Blockchain Technology 1

Final Project

Title: Blockchain-based Charity Crowdfunding Platform AsarChain

Team members: Artur Jaxygaliyev, Alikhan Korazbay, Daryn Musseyev

Instructor: Sayakulova Zarina

Abstract

Traditional crowdfunding platforms are still using the same centralized architecture that has been around for decades to allow contributions to be made for funding projects or ventures on the Internet. With time, no significant improvements have been made to solve the flaws of the system, and the problems have remained intact. The lack of transparency is the cause of the low trust and credibility between funders and fundraisers, therefore, the reason for the low success of campaigns and platforms. Additionally, information such as bank credentials is not secure and malicious actions can occur. A lot of situations where you don't know where your money goes. A lot of advertisements in social media, were children or adults with fatal illnesses, you cannot trust them. This gives our team an opportunity to create web-based application. The system is connected to metamask and users must have an Ethereum account to login to their account on metamask to get access to the crowdfunding system. Shortly, this is the charity system against scammers.

Introduction

In Kazakhstan, charitable crowdfunding platforms often face transparency issues due to centralized intermediaries, which undermines the trust of donors. Blockchain technology, by offering immutable and publicly verifiable transaction records, provides a potential solution to this problem by enabling independent verification of donation flows. Through smart contracts, financial processes can be executed according to predefined rules without reliance on centralized control.

This project presents a hybrid blockchain-based charity crowdfunding platform that integrates Ethereum smart contracts for the transparent collection and verification of donations with off-chain services for managing campaign data and user interactions. Such an architectural approach allows the system to balance the security and immutability of blockchain with the practicality and efficiency of conventional web technologies.

The primary goal of the project is to increase trust and accountability in charitable crowdfunding within Kazakhstan by reducing opportunities for fraudulent activities. Developed as part of an academic study, the platform serves as a practical demonstration of how blockchain technology can be applied to address transparency challenges in real-world charitable systems.

Problem Statement

Nowadays, a lot of people just upload videos of people crying in hospital rooms with gauze, and in description write references of their card numbers, or phone number. People believe that they are doing a good thing, while they just get scammed. The lack of transparency and verifiable mechanism for tracking the donation flows makes it difficult both for donors to get real help, and funders to trust someone. Consequently, the core problem addressed in this project is the lack of a technological framework that ensures transparent, verifiable, and tamper-resistant donation tracking in charitable crowdfunding platforms.

Motivation

The motivation for this project is based on real-world experiences with online charitable donations, where funds were transferred without any reliable way to verify their subsequent use. In such cases, donors are required to rely solely on trust, hoping that their contributions are used properly and reach individuals who genuinely need assistance. This lack of verifiable transparency can reduce confidence in charitable crowdfunding platforms and discourage continued participation in philanthropic initiatives. These concerns motivate the search for technological approaches that provide stronger accountability guarantees. Blockchain technology offers mechanisms for recording donation transactions in an immutable and publicly verifiable manner, enabling donors to independently confirm the flow of funds and increasing trust in charitable systems.

Solution

Our proposed solution is a hybrid blockchain platform for crowdfunding in charity, aimed at increasing transparency and accountability. We have created a system by combining Ethereum smart contracts for financial actions with offline services to manage non-financial campaign data. This approach helps us circumvent the limitations of centralized platforms while maintaining convenience and efficiency.

Transparency is a key element. All transactions are made through smart contracts and recorded in the Ethereum blockchain. This makes each contribution publicly available and verifiable. Donors no longer need to rely on the platform's reports, as the entire transaction history is visible directly on the blockchain.

An important aspect of the solution is immutability. A transaction that has been confirmed and added to the blockchain cannot be changed or deleted. This ensures the integrity of donation data and prevents manipulation. The blockchain storage ensures that the donation history remains unchanged and protected from unauthorized access.

The system reduces dependence on intermediaries by using smart contracts instead of manual management. On conventional platforms, money management depends on third parties. In our system, the rules for accepting and tracking donations are recorded in smart contracts and executed automatically by the blockchain network. This reduces the risk of abuse and eliminates centralized financial control.

The platform ensures automatic compliance with financial regulations. Smart contracts are executed uniformly, ensuring that logic is applied to all users without personal opinion. Trust in people is replaced by trust in the code that brings the behavior of the system in line with the stated logic.

By separating financial logic from data management, we achieve a balance between decentralization and practicality. Blockchain is used where transparency is needed, and web technologies are used for efficient data storage and user interaction. This design shows the practical application of blockchain in charity crowdfunding systems.

System Architecture

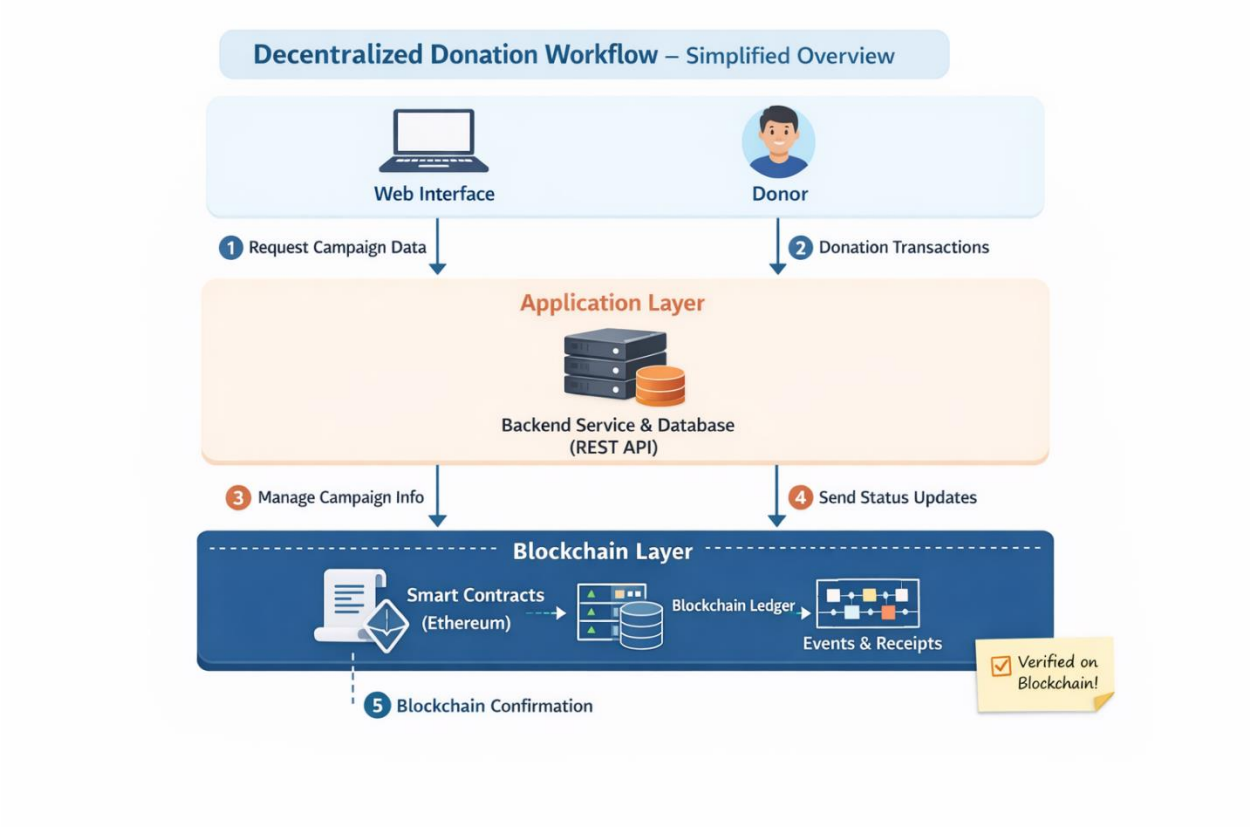
The system is built on a mixed structure, where there is a part for displaying data, a part for running the program, and a part for the blockchain. Each of them is responsible for their own and works with others through clear channels.

What the user sees is made like a website. There he looks at the data on the operation of the system, enters his data and does something, for example, creates a new task or transfers money. This part communicates with the server via the REST API and with the blockchain via a wallet in the browser.

The server part is a service that provides a RESTful API. It monitors data that is not related to money, such as information about the operation of the system and its status and stores them in a database. It checks all requests and monitors the status of the program but does not process financial transactions.

The blockchain consists of Ethereum smart contracts running on the test network. It processes money transfers, updates the financial status of the network, and creates notifications that the transfer has been completed. All financial records cannot be changed here, and anyone can check them.

The interface takes data about the system operation from the server. Money transfers start via the interface and go directly to the blockchain through the user's wallet. After confirming the transfer, the server updates the system status based on events in the blockchain. This allows you to transparently process financial data and at the same time effectively manage the data separately.



1. Campaign Creation

Users can initiate a fundraising effort by calling the createCampaign function.

Parameters: The creator provides a title, a funding goal (in Wei/ETH), and a duration (in days).

Storage: The contract calculates the deadline by adding the duration to the current block timestamp. It stores the campaign in an array, recording the creator's address to ensure only they can finalize it later.

Events: A CampaignCreated event is emitted, allowing the frontend to immediately detect and display the new campaign.

2. Donation Tracking

The contract manages donations through a mapping and a state variable within each campaign.

State Updates: When user calls donate, the totalCollected for that campaign increases, and the specific amount donated by that user is recorded in the donations mapping. Incentive Mechanism: For every 1 ETH donated, the contract automatically triggers the CharityToken contract to mint 1 CRT (Charity Token) to the donor as a "Proof of Donation" reward.

3. Deadline and Goal Enforcement

The contract uses strict requirements to ensure the integrity of the fundraising process:

Active Period: The donate function checks block.timestamp campaign.deadline. Once the deadline passes, no further donations are accepted.

Finalization: A campaign can only be finalized if the goal is reached OR the deadline has passed.

Fund Security: Funds are held securely within the contract until finalizeCampaign is called by the campaign creator, at which point the total amount collected is transferred to their wallet.

4. Transparency and Accounting

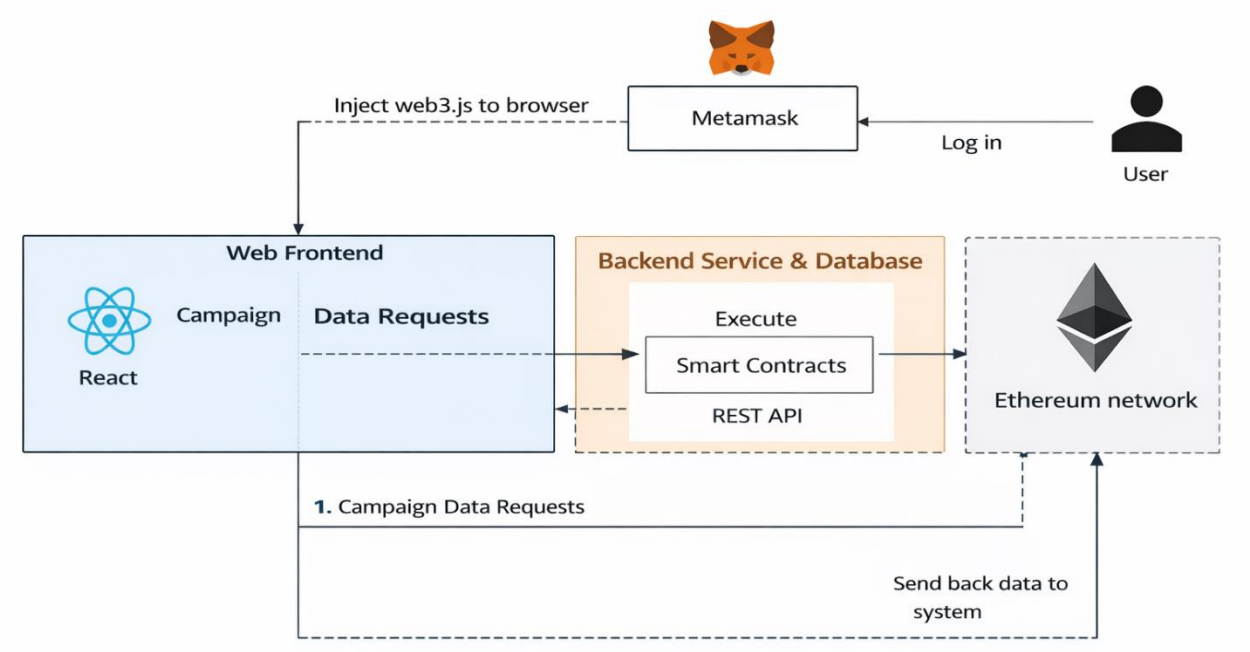
Transparency is achieved through the immutable nature of the Ethereum blockchain:

Public Verification: Every donation and campaign creation is recorded as a transaction. Anyone can verify the totalCollected vs. the goal by querying the public getCampaign function.

Event Logging: By emitting events like DonationMade and RewardMinted, the contract provides a transparent "paper trail" that block explorers and frontends can use to show a real-time history of all charitable activities.

Direct Impact: Since the contract handles the funds, there are no middle-men; donors can be certain their ETH is being managed according to the pre-defined code

Workflow



The user works with the system via the web interface. Authentication and access to the blockchain are provided by MetaMask, which adds a Web3 provider to the browser. This allows the interface to request the signature of transactions and interact with Ethereum without showing the secret keys.

The web interface shows information about the campaign and sends data requests. It receives campaign metadata from the server service via the REST API, requesting campaign details and updating other information.

The server service and database manage external data, such as campaign descriptions, categories, and status. This layer processes requests from the interface, stores information in the database, and returns updated data. He does not deal with financial transactions directly.

Donations are made through smart contracts in Ethereum. When a user wants to make a donation, the interface starts a transaction on the blockchain via MetaMask. The transaction goes to the Ethereum network, where a smart contract processes the donation and writes it to the blockchain.

After the transaction is completed, Ethereum provides confirmation data, including events and receipts. This information is returned to the system and the application status is updated, meaning the interface shows the confirmed donation status. This architecture separates financial logic and data management, ensuring transparency and verifiability of donations.

CharityCrowdfunding.sol

The CharityCrowdfunding.sol contract implements the core crowdfunding logic of the platform. It manages the creation of campaigns, processes donations, enforces deadlines, and controls the release of collected funds without relying on centralized intermediaries.

The contract allows users to create campaigns by defining a funding goal and a deadline. Donations are made in test ETH and are recorded on-chain, updating the total amount raised for each campaign. Campaign finalization is restricted by predefined conditions, ensuring that funds can be claimed only after the campaign goal is reached or the deadline has passed. View functions are provided to supply real-time campaign data to the frontend.

On-chain data includes campaign records containing the creator address, funding goal, deadline, and total ETH collected. Additionally, a mapping tracks individual donation amounts per wallet address for each campaign, enabling transparent and immutable contribution tracking.

CharityToken.sol

The CharityToken.sol contract implements a custom ERC-20 token used as a proof of donation. The token has no real monetary value and is issued solely for educational purposes.

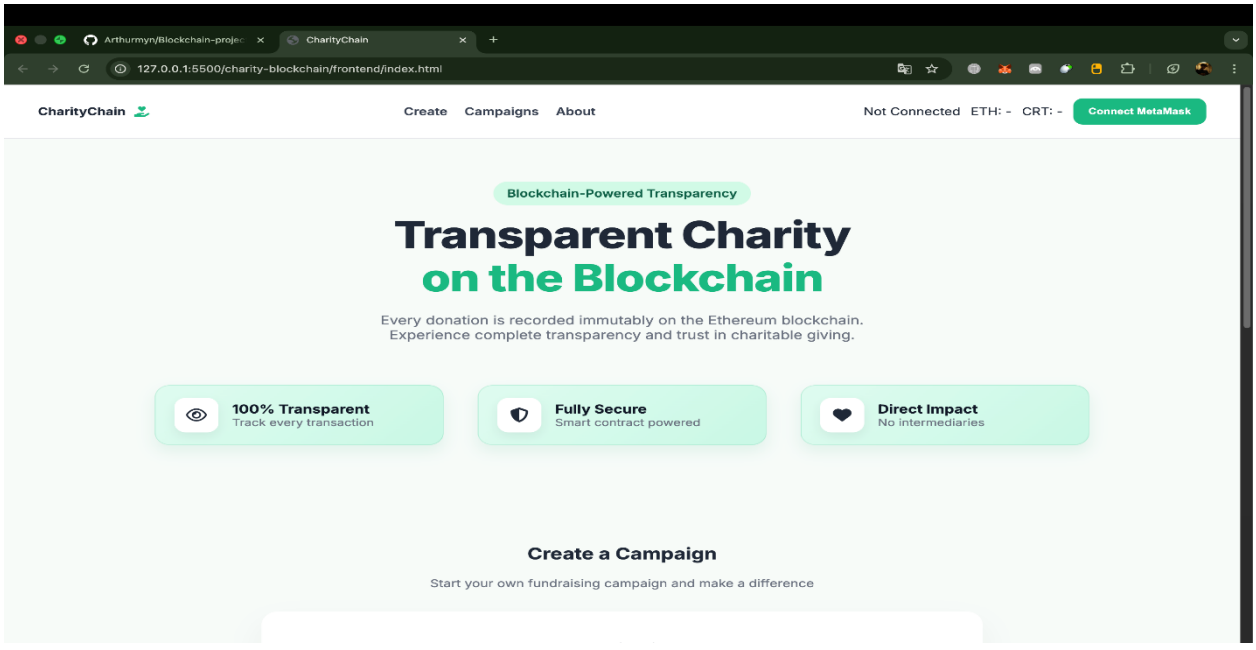
The contract includes standard ERC-20 functionality and a restricted mint function. Token metadata, balance mappings, and total supply are stored on-chain, providing a verifiable record of token ownership.

Token Minting Mechanism

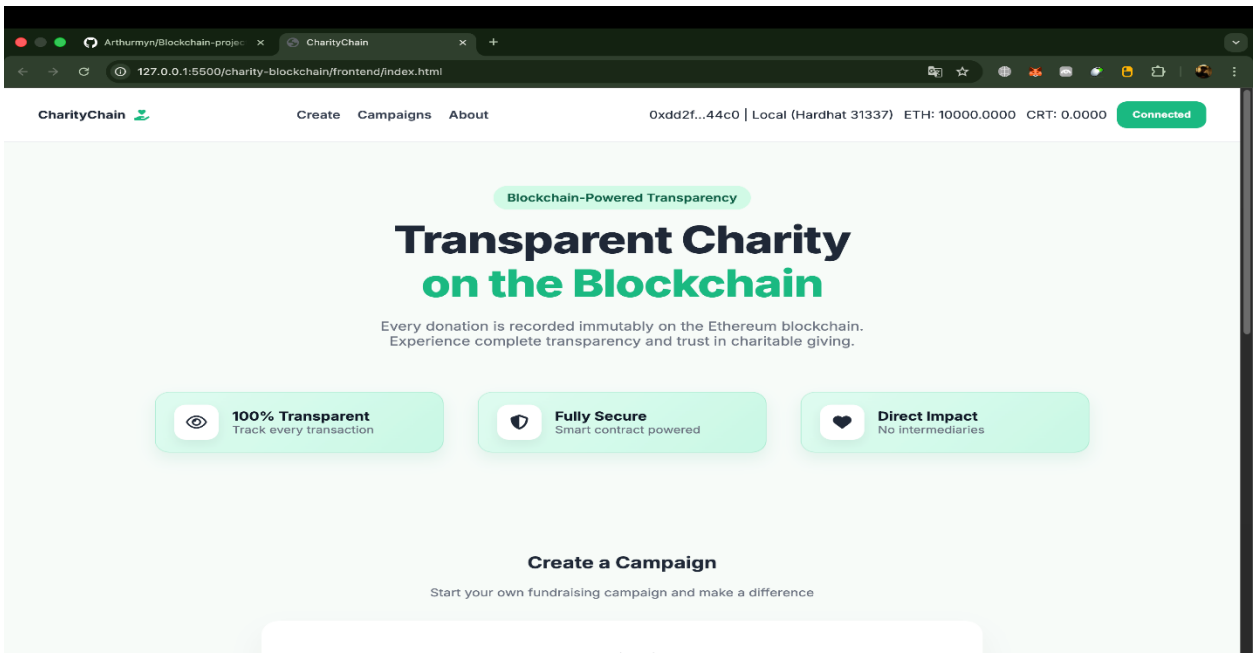
Token minting is triggered during the donation process. When a donor sends ETH via the donate() function, the crowdfunding contract calculates the reward amount using a fixed 1:1 ratio between ETH and CRT tokens. The crowdfunding contract then calls the token contract's mint function.

Minting is restricted to the crowdfunding contract, which is set as the token owner during deployment. Upon successful execution, CRT tokens are minted and transferred directly to the donor's wallet, serving as an on-chain proof of participation.

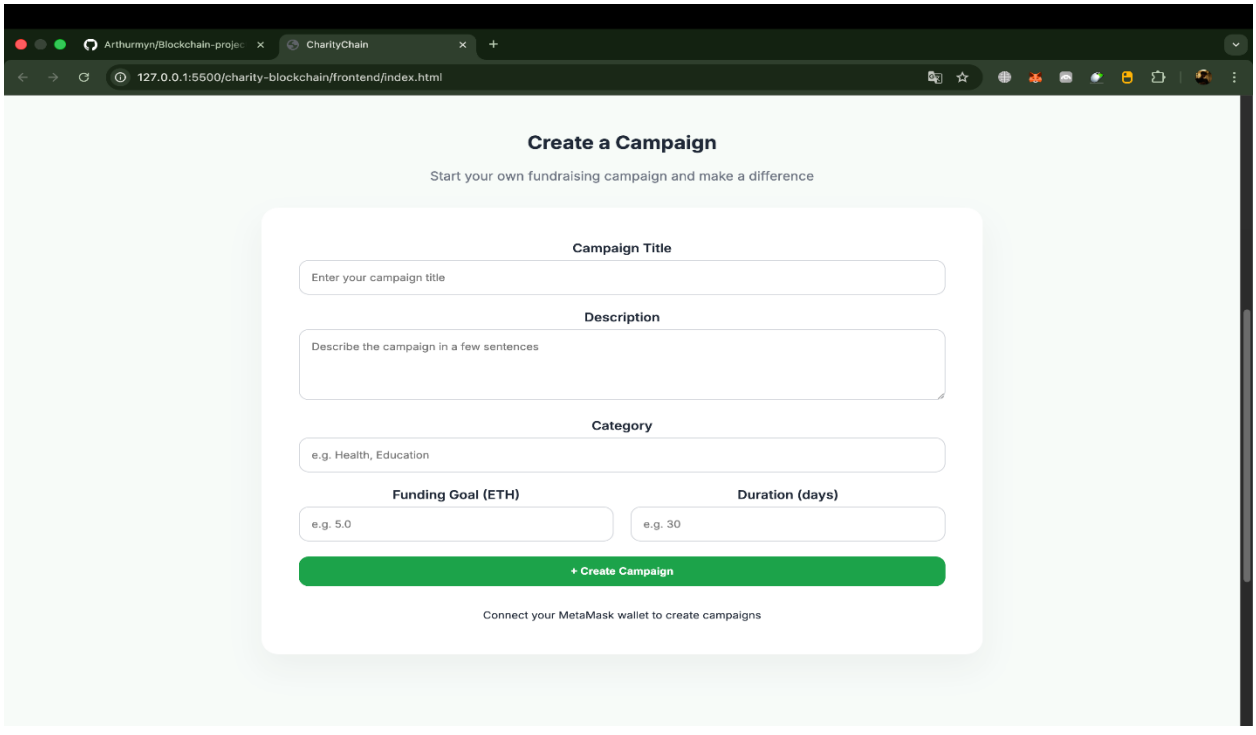
Practical Demonstration



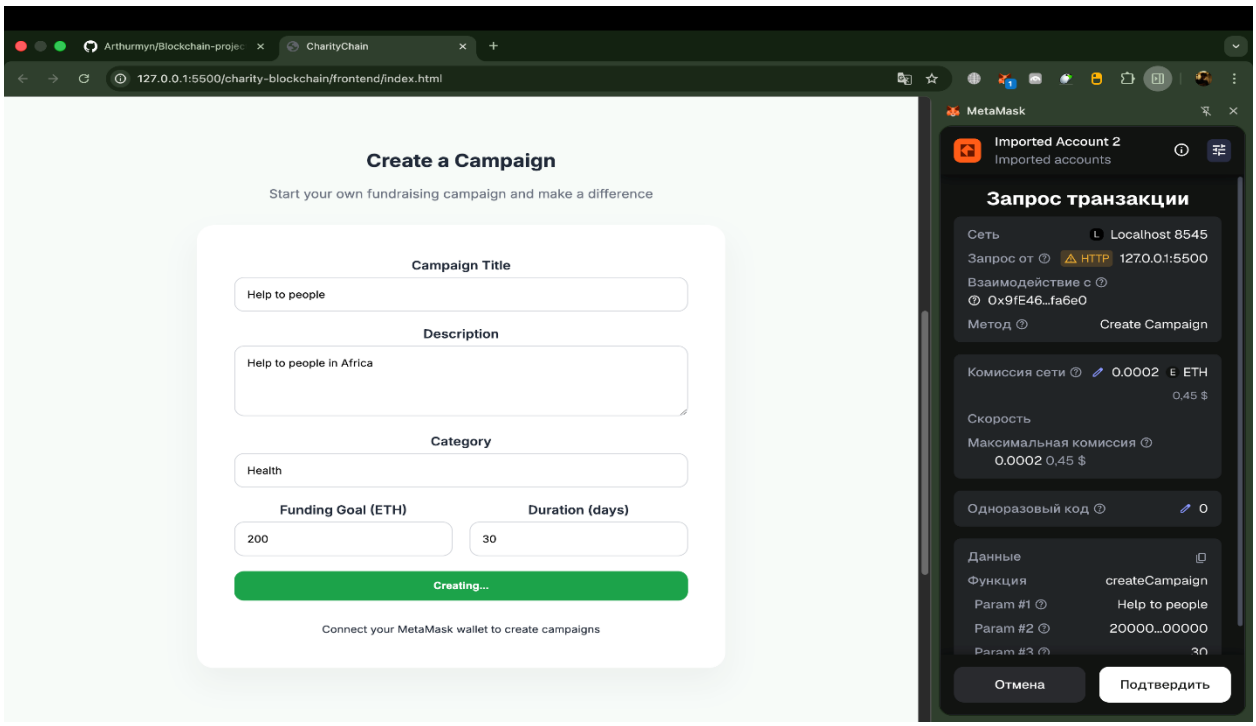
Description: This is our web application where Metamask is not connected, so that's why, blockchain based features are off. So user must authorize first, then interact with smart contracts.



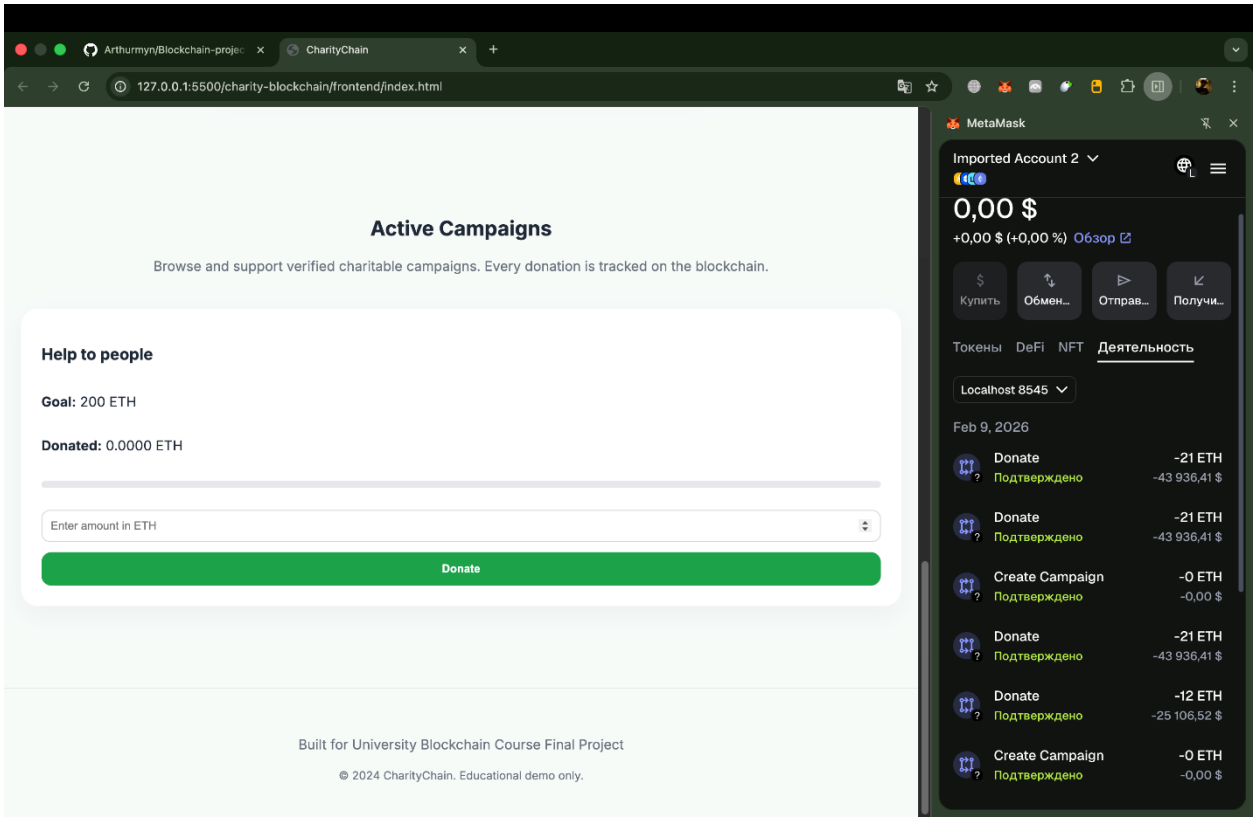
Description: This is our web application where Metamask is connected. After clicking the "Connect MetaMask" button, the user authorizes the application through the MetaMask extension. Once connected, the interface displays the active wallet address, the local Hardhat network identifier, and the available ETH and CRT token balances.



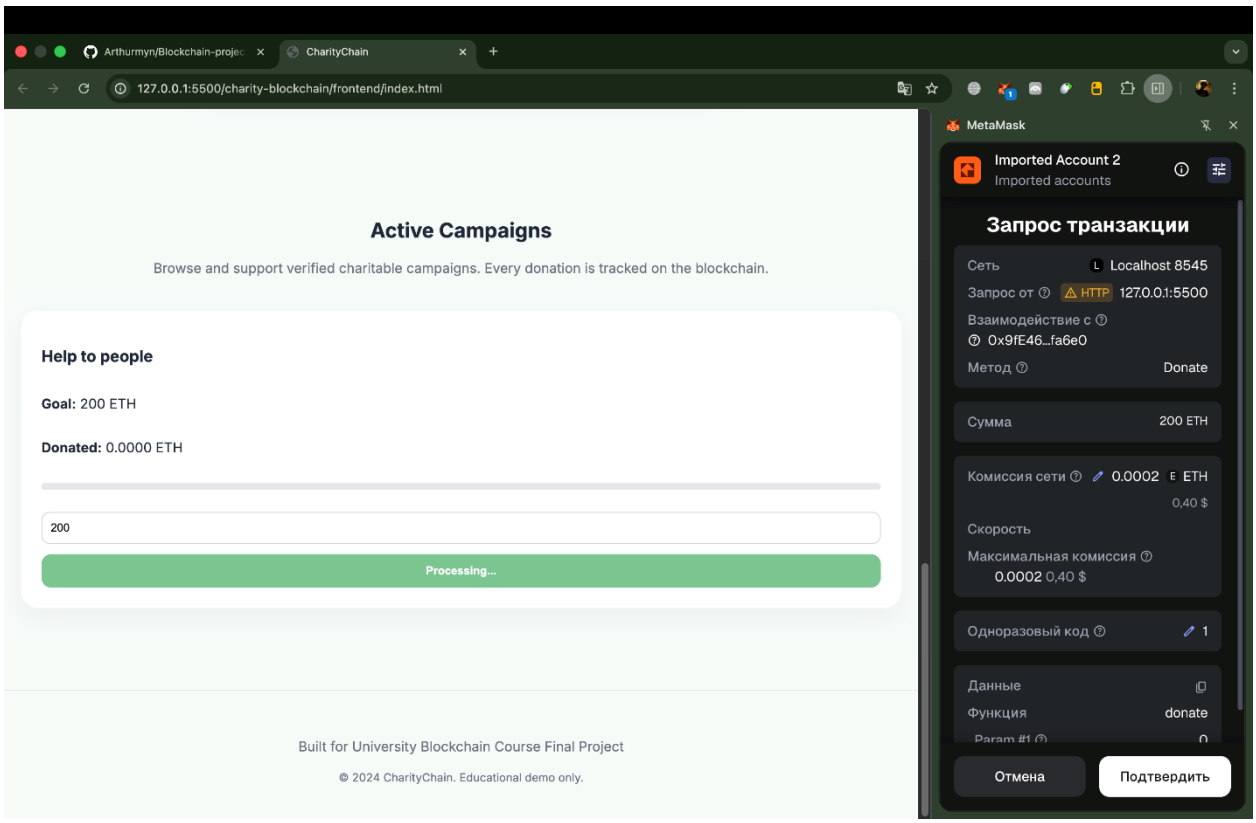
Description: The campaign creation screen allows a connected user to define a new fundraising campaign. The user specifies the campaign title, description, category, funding goal in ETH, and duration in days. These parameters are validated on the frontend and then submitted to the crowdfunding smart contract via MetaMask.



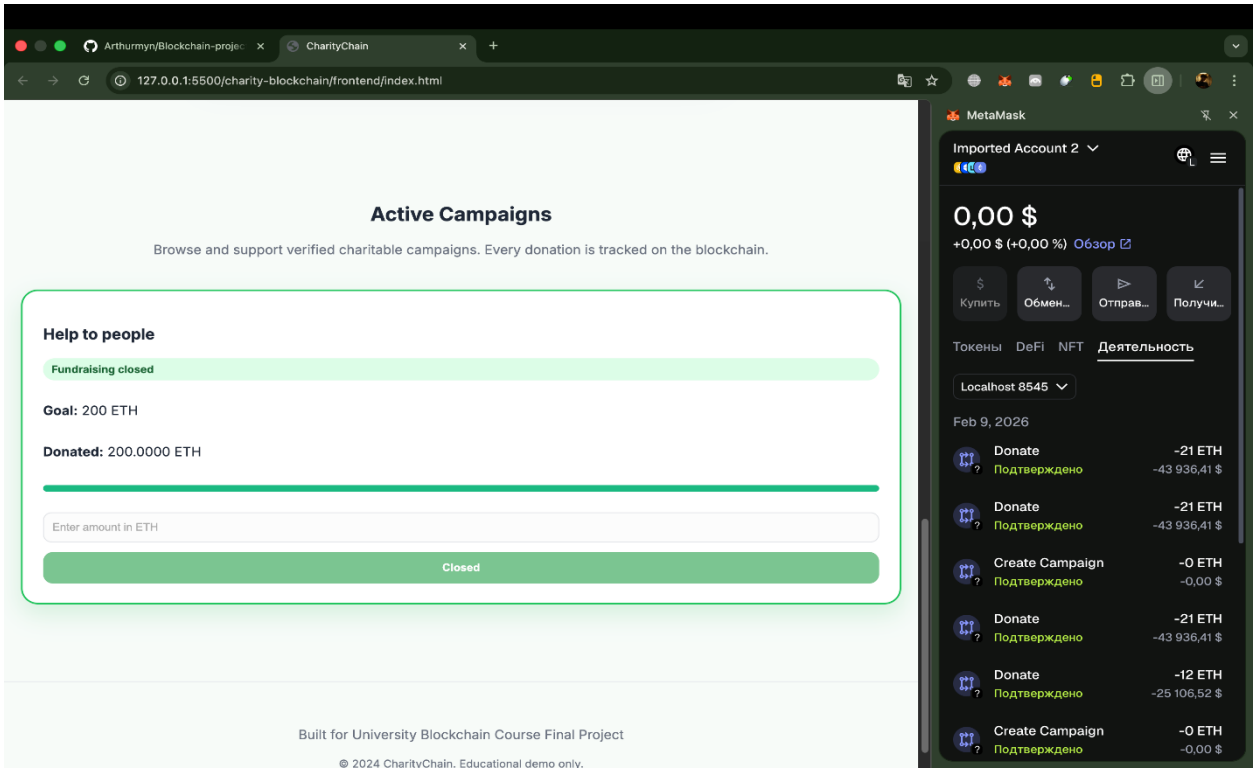
Description: MetaMask displays a transaction confirmation window. This screen shows the target smart contract address, the invoked method (createCampaign), and the associated network fee. Confirming this transaction deploys the campaign data to the blockchain, ensuring that campaign parameters are stored immutably.



After successful transaction confirmation, the newly created campaign appears in the “Active Campaigns” section. The info about campaign is retrieved directly from the smart contract, ensuring real-time synchronization with the blockchain state.



Description: The donation interface allows users to enter a specific amount of ETH and initiate a donation. After clicking donate button, MetaMask prompts the user to confirm the transaction. The donation is then sent directly to the smart contract, which means we don't use any DB.



Description: The MetaMask activity log shows the successful donation transaction, verifying that the ETH transfer has been executed on the Ethereum test network. This confirms transparent and verifiable handling of funds.

Description: After successful donations, the user's CRT token balance increases, as displayed in the interface. These tokens are minted automatically by the token contract and serve as a proof of donation.

Team contribution:

Arthur - responsible for establishing the project repository and version control workflow. This included creating and managing the GitHub repository, organizing branches, and defining the initial project structure. Arthur also implemented the core frontend layout by developing the functional index.html file and provided the initial skeleton of the smart contract components, forming the foundation for further blockchain development.

Daryn - focused on the implementation of the off-chain interaction layer. This work included integrating MetaMask functionality into the application, ensuring that wallet connection and donation actions operated correctly through the user interface. Daryn also contributed to the architectural design of the system by preparing the presentation materials and developing the system architecture flow diagram used to illustrate component interactions.

Alikhan - contributed to the refinement and extension of the off-chain logic, ensuring that all blockchain interactions were restricted to the Ethereum test network and utilized test ETH exclusively. This included adding Web3-related scripts and deployment utilities. Alikhan was also responsible for preparing the project documentation, including the README.md file and the comprehensive PDF technical report.

Conclusion

This project presented the design and implementation of a hybrid blockchain-based charity crowdfunding platform focused on improving transparency and trust in donation processes. By integrating Ethereum smart contracts with a web-based backend, the system ensures that donation transactions are immutably recorded on the blockchain while non-financial campaign data is managed efficiently off-chain.

The platform fulfills the core functional requirements of a decentralized application operating on an Ethereum test network. It demonstrates secure MetaMask integration, transparent handling of test ETH donations, issuance of internal reward tokens, and real blockchain interaction. Through the development of this system, we applied key concepts of smart contract programming, frontend-to-blockchain communication, and decentralized application architecture in a practical and cohesive manner.

Future Work

We could extend the smart contract functionality to support automated campaign finalization, refund mechanisms, and more advanced donation tracking logic. Improvements may also include optimized event indexing to enhance synchronization between on-chain and off-chain data.

Further development could involve investigating decentralized storage solutions for campaign metadata, implementing enhanced verification mechanisms for campaign creators, and refining the user interface to improve usability. These extensions would allow the platform to move closer to real-world applicability while preserving the transparency, security, and trust guarantees demonstrated in this project.

References

Buterin, V. (2014). *Ethereum: A next-generation smart contract and decentralized application platform*. Ethereum Foundation.
https://ethereum.org/en/whitepaper/

Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system*.
https://bitcoin.org/bitcoin.pdf

Wood, G. (2014). *Ethereum: A secure decentralised generalised transaction ledger (Yellow Paper)*.Ethereum Foundation.
https://ethereum.github.io/yellowpaper/paper.pdf

Ethereum Foundation. (2024). *ERC-20 token standard*.
https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

Ethereum Foundation. (2024). *Events and logs in Ethereum*.
https://ethereum.org/en/developers/docs/smart-contracts/anatomy/#events-and-logs

MetaMask. (2024). *Connecting to MetaMask from a web application*.
https://docs.metamask.io/wallet/how-to/connect/

MetaMask. (2024). *Transaction confirmation and signing*.
https://docs.metamask.io/wallet/how-to/sign-data/

Market Data Forecast. (2024). *Crowdfunding market size, share, trends & growth analysis report*.
https://www.marketdataforecast.com/market-reports/crowdfunding-market

Nomic Foundation. (2024). *Hardhat local development network*.
https://hardhat.org/hardhat-network/