Decentralized Crowdfunding Platform

Project Report

Team Name: Team KanyaRashi

Team Members:

- L. Yashwanth Chowhan – 230001046
- M. Mohan Prudhvi Sai – 230001047
- M. Rishik Preetham – 230001048
- M. Rajavardhan – 230001053
- Mohit Katariya – 230001055
- L. Praveen Kumar – 230041018

Introduction

This project is a decentralized crowdfunding platform inspired by Kickstarter, built on the Ethereum blockchain. It allows users to create and fund campaigns in a transparent, secure, and decentralized manner using smart contracts. The backend logic is written in Solidity, while the frontend interface is built using React.js, providing a complete decentralized application (DApp) experience.

Technologies Used

- **Solidity** – For writing the smart contract (`Crowdfunding.sol`)
- **Truffle** – For compiling, deploying, and testing smart contracts
- **Ganache** – A local Ethereum blockchain used for development
- **React.js** – For building the user interface
- **Web3.js** – To connect the frontend with the Ethereum blockchain
- **MetaMask** – For wallet integration and transaction signing

- **OpenZeppelin** – Used for security features like `ReentrancyGuard`

Backend (Smart Contract)

The smart contract (`Crowdfunding.sol`) defines all backend functionality. It enables:

- Campaign creation with a funding goal and deadline
- Contribution of ETH to campaigns
- Claiming of funds by the creator if the goal is met
- Refunds for contributors if the goal is not met

All campaign and transaction data is stored on-chain, ensuring full transparency and immutability.

Core Functions:

- `createCampaign()`
- `contribute()`
- `releaseOrRefund()`
- `getCampaign()`
- `getContribution()`

Security is reinforced by using OpenZeppelin's `ReentrancyGuard`.

Frontend (React.js)

The frontend, built with React.js, allows users to interact with the deployed smart contract via MetaMask. It offers:

- A form to create new campaigns
- A dashboard to view existing campaigns
- Functionality to contribute to a campaign
- Buttons to trigger fund release or refund

The frontend uses Web3.js to read from and write to the blockchain using the deployed contract's ABI and address.

Workflow Summary

1. Start Ganache to simulate a local Ethereum blockchain
2. Compile smart contracts using `truffle compile`
3. Deploy them with `truffle migrate`
4. Paste the deployed contract address into the frontend code
5. Launch the frontend using `npm start` inside the `/client` folder
6. Interact with the DApp via MetaMask

Security Measures

- **Reentrancy Protection**: Blocks nested function calls using `nonReentrant` modifier
- **Input Validation**: Ensures goals, deadlines, and contributions are valid
- **Event Logging**: All key actions like campaign creation and refund are recorded via events

Smart Contract Optimizations

To ensure the smart contract is not only secure but also gas-efficient and performant, the following optimizations were made:

1. **Reentrancy Protection**:
   Used `ReentrancyGuard` from OpenZeppelin to block recursive withdrawal attacks.
2. **Efficient Refund Logic**:
   Refunds are triggered per user request rather than looping through all contributors, saving gas.
3. **Storage Optimization**:
   Grouped related variables in the `Campaign` struct and used `memory` for function arguments to reduce storage costs.
4. **Require Checks First**:
   All validations using `require()` are placed at the start of functions to avoid unnecessary computation.

5. **One-Call Resolution**:
   The `releaseOrRefund()` function was designed to handle both fund transfers and refunds in a single call.
6. **Gas-Safe Transfer Order**:
   ETH transfers occur only after state variables are updated, ensuring transaction safety and gas efficiency.

Conclusion

This decentralized crowdfunding platform demonstrates how blockchain can be used to build trustless, secure, and transparent fundraising solutions. It combines smart contract logic, frontend interaction, and real blockchain simulation through Ganache.

The project emphasizes good security practices and performance optimizations. It can be scaled in the future to include additional features like milestone-based funding, reward distribution, and on-chain governance.