AG RUNBOOK — Admin UI → Roles & Permissions

Purpose:
This runbook defines a safe, UI-focused implementation for Admin → Roles & Permissions management in a multi-tenant SaaS system. The UI must strictly consume existing backend APIs and must not introduce new authorization logic.

Global Constraints:
- Do not modify backend authorization or RBAC logic
- Do not introduce new permissions implicitly
- Do not bypass tenant isolation
- Do not allow cross-tenant role visibility
- Do not expose internal IDs unnecessarily
- Assume Roles, Permissions, and UserRole APIs already exist

Capabilities Covered:
- View roles for a tenant
- Create new roles
- Edit role permissions
- Assign roles to users
- Revoke roles from users
- Delete roles (with safeguards)

Step 1 — Roles List Page:
Route:
- /admin/roles

Behavior:
- Fetch roles scoped to current tenant
- Paginate results
- Display:
- Role name
- Permission count
- Created date
- Actions (View / Edit / Delete)

Step 2 — View Role Details:
- Show role name and description
- List assigned permissions (read-only list with labels)
- List users assigned to the role (email only)
- Do not allow inline editing on this screen

Step 3 — Create Role UI:
- Button: "Create Role"
- Form fields:
- Role name

- Role description
- Permission selection (checkbox list)
- Submit via existing create-role API
- Show confirmation on success

Step 4 — Edit Role Permissions:
- Allow adding or removing permissions
- Require confirmation before saving changes
- Warn that permission changes affect all assigned users
- Submit via update-role API

Step 5 — Assign / Revoke Role to User:
- Role assignment UI within User details or Role details
- Use existing role assignment APIs
- Reflect changes immediately in UI
- Do not cache authorization decisions

Step 6 — Delete Role Safeguards:
- Disable delete if role is assigned to users
- Require explicit confirmation
- Use delete-role API only
- Show clear warning text

Step 7 — Error Handling:
- Generic error messages
- Handle permission-denied gracefully
- Do not expose backend error details

Audit Awareness:
- UI must not generate audit events directly
- Backend APIs will emit ROLE.CREATE, ROLE.UPDATE, ROLE.DELETE, ROLE.ASSIGN, ROLE.REVOKE
- UI only triggers backend actions

Success Criteria:
- Admins can manage roles and permissions safely
- Tenant isolation is preserved
- Authorization logic remains centralized
- UI reflects real permission state accurately

Out of Scope:
- Permission creation or deletion
- Policy-based authorization
- Cross-tenant role templates
- Bulk role assignment