Index.html<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book 7 MEMORY (Global DaaS MVP)</title>
    <!-- PWA MANIFEST LINK -->
    <link rel="manifest" href="manifest.json">

    <!-- PWA METATAGS -->
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="theme-color" content="#1f2937"> <!-- Dark blue header -->

    <!-- Load Tailwind CSS for mobile-first aesthetics -->
    <script src="https://cdn.tailwindcss.com"></script>
    <style>
        @import url('https://fonts.googleapis.com/css2?
family=Inter:wght@100..900&display=swap');
        body { font-family: 'Inter', sans-serif; background-color: #f7f9fb; }
        .main-container { max-width: 450px; }
        .tab-content { display: none; }
        .tab-content.active { display: block; }
        /* Custom scrollbar for memory feed */
        #memory-feed::-webkit-scrollbar { width: 4px; }
        #memory-feed::-webkit-scrollbar-thumb { background: #cbd5e1; border-radius: 2px; }
        .red-confirm-btn {
            background-color: #ef4444; /* Red 500 */
        }
        .red-confirm-btn:hover {
            background-color: #dc2626; /* Red 600 */
        }
        .red-confirm-btn:active {
            transform: scale(0.95);
        }
    </style>
</head>
<body class="min-h-screen flex justify-center p-4">
    <div class="main-container w-full bg-white shadow-xl rounded-2xl overflow-hidden flex
flex-col">

        <!-- Header & Nav Tabs -->
        <header class="p-4 bg-gray-900 text-white shadow-lg">
            <h1 class="text-2xl font-extrabold text-indigo-300">🧠 Book 7 MEMORY</h1>
            Global DaaS Engine - Status: <span
id="status-display" class="font-medium text-yellow-300">Initializing...
            <nav class="mt-4 flex space-x-1 border-b border-gray-700">
                <button data-tab="summary" class="tab-button flex-1 py-2 text-sm font-semibold

```
border-b-2 border-indigo-500 text-indigo-400 focus:outline-none">Summary</button>
            <button data-tab="add" class="tab-button flex-1 py-2 text-sm font-semibold
border-b-2 border-transparent hover:border-gray-500 focus:outline-none">Add
Entry</button>
            <button data-tab="feed" class="tab-button flex-1 py-2 text-sm font-semibold
border-b-2 border-transparent hover:border-gray-500 focus:outline-none">Memory
Feed</button>
            <button data-tab="reports" class="tab-button flex-1 py-2 text-sm font-semibold
border-b-2 border-transparent hover:border-gray-500 focus:outline-none">Reports</button>
            <button data-tab="settings" class="tab-button flex-1 py-2 text-sm font-semibold
border-b-2 border-transparent hover:border-gray-500 focus:outline-
none">Settings</button>
        </nav>
    </header>

    <!-- Main Content Area -->
    <main class="flex-grow p-4 overflow-y-auto">

        <!-- Tab 1: Today's Summary -->
        <section id="summary" class="tab-content active space-y-4">
            <div id="smart-reminders" class="space-y-2">
                <!-- Smart Reminders are injected here -->
            </div>

            <h2 class="text-lg font-bold text-gray-700 border-b pb-2">Today's Balance</h2>
            <div id="summary-metrics" class="bg-gray-50 p-4 rounded-xl shadow-inner grid
grid-cols-2 gap-4">
                <div class="col-span-2 text-center">
                    <p class="text-xs font-semibold text-gray-500">Net Balance (Today)</p>
                    <p id="net-balance" class="text-4xl font-extrabold text-green-600">0.00</p>
                </div>
                <div>
                    <p class="text-sm text-gray-600">Total In</p>
                    <p id="total-in" class="text-xl font-bold text-green-500">0.00</p>
                </div>
                <div>
                    <p class="text-sm text-gray-600">Total Out</p>
                    <p id="total-out" class="text-xl font-bold text-red-500">0.00</p>
                </div>
            </div>

            <div id="sync-status-card" class="mt-4 p-3 bg-blue-100 border-l-4 border-blue-500
rounded-lg text-sm">
                <p class="font-semibold text-blue-700">DB Status:</p>
                <p id="db-status-detail" class="text-blue-600">Connecting to Firestore...</p>
            </div>
        </section>

        <!-- Tab 2: Add Entry -->
        <section id="add" class="tab-content space-y-4">
            <h2 class="text-lg font-bold text-gray-700 mb-4">Tell Book 7 What
```

```
Happened</h2>

            <div class="flex items-center space-x-2">
                <input type="text" id="raw-input" placeholder="e.g., Paid 150 for rent today, or
Sold 50 to Jane"
                       class="flex-grow p-3 border-2 border-indigo-300 rounded-xl focus:ring-
indigo-500 focus:border-indigo-500 transition duration-150 shadow-md">
                <button id="submit-entry"
                        class="bg-indigo-600 text-white p-3 rounded-xl shadow-lg hover:bg-indigo-
700 transition duration-150 active:scale-95">
                    <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6" fill="none"
viewBox="0 0 24 24" stroke="currentColor" stroke-width="2">
                        <path stroke-linecap="round" stroke-linejoin="round" d="M9 5H7a2 2 0 00-2
2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2 0 00-2-2h-2M9 5a2 2 0 002 2h2a2 2 0 002-2M9
5a2 2 0 012-2h2a2 2 0 012 2m-3 7h3m-3 4h3m-6-4h.01M9 16h.01" />
                    </svg>
                </button>
            </div>

            <div id="input-feedback" class="text-sm mt-2 p-2 bg-yellow-50 text-yellow-700
rounded-lg hidden"></div>

            <div id="parsed-fields" class="mt-6 p-4 bg-indigo-50 rounded-xl shadow-inner">
                <p class="font-semibold text-indigo-700 mb-2">Parsed Result (Confirmation)
</p>
                <div class="grid grid-cols-2 gap-3 text-sm">
                    <div><span class="font-medium text-gray-600">Type:</span> <span
id="parsed-type" class="font-bold text-indigo-900">N/A</span></div>
                    <div><span class="font-medium text-gray-600">Currency:</span> <input
id="parsed-currency" type="text" value="USD" class="w-16 p-1 border rounded text-xs text-
center"></div>
                    <div class="col-span-2"><span class="font-medium text-gray-600">Amount:
</span> <span id="parsed-amount" class="text-lg font-extrabold text-green-
700">0.00</span></div>
                    <div class="col-span-2"><span class="font-medium text-gray-
600">Description:</span> <span id="parsed-desc">...</span></div>
                </div>
            </div>
        </section>

        <!-- Tab 3: Memory Feed -->
        <section id="feed" class="tab-content space-y-4">
            <h2 class="text-lg font-bold text-gray-700 border-b pb-2">Business Memory Log
(Real-Time)</h2>
            <div id="memory-feed" class="space-y-3 h-96 overflow-y-auto p-1">
                <!-- Entries are injected here by the Firestore listener -->
                <p class="text-center text-gray-500 italic mt-4" id="feed-placeholder">Loading
memory from cloud...</p>
            </div>
        </section>
```

```
        <!-- Tab 4: Reports (PRO Tier Feature Foundation) -->
        <section id="reports" class="tab-content space-y-6 pt-4">
            <h2 class="text-xl font-bold text-indigo-600 border-b pb-2">PRO Financial Intelligence</h2>
            <p class="text-sm text-gray-600">Aggregated reports are calculated using the standardized $\text{USD}$ value (DaaS Core).</p>

            <div class="p-4 bg-white rounded-xl shadow-lg border border-indigo-200">
                <h3 class="text-lg font-semibold text-gray-800 mb-2">Last 30 Days Net Profit</h3>
                <p id="monthly-net-balance" class="text-3xl font-extrabold text-gray-700">Calculating...</p>
                <p class="text-xs text-gray-500 mt-2">This metric is crucial for the Micro-Lending Risk Score.</p>
            </div>

            <div class="p-4 bg-gray-50 rounded-xl space-y-3">
                <p class="font-semibold text-base text-gray-800">Export & Compliance (PRO Feature)</p>
                <button class="bg-gray-400 text-white px-4 py-2 rounded-lg text-sm shadow-md cursor-not-allowed">
                    Generate Tax Summary (Requires PRO Unlock)
                </button>
                <p class="text-xs text-gray-500">Tier 2 DaaS revenue requires these formalized outputs.</p>
            </div>
        </section>

        <!-- Tab 5: Settings -->
        <section id="settings" class="tab-content space-y-6 pt-4">
            <h2 class="text-xl font-bold text-gray-700">System Settings</h2>

            <div class="p-4 bg-yellow-50 border border-yellow-300 rounded-lg space-y-3">
                <p class="font-semibold text-lg text-yellow-800">Danger Zone</p>
                <p class="text-sm text-yellow-700">This action will delete all locally stored Book 7 data (synced and unsynced). This should only be done if you are migrating devices or clearing sensitive data.</p>

                <button id="delete-all-data"
                        class="red-confirm-btn text-white px-4 py-2 rounded-lg text-sm shadow-md transition duration-150">
                    Wipe All Local Data (Permanent)
                </button>

                <div id="delete-confirm-box" class="hidden mt-3 p-3 bg-red-200 rounded-lg">
                    <p class="text-red-900 font-semibold mb-2">Are you sure?</p>
                    <button id="confirm-delete" class="bg-red-700 text-white px-3 py-1 rounded-md text-xs hover:bg-red-800">
                        YES, Delete All
                    </button>
```

```
                    <button id="cancel-delete" class="bg-gray-500 text-white px-3 py-1 rounded-
md text-xs ml-2 hover:bg-gray-600">
                        Cancel
                    </button>
                </div>
            </div>
            <p id="auth-info" class="text-xs text-gray-500"></p>
        </section>
    </main>
</div>

<script type="module">
    // --- FIREBASE IMPORTS ---
    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, query, onSnapshot, addDoc, updateDoc, deleteDoc,
where, getDocs, orderBy } 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('error'); // Set logging level for production focus

    // --- GLOBAL SETUP AND CONSTANTS ---
    const BASE_CURRENCY = 'USD'; // USF Standard for DaaS aggregation
    const MOCK_FX_API = 'https://api.mock-fx.com/rate'; // Mock FX API URL
    const TODAY = new Date().toISOString().slice(0, 10);

    let db;
    let auth;
    let userId = null;
    let memoryLog = []; // Local cache of the Firestore data

    // --- FIREBASE INITIALIZATION ---

    const appId = typeof __app_id !== 'undefined' ? __app_id : 'default-app-id';
    const firebaseConfig = typeof __firebase_config !== 'undefined' ?
JSON.parse(__firebase_config) : null;
    const initialAuthToken = typeof __initial_auth_token !== 'undefined' ?
__initial_auth_token : null;

    if (firebaseConfig) {
        const app = initializeApp(firebaseConfig);
        db = getFirestore(app);
        auth = getAuth(app);
    } else {
        console.error("Firebase config not found. App cannot connect to cloud database.");
        document.getElementById('db-status-detail').textContent = 'Error: Missing Firebase
```

```
Config.';
        }

        // --- AUTHENTICATION AND DB CONNECTION ---

        onAuthStateChanged(auth, async (user) => {
            if (user) {
                userId = user.uid;
                document.getElementById('auth-info').textContent = `User ID: ${userId} (Signed
In)`;

                // 1. Update UI Status
                document.getElementById('status-display').textContent = 'Cloud Connected';
                document.getElementById('db-status-detail').textContent = 'Real-time data sync
active.';
                document.getElementById('sync-status-card').classList.add('bg-green-100',
'border-green-500');
                document.getElementById('sync-status-card').classList.remove('bg-blue-100',
'border-blue-500');

                // 2. Start Firestore Listener (onSnapshot)
                subscribeToMemoryLog();

            } else {
                console.log("Attempting sign-in...");
                if (initialAuthToken) {
                    try {
                        await signInWithCustomToken(auth, initialAuthToken);
                    } catch (error) {
                        console.error("Custom token sign-in failed:", error);
                        // Fallback to anonymous sign-in if token fails
                        await signInAnonymously(auth);
                    }
                } else {
                    await signInAnonymously(auth);
                }
            }
        });

        // --- FIRESTORE DATA FUNCTIONS ---

        const getCollectionPath = (uid) => `/artifacts/${appId}/public/data/memory_log`;

        function subscribeToMemoryLog() {
            if (!db || !userId) return;

            const memoryRef = collection(db, getCollectionPath(userId));
            const q = query(memoryRef, orderBy('localTimestamp', 'desc'));

            onSnapshot(q, (snapshot) => {
                memoryLog = snapshot.docs.map(doc => ({
                    ...doc.data(),
```

```
                id: doc.id // Store Firestore document ID
            }));

            console.log(`[Firestore] Updated memory log with ${memoryLog.length} entries.`);
            renderMemoryFeed();
            renderSummary();
            renderReports();
        }, (error) => {
            console.error("Firestore subscription error:", error);
            document.getElementById('db-status-detail').textContent = 'ERROR: Sync
disconnected.';
            document.getElementById('sync-status-card').classList.add('bg-red-100', 'border-
red-500');
        });
    }

    /**
     * Simulates the call to the central Currency Management Service (CMS)
     */
    async function fetchExchangeRate(ccy) {
        if (ccy === BASE_CURRENCY) return 1.0;

        // MOCK LOGIC: Use simplified, hardcoded rates for demonstration
        if (ccy === 'EUR') return 1.08;
        if (ccy === 'ZAR') return 0.054;
        if (ccy === 'INR') return 0.012;
        if (ccy === 'GBP') return 1.25;

        // Simulating API latency and failure risk for global deployment
        await new Promise(resolve => setTimeout(resolve, Math.random() * 500));
        if (Math.random() < 0.05) {
            throw new Error("MOCK FX API Timeout/Error");
        }
        return 1.0;
    }

    /**
     * Saves a new entry to Firestore, including USD standardization.
     * @param {Object} entry
     */
    async function saveNewEntry(entry) {
        if (!db || !userId) {
            showToast("DB Error: Not authenticated.", 'error');
            return;
        }

        showToast("Memory Logged. Converting currency...");

        try {
            // 1. USF Core: Fetch Exchange Rate & Standardize
            const rate = await fetchExchangeRate(entry.localCurrency);
```

```
            entry.baseAmount = entry.localAmount * rate;
            entry.conversionRate = rate;

            // 2. Add Authentication Metadata (for DaaS security)
            entry.userId = userId;

            // 3. Save to Firestore
            const memoryRef = collection(db, getCollectionPath(userId));
            await addDoc(memoryRef, entry);

            showToast("Entry saved and synchronized successfully!", 'success');

        } catch (error) {
            console.error("Failed to save entry:", error);
            showToast(`Failed to save: ${error.message}. Check console.`, 'error');
        }
    }

    // --- UTILITY FUNCTIONS ---

    function showToast(message, type = 'success') {
        const toast = document.createElement('div');
        toast.className = `fixed bottom-4 left-1/2 transform -translate-x-1/2 p-3 rounded-xl shadow-2xl text-white text-sm font-semibold transition-opacity duration-300 ${type === 'success' ? 'bg-green-600' : 'bg-red-600'}`;
        toast.textContent = message;
        document.body.appendChild(toast);
        setTimeout(() => {
            toast.style.opacity = '0';
            toast.addEventListener('transitionend', () => toast.remove());
        }, 3000);
    }

    function getTransactionIcon(type) {
        switch (type) {
            case 'Sale': return '💰';
            case 'Expense': return '🧾';
            case 'Debt': return '🤝';
            case 'Note': return '💬';
            default: return '❓';
        }
    }

    function getTodayDateString() {
        return new Date().toISOString().split('T')[0];
    }

    // --- INPUT PARSING (More Robust V2) ---

    /**
     * Parses the raw text input into structured transaction data.
```

```
     */
    function parseInput(text) {
        text = text.trim().toLowerCase();
        let amount = 0;
        let type = 'Note';
        let description = text;

        const amountRegex = /(?:[€$£₹]|eur|usd|zar|inr|gbp)?\s*([\d,]+\.?\d*)/;
        const amountMatch = text.match(amountRegex);

        if (amountMatch && amountMatch[1]) {
            amount = parseFloat(amountMatch[1].replace(/,/g, ''));
            description = description.replace(amountMatch[0], '').trim();
        }

        const expenseKeywords = ['paid', 'spent', 'bought', 'expense', 'materials', 'rent', 'salary',
'utility', 'stock'];
        const saleKeywords = ['sold', 'received', 'got', 'sale', 'from client', 'paid by', 'invoice',
'income'];
        const debtKeywords = ['owe', 'loan', 'lend', 'borrow', 'credit'];

        if (amount > 0) {
            if (expenseKeywords.some(keyword => text.includes(keyword))) {
                type = 'Expense';
            } else if (saleKeywords.some(keyword => text.includes(keyword))) {
                type = 'Sale';
            } else if (debtKeywords.some(keyword => text.includes(keyword))) {
                type = 'Debt';
            }
        } else if (text.includes('decision') || text.includes('task') || text.includes('meeting')) {
             type = 'Note';
        }

        description = description.trim().replace(/\s\s+/g, ' ');

        return { amount: amount || 0, type: type, description: description || text };
    }

    function handleInputSubmission() {
        const rawInput = document.getElementById('raw-input');
        const feedback = document.getElementById('input-feedback');

        if (!userId) {
             feedback.textContent = "Error: Waiting for cloud connection. Please wait a
moment.";
             feedback.classList.remove('hidden');
             return;
        }

        if (rawInput.value.length < 5) {
            feedback.textContent = "Please provide a description of the transaction.";
```

```
            feedback.classList.remove('hidden');
            return;
        }
        feedback.classList.add('hidden');

        const parsed = parseInput(rawInput.value);

        const newTransaction = {
            rawText: rawInput.value,
            localAmount: parsed.amount,
            localCurrency: document.getElementById('parsed-currency').value.toUpperCase()
|| BASE_CURRENCY,
            baseAmount: 0,
            conversionRate: 0,
            localTimestamp: Date.now(),
            type: parsed.type,
            dateOfEvent: getTodayDateString()
        };

        saveNewEntry(newTransaction);

        rawInput.value = '';
        document.getElementById('parsed-type').textContent = 'N/A';
        document.getElementById('parsed-amount').textContent = '0.00';
        document.getElementById('parsed-desc').textContent = '...';
    }

    function updateParsedView(event) {
        const rawInput = event.target.value;
        const parsed = parseInput(rawInput);

        document.getElementById('parsed-type').textContent = parsed.type;
        document.getElementById('parsed-amount').textContent =
parsed.amount.toFixed(2);
        document.getElementById('parsed-desc').textContent = parsed.description;

        const amountEl = document.getElementById('parsed-amount');
        amountEl.classList.remove('text-green-700', 'text-red-700');

        if (parsed.type === 'Sale') {
            amountEl.classList.add('text-green-700');
        } else if (parsed.type === 'Expense' || (parsed.type === 'Debt' && parsed.amount > 0))
{
            amountEl.classList.add('text-red-700');
        } else {
            amountEl.classList.add('text-gray-600');
        }
    }

    // --- UI RENDER FUNCTIONS ---
```

```
    function renderMemoryFeed() {
        const feedContainer = document.getElementById('memory-feed');
        feedContainer.innerHTML = '';

        if (memoryLog.length === 0) {
            feedContainer.innerHTML = '<p class="text-center text-gray-500 italic mt-4">Log
your first entry above to start your memory.</p>';
            return;
        }

        memoryLog.forEach(tx => {
            const isExpense = tx.type === 'Expense' || (tx.type === 'Debt' && tx.localAmount >
0 && tx.type.includes('Given'));
            const isSale = tx.type === 'Sale' || (tx.type === 'Debt' && tx.localAmount > 0 &&
tx.type.includes('Received'));
            const amountClass = isExpense ? 'text-red-600' : (isSale ? 'text-green-600' : 'text-
gray-600');

            const baseAmountDisplay = tx.baseAmount > 0 ?
                `<p class="text-xs text-gray-400">(${(tx.baseAmount).toFixed(2)}
${BASE_CURRENCY})</p>` : '';

            const syncIndicator = tx.baseAmount > 0 ?
                '<span class="text-green-500 text-xs ml-2">✓ Synced</span>' :
                '<span class="text-yellow-500 text-xs ml-2">...Converting</span>'; //
BaseAmount is the sync marker

            const entryHtml = `
                <div class="p-3 bg-white border-l-4 ${tx.baseAmount > 0 ? 'border-indigo-400' :
'border-yellow-400'} rounded-lg shadow-sm">
                    <div class="flex justify-between items-center">
                        <span class="text-xs font-semibold text-gray-500">${new
Date(tx.localTimestamp).toLocaleTimeString()} on ${tx.dateOfEvent}</span>
                        ${syncIndicator}
                    </div>
                    <p class="text-base font-semibold mt-1 flex items-center">
                        ${getTransactionIcon(tx.type)}
                        <span class="ml-2 ${amountClass}">${tx.type}
${tx.localAmount.toFixed(2)} ${tx.localCurrency}</span>
                        ${baseAmountDisplay}
                    </p>
                    <p class="text-sm text-gray-700">${tx.rawText}</p>
                </div>
            `;
            feedContainer.insertAdjacentHTML('beforeend', entryHtml);
        });
    }

    function renderSummary() {
        if (memoryLog.length === 0) {
```

```
            document.getElementById('total-in').textContent = '0.00 ' + BASE_CURRENCY;
            document.getElementById('total-out').textContent = '0.00 ' + BASE_CURRENCY;
            document.getElementById('net-balance').textContent = '0.00 ' + BASE_CURRENCY;
            renderSmartReminders([]);
            return;
        }

        let totalIn = 0;
        let totalOut = 0;
        const todayLog = memoryLog.filter(tx => tx.dateOfEvent === getTodayDateString()
&& tx.baseAmount > 0); // Only use fully processed DaaS data

        todayLog.forEach(tx => {
            const amount = tx.baseAmount;
            if (tx.type === 'Sale' || (tx.type === 'Debt' && tx.type.includes('Received'))) {
                totalIn += amount;
            } else if (tx.type === 'Expense' || (tx.type === 'Debt' && tx.type.includes('Given'))) {
                totalOut += amount;
            }
        });

        const netBalance = totalIn - totalOut;

        document.getElementById('total-in').textContent = totalIn.toFixed(2) + `
${BASE_CURRENCY}`;
        document.getElementById('total-out').textContent = totalOut.toFixed(2) + `
${BASE_CURRENCY}`;

        const netBalanceEl = document.getElementById('net-balance');
        netBalanceEl.textContent = netBalance.toFixed(2) + ` ${BASE_CURRENCY}`;

        netBalanceEl.classList.remove('text-green-600', 'text-red-600');
        if (netBalance > 0) {
            netBalanceEl.classList.add('text-green-600');
        } else if (netBalance < 0) {
            netBalanceEl.classList.add('text-red-600');
        }

        renderSmartReminders(todayLog);
    }

    function renderReports() {
        if (memoryLog.length === 0) {
            document.getElementById('monthly-net-balance').textContent = 'No Data';
            return;
        }

        let monthlyIn = 0;
        let monthlyOut = 0;
        const thirtyDaysAgo = Date.now() - (30 * 24 * 60 * 60 * 1000);
```

```
        const monthlyLog = memoryLog.filter(tx =>
            tx.localTimestamp >= thirtyDaysAgo && tx.baseAmount > 0
        );

        monthlyLog.forEach(tx => {
            const amount = tx.baseAmount;
            if (tx.type === 'Sale') {
                monthlyIn += amount;
            } else if (tx.type === 'Expense') {
                monthlyOut += amount;
            }
        });

        const monthlyNet = monthlyIn - monthlyOut;
        const monthlyNetEl = document.getElementById('monthly-net-balance');

        monthlyNetEl.textContent = monthlyNet.toFixed(2) + ` ${BASE_CURRENCY}`;

        monthlyNetEl.classList.remove('text-green-600', 'text-red-600', 'text-gray-700');

        if (monthlyNet > 0) {
            monthlyNetEl.classList.add('text-green-600');
        } else if (monthlyNet < 0) {
            monthlyNetEl.classList.add('text-red-600');
        } else {
             monthlyNetEl.classList.add('text-gray-700');
        }
    }

    function renderSmartReminders(todayLog) {
        const remindersContainer = document.getElementById('smart-reminders');
        remindersContainer.innerHTML = '';

        const totalSales = todayLog.filter(t => t.type === 'Sale').length;
        const totalExpenses = todayLog.filter(t => t.type === 'Expense').length;

        const lastLogTime = memoryLog.length > 0 ? memoryLog[0].localTimestamp : 0;
        const hoursSinceLastLog = (Date.now() - lastLogTime) / (1000 * 60 * 60);

        let hasReminder = false;

        // 1. Imbalance Reminder
        if (totalSales > 2 && totalExpenses === 0) {
            remindersContainer.insertAdjacentHTML('beforeend', `
                <div class="p-3 bg-red-100 border-l-4 border-red-500 rounded-lg text-sm font-
medium text-red-700">
                    🚨 Imbalance Alert: You logged ${totalSales} sales today but NO expenses.
Did you forget materials or transport?
                </div>
            `);
```

```
            hasReminder = true;
        }

        // 2. Logging Gap Reminder
        if (hoursSinceLastLog > 48 && totalSales + totalExpenses === 0) {
            remindersContainer.insertAdjacentHTML('beforeend', `
                <div class="p-3 bg-yellow-100 border-l-4 border-yellow-500 rounded-lg text-sm font-medium text-yellow-700">
                    🗓️ Logging Gap: It's been over 48 hours! Your business is running—log your latest activity now.
                </div>
            `);
            hasReminder = true;
        }

        if (!hasReminder) {
            remindersContainer.insertAdjacentHTML('beforeend', `
                <div class="p-3 bg-green-100 border-l-4 border-green-500 rounded-lg text-sm font-medium text-green-700">
                    ✅ All Systems Clear! Good job maintaining your Memory Log.
                </div>
            `);
        }
    }

    // --- EVENT HANDLERS ---
    document.addEventListener('DOMContentLoaded', () => {
        // PWA Service Worker Registration
        if ('serviceWorker' in navigator) {
            navigator.serviceWorker.register('./service-worker.js')
                .then(reg => console.log('Service Worker Registered.', reg))
                .catch(err => console.error('Service Worker registration failed:', err));
        }

        // Setup Tab Switching
        const tabs = document.querySelectorAll('.tab-button');
        const contents = document.querySelectorAll('.tab-content');
        tabs.forEach(tab => {
            tab.addEventListener('click', () => {
                tabs.forEach(t => t.classList.remove('border-indigo-500', 'text-indigo-400'));
                tabs.forEach(t => t.classList.add('border-transparent', 'hover:border-gray-500'));

                tab.classList.remove('border-transparent', 'hover:border-gray-500');
                tab.classList.add('border-indigo-500', 'text-indigo-400');

                const target = tab.getAttribute('data-tab');
                contents.forEach(content => {
                    content.classList.remove('active');
                    if (content.id === target) {
                        content.classList.add('active');
                        if (target === 'reports') renderReports();
```

```
            }
        });
    });
});
// Ensure initial tab styling is correct
document.querySelector('.tab-button[data-tab="summary"]').classList.remove('border-transparent', 'hover:border-gray-500');
document.querySelector('.tab-button[data-tab="summary"]').classList.add('border-indigo-500', 'text-indigo-400');

// Setup Input Handlers
document.getElementById('raw-input').addEventListener('input', updateParsedView);
document.getElementById('submit-entry').addEventListener('click', handleInputSubmission);

// Setup Settings Handlers
const deleteAllBtn = document.getElementById('delete-all-data');
const confirmBox = document.getElementById('delete-confirm-box');

deleteAllBtn.addEventListener('click', () => {
    deleteAllBtn.classList.add('hidden');
    confirmBox.classList.remove('hidden');
});

document.getElementById('cancel-delete').addEventListener('click', () => {
    confirmBox.classList.add('hidden');
    deleteAllBtn.classList.remove('hidden');
});

document.getElementById('confirm-delete').addEventListener('click', async () => {
    if (db && userId) {
        try {
            // Deletes all documents in the user's public collection
            const q = query(collection(db, getCollectionPath(userId)));
            const snapshot = await getDocs(q);

            // We must delete documents one by one; there is no bulk delete in client SDK
            for (const docSnapshot of snapshot.docs) {
                await deleteDoc(docSnapshot.ref);
            }

            showToast("All cloud data wiped successfully.", 'success');
        } catch (error) {
            console.error("Error wiping data:", error);
            showToast("Failed to wipe data from cloud.", 'error');
        }
    } else {
        showToast("Database not ready or authenticated.", 'error');
    }
```

```
            confirmBox.classList.add('hidden');
            deleteAllBtn.classList.remove('hidden');
            document.querySelector('.tab-button[data-tab="summary"]').click();
        });

        // Set default currency input
        document.getElementById('parsed-currency').value = BASE_CURRENCY;
    });
  </script>
  <!-- Firebase Initialization scripts are now inside the script tag above -->
</body>
</html>
```