# 451.https://stackoverflow.com/questions/70929408/best-plan-of-attack-for-digital-certificates-for-proof-of-completion-on-evm

**T:**Best plan of attack for digital certificates for proof of completion on EVM

**Q:**I am looking to explore the option of creating a digital certificate (as in proof) when someone has completed a portion of training, and for this to be issued on an EVM-compatible blockchain using Solidity.  
  
I have prototyped using ERC721 NFTs to encode a "certificate" however, I'd like to prevent recipients from being able to transfer these certificates. To prevent transfer, I attempted to use the Pause.sol functionality from OpenZeppelin, however, this would result in the entire contract being paused, as opposed to a specific tokenId.  
  
Does anyone have any recommendation on an approach? Am I overcomplicating it if I don't want recipients to be able to trade the certificates (i.e. for them to remain static)? Any pointers would be much appreciated!

1 **Answer**

**A1:**The simplest and most raw solution is to just set a mapping value.  
  
pragma solidity ^0.8;contract TrainingResults { enum Stage { NONE, STAGE\_1, STAGE\_2, COMPLETED } mapping (address => Stage) public participantStage; function setParticipantStage(address \_graduate, Stage \_stage) external { require(msg.sender == address(0x123), "Not authorized"); participantStage[\_graduate] = \_stage; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
Or if you want them to be able to see some kind of NFT in their wallet (that supports NFTs), you can modify the ERC-721 contract to disallow transfers.  
  
For example the OpenZeppelin implementation uses a function named \_beforeTokenTransfer() (GitHub link) that can be overwritten to disallow transfers altogether.  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
pragma solidity ^0.8;import "@openzeppelin/contracts/token/ERC721/ERC721.sol";contract TrainingResults is ERC721 { constructor() ERC721("TrainingResults", "TR") {} function \_beforeTokenTransfer(address from,address to, uint256 tokenId) override internal { // Allow only for the admin // as this function is called on token mint as well require(msg.sender == address(0x123), "Cannot transfer tokens"); }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]