Joint Account DApp Implementation

VISHAL PHARATE B21CS078

# **Introduction**

This report details the implementation of a decentralized application (DApp) on the Ethereum blockchain. The DApp allows users to create joint accounts, transfer amounts between users, and close accounts. The application ensures that users can only transact if there exists a path between them in the user network.

## Implementation Details

### Smart Contract

## The smart contract is implemented in Solidity and includes the following key features:

## **User Registration**:

## Users can register with a unique ID and name.

## The register User function ensures that a user cannot be registered more than once.

## **Joint Account Creation**:

## Users can create joint accounts with other users.

## The createAcc function initializes the balances for both users and updates their connections.

## **Amount Transfer**:

## Users can transfer amounts to other users if there exists a valid path between them.

## The send Amount function uses a Breadth-First Search (BFS) algorithm to find the shortest path and ensures sufficient balance along the path before transferring the amount.

## **Account Closure**:

## Users can close joint accounts.

## The close Account function deletes the account and updates the user connections.

## **State Reset**:

## The reset State function resets the entire state of the contract, useful for testing purposes.

## Python Script

## The Python script (interact.py) interacts with the deployed smart contract and performs the following tasks:

## User Registration: Registers 100 users with unique IDs and names.

* Joint Account Creation: Creates joint accounts between users with initial balances following an exponential distribution.
* Amount Transfer: Performs 1000 transactions between randomly selected users and tracks the success ratio.
* Plotting Results: Plots the success ratio of transactions over time.

## Findings

1. User Registration: The registration process works as expected, ensuring that each user is registered only once.
2. Joint Account Creation: Joint accounts are created successfully, and initial balances are set correctly.

The connections between users are updated appropriately.

1. Amount Transfer: The BFS algorithm effectively finds the shortest path between users.

Transactions are processed successfully if there is sufficient balance along the path.

The success ratio of transactions is plotted over time, showing the performance of the DApp.

1. Account Closure: Joint accounts are closed successfully, and connections are updated accordingly.
2. State Reset: The state reset function works as intended, allowing for easy testing and reinitialization.

# **Reasoning Behind Implementation Choices**

## Solidity for Smart Contract:

* 1. Solidity is the most widely used language for Ethereum smart contracts, providing robust features and extensive community support.

## Breadth-First Search (BFS) Algorithm:

* + BFS is chosen for finding the shortest path between users due to its simplicity and efficiency in unweighted graphs.

## Exponential Distribution for Initial Balances:

* + An exponential distribution is used to simulate realistic scenarios where some accounts have higher balances than others.

## Python for Interaction Script:

* 1. Python, along with Web3.py, provides a straightforward and powerful way to interact with Ethereum smart contracts.

## Plotting Success Ratios:

* + Plotting the success ratios helps visualize the performance and reliability of the DApp over time.

# Conclusion

## The Joint Account DApp successfully implements the required functionalities, including user registration, joint account creation, amount transfer, account closure, and state reset. The use of Solidity for the smart contract and Python for the interaction script ensures a robust and efficient implementation. The BFS algorithm effectively finds the shortest path between users, and the exponential distribution for initial balances simulates realistic scenarios. The success ratio plot provides valuable insights into the performance of the DApp.