Each **feature or module** of your system will have:

* A **Menu Key** → controls visibility in the sidebar and main routes.
* A set of **Action Keys (Access Levels)** → control what the user can do inside.

The pattern is simple and unified:

{menuKey}:{action}

Example:  
loan:view, loan:create, loan:edit, loan:delete

**🧱 2. Access Levels You Should Define**

Here’s the recommended structure for a **Loan Management SaaS**:

**🏠 Dashboard**

* dashboard:view

**👥 Customers**

* customer:view
* customer:create
* customer:edit
* customer:delete

**💰 Loans**

* loan:view
* loan:create
* loan:edit
* loan:delete
* loan:approve
* loan:renew
* loan:close

**🏦 Payments**

* payment:view
* payment:receive
* payment:refund
* payment:delete

**💍 Appraisal / Collateral**

* appraisal:view
* appraisal:create
* appraisal:edit
* appraisal:approve

**👤 Users (Tenant Users)**

* user:view
* user:create
* user:edit
* user:delete
* user:assignRole

**⚙️ Settings (Tenant Config)**

* settings:view
* settings:update
* settings:branchManage
* settings:loanConfig

**📊 Reports**

* report:view
* report:export

**🧩 System-Level (for SaaS provider only)**

* tenant:view
* tenant:create
* tenant:edit
* tenant:delete
* systemUser:view
* systemUser:create
* systemUser:edit
* systemUser:delete
* billing:view
* billing:update
* audit:view

**🧩 3. How You’ll Use These Keys**

| **Layer** | **Uses Keys For** | **Example** |
| --- | --- | --- |
| **Menu / Sidebar** | Which features to show | Only show “Loans” menu if user has any loan:\* key |
| **Frontend Route Guard** | Which routes can be opened | Block /loan route if loan:view missing |
| **Button Control** | Enable/disable action buttons | Hide Edit button if loan:edit missing |
| **Backend Middleware** | Secure API endpoints | Deny DELETE /loan if loan:delete missing |

**🔐 4. Access Level Tiers (Recommended)**

You can group them conceptually for easier role assignment:

| **Tier** | **Description** | **Example Keys** |
| --- | --- | --- |
| **View-Only** | Can only open and read data | \*:view, report:export |
| **Standard User** | Can view + create + edit own data | loan:create, loan:edit, customer:create |
| **Manager** | Can approve, delete, or modify others’ records | loan:approve, loan:delete, payment:refund |
| **Admin** | Full control within tenant | All tenant and settings keys |
| **System Admin** | Full platform control | All keys including systemUser:\*, tenant:\*, billing:\* |

**🧭 1. Two Levels of Roles**

Your SaaS has **two role categories**:

* **System-Level Roles** → manage the entire platform.
* **Tenant-Level Roles** → manage and operate each tenant’s loan business.

**🏗️ 2. Role → Permission Matrix**

**🧱 System-Level Roles (Platform Control)**

| **Role** | **Description** | **Access Keys** |
| --- | --- | --- |
| **System Super Admin** | Full control over entire SaaS platform | All keys (\*) — includes tenant:\*, systemUser:\*, billing:\*, audit:\* |
| **System Admin / Staff** | Manages tenants and system users | tenant:view, tenant:create, tenant:edit, tenant:delete, systemUser:view, systemUser:create, systemUser:edit |
| **System Support / Helpdesk** | Supports tenant issues, view-only | tenant:view, systemUser:view, audit:view |
| **System Auditor / Compliance** | View all data and audit logs only | tenant:view, systemUser:view, audit:view, billing:view |
| **Billing Manager** | Handles subscriptions and payments | billing:view, billing:update, tenant:view |

**🏢 Tenant-Level Roles (Per Organization)**

| **Role** | **Description** | **Access Keys (per menu/action)** |
| --- | --- | --- |
| **Tenant Admin / Owner** | Full control of their own tenant (manages users, settings, reports) | dashboard:view, loan:\*, customer:\*, payment:\*, user:\*, settings:\*, report:\* |
| **Branch Manager** | Manages one branch, approves loans | dashboard:view, loan:view, loan:create, loan:edit, loan:approve, customer:view, customer:create, report:view, report:export |
| **Loan Officer** | Handles loan creation and customer profiles | dashboard:view, loan:view, loan:create, loan:edit, customer:view, customer:create, customer:edit |
| **Cashier / Teller** | Handles payments and renewals | dashboard:view, loan:view, payment:view, payment:receive, payment:refund |
| **Appraiser** | Evaluates collateral items | dashboard:view, appraisal:view, appraisal:create, appraisal:edit, appraisal:approve |
| **Auditor / Accountant** | Reviews all financial transactions | dashboard:view, loan:view, payment:view, report:view, report:export |
| **Viewer / Read-Only** | Can view data but not modify | dashboard:view, loan:view, customer:view, payment:view, report:view |

**🧩 3. Optional Expansion (Later)**

If you add more features or branches in the future:

* Add roles like RegionalManager or Collector
* Reuse existing permission keys (loan:view, loan:edit) so you don’t need new logic — just assign different combinations.

**🧠 4. Design Notes**

| **Concept** | **Meaning** |
| --- | --- |
| **Menu Key** | Controls access to a main module (Dashboard, Loan, Customer, etc.) |
| **Action Key** | Controls specific actions (view, create, edit, delete, approve) |
| **Role** | A collection of access keys assigned to a user |
| **User** | Gets permissions based on their assigned role |
| **Tenant Isolation** | Tenant users can only act within their own tenant; System users can access all tenants |

**✅ 5. Why This Structure Is Excellent**

* 🔄 One unified permission system for both menu and route.
* 🧩 Extremely flexible — you can add roles or actions later without rewriting logic.
* 🔒 Secure — backend and frontend use the same matrix.
* 🧠 Easy to manage — you can later build a “Role Management” page that simply toggles keys per role.

**🧭 1. How to Structure the Role-Permission GUI**

**For each menu/feature (menu key):**

* **Show the menu name (e.g., “Tenant Users”)**
* **List action checkboxes:**
  + **Create → systemUser:create**
  + **Edit → systemUser:edit**
  + **Delete → systemUser:delete**
* **No need for a separate view checkbox because presence of the menu key already implies view access.**

**Example Table Layout:**

| **Menu / Feature** | **Create** | **Edit** | **Delete** |
| --- | --- | --- | --- |
| **Tenant Users** | **☑** | **☑** | **☐** |
| **Loans** | **☑** | **☐** | **☐** |
| **Payments** | **☐** | **☐** | **☐** |

* **In this example, the role can create and edit Tenant Users, but cannot delete them.**
* **The menu “Tenant Users” will show in the sidebar because at least one action is selected (or you can tie sidebar visibility just to the menu key).**

**🧱 2. Benefits of This Approach**

1. **✅ Very simple**
   * **Admin just toggles checkboxes for each menu; no separate view key needed.**
2. **✅ Action-specific control**
   * **Frontend buttons and backend API calls can all check the assigned action keys.**
3. **✅ Scalable**
   * **Add new actions later (e.g., approve, export) by just adding a new checkbox.**
4. **✅ Clear UX**
   * **Admin sees exactly what the role can do for each feature.**

**🧠 3. Optional Enhancements**

* **“Select All” per menu → auto-check all actions for convenience.**
* **Tooltips / labels → explain what each action does (especially useful for non-technical admins).**
* **Frontend / backend enforcement → check the menu key for route access, check action keys for buttons/API.**

**🔑 Takeaway**

* **Menu key → controls module visibility + route access**
* **Action keys → control buttons / API calls (create, edit, delete)**
* **GUI → checkboxes for action keys only; no separate view checkbox needed**

**This is exactly how enterprise SaaS RBAC systems do it. It’s simple, flexible, and secure.**