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

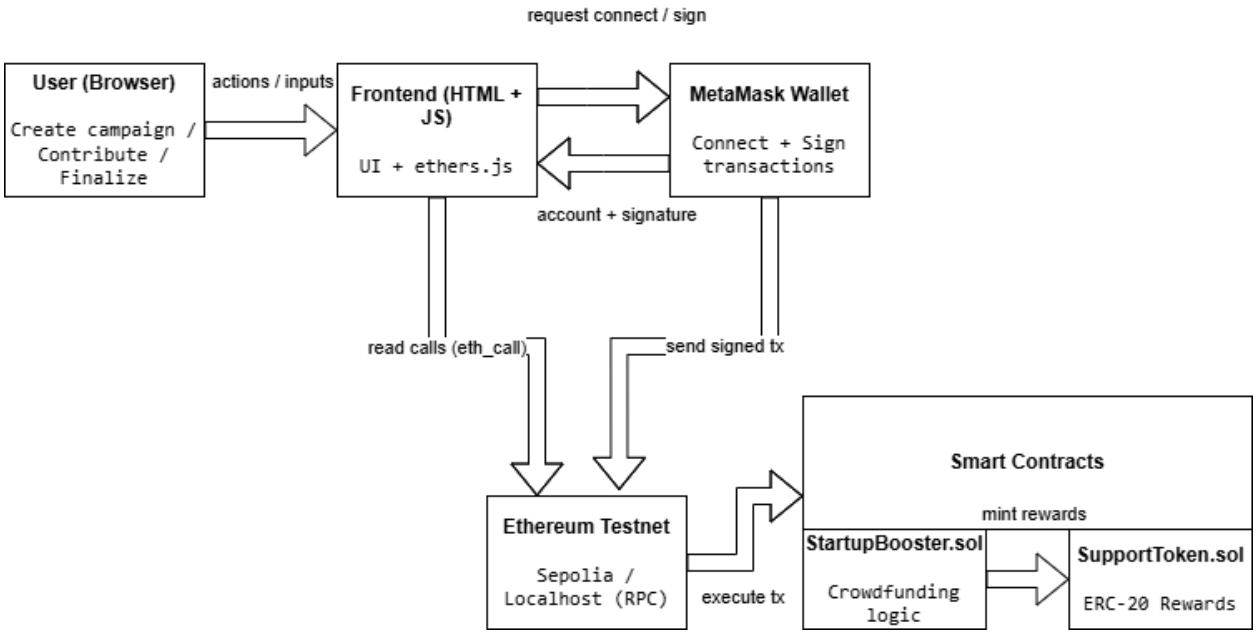
PART 4 — System Architectur:

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:

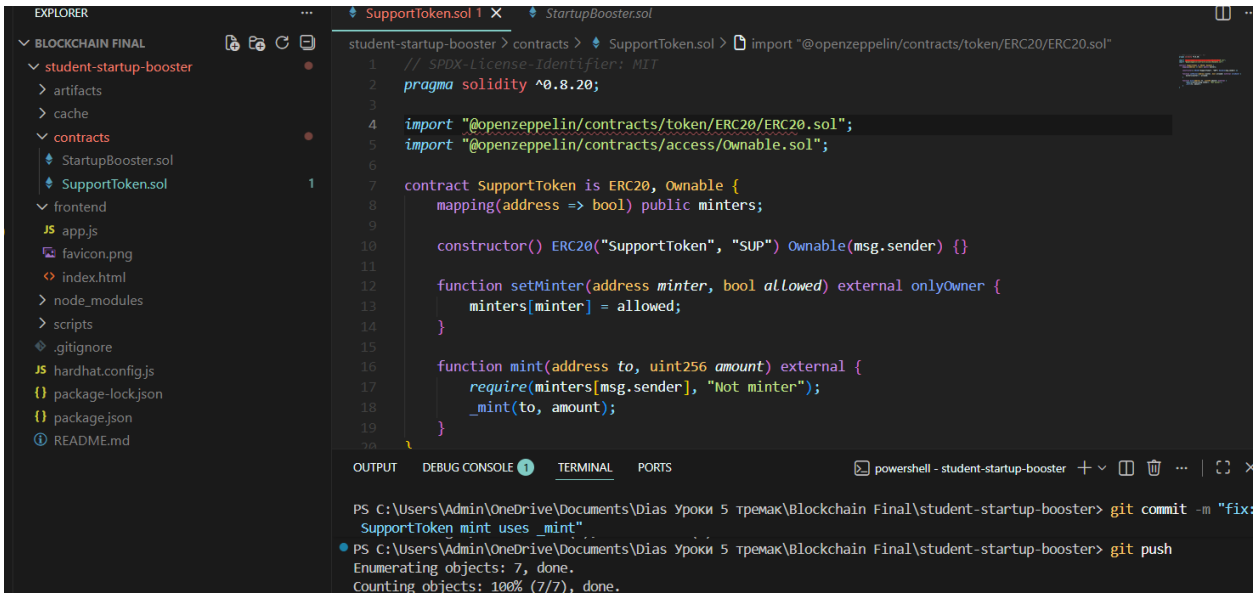
5.1 SupportToken.sol

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

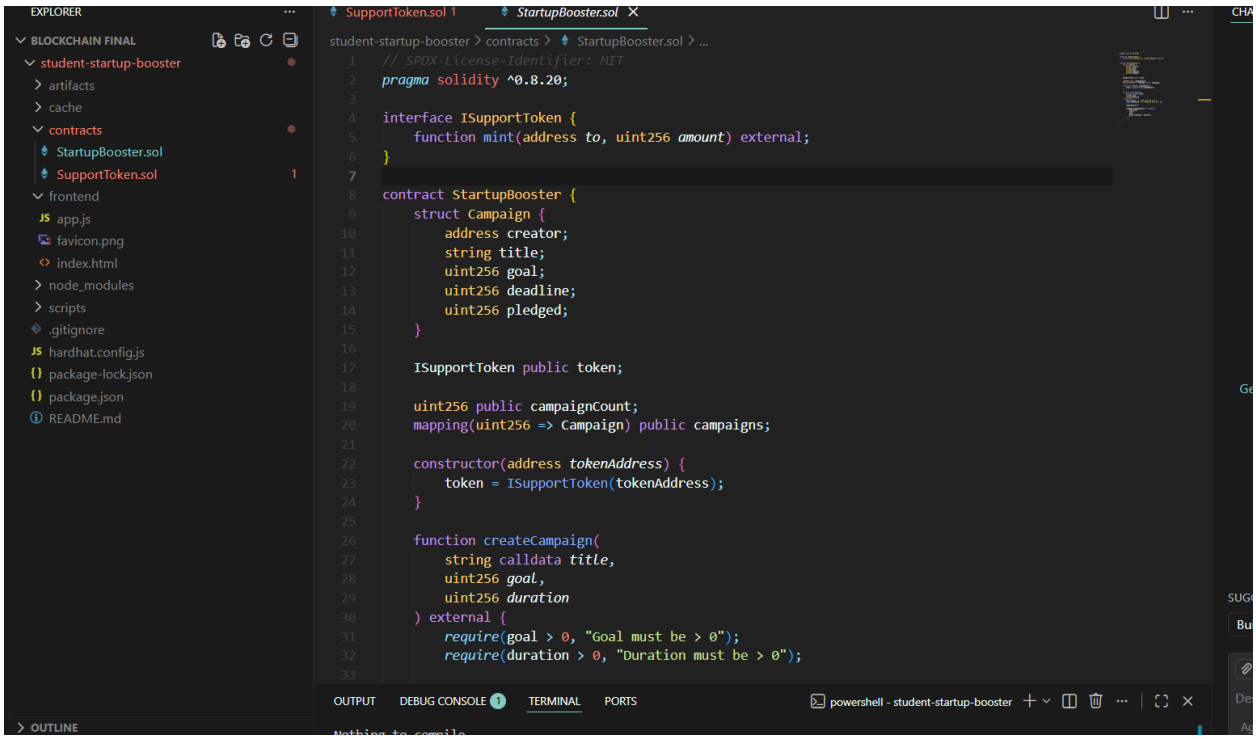


5.2 StartupBooster.sol

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)

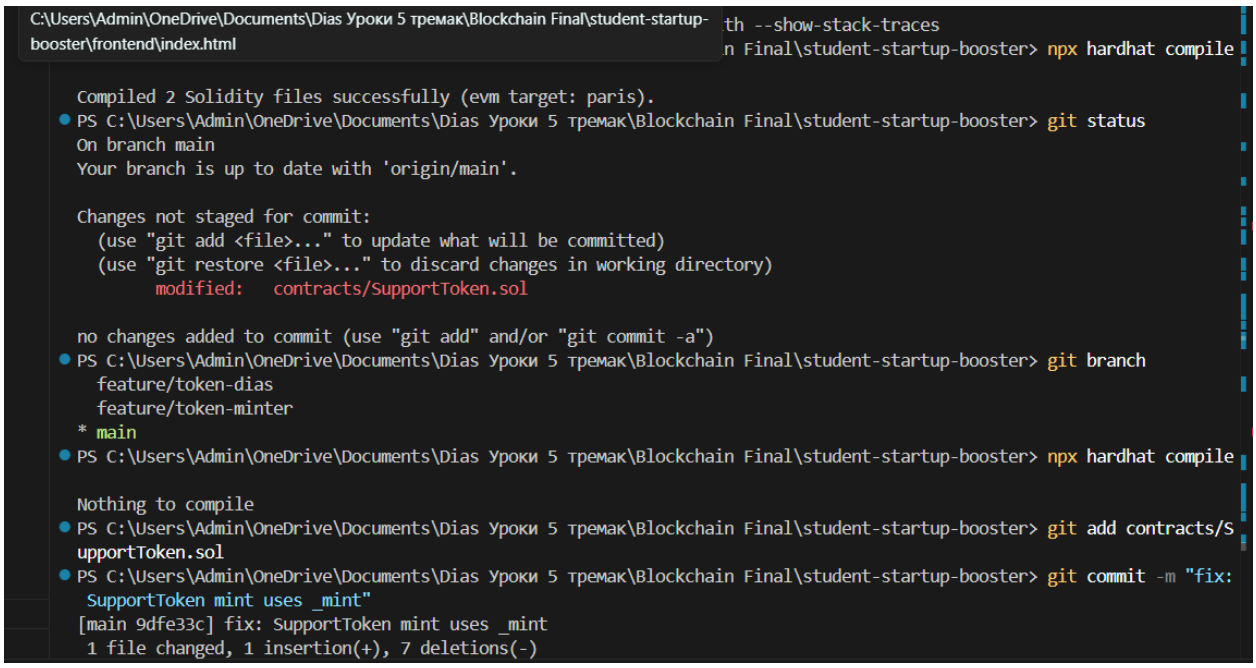


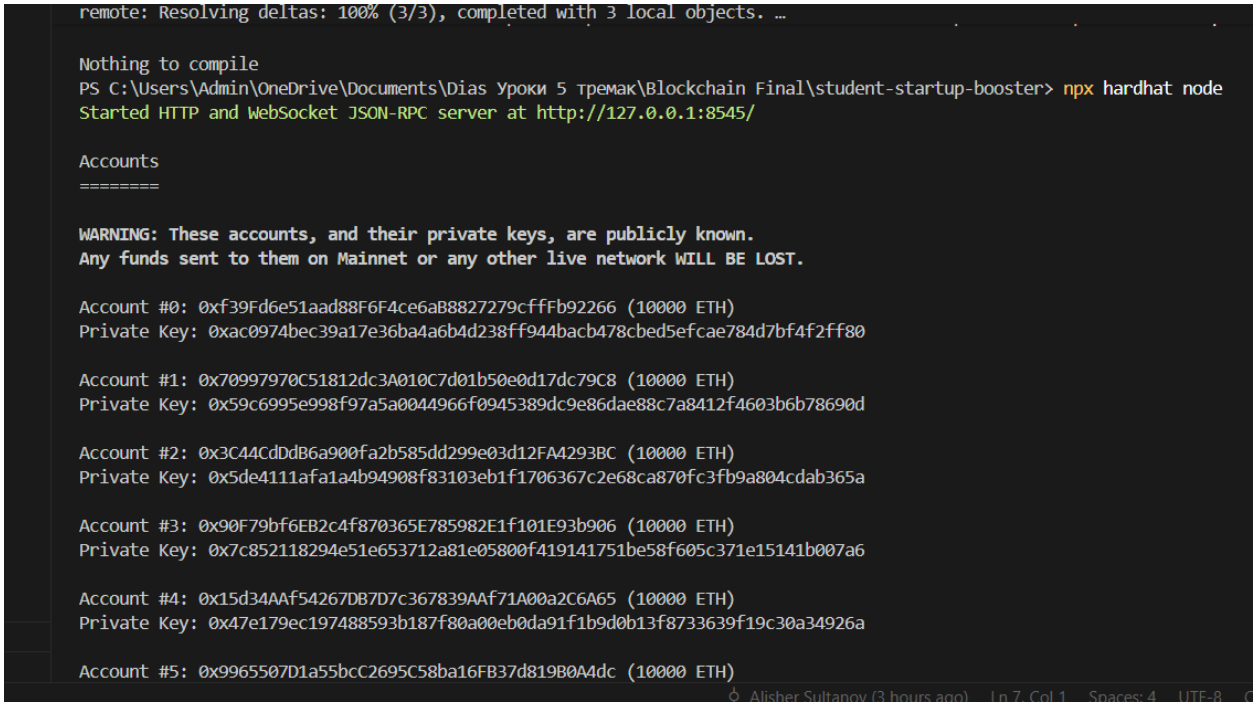
PART 6 — Deployment Process (step-by-step + terminal screenshots)

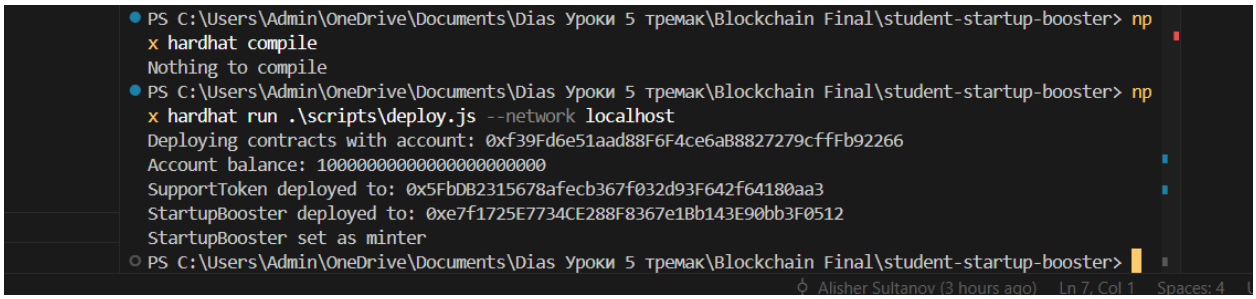
The project is deployed using Hardhat scripts on an Ethereum test network. The deployment process includes compilation, deployment of both contracts, and assigning the StartupBooster contract as an authorized minter for the SupportToken contract.

Steps:

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







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.

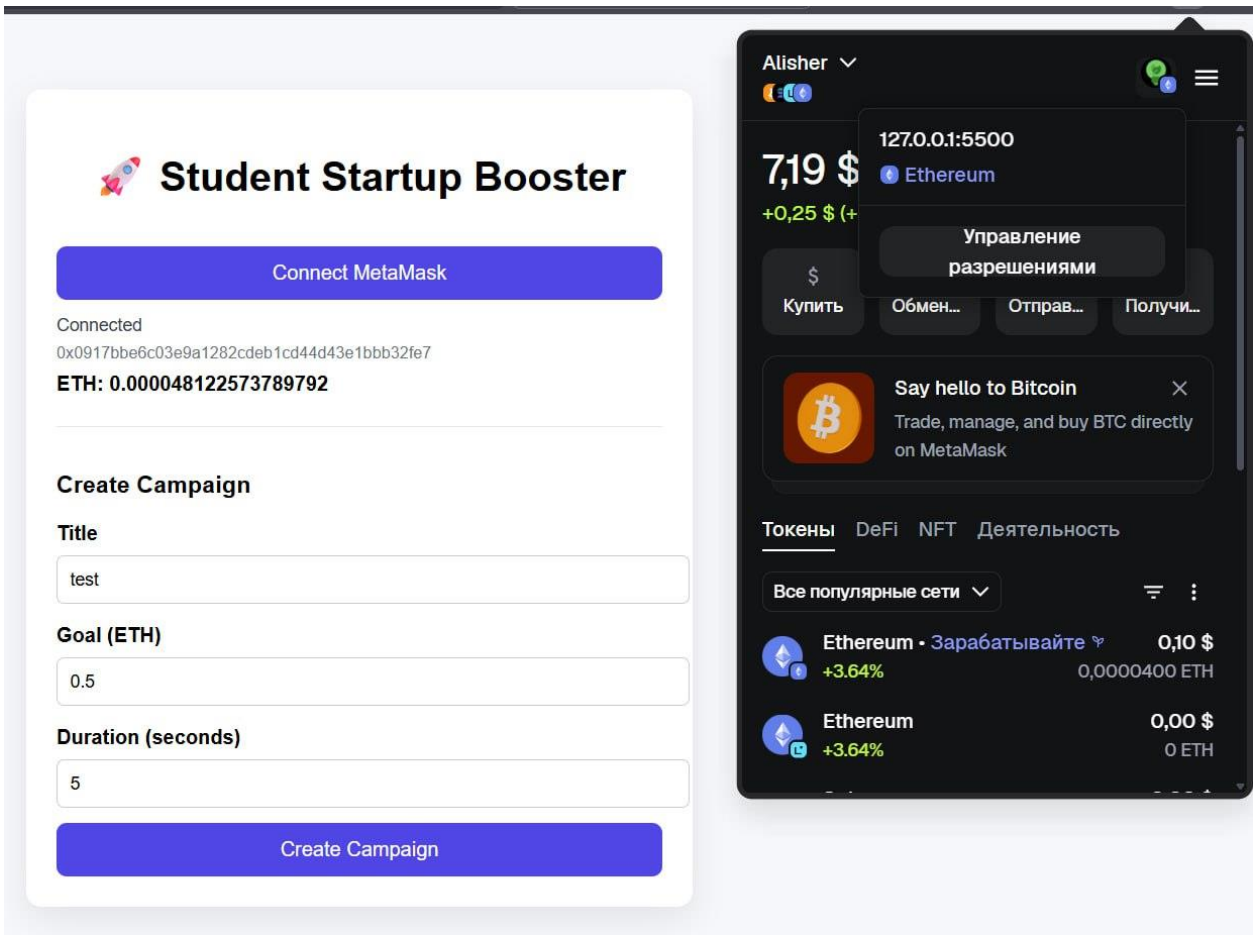
During deployment, the SupportToken contract and StartupBooster contract were deployed and their addresses were printed in the terminal.
Finally, the StartupBooster contract was assigned as a minter for SupportToken to enable minting reward tokens after contributions.

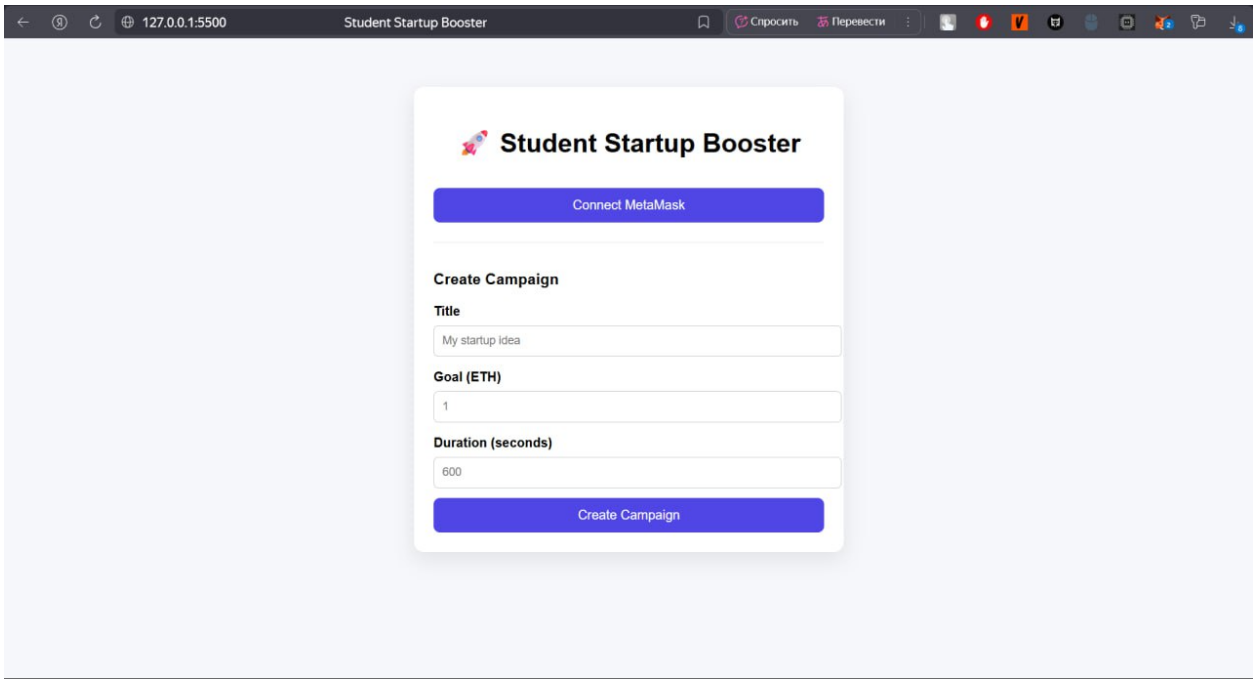
PART 7 — Frontend & MetaMask Integration:

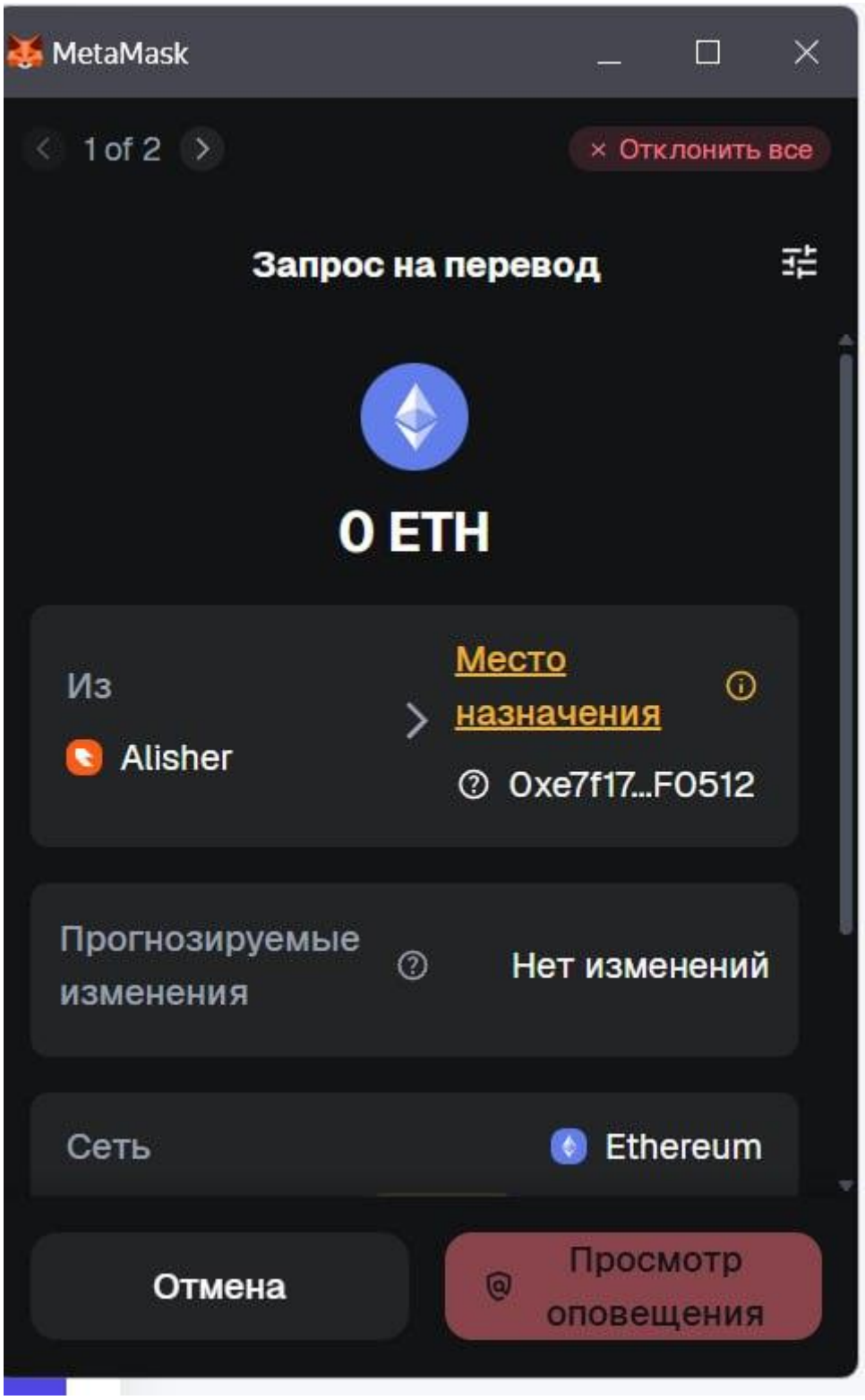
MetaMask integration is required because all blockchain operations must be signed by the user. The frontend requests wallet access, checks the active network, and then uses ethers.js to call smart contract methods and read balances.

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


Student Startup Booster
Connect MetaMask
Connected
0x0917bbe6c03e9a1282cdeb1cd44d43e1bbb32fe7
ETH: 0.000048122573789792
Create Campaign
Title
test
Goal (ETH)
0.5
Duration (seconds)
5
Create Campaign
Alisher
7,19 $
+0,25 $ (+
127.0.0.1:5500
Ethereum
Управление разрешениями
Купить
Обмен...
Отправ...
Получи...
Say hello to Bitcoin
Trade, manage, and buy BTC directly on MetaMask
Токены
DeFi
NFT
Деятельность
Все популярные сети
Ethereum • Зарабатывайте
+3.64%
0,10 $
0,0000400 ETH
Ethereum
+3.64%
0,00 $
0 ETH

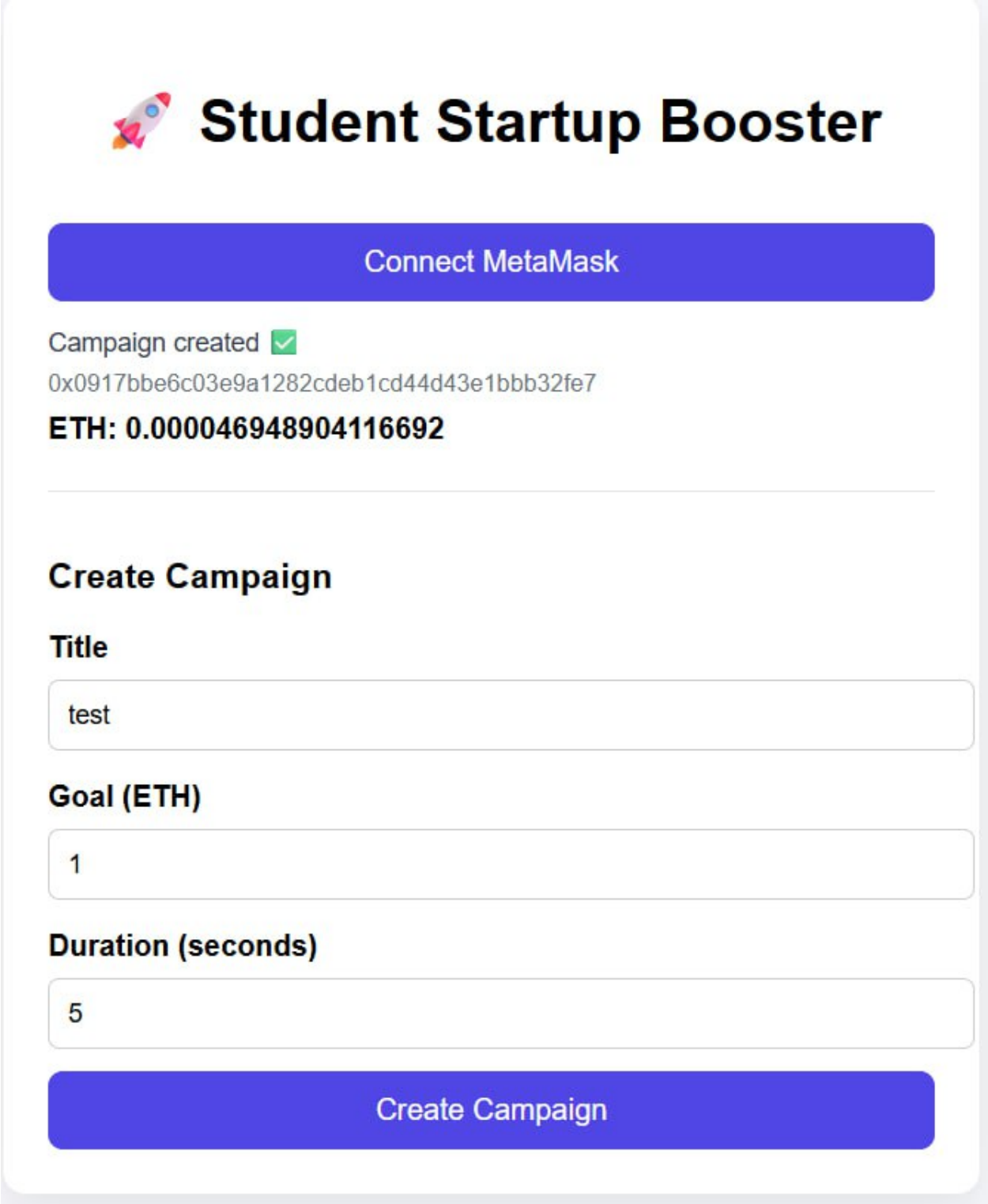

127.0.0.1:5500
Student Startup Booster
Спросить
Перевести
Student Startup Booster
Connect MetaMask
Create Campaign
Title
My startup idea
Goal (ETH)
1
Duration (seconds)
600
Create Campaign



PART 8 — Functional Demonstration:

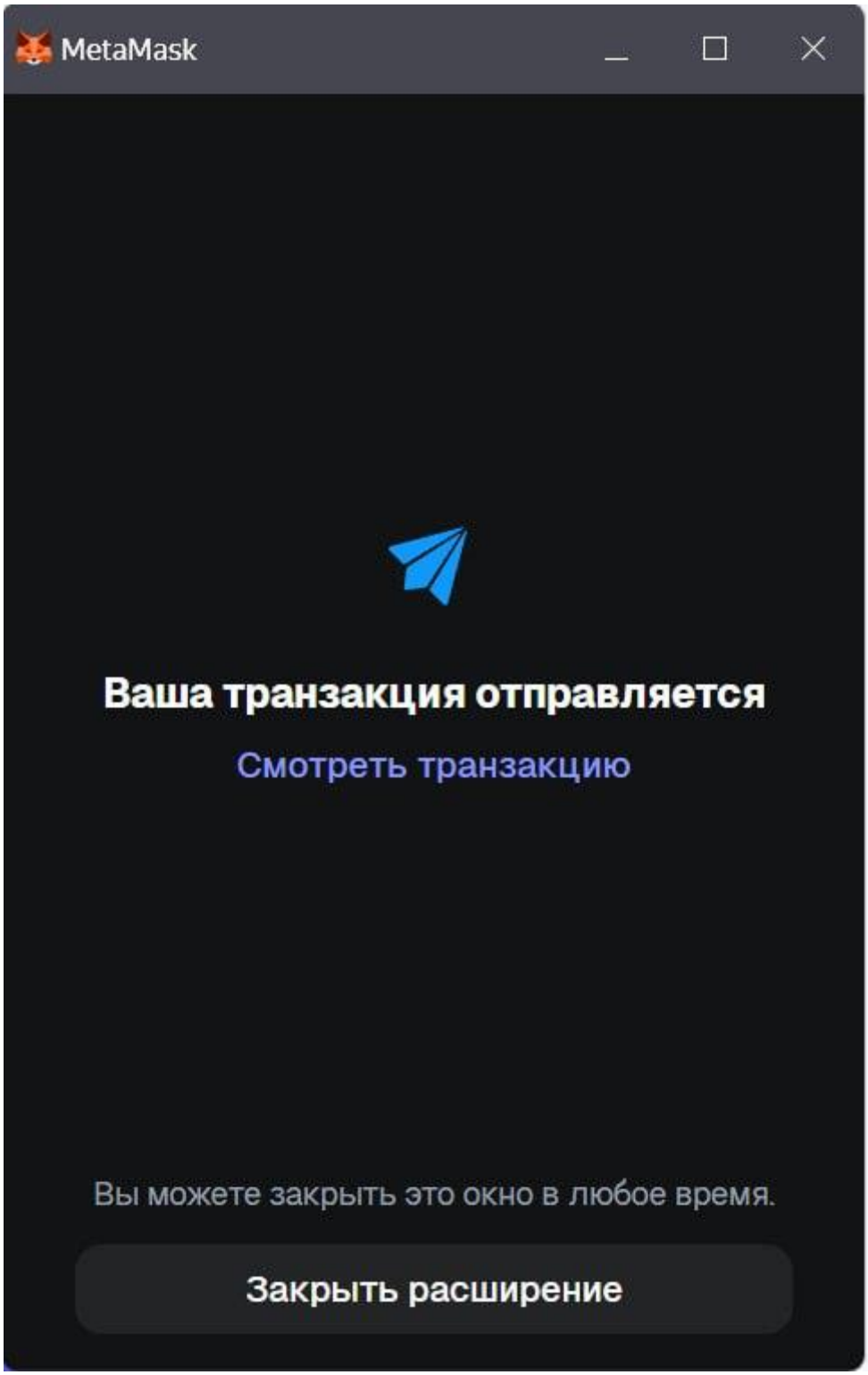
8.1 Create Campaign

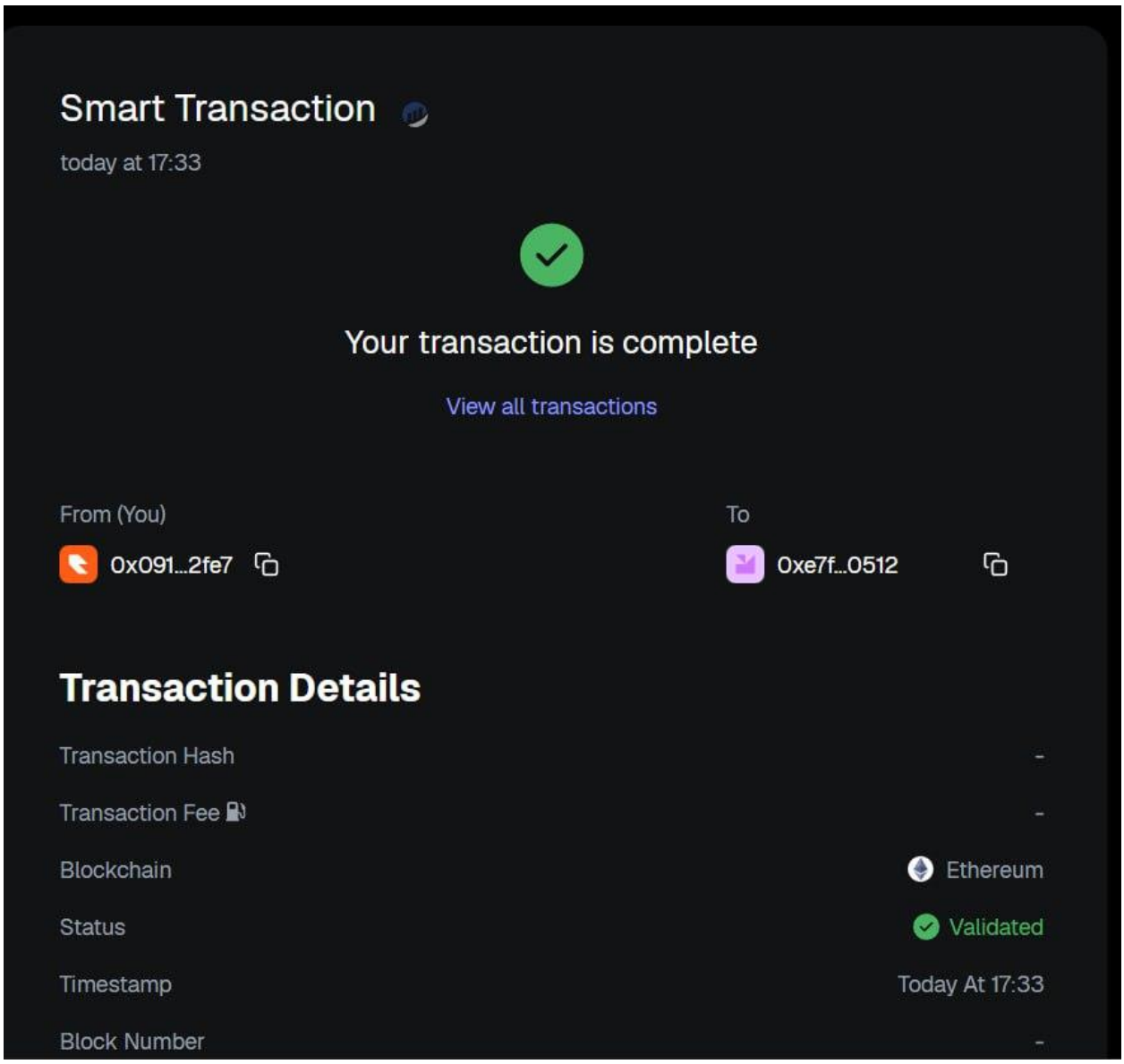
We tested the DApp by creating a new crowdfunding campaign from the frontend.
The campaign was created by entering a title, funding goal (in ETH), and a deadline.
After submitting the form, the campaign appeared in the campaigns list with its parameters (title/goal/deadline) and a unique campaign ID.



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.



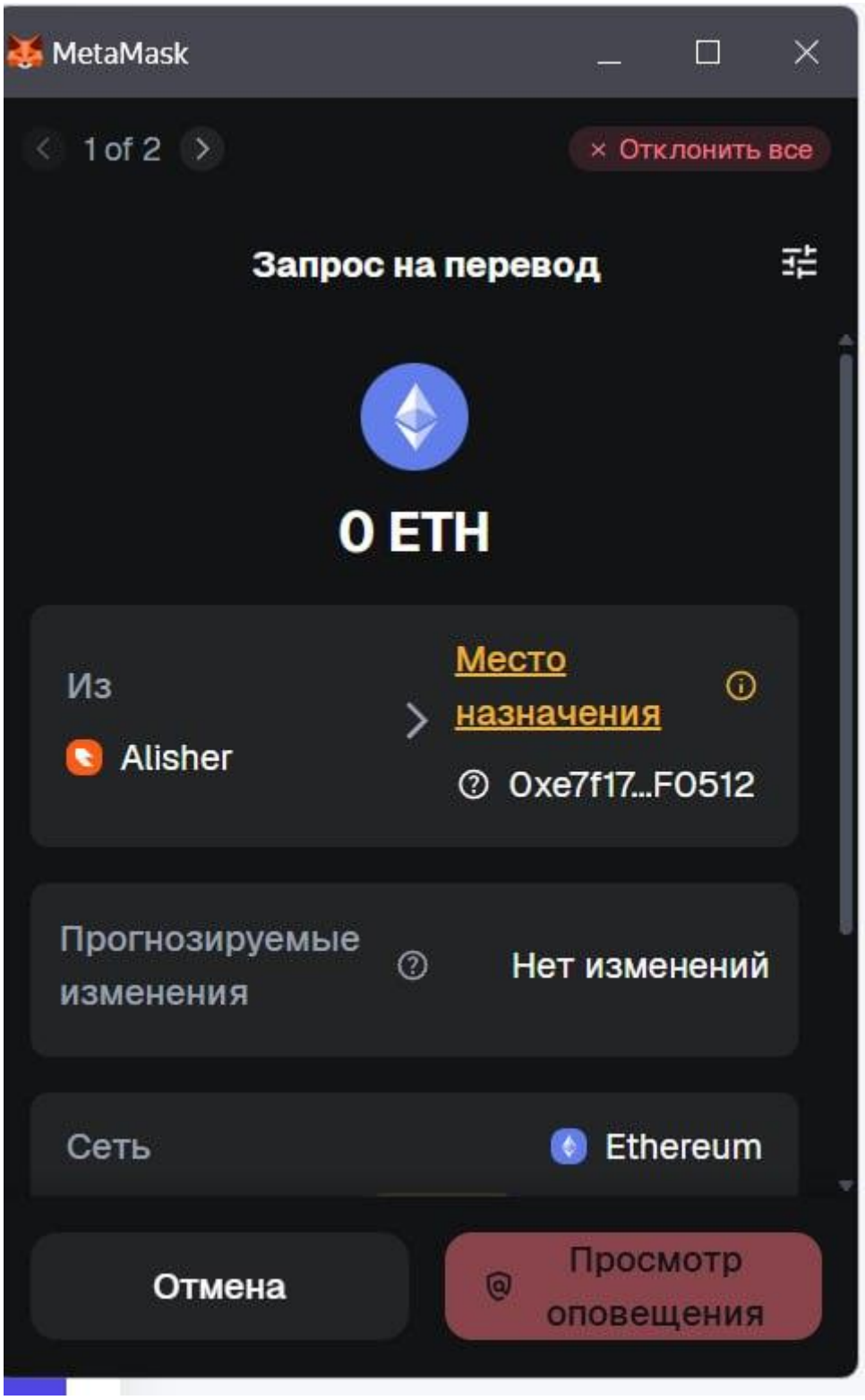


8.3 Finalize Campaign

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.

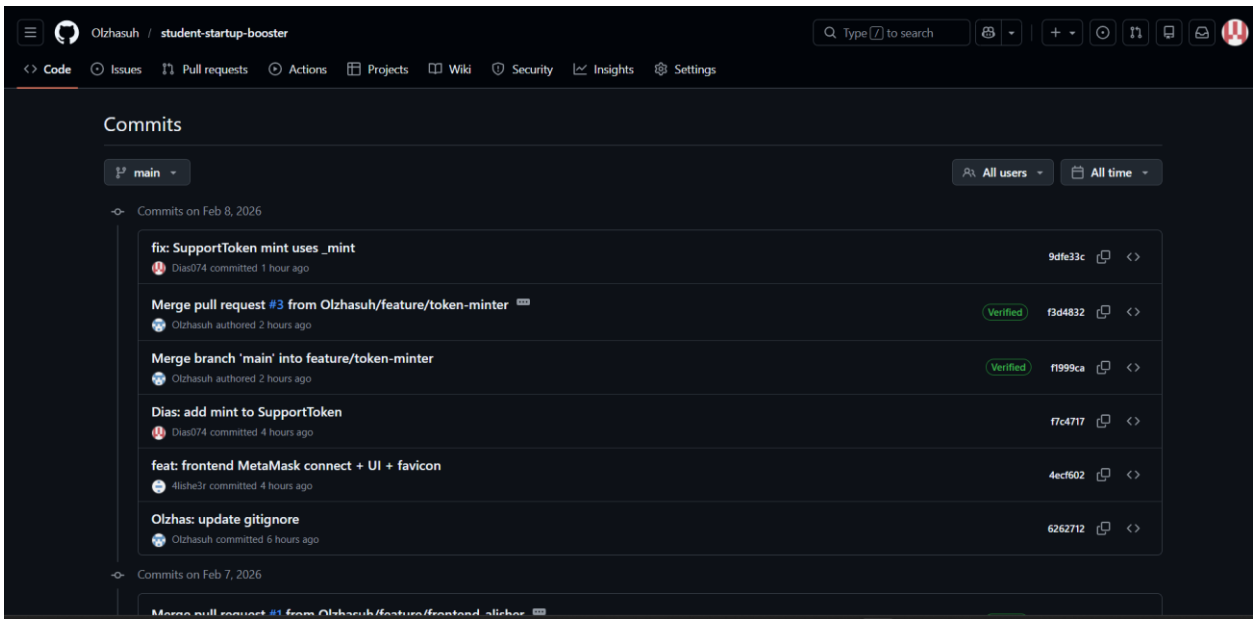
PART 9 — Team Contribution & Version Control:

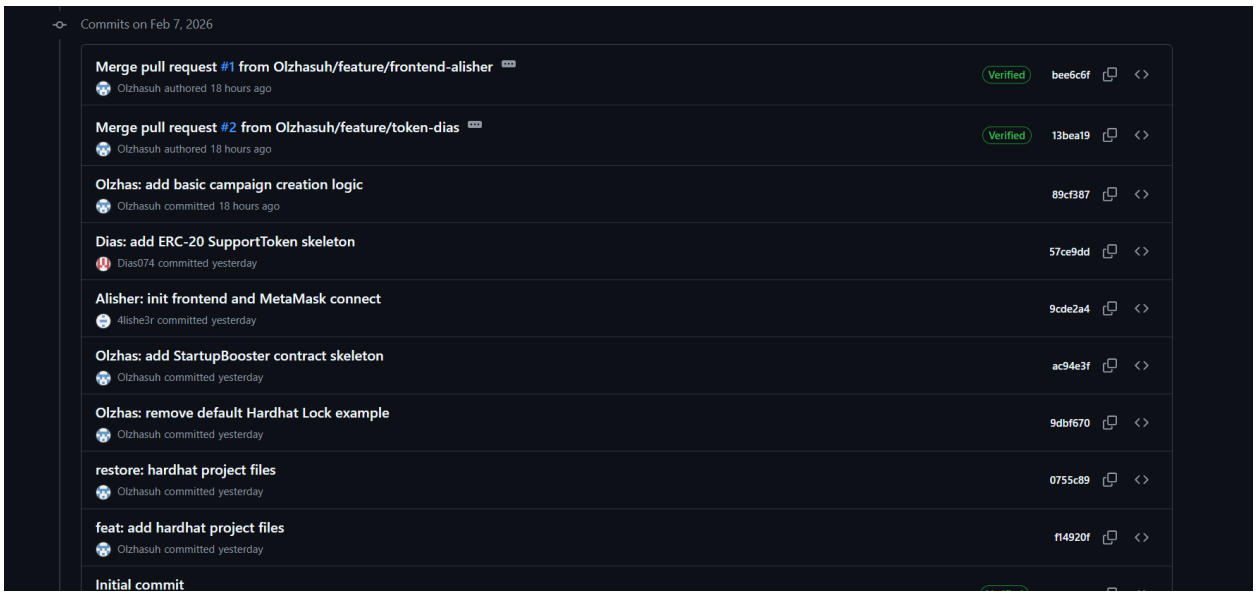
This project was developed by a team of three members, using GitHub for collaboration through branches, commits, and pull requests:

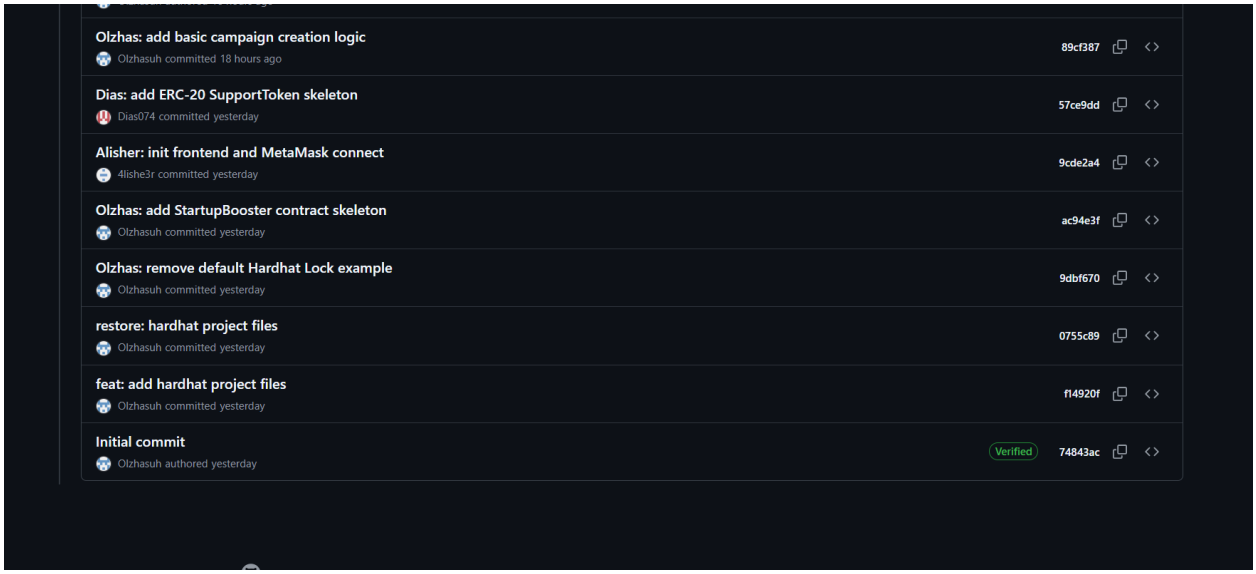
- **Olzhas** — smart contracts & deployment scripts, merging PRs
- **Dias** — ERC-20 SupportToken development (minter role + mint function)
- **Alisher** — frontend interface and MetaMask connection

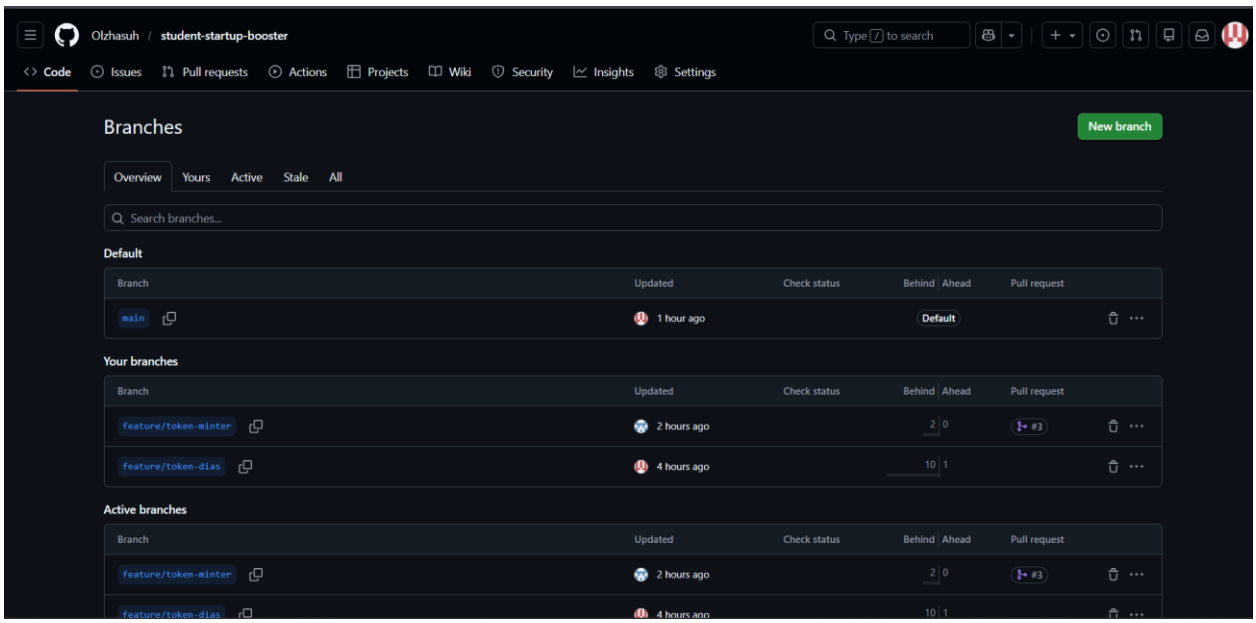
GitHub was used to ensure version control and teamwork:

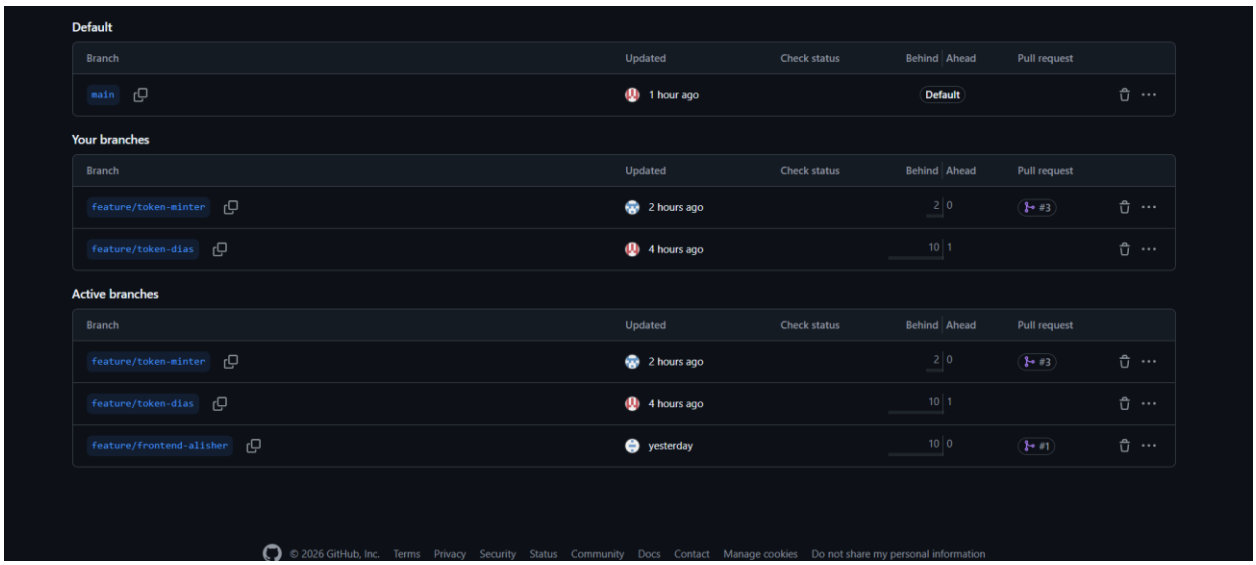
- Each feature was developed on a separate branch
- Changes were reviewed and merged through Pull Requests
- Commit history shows contributions from each member




Commits on Feb 7, 2026
Merge pull request #1 from Olzhasuh/feature/frontend-alisher
Olzhasuh authored 18 hours ago
Verified
bee6c6f
Merge pull request #2 from Olzhasuh/feature/token-dias
Olzhasuh authored 18 hours ago
Verified
13bea19
Olzhas: add basic campaign creation logic
Olzhasuh committed 18 hours ago
89cf387
Dias: add ERC-20 SupportToken skeleton
Dias074 committed yesterday
57ce9dd
Alisher: init frontend and MetaMask connect
4lishe3r committed yesterday
9cde2a4
Olzhas: add StartupBooster contract skeleton
Olzhasuh committed yesterday
ac94e3f
Olzhas: remove default Hardhat Lock example
Olzhasuh committed yesterday
9dbf670
restore: hardhat project files
Olzhasuh committed yesterday
0755c89
feat: add hardhat project files
Olzhasuh committed yesterday
f14920f
Initial commit


Olzhas: add basic campaign creation logic
Olzhasuh committed 18 hours ago
89cf387
Dias: add ERC-20 SupportToken skeleton
Dias074 committed yesterday
57ce9dd
Alisher: init frontend and MetaMask connect
4lishe3r committed yesterday
9cde2a4
Olzhas: add StartupBooster contract skeleton
Olzhasuh committed yesterday
ac94e3f
Olzhas: remove default Hardhat Lock example
Olzhasuh committed yesterday
9dbf670
restore: hardhat project files
Olzhasuh committed yesterday
0755c89
feat: add hardhat project files
Olzhasuh committed yesterday
f14920f
Initial commit
Olzhasuh authored yesterday
Verified
74843ac


Olzhasuh / student-startup-booster
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Branches
New branch
Overview
Yours
Active
Stale
All
Search branches...
Default
Branch
Updated
Check status
Behind
Ahead
Pull request
main
1 hour ago
Default
Your branches
feature/token-minter
2 hours ago
2
0
#3
feature/token-dias
4 hours ago
10
1
Active branches
feature/token-minter
2 hours ago
2
0
#3
feature/token-dias
4 hours ago
10
1



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