// 1. Secure, Decentralized Storage of User Data (Blockchain)

// ResultStorage.sol

pragma solidity ^0.8.0;

contract ResultStorage {

struct Result {

string username;

uint score;

string breakdown;

}

Result[] public results;

event ResultStored(string username, uint score, string breakdown);

function storeResult(string memory username, uint score, string memory breakdown) public {

results.push(Result(username, score, breakdown));

emit ResultStored(username, score, breakdown);

}

function getResult(uint index) public view returns (string memory, uint, string memory) {

Result memory result = results[index];

return (result.username, result.score, result.breakdown);

}

function getResultsCount() public view returns (uint) {

return results.length;

}

}

// 2. Personal Analysis and Breakdown of Results (Backend Logic)

// analysis.js

function analyzeResults(answers) {

// Placeholder for analysis logic

let score = 0;

let breakdown = {};

answers.forEach((answer, index) => {

// Example scoring logic

if (answer.correct) {

score += 10;

breakdown[`Question ${index + 1}`] = "Correct";

} else {

breakdown[`Question ${index + 1}`] = "Incorrect";

}

});

return { score, breakdown };

}

module.exports = analyzeResults;

// 3. Integration with Blockchain for Decentralized Storage (Backend)

// blockchainIntegration.js

const Web3 = require('web3');

const ResultStorageABI = [/\* ABI from compiled ResultStorage.sol \*/];

const contractAddress = "0xYourContractAddress";

const web3 = new Web3("https://mainnet.infura.io/v3/your-infura-project-id");

const resultStorageContract = new web3.eth.Contract(ResultStorageABI, contractAddress);

async function storeResultOnBlockchain(username, score, breakdown) {

const accounts = await web3.eth.getAccounts();

await resultStorageContract.methods

.storeResult(username, score, JSON.stringify(breakdown))

.send({ from: accounts[0] });

console.log("Result stored on blockchain successfully!");

}

module.exports = storeResultOnBlockchain;

// Example usage in backend server (server.js)

const analyzeResults = require('./analysis');

const storeResultOnBlockchain = require('./blockchainIntegration');

app.post('/submit', async (req, res) => {

const { username, answers } = req.body;

// Analyze results

const { score, breakdown } = analyzeResults(answers);

// Store result in database

const result = new Result({ username, score, breakdown });

await result.save();

// Store result on blockchain

await storeResultOnBlockchain(username, score, breakdown);

res.json({ success: true, message: 'Result saved successfully!', score, breakdown });

});