```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PeacePlay - Your Mindful Companion</title>
    <!-- Use Inter font for a clean, modern look -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap"
rel="stylesheet">
    <!-- Load Tailwind CSS -->
    <script src="https://cdn.tailwindcss.com"></script>
    <style>
        body {
            font-family: 'Inter', sans-serif;
            color: #1f2937;
            background: linear-gradient(180deg, #f0f9ff 0%, #dbeafe 100%);
            background-size: cover;
            background-position: center;
            background-attachment: fixed;
        }

        /* Keyframes for the breathing circle animation */
        .breathing-circle {
            width: 200px;
            height: 200px;
            background-color: #60a5fa; /* Medium blue */
            border-radius: 9999px;
            animation: breathe 8s infinite;
            box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);
        }
        @keyframes breathe {
            0%, 100% { transform: scale(1); }
            50% { transform: scale(1.2); background-color: #bae6fd; /* Light sky blue on inhale */ }
        }

        /* bubble pop game's floating bubbles */
        .bubble {
            position: absolute;
            bottom: 0;
            width: 40px;
            height: 40px;
            background-color: #60a5fa;
            border-radius: 9999px;
            cursor: pointer;
            opacity: 0.7;
```

```
            animation: float-up linear infinite;
        }
        @keyframes float-up {
            from { transform: translateY(0); opacity: 0.8; }
            to { transform: translateY(-100vh); opacity: 0; }
        }

        /* Animation for the text that appears after a bubble is popped */
        .pop-text-animation {
            animation: fade-out 1.5s forwards;
        }
        @keyframes fade-out {
            0% { opacity: 1; transform: translateY(0); }
            100% { opacity: 0; transform: translateY(-20px); }
        }

        .message-box {
            position: fixed;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
            background-color: #fff;
            padding: 2rem;
            border-radius: 1rem;
            box-shadow: 0 10px 20px rgba(0,0,0,0.2);
            z-index: 100;
            display: none; /* Hidden by default */
            text-align: center;
        }

        .message-overlay {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background-color: rgba(0,0,0,0.5);
            z-index: 90;
            display: none; /* Hidden by default */
        }
    </style>
</head>
<body class="p-4 sm:p-8 md:p-12 lg:p-16 flex flex-col items-center min-h-screen">
    <!-- Message Box and Overlay -->
    <div id="message-overlay" class="message-overlay"></div>
    <div id="message-box" class="message-box">
        <p id="message-text" class="text-lg font-semibold mb-4"></p>
        <button id="message-ok-btn" class="px-4 py-2 bg-blue-500 text-white rounded-full
```

```
hover:bg-blue-600 transition-colors">OK</button>
    </div>

    <!-- Main Content -->
    <div id="main-container" class="max-w-7xl w-full bg-blue-50 rounded-3xl shadow-2xl p-6
sm:p-10 md:p-14 border-b-8 border-blue-500">
        <header class="text-center mb-10">
            <h1 class="text-4xl sm:text-5xl lg:text-6xl font-extrabold text-blue-700 tracking-tight">
                PeacePlay
            </h1>
            <p class="mt-2 text-lg sm:text-xl text-blue-400 max-w-2xl mx-auto">
                Your peaceful digital space for stress relief and emotional well-being.
            </p>
        </header>

        <!-- Main Layout with two vertical sections -->
        <main class="flex flex-col gap-8">
            <!-- Games Section -->
            <section id="games-section" class="bg-cyan-50 p-6 rounded-2xl shadow-inner
border-t-4 border-blue-400">
                <h2 class="text-2xl sm:text-3xl font-bold text-blue-800 mb-6">Interactive
Mini-Games</h2>
                <p class="text-gray-600 mb-6">Choose a game to take a mindful break and relax
your mind.</p>
                <!-- Game cards -->
                <div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-4">
                    <div id="bubble-pop" class="game-card bg-cyan-200 p-6 rounded-xl text-center
cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                        <span class="text-3xl">🫧</span>
                        <h3 class="mt-2 font-semibold text-cyan-800">Bubble Pop</h3>
                        <p class="text-sm text-cyan-600">A simple, satisfying game.</p>
                    </div>
                    <div id="breathe-with-me" class="game-card bg-blue-200 p-6 rounded-xl
text-center cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                        <span class="text-3xl">🧘♀️</span>
                        <h3 class="mt-2 font-semibold text-blue-800">Breathe with Me</h3>
                        <p class="text-sm text-blue-600">Follow the guide to calm your breath.</p>
                    </div>
                    <div id="affirmation-flip" class="game-card bg-sky-200 p-6 rounded-xl text-center
cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                        <span class="text-3xl">✨</span>
                        <h3 class="mt-2 font-semibold text-sky-800">Affirmation Flip</h3>
                        <p class="text-sm text-sky-600">Turn over cards for positive thoughts.</p>
                    </div>
                    <div id="tic-tac-toe" class="game-card bg-teal-200 p-6 rounded-xl text-center
cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                        <span class="text-3xl">❌⭕</span>
                        <h3 class="mt-2 font-semibold text-teal-800">Tic Tac Toe</h3>
```

```
                    <p class="text-sm text-teal-600">The classic game of X's and O's.</p>
                </div>
                <div id="soundscape-generator" class="game-card bg-indigo-200 p-6 rounded-xl
text-center cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                    <span class="text-3xl">🎶</span>
                    <h3 class="mt-2 font-semibold text-indigo-800">Soothing Tones</h3>
                    <p class="text-sm text-indigo-600">Mix calming sounds to relax.</p>
                </div>
                <div id="thought-journal" class="game-card bg-cyan-300 p-6 rounded-xl
text-center cursor-pointer transition-transform transform hover:scale-105 hover:shadow-lg">
                    <span class="text-3xl">📝</span>
                    <h3 class="mt-2 font-semibold text-cyan-800">Thought Journal</h3>
                    <p class="text-sm text-cyan-600">Write down your feelings.</p>
                </div>
            </div>

            <!-- Game Display Area -->
            <div id="game-display" class="mt-8 bg-blue-50 p-6 rounded-xl flex items-center
justify-center text-center h-96 border border-blue-200 relative overflow-hidden">
                <p class="text-gray-400 text-lg">Select a game to start playing.</p>
            </div>
        </section>

        <!-- Chatbot Section -->
        <section id="chatbot-section" class="bg-blue-100 p-6 rounded-2xl shadow-inner
border-t-4 border-blue-400 flex flex-col h-full">
            <h2 class="text-2xl sm:text-3xl font-bold text-blue-800 mb-6">PeacePlay
Chatbot</h2>
            <p class="text-gray-600 mb-4">I'm here to listen. How are you feeling today?</p>

            <!-- Chatbox -->
            <div id="chat-box" class="flex-1 overflow-y-auto p-4 bg-white rounded-lg border
border-blue-200 mb-4 flex flex-col gap-3 shadow-inner">
                <!-- Initial bot message -->
                <div class="flex justify-start">
                    <div class="bg-blue-100 text-gray-800 p-3 rounded-2xl rounded-bl-none
max-w-xs">
                        Hello there! I'm here to offer some peace and calm.
                    </div>
                </div>
            </div>

            <!-- Chat input form -->
            <form id="chat-form" class="flex gap-2">
                <input type="text" id="user-input" placeholder="Type your message..."
class="flex-1 p-3 rounded-full border border-gray-300 focus:outline-none
focus:border-blue-500 transition-all">
                <button type="submit" class="bg-blue-500 text-white p-3 rounded-full
```

```
font-semibold hover:bg-blue-600 transition-colors shadow-md">
                    Send
                </button>
            </form>
        </section>
    </main>
</div>

<!-- JavaScript Logic -->
<script type="module">
    // --- Firebase Imports and Setup ---
    import { initializeApp } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";
    import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged }
from "https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";
    import { getFirestore, collection, onSnapshot, addDoc, deleteDoc, doc, getDocs,
writeBatch } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";
    import { setLogLevel } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";

    // setLogLevel('Debug'); // Uncomment to see Firebase logs

    let db;
    let auth;
    let userId;
    let isAuthReady = false;

    // Load Tone.js for the soundscape generator and audio effects
    const toneScript = document.createElement('script');
    toneScript.src = "https://cdn.jsdelivr.net/npm/tone@14.7.58/build/Tone.min.js";
    document.head.appendChild(toneScript);

    // Get DOM elements for interaction
    const chatForm = document.getElementById('chat-form');
    const userInput = document.getElementById('user-input');
    const chatBox = document.getElementById('chat-box');
    const gameDisplay = document.getElementById('game-display');
    const gameCards = document.querySelectorAll('.game-card');
    const messageBox = document.getElementById('message-box');
    const messageText = document.getElementById('message-text');
    const messageOkBtn = document.getElementById('message-ok-btn');
    const messageOverlay = document.getElementById('message-overlay');

    // Global variables to control game states and animations
    let bubbleInterval = null;
    let currentSound = null;
    let animationFrameId = null;
    let journalUnsubscribe = null; // To hold the Firestore onSnapshot listener
```

```
        // --- Firebase Initialization and Authentication ---
        const firebaseConfig = JSON.parse(typeof __firebase_config !== 'undefined' ?
__firebase_config : '{}');
        const initialAuthToken = typeof __initial_auth_token !== 'undefined' ? __initial_auth_token
: null;

        if (Object.keys(firebaseConfig).length > 0) {
            const app = initializeApp(firebaseConfig);
            db = getFirestore(app);
            auth = getAuth(app);

            onAuthStateChanged(auth, async (user) => {
                if (user) {
                    userId = user.uid;
                } else {
                    // Sign in anonymously if no user is found (which happens if initialAuthToken is
missing/failed)
                    try {
                        await signInAnonymously(auth);
                        userId = auth.currentUser.uid;
                    } catch (error) {
                        console.error("Anonymous sign in failed:", error);
                        showMessage("Error signing in. Journal functionality disabled.");
                        userId = 'guest-user';
                    }
                }
                isAuthReady = true;
                console.log("Firebase Auth Ready. User ID:", userId);
            });

            // Attempt to sign in with custom token
            if (initialAuthToken) {
                signInWithCustomToken(auth, initialAuthToken).catch(error => {
                    console.error("Custom token sign in failed:", error);
                    // onAuthStateChanged will handle anonymous fallback
                });
            }
        } else {
            console.error("Firebase config is missing. Journal functionality disabled.");
            isAuthReady = true; // Still allow app to run without journal
        }

        // Firestore Helper
        const getJournalCollectionRef = () => {
            const appId = typeof __app_id !== 'undefined' ? __app_id : 'default-app-id';
            if (db && userId) {
                return collection(db, artifacts/${appId}/users/${userId}/journal_entries);
            }
```

```
        return null;
    };


    // Jokes (moved into a const)
    const jokes = [
        "I’m on a seafood diet — I see food and I eat it.",
        "My wallet is like an onion, opening it makes me cry.",
        "If Monday had a face, I would punch it",
        "My boss told me to have a good day… so I went home.",
        "My boss told me to start every presentation with a joke… so I used his salary as the opening slide.",
        "My boss is like clouds. When he disappears, it’s a beautiful day.",
        "I told my computer I needed a break… now it won’t stop sending me KitKats.",
        "What do you call a magic dog? A labracadabrador..",
        "Why did the bicycle fall over? Because it was two tired!",
        "I told my mirror to tell me a joke.I don’t know why…but it just showed me my face.'",
        "I love walking in fog… it’s the only time I can disappear without people noticing.",
        "Relative at a wedding: You’re next!... Me at a funeral: Same to you!",
        "You must be tired..... running from responsibility all these years.",
        "Relative: Can I do something for you? me : block me...",
        "Boss: Where do you see yourself in 5 years? Me: Not in this office",
        "Relatives: What makes you sad? Me : You.",
        "Boss: You’re wasting time on your phone...You: “Better than wasting time listening to you..",
        "Why was the math book sad? Because it had too many problems.",
        "What’s the best thing about Switzerland? I don’t know, but the flag is a big plus.",
        "Did you hear about the two guys who stole a calendar? They each got six months."
    ];

    // --- Message Box Function ---
    const showMessage = (text) => {
        messageText.textContent = text;
        messageBox.style.display = 'block';
        messageOverlay.style.display = 'block';
    };

    messageOkBtn.addEventListener('click', () => {
        messageBox.style.display = 'none';
        messageOverlay.style.display = 'none';
    });

    /**
     * Clears all running animations, audio, and Firestore listeners, ensuring a clean state when switching games.
     */
    const stopAllSounds = () => {
        if (currentSound) {
```

```
            // Tone.js Sequences/Loops need to be stopped and disposed
            if (currentSound.stop) currentSound.stop();
            if (currentSound.dispose) currentSound.dispose();
            currentSound = null;
            // Stop Tone.Transport only if nothing else is scheduled
            if (Tone.Transport.state === "started" || Tone.Transport.state === "paused") {
                Tone.Transport.stop();
                Tone.Transport.cancel();
            }
        }
        if (bubbleInterval) {
            clearInterval(bubbleInterval);
            bubbleInterval = null;
        }
        if (animationFrameId) {
            cancelAnimationFrame(animationFrameId);
            animationFrameId = null;
        }
        if (journalUnsubscribe) {
            journalUnsubscribe(); // Detach the Firestore listener
            journalUnsubscribe = null;
        }
    };

    // --- Chatbot Logic (unchanged) ---
    /**
     * Provides a chatbot response based on a direct match or a keyword-based emotion
analysis.
     * @param {string} userMessage The user's message.
     * @returns {object} An object containing the chatbot's message and a game
recommendation ID.
     */
    const getChatResponse = (userMessage) => {
        const lowerMessage = userMessage.toLowerCase().trim();

        // Handle common greetings and questions with flexibility
        if (lowerMessage.includes("how are you")) {
            return { message: "I'm doing well, thank you. How about you?", recommendation:
"none" };
        }
        if (lowerMessage.includes("what can you do")) {
            return { message: "I'm here to offer a little calm and peace. You can play one of our
mindful games, or just talk to me about how you're feeling.", recommendation: "none" };
        }
        if (lowerMessage.includes("hello") || lowerMessage.includes("hi") ||
lowerMessage.includes("hey")) {
            return { message: "Hello there! How can I help you feel more at ease?",
recommendation: "none" };
```

```
        }
        if (lowerMessage.includes("i'm fine") || lowerMessage.includes("i am fine") ||
lowerMessage.includes("i'm good") || lowerMessage.includes("i am good")) {
            return { message: "That's great! It's wonderful to hear you're in a good place.",
recommendation: "none" };
        }
        if (lowerMessage.includes("thanks") || lowerMessage.includes("thank you")) {
            return { message: "You're welcome! I'm here to help.", recommendation: "none" };
        }
        if (lowerMessage.includes("i can't sleep") || lowerMessage.includes("i cannot sleep")) {
            return { message: "A calm mind can lead to a good night's rest. You could try the
soundscape generator to find some soothing sounds to help you relax.", recommendation:
"soundscape-generator" };
        }
        if (lowerMessage.includes("i need to vent")) {
            return { message: "I'm here to listen without judgment. You can write down your
thoughts in the thought journal. It can be a powerful way to process your feelings.",
recommendation: "thought-journal" };
        }
        if (lowerMessage.includes("what is mindfulness?")) {
            return { message: "Mindfulness is the practice of being present and aware of your
thoughts and feelings without getting caught up in them. Our games are designed to help you
with that.", recommendation: "none" };
        }
        if (lowerMessage.includes("bye")) {
            return { message: "Bye! Have a nice day.", recommendation: "none" };
        }
        if (lowerMessage.includes("tell me a joke")) {
            const randomJoke = jokes[Math.floor(Math.random() * jokes.length)];
            return { message: randomJoke, recommendation: "bubble-pop" };
        }
        if (lowerMessage.includes("i am not sad") || lowerMessage.includes("i'm not sad")) {
             return { message: "That's great! It sounds like you are feeling positive. I'm glad to
hear that!", recommendation: "none" };
        }

        // More general emotion-based responses
        if (lowerMessage.includes("i am not fine") || lowerMessage.includes("i'm not fine") ||
lowerMessage.includes("not happy") || lowerMessage.includes("not good")) {
             return { message: "I'm sorry to hear that. It's okay to feel that way. Perhaps a game
can offer a small distraction.", recommendation: "bubble-pop" };
        }

        // Fallback to the original emotion-based logic for more complex sentences
        const emotion = analyzeEmotion(lowerMessage);
        switch (emotion) {
            case 'anxious':
                return {
```

```
                message: "It sounds like you're feeling anxious. Taking a moment to breathe can really help.",
                recommendation: "breathe-with-me"
            };
        case 'sad':
            return {
                message: "I'm sorry to hear that. How about a little boost of positivity? Let's try some affirmations.",
                recommendation: "affirmation-flip"
            };
        case 'tired':
            return {
                message: "It's okay to feel drained. A slow, gentle activity can help you recharge. How about a quick game of Tic Tac Toe?",
                recommendation: "tic-tac-toe"
            };
        case 'happy':
            return {
                message: "That's wonderful to hear! I'm glad you're feeling so positive. Keep that energy going!",
                recommendation: "none"
            };
        default:
            return {
                message: "As a peace-focused companion, I'm here to listen. I'm not able to answer all questions, but I'm here for you to talk to. Or, you can just enjoy one of our calming games.",
                recommendation: "none"
            };
    }
};

// This function is still needed for the fallback logic
const analyzeEmotion = (text) => {
    const lowerText = text.toLowerCase();

    // Negative and sadness keywords
    const sadKeywords = ['sad', 'down', 'unhappy', 'depressed', 'lonely', 'not happy', 'not good', 'not feeling good', 'bad', 'terrible', 'awful'];
    if (sadKeywords.some(keyword => lowerText.includes(keyword))) {
        return 'sad';
    }

    // Anxiety and stress keywords
    const anxiousKeywords = ['stressed', 'anxious', 'overwhelmed', 'worry', 'nervous'];
    if (anxiousKeywords.some(keyword => lowerText.includes(keyword))) {
        return 'anxious';
    }
```

```
        // Tiredness keywords
        const tiredKeywords = ['tired', 'exhausted', 'drained', 'sleepy'];
        if (tiredKeywords.some(keyword => lowerText.includes(keyword))) {
            return 'tired';
        }

        // Happiness and positive keywords
        const happyKeywords = ['happy', 'good', 'great', 'excited', 'wonderful', 'fine', 'amazing'];
        if (happyKeywords.some(keyword => lowerText.includes(keyword))) {
            return 'happy';
        }

        return 'neutral';
    };

    // --- Chat Interaction Handler (unchanged) ---
    chatForm.addEventListener('submit', (e) => {
        e.preventDefault();
        const userMessage = userInput.value.trim();
        if (userMessage === '') return;

        // Display user message in the chat box
        chatBox.innerHTML += `
            <div class="flex justify-end">
                <div class="bg-blue-300 text-gray-800 p-3 rounded-2xl rounded-br-none
max-w-xs">
                    ${userMessage}
                </div>
            </div>
        `;
        userInput.value = '';
        chatBox.scrollTop = chatBox.scrollHeight;

        // Get bot's response and recommend a game
        const botResponse = getChatResponse(userMessage);

        setTimeout(() => {
            chatBox.innerHTML += `
                <div class="flex justify-start">
                    <div class="bg-blue-100 text-gray-800 p-3 rounded-2xl rounded-bl-none
max-w-xs">
                        ${botResponse.message}
                    </div>
                </div>
            `;
            chatBox.scrollTop = chatBox.scrollHeight;
            if (botResponse.recommendation !== "none") {
```

```
                displayGame(botResponse.recommendation);
            }
        }, 1000); // Simulate a slight delay
    });

    // --- Game Display Logic ---
    /**
     * Clears the game display and shows the content for the selected game.
     * @param {string} gameId The ID of the game to display.
     */
    const displayGame = (gameId) => {
        // Clear any existing game content and stop sounds/listeners
        gameDisplay.innerHTML = '';
        stopAllSounds();

        let content = '';
        switch (gameId) {
            case 'bubble-pop':
                const positiveTexts = [
                    "Be Happy", "Keep smiling", "Stay Strong", "You got this", "Take a breath", "Be
Kind",
                    "You are capable of amazing things.", "Embrace the journey.", "Challenges are
opportunities.",
                    "Your potential is limitless.", "Be kind to yourself.", "Find joy in the little things.",
                    "Every moment is a fresh start.", "You are a masterpiece.", "Your efforts will pay
off."
                ];
                // Ensure Tone is started before creating synth
                Tone.start().then(() => {
                    const popSynth = new Tone.MembraneSynth().toDestination();

                    content = `<div class="w-full h-full relative flex flex-col items-center
justify-center">
                                <p class="text-gray-500 text-sm absolute top-4 left-4">Pop the
bubbles!</p>
                            </div>`;
                    gameDisplay.innerHTML = content;

                    const createBubble = () => {
                        const bubble = document.createElement('div');
                        bubble.classList.add('bubble');
                        const size = Math.random() * 20 + 30;
                        const color = ['#60a5fa', '#bae6fd', '#22c55e'][Math.floor(Math.random() * 3)];
                        bubble.style.width = ${size}px;
                        bubble.style.height = ${size}px;
                        bubble.style.backgroundColor = color;
                        const leftPosition = Math.random() * 95;
                        const duration = Math.random() * 8 + 8;
```

```
                    bubble.style.left = ${leftPosition}%;
                    bubble.style.animationDuration = ${duration}s;

                    bubble.addEventListener('click', () => {
                        popSynth.triggerAttackRelease("C4", "8n");
                        bubble.remove();
                        const randomText = positiveTexts[Math.floor(Math.random() *
positiveTexts.length)];
                        const popText = document.createElement('span');
                        popText.textContent = randomText;
                        popText.classList.add('pop-text-animation', 'absolute', 'font-bold', 'text-sm',
'text-cyan-700');
                        popText.style.left = ${leftPosition}%;
                        popText.style.bottom = ${(Math.random() * 30 + 10)}%;
                        gameDisplay.appendChild(popText);
                        setTimeout(() => popText.remove(), 1500);
                    });
                    gameDisplay.appendChild(bubble);
                };

                bubbleInterval = setInterval(createBubble, 1000);
            }).catch(e => console.error("Tone.js failed to start:", e));

            break;

        case 'breathe-with-me':
            content = `<div class="w-full h-full flex flex-col items-center justify-center
space-y-4">
                    <div class="breathing-circle"></div>
                    <p class="text-xl text-blue-700 font-medium">Breathe in...</p>
                    <p class="text-sm text-gray-500">Inhale as it expands, exhale as it
shrinks.</p>
                </div>`;
            gameDisplay.innerHTML = content;
            break;

        case 'affirmation-flip':
            const affirmations = [
                "You are stronger than you think.", "Don't give up. You are the best.", "Believe in
yourself. You can do this.", "Every day is a fresh start.", "Your efforts will pay off.", "Stay
positive and great things will happen.",
                "You are capable of amazing things.", "Embrace the journey, not just the
destination.", "Challenges are opportunities in disguise.", "Your potential is limitless.",
                "Keep going, you're doing great.", "You are worthy of success.", "Small steps
lead to big changes.", "Trust yourself. You've got this.", "Progress, not perfection, matters
most.",
                "You inspire others without knowing it.", "Stay strong, your breakthrough is
near.", "Believe in the power of your dreams.", "You have what it takes to shine.", "Keep
```

moving forward with courage.",
"Every setback is a setup for a comeback.", "Your smile makes the world brighter.", "Hard work always pays off.", "Your energy attracts positivity.", "You are more powerful than fear.", "Stay hopeful, better days are coming.",
"Your kindness makes a difference.", "Be proud of how far you've come.", "You are a light to others.", "Confidence looks good on you.",
"Your heart knows the way.", "You can handle anything that comes your way.", "Keep believing, miracles happen every day.", "You bring value to the world.", "You are growing stronger each day.",
"Your dreams are within reach.", "You are a fighter, not a quitter.", "Stay patient, everything will align.", "You are braver than you feel.", "Your story is still being written.",
"You make progress even on tough days.", "You are more than enough.", "Your mindset creates your reality.", "Every effort counts, keep going.", "You are a source of inspiration.",
"Stay calm, stay focused, stay strong.", "You deserve happiness and peace.", "You have the courage to succeed.", "Your hard work builds your future.", "You are unstoppable when you believe.",
"Every challenge makes you wiser.", "You have endless possibilities ahead.", "Your determination is your superpower.", "Stay grateful, stay grounded, stay great.", "You are stronger than yesterday.",
"Your positivity creates miracles.", "You are capable of incredible growth.", "Stay hopeful, the best is yet to come.", "You inspire by simply being yourself.", "Your persistence will be rewarded.",
"You can turn struggles into strengths.", "You are destined for greatness.", "Your inner strength is unshakable.", "Stay consistent, success is near.", "You are worthy of love and respect.",
"You are learning and growing daily.", "Your courage opens new doors.", "You bring joy wherever you go.", "You are in charge of your happiness.", "Your possibilities are endless.",
"You can achieve anything you set your mind to.", "Stay motivated, success is closer than you think.", "You are a masterpiece in progress.", "Your resilience is admirable.",
"You radiate strength and confidence.", "You make the world a better place.", "You are a magnet for positivity.", "Stay kind, it always comes back.", "You are creating your own destiny.", "Your spirit is unbreakable.",
"You are doing better than you realize.", "Your passion fuels your success.", "You are enough just as you are.", "Stay humble, stay hungry, stay hopeful.", "You are worthy of every blessing.", "Your future is full of promise.",
"You are stronger with every challenge.", "Your mind is powerful beyond measure.", "You are closer to your goals every day.", "You are destined to achieve greatness.",

"You make the impossible possible.", "You are unique and valuable.", "Your efforts create ripples of success.", "You are stronger than any obstacle.", "Your journey is shaping your greatness.",
"You have the power to rewrite your story.", "You are a warrior of light and hope.", "Stay focused, success loves discipline.", "You are capable of turning dreams into reality.", "Your best is yet to come."
];
const shuffleArray = (array) => {

```
                for (let i = array.length - 1; i > 0; i--) {
                    const j = Math.floor(Math.random() * (i + 1));
                    [array[i], array[j]] = [array[j], array[i]];
                }
                return array;
            };
            content = `<div class="flex flex-col items-center space-y-4 w-full h-full">
                        <p class="text-lg text-gray-700">Click a card for an affirmation.</p>
                        <div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-4 w-full p-4
items-center justify-center flex-1">
                            <div class="affirmation-card-item w-full h-40 bg-sky-300 rounded-xl
shadow-lg flex items-center justify-center cursor-pointer transform transition-transform
hover:scale-105">
                                <p class="text-lg font-bold text-center p-4 text-black">Click to
reveal</p>
                            </div>
                            <div class="affirmation-card-item w-full h-40 bg-blue-300 rounded-xl
shadow-lg flex items-center justify-center cursor-pointer transform transition-transform
hover:scale-105">
                                <p class="text-lg font-bold text-center p-4 text-black">Click to
reveal</p>
                            </div>
                            <div class="affirmation-card-item w-full h-40 bg-cyan-300 rounded-xl
shadow-lg flex items-center justify-center cursor-pointer transform transition-transform
hover:scale-105">
                                <p class="text-lg font-bold text-center p-4 text-black">Click to
reveal</p>
                            </div>
                        </div>
                        <button id="refresh-cards" class="w-full sm:w-auto px-6 py-3 bg-blue-500
text-white rounded-full font-semibold hover:bg-blue-600 transition-colors shadow-md">
                            Refresh
                        </button>
                    </div>`;
            gameDisplay.innerHTML = content;

            // Attach event listeners after the HTML is in the DOM
            const cards = gameDisplay.querySelectorAll('.affirmation-card-item');
            const refreshButton = document.getElementById('refresh-cards');
            let currentAffirmations = shuffleArray([...affirmations]);

            const setupCardListeners = () => {
                cards.forEach((card, index) => {
                    card.onclick = null;
                    card.addEventListener('click', () => {
                        if (card.textContent.trim() === 'Click to reveal') {
                            card.querySelector('p').textContent = currentAffirmations[index];
                        }
```

```
                });
            });
        };

        const resetCards = () => {
            currentAffirmations = shuffleArray([...affirmations]);
            cards.forEach(card => {
                card.querySelector('p').textContent = 'Click to reveal';
            });
            setupCardListeners();
        };

        refreshButton.addEventListener('click', resetCards);
        setupCardListeners();
        break;

    case 'tic-tac-toe':
        content = `<div class="flex flex-col items-center justify-center h-full w-full p-4">
                    <h3 id="tic-tac-toe-status" class="text-xl sm:text-2xl font-bold
text-blue-800 mb-4">Let's Play</h3>
                    <div id="tic-tac-toe-board" class="grid grid-cols-3 gap-2 p-2 rounded-xl
bg-blue-200">
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
font-extrabold text-blue-700"></div>
                        <div class="tic-tac-toe-cell w-[70px] h-[70px] sm:w-[70px] sm:h-[70px]
bg-white rounded-lg cursor-pointer flex items-center justify-center text-4xl sm:text-5xl
```

```
font-extrabold text-blue-700"></div>
                    </div>
                    <button id="play-again-btn" class="mt-4 px-6 py-3 bg-blue-500 text-white
rounded-full font-semibold hover:bg-blue-600 transition-colors shadow-md hidden">Play
Again</button>
                </div>`;
            gameDisplay.innerHTML = content;

            const cells = document.querySelectorAll('.tic-tac-toe-cell');
            const statusText = document.getElementById('tic-tac-toe-status');
            const playAgainBtn = document.getElementById('play-again-btn');

            let board = ['', '', '', '', '', '', '', '', ''];
            let currentPlayer = 'X'; // Human player
            let isGameActive = true;

            const winningConditions = [
                [0, 1, 2], [3, 4, 5], [6, 7, 8],
                [0, 3, 6], [1, 4, 7], [2, 5, 8],
                [0, 4, 8], [2, 4, 6]
            ];

            const handleResultValidation = () => {
                let roundWon = false;
                for (let i = 0; i <= 7; i++) {
                    const winCondition = winningConditions[i];
                    const a = board[winCondition[0]];
                    const b = board[winCondition[1]];
                    const c = board[winCondition[2]];
                    if (a === '' || b === '' || c === '') {
                        continue;
                    }
                    if (a === b && b === c) {
                        roundWon = true;
                        break;
                    }
                }

                if (roundWon) {
                    statusText.innerHTML = You won!;
                    isGameActive = false;
                    playAgainBtn.classList.remove('hidden');
                    return;
                }

                if (!board.includes('')) {
                    statusText.innerHTML = 'Draw!';
                    isGameActive = false;
```

```
            playAgainBtn.classList.remove('hidden');
            return;
        }

        return false;
    };

    const computerMove = () => {
        if (!isGameActive) return;

        // Computer (O) turn
        let bestMove = -1;
        let availableMoves = board.map((cell, index) => cell === '' ? index :
null).filter(val => val !== null);

        // 1. Check for a winning move for the computer
        for (let i = 0; i < winningConditions.length; i++) {
            const [a, b, c] = winningConditions[i];
            const combo = [board[a], board[b], board[c]];
            if (combo.filter(val => val === 'O').length === 2 && combo.includes('')) {
                bestMove = winningConditions[i][combo.indexOf('')];
                break;
            }
        }

        // 2. Block the opponent if they are about to win
        if (bestMove === -1) {
            for (let i = 0; i < winningConditions.length; i++) {
                const [a, b, c] = winningConditions[i];
                const combo = [board[a], board[b], board[c]];
                if (combo.filter(val => val === 'X').length === 2 && combo.includes('')) {
                    bestMove = winningConditions[i][combo.indexOf('')];
                    break;
                }
            }
        }

        // 3. Take the center
        if (bestMove === -1 && board[4] === '') {
            bestMove = 4;
        }

        // 4. Take a corner
        if (bestMove === -1) {
            const corners = [0, 2, 6, 8];
            const availableCorners = availableMoves.filter(move =>
corners.includes(move));
            if (availableCorners.length > 0) {
```

```
                    bestMove = availableCorners[Math.floor(Math.random() *
availableCorners.length)];
                }
            }

            // 5. Take any remaining available spot
            if (bestMove === -1 && availableMoves.length > 0) {
                bestMove = availableMoves[Math.floor(Math.random() *
availableMoves.length)];
            }

            if (bestMove !== undefined && bestMove !== -1) {
                board[bestMove] = 'O';
                cells[bestMove].innerHTML = 'O';
                cells[bestMove].style.color = '#dc2626'; // Red for the computer
                if (!handleResultValidation()) {
                    statusText.innerHTML = "Your turn (X)";
                }
            }
        };

        const handleCellClick = (event) => {
            const clickedCell = event.target;
            const clickedCellIndex = Array.from(cells).indexOf(clickedCell);

            if (board[clickedCellIndex] !== '' || !isGameActive) {
                return;
            }

            // Human player (X) turn
            board[clickedCellIndex] = 'X';
            clickedCell.innerHTML = 'X';
            clickedCell.style.color = '#1d4ed8'; // Blue for the human player

            if (!handleResultValidation()) {
                statusText.innerHTML = "Computer's turn (O)...";
                // Computer takes its turn
                setTimeout(computerMove, 500); // Add a small delay for a more natural feel
            }
        };

        const resetGame = () => {
            board = ['', '', '', '', '', '', '', '', ''];
            isGameActive = true;
            statusText.innerHTML = "Let's Play";
            currentPlayer = 'X';
            playAgainBtn.classList.add('hidden');
            cells.forEach(cell => {
```

```
                cell.innerHTML = '';
                cell.style.color = '';
            });
        };

        cells.forEach(cell => cell.addEventListener('click', handleCellClick));
        playAgainBtn.addEventListener('click', resetGame);
        statusText.innerHTML = "Your turn (X)";
        break;
    case 'soundscape-generator':
        content = `<div class="flex flex-col items-center justify-center h-full space-y-4">
                <p class="text-lg text-gray-700">Choose a calming tune to play.</p>
                <div class="flex flex-wrap justify-center gap-2">
                    <button id="play-piano" class="px-4 py-2 bg-blue-500 text-white rounded-full font-semibold hover:bg-blue-600 transition-colors shadow-md text-sm">Piano</button>
                    <button id="play-synth" class="px-4 py-2 bg-indigo-500 text-white rounded-full font-semibold hover:bg-indigo-600 transition-colors shadow-md text-sm">Synth Pad</button>
                    <button id="play-harp" class="px-4 py-2 bg-purple-500 text-white rounded-full font-semibold hover:bg-purple-600 transition-colors shadow-md text-sm">Gentle Harp</button>
                    <button id="play-gentle-chimes" class="px-4 py-2 bg-teal-500 text-white rounded-full font-semibold hover:bg-teal-600 transition-colors shadow-md text-sm">Gentle Chimes</button>
                    <button id="play-ocean" class="px-4 py-2 bg-sky-500 text-white rounded-full font-semibold hover:bg-sky-600 transition-colors shadow-md text-sm">Ocean Waves</button>
                    <button id="play-drone" class="px-4 py-2 bg-gray-500 text-white rounded-full font-semibold hover:bg-gray-600 transition-colors shadow-md text-sm">Space Drone</button>
                    <button id="play-forest" class="px-4 py-2 bg-green-500 text-white rounded-full font-semibold hover:bg-green-600 transition-colors shadow-md text-sm">Forest Chirps</button>
                    <button id="play-wind" class="px-4 py-2 bg-cyan-500 text-white rounded-full font-semibold hover:bg-cyan-600 transition-colors shadow-md text-sm">Wind Chimes</button>
                    <button id="play-ambient" class="px-4 py-2 bg-rose-500 text-white rounded-full font-semibold hover:bg-rose-600 transition-colors shadow-md text-sm">Ambient Pad</button>
                </div>
                <button id="pause-audio" class="w-full sm:w-auto px-6 py-3 bg-red-400 text-white rounded-full font-semibold cursor-not-allowed shadow-md text-sm" disabled>
                    Pause
                </button>
            </div>`;
        gameDisplay.innerHTML = content;
        const pauseButton = document.getElementById('pause-audio');
```

```
            const playButtons = document.querySelectorAll('#play-piano, #play-synth, #play-harp, #play-gentle-chimes, #play-ocean, #play-drone, #play-forest, #play-wind, #play-ambient');

            playButtons.forEach(button => {
                button.addEventListener('click', async () => {
                    await Tone.start();
                    stopAllSounds(); // Stop current sound and transport

                    // Re-initialize a new Transport state if needed, but Tone.Transport.start() handles it

                    switch(button.id) {
                        case 'play-piano':
                            const synthPiano = new Tone.Synth({ oscillator: { type: "triangle" }, envelope: { attack: 0.01, decay: 0.8, sustain: 0, release: 1 } }).toDestination({ volume: -15 });
                            const notesPiano = ["C4", "E4", "G4", "A4", "G4", "E4"];
                            currentSound = new Tone.Sequence((time, note) => { synthPiano.triggerAttackRelease(note, "8n", time); }, notesPiano, "4n").start(0);
                            break;
                        case 'play-synth':
                            const synthPad = new Tone.PolySynth(Tone.Synth, { oscillator: { type: "sine" }, envelope: { attack: 4, decay: 1, sustain: 0.8, release: 4 } }).toDestination({ volume: -15 });
                            const notesSynth = [["C3", "E3", "G3", "B3"], ["A3", "C4", "E4", "G4"]];
                            currentSound = new Tone.Sequence((time, notes) => { synthPad.triggerAttackRelease(notes, "4n", time); }, notesSynth, "2n").start(0);
                            break;
                        case 'play-harp':
                            const synthHarp = new Tone.PolySynth(Tone.Synth, { oscillator: { type: "triangle" }, envelope: { attack: 0.1, decay: 0.4, sustain: 0, release: 0.5 } }).toDestination({ volume: -15 });
                            const notesHarp = [["C4", "E4", "G4", "C5"], ["A3", "C4", "E4"]];
                            currentSound = new Tone.Sequence((time, notes) => { synthHarp.triggerAttackRelease(notes, "8n", time); }, notesHarp, "4n").start(0);
                            break;
                        case 'play-gentle-chimes':
                            const gentleChimesSynth = new Tone.PolySynth(Tone.Synth, {
                                oscillator: { type: "triangle" },
                                envelope: { attack: 2, decay: 1, sustain: 0, release: 2 }
                            }).toDestination({ volume: -15 });
                            const chimeNotes = ["C4", "E4", "G4", "A4", "G4", "E4"];
                            currentSound = new Tone.Sequence((time, note) => { gentleChimesSynth.triggerAttackRelease(note, "2n", time); }, chimeNotes, "2n").start(0);
                            break;
                        case 'play-ocean':
                            const noise = new Tone.NoiseSynth({ noise: { type: 'pink' }, envelope: { attack: 4, decay: 0, sustain: 1, release: 4 } }).toDestination();
```

```
                        currentSound = new Tone.Loop(time => {
noise.triggerAttackRelease("4n", time); }, "8n").start(0);
                        break;
                    case 'play-drone':
                        const drone = new Tone.DuoSynth({ voice0: { oscillator: { type: 'sine' },
envelope: { attack: 5, release: 5 } }, voice1: { oscillator: { type: 'sine' }, envelope: { attack: 5,
release: 5 } } }).toDestination({ volume: -20 });
                        const droneNotes = [["C2", "G2"], ["F2", "C3"]];
                        currentSound = new Tone.Sequence((time, notes) => {
drone.triggerAttackRelease(notes, "8n", time); }, droneNotes, "8n").start(0);
                        break;
                    case 'play-forest':
                        const forestSynth = new Tone.MembraneSynth({ pitchDecay: 0.05,
octave: 2, oscillator: { type: 'sine' } }).toDestination();
                        const forestSeq = new Tone.Sequence((time, note) => {
forestSynth.triggerAttackRelease(note, "4n", time); }, ["C3", ["E3", "G3"], "C3"], "1n");
                        currentSound = new Tone.Loop(() => { forestSeq.start(Tone.now() +
Math.random() * 2); }, "4n").start(0);
                        break;
                    case 'play-wind':
                        const windChimes = new
Tone.PolySynth(Tone.PluckSynth).toDestination();
                        const windNotes = ["C5", "G4", "A4", "E4"];
                        currentSound = new Tone.Loop(time => {
windChimes.triggerAttack(windNotes[Math.floor(Math.random() * windNotes.length)], time); },
"2n").start(0);
                        break;
                    case 'play-ambient':
                        const ambientPad = new Tone.PolySynth(Tone.Synth, { oscillator: {
type: 'sawtooth' }, envelope: { attack: 2, decay: 2, sustain: 0.5, release: 4 } }).toDestination();
                        const ambientNotes = [["C3", "E3", "G3"], ["A3", "C4", "E4"]];
                        currentSound = new Tone.Sequence((time, notes) => {
ambientPad.triggerAttackRelease(notes, "1n", time); }, ambientNotes, "2n").start(0);
                        break;
                }
                Tone.Transport.loop = true; // Loop the transport to keep the sounds playing
indefinitely
                Tone.Transport.loopEnd = '8m'; // Set a long loop time (8 measures)
                Tone.Transport.start();
                // Enable pause button after sound starts playing
                pauseButton.disabled = false;
                pauseButton.classList.remove('bg-red-400', 'text-white', 'cursor-not-allowed');
                pauseButton.classList.add('bg-red-600', 'hover:bg-red-700', 'text-white',
'cursor-pointer');
                pauseButton.textContent = 'Pause';
            });
        });
        pauseButton.addEventListener('click', () => {
```

```
                if (Tone.Transport.state === "started") { Tone.Transport.pause();
pauseButton.textContent = 'Resume'; } else if (Tone.Transport.state === "paused") {
Tone.Transport.start(); pauseButton.textContent = 'Pause'; }
            });
            if (Tone.Transport.state !== "started") { pauseButton.disabled = true; }
            break;
        case 'thought-journal':
            // Check for Firebase readiness before trying to initialize the journal
            if (!isAuthReady || !db || !userId) {
                content = <p class="text-red-500 text-lg">Journal requires authentication and
database setup. Please wait a moment or check the console.</p>;
                gameDisplay.innerHTML = content;
                return;
            }

            content = `<div class="flex flex-col h-full space-y-4">
                    <textarea id="journal-text" class="flex-1 p-4 rounded-xl border-2
border-gray-300 focus:outline-none focus:border-blue-500 transition-all resize-none"
placeholder="Write down your thoughts here..."></textarea>
                    <div class="flex space-x-4">
                        <button id="save-journal" class="flex-1 px-3 py-1 text-sm bg-cyan-600
text-white rounded-full font-semibold hover:bg-cyan-700 transition-all transform
hover:scale-105 shadow-md">Save Thought</button>
                        <button id="clear-all-entries" class="flex-1 px-3 py-1 text-sm bg-teal-600
text-white rounded-full font-semibold hover:bg-teal-700 transition-all transform hover:scale-105
shadow-md">Delete All</button>
                    </div>
                    <div id="journal-entries-list" class="flex-1 overflow-y-auto p-4 bg-white
rounded-lg border border-blue-200 mt-4 shadow-inner">
                        <p class="text-gray-400 text-sm text-center">Your past thoughts will
appear here.</p>
                    </div>
                </div>`;
            gameDisplay.innerHTML = content;

            const journalTextarea = document.getElementById('journal-text');
            const saveButton = document.getElementById('save-journal');
            const clearAllButton = document.getElementById('clear-all-entries');
            const entriesList = document.getElementById('journal-entries-list');

            const renderEntries = (snapshot) => {
                entriesList.innerHTML = '';
                const entries = [];
                snapshot.forEach(doc => {
                    const data = doc.data();
                    entries.push({ id: doc.id, text: data.text, date: data.date });
                });
```

```
                if (entries.length === 0) {
                    entriesList.innerHTML = '<p class="text-gray-400 text-sm text-center">Your past thoughts will appear here.</p>';
                    return;
                }

                // Sort entries by timestamp in descending order (newest first)
                entries.sort((a, b) => b.date - a.date);

                entries.forEach(entry => {
                    const date = new Date(entry.date).toLocaleString();
                    const entryDiv = document.createElement('div');
                    entryDiv.classList.add('p-3', 'bg-blue-100', 'rounded-xl', 'mb-2', 'shadow-sm');
                    entryDiv.innerHTML = `
                        <div class="flex justify-between items-start">
                            <p class="text-gray-700 text-sm">${entry.text}</p>
                            <button class="delete-btn text-red-500 hover:text-red-700 ml-4 flex-shrink-0" data-id="${entry.id}">
                                &times;
                            </button>
                        </div>
                        <p class="text-xs text-gray-500 mt-2">${date}</p>
                    `;
                    entriesList.appendChild(entryDiv);
                });

                entriesList.querySelectorAll('.delete-btn').forEach(button => {
                    button.addEventListener('click', (e) => {
                        const id = e.target.dataset.id;
                        deleteJournalEntry(id);
                    });
                });
            };

            const deleteJournalEntry = async (docId) => {
                try {
                    await deleteDoc(doc(getJournalCollectionRef(), docId));
                    showMessage("Entry deleted successfully.");
                } catch (error) {
                    console.error("Error deleting document: ", error);
                    showMessage("Error deleting entry. Check console for details.");
                }
            };

            const saveEntry = async () => {
                const newEntryText = journalTextarea.value.trim();
                if (newEntryText) {
                    try {
```

```
                    await addDoc(getJournalCollectionRef(), {
                        text: newEntryText,
                        date: Date.now() // Timestamp for sorting
                    });
                    journalTextarea.value = '';
                    showMessage("Entry saved successfully!");
                } catch (e) {
                    console.error("Error adding document: ", e);
                    showMessage("Error saving entry. Check console for details.");
                }
            }
        };

        const clearAllJournalEntries = async () => {
            try {
                const q = getJournalCollectionRef();
                const snapshot = await getDocs(q);
                if (snapshot.docs.length === 0) {
                    showMessage("Journal is already empty.");
                    return;
                }

                // Use a batch for efficiency
                const batch = writeBatch(db);
                snapshot.docs.forEach((d) => {
                    batch.delete(d.ref);
                });
                await batch.commit();
                showMessage("All entries deleted.");
            } catch (error) {
                console.error("Error deleting all documents: ", error);
                showMessage("Error deleting all entries. Check console for details.");
            }
        };

        // Set up the real-time listener
        journalUnsubscribe = onSnapshot(getJournalCollectionRef(), renderEntries,
(error) => {
            console.error("Firestore real-time error: ", error);
            entriesList.innerHTML = <p class="text-red-500 text-sm text-center">Error
loading journal: ${error.message}</p>;
        });

        saveButton.addEventListener('click', saveEntry);
        clearAllButton.addEventListener('click', clearAllJournalEntries);

        break;
    default:
```

```
                content = <p class="text-gray-400 text-lg">Select a game to start playing.</p>;
                gameDisplay.innerHTML = content;
            }
        };

        // --- Event Listeners for Game Cards ---
        gameCards.forEach(card => {
            card.addEventListener('click', () => {
                displayGame(card.id);
            });
        });

    </script>
</body>
</html>**
```