CASE STUDY 04 OF 07 | CONNECT APP

Presence System& Online Status

Knowing who is available, busy, or away before you reach out

#Presence #WebSocket #RealTime #UX

Overview

Before you send someone a message or start a video call, it helps to know if they are actually there. A presence system that tells you who is online, who is in a meeting, and who is away is one of those features that people take for granted when it works well and notice immediately when it does not. This case study covers how the presence and status system was built in Connect.

Project Snapshot

Project	Connect — Workplace Communication Platform
Feature Area	Presence System and Online Status
Role	Senior Fullstack Engineer
Update Speed	Status changes propagated under 2 seconds organization-wide
Status Types	Online, Offline, Away, Busy, In a Call, Custom
Outcome	Accurate, real-time presence visible across all surfaces in the app

The Problem

Without presence information, every message you send is a shot in the dark. You do not know if the other person is at their desk and will reply in seconds, or if they stepped away and will not see it for hours. In a busy organization, this ambiguity leads to unnecessary follow-ups, missed urgency, and a general anxiety about whether communication is landing.

The presence system needed to solve this in a way that was accurate without being creepy, visible without being intrusive, and fast enough that the status you see reflects reality right now rather than five minutes ago. It also needed to work across the web app, mobile clients, and desktop app simultaneously.

What Was Built

Status Types

Connect surfaces six presence states for each user:

- Online: the user has the app open and is actively interacting with it
- Away: the app is open but the user has not interacted for a configurable idle period
- Busy: manually set by the user to indicate they should not be interrupted
- In a Call: automatically set when the user joins a video call or audio call
- Offline: the user has no active app sessions
- Custom: a user-defined status with a message and optional emoji (for example, 'Focusing until 3pm')

Where Presence Is Shown

Status indicators appear consistently across every surface where people interact: next to names in the contact list, on user avatars in channel member lists, in DM conversation headers, in group participant lists, and in user profile popups. The indicator is always current, not cached from last page load.

Automatic Status Transitions

Certain status changes happen automatically without user intervention. Going idle after a period of no activity switches status from Online to Away. Joining a video call switches status to In a Call. Closing the app or losing connectivity switches status to Offline. These automatic transitions ensure the status others see is actually informative rather than perpetually stuck on Online.

Technical Architecture

Heartbeat-Based Presence Each connected client sends a lightweight heartbeat ping every 30 seconds over the WebSocket connection. The server tracks the last heartbeat time per session. If no heartbeat is received within a threshold, the session is marked as expired and the user's presence is updated accordingly.	**Multi-Session Awareness** A user might have the web app, the mobile app, and the desktop app open simultaneously. Presence is aggregated across all active sessions. The user is Online if any session is active, and Offline only when all sessions have expired or disconnected.
Presence Fan-Out When a user's status changes, the update fans out to all organization members who are currently online. This is done through the Redis pub-sub layer, which distributes the update to all server instances handling connections for those members.	**Idle Detection** Idle detection runs client-side, monitoring mouse movement, keyboard events, and scroll activity. After a configurable idle timeout, the client pushes an Away status update to the server without waiting for the next heartbeat cycle, keeping the transition fast.
In-Call Status Injection The video calling system directly updates the presence service when a user joins or leaves a call. This allows the In a Call status to appear	**Status Caching** On app load, presence state for all organization members is fetched in a single batch request and cached locally. Subsequent updates arrive as real-time events and are applied as patches to the

instantly when a call starts, independent of the heartbeat cycle.	local cache, so the full list never needs to be refetched.

Key Challenges

Presence at Scale Across a Large Organization

For an organization with hundreds of members, fanning out every single presence change to every connected member produces a high volume of real-time events. The fan-out was optimized by batching rapid successive status changes (for example, a user briefly going Away and immediately back to Online) into a single update, and by only sending updates to clients that are currently viewing a surface where the updated user's status is visible.

Accurate Offline Detection

The hardest state to detect correctly is Offline. A user might close their laptop (no graceful disconnect), lose network connectivity (WebSocket drops without explicit close), or open the app then immediately lock their screen. The heartbeat approach handles all of these: if no heartbeat arrives within the window, the user is marked Offline regardless of whether they sent an explicit disconnect signal.

Custom Status Expiry

Custom status messages can optionally include an expiry time, such as 'Out of office until Monday'. The system needs to automatically clear those statuses when the expiry time passes, even if the user is offline at that time. A lightweight scheduled job sweeps for expired custom statuses and resets them, without requiring user action.

Outcome

The presence system gave the organization a shared awareness layer that made communication more considerate and efficient. Users knew whether to expect an immediate reply or to wait before following up. The In a Call status reduced interruptions during video calls noticeably. Custom statuses let people communicate their context without needing to send a message about it. Presence changes were consistently fast enough that the status felt live rather than lagged.

Engineering Highlights

Heartbeat-based presence handling graceful and non-graceful disconnects equally well
Multi-session aggregation keeping status accurate across web, mobile, and desktop
Client-side idle detection for fast Away transitions independent of the heartbeat cycle
Batched fan-out reducing event volume in large organizations without sacrificing accuracy