SDG BLOCKCHAIN ACCELERATOR

Prototype (PoC) Report

1. Project Information

- **Project Name:** Unicorn
- **Challenge & UNDP Office:** UNDP OP CCIT
- **Document Version:** 1

2. Project Overview

(Briefly describe what the prototype does and its main features.)

Example:

> We are building a platform to tokenize and co-finance renewable energy infrastructure vetted by UNDP OP-CCIT, starting with a pilot in Afghanistan. The platform enables global users to crowdinvest in projects like solar PV, microgrids, and small hydro through ADA or USDA stablecoin. Tokens represent fractional participation, while revenues from energy sales flow back via smart contracts. This model brings transparency, impact, and global accessibility to renewable energy finance.

3. Repository Structure

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

Suggested Structure:

```
/prototype
    /contracts   → Aiken smart contract source code
    /tests       → Unit and integration tests
    /scripts     → Deployment scripts and CLI commands
    /docs        → README.md, deployment guide, screenshots
```

4. Build Instructions

Prerequisites

• Node.js: v20 or v22 (recommended: v22).

Note: @utxorpc/spec (pulled by Mesh) requires Node ≥20 and Vercel will deprecate Node 18.

• Package manager: pnpm or yarn or npm

• Next.js: App Router (Next 13/14) or Pages (works with both)

• Cardano wallet extension: Yoroi/Nami/Eternl installed in the browser (for local testing).

Required ENV Variables

Create a .env.local in the project root:

```
# Blockfrost (use the right key per network)

NEXT_PUBLIC_CARDANO_BLOCKFROST_PROJECT_ID=<your_mainnet_key>

NEXT_PUBLIC_CARDANO_BLOCKFROST_PREPROD_PROJECT_ID=<your_preprod_key>

# Donation destination (pick matching network at runtime)

NEXT_PUBLIC_CARDANO_RECIPIENT_MAINNET=addr1...

NEXT_PUBLIC_CARDANO_RECIPIENT_PREPROD=addr_test1...

# Optional: GraphQL backend for recording donations

NEXT_PUBLIC_GQL_ENDPOINT=https://impact-graph.serve.giveth.io/graphql

NEXT_PUBLIC_AUTH_VERSION=2

# JWT, if required for local manual calls

NEXT_PUBLIC_AUTH_BEARER=<optional_jwt>
```

If you display remote wallet icons, add allowed domains in next.config.js (images.remotePatterns). (You cannot whitelist chrome-extension://; use a fallback icon for that case.)

Install Dependencies

Pick one:

```
# pnpm (fastest)
```

```
pnpm install

# yarn

yarn install

# npm

npm install
```

Temporarily install without changing package.json:

npm install --no-save <pkg> or yarn add --no-lockfile <pkg>

Local Development

Node Version 22

```
# Dev
pnpm dev
# or
yarn dev
# or
npm run dev
```

Open http://localhost:3000 and connect a Cardano wallet extension.

Production Build

Node Version 22

```
pnpm build && pnpm start
# or
yarn build && yarn start
# or
npm run build && npm start
```

Vercel Deployment Notes

- In Project Settings → Build & Development → Node.js Version, set 22.x.
- If you see:

```
error @utxorpc/spec: Expected node >=20.0.0
```

you’re on the wrong Node version—switch to 22.

- If you need to install without mutating package.json on CI:

npm ci (strict) or npm install --no-save.

Network Switching (Mainnet vs Preprod)

At runtime, select the correct recipient address based on the connected wallet:

```
const netId = await wallet.getNetworkId(); // 1 = Mainnet, 0 = Testnets
const recipient =
  netId === 1
    ? process.env.NEXT_PUBLIC_CARDANO_RECIPIENT_MAINNET!
    : process.env.NEXT_PUBLIC_CARDANO_RECIPIENT_PREPROD!;
```

Ensure your Blockfrost key matches that network (mainnet vs preprod).

Postman / GraphQL (optional)

For recording donations:

- Endpoint: NEXT_PUBLIC_GQL_ENDPOINT
- Headers:
- Content-Type: application/json
- authversion: 2
- Authorization: Bearer <JWT> (if required)
- Mutation: createCardanoDonation(...) with the variables your backend expects.

Common Troubleshooting

- Wrong network error (WrongNetwork Testnet … [Addr Mainnet …]):

Wallet network and recipient address mismatch. Use the switch above.

- Output too small (BabbageOutputTooSmallUTxO):

When sending tokens, attach ~1.5 ADA to the same output.

- BigInt RangeError:

Always convert human amounts using toUnits(amount, decimals) before building the tx.

- Icons not loading in Next Image:

Fallback for chrome-extension:// icons; only allow https domains in next.config.js.

5. Test Instructions

A) Setup & Preconditions

1. *Install a Cardano wallet extension (Yoroi/Nami/Eternl).*
2. *Select test network in the wallet (Preprod recommended for testing).*
3. *Fund the wallet with test ADA (TADA) from a Preprod faucet.*
4. *Configure env:*

- *NEXT_PUBLIC_CARDANO_BLOCKFROST_PREPROD_PROJECT_ID=<key>*
- *NEXT_PUBLIC_CARDANO_RECIPIENT_PREPROD=addr_test1...*

5. *Run app locally: npm run dev (or yarn dev / pnpm dev), open http://localhost:3000.*

B) Smoke Test: Connect Wallet

1. *Open the app → click Connect Wallet.*
2. *Choose Yoroi (or Nami/Eternl).*
3. *Expected: app shows Connected, lists your address, and reflects wallets in the selector.*
4. *Fail cases to watch:*

- *If icon fails (e.g., chrome-extension://), ensure fallback icon renders.*
- *If connect fails, a readable error toast/console log should appear.*

C) ADA Donation (Main Flow)

1. *Ensure wallet network = Preprod.*
2. *Ensure recipient address = addr_test1… (Preprod).*

3. *Enter amount (e.g., 1, 0.5, or 0,25 to test comma parsing).*
4. *Click Donate.*
5. *Sign in wallet → wait for submission.*
6. *Expected:*

- *Wallet shows an outgoing transaction.*
- *App logs txHash and shows a success state.*
- *Cardanoscan preprod link opens to the tx.*

7. *Assertions:*

- *The output to the recipient matches the amount in TADA.*
- *Fee is displayed separately; no OutputTooSmallUTxO errors.*

D) Native Token Donation

1. *In Select Token, choose a token (e.g., SHEN) that exists on your network.*
2. *Enter a token amount in human units.*
3. *Click Donate → sign tx.*
4. *Expected:*

- *Tx includes one token output to recipient.*
- *Tx includes ADA top-up on the same output (≈ 1–1.5 ADA).*

5. *Assertions:*

- *No BabbageOutputTooSmallUTxO.*
- *Explorer shows correct token quantity and ADA top-up.*

E) Amount Parsing & BigInt Conversion

1. *Try inputs: 0.0002, 0,0002, 8.5, 8,5.*

2. Expected: normalizeAmount() + toUnits() convert to lovelace / smallest units without throwing.

3. Fail case: If you see RangeError: cannot convert 9.58 to BigInt, input wasn't normalized before conversion.

F) Network Mismatch Guard

1. Set wallet to Preprod but recipient to Mainnet (addr1…).

2. Attempt donation.

3. Expected:

- *Donation is blocked by a guard (UI) or*
- *Node rejects with WrongNetwork Testnet … [Addr Mainnet …] and app shows user-friendly error.*

4. Switch both to Mainnet and repeat → should succeed once funded.

G) Minimum ADA / Insufficient Balance

1. For ADA: enter < 1 ADA (e.g., 0.2).

2. Expected: App warns about min-ADA or blocks submission.

3. For low wallet balance: try donating more than available.

4. Expected: hasSufficientBalance returns false; UI blocks with a clear message.

H) Anonymous Donations (UI Only)

1. Toggle Donate Anonymously (if present).

2. Complete a donation.

3. Expected: No transaction link is rendered for anonymous entries in the history component (your fixed conditional is applied).

- *Cardano: link shown only if not anonymous.*

- *EVM/Solana: same rule.*

———

I) Recording the Donation (GraphQL Optional)

1. *Use Postman/Insomnia or app integration to call:*

- *createCardanoDonation(amount, transactionId, projectId, fromWalletAddress, toWalletAddress, token, tokenAddress, anonymous, valueUsd, priceUsd, userId)*

2. *Headers: Authorization: Bearer <JWT>, authversion: 2 (if required).*
3. *Expected: API returns OK; donation visible in backend (if a UI exists).*
4. *Fail case: unAuthorized → verify JWT and headers.*

———

J) Price Display (USD)

1. *Ensure Coingecko fetch for ADA succeeds.*
2. *Ensure MuesliSwap token→ADA price returns non-zero where supported.*
3. *Expected: Donation summary shows accurate USD value = amount_in_token * price_ada * ada_usd.*
4. *Fail case: If price APIs return 0/404, UI should degrade gracefully (e.g., —).*

———

K) Explorers

- *Preprod: verify the tx on Cardanoscan preprod.*
- *Mainnet: verify the tx on Cardanoscan mainnet.*
- *Expected: Amounts and assets match the UI.*

———

L) Regression Checks

- *Switch between ADA and token → amounts remain consistent (no double scaling).*
- *Switch between Mainnet and Preprod → correct explorer and recipient used.*

- *Disconnect/reconnect → selected wallet persists (if you store in localStorage) and reconnects.*

Pass/Fail Criteria (Summary)

- ✅ *Wallet connects; balances and tokens load without errors.*
- ✅ *ADA donation and token donation both submit; explorer confirms outputs.*
- ✅ *Input parsing works with dot or comma decimals.*
- ✅ *Min-ADA and insufficient balance rules enforced.*
- ✅ *Anonymous donations hide transaction link.*
- ✅ *(Optional) Donation recorded via GraphQL.*

6. Deployment Instructions

A) Prerequisites

- *Node.js: v22.x (mesh/utxorpc requires ≥20; Vercel 18 is deprecated)*
- *Env secrets ready: Blockfrost keys (mainnet + preprod), recipient addresses, GraphQL endpoint/JWT (if used)*

B) Environment Variables

Create .env.production (or set in your host's dashboard):

Example:

```
# Blockfrost (match the wallet/network at runtime)
NEXT_PUBLIC_CARDANO_BLOCKFROST_PROJECT_ID=<mainnet_key>
NEXT_PUBLIC_CARDANO_BLOCKFROST_PREPROD_PROJECT_ID=<preprod_key>

# Recipient addresses (used by runtime guard against wrong network)
NEXT_PUBLIC_CARDANO_RECIPIENT_MAINNET=addr1...
NEXT_PUBLIC_CARDANO_RECIPIENT_PREPROD=addr_test1...
```

```
# GraphQL (optional, for recording donations)
NEXT_PUBLIC_GQL_ENDPOINT=https://impact-graph.serve.giveth.io/graphql
NEXT_PUBLIC_AUTH_VERSION=2
# If your API requires it (store as secret, not committed)
NEXT_PUBLIC_AUTH_BEARER=<optional_jwt>
```

Keep secrets in your platform's Environment Variables UI (Vercel/Netlify), not in the repo.

C) Build Commands

```
# Install
pnpm install        # or: yarn install / npm install

# Build
pnpm build          # or: yarn build / npm run build

# Start (self-hosting)
pnpm start          # or: yarn start / npm start
```

D) Vercel (Recommended)

- **Import** the repo in Vercel.
- **Project Settings → Build & Development Settings**
    - Node.js Version: **22.x**
    - Install Command: auto (or pnpm install)
    - Build Command: auto (or pnpm build)
    - Output: .next (default)
- **Environment Variables**: add all keys from section B.
- **Domains** (optional): add custom domain.
- **Images** (optional): if you load remote wallet icons:

```
// next.config.js
module.exports = {
  images: {
    remotePatterns: [{ protocol: 'https', hostname: '**' }],
  },
};
```

6. Deploy. Verify logs show Node 22 and a successful Next build.

E) Self-Hosting (Docker Optional)

Without Docker

```
NODE_ENV=production pnpm build
NODE_ENV=production pnpm start
# app listens on PORT (default 3000)
```

With Docker (example)

```
# Dockerfile
FROM node:22-alpine AS builder
WORKDIR /app
COPY package.json pnpm-lock.yaml ./
RUN corepack enable && corepack prepare pnpm@latest --activate
RUN pnpm install --frozen-lockfile
COPY . .
RUN pnpm build

FROM node:22-alpine AS runner
WORKDIR /app
```

```
ENV NODE_ENV=production
COPY --from=builder /app/.next ./.next
COPY --from=builder /app/public ./public
COPY package.json ./
EXPOSE 3000
CMD ["node", ".next/standalone/server.js"]
```

Build & run:

```
docker build -t giveth-cardano .
docker run -p 3000:3000 --env-file .env.production giveth-cardano
```

F) Runtime Network Guard

Ensure you pick the recipient address based on the connected wallet’s network:

```
const netId = await wallet.getNetworkId(); // 1 = Mainnet, 0 = Testnets
const recipient =
  netId === 1
    ? process.env.NEXT_PUBLIC_CARDANO_RECIPIENT_MAINNET!
    : process.env.NEXT_PUBLIC_CARDANO_RECIPIENT_PREPROD!;
```

Use the matching Blockfrost key for that network.

G) Caching & Rate Limits

- Blockfrost has rate limits; avoid spamming price/asset endpoints.
- Debounce balance fetches; cache token metadata in memory for the session.
- Next.js: for public price endpoints (fetch(...)), consider cache: 'no-store' where real-time is required.

H) Monitoring & Logs

- Watch for:
- WrongNetwork Testnet … [Addr Mainnet …] → network mismatch.
- BabbageOutputTooSmallUTxO → add ~1–1.5 ADA on token outputs.
- BigInt/decimal errors → ensure toUnits(normalizeAmount(...), decimals).

I) Rollback

- Keep previous successful deployments (Vercel keeps history).
- If a build fails (Node 18 deprecation / utxorpc), set Node to 22 and redeploy.

7. Testnet / Emulator Results

Testing Deployment Link:
https://giveth-dapps-v2-git-testcardano-general-magic.vercel.app/cardano/giveth/donate

Environment Used:

- Yoroi wallet (Preprod) connected via @meshsdk/react
- Blockfrost Preprod key
- Cardanoscan Preprod for transaction verification
- Successful Results:
- ✅ Wallet connection established (Yoroi, Eternl, Nami tested).
- ✅ Balance fetching works for both ADA (lovelace) and test tokens (SHEN, TADA).
- ✅ Transactions submitted and confirmed on Preprod Cardanoscan.
- ✅ Correct conversion from ADA ↔ Lovelace (toUnits helper ensured proper BigInt conversion).
- ✅ Min-ADA requirement enforced (amount >= 1 ADA check prevents too-small UTxOs).
- Errors Encountered & Resolved:
- WrongNetwork Testnet vs Mainnet

- Error: WrongNetwork Testnet (Addr Mainnet ...)
- Cause: using Mainnet address on Preprod.
- Fix: added runtime guard to choose recipient address based on wallet.getNetworkId().
- BabbageOutputTooSmallUTxO
- Error when sending < 1 ADA or too little with tokens.
- Fix: enforced min 1 ADA buffer when building transactions.
- BigInt Conversion Errors
- Example: RangeError: The number 9.58 cannot be converted to a BigInt
- Fix: implemented normalizeAmount + toUnits helper to handle decimals safely.
- Test Token Results:
- SHEN (policyId: ...61)
- Price fetched via Muesliswap API.
- Transfer successful on Preprod with correct decimals (6).
- Custom TADA token
- Minted & visible in Yoroi wallet.
- Transfer worked with same transaction flow as ADA.
- User Experience Checks:
- ✅ Donation modal showed correct balances & USD conversion.
- ✅ Yoroi signed transactions normally (no approval flow like ERC-20).
- ✅ Transaction links opened correctly in Preprod Cardanoscan.
- Overall Conclusion:

The donation flow works reliably on Preprod Testnet, with accurate handling of ADA + multi-asset tokens. All main blockers (network mismatch, UTxO min values, BigInt conversions) were identified and fixed.

8. Dependencies & Environment

Frontend Framework

- Next.js 14 (React 18) – used for building the dApp UI
- styled-components – styling system
- @giveth/ui-design-system – shared UI components
- Cardano SDKs & Wallet Integration
- @meshsdk/react – React hooks for Cardano wallet connection (Yoroi, Nami, Eternl, Flint)
- @meshsdk/core – transaction building, signing, and submitting
- Yoroi Wallet (Preprod) – main wallet used for testing

- Blockchain & APIs
- Blockfrost API (Preprod & Mainnet endpoints) – balance, asset, and metadata queries
- Muesliswap API – fetching token prices in ADA (for SHEN, custom tokens, etc.)
- Coingecko API – fetching ADA price in USD
- Backend & GraphQL
- Apollo Client – for GraphQL queries/mutations to Giveth's Impact Graph
- GraphQL Server (Impact Graph) – endpoint to store and fetch donation records
- Language & Utilities
- TypeScript – type safety across frontend and transaction logic
- Viem – utility for unit conversion (formatUnits)
- Custom Helpers
- normalizeAmount – safely parse user input (e.g., 0,5 → 0.5)
- toUnits – converts human-readable amounts into smallest unit (ADA → lovelace)
- hasSufficientBalance – checks donation + buffer vs wallet balance
- Environments
- Preprod Testnet – primary environment for testing transactions and tokens
- Mainnet – used only for read-only validation (no donations executed)
- Local Development
- Node.js v18+
- Yarn/NPM for dependency management
- Environment variables (NEXT_PUBLIC_CARDANO_BLOCKFROST_PROJECT_ID, API keys, etc.)

9. Remaining Issues / Next Steps

- *Transaction Fee Estimation*

- *• Need a more reliable way to estimate Cardano network fees before submission.*
- *• Current workaround uses buffer checks (hasSufficientBalance), but lacks precision.*
- *• Anonymous Donations Handling*
- *• UI condition for anonymous Cardano/EVM donations requires refinement (preventing transaction link exposure).*
- *• Token Handling Improvements*
- *• Standardize conversion functions for ADA (toUnits, normalizeAmount) and ensure consistent BigInt usage across ADA and native tokens.*
- *• Verify decimals handling for all supported Cardano tokens.*
- *• Wallet Detection & UX*
- *• Improve error handling when no wallet is connected or when the user is on the wrong network (Mainnet vs Preprod).*
- *• Add responsive UI feedback if wallet balance or selected token info fails to load.*
- *• Testing Coverage*
- *• Extend test cases for token transfers beyond ADA (e.g., SHEN, test tokens).*
- *• Validate donations with multiple Cardano wallets (Yoroi, Nami, Eternl).*
- *• API Integration*
- *• Ensure GraphQL mutation (createCardanoDonation) aligns with backend schema (types and variable mapping).*
- *• Implement proper error messaging for unauthorized or invalid transaction submissions.*
- *• Next Steps*
- *• Deploy fixes to staging for end-to-end donation flow testing.*
- *• Gather feedback from test users (using Yoroi/Nami wallets) on transaction UX.*
- *• Finalize minimum-ADA validation rules and sync with backend enforcement*