TEAM – NEW00007

DECENTRALIZED VOTING SYSTEM (D POLL) PROJECT ROADMAP

This document outlines our plan to build d Poll, a simple decentralized polling app using Soroban. Since we're still learning, we've designed this project to be achievable within a hackathon timeframe by modifying and experimenting with examples from the Soroban documentation. Our goal is to understand how decentralized apps work while creating something functional and fun.

Project Goal

To build a smart contract that stores counters for two options ("Option A" and "Option B") and a simple webpage that lets users click buttons to vote, displaying the results dynamically.

Phase 1: Environment Setup & Foundation

Objective

Preparing the local development environment for Soroban smart contract and frontend development.

Key Tasks

- Read 00. Setup.md and docs/getting-started/environment-setup.md.
- Installing all "Prerequisites" and tools as listed in the documentation.
- Ensuring the Soroban CLI is working so that we can compile and run the example contracts.

Relevant Docs

- 00. Setup.md
- docs/getting-started/environment-setup.md

Phase 2: The Smart Contract (Backend Logic)

Objective

Creating the **d Poll** smart contract that manages and stores the vote counts. This will be achieved by modifying the increment/ example.

Base Example

- increment/

Key Tasks

- **Understand State:** Read docs/smart-contracts/contract-state.md to understand how to store data.
- **Adapt State:** Modify the contract's state to store **two** counters (e.g., VOTES_A, VOTES_B) instead of one.
- **Modify Functions:**
    - Rename increment () to vote _a ().
    - Update vote _a () to only increment the counter for "Option A".
- **Create New Functions:**
    - Create vote _b (): This function will increment the counter for "Option B".
    - Create get _votes _a (): A "getter" function that returns the current count for "Option A".
    - Create get _votes _b ( ): A "getter" function that returns the current count for "Option B".
- **Test Contract:**
    - Follow docs/smart-contracts/testing-contracts.md.
    - Write and run tests to ensure vote _a, vote _b, and the getter functions all work as expected.
- **Deploy Locally:** Deploy the finalized d Poll contract to your local network.

Relevant Docs

- docs/smart-contracts/contract-state.md
- docs/smart-contracts/testing-contracts.md

Phase 3: The Frontend (Voting Booth UI)

Objective

Build a simple webpage that allows users to interact with the deployed **d Poll** smart contract. This will be achieved by modifying the frontend _hello_ world/ example.

Base Example

- frontend _hello _world/

Key Tasks

- **Connect Contract:**
    - Read docs/frontend/connecting-to-contracts.md.
    - Update the frontend code to use your new d Poll contract ID.
- **Update UI (HTML):**
    - Remove the "Hello World" button and related text.
    - Add two new buttons: "Vote for Option A" and "Vote for Option B".
    - Add a section to display results (e.g., "Current Results:").
- **Update Logic (JavaScript):**
    - Read docs/frontend/user-interaction.md.
    - Connect the "Vote for Option A" button to call the vote _a () function on your smart contract.
    - Connect the "Vote for Option B" button to call the vote _b () function on your smart contract.
- **Display Results:**
    - Write JavaScript that periodically calls the get_ votes _a () and get_ votes _b () functions.
    - Display the returned vote counts on the webpage, updating them after a vote is cast.

Relevant Docs

- docs/frontend/connecting-to-contracts.md
- docs/frontend/user-interaction.md