**Comprehensive Project Documentation: UAMP Built on DALRN**

**1. Project Overview and Vision**

**Project Title**: Universal Agentic Mediation Protocol (UAMP) Built on Decentralized Autonomous Legal Resolution Network (DALRN)

**Vision**: UAMP is a modular, decentralized AI platform for resolving high-stakes disputes across industries, with DALRN as its legal-specific core. It fuses the five primary research topics to enable secure, autonomous mediation without central control, addressing gaps in existing platforms like SAKSHI (arXiv:2307.16562, 2023) or Kleros (ResearchGate, 2022) by integrating self-organizing agents, homomorphic processing, quantum-inspired search, federated learning, and adversarial negotiation [cite: McMahan et al., 2017, "Communication-efficient learning of deep networks from decentralized data," AISTATS, foundational for federated aspects; Nash, 1951, "Non-cooperative games," Annals of Mathematics, for adversarial equilibria; continuity with our prior UAMP as DALRN generalization].

**Objectives**:

* Deliver a privacy-preserving system starting with DALRN for legal disputes (e.g., small claims arbitration) and extending to UAMP for healthcare, finance, supply chain, etc.
* Demonstrate **empirical** retrieval and latency gains over a strong FAISS baseline using a **quantum‑inspired reweighting** reranker (A/B‑tested). Do **not** claim theoretical speedups on classical hardware. Compute Pareto‑efficient and/or **stable** outcomes with transparent selection rules (e.g., Nash social welfare or egalitarian).
* Ensure compliance with GDPR, HIPAA, and EU AI Act via differential privacy and audit trails [cite: Abadi et al., 2016, "Deep learning with differential privacy," CCS; EU AI Act draft, 2024].
* Hypothetically reduce dispute resolution time by 30-50% across industries, based on AI mediation pilots [speculative scenario, grounded in PMC, 2021, reporting efficiency gains in ODR; requires verification from user pilots].

**Scope**:

* In Scope: DALRN core for legal mediation; UAMP adapters for generalization; integration of five topics; no-code UI, AI modules, blockchain settlement.
* Out of Scope: Real quantum hardware (use simulations); full mobile app (web-first); international legal certification (focus on tech compliance).

**Assumptions and Constraints**:

* Development leverages Claude Code for generation, GitHub for CI/CD, AWS for deployment
* Access to libraries like NetworkX, TenSEAL, Qiskit, PySyft, Nashpy (pre-imported in code\_execution environments) [cite: Python 3.12 docs, 2024].
* Constraints: No internet in code\_execution; focus on Python/JS stack.

**Stakeholders**:

* Solo Entrepreneur: Product Owner.
* Developers/Team: Builders using Claude-generated code.
* End Users: Enterprises in legal, healthcare, etc.
* Regulators: Compliance bodies (e.g., for GDPR audits).

**Success Metrics**:

* Technical:  **Encrypted similarity** parity with plaintext (recall@k within ±2% on benchmark corpus); **Search** P95 ≤ 600 ms (defined hardware, dataset size); **SLOs**: per‑service availability targets (e.g., API 99.5%, search 99.5%), excluding planned FL/crypto key maintenance windows
* Business: 100 users in MVP pilot; 20% cost savings in disputes (hypothetical, per McKinsey 2023 AI reports).
* Innovation: Patentable fusion of five topics, validated against gaps in arXiv:2307.16562 (2023).

**2. System Architecture**

**High-Level Architecture**: UAMP extends DALRN as a layered microservices system, with DALRN providing legal-specific workflows and UAMP adding modular adapters for cross-industry use. The architecture ensures decentralized execution, privacy, and scalability [cite: Newman, 2015, "Building Microservices," O'Reilly, for modular design; AWS Well-Architected Framework, 2023, for cloud alignment].

**Text-Based Diagram (Mermaid Syntax for Rendering in Word via Plugins or Online Tools)**:

UAMP Industry Extensions

DALRN Legal Core

User Interface Layer: No-Code Dashboard

Self-Organizing Router: Dynamic Agent Assembly (Topic 1)

Encrypted Data Layer: Homomorphic Processing (Topic 2)

Quantum-Inspired Search Engine: Precedent Matching (Topic 3)

Federated Learning Hub: Privacy-Preserving Training (Topic 4)

Adversarial Negotiation Pool: Game-Theoretic Resolution (Topic 5)

Blockchain Settlement Layer: Immutable Enforcement

Adapters: e.g., Healthcare (HIPAA Module), Finance (Fraud Patterns)

Monitoring & Compliance: Prometheus, Audit Logs

A diagram of a process

AI-generated content may be incorrect.

**Detailed Component Descriptions**:

* **User Interface Layer**: React/Next.js frontend; supports drag-and-drop dispute submission [cite: React v18 docs, 2024].
* **Self-Organizing Router**: NetworkX graph for agent topology; asyncio for parallel task routing [cite: NetworkX v3.3, 2024; Python asyncio docs, 2024].
* **Encrypted Data Layer**: TenSEAL for CKKS operations on sensitive data [cite: Cheon et al., 2017, "Homomorphic encryption for arithmetic of approximate numbers," ASIACRYPT].
* **Quantum-Inspired Search Engine: Qiskit demo only** (education). **Production** uses FAISS (HNSW/IVF‑PQ) with an optional quantum‑inspired **reweighting** loop
* [cite: Tang, 2019, "An affirmative solution to the approximate quantum search problem," arXiv].
* **Federated Learning Hub**: PySyft for secure model aggregation with differential privacy [cite: Abadi et al., 2016, "Deep learning with differential privacy," CCS].
* **Adversarial Negotiation Pool**: Nashpy/PyTorch for RL agents computing equilibria [cite: Osborne & Rubinstein, 1994, "A Course in Game Theory," MIT Press].
* **Blockchain Settlement Layer**: Solidity smart contracts on Ethereum; IPFS for audits [cite: Buterin, 2014, Ethereum Whitepaper; IPFS docs, 2024].
* **Monitoring & Compliance**: Prometheus for metrics; VeraCode for scans [cite: Prometheus v2.48, 2024; VeraCode State of Software Security, 2024].

**Data Flow and Integration**:

* Input: Encrypted dispute data → Router assembles agents → Search retrieves matches → Federated hub adapts models → Negotiation computes outcome → Blockchain records resolution.
* APIs: RESTful with FastAPI; WebSockets for real-time tracking [cite: FastAPI docs, 2024].

**Security Architecture**:

* Zero-trust: OAuth via AWS Cognito; rate limiting with AWS WAF [cite: AWS Cognito docs, 2024].
* Privacy: **Tenant‑scoped DP budgets** tracked in an **ε‑ledger** with RDP accounting; configurable per model/version; enforce budget caps
* ; zk-proofs for verification [cite: arXiv:2405.00295v3, 2025, zkML in decentralized systems].

**Deployment Architecture**:

* AWS: ECS/Fargate for containers, Lambda for serverless, SageMaker for ML [cite: AWS ECS docs, 2024].
* CI/CD: GitHub Actions with Terraform [cite: Terraform v1.5, 2024].

**3. Product Requirements Document (PRD)**

This PRD is exhaustive, expanded to task-level detail for developers (e.g., sub-tasks under each ID) to facilitate Code generation and implementation. Structured by phase, with priorities, acceptance criteria, and DALRN topic links. Covers functional, non-functional, and task breakdowns

**PRD Version**: 1.0  
  
**Phase 1: Foundation (Months 1-2)**

Focus: Marketing site, UI, core router for DALRN legal MVP.

**I. Marketing Website**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| MKT-01 | Landing page with value prop | Task 1: Design layout in Figma; Task 2: Implement in Next.js; Task 3: Add DALRN demo button; Task 4: Optimize for SEO with meta tags. | Loads <2s; explains UAMP/DALRN in ≤30 words; passes Lighthouse score >90. | High | All |
| MKT-02 | Industry-specific pages | Task 1: Create templates for 3 industries; Task 2: Populate with case studies from DALRN simulations; Task 3: Add CTA links. | 3 studies per industry; responsive design; A/B test conversion >5%. | High | Federated (D) |
| MKT-03 | Developer portal with API docs | Task 1: Generate Swagger spec; Task 2: Build sandbox with Postman collection; Task 3: Document DALRN endpoints. | Live testing endpoint; covers all five topics; no 404 errors. | Med | Self-Organizing (A) |
| MKT-04 | Early-access signup form | Task 1: Set up PostgreSQL DB; Task 2: Implement form with React Hook Form; Task 3: Add GDPR checkbox and email confirmation. | Stores data securely; sends welcome email; compliance audit pass. | Med | Homomorphic (B) |

**II. No-Code Platform Dashboard**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| UI-01 | Dispute submission form | Task 1: Build drag-and-drop with React Dropzone; Task 2: Add dropdowns for industry/type; Task 3: Validate inputs client-side; Task 4: Encrypt uploads on submit. | Supports PDF/JPEG; DALRN defaults for legal; form submits <1s. | High | Adversarial (E) |
| UI-02 | Real-time case tracker | Task 1: Set up WebSockets with Socket.io; Task 2: Log agent steps (e.g., "Search started"); Task 3: Poll every 15s; Task 4: Handle disconnects. | Updates in real-time; shows DALRN agent logs; no lag >15s. | High | Quantum (C) |
| UI-03 | Resolution dashboard | Task 1: Display terms/scores in cards; Task 2: Add export PDF button; Task 3: Link to blockchain explorer; Task 4: Allow feedback submission. | Exports accurate PDFs; UAMP industry filters; user satisfaction >80% in tests. | High | Federated (D) |
| UI-04 | WCAG 2.1 compliance | Task 1: Run Axe audit; Task 2: Add ARIA labels; Task 3: Test with screen readers; Task 4: Ensure color contrast >4.5:1. | 0 errors in audit; passes WAVE tool; inclusive for all users. | Med | All |

**Phase 2: Privacy Layer (Months 3-4)**

Focus: Homomorphic and federated modules for UAMP privacy.

**III. Core AI Modules (Continued)** **B. Homomorphic Processing Engine**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| HE-01 | CKKS encryption | Task 1: Import TenSEAL; Task 2: Configure context (poly\_mod=8192); Task 3: Encrypt docs; Task 4: Store in S3 with metadata. | Encrypts DALRN data securely; decryption test 100% success. | High | Homomorphic (B) |
| HE-02 | Encrypted **similarity scoring** (CKKS) on unit‑norm embeddings | Normalize → encrypt embeddings client‑side; server performs CKKS dot products; decrypt & select top‑k client‑side | Encrypted **dot‑product/cosine** scoring with TenSEAL (no raw text under FHE); parity with plaintext retrieval within ±2% recall@k; client‑side decryption of scores; no plaintext leakage in transit or at rest | High | Homomorphic (B) |
| HE-03 | HIPAA/GDPR compliance | Task 1: Add consent logs; Task 2: Run VeraCode scan; Task 3: Generate audit reports; Task 4: Automate data deletion. | Passes scan; logs query history. | Med | Homomorphic (B) |

**D. Federated Learning Hub**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| FL-01 | PySyft integration | Task 1: Set up VirtualWorkers; Task 2: Aggregate from 3+ parties; Task 3: Train on DALRN models; Task 4: Add UAMP data hooks. | Aggregates without sharing data; convergence in <10 rounds. | High | Federated (D) |
| FL-02 | Differential privacy | Task 1: Implement Gaussian noise (ε=0.5); Task 2: Track budget; Task 3: Test on synthetic data; Task 4: Log privacy spends. | Privacy guarantee verified; no over-budget errors. | High | Federated (D) |
| FL-03 | Local training container | Task 1: Build Docker image; Task 2: Include PyTorch; Task 3: Run on-prem tests; Task 4: Deploy to ECS. | Image runs DALRN training; size <500MB. | Med | Federated (D) |

**Phase 3: Intelligence Layer (Months 5-6)**

Focus: Quantum and adversarial modules; e-commerce pilot.

**C. Quantum-Inspired Search**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| QS-01 | Precedent vectorization | Task 1: Use Sentence-BERT for embeddings; Task 2: Vectorize DALRN cases; Task 3: Store in vector DB (e.g., FAISS); Task 4: Add UAMP filters. | Embeddings for 10K cases; cosine similarity >0.9 for matches. | High | Quantum (C) |
| QS-02 | Grover-style search | Task 1: Simulate superposition with NumPy; Task 2: Amplify amplitudes; Task 3: Test on legal DB; Task 4: Generalize for industries. | **Benchmark target** (report corpus size, ANN index type, hardware). A/B: FAISS‑only vs FAISS+reweighting. Publish recall@{10,20}, NDCG, P95 latency. Enable reweighting **flag‑off by default** until lift is shown | High | Quantum (C) |
| QS-03 | Federated search | Task 1: Query via PySyft; Task 2: Aggregate results securely; Task 3: Test privacy; Task 4: Log queries. | No raw data access; federated accuracy >85%. | Med | Quantum (C) |

**E. Adversarial Negotiation**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| NEG-01 | RL agent proxies | Task 1: Build PyTorch agents; Task 2: Train on DALRN simulations; Task 3: Proxy parties; Task 4: Add UAMP reward functions. | Creates agents for disputes; RL convergence <100 epochs. | High | Adversarial (E) |
| NEG-02 | Nash equilibrium solver | Task 1: Use Nashpy to compute; Task 2: Optimize for $1M scales; Task 3: Ensure Pareto; Task 4: Test on multi-party scenarios. | Solves <5s; outcomes fair (Gini <0.3). | High | Adversarial (E) |
| NEG-03 | Human override | Task 1: Add UI button; Task 2: Log overrides; Task 3: Retrain on feedback; Task 4: Notify via email. | Button functional; logs stored securely. | Med | Adversarial (E) |

**Phase 4: Generalization (Months 7-8)**

Focus: UAMP adapters, DAO governance.

**IV. Blockchain & Infrastructure**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| BC-01 | Settlement smart contracts | Task 1: Write Solidity code; Task 2: Deploy on testnet; Task 3: Integrate DALRN outcomes; Task 4: Add UAMP multi-currency support. | Enforces resolutions; **gas budget** documented for chosen L2 with compression/batching; publish P50/P95 gas use for anchor/escrow flows; store only **CIDs/hashes** (no PII) on‑chain | High | All |
| BC-02 | Stripe/Coinbase payout | Task 1: Integrate APIs; Task 2: Trigger on acceptance; Task 3: Handle fiat/crypto; Task 4: Test refunds. | Auto-payments succeed 100%; compliance with PCI-DSS. | High | All |
| BC-03 | IPFS audit trail | Task 1: Upload logs to IPFS; Task 2: Generate hashes; Task 3: Link to dashboard; Task 4: Ensure immutability. | Retrievable via hash; tamper-proof. | Med | All |
| INF-01 | AWS ECS deployment | Task 1: Write Terraform scripts; Task 2: Set up VPC/ECS; Task 3: Auto-scale rules; Task 4: Deploy DALRN/UAMP. | Handles 100 users; deployment time <5min. | High | All |
| INF-02 | Prometheus monitoring | Task 1: Install Prometheus; Task 2: Monitor latency/accuracy; Task 3: Set alerts; Task 4: Dashboard in Grafana. | Alerts on >500ms; 99.9% uptime. | Med | All |

**V. Security & Compliance (Cross-Phase)**

| **ID** | **Requirement** | **Task-Level Breakdown** | **Acceptance Criteria** | **Priority** | **DALRN Topic Link** |
| --- | --- | --- | --- | --- | --- |
| SEC-01 | Penetration testing | Task 1: Run OWASP ZAP scans; Task 2: Fix CVEs; Task 3: Annual external audit; Task 4: Document fixes. | 0 critical CVEs; report generated. | High | Homomorphic (B) |
| SEC-02 | Rate limiting | Task 1: Implement with FastAPI middleware; Task 2: Block >100 RPM; Task 3: Integrate AWS WAF; Task 4: Test DDoS scenarios. | Handles attacks; logs blocked IPs. | High | All |
| SEC-03 | EU AI Act compliance | Task 1: Build policy agent; Task 2: Classify high-risk features; Task 3: Generate reports; Task 4: Update dynamically. | Compliance score >95%; auto-updates on regulations. | Med | Federated (D) |

**Non-Functional Requirements (Task-Level Expansion)**:

* Performance: Task 1: Benchmark each module; Task 2: Optimize with profiling; Acceptance: <1s responses,
* service‑specific SLOs; rollup: 99.5% for Gateway/Search; maintenance windows carved out; error budgets + alert policies defined
* .
* Scalability: Task 1: Set auto-scaling groups in AWS; Task 2: Load test with Locust; Acceptance: Handles 100 disputes.
* Usability: Task 1: Conduct user testing; Task 2: Iterate on feedback; Acceptance: <5 clicks for submission.
* Testing: Task 1: Write unit tests (pytest); Task 2: Integration tests; Task 3: Coverage >80%; Acceptance: CI/CD pass rate 100%.
* **Cross‑referenced in Security & Compliance**
* **Proof‑of‑Due‑Process (PoDP) Receipts:** Every dispute generates a cryptographic chain of step receipts (intake → retrieval → negotiation → approval), each capturing inputs, parameters, outputs (hashed), and policies. The chain commits to a **Merkle root** anchored on L2.
* **ε‑Ledger (DP Ledger):** Per‑tenant, per‑model ledger tracking , clipping bounds, accountant settings, and composition across rounds; surfaced in /status/{id} and referenced in PoDP receipts

**B) PoDP (Proof‑of‑Due‑Process) — Receipt & Merkle Spec**

**B.1 Concepts**

* **Step receipt**: Immutable record of a pipeline step (e.g., Retrieval‑v1, Negotiation‑NE‑v2). Contains *only* metadata and digests (no PII/plaintext).
* **Receipt chain**: Ordered list of step receipts. The client (or orchestrator) computes the **Merkle root** over all receipts.
* **Anchor**: On‑chain transaction recording (disputeId, merkleRoot, metadata).

**B.2 JSON Schemas (Draft‑07 style)**

**Receipt**

{

"$id": "https://dalrn.dev/schemas/receipt.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["receipt\_id", "dispute\_id", "step", "inputs", "params", "artifacts", "hashes", "signatures", "ts"],

"properties": {

"receipt\_id": {"type": "string", "description": "UUID v4"},

"dispute\_id": {"type": "string"},

"step": {"type": "string", "enum": [

"INTAKE\_V1", "EMBED\_ENCRYPT\_V1", "FHE\_DOT\_V1",

"ANN\_SEARCH\_V1", "REWEIGHT\_V1",

"NEG\_NASH\_V1", "NEG\_BARGAIN\_V1",

"HUMAN\_OVERRIDE\_V1", "ESCROW\_UPDATE\_V1"

]},

"inputs": {

"type": "object",

"properties": {

"cid\_bundle": {"type": "string"},

"model\_id": {"type": "string"},

"index\_id": {"type": "string"},

"epsilon\_budget\_before": {"type": "number"}

},

"additionalProperties": false

},

"params": {

"type": "object",

"properties": {

"he\_params": {"type": "object", "properties": {

"scheme": {"type": "string", "enum": ["CKKS"]},

"poly\_modulus\_degree": {"type": "integer"},

"coeff\_mod\_bit\_sizes": {"type": "array", "items": {"type": "integer"}},

"scale": {"type": "number"}

}},

"search\_params": {"type": "object", "properties": {

"index": {"type": "string"},

"k": {"type": "integer"},

"reweight\_iters": {"type": "integer"}

}},

"negotiation\_params": {"type": "object", "properties": {

"solver": {"type": "string"},

"selection\_rule": {"type": "string"},

"batna": {"type": "array", "items": {"type": "number"}}

}}

},

"additionalProperties": true

},

"artifacts": {

"type": "object",

"properties": {

"candidate\_set\_cid": {"type": "string"},

"policy\_card\_cid": {"type": "string"},

"explanation\_memo\_cid": {"type": "string"}

},

"additionalProperties": false

},

"hashes": {

"type": "object",

"properties": {

"inputs\_hash": {"type": "string", "description": "keccak256 of normalized canonical inputs"},

"outputs\_hash": {"type": "string", "description": "keccak256 of canonical outputs (e.g., top‑k IDs, utilities)"}

},

"additionalProperties": false

},

"signatures": {

"type": "array",

"items": {"type": "object", "properties": {

"by": {"type": "string"},

"sig": {"type": "string"}

}, "required": ["by", "sig"]}

},

"ts": {"type": "string", "format": "date-time"}

}

}

**ReceiptChain**

{

"$id": "https://dalrn.dev/schemas/receipt\_chain.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["dispute\_id", "receipts", "merkle\_root", "algo"],

"properties": {

"dispute\_id": {"type": "string"},

"receipts": {"type": "array", "items": {"$ref": "receipt.json"}},

"algo": {"type": "string", "enum": ["keccak256"]},

"merkle\_root": {"type": "string"},

"merkle\_leaves": {"type": "array", "items": {"type": "string"}}

}

}

**B.3 Merkle layout & hashing**

* **Leaf**: leaf\_i = keccak256( canonical\_json(receipt\_i) )
* **Tree**: pairwise keccak256(left || right); if odd, duplicate last.
* **Root**: merkle\_root
* **Canonicalization**: RFC‑8785‑style JSON canonicalization (stable key order, UTF‑8, no whitespace).
* **Anchoring metadata** (URI): optional IPFS CID that stores the full ReceiptChain JSON.

**B.4 Example mini‑chain (abbrev.)**

{

"dispute\_id": "disp\_9f3e...",

"receipts": [

{"receipt\_id": "r1", "step": "EMBED\_ENCRYPT\_V1", "hashes": {"inputs\_hash": "0x...", "outputs\_hash": "0x..."}, "ts": "2025-08-11T01:23:45Z"},

{"receipt\_id": "r2", "step": "FHE\_DOT\_V1", "hashes": {"inputs\_hash": "0x...", "outputs\_hash": "0x..."}, "ts": "2025-08-11T01:23:50Z"}

],

"algo": "keccak256",

"merkle\_root": "0xabc...",

"merkle\_leaves": ["0x11..", "0x22.."]

}

**C) ε‑Ledger (DP Budget) — Schema & Rules**

**C.1 Purpose**

Tenant‑scoped ledger that tracks **DP composition** across FL rounds and model releases; enforces budget caps; surfaces privacy state to users and auditors.

**C.2 Entities**

* **Ledger** → one per tenant.
* **Entry** → one per FL round or DP operation.

**C.3 JSON Schemas**

**EpsLedgerEntry**

{

"$id": "https://dalrn.dev/schemas/eps\_entry.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["tenant\_id", "model\_id", "round", "accountant", "epsilon", "delta", "clipping\_C", "sigma", "ts", "sig"],

"properties": {

"tenant\_id": {"type": "string"},

"model\_id": {"type": "string"},

"round": {"type": "integer"},

"accountant": {"type": "string", "enum": ["RDP", "MA", "zCDP"]},

"epsilon": {"type": "number"},

"delta": {"type": "number"},

"clipping\_C": {"type": "number"},

"sigma": {"type": "number"},

"participants": {"type": "array", "items": {"type": "string"}},

"composition\_state": {"type": "object", "additionalProperties": true},

"ts": {"type": "string", "format": "date-time"},

"sig": {"type": "string"}

}

}

**EpsLedger**

{

"$id": "https://dalrn.dev/schemas/eps\_ledger.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["tenant\_id", "budget", "entries"],

"properties": {

"tenant\_id": {"type": "string"},

"budget": {"type": "number", "description": "Max allowable ε before block"},

"entries": {"type": "array", "items": {"$ref": "eps\_entry.json"}},

"spent": {"type": "number"}

}

}

**C.4 Enforcement rules**

* **Pre‑round check**: if spent + ε\_round > budget → **block** training.
* **Post‑round update**: append signed entry; update spent with accountant’s composed ε.
* **Cross‑ref to PoDP**: include epsilon\_budget\_before/after in relevant receipts.
* **D) Solidity — Anchor Contract Interface (L2)**
* Minimal interface to anchor PoDP Merkle roots and (optionally) individual receipt hashes. Production should add role‑based access control and rate‑limits; here we keep it simple.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.24;

interface IAnchorReceipts {

event RootAnchored(

bytes32 indexed disputeId,

bytes32 merkleRoot,

bytes32 modelHash,

uint256 round,

string uri, // optional: IPFS URI for ReceiptChain JSON

bytes32[] tags // e.g., ["PoDP","DALRN","v0.1"]

);

event ReceiptAnchored(

bytes32 indexed disputeId,

bytes32 receiptHash,

uint256 stepIndex,

string uri

);

function anchorRoot(

bytes32 disputeId,

bytes32 merkleRoot,

bytes32 modelHash,

uint256 round,

string calldata uri,

bytes32[] calldata tags

) external returns (uint256 anchorId);

function anchorReceipt(

bytes32 disputeId,

bytes32 receiptHash,

uint256 stepIndex,

string calldata uri

) external returns (uint256 anchorId);

function latestRoot(bytes32 disputeId)

external

view

returns (bytes32 merkleRoot, uint256 blockNumber);

}

**Notes**

* disputeId: off‑chain UUID hashed to bytes32.
* modelHash: keccak256 of model weights or model card; use to pin outputs to a model version.
* uri: IPFS CID (optional) for the full ReceiptChain or single receipt.
* **Gas**: keep payload minimal; large artifacts stay on IPFS.

**E) Developer instrumentation — PoDP end‑to‑end**

1. **Canonicalize** each step’s inputs/outputs → JSON → keccak256 → leaf\_i.
2. **Build Merkle** over leaves → root.
3. **POST** ReceiptChain JSON to IPFS → uri.
4. **anchorRoot** on L2 with (disputeId, root, modelHash, round, uri, tags).
5. If needed, **anchorReceipt** for critical steps (e.g., human override) with receiptHash and per‑step uri.
6. Expose PoDP artifacts in UI /status/{id} with explorer links.

**F) Ground‑breaking differentiators (concise)**

1. **PoDP receipts**: verifiable due‑process, independent of data access.
2. **ε‑ledger**: auditable privacy budgets per tenant/model with enforced caps.
3. **Negotiation DSL**: machine‑readable fairness criteria + deterministic selection rules.
4. **Robust FL**: secure aggregation + robust statistics + contribution metering.
5. **Crypto‑shredding lifecycle**: erasure = key destruction, recorded in PoDP.

**4. Development Roadmap and Timeline**

**Phased Roadmap** (Hypothetical, Verifiable via Gantt Charts in Tools like Asana):

* **Month 1**: Setup repo, build marketing site/UI (MKT/UI IDs); prototype router (RTR).
* **Month 2**: Implement homomorphic layer (HE); initial federated hub (FL).
* **Month 3**: Add quantum search (QS); test DALRN core integration.
* **Month 4**: Develop adversarial negotiation (NEG); deploy MVP on AWS (INF).
* **Month 5**: Build blockchain (BC); add monitoring (INF-02).
* **Month 6**: Test end-to-end; optimize performance.
* **Month 7-8**: Add UAMP adapters (e.g., healthcare); pilots and iterations [cite: Ries, 2011, "The Lean Startup," for iterative MVPs].

**Milestones**: MVP ready by Month 4; full UAMP by Month 8.

**5. Skill Sets Required for End-to-End Solution**

A lean team (3-5 Coding Agent), is recommended, with skills covering the five topics

* **Founder/Lead Developer (Agent)**: Python/JS; Claude integration; overall architecture (required for all topics).
* **Full-Stack Developer (Agent)**: React/FastAPI/Docker; UI/Backend implementation (essential for UI/RTR/INF).
* **AI/ML Specialist (Agent)**: PyTorch/PySyft/Qiskit/TenSEAL; model development (critical for HE/QS/FL/NEG).
* **Blockchain/DevOps Engineer (Agent)**: Solidity/Terraform/AWS; deployment/CI/CD (key for BC/INF/SEC).
* **Compliance/Legal Consultant (Agent)**: GDPR/HIPAA/EU AI Act expertise; audits (vital for HE/FL/SEC).

**6. Risks, Mitigations, and Assumptions**

**Risks**:

* Technical Complexity: Mitigation: Modular phases; use pre-built libs [cite: Newman, 2015].
* Compliance Failures: Mitigation: Early audits; policy agents [cite: EU AI Act draft, 2024].
* Scalability Issues: Mitigation: AWS auto-scaling; load testing [cite: AWS Batch docs, 2024].

**Assumptions**: Access to cited libs; user pilots for validation. **Dependencies**: Grok for initial docs; Claude for code.

**7. References and Citations**

* Abadi et al., 2016, "Deep learning with differential privacy," CCS.
* arXiv:2307.16562, 2023, "SAKSHI: Decentralized AI Platforms."
* arXiv:2406.00252v1, 2024, "Multi-Modal and Multi-Agent Systems Meet Rationality: A Survey."
* Buterin, 2014, Ethereum Whitepaper.
* Cheon et al., 2017, "Homomorphic encryption for arithmetic of approximate numbers," ASIACRYPT.
* Grover, 1996, "A fast quantum mechanical algorithm for database search," STOC.
* McKinsey Global Institute, 2023, "The economic potential of generative AI."
* McMahan et al., 2017, "Communication-efficient learning of deep networks from decentralized data," AISTATS.
* Nash, 1951, "Non-cooperative games," Annals of Mathematics.
* Newman, 2015, "Building Microservices," O'Reilly.
* Osborne & Rubinstein, 1994, "A Course in Game Theory," MIT Press.
* PMC, 2021, "Using Artificial Intelligence to provide Intelligent Dispute Resolution Support."
* ResearchGate, 2022, "Blockchain Dispute Resolution for Decentralized Autonomous Organizations."
* Ries, 2011, "The Lean Startup."
* Tang, 2019, "An affirmative solution to the approximate quantum search problem," arXiv.
* Watts & Strogatz, 1998, "Collective dynamics of 'small-world' networks," Nature.

This document is the single source of truth for the project, comprehensive for developers and Claude Code. If updates are needed, flag with justification.