1. Create an UUPSUpgradeable ERC721 token smart contract and UUPSUpgradeable ERC20 token smart contract.

Level: **Easy**

* Smart Contract:

The features required of the ERC721 contract:

* The token name will be “YOURNAME”
* Have a mint function with auto increment ID.
* Have a token burn, mint, and pause function function.
* Deploy the smart contract on the Ethereum Testnet and verify the source code in Etherscan
* Execute the following functions - mint, burn and pause function.

Feature of ERC20 contract:

* Create a token name “YOURNAME” with initial token supply of 100,000 tokens.
* Cap Supply: 1,000,000, Minimum supply: 100,000
* Add partial burn function: for every token transfer, burn 10% of transfer amount.
* Total supply = 1,000,000
* Verify the source code in Etherscan

* Frontend:
  + - Create a frontend that allows users to mint above tokens.
    - Deploy application on free hosting service(i.e: netlify, github pages) and send us the dapp URL.
    - Upload source code to github and share the repo with us.
    - Frontend features include but not limited to
    - Mint function
    - Display the token supply on your web page.

1. **Bonus!!** Deploy your own private blockchain on Ethereum and deploy below smart contracts on it. (Bonus question)

Level: Medium

* Set up a private blockchain that is EVM compatible and create a **RPC URL** which we can connect to using Metamask(Sent testing token to this address(Eth) 0x34846BF00C64A56A5FB10a9EE7717aBC7887FEdf)
* Create 3 smart contracts with the below functions.
  + 1. **First smart contract:** Create a function to check if the “msg.sender” is a contract address or a non-smart contract address by using the “extcodesize” function. This is to protect the smart contract from being interacted by other smart contracts.
    2. **Second smart contract:** Create a smart contract to interact with the first contract but it will fail due to the check function of the first contract.
    3. **Third smart contract:** Create a smart contract to interact with the first smart contract and successfully pass the check function of the first contract.
* Deploy all the smart contracts to the Ethereum Testnet (Ropsten/Rinkeby)
* Verify the source code in Etherscan.