Blockchain Final Project

Farkhad Imanbayev, Mendeke Seyitbaev, Timur Bizinskiy

Overview of the application architecture

The BeginUp project is an educational blockchain application.
It is created to show how crowdfunding works with smart contracts.

The project has **three** main parts.

1. **Frontend**
   The frontend is a web application.
   Users can see projects, send test ETH, and check their token balance.
   MetaMask is used to connect the wallet.
2. **Blockchain layer**
   This layer contains smart contracts.
   All main logic is inside the blockchain.
   Contracts are deployed only on a test network.
3. **Connection layer**
   The frontend connects to blockchain using **ethers.js**
   It sends transactions and reads data from smart contracts.

Explanation of Design and Implementation Decisions

Test network is used because real cryptocurrency is not allowed.
ERC-20 standard is used to show basic tokenization concepts and also because of requirements.

Minting is limited to one contract to prevent abuse.
Only the Crowdfund contract can create new tokens.

Burn function is added to show how tokens can be spent.
OpenZeppelin library is used because it is safe and well tested.

All decisions were made to keep the project simple and secure.

Description of Smart Contract Logic

BeginUpToken (ERC-20)

BeginUpToken is a reward token.
It has no real monetary value.

Main logic:

- Tokens are created when a user supports a project
- Only one contract can mint tokens
- Users can burn tokens to spend them

Main functions:

- setMinter() – sets the Crowdfund contract
- mint() – creates tokens for users
- burn() – removes tokens from user balance

BeginUpCrowdfund

BeginUpCrowdfund manages crowdfunding projects.

Main responsibilities:

- Create projects
- Accept test ETH
- Save user contributions
- Give tokens to users

When a user sends test ETH:

1. The contract receives ETH
2. The contribution is saved
3. Reward tokens are minted automatically

Frontend to Blockchain Interaction

The interaction works step by step:

1. User opens the website
2. User connects MetaMask wallet
3. Frontend creates a blockchain provider
4. User clicks “Support project”
5. MetaMask asks for transaction confirmation
6. Transaction is sent to the blockchain
7. Smart contract processes the contribution and mints reward tokens
8. Frontend receives transaction confirmation
9. Activation code is generated and sent to the user (gmail)
10. Frontend updates balances and shows success message

Deployment and Execution Instructions

Install dependencies:

npm install

Start local blockchain

Open a new terminal and run:
npx hardhat node
This command starts a local blockchain network.

Deploy smart contracts

Open another terminal and run:
npx hardhat ignition deploy ./ignition/modules/BeginUpModule.js --network localhost
Smart contracts will be deployed to the local network.

(Optional) Run project check script
make sure that you add deploy address to env file
npx hardhat run scripts/check-project.js --network localhost
This script checks that the project works correctly without starting the server.

Run backend server
node server.js

Seed example data
This script is used only as an example for testing
node seed.js

Description of the Process for Obtaining Test ETH

In our project - test ETH is obtained automatically.
We use a local blockchain provided by **Hardhat**.
When the Hardhat node starts, it creates test accounts with test ETH.

Command used:
npx hardhat node