|  |  |  |
| --- | --- | --- |
|  | **เครื่องมือประเมินมาตรฐานวิชาชีพ (ภาคปฏิบัติ)** | **ปวส. 67: A5** |
| **หลักสูตรประกาศนียบัตรวิชาชีพชั้นสูง พุทธศักราช 2567**  **ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ       สาขาวิชาเทคโนโลยีสารสนเทศ** | |
| **สมรรถนะงานที่ 5 งานพัฒนาระบบ**  **เวลา ...5.... ชั่วโมง  คะแนนเต็ม ....60.... คะแนน** | | |

**กรอบการประเมิน**

1. แก้ไขข้อผิดพลาดและความปลอดภัย ระบุจุดบกพร่อง อธิบายสาเหตุเชิงเทคนิค วิเคราะห์ผลกระทบ จัดทำแพตช์อย่างปลอดภัย และปิดช่องโหว่ที่เกี่ยวข้อง
2. พัฒนาแบบบูรณาการ แปลง SRS/Functional Spec/UML เป็นโค้ดที่ทำงานได้จริง เชื่อม API ↔ DB ↔ Frontend รันบน Docker/VM และอธิบายแนวทาง Auto-scale/Load Balancing
3. การทดสอบแบบบูรณาการ — วาง Test Strategy/Plan นิยาม Entry/Exit Criteria จัดทำ Test Case (Functional/Non-functional/Security) รันทดสอบ บันทึกผล จัด Severity/Defect และสรุปเทียบ Exit
4. จัดทำคู่มือ/ตรวจคู่มือ — สร้างคู่มือผู้ใช้และผู้ดูแลระบบที่ใช้ได้จริง อธิบายการทำงาน/การใช้งาน/ข้อจำกัด และตรวจความครบถ้วนด้วย Checklist

**คำสั่ง**

ให้ผู้รับการประเมินปฏิบัติงานตามขั้นตอนและเงื่อนไขต่อไปนี้ ให้แล้วเสร็จในเวลาที่กำหนด

1. **ขั้นเตรียม** ให้จัดเตรียมเครื่องมือ วัสดุ อุปกรณ์ ตามรายการต่อไปนี้ (เวลา 15 นาที)

1.1 สถานศึกษาจัดเตรียม

1.1.1 เครื่องคอมพิวเตอร์ส่วนบุคคล พร้อมระบบปฏิบัติการ และเชื่อมต่ออินเทอร์เน็ต (1 เครื่อง/คน)

1.1.2 โปรแกรมและแอปพลิเคชันที่ใช้ในการพัฒนาซอฟต์แวร์สำหรับอุปกรณ์เคลื่อนที่

1.2 ผู้รับการประเมินจัดเตรียม

1.2.1 ปากกา 1 ด้าม

1.2.2 กระดาษทดหรือจดบันทึก

2. **ขั้นดำเนินการ** ให้ปฏิบัติงานตามขั้นตอนและเงื่อนไขที่กำหนดดังนี้ (เวลา 5 ชั่วโมง)

หัวข้อ “การพัฒนาระบบประเมินบุคลากร”

2.1.) **เครื่องมือพัฒนาและรันไทม์พื้นฐาน**

**2.1.1. ระบบปฏิบัติการสำหรับนักพัฒนา**: Windows 10/11, macOS 12+, หรือ Ubuntu 22.04+

2.2.2. **ตัวจัดการเวอร์ชันซอร์สโค้ด**: Git (โฮสต์บน GitHub/GitLab/Bitbucket)

2.2.3 **สภาพแวดล้อมรันโค้ดฝั่งเซิร์ฟเวอร์**: Node.js LTS 20.x พร้อม npm

2.2.4 **เครื่องมือจำลอง/คอนเทนเนอร์**: Docker Engine/ Docker Desktop + Docker Compose

2.2.5 **โปรแกรมแก้ไขโค้ด**: Visual Studio Code (ส่วนเสริม: ESLint, Prettier, Vue Language Features(Volar), YAML)

**2.2) ส่วนพัฒนา Backend (API) มีโปรเจคเริ่มต้น**

**#git clone** [**https://github.com/kruoakai/competency2568.git**](https://github.com/kruoakai/competency2568.git)

**สิ่งที่ต้องทำ:**

1. อ่านและทำความเข้าใจคู่มือที่โหลดมา พร้อมติดตั้งตามขั้นตอน

**หมายเหตุ**

**ถ้านักศึกษา ออกแบบระบบเองทั้งหมด ไม่ต้องใช้ ข้อมูลที่จัดเตรียมไว้**

**2.2.1. ภาษาและเฟรมเวิร์ก**

**-** Node.js 20.x, **Express 4.x** (REST API), **Knex 3.x** (เชื่อมฐานข้อมูลด้วย Query Builder + รองรับ SQL ตรง/raw)

2.2.2. **ฐานข้อมูล**

**- MySQL 8.x**

**- MariaDB 10.6+ (LTS) / 10.11+** (กรณีใช้แทน MySQL)

- **เครื่องมือช่วยออกแบบ/บริหารฐานข้อมูล (DB Tools) MySQL Workbench** ,**TablePlus**,

**phpMyAdmin, HeidiSQL (Windows)**

**ตัวอย่าง** dockercompose file ใช้ประกอบการพัฒนา และโครงสร้างฐานข้อมูลตัวอย่างที่โหลดมาจากหัวข้อ 2.2

1. เลือกทางเลือกที่ต้องการกรณีใช้ตัวอย่างที่เตรียมไว้
   1. **MySQL**

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

# UI: http://localhost:8080

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

**1.2. MariaDB**

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

# UI: http://localhost:8080

# phpMyAdmin: http://localhost:8081

2.2.3. **ไลบรารีหลัก**

**-** ความปลอดภัยและมาตรฐาน: helmet, cors, dotenv

- ยืนยันตัวตน/สิทธิ์: jsonwebtoken (JWT), bcryptjs (แฮชรหัสผ่าน)

- อัปโหลดไฟล์หลักฐาน: multer, ตรวจชนิดไฟล์ด้วย file-type

2.2.4. เครื่องสร้าง **เอกสาร API และทดสอบสุ่มโหลด**

**-** ไฟล์ **OpenAPI (openapi.json)** สำหรับอธิบายสัญญา API และทดสอบร่วมกับ Postman/Insomnia

**-** Swagger UI: เครื่องมือสร้างเอกสาร API แบบโต้ตอบ (interactive) ที่ใช้งานง่ายและสามารถนำไปแสดงผลได้ทันที

**-** Redoc: อีกหนึ่งเครื่องมือที่ใช้สร้างเอกสาร API จากไฟล์ OpenAPI แต่เน้นการแสดงผลที่สวยงามและอ่านง่ายกว่า

- Postman/Insomnia (Functional Test), k6/JMeter (Load Test) – ตามกรณีใช้งาน

2.2.5. **การกำหนดคอนฟิก** ไฟล์ .env สำหรับ DATABASE\_URL, JWT\_SECRET, พาธจัดเก็บไฟล์อัปโหลด และค่ากำหนดอื่น ๆ ตัวอย่างอยู่ในข้อมูลที่โหลดมา

2.3. **ส่วนพัฒนา Frontend (ผู้ใช้/ผู้ประเมิน/ผู้ดูแล) มีตัวอย่างไฟล์ประกอบการพัฒนา frontend และนักศึกษาสามารถออกแบบเริ่มต้นใหม่ได้ตามที่ถนัด**

**ขั้นใช้โปรเจคตัวอย่าง**

* + 1. **สแตกเฟรมเวิร์ก**

**- Nuxt 3 (Vue 3)**, **Vuetify 3** (UI Components), **Tailwind CSS** (Utility-first CSS), Vite (Bundler /Dev server)

2.3.2. **ไลบรารีประกอบ**

**-** การเรียก API: $fetch หรือ axios

- ไอคอน/กราฟิก: @mdi/font (ตามต้องการ), Chart.js/echarts (หากต้องมีกราฟสรุปผล)

- คุณภาพโค้ด: ESLint + Prettier

**2.4. เครื่องมือโครงสร้างพื้นฐานและปฏิบัติการ (DevOps)**

**2.4.1. Reverse Proxy / Load Balancer**: Nginx (กำหนด upstream สำหรับหลายอินสแตนซ์ API)

2.4.2. **CI / CD**: GitHub Actions / GitLab CI (Test–Build–Package–Deploy)

2.4.3. **การตรวจสอบ/สังเกตการณ์** (เลือกใช้ตามสเกล): Prometheus + Grafana (Metrics), Loki (Logs), OpenTelemetry (Tracing)  
 2.4.4. **สคริปต์ Compose สำหรับนักพัฒนา**: docker-compose.yml ที่บูท **MySQL + API (หลายอินสแตนซ์) + UI + Nginx** เพื่อทดสอบแบบบูรณาการ

2.5. **แนวทางการกำหนดเวอร์ชันและการจัดการแพ็กเกจ**

**2.5.1** กำหนดเวอร์ชันขั้นต่ำ (Node 20.x, MySQL 8.x, Nuxt 3.x, Vue 3.4+, Vuetify 3.6+)

2.5.2. ใช้ไฟล์ package.json ระบุ dependencies ชัดเจน และ package-lock.json สำหรับ reproducible builds

2.5.3. ใช้สาขา Git: main (เสถียร), develop (ฟีเจอร์รวม), และ feature/\* (งานย่อย) พร้อม Pull Request + Code Review

**2.3) การสร้างฐานข้อมูลสำหรับโปรแกรมพัฒนาระบบการประเมินบุคลากร**

**ทรัพยากรที่เตรียมไว้แล้วในชุดงาน หัวข้อ** 2.2.2. **ฐานข้อมูล เป็นโครงสร้างตัวอย่างสามารถ สร้างตารางเพิ่มเติมได้ตามออกไว้ หรือ ลบโครงสร้างตัวอย่างทั้งหมด แล้วสร้างใหม่ตามที่ออกแบบไว้**

* **เครื่องมือจัดการฐานข้อมูล: phpMyAdmin (port 8081) ใช้ตรวจสอบ และแก้ไข โครงสร้างของฐานข้อมูล**

2.3.1 ศึกษาความต้องการของระบบการประเมินบุคลากร

- รองรับบทบาท 3 กลุ่ม: Admin(ผู้ดูแลระบบ), Evaluator (กรรมการ), Evaluatee (ครูที่ถูกประเมิน)

- โครงสร้างการประเมินแบบ หัวข้อ (Topics) → ตัวชี้วัด (Indicators)

- กำหนด รอบการประเมิน (Evaluation Period), การมอบหมาย (Assignments)

ระหว่างกรรมการ ↔ ครู

- แนบ หลักฐาน (Evidence/Attachments) ต่อผลการประเมิน

- สรุปผลตามตัวบ่งชี้

**หมายเหตุ**: ตัวอย่างโครงสร้างอยู่ใน 02\_schema.sql แล้ว และถูกใช้งานจริงโดย API+Frontend ในแพ็กเกจ หรือ ออกแบบ schema.sql ใหม่ทั้งหมด

**2.4 การเขียนโปรแกรมพัฒนาระบบการประเมินบุคลากร**

**2.4.1 สร้างโปรแกรมโดยใช้เครื่องมือและ Framework ที่เหมาะสม**

**Backend**

* **Express 4** + **Knex (mysql2)**, โครงสร้าง routes/, controllers/, services/, middlewares/
* มาตรฐาน REST + JSON, เอกสาร **OpenAPI** ที่ GET /openapi.json
* ความปลอดภัยพื้นฐาน: CORS, Helmet, JWT (Authorization: Bearer <token>)
* จัดการไฟล์หลักฐาน: multer เก็บไฟล์แบบ disk (หรือ Object Storage ได้ถ้าต่อยอด)

**Frontend**

* **Nuxt 3** (Vue 3) + **Vuetify 3** + **Tailwind**
* โครง route ตามบทบาท (/admin/\*, /evaluator/\*, /me/\*, /reports/\*)
* ใช้ v-data-table, v-form, v-dialog, v-file-input, v-alert เป็นหลัก

**DevOps**

* Docker Compose รวม db + api(2 replicas) + ui + nginx + phpmyadmin
* เลือก **MySQL** หรือ **MariaDB** ได้ (มี compose ให้ทั้งสอง พร้อม volumes ถาวร)

**2.4.2 สร้างหน้าจอหลักของโปรแกรม (Home Page)**

**/home (หรือ /)**: แดชบอร์ดแยกตามบทบาท

* แสดง “บทบาท/สิทธิ์” ของผู้ใช้ล็อกอิน
* **Admin**: ตัวเลขสรุป ผู้ใช้/หัวข้อ/ตัวชี้วัด/รอบ/การมอบหมาย + ลิงก์ไปยังหน้าบริหาร (Users/Topics/Indicators/Periods/Assignments + ปุ่มไปยัง /admin/\*)
* **Evaluator**: รายการมอบหมายที่ต้องประเมิน (สถานะ: *draft/submitted/locked*)
* **Evaluatee**: งานประเมินของฉัน + สถานะการแนบหลักฐาน  
  **Acceptance Criteria (AC) ที่ยอมรับได้:**
* - แสดงบทบาทของผู้ใช้ที่ล็อกอินอย่างชัดเจน
* - มี Shortcuts ไป /reports/normalized และ /system/health
* - ใช้ v-data-table/v-card (Vuetify) และเงื่อนไขแสดงผลตาม บทบาทของผู้ใช้ (role-based conditional rendering)

**2.4.4 การสมัครสมาชิกเพื่อใช้งานระบบ (Registration)**

* ฟอร์ม POST /auth/register (อีเมล, ชื่อ, รหัสผ่าน, แผนก,ฯลฯ) → role=Evaluatee โดยปริยาย

POST /auth/register — รับ (email, name, password, department\_id) → บันทึก role=Evaluatee โดยปริยาย (hash password)

* ส่งอีเมลยืนยัน (ถ้าต่อยอด) หรืออนุมัติ role โดย Admin ใน /admin/users

- ล็อกอิน /auth/login (ใช้ v-form + validation)   
 **โครงร่างหน้าใช้งาน (Role-based Information Architecture)**

**1.Admin**

1. **Users** (/admin/users)
   * List/Search/Filter, Create/Edit/Delete, ตั้ง **Role** (Admin/Evaluator/Evaluatee),

ผูกแผนก/กลุ่มงาน

1. **Evaluation Topics** (/admin/topics)
   * CRUD หัวข้อ (A5-1..A5-4), กำหนด **weight** (15/21/15/9) และสถานะ active
2. **Indicators** (/admin/indicators)
   * CRUD ตัวชี้วัด ต่อ Topic, กำหนด type (score\_1\_4/yes\_no), ช่วงคะแนน, น้ำหนัก
   * ผูกชนิดหลักฐาน (indicator\_evidence)
3. **Evaluation Periods** (/admin/periods)
   * CRUD รอบ (ชื่อ/วันเริ่ม–สิ้นสุด/active), ยกเลิกรอบ
4. **Assignments** (/admin/assignments)
   * มอบหมายกรรมการ ↔ ครู (ตาม period/department), ปรับปรุง/ยกเลิก
5. **Evaluation Results (overview)** (/admin/results)
   * สรุปผลตามหัวข้อ/ตัวชี้วัด/บุคคล/แผนก, Export CSV/Excel/PDF
6. **Reports** (/reports/\*)

* **Normalized 60** (/reports/normalized) + รายงานความคืบหน้า

**2.Evaluator (กรรมการ)**

1. **My Assignments** (/evaluator/assignments)
   * รายการครูที่ต้องประเมิน (filter by period/department/status)
2. **Fill Scores** (/evaluator/assignments/:id)
   * แบบกริดตัวชี้วัด (score/yes-no/note), อัปโหลดไฟล์หลักฐานประกอบ, **บันทึก**/*ยืนยันผล*
3. **History** (/evaluator/history)
   * ประวัติการส่ง/แก้ไขผลของฉัน (ตาม period)

**3. Evaluatee (ครูผู้ถูกประเมิน)**

1. **My Evaluation** (/me/evaluation)
   * ดูหัวข้อ/ตัวชี้วัดของฉัน, สถานะผล (draft/submitted/locked) และคะแนนที่ได้รับ
2. **Upload Evidence** (/me/evidence)
   * อัปโหลดไฟล์หลักฐานตามรายการที่ระบบกำหนด (type/ข้อกำหนด)
3. **Personal Report** (/me/report)
   * พิมพ์รายงานผลส่วนบุคคล (PDF/Print)

**4.รายงาน (Reports)**

* **Normalized 60** (/reports/normalized) — เลือก period/department → ตาราง A5-1..A5-4 และรวม /60
* **Progress** (/reports/progress) — % การประเมินเสร็จสิ้นต่อ period/แผนก

**2.4.5 “การเพิ่ม” (Create) — กรอบงานที่ต้องทำ**

* **Admin**: เพิ่ม Users/Topics/Indicators/Periods/Assignments
  + แบบฟอร์ม v-form + ตรวจความถูกต้อง (required, range, enum)
* **Evaluator**: เพิ่ม “ผลการประเมิน” (score/yes-no/note) ต่อ indicator ของ assignment
  + สถานะเริ่ม draft → submitted เมื่อยืนยัน
* **Evaluatee**: เพิ่ม/อัปโหลด Evidence (ไฟล์) ผูกกับ result\_id ตาม type ที่กำหนด

**2.4.6 “แสดงรายการ” (Read/List)**

* หน้า List ทุกโมดูลใช้ **v-data-table** + ค้นหา/เรียง/กรอง/แบ่งหน้า
* ฝั่ง API รองรับ query params: ?q=&page=&pageSize=&sort=
* แสดงคอลัมน์สำคัญ + ปุ่ม **ดู/แก้ไข/ลบ** ตามสิทธิ์

**2.4.7 “แสดงสถานะ” (Status/State)**

* **ผลการประเมิน**: draft (กำลังกรอก) / submitted (ยืนยันแล้ว) / locked (ปิดรอบ/ปิดแก้ไข)
* **การมอบหมาย**: active / cancelled
* **รอบการประเมิน**: active / closed

ใช้ badge สี (success/warning/error) และ filter ตามสถานะ

**2.4.8–2.4.10 “สร้างหน้าจอ/แสดงรายการ” (ขยายจาก 2.4.5–2.4.7)**

* นักศึกษาต้อง **สร้างหน้าจอครบชุด** สำหรับ **Users/Topics/Indicators/Periods/Assignments/Results/Evidence/Reports**
* ครบ **CRUD** + **List/Filter/Status** + **Dialog ยืนยัน**
* รองรับ **อัปโหลดไฟล์** (จำกัดขนาด/ชนิดไฟล์) และแสดงรายการไฟล์ที่แนบ

**2.4.11 “แสดงหน้าจอประวัติ” (History/Audit)**

* หน้าประวัติแยกตามบทบาท
  + Evaluator: ประวัติการส่งผล/แก้คะแนน
  + Evaluatee: ประวัติการแนบหลักฐาน/อัปเดต
* แหล่งข้อมูลขั้นต่ำ: ใช้ created\_at/updated\_at/submitted\_at ของ evaluation\_results และ attachments
* (ต่อยอด) เพิ่มตาราง evaluation\_result\_logs เพื่อเก็บเวอร์ชัน/ผู้แก้ไข
  1. **แก้ไขข้อผิดพลาดและความปลอดภัย**

**2.5.1 IDOR Guard**

* **URL: GET /task1/evaluation-results?user\_id=...&assignment\_id=...**
* **เงื่อนไข:**
  + **admin ดูได้ทุก assignment**
  + **evaluator ดูได้เฉพาะ assignment ที่ตัวเองเป็นผู้ประเมิน**
  + **evaluatee ดูได้เฉพาะ assignment ของตนเอง**
* **ถ้าผิดสิทธิ์ → 403 พร้อม { error: 'forbidden' }**
* **ถ้าถูกสิทธิ์ → 200 + ข้อมูลผลเฉพาะ assignment\_id นั้น**

**2.5.2 Evidence Submit Rule**

* **URL: PATCH /task2/results/:id/submit**
* **เงื่อนไข: ถ้า indicator.type='yes\_no' และ yes\_no=1 แต่ไม่มีไฟล์ใน attachments ของผลนั้น → 400 { error:'EVIDENCE\_REQUIRED' }**
* **กรณีอื่น ๆ → อัปเดตเป็น submitted พร้อม submitted\_at**

**2.5.3 Normalized /60**

* **URL: GET /task3/reports/normalized?period\_id=1**
* **สูตร:**
  + **score\_1\_4: r=(score−1)/3r=(score-1)/3r=(score−1)/3**
  + **yes\_no: 0 หรือ 1**

**2.5.4 Unique Assignment**

* **URL: POST /task4/assignments (body: { evaluator\_id, evaluatee\_id, period\_id, dept\_id })**
* **ถ้าซ้ำ (evaluator\_id, evaluatee\_id, period\_id) → 409 { error:'DUPLICATE\_ASSIGNMENT' }**
* **ถ้าไม่ซ้ำ → แทรกข้อมูลและคืน row**

**2.5.5: Progress by Department (กลาง)**

* **URL: GET /task5/reports/progress?period\_id=1**
* **คืนอาร์เรย์ { department, submitted, total, percent }**
* **% = submitted/total\*100 (ปัด 2 ตำแหน่ง; total=0 → 0)**
  1. การทดสอบแบบบูรณาการ

Test Cases เกณฑ์การทดสอบของระบบ

|  |  |  |  |
| --- | --- | --- | --- |
| หมวด | Test Case(TC) | Input/Step(ขั้นตอนการทดสอบ) | Expected (รหัสสถานะ/เนื้อหา/ผลลัพธ์ที่คาดหวัง) |
| Functional | สมัคร/ล็อกอิน (ถ้ามี) | 1) POST /auth/register (optional) 2) POST /auth/login บันทึก token เป็น {{jwtToken}} | Register: 201/200 Login: 200 + JSON มี token |
| Functional | Home ถูกบทบาท (ตรวจด้วยตา) | เปิด /home (หรือหน้า mock ใน Frontend) หลัง login ด้วยบทบาทต่าง ๆ | Admin: การ์ดสรุป + ลิงก์ /admin/\* Evaluator: My Assignments Evaluatee: My Evaluation + หลักฐาน |
| Functional | ดูผลของตนเอง | GET /task1/evaluation-results?user\_id=3&assignment\_id=10 | 200 + rows เฉพาะ assignment 10 |
| Functional | มอบหมายซ้ำ | POST /task4/assignments (แทรกรายการที่มีอยู่แล้ว) | 409 DUPLICATE\_ASSIGNMENT |
| Security | IDOR | GET /task1/evaluation-results?user\_id=3&assignment\_id=11 (ไม่ใช่เจ้าของ) | 403 forbidden |
| Security | Evidence Rule | ลบ attachments ของ result\_id=101 → PATCH /task2/results/101/submit | 400 EVIDENCE\_REQUIRED |
| Non-functional | อัปโหลดไฟล์ >10MB | POST /me/evidence (ถ้ามี endpoint) แนบไฟล์ใหญ่กว่า 10MB | 413 Payload Too Large |
| Non-functional | ชนิดไฟล์ต้องห้าม | POST /me/evidence แนบ .exe | 415 Unsupported Media Type |
| เพิ่มเอง1 |  |  |  |
| เพิ่มเอง2 |  |  |  |

2.8 Docker/VM + Auto-scale/Load Balancing  
  
**เนื้อหาข้อสอบ**

* ใช้ lb\_scale\_pack.zip ที่ทำไว้เป็นสตาร์ตเตอร์
* งานที่ต้องทำ: Compose+Scale api=2, ทดสอบ Reverse Proxy (health), เปิดหน้า Nuxt (3 กล่อง), ยิงโหลดด้วย k6/autocannon และสรุปผล p95<500ms (ถ้าเกินให้วิเคราะห์+ปรับแล้วเทสซ้ำ)
* ส่งหลักฐาน: ภาพ docker ps, ผล curl /system/health, สรุป k6/autocannon, และไฟล์คอนฟิกจริง (docker-compose.lb.yml, nginx.lb.conf) + รายงานสั้น

**หมายเหตุสำหรับผู้พัฒนาและผู้ตรวจระบบ**

**การพัฒนาระบบนี้มีวัตถุประสงค์เพื่อประเมินความสามารถในการเขียนโค้ดและพัฒนาฟีเจอร์ให้เป็นไปตามโจทย์ที่กำหนด ผู้พัฒนาสามารถเลือกใช้ Technology Stack ที่มีความถนัดได้ตามความเหมาะสม โดยมีเงื่อนไขดังต่อไปนี้:**

1. **การทำงานของระบบ (Functional Requirements):**
   * **ระบบจะต้องสามารถทำงานได้ครบถ้วนตามความต้องการที่ระบุไว้ในโจทย์ (เช่น การจัดการข้อมูล, การประมวลผล, การแสดงผล)**
   * **ฟีเจอร์หลักทั้งหมดจะต้องทำงานได้อย่างถูกต้องและมีประสิทธิภาพ**
2. **ประสิทธิภาพและคุณภาพของโค้ด (Performance & Code Quality):**
   * **โค้ดที่เขียนขึ้นควรมีคุณภาพดี อ่านง่าย และสามารถบำรุงรักษาได้ในอนาคต**
   * **ประสิทธิภาพของระบบโดยรวมจะต้องอยู่ในเกณฑ์ที่ยอมรับได้**
3. **การส่งมอบงาน:**
   * **ผู้พัฒนาต้องจัดเตรียมคู่มือหรือเอกสารที่ระบุ Technology Stack ที่ใช้ และวิธีการติดตั้ง/รันระบบอย่างชัดเจน เพื่อให้ผู้ตรวจสามารถตรวจสอบผลงานได้อย่างสะดวก**
   * **งานที่ส่งมอบจะต้องประกอบด้วย Frontend และ Backend ที่เชื่อมต่อกันและสามารถทำงานร่วมกันได้อย่างสมบูรณ์**

***ตัวอย่างเช่น: หากผู้พัฒนาถนัด Angular, React หรือ Svelte สามารถใช้เป็น Frontend ได้แทน Nuxt3 และสามารถใช้ Express หรือ NestJS กับฐานข้อมูลอื่น ๆ เช่น PostgreSQL, MongoDB แทน Knex และ MySQL/MariaDB ได้ โดยไม่ขัดต่อเงื่อนไขข้างต้น***

**รายการประเมินและน้ำหนักคะแนน**

|  |  |
| --- | --- |
| **รายการประเมิน** | **คะแนน** |
| **1. ด้านกระบวนการ** | **15** |
| 1.1 ขั้นเตรียม | (4) |
| 1.2 ขั้นดำเนินการ | (11) |
| **2. ด้านผลงาน** | **40** |
| **3. ด้านพฤติกรรมลักษณะนิสัย** | **5** |
| **รวมคะแนน** | **60** |

**แบบประเมินและเกณฑ์การให้คะแนน**

| **รายการประเมิน** | **ผลการประเมิน** | | **เกณฑ์การให้คะแนน** |
| --- | --- | --- | --- |
| **คะแนนเต็ม** | **คะแนนที่ได้** |
| **1. ด้านกระบวนการ (Process)** | **15** |  |  |
| **1.1 ขั้นเตรียม (4 คะแนน)** |  |  |  |
| 1.1.1 เตรียมความพร้อมด้านเครื่องมือ/อุปกรณ์ | 2 |  | มีเครื่องมือและอุปกรณ์ตามที่ระบุครบถ้วนพร้อมใช้งาน (2 คะแนน) |
| 1.1.2 เริ่มต้นปฏิบัติงานภายในเวลาที่กำหนด | 2 |  | เริ่มต้นปฏิบัติงานได้ทันทีหลังสิ้นสุดขั้นเตรียม  (2 คะแนน) |
| **1.2 ขั้นปฏิบัติ (11 คะแนน)** |  |  |  |
| 1.2.1 **เครื่องมือพัฒนาและรันไทม์พื้นฐาน 5 หัวข้อย่อย** | 2 |  | - เตรียมได้ 1 – 3 ข้อ (1 คะแนน)  - เตรียมได้ 4 – 5 ข้อ (1 คะแนน) |
| 1.2.2 **ส่วนพัฒนา Backend (API)** | 5 |  |  |
| **- ภาษาและเฟรมเวิร์ก** | 1 |  | - เตรียมได้พร้อมพัฒนา(1 คะแนน)  - เตรียมได้ไม่พร้อมพัฒนา(0 คะแนน) |
| **- ฐานข้อมูล** | 2 |  | - สร้าง databases ของระบบ พร้อมฐานข้อมูลตั้งต้นได้มากกว่า  4 ตาราง (2 คะแนน)  2-3 ตาราง (1 คะแนน)  1 ตาราง (0 คแนน) |
| - **ไลบรารีหลัก** | 1 |  | 3 ข้อ (1 คะแนน)  2 ข้อ (0.5 คะแนน)  1 ตาราง (0 คะแนน) |
| - เครื่องสร้าง **เอกสาร API และทดสอบสุ่มโหลด** | 1 |  | มี (1 คะแนน)  ไม่มี (0 คะแนน) |
| 1.2.3 **ส่วนพัฒนา Frontend (ผู้ใช้/ผู้ประเมิน/ผู้ดูแล)** | 6 |  |  |
| **- สแตกเฟรมเวิร์ก** | 2 |  | - เตรียมได้พร้อมพัฒนาครบตามข้อหนดของโจทย์ (2 คะแนน)  - เตรียมได้ไม่พร้อมพัฒนา(0 คะแนน) |
| **- ไลบรารีประกอบ** | 2 |  | 3 ข้อ (2 คะแนน)  2 ข้อ (1 คะแนน)  1 ข้อ (0.5 คะแนน) |
| **- เครื่องมือโครงสร้างพื้นฐานและปฏิบัติการ (DevOps)** | 2 |  | ทำได้ 1 ข้อ (2 คะแนน) |
|  |  |  | - |
| **2. ด้านผลงาน (Product)** | **40** |  |  |
| **2.1 Backend (API) (20 คะแนน)** | **20** |  |  |
| **ความครบถ้วนของโมดูล API (CRUD)** สำหรับ Users, Topics, Indicators, Periods, Assignments, Results, Evidence + query params ?q=&page=&pageSize=&sort= **ทรัพยากรที่เกี่ยวข้อง:** Users, Topics, Indicators, Periods, Assignments, Results, Evidence **หลักฐานรวม:** Postman Collection หรือ Swagger/OpenAPI (ครอบคลุมทุก endpoint) + ผลรันทดสอบจริง |  |  |  |
| - สเปกพร้อมใช้งาน  เปิด Swagger UI หรือ Postman (Import Postman Collection) แล้วเรียกได้ทุกหมวด (Users/Topics/Indicators/Periods/Assignments/Results/Evidence) อย่างน้อยหมวดละ 1 endpoint + | 1 |  | ไฟล์ openapi.yaml/json หรือ Postman\_Collection.json + สกรีนช็อต |
| - Search (q) ทำงานถูกต้อง GET /api/<resource>?q=keyword บนอย่างน้อย 2 หมวด (เช่น users, indicators) → ผลลัพธ์ต้องสัมพันธ์กับคีย์เวิร์ด และ q ที่ไม่มีข้อมูลต้องคืนลิสต์ว่าง (ไม่ error) | 1 |  | - ค่าคำขอ (request) + ค่าคำตอบ (response) 2 กรณี: พบ/ไม่พบ |
| -Sort (sort=field:asc/desc) ถูกต้อง เรียก 2 ครั้งด้วย asc และ desc บนฟิลด์เดียวกัน (เช่น created\_at) → ลำดับรายการต้องกลับด้านกันจริง | 1 |  | สกรีนช็อต/Log ผลลัพธ์ก่อน–หลัง พร้อมไฮไลต์ความต่างของลำดับ |
| - Pagination ถูกต้อง GET /api/<resource>?page=2&pageSize=10 ต้องได้ 10 แถว (หรือจำนวนน้อยลงในหน้าท้าย) และ response มี meta.total, meta.page, meta.pageSize ถูกต้อง (ตรวจ page=1 และหน้าท้ายสุด) | 1 |  | 2–3 ชุด request+response (หน้าแรก/กลาง/ท้าย) พร้อมตรวจค่า meta |
| -คิวรีรวม (q+sort+page) คงเสถียร เรียกซ้ำ GET /api/<resource>?q=…&sort=…&page=…&pageSize=… 2 รอบ ต้องได้ผล **เหมือนเดิมทุกประการ** (idempotent) | 1 |  | ค่าคำตอบสองรอบ (diff เท่ากัน) หรือ Postman Test ที่ assert เท่ากัน |
| - Authentication & 401 POST /api/auth/login ด้วยบัญชีที่ให้มา → ได้ 200 + access\_token; เรียก GET /api/results โดย **ไม่ใส่** token → ต้อง 401 | 1 |  | Log คำขอ/คำตอบทั้งสองกรณี (มี/ไม่มี token |
| - Admin ทะลุมองเห็นทั้งหมด (สิทธิ์ถูกต้อง) ใช้ token admin เรียก GET /api/results?period\_id=… → เห็นผลหลายผู้ถูกประเมิน (มากกว่า 1 คน) โดยไม่ถูกจำกัดด้วย assignments ของตน | 1 |  | Response ตัวอย่าง (ตัดเฉพาะฟิลด์จำเป็น) + ชี้ให้เห็นว่ามีหลาย evaluatee\_id |
| - Evaluator: อนุญาตเฉพาะงานที่ได้รับมอบหมาย (403 เมื่อเกินสิทธิ์) ใช้ token evaluator เรียกผล **ที่อยู่ใน assignments ของตน** → 200; และลองเข้าถึงผล **ของคนที่ไม่อยู่ใน assignments → 403** | 1 | 1 | 2 ชุด request+response: case allowed (200) และ case forbidden (403) |
| -Evaluatee: เห็นเฉพาะผลของตน (403 เมื่อเกินสิทธิ์)  ใช้ token evaluatee เรียกผล ของตนเอง → 200; และลองเข้าถึงผล ของคนอื่น → 403 | 1 |  | 2 ชุด request+response: case self (200) และ case others (403) |
| -Score Range Validation (score\_1\_4) POST/PUT ผลด้วย score=0 หรือ 5 → ต้อง 400 พร้อมข้อความชัดเจน; score 1–4 → 200/201 | 1 |  | Request+Response 2 เคส (นอกช่วง=400, ในช่วง=ผ่าน) |
| - Evidence Submit Rule (yes\_no=1 → ต้องมีไฟล์ก่อน submit)  PATCH /results/:id status=submitted โดย value\_yes\_no=1 แต่ไม่มีไฟล์ใน attachments → 400 EVIDENCE\_REQUIRED; กรณีมีไฟล์ → ผ่าน | 1 |  | สองกรณี Before/After แนบไฟล์ + Response code |
| -413 Payload Too Large (จำกัดขนาดไฟล์)  อัปโหลดไฟล์ > ขนาดกำหนด (เช่น >10MB) → 413 | 1 |  | ตัวอย่างไฟล์/สคริปต์อัปโหลด + Response 413 |
| -415 Unsupported Media Type (ชนิดไฟล์ต้องห้าม)  อัปโหลดไฟล์ชนิดต้องห้าม (เช่น .exe) → 415 พร้อมข้อความอธิบาย | 1 |  | Request+Response 415 + รายการ MIME ที่ยอมรับ/ไม่ยอมรับ |
| -Create Assignment สำเร็จ (กรณีไม่ซ้ำ)  POST /assignments ด้วยชุดข้อมูลใหม่ → 201 และ GET ยืนยันว่ามี 1 ระเบียน | 1 |  | Request+Response 201 + GET แสดงระเบียนที่สร้าง |
| - Duplicate Assignment → 409  POST /assignments ซ้ำชุดเดิม (period\_id,evaluator\_id,evaluatee\_id) → 409 DUPLICATE\_ASSIGNMENT | 1 |  | Request+Response 409 พร้อม error code |
| - Reports: Normalized /60 ถูกสูตร  /reports/normalized?period\_id=…&evaluatee\_id=… → ได้ฟิลด์ A5-1..A5-4 + total และค่าตรงกับการคำนวณมือ (r=(score-1)/3, yes\_no=0/1, ชั่งน้ำหนัก indicators & topics) | 1 |  | Response JSON + ตารางคำนวณมือ (แนบไฟล์หรือภาพ) |
| - Reports: Progress per Department ถูกต้อง  GET /reports/progress?period\_id=… → ตรวจ submitted/total/percent ต่อแผนก ตรงกับการนับจากฐานข้อมูลจริง | 1 |  | Response JSON + สคริปต์/คิวรีที่นับเทียบ |
| -Filters/Idempotency เสถียร ลองเปลี่ยนพารามิเตอร์ filter (period\_id/department\_id/evaluatee\_id) แล้วผลลดลง/ตรงเงื่อนไข; เรียกซ้ำพารามิเตอร์เดิม 2 รอบ → ผลเท่ากัน Login → ดู assignments → อัปโหลด evidence → กรอกผล → submit → ตรวจผลใน reports | 1 |  | Request+Response หลายชุด + ผลซ้ำเท่ากัน |
| - Functional E2E (อย่างน้อย 1 เส้นทางครบ)  Login → ดู assignments → อัปโหลด evidence → กรอกผล → submit → ตรวจผลใน reports | 1 |  | สรุปผลรันจาก Postman/Newman แสดงทุกขั้นตอนผ่าน |
| -Security/Non‑functional ในชุด E2E  แทรก TC IDOR (403) + อัปโหลดไฟล์เกินกำหนด (413) หรือชนิดต้องห้าม (415) ในชุดเดียวกัน แล้วสรุปผลผ่าน | 1 |  | รายงานรวม (HTML/JSON) จาก Newman/k6/JMeter ระบุ TC ดังกล่าวผ่าน |
| 2.2 Frontend (ผู้ใช้/ผู้ประเมิน/ผู้ดูแล) (15 คะแนน) | 15 |  |  |
| -เมนู/หน้า Home แตกต่างตามบทบาท Login เป็น 3 บทบาท → ตรวจเมนู/หน้า Home แตกต่าง (เช่น Admin มี Users/Reports; Evaluatee ไม่มี) | 1 |  | สกรีนช็อตหลัง Login ของแต่ละบทบาท (3 รูป) |
| -Router Guard กันเข้าหน้าเกินสิทธิ์  Login เป็น evaluator แล้วเข้าลิงก์ลึก /admin/users → ต้อง Redirect หรือแสดง 403 page | 1 |  | สกรีนช็อตผล 403/Redirect |
| - Deep‑link + Refresh คงสิทธิ์ถูกต้อง  เปิด /evaluator/assignments แล้ว Refresh → ยังอยู่/Redirect ตามสิทธิ์ถูกต้อง (ไม่หลุด session); ทดสอบ Logout แล้วเข้าลิงก์เดิม → กลับหน้า Login | 1 |  | สกรีนช็อตผล (ก่อน/หลัง Refresh และหลัง Logout) |
| - Required/Pattern/MaxLength  เว้นค่าว่าง/ผิดรูปแบบ/เกินความยาว → แสดง error ที่ field และปุ่ม Submit disabled | 1 |  | สกรีนช็อต error states + ปุ่มถูก disable |
| - Range/Enum (เช่น score 1–4, yes/no)  กรอกค่าผิดช่วงหรือ enum ผิด → บล็อก; กรอกถูก → บันทึกได้ | 1 |  | ก่อน/หลัง แสดง error → success toast |
| - Server‑side Error Handling (เช่น 409)  กดบันทึกให้เกิด 409 (Duplicate Assignment) → UI แสดงข้อความอ่านง่าย ไม่ค้าง | 1 |  | สกรีนช็อต+ข้อความผิดพลาด |
| - UX States (loading/disabled/success)  ขณะ submit มี loading และปุ่ม disabled; สำเร็จแล้ว clear/redirect + success toast | 1 |  | สกรีนช็อต ขั้นตอน submit เต็ม flow |
| - Filter + Sort ทำงานถูกต้อง  ใช้กล่องค้นหา/ตัวกรอง แล้วสลับ sort asc/desc บนคอลัมน์เดียวกัน → ผลเปลี่ยนตามคาดและลำดับกลับจริง | 1 |  | สกรีนช็อตก่อน–หลัง พร้อมระบุพารามิเตอร์ที่ใช้ |
| - Pagination + เมตาดาตา  เปลี่ยนหน้า (1 → 2 → หน้าท้าย) ตรวจจำนวนแถว/ตัวบ่งชี้หน้าปัจจุบัน และค่ารวม total ตรงกับ API | 1 |  | สกรีนช็อตอย่างน้อย 3 หน้า + Response JSON meta |
| -จำกัดขนาดไฟล์ (เช่น >10MB → ถูกบล็อก)  อัปโหลดไฟล์ใหญ่เกินกำหนด → UI แสดงข้อความแจ้งและไม่ส่งต่อไปยัง API | 1 |  | สกรีนช็อตข้อความแจ้ง + ตัวอย่างไฟล์ทดสอบ |
| -ชนิดไฟล์ต้องห้าม (เช่น .exe) ถูกบล็อก  อัปโหลดไฟล์ชนิดต้องห้าม → UI แสดง error และไม่ส่งต่อ; ชนิดที่ยอมรับ → อัปโหลดสำเร็จ | 1 |  | สกรีนช็อต error และ success ในเคสชนิดไฟล์ต่างกัน |
| Status/Badges/Filter —สถานะที่เกี่ยวข้อง: draft / submitted / locked - Badge แสดงสถานะตรงกับ API  เปิดรายการ Results/Assignments → badge สี/ข้อความต้องตรงกับค่าสถานะจาก API | 1 |  | สกรีนช็อต list ชี้ให้เห็น mapping สถานะ |
| - Filter ตามสถานะ + จำนวนตรง  เลือก filter สถานะ (เช่น submitted) → จำนวนรายการบนหน้าจอตรงกับที่ API ส่ง และคงค่าตัวกรองเมื่อกลับหน้า/รีเฟรช | 1 |  | สกรีนช็อต แสดง flow และข้อความเตือน |
| - Evaluatee Flow  Login → อัปโหลด evidence ตามตัวชี้วัด → เห็นรายการไฟล์แนบในหน้าตนเอง; ถ้าไม่มีไฟล์ ระบบไม่ให้ submit (UI แจ้งเตือน) | 1 |  | สกรีนช็อต แสดง flow และข้อความเตือน |
| -Evaluator Flow  Login → เปิด assignments ของตน → กรอกผล/บันทึก → เห็นสถานะ/คะแนนอัปเดต และในรายงาน/ตารางสะท้อนผลทันที | 1 |  | สกรีนช็อต การทำงาน + สกรีนช็อต list/รายงานหลังบันทึก |
| 2.3 เครื่องมือโครงสร้างพื้นฐานและปฏิบัติการ (DevOps) (5 คะแนน) | 5 |  |  |
| - **Dockerized Stack** (Compose: API+DB(+phpMyAdmin), ไฟล์ .env) | 2 |  | docker compose up -d รันผ่าน; แนบ docker-compose.yml, .env |
| - **Health/Logs/Restart** — endpoint สุขภาพ/เช็กสถานะคอนเทนเนอร์/แนวทางเก็บล็อก | 1 |  | คำสั่งตรวจ curl /health, docker logs + หมายเหตุ restart policy |
| - Reverse Proxy/Scale (เบื้องต้น) + mini-load test | 1 |  | Config Nginx (ถ้ามี) หรือ scale API=2 + แนบผลทดสอบสั้น ๆ |
| **-Runbook & Backup** — README วิธีรัน/ทดสอบ + สคริปต์ dump/restore ฐานข้อมูล | 1 |  | แนบไฟล์ README + ตัวอย่าง mysqldump |
| **3. ด้านพฤติกรรมลักษณะนิสัย** | **5** |  |  |
| 3.1 ความละเอียดรอบคอบ | 2 |  | ตรวจสอบงานอย่างละเอียดก่อนส่ง ทำให้งานมีความผิดพลาดน้อยที่สุด |
| 3.2 การทำงานเป็นระบบ | 2 |  | ปฏิบัติงานตามขั้นตอนที่กำหนดอย่างเป็นระเบียบ มีการวางแผนและจัดลำดับความสำคัญของงาน |
| 3.3 การส่งงานตรงเวลาส่งงานครบถ้วน | 1 |  | ส่งงานครบถ้วนภายในเวลาที่กำหนด หรือเร็วกว่าที่กำหนด |
| **รวมคะแนน** | **60** |  |  |