แผนการสอน Bootcamp 10 วัน: KKU Library System (Node.js + Express + PostgreSQL)

> โปรเจกต์ต่อเนื่องตลอดคอร์ส: **KKU Library System**
> ฟีเจอร์หลัก: รายการหนังสือ, สมาชิก, ยืม-คืน, Dashboard

Day 1: บทนำการพัฒนาเว็บและพื้นฐาน JavaScript

แนะนำแนวคิดพื้นฐานการพัฒนาเว็บ (Duration: 60 นาที)

- ทำความเข้าใจหลักการทำงานแบบ Client–Server และ HTTP (Browser ส่ง request, Server ตอบ response)
- ยกตัวอย่างการสื่อสารระหว่าง Web Browser กับ Server
- แนะนำภาพรวมโปรเจกต์ “KKU Library System” (ระบบรายการหนังสือ, สมาชิก, ยืม-คืน, Dashboard)

ติดตั้งและเตรียมเครื่องมือ (Duration: 50 นาที)

- ติดตั้ง Node.js และ npm (เป็นสภาพแวดล้อม runtime สำหรับรัน JavaScript ฝั่งเซิร์ฟเวอร์)
- เปิดตัว VS Code (หรือ Editor ที่ถนัด) และติดตั้ง Extension พื้นฐาน
- สร้างโปรเจกต์ Node เบื้องต้น (`npm init`), ติดตั้ง Express.js (เฟรมเวิร์ก Node.js ยอดนิยมสำหรับสร้างเว็บ)

พื้นฐาน JavaScript (ES6) (Duration: 220 นาที)

- แนะนำตัวแปร (`let`, `const`) และชนิดข้อมูลพื้นฐาน (ตัวเลข, ข้อความ, อาร์เรย์, อ็อบเจกต์)
- คำสั่งเงื่อนไข (`if/else`) และลูป (`for`, `while`) ใน JavaScript
- ฟังก์ชันพื้นฐาน, Arrow Function, การใช้งาน `console.log` เพื่อตรวจสอบการทำงานของโค้ด
- เรียนรู้การใช้ NPM ติดตั้งแพ็กเกจพื้นฐาน (เช่น Express)

พื้นฐาน HTML และ Tailwind CSS (Duration: 150 นาที)

- สร้างโครงสร้างหน้าเว็บด้วย HTML เบื้องต้น (แท็ก `<!doctype html>`, `<html>`, `<head>`, `<body>`)
- แนะนำ Tailwind CSS (ใช้ผ่าน CDN): เฟรมเวิร์ก CSS แบบ utility-first มีคลาสช่วยให้สร้างดีไซน์ได้ง่าย
- ฝึกใช้ Tailwind จัดแต่ง layout แบบง่ายๆ (กำหนดสี, ขนาด, การจัดวางด้วยคลาสเช่น `flex`, `text-center` ฯลฯ)

Day 2: เริ่มต้นใช้งาน Node.js และ Express.js

เรียนรู้ Node.js พื้นฐาน (Duration: 80 นาที)

- เรียกใช้คำสั่ง `node` เพื่อรันไฟล์ `.js` ง่ายๆ (เช่น `console.log`)
- ทำความเข้าใจโฟลเดอร์โปรเจกต์ Node และไฟล์ `package.json`
- ใช้ npm ติดตั้งแพ็กเกจ (เช่น Express) และดูวิธีจัดการ dependencies

สร้างโปรเจกต์ Node แรก (Duration: 100 นาที)

- สร้างไฟล์ `app.js` แล้วเขียนโค้ด Node.js เบื้องต้น (ใช้ CommonJS `require` นำเข้าโมดูล)
- ทดลองใช้โมดูลในตัว (เช่น `fs` สำหรับอ่าน/เขียนไฟล์) เพื่อเข้าใจพื้นฐาน Node

- เรียนรู้แนวคิด event-driven และ non-blocking I/O ของ Node.js (เช่น ติดป้ายเหตุการณ์ `listen`, `callback`)

แนะนำ Express.js (Duration: 100 นาที)

- ติดตั้ง Express และเรียกใช้งานในโปรเจกต์ (`const express = require('express')`)
- เข้าใจแนวคิดการทำงานของ Express (Framework บางเบาบน Node.js ที่ช่วยจัดการ routing ได้ง่าย)
- ทำความเข้าใจหลักการ Middleware เบื้องต้น (เช่น `express.json()`, `express.static()`)

สร้างเว็บเซิร์ฟเวอร์ด้วย Express (Duration: 100 นาที)

- เขียนตัวอย่าง “Hello World” ด้วย Express (สร้าง server, `app.get('/', ...)`, `app.listen`)
- ทดลองส่งข้อความกลับผู้ใช้ผ่าน `res.send()` และเปิดดูผลลัพธ์ในเบราว์เซอร์
- ฝึกใช้ Express สร้างเส้นทาง (Route) แบบง่ายๆ (`GET`, `POST`) และทดสอบด้วย Postman หรือเบราว์เซอร์

สรุปและทบทวน (Duration: 50 นาที)

- ทบทวนเนื้อหาทั้งหมดในวัน (ถาม-ตอบ, แก้ไขปัญหาที่พบ)

Day 3: การสร้าง REST API ด้วย Express.js

Routing ใน Express (Duration: 120 นาที)

- สร้างเส้นทางหลายแบบ (`GET`, `POST`, `PUT`, `DELETE`) และทำความเข้าใจกับ `req` และ `res`
- ใช้พารามิเตอร์ใน URL (เช่น `/books/:id`) และ query parameters (เช่น `?q=keyword`)
- กำหนดโครงสร้างโปรเจกต์ (แยกไฟล์ Route ต่างๆ, ใช้ `express.Router()`)

ส่งไฟล์ static และ Template (Duration: 100 นาที)

- ใช้ Express เสิร์ฟไฟล์ HTML/CSS/JS แบบ static (`express.static()`)
- (อาจใช้ EJS/Pug เบื้องต้น) ทดลองสร้าง HTML แบบ dynamic เช่น แสดงชื่อหนังสือจากตัวแปรในหน้า Template
- ฝึกสร้างหน้าฟอร์ม HTML (เช่น ฟอร์มเพิ่มหนังสือใหม่) แล้วส่งข้อมูลด้วยวิธี `POST`

การจัดการข้อมูลแบบง่าย (Duration: 80 นาที)

- สร้างตัวแปร/อาร์เรย์เก็บข้อมูลหนังสือแบบชั่วคราว (ในหน่วยความจำ) เพื่อฝึก CRUD
- เขียนโค้ดสำหรับเพิ่ม (create) อ่าน (read) อัพเดต (update) ลบ (delete) หนังสือ โดยไม่ใช้ฐานข้อมูล
- ทดสอบ API เหล่านี้โดยใช้งานกับ Postman และดู JSON response

ฝึกใช้งาน JSON และ REST (Duration: 80 นาที)

- เรียนรู้รูปแบบการส่งข้อมูล JSON ในการสื่อสาร Client–Server
- บันทึกเนื้อหาใน `package.json` และใช้งาน body-parser (หรือ `express.json()`) เพื่อรับ JSON payload
- ทดสอบโค้ดด้วยตนเอง (เช่น ส่ง JSON เข้า API ด้วย Postman)

สรุปและทบทวน (Duration: 50 นาที)

- ถาม-ตอบปัญหา, ทบทวนแนวคิด REST API และฟังก์ชันพื้นฐานของ Express

---

Day 4: เริ่มต้นกับฐานข้อมูล PostgreSQL

พื้นฐาน PostgreSQL และ SQL (Duration: 100 นาที)

- แนะนำ PostgreSQL (Postgres) เป็นฐานข้อมูลเชิงสัมพันธ์ (RDBMS) แบบโอเพ่นซอร์ส
- คำสั่ง SQL เบื้องต้น: `CREATE TABLE`, `INSERT`, `SELECT`, `UPDATE`, `DELETE`
- อธิบายโครงสร้างข้อมูล (`table`, `row`, `column`, `schema`) และความสัมพันธ์ (book ↔ author, member ↔ loans)

ออกแบบฐานข้อมูลระบบห้องสมุด (Duration: 80 นาที)

- วางโครงสร้างตารางหลัก: ตาราง `books` (id, title, author, year, status ฯลฯ)
- ตาราง `members` (id, name, email, password ฯลฯ) และ `loans` (id, book_id, member_id, borrow_date, return_date)
- สรุปความสัมพันธ์: หนังสือ 1 เล่มยืมได้หลายครั้ง, สมาชิก 1 คนยืมได้หลายเล่ม

ติดตั้งและเริ่มต้นใช้งาน PostgreSQL (Duration: 60 นาที)

- ติดตั้ง PostgreSQL บนเครื่อง (หรือใช้ Docker/เครื่องมือบน Cloud หากสะดวก)
- ใช้ `psql` หรือ pgAdmin สร้างฐานข้อมูลใหม่และล็อกอิน (สอนคำสั่งเบื้องต้น เช่น `\\c`, `\\dt`)
- สร้าง role/user ใหม่สำหรับโปรเจกต์ (ตัวอย่าง `me` พร้อมสิทธิ์สร้าง DB)

สร้างตารางในฐานข้อมูล (Duration: 100 นาที)

- เขียนคำสั่ง SQL สร้างตาราง `books`, `members`, `loans` ตามออกแบบ (กำหนด `PRIMARY KEY`, `FOREIGN KEY`)
- ทดลองเพิ่มข้อมูลตัวอย่าง (`INSERT`) ด้วยคำสั่ง SQL ผ่าน `psql`
- ดูการใช้งานคำสั่ง `\\dt` เพื่อตรวจสอบตาราง และ `SELECT` เพื่อตรวจสอบข้อมูล

เชื่อมต่อ Node.js กับ PostgreSQL (Duration: 80 นาที)

- ใช้ไลบรารี node-postgres (`pg`) ใน Node.js (`npm install pg`)
- เขียนโค้ดเชื่อมต่อฐานข้อมูล (ตั้งค่าการเชื่อมต่อ host, user, database, password)
- ทดสอบการ query ด้วย Node (`SELECT * FROM books`) และแสดงผลผ่าน `console.log`

ทบทวนและสรุป (Duration: 20 นาที)

- ทบทวนคำสั่ง SQL และขั้นตอนการเชื่อมต่อ, แก้ปัญหาที่เจอ

Day 5: สร้าง API สำหรับระบบรายการหนังสือ (Backend)

สร้าง API ดึงรายการหนังสือทั้งหมด (`GET /books`) (Duration: 60 นาที)

- เขียนโค้ด Express สร้าง route `GET /books` เพื่อดึงข้อมูลจากตาราง `books` (`SELECT * FROM books`)
- คืนค่า JSON ของรายการหนังสือทั้งหมด

สร้าง API แสดงรายละเอียดหนังสือ (`GET /books/:id`) (Duration: 60 นาที)

- เขียน route `GET /books/:id` ดึงข้อมูลหนังสือตาม id (`SELECT * FROM books WHERE id = $1`)
- ส่งคืนข้อมูลหนังสือที่ค้นหาเป็น JSON

สร้าง API เพิ่มหนังสือใหม่ (`POST /books`) (Duration: 60 นาที)

- เขียน route `POST /books` รับ JSON จาก front-end (เช่น `title`, `author`)
- ใช้ SQL `INSERT INTO books ... RETURNING *` เพื่อเพิ่มหนังสือและส่งกลับข้อมูลที่เพิ่มแล้ว

สร้าง API อัพเดตหนังสือ (`PUT /books/:id`) (Duration: 60 นาที)

- เขียน route `PUT /books/:id` รับข้อมูลใหม่จาก front-end
- ใช้ SQL `UPDATE books SET ... WHERE id = $1` แล้วส่งคืนข้อมูลที่อัพเดตแล้ว

สร้าง API ลบหนังสือ (`DELETE /books/:id`) (Duration: 40 นาที)

- เขียน route `DELETE /books/:id` ใช้ SQL `DELETE FROM books WHERE id = $1`
- ส่งสถานะสำเร็จหรือข้อความยืนยันการลบกลับไป

ทดสอบ API ด้วย Postman (Duration: 60 นาที)

- ส่งคำขอ (`GET/POST/PUT/DELETE`) จาก Postman ไปยัง API ที่สร้าง
- ตรวจสอบโครงสร้าง JSON ที่ได้กลับมา
- แก้ไขข้อผิดพลาดที่พบจากการทดสอบ

ภาพรวม JSON และ REST (Duration: 60 นาที)

- ทบทวนแนวคิด CRUD และ HTTP Methods (`GET/POST/PUT/DELETE`) ที่สัมพันธ์กับ REST API
- ย้ำการใช้ JSON เป็นรูปแบบแลกเปลี่ยนข้อมูลระหว่าง client-server

สรุปและฝึกปฏิบัติ (Duration: 60 นาที)

- ฝึกเชื่อมต่อ front-end (ทดสอบด้วย `curl` หรือ `fetch` เริ่มต้น)
- สรุปสิ่งที่ได้เรียนรู้ในวัน

Day 6: สร้างหน้าแสดงรายการหนังสือ (Frontend)

สร้างโครงสร้างหน้ารายการหนังสือ (HTML & Tailwind) (Duration: 80 นาที)

- สร้างไฟล์ `index.html` สำหรับหน้าแสดงรายการหนังสือ
- ออกแบบด้วย Tailwind: เพิ่ม navbar ง่ายๆ, ตารางหรือการ์ดสำหรับแสดงหนังสือ (ใช้คลาสเช่น `container`, `grid`, card ของ Tailwind)
- ตรวจสอบ layout ให้เป็น Responsive เบื้องต้น (ใช้ breakpoints ของ Tailwind)

เรียนรู้ Fetch API (`GET`) ใน JavaScript (Duration: 80 นาที)

- ใช้ฟังก์ชัน `fetch()` เรียกใช้งาน API `GET /books` ที่สร้างเมื่อวาน
- รู้จัก Promises และ `async/await` เพื่อจัดการ response (แปลงเป็น JSON ด้วย `response.json()`)
- จัดการกรณีเกิดข้อผิดพลาด (`catch` errors ง่ายๆ)

แสดงข้อมูลหนังสือบนหน้าเว็บ (Duration: 80 นาที)

- เขียนฟังก์ชัน JavaScript ดึงข้อมูลหนังสือ จากนั้นวนลูปสร้าง element (เช่น `<tr>` หรือ `<div>`) แทรกเข้าในตารางหรือ grid บน HTML

- ใช้ Tailwind จัดตกแต่ง (เช่น ข้อความโทนสี, ขนาด font) ให้ดูสวยงาม
- เพิ่มลิงก์หรือปุ่มบนแต่ละรายการเพื่อนำไปหน้ารายละเอียดหนังสือ

เพิ่มหน้ารายละเอียดหนังสือ (Duration: 40 นาที)

- สร้างไฟล์ `book.html` (หรือใช้ dynamic routing) สำหรับแสดงรายละเอียดหนังสือ
- รับพารามิเตอร์ id จาก URL (เช่น `?id=123`) และดึงข้อมูลเฉพาะหนังสือด้วย Fetch อีกครั้ง (`GET /books/:id`)
- แสดงข้อมูลเช่น ชื่อเรื่อง, ผู้แต่ง, สถานะ เป็นต้น

ฝึกเขียนโค้ดและดีบัก (Duration: 40 นาที)

- ทดลองเปิดหน้าเว็บ, ดู console log และ network tab ในเบราว์เซอร์ Developer Tools
- แก้ไขบั๊กที่พบ (เช่น ปัญหา CORS, path ผิดพลาด)

สรุปและทบทวน (Duration: 40 นาที)

- สอบถาม-ตอบปัญหา, สรุปการใช้ Fetch และ DOM manipulation

Day 7: จัดการรายละเอียดหนังสือและฟอร์ม (Full Stack)

สร้างหน้ารายละเอียดหนังสือ (Frontend) (Duration: 60 นาที)

- ออกแบบหน้าเว็บ `book.html` ด้วย Tailwind ให้แสดงข้อมูลหนังสือแบบอ่านง่าย (เช่น card หรือ sections)
- ดึง id จาก URL และใช้ Fetch เรียก API `/books/:id` มาแสดงข้อมูลในฟอร์มหรือหน้า
- เพิ่มปุ่ม “แก้ไข” และ “ลบ” ในหน้านี้ (เชื่อมโยงกับฟังก์ชันด้านหลัง)

สร้างฟอร์มเพิ่มหนังสือ (Frontend) (Duration: 60 นาที)

- ออกแบบฟอร์ม HTML หน้าใหม่ (เช่น `new-book.html`) ให้มีช่องกรอกข้อมูลหนังสือ (title, author, year)
- ใช้ Tailwind จัดรูปแบบฟอร์มให้ง่ายต่อการใช้งาน

ส่งฟอร์มเพิ่มหนังสือ (Frontend) (Duration: 60 นาที)

- เขียน JavaScript รับค่าจากฟอร์มและใช้ Fetch ส่ง `POST /books` ในรูปแบบ JSON
- หลังส่งสำเร็จ นำผู้ใช้กลับไปหน้าแรก (หรือแสดงข้อความยืนยัน)
- ทดสอบด้วยการเพิ่มหนังสือใหม่แล้วตรวจสอบว่าอยู่ในฐานข้อมูล

แก้ไขหนังสือ (Edit) (Duration: 60 นาที)

- เมื่อกด “แก้ไข” ในหน้ารายละเอียด ให้แสดงฟอร์มแบบเติมค่าปัจจุบันของหนังสือ
- เขียน JavaScript ส่งคำขอ `PUT /books/:id` พร้อมข้อมูลใหม่ที่กรอก
- อัปเดตหน้า UI หลังแก้ไขเสร็จ (อาจ redirect กลับ หรือแสดงข้อความสำเร็จ)

สรุปการเชื่อมต่อ Front-End และ Back-End (Duration: 40 นาที)

- ทบทวนการ map ระหว่างหน้าต่างๆ ของเว็บ กับ API (เช่น ฟอร์ม -> `POST`, หน้ารายการ -> `GET`)
- แนะนำวิธีตรวจสอบว่า Request ไปถึง Server หรือไม่ (ดู Network tab)

ฝึกปฏิบัติ (Duration: 80 นาที)

- ให้ผู้เรียนทำแบบฝึกหัดเขียนโค้ดเพิ่ม/แก้ไขหนังสือตามโจทย์
- ครูช่วยแก้ปัญหาเฉพาะหน้า และอธิบายเพิ่มเติมตามต้องการ

สรุปและถาม-ตอบ (Duration: 10 นาที)

Day 8: ระบบสมาชิกและการล็อกอินพื้นฐาน

ออกแบบตารางสมาชิก (Members) (Duration: 40 นาที)

- สร้างตาราง `members` (เช่น id, username, email, password (hashed) ฯลฯ)
- ทบทวนหลักการ hashing รหัสผ่าน (เช่น bcrypt เบื้องต้น) เพื่อความปลอดภัย (คงไม่ลงลึกมาก)

สร้าง API จัดการสมาชิก (Duration: 40 นาที)

- เขียน `POST /members` สำหรับสมัครสมาชิก (รับข้อมูล user/password)
- สร้าง `GET /members` (ดึงสมาชิกทั้งหมด), `GET /members/:id`, `PUT /members/:id`, `DELETE /members/:id` (ถ้ามีเวลา)

ยืนยันตัวตน (Authentication) (Duration: 60 นาที)

- อธิบายแนวคิดเบื้องต้น: session vs token (JWT) (เน้นง่ายๆ ว่าเป็นวิธีเก็บสถานะผู้ใช้)
- ใช้ Express session หรือ Cookie เพื่อตรวจสอบว่าใครล็อกอิน (ตั้งค่า `express-session`)
- สอนการเข้ารหัส / ตรวจสอบรหัสผ่านด้วย bcrypt (หรือบอกว่าเป็น step ถัดไป)

เขียนฟังก์ชันล็อกอิน/ล็อกเอาต์ (Duration: 60 นาที)

- เขียน API `POST /login` รับ username/password, ตรวจสอบกับฐานข้อมูล ถ้าถูกต้องเซ็ต session
- เขียน API `GET /logout` เคลียร์ session ของผู้ใช้

สร้างฟอร์มล็อกอินและลงทะเบียน (Frontend) (Duration: 60 นาที)

- สร้างหน้า `login.html` และ `register.html` ด้วย HTML/Tailwind
- มีช่องกรอก username/password และปุ่มยืนยัน
- เขียน JS ส่ง request: `POST /login` และ `POST /members` ตามลำดับ

ปกป้องเส้นทางที่สำคัญ (Middleware) (Duration: 40 นาที)

- สร้าง middleware ตรวจสอบ session (เช่น `if (!req.session.user) return res.redirect('/login');`)
- ใช้ middleware นี้กับ route ที่ต้องล็อกอิน เช่น หน้า Dashboard หรือฟังก์ชันแก้ไขข้อมูล

ฝึกปฏิบัติและทดสอบ (Duration: 30 นาที)

- ทดสอบไหล่สอน: สร้างสมาชิกใหม่, ล็อกอิน, เข้าถึงข้อมูลส่วนตัว
- แก้ไขบั๊ก (เช่น cookie ไม่ทำงาน, ข้อมูลไม่ถูกส่ง)

สรุปและถาม-ตอบ (Duration: 20 นาที)

Day 9: ระบบยืม-คืนหนังสือ

ออกแบบตาราง Loans (Duration: 60 นาที)

- สร้างตาราง `loans` (เช่น id, book_id (FK), member_id (FK), borrow_date, return_date)
- เพิ่มสถานะหนังสือในตาราง `books` (เช่น available หรือ not available) เพื่อใช้ตรวจสอบการยืม

สร้าง API สำหรับยืมหนังสือ (`POST /loans`) (Duration: 60 นาที)

- เขียน route `POST /loans` เพื่อบันทึกการยืมใหม่ (รับ `book_id` และ `member_id`)
- อัปเดตสถานะในตาราง `books` เป็นยืมแล้ว (`available = false`)
- ส่งข้อมูล loan กลับ (หรือสถานะสำเร็จ)

สร้าง API สำหรับคืนหนังสือ (`PUT /loans/:id` หรือ `DELETE`) (Duration: 40 นาที)

- เขียน route `PUT /loans/:id` เพื่อบันทึกคืนหนังสือ (เช่น set `return_date`)
- อัปเดตสถานะใน `books` เป็นว่าง (`available = true`)
- (หรือใช้ `DELETE /loans/:id` ถ้าไม่เก็บประวัติ)

สร้าง API แสดงประวัติยืม (`GET /loans`) (Duration: 40 นาที)

- รวบรวมข้อมูลการยืมทั้งหมด (join กับหนังสือและสมาชิกเพื่อดึงชื่อ)
- ส่งคืน JSON ให้ front-end เพื่อแสดงใน Dashboard หรือหน้า members

สร้าง UI ยืม-คืน (Frontend) (Duration: 100 นาที)

- สร้างฟอร์มบนหน้าเว็บ เช่น ปุ่ม “ยืมหนังสือ” ในหน้ารายละเอียดหนังสือ (ถ้าว่าง)
- เมื่อกดปุ่ม ส่งคำขอ `POST /loans` ผ่าน Fetch (ใส่ `book_id`, `member_id` จาก session)
- สร้างฟอร์ม “คืนหนังสือ” ในหน้ารายละเอียดการยืมหรือที่หน้า Dashboard (ขึ้นกับโครงสร้าง)
- ส่งคำขอ `PUT /loans/:id` เมื่อยืนยันคืน

ฝึกใช้งานและแก้ปัญหา (Duration: 30 นาที)

- ทดสอบการยืมและคืน ดูผลในฐานข้อมูล (status หนังสือต้องเปลี่ยน)
- แก้ไขบั๊ก เช่น กรณีไม่มีหนังสือ หรือหนังสือไม่ว่าง

สรุปและถาม-ตอบ (Duration: 10 นาที)

Day 10: หน้า Dashboard และสรุปโครงการ

ออกแบบหน้า Dashboard (Duration: 60 นาที)

- สร้าง `dashboard.html` ด้วย Tailwind ให้แสดงสรุปข้อมูล (เช่น จำนวนหนังสือทั้งหมด, ยืมไปแล้ว, สมาชิก, ฯลฯ)
- อาจใช้การ์ดหรือกราฟเสริม (ถ้าเวลามี)

ดึงข้อมูลสถิติจาก API (Duration: 40 นาที)

- เขียน API เช่น `GET /stats` หรือใช้หลาย route (`/books`, `/loans`) เพื่อดึงตัวเลขสรุป
- ใช้ Fetch ในหน้า Dashboard มาเติมลงใน HTML (เช่น วนแสดงตัวเลข)

แสดงข้อมูลบน Dashboard (Duration: 40 นาที)

- แสดงรายชื่อสมาชิกใหม่ล่าสุดหรือหนังสือที่ถูกยืมบ่อย (ถ้าออกแบบมา)
- ปรับแต่งด้วย Tailwind ให้น่าใช้งาน

สรุปโครงสร้างโปรเจกต์ (Duration: 100 นาที)

- ทบทวนการทำงานโดยรวม: Frontend ↔ Backend ↔ Database
- สรุปไฟล์สำคัญในโปรเจกต์ (เช่น `app.js`, ไฟล์ route แต่ละอัน, structure ใน VS Code)
- ให้ผู้เรียนลองอธิบาย flow การทำงานของระบบ (ตั้งแต่เปิดหน้าเว็บถึงแสดงข้อมูล)

ทดสอบและแก้ไขปัญหาทั้งระบบ (Duration: 60 นาที)

- ให้ผู้เรียนทดสอบฟีเจอร์ทั้งหมด (เพิ่ม/แก้หนังสือ, ยืม/คืน, ลงทะเบียน/ล็อกอิน)
- แก้ไขบั๊กที่พบ (เช่น Route ผิด, ข้อมูลไม่ถูกบันทึก)

สรุปบทเรียนและถาม-ตอบ (Duration: 60 นาที)

- สรุปหัวข้อสำคัญทั้งหมดที่เรียนมา
- เปิดโอกาสให้ผู้เรียนถามคำถามเพิ่มเติม
- ให้คำแนะนำต่อยอด (เช่น นำไปพัฒนาต่อ, เรียนรู้ security, deployment ในอนาคต)