RBAC API Documentation

1. Users API

Base Path: `/api/users`

1.1. Get All Users

- **Endpoint:** `GET /`
- **Description:** Retrieves a list of all users, including their assigned roles and groups.
- **Success Response (200 OK):**

```
[
  {
    "id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "nafath_id": "1122334455",
    "email": "admin@example.com",
    "full_name_en": "Admin System Main One",
    "status": "Active",
    // ... other user fields
    "roles": ["admin"],
    "groups": []
  },
  {
    "id": "b2c3d4e5-f6a7-8901-2345-67890abcdef1",
    "nafath_id": "2233445566",
    "email": "developer@example.com",
    "full_name_en": "Developer System Secondary Two",
    "status": "Active",
    // ... other user fields
    "roles": [],
    "groups": ["engineering"]
  }
]
```

1.2. Create a New User

- **Endpoint:** `POST /`
- **Description:** Creates a new user.
- **Request Body:**

```
{
  "nafath_id": "3344556677",
  "email": "newuser@example.com",
  "phone_number": "5559876543",
  "first_name_en": "New",
  "full_name": "New User System Three",
  "status": "Active",
  // ...  user fields
}
```

- **Success Response (201 Created):**

```
{
  "id": "c3d4e5f6-a7b8-9012-3456-7890abcdef12",
  "nafath_id": "3344556677",
  "email": "newuser@example.com",
  // ...  user fields
}
```

1.3. Delete a User

- **Endpoint:** `DELETE /:id`
- **Description:** Deletes a user by their ID and removes their associated Casbin policies.
- **URL Parameter:** `id` (User's UUID)
- **Success Response (204 No Content):** An empty response body.

---

2. Roles API

Base Path: `/api/roles`

2.1. Get All Roles

- **Endpoint:** `GET /`
- **Description:** Retrieves a list of all roles and the permissions (policies) assigned to them.
- **Success Response (200 OK):**

```
[
  {
    "id": 1,
    "name": "admin",
    "description": "Administrator with full access",
    "policies": [
      ["admin", "dashboard", "read"],
      ["admin", "dashboard", "write"]
    ]
  },
  {
    "id": 2,
    "name": "editor",
    "description": "Can edit content",
    "policies": [
      ["editor", "profile", "read"]
    ]
  }
]
```

2.2. Create a New Role

- **Endpoint:** `POST /`
- **Description:** Creates a new role.
- **Request Body:**

```
{
  "name": "manager",
  "description": "Can manage projects"
}
```

- **Success Response (201 Created):**

```
{
  "id": 3,
  "name": "manager",
  "description": "Can manage projects"
}
```

2.3. Get a Single Role

- **Endpoint:** `GET /:id`

- **Description:** Retrieves a single role by its ID, including its policies.
- **URL Parameter:** `id` (Role's integer ID)
- **Success Response (200 OK):**

```
{
  "id": 1,
  "name": "admin",
  "description": "Administrator with full access",
  "policies": [
    ["admin", "dashboard", "read"],
    ["admin", "dashboard", "write"]
  ]
}
```

2.4. Update a Role

- **Endpoint:** `PUT /:id`
- **Description:** Updates a role's details (name, description).
- **URL Parameter:** `id` (Role's integer ID)
- **Request Body:**

```
{
  "description": "Administrator with full system access"
}
```

- **Success Response (200 OK):**

```
{
  "id": 1,
  "name": "admin",
  "description": "Administrator with full system access"
}
```

2.5. Delete a Role

- **Endpoint:** `DELETE /:id`
- **Description:** Deletes a role by its ID and removes all associated Casbin policies.
- **URL Parameter:** `id` (Role's integer ID)
- **Success Response (204 No Content):** An empty response body.

---

3. Groups API

Base Path: `/api/groups`

3.1. Get All Groups

- **Endpoint:** `GET /`
- **Description:** Retrieves a list of all groups, including their members (users) and assigned roles.
- **Success Response (200 OK):**

```
[
  {
    "id": 1,
    "name": "engineering",
    "description": "Engineering department",
    "users": ["2233445566"],
    "roles": ["editor"]
  },
  {
    "id": 2,
    "name": "marketing",
    "description": "Marketing department",
    "users": [],
    "roles": ["viewer"]
  }
]
```

3.2. Create a New Group

- **Endpoint:** `POST /`
- **Description:** Creates a new group.
- **Request Body:**

```
{
  "name": "sales",
  "description": "Sales department"
}
```

- **Success Response (201 Created):**

```
{
  "id": 3,
  "name": "sales",
  "description": "Sales department"
}
```

3.3. Get a Single Group

- **Endpoint:** `GET /:id`
- **Description:** Retrieves a single group by its ID.
- **URL Parameter:** `id` (Group's integer ID)
- **Success Response (200 OK):**

```
{
  "id": 1,
  "name": "engineering",
  "description": "Engineering department"
}
```

3.4. Update a Group

- **Endpoint:** `PUT /:id`
- **Description:** Updates a group's details.
- **URL Parameter:** `id` (Group's integer ID)
- **Request Body:**

```
{
  "description": "Software Engineering Department"
}
```

- **Success Response (200 OK):**

```
{
  "id": 1,
  "name": "engineering",
  "description": "Software Engineering Department"
}
```

3.5. Delete a Group

- **Endpoint:** `DELETE /:id`
- **Description:** Deletes a group by its ID and removes all associated Casbin grouping policies.
- **URL Parameter:** `id` (Group's integer ID)
- **Success Response (204 No Content):** An empty response body.

---

4. Resources API

Base Path: `/api/resources`

- This API follows standard CRUD patterns (`GET /`, `POST /`, `GET /:id`, `PUT /:id`, `DELETE /:id`) for managing resources.

4.1. Get All Resources

- **Endpoint:** `GET /`
- **Success Response (200 OK):**

```
[
  { "id": 1, "name": "dashboard", "description": "Main application dashboard" },
  { "id": 2, "name": "profile", "description": "User profile page" }
]
```

4.2. Create a Resource

- **Endpoint:** `POST /`
- **Request Body:**

```
{
  "name": "analytics",
  "description": "Data analytics page"
}
```

- **Success Response (201 Created):**

```
{ "id": 3, "name": "analytics", "description": "Data analytics page" }
```

---

5. Permissions API

Base Path: `/api/permissions`

- This API follows standard CRUD patterns (`GET /`, `POST /`, `GET /:id`, `PUT /:id`, `DELETE /:id`) for managing permissions (actions).

5.1. Get All Permissions

- **Endpoint:** `GET /`
- **Success Response (200 OK):**

```
[
  { "id": 1, "name": "read", "description": "Read access to a resource" },
  { "id": 2, "name": "write", "description": "Write access to a resource" }
]
```

5.2. Create a Permission

- **Endpoint:** `POST /`
- **Request Body:**

```
{
  "name": "execute",
  "description": "Execute a resource"
}
```

- **Success Response (201 Created):**

```
{ "id": 3, "name": "execute", "description": "Execute a resource" }
```

---

6. Associations API

Base Path: `/api/associations`

6.1. Assign a Role to a User

- **Endpoint:** `POST /users/:userId/roles`
- **Description:** Assigns a role to a user.
- **URL Parameter:** `userId` (User's UUID)
- **Request Body:**

```
{
  "roleId": 1
}
```

- **Success Response (200 OK):**

```
{
  "message": "Role 'admin' assigned to user '1122334455'"
}
```

6.2. Remove a Role from a User

- **Endpoint:** `DELETE /users/:userId/roles/:roleId`
- **Description:** Removes a role from a user.
- **URL Parameters:** `userId` (User's UUID), `roleId` (Role's integer ID)
- **Success Response (200 OK):**

```
{
  "message": "Role 'admin' removed from user '1122334455'"
}
```

6.3. Add a Permission to a Role

- **Endpoint:** `POST /roles/:roleId/permissions`
- **Description:** Adds a specific permission (action) on a resource to a role.
- **URL Parameter:** `roleId` (Role's integer ID)
- **Request Body:**

```
{
  "resourceId": 3,
  "permissionId": 1
}
```

- **Success Response (200 OK):**

```
{
  "message": "Permission 'read' for resource 'analytics' assigned to role 'viewer'"
}
```

6.4. Add a User to a Group

- **Endpoint:** `POST /groups/:groupId/users`
- **Description:** Adds a user to a group.
- **URL Parameter:** `groupId` (Group's integer ID)
- **Request Body:**

```
{
  "userId": "b2c3d4e5-f6a7-8901-2345-67890abcdef1"
}
```

- **Success Response (200 OK):**

```
{
  "message": "User '2233445566' added to group 'engineering'"
}
```

6.5. Assign a Role to a Group

- **Endpoint:** `POST /groups/:groupId/roles`
- **Description:** Assigns a role to a group, effectively granting the role's permissions to all users in that group.
- **URL Parameter:** `groupId` (Group's integer ID)
- **Request Body:**

```
{
  "roleId": 2
}
```

- **Success Response (200 OK):**

```
{
  "message": "Role 'editor' assigned to group 'engineering'"
}
```

6.6. Get All Policies and Groupings

- **Endpoint:** `GET /`
- **Description:** A utility endpoint to retrieve all raw policies and grouping policies from Casbin.
- **Success Response (200 OK):**

```
{
  "policies": [
    ["admin", "dashboard", "read"],
    ["admin", "dashboard", "write"]
  ],
  "groupings": [
    ["1122334455", "admin"],
    ["2233445566", "engineering"],
    ["engineering", "editor"]
  ]
}
```

---

7. Test API

Base Path: `/api/test`

7.1. Test Access

- **Endpoint:** `GET /`
- **Description:** A hardcoded test to check if user `1234567890` has `read` access to `test-resource`. This is primarily for internal testing.
- **Success Response (200 OK):**

```
You have access to the test resource!
```

- **Error Response (403 Forbidden):**

```
{
  "error": "Forbidden"
}
```