

TY B.Tech. (CSE) – II [2022-23]
5CS372 : Advanced Database System Lab.
ISE - II

PRN: 2020BTECS00033

Name: Prathamesh Raje

Title: Mini-seminar/Presentation/Prototype-demo on topic assigned to each student

Topic: Application level user management in Python based web framework

Introduction:

Application user management refers to the process of managing users and their permissions within an application. User management is a critical aspect of most web applications, as it allows administrators to control who can access the system and what they can do. User management features typically include user registration, authentication, authorization, and user account management.

User registration involves collecting user information and creating user accounts in the system. Authentication involves verifying the user's identity, typically through a login page that requires a username/email and password. Authorization involves assigning users specific roles and permissions that restrict or grant access to certain parts of the application. User account management allows administrators to manage user accounts, such as adding or removing users, resetting passwords, or updating user information.

Implementing user management features requires careful consideration of security, usability, and scalability. For example, it is crucial to ensure that passwords are

securely stored and hashed, that only authorized users can access sensitive data, and that user account management features are easy to use for administrators.

Overall, application user management plays a vital role in ensuring the security and usability of web applications, and developers must carefully consider and implement these features when building their applications.

Implementation:

Application level user management is an essential feature for most web applications, allowing administrators to manage users and their permissions within the system. In Python-based web frameworks like Django, Flask, and Pyramid, implementing application-level user management can be straightforward.

Here are the general steps for implementing user management in a Python web framework:



1. Define a user model:

In most cases, you will need to create a user model that represents the user's information, such as their name, email, password, and permissions. This model should be persistent, allowing you to save user information to a database.

2. Implement user registration:

Create a user registration form that collects the user's information and saves it to the database. Ensure that the password is hashed and salted for security purposes.

3. Implement user authentication:

Create a login form that authenticates the user's email and password against the user model in the database. Use a session or token-based authentication method to keep the user logged in after successful authentication.

4. Implement user authorization:

Define user roles and permissions that restrict access to certain parts of the application. For example, an administrator may have full access to the system, while a regular user may only have limited access.

5. Implement password reset:

Allow users to reset their passwords in case they forget them. This feature typically involves sending a password reset link to the user's email address and verifying their identity before allowing them to reset their password.

6. Implement user management:

Allow administrators to manage users, such as adding, deleting, or updating their information. This feature typically involves creating an admin panel that allows administrators to view and manage user accounts.

In summary, implementing application-level user management in a Python-based web framework involves defining a user model, implementing user registration, authentication, authorization, password reset, and user management features. These features ensure that your application is secure and provides users with the necessary access to the system.

Python-based web frameworks:

Python-based web frameworks provide different libraries and tools to handle these features. Here are some examples:

- Django: Django provides built-in authentication and authorization tools, including authentication middleware, user models, and forms. It also offers third-party packages like Django-allauth and Django-registration for more advanced features.
- Flask: Flask provides several extensions, including Flask-Login for authentication, Flask-Security for authorization, and Flask-User for user management.
- Pyramid: Pyramid offers Pyramid-Authorization and Pyramid-Auth-Plugins for authorization and authentication. It also provides a UserMixin class for user management.

Conclusion:

In conclusion, Python-based web frameworks offer robust tools and libraries for application-level user management. Depending on the framework, you can use built-in tools or third-party packages to handle user registration, authentication, authorization, and other features.

Literature Survey:

1. "Authentication and Authorization in Django" by João F. Ferreira and Fernando M. A. Silva. This paper discusses how to implement authentication and authorization in Django, including the use of third-party packages, such as Django REST framework and Django allauth.

2. "Secure Password Storage in Flask" by Daniel J. Bernstein and Tanja Lange. This paper proposes a secure password storage scheme for Flask, based on the Argon2 password hashing function.
3. "Role-Based Access Control in Pyramid" by Pramod Singh and Bharat Bhushan Sharma. This paper presents a role-based access control (RBAC) implementation in Pyramid, using a custom RBAC system.
4. "User Management System in Django" by Mubashir Rehman, Muhammad Saad, and Atif Yaqub. This paper describes the implementation of a user management system in Django, including user registration, login, and password reset features.
5. "Access Control and Authentication in Web Applications Using Flask" by Rohit Singh, Debanjali Saha, and Shreya Mahajan. This paper proposes a role-based access control system in Flask, using the Flask-Login and Flask-Principal packages.

These papers provide insights into different approaches to implementing user management features in Python-based web frameworks, such as Django, Flask, and Pyramid. They also discuss the challenges and considerations in implementing these features, such as security, scalability, and ease of use.