WorkspaceOps — HLR → FRD → API → Module Mapping

How to read this (important)

- **HLR** = capability (business)
- **FRD** = behavior (functional)
- **API** = contract (technical boundary)
- **Module** = code ownership (clean architecture)

This is the **bridge** from requirements → code.

1️⃣ Tenant & Workspace Onboarding

🔹 HLR0001 – User signup & tenant creation

Layer	Mapping
FRD	FRD0001 – User Signup & Tenant Creation
API	`POST /auth/signup`
Module	`auth`

Module responsibility

- Validate signup
- Create user
- Create tenant

🔹 HLR0002 – Default workspace creation

Layer	Mapping
FRD	FRD0002 – Default Workspace Initialization
API	`POST /workspaces` (internal call after signup)
Module	`workspace`

Module responsibility

- Create workspace
- Assign OWNER role

🔹 HLR0004 – Invite users

Layer	Mapping
FRD	FRD0003 – Workspace User Invitation
API	`POST /workspaces/:id/invite`
Module	`workspace-user` (or `user`)

Module responsibility

- RBAC check
- Create workspaceUsers entry

2️⃣ RBAC (Cross-cutting)

🔹 HLR0005 / HLR0006 / HLR0007 – Roles & restrictions

Layer	Mapping
FRD	Implicit across all FRDs
API	N/A (middleware enforced)
Module	`common/middleware/rbac`

Module responsibility

- Validate role
- Enforce permissions per route

📌 This is **not a single API** — it's **infrastructure logic**.

3️⃣ Entity Management

🔹 HLR0008 / HLR0009 – Create entity & assign role

Layer	Mapping
FRD	FRD0004 – Entity Creation
API	`POST /entities`
Module	`entity`

Module responsibility

- Validate entity role
- Ensure workspace scope

🔹 HLR0010 – Entity as subject of work & documents

Layer	Mapping
FRD	FRD0005 – Entity Association Behavior
API	Used implicitly in `/documents`, `/work-items`
Module	`entity` (referenced), `document`, `work-item`

📌 This HLR is **behavioral**, not endpoint-specific.

4️⃣ Document Configuration & Upload

🔹 HLR0011 / 0012 / 0013 – Document type definition

Layer	Mapping
FRD	FRD0006 – Document Type Configuration
API	`POST /document-types`
Module	`document-type`

Module responsibility

- Store schema config
- Enforce workspace ownership

🔹 HLR0014–0018 – Document upload & metadata

Layer	Mapping
FRD	FRD0007 – Document Upload Behavior
API	`POST /documents`
Module	`document`

Module responsibility

- Load document type
- Validate metadata
- Store file
- Save document record

🔹 HLR0019 / 0020 – Expiry calculation

Layer	Mapping
FRD	FRD0008 – Document Expiry Evaluation

API	`GET /documents` (computed on read)
Module	`document`

📌 No cron jobs.
📌 Computed dynamically.

5️⃣ Work Items (Generalized Compliance)

🔹 HLR0021 – Work item types

Layer	Mapping
FRD	Work Item Type Definition
API	`POST /work-item-types`
Module	`work-item-type`

🔹 HLR0022 / 0023 – Create work item

Layer	Mapping
FRD	FRD0009 – Work Item Creation
API	`POST /work-items`
Module	`work-item`

Module responsibility

- Validate entity
- Validate owner
- Set initial state

🔹 HLR0024 – Work item lifecycle

Layer	Mapping

FRD	FRD0010 – Work Item Lifecycle
API	`PATCH /work-items/:id/status`
Module	`work-item`

🔹 HLR0025 – Link documents to work items

Layer	Mapping
FRD	FRD0011 – Work Item Document Linking
API	`POST /work-items/:id/documents`
Module	`work-item-document`

6️⃣ Audit Logging

🔹 HLR0026 / 0027 – Audit logs

Layer	Mapping
FRD	Implicit logging behavior
API	N/A (middleware / hooks)
Module	`audit`

📌 Triggered by:

- document upload
- work item status change
- entity creation

7️⃣ Overview (Counts Only)

🔹 HLR0028 / 0029 – Workspace overview

Layer	Mapping
FRD	Overview Aggregation
API	`GET /overview`
Module	`overview` (or `workspace`)

Module responsibility

- Aggregate counts
- No business logic

Why this mapping is IMPORTANT

This proves:

- You didn't "randomly write APIs"
- Every endpoint exists **because of a requirement**
- Every module has **clear ownership**
- RBAC & audit are **cross-cutting**, not scattered

This is **senior-level structure**.

What this means practically

You can now code like this:

> "I am implementing FRD0007 → `/documents` → `document` module"

No confusion. No wandering.

Final checkpoint (facts)

- ✅ You now have full traceability
- ✅ You can justify every endpoint
- ✅ You can explain architecture clearly
- ✅ You are ready to start