# 📘 Implementation Guide for SDG-Focused Blockchain Projects

**EMURGO x UNDP Blockchain Accelerator**
**Version:** May 2025

## Content

# 🎯 Purpose

This guide provides SDG teams with a comprehensive, step-by-step process to design and deploy blockchain-powered Proofs-of-Concept (PoCs) using the Cardano ecosystem. It draws from real challenges submitted by UNDP country offices and maps each challenge to sample verticals: **Climate Action**, **Digital Identity**, **Financial Inclusion** and **Supply Chain & Product Traceability.**

# 🔧 Core Design Considerations

| Layer | Component | Purpose | Recommended Tools on Cardano |
|---|---|---|---|
| 🧾 Identity | DID & Verifiable Credential Issuance | Attest roles, permissions, eligibility, and impact participation | 🔸 CIP-68-based identifiers<br>🔸 Off-chain VC creation via Verifiable Credential Schema (W3C) |
| 🔐 Wallet | Custody & Signing | Secure access to assets, smart contract signing, and VC presentation | 🔸 Lace Wallet<br>🔸 Nami<br>🔸 Eternl<br>🔸 Flint Wallet<br>🔸 Smart wallet interfaces (via Lucid) |
| 📜 Smart Contracts | Business Logic & Enforcement | Execute logic such as loan disbursement, payout criteria, VC anchoring | 🔸 Aiken (lightweight validator authoring)<br>🔸 Marlowe (low-code financial logic)<br>🔸 Plutus (custom logic) |
| 🧠 Data & Metadata | Asset Tracking & Representation | Represent credentials, tokenized data, off-chain data links | 🔸 CIP-68 (NFT metadata pointers)<br>🔸 IPFS (off-chain VC or proof data)<br>🔸 Koios / Blockfrost (indexers) |
| 🖥️ UI & APIs | Human Interaction Layer | User access, dashboards, verification portals, VC requests | 🔸 Lucid Evolution SDK<br>🔸 MeshJS (wallet & smart contract tools)<br>🔸 GraphQL APIs<br>🔸 React + Typescript for frontends |

# 🌱 Vertical 1: Climate Action

## 🧠 Use Case

Design a transparent, blockchain-based Payment for Ecosystem Services (PES) program.

The goal is to reward verified ecological actions (e.g., tree planting, soil restoration) with programmable payments in ADA or stablecoins using Verifiable Credentials (VCs), smart contracts, and metadata NFTs — implemented using **Cardano-native tools** with no dependency on external oracle-based systems or deprecated identity frameworks.

### 1. Functional Flow

1. A farmer performs an ecological activity (e.g., tree planting).
2. A verifier (e.g., NGO or cooperative agent) inspects and validates the activity.
3. A **Verifiable Credential (VC)** is issued to confirm the ecological action.
4. A **CIP-68 NFT** is minted, embedding:
    - Structured metadata (tree species, GPS, timestamp)
    - A **SHA-256 hash of the VC** for verifiability
5. A **smart contract** validates the hash and disburses funds to the farmer's wallet.

### 2. Architecture Diagram

```
[Farmer] → [Verifier Mobile App] → [VC Issued]
                  ↓
      [NFT Minted w/ CIP-68 Metadata + VC Hash]
                  ↓
   [Smart Contract Validates VC Hash → Pays ADA]
                  ↓
        [Farmer Wallet (Lace, Eternl, Typhon)]
```

### 3. Tools & Libraries

| Layer | Toolset | Description |
|---|---|---|
| Identity | VC Hashing (W3C JSON-LD) | Attestation of verified ecosystem action |
| Metadata | CIP-68 + IPFS | Anchor VC hash and location metadata on-chain |
| Smart Contracts | Aiken | Payout contract verifying credential hash match |
| UI / Wallets | **Mesh SDK + Lucid Evolution** | Wallet interaction, NFT minting, verifier/farmer dashboards |
| Storage | IPFS or GitHub | Off-chain storage of VC files, photos, geodata, logs |

### 4. Credential Format (VC Example)

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "id": "vc:climate:12345",
  "type": ["VerifiableCredential", "ClimateActivity"],
  "issuer": "did:cardano:verifier123",
  "issuanceDate": "2025-07-03T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:farmer987",
    "activityType": "TreePlanting",
```

```
    "treeSpecies": "Acacia Senegal",
    "location": {
      "lat": "15.8921",
      "lng": "35.1831"
    },
    "timestamp": "2025-07-01T13:00:00Z"
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-07-03T01:00:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:cardano:verifier123#keys-1",
    "jws": "eyJhbGciOiJFZERTQSJ9.."
  }
}
```

### 5. CIP-68 NFT Metadata (Example)

```
{
  "name": "TreePlanting #12345",
  "description": "Verified climate action - Acacia planting",
  "image": "ipfs://QmExampleHash/photo.jpg",
  "vc_hash": "0x9f2a38...d21b7a",
  "lat": "15.8921",
  "lng": "35.1831",
  "tree_species": "Acacia Senegal",
  "verified_by": "did:cardano:verifier123",
  "timestamp": "2025-07-01T13:00:00Z"
}
```

### 6. Smart Contract Logic (Aiken Sample)

```
fn payout(vc_hash: ByteArray, submitted_hash: ByteArray) -> Bool {
  assert(vc_hash == submitted_hash, "Credential mismatch");
  true
}
```

- Hash must match the one stored in the NFT metadata.
- In future versions, issuer whitelisting or revocation checks can be added.

## 7. Frontend Features

- **Verifier UI**: Credential issuance form with GPS, photo upload
- **NFT Mint UI**: Embed VC hash and metadata via Mesh + Lucid Evolution
- **Farmer Dashboard**: Display NFTs, payment status, QR export
- **Admin Panel**: Track verifications, contract executions, disbursed funds

## 8. Deployment Simulation (Testnet)

- Use testnet wallets: **Lace, Eternl, Typhon**
- Issue VC, mint NFT via **Mesh + Lucid Evolution**
- Submit payout transaction to Aiken contract (manual TX or via UI)
- Log VC → NFT → payout steps in GitHub, include TX hashes

## 9. Output Expectations

| Deliverable | Description |
|---|---|
| `contracts/payout.aiken` | Smart contract that verifies credential hash |
| `vc-schema/climate-activity.json` | VC JSON schema with location and activity data |
| `mint-nft.tsx` | Frontend flow to mint CIP-68 NFT via Lucid Evolution |
| `demo/funding-flow.mp4` | Optional demo recording of full PES cycle |

| | |
|---|---|
| `docs/dashboard-mockup.png` | Optional UI concept for verifier/farmer view |
| `README.md` | Instructions for running full simulation locally |

### 10. Best Practices for Climate Action PoCs

✅ **Start Small**
Use one verifier and a handful of farmers. Scale only after first cycle is documented.

✅ **Use Standard Schemas**
Design VCs using `@context`, `proof`, and structured `credentialSubject` fields.

✅ **Anchor Hashes Only**
Only store VC hashes on-chain. Keep documents/images in IPFS or GitHub.

✅ **Define Governance**
Use multisig or DAO to control payout contract. Define who can revoke VCs.

✅ **Ensure Field Usability**
Offline UI caching for verifiers in rural areas. Multilingual UI tags included.

✅ **Document Every Step**
Log VC issuance, NFT mint, and payout TXs. Link them in a public GitHub repo or dashboard.

# 🧾 Vertical 2: Digital Identity

## 🧠 Use Case

**Design a decentralized identity solution for underserved populations to access public and private services with verifiable credentials.**

In many low-infrastructure regions, people lack formal documentation. This solution enables identity creation and access using Cardano-native verifiable credentials (VCs), anchored via CIP-68 NFTs and validated through off-chain hash comparisons — with optional on-chain enforcement for gated access.

## 1. Functional Flow Diagram

```
[NGO Agent / Self Onboarding]
          ↓
[VC Created & Signed (JSON-LD)]
          ↓
[SHA-256 Hash of CredentialSubject]
          ↓
[Anchor hash in CIP-68 NFT (optional)]
          ↓
[Store VC in Wallet / Mobile App]
          ↓
[Verifier App → Upload or Scan VC]
          ↓
[VC Hash Compared to On-Chain Anchor or Registry]
          ↓
[Access Granted / Denied]
```

## 2. Verifiable Credential (VC) – Schema Breakdown

📁 vc-schema/basic-identity.json

| Field | Description | Example |
|---|---|---|
| `@context` | W3C standard credential context | `https://www.w3.org/2018/credentials/v1` |
| `type` | Credential type | `["VerifiableCredential", "BasicIDCredential"]` |
| `issuer` | DID of the credential issuer | `did:cardano:ngo789` |

| | | |
|---|---|---|
| `issuanceDate` | Date of issuance | `2025-07-10T00:00:00Z` |
| `credentialSubject` | Data about the subject (user) | `{ name, birthYear, nationality }` |
| `proof` | Digital signature block | Ed25519 signature |

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiableCredential", "BasicIDCredential"],
  "issuer": "did:cardano:ngo789",
  "issuanceDate": "2025-07-10T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:user123",
    "fullName": "Fatima Diallo",
    "birthYear": "1990",
    "nationality": "Senegalese",
    "verifiedBy": "Red Cross Dakar"
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-07-10T00:00:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:cardano:ngo789#keys-1",
    "jws": "eyJhbGciOiJFZERTQSJ9..."
  }
}
```

### 3. CredentialSubject Hashing Script

📁 `utils/hash.ts`

```ts
import { createHash } from 'crypto'

export function hashCredentialSubject(credentialSubject: object): string {
  const canonical = JSON.stringify(credentialSubject)
  return createHash('sha256').update(canonical).digest('hex')
}

// Example usage
const subject = {
  fullName: "Fatima Diallo",
  birthYear: "1990",
  nationality: "Senegalese",
  verifiedBy: "Red Cross Dakar"
}

console.log("VC Hash:", hashCredentialSubject(subject))
```

⚠ Exclude the `proof` field from hashing to ensure verifiability and privacy.

### 4. CIP-68 NFT Metadata (for Anchoring VC)

📁 `nft-metadata.json`

```json
{
  "name": "ID Credential - Fatima Diallo",
  "description": "Anchored VC for identity verification",
  "vc_hash": "0x9f2a3878c1ed2f...d21b7a",
  "issuer_did": "did:cardano:ngo789",
  "credential_type": "BasicIDCredential",
  "schema_id": "ipfs://QmSchemaRef123",
  "issued_at": "2025-07-10T00:00:00Z"
}
```

Mint using **Mesh SDK + Lucid Evolution** with an inline datum and reference NFT under CIP-68.

### 5. Smart Contract – VC Hash Verification

📁 `contracts/verify_id.aiken`

```
fn main(submitted_hash: ByteArray, expected_hash: ByteArray) -> Bool {
  assert(submitted_hash == expected_hash, "Invalid Credential")
  true
}
```

Deployment:

- Use this contract to **gate access** (e.g., claim subsidy, mint token, access portal).
- Optional: Extend to support datum-based hash registry.

### 6. Tooling Table

| Layer | Toolset | Function |
|---|---|---|
| Issuance | Custom issuer dashboard / CLI | NGO admins generate and sign VCs |
| Hashing | `crypto` (NodeJS), `sha2` (Rust) | VC integrity check via SHA-256 |
| Anchoring | CIP-68 NFT, Mesh SDK | Optional hash storage on-chain |
| Wallet UI | Eternl, Lace, PWA | Display VC and QR for offline use |

| | | |
|---|---|---|
| Smart Contracts | Aiken | Validate hash match during access |
| Frontend | Mesh SDK + Lucid Evolution | dApp for validators, verifiers |
| Storage | IPFS, Arweave, GitHub | Store VC, schemas, revocation files |

### 7. Frontend Validator Flow (React/TS)

📁 frontend/verify.tsx

```
const handleFileUpload = async (file: File) => {
  const text = await file.text()
  const vc = JSON.parse(text)

  const subjectHash = hashCredentialSubject(vc.credentialSubject)
  const knownHash = await fetchAnchor(vc.credentialSubject.id)

  if (subjectHash === knownHash) {
    showResult("✅ Credential Valid")
  } else {
    showResult("❌ Credential Invalid or Unrecognized")
  }
}
```

## 8. Deployment Simulation Plan

| Step | Actor | Tool |
|---|---|---|
| Create VC | Issuer | Custom CLI / JSON generator |
| Hash credentialSubject | Issuer | `utils/hash.ts` |
| Mint CIP-68 NFT | Issuer | Mesh SDK + Lucid Evolution |
| Upload VC (QR or file) | User | Wallet / PWA export |
| Validate hash | Verifier dApp | React app, smart contract call |

## 9. Output Deliverables

| Path / File | Description |
|---|---|
| `/vc-schema/basic-identity.json` | JSON-LD credential schema |
| `/contracts/verify_id.aiken` | Hash-checking Aiken script |

| | |
|---|---|
| `/utils/hash.ts` | CredentialSubject hashing util |
| `/nft-metadata.json` | CIP-68 NFT metadata template |
| `/frontend/verify.tsx` | Basic VC validation dApp logic |
| `/demo/verify-flow.mp4` | Optional demo screencast |
| `/README.md` | Instructions for testnet setup |

## 🔐 Best Practices for Digital Identity PoCs on Cardano

| ✅ Practice | 💡 Rationale |
|---|---|
| **Use Hashes, Not Signatures On-Chain** | Cardano smart contracts cannot verify Ed25519 signatures. Only hash-matching is supported. |
| **Use JSON-LD + W3C-Compatible Schemas** | Ensures VC interoperability across issuers and chains. |
| **Support Offline Use** | Verifiers in the field can validate VCs via QR scan + local hash check. |

| | |
|---|---|
| **Plan Revocation via Metadata** | Use `expirationDate` or CIP-68 pointer to IPFS-based revocation list. |
| **Minimize Sensitive Data** | Avoid collecting full addresses, ID numbers, religion, or health records. |
| **Structure Codebases Cleanly** | Easy auditability and testing in repositories and workshops. |

**Recommended Directory Structure:**

```
/contracts/verify_id.aiken          -- On-chain VC hash enforcement
/vc-schema/basic-identity.json      -- Credential schema
/utils/hash.ts                      -- VC hashing utility
/frontend/verify.tsx                -- Web or kiosk VC verification
/demo/verify-flow.mp4               -- PoC screencast (optional)
/README.md                          -- Setup and deployment instructions
```

# 🔧 Advanced Practices for Digital Identity PoCs on Cardano

## 1. ✅ Credential Anchoring with CIP-68 NFTs

**Purpose**: Create tamper-evident, on-chain reference to a VC.
**Implementation**:

- Hash the `credentialSubject` portion of the VC.
- Mint a CIP-68 NFT with:
    - `vc_hash`: SHA-256 hash
    - `issuer_did`: e.g., `did:cardano:ngo789`
    - `schema_ref`: IPFS or URL of the schema

    - `purpose`: “Basic ID”, “Training”, etc.

**Benefits**:

- Off-chain VC can be verified by comparing hash.
- Preserves privacy — no personal data on-chain.

## 2. ✅ Offline Presentation + Kiosk Verification

**Purpose**: Ensure usability in remote or low-connectivity areas.
**Implementation**:

- Export VC as:
    - ✅ QR code (with embedded JSON-LD VC)
    - ✅ Printable format (with visible hash or issuer info)
- Verifier app performs:
    - ✅ Local hash computation
    - ✅ Offline schema validation (preloaded schema + trusted DIDs)
- Optionally, later online sync with CIP-68 hash.

**Benefit**:

- Works in humanitarian deployments or borderless field contexts.

## 3. ✅ Revocation Strategy via IPFS + CIP-68

**Purpose**: Enable revocation without central APIs.
**Implementation**:

- Issuer maintains a list of revoked `vc_hash`es (CSV or JSON).
- Push this list to IPFS.

- Mint or update a CIP-68 NFT with:
    - `revocation_list_ipfs`: `ipfs://QmHash...`
    - `last_updated`: `2025-07-12T00:00Z`

**Frontend Logic**:

```
const isRevoked = async (vcHash) => {
  const list = await fetch("https://ipfs.io/ipfs/QmRevocationList")
  return list.includes(vcHash)
}
```

**Benefit**:

- Audit trail of credential lifecycle changes on-chain.

## 4. ✅ Smart Contract Gating (Hash Matching Only)

**Purpose**: Enforce access control (e.g., subsidies, voting, scholarships).
**Implementation**:

📁 `contracts/is_verified.aiken`

```
fn main(vc_hash: ByteArray, allowed: ByteArray) -> Bool {
  vc_hash == allowed
}
```

- Submit user VC hash as redeemer.
- Match against a known on-chain hash in datum or inline redeemer.

**Limitation**:

- No VC field logic or signature validation is possible on-chain.

## 5. ✅ GLEIF Tiered Issuer Reputation

**Purpose**: Help users quickly assess VC issuer credibility.
**Implementation**:

- Maintain a JSON registry (IPFS or frontend file):

```
{
  "did:cardano:gleif-org": {
    "tier": "GLEIF",
    "name": "Gov ID Authority",
    "country": "DE"
  },
  "did:cardano:ngo789": {
    "tier": "Accredited NGO",
    "name": "Red Cross Dakar",
    "country": "SN"
  }
}
```

- Add issuer tier icon in UI:
  🟢 = GLEIF | 🟡 = NGO | ⚪ = Community

**Benefit**:

- Frictionless trust signal in both mobile and kiosk flows.

## 6. ✅ Traceability of Issuance Events

**Purpose**: Monitor where, when, and by whom VCs were issued.
**Implementation**:

- Mint metadata-only transaction with:

```
{
  "vc_hash": "0xabc123...",
  "issuer_did": "did:cardano:ngo789",
  "recipient_did": "did:cardano:user123",
  "issued_at": "2025-07-10T00:00Z"
}
```

- Pull data via Koios or Blockfrost for dashboards.

**Benefit**:

- Track credential issuance at scale (map by issuer or region).

## 7. ✅ Progressive Credentialing

**Purpose**: Lower friction for first-time users.
**Implementation**:

- Begin with low-risk credentials:
    - 🟡 Proof of Attendance
    - 🟡 Community Member Verified
    - 🟡 Phone or Email Linked
- Allow users to build up credential layers over time.

**Benefit**:

- Higher adoption rates and user safety in sensitive areas.

## 8. ✅ UX Enhancements for Real-World Use

| Feature | Description |
|---|---|
| Language Localization | Translate schemas/UI dynamically by locale |
| Trust Icons | Show issuer credibility visually (🟢 / 🟡 / ⚪) |
| Expiration Alerts | Highlight VCs nearing expiration or revoked |

| | |
|---|---|
| Offline Mode | Cache VC schemas and trusted issuer list locally |
| VC Export + Recovery | QR download + encrypted backup of VC (optional) |

## 🗒️ Summary Table: Advanced Enhancements

| Feature | Realistic on Cardano | Tools Involved |
|---|---|---|
| CIP-68 VC Hash Anchoring | ✅ | Mesh, Lucid Evolution, NFT metadata |
| Offline QR Presentation | ✅ | PWA, QR SDK, IPFS |
| IPFS-Based Revocation | ✅ | IPFS, CIP-68, frontend checker |
| Hash-Based Access Control | ✅ | Aiken contracts |
| GLEIF Tiered Trust Registry | ✅ | JSON + frontend icons |
| Issuance Event Logging | ✅ | Metadata TXs, Koios APIs |
| Progressive Schemas | ✅ | JSON-LD schemas |
| Dashboards / Analytics | ✅ | GraphQL, Koios / custom endpoints |

# 💸 Vertical 3: Financial Inclusion & Payments

## 🧠 Use Case::

*Build a decentralized microloan platform for underserved small businesses.*

Smallholder farmers, women-led cooperatives, and informal vendors often lack access to traditional financial services. This PoC enables credential-based microloans, with VC-anchored eligibility, Aiken smart contracts for disbursement/repayment, and clear Cardano-native tooling for frontend + on-chain coordination.

## 1. 🧩 Functional Flow (Cardano-Compatible)

```
[Borrower Onboards]
       ↓
[VC Issued (Eligibility, KYC, Score)]
       ↓
[VC Hash Anchored in CIP-68 NFT (optional)]
       ↓
[Loan Contract Deployed w/ Terms]
       ↓
[Loan Claimed Manually → Funds Sent]
       ↓
[Manual Repayment via Wallet TX]
       ↓
[NGO Reviews Loan Status via Dashboard]
```

## 2. 🧾 Credential Schema – `vc-schema/msme-eligibility.json`

| Field | Description |
|---|---|
| `type` | Type of VC (e.g. MSMEEligibility) |
| `issuer` | DID of issuing NGO |
| `credentialSubject.name` | Borrower's legal name |
| `businessType` | Informal sector classification |
| `location` | Region or city |

| | |
|---|---|
| `incomeBracket` | Self-reported or verified income |
| `eligibilityScore` | A simple float between 0–1 |

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiableCredential", "MSMEEligibility"],
  "issuer": "did:cardano:ngo123",
  "issuanceDate": "2025-08-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:user789",
    "name": "Ama Diarra",
    "businessType": "Agriculture",
    "location": "Bauchi, Nigeria",
    "incomeBracket": "USD 200-400/mo",
    "eligibilityScore": 0.72
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-08-01T01:00:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:cardano:ngo123#key-1",
    "jws": "eyJhbGciOiJFZERTQSJ9..."
  }
}
```

### 3. 🔐 CredentialSubject Hashing Script

📁 `utils/hash.ts`

```
import { createHash } from 'crypto'

export function hashCredentialSubject(input: object): string {
  const canonical = JSON.stringify(input)
  return createHash('sha256').update(canonical).digest('hex')
}

// Example usage
const subject = {
```

```
  name: "Ama Diarra",
  businessType: "Agriculture",
  location: "Bauchi, Nigeria",
  incomeBracket: "USD 200-400/mo",
  eligibilityScore: 0.72
}

console.log("VC Hash:", hashCredentialSubject(subject))
```

### 4. 📜 Smart Contract Logic (Aiken) – `contracts/loan_repayment.aiken`

```
// Validate that the VC used is one of the authorized hashes
fn validate_eligibility(vc_hash: ByteArray, expected_hash: ByteArray) ->
Bool {
    vc_hash == expected_hash
}

// Validate repayment meets deadline and minimum repayment
fn validate_repayment(due_date: Int, amount: Int, paid_on: Int) -> Bool {
    paid_on <= due_date && amount >= 1000000 // min 1 ADA
}

// Combine checks for loan claim
fn main(
    submitted_hash: ByteArray,
    expected_hash: ByteArray,
    amount: Int,
    due_date: Int,
    paid_on: Int
) -> Bool {
    validate_eligibility(submitted_hash, expected_hash)
    && validate_repayment(due_date, amount, paid_on)
}
```

> 🔒 Smart contracts only validate — the borrower must manually submit loan claim and repayment TXs.

## 5. 🧰 Tools & Libraries

| Layer | Toolset | Description |
|---|---|---|
| VC Issuance | CLI or NGO form-based UI | Create JSON-LD VC + Ed25519 signature |
| Hashing | `js-sha256`, `hash.ts` | Calculate VC hash to anchor on-chain |
| On-Chain Logic | Aiken | Validator for loan claim + repayment TX |
| Frontend | Mesh SDK + Lucid Evolution | Wallet connect, TX builder, status dashboard |
| Wallets | Eternl / Lace / Typhon / PWA | Borrower interface to claim and repay loans |
| APIs | Koios / Blockfrost | Fetch TX status, repayment status |
| Storage | IPFS | VC backup, signed loan agreements (optional) |

## 6. 🖥️ Frontend Flows (borrower & NGO)

### 📲 Borrower Dashboard – `frontend/repayment.tsx`

- View loan offer (terms: amount, due date)
- Upload VC → Hash computed locally
- Submit loan claim transaction
- Track repayment status (Koios)

### 🧑‍💼 NGO/Funder Dashboard

- Issue VCs to pre-qualified users
- Monitor on-chain status (Repaid, Defaulted)
- Export repayment logs per borrower ID

## 7. 🧪 Deployment Simulation (Testnet)

| Step | Artifact |
| --- | --- |
| VC Issuance | `vc-schema/msme-eligibility.json` |
| VC Hashing | `utils/hash.ts` |
| Contract Compilation | `contracts/loan_repayment.aiken` |
| Loan Claim TX | `logs/loan_disbursement_tx.txt` |
| Repayment TX | `logs/repayment_tx.txt` |
| Frontend UI | `frontend/repayment.tsx` |
| Demo | `demo/loan_flow.mp4` |

## 8. ✅ Best Practices for Microloan PoCs

| ✅ Practice | 🧠 Notes |
| --- | --- |
| **Use fixed loan terms in PoC** | Simplify demo — hardcode amount/due date for testing |
| **Hardcode 1–2 VC hashes** | Avoid needing full VC oracle system |
| **Repayment must be manual** | User submits TX, contract validates timing + amount |
| **Hash-based enforcement only** | VC fields not evaluated on-chain; only hash compared |
| **Track status off-chain (Koios)** | Use wallet address or UTXO tags to track repayment |
| **Revocation not required in PoC** | Assume short-term loans; can add in full system with IPFS later |

### 9. 📦 Output Expectations

| Deliverable | Description |
|---|---|
| `contracts/loan_repayment.aiken` | VC + repayment-checking Aiken contract |
| `vc-schema/msme-eligibility.json` | Credential schema for MSME eligibility |
| `utils/hash.ts` | CredentialSubject hashing utility |
| `frontend/repayment.tsx` | Frontend TX form and repayment checker |
| `demo/loan_flow.mp4` | Optional screen capture of end-to-end flow |
| `README.md` | Setup + signed testnet TXs + links/logs |

# 🔄 Advanced Design Additions

*Optional Enhancements for Financial Inclusion PoC Teams*

These optional components help elevate your PoC from a minimal flow to a **mature, system-level pilot** — covering user lifecycle, behavioral scoring, and impact tracking. Each is modular and Cardano-compatible.

## 🧠 1. Credit Scoring & Risk Segmentation

**Purpose**: Reflect user creditworthiness via off-chain logic, encoded into a Verifiable Credential (VC).

**Why off-chain?** Cardano smart contracts cannot access external data or perform dynamic scoring — credit assessment must be done externally, then attested via a signed VC.

📄 **VC Example: `vc-schema/credit-assessment.json`**

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiableCredential", "CreditAssessment"],
```

```
  "issuer": "did:cardano:ngo123",
  "issuanceDate": "2025-07-15T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:user789",
    "creditTier": "B",
    "maxLoanLimit": 150,
    "scoreBasis": "Field assessment by agent #4"
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "jws": "eyJhbGciOiJFZERTQSJ9...",
    "created": "2025-07-15T01:00:00Z"
  }
}
```

🧠 **Smart Contract Tip:**

Precompute hash of creditTier and validate it in loan disbursement logic:

```
fn check_credit(vc_hash: ByteArray, tier_hash: ByteArray) -> Bool {
    vc_hash == tier_hash // Assume B-tier hash embedded in contract
}
```

💡 **Use Case**: Approve higher loan limits only for B+ and above tiers.

## 🔁 2. Loan Renewal, Reward, and Escalation Flows

**Purpose**: Represent successful repayment behavior with a follow-up VC — enabling progressive access to larger loans or better terms.

🧭 **Flow:**

```
[Loan Repaid] → [New VC Issued: "LoanRepaid": true]
                ↓
[New Loan Request → Higher Tier Validator → Approval]
```

📄 **VC Example: `vc-schema/loan-history.json`**

```
{
  "type": ["VerifiableCredential", "LoanRepaymentStatus"],
  "issuer": "did:cardano:ngo123",
  "issuanceDate": "2025-10-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:user789",
    "loanRepaid": true,
    "dateRepaid": "2025-10-01"
  }
}
```

🧠 **Implementation Tip:**

Use a new Aiken validator for **Tier 2 Loan Contract**, checking for prior `LoanRepaymentStatus` hash.

## 📉 3. Grace Periods & Default Recovery Paths

**Purpose**: Simulate realistic repayment behavior (e.g., delays, grace periods, status flagging).

🔐 **Smart Contract Extension:**

```
fn validate_with_grace(today: Int, due: Int, grace_days: Int) -> Bool {
    today <= due + grace_days
}
```

🔁 **Off-Chain VC Logic:**

- If repayment missed: Issue `LoanDefaultStatus` VC with reason and lockout period.
- Recovered users can reapply after NGO re-issues eligibility VC.

```
{
  "type": ["VerifiableCredential", "LoanDefaultStatus"],
  "credentialSubject": {
    "id": "did:cardano:user789",
    "defaultedOn": "2025-10-15",
    "status": "Ineligible",
    "recheckAfter": "2025-12-01"
  }
}
```

## 📊 4. Impact-Linked Repayment Metrics (Optional VC Layer)

**Purpose**: Track non-financial success metrics to attract impact funders or ESG-aligned capital.

📄 **VC Example: `vc-schema/loan-impact-report.json`**

```
{
  "type": ["VerifiableCredential", "LoanImpactReport"],
  "issuer": "did:cardano:ngo123",
  "issuanceDate": "2025-11-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:user789",
    "hiredEmployees": 1,
    "businessGrowth": "15%",
    "fieldVisit": "Passed"
  }
}
```

> 💡 Can be minted as a **CIP-68 NFT** for display in public dashboards or exported for donor reporting.

🧠 **Bonus:**

- Use this VC in **impact-weighted repayment** models or bonus logic (e.g., reduce interest for high-impact borrowers in follow-up Aiken contract).

### 📦 Summary Table: Optional Enhancements

| Feature | Purpose | On-Chain ? | Tools Used |
|---|---|---|---|
| **Credit Tier VC** | Encodes risk segmentation | ✅ (hash) | JSON VC + Aiken validator |
| **Repayment History VC** | Unlocks renewal or escalated loan logic | ✅ (hash) | NGO-issued VC + tier logic |
| **Grace Period Validator** | Accepts late repayment within tolerance | ✅ | Aiken `validate_with_grace` |
| **Default Status VC** | Encodes borrower lockout period | ✅ (hash) | Off-chain issuance + dashboard |
| **Impact VC** | Reflects outcome metrics post-loan | ✅ (NFT) | IPFS + metadata dashboard |

## 📦 Vertical 4: Supply Chain & Product Traceability

### 🧠 Use Case:

**Create a blockchain-based traceability system for agricultural exports (e.g., coffee, cocoa), ensuring product origin, sustainability claims, and fair trade certifications are verifiable across the supply chain.**

This vertical enables a verifiable trail for certified commodities using Verifiable Credentials (VCs), CIP-68 NFTs to represent product batches, and smart contracts to validate transfers between authorized supply chain actors.

## 1. 🔁 Functional Flow

```
[Farmer / Coop Onboards]
      ↓
[VC Issued: Organic / Fair Trade Certified]
      ↓
[Batch NFT Minted (CIP-68)]
      ↓
[On-Chain Transfers (Processor → Exporter → Retailer)]
      ↓
[QR Scan → Traceability Viewer for Consumers]
```

**Actors**:

- ✅ Farmer or Coop
- ✅ Certifier (e.g., Fair Trade NGO)
- ✅ Supply Chain Participants
- ✅ Retailer + Buyer

## 2. 📄 Credential Schema (Fair Trade VC)

📁 `vc-schema/fair-trade-certification.json`

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiableCredential", "FairTradeCertification"],
  "issuer": "did:cardano:certifierXYZ",
  "issuanceDate": "2025-06-12T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:farmer123",
```

```
    "farmName": "Alto Sierra Coop",
    "certification": "Fair Trade Certified",
    "validUntil": "2026-06-12",
    "region": "Antioquia, Colombia",
    "product": "Coffee Arabica"
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-06-12T12:00:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:cardano:certifierXYZ#key-1",
    "jws": "eyJhbGciOiJFZERTQSJ9..."
  }
}
```

Hash the `credentialSubject` and anchor it in the batch NFT.

### 3. 🎯 Batch NFT Metadata (CIP-68 Format)

📁 `nft-metadata/coffee-batch-234.json`

```
{
  "name": "Coffee Batch #234",
  "product": "Coffee",
  "harvestDate": "2025-07-15",
  "farm": "Alto Sierra Coop",
  "origin": {
    "lat": 6.25184,
    "lon": -75.56359,
    "country": "Colombia"
  },
  "cert_vc_hash": "0xc6f0d3a1...b0f2",
  "supplyChain": [
    {
      "actor": "ProcessorCo",
      "timestamp": "2025-08-01T12:00:00Z",
      "txHash": "abcd123..."
    }
  ],
  "qrUrl": "https://trace.app/coffee/234"
```

```
}
```

The **VC hash** links to the certification claim.
The **supplyChain array** updates with each verified hop.

## 4. 🔐 Transfer Smart Contract (Aiken)

📁 `contracts/verify-transfer.aiken`

```
fn validate_transfer(
  expected_holder: PubKeyHash,
  signer: PubKeyHash,
  approved_list: List<PubKeyHash>
) -> Bool {
  signer == expected_holder && List.member(signer, approved_list)
}
```

💡 More advanced flows can track current holder via datum and enforce hop order dynamically.

## 5. 🧰 Tools & Libraries

| Layer | Toolset | Purpose |
|---|---|---|
| Identity | DID + VC Hashing (JSON-LD) | Issue and verify certifications |
| Tokenization | CIP-68 NFT | Represent product batch and traceability trail |
| Contracts | Aiken | Validate signed transfer hops |
| UI / Wallets | Lucid Evolution, MeshJS, Lace | Mint, sign, trace, and display product paths |
| Storage | IPFS, Arweave | Store VC proof or product images |
| QR Layer | Mesh QR or custom PWA | Consumer-facing scan interface |

## 6. 🔍 Demo Simulation Flow

| Step | Artifact |
|---|---|
| Issue VC | `vc-schema/fair-trade-certification.json` |
| Mint batch NFT | `nft-metadata/coffee-batch-234.json` |
| Simulate 3 transfers | Aiken + wallet signatures |
| QR-linked display | `frontend/trace-ui.tsx` + `scan-qr.mp4` |

## 7. 🧪 Output Expectations

| Deliverable | Description |
|---|---|
| `vc-schema/fair-trade-cert.json` | VC proving sustainability and certification |
| `nft-metadata/coffee-batch.json` | Full batch trace with certification hash |
| `contracts/verify-transfer.aiken` | Validator ensuring legit hops only |
| `demo/scan-qr.mp4` | QR-to-view traceability journey (consumer UX) |
| `frontend/trace-ui.tsx` | React component showing product lifecycle |

## 8. ✅ Best Practices for Supply Chain Traceability PoCs

| Best Practice | Reason |
|---|---|
| Anchor cert VC hash in batch NFT | Maintains integrity and privacy of claims |
| Use CIP-68 to enable NFT updates per hop | Allows progressive updates without reminting |

| | |
|---|---|
| Ensure signed transfers from wallet owners | Validates actor authenticity at each step |
| Separate off-chain and on-chain roles | VCs off-chain; signatures + metadata on-chain |
| Use QR links to live data not static files | Ensures real-time verification (IPFS + Koios fallback recommended) |
| Demo with 3 hops minimum | Simulates real product lifecycle (Processor → Exporter → Retailer) |

# 🆘 Vertical 5: Humanitarian Aid Delivery & Crisis Response (Expanded)

## 🧠 Use Case

**Design a blockchain-based aid delivery system** to verify eligibility, issue traceable payments (or vouchers), and audit aid impact in crisis zones — including refugee camps, natural disasters, or food insecurity regions.

The system ensures that:

- Only verified individuals receive aid.
- Disbursements are logged immutably.
- Aid eligibility persists post-crisis as a digital asset for recovery or follow-up services.

# 1. 🔁 Functional Architecture Overview

```
[NGO Registers Beneficiary]
        ↓
[VC Issued (Eligibility Claim)]
        ↓
[Beneficiary Wallet (PWA / SMS / Lace)]
        ↓
[Smart Contract (Disbursement Logic)]
        ↓
```

```
[Aid (e.g. USDA) Sent + Metadata Logged]
         ↓
[Donor/Aid Dashboard → TX Log via Koios]
         ↓
[Beneficiary Retains Identity for Future Access]
```

### 🧑 Actors:

- **NGO or Field Agent** – Issues the VC
- **Beneficiary** – Receives aid, stores credentials
- **Smart Contract** – Releases funds if VC hash is valid and within aid window
- **Public Verifier** – Monitors disbursements via dashboards

## 2. 📄 Verifiable Credential (VC) for Aid Eligibility

📁 vc-schema/aid-eligibility.json

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiableCredential", "AidEligibilityCredential"],
  "issuer": "did:cardano:ngo123",
  "issuanceDate": "2025-05-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:cardano:beneficiary123",
    "eligibility": "YES",
    "location": "Zaatari Camp, Jordan",
    "familySize": 6,
    "aidType": "Shelter & Food"
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2025-05-01T00:00:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:cardano:ngo123#key-1",
    "jws": "eyJhbGciOiJFZERTQSJ9..."
  }
```

```
}
```

> ⚠️ This VC is validated off-chain. Only the hash of `credentialSubject` is used in smart contract logic.

## 3. 🔐 VC Hashing (Credential Anchoring)

📁 `utils/hash.ts`

```ts
import { createHash } from 'crypto'

export function hashCredentialSubject(subject: object): string {
  const canonical = JSON.stringify(subject)
  return createHash('sha256').update(canonical).digest('hex')
}

// Sample Input:
const subject = {
  eligibility: "YES",
  location: "Zaatari Camp, Jordan",
  familySize: 6,
  aidType: "Shelter & Food"
}

console.log(hashCredentialSubject(subject))
```

> ✅ Use this hash in a **CIP-68 NFT**, and also as an input for smart contract validation.

## 4. 🎯 Aiken Smart Contract – Aid Claim & Disbursement

📁 `contracts/aid_disbursement.aiken`

```
fn main(
  vc_hash: ByteArray,
  allowed_hash: ByteArray,
  now: Int,
```

```
  start: Int,
  end: Int
) -> Bool {
  vc_hash == allowed_hash && now >= start && now <= end
}
```

### 💡 Explanation:

- `allowed_hash`: Stored in datum or passed via redeemer.
- `now`: Current slot/time.
- `start` and `end`: Disbursement window (e.g., 7-day emergency aid).
- ✅ User submits a transaction including their VC hash → if conditions match, contract validates.

## 5. 📦 CIP-68 NFT Metadata – Aid Token

📁 `nft-metadata/aid-voucher-103.json`

```
{
  "name": "Emergency Aid Voucher #103",
  "description": "USDA-backed shelter and food assistance",
  "beneficiary_did": "did:cardano:beneficiary123",
  "vc_hash": "0x9c4a88...ff12",
  "aid_type": "Shelter & Food",
  "disbursed_at": "2025-05-03T14:00:00Z",
  "location": "Zaatari Camp, Jordan"
}
```

✅ Use for audit visibility, traceability, and dashboard display.
Can also be burned after redemption to indicate use.

## 6. 📱 Beneficiary Experience

| Step | Device Used | UX Design Features |
|---|---|---|
| Receive VC | Android / PWA / Paper QR | VC sent via QR, SMS link, or preloaded PWA |
| Submit Aid Claim TX | Wallet or Mobile Agent | Pre-filled transaction → submit button |
| See Confirmation | Local dApp / SMS alert | Confirm receipt (voucher, stablecoin) |
| View History | Optional Aid Journal | VC + token + disbursement records (offline) |

## 7. 🧪 Simulation & Output Expectations

| Deliverable | Description |
|---|---|
| `vc-schema/aid-eligibility.json` | Sample credential used in testnet flow |
| `utils/hash.ts` | Script to hash and anchor VC |
| `contracts/aid_disbursement.aiken` | Disbursement contract with VC hash logic |
| `nft-metadata/aid-voucher-103.json` | Aid token metadata (CIP-68) |
| `frontend/aid-ui.tsx` | React app for NGO aid agents |
| `demo/aid-flow.mp4` | Screencast: onboarding → claim → log |
| `README.md` | Deployment + signed TXs |

# 8. ✅ Best Practices for Aid Delivery Systems

## Identity Onboarding

- Use face scan, mobile ID, or document upload.
- Prefer **in-person agent onboarding** in fragile zones.
- Avoid collecting sensitive data like religion or political status.

## Disbursement Logic

- Time-locked disbursement via Aiken contracts.
- Use USDA or voucher tokens, not volatile ADA.
- NGO admin override (e.g., fraud freeze) is stored off-chain or in metadata.

## Transparency & Auditing

- Log only `vc_hash`, timestamp, and region on-chain.
- IPFS or Arweave for supporting documents or NGO certifications.
- Open GraphQL dashboards for donors/regulators (e.g., via Koios feeds).

## Post-Crisis Continuity

- Beneficiary retains wallet or printed DID for:
    - Education enrollment
    - Job registration
    - Medical follow-up
- Optional biometric recovery using ZK-based face vector or fingerprint hash.

### Humanitarian UX Design

- Offline-first design: PWA + QR fallback
- SMS-based notification flows
- Local language support for VC display + claim process

## 9. 📋 Implementation Checklist

| Task | Tools / Outcome |
|---|---|
| Define identity + issuance flow | JSON-LD VC + hashing script |
| Anchor credentials | CIP-68 NFT w/ `vc_hash` |
| Build disbursement contract | Aiken (time-bound + hash check) |
| UI for NGO agents | Lucid Evolution + React, PWA support |
| Simulate testnet flow | Mint token, call TX, validate payout |
| Log results + docs | GitHub, GraphQL TX explorer |

## 📚 Reference Examples & Learning Resources

### 🔐 Vertical 2: Digital Identity

**GitHub Repositories:**

### Vertical 2: Digital Identity

- **Lucid Evolution** A lightweight SDK for creating CIP-68 NFTs and interacting with Cardano assets, facilitating decentralized identity solutions: https://github.com/Anastasia-Labs/lucid-evolution
- **CIP 68, Datum Metadata Standard** https://cips.cardano.org/cip/CIP-68

## 🌿 Vertical 4: Climate Action & Carbon Credits

**GitHub Repositories:**

- **BlockCarbon Market**: Development of a hybrid marketplace for Cardano carbon credit tokenization, including token minting, swapping, and retirement mechanisms.
  🔗 https://github.com/BlockCarbon/market

- **Carbon Credits Contract**: A blockchain-based network enabling commercial carbon credit trading in a public and secure setting.
  🔗 https://github.com/hrushikesh-choudhary/carbon-credits-contractGitHub

**Demonstrations:**

- **Hyperledger Climate Action and Accounting**: Implementations for climate action and accounting, including emissions calculations and carbon trading.
  🔗 https://github.com/hyperledger-labs/blockchain-carbon-accounting

## 📦 Vertical 5: Supply Chain Traceability

**GitHub Repositories:**

- **Supply Chain Library**: Integration of GS1 standards like GTIN and GLN into the Cardano blockchain to enhance product traceability and transparency.
  🔗 https://github.com/Agrow-Labs/supply-chain-lib

- **Supply Chain Traceability DApp**: A decentralized application using multichain blockchain and front-end technologies to trace products in a supply chain.
  🔗 https://github.com/neo-9981/Supply-Chain-Traceability-using-BlockchainGitHub

**Demonstrations:**

- **Cardano Foundation Traceability Solution**: An enterprise-grade solution for end-to-end supply chain visibility using the Cardano blockchain.
  🔗 https://cardanofoundation.org/solutions/traceabilityhttps://cardanofoundation.org

## 🎓 Vertical 6: Education & Credentialing

**GitHub Repositories:**

- **Verifiable Credentials**: A repository detailing requirements to earn verifiable credentials, focusing on digital identity and credentialing.
  🔗 https://github.com/rcl-coding/verifiable-credentials

- **Best of Digital Identity**: A curated list of open-source projects related to digital identity, including credentialing systems.
  🔗 https://github.com/jruizaranguren/best-of-digital-identity

## 🏥 Vertical 7: Health Records & Verifiable Credentials

**GitHub Repositories:**

- **VC ZK Health Records**: A project demonstrating the use of verifiable credentials and zero-knowledge proofs in managing health records.
  🔗 https://github.com/csalvador58/verifiable-credential-zk-health-record

**Demonstrations:**

- **Verifiable Credentials in Healthcare**: An exploration of how verifiable credentials can be applied in the healthcare industry for secure and private data sharing.
  🔗 https://github.com/csalvador58/verifiable-credential-zk-health-record