CIS*3530 Data Base Systems and Concepts
Assignment 4

Dr. Charlie Obimbo **Due: Monday, November 25th 11:59 pm**

Each Mark in Questions 1 – 2 is worth 6 marks of the assignment

Question 1 (2 marks)

Mark **True** or **False** for each of the following.

1. $\{X \to Y, YW \to Z\} \Rightarrow \{X \to Z\}$ _____
2. $\{W \to Y, X \to Z\} \Rightarrow \{WX \to Y\}$ _____
3. $\{X \to Y, X \to W, WY \to Z\} \Rightarrow \{X \to Z\}$ _____
4. $\{X \to Y, Y \to W, W \to Z\} \Rightarrow \{X \to Z\}$ _____

Question 2

1. (a) Describe MAC, DAC and Role-Based Access Control. (2 Marks)
   (b) In what way is Role-Based Access Control superior to MAC and DAC. (2 Marks)

Q3: Company Mgt System with Role-Based Access Control [12]

This project will build on the existing system by introducing user accounts, roles, and permissions. The system will allow an **Admin** to manage users and roles, while **department-specific users** can access data relevant only to their departments. This structure introduces new tables, views, and functions, making it a practical learning experience.

Note

Partial work will receive partial marks. However, each partial implementation must be able to complete a task fully. For example, if only authentication is implemented, then the user table, login, and logout functionalities should work and perform the task independently.

The project is worth 12 out of 15 marks, with an option to earn up to 4 more marks as a bonus. Details on how to earn bonus marks are provided at the end.

Requirements:

- **Documentation**: Include a README file with setup instructions and an overview of implemented features. List all features implemented in the README.

Note

- **Group Member Contributions**: Completing all the required tasks perfectly and ensuring that all group members can demonstrate and modify the project during the evaluation session—such as adding a module with different levels of access or creating a new HTML view with a special join—are necessary to award full marks to the entire group.
- **Programming**: You can use either Python or PHP. Pure HTML is fine, and No marks for CSS/style.

Project Objectives

1. **Account Management**: Allow admins to manage user accounts and assign roles.
2. **Role-Based Access**: Define user roles with different permissions and access scopes:
   - **Super Admin**: Can manage all users, roles, departments, and employees.
   - **Department Admin**: Can manage employees only within their assigned department.
   - **Normal User**: Can only view data for their department.
3. **Department-Specific Data Access**: Users should only access data related to their department, even when joining tables.
4. **Views and Data Filtering**: Use SQL views or queries to restrict access based on the user's department.

Project Requirements

1. **User and Role Management**
   - Add an **Account Manager** module that allows the Super Admin to:
     - Create new users.
     - Assign roles to users (Super Admin, Department Admin, Normal User).
     - Remove users.
   - Add roles with permissions:
     - **Super Admin**: Can create, read, update, and delete (CRUD) all data.
     - **Department Admin**: Can CRUD all data only within their department (all related data to their department).
     - **Normal User**: Can view department data but cannot modify it.
   - Store user data in a `Users` table and roles in a `Roles` table, with `department_id` in `Users` to assign users to specific departments (May have `NULL` in `department_id` for Super Admin to indicate unrestricted access.).

2. **Data Access Control**
    - Use the `department_id` in the `Users` table to enforce department-level access.
    - Only allow Department Admins to view or modify employee data within their department.
    - Normal Users should only view data, without edit permissions, and only within their department.
3. **Views and Queries for Department-Specific Data**
    - Define views or filtered queries to ensure that each user only sees data relevant to their department.
    - For example:
        - A **Normal User** in the Research department should only see employees, projects, and departments related to Research.
    - Implement these restrictions using SQL views that are filtered by department.
4. **Authentication and Authorization**
    - Implement user authentication (e.g., login/logout).
    - After login, set session data based on the user's role and department.
    - Enforce authorization by checking the user's role and department before displaying or modifying data.
    - You are free to use `flask_login` or any other modules to handle Authentication.
5. **User Interface with Role-Based Display**
    - Create a dashboard where:
        - Super Admins can view all departments and users and have full access.
        - Department Admins can only see data for their departments.
        - Normal Users have a restricted view of only their department's data.
    - Define a `base.html` template to share common UI elements and display relevant links based on user roles.

Suggested Database Schema Extensions

Add these tables to manage users, roles, and departments:

- **Users**

```
CREATE TABLE Users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role_id INT REFERENCES Roles(id),
    department_id INT REFERENCES Department(dnumber)  -- Nullable for
       Super Admin
);
```

- **Roles**

```
CREATE TABLE Roles (
    id SERIAL PRIMARY KEY,
    role_name VARCHAR(50) UNIQUE NOT NULL
);
```

Sample Views for Department-Specific Data Access

1. **Employee View per Department**:

```
CREATE VIEW DepartmentEmployees AS
SELECT e.*, d.dname
FROM Employee e
JOIN Department d ON e.dno = d.dnumber
WHERE e.dno = (
        SELECT department_id
        FROM Users
        WHERE id = CURRENT_USER_ID
    )
    OR (
        SELECT department_id
        FROM Users
        WHERE id = CURRENT_USER_ID
    ) IS NULL;
```

2. **Project View per Department**:

```
CREATE VIEW DepartmentProjects AS
SELECT p.*, d.dname
FROM Project p
JOIN Department d ON p.dnum = d.dnumber
WHERE
    p.dnum = (
        SELECT department_id
        FROM Users
        WHERE id = CURRENT_USER_ID
    )
    OR (
        SELECT department_id
        FROM Users
        WHERE id = CURRENT_USER_ID
    ) IS NULL;
```

This adjusted design uses `department_id` in `Users` for filtering access to department-specific data and supports a **Super Admin** user with unrestricted access by setting their `department_id` to `NULL`.

Bonus Requirements

To qualify for the bonus, all core project requirements must be fully implemented. Bonus marks can be earned by completing additional features as follows:

- **Filtering for Joined Data (1 mark):** Add a filtering option for lists that include joined data, allowing users to filter based on a related field.
- **Export Filtered Data to Excel (1 mark):** Implement functionality to extract the filtered list into an Excel file that users can download.
- **Excel File Upload and Data Insertion (2 marks):** Add the ability to upload an Excel file, read its contents, and insert the data into one of the database tables.