**LicenseIQ Development Plan: Sprint-0 and Starter Guide**

This document outlines the Sprint-0 exit targets, key steps, domain models, service architecture, implementation priorities, and starter artifacts for building LicenseIQ—a compliance tool for license determination, sanctions screening, and auditability. The plan emphasizes starting with API contracts, domain schemas, and core decision logic, while baking in governance, security, and observability from day one. All elements align with board-ready requirements and comprehensive expert guidance.

**0) Sprint-0 Exit Targets (What “Ready” Means)**

* OpenAPI v1 published with mockable examples; contract tests pass.
* SLOs & guards in place (performance, availability), CI/CD + IaC standing up Dev/Stage, blue-green/canary demonstrable.
* SSO/MFA + RBAC/ABAC enforced on mock endpoints; residency/retention policies codified.

**1) Start with the API Contract (Day 0–1)**

Create OpenAPI v1 with these core endpoints and shared error schema. This unblocks frontend development and locks your domain language.

**Must-Have Endpoints**

* POST /determine → returns ALLOW/REVIEW/BLOCK + WHY/FIX + audit\_id
* POST /simulate → what-if on historic orders (no side effects)
* GET/POST /rules (+ /rules/{id}/deploy) → CRUD + lifecycle
* GET/POST /masterdata/products and /masterdata/parties
* GET /sanctions/sources, POST /sanctions/refresh
* POST /documents (upload), GET /documents/{id} (versions/metadata)
* POST /webhooks/subscriptions (+ outbound events)
* POST /integrations/sap/so (adapter stub)

**Principles & Guards**

* /api/v1 base
* OAuth2/OIDC
* Idempotency-Key on POST
* 600 rpm/tenant default
* limit/offset pagination
* Standard error with trace\_id

**2) Freeze the Initial Domain Models (Day 1)**

Define schemas your engine and UI depend on.

**Products/Materials (Minimum)**

* material\_id
* sku
* description
* uom
* category
* hs\_code
* eccn
* dual\_use\_flag
* origin\_country
* bom\_pct
* licensing\_notes
* controlled\_reasons

**Parties/Customers (Minimum)**

* party\_id
* external\_ids {duns, lei}
* tax\_ids
* names/addresses
* country
* region
* party\_type
* screening\_status
* risk\_score
* last\_reviewed\_at
* kyc\_docs

**Rules/Quotas**

* Rule (id, condition, action, priority)
* LicenseType (id, code, jurisdiction, validity\_period)
* Quota (id, license\_id, remaining\_qty, reset\_date)

**Documents/OCR**

* Document (id, type, version, status, upload\_date, expiry\_date)
* TransactionLink (document\_id, order\_id, license\_id)
* Metadata (document\_id, tag\_key, tag\_value)

**3) Lay Down the Repository & Services (Day 1–2)**

Service layout (monorepo or polyrepo):

* **gateway**: auth, rate limit, request logging
* **determination**: rules engine, de-minimis, quotas
* **rules**: CRUD, lifecycle (draft → shadow → canary → prod)
* **masterdata**: products, parties
* **sanctions**: sources, sync jobs, freshness timestamps
* **documents**: uploads, OCR pipeline, versions
* **audit**: immutable events + evidence pack generator
* **integrations**: SAP/SFDC adapters, webhooks

Back each service with:

* PostgreSQL (relational)
* Object storage for docs
* Redis for hot caches (quotas, sanctions hits)

**4) Implement the “Core Decision Loop” First (Days 2–5)**

**a) Determination API (Sync)**

Flow: Trigger → Pull master data → Apply rules → De-minimis → Check quotas → Decide → WHY/FIX → Audit.

Response must include: decision, why, fix, audit\_id for auditability from day one.

**b) Rules Evaluator**

Start with a few high-value rules (ECCN × destination, SPL hit, de-minimis %, quota >80% alert). Keep rules deterministic and traceable.

**c) Audit Events**

Every /determine writes an immutable event (who/when/why, rule IDs, sources).

**5) Hardcode Governance from the Start (Days 4–6)**

* Lifecycle: draft → shadow test (mirrored, read-only) → canary (≤10%) → prod; one-click rollback; 4-eyes approval.
* Simulation: Before enabling risky rules, simulate on last 90 days.

**6) Security, Privacy, and Tenancy (Days 2–6; Keep Iterating)**

* SSO (OIDC), MFA
* Roles: Admin, Compliance Manager, Operator, Auditor, Integration
* ABAC: region/BU/customer-group
* TLS 1.3, AES-256-GCM at rest
* Central vault + 90-day key rotation
* Per-tenant keys & rate limits
* Residency pinning (EU ↔ EU, US ↔ US)
* Ops retention: 7 years / logs: 13 months

**7) DevOps & Observability (Days 1–3 for Scaffolding)**

* CI/CD with tests, lint, SAST/DAST, SBOM
* IaC (Terraform + Helm)
* Dev/Stage/Prod envs
* Blue-green/canary with SLO-based rollback
* Metrics/logs/traces dashboards per service
* Error-budget alerts

**8) Concrete Starter Artifacts You Can Create Today**

**A) Minimal OpenAPI v1 (Excerpt)**

yaml

openapi: 3.0.3

info: { title: LicenseIQ API, version: "1.0.0" }

servers: [{ url: https://api-staging.licenseiq.com/api/v1 }]

paths:

/determine:

post:

summary: Evaluate order lines

parameters: [{ in: header, name: Idempotency-Key, schema: { type: string } }]

responses:

"200":

description: Decision

content:

application/json:

schema:

type: object

properties:

decision: { type: string, enum: [ALLOW, REVIEW, BLOCK] }

why: { type: array, items: { type: string } }

fix: { type: array, items: { type: string } }

audit\_id: { type: string, format: uuid }

"400": { $ref: "#/components/responses/Error" }

components:

responses:

Error:

description: Standard error

content:

application/json:

schema:

type: object

properties:

error:

type: object

properties:

code: { type: string }

message: { type: string }

details: { type: array, items: { type: object } }

trace\_id: { type: string, format: uuid }

**B) First SQL Tables (PostgreSQL)**

sql

-- products

create table product (

material\_id uuid primary key,

sku text not null,

description text,

uom text,

category text,

hs\_code text,

eccn text,

dual\_use\_flag boolean default false,

origin\_country char(2),

bom\_pct numeric(5,2),

licensing\_notes text,

controlled\_reasons text[]

);

-- parties

create table party (

party\_id uuid primary key,

duns text, lei text, tax\_ids jsonb,

name text not null,

address jsonb, country char(2), region text,

party\_type text, screening\_status text,

risk\_score numeric(5,2), last\_reviewed\_at timestamptz,

kyc\_docs jsonb

);

-- rules (simplified)

create table rule (

id uuid primary key,

condition jsonb not null, -- e.g., {"eccn":"5A002","dest":"CN"}

action jsonb not null, -- e.g., {"require\_license":"BIS-XYZ"}

priority int not null default 100,

status text not null default 'draft' -- draft/shadow/canary/prod

);

-- audit

create table audit\_event (

id uuid primary key,

audit\_id uuid not null,

occurred\_at timestamptz not null default now(),

actor text, tenant\_id uuid,

event\_type text, -- determination.created, rule.deployed, etc.

payload jsonb -- full trace incl. rules fired (WHY) and FIX tips

);

**C) “Happy Path” Determine Handler (Pseudo)**

python

def determine(order):

md = masterdata.enrich(order) # products+parties

ctx = RuleContext(md)

fired = rules.evaluate(ctx) # returns list of rule ids

decision, fixes = decide(ctx) # ALLOW/REVIEW/BLOCK + FIX list

audit\_id = audit.write("determination.created", order, decision, fired, fixes)

return { "decision": decision, "why": fired, "fix": fixes, "audit\_id": audit\_id }

**9) 7-Day Starter Schedule (Focused Only on Back-End)**

* **Day 1**: OpenAPI v1 + examples; mock server on CI; repo + service skeletons.
* **Day 2**: DB migrations for Products/Parties/Rules/Audit; OAuth/OIDC guard at gateway.
* **Day 3**: Implement /determine (happy path) + audit write; unit tests with golden fixtures.
* **Day 4**: /rules CRUD + /simulate; start shadow test scaffolding.
* **Day 5**: masterdata CRUD + sanctions/sources freshness endpoint.
* **Day 6**: CI/CD (signed images, SBOM), Terraform+Helm for Dev/Stage, SLO dashboards.
* **Day 7**: Canary deploy of determination service; rollback drill; DPIA checklist started.

**Bottom Line**

Start with OpenAPI v1 + domain schemas, then implement the core decision loop with auditability and governance baked in. Everything above maps directly to board-ready requirements and comprehensive guidelines, so Engineering can begin immediately with zero ambiguity.

s