EHR + Smart Contracts – Patient Consent Registry

Initiative end-to-end development map

Top-Level Workstreams

1) Smart Contracts (EVM / Solidity)

Purpose: On-chain source of truth for consent, identities, and audit proofs (no PHI on-chain).

Modules

- **ConsentRegistry** (already drafted): grant/update/revoke consent; events for audits.
- **ProviderIdentityRegistry**: verifies clinics/providers (NPI/PKI/DID → EVM address), status (active/suspended).
- **HospitalRegistry**: registers hospital EHR gateways, org identifiers, public keys, policies.
- **AccessEventAnchor** (optional): emits hashed access receipts (off-chain log → on-chain anchor for integrity).

Key Artifacts

- Solidity contracts + Hardhat/Foundry tests
- ABI + TypeChain types for SDKs
- Contract docs (NatSpec), threat model, gas profile

Acceptance

- Reproducible deployments to **local**, **testnet**, **permissioned net (Besu/Quorum)**.
- Full event coverage and revert paths; 95%+ unit test coverage.

---

2) Off-Chain Services (Gateways & Indexers)

Purpose: Bridge chain ↔ EHR (FHIR), enforce policy off-chain, keep PHI off-chain.

Services

- **Blockchain Gateway** (Node.js or Python): validates consent against chain, signs/verifies EIP-712 requests, calls FHIR.
- **Consent Indexer**: subscribes to contract events, builds a queryable cache (Postgres/Elastic) for portals & reporting.
- **Audit Pipeline**: streams access events to SIEM; batches periodic **on-chain anchors** (Merkle root) for tamper-evidence.
- **Notification Service**: pushes consent requests/approvals to patients & clinics (email/SMS/push/Webhooks).

APIs (sample)

- POST /consent/requests (clinic → patient): create signed access request
- POST /consent/approve (patient): triggers on-chain grant
- GET /consent/state?patient=...&requester=...
- POST /ehr/access (clinic): consent check → proxy FHIR query
- POST /audit/anchor (ops): publish batch hash to chain

Acceptance

- End-to-end consent check latency < 400ms (cached).
- FHIR calls blocked if consent expired/revoked.
- Signed request verification required; full trace in logs.

---

3) EHR Connectors (FHIR)

Purpose: Vendor-agnostic adapters that translate allowed scopes → FHIR queries.

Adapters

- **HAPI FHIR** dev adapter (reference)
- **Epic/Cerner connectors** (as available), with scoped resource mapping: Patient/*.read, Observation.read, MedicationRequest.read, etc.

Acceptance

- Scope → query mapping table
- Pagination, throttling, and error normalization

- Synthetic data harness for testing (no PHI)

---

4) Frontends (Portals & dApps)

Purpose: Human-friendly UIs for each role.

Apps

- **Patient Portal** (web/mobile): approve/revoke, time windows, scope selection, history, notifications.
- **Clinic Console**: request access, view status, download records, renewals, org settings.
- **Hospital Admin**: onboard providers, manage policies/keys, incident/audit views.
- **Research Workspace** (phase 2): cohort consent, anonymized exports.

Acceptance

- Accessibility (WCAG 2.1 AA), audit trails surfaced, no PHI in logs.
- Simple “Invite Provider” (email/QR) flow works cross-org.

---

5) Identity, Security & Keys

Purpose: Bind real-world entities to cryptographic identities.

Components

- **OIDC/SAML** for user login (patients/clinicians).
- **DID/PKI binding** for clinics/hospitals → **EVM addresses** in registries.
- **KMS/HSM** for server keys; wallet custody options (custodial for clinics, app-embedded for patients).
- **Policy Engine**: timeboxing, audience, purpose-of-use, jurisdiction flags.

Acceptance

- Rotating keys w/ audit; compromise playbook.
- mTLS across internal services; secrets in vault.

---

6) SDKs & Developer Experience

Purpose: Make integration easy for partners and future teams.

Deliverables

- **JS/TS SDK** (@health-consent/sdk): contract calls, EIP-712 helpers, gateway client.
- **Python SDK** (for data teams).
- **CLI**: bootstrap orgs, simulate requests, tail events.
- **Postman collection** / OpenAPI spec.

Acceptance

- Quickstarts that run green in <10 minutes (local stack).
- Versioned APIs and semver packages.

---

7) Compliance, Governance & Policy

Purpose: Operate safely under HIPAA/GDPR and org policies.

Artifacts

- **Data flow diagrams** (PHI boundaries), RoPA (record of processing activities).
- **BAA** templates, consent language, retention schedules.
- **Incident response & breach notification** runbook.
- **PIA/TRA** (privacy/threat risk assessments).

Acceptance

- External audit checklist ready; evidence in repo.
- Logging/monitoring meets compliance (no PHI leakage).

---

8) Observability, SRE & DevOps

Purpose: Reliable, repeatable environments.

Deliverables

- IaC (Terraform) for **dev/test/stage/prod** (+ permissioned chain if used).
- CI/CD (GitHub Actions) with contract deploy gates & migration safety.

- Metrics (latency, error rates), logs, traces; on-call playbooks.
- Canary releases, feature flags.

Acceptance

- Blue/green deploys; rollback < 5 minutes.
- SLOs: 99.9% for gateway; alerts wired.

---

9) QA & Test Harness

Purpose: Prove correctness and safety end-to-end.

Layers

- Contract unit & property tests.
- Integration tests: consent → FHIR response paths (happy/sad).
- Red-team tests (mis-scoped requests, replay, key rotation).
- Load & soak tests; chaos (gateway failover).

Acceptance

- Test matrix automated in CI; synthetic data only.

Suggested Repo / Package Layout (poly-repo or mono with workspaces)

```
/contracts
 /consent-registry
 /provider-identity
 /hospital-registry
 /access-anchor
 hardhat.config.ts
 README.md

/services
 /gateway-api          # Node/Python; REST; EIP-712; FHIR proxy
 /consent-indexer       # Event subscribers → Postgres/Elastic
 /audit-pipeline        # SIEM + anchor batching
 /notification-service

/sdk
 /js
 /python

/frontends
 /patient-portal        # React/Next + OIDC; wallet-lite
 /clinic-console
 /hospital-admin
 shared-ui             # design system

/connectors
 /fhir-hapi-adapter
 /fhir-epic-adapter
 /fhir-cerner-adapter


/devops
 /terraform
 /helm
 /github-actions

/docs
 /architecture
 /openapi
 /threat-model
 /runbooks
```

Minimal API & Contract Shapes (for planning)

ConsentRegistry.sol (events)

- event ConsentGranted(bytes32 consentId, address patient, address requester, uint64 start, uint64 end)
- event ConsentRevoked(bytes32 consentId, address patient, address requester, uint64 revokedAt)
- function isConsentActive(bytes32) view returns (bool)

Gateway REST (OpenAPI)

- POST /consent/requests → { patientId, requesterId, purpose, scopes[], duration }
- POST /consent/approve → { consentId, signature? } → writes on-chain
- GET /consent/state → { active, start, end, scope }
- POST /ehr/access → { consentId, fhirQuery } → proxied response (PHI)

Phased Delivery Plan

Phase 0 — Foundations (2–3 sprints)

- Contracts v1 + tests
- Gateway skeleton + HAPI FHIR adapter
- Patient Portal MVP (grant/revoke)
- JS SDK alpha, local dev cluster (docker-compose)

Phase 1 — Pilot (3–5 sprints)

- ProviderIdentity/HospitalRegistry
- Clinic Console MVP + signed requests
- Indexer + basic audit dashboard
- Synthetic data E2E + load test
- Compliance docs v1 (PIA/TRA, data flows)

Phase 2 — Hardening (3–4 sprints)

- SIEM integration + anchor batching

- Key rotation, EIP-712 everywhere
- Epic/Cerner adapter
- SLOs, canary deploys, chaos tests

Phase 3 — Expansion

- Research workspace
- Patient data-sharing incentives (tokenless or credits)
- Consent templates & multi-org workflows

Initial Backlog (concrete, buildable next steps)

1. Contracts: split **ConsentRegistry** into library + main; add pause/upgrade pattern (proxy optional).
2. Gateway: /ehr/access happy path with **HAPI FHIR** and **isConsentActive** check.
3. Patient Portal: Approve/Revoke UI with event-driven status.
4. ProviderIdentityRegistry: minimal allowlist (NPI + domain) + admin UI in **Hospital Admin**.
5. Indexer: ConsentGranted/Revoked → Postgres; /consents?patient=... API for UIs.
6. Docs: OpenAPI for gateway, README quickstart, threat model v0.1.
