Dexponent Hackathon Festival - Round 1

Building a DeFi protocol for global investment finance.

Who can participate

- Above the legal age of majority in the country of residence
- All countries/territories, excluding standard exceptions

Overview:

Welcome to the inaugural round of the Dexponent Hackathon Festival! This event invites developers worldwide to contribute to the Dexponent ecosystem by building innovative solutions. In this round, participants will focus on creating an intuitive, multi-step wizard that enables Asset Managers to define and deploy new Farms within the Dexponent protocol.

Objective:

Develop a user-friendly, multi-step wizard interface that allows Asset Managers to:

1. Define Farm parameters (e.g., strategy type, target APY, collateral assets).
2. Set verifier requirements and risk profiles.
3. Preview expected Sharpe benchmarks and fee structures.
4. Deploy the Farm with a clear transaction summary.

Requirements

🛠️ What to Build

Objective: Develop a user-friendly, multi-step wizard interface that enables Asset Managers to define and deploy new.

Farms within the Dexponent protocol.

🔹 Core Features

1. Multi-Step Form Wizard:

Implement a multi-step form that guides users through the Farm creation process. Each step should collect specific information, such as strategy type, target APY, collateral assets, and verifier requirements. Ensure smooth navigation between steps, with the ability to go back and edit previous inputs.

2. Real-Time Data Integration:

Integrate with Uniswap V3's TWAP oracle to fetch real-time price data for selected assets. Use this data to provide immediate feedback on expected Sharpe benchmarks and fee structures.

3. Form Validation and User Feedback:

Implement client-side validation for each form step to ensure data integrity. Provide clear error messages and guidance to assist users in correcting input errors.

4. Deployment Summary:

Before final submission, present a comprehensive summary of all inputs and calculated metrics. Allow users to review and confirm their configurations before deploying the Farm.

🔹 Recommended Technologies

- Frontend:
  - React.js with libraries like Formik or React Hook Form for form management.
  - Tailwind CSS for styling and responsive design.
  - State management tools like Zustand or Context API.
- Data Fetching:
  - Use React Query or Axios for API calls to fetch real-time data.
- Blockchain Interaction:
  - Ethers.js or Web3.js for interacting with smart contracts.
  - Integration with MetaMask or WalletConnect for user authentication and transaction signing.

📤 What to Submit

Participants are expected to submit the following components:

1. Source Code Repository:
   a. A public GitHub repository containing all source code, organized and documented.
   b. Include a README file with setup instructions, technologies used, and any other relevant information.
2. Deployed Application:
   a. A live, deployed version of the application accessible via a public URL (e.g., Vercel, Netlify).
   b. Ensure the application is fully functional and mirrors the features described in the "What to Build" section.
3. Documentation:
   a. Detailed documentation explaining the application's architecture, components, and any assumptions made.
   b. Instructions for setting up the development environment and deploying the application.

🛠️ Technical Requirements

Frontend:

- Implement a multi-step form wizard using frameworks like React with react-hook-form and Tailwind CSS for Styling.
- Ensure responsive design and intuitive navigation between steps.

- Include real-time validation and contextual help for form fields.

Backend/Smart Contracts:

- Integrate with Uniswap V3's TWAP oracle to fetch real-time price data for assets.
- Utilize or extend existing smart contracts to handle Farm deployment logic.
- Ensure secure and efficient interaction with the blockchain for deploying Farms.

Data Handling:

- Manage form state across steps using state management libraries like Zustand.
- Handle user inputs securely and validate data before submission.

📥 Input Data

- Participants will work with the following inputs:
    - User Inputs:
        - Strategy type selection (e.g., yield farming, liquidity provision).
        - Target APY and risk tolerance levels.
        - Selection of collateral assets from a predefined list.
        - Verifier requirements and staking rules.
    - External Data:
        - Real-time price feeds from Uniswap V3 TWAP oracle for selected assets.
        - Predefined templates or examples of existing Farms for reference.

1. Deployed Application:
    a. A live, deployed version of the application accessible via a public URL (e.g., Vercel, Netlify).
    b. Ensure the application is fully functional and mirrors the features described in the "What to Build" section.
2. Documentation:
    a. Detailed documentation explaining the application's architecture, components, and any assumptions made.
    b. Instructions for setting up the development environment and deploying the application.

✅ Expected Outcomes

- Functional Wizard Interface:
    - A seamless, multi-step form guiding Asset Managers through Farm creation.
    - Real-time previews of expected returns and risk assessments based on inputs.
    - Clear transaction summaries before deployment.
- Smart Contract Interaction:
    - Successful deployment of new Farms to the blockchain via the wizard.
    - Proper handling of verifier staking requirements and collateral management.

- User Experience:
    - Responsive and accessible design suitable for various devices.
    - Informative feedback and error messages to guide users.
    - Integration of contextual help and tooltips for complex fields.