**BG Safety Training project**

**Flow in Your Project**

* **Admin logs in** → Can add/delete employees, create training types, assign trainings.
* **User logs in** → Can only update status or view stats (no add/delete employee).
* **Unique training rule** is enforced by the UNIQUE(employee\_id, training\_id) constraint.
* **Filter by training type** is a simple WHERE training\_id = ? query.

### ****Development Plan for You****

**Week 1 (Now - Aug 17)**  
✅ Create DB in MySQL Workbench (tables above)  
✅ Connect Java backend to MySQL (JDBC)

**Week 2 (Aug 18 - Aug 24)**  
✅ Build backend APIs (Servlets) for CRUD operations  
✅ Implement login with role-based access

**Week 3 (Aug 25 - Aug 31)**  
✅ Build HTML/CSS/JS pages for dashboard & training assignment  
✅ Connect frontend to backend APIs with Fetch/AJAX

**Week 4 (Sep 1 - Sep 10)**  
✅ Test & fix bugs  
✅ Export DB (.sql file)  
✅ Prepare presentation for company

### ****How We’ll Structure Your Database****

Since you’re making your own DB, we’ll store:

1. **Employee Details** (with unique employee ID)
2. **Training Types** (list of available trainings)
3. **Employee Training Assignments** (linking employees to trainings with status)
4. **Users** (for login & role-based permissions — Admin/User)

## ****1️⃣ Roles and Permissions****

* **Admin**
  + Can **Add/Delete** employees in the system
  + Can assign trainings to any employee
* **Other Users**
  + Can only **view** training status
  + Can update training progress/status (maybe, if allowed)
  + Cannot add/delete employees

## ****5️⃣ Tech Stack****

* **Frontend** → HTML, CSS, JS (Vanilla)
* **Backend** → Java Servlet (Eclipse + Tomcat)
* **Database** → MySQL (Workbench)
* **Testing** → Postmam
* **Stats**:
  + Number of employees **Trained**
  + Number of employees **In Training**
  + Number of employees **Not Trained**
* **Filter** by **Training Type** (e.g., Safety, Technical, Soft Skills, etc.)
* **Rule**:
  + An employee **cannot** repeat the **same training**.
  + But can attend **different trainings**.

**SQL Design:**

-- 1. Employees Table

CREATE TABLE employees (

id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_code VARCHAR(50) UNIQUE NOT NULL, -- unique code like EMP123

name VARCHAR(100) NOT NULL,

department VARCHAR(100) NOT NULL

);

-- 2. Training Types

CREATE TABLE training\_types (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) UNIQUE NOT NULL

);

-- 3. Employee Trainings

CREATE TABLE employee\_training (

id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT NOT NULL,

training\_id INT NOT NULL,

status ENUM('Not Trained', 'In Training', 'Trained') DEFAULT 'Not Trained',

UNIQUE (employee\_id, training\_id), -- no duplicate same training for same employee

FOREIGN KEY (employee\_id) REFERENCES employees(id) ON DELETE CASCADE,

FOREIGN KEY (training\_id) REFERENCES training\_types(id) ON DELETE CASCADE

);

-- 4. Users Table

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL, -- store hashed password

role ENUM('Admin', 'User') DEFAULT 'User'

);