1. User management Module

1. User account creation and management: The module allows administrators to create and manage user accounts, including adding new users, modifying user information, and deleting users.
2. Role-based access control: The module allows administrators to define roles and assign permissions to each role.
3. Authentication and password security: The module provides authentication functionality to ensure that only authorized users can access the application.

Consideration :

* 1. Password Policies (Front-end)
  2. multi-factor authentication
  3. Implement rate limiting
  4. Use secure cookie settings
  5. token-based authentication:
  6. Use secure cookie settings: Use secure cookie settings, such as the Secure and HttpOnly flags, to prevent cookie theft and session hijacking attacks.

Security CSRF Token in Django

* UserRoles model : a model for authentication system for PCMS
  + username
  + password
  + user\_role
* UserProfileInformation: a data model for storing the User Information
  + First name,
  + Last\_name,
  + age, Gender,
  + date\_of\_birth etc.

1. User activity logging and audit trails: The module tracks user activity within the application, including login attempts, password changes, and other actions. This provides an audit trail that can be used for security and compliance purposes.
2. User self-service features: The module may provide self-service features such as password resets and account updates to enable users to manage their own accounts without the need for administrator intervention.
3. Integration with external systems: The module may integrate with external systems such as LDAP or Active Directory to enable centralized user management and authentication.

Security Used

The Django CSRF Middleware generates a unique CSRF token for each user session using a cryptographically secure pseudorandom number generator (PRNG). This ensures that the CSRF token is random and unpredictable, making it difficult for an attacker to guess the token and mount a CSRF attack. When a user visits a Django site, the CSRF Middleware generates a new CSRF token and stores it in the user's session. The CSRF token is also included in a cookie called `csrftoken`, which is sent to the user's browser and stored there. When the user submits a form or makes a POST request to the Django site, the CSRF Middleware checks that the CSRF token in the request matches the CSRF token in the user's session and in the `csrftoken` cookie. If the CSRF tokens match, the request is considered valid. If the CSRF tokens don't match, the request is rejected as a potential CSRF attack.

By including a CSRF token in each form and API request, the CSRF Middleware protects against CSRF attacks by ensuring that each request comes from a legitimate source and not from a malicious attacker.

Settings.py

CSRF\_COOKIE\_SECURE = False

1. CSRF\_COOKIE\_SECURE: used to access csrftoken only using https protocol
2. CSRF\_COOKIE\_HTTPONLY = True
3. CSRF\_COOKIE\_SAMESITE : the browser will only send the CSRF cookie on requests that originate from the same site as the server that set the cookie
4. CSRF\_USE\_SESSIONS = True
5. X\_FRAME\_OPTIONS = "ALLOWALL"
6. CSRF\_COOKIE\_NAME = 'csrftoken'
7. CSRF\_HEADER\_NAME = 'HTTP\_X\_CSRFTOKEN'

