```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-g">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Collaborative IFS Toolkit</title>
    <!-- Load Tailwind CSS -->
    <script src="https://cdn.tailwindcss.com"></script>
    <!-- Load Firebase -->
    <script type="module">
        // --- Firebase Imports ---
        import { initializeApp } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";
        import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged,
setLogLevel } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";
        import { getFirestore, doc, onSnapshot, setDoc, updateDoc, collection } from
"https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";

        // --- Firebase Configuration ---
        // These variables are provided by the environment.
        const appId = typeof __app_id !== 'undefined' ? __app_id : 'default-ifs-session';
        const firebaseConfig = typeof __firebase_config !== 'undefined' ?
JSON.parse(__firebase_config) : {};
        const initialAuthToken = typeof __initial_auth_token !== 'undefined' ? __initial_auth_token :
null;

        let db, auth, userId;
        let sessionDocRef;
        let debounceTimer;

        // --- Main Data Store ---
        // This will hold the local state, synced with Firestore
        let sessionData = {
            parts: [],
            educationNotes: {
                managers: "",
                firefighters: "",
                exiles: "",
                self: ""
            },
            woundMap: ""
        };

        // --- 1. Initialize Firebase ---
        async function initFirebase() {
```

```
    try {
        const app = initializeApp(firebaseConfig);
        db = getFirestore(app);
        auth = getAuth(app);
        setLogLevel('Debug'); // Enable Firestore logging

        // Define the single, shared document for this session
        // This uses a public path based on the appId, allowing collaboration
        const sessionPath = `/artifacts/${appId}/public/data/ifs-session`;
        sessionDocRef = doc(db, sessionPath, 'main');

        listenForAuth();
    } catch (e) {
        console.error("Error initializing Firebase:", e);
        displayError("Could not connect to the collaborative session. Please refresh.");
    }
}

// --- 2. Handle Authentication ---
function listenForAuth() {
    onAuthStateChanged(auth, async (user) => {
        if (user) {
            userId = user.uid;
            console.log("Authenticated. User ID:", userId);
            document.getElementById('loading-overlay').classList.add('hidden');
            document.getElementById('session-id').textContent = appId;
            // Now that we're authenticated, start listening to the session document
            loadSessionData();
        } else {
            // No user, try to sign in
            signIn();
        }
    });
}

async function signIn() {
    try {
        if (initialAuthToken) {
            console.log("Signing in with custom token...");
            await signInWithCustomToken(auth, initialAuthToken);
        } else {
            console.log("Signing in anonymously...");
            await signInAnonymously(auth);
        }
```

```
    } catch (e) {
        console.error("Error signing in:", e);
        displayError("Authentication failed. Collaboration will not work.");
    }
}

// --- 3. Load & Sync Session Data (Real-time) ---
async function loadSessionData() {
    try {
        // Ensure the document exists
        await setDoc(sessionDocRef, sessionData, { merge: true });

        // Listen for real-time updates
        onSnapshot(sessionDocRef, (doc) => {
            if (doc.exists()) {
                sessionData = doc.data();
                console.log("Real-time data received:", sessionData);
                renderUI();
            } else {
                console.log("No session data, creating new document.");
                setDoc(sessionDocRef, sessionData); // Create it
            }
        }, (error) => {
            console.error("Error in onSnapshot:", error);
            displayError("Lost connection to the session.");
        });
    } catch (e) {
        console.error("Error loading session data:", e);
    }
}

// --- 4. Render UI from Local State ---
// This function re-draws the UI based on the `sessionData` object
function renderUI() {
    renderPartsMap();
    renderEducationNotes();
    renderWoundMap();
}

function renderPartsMap() {
    const container = document.getElementById('parts-map-container');
    container.innerHTML = ''; // Clear existing parts

    if (!sessionData.parts || sessionData.parts.length === 0) {
```

```
            container.innerHTML = `<p class="text-gray-500 italic">No parts added yet. Use the
form above to add one.</p>`;
            return;
        }

        sessionData.parts.forEach(part => {
            const partId = part.id;
            const card = document.createElement('div');
            card.className = `bg-white p-4 rounded-lg shadow-md border-l-4
${getPartBorderColor(part.type)}`;

            card.innerHTML = `
                <div class="flex justify-between items-center mb-3">
                    <h3 class="text-xl font-bold text-gray-800">${part.name}</h3>
                    <span class="text-sm font-medium px-3 py-1 rounded-full
${getPartBgColor(part.type)} ${getPartTextColor(part.type)}">${part.type}</span>
                </div>

                ${createTextArea(partId, 'role', 'Role (Its job?):', part.role)}
                ${createTextArea(partId, 'feelings', 'Feelings:', part.feelings)}
                ${createTextArea(partId, 'bodySensation', 'Body Sensation:', part.bodySensation)}
                ${createTextArea(partId, 'clientNotes', 'Client Notes:', part.clientNotes, 'bg-blue-50')}
                ${createTextArea(partId, 'therapistNotes', 'Therapist Notes:', part.therapistNotes,
'bg-green-50')}

                <button data-part-id="${partId}" class="delete-part-btn mt-3 text-red-500
hover:text-red-700 text-sm">
                    Delete Part
                </button>
            `;
            container.appendChild(card);
        });

        // Re-attach event listeners for textareas and delete buttons
        attachPartListeners();
    }

    function createTextArea(partId, field, label, value, bgClass = 'bg-gray-50') {
        return `
            <label class="block text-sm font-medium text-gray-700 mt-2">${label}</label>
            <textarea
                data-part-id="${partId}"
                data-field="${field}"
```

```
                class="part-textarea mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 sm:text-sm ${bgClass}"
                rows="2"
            >${value || ''}</textarea>
        `;
    }

    function renderEducationNotes() {
        if (sessionData.educationNotes) {
            document.getElementById('notes-managers').value =
sessionData.educationNotes.managers || '';
            document.getElementById('notes-firefighters').value =
sessionData.educationNotes.firefighters || '';
            document.getElementById('notes-exiles').value = sessionData.educationNotes.exiles
|| '';
            document.getElementById('notes-self').value = sessionData.educationNotes.self || '';
        }
    }

    function renderWoundMap() {
        if (sessionData.woundMap) {
            document.getElementById('wound-map-textarea').value = sessionData.woundMap;
        }
    }


    // --- 5. Handle UI Interactions & Update Firestore ---

    function attachPartListeners() {
        // Textarea listeners
        document.querySelectorAll('.part-textarea').forEach(textarea => {
            textarea.addEventListener('input', (e) => {
                const { partId, field } = e.target.dataset;
                const value = e.target.value;

                // Optimistic local update
                const partIndex = sessionData.parts.findIndex(p => p.id === partId);
                if (partIndex > -1) {
                    sessionData.parts[partIndex][field] = value;
                }

                // Debounced Firestore update
                debouncedUpdate(`parts.${partId}.${field}`, value, () => {
                    const newPartsArray = [...sessionData.parts];
```

```
                const partToUpdateIndex = newPartsArray.findIndex(p => p.id === partId);
                if (partToUpdateIndex > -1) {
                    newPartsArray[partToUpdateIndex] = { ...newPartsArray[partToUpdateIndex],
[field]: value };
                }
                return { parts: newPartsArray };
            });
        });
    });

    // Delete button listeners
    document.querySelectorAll('.delete-part-btn').forEach(btn => {
        btn.addEventListener('click', (e) => {
            const partId = e.target.dataset.partId;
            handleDeletePart(partId);
        });
    });
}

async function handleAddPart(e) {
    e.preventDefault();
    const partNameInput = document.getElementById('part-name');
    const partTypeInput = document.getElementById('part-type');

    const newPart = {
        id: `part_${Date.now()}`,
        name: partNameInput.value,
        type: partTypeInput.value,
        role: "",
        feelings: "",
        bodySensation: "",
        clientNotes: "",
        therapistNotes: ""
    };

    if (!newPart.name) {
        displayError("Please enter a name for the part.");
        return;
    }

    try {
        // Optimistic local update
        sessionData.parts.push(newPart);
        renderPartsMap();
```

```
        // Update Firestore
        await updateDoc(sessionDocRef, {
            parts: sessionData.parts
        });

        partNameInput.value = ''; // Clear form
    } catch (e) {
        console.error("Error adding part:", e);
        displayError("Could not add part. Check connection.");
        // Revert optimistic update
        sessionData.parts.pop();
        renderPartsMap();
    }
}

async function handleDeletePart(partId) {
    // Confirm deletion
    const modal = document.getElementById('delete-modal');
    modal.classList.remove('hidden');

    document.getElementById('confirm-delete').onclick = async () => {
        modal.classList.add('hidden');

        const oldParts = [...sessionData.parts];

        // Optimistic local update
        sessionData.parts = sessionData.parts.filter(p => p.id !== partId);
        renderPartsMap();

        try {
            // Update Firestore
            await updateDoc(sessionDocRef, {
                parts: sessionData.parts
            });
        } catch (e) {
            console.error("Error deleting part:", e);
            displayError("Could not delete part. Check connection.");
            // Revert optimistic update
            sessionData.parts = oldParts;
            renderPartsMap();
        }
    };
```

```
    document.getElementById('cancel-delete').onclick = () => {
        modal.classList.add('hidden');
    };
}

// Generic function for shared textareas (education, wound map)
function handleSharedTextUpdate(e) {
    const { field } = e.target.dataset;
    const value = e.target.value;

    // Optimistic local update
    if (field.startsWith('educationNotes.')) {
        const key = field.split('.')[1];
        sessionData.educationNotes[key] = value;
    } else {
        sessionData[field] = value;
    }

    // Debounced Firestore update
    debouncedUpdate(field, value, () => {
        // This structure creates the update object, e.g., { woundMap: "new value" }
        // or { "educationNotes.managers": "new value" }
        let updateObj = {};
        updateObj[field] = value;
        return updateObj;
    });
}

// Debounce utility to prevent spamming Firestore
// This is more complex because it batches updates
let pendingUpdates = {};
function debouncedUpdate(field, value, getUpdateObject) {
    // Store the latest value for this field
    pendingUpdates[field] = value;

    clearTimeout(debounceTimer);
    debounceTimer = setTimeout(async () => {
        try {
            // Create a single update object from all pending changes
            let batchedUpdate = {};
            for (const [key, val] of Object.entries(pendingUpdates)) {
                // This logic handles nested objects like "educationNotes.managers"
                // and also flat arrays like "parts"
                if (key.startsWith('parts.')) {
```

```
                // Re-calculate the full 'parts' array on update
                batchedUpdate['parts'] = [...sessionData.parts];
            } else {
                batchedUpdate[key] = val;
            }
        }

        console.log("Debounced update to Firestore:", batchedUpdate);
        await updateDoc(sessionDocRef, batchedUpdate);
        pendingUpdates = {}; // Clear the queue
    } catch (e) {
        console.error("Error in debounced update:", e);
        displayError("Error saving changes. Re-syncing...");
        // Force a re-load from server on error
        onSnapshot(sessionDocRef, (doc) => {
            if (doc.exists()) {
                sessionData = doc.data();
                renderUI();
            }
        });
    }
  }, 1000); // 1-second debounce
}

// --- 6. UI Utilities ---

function displayError(message) {
  const errorEl = document.getElementById('error-message');
  errorEl.textContent = message;
  errorEl.classList.remove('hidden');
  setTimeout(() => {
    errorEl.classList.add('hidden');
  }, 3000);
}

// Tab switching logic
function setupTabs() {
  const tabs = document.querySelectorAll('[data-tab-target]');
  const tabContents = document.querySelectorAll('[data-tab-content]');

  tabs.forEach(tab => {
    tab.addEventListener('click', () => {
      const target = document.querySelector(tab.dataset.tabTarget);
```

```
            tabContents.forEach(tabContent => {
                tabContent.classList.add('hidden');
            });
            tabs.forEach(t => {
                t.classList.remove('text-indigo-600', 'border-indigo-600');
                t.classList.add('text-gray-500', 'border-transparent');
            });

            tab.classList.add('text-indigo-600', 'border-indigo-600');
            tab.classList.remove('text-gray-500', 'border-transparent');
            target.classList.remove('hidden');
        });
    });
}

// Helper functions for styling
function getPartBorderColor(type) {
    switch (type) {
        case 'Manager': return 'border-blue-500';
        case 'Firefighter': return 'border-red-500';
        case 'Exile': return 'border-purple-500';
        default: return 'border-gray-500';
    }
}
function getPartBgColor(type) {
     switch (type) {
        case 'Manager': return 'bg-blue-100';
        case 'Firefighter': return 'bg-red-100';
        case 'Exile': return 'bg-purple-100';
        default: return 'bg-gray-100';
    }
}
function getPartTextColor(type) {
     switch (type) {
        case 'Manager': return 'text-blue-800';
        case 'Firefighter': return 'text-red-800';
        case 'Exile': return 'text-purple-800';
        default: return 'text-gray-800';
    }
}

// --- 7. Initial Load ---
window.addEventListener('load', () => {
    initFirebase();
```

```
            setupTabs();
            document.getElementById('add-part-form').addEventListener('submit', handleAddPart);

            // Add listeners for shared textareas
            document.querySelectorAll('.shared-textarea').forEach(textarea => {
                textarea.addEventListener('input', handleSharedTextUpdate);
            });
        });

    </script>
</head>
<body class="bg-gray-100 font-sans leading-normal tracking-normal" style="font-family: 'Inter', sans-serif;">

    <!-- Loading Overlay -->
    <div id="loading-overlay" class="fixed inset-0 bg-white bg-opacity-75 z-50 flex items-center justify-center">
        <div class="animate-spin rounded-full h-16 w-16 border-t-4 border-b-4 border-indigo-600"></div>
        <p class="ml-4 text-lg text-gray-700">Connecting to session...</p>
    </div>

    <!-- Error Message Bar -->
    <div id="error-message" class="hidden fixed top-0 left-0 right-0 bg-red-600 text-white text-center p-2 z-50">
        Error message
    </div>

    <!-- Delete Confirmation Modal -->
    <div id="delete-modal" class="hidden fixed inset-0 bg-gray-600 bg-opacity-50 overflow-y-auto h-full w-full z-40">
        <div class="relative top-20 mx-auto p-5 border w-96 shadow-lg rounded-md bg-white">
            <div class="mt-3 text-center">
                <h3 class="text-lg leading-6 font-medium text-gray-900">Delete Part</h3>
                <div class="mt-2 px-7 py-3">
                    <p class="text-sm text-gray-500">Are you sure you want to delete this part? This action cannot be undone.</p>
                </div>
                <div class="items-center px-4 py-3 gap-2 flex justify-center">
                    <button id="cancel-delete" class="px-4 py-2 bg-gray-200 text-gray-800 rounded-md hover:bg-gray-300">
                        Cancel
                    </button>
```

```
                <button id="confirm-delete" class="px-4 py-2 bg-red-600 text-white rounded-md
hover:bg-red-700">
                    Delete
                </button>
            </div>
        </div>
    </div>

    <!-- Main Content -->
    <div class="container mx-auto max-w-4xl p-4 pt-8">

        <header class="mb-6">
            <h1 class="text-4xl font-bold text-gray-900">Collaborative IFS Toolkit</h1>
            <p class="text-lg text-gray-600">A shared space for client and therapist to map the inner
world.</p>
            <div class="mt-4 bg-indigo-100 border-l-4 border-indigo-500 text-indigo-700 p-4
rounded-md">
                <strong>Share this Session ID:</strong>
                <span id="session-id" class="font-mono bg-indigo-200 px-2 py-1
rounded">loading...</span>
                <p class="text-sm mt-1">Both users must join this session (use the same link) to
collaborate.</p>
            </div>
        </header>

        <!-- Tool: Parts Mapper -->
        <section class="bg-white rounded-lg shadow-lg p-6 mb-8">
            <h2 class="text-2xl font-bold text-gray-800 mb-4">Activity 1: Parts Mapper</h2>

            <!-- Add Part Form -->
            <form id="add-part-form" class="mb-6 p-4 bg-gray-50 rounded-md grid grid-cols-1
md:grid-cols-3 gap-4">
                <div>
                    <label for="part-name" class="block text-sm font-medium text-gray-700">Part
Name</label>
                    <input type="text" id="part-name" class="mt-1 block w-full rounded-md
border-gray-300 shadow-sm focus:border-indigo-500 focus:ring-indigo-500" placeholder="e.g.,
The Anxious Planner">
                </div>
                <div>
                    <label for="part-type" class="block text-sm font-medium text-gray-700">Part
Type</label>
```

```
                <select id="part-type" class="mt-1 block w-full rounded-md border-gray-300
shadow-sm focus:border-indigo-500 focus:ring-indigo-500">
                    <option>Manager</option>
                    <option>Firefighter</option>
                    <option>Exile</option>
                </select>
            </div>
            <button type="submit" class="md:mt-6 w-full md:w-auto bg-indigo-600 text-white px-4
py-2 rounded-md shadow-sm hover:bg-indigo-700 focus:outline-none focus:ring-2
focus:ring-indigo-500 focus:ring-offset-2">
                Add Part
            </button>
        </form>

        <!-- Parts Container -->
        <div id="parts-map-container" class="space-y-4">
            <!-- Part cards will be injected here by JavaScript -->
        </div>
    </section>

    <!-- Tool: Educational Section -->
    <section class="bg-white rounded-lg shadow-lg p-6 mb-8">
        <h2 class="text-2xl font-bold text-gray-800 mb-4">Activity 2: Understanding Your
Parts</h2>
        <p class="text-gray-600 mb-4">Read about each part type and use the collaborative
space to reflect on your own parts.</p>

        <!-- Tabs -->
        <div class="border-b border-gray-200">
            <nav class="-mb-px flex space-x-8" aria-label="Tabs">
                <button data-tab-target="#tab-managers" class="whitespace-nowrap py-4 px-1
border-b-2 font-medium text-sm text-indigo-600 border-indigo-600">
                    Managers
                </button>
                <button data-tab-target="#tab-firefighters" class="whitespace-nowrap py-4 px-1
border-b-2 font-medium text-sm text-gray-500 border-transparent hover:text-gray-700
hover:border-gray-300">
                    Firefighters
                </button>
                <button data-tab-target="#tab-exiles" class="whitespace-nowrap py-4 px-1
border-b-2 font-medium text-sm text-gray-500 border-transparent hover:text-gray-700
hover:border-gray-300">
                    Exiles (Inner Child)
                </button>
```

```
                <button data-tab-target="#tab-self" class="whitespace-nowrap py-4 px-1 border-b-2
font-medium text-sm text-gray-500 border-transparent hover:text-gray-700
hover:border-gray-300">
                    The Self
                </button>
            </nav>
        </div>

        <!-- Tab Content -->
        <div class="pt-6">
            <!-- Managers Tab -->
            <div id="tab-managers" data-tab-content>
                <h3 class="text-lg font-semibold text-blue-800">Managers (The "Proactive
Protectors")</h3>
                <p class="text-gray-700 mt-2">
                    These parts manage daily life to keep you safe and in control. They strive to
prevent pain and avoid triggering "Exiles." Anxiety is often the energy a Manager uses to do its
job.
                    <br><strong>Examples:</strong> The Planner, The Inner Critic, The
People-Pleaser, The Perfectionist.
                </p>
                <textarea
                    id="notes-managers"
                    data-field="educationNotes.managers"
                    class="shared-textarea mt-4 block w-full rounded-md border-gray-300
shadow-sm focus:border-blue-500 focus:ring-blue-500 sm:text-sm bg-blue-50"
                    rows="4"
                    placeholder="Collaborative reflections on Manager parts..."
                ></textarea>
            </div>
            <!-- Firefighters Tab -->
            <div id="tab-firefighters" data-tab-content class="hidden">
                <h3 class="text-lg font-semibold text-red-800">Firefighters (The "Reactive
Protectors")</h3>
                <p class="text-gray-700 mt-2">
                    These parts react *after* a wound has been triggered. Their goal is to douse the
emotional fire at all costs, often with impulsive or extreme behaviors, without regard for
consequences.
                    <br><strong>Examples:</strong> Addictions, Binge-eating, Dissociation (zoning
out), Rage.
                </p>
                <textarea
                    id="notes-firefighters"
                    data-field="educationNotes.firefighters"
```

```
                    class="shared-textarea mt-4 block w-full rounded-md border-gray-300
shadow-sm focus:border-red-500 focus:ring-red-500 sm:text-sm bg-red-50"
                    rows="4"
                    placeholder="Collaborative reflections on Firefighter parts..."
                ></textarea>
            </div>
            <!-- Exiles Tab -->
            <div id="tab-exiles" data-tab-content class="hidden">
                <h3 class="text-lg font-semibold text-purple-800">Exiles (The "Inner
Children")</h3>
                <p class="text-gray-700 mt-2">
                    These are the young, vulnerable parts that hold the original pain, trauma, and
overwhelming emotions. The rest of the system works to keep them "exiled" to prevent their
pain from flooding you.
                    <br><strong>Common Feelings:</strong> Shame, loneliness, fear,
unworthiness.
                </p>
                <textarea
                    id="notes-exiles"
                    data-field="educationNotes.exiles"
                    class="shared-textarea mt-4 block w-full rounded-md border-gray-300
shadow-sm focus:border-purple-500 focus:ring-purple-500 sm:text-sm bg-purple-50"
                    rows="4"
                    placeholder="Collaborative reflections on Exiled parts..."
                ></textarea>
            </div>
            <!-- Self Tab -->
            <div id="tab-self" data-tab-content class="hidden">
                <h3 class="text-lg font-semibold text-green-800">The Self (The "Reparenting"
Agent)</h3>
                <p class="text-gray-700 mt-2">
                    The Self is not a part. It is your core essence—naturally calm, compassionate,
curious, and confident. It cannot be damaged. The goal of IFS is to lead from this Self-energy.
                    <br><strong>The 8 C's:</strong> Compassion, Curiosity, Calm, Confidence,
Clarity, Creativity, Courage, Connectedness.
                </p>
                <textarea
                    id="notes-self"
                    data-field="educationNotes.self"
                    class="shared-textarea mt-4 block w-full rounded-md border-gray-300
shadow-sm focus:border-green-500 focus:ring-green-500 sm:text-sm bg-green-50"
                    rows="4"
                    placeholder="Collaborative reflections on moments of Self-energy..."
                ></textarea>
```

```
            </div>
        </div>
    </section>

    <!-- Tool: Childhood Wound Mapping -->
    <section class="bg-white rounded-lg shadow-lg p-6 mb-8">
        <h2 class="text-2xl font-bold text-gray-800 mb-4">Activity 3: Reflecting on Roots</h2>
        <p class="text-gray-600 mb-4">Use this shared space to explore potential childhood wounds. These are early experiences where you felt overwhelming emotion without support. Consider the themes of rejection, abandonment, injustice, betrayal, or neglect.</p>

        <label for="wound-map-textarea" class="block text-sm font-medium text-gray-700">Collaborative Reflection Space</label>
        <textarea
            id="wound-map-textarea"
            data-field="woundMap"
            class="shared-textarea mt-2 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500 focus:ring-indigo-500 sm:text-sm"
            rows="10"
            placeholder="What early memories or family dynamics come to mind? What feelings (e.g., loneliness, shame, fear) are connected to them? What protectors (parts) seem to have shown up around that time?"
        ></textarea>
    </section>
  </div>
</body>
</html>
```