BLOCKCHAIN TECHNOLOGIES 1

Final Project Presentation

TEAM

Assem Rakhmanova
Aisana Kuanyshbek
Nurassyl Nurdilda

Technology Stack:
Solidity | Hardhat | JavaScript | MetaMask | Ethereum Testnet

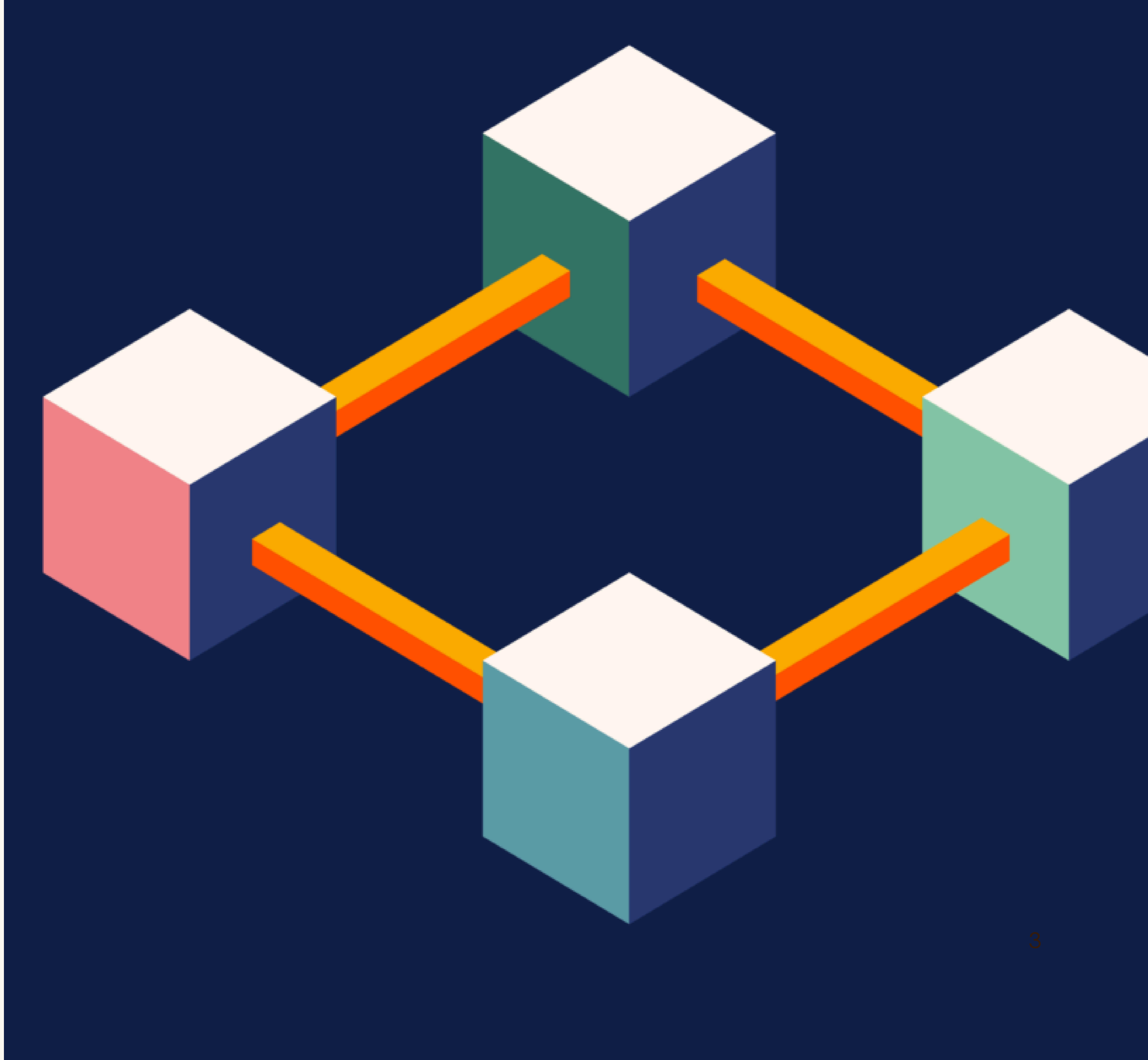
INTRODUCTION

Project Purpose

The purpose of this project is to:

- Design and implement smart contracts in Solidity
- Develop a decentralized application (DApp)
- Integrate MetaMask wallet
- Work with Ethereum test network
- Implement ERC-20 tokenization

The project demonstrates full blockchain interaction without using real cryptocurrency.



Project Idea

Our platform allows:

- A user to create a research project
- Other users to contribute test ETH
- Contributors to receive ERC-20 reward tokens
- Automatic status change after deadline

That’s it , simple and fully decentralized.

System Architecture

1

ResearchFunding.sol – Crowdfunding logic

2

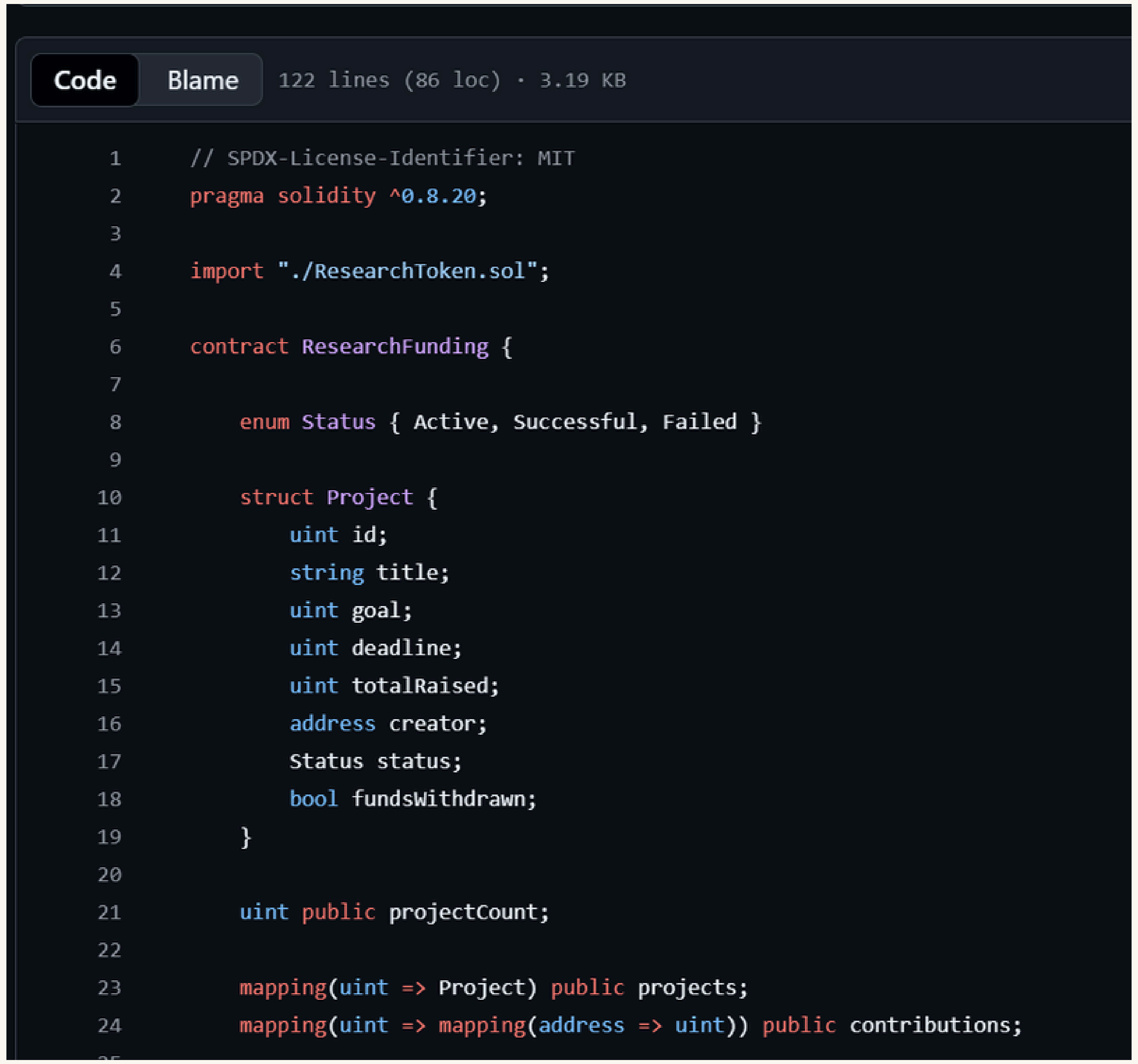
ResearchToken.sol – ERC-20 reward token

3

Frontend (HTML + JavaScript + MetaMask)

Flow:
User → Frontend → MetaMask → Smart Contract → Blockchain

Smart Contract: ResearchFunding.sol



Code Blame 122 lines (86 loc) · 3.19 KB

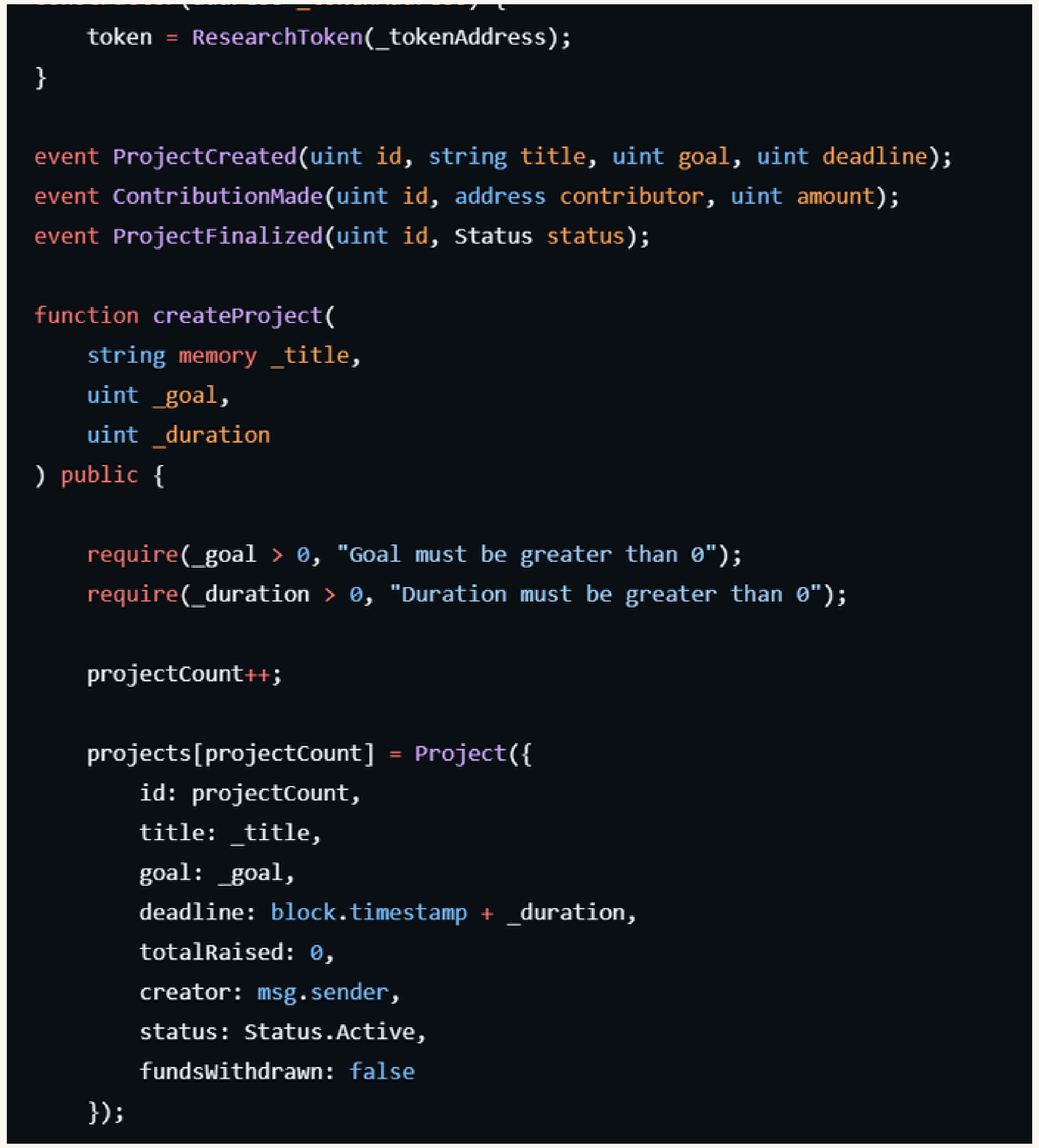
```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "./ResearchToken.sol";

contract ResearchFunding {

    enum Status { Active, Successful, Failed }

    struct Project {
        uint id;
        string title;
        uint goal;
        uint deadline;
        uint totalRaised;
        address creator;
        Status status;
        bool fundsWithdrawn;
    }

    uint public projectCount;

    mapping(uint => Project) public projects;
    mapping(uint => mapping(address => uint)) public contributions;
```

Main functionality:

- Create campaign (title, goal, duration)
- Contribute ETH
- Track individual contributions
- Finalize campaign
- Withdraw funds (if successful)
- Refund contributors (if failed)

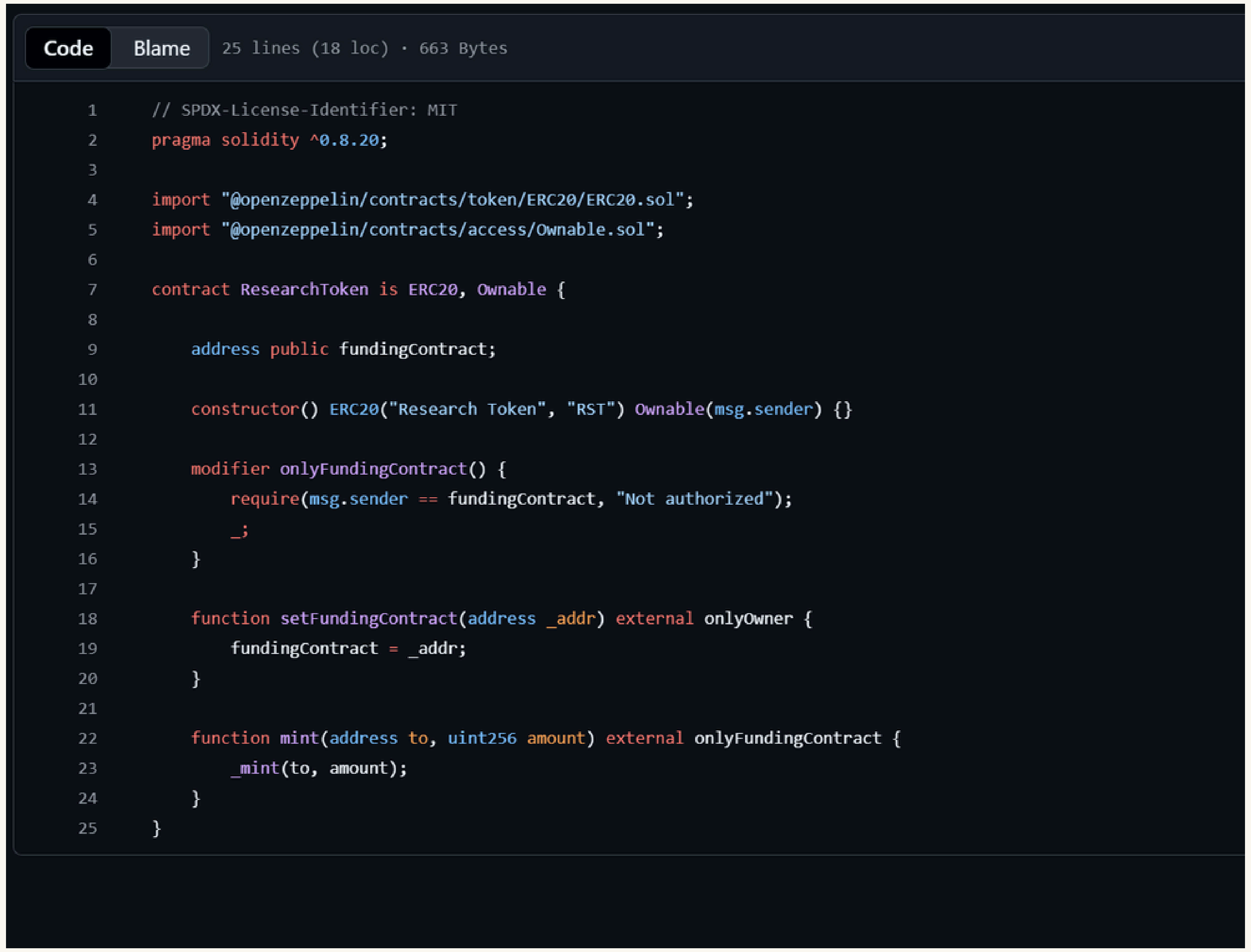
Key elements:

- struct Project
- enum Status { Active, Successful, Failed }
- mapping for contributions
- require validations
- events



```solidity
    token = ResearchToken(_tokenAddress);
}

event ProjectCreated(uint id, string title, uint goal, uint deadline);
event ContributionMade(uint id, address contributor, uint amount);
event ProjectFinalized(uint id, Status status);

function createProject(
    string memory _title,
    uint _goal,
    uint _duration
) public {

    require(_goal > 0, "Goal must be greater than 0");
    require(_duration > 0, "Duration must be greater than 0");

    projectCount++;

    projects[projectCount] = Project({
        id: projectCount,
        title: _title,
        goal: _goal,
        deadline: block.timestamp + _duration,
        totalRaised: 0,
        creator: msg.sender,
        status: Status.Active,
        fundsWithdrawn: false
    });
```

Smart Contract: ResearchToken.sol



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

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

contract ResearchToken is ERC20, Ownable {

    address public fundingContract;

    constructor() ERC20("Research Token", "RST") Ownable(msg.sender) {}

    modifier onlyFundingContract() {
        require(msg.sender == fundingContract, "Not authorized");
        _;
    }

    function setFundingContract(address _addr) external onlyOwner {
        fundingContract = _addr;
    }

    function mint(address to, uint256 amount) external onlyFundingContract {
        _mint(to, amount);
    }
}
```

Custom ERC-20 token:
Minted automatically during contribution
No real monetary value
Used only for educational purposes
Key features:
mint() function
onlyFundingContract modifier
1 ETH = 100 RST tokens
Demonstrates tokenization principles.

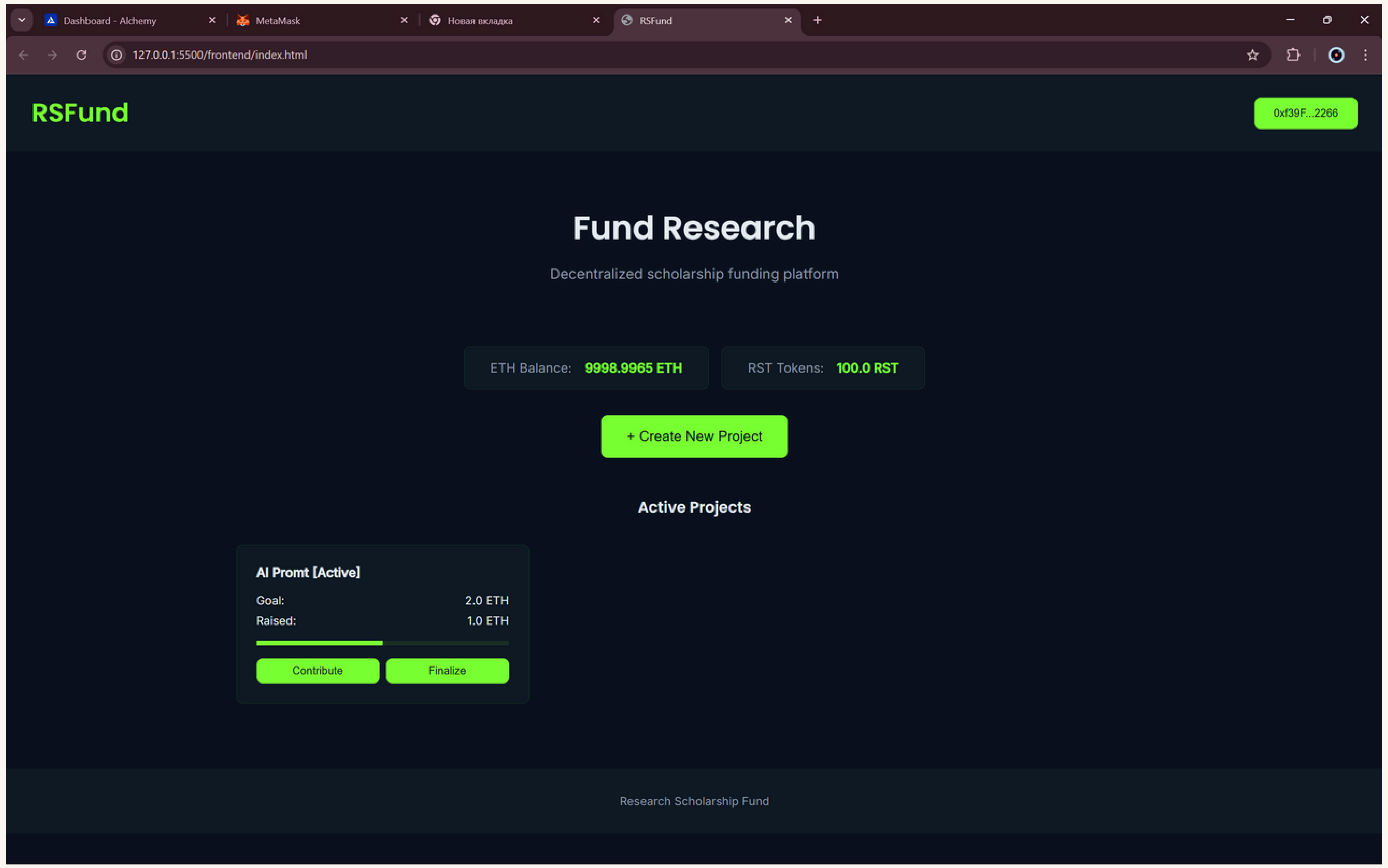
Crowdfunding Logic

User creates project → Users contribute ETH → Tokens are minted → After deadline:

- If goal reached → Successful
- Else → Failed

After a user creates a project, contributors send test ETH and receive ERC-20 tokens, and once the deadline is reached, the contract automatically sets the project status to Successful if the goal is met or Failed otherwise

MetaMask Integration



Fund Research

Decentralized scholarship funding platform

ETH Balance: 9998.9965 ETH

RST Tokens: 100.0 RST

Active Projects

AI Promt [Active]

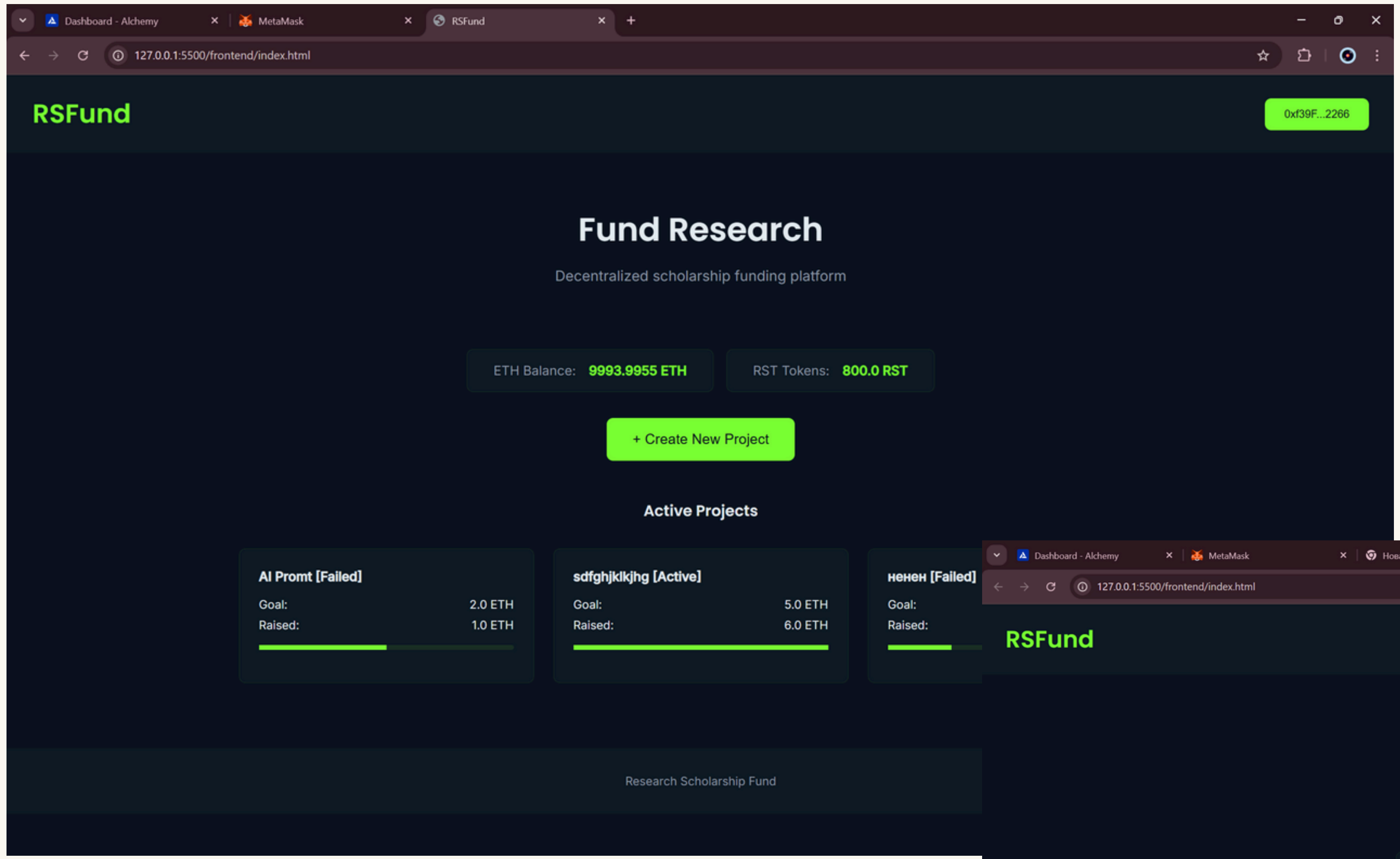
Goal:	2.0 ETH
Raised:	1.0 ETH

Research Scholarship Fund

The application:
Requests wallet access
Verifies selected test network
Sends transactions via MetaMask
Displays wallet address
Displays ETH balance
Displays token balance
All transactions are executed on Ethereum test network.

Frontend Functionality

User can:
Connect wallet
Create project
Contribute ETH
Finalize project
Withdraw or refund
See project status
See funding progress
The interface dynamically reads blockchain data.



Dashboard - Alchemy | MetaMask | RSFund

127.0.0.1:5500/frontend/index.html

RSFund

0xf39F...2266

Fund Research

Decentralized scholarship funding platform

ETH Balance: 9993.9955 ETH | RST Tokens: 800.0 RST

+ Create New Project

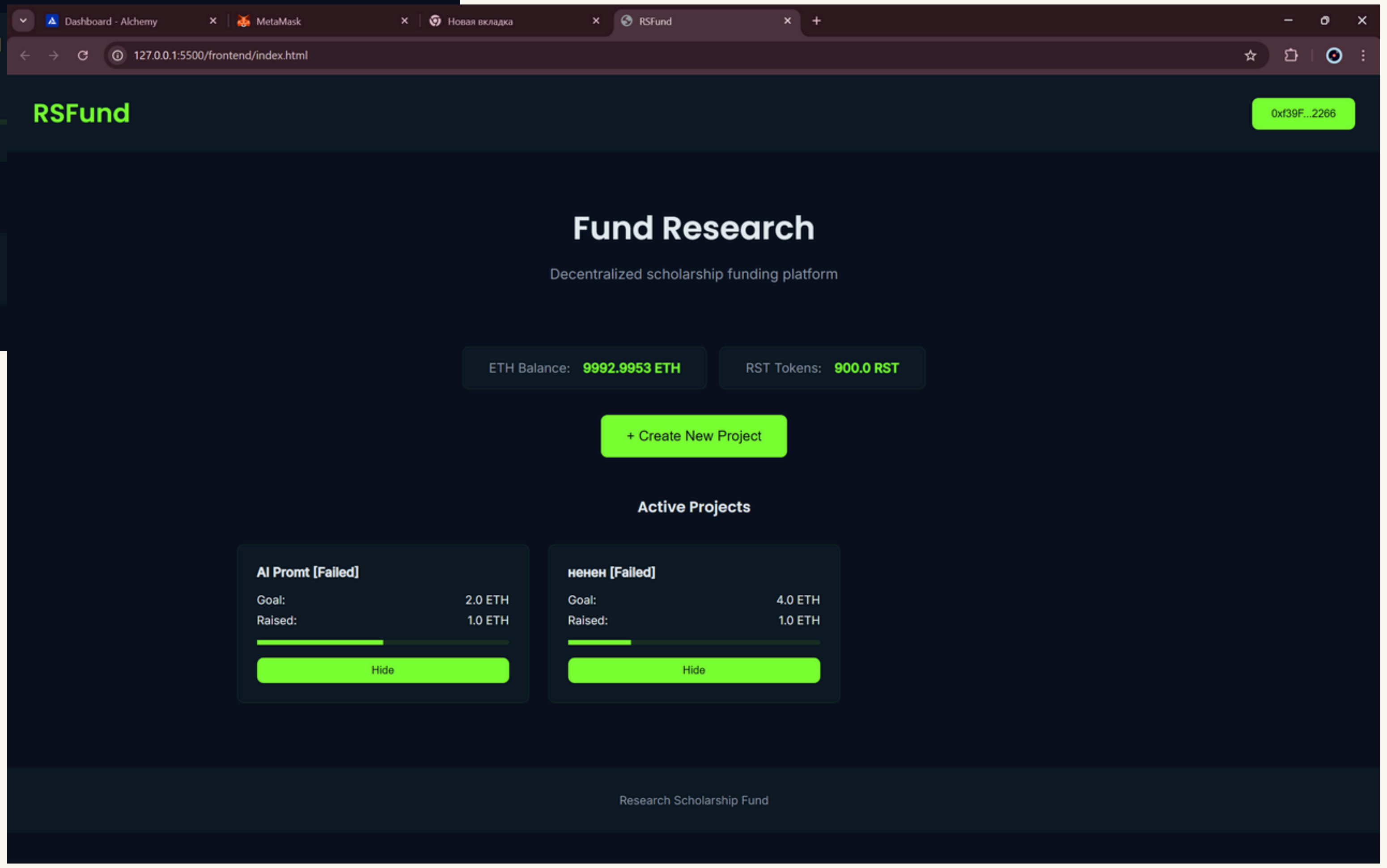
Active Projects

AI Promt [Failed]
Goal: 2.0 ETH
Raised: 1.0 ETH

sdfghjklkjhg [Active]
Goal: 5.0 ETH
Raised: 6.0 ETH

нененен [Failed]
Goal:
Raised:

Research Scholarship Fund



Dashboard - Alchemy | MetaMask | Новая вкладка | RSFund

127.0.0.1:5500/frontend/index.html

RSFund

0xf39F...2266

Fund Research

Decentralized scholarship funding platform

ETH Balance: 9992.9953 ETH | RST Tokens: 900.0 RST

+ Create New Project

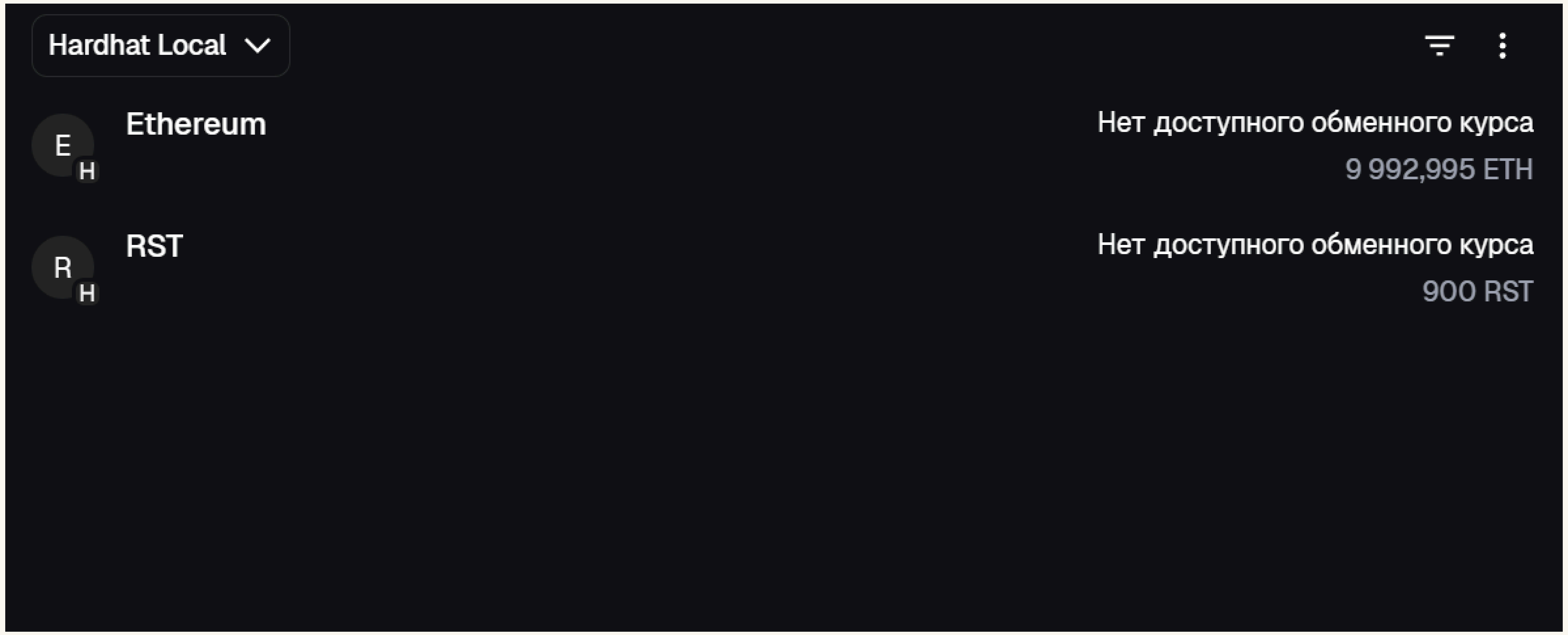
Active Projects

AI Promt [Failed]
Goal: 2.0 ETH
Raised: 1.0 ETH
Hide

нененен [Failed]
Goal: 4.0 ETH
Raised: 1.0 ETH
Hide

Research Scholarship Fund

Test Network Usage

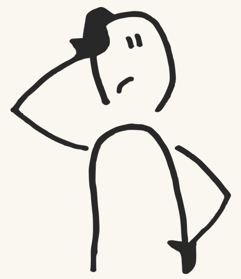

Hardhat Local

Ethereum — Нет доступного обменного курса — 9 992,995 ETH

RST — Нет доступного обменного курса — 900 RST

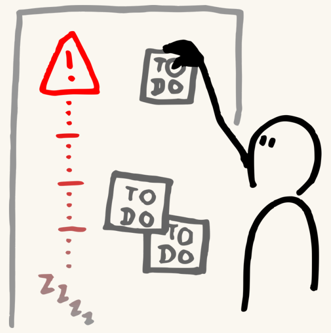
The project runs exclusively on:
Ethereum test network / local
Hardhat network
No real ETH is used.
All transactions use free test ETH.

Roles Distribution



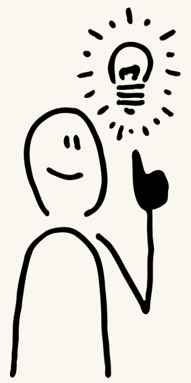
Nurassyl Nurdilda

- ResearchFunding.sol implementation
- Crowdfunding logic
- Deployment
- Testing
- Documentation preparation



Aisana Kuanyshbek

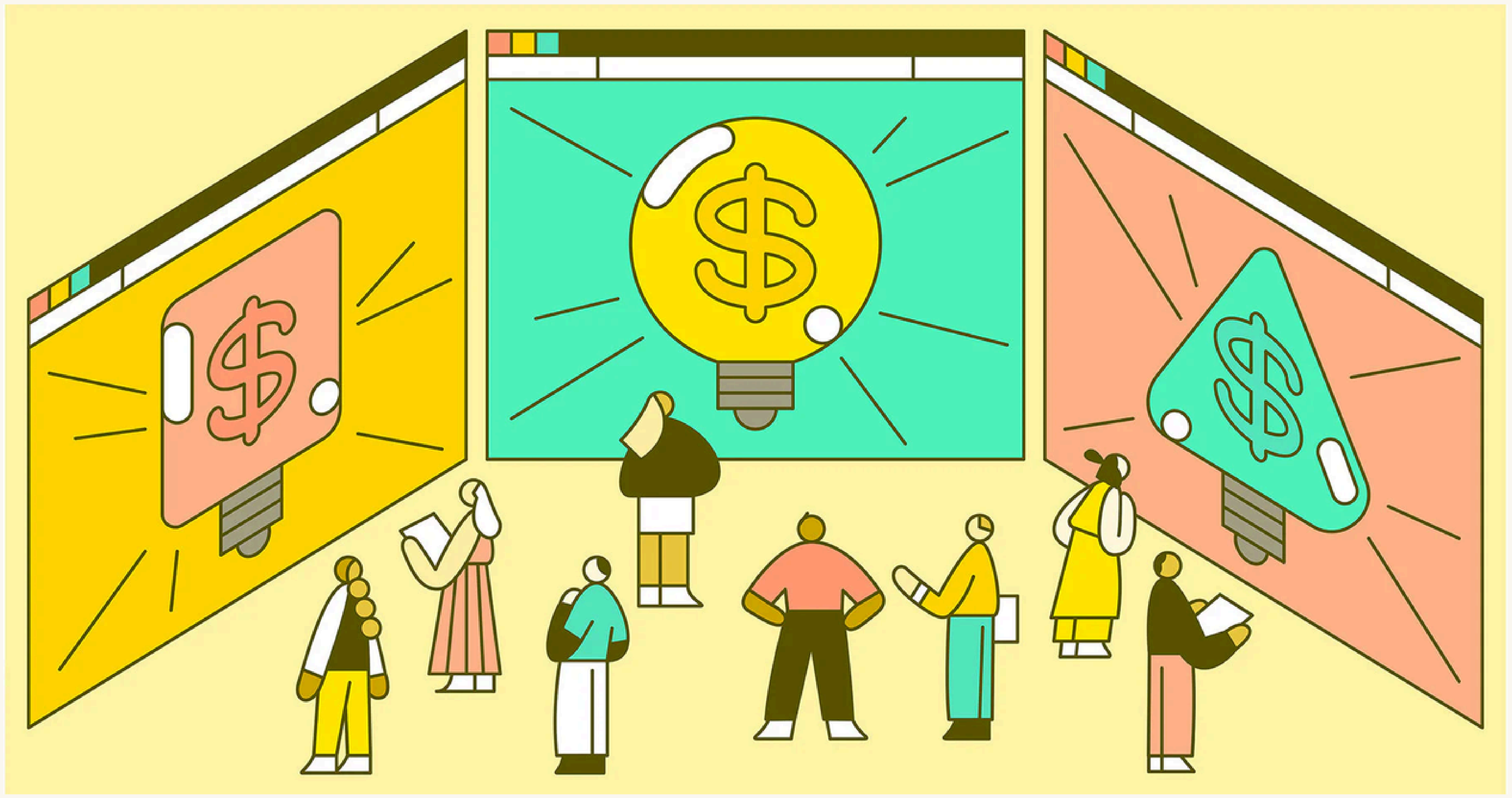
- ResearchFunding.sol implementation
- Crowdfunding logic
- Deployment
- Testing
- Documentation preparation



Assem Rakhmanova

- Frontend development
- ResearchToken.sol
- MetaMask integration
- Deployment
- Testing
- Documentation preparation

Conclusion



This project demonstrates smart contract architecture, ERC-20 token implementation, decentralized crowdfunding logic, secure MetaMask integration, and full blockchain interaction, fully satisfying all functional and academic requirements of the course.