**FRONTEND:-**

**1.QUIZ.HTML:**

<!DOCTYPE html>

<html>

<head>

<title>Quiz App</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<h1>Take the Quiz</h1>

<div id="quiz-container"></div>

<button onclick="submitQuiz()">Submit</button>

<p id="result"></p>

<script src="https://cdn.jsdelivr.net/npm/web3@1.8.2/dist/web3.min.js"></script>

<script src="abi.js"></script>

<script src="app.js"></script>

</body>

</html>

**2.QUIZ \_LOGIC.JS:**

let web3;

let quizContract;

let account;

window.addEventListener('load', async () => {

if (window.ethereum) {

web3 = new Web3(window.ethereum);

await window.ethereum.enable();

account = (await web3.eth.getAccounts())[0];

quizContract = new web3.eth.Contract(abi, contractAddress);

loadQuiz();

} else {

alert("Install MetaMask.");

}

});

async function loadQuiz() {

const count = await quizContract.methods.getQuestionCount().call();

const container = document.getElementById("quiz-container");

container.innerHTML = "";

for (let i = 0; i < count; i++) {

const [question, options] = await quizContract.methods.getQuestion(i).call();

const div = document.createElement("div");

div.innerHTML = `<p>${question}</p>`;

options.forEach((opt, index) => {

div.innerHTML += `

<input type="radio" name="q${i}" value="${index}" /> ${opt}<br/>

`;

});

container.appendChild(div);

}

}

async function submitQuiz() {

const count = await quizContract.methods.getQuestionCount().call();

const answers = [];

for (let i = 0; i < count; i++) {

const options = document.getElementsByName(`q${i}`);

let selected = -1;

for (let opt of options) {

if (opt.checked) {

selected = parseInt(opt.value);

break;

}

}

if (selected === -1) {

alert("Please answer all questions.");

return;

}

answers.push(selected);

}

await quizContract.methods.submitAnswers(answers).send({ from: account });

const score = await quizContract.methods.getScore(account).call();

document.getElementById("result").innerText = `Your Score: ${score}/${count}`;

}

**3.STYLE.CSS:**

/\* General Reset and Body Styling \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #f4f6f8;

color: #333;

}

/\* Header \*/

header {

background-color: #2d3748;

color: #fff;

padding: 1rem 2rem;

text-align: center;

}

/\* Container \*/

.container {

max-width: 800px;

margin: 2rem auto;

background-color: #ffffff;

padding: 2rem;

border-radius: 8px;

box-shadow: 0 4px 12px rgba(0,0,0,0.1);

}

/\* Buttons \*/

button {

padding: 10px 20px;

background-color: #4a90e2;

color: white;

border: none;

border-radius: 6px;

cursor: pointer;

font-size: 16px;

margin-top: 1rem;

}

button:hover {

background-color: #357ab8;

}

/\* Input Fields \*/

input[type="text"],

select {

width: 100%;

padding: 10px;

margin: 0.5rem 0 1rem;

border: 1px solid #ccc;

border-radius: 4px;

}

/\* Quiz Question Card \*/

.question-card {

border: 1px solid #ddd;

border-radius: 6px;

padding: 1rem;

margin-bottom: 1.5rem;

background-color: #fafafa;

}

/\* Radio Options \*/

.quiz-option {

display: block;

padding: 8px;

margin: 8px 0;

background-color: #e6f0fa;

border-radius: 4px;

cursor: pointer;

}

.quiz-option:hover {

background-color: #d0e4f5;

}

/\* Score Display \*/

.score-box {

text-align: center;

font-size: 24px;

font-weight: bold;

margin-top: 1.5rem;

}

/\* Admin Panel Section \*/

.admin-panel {

border-top: 2px solid #ccc;

padding-top: 2rem;

margin-top: 2rem;

}

.admin-title {

font-size: 20px;

margin-bottom: 1rem;

color: #2c5282;

}

.hidden {

display: none;

}

/\* Footer \*/

footer {

margin-top: 3rem;

text-align: center;

font-size: 14px;

color: #aaa;

}

**4.ADMIN.HTML:**

<!DOCTYPE html>

<html>

<head>

<title>Admin Panel</title>

</head>

<body>

<h1>Quiz Admin Panel</h1>

<div>

<h3>Add New Question</h3>

<input id="newQuestion" placeholder="Enter question" /><br/>

<input id="opt0" placeholder="Option 1" /><br/>

<input id="opt1" placeholder="Option 2" /><br/>

<input id="opt2" placeholder="Option 3" /><br/>

<input id="opt3" placeholder="Option 4" /><br/>

<input id="correct" placeholder="Correct option index (0-3)" type="number" /><br/>

<button onclick="addQuestion()">Add Question</button>

</div>

<hr/>

<div>

<h3>Existing Questions</h3>

<div id="questionList"></div>

</div>

<script src="https://cdn.jsdelivr.net/npm/web3@1.8.2/dist/web3.min.js"></script>

<script src="abi.js"></script>

<script src="admin.js"></script>

</body>

</html>

**5.ADMIN\_LOGIC.JS:**

let web3;

let contract;

let account;

window.addEventListener("load", async () => {

if (window.ethereum) {

web3 = new Web3(window.ethereum);

await window.ethereum.enable();

account = (await web3.eth.getAccounts())[0];

contract = new web3.eth.Contract(abi, contractAddress);

const owner = await contract.methods.owner().call();

if (account.toLowerCase() !== owner.toLowerCase()) {

alert("Access denied: not contract owner.");

document.body.innerHTML = "Access denied.";

return;

}

loadQuestions();

} else {

alert("Install MetaMask.");

}

});

async function addQuestion() {

const question = document.getElementById("newQuestion").value;

const options = [

document.getElementById("opt0").value,

document.getElementById("opt1").value,

document.getElementById("opt2").value,

document.getElementById("opt3").value

];

const correct = parseInt(document.getElementById("correct").value);

if (!question || options.some(o => !o) || isNaN(correct) || correct < 0 || correct > 3) {

alert("Please fill in all fields correctly.");

return;

}

await contract.methods.addQuestion(question, options, correct).send({ from: account });

alert("Question added.");

loadQuestions();

}

async function loadQuestions() {

const count = await contract.methods.getQuestionCount().call();

const container = document.getElementById("questionList");

container.innerHTML = "";

for (let i = 0; i < count; i++) {

const [q, options] = await contract.methods.getQuestion(i).call();

const div = document.createElement("div");

div.innerHTML = `

<strong>Q${i + 1}: ${q}</strong><br/>

${options.map((opt, idx) => `${idx}. ${opt}`).join("<br/>")}<br/><br/>

`;

container.appendChild(div);

}

}

**SMART CONTRACTS:-**

**1.CONTRACT.SOL**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract QuizContract {

address public owner;

struct Question {

string question;

string[] options;

uint correctOption;

}

Question[] public questions;

mapping(address => uint) public scores;

modifier onlyOwner() {

require(msg.sender == owner, "Only owner can call this.");

\_;

}

constructor() {

owner = msg.sender;

}

function addQuestion(

string memory \_question,

string[] memory \_options,

uint \_correctOption

) public onlyOwner {

questions.push(Question(\_question, \_options, \_correctOption));

}

function editQuestion(

uint index,

string memory \_question,

string[] memory \_options,

uint \_correctOption

) public onlyOwner {

require(index < questions.length, "Invalid question index.");

questions[index] = Question(\_question, \_options, \_correctOption);

}

function getQuestionCount() public view returns (uint) {

return questions.length;

}

function getQuestion(uint index) public view returns (

string memory,

string[] memory

) {

require(index < questions.length, "Invalid index");

Question storage q = questions[index];

return (q.question, q.options);

}

function submitAnswers(uint[] memory userAnswers) public {

require(userAnswers.length == questions.length, "Answer count mismatch.");

uint score = 0;

for (uint i = 0; i < questions.length; i++) {

if (userAnswers[i] == questions[i].correctOption) {

score++;

}

}

scores[msg.sender] = score;

}

function getScore(address user) public view returns (uint) {

return scores[user];

}

}

**2.ABI.JS:**

// abi.js

const abi = [ /\* <-- Paste ABI from Remix/Hardhat here \*/ ];

const contractAddress = "YOUR\_DEPLOYED\_CONTRACT\_ADDRESS";