SDG BLOCKCHAIN ACCELERATOR

Prototype (PoC) Report

Atlas Ledger

1. Project Information

- **Project Name:** Atlas Ledger
- **Challenge & UNDP Office:** Reforestation projects with UNDP Burkina Faso
- **Document Version:** V 1.0

2. Project Overview

Atlas Ledger is a climate impact platform that connects reforestation projects with eco-conscious donors/contributers on the Cardano blockchain. The prototype implements a comprehensive full-stack decentralized application featuring:

- **Comprehensive Admin Dashboard:** Full administrative control panel with extensive management capabilities:
    - **User Management:** Complete user oversight with KYC **approval/rejection, wallet creation, and role assignment**
    - **Project Administration:** Review and approve/reject climate projects, manage project statuses and milestones
    - **Smart Contract Operations:** Deploy and monitor V6 milestone-based contracts, view all deployed contracts with status tracking
    - **Financial Management:** Treasury wallet management, fund user wallets, track all transactions with real-time Cardano blockchain status
    - **Transaction Monitoring:** Complete transaction history with Preprod testnet explorer integration and automatic status synchronization
    - **Milestone Control:** Trigger milestone payouts, reset projects for testing, manage milestone progressions
    - **System Health:** Monitor platform health, donation simulation status, and blockchain connectivity
    - **Wallet Operations:** Create donor/project owner wallets, check live balances, fund wallets from treasury

- **Climate Project Management**: Project owners can create and manage reforestation projects with milestone-based funding
- **Donor/Contributor Portal:** Donors/contributors can browse, evaluate, and fund climate projects
- **Milestone-Based Payouts**: Smart contract-powered milestone verification and automated payments
- **Impact Assessment Chatbot:** Interactive conversational flow to collect project data and calculate carbon sequestration metrics
- **Carbon Credit Calculations:** Comprehensive carbon sequestration modeling with species-specific rates, survival calculations, and buffer/leakage deductions
- **Multi-Impact Data Display:** Climate, social, biodiversity, circular economy, and avoided emissions impact calculations and visualization
- **Role-Based Access Control:** Admin, project owner, and donor role management
- **Cardano Integration:** Full blockchain integration with custodial wallets and on-chain transactions
- **KYC/Compliance**: User verification and compliance features
- **Real-Time Notifications:** Project updates and transaction notifications

3. Repository Structure

```
/atlas-ledger
├── /src                    → Next.js 15 frontend application
│   ├── /app                → App Router pages and layouts
│   ├── /components         → Reusable UI components (shadcn/ui)
│   ├── /contexts           → React contexts for state management
│   ├── /hooks              → Custom React hooks
│   ├── /lib                → Utility libraries and configurations
│   ├── /services           → API service layer
│   ├── /store              → Zustand state management
│   ├── /types              → TypeScript type definitions
│   └── /utils              → Helper utilities
├── /backend                → NestJS backend API
│   ├── /src                → Source code
│   │   ├── /admin          → Admin management endpoints
│   │   ├── /auth           → Authentication & JWT handling
│   │   ├── /contracts      → Cardano contract services
│   │   ├── /projects       → Project CRUD operations
│   │   ├── /users          → User management & KYC
│   │   ├── /wallets        → Cardano wallet management
│   │   └── /test           → Integration test suites
├── /milestone-contracts    → Aiken smart contracts
│   ├── /validators         → Aiken validator source code
│   ├── aiken.toml          → Project configuration
│   ├── plutus.json         → Compiled contract artifacts
│   └── /build              → Build artifacts
├── /docs                   → Documentation and guides
└── /public                 → Static frontend assets

The testfile for the contract is in the validators directory (aiken check)
```

4. Build Instructions

```
# Navigate to contracts directory
cd milestone-contracts

# Compile Aiken contracts
aiken build
```

Environment Requirements:

- Aiken CLI v1.1.19+e525483
- Plutus v3 compatibility
- Dependencies: aiken-lang/stdlib v2

Compatible Versions:

- Aiken CLI: v1.1.19+
- Cardano node: Compatible with Preprod testnet
- Plutus version: v3

5. Test Instructions

```
# Run Aiken unit tests
cd milestone-contracts
aiken check

integration tests are done on the preprod net already - no emu
```

Key Test Cases:

- **v6_happy_multiple_inputs:** Tests handling multiple UTXOs in a single transaction
- **v6_happy_with_extra_outputs:** Validates contract works with additional transaction outputs
- **v6_happy_reached_up_to_higher:** Confirms milestone gating works when reached_up_to > milestone_ix
- **v6_fail_missing_admin_signature:** Security test - rejects transactions without required admin signature
- **v6_fail_underpay_beneficiary:** Validates contract prevents insufficient payments to beneficiaries
- **v6_fail_change_back_to_script:** Prevents funds from being returned to the contract address
- **v6_fail_wrong_beneficiary_address:** Rejects payments to incorrect beneficiary addresses
- **v6_fail_reached_up_to_too_low:** Enforces milestone index validation (reached_up_to >= milestone_ix)
- **v6_fail_beneficiary_is_script_addr:** Security guard preventing beneficiary == contract address
- **v6_happy_split_payment_to_beneficiary:** Tests multiple outputs to same beneficiary address
- **v6_happy_multi_input_exact_total:** Complex scenario with multiple inputs paying exact total

```
 Compiling atlas-ledger/milestone-contracts 0.0.0 (.)
 Resolving atlas-ledger/milestone-contracts
   Fetched 1 package in 0.01s from cache
 Compiling aiken-lang/stdlib v2 (./build/packages/aiken-lang-stdlib)
Collecting all tests scenarios across all modules
   Testing ...

┌─ oracle_payout_simple_v6_admin ──────────
│ PASS [mem:    200.0, cpu:   16.1 K] smoke
└──────────────── 1 tests | 1 passed | 0 failed


┌─ oracle_payout_simple_v6_admin_test ─────────────────────
│ PASS [mem: 159.70 K, cpu:  57.00 M] v6_admin_payout_smoke
│ PASS [mem: 180.04 K, cpu:  65.61 M] v6_happy_with_extra_outputs
│ PASS [mem: 186.41 K, cpu:  68.23 M] v6_happy_multiple_inputs
│ PASS [mem: 159.70 K, cpu:  57.00 M] v6_happy_reached_up_to_higher
│ PASS [mem:  85.79 K, cpu:  28.09 M] v6_fail_missing_admin_signature
│ PASS [mem: 149.09 K, cpu:  50.74 M] v6_fail_underpay_beneficiary
│ PASS [mem: 175.52 K, cpu:  62.69 M] v6_fail_change_back_to_script
│ PASS [mem:  79.82 K, cpu:  25.50 M] v6_fail_wrong_own_ref
│ PASS [mem:  64.29 K, cpu:  18.85 M] v6_fail_no_datum
│ PASS [mem: 130.85 K, cpu:  45.00 M] v6_fail_wrong_beneficiary_address
│ PASS [mem:  80.35 K, cpu:  25.87 M] v6_fail_reached_up_to_too_low
│ PASS [mem: 197.12 K, cpu:  71.08 M] v6_happy_split_payment_to_beneficiary
│ PASS [mem: 196.12 K, cpu:  66.79 M] v6_fail_multi_input_underpay_total
│ PASS [mem: 334.34 K, cpu: 124.32 M] v6_happy_multi_input_exact_total
│ PASS [mem: 155.11 K, cpu:  54.15 M] v6_fail_beneficiary_is_script_addr
└──────────────────────────────────── 15 tests | 15 passed | 0 failed


   Summary 16 checks, 0 errors, 0 warnings
```

6. Deployment Instructions

Transaction Building Process

Milestone-Based Deployment: The system creates multiple UTXOs for milestone-gated payouts using Lucid-Evolution:

```
// Complete milestone deployment process with immediate payout + contract
locking
async deployV6ContractWithMilestones(params: V6MilestoneParams) {
  // Calculate milestone amounts - always 4 milestones total
  const fixedMilestoneCount = 4;
  const milestoneAmounts = this.v6Transaction.calculateMilestoneAmounts(
    params.amount,
    fixedMilestoneCount
  );

  // SPLIT: Milestone 0 (immediate) vs Contract milestones (1,2,3)
  const milestone0Amount = milestoneAmounts[0];
 // 25% immediate payout
  const contractMilestoneAmounts = milestoneAmounts.slice(1);
 // 75% locked in contract
  const contractTotalAmount = contractMilestoneAmounts.reduce((a, b) => a +
b, 0n);

  // Create different datum for each contract milestone (1, 2, 3)
  const projectIdHex = Buffer.from(params.projectId,
'utf8').toString('hex');
  const milestoneDatums: string[] = [];

  for (let i = 1; i <= 3; i++) { // Create datums for milestones 1, 2, 3
    const datumCbor = this.v6Datum.buildDatumV6({
      projectIdHex,
      beneficiaryAddr: params.beneficiaryAddr,
      adminVkhHex: adminVkhHex,
      milestoneIx: i,  // Each UTXO gets unique milestone index
    });
    milestoneDatums.push(datumCbor);
  }

  // BUILD TRANSACTION: Immediate payout + contract deposits
  const tx = await this.v6Transaction.buildMilestoneDeploymentTx({
    lucid,
```

```
    contractAddress,
    beneficiaryAddr: params.beneficiaryAddr,
    milestone0Amount,        // Direct to beneficiary (immediate)
    contractTotalAmount,     // Total for contract milestones
    contractMilestoneAmounts, // Individual amounts [M1, M2, M3]
    milestoneDatums,         // Individual datums [D1, D2, D3]
    metadata,
  });

  // Sign and submit
  const signed = await tx.sign.withWallet().complete();
  const txHash = await signed.submit();

  return { txHash, milestone0Amount, contractTotalAmount };
}

// Transaction building implementation
async buildMilestoneDeploymentTx(params: V6MilestoneDeploymentTxParams) {
  const {
    lucid, contractAddress, beneficiaryAddr,
    milestone0Amount, contractMilestoneAmounts, milestoneDatums
  } = params;

  let txb = lucid.newTx();

  // OUTPUT 1: Immediate payout to beneficiary (milestone 0 - 25%)
  console.log(`Immediate payout: ${milestone0Amount} lovelace to
${beneficiaryAddr}`);
  txb = txb.pay.ToAddress(beneficiaryAddr, { lovelace: milestone0Amount });

  // OUTPUTS 2-4: Lock remaining milestones in contract (75% total)
  for (let i = 0; i < contractMilestoneAmounts.length; i++) {
    const amount = contractMilestoneAmounts[i];
    const datumCbor = milestoneDatums[i];
    const milestoneIx = i + 1; // Milestone indices 1, 2, 3

    console.log(`Milestone ${milestoneIx}: ${amount} lovelace locked in
contract`);
    txb = txb.pay.ToContract(
      contractAddress,
      { kind: 'inline', value: datumCbor },
 // Inline datum with milestone_ix
      { lovelace: amount }
```

```
    );
  }

  // Complete with metadata and coin selection
  const tx = await txb
    .attachMetadata(674, this.meta674Base(metadata))
    .complete({ coinSelection: true });

  return tx;
}

// Milestone amount calculation (even distribution)
calculateMilestoneAmounts(totalAmount: bigint, milestoneCount: number):
bigint[] {
  const milestoneAmounts: bigint[] = [];
  const baseAmount = totalAmount / BigInt(milestoneCount);
  const remainder = totalAmount % BigInt(milestoneCount);

  for (let i = 0; i < milestoneCount; i++) {
    // Distribute remainder across first milestones
    const amount = baseAmount + (i < remainder ? 1n : 0n);
    milestoneAmounts.push(amount);
  }

  return milestoneAmounts; // [M0, M1, M2, M3] - each ~25%
}
```

Milestone Payout Process

Admin-Triggered Milestone Payouts: When a project milestone is completed, admins can trigger payouts:

```
// Backend builds payout transaction
// Spends contract UTXOs with milestone_ix <= reached_milestone

// Build payout transaction
lucid.newTx()
  .attach.SpendingValidator(validator)
  .collectFrom([utxo1, utxo2, utxo3], redeemer) # Collect milestone UTXOs
  .pay.ToAddress(beneficiaryAddr, { lovelace: totalAmount })
  .addSignerKey(adminVkhHex) # Required admin signature
  .complete({ changeAddress: adminWalletAddress })

// Redeemer: Payout(reached_up_to: milestone_number)
// Validation: reached_up_to >= datum.milestone_ix for each UTXO
```

7. Testnet / Emulator Results

Transaction ID	Type	Status	CPU	Memory	Notes
83730d50102cb51a7b8a9cb2b0cd9a5e94cfdaf2992a303fc874bf83b0757949	Milestone 1 (index) pay out - 1 donor	Success	55M	127k	Validator passed all checks Admin signature and milestone index verified successfully
077ae2d79d1de8113400c619a8e16c351c9e47c24a84c5a4bba63a00c6e7d537	Milestone 2 (index) payout - 2 donors	Success	avg for every donor 67M	avg for every donor 155k	Validator passed all checks Admin signature and milestone index verified successfully
a79c2f27e3646d935741f8f80311fa213d1a5debfd9e62838359864386965926	Milestone 3 (index) payout - 3 donors	Success	avg for every donor 81M	avg for every donor 186k	Validator passed all checks Admin signature and milestone index verified successfully

Contract Addresses (HEX):

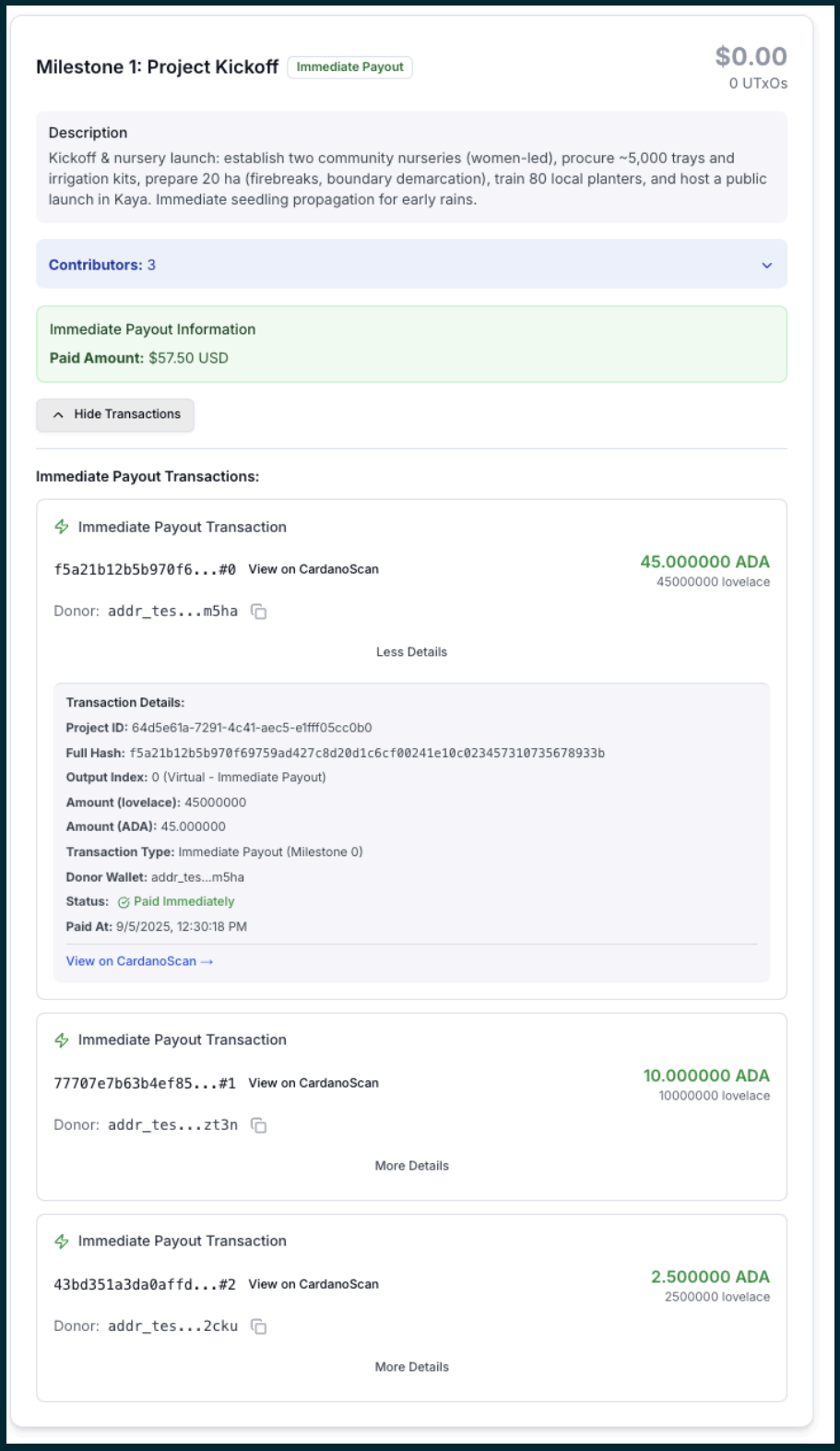
- 106c8c2497ab7173c16a389f9948aa798a22d2d476e1785b0706dfa7af534b62ad660488b3c51fd524faeed907f5612f6682b37f7ae48393d8 (used for the table data)
- 108c22570eeb82089ada1539e4709841e3d3b740fde3577ad700080e90b19bd26b60b052cd99d01cac3cdbf3ed4531d6483d019c6e49998ffd

8. Dependencies & Environment

- **Aiken/Smart Contracts**
    - Aiken CLI version: v1.1.19+e525483
    - Plutus version: v3
    - Dependencies: aiken-lang/stdlib v2
    - Testnet: Cardano Preprod
- **Backend Dependencies**
    - Framework: NestJS 11.1.6
    - Database: PostgreSQL with TypeORM 0.3.25
    - Blockchain: @lucid-evolution/lucid 0.4.29, @blockfrost/blockfrost-js 6.0.0
    - Authentication: JWT with passport
    - Runtime: Node.js with TypeScript 5.9.2
- **Frontend Dependencies**
    - Framework: Next.js 15.1.7 with React 19.0.0
    - UI: Tailwind CSS + shadcn/ui components
    - State Management: Zustand 5.0.3
    - Testing: Vitest 3.2.4 + React Testing Library
    - Blockchain: @lucid-evolution/lucid 0.4.29
- **Off-chain Services**
    - Blockfrost API: Cardano blockchain queries
    - Firebase: Optional file storage integration

9. Demo / Walkthrough

Milestone Payout Screenshots:



Milestone 2: Implementation Phase 1 Paid Out

$0.00
0 UTxOs

Description

Implementation Phase 1: plant ~25,000 trees across ~22.5 ha (50% of total). Install basic monitoring (GPS plot mapping, photo-points), create water-harvesting micro-basins on slopes, and form 4 community maintenance brigades with tools & safety gear.

Contributors: 1

Top Contributions:

Alice Johnson $2.50

Payout Information

Paid Amount: $2.50

Transaction: View

Paid At: 9/5/2025 12:08:59

Hide Transactions

Milestone Payout Transactions:

Milestone Payout Transaction Paid Out

`83730d50102cb51a...#0` View

2.500000 ADA
2500000 lovelace

Donor: addr_tes...h8pr

Less Details

Transaction Details:

Project ID: 64d5e61a-7291-4c41-aec5-e1fff05cc0b0

Full Hash: 83730d50102cb51a7b8a9cb2b0cd9a5e94cfdaf2992a303fc874bf83b0757949

Output Index: 0 (Contract Milestone Payout)

Amount (lovelace): 2500000

Amount (ADA): 2.500000

Transaction Type: Milestone Payout (Milestone 2)

Donor Wallet: addr_tes...h8pr

Status: Paid Out

Paid At: 9/5/2025 12:08:59

View on CardanoScan →

Milestone 3: Implementation Phase 2 Paid Out

$0.00
0 UTxOs

Description
Implementation Phase 2: complete remaining ~25,000 trees on the final ~22.5 ha. Expand monitoring (sample survival surveys, ODK records, drone overpass), deploy soil-moisture pits in each plot, and run refresher training on pest management and FMNR.

Contributors: 2

Top Contributions:

Markus Doppelreiter	$5.00
Alice Johnson	$2.50

Payout Information

Paid Amount: $7.50
Transaction: View
Paid At: 9/5/2025 12:11:52

Hide Transactions

Milestone Payout Transactions:

Milestone Payout Transaction — Paid Out

`077ae2d79d1de811...#0` View

7.500000 ADA
7500000 lovelace

Donor: addr_tes...h8pr

Less Details

Transaction Details:

Project ID: 64d5e61a-7291-4c41-aec5-e1fff05cc0b0
Full Hash: 077ae2d79d1de8113400c619a8e16c351c9e47c24a84c5a4bba63a00c6e7d537
Output Index: 0 (Contract Milestone Payout)
Amount (lovelace): 7500000
Amount (ADA): 7.500000
Transaction Type: Milestone Payout (Milestone 3)
Donor Wallet: addr_tes...h8pr
Status: Paid Out
Paid At: 9/5/2025 12:11:52

View on CardanoScan →

Milestone 4: Project Completion

Paid Out

$0.00
0 UTxOs

Description

Completion & Monitoring: conduct year-1 survival audit (>85% target) and gap-filling, finalize monitoring system (quarterly surveys + satellite snapshots), publish impact report (jobs, women's income from shea/moringa), and hand over maintenance plan to local committees.

Contributors: 3

Top Contributions:

Bob Chen	$15.00
Markus Doppelreiter	$5.00
Alice Johnson	$2.50

Payout Information

Paid Amount: $22.50
Transaction: View
Paid At: 9/5/2025 12:31:37

Hide Transactions

Milestone Payout Transactions:

Milestone Payout Transaction — Paid Out

a79c2f27e3646d93...#0 View

22.500000 ADA
22500000 lovelace

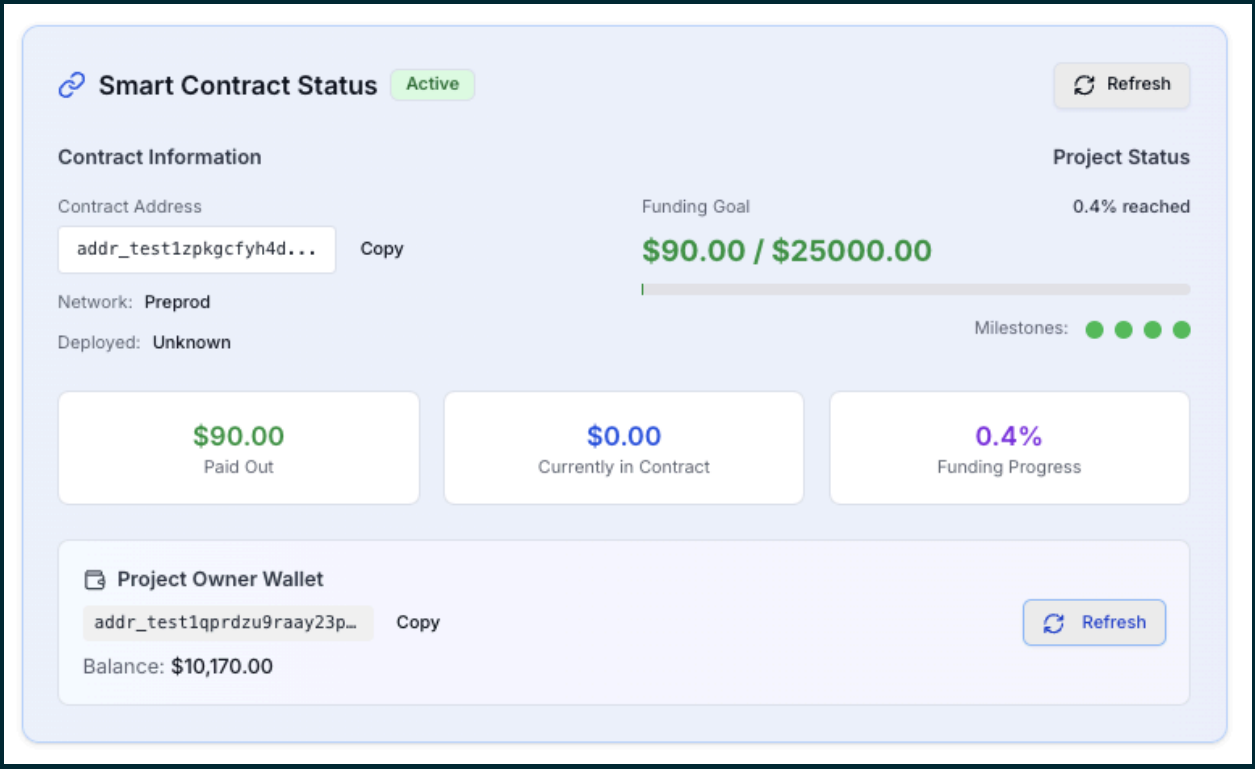
Donor: addr_tes...h8pr

Less Details

Transaction Details:

Project ID: 64d5e61a-7291-4c41-aec5-e1fff05cc0b0
Full Hash: a79c2f27e3646d935741f8f80311fa213d1a5debfd9e628383598643869659 26
Output Index: 0 (Contract Milestone Payout)
Amount (lovelace): 22500000
Amount (ADA): 22.500000
Transaction Type: Milestone Payout (Milestone 4)
Donor Wallet: addr_tes...h8pr
Status: Paid Out
Paid At: 9/5/2025 12:31:37

View on CardanoScan →



Walktrough Videos (direct download from G-Drive):

01 - Project Creation (7:00 min ~560MB)

02 - Admin Dashboard - Donation - Milestone Payout (5:40 ~500MB)

03 - Project Display (0:40 ~90MB)

10. Remaining Issues / Next Steps

1. **Critical Scaling Issue - Many-Donors Transactions:**
    - **Problem**: Current V6 validator scans entire transaction for each donor UTXO, causing CPU/memory to grow quadratically with donor count
    - **Impact**: Transactions with many donors can approach Cardano's per-transaction CPU limits
    - **Proposed Solutions:**
        - On-chain: Add consolidation redeemer branch to merge multiple donor UTXOs into single script UTXO before payout
        - Off-chain: Implement batching (≤20 donors/tx) with EvaluateTx validation and dynamic batch sizing based on execution units
        - Monitoring: Pre-flight transaction cost estimation to prevent failures
2. **Performance Optimizations:**
    - Monitor real-world execution costs and efficiency on Preprod testnet
    - Optimize batch transaction handling for large milestone collections
    - Implement UTXO consolidation strategies for improved efficiency
    - Cache optimization for frequent blockchain queries
3. **Security Enhancements:**
    - Conduct comprehensive security audit of V6 validator logic
    - Implement additional validation for edge cases in multi-input scenarios
    - Enhance admin signature verification with multi-sig support (future V7)
    - Strengthen input validation across all contract entry points
4. **Mainnet Preparation:**
    - Stress testing with higher transaction volumes on Preprod
    - Documentation completion for production deployment procedures
    - Monitoring and alerting system setup for mainnet operations
    - Backup and disaster recovery procedures for contract management

5. **Feature Enhancements:**
    - Enhanced project media and documentation support
    - Real AI Agent to help the Project Owner to create a compelling Donation Projects
    - Text suggestions for Project Descriptions
    - Improved Dashboards for all user groups
    - Design System which fits the Use Case
    - Mobile-responsive optimizations
6. **Database & Infrastructure Scalability:**
    - Optimize database queries for large project datasets
    - Implement caching layers for frequently accessed data
    - Consider layer-2 solutions for high-frequency transactions