UPSTATE INTERACTIVE

CASE STUDY:

OpenZeppelin

```
contract DutchAuction is Ownable{
  using SafeMath for uint256;
  /// The address of the user auctioning off their NFT
  address public beneficiary;
  /// The address of the user that bids on the NFT
  address public bidder;
  /// The value of the bid from the bidder
  uint public bid;
  /// The difference between the bid and currentAskingPrice if bid is higher
  uint public overage;
  /// The number of days the auction will run for.
  /// See `startAuction` and `getCurrentAskingPrice` for how _auctionLength is used and why it's in days
  uint public auctionLength;
  /// The time at which the auction should start
  uint public startTime;
  /// The time at which the auction should end
  uint public endTime;
  /// The high asking price sought for the NFT
  uint public highAskingPrice;
  /// The absolute lowest price that would be accepted in returnfor the NFT
  uint public lowAskingPrice;
  /// The auction's current asking price for the NFT
  uint public currentAskingPrice;
  /// The NFT's contract's address
  address public tokenContract;
  /// The NFT's unique ID
  uint public tokenId;
  /// @dev create user-defined "Stages" type to manage state of the auction
  Stages public stage;
  enum Stages {
      AuctionDeployed,
      AuctionStarted,
      AuctionEnded
  }
  /// @dev Event for auction logging
  /// @param beneficiary who sold the NFT
  /// @param bidder who paid for the NFT
  /// @param bid amount in weis paid for the NFT
  event TokenAuctioned(
    address indexed beneficiary,
    address indexed bidder,
    uint256 bid,
    uint256 indexed tokenId
  );
  /// @dev Modifier use to ensure the auction is at the appropriate stage
  modifier validBid() {
    require(msg.value > 0);
    require(block.timestamp >= startTime);
    require(block.timestamp <= endTime);
    _;
  }
```

openzeppelin

CASE STUDY:

OpenZeppelin

BACKGROUND

Smart contracts deployed to the Ethereum blockchain often deal with large sums of real money. Ensuring their code is free of errors and secure is absolutely essential.

```
contract DutchAuction is Ownable{
  using SafeMath for uint256;
  /// The address of the user auctioning off their NFT
  address public beneficiary;
  /// The address of the user that bids on the NFT
  address public bidder;
  /// The value of the bid from the bidder
  uint public bid;
  /// The difference between the bid and currentAskingPrice if bid is higher
  uint public overage;
  /// The number of days the auction will run for.
  /// See `startAuction` and `getCurrentAskingPrice` for how _auctionLength is used and why it's in days
  uint public auctionLength;
  /// The time at which the auction should start
  uint public startTime;
  /// The time at which the auction should end
  uint public endTime;
  /// The high asking price sought for the NFT
  uint public highAskingPrice;
  /// The absolute lowest price that would be accepted in returnfor the NFT
  uint public lowAskingPrice;
  /// The auction's current asking price for the NFT
  uint public currentAskingPrice;
  /// The NFT's contract's address
  address public tokenContract;
  /// The NFT's unique ID
  uint public tokenId;
  /// @dev create user-defined "Stages" type to manage state of the auction
  Stages public stage;
  enum Stages {
      AuctionDeployed,
      AuctionStarted,
      AuctionEnded
  }
  /// @dev Event for auction logging
  /// @param beneficiary who sold the NFT
  /// @param bidder who paid for the NFT
  /// @param bid amount in weis paid for the NFT
  event TokenAuctioned(
    address indexed beneficiary,
    address indexed bidder,
    uint256 bid,
    uint256 indexed tokenId
  );
  /// @dev Modifier use to ensure the auction is at the appropriate stage
  modifier validBid() {
    require(msg.value > 0);
    require(block.timestamp >= startTime);
    require(block.timestamp <= endTime);
    _;
  }
```

openzeppelin

CASE STUDY:

OpenZeppelin

BACKGROUND

OpenZeppelin is a “battle-tested” open source framework comprised of reusable Ethereum smart contracts. The framework helps smart contract developers reduce the risk of vulnerabilities in their distributed applications (dapps) by using standard, tested, community-reviewed code.

OpenZeppelin smart contracts power $4,500,000,000 worth of digital assets as of the time of this writing. That’s right, $4.5 billion!

```
contract DutchAuction is Ownable{
  using SafeMath for uint256;
  /// The address of the user auctioning off their NFT
  address public beneficiary;
  /// The address of the user that bids on the NFT
  address public bidder;
  /// The value of the bid from the bidder
  uint public bid;
  /// The difference between the bid and currentAskingPrice if bid is higher
  uint public overage;
  /// The number of days the auction will run for.
  /// See `startAuction` and `getCurrentAskingPrice` for how _auctionLength is used and why it's in days
  uint public auctionLength;
  /// The time at which the auction should start
  uint public startTime;
  /// The time at which the auction should end
  uint public endTime;
  /// The high asking price sought for the NFT
  uint public highAskingPrice;
  /// The absolute lowest price that would be accepted in returnfor the NFT
  uint public lowAskingPrice;
  /// The auction's current asking price for the NFT
  uint public currentAskingPrice;
  /// The NFT's contract's address
  address public tokenContract;
  /// The NFT's unique ID
  uint public tokenId;
  /// @dev create user-defined "Stages" type to manage state of the auction
  Stages public stage;
  enum Stages {
      AuctionDeployed,
      AuctionStarted,
      AuctionEnded
  }
  /// @dev Event for auction logging
  /// @param beneficiary who sold the NFT
  /// @param bidder who paid for the NFT
  /// @param bid amount in weis paid for the NFT
  event TokenAuctioned(
    address indexed beneficiary,
    address indexed bidder,
    uint256 bid,
    uint256 indexed tokenId
  );
  /// @dev Modifier use to ensure the auction is at the appropriate stage
  modifier validBid() {
    require(msg.value > 0);
    require(block.timestamp >= startTime);
    require(block.timestamp <= endTime);
    _;
  }
```

openzeppelin

CASE STUDY:

OpenZeppelin

THE CHALLENGE

Earlier this year members of our team had a desire to sell digital collectibles (i.e., non-fungbile tokens or ERC-721 tokens). We sought out different platforms to use to sell them and realized very quickly that our options were limited.

At the same time, we were making contributions to the OpenZeppelin framework when we realized OpenZeppelin was missing a secure, smart contract that could be used to auction off digital collectibles. We decided to build it and share it with the community.

```
contract DutchAuction is Ownable{
  using SafeMath for uint256;
  /// The address of the user auctioning off their NFT
  address public beneficiary;
  /// The address of the user that bids on the NFT
  address public bidder;
  /// The value of the bid from the bidder
  uint public bid;
  /// The difference between the bid and currentAskingPrice if bid is higher
  uint public overage;
  /// The number of days the auction will run for.
  /// See `startAuction` and `getCurrentAskingPrice` for how _auctionLength is used and why it's in days
  uint public auctionLength;
  /// The time at which the auction should start
  uint public startTime;
  /// The time at which the auction should end
  uint public endTime;
  /// The high asking price sought for the NFT
  uint public highAskingPrice;
  /// The absolute lowest price that would be accepted in returnfor the NFT
  uint public lowAskingPrice;
  /// The auction's current asking price for the NFT
  uint public currentAskingPrice;
  /// The NFT's contract's address
  address public tokenContract;
  /// The NFT's unique ID
  uint public tokenId;
  /// @dev create user-defined "Stages" type to manage state of the auction
  Stages public stage;
  enum Stages {
      AuctionDeployed,
      AuctionStarted,
      AuctionEnded
  }
  /// @dev Event for auction logging
  /// @param beneficiary who sold the NFT
  /// @param bidder who paid for the NFT
  /// @param bid amount in weis paid for the NFT
  event TokenAuctioned(
    address indexed beneficiary,
    address indexed bidder,
    uint256 bid,
    uint256 indexed tokenId
  );
  /// @dev Modifier use to ensure the auction is at the appropriate stage
  modifier validBid() {
    require(msg.value > 0);
    require(block.timestamp >= startTime);
    require(block.timestamp <= endTime);
    _;
  }
```

openzeppelin

CASE STUDY:

OpenZeppelin

THE CONTRIBUTION

With our secure and reusable dutch auction smart contract, a user can submit their high asking price, their low asking price, and the length of time they would like their auction to run for. The high asking price decrements over time. Then, the first one to bid on the item is awarded the auctioner's digital collectible.

To satisfy OpenZeppelin's contribution guidelines we included unit tests, performed JavaScript and Solidity lint testing, and documented our code using Ethereum's NatSpec format.

```
contract DutchAuction is Ownable{
  using SafeMath for uint256;
  /// The address of the user auctioning off their NFT
  address public beneficiary;
  /// The address of the user that bids on the NFT
  address public bidder;
  /// The value of the bid from the bidder
  uint public bid;
  /// The difference between the bid and currentAskingPrice if bid is higher
  uint public overage;
  /// The number of days the auction will run for.
  /// See `startAuction` and `getCurrentAskingPrice` for how _auctionLength is used and why it's in days
  uint public auctionLength;
  /// The time at which the auction should start
  uint public startTime;
  /// The time at which the auction should end
  uint public endTime;
  /// The high asking price sought for the NFT
  uint public highAskingPrice;
  /// The absolute lowest price that would be accepted in returnfor the NFT
  uint public lowAskingPrice;
  /// The auction's current asking price for the NFT
  uint public currentAskingPrice;
  /// The NFT's contract's address
  address public tokenContract;
  /// The NFT's unique ID
  uint public tokenId;
  /// @dev create user-defined "Stages" type to manage state of the auction
  Stages public stage;
  enum Stages {
      AuctionDeployed,
      AuctionStarted,
      AuctionEnded
  }
  /// @dev Event for auction logging
  /// @param beneficiary who sold the NFT
  /// @param bidder who paid for the NFT
  /// @param bid amount in weis paid for the NFT
  event TokenAuctioned(
    address indexed beneficiary,
    address indexed bidder,
    uint256 bid,
    uint256 indexed tokenId
  );
  /// @dev Modifier use to ensure the auction is at the appropriate stage
  modifier validBid() {
    require(msg.value > 0);
    require(block.timestamp >= startTime);
    require(block.timestamp <= endTime);
    _;
  }
```

openzeppelin

CASE STUDY:

OpenZeppelin

THE FUTURE

We plan to continue making contributions to the OpenZeppelin framework and we intend to use OpenZeppelin's collection of vetted smart contracts to create more secure dapps for our clients in less time.