Medication Side Effects Module

High-Level Design

SolDevelo

Platform	*OpenMRS 3.x (O3)*
Document Type	*High-Level Design (HLD)*
Version	*1.2*
Status	*For Review*
Date	*July 2026*

Table of contents

1. Introduction & Purpose

This High-Level Design document describes the architecture of **openmrs-module-medicationmetadata**, a new self-contained OpenMRS module that brings structured medication side-effect data into the O3 platform without modifying any core OpenMRS tables. Side-effect data will be exposed exclusively as a FHIR R5 ClinicalUseDefinition resource (type: undesirable-effect); the O3 frontends consume this FHIR endpoint directly. Records are authored as versioned CSV files loaded by the OpenMRS Initializer at startup - the only write path.

The module brings structured adverse-effect data into the clinical workflow - visible to clinicians during prescribing and to pharmacists during dispensing. Side effects are linked to the **Drug (drug.drug_id)**, not to the Concept (concept.concept_id).

1.1 Problem Statement

Currently OpenMRS lacks a structured, FHIR-aligned mechanism to store and surface medication side effects. Clinical teams depend on unstructured notes or external references, leading to:

- Inconsistent counselling at the point of care.
- No programmatic visibility of adverse effects during order entry.
- No dispenser-facing risk information that can be toggled per deployment context.
- No single maintainable source of truth owned by the pharmacy team.

1.2 Goals

- Provide a clean, dedicated module that does not modify core OpenMRS Drug or Concept tables.
- Link side-effect data at the Drug (drug.drug_id) level.
- Expose side-effect data as a FHIR R5 ClinicalUseDefinition resource (type: undesirable-effect) - the single read API, consumed directly by the O3 frontends. This requires enabling R5 in openmrs-module-fhir2 (see 4.5).
- Classify each side effect as Common or Serious and carry a recommended patient-management (counselling) action (usually for serious effects)
- Surface side-effect information in the Patient Chart Order Form (clinician) and the Dispensing Workspace (pharmacist).

- Allow per-app visibility control of the side-effects panel - in both the Patient Chart Order Form and the Dispensing Workspace - via an O3 config flag (displaySideEffects, read with useConfig; default true).
- Add a new Initializer domain (medicationsideeffects) to openmrs-module-initializer so side-effect data is managed as versioned configuration files.

2. Stakeholders & Roles

Role	System	Responsibilities
Patient	Dispensing App ESM / Printed Prescription	Receives verbal counselling on side effects from the dispenser; may receive a printed prescription including side effects.
Clinician	Patient Chart ESM	Views drug side effects during order entry; uses info for shared decision-making.
Dispenser / Pharmacist	Dispensing App ESM	Reviews side effects before dispensing and counsels patients on Common and Serious effects. Visibility gated by O3 configuration.
Pharma Team	External	Primary source of truth for side-effect data. Reviews and approves the *configuration/medication sideeffects/* CSV files before deployment.
OpenMRS Developer	Module / API / Deployment	Maintains the backend module, data model, and the FHIR R5 ClinicalUseDefinition provider. Maintains the

Role	System	Responsibilities
		configuration*/medication sideeffects/* CSV files in version control and deploys them to target environments.

3. System Architecture Overview

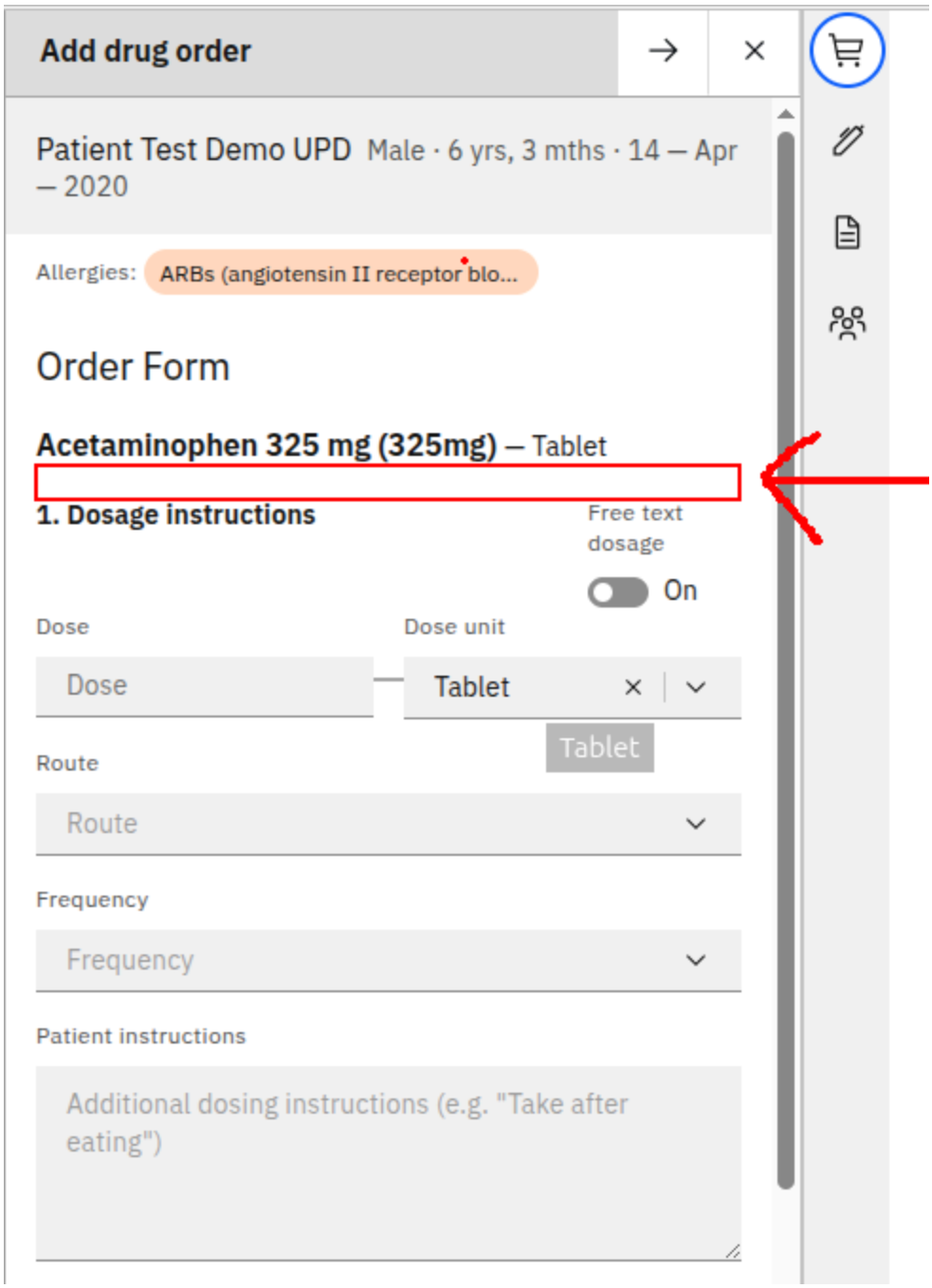
3.1 Architectural Principles

- **Separation of Concerns:** Side-effect metadata lives in its own module, leaving core OpenMRS tables unchanged.
- **FHIR Alignment:** Side-effect data is exposed solely as a FHIR R5 ClinicalUseDefinition resource; the frontends consume FHIR directly. This requires adding R5 support to openmrs-module-fhir2 (see 4.5).
- **Micro-frontend Compatibility:** Frontend integrations follow O3 ESM conventions. The patient-chart Order Form uses an ESM extension slot (order-form-side-effects-slot); the dispensing view renders the panel directly in the Prescription Details component. Both are gated by config.
- **Configuration-Driven Data:** Side-effect records are managed as versioned CSV files loaded by the Initializer module at startup, rather than entered via UI. The FHIR endpoint is read-only.
- **Configuration-Driven Visibility:** A per-app O3 config flag (displaySideEffects, read via useConfig) controls whether each surface — the Order Form and the Dispensing view — shows the side-effects panel. No backend Global Property is used.

3.2 High-Level Components

3.2.1 Frontend - openmrs-esm-patient-chart

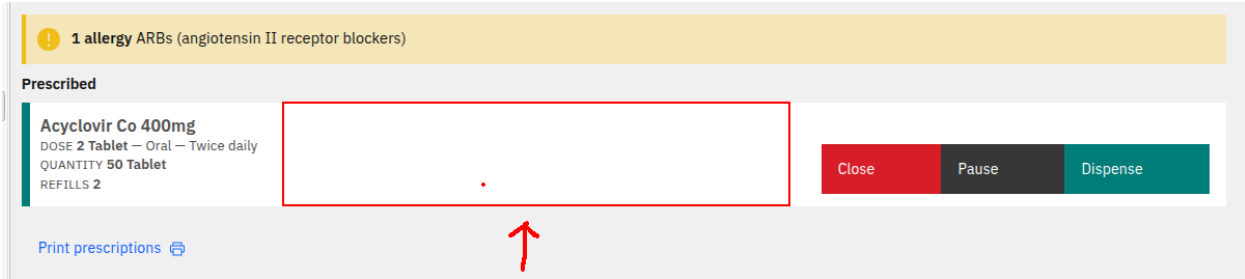
A NEW extension slot in the drug Order Form workspace renders a side-effects panel for the selected drug, fed by the FHIR endpoint and gated by the app's displaySideEffects config (the slot does not exist today and will be created — see 5.1)



The screenshot shows the current O3 Order Form for drug ordering. The red rectangle marks the proposed location for the Side Effects panel.

3.2.2 Frontend - openmrs-esm-dispensing-app

A NEW side-effects panel is rendered directly in the Prescription Details view (no new extension slot), gated by config.



The screenshot shows the current O3 Dispensing App, prescription details view. The red rectangle marks the proposed location for the Side Effects panel.

3.2.3 Configuration - openmrs-module-initializer

Side-effect data is loaded at startup via the OpenMRS Initializer module. A new medicationsideeffects domain is added directly to openmrs-module-initializer - a new MEDICATION_SIDE_EFFECTS value in the Domain enum plus a BaseCsvLoader + CsvParser + line-processor - following the pattern Initializer uses for other add-on modules (billing/cashier, queue). The loader classes are gated with @OpenmrsProfile(modules={"medicationmetadata:..."}) and Initializer takes a provided-scope dependency on medicationmetadata-api, so the domain is active precisely when openmrs-module-medicationmetadata is installed. openmrs-module-medicationmetadata carries no loader and does not depend on Initializer.

Configuration files are maintained in version control and placed in the standard Initializer configuration directory:

```
None
configuration/
└── medicationsideeffects/
    └── medication_side_effects.csv
```

Each row in the CSV represents one side-effect record. The supported columns are:

Column	Required	Description
Uuid	No	If omitted, a new UUID is generated on first load. If provided, Initializer updates the existing record with that UUID.
Void/Retire	No	Set to true to soft-delete the record. When true, parsing of the remaining columns is skipped - only Uuid is needed alongside this flag.
Drug Uuid	Yes	UUID of the target OpenMRS Drug entity.
Classification	Yes	One of: common, serious.
Side Effect Concept Uuid	Conditional	UUID of the OpenMRS Concept for the effect, when the effect is represented as a coded Concept.
Side Effect Text	Conditional	Free-text description of the effect, when the effect is not coded.
Recommended Action	No	Recommended patient-management / counselling action to take if the effect occurs (typically relevant for serious effects).
Notes	No	Free-text clinical notes

3.2.4 Backend/Data

Layer	Component	Description
Backend	openmrs-module-medicationmetadata	Owns all business logic and data access. Contributes a ClinicalUseDefinition (R5) ResourceProvider to openmrs-module-fhir2 (the single read API). No Initializer loader here - see the initializer row. No OpenMRS REST resource is exposed.
Backend	openmrs-module-initializer	Defines and hosts the medicationsideeffects domain (loader/parser/line-processor, gated by @OpenmrsProfile on medicationmetadata) and loads configuration/medicationsideeffects/ CSV files on startup into medicationmetadata's MedicationSideEffectService.
Backend	openmrs-module-fhir2	Hosts the FHIR API. R5 support (context, servlet, routing, @R5Provider annotation) is added here so the ClinicalUseDefinition ResourceProvider can be registered. No side-effect business logic lives here.
Backend	OpenMRS Core (Drug entity)	Read-only reference. Drug records are not modified.

Layer	Component	Description
		Drugs are exposed in FHIR as Medication.
Data	medication_side_effect	Single table linking Drug records to side effects (classification, coded/free-text effect, action, audit).
Data	O3 app config (displaySideEffects)	Per-app boolean (config-schema, useConfig) in BOTH the patient-chart medications app and the dispensing app; controls whether each shows the side-effects panel. Default true. No backend Global Property.

4. Backend Module Design

4.1 Module

Module Id	*openmrs-module-medicationmetadata*

4.2 Data Model

4.2.1 Table: **medication_side_effect**

A single table linking a Drug record to its known side effects, including coding and audit fields.

Column	Constraints	Description
medication_side_effect_id	PK, AUTO_INCREMENT	Primary key.
drug_id	FK → drug.drug_id, NOT NULL	Links to the OpenMRS Drug entity.
classification	NOT NULL, VARCHAR(50)	DB values are the enum names COMMON/SERIOUS. Common = acceptable effect to mention to the patient; Serious = may require action. Drives grouping and the colour-coded badge in the UI.
side_effect_concept_id	FK → concept.concept_id, NULLABLE	Coded representation of the effect, when it maps to an OpenMRS Concept.
side_effect_text	NULLABLE (TEXT)	Free-text representation of the effect, when it is not coded.

Column	Constraints	Description
recommended_action	NULLABLE (TEXT)	Recommended patient-management / counselling action to take if the effect occurs (typically relevant for serious effects).
notes	NULLABLE (TEXT)	Optional general clinical notes.
creator	FK → users, NOT NULL	OpenMRS audit: record creator.
date_created	NOT NULL (datetime)	OpenMRS audit: creation timestamp.
voided	DEFAULT 0, NOT NULL	Soft-delete flag.
void_reason	VARCHAR(255)	Reason for soft-delete.
voided_by	FK → users, NULLABLE	OpenMRS audit: user who voided the record.
date_voided	NULLABLE (datetime)	OpenMRS audit: timestamp of soft-delete.
changed_by	FK → users, NULLABLE	User who last changed the record
date_changed	NULLABLE (datetime)	Timestamp of last change
uuid	UNIQUE, NOT NULL	UUID.

Constraint: CHECK (side_effect_concept_id IS NOT NULL OR side_effect_text IS NOT NULL). Index on drug_id supports the primary query (all effects for a drug).

4.3 Configuration - Side-Effects Visibility

Panel visibility is controlled generically per app via an O3 configuration flag, following the standard config-schema / useConfig pattern. Both consuming apps declare the same key - displaySideEffects (boolean) - in their own config-schema, so each surface

can be toggled independently by the implementation. When the flag is false, the extension renders nothing and makes no FHIR call. No backend Global Property is used; there is no separate system setting round-trip.

Config key	displaySideEffects (per app)
Declared in	openmrs-esm-patient-medications-app (Order Form) and openmrs-esm-dispensing-app config-schema
Type / Default	Boolean, default true (in both apps)
Read via	useConfig
Effect	When true, the app shows the side-effects panel; when false, it renders nothing and makes no FHIR call.

4.4 API Design — FHIR R5 (ClinicalUseDefinition)

4.4.1 Endpoint & query

Method	Endpoint	Access	Description
GET	/fhir2/R5/ClinicalUseDefinition?type=undesirable-effect&subject=Medication/{uuid}	Authenticated	Bundle of ClinicalUseDefinition (type: undesirable-effect), one per side-effect record for the drug.
GET	/fhir2/R5/ClinicalUseDefinition/{id}	Authenticated	A single ClinicalUseDefinition by id.

4.4.2 Field mapping: medication_side_effect → ClinicalUseDefinition

Stored field	ClinicalUseDefinition (R5) path	Notes
(fixed)	type = "undesirable-effect"	Required (1..1).
drug_id	subject[0] = Reference(Medication/{drug-uuid})	fhir2 maps org.openmrs.Drug ↔ Medication
side_effect_concept_id	undesirableEffect.symptomConditionEffect.concept.coding	Coded representation → coding from the Concept
side_effect_text	undesirableEffect.symptomConditionEffect.concept.text	Free-text representation → concept.text
classification	undesirableEffect.classification (CodeableConcept)	CodeSystem https://fhir.example.org/CodeSystem/se-classification (common | serious)
recommended_action	undesirableEffect.extension[managementAction].valueString	No native field on undesirableEffect → carried as an extension (see 4.4.3)
uuid	ClinicalUseDefinition.id	Resource id

4.4.3 Coded values & the recommended-action field

Two module-specific values are exposed. Both use standard FHIR mechanisms and require NO hosted artifacts and NO profile:

- **classification**: exposed as undesirableEffect.classification.coding, using a stable code-system URI plus the codes common | serious.

- **recommended_action**: carried as a simple extension (url + valueString) on undesirableEffect, because the R5 resource has no native management field there. The extension url is likewise just a stable identifier and works without publishing a StructureDefinition

4.5 FHIR - Enabling R5 ClinicalUseDefinition (technical)

Openmrs-module-fhir2 today ships DSTU3 + R4 structures only (HAPI FHIR 5.7.9); the ForwardingFilter and FhirVersionUtils route only R3 and R4, so /ws/fhir2/R5/... currently returns 404, and there is no @R5Provider annotation. Because the frontends now depend on the FHIR endpoint, R5 enablement is on the critical path - it is a deliberate upstream contribution, not a trivial provider registration. Work items:

- in openmrs-module-fhir2: add the R5 HAPI structures dependency, an R5 FhirContext bean (forR5Cached), an @R5Provider annotation, an FhirR5RestServlet (+ config.xml servlet mapping), and R5 routing in ForwardingFilter and the FhirVersion enum in FhirVersionUtils.
- Verification / spike: confirm whether HAPI 5.7.9 provides the ClinicalUseDefinition class (5.7.x carries only a preview/ballot R5; final FHIR R5 = 5.0.0 is stable from HAPI ~6.4+). If unavailable or preview-only, plan a HAPI upgrade to 6.4+ - a broader change to fhir2 to be agreed with the community
- In openmrs-module-medicationmetadata: a ClinicalUseDefinition IResourceProvider annotated @Component + @R5Provider, registered into the fhir2 R5 servlet via Spring component-scan, and a translator mapping each record to ClinicalUseDefinition (see 4.4.2). The provider reads through the module's existing MedicationSideEffectService; no dedicated FhirService/FhirDao and no META-INF/services SPI file are used. Provider + translator live in omod (not api) so the api jar - which openmrs-module-initializer depends on - carries no HAPI/R5 code.

5. Frontend Design

5.1 Patient Chart - Order Form Extension

5.1.1 Location

Module: openmrs-esm-patient-chart
Extension Slot: A new extension slot (e.g. order-form-side-effects-slot) will be created in the drug Order Form workspace. Natural insertion point: in packages/esm-patient-medications-app/src/add-drug-order/drug-order-form.component.tsx, between the medication info header and the "Dosage instructions" section (directly below the drug selector).

5.1.2 Behavior

1. The extension reads the displaySideEffects config (useConfig, default true); if false it renders nothing and makes no FHIR call.
2. When a clinician selects a drug in the Order Form, the drug UUID is resolved.
3. The extension calls GET **/ws/fhir2/R5/ClinicalUseDefinition?type=undesirable-effect&subject=Medication/{drugUuid}.**
4. The returned Bundle is parsed: each entry is one side effect. The effect name is symptomConditionEffect.concept.coding[0].display (Common) or .concept.text (Serious); classification comes from undesirableEffect.classification; the action is read from the managementAction extension.
5. Entries render in a collapsible panel below the drug selector, grouped by classification (Common first, then Serious).
6. For any effect that has a recommended action (common or serious), the action is shown via a warning icon + tooltip next to the effect.
7. If the Bundle is empty, the panel is hidden (no empty state). If the call fails, a non-blocking inline notification warns the clinician (read-only; never blocks ordering).

5.1.3 UI Components (Carbon Design System)

- **Section label:** Known side effects
- **Tag (classification)**: magenta for Serious, teal for Common

- **Warning icon (red) + Tooltip:** showing the recommended action, next to any effect that has one
- **InlineNotification**: error/loading feedback and the counselling note.
- **SkeletonText**: loading placeholder.

5.2 Dispensing App - Dispensing Workspace Extension

5.2.1 Location

Module: openmrs-esm-dispensing-app
Integration: the panel is rendered directly in prescription-details.component.tsx (per medication, inside the MedicationEvent component). No extension slot; prescription-expanded.component.tsx is not involved.

5.2.2 Conditional Rendering Logic

Visibility is driven by the same generic O3 config flag read with useConfig (a displaySideEffects boolean in the dispensing app config-schema)

- If displaySideEffects is **false**: the extension renders nothing and makes no FHIR call.
- If displaySideEffects is **true** (default): the extension fetches from the same FHIR endpoint and renders side effects identically to the Order Form panel

5.2.3 Dispenser Panel Additions

In addition to the Order Form panel, the Dispensing view adds a counselling prompt:

- A highlighted InlineNotification: 'Counsel the patient on the above side effects' (shown below the effects). The same counselling note appears in both surfaces (Order Form and Dispensing) for consistency. The recommended action for any effect is surfaced via a warning icon + tooltip next to that effect.

6. Integration & Data Flow

6.1 End-to-End Flow: Clinician Order Entry

1. The clinician opens the New Order Form in the Patient Chart and selects a drug.
2. The extension reads displaySideEffects (useConfig, default true); if false it renders nothing and stops.
3. The Order Form ESM resolves the drug UUID.
4. The side-effects panel fires GET /ws/fhir2/R5/ClinicalUseDefinition?type=undesirable-effect&subject=Medication/{drugUuid}.
5. Fhir2 routes to the ClinicalUseDefinition provider → service → DAO, which queries medication_side_effect WHERE drug_id = {id} AND voided = 0.
6. The translator maps each record to a ClinicalUseDefinition; HAPI returns a Bundle.
7. The frontend groups entries by classification and shows the recommended action for serious effects.
8. The clinician proceeds to submit the order (the display is read-only and non-blocking).

6.2 End-to-End Flow: Pharmacist Dispensing

1. Pharmacist opens a prescription in the Dispensing App workspace.
2. The dispensing extension checks the displaySideEffects config (useConfig, default true).
3. If true: fetches from the FHIR endpoint and renders the counselling panel within the Prescription Details view.
4. If false: the extension renders nothing; no FHIR call is made.
5. The pharmacist reviews side effects, counsels the patient, and completes the encounter.

6.3 Initializer Load Flow (the only write path)

1. The OpenMRS Developer updates the *configuration/medicationsideeffects/* CSV file in version control.

2. The updated file is deployed to the OpenMRS configuration directory on the target server.
3. On the next backend startup, invokes the medicationsideeffects domain loader (defined in openmrs-module-initializer, active because openmrs-module-medicationmetadata is installed).
4. Each row is processed per Initializer conventions: rows with a provided uuid update existing records; rows without a uuid create new records; rows with void/retire = true retire the matching record. Rows with unresolved references fail that row (with retry and an error summary) while other rows still load.
5. Initializer writes a new checksum for the processed file. The file will not be reprocessed until its content changes again.

7. Security & Access Control

Access control follows the OpenMRS privilege framework. The module defines the following privileges. Role-to-privilege assignment is performed per deployment by the System Administrator.

Privilege	Permitted Operations
Get Medication Side Effects	Read side-effect records via the FHIR ClinicalUseDefinition endpoint.
Manage Medication Side Effects	Create/update/void side-effect records; used by the Initializer write path

All API endpoints enforce authentication via OpenMRS session tokens. Write operations are performed exclusively by the Initializer module at startup and are not exposed via any API (the FHIR endpoint is read-only).

8. Risks

Risk	Likelihood	Mitigation
FHIR R5 enablement requires upstream changes to openmrs-module-fhir2 (possibly a HAPI upgrade), and it is on the critical path because the UI depends on it.	High	Treat R5 enablement as a scoped upstream contribution (context/servlet/routing/@R5Provider); run a spike to confirm ClinicalUseDefinition availability in HAPI 5.7.9 vs. upgrading to 6.4+. Coordinate with the OpenMRS community and sequence it before the frontend work.
New frontend extension slots must be created	Low	Create order-form-side-effects-slot in patient-chart; in dispensing-app render the panel directly in the Prescription Details view (per medication), gated by config - no custom tab.
Concept UUID resolution fails at render time, leaving side effect names blank in the UI.	Low	The frontend should handle concept API failures gracefully, falling back to displaying the raw UUID with an inline warning rather than an empty or broken panel.
Classification is inconsistent across CSV files authored by different contributors.	Low	classification is a fixed ENUM (common, serious) backed by the se-classification CodeSystem; the Pharma Team agrees classifications before

Risk	Likelihood	Mitigation
		authoring configuration files.
Query performance degrades as the dataset grows.	Low	Index on drug_id. The dataset is small (order of a few hundred drugs), so performance is not a concern.