**PROJECT CÔNG NGHỆ MỚI**

1. **Thiết kế Database Diagram:**

**- Bảng Users** gồm các ràng buộc sau:

|  |  |
| --- | --- |
| User\_ID | PK, Varchar, Auto Increment |
| Full\_Name | Varchar |
| Username | Varchar, Unique |
| Password | Varchar, (8 ký tự trở lên, có 1 hoặc 2 ký hiệu đặc biệt, không được giống username, không toàn là số) |
| Email | Varchar, unique |
| Role | Enum(Admin, User, Maint) |
| SDT | Varchar, unique |

**Lưu ý:**

1. Auto Increment: Tự động update ID theo lượt đăng ký và tự cho mỗi một ID sẽ độc nhất với nhau.

VD: Người đầu tiên đăng ký vào hệ thống sẽ có ID: U001 thì các người sau sẽ là U002,…Un+1.

1. Username phải độc lập, không trùng với nhau.
2. Password phải ràng buộc kĩ như: 8 ký tự trở lên, có 1 hoặc 2 ký hiệu đặc biệt, không được giống username, không toàn là số.

- **Bảng BUILDING:**

Building\_ID: Varchar

- **Bảng Location:**

Location\_id: Varchar

building\_id -> Foreign Key với Building.building\_id

Thêm Descripton để mô tả vị trí

**- Bảng Equipment:**

Category thì nên làm type để phân loại thiết bị loại nào

Equipment\_Id: varchar

Location\_Id: ForeignKey -> Location.location\_Id

Thêm Code để định danh mã thiết bị riêng, mang tính kinh doanh và dùng cho hiển thị, tìm kiếm hoặc báo cáo thống kê.

### - ****Bảng EquipmentLog (Lịch sử thiết bị)****

**Mục đích:** Lưu trữ lịch sử sử dụng, sự cố, cập nhật của từng thiết bị.

**Thuộc tính:**

* + log\_id (PK, varchar)
  + equipment\_id (FK -> Equipment.equipment\_id)
  + user\_id (FK -> Users.user\_id) – Người thực hiện hành động (có thể là người dùng thông thường báo sự cố, hoặc người bảo trì cập nhật thông tin)
  + action (VARCHAR) – Loại hành động (Thêm, Sửa, Xóa, Báo cáo sự cố,…)
  + timestamp (DATETIME)
  + details (Varchar)

**- Bảng Maintenance:**

**Mục đích:** Theo dõi các lần bảo trì, sửa chữa trang thiết bị.

**Thuộc tính:**

· maintenance\_id (PK, Varchar, auto increment)

· equipment\_id (FK -> Equipment.equipment\_id)

· maintenance\_date (DATE)

· performed\_by (FK -> Users.user\_id) – Người bảo trì (ràng buộc chỉ thuộc nhóm MAINT (người bảo trì))

· description (varchar) – Mô tả công việc bảo trì/sửa chữa

· cost (DECIMAL)

· status (ENUM: 'Scheduled', 'Completed')

**- Bảng Feedback:**

### Mục đích:

* Lưu trữ phản hồi (feedback) từ người dùng về trang thiết bị hoặc hệ thống.
* Cho phép người dùng (hoặc nhân viên bảo trì) gửi nhận xét, đánh giá chất lượng trang thiết bị, báo cáo sự cố hoặc góp ý.

### Các thuộc tính cần sửa :

* **feedback\_id**: Khóa chính (Primary Key), kiểu tự động tăng (varchar, auto-increment).
* **user\_id**: Khóa ngoại (Foreign Key) liên kết tới bảng **Users** (để biết ai đã gửi feedback).
* **equipment\_id**: Khóa ngoại liên kết tới bảng **Equipment** (nếu feedback liên quan đến một thiết bị cụ thể; có thể để NULL nếu feedback chung về hệ thống).
* **content**: Nội dung phản hồi (varchar).
* **rating**: Đánh giá (INTEGER, tùy chọn, ví dụ từ 1 đến 5).
* **created\_at**: Thời gian gửi phản hồi (TIMESTAMP).

### Mối quan hệ:

* Feedback liên kết với bảng **Users**: Mỗi feedback được gửi bởi một người dùng.
* Feedback có thể liên kết với bảng **Equipment**: Nếu feedback được gửi về một thiết bị cụ thể.

**- Bảng Notifications:**

### Mục đích:

* Lưu trữ các thông báo (notification) được gửi đến người dùng.
* Thông báo có thể liên quan đến các sự kiện như: thiết bị cần bảo trì, thông báo sự cố, hoặc thông báo hệ thống từ Quản trị.

### Các thuộc tính đề xuất:

* **notification\_id**: Khóa chính (Primary Key), kiểu tự động tăng (varchar, auto-increment).
* **user\_id**: Khóa ngoại liên kết tới bảng **Users** – người nhận thông báo.
* **message**: Nội dung thông báo (varchar).
* **status**: Trạng thái thông báo (ví dụ: 'unread', 'read'; dùng kiểu ENUM).
* **created\_at**: Thời gian thông báo được tạo (TIMESTAMP).

### Mối quan hệ:

* Notifications liên kết với bảng **Users**: Mỗi thông báo được gửi đến một người dùng cụ thể.
* (Tùy chọn) Nếu hệ thống cần thông báo liên quan đến bảo trì hoặc thiết bị, có thể thêm các khóa ngoại đến bảng **Maintenance** hoặc **Equipment**.

Bảng Notification chỉ lưu trữ thông tin thông báo; tự nó không "gửi" thông báo tự động. Có thể tích hợp chức năng tự động gửi thông báo bằng cách kết hợp bảng Notification với các công cụ và kỹ thuật như:

**Django Signals:** Khi có sự kiện (ví dụ, thiết bị cần bảo trì), có thể dùng signal (như post\_save) để tạo một bản ghi thông báo trong bảng Notification và đồng thời gọi một hàm gửi email hoặc SMS.