คู่มือผู้เรียน (Student Manual)

หลักสูตร: Vibe Coding & Modern App Development with Antigravity

บทนำ (Introduction)

ภาพรวมหลักสูตร

หลักสูตรนี้ออกแบบมาเพื่อเปลี่ยนวิธีการพัฒนาซอฟต์แวร์จาก "Coding by Syntax" (การเขียนโค้ดทีละบรรทัด) เป็น "Coding by Intent" (การเขียนโค้ดด้วยเจตนา)

ผ่านแนวคิด **Vibe Coding** โดยใช้เครื่องมือ **Antigravity** ร่วมกับ AI (LLMs) เพื่อให้คุณสามารถสร้าง Application ได้จริง รวดเร็ว และมีประสิทธิภาพ

หลักสูตรนี้เหมาะสำหรับ

- **Product Owner (PO) / System Analyst (SA):** ที่ต้องการเข้าใจวิธีการใช้ AI เพื่อเปลี่ยน Requirement เป็น Technical Spec และ Prototype
- **Developer (Dev) / System Engineer (SE):** ที่ต้องการเพิ่มความเร็วในการพัฒนา (Velocity) และเรียนรู้วิธีการทำงานร่วมกับ AI ในฐานะผู้สั่งการ

พื้นฐานที่ควรมี (Prerequisites)

เพื่อให้การเรียนเป็นไปอย่างราบรื่น คุณควรมีพื้นฐานดังนี้:

1. **Web Fundamentals:** เข้าใจพื้นฐาน HTML, CSS และการทำงานของ Web Browser เบื้องต้น
2. **Basic Programming Logic:** เข้าใจ Concept ของตัวแปร (Variables), เงื่อนไข (If/Else), และการวนซ้ำ (Loops)
3. **Basic Database:** เข้าใจ Concept ของ Database (Table, Column, Row, PK, FK)
4. **Terminal Basics:** คุ้นเคยกับการใช้ Command Line เบื้องต้น (`cd`, `ls`, `npm install`)
5. **Docker:** คุ้นเคยกับการใช้ Docker เบื้องต้น
6. **Open Mindset:** พร้อมที่จะเรียนรู้วิธีการทำงานรูปแบบใหม่ (AI-Driven)

เครื่องมือที่ต้องเตรียม (Required Tools)

- **Antigravity:** IDE/Platform หลักสำหรับการพัฒนา
- **Docker Desktop:** สำหรับ run เครื่องมือที่ใช้ในการพัฒนาอื่นๆ
- **Git:** สำหรับจัดการ Version Control

บัญชีที่ต้องเตรียม (Required Accounts)

ควรสมัครบัญชีเหล่านี้ให้พร้อมก่อนเริ่มเรียน:

1. **GitHub:** สำหรับเก็บ Source Code
2. **Vercel:** สำหรับ Deploy Frontend (Login with GitHub ได้)
3. **Neon:** สำหรับ Cloud Database (PostgreSQL)
4. **AI Provider (Optional):** OpenAI Platform หรือ Anthropic Console (กรณีต้องการใช้ API Key ส่วนตัว)

Technology Stack ที่ใช้ใน Workshop

- **Database:** PostgreSQL (via Neon)
- **LLMs:** Gemini 3 Pro / Claude Sonnet 4.5 / GPT-4o
- **Stack:** Next.js (Fullstack), Prisma (ORM)
- **Infrastructure:** Docker

Day 1: The Foundation & The Blueprint (จากไอเดียสู่โครงสร้าง)

เป้าหมายประจำวัน

- เตรียมสภาพแวดล้อมการทำงาน (Environment Setup)
- เข้าใจ Concepts ของ Vibe Coding และ AI-Driven Development
- ฝึกฝนการทำ Requirements Engineering ร่วมกับ AI (PRD & UI Spec)

ช่วงเช้า: Introduction & Setup

1. Concept of Vibe Coding & AI-Driven Dev

The Evolution of Development

- **ยุคที่ 1: Machine Code & Assembly** - สั่งงาน Hardware โดยตรง
- **ยุคที่ 2: High-Level Languages** - สั่งงานด้วย Logic (C, Java, Python)
- **ยุคที่ 3: AI-Driven / Vibe Coding** - สั่งงานด้วย **"เจตนา" (Intent)** และ **"บริบท" (Context)**

Core Concept: "Vibe Coding" คือการรักษา **"Flow"** ของการทำงาน ไม่ให้สะดุดด้วย Syntax Error หรือการค้นหา Document เน้นการมองภาพรวม (Big Picture) และ Logic ของระบบ มากกว่ารายละเอียดปลีกย่อย

บทบาทใหม่ของคุณ: "The Director" เราไม่ได้เขียน Code เองทุกบรรทัด แต่เราคือ **"ผู้กำกับ"** ที่มี AI เป็นทีมงานมืออาชีพ หน้าที่ของคุณคือ:

1. **Clear Brief:** สั่งงานให้ชัดเจน (Prompting)
2. **Context Management:** ให้ข้อมูลที่จำเป็นแก่ AI (Files, Docs, Rules)
3. **Review & Refine:** ตรวจสอบผลลัพธ์และสั่งแก้

ตัวอย่าง: การสร้างปุ่ม

- *Traditional:* เขียน HTML/CSS/JS เอง (ใช้เวลา 5-10 นาที)
- *Vibe Coding:* พิมพ์ Prompt "Create a blue button that turns red when clicked." (ใช้เวลา 30 วินาที)
- *ข้อคิด:* เวลาที่ประหยัดได้ สามารถนำไปโฟกัสที่ Business Logic, UX, และ Quality

2. Environment & Tool Installation

Checklist การติดตั้ง:

1. **Antigravity:** ติดตั้งและเปิดใช้งานได้ พร้อม Extensions
2. **Docker Desktop:** รัน `docker run hello-world` ผ่าน
3. **Node.js:** ตรวจสอบ version (`node -v`) เป็น v18 ขึ้นไป

การเตรียม Project (Test Run):

1. เปิด Terminal ใน Antigravity

2. สร้าง Next.js Project: `npx create-next-app@latest test-env --typescript --tailwind --eslint`
3. รัน Project: `npm run dev`
4. เปิด Browser ดูผลลัพธ์ที่ Localhost

ช่วงบ่าย: AI-Driven Requirements Engineering

3. AI-Driven Requirements Engineering

ทฤษฎี: From Vibe to Spec การสั่งงาน AI ที่ดีต้องมี **"Context Triangle"**:

1. **Role:** "Act as..." (บทบาท)
2. **Task:** "Create..." (งานที่ให้ทำ)
3. **Constraint:** "Use..." (ข้อจำกัด/เงื่อนไข)

โจทย์ Workshop: Fixed Asset Tracking System (MVP) เราจะพัฒนาระบบบริหารจัดการทรัพย์สินถาวร โดยมีฟังก์ชันหลัก:

1. บริหารจัดการประเภทสินทรัพย์
2. เพิ่ม แก้ไข ลบสินทรัพย์ได้
3. บันทึกผู้ครอบครองสินทรัพย์ (ส่งมอบ)
4. บันทึกคืนสินทรัพย์

Activity 1: Prompt to Create PRD ให้ลองเขียน Prompt เพื่อสร้าง PRD (Product Requirements Document) ภาษาไทย โดยใช้ข้อมูลจากโจทย์

- **Model:** Gemini 3 Pro
- **Example Prompt:**

```
ต้องการพัฒนาระบบบริหารจัดการ fixed asset
ที่มีความสามารถพื้นฐานพอ ไม่ต้อง advance
... (ใส่รายละเอียดฟังก์ชัน) ...
อยากให้เขียน product requirement definition ก่อน
เขียนเป็นภาษาไทย
```

Activity 2: Review & Refine PRD

- ตรวจสอบ PRD ที่ AI สร้างขึ้น ว่าครบถ้วนหรือไม่
- เปรียบเทียบกับ Template มาตรฐาน (`fixed_asset_prd.md`) ที่เตรียมไว้ให้

4. UI/UX Conceptualization

Activity 3: Create UI Spec สวมบทบาท UX Expert เพื่อสร้างเอกสาร UI Spec

- **Prompt:**

```
@fixed_asset_prd.md
คุณคือ ux expert
```

```
ให้สร้างเอกสาร ui spec ตาม requirement ไฟล์นี้
```

Activity 3.1: Tuning UI Spec (Deep Dive) ปรับปรุง UI Spec ให้ละเอียดระดับที่ Developer นำไปใช้ได้

- **Prompt:**

```
ช่วยสร้าง ui spec แบบละเอียด
ลงถึงระดับที่จะส่งให้ dev สร้างต่อได้เลย
เช่นลงรายละเอียด theme สีที่ระบุเลย
ผังโครงสร้างของระบบว่ามีหน้าจออะไรบ้าง เชื่อมต่อกันอย่างไร (Mermaid)
หน้าจอแต่ละหน้าจอ ควรมีการออกแบบ components , layout อย่างไร
เพื่ออะไร
```

5. Tech Architecture Design

Activity 4: Tech Architecture Design สวมบทบาท Senior Tech Architect เพื่อออกแบบโครงสร้างระบบ

- **Prompt:**

```
คุณคือ tech architect อาวุโส
ชวยออกแบบ architecture ของระบบนี้
โดยมีโครงสร้างตาม template นี้
@architecture-tmpl.yaml
```

สิ่งที่ควรพิจารณา:

- ทำไมใช้ Next.js? (Fullstack capabilities)
- ทำไมใช้ Prisma? (Type-safety)
- ทำไมใช้ Postgres? (Relational data integrity)

Day 2: The Construction & Deployment (สร้างและขึ้นระบบจริง)

เป้าหมายประจำวัน

- เปลี่ยน Requirements เป็น User Stories
- สร้าง Application จริง (Alpha Version) ด้วย AI
- Deploy ระบบขึ้นใช้งานจริง

ช่วงเช้า: Planning & Construction

1. Agile Planning with AI

Activity 5: Create Epic / User Stories แปลง PRD และ UI Spec เป็น User Stories สำหรับการพัฒนา

- **Prompt:**

```
@fixed_asset_prd.md @fixed_asset_ui_spec.md
สร้าง user story file ละเอียดตาม template @story-tmpl.yaml
```

- **Check:** ตรวจสอบ Acceptance Criteria ของแต่ละ Story ว่าชัดเจนหรือไม่

2. Vibe Coding: Alpha Version

Activity 6: Start Coding เมื่อมี Blueprint ครบแล้ว (Arch + UI Spec + Stories) ก็เริ่มก่อสร้าง

- **Prompt:**

```
@fixed_asset_architecture.md @fixed_asset_ui_spec.md
@fixed_asset_stories.md
เริ่ม develop ได้เลย
```

- AI จะเริ่มสร้าง Project Structure, Database Schema, API และ UI Components

Activity 7: ระหว่างรอ AI (The Waiting Game)

- สังเกตการทำงานของ AI
- ศึกษา Code Structure ที่ AI สร้างขึ้น
- เตรียมคำถามหรือจุดที่ต้องการปรับปรุง

3. Refinement & Enhancement

Activity 8: Refine Alpha Version ตรวจสอบและปรับปรุงระบบ:

- **Functionality:** ลองเพิ่มฟังก์ชัน เช่น อัปโหลดรูปภาพ, คำนวณค่าเสื่อม
- **Database:** สั่ง AI ให้ update schema และ migrate db หากมีการแก้ไขข้อมูล
- **UI/UX:** ปรับ Theme สี หรือ Font ให้สวยงาม

ช่วงบ่าย: Deployment & Closing

4. Final Polish & Deployment Prep

Activity 9: Final Polish

- เพิ่ม Authentication ง่ายๆ
- **Bug Bash:** ลองสลับกันทดสอบระบบกับเพื่อน หาจุดผิดพลาด (Login, Flow การใช้งาน, Mobile View)

Activity 10: Deployment to Vercel นำระบบขึ้น Cloud (Production)

1. **Cloud DB:** เตรียม Connection String จาก Neon
2. **GitHub:** Push Code ขึ้น GitHub Repository
3. **Vercel:** Import Project -> ตั้งค่า Environment Variables (`DATABASE_URL`)
4. **Deploy:** สั่ง Deploy และรอรับ URL

5. Documentation & Handover

Activity 11: Auto-Docs สร้างคู่มือการใช้งานด้วย AI

- **User Manual:** Prompt "Generate a User Guide for this Booking System based on the current UI features."
- **Technical Docs:** Prompt "Generate a README.md explaining how to setup, run, and deploy this project."

ภาคผนวก (Appendix)

Role Matrix (ตารางบทบาท)

Activity	PM/SA	Dev/SE
Requirements	**Lead:** Business Context	**Support:** Tech Feasibility
Architecture	**Review:** Flow	**Lead:** Stack & DB Design
Vibe Coding	**Review:** UI/UX	**Lead:** Prompting & Logic
Testing	**Lead:** UAT Scenarios	**Lead:** Unit/Load Test
Deployment	**Sign-off**	**Lead:** CI/CD & Pipeline

Prompt Library (ตัวอย่างคำสั่ง)

- **Requirement to DB:** "Act as a DB Architect. Create a schema for [App Description]..."
- **Text to UI:** "Create a modern dashboard using Tailwind CSS. It should have..."
- **Refactoring:** "Refactor this component to use React Hooks..."