CASE STUDY 06 OF 07 | CONNECT APP

Chat EngagementFeatures

Emoji reactions, @mentions, threaded replies and message interactions

#UX #Threads #Emoji #Mentions #Reactions

Overview

The mechanics of how people interact with messages in a chat platform shape the entire tone of communication. Being able to react to a message with an emoji instead of sending 'Got it!' keeps channels clean. Threading a reply to a specific message keeps long conversations organized. Mentioning someone by name ensures they do not miss something meant for them. This case study covers the design and engineering of the engagement layer in Connect.

Project Snapshot

Project	Connect — Workplace Communication Platform
Feature Area	Engagement: Emoji, Reactions, Mentions, Threads, Replies
Role	Senior Fullstack Engineer
Emoji Set	Full Unicode emoji set plus custom organization emoji
Thread Model	Inline threaded replies branching from any message
Outcome	Rich, expressive interaction layer that keeps channels focused and organized

The Problem

Without a rich interaction layer, chat channels become either too noisy or too silent. Noisy because every acknowledgment becomes a full message ('Thanks!', 'Sounds good', 'Makes sense'). Or silent because people do not want to add noise, so important messages get no acknowledgment at all. Neither extreme is healthy for a team communication tool.

Threads solve a different problem: keeping focused side conversations from burying the main conversation. Without threading, a ten-message exchange about one specific point floods the channel for everyone following the main topic. Threading lets that conversation happen without interrupting the primary stream.

What Was Built

Emoji Reactions

Hovering over any message reveals a reaction button. Clicking it opens the emoji picker, from which the user can choose any emoji. The reaction appears on the message, grouped by emoji type with a count showing how many people reacted with the same emoji. Clicking an existing reaction adds your own, clicking it again removes it. Reactions are visible to everyone in the conversation and update in real time.

Organizations can define custom emoji by uploading an image and assigning a shortcode. Custom emoji appear in the picker alongside the full Unicode set and work identically in reactions and in message text.

@Mentions

Typing @ in the message input triggers an autocomplete dropdown showing organization members and channel names. Selecting a person mentions them by name, highlighted in the message. Mentioned users receive a notification that cuts through their regular notification settings, meaning a mention will alert them even if they have channel notifications muted. A dedicated mentions view aggregates every message where the user has been mentioned across all conversations, so nothing important gets buried.

Message Replies

Any message can be replied to directly, creating a visual link between the reply and the original message. Replies in the main conversation keep the context of what they are responding to visible, which prevents the confusion of 'wait, who was that replying to?' that plagues flat chat streams.

Threaded Conversations

For longer discussions, replies can be taken into a thread. A thread opens as a panel alongside the main conversation, keeping the discussion contained without interrupting the main channel stream. Thread participants receive notifications about new thread messages. The main conversation shows a summary of thread activity, the number of replies and the avatars of participants, without displaying every thread message inline.

Message Actions

Every message exposes a set of actions on hover: react, reply, start thread, pin, copy link, and for own messages, edit or delete. Pinned messages are collected in a per-conversation pins panel, making important reference information findable without scrolling.

Technical Architecture

Reaction Data Model Reactions are stored as a many-to-many relationship between users and messages, with the emoji identifier as an attribute. Aggregated counts are computed at query time with a simple GROUP BY and cached per message for frequently viewed conversations.	**Mention Parsing** The message input uses a rich text editor that recognizes @-prefixed tokens and resolves them to user IDs at compose time. The stored message format includes structured mention entities alongside the display text, making mention extraction reliable regardless of name changes.
Thread Data Model	**Real-Time Reaction Updates**

Threads are stored as child conversations linked to a parent message ID. Each thread has its own message history and participant list. The thread summary (reply count, participant avatars, last reply time) is maintained as a denormalized record updated on each new reply.	Reaction changes are broadcast via the real-time layer to all conversation members. The client merges incoming reaction updates into its local message cache without re-rendering the entire message list, keeping updates fast and visually smooth.
Mention Notification Routing Mentions trigger a high-priority notification path that bypasses channel mute settings. The notification service checks the mentioned user's current presence and routes accordingly: WebSocket for online users, push notification for mobile, email digest for offline users.	**Custom Emoji Storage** Custom emoji images are stored in object storage and served via CDN with long cache TTLs. The emoji registry (shortcode to URL mapping) is loaded at app start and updated when new emoji are added, with a client-side cache invalidation signal pushed via the real-time layer.

Key Challenges

Emoji Picker Performance with a Full Unicode Set

The full Unicode emoji set contains over 3,500 emoji across dozens of categories. Rendering all of them in a picker without lag required virtualization: only the emoji currently in the viewport are rendered as DOM nodes, with the rest represented as empty placeholder elements. Search across the full set uses a pre-built index that returns results in milliseconds.

Thread Summary Staleness

The thread summary shown in the main conversation (reply count, last reply time, participant avatars) needs to stay accurate without querying the database on every message load. The summary is kept as a denormalized record updated atomically with each new thread reply. On the rare occasion that a reply is deleted, a reconciliation job verifies and corrects the summary to prevent permanent inaccuracy.

Mention Resolution Across Name Changes

If a user changes their display name after being mentioned in a message, the stored mention should ideally still link to the right person. Mentions are stored by user ID internally, not by display name. The display name rendered in the message is resolved at read time from the current user profile, so name changes are reflected automatically in old messages while still linking to the correct person.

Outcome

The engagement layer measurably changed the way teams communicated in Connect. Channels stayed cleaner because acknowledgments moved to reactions rather than messages. Threaded conversations kept complex discussions from dominating the main channel stream. Mentions gave people confidence that important messages would reach the right person. And the expressive range of emoji made the communication feel more human, which matters for a tool that people use for many hours every day.

Engineering Highlights

Virtualized emoji picker handling 3,500+ emoji without render lag
User ID-based mention storage ensuring correct resolution after name changes
Denormalized thread summaries keeping the main conversation view fast at any thread depth
High-priority mention notification path bypassing mute settings for critical messages