NSS Council Management System

A CS814 Course Project Report

Under the Guidance of
Prof. Mahendra Pratap Singh

Submitted by
Karan Singh Bisht 202CS015
Sayali Arun Deo 202IS009

Department of Computer Science and Engineering
National Institute of Technology Karnataka
P.O. Srinivasnagar, Surathkal, Mangalore-575025
Karnataka, India
January 2021

Table of Contents

Page No.

1. Introduction

Give detailed information about your application, like use of application, types of users and their functionalities, etc.

1.1. About the application :

The **National Service Scheme (NSS)** is a Central Sector Scheme of the Government of India, Ministry of Youth Affairs & Sports with the sole aim to provide hands on experience to young students in delivering community service. Within states, each university has a University level NSS cell under which institutions (schools and colleges) based NSS units operate. A unit within an institution typically comprises 100 students with 10-20 council members.

The focus of our application is to provide an easy to use interface for such institution-level NSS units to manage user roles, permissions, etc. using RBAC policy.

1.2. Technology Stack

i. Browser : Google Chrome, Firefox, etc.
ii. Code Editor : Visual Studio Code, Sublime Text, etc.
iii. Version Control : Github for Desktop, Gitkraken, etc.
iv. Technology Stack :
    1. Frontend : HTML, CSS, Jinja 2 Templates
    2. Backend : Python Flask Framework.
    3. Database : SQLAlchemy - A Python SQL toolkit

1.3. Use of the application :

It can become increasingly tedious to monitor individual user's assignments and allowing them permissions to perform their respective tasks. Role Based Access Control Model is the perfect solution for such a functionality driven system. Every NSS unit consisting of the roles mentioned below can use the system for managing their users and tasks.

- For Registering new members in NSS.
- For keeping a track of the number of activities and events they have participated in.
- For recording the amount of funds collected for the events.
- For keeping a track of colleges where advertising is carried out.
- For documenting the report of each event, etc.

1.4. Types of users and their functionality :

No	Role	Abbreviation	Functionalities
1	Administrator	Admin	Add role, Delete role, Assign role, Revoke Role
2	Chairperson	CP	Approve Funds, Approve Report, Approve Event, Approve Advt.
3	Event Head	EH	Schedule Event, Modify Event
4	Public Relations Head	PRH	Add advertising, Delete Advertising
5	Treasurer	TR	Disburse Fund, Collect Fund
6	Documentation Head	DH	Add members, Make Report
7	Event Team	ET	Execute Event
8	PR Team	PRT	Perform Advertising
9	NSS Member	NM	Attend Event
10	Registered Visitor	RV	View Events

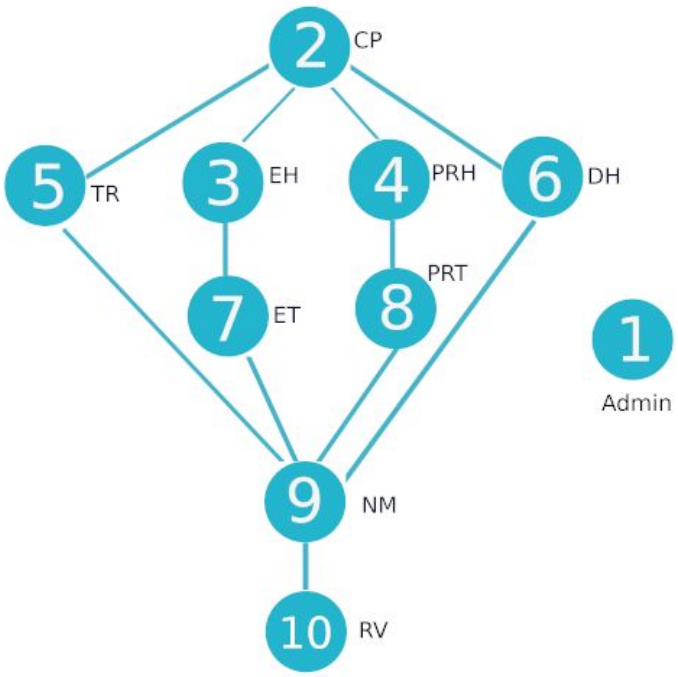

Fig 1. RBAC Hierarchy of the system

2. Authorization

2.1. Need of RBAC based authorization.

- Through RBAC, you can control what end-users can do at all levels. We can designate user different roles, and align roles and access permissions with the user positions in the organization.
- **Improving operational efficiency :-** RBAC lets organizations quickly add and change roles, as well as implement them across platforms, operating systems (OSes) and applications. It also cuts down on the potential for error when user permissions are being assigned.
- **Giving administrators increased visibility.** RBAC gives network administrators and managers more visibility and oversight into the system, while also guaranteeing that authorized users and guests on the system are only given access to what they need to do their jobs.
- **Reducing costs.** By not allowing user access to certain processes and applications, we may conserve or more cost-effectively use resources, such as network bandwidth, memory and storage.
- **Decreasing risk of breaches and data leakage.** Implementing RBAC means restricting access to sensitive information, thus reducing the potential for data breaches or data leakage.

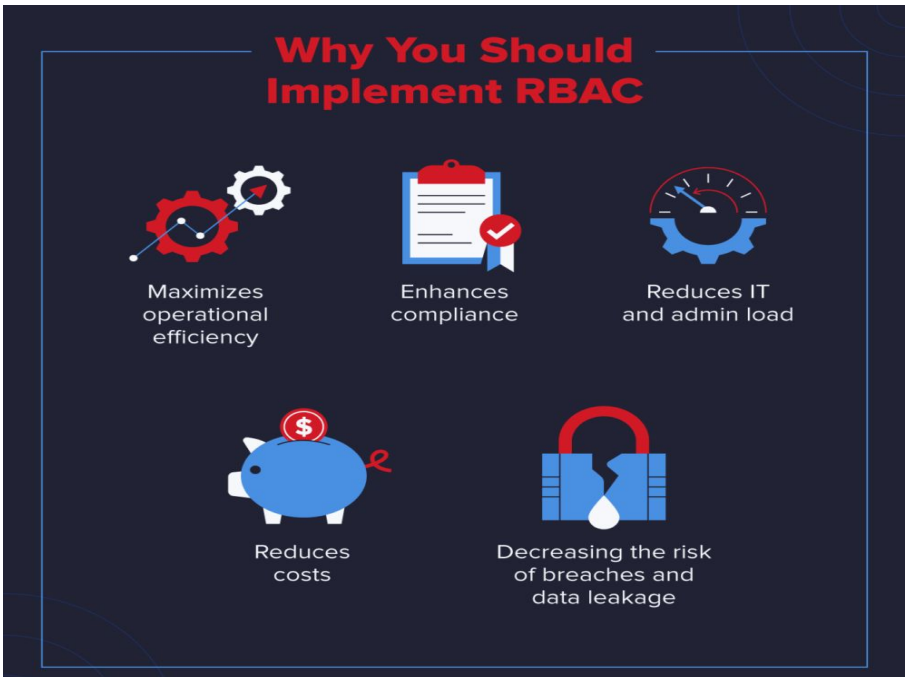

Fig 2. Need of RBAC in the system

2.2. Components of RBAC

1. **List of Subjects :**
   {Admin, CP, EH, PRH, DH, TR, ET, PRT, NM}

2. **List of Objects :**
   {Event, Fund, Advertising, Report}

3. **Access Control** :
   (Subject s can access object o denoted as s -> o)
   {EH -> Event, PRH -> Advertising, DH -> Report, TR -> Fund}

4. **Administrative Role :**
   Admin is the Administrative Role in our system.
   The Admin can create new roles, delete existing roles, assign roles to users and revoke roles from users.

5. **Permissions :**
   Admin -> {AddRole( ), DeleteRole( ), AssignRole( ), RevokeRole( ) }
   CP -> {ApproveFund( ), ApproveEvent( ), AllowAdvertising( ), ApproveReport( )}
   ET -> {PlanEvent( )}
   PRT -> {AdvertiseEvent( )}
   DH -> {MakeReport( )}
   TR -> {CollectFund( ), DisburseFund( )}
   ET -> {ExecuteEvent( )}
   PRT -> {PerformAdvertising( )}
   NM -> {AttendEvent( )}
   RV -> {ViewEvents( )}

6. **Constraints :**
   a. **Cardinality** :
      Only a single user is assigned the role of CP at any time instance. Also, the maximum number of NSS Members in the system is 100.
   b. **Prerequisites** :
      The minimum requirement to be assigned any role is that the user must be an NSS Member i.e. (NM). Thus, admin cannot assign higher roles to any user who has the role RV only.
   c. **Mutual Exclusion** :
      An EH cannot be assigned the role of PRH, DH, TR and vice-versa as these roles have mutually exclusive tasks and cannot execute both.

7. **RBAC Model :**
   Our System follows the **RBAC3 Model**.
   RBAC3 comprises of the following :
   1. Role-based Access Control as in RBAC
   2. Hierarchy in Roles as in RBAC1
   3. Constraints present in RBAC2
   4. Combination of RBAC1 and RBAC2

2.2.1. Support for Security Principles :

Least Privilege
Eg. Event Head only needs to plan events and thus does not have permission to access reports or advertising or funds.
Separation of Duty
Eg. In order to disburse funds, treasurer TR first needs to ask Vice chairperson VCP for approval only after which funds can be dispersed thus ensuring a check at two-levels.
Data Abstraction
Eg. Instead of read, write, etc., permissions, abstract permissions e.g., CollectFund and DisburseFund.

2.3. Components of the Administrative Model

Admin is the Administrative Role in our system.
The Admin can create new roles, delete existing roles, assign roles to users and revoke roles from users.

2.4. Implementation of Authorization

Authorization is implemented using Flask-login library. It provides user session management for Flask. It handles the common tasks of logging in, logging out, and remembering your users' sessions over extended periods of time. It will enable the following :

- Store the active user’s ID in the session, and let you log them in and out easily.
- Let you restrict views to logged-in (or logged-out) users.
- Handle the normally-tricky “remember me” functionality.
- Help protect your users’ sessions from being stolen by cookie thieves.
- Possibly integrate with Flask-Principal or other authorization extensions later on.

Two parameters that are essential for logging in our application are *'Username'* and *'Password'*. New users can choose a *'Username'* and *'Password'* for themselves at the time of registering on the application.

The authorization interface is common across all roles present in the system. The system internally decides to guide the user to its respective dashboard based on its role.

3. Conclusion

The essential functionality of the NSS Council Management System is managing the different tasks of the various authorized roles in a systematic manner. This is very effectively achieved using the RBAC model. Role based access control interference is a relatively new issue in security applications, where multiple user accounts with dynamic access levels may lead to encryption key instability, allowing an outside user to exploit the weakness for unauthorized access. Key sharing applications within dynamic virtualized environments have shown some success in addressing this problem. Thus we can conclude that the Role Based Access Control model can be successfully implemented for the NSS Event management system.

4. References

[1] Sandhu, R., et al., Role-based Access Control Models. IEEE Computer, 38–47 (1996).

[2] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer George Mason University, The ARBAC97 Model for Role Based Administration of Roles,105-135 (1999).

[3] 15th National Computer Security Conference (1992) Baltimore, Oct 13-16, 1992. pp. 554 - 563.

[4] A Review Paper Role Based Access Control, Anthony Rhodes and William Caelli.

[5] ACM Transactions on Information and System Security, Vol. 4, No. 3, August 2001, Pages 191-223.

[6] Python Documentation : https://docs.python.org/3/

[7] Flask Documentation : https://flask.palletsprojects.com/en/1.1.x/

[8] SQLAlchemy Documentation : https://docs.sqlalchemy.org/en/13/