ขั้นตอนการสร้าง GoDrink App ด้วย Next.js และ Google Sheets บน VS Code

เอกสารนี้จะแนะนำขั้นตอนการสร้างโปรเจกต์ GoDrink สำหรับใช้งานบนเครื่องของคุณ (Local Project) โดยใช้ Next.js, Tailwind CSS และดึงข้อมูลสินค้ามาจาก Google Sheets โดยตรง

สิ่งที่ต้องมีก่อนเริ่ม (Prerequisites)

1. **Node.js**: ตรวจสอบว่าคุณติดตั้ง Node.js เวอร์ชั่น 18.17 หรือสูงกว่า
2. **Visual Studio Code (VS Code)**: โปรแกรม Code Editor
3. **บัญชี Google**: สำหรับสร้าง Google Sheets

ขั้นตอนที่ 1: เตรียมข้อมูลใน Google Sheets

เราจะสร้างไฟล์ Google Sheets เพื่อใช้เป็นฐานข้อมูล (Database) จำลองสำหรับรายการเครื่องดื่มของเรา

1. ***สร้าง Google Sheet ใหม่***:
   - ไปที่ sheets.google.com และสร้างเอกสารใหม่ (Blank spreadsheet)
   - ตั้งชื่อชีตว่า GoDrink-Products
   - ในชีตแรก (ตั้งชื่อว่า Sheet1) ให้สร้างตารางข้อมูลตามนี้ และคัดลอกข้อมูลตัวอย่างไปวางได้เลย:

id	name	thaiName	description	thaiDescription	price	category	image
1	Signature Cold Brew	ซิกเนเจอร์ โคลด์ บรูว์	Our smooth and rich cold brew.	โคลด์ บรูว์รสชาติเข้มข้นกลมกล่อม	120	Coffee	https://images.pexels.com/photos/312418/pexels-photo-312418.jpeg
2	Tropical Smoothie	ทรอปิคอล สมูทตี้	A refreshing blend of mango and passion fruit.	การผสมผสานที่สดชื่นของมะม่วงและเสาวรส	100	Smoothies	https://images.pexels.com/photos/1484516/pexels-photo-1484516.jpeg

3	Rose Milk Tea	ชานมกุหลาบ	Fragrant rose tea with creamy milk.	ชากุหลาบหอมกรุ่นกับนมรสนุ่ม	90	Tea	https://images.pexels.com/photos/3755710/pexels-photo-3755710.jpeg
4	Sparkling Lychee	ลิ้นจี่โซดา	Sweet lychee with a refreshing fizz.	ลิ้นจี่หวานชื่นใจพร้อมโซดาซ่า	85	Special	https://images.pexels.com/photos/1293268/pexels-photo-1293268.jpeg
5	Caramel Macchiato	คาราเมลมัคคิอาโต้	Rich espresso with vanilla syrup and caramel.	เอสเพรสโซ่เข้มข้นกับไซรัปวานิลลาและคาราเมล	110	Coffee	https://images.pexels.com/photos/2983101/pexels-photo-2983101.jpeg
6	Matcha Latte	มัทฉะลาเต้	Ceremonial grade matcha with steamed milk.	มัทฉะเกรดพิธีชงกับนมอุ่น	115	Tea	https://images.pexels.com/photos/461428/pexels-photo-461428.jpeg
7	Berry Blast Smoothie	เบอร์รี่ บลาสท์ สมูทตี้	A mix of strawberries, blueberries, and raspberries.	ส่วนผสมของสตรอเบอร์รี่ บลูเบอร์รี่ และราสเบอร์รี่	125	Smoothies	https://images.pexels.com/photos/109275/pexels-photo-1

							09275.jpeg
8	Butterfly Pea Limeade	อัญชันมะนาว	A magical color-changing drink.	เครื่องดื่มเปลี่ยนสีได้สุดมหัศจรรย์	95	Special	https://images.pexels.com/photos/814036/pexels-photo-8140366.jpeg

2. **เปิดการเข้าถึง (Share)**:
   - คลิกที่ปุ่ม **"Share"** (แชร์) มุมขวาบน
   - ในส่วน **"General access"** (การเข้าถึงทั่วไป) ให้เปลี่ยนจาก "Restricted" (จำกัด) เป็น **"Anyone with the link"** (ทุกคนที่มีลิงก์)
   - ตั้งค่าสิทธิ์เป็น **"Viewer"** (ผู้มีสิทธิ์อ่าน)
   - คลิก **"Done"** (เสร็จสิ้น)
3. **คัดลอก Spreadsheet ID**:
   - ดูที่ URL ของเบราว์เซอร์ จะมีลักษณะดังนี้:
     https://docs.google.com/spreadsheets/d/SPREADSHEET_ID/edit#gid=0
   - คัดลอกค่า SPREADSHEET_ID เก็บไว้ (จะเป็นสตริงยาวๆ)

ขั้นตอนที่ 2: สร้างโปรเจกต์ Next.js

1. **เปิด Terminal (หรือ Command Prompt) ขึ้นมา**
2. **ใช้คำสั่งเพื่อสร้างโปรเจกต์ Next.js ใหม่**:
   npx create-next-app@latest godrink-app

3. **ตอบคำถามตามนี้**:
   - Would you like to use TypeScript? **No**
   - Would you like to use ESLint? **Yes**
   - Would you like to use Tailwind CSS? **Yes**
   - Would you like to use 'src/' directory? **No**
   - Would you like to use App Router? **Yes**
   - Would you like to customize the default import alias? **No**
4. **เข้าสู่โฟลเดอร์โปรเจกต์**:
   cd godrink-app

5. **เปิดโปรเจกต์ใน VS Code**:
   code .

ขั้นตอนที่ 3: สร้างฟังก์ชันสำหรับดึงข้อมูล

เราจะใช้วิธี fetch API มาตรฐานเพื่อดึงข้อมูลจาก Google Sheets โดยตรง ไม่ต้องติดตั้ง Library เพิ่มเติม

1. **สร้างไฟล์สำหรับดึงข้อมูล**:
    - สร้างโฟลเดอร์ใหม่ชื่อ lib ใน root ของโปรเจกต์
    - ในโฟลเดอร์ lib สร้างไฟล์ใหม่ชื่อ sheets.js
    - ใส่โค้ดนี้ลงใน lib/sheets.js:

```
export async function getProducts() {
  try {
    const SPREADSHEET_ID = 'YOUR_SPREADSHEET_ID'; // <--- วาง SPREADSHEET_ID ของคุณที่นี่
    const SHEET_NAME = 'Sheet1';
    const URL = `https://docs.google.com/spreadsheets/d/${SPREADSHEET_ID}/gviz/tq?tqx=out:json&sheet=${SHEET_NAME}`;

    const res = await fetch(URL);
    if (!res.ok) {
      throw new Error(`Failed to fetch sheet data: ${res.statusText}`);
    }

    // Google Sheets API คืนค่ามาเป็น Text ที่ต้องตัดส่วนที่ไม่ใช่ JSON ออก
    let text = await res.text();
    const jsonString = text.match(/google\.visualization\.Query\.setResponse\((.*)\)/s)[1];
    const data = JSON.parse(jsonString);

    // แปลงข้อมูลจากโครงสร้างของ Google Sheets ให้เป็น Array ของ Object ที่เราใช้งานได้
    const headers = data.table.cols.map(col => col.label);
    const rows = data.table.rows;

    const products = rows.map(row => {
      const product = {};
      headers.forEach((header, index) => {
        const cell = row.c[index];
        product[header] = cell ? cell.v : null;
      });
      return product;
    });

    // แปลง price ให้เป็นตัวเลข
    const formattedProducts = products.map(product => ({
```

```
    ...product,
    price: Number(product.price) || 0,
  }));

  return formattedProducts;

 } catch (error) {
  console.error('Error fetching from Google Sheets:', error);
  return []; // คืนค่าเป็น array ว่างถ้าเกิดข้อผิดพลาด
 }
}
```

สำคัญ: อย่าลืมเปลี่ยน YOUR_SPREADSHEET_ID เป็น ID ที่คุณคัดลอกมาจากขั้นตอนที่ 1

ขั้นตอนที่ 4: นำโค้ด React มาปรับใช้และแสดงผล

ตอนนี้เราจะนำโค้ด React จากที่คุณมีอยู่ มาใส่ในโปรเจกต์ Next.js ของเรา

1. ล้างไฟล์ app/page.js:
   เปิดไฟล์ app/page.js และลบโค้ดทั้งหมดออก แล้วแทนที่ด้วยโค้ดนี้:

   ```
   import { getProducts } from '../lib/sheets';
   import ClientPage from './ClientPage'; // เราจะสร้างไฟล์นี้ต่อไป

   export default async function HomePage() {
    // Fetch data on the server
    const products = await getProducts();

    // Pass data to a Client Component
    return <ClientPage initialProducts={products} />;
   }
   ```

   Next.js 13+ (App Router) จะทำการดึงข้อมูลบนฝั่ง Server ก่อน ซึ่งดีต่อ SEO และประสิทธิภาพ
2. สร้าง Client Component:
   เนื่องจากหน้าเว็บของเรามีการโต้ตอบกับผู้ใช้ (เช่น การกดปุ่ม, การจัดการ state) เราต้องสร้างเป็น Client Component
   - สร้างไฟล์ใหม่ในโฟลเดอร์ app ชื่อ ClientPage.js
   - คัดลอกโค้ด React เดิม (จาก immersive godrink-react-app) มาวางในไฟล์นี้
   - **แก้ไขส่วนบนสุดของไฟล์ ClientPage.js** ดังนี้:

     ```
     'use client'; // บรรทัดนี้สำคัญมาก! เพื่อบอก Next.js ว่านี่คือ Client Component

     import React, { useState, useMemo } from 'react';

     // --- SVG Icons ---
     ```

```
// (โค้ด SVG ทั้งหมดของคุณอยู่ที่นี่... ไม่ต้องเปลี่ยนแปลง)
const HomeIcon = ({ className }) => { /* ... */ };
// ... ไอคอนอื่นๆทั้งหมด

// --- Mock Data ---
// ลบ const products และ const categories ออกไปได้เลย
// เพราะเราจะรับข้อมูลผ่าน props แทน

const categories = [
  { name: 'Coffee', thaiName: 'กาแฟ', icon: CoffeeIcon, color: 'bg-pink-100' },
  { name: 'Tea', thaiName: 'ชา', icon: TeaIcon, color: 'bg-sky-100' },
  { name: 'Smoothies', thaiName: 'สมูทตี้', icon: SmoothieIcon, color:
'bg-green-100' },
  { name: 'Special', thaiName: 'เมนูพิเศษ', icon: SpecialDrinkIcon, color:
'bg-yellow-100' },
];


// --- Components ---
// (โค้ด Component ทั้งหมดของคุณอยู่ที่นี่... ไม่ต้องเปลี่ยนแปลง)
const Header = () => { /* ... */ };
// ... Component อื่นๆทั้งหมด

// --- Main App Component ---
// แก้ไข Component หลัก (ที่เคยชื่อ App) เป็น ClientPage
export default function ClientPage({ initialProducts }) {
  const [activeView, setActiveView] = useState('home');
  const [cart, setCart] = useState([]);
  const [selectedCategory, setSelectedCategory] = useState(null);

  // ใช้ initialProducts ที่ได้รับมาจาก Server
  const products = initialProducts;

  const handleAddToCart = (productToAdd) => {
    // (โค้ดส่วนนี้เหมือนเดิม)
  };

  const handleUpdateCart = (productToUpdate, newQuantity) => {
    // (โค้ดส่วนนี้เหมือนเดิม)
  };

  const filteredProducts = useMemo(() => {
    if (!selectedCategory) return products;
    return products.filter(p => p.category === selectedCategory);
  }, [selectedCategory, products]); // เพิ่ม products เข้าไปใน dependency array
```

```
    const handleSelectCategory = (categoryName) => {
      // (โค้ดส่วนนี้เหมือนเดิม)
    }

    const renderView = () => {
      // (โค้ดส่วนนี้เหมือนเดิม)
    };

    return (
      // (โค้ด JSX ที่ return เหมือนเดิม)
    );
  }
```

สรุปการแก้ไขใน ClientPage.js:

- เพิ่ม 'use client'; ที่บรรทัดแรกสุด
- ลบ const products ที่เป็น mock data ออก
- เปลี่ยนชื่อ Component หลักจาก App เป็น ClientPage และรับ initialProducts ผ่าน props
- กำหนด const products = initialProducts; เพื่อให้โค้ดส่วนที่เหลือทำงานได้
- ใน useMemo ของ filteredProducts ให้เพิ่ม products เข้าไปใน dependency array

3. ล้างไฟล์ app/globals.css:
   เปิดไฟล์ app/globals.css และลบโค้ด CSS ที่มีอยู่ทั้งหมดออก เหลือไว้แค่ 3 บรรทัดนี้:
   @tailwind base;
   @tailwind components;
   @tailwind utilities;

ขั้นตอนที่ 5: รันโปรเจกต์

1. **กลับไปที่ Terminal ใน VS Code**
2. **รันคำสั่งเพื่อเริ่ม Development Server**:
   npm run dev

3. **เปิดเบราว์เซอร์แล้วไปที่**: http://localhost:3000