### Project Requirement Document for Gocloak

#### 1. **Introduction**

* **Project Name**: Gocloak
* **Purpose**: To develop an alternative to Keycloak, providing enhanced features and addressing its limitations.
* **Document Version**: 1.0
* **Date**: April 24, 2025

### 2. **2. Product Features**

**2.1 Admin Panel (Internal Use)**

* **Realm Management**: Administrators can create, manage, and delete realms.
* **User & Role Management**: Add, update, and delete users, assign roles and permissions.
* **Client Management**: Register and manage client applications for authentication.
* **Audit Logs**: Track changes and actions performed within the system.
* **Security**: Admin access will be secured via a dedicated admin login system with role-based access control.

**2.2 REST API (External Use)**

* **Authentication**: OAuth2 and OpenID Connect (OIDC) token endpoints.
* **User Management**: Endpoints for creating, updating, and deleting users within realms.
* **Realm Management**: Create, update, and list realms via API.
* **Client Management**: Endpoints to register and manage clients (applications) within a realm.
* **Roles & Permissions**: Manage roles and assign them to users.

**2.3 Authentication Flows**

* **Password Grant**: Secure login using username/password.
* **Authorization Code Flow**: Secure authorization for web and mobile apps.
* **Implicit Flow**: For client-side applications requiring tokens directly.

**3. Technical Requirements**

**3.1 Platform**

* **Programming Language**: Go (Golang) for performance, concurrency, and scalability.
* **Web Framework**: **Gin** or **Echo** for building the REST APIs.
* **Database**: **PostgreSQL** for production; **SQLite** for development.
* **Token Format**: **JWT (JSON Web Tokens)** for OAuth2 and OIDC support.
* **Authentication Protocols**: **OAuth2**, **OIDC**.
* **Deployment**: Dockerized for easy deployment; Kubernetes-compatible via Helm chart.
* **Security**: Enforce **HTTPS**, **CSRF protection** for the Admin Panel, and **JWT validation** for API access.

**3.2 Security**

* Use **JWT** for token management (RS256 or HS256).
* Ensure **HTTPS** for all public-facing endpoints.
* Implement **CSRF protection** in Admin Panel and input validation throughout.

**4. User Stories**

**4.1 As an administrator:**

* I want to manage realms so that I can separate identity data for different clients/applications.
* I want to add/remove users from realms and assign roles so that I can control access to resources.
* I want to monitor activities through audit logs to track user actions for security purposes.

**4.2 As a developer:**

* I want to integrate OAuth2/OIDC authentication with GoCloak to secure user access in my app.
* I want to use the GoCloak API to manage users, roles, and clients programmatically.

**4.3 As a system:**

* I want to handle up to 100k concurrent users per realm and ensure API responses are under 200ms to ensure scalability and performance.

#### 5. **Stakeholders**

* **Project Sponsor**: [Name]
* **Project Manager**: [Name]
* **Development Team**: [Names]
* **End Users**: Organizations requiring identity and access management solutions.