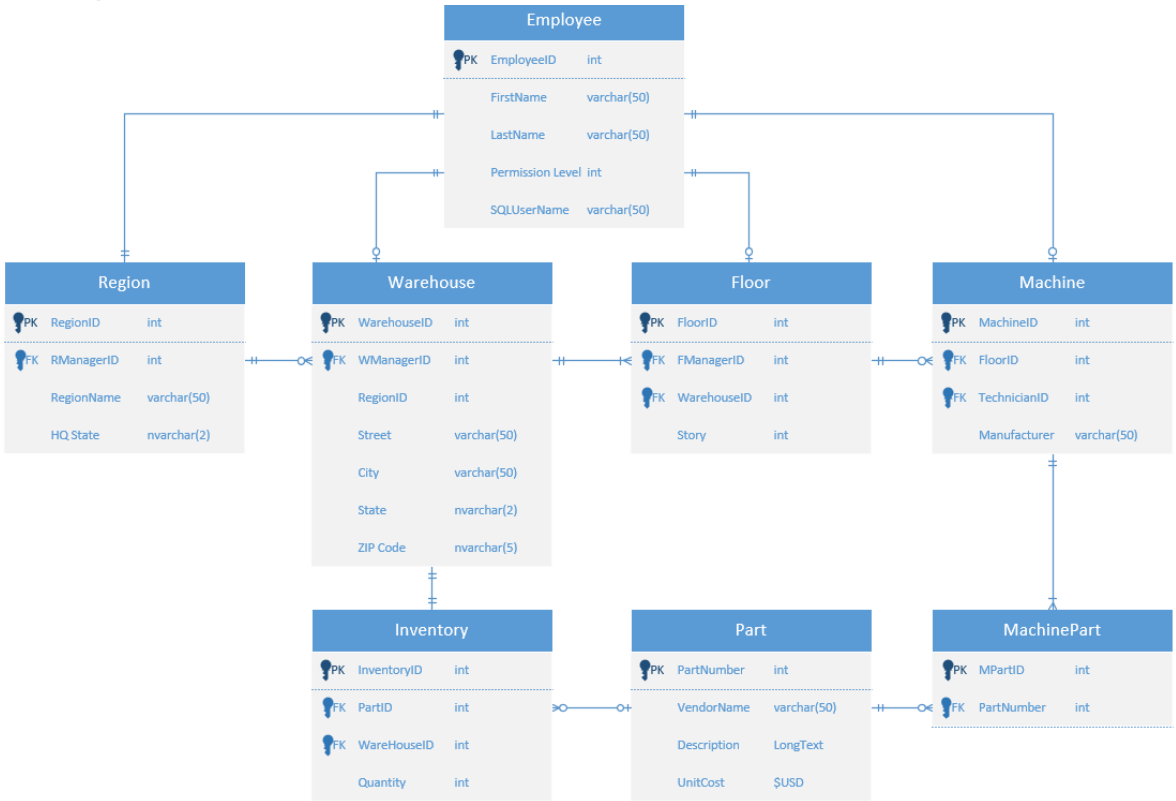
Team 2: Skylar Love J00708381, Princeton Epeagba J00701287, Andrew Mills J00630787,
Raven Hall J00702102, Sarah Pyle J00709425
CSC – 440 Fall 2024

Problem Description: Our team will develop a Factory Management System that records organizational data, warehouse, floor, machine, part, and employee data. This will use an MS SQL Server Database with a C# and .NET User Interface for employees at 4 different security levels to access and create records.

Research goals and expected outcome: to create a database that stores critical and sensitive information regarding an industrial factory setting that can impact managerial and corporate level decisions and finances. To keep this database secure from its users and potential attackers by at least implementing principles such as separation of privilege, least privilege, complete mediation, fail-safe defaults, and economy of mechanism. To prevent attacks and mistakes such as SQL Injection, Unauthorized or Abuse of Access, Denial of Service, Security Misconfiguration, and Human Error.

Project Plan:

- **Implement an MS SQL database** such as:



  i. **Separate tables at each managerial level** so that managers for one floor cannot access the machine tables for another floor and so on. – prevents human error, misuse, and abuse. i.e. Floor 1 Manager cannot access the machines for Floor 2 and vice versa. This applies in the context of warehouses, floors, machines, and inventories.

- **Implement secure login using credentials and automatic timeout**
  i. Prevent SQL Injection

- **Login Sequence/Algorithm**

The user inputs username and password into the GUI, there must be a matching username in the Employee Table, if it matches* a username in the table, the interface's backend then hashes the user-input password, queries for the SQL server's hashed password from the matching Employee table entry, then the C# interface's backend compares the hashed passwords**.

*If the username does not match, it responds with a message saying "No user with username 'userInput' found. Please try again."
**If the hashed passwords do not match, the user receives an error message along the lines of "Incorrect Password."
*** if it succeeds, the screen in the GUI changes to the next screen, AKA, the table selection screen (subject to change?). This process should use a method that prevents SQL injection. Throughout this process, we should also Audit both successful and unsuccessful username and password checks with datetimes. What the interface Developer(s) need to know/learn:

i. How to hash passwords, using strong algorithms such as: bcrypt, PBKDF2, or Argon2 AND NOT MD5 or SHA-1
ii. How to make a SQL connection using connection Strings
iii. One of many SQL Injection prevention methods such as parameterized queries which uses prepared statements to separate the actual data (in this case user input) from the actual SQL query code. Thus, if the user DOES input an '=', it's not relevant to the query and is JUST data.
iv. SALT-ing passwords. Stored Hash passwords consist of the user's password concatenated with a random data string (known as a SALT) that is then hashed.

- **Implement User Privilege Levels** users must have their privilege level (0,1,2,3,4,5) checked against the privilege requirement of the table.

- **Implement specific table edit/access permissions** that are required to access or edit the table. The user must hold the appropriate permission for each action for each table.

- **Implement "Are you sure" checks** such as "Are you sure you want to permanently alter this record? Yes/No"

- **Implement Auditing/Logging** such as when a table is accessed or edited, record that access or edit, storing values such as TableName, EmployeeID, Permission granted or denied/reason for denial, etc.

- **Implement input validation methods** i.e. acceptable range of values or set of entities

- **Implement password encryption method(s)**
  i. StackOverflow the-proper-way-of-implementing-user-login-system
  ii. ASP.NET Login Controls Overview

iii. SQL Server Central how-to-implement-a-secure-user-login-concept-with-a-winform-application-and-a-sql-server

Milestones (subject to change):

Complete by:

- Sept 20th: Finalize and Publish Diagrams/ Design Plans
- Sept 27th: Create Database Tables: Region, Employee Table, Warehouse table, Floor Table, Machine Table
- Oct 4th: Create: Inventory, Vendor, Part, InventoryPart, MachinePart
- Oct 9th: Add integrity constraints
- Oct 18th: Create a Secure Login System → 50% milestone
- Oct 23rd: Project DRAFT Technical Write-Up
- Oct 30th: Create Table Selection Screen
- Nov 6th: Create Query Screen
    - When a query is made, audit the query (Read operation)
- Nov 15th: Create a screen for adding/ modifying entries.
    - When a modification is made, audit the modification as well as if it is a New Record or Existing Record. (Create, Update, Delete)
- Nov 22nd: Complete remaining work, Create PPT/Slides
- Nov 22nd: Consider starting report.
- Dec 2nd: Present slides
- Dec 4th: Finalize and Submit Project Report