โครงการระบบประเมินบุคลากร — เอกสารฉบับรวม (Compiled)

อ้างอิงจาก: A5\_อลงกรณ์ส่ง.docx, 02\_schema.sql และโครงสร้างโค้ด backend.zip

# สารบัญ (Table of Contents)

*\*\*หมายเหตุ:\*\* เปิดเอกสารนี้ใน Microsoft Word แล้วกด Update Table เพื่อสร้างสารบัญอัตโนมัติ.*

# บทนำ

เอกสารฉบับนี้รวบรวมคู่มือการติดตั้ง การใช้งาน API/Frontend การปฏิบัติการ (Admin/Ops) แผนทดสอบ DevOps และสคีมา โดยอ้างอิงไฟล์ข้อกำหนดและสคีมาที่ให้มา พร้อมสรุปโครงสร้างโค้ดจาก backend.zip เพื่อเริ่มงานได้ทันที.

# INSTALL — ขั้นตอนติดตั้ง/รันระบบ

# INSTALL — ขั้นตอนติดตั้ง/รันระบบ

## 1) ข้อกำหนดเครื่องมือ (ขั้นต่ำ)

- OS: Windows 10/11, macOS 12+, หรือ Ubuntu 22.04+

- Git, Node.js LTS 20.x, npm

- Docker Engine / Docker Desktop + Docker Compose

- VS Code + ส่วนเสริม (ESLint, Prettier, Vue Language Features, YAML)

## 2) เตรียมฐานข้อมูล (เลือก MySQL หรือ MariaDB)

ตัวอย่าง Compose สำหรับพัฒนา:

```bash

# MySQL

docker-compose -f mysql\_phpmyadmin.yaml up -d --build

# UI Frontend (หากมี): http://localhost:8080

# phpMyAdmin: http://localhost:8081 (host=db, user=root, pass=rootpassword)

# MariaDB

docker-compose -f maria\_phpmyadmin.yaml up -d --build

# UI Frontend: http://localhost:8080

# phpMyAdmin: http://localhost:8081

```

นำเข้าโครงสร้างตัวอย่างจาก `02\_schema.sql` (หรือปรับ/สร้างใหม่ตามออกแบบทีม)

## 3) ตั้งค่าไฟล์ .env (ตัวอย่าง)

```env

# API

PORT=3000

NODE\_ENV=development

JWT\_SECRET=change\_me\_please

# DB

DB\_CLIENT=mysql2

DATABASE\_URL=mysql://root:rootpassword@localhost:3306/skills\_db

# Uploads

UPLOAD\_DIR=./uploads

MAX\_UPLOAD\_MB=10

ALLOWED\_MIME=application/pdf,image/png,image/jpeg,application/zip

```

## 4) ติดตั้งและรัน (โครง Express/Knex + Nuxt 3)

```bash

# Backend

cd backend

npm ci

npm run dev # หรือ npm run start

# Frontend

cd frontend

npm ci

npm run dev

```

ตรวจสุขภาพระบบ: `GET /system/health` (ผ่าน Nginx/Reverse Proxy หากตั้งค่าแล้ว)

## 5) โครงสร้างงาน Git (แนะนำ)

- main (เสถียรพร้อมส่งมอบ), develop (รวมฟีเจอร์), feature/\* (งานย่อย)

- เปิด Pull Request + Code Review ก่อน merge

# API GUIDE — โครงสร้างและตัวอย่าง API

# API GUIDE — โครงสร้างและตัวอย่าง API

## 1) ภาพรวม

- REST + JSON, เอกสาร OpenAPI ให้บริการที่ `GET /openapi.json` และ/หรือ Swagger UI

- Authentication: JWT (Authorization: Bearer <token>)

- พารามิเตอร์สำหรับ List:

- `q` (ค้นหา), `sort=field:asc|desc`, `page`, `pageSize`

ทรัพยากรหลัก: Users, Topics, Indicators, Periods, Assignments, Results, Evidence

## 2) Authentication

```http

POST /api/auth/login

POST /api/auth/register # (กรณีเปิดลงทะเบียน) role เริ่มต้น evaluatee

```

## 3) ตัวอย่างงาน (Task Endpoints)

> ใช้สำหรับทดสอบตามข้อกำหนดและการให้คะแนน

### 3.1 IDOR Guard

```http

GET /task1/evaluation-results?user\_id={uid}&assignment\_id={aid}

# admin: เห็นได้ทุก assignment

# evaluator: เห็นเฉพาะ assignment ของตน

# evaluatee: เห็นเฉพาะของตนเอง

# เกินสิทธิ์ → 403 { "error":"forbidden" }

```

### 3.2 Evidence Submit Rule

```http

PATCH /task2/results/:id/submit

# ถ้า indicator.type = 'yes\_no' และ yes\_no = 1 แต่ไม่มีไฟล์แนบ -> 400 { "error":"EVIDENCE\_REQUIRED" }

# กรณีผ่าน -> อัปเดตเป็น submitted พร้อม submitted\_at

```

### 3.3 Normalized /60

```http

GET /task3/reports/normalized?period\_id=1

# score\_1\_4: r = (score - 1)/3

# yes\_no: 0 หรือ 1

```

### 3.4 Unique Assignment

```http

POST /task4/assignments

# body: { evaluator\_id, evaluatee\_id, period\_id, dept\_id }

# ซ้ำชุด (evaluator\_id, evaluatee\_id, period\_id) -> 409 { "error":"DUPLICATE\_ASSIGNMENT" }

```

### 3.5 Progress by Department

```http

GET /task5/reports/progress?period\_id=1

# คืนอาร์เรย์ { department, submitted, total, percent }

# percent = submitted/total\*100 (ปัด 2 ตำแหน่ง; total=0 → 0)

```

## 4) ตัวอย่างการเรียก List

```http

GET /api/indicators?q=MEDIA&sort=created\_at:desc&page=1&pageSize=10

```

## 5) โครงสร้างโฟลเดอร์ (แนะนำ Backend)

```

backend/

src/

routes/

controllers/

services/

middlewares/

db/ (knex, migrations, seeds)

.env.example

```

> หมายเหตุ: เพิ่มตัวอย่าง Request/Response จริงหลังเริ่มพัฒนาและมี Postman/Swagger

# USER GUIDE — คู่มือผู้ใช้

# USER GUIDE — การใช้งานสำหรับผู้ใช้ตามบทบาท

## บทบาท (Roles)

- Admin: บริหาร Users/Topics/Indicators/Periods/Assignments/Reports

- Evaluator: รับมอบหมายงาน → กรอกผล/อัปโหลดหลักฐาน → ยืนยันส่ง

- Evaluatee: แนบหลักฐาน/ติดตามสถานะ/ดูรายงานส่วนบุคคล

## โครงร่างหน้าใช้งาน (IA)

- `/home` (หรือ `/`) แดชบอร์ดตามบทบาท + ช็อตคัตไป `/reports/normalized` และ `/system/health`

- Admin: `/admin/users`, `/admin/topics`, `/admin/indicators`, `/admin/periods`, `/admin/assignments`, `/admin/results`

- Evaluator: `/evaluator/assignments`, `/evaluator/history`

- Evaluatee: `/me/evaluation`, `/me/evidence`, `/me/report`

- Reports: `/reports/normalized`, `/reports/progress`

## ตัวอย่างขั้นตอน (Evaluatee)

1) Login

2) เปิด `/me/evidence` → อัปโหลดไฟล์ตามชนิดที่กำหนด

3) ตรวจสถานะใน `/me/evaluation` (draft/submitted/locked)

4) พิมพ์รายงานใน `/me/report`

## แนวปฏิบัติการใช้งาน

- ฟอร์มทุกหน้าควรมี validation พร้อมข้อความชัดเจน

- ตาราง (v-data-table) รองรับค้นหา/เรียง/แบ่งหน้า

- แสดง badge สถานะตามค่าจริงจาก API

# ADMIN/OPERATIONS GUIDE

# ADMIN / OPERATIONS GUIDE

## Deploy แบบนักพัฒนา (Compose)

- ใช้ Compose รวม: db + api (สามารถ scale ≥2) + ui + nginx + phpmyadmin

- ตั้งค่า `.env` ให้ชัดเจนสำหรับ DB/JWT/Uploads

## งานดูแลประจำวัน

- ตรวจสุขภาพระบบ: `GET /system/health`

- ดู log: `docker compose logs -f api` (หรือใช้ Loki/Promtail หากมี)

- สำรอง/กู้คืน (ตัวอย่าง):

```bash

# Backup

mysqldump -h 127.0.0.1 -u root -p skills\_db > backup.sql

# Restore

mysql -h 127.0.0.1 -u root -p skills\_db < backup.sql

```

## บริหารผู้ใช้/สิทธิ์

- แก้ไข/ตั้งค่า role: admin / evaluator / evaluatee

- บริหารรอบการประเมิน/มอบหมายกรรมการ

## ปรับปรุงระบบ

- ใช้ Git flow (main/develop/feature/\*)

- เปิด PR และทำ Code Review

# DEVOPS — Compose/Reverse Proxy/Scale

# DEVOPS — Compose, Reverse Proxy/Load Balancer, Scale

## โครง Compose (แนวทาง)

- services: db, api(2 replicas), ui, nginx (reverse proxy), phpmyadmin

- ตัวอย่างปรับแต่ง Nginx ให้ health‑check และกระจายโหลดไปยัง api ทั้งสองอินสแตนซ์

- ยิงโหลดด้วย k6/autocannon → เก็บค่า p95 และวิเคราะห์หากเกินเกณฑ์ (เช่น 500ms)

## สิ่งที่ต้องจัดเตรียม/ส่งมอบ

- `docker-compose.yml` (หรือ `docker-compose.lb.yml`) ที่รันได้จริง

- `nginx.lb.conf` พร้อม upstream หลายอินสแตนซ์

- หลักฐานทดสอบ: ภาพ `docker ps`, ผล `curl /system/health`, รายงาน k6/autocannon

# TEST PLAN — แผนทดสอบแบบบูรณาการ

# TEST PLAN — การทดสอบแบบบูรณาการ

## กลยุทธ์

- Functional: สมัคร/ล็อกอิน, Home ตามบทบาท, CRUD และรายการ, กฎธุรกิจ (Evidence Rule, Unique Assignment)

- Security: IDOR guard (403), auth (401/403), อัปโหลดชนิดต้องห้าม/เกินขนาด (415/413)

- Non-functional: โหลด/สเกล p95, ขนาดไฟล์, ข้อความผิดพลาดอ่านง่าย

## Entry/Exit Criteria

- Entry: ติดตั้ง/รันระบบสำเร็จ, DB พร้อม seed

- Exit: โฟลว์หลักผ่าน, ข้อผิดพลาดร้ายแรงถูกแก้, เอกสารถูกอัปเดต

## Test Cases (อย่างย่อ)

- POST /auth/register → 201/200; POST /auth/login → 200 + token

- GET /home ตามบทบาท (ตรวจด้วยตา)

- GET /task1/evaluation-results unauthorized → 403

- PATCH /task2/results/:id/submit (ไม่มีไฟล์) → 400 EVIDENCE\_REQUIRED; (มีไฟล์) → submitted

- POST /task4/assignments ซ้ำ → 409

- GET /task5/reports/progress → โครงสร้าง/สูตรถูกต้อง

- อัปโหลดไฟล์ >10MB → 413; ชนิดต้องห้าม → 415

- ยิงโหลด k6/autocannon → p95 เทียบเกณฑ์

> แนบหลักฐาน: Postman/Swagger, Newman report, สกรีนช็อต UI/ตาราง, รายงานโหลด

# SCHEMA — สรุปตารางและความสัมพันธ์ (อ้างอิง 02\_schema.sql)

# SCHEMA — พจนานุกรมข้อมูล (อ้างอิง `02\_schema.sql`)

ตารางหลัก: `vocational\_categories`, `vocational\_fields`, `org\_groups`, `departments`, `dept\_fields`, `users`, `evaluation\_periods`, `evaluation\_topics`, `indicators`, `evidence\_types`, `indicator\_evidence`, `assignments`, `evaluation\_results`, `attachments`

ความสัมพันธ์สำคัญ (ตัวอย่าง):

- indicators.topic\_id → evaluation\_topics.id

- indicator\_evidence: (indicator\_id, evidence\_type\_id) PK

- assignments: UNIQUE(period\_id, evaluator\_id, evaluatee\_id)

- evaluation\_results/evidence อ้าง period/evaluatee/indicator ตามเงื่อนไข

> แนะนำให้นำ `02\_schema.sql` ไปเปิดใน DB Tool แล้ว export ER diagram แนบเข้าคู่มือ

# โครงสร้างโค้ดจาก backend.zip (สำรวจเบื้องต้น)

โครงสร้างไฟล์/โฟลเดอร์ (ลึกไม่เกิน 2 ชั้น):

backend/  
 controllers/  
 assignments.controller.js  
 auth.controller.js  
 reports.controller.js  
 results.controller.js  
 upload.controller.js  
 users.controller.js  
 db/  
 knex.js  
 middlewares/  
 auth.js  
 error.js  
 requireAuth.js  
 upload.js  
 node\_modules/  
 @apidevtools/  
 @esbuild/  
 @jest/  
 @jridgewell/  
 @jsdevtools/  
 @mapbox/  
 @noble/  
 @paralleldrive/  
 @rollup/  
 @scarf/  
 @sinclair/  
 @types/  
 @vitest/  
 abbrev/  
 accepts/  
 agent-base/  
 ansi-regex/  
 anymatch/  
 append-field/  
 aproba/  
 are-we-there-yet/  
 argparse/  
 array-flatten/  
 asap/  
 assertion-error/  
 asynckit/  
 aws-ssl-profiles/  
 balanced-match/  
 basic-auth/  
 bcrypt/  
 binary-extensions/  
 body-parser/  
 brace-expansion/  
 braces/  
 buffer-equal-constant-time/  
 buffer-from/  
 busboy/  
 bytes/  
 cac/  
 call-bind-apply-helpers/  
 ...  
 repositories/  
 db.js  
 users.js  
 routes/  
 assignments.routes.js  
 auth.routes.js  
 reports.routes.js  
 results.routes.js  
 upload.routes.js  
 users.routes.js  
 tests/  
 auth.test.mjs  
 health.test.mjs  
 users.server.test.mjs  
 uploads/  
 utils/  
 authz.js  
 app.js  
 Dockerfile  
 openapi.json  
 package-lock.json  
 package.json  
 README.md  
 server.js  
 TESTS.md  
frontend/  
 assets/  
 css/  
 pages/  
 users/  
 index.vue  
 login.vue  
 upload.vue  
 users.new.vue  
 plugins/  
 vuetify.ts  
 stores/  
 auth.ts  
 tests/  
 login.spec.ts  
 setup.ts  
 users.new.spec.ts  
 app.vue  
 Dockerfile  
 nuxt.config.ts  
 package.json  
 postcss.config.js  
 README.md  
 tailwind.config.js  
 TESTS.md  
 vitest.config.ts  
02\_schema.sql  
docker-compose.yml  
docker-compose\_mysql.yml  
postman\_collection.json  
README.md  
README\_A5\_ALIGN.md  
TESTS.md

ไลบรารี/แพ็กเกจที่ตรวจพบ:

(ไม่พบบัญชีรายชื่อไลบรารี)

# ภาคผนวก: รูบริก/เช็กลิสต์การตรวจ

- A5\_Rubric\_Backend\_B1\_B6\_Checklist\_Full.docx — เช็กลิสต์ Backend B1–B6 เงื่อนไขละ 1 คะแนน

- A5\_Rubric\_Frontend\_F1\_F6\_Checklist\_Full.docx — เช็กลิสต์ Frontend F1–F6 เงื่อนไขละ 1 คะแนน

- A5\_Rubric\_Manuals\_User\_Admin\_Checklist.docx — เช็กลิสต์คู่มือผู้ใช้/ผู้ดูแล (5 คะแนน)

- A5\_Rubric\_Backend\_Frontend\_DevOps.docx — รูบริกฉบับรวม