

FINAL EXAMINATION PROJECT

Course: Blockchain 1

Format: Group Project Technology

Stack: Solidity, JavaScript, MetaMask, Ethereum (Testnet)

Team members: Dias N, Olzhas E, Alisher S.

PART 1 — Introduction & Problem Statement:

Blockchain is a distributed ledger technology where transactions are recorded in a secure and immutable way across a network of computers. Instead of relying on a centralized authority, blockchain uses consensus rules to validate and store data. A decentralized application (DApp) is an application that runs using smart contracts on a blockchain. The logic and rules are enforced by the smart contract, while users interact through a web interface and a wallet such as MetaMask.

In real life, many student teams have promising startup ideas but face difficulties in collecting initial funding. Traditional fundraising platforms depend on intermediaries, centralized payment systems, and may introduce extra fees or trust issues. In addition, contributors often have limited transparency: it is not always clear how funds are collected and distributed.

This project proposes a solution called **Student Startup Booster** — a decentralized crowdfunding platform implemented as a DApp. The system allows creators to publish crowdfunding campaigns on an Ethereum test network and allows contributors to send test ETH to support campaigns. The fundraising process is transparent because all transactions are recorded on-chain, and campaign status can be verified directly from the blockchain. To demonstrate tokenization concepts, contributors receive internal ERC-20 reward tokens proportionally to their contribution amount. This reward token has no real monetary value and is used only for educational purposes.

PART 2 — Project Overview:

Project name: Student Startup Booster

Type: Decentralized Crowdfunding DApp (Ethereum test network)

Goal:

To create a blockchain-based crowdfunding system where users can create campaigns and other users can contribute test ETH using MetaMask, while receiving reward ERC-20 tokens for participation.

Key features:

- Create crowdfunding campaigns (title, goal, deadline/duration)
- Contribute test ETH to active campaigns
- Track contributions on-chain per user
- Finalize campaigns after deadline
- Mint and distribute ERC-20 reward tokens to contributors

User roles:

- **Creator:** creates a campaign and finalizes it
- **Contributor:** donates test ETH and receives reward tokens

PART 3 — Technology Stack:

- **Solidity** — smart contract development (crowdfunding logic + ERC-20 reward token)
- **OpenZeppelin Contracts** — secure ERC-20 and Ownable implementations
- **Hardhat** — local development environment, compilation, deployment scripts
- **Ethereum Test Network** — Sepolia / Holesky / localhost (only free test ETH)

FINAL EXAMINATION PROJECT

- **MetaMask** — wallet connection + transaction signing
- **ethers.js** — frontend interaction with deployed contracts
- **HTML + JavaScript** — frontend interface
- **GitHub** — version control, branches, pull requests, collaboration

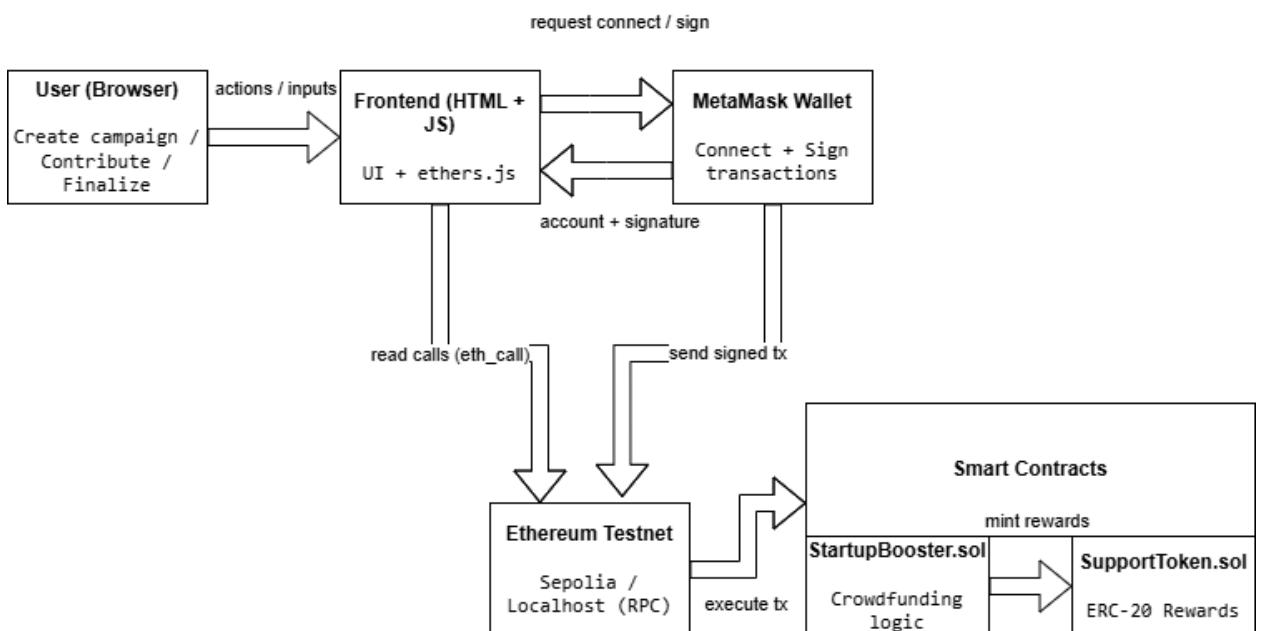
PART 4 — System Architectur:

Architecture description:

1. The user opens the web frontend in a browser.
2. The frontend requests connection to MetaMask and reads the selected network and wallet address.
3. MetaMask acts as a secure bridge between the UI and the blockchain: it signs transactions and sends them to the Ethereum test network.
4. The smart contracts (SupportToken and StartupBooster) store campaign state, contributions, and token minting rules.
5. The frontend reads on-chain data (campaign list, user balances) using ethers.js and displays it to the user.

Smart contracts:

- **SupportToken (ERC-20)**: reward token minted for contributors
- **StartupBooster**: crowdfunding logic (create campaign, contribute, finalize, reward mint)



The DApp consists of a web frontend (HTML/JavaScript) that communicates with the blockchain via ethers.js and MetaMask. The user

interacts with the UI to create campaigns, contribute ETH, and finalize campaigns. MetaMask handles account connection and transaction signing, then sends signed transactions to the Ethereum test network (Sepolia or localhost RPC). The network executes transactions on the deployed smart contracts. The core logic is implemented in StartupBooster.sol (crowdfunding features), while SupportToken.sol is an ERC-20 reward token. After a contribution, StartupBooster triggers minting of reward tokens to the contributor through SupportToken.

PART 5 — Smart Contracts Implementation:

5.1 SupportToken.sol

SupportToken is a custom ERC-20 token used as an internal reward for contributors. The token is created for educational purposes and has no real monetary value, which satisfies the assignment constraints.

FINAL EXAMINATION PROJECT

Main points:

- Inherits from **ERC20** (OpenZeppelin)
- Uses **Ownable** to restrict administration
- Has a minters mapping to control which addresses can mint
- Owner can enable or disable minter addresses via setMinter
- When a user contributes through the crowdfunding system, the authorized contract (StartupBooster) can mint reward tokens for that user

```

SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract SupportToken is ERC20, Ownable {
    mapping(address => bool) public minters;

    constructor() ERC20("SupportToken", "SUP") Ownable(msg.sender) {}

    function setMinter(address minter, bool allowed) external onlyOwner {
        minters[minter] = allowed;
    }

    function mint(address to, uint256 amount) external {
        require(minters[msg.sender], "Not minter");
        _mint(to, amount);
    }
}

PS C:\Users\Admin\OneDrive\Documents\Diagrams\Blockchain Final\student-startup-booster> git commit -m "fix: SupportToken mint uses _mint"
PS C:\Users\Admin\OneDrive\Documents\Diagrams\Blockchain Final\student-startup-booster> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.

```

5.2 StartupBooster.sol

StartupBooster is the main crowdfunding contract. It stores campaigns and handles contributions and finalization. It also integrates tokenization: after contribution, contributors receive SupportToken rewards proportional to the contribution amount.

Main points:

- Campaign structure includes: title, goal, deadline, owner, raised amount, status
- createCampaign(...) creates a new campaign on-chain
- contribute(...) sends ETH to a campaign and stores per-user contribution data
- finalize(...) is used after deadline to finalize success/failure and release logic
- Reward token minting is triggered after a successful contribution (through SupportToken)

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

interface ISupportToken {
    function mint(address to, uint256 amount) external;
}

contract StartupBooster {
    struct Campaign {
        address creator;
        string title;
        uint256 goal;
        uint256 deadline;
        uint256 pledged;
    }

    ISupportToken public token;

    uint256 public campaignCount;
    mapping(uint256 => Campaign) public campaigns;

    constructor(address tokenAddress) {
        token = ISupportToken(tokenAddress);
    }

    function createCampaign(
        string calldata title,
        uint256 goal,
        uint256 duration
    ) external {
        require(goal > 0, "Goal must be > 0");
        require(duration > 0, "Duration must be > 0");
    }
}

```

PART 6 — Deployment Process (step-by-step + terminal screenshots)

The project is deployed using Hardhat scripts on an Ethereum test network. The deployment process includes compilation, deployment of both contracts, and assigning the StartupBooster contract as an authorized minter for the SupportToken contract.

Steps:

1. Install dependencies:
 - o `npm install`
2. Compile contracts:
 - o `npx hardhat compile`
3. Deploy contracts using Hardhat:
 - o `npx hardhat run scripts/deploy.js --network <network>`
4. After deployment, call `SupportToken setMinter(boosterAddress, true)` so the booster contract can mint reward tokens.

```
C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster\frontend\index.html
    th --show-stack-traces
    n Final\student-startup-booster> npx hardhat compile
    Compiled 2 Solidity files successfully (evm target: paris).
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> git status
  On branch main
  Your branch is up to date with 'origin/main'.

  Changes not staged for commit:
    (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   contracts/SupportToken.sol

  no changes added to commit (use "git add" and/or "git commit -a")
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> git branch
  feature/token-dias
  feature/token-minter
* main
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> npx hardhat compile
  Nothing to compile
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> git add contracts/SupportToken.sol
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> git commit -m "fix: SupportToken mint uses _mint"
[main 9dfe33c] fix: supportToken mint uses _mint
  1 file changed, 1 insertion(+), 7 deletions(-)

remote: Resolving deltas: 100% (3/3), completed with 3 local objects. ...

Nothing to compile
PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
Accounts
=====
WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf9Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cb5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de411afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0xd15d34AAf54267DB7D7c367839AAf71A00a2C6A65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a

Account #5: 0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc (10000 ETH)

```

```
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> npx hardhat compile
  Nothing to compile
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster> npx hardhat run .\scripts\deploy.js --network localhost
  Deploying contracts with account: 0xf9Fd6e51aad88F6F4ce6aB8827279cffFb92266
  Account balance: 1000000000000000000000000
  SupportToken deployed to: 0x5FbDB2315678afeccb367f032d93F642f64180aa3
  StartupBooster deployed to: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
  StartupBooster set as minter
● PS C:\Users\Admin\OneDrive\Documents\Dias Уроки 5 тремак\Blockchain Final\student-startup-booster>
```

Deployment was performed using Hardhat on the localhost network.

First, the contracts were compiled with npx hardhat compile.

Then the deployment script was executed using npx hardhat run scripts/deploy.js --network localhost.

During deployment, the SupportToken contract and StartupBooster contract were deployed and their addresses were printed in the terminal. Finally, the StartupBooster contract was assigned as a minter for SupportToken to enable minting reward tokens after contributions.

PART 7 — Frontend & MetaMask Integration:

MetaMask integration is required because all blockchain operations must be signed by the user. The frontend requests wallet access, checks the active network, and then uses ethers.js to call smart contract methods and read balances.

Implemented functionality:

- Connect MetaMask button
- Display connected wallet address
- Check selected network is a testnet (Sepolia/Holesky/local)
- Create campaign via contract function
- Contribute via MetaMask transaction
- Display user test ETH balance and reward token (SUP) balance

The image shows two screenshots of a web application and a MetaMask wallet interface.

Top Screenshot: A screenshot of a web browser showing a campaign creation form for "Student Startup Booster". The title is "Create Campaign". The fields are as follows:

- Title:** test
- Goal (ETH):** 0.5
- Duration (seconds):** 5

A large blue "Create Campaign" button is at the bottom.

Bottom Screenshot: A screenshot of a MetaMask wallet interface. It shows the following information:

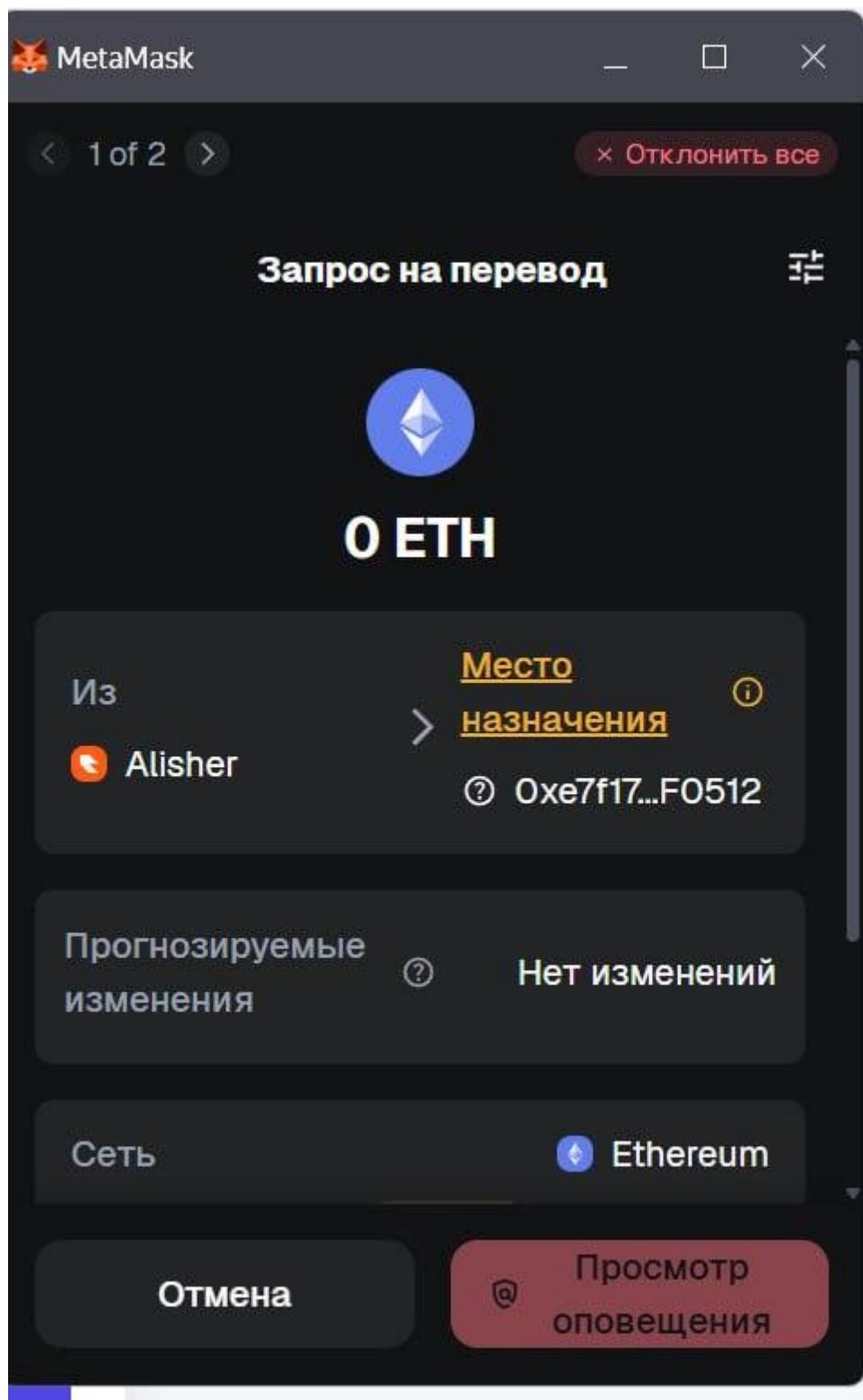
- User: Alisher
- Network: Ethereum
- Address: 127.0.0.1:5500
- Balances: 7,19 \$ (ETH) + 0,25 \$ (+)
- Buttons: Купить, Обмен..., Отправ..., Получи...
- Section: Управление разрешениями
- Card: Say hello to Bitcoin - Trade, manage, and buy BTC directly on MetaMask
- Tab: Токены (selected), ДеFi, NFT, Деятельность
- Table: Ethereum - Зарабатывайте 0,10 \$ +3.64% 0,0000400 ETH
- Table: Ethereum 0,00 \$ 0 ETH +3.64%

The image shows a screenshot of a web browser displaying the same campaign creation form as the top screenshot.

The title is "Create Campaign". The fields are as follows:

- Title:** My startup idea
- Goal (ETH):** 1
- Duration (seconds):** 600

A large blue "Create Campaign" button is at the bottom.



PART 8 — Functional Demonstration:

8.1 Create Campaign

We tested the DApp by creating a new crowdfunding campaign from the frontend.

The campaign was created by entering a title, funding goal (in ETH), and a deadline.

After submitting the form, the campaign appeared in the campaigns list with its parameters (title/goal/deadline) and a unique campaign ID.



Student Startup Booster

[Connect MetaMask](#)

Campaign created ✓
0x0917bbe6c03e9a1282cdeb1cd44d43e1bbb32fe7
ETH: 0.000046948904116692

Create Campaign

Title

Goal (ETH)

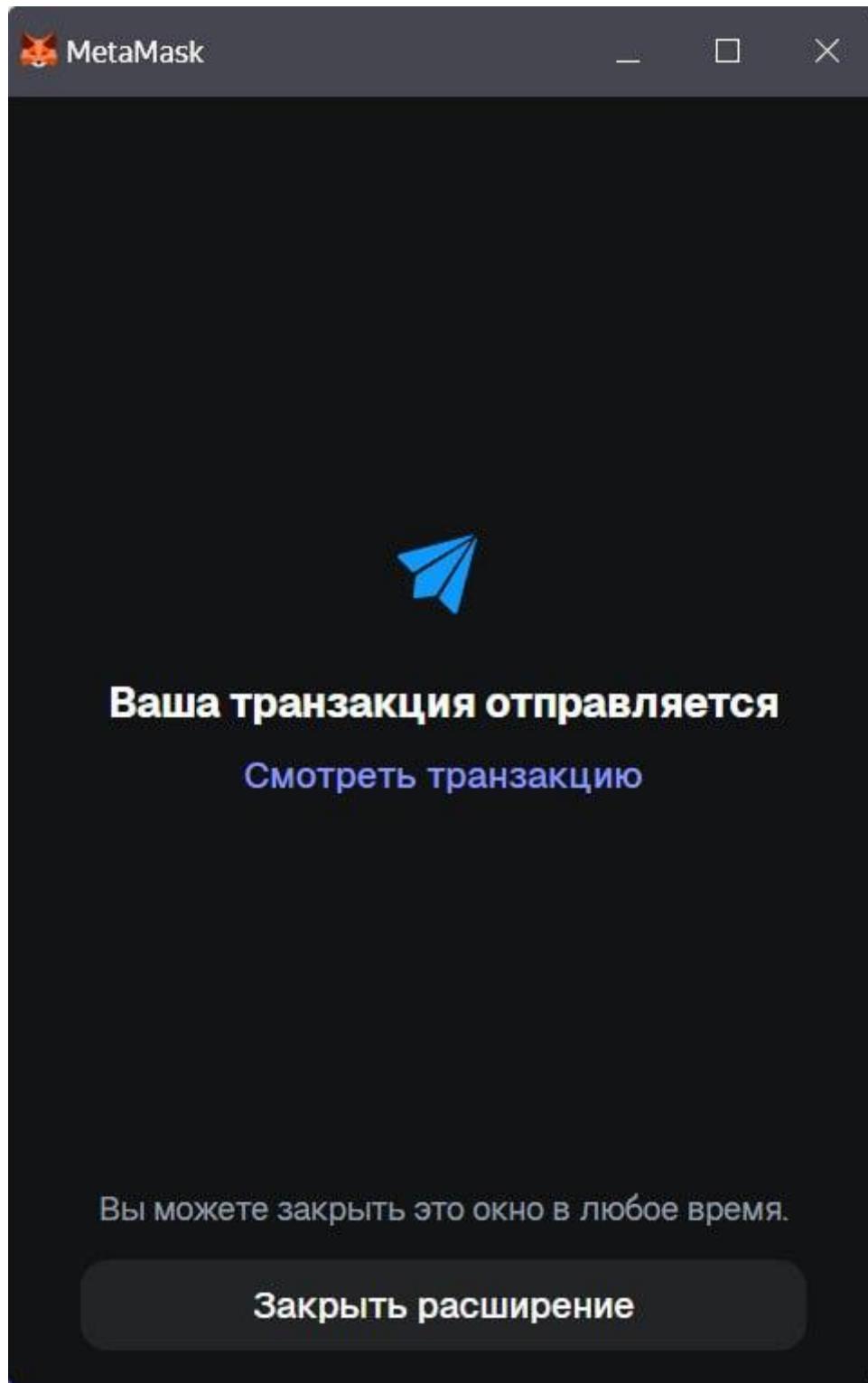
Duration (seconds)

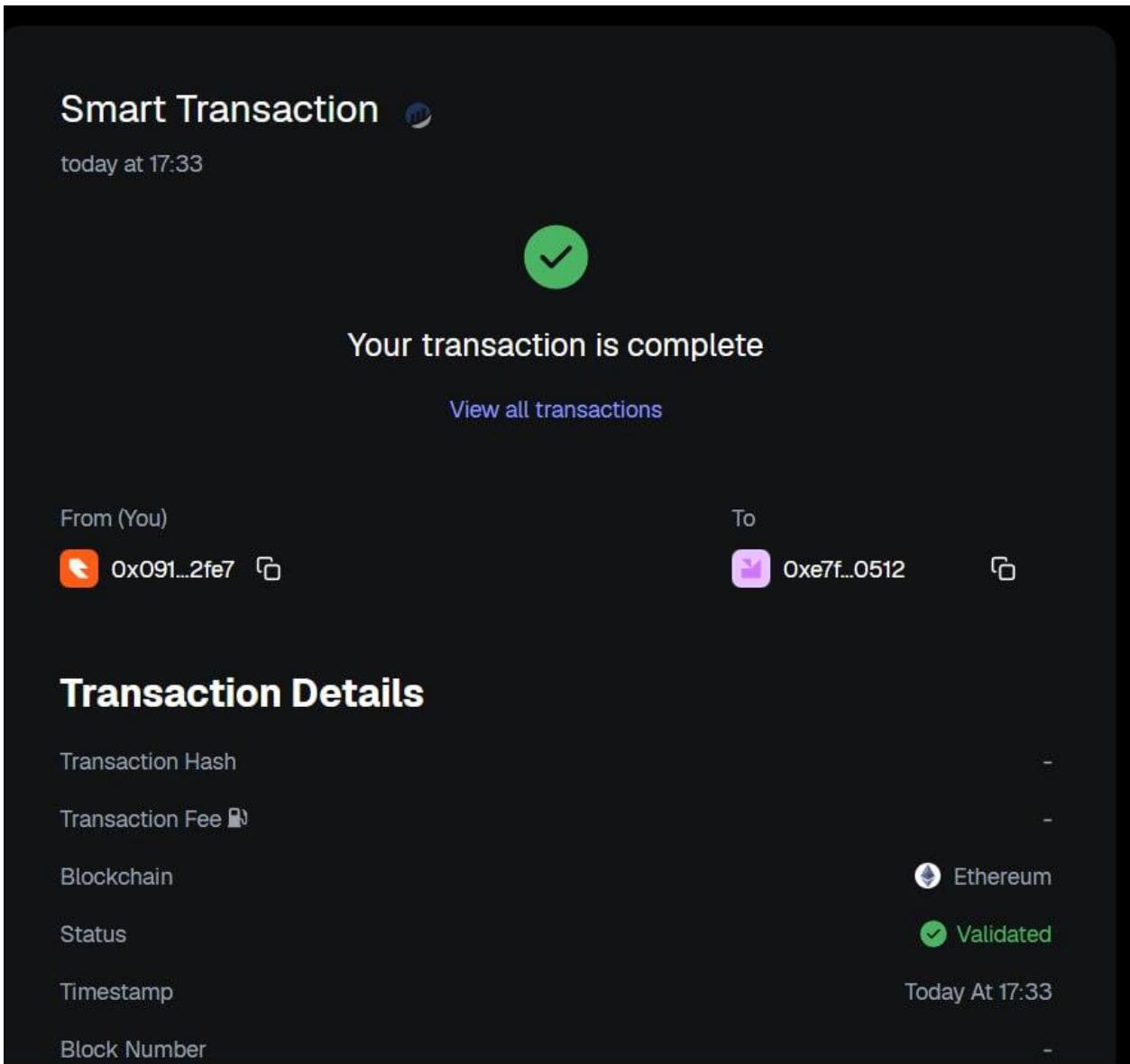
Create Campaign

8.2 Contribute

After creating a campaign, we contributed test ETH from a second user account (or the same account) through MetaMask.

The frontend triggered the contribute transaction, MetaMask requested confirmation, and after signing, the contribution was recorded on-chain. The campaign's raised amount increased accordingly.





The image shows a mobile application interface for a "Smart Transaction". At the top, it says "Smart Transaction" with a blue circular icon. Below that, it says "today at 17:33". In the center, there is a green circle with a white checkmark. Below the checkmark, the text "Your transaction is complete" is displayed. Underneath that, there is a blue link "View all transactions".

Below this, there are two sections: "From (You)" and "To". Each section has an orange square icon with a white symbol (a hand or a person), followed by a hex address (e.g., Oxo91...2fe7 or Ox...0512) and a copy icon.

At the bottom, there is a section titled "Transaction Details" with the following items:

- Transaction Hash: -
- Transaction Fee: -
- Blockchain: Ethereum
- Status: Validated
- Timestamp: Today At 17:33
- Block Number: -

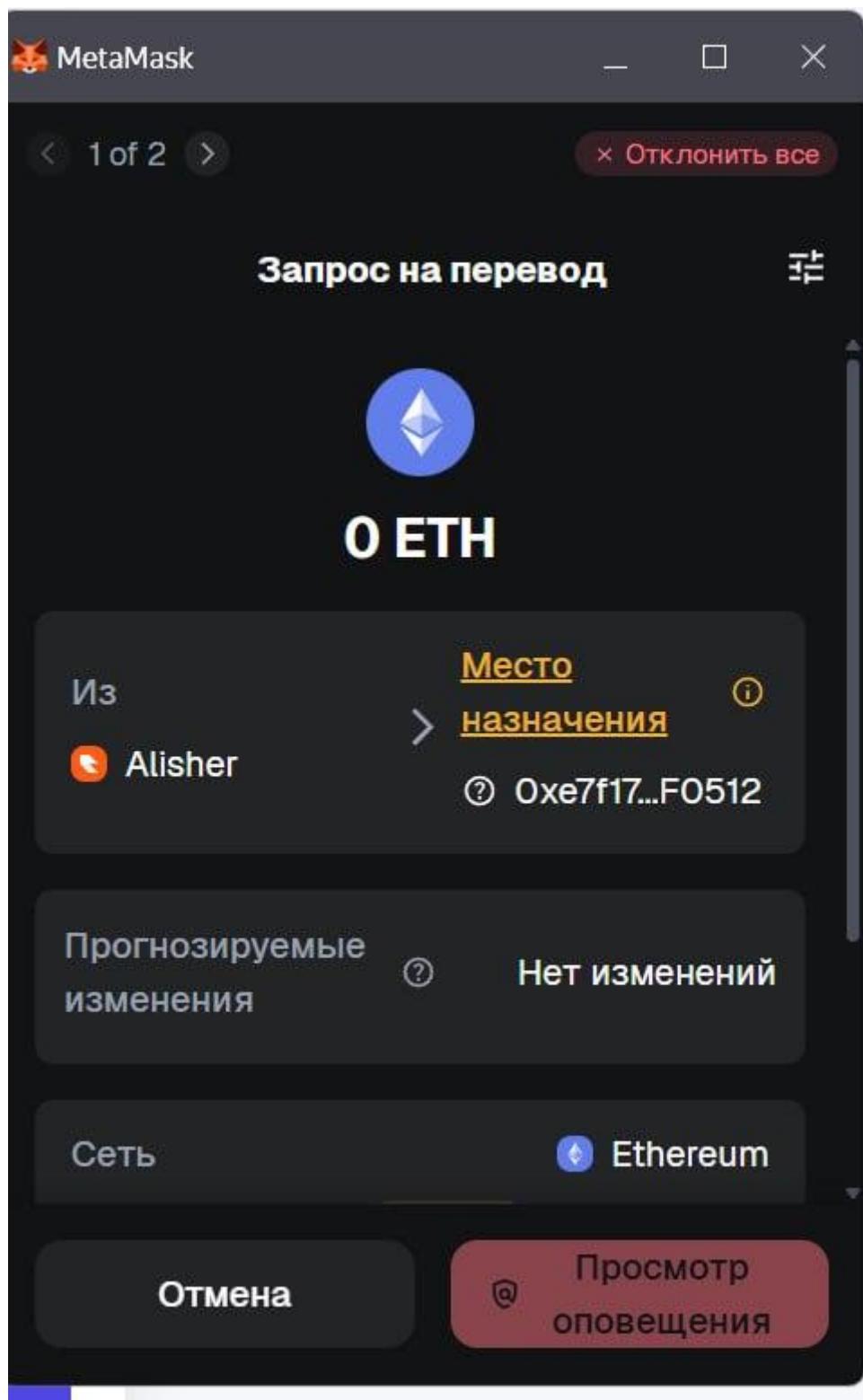
8.3 Finalize Campaign

After the deadline passed, we finalized the campaign using the “Finalize” function from the frontend.

Finalize completes the campaign logic and triggers reward distribution (minting SUP tokens) to contributors according to the contribution amount.

8.4 Verify SUP Rewards

After contributing ETH, the contributor received reward tokens (SUP). We verified this by checking the SUP token balance in the frontend (or in MetaMask after importing the token contract address). The SUP balance increased after the donation, confirming that the minting mechanism works.



We deployed and tested on Sepolia testnet using free faucet ETH.
Then include transaction hash (txid) for donation.

PART 9 — Team Contribution & Version Control:

This project was developed by a team of three members, using GitHub for collaboration through branches, commits, and pull requests:

- **Olzhas** — smart contracts & deployment scripts, merging PRs
- **Dias** — ERC-20 SupportToken development (minter role + mint function)
- **Alisher** — frontend interface and MetaMask connection

GitHub was used to ensure version control and teamwork:

- Each feature was developed on a separate branch
- Changes were reviewed and merged through Pull Requests
- Commit history shows contributions from each member

The screenshot shows the GitHub repository interface for 'student-startup-booster'. The 'Code' tab is selected. The commit history for the 'main' branch is displayed, showing the following commits:

- fix: SupportToken mint uses _mint (Dias074 committed 1 hour ago) - Commit ID: 9dfe33c
- Merge pull request #3 from Olzhasuh/feature/token-minter (Olzhasuh authored 2 hours ago) - Verified - Commit ID: f3d4832
- Merge branch 'main' into feature/token-minter (Olzhasuh authored 2 hours ago) - Verified - Commit ID: f1999ca
- Dias: add mint to SupportToken (Dias074 committed 4 hours ago) - Commit ID: f7c4717
- feat: frontend MetaMask connect + UI + favicon (4lishe3r committed 4 hours ago) - Commit ID: 4ecf602
- Olzhas: update gitignore (Olzhasuh committed 6 hours ago) - Commit ID: 6262712

At the bottom of the commit list, there is a note: "Commits on Feb 7, 2026" followed by "Merge pull request #1 from Olzhasuh/feature/frontend_alisher".

Commits on Feb 7, 2026					
Merge pull request #1 from Olzhasuh/feature/frontend-alisher	bee6c6f				
Olzhasuh authored 18 hours ago					
Merge pull request #2 from Olzhasuh/feature/token-dias	13bea19				
Olzhasuh authored 18 hours ago					
Olzas: add basic campaign creation logic	89cf387				
Olzhasuh committed 18 hours ago					
Dias: add ERC-20 SupportToken skeleton	57ce9dd				
Dias074 committed yesterday					
Alisher: init frontend and MetaMask connect	9cd2a4				
4liske3r committed yesterday					
Olzas: add StartupBooster contract skeleton	ac94e3f				
Olzhasuh committed yesterday					
Olzas: remove default Hardhat Lock example	9dbf670				
Olzhasuh committed yesterday					
restore: hardhat project files	0755c89				
Olzhasuh committed yesterday					
feat: add hardhat project files	f14920f				
Olzhasuh committed yesterday					
Initial commit					
Olzas: add basic campaign creation logic	89cf387				
Olzhasuh committed 18 hours ago					
Dias: add ERC-20 SupportToken skeleton	57ce9dd				
Dias074 committed yesterday					
Alisher: init frontend and MetaMask connect	9cd2a4				
4liske3r committed yesterday					
Olzas: add StartupBooster contract skeleton	ac94e3f				
Olzhasuh committed yesterday					
Olzas: remove default Hardhat Lock example	9dbf670				
Olzhasuh committed yesterday					
restore: hardhat project files	0755c89				
Olzhasuh committed yesterday					
feat: add hardhat project files	f14920f				
Olzhasuh committed yesterday					
Initial commit					
Olzhasuh authored yesterday	74843ac				

Olzhasuh / student-startup-booster

Type to search

Branches

Overview	Yours	Active	Stale	All
Search branches...				

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	1 hour ago		Default		

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature/token-minter	2 hours ago		2 0	#3	
feature/token-dias	4 hours ago		10 1		

Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature/token-minter	2 hours ago		2 0	#3	
feature/token-dias	4 hours ago		10 1		

The screenshot shows a GitHub repository interface. At the top, there's a header with the repository name and a 'Code' button. Below the header, there are three sections: 'Default', 'Your branches', and 'Active branches', each containing a table of branch information.

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	1 hour ago	Default	0	0	...

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature/token-minter	2 hours ago	2 0	3	0	...
feature/token-dias	4 hours ago	10 1	0	1	...

Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature/token-minter	2 hours ago	2 0	3	0	...
feature/token-dias	4 hours ago	10 1	0	1	...
feature/frontend-alisher	yesterday	10 0	1	0	...

At the bottom of the page, there are links for GitHub, Terms, Privacy, Security, Status, Community, Docs, Contact, Manage cookies, and a link to 'Do not share my personal information'.

PART 10 — Conclusion & Future Work:

The Student Startup Booster project successfully meets the main course requirements: it includes Solidity smart contracts, ERC-20 reward token minting, MetaMask integration, and operates on an Ethereum test network using free test ETH only.

FINAL EXAMINATION PROJECT

The platform demonstrates core DApp principles: decentralized logic, transparent fundraising, and tokenization rewards for participation.

Possible future improvements:

- Refund functionality for unsuccessful campaigns
- Better UI/UX (validation, notifications, loading states)
- Storing campaign metadata (images/descriptions) in IPFS
- DAO-style voting for campaign approval
- More detailed analytics of contributions and campaign status