Technical Challenge

Your challenge is to create a simple application that consists of three components: a smart contract, a server, and a web application.

The purpose is to create a system which demonstrates simple linear data flow.

1. User inputs data on webapp
2. Webapp broadcasts data to smart contract
3. Smart contract emits events which include data
4. API listens for emitted events from the smart contract
5. API transforms data in some trivial way and keeps records
6. Webapp displays transformed data from the API

Ethereum Smart Contract

- Function to accept string data
- Emit events
- Write in Solidity or Vyper
- Hint: you'll probably want to use a development framework like Hardhat, Truffle, or Waffle

Server (API)

- Listen for events on the smart contract
- Perform some trivial string manipulation
    - ROT13, alter casing, hash, encrypt, whatever, etc
- Store resulting string for future GET request
    - Redis, database, in-memory, etc
- Write in Node.js or GoLang

Web Application

- Form to input string data, place in a transaction, and broadcast to Ethereum
    - Testnet or local network is 100% expected! Don't waste mainnet ETH!
- Display a list of transformed strings from the API
- Write in React
- You'll need a Web3 library, like "Web3" or "Ethers.js"

Considerations

You may use whatever resources you desire to complete this challenge. The purpose of this exercise is to evaluate your ability to develop software from a set of feature requirements given any resources at your disposal in a working environment.

Presentation

You may present your solution on any platform you choose, whether local, containerized, and/or deployed to a cloud service provider. Expect to present the functioning app and share the underlying source code.

Try Your Best

If you cannot complete every aspect of this exercise, no worries. Please complete each component (contract, api, app) to the best of your ability.