⌨️ MWIT Interactive Web Programming Workbook

Upgrade Your Web App 🚀

เปลี่ยนการบันทึกข้อมูลจาก Local Storage ไปสู่ Supabase Cloud Database

ความคืบหน้าของบทเรียน 100%

MISSION

☁️ ภารกิจ: พาเว็บของเราไปอยู่บน Cloud

เว็บ Mini Project ของคุณทำงานได้แล้ว แต่ข้อมูลยังอยู่เฉพาะ Browser เครื่องเดิม บทนี้เราจะอัปเกรดเฉพาะระบบ Save และ Load โดยรักษาการคำนวณและ UI เดิมเอาไว้

เดิม		บทนี้
localStorage ข้อมูลอยู่ใน Browser	→	**Supabase** ข้อมูลอยู่ใน Cloud Database

✅ เมื่อจบบทนี้ คุณจะทำได้

- ✓ ออกแบบ table ให้ตรงกับข้อมูลในเว็บ
- ✓ เชื่อมเว็บกับ Supabase
- ✓ บันทึก row ใหม่ด้วย `insert()`
- ✓ โหลดหลายรายการด้วย `select()`
- ✓ ทดสอบข้อมูลข้าม Browser

🧭 ขอบเขตของบทนี้

เรียนตอนนี้: Create + Read

ยังไม่เรียน: Update, Delete, Authentication

> ถ้าบันทึกผิด เราจะลบ row ไหน? นี่คือภารกิจบทถัดไป

ก่อนเริ่ม: เลือกโปรเจกต์ของคุณ

ตัวอย่างโค้ดและชื่อตารางจะปรับตามตัวเลือกนี้

📚 Personal Study Planner

💰 Financial Helper

🧩 โปรเจกต์อื่น ๆ

ชื่อเว็บของฉัน

FinGoal — วางแผนการเงิน

• บันทึกคำตอบอัตโนมัติใน Browser นี้

STEP 1

🧱 เปลี่ยนข้อมูลในเว็บให้เป็น Table

หนึ่ง object ใน JavaScript จะกลายเป็นหนึ่ง row ส่วน property แต่ละตัวจะกลายเป็น column

JavaScript object → Database row

สำรวจเว็บเดิมของคุณ

เว็บรับข้อมูลอะไรบ้าง?

ฟอร์มบันทึกรายการ รับข้อมูล: ชื่อรายการ (item), จำนวนเงินรับ (income), จำนวนเงินจ่าย (expense)

ตอนนี้ Local Storage เก็บอะไร?

เก็บ array ของ object รายการทั้งหมดด้วย localStorage.setItem(...) เช่น key "fingoal_transactions" เก็บเป็น JSON.stringify(data)

หนึ่ง row ในเว็บของคุณหมายถึงอะไร?

1 row = 1 รายการรายรับหรือรายจ่าย ที่ผู้ใช้บันทึกหนึ่งครั้ง

Table ตัวอย่าง: `finance_logs`

สร้างผ่าน Table Editor ตามโครงสร้างนี้ หรือใช้ SQL ด้านล่าง

Financial Helper

Column	Type	หน้าที่	ตัวอย่าง
`id`	int8	รหัสของแต่ละ row	อัตโนมัติ
`created_at`	timestamptz	เวลาที่บันทึก	อัตโนมัติ
`item`	text	ข้อมูลจากเว็บ	ค่าอาหาร
`income`	float8	ข้อมูลจากเว็บ	500
`expense`	float8	ข้อมูลจากเว็บ	120

▶ ทางเลือก: ดู SQL สำหรับสร้างตาราง

⚠️ **ใช้ข้อมูลทดลองเท่านั้น** บทนี้ยังไม่มีระบบ Login ข้อมูลอาจมองเห็นร่วมกันตาม RLS policy ที่ครูกำหนด

STEP 2

🔌 เชื่อมโปรเจกต์เดิมกับ Supabase

ทำงานในโฟลเดอร์สำเนาของ Mini Project และเก็บเวอร์ชัน Local Storage เดิมไว้เสมอ

```
my-project-supabase/
  ├── index.html
  ├── style.css
  └── script.js
```

1. เพิ่ม Supabase JavaScript Library

วางก่อน `</body>` และก่อน `script.js`

```
<script src="https://cdn.jsdelivr.net/npm/@supabase/supabase-js@2"></script>
<script src="script.js"></script>
```

2. สร้าง Client ใน `script.js`

```
const SUPABASE_URL = "YOUR_SUPABASE_URL";
const SUPABASE_KEY = "YOUR_PUBLISHABLE_KEY";

const supabaseClient = supabase.createClient(
  SUPABASE_URL,
  SUPABASE_KEY
);
```

🔑 **ใช้เฉพาะ Publishable Key**
ห้ามนำ Service Role Key มาใส่ในหน้าเว็บ

ตรวจสอบก่อนเดินต่อ

- ☑ สร้าง table แล้ว
- ☑ ใส่ URL และ Publishable Key แล้ว
- ☑ Console ไม่พบ supabase is not defined

STEP 3

💾 Upgrade Save: จาก setItem() เป็น insert()

เปลี่ยนเฉพาะปลายทางที่เก็บข้อมูล ส่วนการอ่านค่าจาก input และการคำนวณเดิมยังใช้ต่อได้

BEFORE · LOCAL

```
localStorage.setItem(
  "myData",
  JSON.stringify(data)
);
```

AFTER · CLOUD

```
await supabaseClient
  .from("finance_logs")
  .insert([data]);
```

ตัวอย่างฉบับเต็มสำหรับ Financial Helper

ปรับชื่อ id ของ input ให้ตรงกับโปรเจกต์จริง

finance_logs

```
async function saveData() {
  const newRecord = {
  item: itemInput.value.trim(),
  income: Number(incomeInput.value),
  expense: Number(expenseInput.value)
};

  const { error } = await supabaseClient
    .from("finance_logs")
    .insert([newRecord]);

  if (error) {
    console.error(error.message);
    return;
  }

  alert("บันทึกข้อมูลสำเร็จ");
}
```

Predict ก่อน Run

ถ้ากด Save สองครั้ง จะมีข้อมูลใน table กี่ row และเพราะเหตุใด?

2 rows — เพราะทุกครั้งที่เรียก insert() จะเพิ่ม row ใหม่ ไม่เหมือน setItem() ที่เขียนทับค่าของ key เดิม

▶ ดูแนวคิดหลังจากทดลองแล้ว

STEP 4

📊 Upgrade Load: อ่านหลาย rows ด้วย select()

`select("*")` ขอข้อมูลทุก column และ `order()` เรียงรายการใหม่ไว้ด้านบน

โหลดและแสดงประวัติ

```
async function loadData() {
  const { data, error } = await supabaseClient
    .from("finance_logs")
    .select("*")
    .order("created_at", { ascending: false });

  if (error) { console.error(error.message); return; }

  historyList.innerHTML = data.map(item => `
    <div class="history-item">
      ${item.item} · รับ ${item.income} บาท · จ่าย ${item.expense} บาท
    </div>
  `).join("");
}
```

select() ขอข้อมูล → **data** array ของ objects → **map()** สร้าง HTML

ทดลองข้าม Browser

1. บันทึกข้อมูลจาก Browser แรก
2. เปิดเว็บในอีก Browser หรืออุปกรณ์
3. กด Load แล้วสังเกตผล

ผลต่างจาก Local Storage ที่สังเกตได้คืออะไร?

ข้อมูลใน Browser ที่สองโผล่ขึ้นมาเหมือนกัน เพราะถูกดึงจาก Supabase cloud database ตัวเดียวกัน — ต่างจาก Local Storage ที่เก็บข้อมูลเฉพาะ browser ที่บันทึก ถ้าเปิดอีกเครื่องจะไม่เห็น

🛠️ ถ้ายังไม่สำเร็จ

- ▶ relation ... does not exist
- ▶ row-level security policy
- ▶ column ... does not exist
- ▶ supabase is not defined

STEP 5

🧪 Test Like a Developer

อย่าเชื่อว่าเว็บทำงานจนกว่าจะทดสอบทั้งกรณีปกติและกรณีผิดพลาด

- ☑ กรอกข้อมูลครบแล้ว Save สำเร็จ
- ☑ ข้อมูลปรากฏเป็น row ใหม่ใน Supabase
- ☑ Refresh แล้ว Load ข้อมูลกลับมาได้
- ☑ เปิดอีก Browser แล้วเห็นข้อมูล
- ☑ กรอกไม่ครบแล้วเว็บไม่บันทึก
- ☑ เว็บเดิมยังคำนวณและแสดงผลได้
- ☑ ไม่มี Service Role Key และข้อมูลส่วนตัวจริง

Reflection: อธิบายสิ่งที่คุณเรียนรู้

1. Local Storage และ Supabase ต่างกันอย่างไร?

Local Storage เก็บข้อมูลเฉพาะใน browser/เครื่องนั้น เปิดอีกเครื่องจะไม่เห็นและล้างได้ง่าย ส่วน Supabase เก็บบน cloud server กลาง ทุกอุปกรณ์ที่เชื่อมต่อ table เดียวกันเห็นข้อมูลชุดเดียวกัน ซิงก์ผ่านอินเทอร์เน็ต

2. เหตุใดชื่อ property จึงต้องตรงกับชื่อ column?

เพราะ insert() / select() จะจับคู่ชื่อ property ใน object กับชื่อ column ในตารางโดยตรง ถ้าชื่อไม่ตรงกันจะเจอ error "column ... does not exist" ทำให้บันทึกหรือโหลดข้อมูลไม่ได้

3. หนึ่ง object กลายเป็นหนึ่ง row ได้อย่างไร?

เมื่อเรียก insert([object]) Supabase จะสร้าง 1 row ใหม่ในตาราง โดยแต่ละ property กลายเป็นค่าในแต่ละ column เช่น { item: "ค่าอาหาร", income: 500 } กลายเป็น row ที่มี item = "ค่าอาหาร" และ income = 500 ส่วน id กับ created_at เติมให้อัตโนมัติ

4. ถ้ามีข้อมูลผิดหนึ่งรายการ เราจะรู้ได้อย่างไรว่าต้องลบ row ใด?

ดูจาก column id ซึ่งเป็น primary key ไม่ซ้ำกันในแต่ละ row — ใช้ id ระบุ row ที่ต้องการลบ เช่น .delete().eq("id", id) (เนื้อหาบทถัดไป)

MISSION CHECK

Upgrade สำเร็จแล้ว!

ทำเครื่องหมาย “ทำส่วนนี้แล้ว” ให้ครบทุกแท็บ

100%

บทถัดไป

Delete Data: จะลบ row ที่เลือกได้อย่างไร?

`id` → ปุ่มลบ → `.delete().eq("id", id)`