Blockchain Project Report

KYC Ethereum DAPP

Group 8

Group Members:

Ananya Gautam
(2020AAPS2096H)
Shikhar Sachan
(2020A3PS2140H)
Shashwat Tripathi
(2020A3PS2210H)

KYC - DAPP Example

KYC means Know Your Customer. For many organizations, it is essential that they store their customer information on a reliable storage system.
For example, in the case of banks and finance companies, (or other identity sensitive institutions) if the customer fails to meet their KYC requirements banks will have the right to refuse to open their account and abort any business relationship.
This can be accomplished by employing a blockchain architecture for each individual customer, assigning them a unique identifier according to the info they provide.
Our project uses Ethereum blockchain for the same
This is a sample implementation of the solution of the KYC problem faced by many financial and other organizations.

Dependencies:

- Node: Run Javascript files locally
- NPM: Installing and managing the dependencies versions on the local environment
- Truffle: Compile all the smart contracts locally
- Ganache: Set up a local blockchain
- Metamask: Used to store account information of a customer

4 contracts-

1. Integers.sol

- Imports a library Integer - In summary this is a simple library of integer functions which allow a simple conversion to and from strings
- This file handles the conversion of unsigned integer, 8 bit integers to bytes

2. Strings.sol
    - This file imports the String library to handle strings without much difficulty in solidity
    - Helps in identifying if the customer is different from the previous one by checking its unique message generated

3. Migrations.sol
    - Contract to safely migrate a unique customer
    - Provide a new unique message

4. Verification.sol
    - Helps in storing the bank accounts with signature,
    - Insert new customer's signature
    - Verify new signatures

Steps to properly run the DAPP:

Step 1. Start Ganache

Step 2. Compile & Deploy Election Smart Contract

Step 3. Configure Metamask

Step 4. Run the Front End Application