SecureMvc-Preptest - Wipro NGA .Net Fullstack Angular

Project Report: SecureMvcRbac

1. Executive Summary

SecureMvcRbac is a Role-Based Access Control (RBAC) implementation for ASP.NET Core MVC applications. It enhances security by managing user permissions based on roles, ensuring that users can access only the resources they're authorized for. This project demonstrates the integration of RBAC in a modern web application, focusing on scalability, maintainability, and security.

---

2. Project Objectives

- **Implement RBAC**: Develop a robust RBAC system to manage user roles and permissions.
- **Enhance Security**: Ensure that users have access only to authorized resources.
- **Demonstrate Best Practices**: Showcase the use of ASP.NET Core MVC in building secure applications.
- **Provide a Template**: Offer a reusable RBAC implementation for future

---

3. Methodology

- **Framework**: ASP.NET Core MVC
- **Authentication**: ASP.NET Identity
- **Authorization**: Custom Role-Based Authorization Filters
- **Database**: SQL Server with Entity Framework Core
- **UI**: Razor Views with Bootstrap for responsive

---

4. System Architecture

- **Model-View-Controller (MVC)**: Separation of concerns for better maintainability.
- **Role-Based Access Control**: Users assigned to roles; roles have permissions to access resources.
- **Custom Authorization Filters**: Interceptors to enforce role-based access on controllers and actions.

---

5. Key Features

- **User Registration and Login**: Secure user authentication using ASP.NET Identity.

- **Role Management**: Admin interface to assign and manage roles.
- **Access Control**: Authorization filters to restrict access based on user roles.
- **Audit Logs**: Logging of user activities for security auditing

---

6. Challenges and Solutions

- **Challenge**: Ensuring secure password storage and management.
    - **Solution**: Utilized ASP.NET Identity's built-in password hashing and salting mechanisms.
- **Challenge**: Implementing granular access control.
    - **Solution**: Developed custom authorization filters to handle complex permission scenarios.

---

7. Results and Outcomes

- **Enhanced Security**: Implemented role-based access, reducing unauthorized access incidents.
- **Scalability**: The RBAC system can be easily extended to accommodate additional roles and permissions.
- **Maintainability**: Clear separation of concerns and modular design facilitate future updates and maintenance.

---

8. Conclusion

SecureMvcRbac successfully demonstrates the implementation of RBAC in an ASP.NET Core MVC application. It provides a secure, scalable, and maintainable solution for managing user access based on roles. This project serves as a valuable reference for developers looking to integrate RBAC into their applications.
