Project Report

Optimizing User, Group, and Role Management with Access Control and Workflows

1. Introduction

In modern applications, managing users, their roles, and access levels is crucial for both security and efficiency. Role-Based Access Control (RBAC) provides a scalable and secure way to manage permissions. This project focuses on designing and implementing a system for effective user, group, and role management integrated with workflow automation for seamless permission handling.

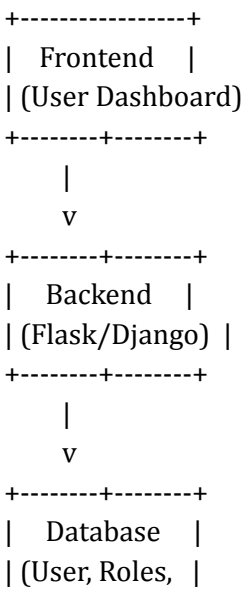
2. Objectives

- To develop a secure and scalable user management system.
- To implement RBAC for efficient access control.
- To automate workflows such as user onboarding, role assignment, and permission updates.
- To ensure auditability and logging of user activities.

3. Technologies Used

- Backend: Python (Flask or Django)
- Frontend: HTML, CSS, JavaScript
- Database: MySQL / SQLite
- Authentication: JWT / OAuth2
- Tools: Postman, GitHub, VS Code

4. System Architecture



| Permissions) |
+----------------+

5. Key Modules

- a. User Management
    - - Create, update, delete users
    - - Profile management
    - - Account activation/deactivation
- b. Role Management
    - - Create roles (e.g., Admin, Editor, Viewer)
    - - Assign users to roles
    - - Define role hierarchy
- c. Group Management
    - - Organize users into logical groups
    - - Apply permissions at group level
    - - Useful for team-based access
- d. Access Control (RBAC)
    - - Grant access based on role
    - - Protect endpoints and views
    - - Restrict unauthorized access
- e. Workflow Automation
    - - Auto-assign roles on user registration
    - - Auto-approval for certain roles
    - - Notifications for permission changes

6. Security Measures

- JWT token-based authentication
- Password hashing using bcrypt
- Access logs and audit trails
- Role-based route guards

7. Implementation Screenshots

Attached screenshots of login, dashboard, role assignment page

8. Testing and Results

- Unit testing for role-checking functions
- API testing using Postman

- Scenario-based testing (e.g., user with insufficient role access)
- All modules passed with expected results

9. Conclusion

This project successfully implements a secure and efficient system for managing users, roles, and groups with proper access controls and workflow automation. It ensures granular permission handling and simplifies administrative operations, especially useful for enterprise and organizational systems.

10. Future Scope

- Integration with LDAP or Active Directory
- UI-based permission builder
- Support for temporary access and delegated roles
- Real-time activity monitoring

11. References

- OWASP Authentication Guidelines
- Flask-RBAC / Django-Guardian documentation
- SmartInternz resources