Blockchain-Based Simple Auction System

Objective:

Design and implement a smart contract that allows users to participate in a simple auction. Users can place bids using Ether (ETH), and after the auction duration ends, the highest bidder wins and the seller receives the funds.

Functional Requirements:

1. Auction Initialization:

- The contract should be initialized by the seller (owner) with a defined bidding duration (in seconds).

- The start time and end time of the auction should be recorded.

2. Place Bid:

- Any user (except the owner) can place a bid by sending Ether to the contract.

- A new bid must be higher than the current highest bid.

- If a user places a higher bid, the previous highest bidder must be eligible to withdraw their bid amount.

3. Withdraw Previous Bids:

- Users who are outbid should be able to withdraw their ETH safely.

4. End Auction:

- Only the owner can end the auction after the deadline.

- Once ended:

- No further bids are allowed.

- The highest bid amount is sent to the owner.

- The winner of the auction is recorded.

Blockchain-Based Simple Auction System

5. Events:

- Emit events for:

- New bids (NewBid)

- Auction end (AuctionEnded)

- Withdrawals (optional)

Constraints:

- Only bids placed before the end time are valid.

- No one should be able to place a bid after the auction ends.

- Only the contract owner can end the auction.

- ETH handling must be secure: no reentrancy vulnerabilities.

Bonus Objectives (Optional):

- Add a function to cancel the auction before it ends.

- Implement minimum bid increment (e.g., bids must be at least 0.01 ETH higher).

- Display the remaining time or status of the auction via a view function.

- Add a frontend using React + Ethers.js to interact with the contract.

Expected Outcome:

By the end of the project, you should be able to:

- Deploy and interact with a secure auction smart contract on a testnet.

- Allow multiple users to participate in real-time bidding.

- Handle bid tracking, refunding, and auction finalization using Solidity.