Decentralized Crowdfunding App

- Student Name: Lai Quang Huy
- Student ID: 20194438
- Instructor: Prof. Hai Van Pham

Introduction

Overview of the decentralized crowdfunding app

- Problem statement
  - Traditional crowdfunding has centralized control, high fees, and lack of transparency.
- Solution
  - Decentralized crowdfunding app built on Ethereum blockchain.
- Goal
  - Empower users to create and fund campaigns with low fees and complete transparency.

Problem

- Traditional crowdfunding platforms:
  - High fees
  - Limited access to certain countries
  - Centralized control
- Solution: A decentralized crowdfunding app that:
  - Lowers fees
  - Offers global access
  - Removes centralized control

Solution

- Decentralized crowdfunding app built on Ethereum blockchain
- Smart contract to handle:
    - Campaign creation
    - Funding
- User-friendly interface

Overview of the decentralized crowdfunding app

- Technologies used:
  - Solidity
  - Hardhat
  - Ether.js
  - Next.js

Architecture

- The app consists of 2 main components: the smart contract and the frontend.
- The smart contract is written in Solidity and deployed on the Goerli test network.
- The frontend is built with Next.js and connects to the smart contract using the Ether.js library.

Functionality

- Home page:
  - Connect wallet button
- Browse page:
  - View all campaigns
  - Click on campaign to view details and donate
  - Modal view for each campaign with funding progress, input box, and submit button.
- Create page:
  - Submit new campaign
- Profile page:
  - View all user's campaigns

Functionality

- Smart contract:
  - Handles campaign creation, funding, and distribution
  - Ability to connect user's wallet, fund campaigns, and view transaction history.
  - Complete transparency and low fees.

Smart Contract

- Written in Solidity
- Contains functions to create, read, and fund campaigns.
- Implements security features such as access control and checks on input data.
- Deployed on the Goerli test network using the Hardhat development environment.
- Fully tested using the Hardhat testing framework.

Testing and Deployment

Testing and Deployment

- Smart contract testing is an essential step in developing secure and reliable decentralized applications
- Smart contract tested using Hardhat development environment and Hardhat testing framework.
- Smart contract deployed on the Goerli test network using the ThirdWeb deployment platform.
- Frontend deployed on Vercel.
- DApp tested using various test scenarios to ensure functionality and security.

Hardhat

- Popular development environment for Ethereum that comes with a built-in testing framework
- Features:
  - contract mocking
  - test snapshots
  - coverage reports
- Write and run automated tests for smart contracts, ensuring they are functioning as intended
- By testing the smart contract before deployment, we can catch bugs and vulnerabilities early in the development process, saving time and reducing risks

Ether.js

- A library for interacting with the Ethereum blockchain
- Key features:
  - Accounts and Wallets
  - Contract Interactions
  - Provider abstraction
- Used to interact with the smart contract in the app

Frontend

- Built with Next.js and styled with Tailwind CSS.
- Connects to the smart contract using the Ether.js library.
- Uses React components for modularity and reusability.
- Implements functionality for connecting wallet, creating and funding campaigns, and viewing transaction history.

Testing

- Testing with Hardhat:
  - Ensures functionality of smart contract
  - Ensures security and safety of funds
- Testing the DApp:
  - Ensures functionality of user interface
  - Ensures smooth integration between front-end and back-end

Deployment

- Deployed to the Goerli test network:
  - Ensures security and safety of funds
  - Allows for testing and development without spending real Ether
- Thirdweb:
  - Easy deployment process
- Vercel:
  - Frontend deployment

Evaluation

- Pros:
  - Lower fees
  - Global access
  - Decentralized control
- Cons:
  - Learning curve for non-technical users
  - Dependence on blockchain technology
- Overall success of the app will depend on adoption by users and the strength of the Ethereum network

Conclusion

- The decentralized crowdfunding app provides a secure and transparent alternative to traditional crowdfunding.
- Built using the Ethereum blockchain and Next.js frontend, it is a modern and efficient solution.
- Testing and deployment were completed with rigorous testing and best practices.
- The app has the potential for future expansion and improvement.

Future Work

- Addition of social features for sharing campaigns and updates.
- Integration with other blockchain networks for broader support.
- Implementation of a reputation system for campaign creators and funders.
- Integration of IPFS for decentralized storage of campaign information.
- Improved UI/UX and accessibility features.

Thank you!

Thank you for your attention and interest in my decentralized crowdfunding app.

References

- Solidity: https://docs.soliditylang.org
- Hardhat: https://hardhat.org
- Ether.js: https://docs.ethers.io/v5
- Next.js: https://nextjs.org
- Goerli Test Network: https://goerli.net
- Thirdweb: https://thirdweb.com