RBAC Module Documentation for Web Application

Introduction:

This document outlines the requirements for a Role-Based Access Control (RBAC) module that restricts access to specific screens and features based on user roles. The module aims to provide flexibility in managing permissions and ensure secure access.

Scope:

The RBAC module will:

- Allow admins to define roles and assign permissions.
- Provide role-based restricted views for users after login.
- Ensure secure and efficient permission management.

Functional Requirements:

1. **Admin Module**
    - **Manage Roles:**
        - Create, edit, or delete roles.
    - **Assign Permissions:**
        - Assign access to specific screens or features for each role.
        - View existing permissions assigned to roles.
    - **Audit Logs:**
        - Maintain logs of role and permission changes for accountability.
2. **User Module**
    - **Role-Based UI Access:**
        - Display only the permitted screens and features after user login.
    - **Unauthorized Access Handling:**

    - Restrict access to unauthorized features with an appropriate error message or redirection.

Non-Functional Requirements:

- **Performance:** Role checks should occur within 200ms to avoid login delays.
- **Security:**
    - Permissions stored securely in the backend.
    - Prevent unauthorized access via direct URL manipulation.
- **Scalability:** Support up to 10,000 roles and permissions.

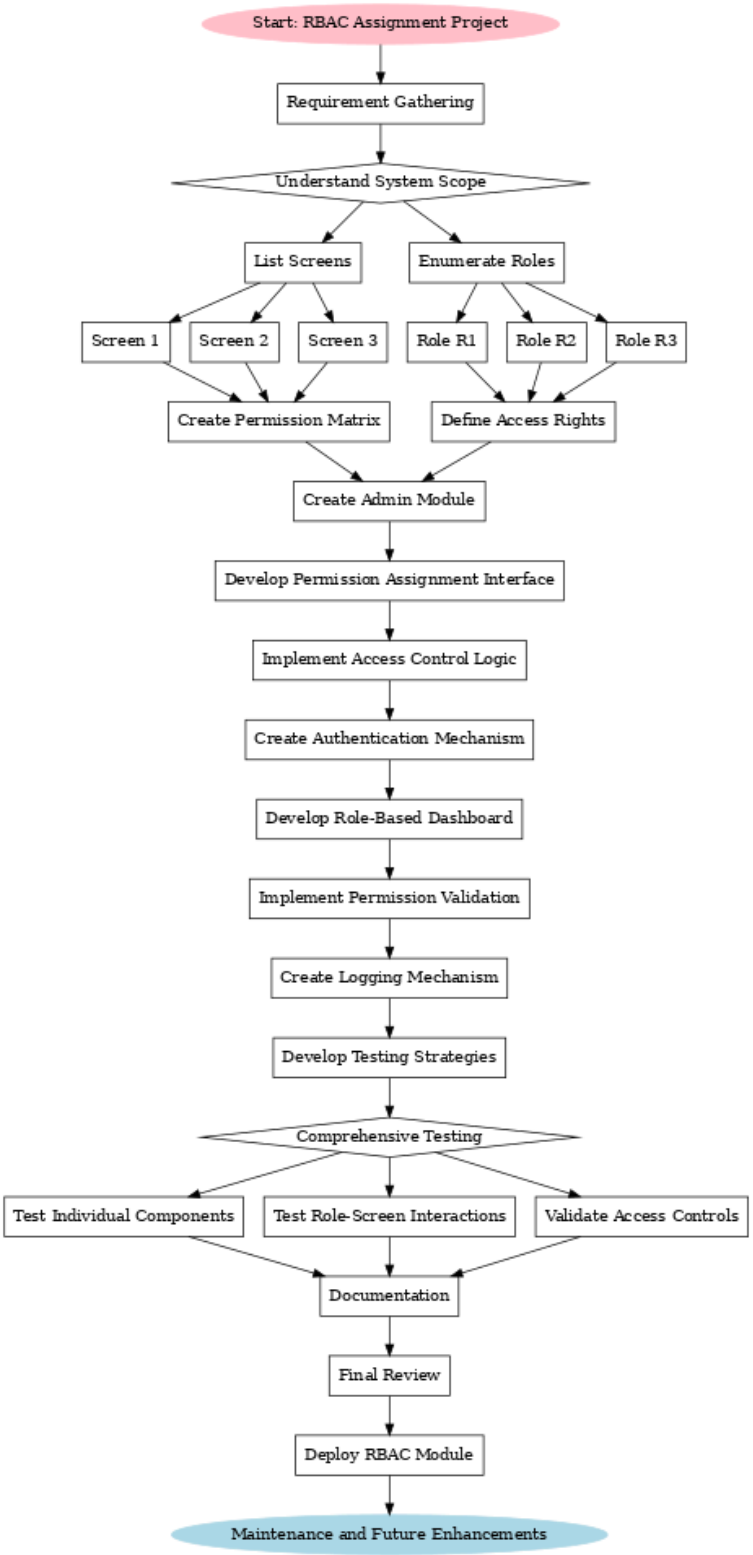
Use Case Scenarios:

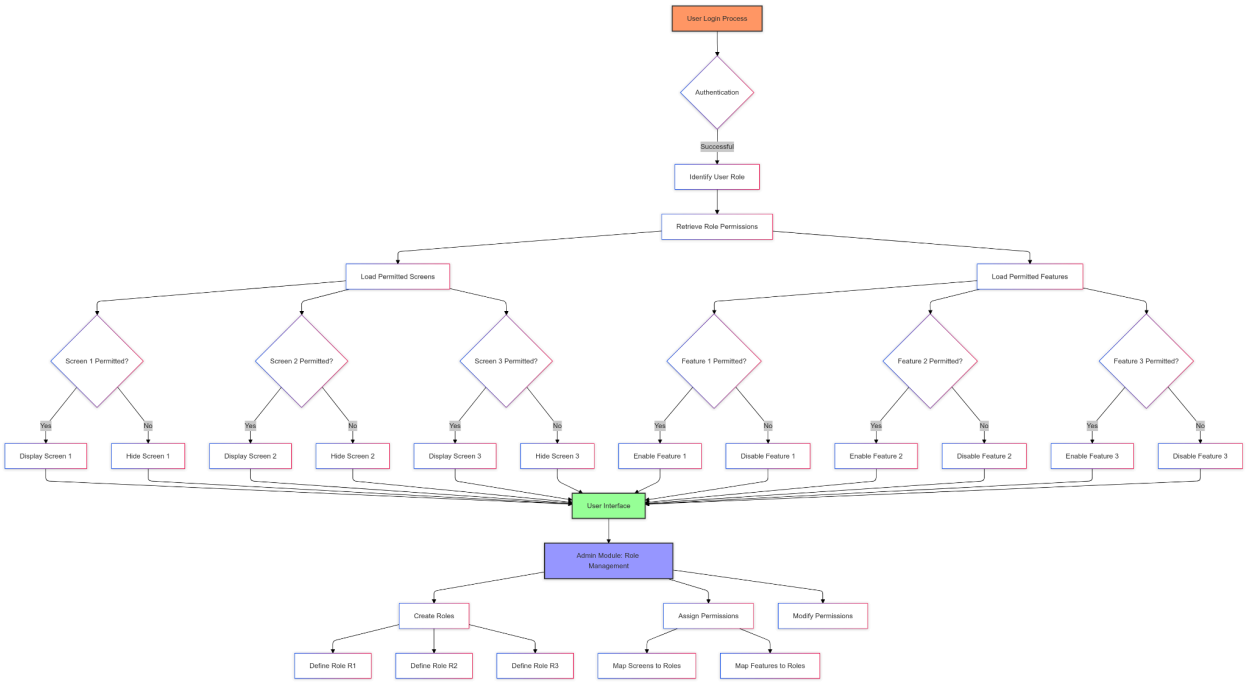
- **Admin Assigns Permissions to Roles**
    1. Admin logs in.
    2. Navigates to "Manage Roles" in the Admin Dashboard.
    3. Selects a role (e.g., R1).
    4. Assigns screens (Screen 1, Screen 2) and features (Feature 1, Feature 2).
    5. Saves changes.

- **User Logs In and Sees Role-Based UI**
    1. User logs in.
    2. System checks assigned role and permissions.
    3. Displays only the allowed screens and features.

- **Unauthorized Access Attempt**
    1. User attempts to access an unauthorized screen or feature.
    2. System displays an error message: “Access Denied.”

Flowchart:


Start: RBAC Assignment Project
Requirement Gathering
Understand System Scope
List Screens
Enumerate Roles
Screen 1
Screen 2
Screen 3
Role R1
Role R2
Role R3
Create Permission Matrix
Define Access Rights
Create Admin Module
Develop Permission Assignment Interface
Implement Access Control Logic
Create Authentication Mechanism
Develop Role-Based Dashboard
Implement Permission Validation
Create Logging Mechanism
Develop Testing Strategies
Comprehensive Testing
Test Individual Components
Test Role-Screen Interactions
Validate Access Controls
Documentation
Final Review
Deploy RBAC Module
Maintenance and Future Enhancements



Roles and Permissions Matrix:

Role	Screen 1	Screen 2	Feature 1	Feature 2	Feature 3
R1	✅	❌	✅	❌	✅
R2	✅	✅	✅	✅	❌
R3	❌	✅	❌	✅	✅

Risks and Assumptions:

- **Risks**
  - Misconfigured roles may block valid user access.
  - Potential security risks if permissions are not encrypted.
- **Assumptions**
  - Admins are trained to manage roles and permissions.
  - Users cannot bypass role checks through backend manipulation.

Test Scenarios:

Test Case	Description	Expected Result
Login with Role R1	Verify user R1 only sees permitted screens	Only Screen 1 and Feature 1 visible
Login with Role R2	Verify user R2 sees all assigned features	Screen 1, Screen 2, Feature 2 visible
Modify Role Permissions	Admin changes permissions for R3	R3's interface updates accordingly
Unauthorized API Access	User attempts to access a blocked feature	Access is denied