

LOTTERY SMART CONTRACT

## INTERNSHIP PROJECT REPORT

***by***

# GESWANTH REDDY SUNKARA 21KQ1A0462

****

## Department of ECE

Pace Institute of Technology and Sciences Ongole

June & 2024

# BONAFIDE CERTIFICATE

This is to certify that this project report entitled **Lottery Smart Contract** submitted to **Pace Institute of Technology and ,Ongole** is a bonafide record of work done by **GESWANTH REDDY SUNKARA** from 20/5/2024to 8/6/2024.

# Declaration by Author(s)

## This is to declare that this report has been written by us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be plagiarized, we are shall take full responsibility for it.

Geswanth reddy Sunkara

Roll.no:21KQ1A0462

## Place: Date :

# Abstract

The lottery smart contract in Solidity functions as a decentralized platform for conducting lotteries on the Ethereum blockchain. Managed by an administrator, participants enter the lottery by sending a designated amount of ether to the contract. After a predetermined period or when a specified condition is met, such as a set number of participants, the contract randomly selects a winner from the pool of participants. The winner receives the entire sum of ether collected, promoting transparency and fairness in the selection process. Utilizing blockchain technology, the contract ensures the integrity of the lottery by recording all transactions on the immutable blockchain ledger, providing a trustless environment for lottery participants.

**Introduction:**

**Problem Addressed:** This project aims to tackle the lack of fairness and transparency in traditional lottery systems by developing a decentralized lottery application using blockchain technology. The Lottery smart contract ensures that the selection process is random and verifiable, reducing the possibility of manipulation or fraud, and providing participants with confidence in the integrity of the lottery.

**Importance/Novelty:** The novelty of this project lies in harnessing the capabilities of blockchain to create a trustless and transparent environment for conducting lotteries. By automating the selection process through smart contracts, participants can be assured that winners are chosen randomly and fairly, without the need for intermediaries.

**Scope of the Project**: This project focuses on developing a functional smart contract for conducting lotteries on the Ethereum blockchain. The contract will handle participant entries, random selection of winners, and distribution of prizes. Additionally, the project will include testing and deployment of the smart contract.

**Brief Statements on What Subsequent Chapters Contain:** Subsequent chapters will delve into the detailed design and implementation of the lottery smart contract, including the algorithm for random selection and the mechanisms for participant interaction. The results and discussion chapter will present the outcomes of testing the contract's functionality, and the conclusion will summarize the project's achievements and propose areas for further development.

**Approach Used:**

The approach used in this project involves designing and implementing a lottery smart contract using Solidity. The contract includes the following key functionalities:

**Participant Entry:** Participants can enter the lottery by sending a specified amount of cryptocurrency to the contract.

**Random Selection:** The contract employs a verifiably random method to select a winner from the pool of participants.

**Prize Distribution:** Once a winner is selected, the contract automatically distributes the prize to the winner's address.

**Security Measures:** The contract incorporates various security measures to prevent tampering or manipulation of the lottery process.

**Results and Discussion:**

The lottery smart contract was thoroughly tested on the Ethereum blockchain to ensure its functionality and security.

Various test cases were executed to simulate different scenarios, including participant entries, random selection of winners, and prize distribution.

The results demonstrated that the contract functions as intended, with winners selected randomly and prizes distributed accurately.

**Conclusions and Future Work:**

The decentralized lottery smart contract developed in this project successfully addresses the shortcomings of traditional lottery systems by providing a fair and transparent alternative.

Future work could involve further optimizing the contract for scalability and efficiency, as well as exploring additional features such as multi-round lotteries or integration with other blockchain applications.

**References:** [1] Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform by Vitalik Buterin, 2014.

[2] Smart Contracts: How They Work and Why They Are So Important by CoinDesk, 2020.

[3] Introduction to Blockchain Technology and Smart Contracts by IBM, 2019.

[4] Solidity Documentation: The Ethereum Smart Contract Programming Language by the Ethereum Foundation.

### SOURCE CODE:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.11;

contract SimpleLottery {

    address public manager;

    address[] public players;

    uint public minimumBet;

    address public winner;

    uint public numOfTickets;

    constructor(uint \_minimumBet) {

        manager = msg.sender;

        minimumBet = \_minimumBet;

    }

    modifier onlyManager() {

        require(msg.sender == manager, "Only manager can call this function");

        \_;

    }

    function buyTicket() public payable {

        require(msg.value >= minimumBet, "Insufficient payment");

        players.push(msg.sender);

        numOfTickets++;

    }

    function random() private view returns(uint) {

        return uint(keccak256(abi.encodePacked(block.difficulty, block.timestamp, players.length)));

    }

    function pickWinner() public onlyManager {

        require(players.length > 0, "No players in the lottery");

        uint index = random() % players.length;

        winner = players[index];

        payable(winner).transfer(address(this).balance);

        players = new address[](0) ;

        numOfTickets = 0;

    }

    function getContractBalance() public view returns(uint) {

        return address(this).balance;

    }

}