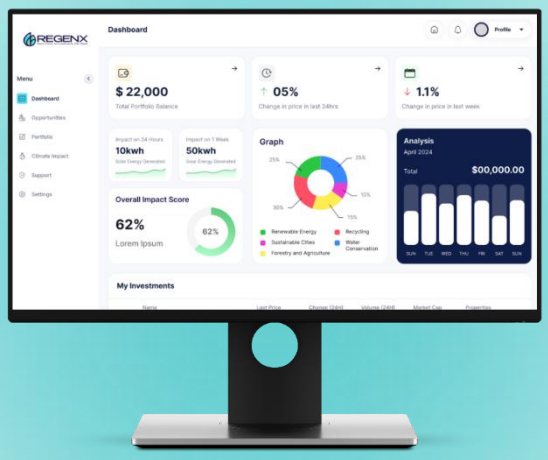
RegenX
Technical Architecture Guide

Version: July 2025
Prepared for: Stellar Community Fund Build Award — SCF #37
Project: RegenX — Tokenized Infrastructure for Climate Capital



REGENX
Impact Investing, Made Simple.

Contents

1. Introduction

RegenX is a tokenized investment platform enabling accredited investors to fund clean energy infrastructure — starting with mid scale solar — through fractionalised digital ownership. The platform integrates environmental data from IoT sources (Sunified) to verify project milestones and measure carbon benefits. RegenX combines traditional investment processes with the transparency, security, and programmability of the Stellar blockchain and Soroban smart contracts.

This technical guide documents RegenX's architecture, including its existing components and how the Stellar Build Award will support its on chain evolution.

2. Part I: Current RegenX Platform Architecture

2.1 System Model

RegenX is currently built on a robust, proven, modern SaaS architecture:

- Frontend: React + TypeScript
- Backend: NestJS with PostgreSQL
- Authentication: Auth0 for role based access
- DevOps: Hosted on Coolify (self managed PaaS)
- Continuous Integration: GitHub Actions
- Secure Secrets & Credentials: managed via .env configurations and Auth0
- Staging & Production Environments: fully separated with secure deployment workflows

This separation of concerns provides a scalable, maintainable foundation.

Diagram 1a: RegenX System Context (C4 Level 1)

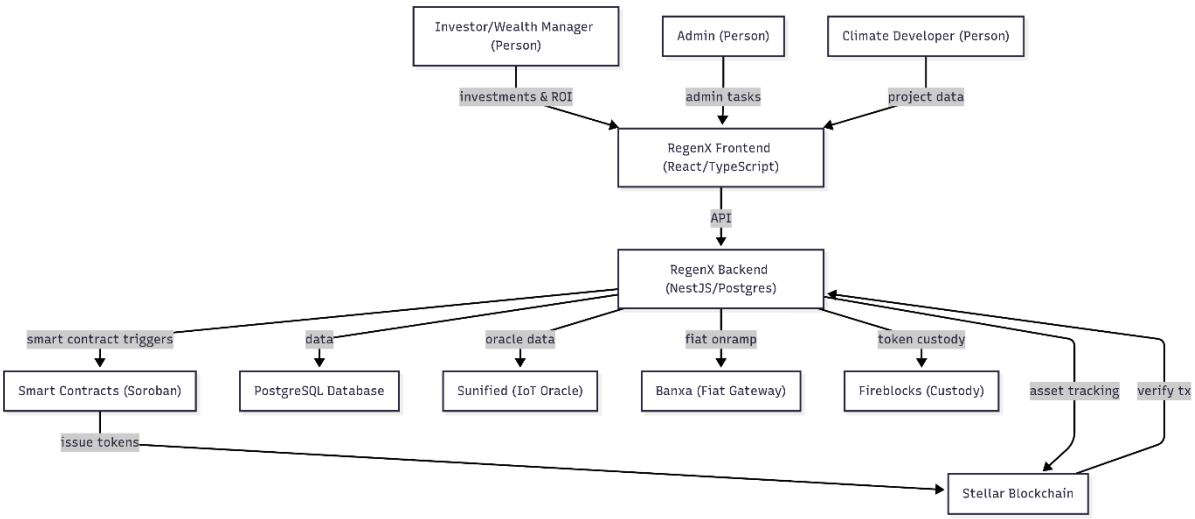
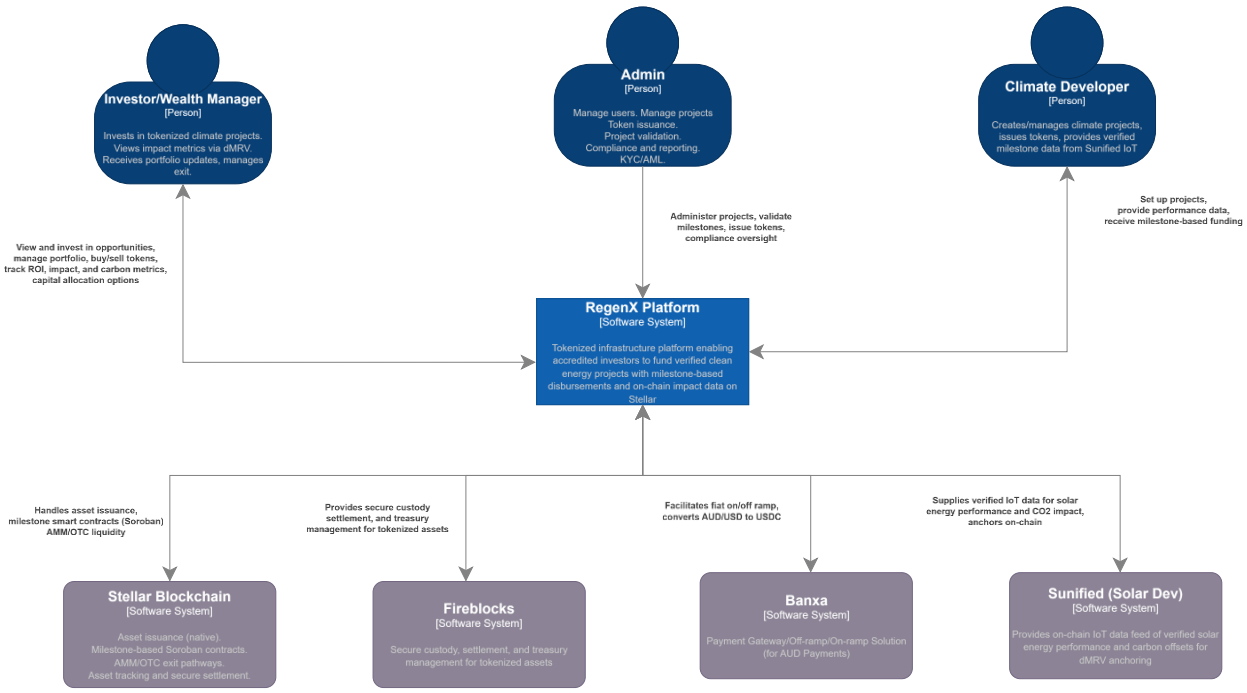


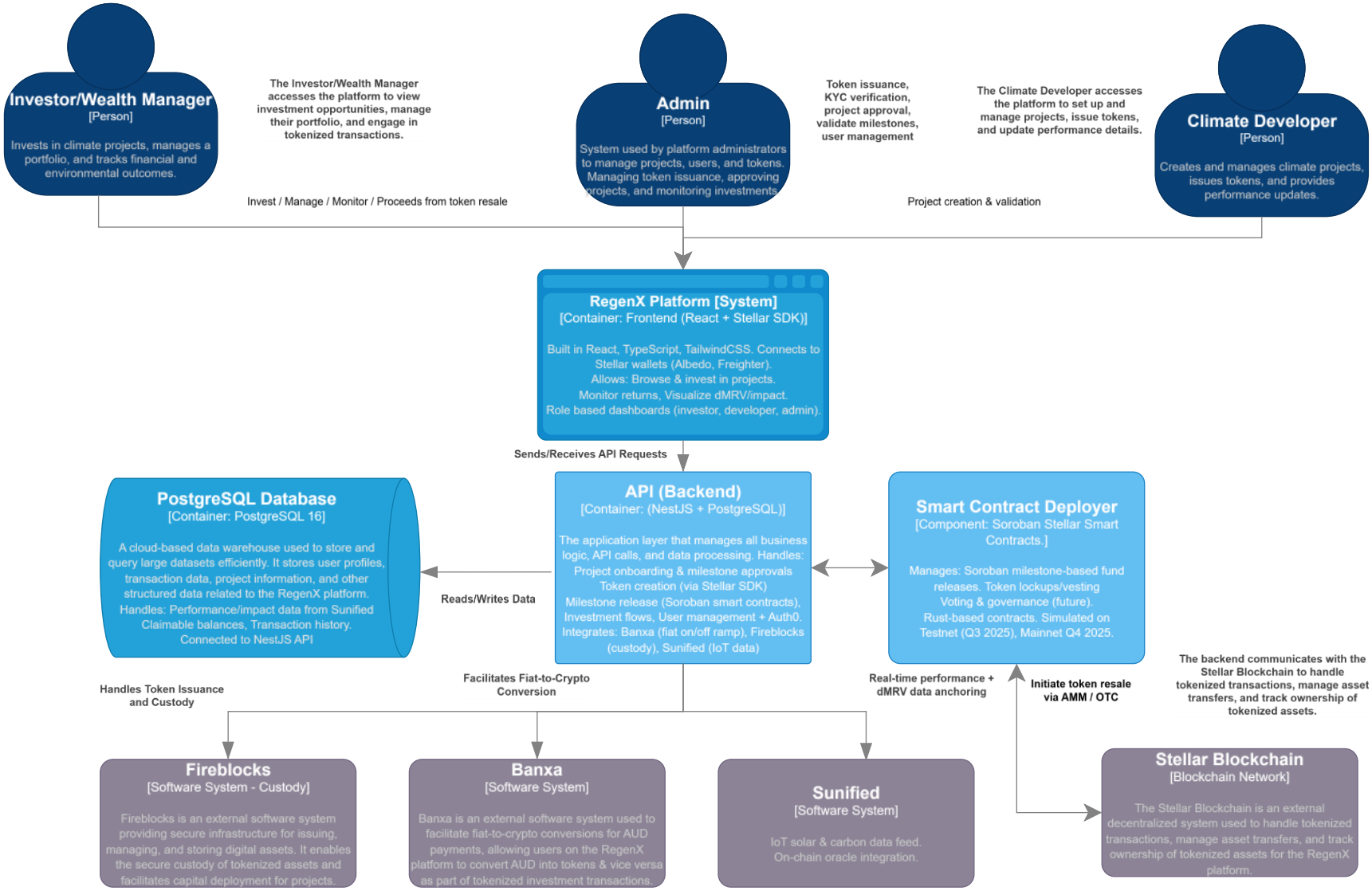
Diagram 1b: RegenX System Context (C4 Level 1)



2.2 Core Components & Dependencies

- **Frontend**
  - React, Vite, TailwindCSS
  - Stellar SDK for wallet connectivity
  - Cypress for testing
- **Backend**
  - NestJS framework
  - TypeORM ORM
  - PostgreSQL relational database
  - Fireblocks planned for secure custody
  - Banxa for fiat ramps
  - Interfaces with Sunified Oracle for dMRV
  - Soroban SDK integration planned for milestone payments
- **Infrastructure**
  - Coolify (Docker based orchestration)
  - GitHub Actions for CI/CD
  - Auth0 for secure user session management
  - SonarQube for quality gating
  - PostgreSQL on cloud or self managed server

Diagram 2: RegenX Container View (C4 Level 2)



2.3 Key Features of the Existing Platform

- Role based dashboards for investors, project developers, and administrators
- Onboarding flows with investor KYC
- Project creation and milestone tracking
- Transaction flow simulation on Stellar Testnet
- Basic investor reporting and ROI dashboards
- Newsletter sign up, early access lists, and investor engagement modules
- Ability to bundle carbon credits and impact scores with investment

---

3. Part II: Proposed Stellar & Soroban Integration

3.1 Overview

The RegenX Build Award will transition core platform logic on chain through Stellar's native asset issuance and Soroban smart contracts.

RegenX will leverage:

- Stellar for native token issuance
- Claimable balances for milestone based disbursements
- Soroban smart contracts to enforce milestone release
- dMRV data timestamped and anchored on chain from Sunified IoT devices
- Stellar AMM or OTC pathways for secondary market liquidity
- White label portals for wealth managers to syndicate investments

This hybrid architecture balances off chain onboarding and investor UX with on chain security and transparency.

3.2 Stellar Integration Components

- **Tokenized Infrastructure**
  - Each project represented by a Stellar issued asset
  - Fractional ownership recorded on chain
- **Milestone based Capital Release**
  - Soroban contract logic
  - Tied to dMRV triggers from Sunified IoT
- **Fiat Onboarding**
  - Investors deposit AUD/USD via Banxa
  - Converted to USDC on Stellar
- **Liquidity**
  - Post hold period, investors exit via Stellar AMM or OTC

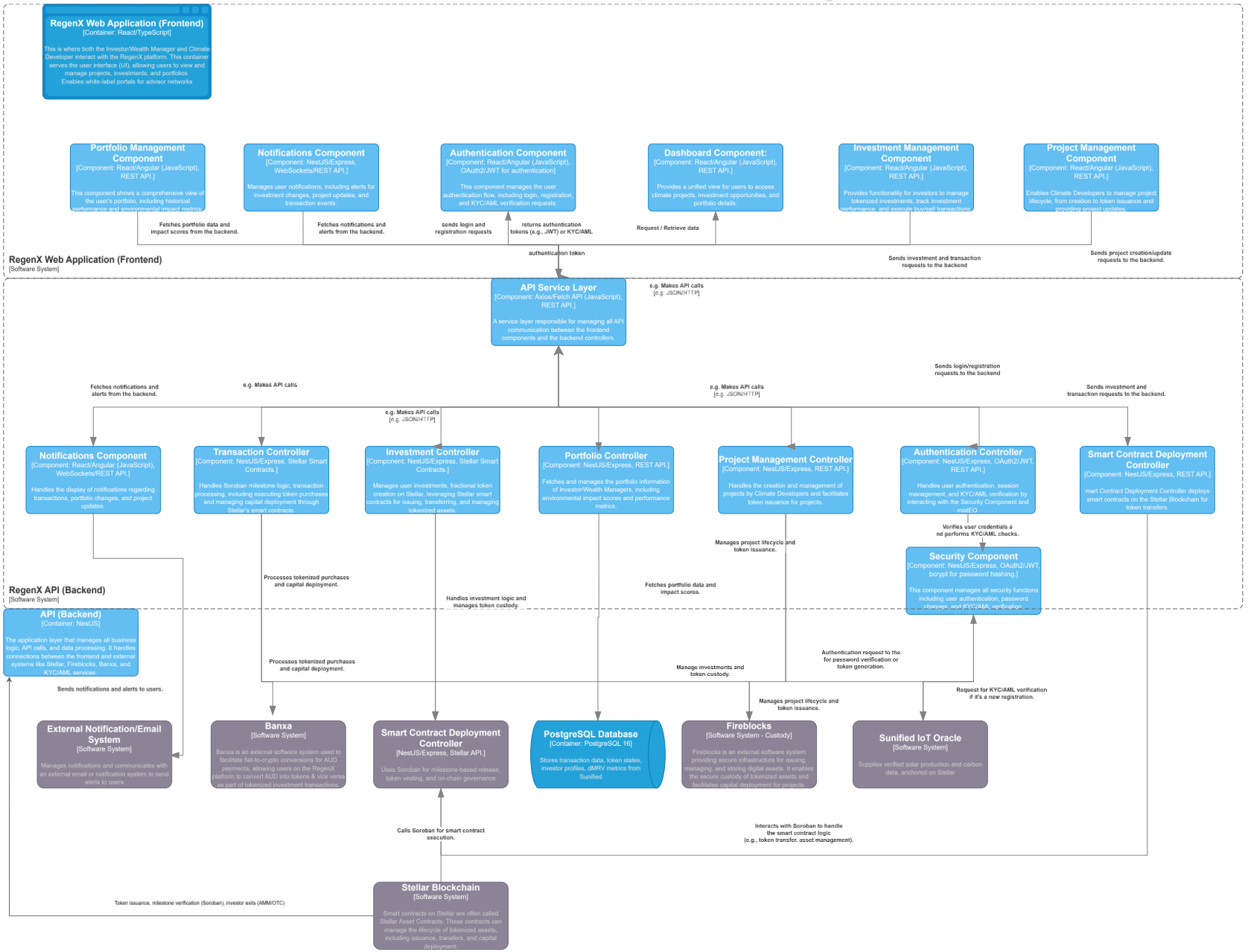
- **Compliance**
  - Soroban milestone contracts will gate fund release
  - RegenX maintains off chain compliance verification to support licensing

3.3 Soroban Smart Contract Design

The Soroban smart contracts will include:

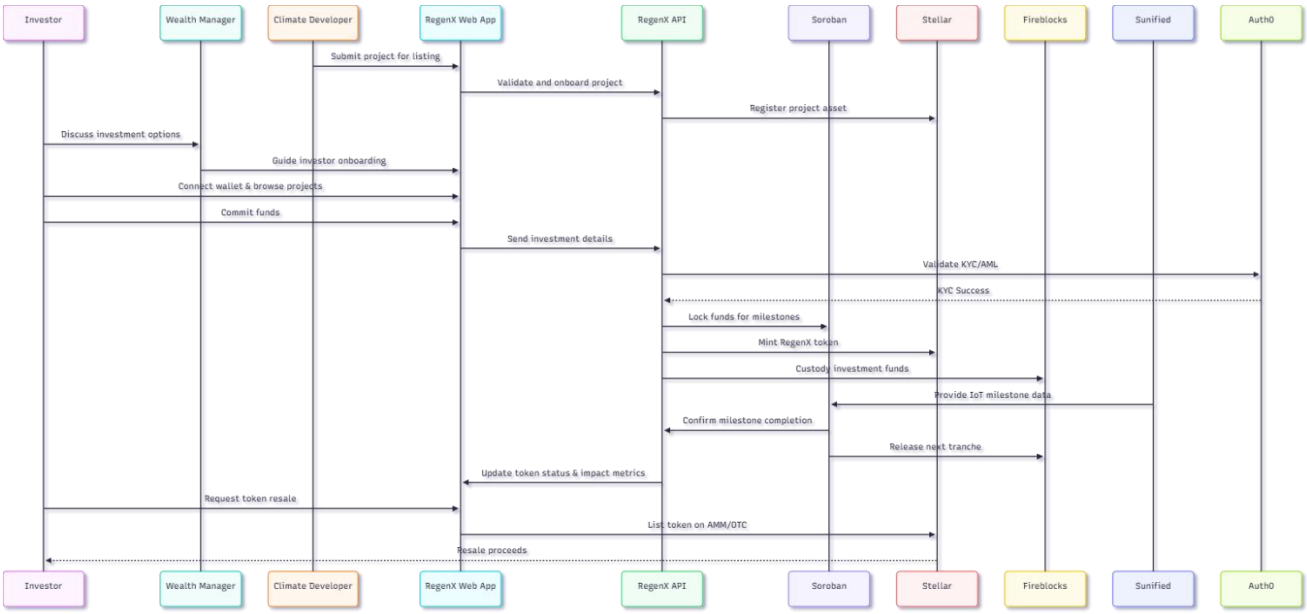
- Milestone validation logic connected to Sunified IoT events
- Enforced staged fund release based on construction or performance verification
- Token lockups and vesting periods
- Optional investor governance voting features
- Cross contract calls to monitor dMRV performance on chain
- Use of Stellar memo fields to anchor IoT data references
- Soroban contracts written in Rust, planned to be open sourced after audit

Diagram 3: Soroban Contract Component View (C4 Level 3)


RegenX Web Application (Frontend)
[Container: React/TypeScript]
This is where both the Investor/Wealth Manager and Climate Developer interact with the RegenX platform. This container serves the user interface (UI), allowing users to view and manage projects, investments, and portfolios. Enables white-label portals for advisor networks
Portfolio Management Component
[Component: React/Angular (JavaScript), REST API.]
This component shows a comprehensive view of the user's portfolio, including historical performance and environmental impact metrics
Notifications Component
[Component: NestJS/Express, WebSockets/REST API.]
Manages user notifications, including alerts for investment changes, project updates, and transaction events.
Authentication Component
[Component: React/Angular (JavaScript), OAuth2/JWT for authentication]
This component manages the user authentication flow, including login, registration, and KYC/AML verification requests.
Dashboard Component:
[Component: React/Angular (JavaScript), REST API.]
Provides a unified view for users to access climate projects, investment opportunities, and portfolio details.
Investment Management Component
[Component: React/Angular (JavaScript), REST API.]
Provides functionality for investors to manage tokenized investments, track investment performance, and execute buy/sell transactions
Project Management Component
[Component: React/Angular (JavaScript), REST API.]
Enables Climate Developers to manage project lifecycle, from creation to token issuance and providing project updates.
Fetches portfolio data and impact scores from the backend.
Fetches notifications and alerts from the backend.
sends login and registration requests
returns authentication tokens (e.g., JWT) or KYC/AML
Request / Retrieve data
authentication token
Sends investment and transaction requests to the backend
Sends project creation/update requests to the backend.
RegenX Web Application (Frontend)
[Software System]
API Service Layer
[Component: Axios/Fetch API (JavaScript), REST API.]
A service layer responsible for managing all API communication between the frontend components and the backend controllers.
e.g. Makes API calls
[e.g. JSON/HTTP]
Sends login/registration requests to the backend
Fetches notifications and alerts from the backend.
e.g. Makes API calls
e.g. Makes API calls
[e.g. JSON/HTTP]
Sends investment and transaction requests to the backend.
Notifications Component
[Component: React/Angular (JavaScript), WebSockets/REST API.]
Handles the display of notifications regarding transactions, portfolio changes, and project updates.
Transaction Controller
[Component: NestJS/Express, Stellar Smart Contracts.]
Handles Soroban milestone logic, transaction processing, including executing token purchases and managing capital deployment through Stellar's smart contracts.
Investment Controller
[Component: NestJS/Express, Stellar Smart Contracts.]
Manages user investments, fractional token creation on Stellar, leveraging Stellar smart contracts for issuing, transferring, and managing tokenized assets.
Portfolio Controller
[Component: NestJS/Express, REST API.]
Fetches and manages the portfolio information of Investor/Wealth Managers, including environmental impact scores and performance metrics.
Project Management Controller
[Component: NestJS/Express, REST API.]
Handles the creation and management of projects by Climate Developers and facilitates token issuance for projects.
Authentication Controller
[Component: NestJS/Express, OAuth2/JWT, REST API.]
Handles user authentication, session management, and KYC/AML verification by interacting with the Security Component and mistEO.
Smart Contract Deployment Controller
[Component: NestJS/Express, REST API.]
mart Contract Deployment Controller deploys smart contracts on the Stellar Blockchain for token transfers.
Verifies user credentials a nd performs KYC/AML checks.
Manages project lifecycle and token issuance.
Security Component
[Component: NestJS/Express, OAuth2/JWT, bcrypt for password hashing.]
This component manages all security functions including user authentication, password changes, and KYC/AML verification.
Processes tokenized purchases and capital deployment.
Handles investment logic and manages token custody.
Fetches portfolio data and impact scores.
RegenX API (Backend)
[Software System]
API (Backend)
[Container: NestJS]
The application layer that manages all business logic, API calls, and data processing. It handles connections between the frontend and external systems like Stellar, Fireblocks, Banxa, and KYC/AML services.
Processes tokenized purchases and capital deployment.
Manage investments and token custody.
Authentication request to the for password verification or token generation.
Request for KYC/AML verification if it's a new registration.
Sends notifications and alerts to users.
Manages project lifecycle and token issuance.
External Notification/Email System
[Software System]
Manages notifications and communicates with an external email or notification system to send alerts to users
Banxa
[Software System]
Banxa is an external software system used to facilitate fiat-to-crypto conversions for AUD payments, allowing users on the RegenX platform to convert AUD into tokens & vice versa as part of tokenized investment transactions.
Smart Contract Deployment Controller
[NestJS/Express, Stellar API.]
Uses Soroban for milestone-based release, token vesting, and on-chain governance
PostgreSQL Database
[Container: PostgreSQL 16]
Stores transaction data, token states, investor profiles, dMRV metrics from Sunified
Fireblocks
[Software System - Custody]
Fireblocks is an external software system providing secure infrastructure for issuing, managing, and storing digital assets. It enables the secure custody of tokenized assets and facilitates capital deployment for projects.
Sunified IoT Oracle
[Software System]
Supplies verified solar production and carbon data, anchored on Stellar
Calls Soroban for smart contract execution.
Interacts with Soroban to handle the smart contract logic (e.g., token transfer, asset management).
Stellar Blockchain
[Software System]
Smart contracts on Stellar are often called Stellar Asset Contracts. These contracts can manage the lifecycle of tokenized assets, including issuance, transfers, and capital deployment.
Token issuance, milestone verification (Soroban), investor exits (AMM/OTC)

3.4 Flow Diagrams

Diagram 4: Sequence Diagram
(Sequence diagram showing: investor funds -> RegenX backend -> Soroban milestone release -> Sunified dMRV -> payout)



4. Part III: Technical Workflows & Security

4.1 Key Technical Workflows

- **Investor Onboarding**
  - Auth0 secures investor session
  - KYC off chain
  - Funding flow via Banxa
  - Token minted on Stellar
  - Soroban locks initial funds
- **Project Funding**
  - Developer uploads milestone plan
  - Funds locked into claimable balance
  - Each milestone triggers Soroban release after dMRV confirmation
- **Project Operations**
  - Sunified IoT streams real world data
  - On chain anchoring of kWh and CO2 data
  - Dashboard shows real time investor returns
- **Investor Exit**

- After hold period, investor sells token on Stellar AMM or OTC
- RegenX platform updates off chain portfolio records

4.1a: API Components

The RegenX backend exposes modular API controllers to manage all project and investor workflows. These controllers follow a clean, maintainable structure:

- **Auth Controller**: Handles authentication using Auth0 JWTs and enforces role based permissions.
- **Project Controller**: Manages project listings, updates, and milestone tracking.
- **Investment Controller**: Accepts investor commitments, creates tranches, and triggers token issuance.
- **Token Controller**: Interacts with the Stellar SDK to mint and distribute fractional ownership tokens.
- **Impact Controller**: Anchors solar generation and carbon data from Sunified's IoT oracle on chain.
- **Admin Controller**: Provides administrative functions for approvals, user oversight, and platform settings.

Each controller uses NestJS decorators to define routes, manage guards, and integrate validation for a secure and scalable API.

4.2 Testing & Security

Authentication & Security

The RegenX platform uses a layered security model:

- **Auth0** provides identity and access management, with JWT based session tokens.
- **Session Security** uses signed JWTs with role based claims.
- **NestJS Guards and Decorators** apply fine grained role access across all API endpoints.
- **User Verification Flow** is enforced via an Auth0 triggered serverless function during signup, preventing unverified accounts from investing or issuing tokens.

RegenX is committed to robust testing and a secure architecture to protect both investor capital and project developer assets. Our testing strategy includes:

- **Unit Testing**: We will implement unit tests for all smart contract functions using Rust testing frameworks, ensuring correctness of the business logic at a granular level.
- **Integration Testing**: Conduct integration testing across the RegenX backend, Soroban smart contracts, and the Stellar Testnet to verify correct interactions and data flows.
- **End to End Testing**: Simulate full user journeys from onboarding to investing and redeeming tokens, ensuring functional correctness across the entire stack.
- **Testnet Environment**: We will deploy to the Stellar Testnet to validate milestone based fund release functionality using actual Sunified IoT data triggers before deploying to mainnet.
- **Security Strategy**:
    - Strict use of `require_auth()` for all privileged contract functions
    - Role based permissions enforced via Auth0
    - Continuous monitoring for known vulnerabilities in dependencies
    - Adherence to Stellar best practices for token issuance, claimable balances, and multisig
    - External third party audit via the Soroban Audit Bank, as recommended by SCF guidelines

This approach will ensure RegenX meets high security standards while delivering confidence to the Stellar ecosystem.

5: Deployment & DevOps

RegenX is deployed on a containerised, modern stack to ensure high security and scalability.

- **Hosting**: Coolify (self hosted PaaS)
- **Environments**:
    - Staging (automated deploys for quality assurance)
    - Production (manual promotion after review)
- **CI/CD**: GitHub Actions pipeline running ESLint, Prettier, Supertest, Jest, and SonarQube for continuous quality and security checks.
- **Infrastructure**:
    - Two web APIs (NestJS containers)
    - Two PostgreSQL databases for transactional and reporting data
    - Auth0 for identity and secure user authentication
    - Serverless functions (via Vercel or AWS Lambda) to manage signup, verification, and asynchronous tasks
    - Secure secret management and API credentials held in a managed vault

This infrastructure guarantees consistent build, deploy, and rollback procedures while meeting Stellar ecosystem standards.

---

6. Conclusion

RegenX represents a practical, impactful tokenized clean energy investment solution. By leveraging Stellar's proven asset issuance and the programmability of Soroban smart contracts, RegenX can ensure secure, milestone based, transparent funding of renewable energy infrastructure, verified by on chain dMRV data.