SDG BLOCKCHAIN ACCELERATOR

Prototype (PoC) Report – Genius Tags

1. Project Information

- **Project Name:** _ClimateAid

- **Challenge & UNDP Office:** UNDP Malawi
- **Document Version:**1

2. Project Overview

This prototype implements a decentralized safe data sharing and coordination with beneficiary deduplication on Cardano using Aiken smart contracts. The system enables organizations to register, manage users with granular permissions, create and share projects, and commit verifiable transactions including beneficiary UID generation and duplication status to the blockchain. The platform provides a REST API service that acts as a bridge between user applications and the Cardano blockchain, handling wallet management, transaction processing, and blockchain state queries.

3. Repository Structure

(Outline how your code and related files are organized.)

Suggested Structure:

```
GeniusChain/
├── contract/                    # Aiken smart contracts
│   ├── validators/
│   │   ├── genius-chain.ak     # Main contract (1,692 lines)
│   │   ├── comprehensive-tests.ak
│   │   ├── simple-tests.ak
│   │   └── deploy.ak
│   ├── aiken.toml              # Contract configuration
│   ├── deployment-info.json    # Deployment metadata
│   └── test-*.ts               # Integration tests
├── service/                     # Cardano blockchain service
│   ├── src/
│   │   ├── controllers/        # API endpoints
│   │   ├── services/           # Business logic
│   │   ├── middleware/         # Express middleware
│   │   └── routes/             # API routes
│   ├── package.json
│   └── README.md
├── web/                        # Vue.js web application
├── mobile/                    # React Native mobile app
├── client/                    # TypeScript client library
├── server/                    # Backend server
└── shared/                    # Shared utilities and types
```

4. Build Instructions

```
Smart Contract Compilation:
# Navigate to contract directory
cd contract

# Install dependencies
npm install

# Compile Aiken contracts
aiken build

# Run tests
aiken check
Service Build:
# Navigate to service directory
cd service

# Install dependencies
npm install

# Build TypeScript
npm run build

# Start development server
npm run dev
Environment Variables Required:
# Blockfrost API configuration
BLOCKFROST_PROJECT_ID=your_blockfrost_project_id
NETWORK_ID=0  # 0 for testnet, 1 for mainnet

# Service configuration
PORT=3001
CORS_ORIGIN=http://localhost:3000
```

```
NODE_ENV=development
Compatible Versions:
  •   Aiken CLI: v1.1.19+
  •   Node.js: v18.0.0+
  •   Cardano-node: Compatible with MeshSDK v1.5.11+
```

5. Test Instructions

(Explain how to run unit and integration tests.)

Example:

```
Unit Tests:
# Run Aiken contract tests
cd contract
aiken check

# Run TypeScript integration tests
npm run test:contract
npm run test:blockchain
npm run test:all
Integration Tests:
# Test blockchain interactions
npm run test:onchain

# Test service API endpoints
cd service
npm test
```

Expected Test Output:

- Contract validation tests pass
- Transaction execution tests succeed
- API endpoint tests return expected responses
- Edge cases (invalid permissions, non-existent projects) properly rejected

Edge Cases Tested:

- Invalid organization addresses
- Unauthorized user actions
- Non-existent project access
- Batch transaction limits
- Security validation (zero addresses rejected)

6. Deployment Instructions

```
Smart Contract Deployment:
# Build transaction
cardano-cli transaction build \
  --testnet-magic 2 \
  --tx-in <UTXO> \
  --tx-out <SCRIPT_ADDRESS>+2000000 \
  --change-address <CHANGE_ADDR> \
  --out-file tx.raw

# Sign transaction
cardano-cli transaction sign \
  --tx-body-file tx.raw \
  --signing-key-file payment.skey \
  --testnet-magic 2 \
  --out-file tx.signed

# Submit transaction
cardano-cli transaction submit \
  --testnet-magic 2 \
  --tx-file tx.signed
Service Deployment:
# Build Docker image
docker build -t genius-chain-service .

# Run with Docker Compose
docker-compose up --build
```

```
# Or deploy to Kubernetes
kubectl apply -f k8s/
```

Prerequisites:

- Testnet ADA for transaction fees
- Blockfrost API key
- Cardano node access (or use Blockfrost)
- Docker and Docker Compose (for containerized deployment)

7. Testnet / Emulator Results

Deployment Information:

- Network: Cardano Testnet
- Script Address: addr_test1wztsa3xlr60hhv35d5ek2afq3kml7h9ku0j9pkzuswrfcfce9ln58
- Transaction Hash: e51fab47be03f4b4c081d8891e19afbb2b49b7bc7e6b7c2d4f88ff339e1a9576
- Contract Amount: 2,000,000 lovelace

8. Dependencies & Environment

Core Dependencies:

- Aiken CLI: v1.1.19+e525483
- Node.js: v20.19.2
- MeshSDK: v1.5.11-beta.4
- Express: v4.18.2
- TypeScript: v5.3.0

Cardano Network:

- Testnet: Cardano Preview/Preprod

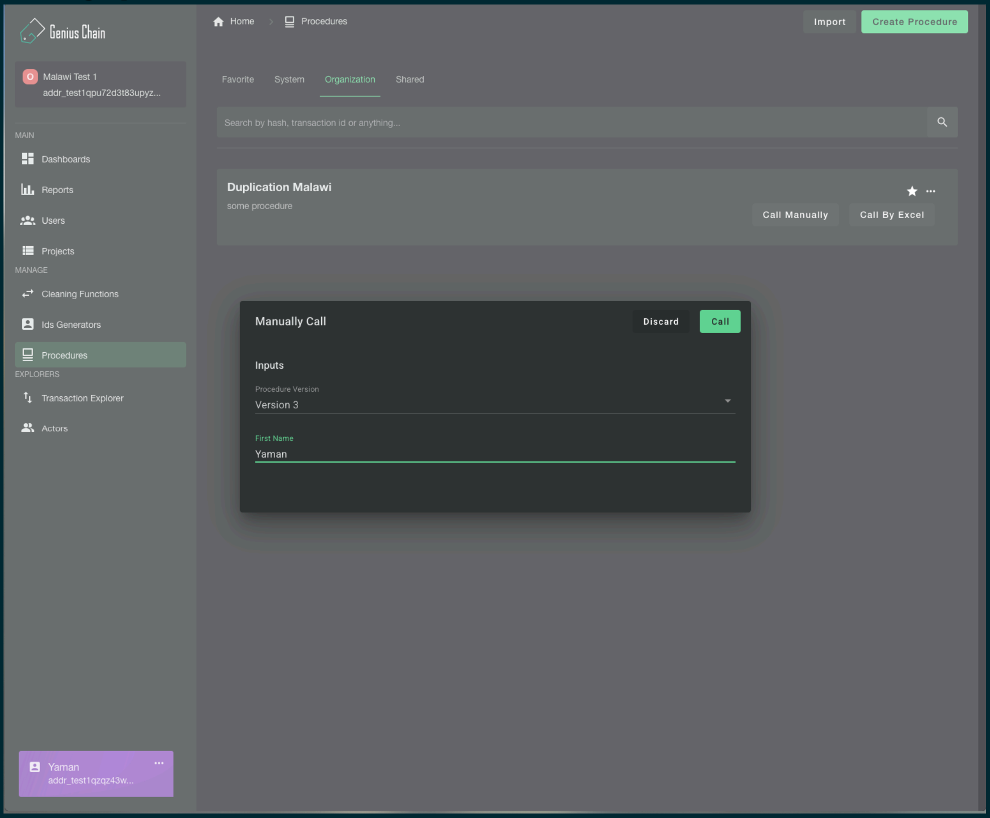
- Blockfrost Provider for network access
- MeshSDK for wallet and transaction management

Libraries & SDKs:

- @meshsdk/core: Cardano blockchain interactions
- @meshsdk/core-cst: Cardano serialization
- bip39: Mnemonic generation
- express-rate-limit: API rate limiting

9. Demo / Walkthrough

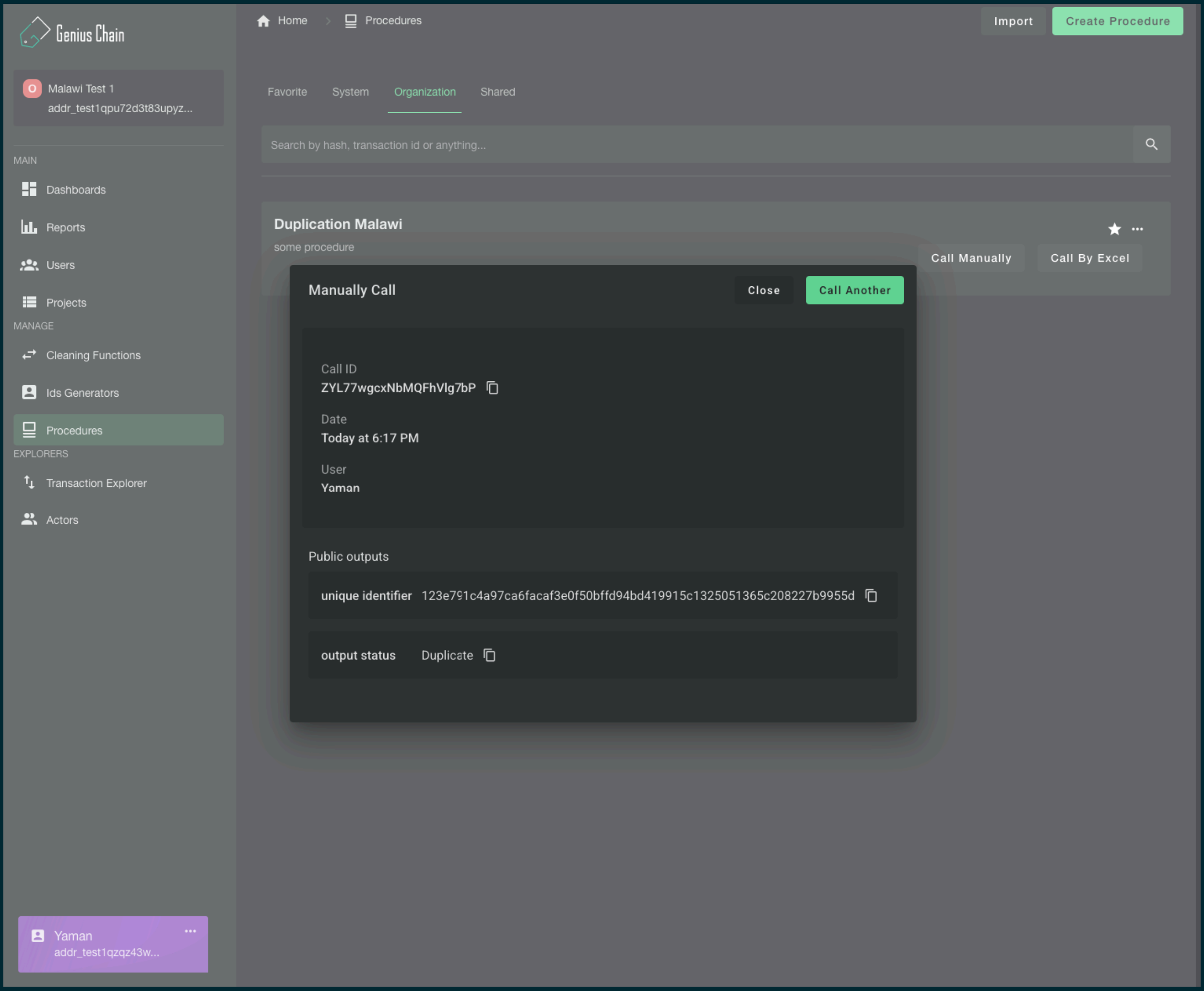
- Running a procedure



Procedure Version: Version 3

First Name: Yaman

- Getting a result



Duplication Malawi

some procedure

Manually Call

Call ID
ZYL77wgcxNbMQFhVlg7bP

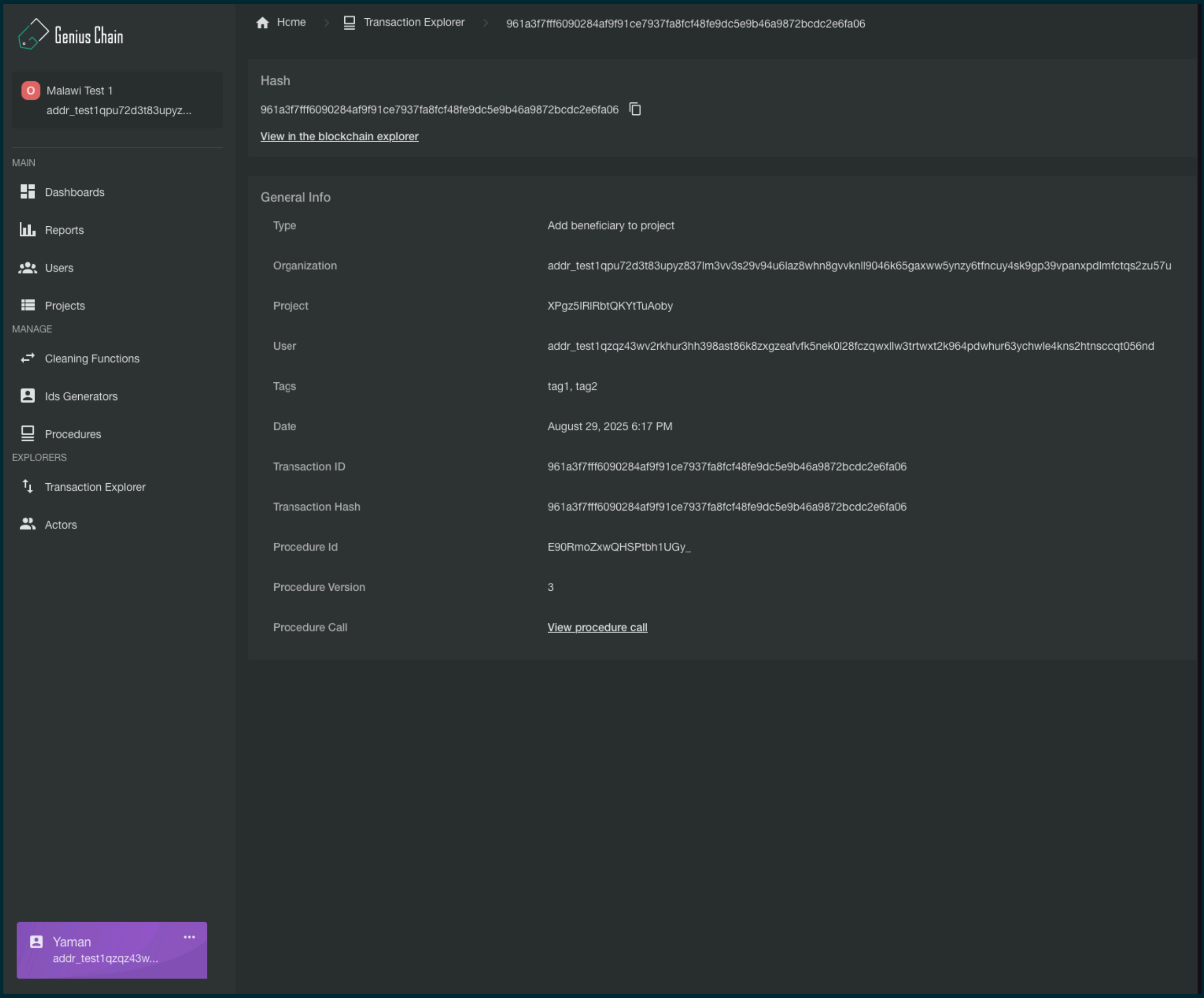
Date
Today at 6:17 PM

User
Yaman

Public outputs

unique identifier	123e791c4a97ca6facaf3e0f50bffd94bd419915c1325051365c208227b9955d
output status	Duplicate

- Checking a transaction



Hash

961a3f7fff6090284af9f91ce7937fa8fcf48fe9dc5e9b46a9872bcdc2e6fa06

General Info

Type	Add beneficiary to project
Organization	addr_test1qpu72d3t83upyz837lm3vv3s29v94u6laz8whn8gvvknll9046k65gaxww5ynzy6tfncuy4sk9gp39vpanxpdlmfctqs2zu57u
Project	XPgz5IRlRbtQKYtTuAoby
User	addr_test1qzqz43wv2rkhur3hh398ast86k8zxgzeafvfk5nek0l28fczqwxllw3trtwxt2k964pdwhur63ychwle4kns2htnsccqt056nd
Tags	tag1, tag2
Date	August 29, 2025 6:17 PM
Transaction ID	961a3f7fff6090284af9f91ce7937fa8fcf48fe9dc5e9b46a9872bcdc2e6fa06
Transaction Hash	961a3f7fff6090284af9f91ce7937fa8fcf48fe9dc5e9b46a9872bcdc2e6fa06
Procedure Id	E90RmoZxwQHSPtbh1UGy_
Procedure Version	3
Procedure Call	View procedure call

https://preview.cardanoscan.io/transaction/961a3f7fff6090284af9f91ce7937fa8fcf48fe9dc5e9b46a9872bcdc2e6fa06

API Endpoints Available:

- `POST /api/v1/wallet/create` - Create new wallet
- `POST /api/v1/wallet/login` - Login with existing wallet
- `GET /api/v1/wallet/balance` - Get wallet balance
- `POST /api/v1/transaction/call` - Execute blockchain transaction
- `GET /api/v1/transaction/status/:txHash` - Check transaction status

Example API Usage:

```
# Create wallet
curl -X POST http://localhost:3001/api/v1/wallet/create

# Execute transaction
curl -X POST http://localhost:3001/api/v1/transaction/call \
  -H "Content-Type: application/json" \
  -d '{
    "function": "CreateProject",
    "payload": {
      "project_hash": "abc123",
      "status": "active"
    }
  }'
```

Delivered DEMO: https://youtu.be/m2wakC1js4E

10. Remaining Issues / Next Steps

Current Issues:

- *Need to keep chain logic on front-end instead of having a service*

Planned Improvements:

- *Update legacy code to use newer versions of nodejs, vue, etc.*
- *Move all chain logic for calling transaction to run the frontend*

Next Steps:

1. *Complete comprehensive testing suite*
2. *Performance optimization*
3. *Security audit*
4. *Mainnet deployment preparation*

Technical Debt:

- *Refactor contract code for better modularity*
- *Implement comprehensive logging*
- *Add automated testing pipeline*
- *Optimize gas usage for complex operations*
- *Enhance error messages and debugging*