Homework: Crowdfunding Platform

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 contract.

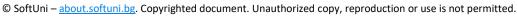
2. Smart Contract Functionality

- Project Creation
 - Allow users to create crowdfunding campaigns with a unique identifier, project name, description, funding goal (funds that must be collected), and duration.
- Contribution Mechanism
 - Enable users to contribute funds in ETH to support projects they are interested in.
 - Contributions can't be more than the funding goal.
- Release of Funds
 - The smart contract should hold contributed funds during the campaign duration and release them to the project creator upon reaching the funding goal.
- - Allow users to collect back the contributed funds in case the funding goal is not reached.
- **Basic Reward Distribution**
 - Implement a reward mechanism where the contributed funds are distributed proportionally to the project backers based on their contribution amount.
 - Example: The campaign creator distributes 100 ETH to the contract. Every supporter can claim their part of the reward which is proportional to their contribution. Let's say I've contributed 20 ETH and the total funds collected are 200 ETH, then I own 10% of the crowdfunding pool. So, I will receive 10 ETH from the distributed reward. Then the campaign creator distributes 50 ETH, and I can claim 5 ETH. Every distribution must be claimed separately.
- The shares must be kept by inheriting the ERC-20 standard. It allows users to be able to transfer their shares (buy/sell). Every campaign must be a separate ERC-20 token. (Hint: Create a main contract that handles the creation of new campaigns. Create a separate contract for every single campaign - because the ERC-20 standard needs a separate address for a token)

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. (campaign creator, e.g. owner)
- Smart contracts should be based on **OpenZeppelin contracts**.
- Unit tests for 100% of the Reward Distribution and Refund functionalities logic must be implemented.
- **Alchemy** provider must be used for interaction with the blockchain.
- Deployment task must be implemented
- Interaction tasks must be implemented for the Project Creation and Contribution mechanism functionalities

















4. Other requirements

- Proper Licensing information should be added
- 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

- Use CREATE2 and/or Minimal Proxy Factory as a deployment method
- 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.













