Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose

This document specifies the requirements for developing a Role-Based Access Control (RBAC) module. The system will allow applications to manage users, roles, and permissions, ensuring that only authorized users can perform specific actions. It is designed to function in two modes: SaaS API Service (centralized RBAC system for multiple clients) and Standalone Package (plug-and-play module for existing projects).

1.2 Scope

The RBAC system will: - Manage users, roles, and permissions. - Provide authentication and authorization with JWT. - Allow SaaS mode deployment with multi-tenant support. - Provide REST APIs for integration. - Offer standalone library for NestJS/Node.js projects.

2. Functional Requirements

- User Management: Register, login, device restriction, and session management. - Role Management: Create, update, delete roles. - Permission Management: Define granular actions (read, write, delete). - Role Assignment: Assign/revoke roles to/from users. - Access Check: Validate user permissions in real time via API.

3. Non-Functional Requirements

- Scalability: Should support thousands of requests per second. - Security: JWT authentication, password hashing, HTTPS. - Availability: 99.9% uptime for SaaS deployment. - Performance: API response < 200ms. - Database: PostgreSQL optimized with indexing.

4. Tech Stack

- Backend: NestJS (Node.js Framework) - Database: PostgreSQL - Authentication: JWT + Role Guards - Deployment: Docker + Kubernetes - Client Tools: Postman for API testing, Swagger for docs

5. Database Schema

Tables: - users (id, username, email, password_hash) - roles (id, name) - permissions (id, name) - role_permissions (role_id, permission_id) - user_roles (user_id, role_id)

6. API Endpoints

Authentication: - POST /auth/register - POST /auth/login Roles: - POST /roles - GET /roles - PUT /roles/{id} - DELETE /roles/{id} Permissions: - POST /permissions - GET /permissions Assignments: - POST /roles/assign - POST /permissions/check

7. Deployment

- SaaS Mode: Deploy on cloud (AWS/GCP/Azure) with Kubernetes. - Standalone Mode: Install via npm package and connect to existing DB.

8. Conclusion

This RBAC module provides a flexible and secure way to manage access control. It is designed for both SaaS-based centralized usage and standalone deployment, making it reusable across multiple industries.