# ใบงานที่ 1: ติดตั้งระบบ & Healthcheck

**วัตถุประสงค์:**

- รัน API Starter เชื่อมฐานข้อมูล

- ตรวจสอบ /system/health และบันทึกผล

**ความรู้พื้นฐานที่ควรมีก่อนเริ่ม:**

- Node.js + npm

- MySQL หรือ Docker

- Express.js เบื้องต้น

**ขั้นตอนการปฏิบัติ (Step-by-Step):**

1. แตก a5\_compose\_stack.zip หรือ a5\_api\_starter.zip

2. ตั้งค่า .env และสร้างฐานข้อมูลตาม 02\_schema.sql

3. รัน `npm start` หรือ `docker compose up -d --build`

4. ทดสอบ GET /system/health และบันทึกผล

**หลักฐานที่ต้องส่ง (Deliverables):**

- สกรีนช็อต /system/health

- ไฟล์ .env (ซ่อนรหัสก่อนส่ง)

- สรุปปัญหาและวิธีแก้

เกณฑ์การให้คะแนน (รวม 25 คะแนน)

|  |  |  |
| --- | --- | --- |
| รายการประเมิน | รายละเอียดเกณฑ์ | คะแนน |
| ตั้งค่า .env ถูกต้อง | ค่า DB/JWT/PORT ถูกต้อง | 5 |
| /system/health 200 | db: up ภายใน JSON | 10 |
| หลักฐานครบ | สกรีนช็อต + บันทึกขั้นตอน | 10 |

# ใบงานที่ 2: อัปโหลดหลักฐาน (Attachments)

**วัตถุประสงค์:**

- ใช้ POST /api/attachments/upload

- บันทึกเรคอร์ดลง DB

**ความรู้พื้นฐานที่ควรมีก่อนเริ่ม:**

- บัญชี evaluatee/evaluator

- เข้าใจ indicator ↔ evidence\_type

**ขั้นตอนการปฏิบัติ (Step-by-Step):**

1. เตรียม indicators/evidence\_types ข้อมูลขั้นต่ำ

2. เรียกอัปโหลดแบบ form-data: file + metadata

3. ตรวจสอบเรคอร์ดใน attachments และเส้นทางไฟล์

4. ทดลองกรณีข้อมูลไม่ครบและบันทึกผลการทดสอบ

**หลักฐานที่ต้องส่ง (Deliverables):**

- สกรีนช็อตคำตอบ API

- SQL select attachments

- สรุปเคส fail/success

เกณฑ์การให้คะแนน (รวม 25 คะแนน)

|  |  |  |
| --- | --- | --- |
| รายการประเมิน | รายละเอียดเกณฑ์ | คะแนน |
| อัปโหลดสำเร็จ | API ตอบ ok มี path ไฟล์ | 10 |
| ตรวจ DB | เรคอร์ดถูกต้องครบฟิลด์ | 10 |
| รายงานทดสอบ | สรุปเคสชัดเจน | 5 |

# ใบงานที่ 3: Submit ผลประเมิน (+Evidence Rule)

**วัตถุประสงค์:**

- ใช้ POST /api/results/:id/submit

- บังคับแนบหลักฐานถ้ามี mapping

**ความรู้พื้นฐานที่ควรมีก่อนเริ่ม:**

- มี evaluation\_results = draft

- ผ่าน Lab 2

**ขั้นตอนการปฏิบัติ (Step-by-Step):**

1. ทดลอง submit โดยยังไม่มีหลักฐาน (ควร error)

2. แนบหลักฐานตาม mapping แล้ว submit อีกครั้ง (ควรผ่าน)

3. ตรวจค่า status และ submitted\_at

4. สรุปแนวคิด IDOR Guard / Data Integrity

**หลักฐานที่ต้องส่ง (Deliverables):**

- สกรีนช็อต fail/success

- SQL แสดงผลการอัปเดต

- สรุปบทเรียน

เกณฑ์การให้คะแนน (รวม 25 คะแนน)

|  |  |  |
| --- | --- | --- |
| รายการประเมิน | รายละเอียดเกณฑ์ | คะแนน |
| ทดสอบครบเคส | ทั้ง fail และ success | 10 |
| ค่าอัปเดตถูกต้อง | status/submitted\_at ถูกต้อง | 10 |
| สรุปบทเรียน | อธิบายเหตุผลเชิงเทคนิค | 5 |

# ใบงานที่ 4: Reports: Progress & Normalized/60

**วัตถุประสงค์:**

- อ่านผลจาก /api/reports/progress และ /api/reports/normalized

- คำนวณมือเทียบผล API

**ความรู้พื้นฐานที่ควรมีก่อนเริ่ม:**

- มีข้อมูล topics/indicators/results พอสำหรับรายงาน

**ขั้นตอนการปฏิบัติ (Step-by-Step):**

1. เรียก /reports/progress?period=&dept=

2. เรียก /reports/normalized?period=&evaluatee=

3. คำนวณมือ (Excel/CSV) เปรียบเทียบผล

4. อธิบายสูตร normalize 0..1 และถ่วงน้ำหนักหัวข้อ

**หลักฐานที่ต้องส่ง (Deliverables):**

- ผล JSON ทั้งสองทรัพยากร

- ไฟล์คำนวณมือ และคำอธิบายสูตร

เกณฑ์การให้คะแนน (รวม 25 คะแนน)

|  |  |  |
| --- | --- | --- |
| รายการประเมิน | รายละเอียดเกณฑ์ | คะแนน |
| ดึงรายงานสำเร็จ | ตอบ 200 และ logic ถูกต้อง | 10 |
| ตรวจสอบข้าม | ผลใกล้เคียงการคำนวณมือ | 10 |
| อธิบายสูตร | อธิบาย normalize/weight | 5 |

เกณฑ์ผ่านรายใบงาน: ได้อย่างน้อย 15/25 คะแนน (รวม ≥ 60/100 ถือว่าผ่าน).