<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Time & Reminders Organizer</title>

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

    <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;500;600;700&display=swap" rel="stylesheet">

    <style>

        body {

            font-family: 'Inter', sans-serif;

            background-color: #f3f4f6;

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            padding: 1rem;

        }

        .container {

            max-width: 600px;

            width: 100%;

        }

        /\* Custom styling for the message box \*/

        .message-box {

            position: fixed;

            top: 20px;

            right: 20px;

            padding: 1rem;

            border-radius: 0.5rem;

            background-color: #374151;

            color: white;

            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

            z-index: 1000;

            display: none;

        }

        /\* Loading spinner \*/

        .loader {

            border: 4px solid #f3f3f3;

            border-top: 4px solid #3b82f6;

            border-radius: 50%;

            width: 40px;

            height: 40px;

            animation: spin 1s linear infinite;

        }

        @keyframes spin {

            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

        .done-reminder {

            text-decoration: line-through;

            opacity: 0.6;

        }

        .no-data {

            color: #6b7280;

            text-align: center;

            padding: 2rem;

        }

    </style>

</head>

<body class="bg-gray-100 flex items-center justify-center p-4">

    <!-- Message Box -->

    <div id="message-box" class="message-box"></div>

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

        <div class="flex flex-col items-center mb-6">

            <h1 class="text-3xl md:text-4xl font-bold text-gray-800 mb-2">My Reminders</h1>

            <p id="user-id-display" class="text-sm text-gray-500 font-mono"></p>

        </div>

        <!-- Loader -->

        <div id="loader" class="flex justify-center my-4 hidden">

            <div class="loader"></div>

        </div>

        <!-- Add Reminder Form -->

        <form id="reminder-form" class="flex flex-col md:flex-row gap-4 mb-8">

            <input type="text" id="reminder-text" placeholder="What do you need to do?" required

                   class="flex-1 p-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">

            <input type="datetime-local" id="reminder-date"

                   class="p-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">

            <button type="submit"

                    class="p-3 bg-blue-600 text-white font-semibold rounded-lg shadow-md hover:bg-blue-700 transition-colors">

                Add Reminder

            </button>

        </form>

        <!-- Reminders List -->

        <div id="reminders-list" class="space-y-4">

            <!-- Reminders will be dynamically added here -->

            <p class="no-data">No reminders yet. Add one above!</p>

        </div>

    </div>

    <!-- Firebase SDKs -->

    <script type="module">

        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, doc, getDoc, addDoc, setDoc, updateDoc, deleteDoc, onSnapshot, collection, query, where, getDocs, orderBy, serverTimestamp, setLogLevel } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";

        const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-app-id';

        const firebaseConfig = typeof \_\_firebase\_config !== 'undefined' ? JSON.parse(\_\_firebase\_config) : {};

        const initialAuthToken = typeof \_\_initial\_auth\_token !== 'undefined' ? \_\_initial\_auth\_token : null;

        // Set Firebase logging level to debug

        setLogLevel('debug');

        // UI elements

        const form = document.getElementById('reminder-form');

        const reminderText = document.getElementById('reminder-text');

        const reminderDate = document.getElementById('reminder-date');

        const remindersList = document.getElementById('reminders-list');

        const loader = document.getElementById('loader');

        const userIdDisplay = document.getElementById('user-id-display');

        const messageBox = document.getElementById('message-box');

        let db, auth;

        let userId = null;

        let isAuthReady = false;

        // Message display function

        const showMessage = (message, duration = 3000) => {

            messageBox.textContent = message;

            messageBox.style.display = 'block';

            setTimeout(() => {

                messageBox.style.display = 'none';

            }, duration);

        };

        // Loading spinner functions

        const showLoading = () => { loader.classList.remove('hidden'); };

        const hideLoading = () => { loader.classList.add('hidden'); };

        // Initialize Firebase and authenticate

        try {

            const app = initializeApp(firebaseConfig);

            auth = getAuth(app);

            db = getFirestore(app);

            onAuthStateChanged(auth, async (user) => {

                if (user) {

                    userId = user.uid;

                    userIdDisplay.textContent = `User ID: ${userId}`;

                    isAuthReady = true;

                    console.log("Authenticated with user ID:", userId);

                    startListeningForReminders();

                } else {

                    console.log("No user signed in. Attempting to sign in.");

                    try {

                        if (initialAuthToken) {

                            await signInWithCustomToken(auth, initialAuthToken);

                            console.log("Signed in with custom token.");

                        } else {

                            await signInAnonymously(auth);

                            console.log("Signed in anonymously.");

                        }

                    } catch (error) {

                        console.error("Authentication failed:", error);

                        showMessage("Failed to authenticate. Please try again.");

                    }

                }

            });

        } catch (error) {

            console.error("Firebase initialization failed:", error);

            showMessage("Error setting up the app. Check the console for details.");

        }

        // Function to start the real-time listener for reminders

        const startListeningForReminders = () => {

            if (!db || !userId) {

                console.log("Firestore or user ID not ready. Skipping listener setup.");

                return;

            }

            const remindersCollectionRef = collection(db, `/artifacts/${appId}/users/${userId}/reminders`);

            showLoading();

            // Setup real-time listener

            onSnapshot(remindersCollectionRef, (snapshot) => {

                hideLoading();

                const reminders = [];

                snapshot.forEach((doc) => {

                    reminders.push({ id: doc.id, ...doc.data() });

                });

                renderReminders(reminders);

            }, (error) => {

                hideLoading();

                console.error("Error fetching reminders:", error);

                showMessage("Failed to load reminders. See console for details.");

            });

        };

        // Render reminders to the DOM

        const renderReminders = (reminders) => {

            remindersList.innerHTML = '';

            if (reminders.length === 0) {

                remindersList.innerHTML = `<p class="no-data">No reminders yet. Add one above!</p>`;

                return;

            }

            reminders.forEach((reminder) => {

                const reminderItem = document.createElement('div');

                reminderItem.className = `flex justify-between items-center bg-gray-50 p-4 rounded-lg shadow-sm border border-gray-200 transition-all duration-200 ${reminder.completed ? 'done-reminder' : ''}`;

                reminderItem.setAttribute('data-id', reminder.id);

                const reminderContent = document.createElement('div');

                reminderContent.className = 'flex-1 pr-4';

                const reminderTextElement = document.createElement('p');

                reminderTextElement.className = 'text-lg text-gray-800 font-medium break-words';

                reminderTextElement.textContent = reminder.text;

                const reminderDateElement = document.createElement('p');

                reminderDateElement.className = 'text-sm text-gray-500 mt-1';

                if (reminder.dueDate) {

                    const date = new Date(reminder.dueDate);

                    reminderDateElement.textContent = `Due: ${date.toLocaleString()}`;

                }

                reminderContent.appendChild(reminderTextElement);

                if (reminder.dueDate) {

                    reminderContent.appendChild(reminderDateElement);

                }

                const actionsContainer = document.createElement('div');

                actionsContainer.className = 'flex items-center space-x-2';

                const completeButton = document.createElement('button');

                completeButton.className = `p-2 rounded-full text-white ${reminder.completed ? 'bg-green-600' : 'bg-gray-400 hover:bg-gray-500'} transition-colors`;

                completeButton.innerHTML = `<svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5" viewBox="0 0 20 20" fill="currentColor"><path fill-rule="evenodd" d="M16.707 5.293a1 1 0 010 1.414l-8 8a1 1 0 01-1.414 0l-4-4a1 1 0 011.414-1.414L8 12.586l7.293-7.293a1 1 0 011.414 0z" clip-rule="evenodd" /></svg>`;

                completeButton.onclick = () => toggleReminderStatus(reminder.id, reminder.completed);

                const deleteButton = document.createElement('button');

                deleteButton.className = 'p-2 rounded-full bg-red-500 text-white hover:bg-red-600 transition-colors';

                deleteButton.innerHTML = `<svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5" viewBox="0 0 20 20" fill="currentColor"><path fill-rule="evenodd" d="M9 2a1 1 0 00-.894.553L7.382 4H4a1 1 0 000 2v10a2 2 0 002 2h8a2 2 0 002-2V6a1 1 0 100-2h-3.382l-.724-1.447A1 1 0 0011 2H9zM7 8a1 1 0 012 0v6a1 1 0 11-2 0V8zm6 0a1 1 0 011 1v5a1 1 0 11-2 0V9a1 1 0 011-1z" clip-rule="evenodd" /></svg>`;

                deleteButton.onclick = () => deleteReminder(reminder.id);

                actionsContainer.appendChild(completeButton);

                actionsContainer.appendChild(deleteButton);

                reminderItem.appendChild(reminderContent);

                reminderItem.appendChild(actionsContainer);

                remindersList.appendChild(reminderItem);

            });

        };

        // Add a new reminder

        const addReminder = async (e) => {

            e.preventDefault();

            const text = reminderText.value.trim();

            const date = reminderDate.value;

            if (!text || !db || !userId) {

                showMessage("Please enter a reminder.");

                return;

            }

            const remindersCollectionRef = collection(db, `/artifacts/${appId}/users/${userId}/reminders`);

            try {

                showLoading();

                const newReminder = {

                    text,

                    completed: false,

                    dueDate: date || null,

                    createdAt: serverTimestamp()

                };

                await addDoc(remindersCollectionRef, newReminder);

                showMessage("Reminder added successfully!");

                reminderText.value = '';

                reminderDate.value = '';

            } catch (error) {

                console.error("Error adding document: ", error);

                showMessage("Failed to add reminder. Please try again.");

            } finally {

                hideLoading();

            }

        };

        // Toggle reminder completion status

        const toggleReminderStatus = async (id, currentStatus) => {

            if (!db || !userId) {

                showMessage("App not ready. Please wait.");

                return;

            }

            const reminderDocRef = doc(db, `/artifacts/${appId}/users/${userId}/reminders/${id}`);

            try {

                await updateDoc(reminderDocRef, {

                    completed: !currentStatus

                });

                showMessage(currentStatus ? "Reminder marked incomplete." : "Reminder completed!");

            } catch (error) {

                console.error("Error updating document: ", error);

                showMessage("Failed to update reminder status.");

            }

        };

        // Delete a reminder

        const deleteReminder = async (id) => {

            if (!db || !userId) {

                showMessage("App not ready. Please wait.");

                return;

            }

            const reminderDocRef = doc(db, `/artifacts/${appId}/users/${userId}/reminders/${id}`);

            try {

                await deleteDoc(reminderDocRef);

                showMessage("Reminder deleted successfully.");

            } catch (error) {

                console.error("Error deleting document: ", error);

                showMessage("Failed to delete reminder.");

            }

        };

        // Event listener for form submission

        form.addEventListener('submit', addReminder);

    </script>

</body>

</html>