FINAL EXAMINATION PROJECT

Studetns: Assel, Gaziza, Zhandos

Topic: Course Platform DApp

PURPOSE OF THE FINAL PROJECT

The global education industry is increasingly moving towards digital platforms that provide online courses. However, most existing platforms are centralized, meaning that user achievements, certificates, and rewards are controlled by a single entity. This can lead to issues such as fraudulent certificates, lack of transparency, and limited ownership of rewards.

Blockchain technology offers a solution by enabling decentralized applications (DApps) that are transparent, immutable, and trustless. By storing records and managing transactions on a blockchain, users can independently verify their achievements, own their certificates, and receive automated rewards without relying on a central authority.

Our project develops a decentralized course platform that demonstrates these principles. Using Ethereum smart contracts, the platform allows users to purchase courses using test ETH. Also, to receive ERC-20 bonus tokens as rewards upon course completion. Mint ERC-721 NFT certificates that prove course completion. As weel as, interact securely with the blockchain via MetaMask, ensuring safe transaction signing and wallet integration.

Tge purpose of this project is to provide students with a practical, hands-on experience in building blockchain applications, including smart contract development, tokenos, frontend blockchain integration, and decentralized identity verification through NFTs.

With this project we want to apply theoretical blockchain knowledge in a real-world scenario by creating a functional decentralized education platform. It demonstrates the principles of smart contract design and deployment on Ethereum testnets. Moreover, it has ERC-20 token

economics for rewarding users and ERC-721 NFT issuance for certification. Frontend-to-blockchain interaction using MetaMask. Transparent, immutable record-keeping on the blockchain.

Our decentralized application consists of three major components: 1.Smart Contracts -Logic deployed on the Ethereum test network that governs course purchases, reward token issuance, and certificate minting. 2.Frontend Interface - A user interface built in JavaScript that interacts with smart contracts and MetaMask. 3.MetaMask Wallet Integration - A browser extension wallet that enables authenticated users to sign transactions, verify network settings, and interact securely with the blockchain.

In this architecture, the frontend acts as the intermediary between the user's wallet and the smart contracts on the Ethereum test network. MetaMask provides a secure account and cryptographic signature support, and the smart contracts enforce the business logic and store persistent data on the blockchain. This architectural separation ensures decentralization, transparency, and immutability - core principles of blockchain systems.

We adopted the following design principles: Modularity: Smart contract functionality is separated into three contracts - BonusToken (ERC-20 token), CertificateNFT (ERC-721 certificate), and Course Platform. This modularity supports maintainability and future extensibility. Standards Compliance: Using OpenZeppelin components ensures that the ERC-20 and ERC-721 implementations are secure, well-audited, and adhere to Ethereum token standards. MetaMask for UX: MetaMask integration was chosen because it is the most widely used wallet for Ethereum DApps and supports secure signing of transactions directly in the user's browser.

Every user must transfer or state change (such as buying a course, minting tokens, or getting certificates) is executed through a transaction from MetaMask. These decisions align with recommended decentralized application patterns where client-side applications request user authorization to interact with smart contracts.

The smart contracts implement the key business logic of the platform. Course Registration and Pricing: The owner (deployer) sets available courses with their respective prices. Purchasing a Course: Users can pay test ETH to unlock access to a course. Completion and Rewards: Upon completion, users receive a proportional amount of ERC-20 reward tokens demonstrating the application of token economics. Certificate Issuance: Completed learners can mint an ERC-721 certificate NFT (CertificateNFT), which contains metadata such as the learner's name and associated course ID. All operations that change state (are transactions that get submitted to the Ethereum test network. These transactions update the contract state directly on the blockchain, ensuring transparency and traceability of user actions.

Our frontend, built in JavaScript, uses a blockchain library such as ethers.js or web3.js to connect to the Ethereum provider exposed by MetaMask. When a user interacts with the UI the frontend. It verifies the network is the selected Ethereum testnet. Sends signed transactions to the smart contract through MetaMask. Reads state from the blockchain for display purposes (like balances or owned certificates). This pattern follows the typical interaction model recommended by the Ethereum developer community, where the frontend never stores private keys - MetaMask handles signing securely.

To deploy and run the smart contract portion of the project we need to:

Install Dependencies.

Initialize the project and install development tools such as Hardhat, Solidity compiler, and Ethereum libraries.

Configure Networks

Include the Ethereum test network configuration in the Hardhat config file.

Compile Contracts

Use Hardhat to compile the smart contracts locally.

Deploy Contracts

Deploy the smart contracts to the Ethereum testnet via Hardhat scripts. Each deployed contract address should be noted for frontend configuration.

Configure Contract Interaction

Update the frontend application with the deployed contract addresses and ABI definitions.

Run Frontend

Start the frontend application locally and connect it to MetaMask.

These steps are standard in Ethereum smart contract development, as seen in official developer tutorials where tools like Hardhat, MetaMask, and test networks are used for testing before production deployment.

Interacting with the Ethereum blockchain requires ETH to pay for gas fees. ETH has no real monetary value and can be obtained for free using a testnet faucet. A faucet is a service that dispenses small amounts of test ETH to a user address. To obtain test ETH: Open MetaMask and switch to the test network. Enter your wallet address and request test ETH. Wait for the transaction to be completed; the test ETH will appear in your MetaMask balance. This process is part of standard Ethereum test network usage, allowing developers to pay for smart contract deployment and interaction without using real funds.