บทที่ 4
ผลลัพธ์การออกแบบและการพัฒนาระบบ

จากการศึกษาทฤษฎีและหลักการที่เกี่ยวข้องชทำให้สามารถพัฒนาระบบเว็บไซต์คลังผลงานโครงงาน SciDigiKnowledge ได้อย่างเป็นรูปธรรม โดยระบบที่พัฒนาขึ้นมีฟังก์ชันการทำงานตรงตามความต้องการของผู้ใช้ในระดับหนึ่ง ดังนั้นในบทนี้จึงนำเสนอผลการดำเนินงานของระบบตามลำดับขั้นตอนการพัฒนา

4.1 บทนำ

บทนี้นำเสนอผลลัพธ์ที่ได้จากการออกแบบและพัฒนาระบบเว็บไซต์คลังผลงานโครงงาน SciDigiKnowledge โดยสรุปการดำเนินงานตั้งแต่ขั้นตอนการออกแบบตาม SDLC จนถึงการพัฒนา (Frontend: React, Backend: Express, Database: MySQL) รวมถึงการเตรียมการทดสอบและเกณฑ์การประเมินต่าง ๆ

4.2 ผลลัพธ์ของการออกแบบระบบ

4.2.1 Context Diagram (แผนภาพบริบท)

แสดงความสัมพันธ์ระหว่างระบบ SciDigiKnowledge กับปัจจัยภายนอก (External Entities) โดยมีผู้มีส่วนได้ส่วนเสียหลัก ดังนี้:

- นักศึกษา (User): อัปโหลดผลงาน ขอสิทธิ์ แก้ไข ดูผลการอนุมัติ และดาวน์โหลดไฟล์
- อาจารย์/ผู้ตรวจ (Teacher): ตรวจสอบ ดาวน์โหลด และให้คำปรึกษา
- ผู้ดูแลระบบ/แอดมิน (Admin): จัดการบัญชีผู้ใช้ สำรองข้อมูล และดูสถิติระบบ ข้อมูลเข้า-ออกที่สำคัญ เช่น ข้อมูลบัญชีผู้ใช้ ข้อมูลเมทาดาทาของผลงาน ไฟล์ผลงาน คำขออนุมัติ และผลสรุปสถิติ

4.2.2 Data Flow Diagram (DFD)

Level 0 : ระบบ SciDigiKnowledge เป็นกล่องเดียวที่เชื่อมต่อกับผู้ใช้งาน (นักศึกษา, อาจารย์, แอดมิน) โดยมีการไหลของข้อมูลหลักคือ "ข้อมูลผลงาน" และ "ข้อมูลผู้ใช้/คำสั่งควบคุม"

Level 1 (Process ย่อย): แยกออกเป็นกระบวนการหลักตามที่ออกแบบไว้ ดังนี้:

1. จัดการบัญชีผู้ใช้ (User Management)
2. การสำรองและดูสถิติ (Backup & Report)
3. อัปโหลดและจัดการผลงาน (Upload & Manage Project)
4. การค้นหาและดาวน์โหลด (Search & Download)

4.2.3 ER Diagram และสรุป Data Dictionary

จากการออกแบบ ER-Diagram แยกเป็นเอนทิตีหลักและความสัมพันธ์ระหว่างเอนทิตี โดยสรุปตารางหลักและคีย์สำคัญดังนี้:

ตาราง users (ข้อมูลสมาชิก)

Attribute	Type	Description	Key
id	INT(11)	รหัสผู้ใช้	PK
username	VARCHAR(100)	ชื่อบัญชีผู้ใช้	
password	VARCHAR(255)	รหัสผ่าน (เก็บแบบ hash)	
role	ENUM('student','teacher','admin')	บทบาทของผู้ใช้	
email	VARCHAR(255)	อีเมลผู้ใช้	
created_at	TIMESTAMP	วันที่สร้างบัญชี	
updated_at	TIMESTAMP	วันที่แก้ไขล่าสุด	

ตาราง documents (ผลงานหลัก)

Attribute	Type	Description	Key
id	INT(11)	รหัสผลงาน	PK
user_id	INT(11)	ผู้ใช้ที่อัปโหลด	FK

		ผลงาน (เชื่อม users.id)	
title	VARCHAR(255)	ชื่อผลงาน	
keywords	TEXT	คำค้น	
academic_year	VARCHAR(20)	ปีการศึกษา	
status	ENUM('draft','pending','approved','rejected')	สถานะผลงาน	
uploaded_at	TIMESTAMP	วันที่อัปโหลด	
download_count	INT(11)	จำนวนครั้งที่ถูกดาวน์โหลด	

ตาราง document_files (ไฟล์ย่อยของแต่ละผลงาน)

Attribute	Type	Description	Key
id	INT(11)	รหัสไฟล์	PK
document_id	INT(11)	ผลงานหลักที่เชื่อม (documents.id)	FK
file_path	VARCHAR(500)	ตำแหน่งไฟล์บน Storage	
file_type	VARCHAR(50)	นามสกุล/ประเภทไฟล์	
uploaded_at	TIMESTAMP	วันที่อัปโหลด	

ตาราง categories (หมวดหมู่เอกสาร)

Attribute	Type	Description	Key
id	INT(11)	รหัสหมวดหมู่	PK
name	VARCHAR(100)	ชื่อหมวดหมู่	

ตาราง document_categories (เชื่อมผลงานกับหมวดหมู่)

Attribute	Type	Description	Key
document_id	INT(11)	รหัสผลงาน	PK, FK
category_id	INT(11)	รหัสหมวดหมู่	PK, FK

ตาราง downloads (ประวัติการดาวน์โหลด)

Attribute	Type	Description	Key
id	INT(11)	รหัสการดาวน์โหลด	PK
user_id	INT(11)	ผู้ใช้ที่ดาวน์โหลด	FK
document_id	INT(11)	ผลงานที่ถูกดาวน์โหลด	FK
downloaded_at	TIMESTAMP	วันที่ดาวน์โหลด	

ความสัมพันธ์ระหว่างตาราง:

- users (1) ➜ documents (N)
- documents (1) ➜ document_files (N)
- documents (N) ↔ categories (N) (ผ่าน document_categories)
- users (1) ➜ downloads (N), documents (1) ➜ downloads (N)

4.3 ผลลัพธ์ของการพัฒนาระบบ

4.3.1 ส่วนของฐานข้อมูล (MySQL)

ฐานข้อมูลออกแบบเป็น MySQL ตาม Data Dictionary ข้างต้น โดยมีความสัมพันธ์แบบ 1-N และ N-M ตามที่สรุปไว้ ตัวอย่าง DDL (ตัวอย่างให้เพื่อใช้อ้างอิงและทดสอบในสภาพแวดล้อมการพัฒนา)

ตัวอย่าง SQL DDL (ย่อ)

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  role ENUM('student','teacher','admin') NOT NULL DEFAULT 'student',
  email VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
CREATE TABLE documents (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  title VARCHAR(255) NOT NULL,
  keywords TEXT,
  academic_year VARCHAR(20),
  status ENUM('draft','pending','approved','rejected') DEFAULT 'draft',
  uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  download_count INT DEFAULT 0,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
CREATE TABLE document_files (
```

```
  id INT AUTO_INCREMENT PRIMARY KEY,
  document_id INT NOT NULL,
  file_path VARCHAR(500),
  file_type VARCHAR(50),
  uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (document_id) REFERENCES documents(id)
);
CREATE TABLE categories (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
);
CREATE TABLE document_categories (
  document_id INT NOT NULL,
  category_id INT NOT NULL,
  PRIMARY KEY (document_id, category_id),
  FOREIGN KEY (document_id) REFERENCES documents(id),
  FOREIGN KEY (category_id) REFERENCES categories(id)
);
CREATE TABLE downloads (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  document_id INT NOT NULL,
  downloaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (document_id) REFERENCES documents(id)
);
```

4.3.2 ส่วนของ Backend (Express)

โครงสร้างตัวอย่างของโฟลเดอร์ Backend (Express) ที่แนะนำ

backend/

```
├── src/
│   ├── controllers/
│   ├── routes/
│   ├── models/ (หรือใช้ ORM เช่น Sequelize)
│   ├── middlewares/
│   ├── services/
│   └── app.js (หรือ server.js)
├── uploads/ (เก็บไฟล์ชั่วคราว/หรือเชื่อมกับ Cloud Storage)
└── package.json
```

ตัวอย่าง endpoint ที่สำคัญ:

- POST /api/auth/login (เข้าสู่ระบบ)
- POST /api/users (สร้างผู้ใช้)
- GET /api/documents (รายการผลงาน)
- POST /api/documents (อัปโหลดผลงาน)
- GET /api/documents/:id (รายละเอียดผลงาน)
- POST /api/downloads/:id (บันทึกการดาวน์โหลด)

การอัปโหลดไฟล์สามารถใช้ multer เป็น middleware และบันทึกตำแหน่งไฟล์ในตาราง document_files หรือเก็บใน Cloud Storage (เช่น AWS S3) แล้วเก็บ URL ในฐานข้อมูล.

ตัวอย่างโค้ด (ตัวอย่างย่อของ route อัปโหลด):

```
const express = require('express');
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });
const router = express.Router();
router.post('/documents', upload.single('file'), async (req, res) => {
  // ตรวจสอบข้อมูล, บันทึก metadata ลง table documents และ path ลง document_files
  res.json({ success: true });
```

```
});
module.exports = router;
```

4.3.3 ส่วนของ Frontend (React)

โครงสร้างตัวอย่างของโฟลเดอร์ Frontend (React):

```
frontend/
├── src/
│   ├── components/ (เช่น UploadForm, DocumentCard, Header, Footer)
│   ├── pages/ (เช่น Home, Upload, Documents, Admin)
│   ├── services/ (api.js — เก็บฟังก์ชันเรียก API)
│   ├── hooks/ (ใช้ custom hooks)
│   └── App.js
```

หน้าที่สำคัญในระบบ (UI):

- หน้าเข้าสู่ระบบ (Login)
- หน้าแดชบอร์ด/หน้าหลัก (แสดงผลงานแนะนำ)
- หน้าอัปโหลดผลงาน (ฟอร์มแยกหัวข้อ)
- หน้าแสดงเอกสารทั้งหมด (ค้นหา/กรอง/ดาวน์โหลด)
- หน้าโปรไฟล์ผู้ใช้/จัดการบัญชี

ตัวอย่างการเรียก API (service ด้วย axios หรือ fetch):

```
import axios from 'axios';
export function uploadDocument(formData) {
  return axios.post('/api/documents', formData, {
    headers: { 'Content-Type': 'multipart/form-data' }});
```

}

4.3.4 ส่วนติดต่อผู้ใช้ (UI Screenshots / Mockups)

เอกสารฉบับนี้ใช้การออกแบบ UI ตามแนวทางที่เน้นความเรียบง่าย การจัดวางเมนูที่ชัดเจน และการใช้งานง่าย (ตัวอย่างตำแหน่งเมนู: โลโก้, เมนูนำทาง, ผลงานแนะนำ, การ์ดผลงาน ฯลฯ).

หากต้องการ ผมสามารถช่วยแปลงภาพต้นแบบ (wireframe) เป็นไฟล์ภาพหรือเพิ่มหน้าจอจริงจากโค้ดเมื่อคุณมีไฟล์ภาพหรือสกรีนช็อตของระบบ.

4.4 ผลการทดสอบระบบ

ในที่นี้จัดทำแบบฟอร์มและตัวอย่าง Test Case เพื่อใช้เป็นแนวทางทดสอบ (Unit, Integration, System, UAT) เมื่อเริ่มการทดสอบจริง:

4.4.1 Test Case ตัวอย่าง

ID	Test Case	Steps	Expected Result	Status/Result
TC001	Login (สำเร็จ)	กรอก username/password ที่ถูกต้อง แล้วกด Login	เข้าสู่ระบบและไปหน้า Dashboard	ยังไม่ได้ทดสอบ
TC002	Login (ล้มเหลว)	กรอกข้อมูลผิด/ว่าง แล้วกด Login	แสดงข้อความข้อผิดพลาด	ยังไม่ได้ทดสอบ
TC003	อัปโหลดผลงาน	เลือกไฟล์และกรอกข้อมูล แล้วกดอัปโหลด	บันทึก metadata ลง DB และไฟล์ถูกเก็บ	ยังไม่ได้ทดสอบ
TC004	ค้นหา/กรอง	พิมพ์คำค้นและกดค้นหา	แสดงรายการผลงานที่ตรงตามเงื่อนไข	ยังไม่ได้ทดสอบ
TC005	ดาวน์โหลด	กดปุ่มดาวน์โหลดบนผลงาน	ไฟล์ถูกดาวน์โหลดและเพิ่ม download_count	ยังไม่ได้ทดสอบ

4.4.2 ความพึงพอใจของผู้ใช้ (Template แบบสำรวจ)

สามารถใช้แบบสอบถามสั้นเพื่อวัดความพึงพอใจหลังการใช้งาน เช่น ความพึงพอใจโดยรวม (1–5), ความง่ายในการใช้งาน, ความเร็วในการค้นหา, ความครบถ้วนของข้อมูล และช่องว่างให้เสนอแนะ

4.5 สรุปผลการดำเนินโครงงาน

จากการออกแบบและเตรียมพัฒนาระบบ SciDigiKnowledge ตามที่สรุปในบทนี้ ระบบออกแบบให้รองรับการจัดเก็บผลงานในรูปแบบดิจิทัล ใช้ React เป็นส่วนติดต่อผู้ใช้, Express เป็น Backend และ MySQL เป็นฐานข้อมูล ซึ่งสอดคล้องกับเป้าหมายในการลดความเสี่ยงการสูญหายของข้อมูลและเพิ่มความสะดวกในการสืบค้นและเผยแพร่ผลงาน