Exam Preparation: NFT Marketplace

Your task is to create a **Solidity smart contract** named "CrowdfundingPlatform" that implements a decentralized crowdfunding platform.

1. Project Structure

Create a Hardhat project as an environment for the involved contracts, following the recommended structure and guidelines provided by Hardhat.

2. Smart Contract Functionality

NFT Creation

- Functionality
  The NFT Marketplace allows users to create and mint their NFTs with unique identifiers, names, descriptions, and associated metadata.
- Execution Process

  Users call the createNFT() function, providing the necessary parameters such as the metadata URL.

  The smart contract mints a new NFT with a unique identifier and assigns it to the user's address.

  The NFT URI is stored in the contract, providing details and a visual representation of the NFT (**proper metadata URL is not part of the requirements**).

NFT Listing

- Functionality

  Users can list their created NFTs on the marketplace for sale.

- Execution Process

  After creating an NFT, users can call the listNFTForSale() function, specifying the NFT identifier and the sale price in ETH.

  The NFT is listed on the marketplace with the specified sale price.

  Other users can now view the NFT listed for sale and have the opportunity to purchase it.

NFT Purchase

- Functionality
  Users can purchase NFTs listed on the marketplace using ETH and transfer ownership of the purchased NFTs to themselves.
- Execution Process:

  Interested users call the purchaseNFT() function, providing the NFT identifier and the payment amount in ETH.

  The smart contract checks if the NFT is available for sale and if the provided payment matches the sale price.

  If the conditions are met, the smart contract transfers ownership of the NFT from the seller to the buyer's address.

  The buyer is now the new owner of the NFT and can manage it accordingly.

Claim Profit

Apply a marketplace fee for each transaction, which is a percentage deducted from the sale price as a fee. Implement a mechanism to collect and distribute the marketplace fees.

- Functionality:
  Allow NFT creators to claim their share of the profits generated from secondary sales of their NFTs.

  Allow NFT sellers to claim their profits generated from sales of their NFTs.
- Execution Process:

  When an NFT creator calls the claimProfit() function, the smart contract calculates the total profits generated from sales for the specific user.

  The contract distributes the profits to the seller's address.

NB: The smart contract should be able to work with different NFT collections, not only the one which is created by the Marketplace.

3. General Requirements

- Smart Contracts must be implemented using the **Hardhat Development Environment**
- Smart Contracts must be written in **Solidity**.
- The application should have **Access Control** functionality. (e.g. NFT Listing creator)
- Smart contracts should be based on **OpenZeppelin contracts**.
- Unit tests for 100% of the **Listing** and **Purchase** functionalities logic must be implemented.
- **Deployment** task must be implemented
- **Interaction** tasks must be implemented for the **NFT Creation** and **Claim Profit** functionalities

4. Other requirements

- Proper **Licensing** information should be added
- **Alchemy** provider must be used for interaction with decentralized blockchain networks.
- **Deploy and Verify the Smart Contracts** on the Sepolia Ethereum Testnet network.
- Apply **error handling** and **data validation** to avoid crashes when invalid data is entered
- Demonstrate use of programming concepts - Smart Contracts Security, Gas Optimization, and Design Patterns

5. Bonuses

- Royalties and Secondary Sales: Implement a royalty mechanism where the original creator of an NFT receives a percentage of the sale price each time the NFT is sold in the future.
- Brief **documentation** on the project and project architecture (as .md file)

6. Assessment Criteria

General Requirements – 80 %

Other Requirements – 20 %

Bonuses – up to 10 %

Additional functionality or libraries outside the general requirements, with motivated usage.