1.Architecture Overview (MVC)

ระบบนี้ใช้ MVC (Model–View–Controller) Architecture
เพื่อแยกหน้าที่ของระบบออกเป็นส่วน ๆ อย่างชัดเจน ทำให้ระบบดูแลรักษาง่าย และรองรับผู้ใช้งานหลายบทบาท

2. MVC Layers

2.1 Model

หน้าที่

- จัดการข้อมูลและ Business Logic
- ติดต่อฐานข้อมูล

ตัวอย่าง

- User
- Role
- Complaint
- Notification
- Category
- Location

2.2 View

หน้าที่

- แสดงผลให้ผู้ใช้
- รับข้อมูลจากผู้ใช้

ตัวอย่าง

- Login Page
- Student Dashboard
- Samo Dashboard
- Admin Panel
- Complaint Form

2.3 Controller

หน้าที่

- รับ Request จาก View
- เรียกใช้งาน Model
- ส่งผลลัพธ์กลับไปที่ View

ตัวอย่าง

- AuthController
- UserController
- ComplaintController
- NotificationController
- AdminController

3. Components & Relationships

User

↓↑

View

↓↑

Controller

↓↑

Model

↓↑

Database

4. Technology Stack Selection (MVC-based)

View

- HTML / CSS / JavaScript

Controller

- Node.js (Express)

Model

- ORM
- MySQL หรือ PostgreSQL

5. Design Patterns

5.1 MVC Pattern

เหตุผล

- โครงสร้างชัดเจน
- แยกหน้าที่ชัด
- เหมาะกับ Web Application

5.2 Repository Pattern

ใช้กับ

- Model Layer

เหตุผล

- แยกการเข้าฐานข้อมูลออกจาก Controller
- โค้ดเป็นระเบียบและดูแลง่าย

Controller → Service → Repository → Database

5.3 Observer Pattern

ใช้กับ

- ระบบแจ้งเตือน

เหตุผล

- เมื่อสถานะคำร้องเปลี่ยน ต้องแจ้งหลายฝ่าย
- ลดการเขียนโค้ดผูกกันโดยตรง

5.4 Strategy Pattern

ใช้กับ

- การกำหนดสิทธิ์ตามบทบาท (Role)

เหตุผล

- Student, Samo, Officer, Admin มีสิทธิ์ต่างกัน
- เพิ่มบทบาทใหม่ได้ง่าย