FINAL EXAMINATION PROJECT
Course: *Blockchain 1*

Milestone Fund

Decentralized Crowdfunding Application with Milestones

Our Group: Yesset Kozhakhmet, Iskander Ismagulov, Nuraly Zhandossov

Github link: https://github.com/Ne1ronn/MilestoneFund

1. Project Overview

This project develops a decentralized crowdfunding application using the Ethereum blockchain technology. The application enables users to create crowdfunding campaigns, make contributions of test ETH, and claim reward tokens. Project introduces a milestone-based crowdfunding system, where the funds are gradually released in stages. The application only runs on the Sepolia Ethereum test network and is for educational purposes only

2. Application Architecture

The system follows a decentralized application (DApp) architecture consisting of the following components:

- Frontend implemented using HTML and JavaScript
- MetaMask wallet for user authentication and transaction signing
- Crowdfunding smart contract responsible for campaign logic and ETH management
- RewardToken smart contract implementing an ERC-20 token

Users interact with the blockchain through the frontend application, which communicates with smart contracts using the Ethers.js library and MetaMask provider.

3. Smart Contract Design

3.1 Crowdfunding Smart Contract

The Crowdfunding smart contract manages all crowdfunding logic.
Each campaign contains the following properties:

- campaign creator address
- total amount of ETH raised
- campaign deadline
- list of milestones
- current milestone index

Each milestone defines a fixed ETH amount that can be released only after confirmation by the campaign creator.

Contributions are stored in the smart contract until a milestone is confirmed.
When a milestone is confirmed:

- the corresponding ETH amount is transferred to the campaign creator
- the milestone is marked as completed
- the next milestone becomes active

Only the campaign creator is allowed to confirm milestones.

3.2 RewardToken Smart Contract

The RewardToken contract implements an ERC-20 token using the OpenZeppelin library.

Key properties:

- tokens have no real monetary value
- tokens are minted automatically
- minting is restricted to the Crowdfunding contract

Ownership of the RewardToken contract is transferred to the Crowdfunding contract during deployment.
This ensures that reward tokens can only be minted as part of valid crowdfunding logic.

Reward tokens are minted proportionally to the amount of ETH released during each milestone.

Deploy the contracts:

```
yesset@Mac MilestoneFund % npx hardhat run scripts/deploy.js --network sepolia
[dotenv@17.2.4] injecting env (2) from .env -- tip: ⚙ suppress all logs with { quiet: true }
[dotenv@17.2.4] injecting env (0) from .env -- tip: 🔐 prevent building .env in docker: https://dotenvx.com/prebuild
Deployer: 0x4FeDd25f27B8C790bdD231dAEF87D0545dBE41FF
Balance: 0.051593595975894413 ETH
RewardToken deployed: 0x71E0f73a272E2f6712042d2433EDf535Ac0b94e1
Crowdfunding deployed: 0x138546d24c67A74b6e75E8113a62744aC8b39b28
RewardToken ownership transferred to Crowdfunding
```

4. Frontend and MetaMask Integration

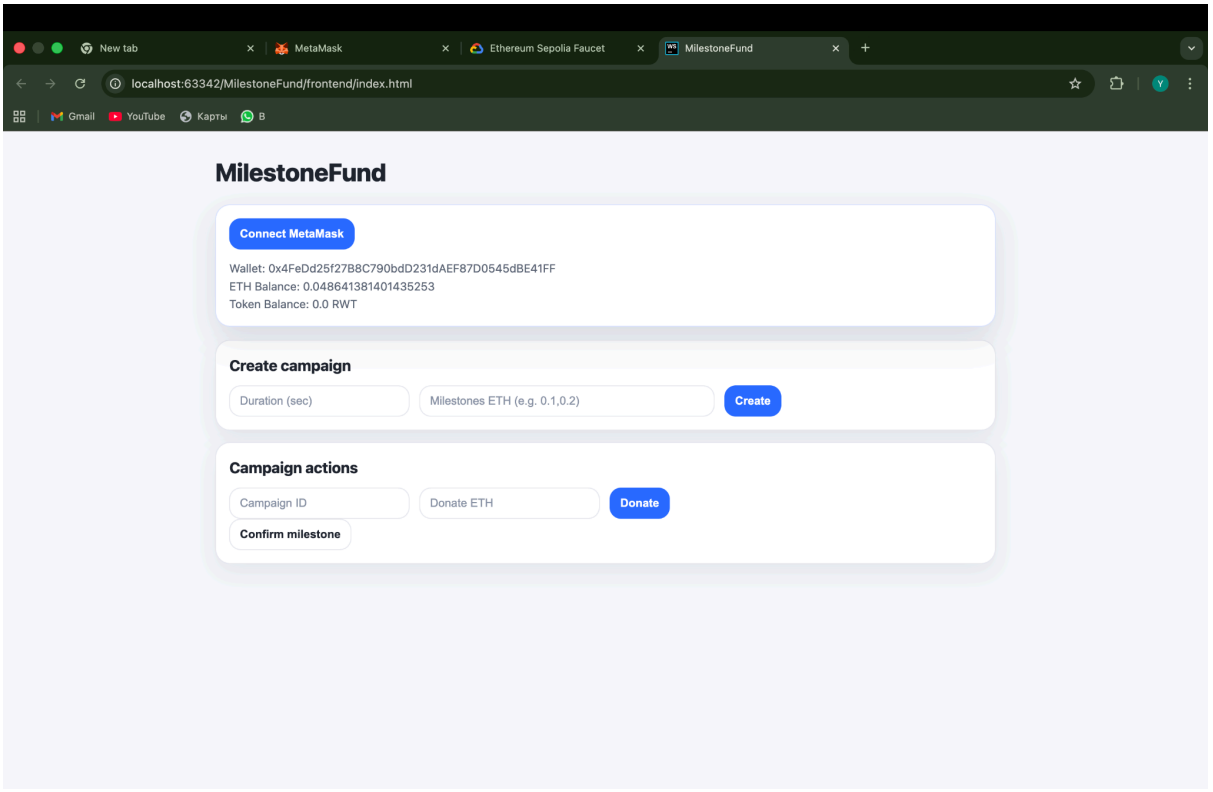
The frontend application provides user interaction with the blockchain through MetaMask.

Implemented features:

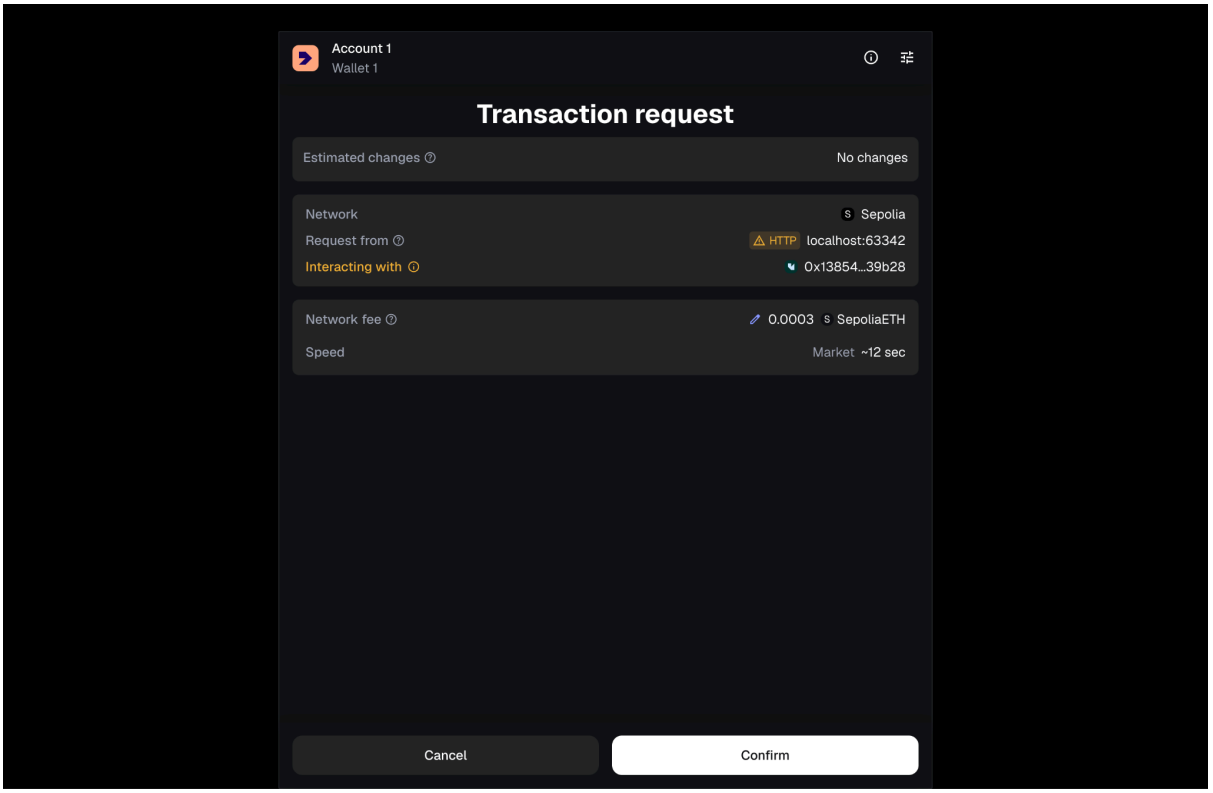
- requesting wallet connection using MetaMask
- verification of active Ethereum network (Sepolia)
- sending transactions for campaign creation and contributions
- confirming milestones
- displaying user ETH balance
- displaying ERC-20 reward token balance

All blockchain transactions are signed by the user through MetaMask.

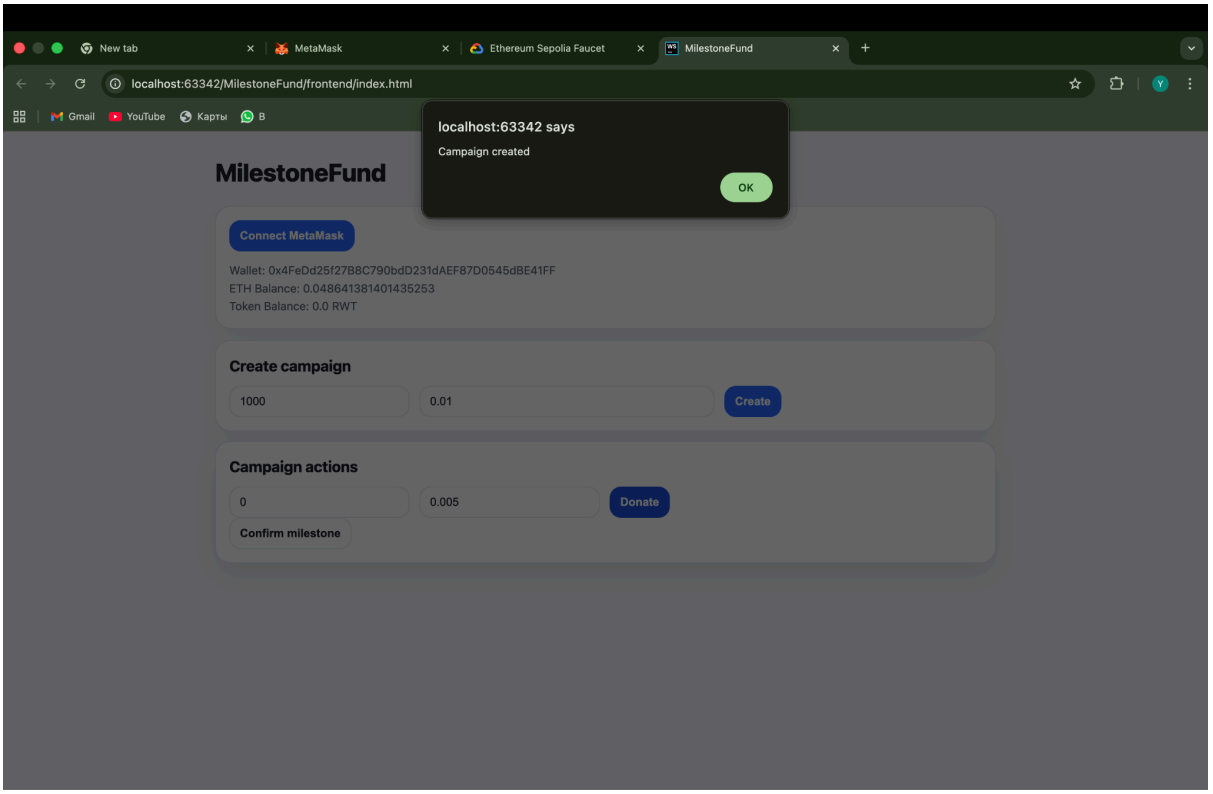
Connecting to Metamask:



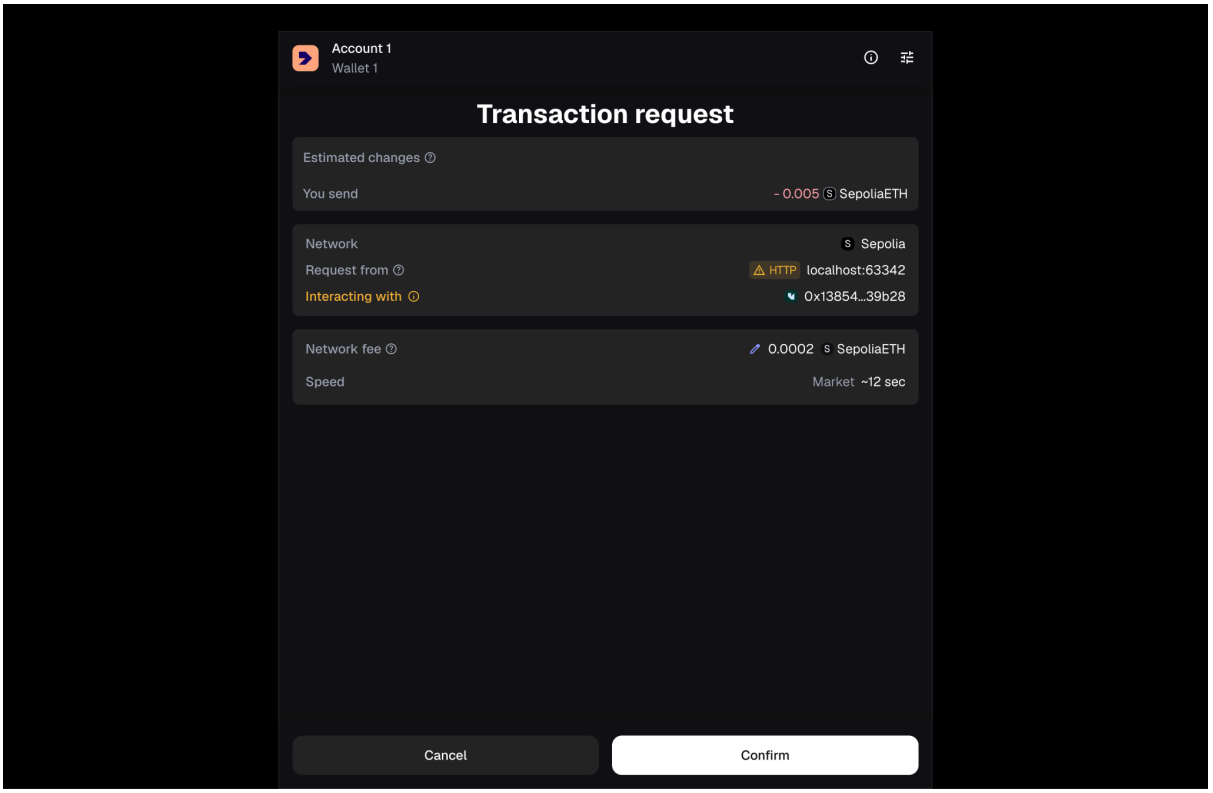
Creat campaign transaction:



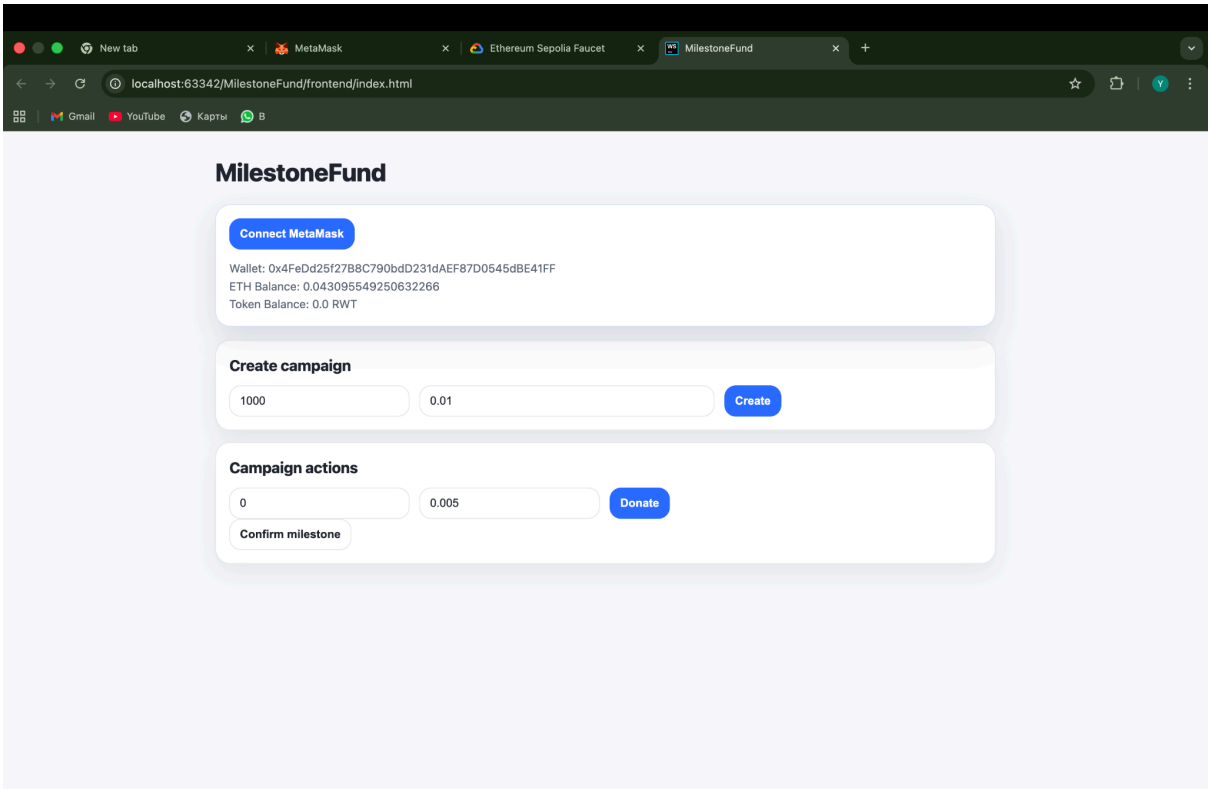
Successful transaction:



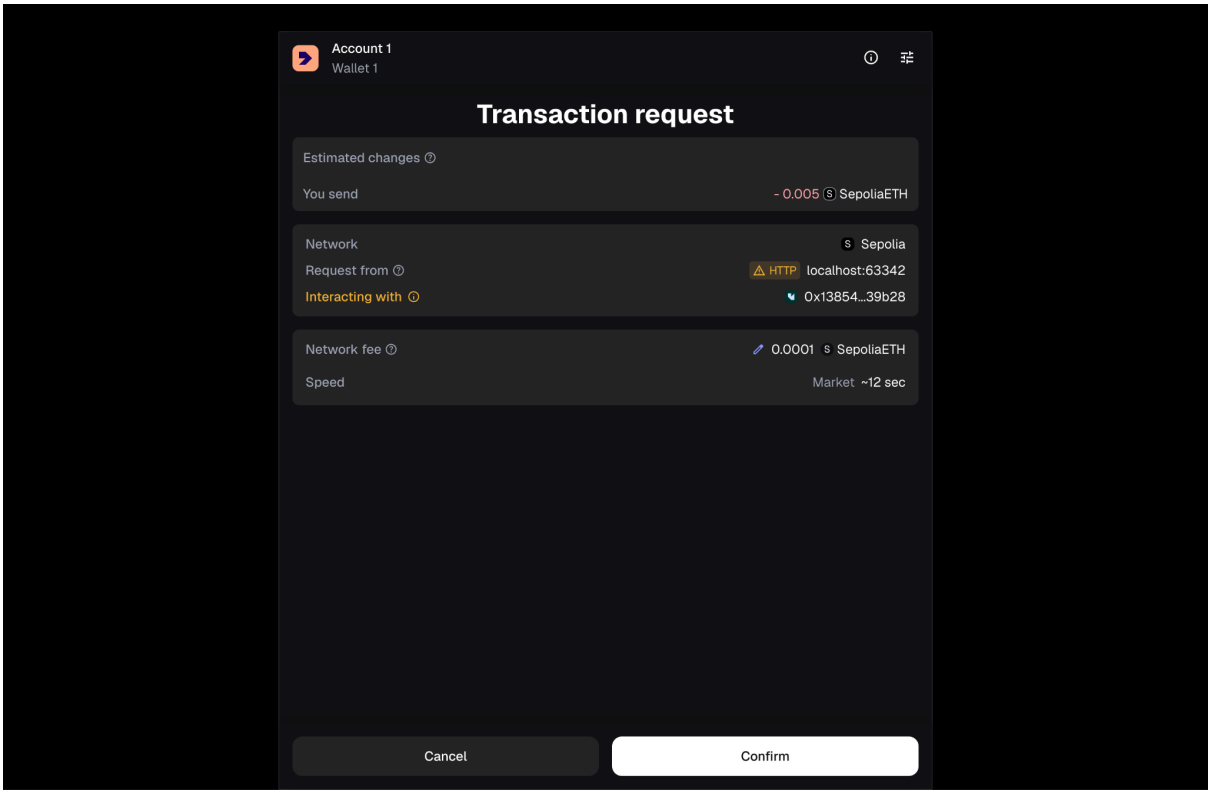
Donate Transaction:

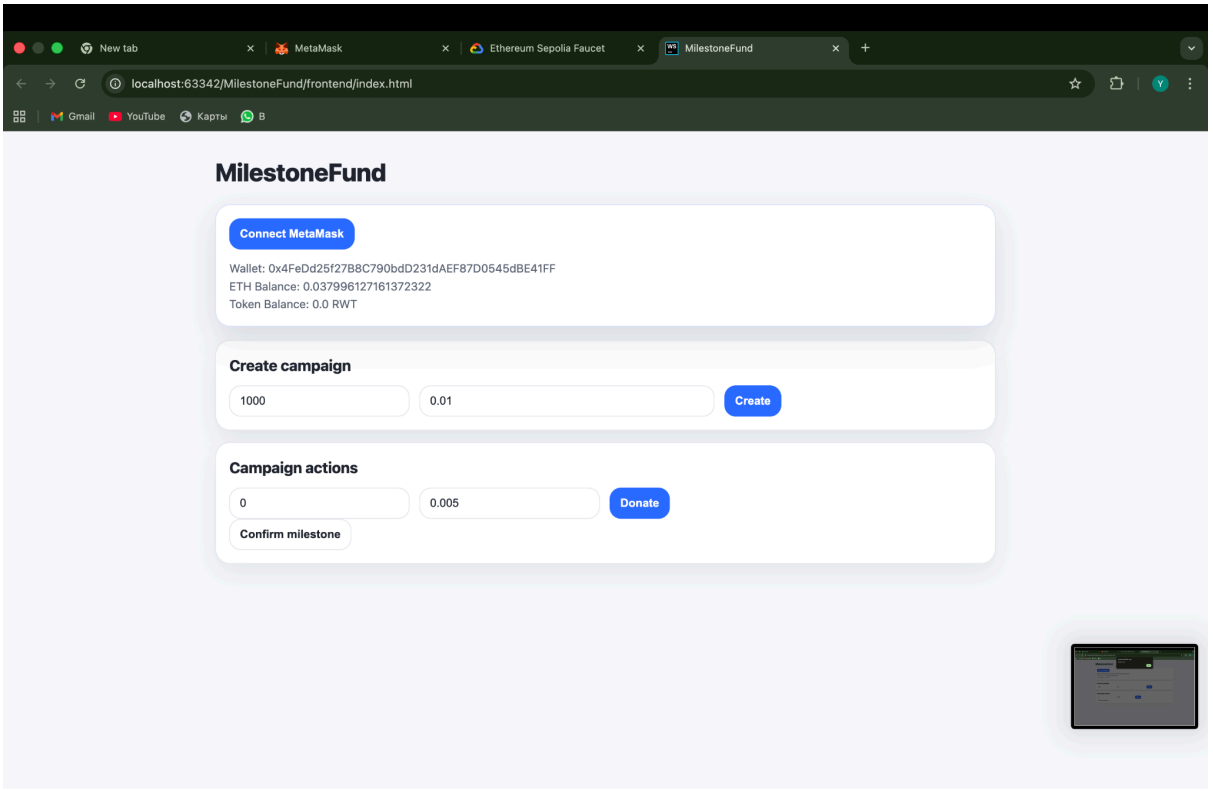


Successful transaction and balance change:

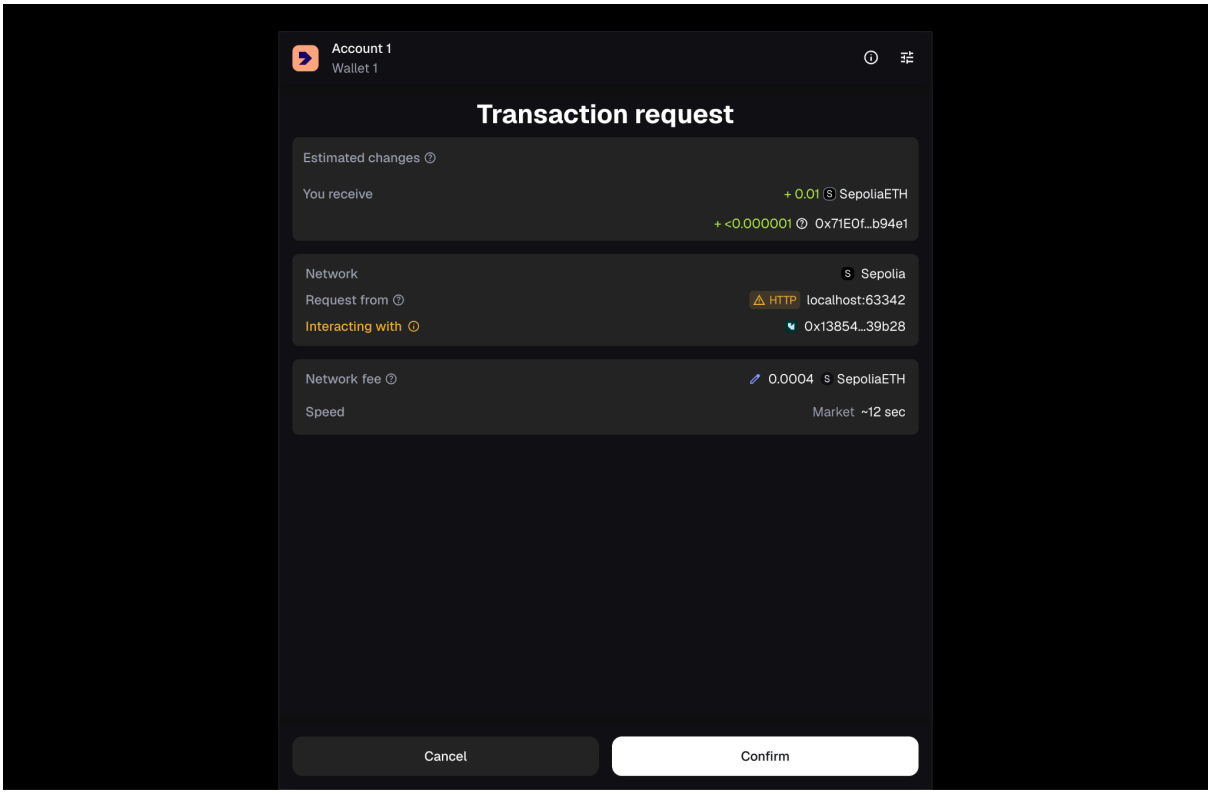


Second donate to test confirming milestone:

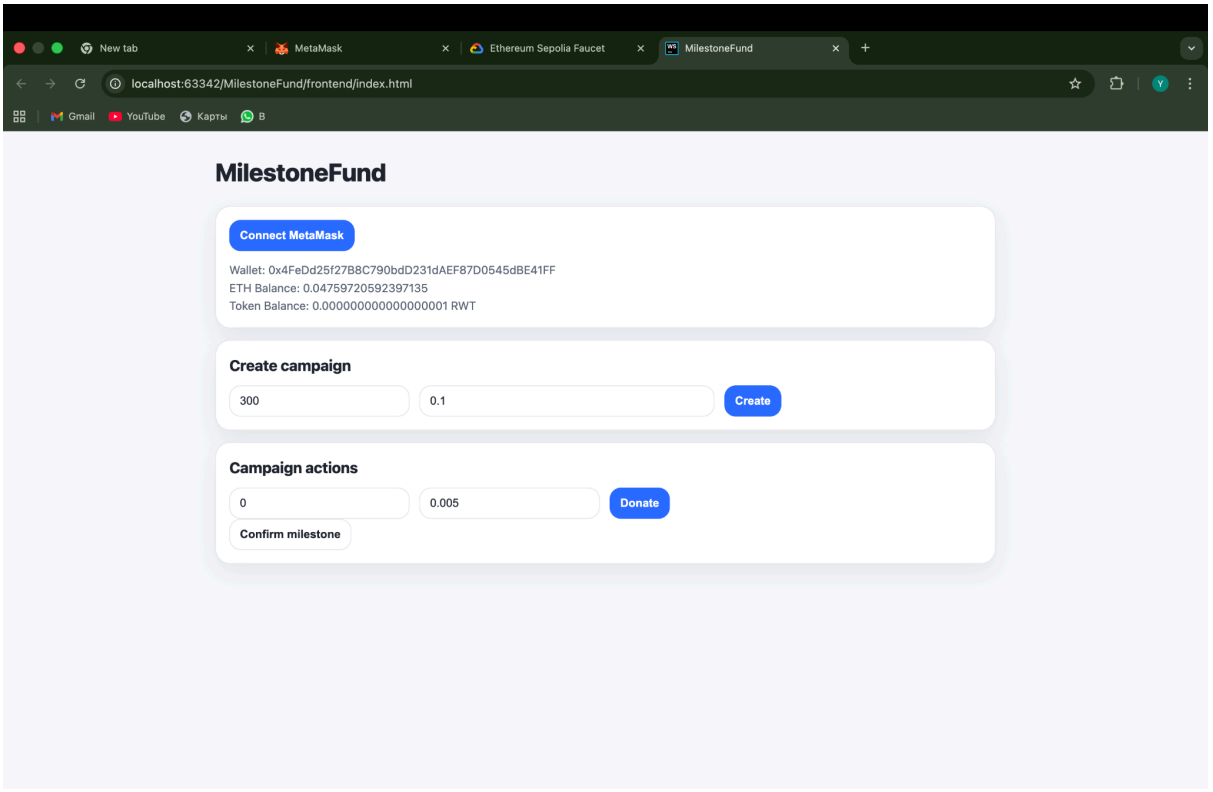




Confirm milestone:



Successful confirm and balance change:



5. Application Workflow

The typical application flow consists of the following steps:

1. User connects MetaMask wallet to the application
2. Campaign creator creates a new crowdfunding campaign with defined milestones
3. Users contribute test ETH to the campaign
4. ETH remains locked in the smart contract
5. Campaign creator confirms a milestone
6. ETH is released to the creator
7. Reward tokens are minted automatically
8. Process repeats until all milestones are completed

6. Deployment and Test Network Usage

The application is deployed and tested exclusively on the **Sepolia Ethereum test network**.

- Test ETH is obtained from public Sepolia faucets
- No real cryptocurrency is used
- Deployment is performed using Hardhat
- MetaMask is used for wallet management and transaction signing

Deployment on Ethereum mainnet is not used or supported.

7. Conclusion

This project demonstrates a complete decentralized crowdfunding application with milestone-based fund release and ERC-20 reward token issuance.

The system fulfills all functional requirements of the course project and demonstrates practical knowledge of:

- smart contract development
- decentralized application architecture
- MetaMask integration
- ERC-20 token usage
- Ethereum test networks

The project is intended solely for educational purposes.