CSIT 6000Q - Blockchain and Smart Contracts
Assignment 2

Due: 11:59 PM on Sunday, Nov. 10, 2024
Released: October 12, 2024

Instructions

This assignment consists of four questions. Ensure that your solutions are well-documented and adhere to best practices in Solidity programming. For each question:

- **Problem Statement** describes the background and context of the problem to be solved.
- **System Features** lists the features that you need to implement in your smart contract.
- **Contract Interface** provides guidelines on what functions or constructs you might use to implement the solution. You are free to modify the interfaces as needed, but ensure that your implementation addresses all the features mentioned.

Submit your solutions as a ZIP file containing your Solidity files (.sol) and a README file explaining your design choices and any additional information that might be relevant.

Question 1: Decentralized Charity Fund Allocation with Governance Voting (25 points)

Problem Statement: A global charity organization aims to introduce a decentralized platform for managing its donations and fund allocations. The organization wants to give its donors more control and transparency over how the funds are utilized. To achieve this, they propose a system where donors contribute to a central fund and vote on fund allocation proposals submitted by charitable projects.

The organization plans to create a smart contract that facilitates token issuance, proposal submission, and voting. Donors will receive voting power proportional to their contribution amount, and charitable projects can submit funding requests to the system. When a funding request receives more than 50% of the total voting power, the requested funds are automatically disbursed to the project's address.

System Features:

- Donors can contribute Ether to the charity fund and receive proportional voting power.
- Charitable projects can submit funding requests, including the project address, requested amount, and project description.
- Donors can vote on funding requests using their voting power.
- A request is approved if it receives votes representing more than 50% of the total voting power.
- Upon approval, the requested amount is disbursed to the project's address, and the contract maintains a record of disbursements.

Contract Interface:

- `constructor: function DecentralizedCharityFund()`
- `donate: function donate() payable`
- `submitFundingRequest: function submitFundingRequest(address projectAddress, uint256 requestedAmount, string projectDescription)`
- `voteOnRequest: function voteOnRequest(uint256 requestId) returns (bool)`
- `finalizeRequest: function finalizeRequest(uint256 requestId) returns (bool)`
- `getFundingHistory: function getFundingHistory() view returns (address[] memory, uint256[] memory, string[] memory)`

Question 2: Decentralized Fan Engagement and Reward System (25 points)

Problem Statement: A popular sports league wants to enhance fan engagement by implementing a decentralized reward system that leverages blockchain technology. The goal is to create a platform where fans can earn and spend reward tokens based on their engagement activities, such as attending events, participating in social media campaigns, and supporting their favorite teams.

The league wants to create a smart contract that handles the issuance, transfer, and redemption of reward tokens, as well as manage fan activities and track their loyalty status. Fans can earn tokens by completing specific activities, and the league can also issue special non-fungible tokens (NFTs) as badges to recognize top contributors. Fans can redeem their tokens for merchandise or special experiences, such as VIP event access or a meet-and-greet with players.

System Features:

- **Token Issuance and Transfer:** A fan can earn tokens based on their engagement activities, and tokens should be transferable between fans.

- **Activity Submission and Verification:** Fans submit proof of their activities (e.g., posting about a team on social media), and a verifier can approve the activity to award tokens.
- **Loyalty Tiers:** Fans are classified into different loyalty tiers (e.g., Bronze, Silver, Gold) based on their accumulated token balance.
- **Special NFT Badges:** The league can mint special NFT badges for fans who reach specific milestones, such as attending 10 live games in a season.
- **Token Redemption:** Fans can redeem their tokens for rewards, such as merchandise or exclusive experiences. The contract should ensure that tokens are burned after redemption.
- **Proposal and Voting Mechanism:** Fans holding tokens can propose new reward ideas or vote on suggestions, making the system community-driven.

Contract Interface:

- `constructor: function FanEngagementSystem()`
- `earnTokens: function earnTokens(address fan, uint256 amount, string activityType, string activityProof)`
- `transferTokens: function transferTokens(address to, uint256 amount)`
- `redeemTokens: function redeemTokens(uint256 amount, string rewardType)`
- `mintNFTBadge: function mintNFTBadge(address fan, string badgeName)`
- `submitProposal: function submitProposal(string proposalDescription)`
- `voteOnProposal: function voteOnProposal(uint256 proposalId)`
- `getFanLoyaltyTier: function getFanLoyaltyTier(address fan) view returns (string)`
- `getRewardHistory: function getRewardHistory(address fan) view returns (string[])`

Question 3: Decentralized Rental Agreement Management (25 points)

Problem Statement: A property management company wants to digitize its rental agreements and provide tenants with a secure and transparent way to manage their lease. The company aims to implement a blockchain-based system to create rental agreements, record rent payments, and allow landlords to manage lease agreements.

The smart contract should allow landlords to create rental agreements, specifying tenant details, rent amount, and lease duration. Tenants can pay their rent through the smart contract, and the contract should keep a record of all payments. Landlords should have the ability to terminate agreements and retrieve agreement status at any time. The contract should emit events for key actions such as agreement creation, rent payments,

and termination.

System Features:

- Landlords can create rental agreements by specifying tenant details, rent amount, and duration.
- Tenants can pay rent through the contract, and payments are recorded.
- Landlords can terminate rental agreements at the end of the lease period or in case of violations.
- Events are emitted for key actions: agreement creation, rent payments, and termination.
- The contract maintains a record of the rental agreement status and payment history.

Contract Interface:

- `constructor: function RentalAgreementManagement()`
- `createAgreement: function createAgreement(address tenant, uint256 rentAmount, uint256 duration)`
- `payRent: function payRent(uint256 agreementId)`
- `terminateAgreement: function terminateAgreement(uint256 agreementId)`
- `getAgreementStatus: function getAgreementStatus(uint256 agreementId) view returns (string)`

Question 4: Decentralized Auction House (25 points)

Problem Statement: An artist collective wants to create a decentralized auction platform for selling digital artwork. The platform should allow artists to create auctions, set reserve prices, and specify auction durations. Interested buyers can place bids, and at the end of the auction period, the highest bidder should win the auction.

The system should support bid withdrawals if the auction is still active, and once the auction ends, the artist should be able to finalize it and transfer ownership of the digital artwork to the winning bidder. The contract should handle auction extensions if no valid bids are placed and prevent malicious behavior (e.g., placing bids and withdrawing them immediately).

System Features:

- Artists can create auctions by specifying the artwork details, reserve price, and auction duration.
- Buyers can place bids, and the highest bid at the end of the auction duration wins.
- Buyers can withdraw their bids if they are not the highest bidder and the auction is still active.

- Artists can finalize the auction, transferring ownership and disbursing funds.
- The contract should prevent malicious behavior and handle cases of auction extension.

Contract Interface:

- `constructor: function DecentralizedAuctionHouse()`
- `createAuction: function createAuction(string itemName, uint256 reservePrice, uint256 auctionDuration)`
- `placeBid: function placeBid(uint256 auctionId, uint256 bidAmount)`
- `withdrawBid: function withdrawBid(uint256 auctionId)`
- `finalizeAuction: function finalizeAuction(uint256 auctionId)`
- `getAuctionDetails: function getAuctionDetails(uint256 auctionId) view returns (string memory, uint256, uint256, bool)`