คู่มือปฏิบัติการ (Student Lab Guide)

หลักสูตร: AI Pair Programming (General Vibe)

Project: Corporate Asset Tracker

🎯 Project Brief (โจทย์ตั้งต้น)

นี่คือโจทย์ที่เราจะใช้ "Vibe" กับ AI ตลอดหลักสูตร คุณสามารถนำข้อความนี้ไปปรับแต่งหรือเพิ่มเติมฟีเจอร์ได้ตามต้องการ

"Corporate Asset Tracker" คือระบบบริหารจัดการทรัพย์สินภายในองค์กร ที่เน้นความถูกต้องแม่นยำของการถือครอง (Chain of Custody)

Key Requirements (ความต้องการหลัก):

1. **Asset Identity:** ต้องแยกความแตกต่างระหว่าง "รุ่นสินค้า" (เช่น MacBook Pro M3) กับ "ของรายชิ้น" (เช่น Serial No. C02XYZ) ได้
2. **Assignment Flow:** ทรัพย์สิน 1 ชิ้น อยู่กับพนักงานได้แค่ 1 คนในเวลาเดียวกัน (ป้องกันการยืมซ้อน)
3. **Offboarding Safety:** เมื่อพนักงานลาออก (Resign) ระบบต้องแจ้งเตือนได้ทันทีว่าเขายังถือครองทรัพย์สินอะไรอยู่บ้าง เพื่อป้องกันความเสียหาย

Tip: คุณสามารถเพิ่มไอเดียของตัวเองลงไปได้ เช่น ระบบจองล่วงหน้า, การคำนวณค่าเสื่อมราคา, หรือการแนบรูปถ่ายสภาพของ

Lab 0: Setup & Preparation

เป้าหมาย: เตรียมเครื่องมือและ Project ให้พร้อม

1. **Create Project:**

```
npx create-next-app@latest asset-tracker --typescript --tailwind --
eslint
cd asset-tracker
code . # หรือเปิด Folder นี้ด้วย Antigravity / VS Code
```

2. **Setup Database (Docker):**
   - สร้างไฟล์ `docker-compose.yml` (ใช้ AI Gen ได้)
   - รัน `docker-compose up -d`
3. **Install Prisma:**

```
npm install prisma --save-dev
npx prisma init
```

Lab 1: The Blueprint (ออกแบบฐานข้อมูล)

เป้าหมาย: สร้างตารางเก็บข้อมูลทรัพย์สินและพนักงาน

1.1 Prompting for Schema

เปิด Chat กับ AI แล้วพิมพ์ Prompt ดังนี้:

> "Act as a Database Architect. Design a Prisma schema for a **Fixed Asset Tracking System**.
>
> We need to track:
>
> 1. **Users** (Employees): id, name, email, status (Active/Resigned).
> 2. **AssetModels** (The type of product): id, name, category, image, specs.
> 3. **AssetItems** (The physical item): id, serialNumber, status (Available, InUse, Maintenance, Retired).
>   - Must belong to an AssetModel.
> 4. **Assignments** (History of usage): id, assignedDate, returnedDate.
>   - Links a User to an AssetItem.
>
> Please provide the `schema.prisma` code."

1.2 Apply Schema

1. Copy Code ไปใส่ในไฟล์ `prisma/schema.prisma`
2. รันคำสั่ง: `npx prisma db push`
3. **Check:** เปิด Database ดูว่าตารางมาครบไหม

Lab 2: Master Data (จัดการรุ่นสินค้า)

เป้าหมาย: สร้างหน้าจอสำหรับเพิ่ม/ลบ รุ่นสินค้า (เช่น MacBook Pro M3, Dell Monitor)

2.1 Create API

> "Create a Next.js API route set (GET, POST) for `AssetModel`.
>
> - GET: Fetch all models.
> - POST: Create a new model."

2.2 Create UI

> "Create a page `/models` using Tailwind CSS and Shadcn UI.
>
> - Display a grid of Asset Models (show image and name).
> - Add a button 'Add Model' that opens a Dialog with a form to create a new model.
> - Connect to the API we just created."

2.3 Seed Data

> "Generate a `seed.ts` script to populate the database with 5 dummy Asset Models (Laptops, Phones) and 10 Users."

Lab 3: The Inventory & Assignment (ระบบเบิกจ่าย)

เป้าหมาย: จัดการของรายชิ้น (Serial Number) และระบบยืม-คืน

3.1 Inventory Management

> "Create a page `/inventory`.
>
> - Display a table of `AssetItems`.
> - Columns: Serial Number, Model Name, Status.
> - Add a filter dropdown for Status."

3.2 The Assignment Logic (The Hard Part)

เราจะสร้างปุ่ม "Assign" ที่หน้า Inventory

> "Update the Inventory table. Add an 'Assign' button next to items with status 'Available'. When clicked, open a Dialog to select a User.
>
> On submit, call an API to:
>
> 1. Create a new `Assignment` record.
> 2. Update the `AssetItem` status to 'InUse'. **IMPORTANT:** Use a Prisma transaction to ensure both happen together."

3.3 The Return Logic

> "Add a 'Return' button next to items with status 'InUse'. When clicked:
>
> 1. Update the current `Assignment` record (set `returnedDate` to now).
> 2. Update the `AssetItem` status to 'Available'."

Lab 4: The Offboarding Dashboard (ตรวจสอบคนลาออก)

เป้าหมาย: สร้าง Report เพื่อดูว่าพนักงานที่กำลังจะออก ยังคืนของไม่ครบหรือไม่

4.1 Employee List

> "Create a page `/employees`. List all users. Show a badge for their status (Active/Resigned)."

4.2 Asset Holding View

> "When clicking on a User, go to `/employees/[id]`. Show user details and a list of **'Currently Held Assets'**. Logic: Find Assignments where `userId` matches AND `returnedDate` is null."

4.3 The Alert System

> "Create a Dashboard Widget on the homepage. Title: 'Offboarding Alert'. Show a list of Users who have `status = Resigned` BUT still have active assignments (holding assets). This is critical for HR."

Lab 5: Deployment

1. Push code to **GitHub**.
2. Import project in **Vercel**.
3. Setup **Supabase/Neon** database and update Environment Variables (`DATABASE_URL`).
4. Deploy and Test!