<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Online Test Series - Daily Test</title>

    <!-- Load Tailwind CSS -->

    <script src="https://cdn.tailwindcss.com"></script>

    <style>

        /\* Custom scrollbar for professional look \*/

        ::-webkit-scrollbar { width: 8px; height: 8px; }

        ::-webkit-scrollbar-thumb { background: #94a3b8; border-radius: 10px; } /\* Slate for a cleaner look \*/

        ::-webkit-scrollbar-track { background: #f1f5f9; }

        /\* Set default font to Inter \*/

        html { font-family: 'Inter', sans-serif; }

        /\* Hide specific screens initially \*/

        #test-view, #results-view { display: none; }

        /\* Styling for the sticky Subject/Question Panel on desktop \*/

        @media (min-width: 1024px) {

            #subject-panel {

                position: sticky;

                top: 0;

                height: 100vh;

                overflow-y: auto;

            }

        }

    </style>

</head>

<body class="bg-gray-50 min-h-screen">

    <script>

        // Use the Inter font, loaded via Tailwind

        tailwind.config = {

            theme: {

                extend: {

                    fontFamily: {

                        sans: ['Inter', 'sans-serif'],

                    },

                    colors: {

                        // New Professional Blue Palette (Sky Blue)

                        'primary': {

                            500: '#0ea5e9', // sky-500

                            600: '#0284c7', // sky-600 for hover/darker elements

                        }

                    }

                }

            }

        }

    </script>

    <!--

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    \* MAIN APPLICATION LOGIC (Module Script - All functions are exposed globally for onclick handlers)

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    -->

    <script type="module">

        import { createIcons, Timer, CheckCircle, List, ArrowLeft, ArrowRight, XCircle, Settings } from 'https://cdn.skypack.dev/lucide';

        // Initial static icons rendering

        createIcons({ icons: { Timer, CheckCircle, List, ArrowLeft, ArrowRight, XCircle, Settings } });

        const DEFAULT\_TESTS = [

            {

                id: "gk\_mock\_test\_1",

                title: "General Knowledge & Current Affairs - Test 1",

                description: "This test covers History, Geography, and Science with a strict 60-minute limit. Focus on accuracy and time management.",

                durationMinutes: 60,

                subjects: [

                    {

                        name: "History",

                        questions: [

                            { q: "Who built the Taj Mahal?", options: ["Akbar", "Shah Jahan", "Jahangir", "Aurangzeb"], isCorrect: 1 },

                            { q: "The Battle of Plassey was fought in which year?", options: ["1757", "1764", "1857", "1862"], isCorrect: 0 },

                            { q: "The Magna Carta was signed in?", options: ["1215", "1315", "1415", "1515"], isCorrect: 0 }

                        ]

                    },

                    {

                        name: "Geography",

                        questions: [

                            { q: "Which continent is the Amazon River primarily located in?", options: ["Africa", "Asia", "South America", "Europe"], isCorrect: 2 },

                            { q: "What is the capital city of Australia?", options: ["Sydney", "Melbourne", "Canberra", "Perth"], isCorrect: 2 },

                            { q: "The deepest oceanic trench on Earth is?", options: ["Tonga Trench", "Philippine Trench", "Mariana Trench", "Puerto Rico Trench"], isCorrect: 2 }

                        ]

                    },

                    {

                        name: "Science",

                        questions: [

                            { q: "What is the chemical symbol for gold?", options: ["Ag", "Au", "Fe", "Cu"], isCorrect: 1 },

                            { q: "Which planet is known as the 'Red Planet'?", options: ["Jupiter", "Mars", "Venus", "Saturn"], isCorrect: 1 },

                            { q: "The functional unit of the kidney is the:", options: ["Neuron", "Alveolus", "Nephron", "Hepatocyte"], isCorrect: 2 }

                        ]

                    }

                ]

            },

            {

                id: "math\_mock\_test\_2",

                title: "Quantitative Aptitude - Test 2",

                description: "Focused practice on core mathematics concepts, including algebra and geometry. (Mock test for demonstration).",

                durationMinutes: 45,

                subjects: [

                    {

                        name: "Mathematics",

                        questions: [

                            { q: "What is the square root of 144?", options: ["10", "12", "14", "16"], isCorrect: 1 },

                            { q: "If x + 5 = 10, what is x?", options: ["2", "3", "4", "5"], isCorrect: 3 }

                        ]

                    }

                ]

            }

        ];

        let TEST\_SERIES\_DATA = [];

        let currentTest = null;

        let userAnswers = {}; // Stores { questionKey: selectedOptionIndex }

        let timerInterval;

        let timeRemaining = 0; // in seconds

        let currentSubjectIndex = 0;

        let currentQuestionIndex = 0;

        let TOTAL\_QUESTIONS = 0;

        // --- Data Management Functions (Using localStorage to simulate persistent data) ---

        function loadTestsFromLocalStorage() {

            try {

                const storedData = localStorage.getItem('daily\_test\_series');

                // Only load stored data if it exists and contains more tests than the default code,

                // otherwise, use the code's DEFAULT\_TESTS (manual update)

                if (storedData && JSON.parse(storedData).length > DEFAULT\_TESTS.length) {

                    TEST\_SERIES\_DATA = JSON.parse(storedData);

                } else {

                    TEST\_SERIES\_DATA = DEFAULT\_TESTS;

                    saveTestsToLocalStorage();

                }

            } catch (e) {

                console.error("Error loading tests from localStorage:", e);

                TEST\_SERIES\_DATA = DEFAULT\_TESTS; // Fallback

            }

        }

        function saveTestsToLocalStorage() {

            try {

                localStorage.setItem('daily\_test\_series', JSON.stringify(TEST\_SERIES\_DATA));

            } catch (e) {

                console.error("Error saving tests to localStorage:", e);

            }

        }

        // --- Core Application Logic ---

        function getTotalQuestions(test) {

             return test.subjects.reduce((sum, s) => sum + s.questions.length, 0);

        }

        function handleRoute() {

            loadTestsFromLocalStorage(); // Always load fresh data

            const hash = window.location.hash.substring(1);

            const params = new URLSearchParams(hash);

            const view = params.get('view') || 'dashboard';

            const testId = params.get('test');

            document.getElementById('dashboard-view').style.display = 'none';

            document.getElementById('test-view').style.display = 'none';

            document.getElementById('results-view').style.display = 'none';

            stopTimer();

            // NEW: Contextual Header Logic

            const contextHeader = document.getElementById('context-header');

            switch (view) {

                case 'test':

                    currentTest = TEST\_SERIES\_DATA.find(t => t.id === testId);

                    if (currentTest) {

                        TOTAL\_QUESTIONS = getTotalQuestions(currentTest); // Update total for the current test

                        loadTest(currentTest);

                        document.getElementById('test-view').style.display = 'flex';

                        contextHeader.classList.remove('hidden');

                        createIcons({ icons: { ArrowLeft } }); // Re-render icon

                    } else {

                        window.location.hash = ''; // Fallback to dashboard

                    }

                    break;

                case 'results':

                    document.getElementById('results-view').style.display = 'block';

                    loadResults();

                    contextHeader.classList.remove('hidden');

                    createIcons({ icons: { ArrowLeft } }); // Re-render icon

                    break;

                case 'dashboard':

                default:

                    loadDashboard();

                    document.getElementById('dashboard-view').style.display = 'block';

                    contextHeader.classList.add('hidden'); // Hide on dashboard

                    break;

            }

        }

        function initializeAnswers() {

            userAnswers = {};

            if (currentTest) {

                currentTest.subjects.forEach((subject, sIndex) => {

                    subject.questions.forEach((q, qIndex) => {

                        userAnswers[`${sIndex}-${qIndex}`] = null; // null means unanswered

                    });

                });

            }

        }

        function loadTest(test) {

            initializeAnswers();

            timeRemaining = test.durationMinutes \* 60;

            currentSubjectIndex = 0;

            currentQuestionIndex = 0;

            renderTimer();

            renderSubjectPanel();

            renderQuestion();

            startTimer();

        }

        function startTimer() {

            if (timerInterval) clearInterval(timerInterval);

            const timerDisplay = document.getElementById('timer-display');

            timerInterval = setInterval(() => {

                timeRemaining--;

                if (timeRemaining <= 0) {

                    clearInterval(timerInterval);

                    timeRemaining = 0;

                    autoSubmitTest();

                }

                renderTimer();

                // Highlight timer when 5 minutes (300 seconds) remain

                if (timeRemaining <= 300) {

                    timerDisplay.classList.add('bg-red-50', 'text-red-600');

                    timerDisplay.classList.remove('bg-white', 'text-primary-600');

                }

            }, 1000);

        }

        function stopTimer() {

            if (timerInterval) clearInterval(timerInterval);

        }

        function renderTimer() {

            const minutes = Math.floor(timeRemaining / 60);

            const seconds = timeRemaining % 60;

            document.getElementById('timer-display').innerHTML = `

                <i data-lucide="Timer" class="w-4 h-4 mr-2"></i>

                ${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}

            `;

            createIcons({ icons: { Timer } });

        }

        function autoSubmitTest() {

            alertModal('Time Up!', 'Your time for the test has expired. The test will be automatically submitted now.', () => {

                submitTest();

            });

        }

        function renderSubjectPanel() {

            const panel = document.getElementById('subject-panel');

            let html = `<h2 class="text-xl font-bold mb-4 text-gray-700">Question Palette (${TOTAL\_QUESTIONS} Qs)</h2>`;

            currentTest.subjects.forEach((subject, sIndex) => {

                const isActiveSubject = sIndex === currentSubjectIndex;

                const answeredCount = subject.questions.filter((q, qIndex) => userAnswers[`${sIndex}-${qIndex}`] !== null).length;

                html += `

                    <div class="mb-4">

                        <button onclick="changeSubject(${sIndex})"

                            class="w-full text-left py-3 px-4 rounded-xl font-bold transition-colors flex justify-between items-center ${isActiveSubject ? 'bg-primary-600 text-white shadow-lg' : 'bg-white text-gray-800 hover:bg-primary-500/10 border border-gray-200'}">

                            <span>${subject.name}</span>

                            <span class="text-sm font-semibold ${isActiveSubject ? 'bg-white text-primary-600' : 'bg-gray-200 text-gray-700'} px-3 py-0.5 rounded-full">${answeredCount}/${subject.questions.length}</span>

                        </button>

                        ${isActiveSubject ? `

                            <div class="grid grid-cols-5 sm:grid-cols-6 gap-2 p-3 bg-white border border-t-0 rounded-b-xl shadow-inner">

                                ${subject.questions.map((q, qIndex) => {

                                    const key = `${sIndex}-${qIndex}`;

                                    const isAnswered = userAnswers[key] !== null;

                                    const isCurrent = sIndex === currentSubjectIndex && qIndex === currentQuestionIndex;

                                    let qClass = 'bg-gray-200 text-gray-700';

                                    if (isAnswered) {

                                        qClass = 'bg-primary-500 text-white shadow-md';

                                    } else if (isCurrent) {

                                        qClass = 'bg-yellow-400 text-black shadow-lg ring-2 ring-yellow-600';

                                    }

                                    return `<button onclick="changeQuestion(${sIndex}, ${qIndex})"

                                                class="w-full aspect-square text-base font-semibold rounded-lg transition-all ${qClass} hover:opacity-80">

                                                ${qIndex + 1}

                                            </button>`;

                                }).join('')}

                            </div>

                        ` : ''}

                    </div>

                `;

            });

            panel.innerHTML = html;

        }

        function renderQuestion() {

            const subject = currentTest.subjects[currentSubjectIndex];

            const question = subject.questions[currentQuestionIndex];

            const questionKey = `${currentSubjectIndex}-${currentQuestionIndex}`;

            const answeredIndex = userAnswers[questionKey];

            document.getElementById('question-header').textContent = `${subject.name} - Question ${currentQuestionIndex + 1} of ${subject.questions.length}`;

            const questionArea = document.getElementById('question-area');

            let html = `

                <div class="mb-8">

                    <p class="text-xl font-semibold text-gray-900 leading-relaxed">${currentQuestionIndex + 1}. ${question.q}</p>

                </div>

                <div id="options-container" class="space-y-4">

                    ${question.options.map((option, index) => {

                        const isSelected = answeredIndex === index;

                        const optionClass = isSelected

                            ? 'bg-primary-500 text-white shadow-lg ring-2 ring-primary-400'

                            : 'bg-white text-gray-800 hover:bg-gray-50 border border-gray-200';

                        return `

                            <div onclick="selectAnswer(${index})"

                                class="flex items-center p-5 rounded-2xl cursor-pointer transition-all duration-300 ${optionClass} hover:ring-2 hover:ring-primary-500/50">

                                <span class="w-8 h-8 flex items-center justify-center font-bold text-lg rounded-full border-2 ${isSelected ? 'border-white bg-white text-primary-600' : 'border-gray-400 text-gray-700 bg-gray-100'} mr-4 transition-all">

                                    ${String.fromCharCode(65 + index)}

                                </span>

                                <span class="flex-1 text-base font-medium">${option}</span>

                            </div>

                        `;

                    }).join('')}

                </div>

            `;

            questionArea.innerHTML = html;

            renderNavigationButtons();

            renderSubjectPanel(); // Update panel to reflect current question status

        }

        function renderNavigationButtons() {

            const navContainer = document.getElementById('question-navigation');

            // Check if there is a previous question

            let hasPrev = true;

            if (currentQuestionIndex === 0) {

                if (currentSubjectIndex === 0) {

                    hasPrev = false;

                }

            }

            // Check if there is a next question

            let hasNext = true;

            if (currentQuestionIndex === currentTest.subjects[currentSubjectIndex].questions.length - 1) {

                if (currentSubjectIndex === currentTest.subjects.length - 1) {

                    hasNext = false;

                }

            }

            navContainer.innerHTML = `

                <button onclick="navigateQuestion('prev')"

                    class="flex items-center px-6 py-3 rounded-xl font-semibold transition-colors disabled:opacity-50 disabled:cursor-not-allowed ${hasPrev ? 'bg-white text-gray-700 border border-gray-300 hover:bg-gray-100 shadow-md' : 'bg-gray-200 text-gray-500'}"

                    ${!hasPrev ? 'disabled' : ''}>

                    <i data-lucide="ArrowLeft" class="w-5 h-5 mr-2"></i> Previous

                </button>

                <button onclick="submitPrompt()"

                    class="px-6 py-3 bg-red-600 text-white rounded-xl font-bold shadow-lg shadow-red-500/50 hover:bg-red-700 transition-all transform hover:scale-[1.02]">

                    Finish & Submit Test

                </button>

                <button onclick="navigateQuestion('next')"

                    class="flex items-center px-6 py-3 rounded-xl font-semibold transition-colors disabled:opacity-50 disabled:cursor-not-allowed ${hasNext ? 'bg-primary-500 text-white shadow-md hover:bg-primary-600 shadow-primary-500/50' : 'bg-gray-200 text-gray-500'}"

                    ${!hasNext ? 'disabled' : ''}>

                    Next <i data-lucide="ArrowRight" class="w-5 h-5 ml-2"></i>

                </button>

            `;

            createIcons({ icons: { ArrowLeft, ArrowRight } });

        }

        // Expose to window for inline calls

        window.selectAnswer = function(optionIndex) {

            const questionKey = `${currentSubjectIndex}-${currentQuestionIndex}`;

            userAnswers[questionKey] = optionIndex;

            renderQuestion(); // Re-render to show selection

        }

        // Expose to window for inline calls

        window.navigateQuestion = function(direction) {

            let nextSIndex = currentSubjectIndex;

            let nextQIndex = currentQuestionIndex;

            if (direction === 'next') {

                if (nextQIndex < currentTest.subjects[nextSIndex].questions.length - 1) {

                    nextQIndex++;

                } else if (nextSIndex < currentTest.subjects.length - 1) {

                    nextSIndex++;

                    nextQIndex = 0;

                } else {

                    return;

                }

            } else if (direction === 'prev') {

                if (nextQIndex > 0) {

                    nextQIndex--;

                } else if (nextSIndex > 0) {

                    nextSIndex--;

                    nextQIndex = currentTest.subjects[nextSIndex].questions.length - 1;

                } else {

                    return;

                }

            }

            currentSubjectIndex = nextSIndex;

            currentQuestionIndex = nextQIndex;

            renderQuestion();

        }

        // Expose to window for inline calls

        window.changeQuestion = function(sIndex, qIndex) {

            currentSubjectIndex = sIndex;

            currentQuestionIndex = qIndex;

            renderQuestion();

        }

        // Expose to window for inline calls

        window.changeSubject = function(sIndex) {

            currentSubjectIndex = sIndex;

            currentQuestionIndex = 0;

            renderQuestion();

        }

        // Expose to window for inline calls

        window.submitPrompt = function() {

            const unansweredCount = Object.values(userAnswers).filter(a => a === null).length;

            const message = unansweredCount > 0

                ? `You have ${unansweredCount} unanswered questions. Are you sure you want to submit the test now?`

                : `You have answered all questions. Submit the test and view your results?`;

            alertModal('Confirm Submission', message, submitTest);

        }

        function submitTest() {

            stopTimer();

            window.location.hash = `#view=results&test=${currentTest.id}`;

        }

        function alertModal(title, message, onConfirm = null) {

            const modal = document.getElementById('custom-modal');

            document.getElementById('modal-title').textContent = title;

            document.getElementById('modal-message').textContent = message;

            const confirmButton = document.getElementById('modal-confirm');

            const cancelButton = document.getElementById('modal-cancel');

            const closeModal = () => {

                modal.classList.add('hidden');

                confirmButton.onclick = null;

            };

            cancelButton.onclick = closeModal;

            if (onConfirm) {

                confirmButton.textContent = 'Yes, Submit';

                confirmButton.classList.remove('hidden');

                confirmButton.onclick = () => {

                    closeModal();

                    onConfirm();

                };

            } else {

                confirmButton.textContent = 'Okay';

                confirmButton.classList.remove('hidden');

                confirmButton.onclick = closeModal;

            }

            modal.classList.remove('hidden');

        }

        function loadDashboard() {

            const dashboardContent = document.getElementById('dashboard-content');

            if (TEST\_SERIES\_DATA.length === 0) {

                dashboardContent.innerHTML = `<p class="text-xl text-gray-500 p-8 bg-white rounded-xl shadow-lg">No tests currently available. Add new tests manually in the code's DEFAULT\_TESTS array.</p>`;

            } else {

                dashboardContent.innerHTML = TEST\_SERIES\_DATA.map(test => {

                    const testTotalQuestions = getTotalQuestions(test);

                    return `

                        <div onclick="window.location.hash='#view=test&test=${test.id}'"

                            class="group bg-white p-8 rounded-3xl shadow-xl transition-all duration-300 border-b-4 border-primary-300 hover:border-primary-600

                                  transform hover:-translate-y-1 hover:shadow-2xl hover:shadow-primary-500/30 cursor-pointer flex flex-col justify-between">

                            <div>

                                <h2 class="text-3xl font-extrabold text-gray-900 mb-2 group-hover:text-primary-600 transition-colors">${test.title}</h2>

                                <p class="text-gray-600 mb-6 text-base">${test.description}</p>

                                <div class="flex flex-wrap items-center space-x-6 text-sm text-gray-500 pt-4 border-t border-gray-100">

                                    <span class="flex items-center font-semibold text-gray-700"><i data-lucide="List" class="w-5 h-5 mr-2 text-primary-500"></i> ${testTotalQuestions} Questions</span>

                                    <span class="flex items-center font-semibold text-gray-700"><i data-lucide="Timer" class="w-5 h-5 mr-2 text-primary-500"></i> ${test.durationMinutes} Minutes</span>

                                </div>

                            </div>

                            <div class="mt-6">

                                <button class="w-full bg-primary-500 text-white py-3 rounded-xl font-bold hover:bg-primary-600 transition-colors shadow-lg shadow-primary-500/40">

                                    Start Now

                                </button>

                            </div>

                        </div>

                    `;

                }).join('');

            }

            createIcons({ icons: { Timer, List } });

        }

        function loadResults() {

            if (!currentTest) {

                window.location.hash = '';

                return;

            }

            let correctCount = 0;

            let attemptedCount = 0;

            const totalQuestions = TOTAL\_QUESTIONS;

            // 1. Calculate Score

            const reviewSections = currentTest.subjects.map((subject, sIndex) => {

                let subjectCorrect = 0;

                const reviewQuestions = subject.questions.map((q, qIndex) => {

                    const key = `${sIndex}-${qIndex}`;

                    const userAnswer = userAnswers[key];

                    const isAnswered = userAnswer !== null;

                    const isCorrect = isAnswered && userAnswer === q.isCorrect;

                    if (isAnswered) attemptedCount++;

                    if (isCorrect) {

                        correctCount++;

                        subjectCorrect++;

                    }

                    return {

                        ...q,

                        userAnswer: userAnswer,

                        isCorrect: isCorrect,

                        correctOptionIndex: q.isCorrect,

                        isAnswered: isAnswered

                    };

                });

                return {

                    name: subject.name,

                    correct: subjectCorrect,

                    total: subject.questions.length,

                    questions: reviewQuestions

                };

            });

            const incorrectCount = attemptedCount - correctCount;

            const unattemptedCount = totalQuestions - attemptedCount;

            const percentage = (correctCount / totalQuestions) \* 100;

            // 2. Render Score Summary

            const summaryHTML = `

                <h1 class="text-4xl font-extrabold text-gray-800 mb-2">${currentTest.title}</h1>

                <p class="text-lg text-gray-600 mb-6">Test Results & Analysis</p>

                <div class="bg-white p-10 rounded-2xl shadow-2xl border-t-8 border-primary-500 mb-10">

                    <div class="flex flex-col sm:flex-row items-center justify-around text-center border-b pb-6 mb-6">

                        <div class="p-4">

                            <div class="text-7xl font-black text-gray-900">${correctCount} / ${totalQuestions}</div>

                            <p class="text-gray-500 text-xl font-semibold mt-2">Questions Correct</p>

                        </div>

                        <div class="p-4">

                            <div class="text-7xl font-black ${percentage >= 60 ? 'text-green-600' : 'text-orange-600'}">${percentage.toFixed(1)}%</div>

                            <p class="text-gray-500 text-xl font-semibold mt-2">Overall Accuracy</p>

                        </div>

                    </div>

                    <div class="mt-4 grid grid-cols-2 md:grid-cols-4 gap-4 text-center">

                        <div class="p-3 bg-green-50 rounded-xl shadow-md"><p class="font-bold text-2xl text-green-700">${correctCount}</p><p class="text-sm text-green-600">Correct</p></div>

                        <div class="p-3 bg-red-50 rounded-xl shadow-md"><p class="font-bold text-2xl text-red-700">${incorrectCount}</p><p class="text-sm text-red-600">Incorrect</p></div>

                        <div class="p-3 bg-blue-50 rounded-xl shadow-md"><p class="font-bold text-2xl text-blue-700">${attemptedCount}</p><p class="text-sm text-blue-600">Attempted</p></div>

                        <div class="p-3 bg-yellow-50 rounded-xl shadow-md"><p class="font-bold text-2xl text-yellow-700">${unattemptedCount}</p><p class="text-sm text-yellow-600">Skipped</p></div>

                    </div>

                </div>

            `;

            document.getElementById('results-summary').innerHTML = summaryHTML;

            // 3. Render Detailed Review

            const reviewHTML = reviewSections.map(subject => {

                const subjectBadge = `<span class="text-sm font-semibold bg-primary-100 text-primary-700 px-3 py-1 rounded-full">${subject.correct}/${subject.total} Correct</span>`;

                const questionsHTML = subject.questions.map((q, qIndex) => {

                    const statusClass = q.isCorrect ? 'border-green-500 bg-green-50' : (q.isAnswered ? 'border-red-500 bg-red-50' : 'border-gray-300 bg-white');

                    const statusIcon = q.isCorrect ? 'CheckCircle' : (q.isAnswered ? 'XCircle' : 'List');

                    const statusText = q.isCorrect ? 'Correct Answer' : (q.isAnswered ? 'Incorrect Answer' : 'Unattempted');

                    return `

                        <div class="p-5 rounded-xl border-l-4 ${statusClass} mb-4 shadow-sm">

                            <p class="text-lg font-semibold text-gray-800 mb-3 flex items-center">

                                <i data-lucide="${statusIcon}" class="w-5 h-5 mr-2 ${q.isCorrect ? 'text-green-600' : (q.isAnswered ? 'text-red-600' : 'text-gray-500')}"></i>

                                Question ${qIndex + 1}: ${q.q}

                            </p>

                            <p class="text-sm font-medium text-gray-600 mb-2">${statusText}</p>

                            <ul class="space-y-2 text-sm mt-3">

                                ${q.options.map((option, index) => {

                                    let optionClass = 'p-2 rounded';

                                    if (index === q.correctOptionIndex) {

                                        optionClass += ' bg-primary-200 font-bold text-primary-800 ring-2 ring-primary-500'; // Correct Answer Highlight

                                    } else if (index === q.userAnswer && !q.isCorrect) {

                                        optionClass += ' bg-red-300 font-bold text-red-900'; // User's Wrong Answer

                                    } else {

                                        optionClass += ' bg-gray-100 text-gray-700';

                                    }

                                    return `<li class="${optionClass}">${String.fromCharCode(65 + index)}. ${option}</li>`;

                                }).join('')}

                            </ul>

                        </div>

                    `;

                }).join('');

                return `

                    <div class="mb-8 p-6 bg-white rounded-xl shadow-lg">

                        <h3 class="text-2xl font-bold text-gray-800 mb-4 flex justify-between items-center">${subject.name} ${subjectBadge}</h3>

                        ${questionsHTML}

                    </div>

                `;

            }).join('');

            document.getElementById('results-review').innerHTML = reviewHTML;

            createIcons({ icons: { CheckCircle, XCircle, List } });

        }

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

            // Initial render based on URL hash

            handleRoute();

            // Listen for hash changes to enable navigation

            window.addEventListener('hashchange', handleRoute);

        });

    </script>

    <!--

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    \* MAIN HTML STRUCTURE

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    -->

    <!-- Minimal Header for Navigation Context (Appears on Test/Results screens) -->

    <header id="context-header" class="bg-white shadow-md sticky top-0 z-10 hidden">

        <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-3 flex justify-between items-center">

            <a href="#" class="text-2xl font-black text-gray-900"><span class="text-primary-500">Daily</span>Test</a>

            <button onclick="window.location.hash=''" class="text-sm font-semibold text-primary-600 hover:text-primary-700 transition-colors flex items-center">

                <i data-lucide="ArrowLeft" class="w-4 h-4 mr-1 inline-block"></i> Back to Tests

            </button>

        </div>

    </header>

    <!-- 1. DASHBOARD VIEW (Default View) -->

    <main id="dashboard-view" class="max-w-6xl mx-auto p-4 sm:p-6 lg:p-8">

        <!-- Hero Section - Integrated Logo -->

        <div class="text-center mb-12 pt-12 pb-12 px-6 bg-white rounded-2xl shadow-2xl border-b-4 border-primary-500">

            <!-- New Impressive Logo Design -->

            <a href="#" class="text-7xl font-black text-gray-900 tracking-tight mb-8 block transition-colors leading-none sm:text-8xl">

                <span class="text-primary-500">Daily</span>Test

            </a>

            <h1 class="text-4xl font-extrabold text-gray-900 mb-4 tracking-tight">Your Daily Test Prep Starts Here</h1>

            <p class="text-xl text-gray-600 font-light max-w-3xl mx-auto">

                Select a test series below to begin your timed assessment.

                Focus on accuracy and time management to track your progress.

            </p>

        </div>

        <h2 class="text-3xl font-bold text-gray-800 mb-6 border-b-2 pb-2">Available Test Series</h2>

        <p class="text-gray-600 mb-8">New tests are added frequently. Check back daily!</p>

        <div id="dashboard-content" class="grid grid-cols-1 md:grid-cols-2 gap-8">

            <!-- Test Cards will be rendered here by JS -->

        </div>

    </main>

    <!-- 2. TEST ACTIVE VIEW -->

    <main id="test-view" class="max-w-7xl mx-auto p-4 sm:p-6 lg:p-8 hidden lg:grid lg:grid-cols-12 lg:gap-8">

        <!-- A. Question & Navigation Area (Col 1-9) -->

        <div class="lg:col-span-9">

            <div class="bg-white p-6 rounded-xl shadow-lg border border-gray-100">

                <div class="flex flex-col sm:flex-row justify-between items-start sm:items-center mb-6 border-b pb-4">

                    <h1 id="question-header" class="text-2xl font-bold text-gray-800"></h1>

                    <div id="timer-display" class="mt-4 sm:mt-0 px-4 py-2 bg-white text-primary-600 font-bold rounded-lg shadow-inner flex items-center text-lg transition-colors border-2 border-primary-100">

                        <i data-lucide="Timer" class="w-4 h-4 mr-2"></i> 00:00

                    </div>

                </div>

                <div id="question-area">

                    <!-- Question text and options rendered here -->

                </div>

                <!-- Navigation Buttons -->

                <div id="question-navigation" class="mt-8 pt-4 border-t flex justify-between">

                    <!-- Buttons rendered here -->

                </div>

            </div>

        </div>

        <!-- B. Subject/Question Panel (Col 10-12) -->

        <div class="lg:col-span-3">

            <div id="subject-panel" class="bg-gray-100 p-4 rounded-xl shadow-inner h-full">

                <!-- Subject navigation and question grid rendered here -->

            </div>

        </div>

    </main>

    <!-- 3. RESULTS VIEW -->

    <main id="results-view" class="max-w-4xl mx-auto p-4 sm:p-6 lg:p-8">

        <div id="results-summary">

            <!-- Summary stats rendered here -->

        </div>

        <h2 class="text-3xl font-bold text-gray-800 mt-10 mb-6 border-b pb-2">Detailed Solution Review</h2>

        <div id="results-review">

            <!-- Question-by-question review rendered here -->

        </div>

        <div class="mt-8 text-center">

            <button onclick="window.location.hash=''"

                class="px-8 py-4 bg-primary-500 text-white rounded-xl font-bold text-lg shadow-xl shadow-primary-500/30 hover:bg-primary-600 transition-all transform hover:scale-[1.02]">

                Return to Dashboard

            </button>

        </div>

    </main>

    <!-- Custom Modal for Alerts and Confirmation (Avoids alert/confirm) -->

    <div id="custom-modal" class="hidden fixed inset-0 bg-gray-900 bg-opacity-75 z-50 flex items-center justify-center p-4">

        <div class="bg-white rounded-xl shadow-2xl max-w-sm w-full p-6 transform transition-all">

            <h3 id="modal-title" class="text-xl font-bold text-gray-800 mb-3 border-b pb-2">Confirm Action</h3>

            <p id="modal-message" class="text-gray-600 mb-6"></p>

            <div class="flex justify-end space-x-3">

                <button id="modal-cancel" class="px-4 py-2 text-gray-700 border border-gray-300 rounded-lg hover:bg-gray-100 transition-colors">

                    Cancel

                </button>

                <button id="modal-confirm" class="px-4 py-2 bg-primary-500 text-white rounded-lg font-semibold hover:bg-primary-600 transition-colors">

                    Confirm

                </button>

            </div>

        </div>

    </div>

</body>

</html>