A logo with text on it

AI-generated content may be incorrect.

**NFT Ticketing System**

**Course title:** Blockchain Practices

**Instructor:** Akhmetov Timur

**Students:** Sagatova I., Kazbekova K., Kapatayaeva A.

Astana 2025

**Contents**

1. Introduction
2. Aims and Objectives
3. Concept and Idea
4. Research
5. Why We Chose This Topic
6. Code Implementation (to be added later)
7. Team Member Roles
8. Challenges
9. Conclusion

**Introduction**

The NFT Ticketing System is an innovative approach to event ticketing that leverages blockchain technology to ensure transparency, security, and immutability. Unlike traditional ticketing systems, NFTs (Non-Fungible Tokens) provide a unique, verifiable, and tamper-proof solution for issuing and transferring tickets.

**Aims and Objectives**

**Aims:**

* To develop a secure, transparent, and decentralized ticketing system using blockchain technology.
* To eliminate fraud and unauthorized reselling of tickets.
* To enhance user experience by enabling digital ownership and seamless transactions.

**Objectives:**

* Design and deploy a smart contract that mints NFT-based tickets.
* Develop a user-friendly interface for ticket buyers and event organizers.
* Utilize blockchain immutability to ensure ticket authenticity.
* Test and validate the security and efficiency of the system.

**Concept and Idea**

The NFT Ticketing System is based on smart contracts deployed on a blockchain network. Event organizers can mint NFT tickets and sell them to users. These NFTs are unique and cannot be duplicated, reducing the risk of counterfeiting. The system also allows secondary ticket sales through smart contract-enforced royalties, ensuring fair compensation to event organizers.

**Research**

We conducted research on:

* Existing ticketing challenges such as scalping, fraud, and lack of transparency.
* Blockchain applications in ticketing systems and their advantages.
* Case studies of NFT ticketing in real-world events (e.g., Coachella, UEFA).
* Ethereum smart contracts and NFT standards (ERC-721, ERC-1155).

**Why We Chose This Topic**

The current event ticketing system faces significant issues, including fraud, unauthorized resale, and lack of transparency. Blockchain and NFTs provide a robust solution by ensuring the authenticity and security of digital tickets. Additionally, our project aligns with the growing adoption of Web3 technologies and their practical applications in everyday transactions.

**Code Implementation**

**Code Aims**

* Implement a smart contract to mint and transfer NFT tickets.
* Integrate functionalities such as event creation, ticket sale, and resale tracking.
* Ensure security through Solidity best practices.

**Code Functions**

* **mintTicket:** Allows the event organizer to create and distribute NFT tickets.
* **transferTicket:** Enables users to securely transfer ownership of NFT tickets.
* **useTicket:** Marks a ticket as used to prevent reuse.

**Tools**

* **Remix 2:** Used for writing and deploying smart contracts.
* **Solidity:** Programming language for Ethereum smart contracts.
* **Metamask:** To interact with the blockchain.
* **Ethers.js:** Used for wallet creation and transaction handling.

**Smart Contract Code**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

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

contract TicketNFT is ERC721URIStorage, Ownable {

uint256 private \_ticketIdCounter;

mapping(uint256 => uint256) public ticketPrices;

mapping(uint256 => bool) public isTicketUsed;

event TicketMinted(uint256 indexed ticketId, address indexed owner, uint256 price);

event TicketTransferred(uint256 indexed ticketId, address indexed from, address indexed to);

event TicketUsed(uint256 indexed ticketId, address indexed owner);

constructor() ERC721("EventTicket", "ETIX") Ownable(msg.sender) {}

function mintTicket(address recipient, string memory tokenURI, uint256 price) public onlyOwner {

\_ticketIdCounter++;

uint256 ticketId = \_ticketIdCounter;

\_safeMint(recipient, ticketId);

\_setTokenURI(ticketId, tokenURI);

ticketPrices[ticketId] = price;

emit TicketMinted(ticketId, recipient, price);

}

function transferTicket(address to, uint256 ticketId) public {

require(ownerOf(ticketId) == msg.sender, "You are not the ticket owner");

require(!isTicketUsed[ticketId], "This ticket has already been used");

\_transfer(msg.sender, to, ticketId);

emit TicketTransferred(ticketId, msg.sender, to);

}

function useTicket(uint256 ticketId) public {

require(ownerOf(ticketId) == msg.sender, "You are not the ticket owner");

require(!isTicketUsed[ticketId], "This ticket has already been used");

isTicketUsed[ticketId] = true;

emit TicketUsed(ticketId, msg.sender);

}

}

**Generating an Ethereum Wallet**

To interact with the NFT Ticketing System, we first generate an Ethereum wallet. This wallet allows users to store, receive, and send NFT tickets securely. The wallet is created using ethers.js as follows:

import { ethers } from "ethers";

const wallet = ethers.Wallet.createRandom();

console.log("🔹 New Ethereum-wallet:");

console.log("address:", wallet.address);

console.log("private key:", wallet.privateKey);

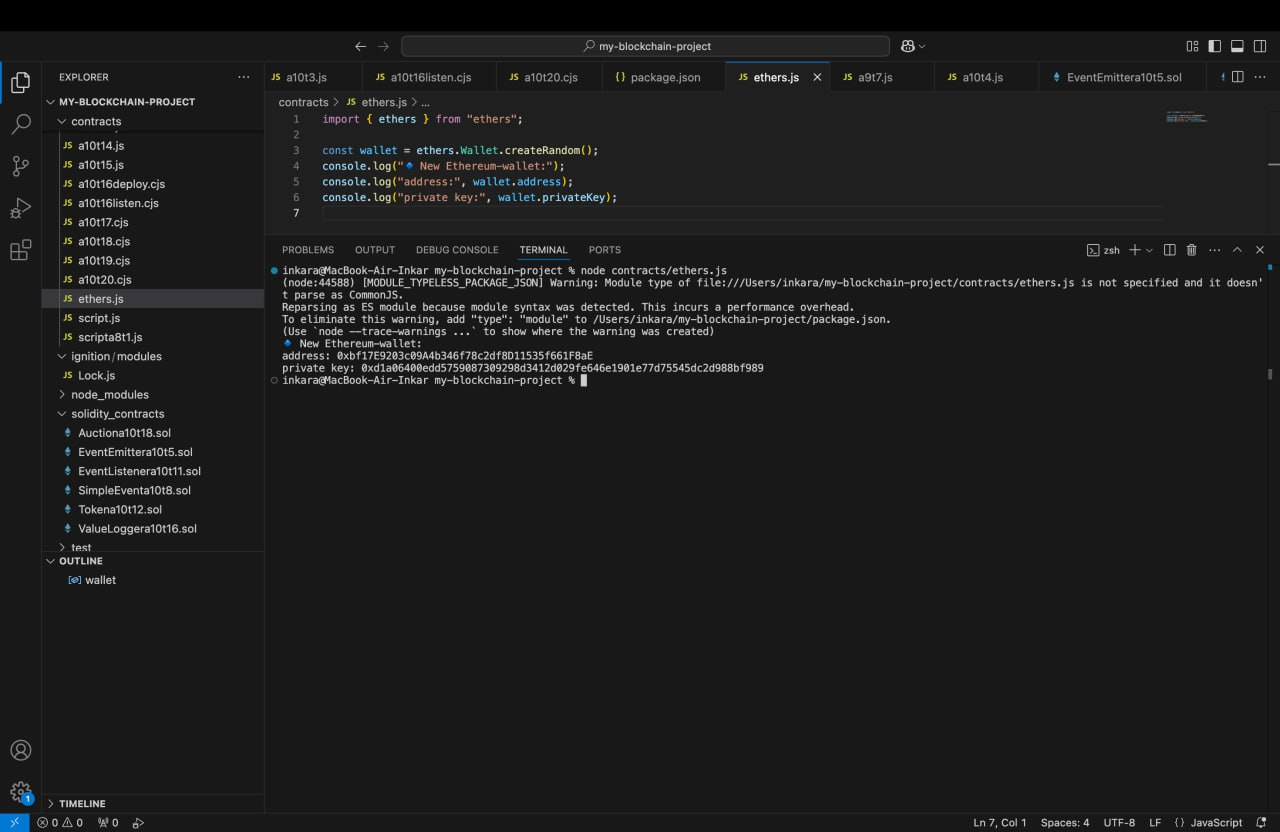
This code generates a completely new Ethereum wallet with a unique address and private key. The wallet address is used for blockchain interactions, while the private key should be securely stored and never shared. This step is crucial to ensure that only the rightful owner can interact with their NFT tickets.

**NFT Metadata Storage**

We uploaded the ticket image and metadata file to IPFS using Pinata, allowing for decentralized and immutable storage. The metadata file ticket.json contains attributes such as:

* **Name**: "VIP Ticket"
* **Description**: "Exclusive NFT ticket for an event."
* **Image**: IPFS URL
* **Attributes**: Seat number and Zone

The CID (Content Identifier) obtained from Pinata is then used in our smart contract to assign metadata to minted NFT tickets.



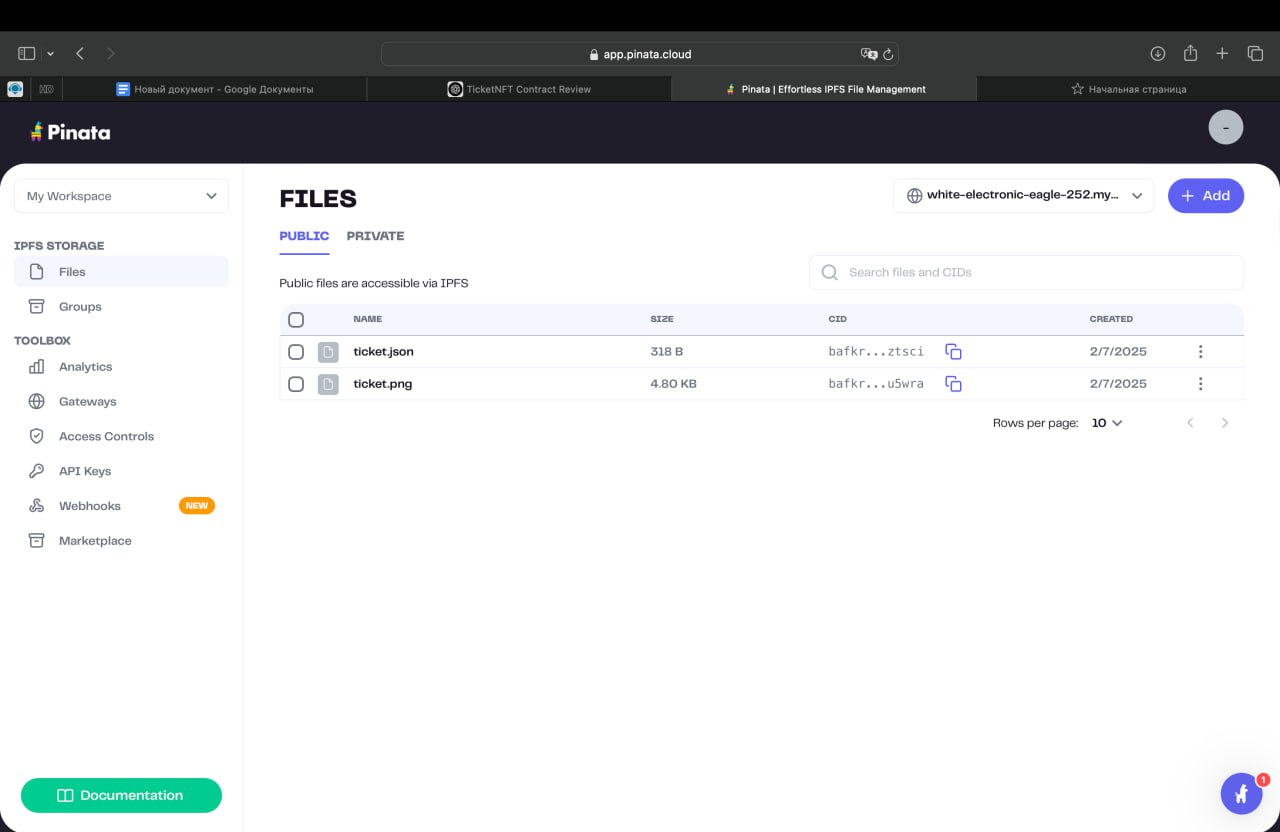
**Contract Deployment Steps**

To successfully deploy our NFT Ticketing System, we followed these steps:

**Step 1: Upload Ticket Image and Metadata to IPFS**

We used [Pinata Cloud](https://app.pinata.cloud/) to store our NFT metadata and ticket image on IPFS.

* The **ticket.png** image was uploaded to IPFS.
* We created **ticket.json**, which includes details like name, description, image link, and attributes (seat and zone).
* Pinata provided a **CID (Content Identifier)** that we used as the token URI in our smart contract.



A screenshot of a computer

AI-generated content may be incorrect.

**Step 2: Smart Contract Development**

We wrote a Solidity smart contract using OpenZeppelin’s ERC721 implementation. Our contract supports:

* **Minting NFT tickets** with a price.
* **Transferring tickets** between users.
* **Marking tickets as used** to prevent duplicate usage.

**Step 3: Deploy and Test in Remix IDE**

Using Remix IDE:

1. We compiled the **TicketNFT.sol** smart contract.
2. Selected **Injected Web3** to connect Metamask and deployed the contract to a testnet.
3. Used the mintTicket function to create a new NFT ticket with the IPFS metadata URL.
4. Verified that transactions were recorded on the blockchain.

**Step 4: Interacting with the Smart Contract**

* **mintTicket:** We minted an NFT ticket by passing the recipient address, IPFS metadata URL, and price.
* **transferTicket:** Transferred the NFT to another address.
* **useTicket:** Marked the ticket as used to ensure it couldn’t be reused.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**How We Called It in Remix IDE**

1. Opened the **deployed contract**.
2. Located and selected the **ownerOf function**.
3. Entered the **Token ID** (e.g., 1).
4. Clicked **"call"** to get the owner's address.

**Why It Matters**

* **Verifies ticket ownership** before transfer or use.
* Ensures **only the rightful owner** can take actions.
* Prevents fraud by **confirming authenticity** on the blockchain.

**Team Member Roles**

* **Sagatova I.:** Smart contract development, blockchain security.
* **Kazbekova K.:** Documentation, partly smart contract development.
* **Kapatayaeva A.:** Documentation, research, testing.

**Challenges**

* Understanding blockchain and NFT standards.
* Ensuring smart contract security and efficiency.
* Overcoming gas fees and blockchain transaction delays.

**Conclusion**

The NFT Ticketing System presents a viable alternative to traditional ticketing by leveraging blockchain’s decentralization, security, and transparency. This project demonstrates how NFTs can revolutionize the event industry by eliminating fraud, streamlining ticket transfers, and ensuring fair revenue distribution for event organizers.

By removing intermediaries, the system reduces costs and ensures fairer pricing for both event organizers and attendees. Additionally, the integration of blockchain technology ensures that tickets remain verifiable and cannot be counterfeited or resold at exploitative prices. Despite challenges like gas fees and adoption hurdles, NFT ticketing has the potential to become a standard in the event industry, offering enhanced trust, security, and efficiency for all stakeholders involved.