Decentralized Identity Verification System Using Blockchain

Guided by:
Dr. Nitish Andola

Presented By :
Aditya Khedkar (24MCSA01)
Aditya Sharma (24MCSA02)
Amar Kumar (24MCSA03)
Ishita tyagi (24MCSA11)
Nehal Sahu (24MCSA16)

Table of contents

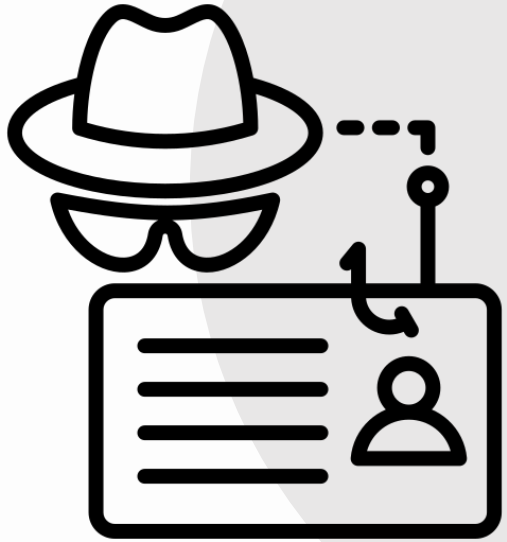
01

Introduction

What is Identity, Really?

Traditional Identity Systems

- Centralized
- Prone to hacks, misuse, and surveillance
- You don’t control your data — **they do**



The Identity Crisis

✓ *"Why Traditional Systems Fail"*

- Centralized Vulnerabilities: 1.5B+ records leaked in 2023 (single points of failure)
- Ownership Issues: Users don’t control their identity data
- Slow Verification: Manual processes cost businesses time/money
- $50B Annual Fraud: FTC-reported losses from identity theft

→ Our Target:

✓ *"Replace fragile centralized systems with blockchain-powered self-sovereign identity."*

Our Solution

✓ *"Decentralized Identity Verification System"*

→ *Key Features:*

- *Ethereum-based DApp for identity registration/verification*
- *User-owned identities via MetaMask wallets*
- *Tamper-proof records with smart contracts*
- *Admin-controlled verification system*

→ *Why It's Better:*

- *No central authority (fully decentralized)*
- *Instant verification status checks*
- *70%+ cost reduction vs manual processes*

02

PREREQUISITES

- ✓ **MetaMask** extension (connected to Sepolia test network).
- ✓ Some **test ETH** in your MetaMask wallet
- ✓ **Remix IDE** to compile and deploy smart contract
- ✓ **Solidity Compiler** version ^0.8.x
- ✓ **Contract ABI** saved in contractAbi.json
- ✓ Contract Address after deployment
- ✓ index.html with Web3.js for frontend interaction
- ✓ Local server to run frontend (Live Server preferred)
- ✓ Basic knowledge of Solidity, **JavaScript, and Web3.js**

03

Run the Project

Steps :

✓ Clone the Repository

git clone https://github.com/NehalSahu8055/Decentralised-Identity-Verification-Sysytem-Using-Blockchain

cd <project-folder>

✓ Deploy Smart Contract

→ Open *IdentityVerification.sol* in Remix IDE

→ Deploy it using Injected Provider - MetaMask

→ Copy the Deployed Contract Address

✓ Get the ABI

→ Go to artifacts → *IdentityVerification_metadata.json*

→ Copy the abi array

→ Paste it into your *contractAbi.json* file

✓ Update Frontend

→ In *index.html*, update:

→ *const contractAddress = "your_deployed_address_here";*

✓ Run the Frontend

→ Simply open *index.html* in your browser

→ Make sure MetaMask is connected to the same network

You can now Register, Verify, and View Identity via the UI!

04

Understanding Code

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

contract IdentityVerification {

    struct Identity {
        string fullName;
        uint256 idNumber;
        bool isVerified;
    }
```

```
mapping (address => Identity) public identities;

    event IdentityVerified(address indexed user);

    modifier onlyAdmin() {
        require(msg.sender == admin, "Only admin can call this
function");
        _;
    }

    address public admin;

    constructor() {
        admin = msg.sender;
    }
```

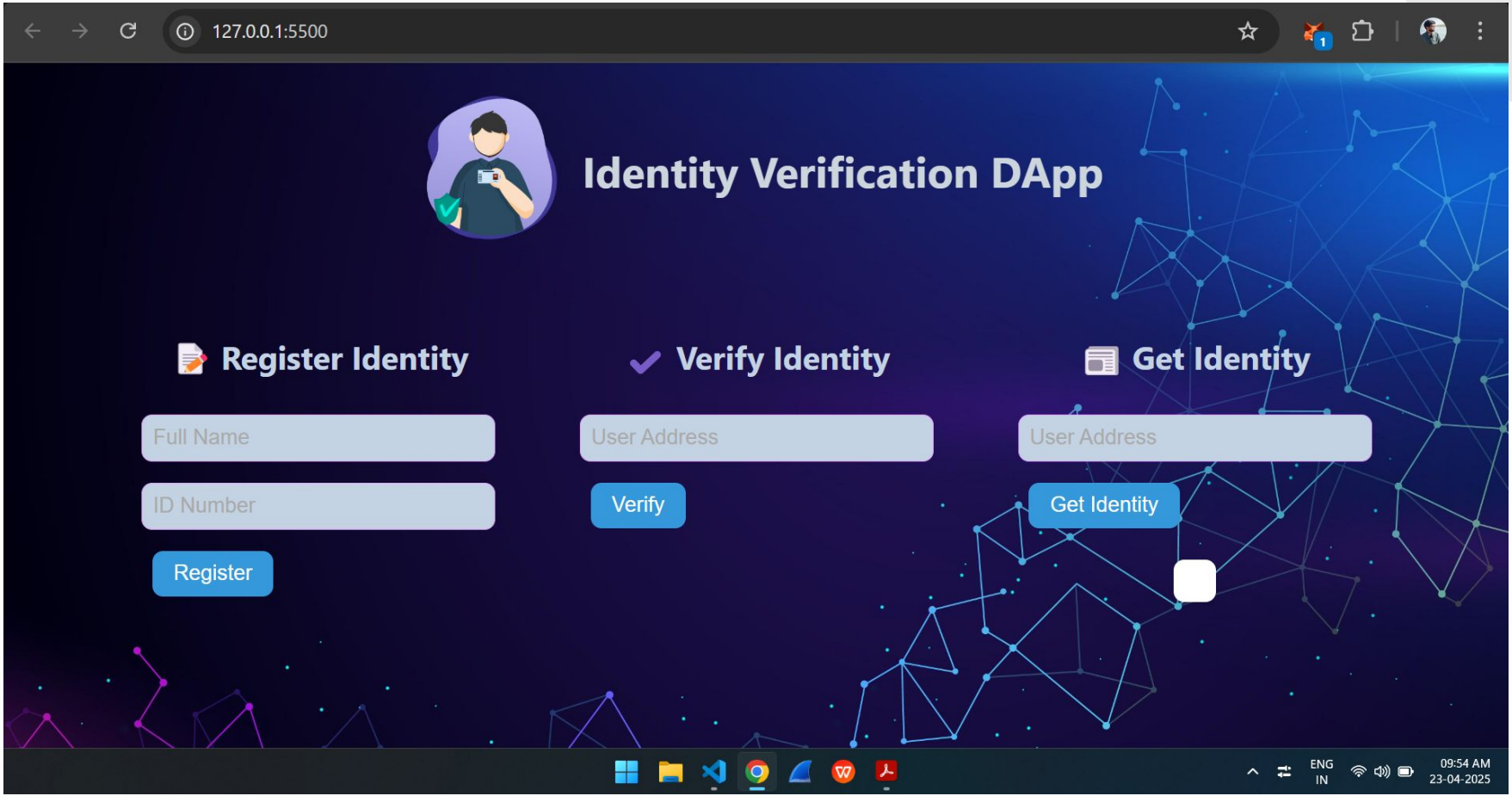
```
function registerIdentity(string memory _fullName, uint256
_idNumber) external {
        require(_idNumber > 0, "ID Number must be greater than
zero");
        require(bytes(_fullName).length > 0, "Full name must not
be empty");

        identities[msg.sender] = Identity({
            fullName: _fullName,
            idNumber: _idNumber,
            isVerified: false
        });
    }
```

```
function verifyIdentity(address _user) external onlyAdmin {
        Identity storage identity = identities[_user];
        require(bytes(identity.fullName).length > 0, "Identity
not registered");
        require(identity.idNumber > 0, "Invalid ID Number");
        require(!identity.isVerified, "Identity already
verified");

        identity.isVerified = true;
        emit IdentityVerified(_user);
    }
```

```
function getIdentity(address _user) external view returns
(string memory, uint256, bool) {
        Identity memory identity = identities[_user];
        return (identity.fullName, identity.idNumber,
identity.isVerified);
    }
```


127.0.0.1:5500
Identity Verification DApp
Register Identity
Full Name
ID Number
Register
Verify Identity
User Address
Verify
Get Identity
User Address
Get Identity
ENG
IN
09:54 AM
23-04-2025

05

Working

System Components

- **Main Components:**
  - **Smart Contract (IdentityVerification.sol)**:
    - Registers, verifies, and retrieves identities on the blockchain.
  - **Frontend (HTML, CSS, JS)**:
    - User interface for registering and verifying identities.
  - **MetaMask**:
    - Web3 wallet integration for interacting with the Ethereum network.
  - **IPFS** (Optional, for future upgrade):
    - Store identity data securely off-chain (CID).

06

Future Scope

Multi-Role Verification System

Current Limitation: Only the admin can verify users.

Future Scope:

- Add multiple verifiers (e.g., institutions, trusted individuals).
- Use DAO (Decentralized Autonomous Organization) for community voting to approve verifiers.

Build a verifier reputation system to track their credibility

- Document Upload via IPFS/Arweave
- Problem: Blockchain storage is expensive and not suitable for large files.
- Future Scope:
- Store identity documents (e.g., passport, Aadhar) on IPFS or Arweave.
- Store only the hash (unique fingerprint) on the blockchain.
- Ensures the document is authentic and untampere.

“In a world where data is currency, owning your identity is power.”

Thanks!

Any questions?