที่มักพลาด #1

3 ท่าพลาดตอนสั่งงาน AI

✗ สั่งทุกอย่างในคำสั่งเดียวยาวเป็นพืด — AI ทำหลุดไปครึ่งหนึ่ง → แก้: สั่งทีละเรื่อง

✗ เชื่อผลลัพธ์ AI ทันที ไม่กดทดสอบ → แก้: ขั้น "สั่งตรวจสอบ" ห้ามข้าม

✗ ไม่ระบุพิกัด P-S-C / E-F-R-A → AI แก้ผิดจุด แล้วเราโทษว่า AI โง่ ทั้งที่เราสั่งไม่ชัดเอง

06

AI Builder vs Coding Agent

สองแนวทางของ Vibe Coding — เร็วแบบอัตโนมัติ vs คุมได้แบบประกอบเอง

สองแนวทาง

เลือกตามงาน ไม่ใช่ตามกระแส

🤖 AI Builder

คุยแล้วได้แอปเลย ในเบราว์เซอร์

- รวม Frontend+Backend+Database+Hosting อัตโนมัติ
- ไม่ต้องแตะ terminal เลย
- เร็วมาก — เหมาะต้นแบบ/งานส่วนตัว
- แลกกับ: คุมรายละเอียดได้น้อยกว่า
- เช่น Lovable · Base44 · Google AI Studio

⌨️ Coding Agent

AI เขียน/แก้ไฟล์โค้ดจริงในเครื่องเรา

- ทำงานผ่าน terminal / IDE
- ยืดหยุ่น — เลือกประกอบเครื่องมือแต่ละชั้นเอง
- คุมได้ทุกจุด เห็นทุกไฟล์
- แลกกับ: ต้องมีพื้นเทคนิคนิดหน่อย
- เช่น Claude Code · Codex · Antigravity

วันนี้ได้เห็นทั้งสองแบบสดๆ ในช่วง **Live Demo**

CODING AGENT หลัก

3 ตัวที่ควรรู้จัก (ณ ก.ค. 2026)

Claude Code (Anthropic)

Terminal — มองเป็น IDE ตัวหนึ่งที่อยู่ในเทอร์มินัล

ฟรีได้แค่ไหน

ไม่มีในแผนฟรี — ใช้ได้เฉพาะแผน Pro กับ Max เท่านั้น

ChatGPT / Codex (OpenAI)

Terminal / CLI

ฟรีได้แค่ไหน

มี Free Tier ให้ใช้ — งานโค้ดพื้นฐาน (ลิมิตไม่ประกาศตัวเลข) แผน Plus ขึ้นไปได้โควตามากขึ้น

Demo วันนี้

Google Antigravity (Google)

IDE เต็มรูปแบบ + CLI

ฟรีได้แค่ไหน

ฟรีถาวรแผน Individual $0 — Tab completion + Command ไม่จำกัด ส่วนงาน Agent มี rate limit รายสัปดาห์ (อัปเกรดผ่าน Google AI Pro/Ultra)

⚠ **Gemini CLI ปิดบริการแล้ว (18 มิ.ย. 2026) — ถูกแทนที่ด้วย Antigravity CLI · ตรวจสอบจากหน้าทางการ ณ ก.ค. 2026**

CODING AGENT / IDE เพิ่มเติม

รู้จักชื่อไว้ — รายละเอียดครบใน Cheat Sheet

Cursor (Anysphere)

IDE (fork VS Code) — Hobby ฟรีแบบจำกัด (ไม่ประกาศตัวเลข) · Pro $20/เดือน

GitHub Copilot CLI (GitHub)

Copilot Free — 2,000 completions/เดือน + chat แบบจำกัด · Pro $10/เดือน

Factory Droid (Factory)

ไม่มีแผนฟรี · เริ่ม Pro $20/เดือน

OpenCode (โอเพนซอร์ส)

ฟรี — มีโมเดลฟรีในตัว หรือใช้ API key / สิทธิ์ Copilot·ChatGPT ที่มีอยู่

Grok Build (xAI)

ไม่มีแผนฟรี — ต้องสมัคร SuperGrok ($30/เดือน) หรือ X Premium+

Pi — Agent Harness (โอเพนซอร์ส)

ฟรี 100% แบบ bring-your-own-key — จ่ายเฉพาะค่าโมเดลที่เลือกใช้

ทุกตัวมีลิงก์หน้าราคาทางการใน Handout "Tool Stack Cheat Sheet" — เช็คก่อนสมัครเสมอ

เกณฑ์เลือกใช้

สูตรเลือกง่ายๆ 3 ข้อ

งานส่วนตัว / ต้นแบบเร็วๆ ไม่ซับซ้อน	→ **AI Builder**	*Lovable · Base44 · Google AI Studio*
งานคุมโค้ดละเอียด อยากเลือกเครื่องมือเอง	→ **Coding Agent + เครื่องมือแยกชั้น**	*เช่น Antigravity + Supabase + Vercel*
แค่ระบบลงบันทึกข้อมูลง่ายๆ	→ **Google Sheets + Apps Script ก็พอ**	*ฟรี 100% ไม่ต้องสมัครอะไรเพิ่ม*

คำถามที่ถูกไม่ใช่ "อันไหนดีกว่า" — แต่คือ "งานของคุณต้องการความเร็ว หรือความคุม"

07

Live Demo

Demo 1: เว็บร้านขายของออนไลน์ (AI Builder) · Demo 2: เว็บ "เส้นทางสู่เงินล้าน" (Coding Agent)

DEMO 1 · AI BUILDER

Google AI Studio — สร้าง "เว็บแคตตาล็อกร้านขายของออนไลน์"

สิ่งที่จะได้เห็นใน Demo นี้

- โจทย์: เว็บหน้าร้านแม่ค้าออนไลน์ — โครงตามตาราง A: Hero → สินค้า → รีวิว → CTA "ทักแชทสั่งเลย" → FAQ
- สั่งสร้าง — prompt แรกใช้ RCTC เต็มรูป (ดูทีละท่อน)
- สั่งแก้ไข — ระบุพิกัด Page > Section > Component
- สั่งตรวจสอบ — กดทุกปุ่มจริง เปิดบนมือถือ ดูว่าพังไหม

👀 จุดที่ควรสังเกต

รอบแรก AI สร้างได้กี่หน้าจอ?
Empty State มามั้ย?
อะไรที่ยังไม่ตรงใจ?

สังเกตไว้ — เดี๋ยวใช้เป็นโจทย์ตอน "สั่งแก้ไข" ต่อทันที

จุดที่มักพลาด #2

สั่งกว้าง ได้ผลกว้าง

- ✗ "ทำแอปให้สวย" / "ทำให้ดีขึ้น" — AI ตีความเอง ผลออกมาไม่ตรงใจ
- ✗ แก้: ใช้ RCTC ตอนสั่งสร้าง + ระบุพิกัด Section/Component ตอนสั่งแก้
- ✗ สังเกตใน Demo: ทุกคำสั่งของผมมีพิกัดเสมอ ไม่มีคำว่า "หน่อย" ลอยๆ

Google Antigravity — เว็บ "เส้นทางสู่เงินล้าน" 💰

สิ่งที่จะได้เห็น

- โจทย์: กรอกเงินตั้งต้น + ออม/เดือน + เลือกสินทรัพย์ + slider ปี → กราฟ 2D วิ่งไปหาเส้นเป้า 1,000,000 พร้อมป้าย "ถึงล้านแรกในอีก X ปี"
- Antigravity เขียน/แก้ไฟล์โค้ดจริงทีละไฟล์ — เราเห็นทุกอย่าง คุมดีไซน์ได้ละเอียดระดับที่ AI Builder ทำไม่ได้
- ดีไซน์ 2D flat โทนแบรนด์ + อะนิเมชันกราฟ/ตัวเลขวิ่ง (ไม่ใช้ 3D)
- ดูเป็น ไม่ต้องทำตาม — ตัวเลขผลตอบแทนสมมุติเพื่อการสอน ไม่ใช่คำแนะนำการลงทุน

เกร็ดน่าคิด

Demo ทั้งสองตัววันนี้เป็นของ Google เหมือนกัน — เจ้าเดียวกันทำทั้งสายอัตโนมัติเต็มรูปแบบ และสายคุมเองผ่านโค้ด

สะท้อนว่าทั้งสองแนวทางจะอยู่คู่กันไปอีกนาน

สรุปจาก **DEMO**

ความเร็ว vs ความคุม — สิ่งที่เพิ่งเห็นกับตา

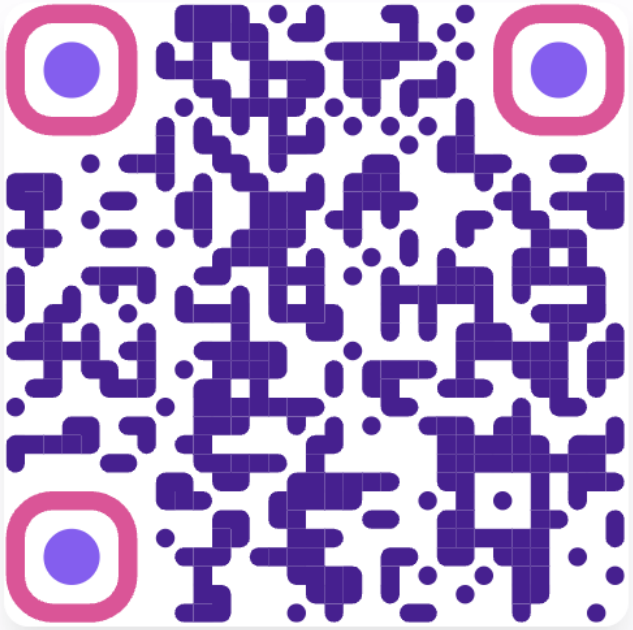
	AI Builder	Coding Agent
ความเร็วได้แอปแรก	AI Builder ชนะขาด — นาทีเดียวมีของ	Coding Agent ต้อง setup ก่อน
ความคุมรายละเอียด	แก้ได้เท่าที่เครื่องมือเปิดให้แก้	Coding Agent คุมได้ทุกไฟล์ทุกบรรทัด
ความรู้ที่ต้องมี	แค่พิมพ์ภาษาไทยเป็น	ต้องรู้จัก terminal/ไฟล์โค้ดพื้นฐาน
เหมาะกับ	ต้นแบบเร็ว งานส่วนตัว มือใหม่	งานจริงจัง อยากโตสายนี้ต่อ

ดูผมทำมาพอแล้ว — ถึงตาคุณ!

08

แนวทางลงมือสร้างแอปของตัวเอง

Guided Hands-on — เอาไอเดียจาก Worksheet มาสร้างจริงด้วย Google AI Studio



https://botplease.me/feedback/

FRAMEWORK หาไอเดีย

ยังคิดไม่ออกว่าจะสร้างอะไร? — P-S-W: เจ็บก่อน ค่อยสร้าง

0

Research ตลาดใกล้ตัว

ตลาดของแอปแรก = ตัวคุณ + คนรอบตัว 3-5 คน — งานไหนทำซ้ำแบบ manual ทุกวัน? ใครบ่นเรื่องอะไรบ่อย?

P

Pain — เลือกปัญหาที่เจ็บจริง

คัดด้วยสูตร บ่อย × เจ็บ: เกิดอย่างน้อยทุกสัปดาห์ + คนเจอบ่นเองโดยไม่ต้องถามนำ

S

Solution — สรุปวิธีแก้ใน 1 ประโยค

"แอป/เว็บที่ช่วยให้ [ใคร] ทำ [อะไร] ได้โดยไม่ต้อง [วิธี manual เดิม]" — เขียนไม่ได้ = ปัญหายังไม่ชัด

W

Why switch? — คำถามฆ่าไอเดีย

เขาทำ manual มาได้ทุกวันอยู่แล้ว ทำไมต้องเปลี่ยนมาใช้ของคุณ? ตอบไม่ได้ = กลับไปข้อ P

ใช้ก่อนพิมพ์ prompt แรกเสมอ — Product คือ Solution ไม่ใช่ของโชว์ · กรอกตามฟอร์มใน App Idea Worksheet

คำถามฆ่าไอเดีย (W)

"ทำไมเขาต้องเลิกทำแบบเดิม มาใช้ของคุณ?"

คำตอบต้องเป็นรูปธรรม อย่างน้อย 1 ใน 3 ข้อนี้:

เร็วกว่าเห็นๆ

จาก 30 นาที เหลือ 3 นาที

หายห่วง

ข้อมูลไม่หาย ไม่ลืม ไม่ตกหล่น — สิ่งที่กระดาษให้ไม่ได้

เห็นสิ่งที่เดิมมองไม่เห็น

ยอดรวม กราฟ แนวโน้ม ที่วิธี manual สรุปให้ไม่ได้

ตัวอย่าง: เพื่อนขายเสื้อ IG ลืมจดออเดอร์ (P) → "แอปจดออเดอร์+สถานะส่ง ไม่ต้องไล่แชทย้อนหลัง" (S) → หายห่วง + เห็นยอดขายรายวัน (W) ✓ สร้างได้

ขั้นตอนการทำงาน

1 **เปิด Google AI Studio**

ตาม Quick Guide หน้า 1 (ลิงก์ในแชท) — ใช้บัญชี Google ปกติ

2 **ใช้โจทย์จาก App Idea Worksheet (ผ่าน P-S-W แล้ว)**

ยังไม่มีไอเดีย? โจทย์สำรอง: แอปรายรับ-รายจ่าย หรือ แอปหลังบ้านแม่ค้า (จดออเดอร์+สถานะส่ง)

3 **ทำครบ Loop 3 ขั้น**

สั่งสร้าง (RCTC) → สั่งแก้ไข (ทีละจุด + พิกัด) → สั่งตรวจสอบ (กดจริงทุกปุ่ม)

4 **ติดตรงไหน ถามใน Chat**

แนบ screenshot + บอกว่าติดขั้นไหน (สร้าง/แก้/ตรวจ) — ผมไล่ตอบตลอด

เป้าหมายช่วงนี้: ทำครบ Loop 3 ขั้น — มีหน้าแรก → สั่งแก้ไขอย่างน้อย 2 รอบ → สั่งตรวจสอบก่อนจบ

จุดที่มักพลาด #3

ก่อนโทษเครื่องมือ — เช็ค prompt ตัวเองก่อน

- ✗ แอปไม่เป็นแบบที่คิด? 8 ใน 10 ครั้ง ปัญหาอยู่ที่คำสั่งเราไม่ชัด ไม่ใช่ AI
- ✗ เปิด prompt ตัวเองอ่านซ้ำ: มี RCTC ครบไหม? ระบุพิกัดหรือยัง?
- ✗ นี่คือทักษะที่แยก "คนใช้ AI เก่ง" กับ "คนใช้ AI ไม่เก่ง" — ฝึกได้ตั้งแต่วันนี้

Screenshot แอปของคุณ ส่งในแชทเลย!

อวดผลงานแรกของคุณให้เพื่อนร่วมคลาสดู

นี่คือแอปแรกในชีวิตของหลายคนในห้องนี้ —
เมื่อ 3 ชั่วโมงที่แล้ว คุณยังคิดว่าต้องเขียนโค้ดเป็นถึงจะทำได้

09

Wrap-up + Homework

เช็คลิสต์สิ่งที่ได้ · การบ้าน 7 วัน · ก้าวถัดไป

WRAP-UP

เช็คลิสต์: วันนี้คุณได้อะไรกลับบ้าน

- ✓ มีแอป 1 ตัวที่ใช้งานได้จริง (หลักฐาน: screenshot ที่เพิ่งส่ง)
- ✓ มองทะลุองค์ประกอบเว็บ/แอป 8 ส่วน + ส่วนประกอบเว็บไซต์ vs แอป
- ✓ รู้จักเครื่องมือแยกชั้น 5 หมวด + AI Builder vs Coding Agent
- ✓ ใช้ Framework เป็น: Loop 3 ขั้น · RCTC · P-S-C · E-F-R-A
- ✓ รู้ว่าก้าวถัดไปคืออะไร: deploy + ประกอบเครื่องมือเอง

จำคำถามแรกของวันได้ไหม — "อะไรหยุดคุณไว้?" … ตอนนี้คำตอบนั้นใช้ไม่ได้แล้ว

HOMEWORK 7 วัน

แอปที่สองของคุณ เริ่มพรุ่งนี้

Day 1	ตั้งโจทย์แอปตัวที่ 2 (ใช้ App Idea Worksheet)
Day 2–3	สร้างด้วย Framework ครบ Loop 3 ขั้น
Day 4	ลองสลับเครื่องมืออีกสาย — ใช้ AI Builder อยู่ ลอง Coding Agent
Day 5–6	ปรับปรุงจนพอใจ (Growth Loop: ทดลอง→ ดูผล→ ปรับ)
Day 7	สรุปสิ่งที่เรียนรู้ + โพสต์แชร์ผลงาน แท็ก Bot Please — ผมรีวิวให้ทุกคน

ติ๊กตามได้ใน Handout "7-Day Homework Tracker" — ทำครบ 7 วัน = แอปตัวที่ 2 + ความมั่นใจที่ไม่ต้องพึ่งคลาสอีก

ขอบคุณครับ 🤖

ผมอยู่ต่ออีกสักครู่สำหรับคำถามค้าง — ถามได้เลยครับ

อยากไปต่อ?

โปรดติดตามคอร์สถัดไป หรือ Consult กับ Bot Please

REFERENCES

แหล่งอ้างอิงหลัก (1/2) — ที่มา + AI Builder + Coding Agent

ที่มาของคำ "Vibe Coding"

Andrej Karpathy (วิเคราะห์โดย Simon Willison): simonwillison.net/2025/Mar/19/vibe-coding
ประวัติศัพท์: coderabbit.ai/blog/a-semantic-history-...-vibe-coding
Collins Dictionary Word of the Year 2025

AI Builder

Lovable: lovable.dev/pricing
Base44: base44.com/pricing
Google AI Studio Build mode: ai.google.dev/gemini-api/docs/aistudio-build-mode

Coding Agent หลัก

Claude Code (Anthropic): claude.com/pricing
ChatGPT / Codex (OpenAI): developers.openai.com/codex/pricing
Google Antigravity: antigravity.google/pricing
Gemini CLI → Antigravity CLI (18 มิ.ย. 2026): developers.googleblog.com

Coding Agent / IDE เพิ่มเติม

Cursor: cursor.com/pricing · Copilot: github.com/features/copilot/plans
Factory Droid: factory.ai/pricing · OpenCode: opencode.ai
Grok Build: x.ai/news/grok-build-cli · Pi: pi.dev

ข้อมูลราคา/โควตาทั้งหมดตรวจสอบ ณ ก.ค. 2026 — โปรดเช็คหน้า pricing ทางการก่อนใช้เสมอ

REFERENCES

แหล่งอ้างอิงหลัก (2/2) — เครื่องมือแยกตามชั้น

Frontend

v0 by Vercel: v0.app/pricing
Framer: framer.com/pricing
Claude Design: claude.com/product/design
Google Stitch: stitch.withgoogle.com

Backend-as-a-Service

Supabase: supabase.com/pricing
Firebase: firebase.google.com/pricing
Google Apps Script quotas: developers.google.com/apps-script/guides/services/quotas

Cloud

AWS Free Tier: aws.amazon.com/free
Cloudflare Workers/Pages: developers.cloudflare.com/workers/platform/pricing

Hosting

Vercel: vercel.com/pricing · Netlify: netlify.com/pricing
Railway: railway.com/pricing
Google Cloud Free Tier: cloud.google.com/free

ข้อมูลราคา/โควตาทั้งหมดตรวจสอบ ณ ก.ค. 2026 — โปรดเช็คหน้า pricing ทางการก่อนใช้เสมอ