The code

Candidate name : Umang Shrestha

Note : Red coloured lines are commented out lines and are only for explanation of the code

```
pragma solidity >0.5.16<0.9;

contract BankKYC {

    address kycAdmin;

    constructor() public{
        kycAdmin = msg.sender;
    }

    modifier onlyKYCAdmin(){
        require(msg.sender==kycAdmin);
        _;
    }

    struct BankDetails {
        string name;
        uint256 kycCount;
        address bankAddress;
        bool isAllowedToAddCustomer;
        bool kycPrivilege;
    }

    struct CustomerDetails {
        string name;
        string data;
        address validatedBank;
        bool kycStatus;
    }

    mapping(address => BankDetails) banksList; //  Mapping a bank's address to the Bank
    mapping(string => CustomerDetails) customersInfo;  //  Mapping a customer's username to the
Customer


/**************************************************************************************
*************************************
     *
     *  Name            :  addNewBankToKYCContract
     *  Access rights   :  KYC Contract Admin
     *  Description     :  This function can be used to add a new bank to the KYC Contract.
     *  Parameters      :        @param bankName :  Name of the bank/organisation.
```

```
    *                                   @param add :  Unique Ethereum address of the
bank/organisation
    *
**************************************************************************************
**********************************/
   function addNewBankToKYCContract(string memory bankName, address add) public
onlyKYCAdmin {
      require(!areStringsEqual(banksList[add].name, bankName), "Cannot add requested bank.
Bank with same name already exists.");
      banksList[add] = BankDetails(bankName, 0, add, true, true);
   }


/**************************************************************************************
************************************
    *
    *  Name             :  allowBankToDoKYC
    *  Access rights    :  KYC Contract Admin
    *  Description      :  This function can be used to enable the KYC Permission of
bank/organisation
    *  Parameters       :        @param add :  Unique Ethereum address of the bank/organisation
    *
**************************************************************************************
**********************************/
   function allowBankToDoKYC(address add) public onlyKYCAdmin returns(int) {
      require(!isValidBankAddress(add), "Invalid bank address");
      //require(banksList[add].bankAddress != address(0), "Invalid bank address");
      banksList[add].kycPrivilege = true;
      return 1;
   }


/**************************************************************************************
************************************
    *
    *  Name             :        blockBankToDoKYC
    *  Access rights    :  KYC Contract Admin
    *  Description      :  This function can be used to disable the KYC Permission of
bank/organisation
    *  Parameters       :        @param add :  Unique Ethereum address of the bank/organisation
    *
**************************************************************************************
**********************************/
   function blockBankToDoKYC(address add) public onlyKYCAdmin returns(int) {
      require(!isValidBankAddress(add), "Invalid bank address");
      //require(banksList[add].bankAddress != address(0), "Invalid bank address");
      banksList[add].kycPrivilege = false;
      return 1;
   }
```

```
/*************************************************************************************
************************************
    *
    *  Name             :  allowBankToAddNewCustomers
    *  Access rights    :  KYC Contract Admin
    *  Description      :  This function can be used to allow bank/organisation to add new customer.
    *  Parameters       :        @param add :  Unique Ethereum address of the bank/organisation
    *

*************************************************************************************
************************************/
   function allowBankToAddNewCustomers(address add) public onlyKYCAdmin returns(int){
      require(!isValidBankAddress(add), "Invalid bank address");
      //require(banksList[add].bankAddress != address(0), "Invalid bank address");
      require(!banksList[add].isAllowedToAddCustomer, "Requested bank is already allowed to add
new customers");
      banksList[add].isAllowedToAddCustomer = true;
      return 1;
   }


/*************************************************************************************
************************************
    *
    *  Name             :  blockBankToAddNewCustomers
    *  Access rights    :  KYC Contract Admin
    *  Description      :  This function can be used to block bank/organisation to add new customer.
    *  Parameters       :        @param add :  Unique Ethereum address of the bank/organisation
    *

*************************************************************************************
************************************/
   function blockBankToAddNewCustomers(address add) public onlyKYCAdmin returns(int){
      require(!isValidBankAddress(add), "Invalid bank address");
      //require(banksList[add].bankAddress != address(0), "Invalid bank address");
      require(banksList[add].isAllowedToAddCustomer, "Requested bank is already blocked to add
new customers");
      banksList[add].isAllowedToAddCustomer = false;
      return 1;
   }


/*************************************************************************************
************************************
    *
    *  Name             :  addNewCustomerToBank
    *  Access rights    :  Bank / Organisation
    *  Description      :  This function can be used by a valid bank/organisation to add new
customer.
    *  Parameters       :        @param custName :  Name of the customer
```

```
    *                                   @param custData :  Hash of the customer data as a string.
    *

**********************************************************************************
**********************************/
   function addNewCustomerToBank(string memory custName, string memory custData) public {
      require(banksList[msg.sender].isAllowedToAddCustomer, "Requested bank is blocked to add
new customers");
      require(customersInfo[custName].validatedBank == address(0), "Requested customer already
exists");

      customersInfo[custName] = CustomerDetails(custName, custData, msg.sender, false);
   }


/**********************************************************************************
************************************
    *
    *  Name             :  addNewCustomerRequestForKYC
    *  Access rights    :  Bank / Organisation
    *  Description      :  This function can be used to add the KYC request to the requests list. If
kycPermission is set to false then bank won't be allowed to add requests for any customer.
    *  Parameters       :        @param custName :  Name of the customer for whom KYC is to be
done
    *

**********************************************************************************
**********************************/
   function addNewCustomerRequestForKYC(string memory custName) public returns(int){
      require(banksList[msg.sender].kycPrivilege, "Requested bank does'nt have KYC privilege");
      customersInfo[custName].kycStatus= true;
      banksList[msg.sender].kycCount++;

      return 1;
   }


/**********************************************************************************
************************************
    *
    *  Name             :  viewCustomerData
    *  Access rights    :  Bank / Organisation
    *  Description      :  This function allows a bank to view details of a customer.
    *  Parameters       :        @param custName :  Name of the customer
    *

**********************************************************************************
**********************************/
   function viewCustomerData(string memory custName) public view returns(string memory,
bool){
      require(customersInfo[custName].validatedBank != address(0), "Requested customer doesn't
exists");
```

```
        return (customersInfo[custName].data, customersInfo[custName].kycStatus);
    }


/**********************************************************************************************************************************
    *
    * Name              :   getCustomerKycStatus
    * Access rights     :   Bank / Organisation
    * Description       :   This function is used to fetch customer kyc status from the smart contract. If true then the customer is verified.
    * Parameters        :         @param custName :  Name of the customer
    *
**********************************************************************************************************************************/
    function getCustomerKycStatus(string memory custName) public view returns(bool){
        require(customersInfo[custName].validatedBank != address(0), "Requested customer doesn't exists");
        return (customersInfo[custName].kycStatus);
    }


/**********************************************************************************************************************************
    *
    * Name       :   areStringsEqual
    * Description :   This is an internal function to verify equality of strings
    * Parameters  :     @param firstString :   1st String
    *                           @param secondString :   2nd String
    *
**********************************************************************************************************************************/
    function areStringsEqual(string memory firstString, string memory secondString) private pure returns (bool) {
        if(bytes(firstString).length != bytes(secondString).length) {
            return false;
        } else {
            return keccak256(bytes(firstString)) == keccak256(bytes(secondString));
        }
    }

        /**********************************************************************************************************************************
    *
    * Name       :   isValidBankAddress
    * Description :   This is an internal function to validate bank/organisation Ethereum address
    * Parameters  :     @param add :  Unique Ethereum address of the bank/organisation
    *
```

```
**********************************************************************************************
**********************************/
    function isValidBankAddress(address add) private view returns (bool) {
        if(banksList[add].bankAddress != address(0)) {
            return false;
        } else {
            return true;
        }
    }
}
```