TASK 4 -NFT CREATION

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

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

import "@openzeppelin/contracts/utils/Strings.sol";

import "@openzeppelin/contracts/utils/Base64.sol";

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

contract StudentIDNFT is ERC721, Ownable {

using Strings for uint256;

struct Student {

string name;

string vtu; // e.g., "VTU22439"

}

uint256 public nextTokenId;

mapping(uint256 => Student) private \_studentOf;

event Minted(address indexed to, uint256 indexed tokenId, string name, string vtu);

// If your OZ version is v5+, keep Ownable(msg.sender).

// If you get a constructor-arg error, switch to: constructor() ERC721("StudentID NFT","SID") Ownable {}

constructor() ERC721("StudentID NFT", "SID") Ownable(msg.sender) {}

function mintStudentID(string calldata name\_, string calldata vtu\_)

external

returns (uint256 tokenId)

{

tokenId = nextTokenId++;

\_studentOf[tokenId] = Student({ name: name\_, vtu: vtu\_ });

\_safeMint(msg.sender, tokenId);

emit Minted(msg.sender, tokenId, name\_, vtu\_);

}

function studentOf(uint256 tokenId)

external

view

returns (string memory name\_, string memory vtu\_)

{

// ✅ Guard without \_requireMinted

ownerOf(tokenId); // reverts if not minted

Student memory s = \_studentOf[tokenId];

return (s.name, s.vtu);

}

function \_buildSVG(uint256 tokenId) internal view returns (string memory) {

// ✅ Guard without \_requireMinted

ownerOf(tokenId); // reverts if not minted

Student memory s = \_studentOf[tokenId];

string memory svg = string.concat(

'<svg xmlns="http://www.w3.org/2000/svg" width="720" height="420">',

'<defs><linearGradient id="g" x1="0" x2="1" y1="0" y2="1">',

'<stop offset="0%" stop-color="#4F46E5"/><stop offset="100%" stop-color="#06B6D4"/>',

'</linearGradient></defs>',

'<rect width="100%" height="100%" fill="url(#g)" rx="24"/>',

'<rect x="24" y="24" width="672" height="372" fill="white" rx="16" opacity="0.15"/>',

'<text x="50%" y="34%" dominant-baseline="middle" text-anchor="middle" ',

'font-family="Verdana, sans-serif" font-size="36" fill="#FFFFFF" font-weight="700">',

'STUDENT ID NFT</text>',

'<text x="50%" y="54%" dominant-baseline="middle" text-anchor="middle" ',

'font-family="Verdana, sans-serif" font-size="28" fill="#ECFEFF">Name: ', s.name, '</text>',

'<text x="50%" y="68%" dominant-baseline="middle" text-anchor="middle" ',

'font-family="Verdana, sans-serif" font-size="24" fill="#D1FAE5">VTU: ', s.vtu, '</text>',

'<text x="50%" y="84%" dominant-baseline="middle" text-anchor="middle" ',

'font-family="Verdana, sans-serif" font-size="16" fill="#E5E7EB">Token #', tokenId.toString(), '</text>',

'</svg>'

);

return string.concat("data:image/svg+xml;base64,", Base64.encode(bytes(svg)));

}

function tokenSVG(uint256 tokenId) external view returns (string memory) {

// ✅ Guard without \_requireMinted

ownerOf(tokenId); // reverts if not minted

return \_buildSVG(tokenId);

}

function tokenURI(uint256 tokenId) public view override returns (string memory) {

// ✅ Guard without \_requireMinted

ownerOf(tokenId); // reverts if not minted

Student memory s = \_studentOf[tokenId];

string memory attributes = string.concat(

'[{"trait\_type":"Name","value":"', s.name,

'"},{"trait\_type":"VTU","value":"', s.vtu, '"}]'

);

string memory imageDataURI = \_buildSVG(tokenId);

string memory json = string.concat(

'{"name":"Student ID #', tokenId.toString(),

'","description":"On-chain Student Identity NFT with name and VTU.",',

'"attributes":', attributes, ',',

'"image":"', imageDataURI, '"}'

);

return string.concat("data:application/json;base64,", Base64.encode(bytes(json)));

}

}

**DATA LINKS :**

**TOKEN SVG**

data:image/svg+xml;base64,

**TOKEN URL**

data:application/json;base64,

OUTPUT :

