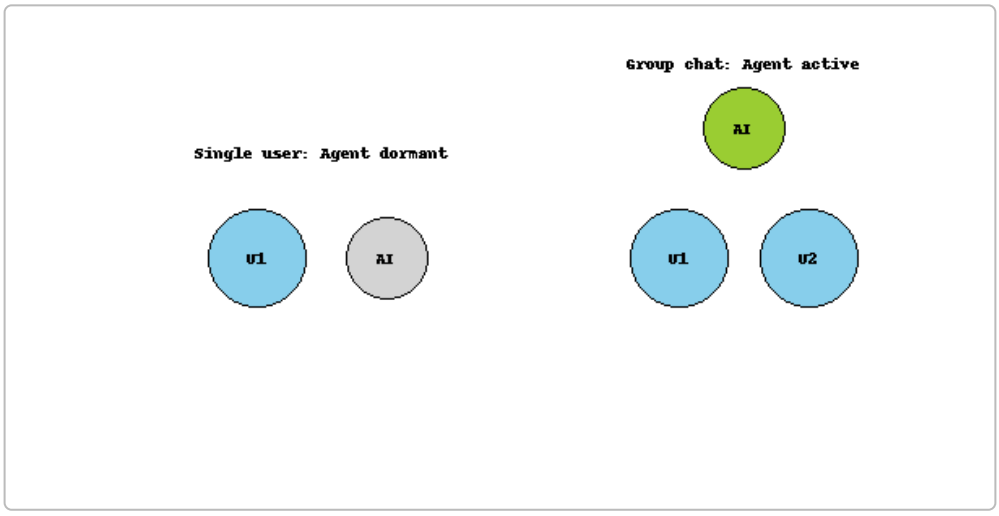
ChatGPT

Designing a Context-Aware Group Chat Agent for Collaboration, Gaming, and Mediation



Conceptual illustration: the agent remains dormant in one-on-one chats (left), but becomes active once multiple participants are present (right). This ensures it behaves like a group member rather than a personal assistant.

In multi-user chats, a chatbot can function as an intelligent team participant rather than a simple one-on-one assistant. We envision an AI agent that *only activates when at least two people are in the conversation*, providing support in group contexts. Such an agent should understand its role within a team, knowing **when to contribute and when to stay back**, and adapt to the social dynamics of each scenario [1]. Below, we explore this agent's design across three contexts – collaborative productivity, gaming, and conflict mediation – detailing (1) its activation conditions, (2) user experience on web, mobile, and integrated platforms, (3) how it signals its presence, (4) customization controls for users, and (5) how its behavior and interface differ by context.

Collaborative Productivity

In a workplace or project team chat, the agent acts as a virtual facilitator to boost productivity and organization without disrupting the human workflow. Studies show that groups appreciate an AI "team member" for brainstorming and planning – it can keep the conversation moving and offer ideas – **as long as it doesn't dominate the discussion** [2]. Design focuses on being helpful and context-aware in meetings, project channels, or collaborative discussions.

1. Activation Conditions: The agent should enter the conversation only under clear, relevant triggers to avoid spamming busy channels. Key conditions include:

- **Direct Mention or Command:** It activates when someone addresses it or uses a slash command (e.g. `@Assistant, summarize this`), ensuring it only speaks when explicitly invited [3]. In reactive mode, the agent strictly waits for such prompts before responding [2].
- **Task/Keyword Triggers:** If team members use certain keywords (e.g. "next steps?", "deadline") or phrases indicating help is needed ("I'm stuck," "Any ideas?"), the agent can gently offer assistance. For example, seeing *"What are the next steps?"* might trigger the agent to draft an action list.
- **Inactivity or Deadlock:** During meetings or brainstorming, a long pause from the group could prompt the agent to intervene. For instance, if discussion lulls for a few minutes, it might ask *"Need a summary or suggestions to proceed?"* [3]. Likewise, if back-and-forth conversation stalls without resolution, it can surface points that haven't been addressed to rekindle progress.
- **Contextual Cues:** The agent may also monitor context like scheduled meeting times or project milestones. At the end of a meeting, if multiple people are present, it could automatically prompt *"Shall I compile the decisions and tasks from today's meeting?"* Similarly, if the due date for a discussed task passes with two or more team members chatting, it might remind them of that deadline.

2. User Experience & Interface (Web, Mobile, Embedded): Across platforms, the agent's presence should feel like a natural part of the chat, with minimal intrusion. On **web/desktop**, it appears as another user in the channel – e.g. a chat message with the bot's name and avatar (perhaps labeled "Assistant"). Its messages might use formatting to stand out (such as italics or a subtle highlight) when providing summaries or lists. The agent could also utilize threads or side panels for lengthy output to keep the main chat tidy. For instance, in Slack the bot might reply in a thread attached to the prompt, especially if it's sharing a detailed project plan, to avoid cluttering the channel [4]. On **mobile**, screen space is limited, so the agent might deliver assistance via compact UI elements. It could send a brief message (e.g. *"3 tasks noted – tap to expand"*), which users can tap to see full details. Mobile push notifications from the agent should be concise (e.g. *"Project Bot: 2 new tasks assigned"*) to avoid overwhelming users on the go. For **integrations like Slack or Microsoft Teams**, the agent can leverage platform-specific UI components. In Slack, it might use app **modals and buttons** – for example, posting an interactive checklist that team members can click to assign or complete tasks. It can also use ephemeral messages (visible only to specific users) for personal reminders. On platforms like **Discord** (less common for workplace use, but similar concept), it would post in the text channel or via a bot DM as needed. Overall, the UX pattern is to be *in the flow of work*: on desktop, perhaps a sidebar "AI assistant" panel for private interaction, and in-channel messages for group-visible contributions. The design should be consistent – the agent's messages use clear language, maybe prefaced with an agent icon, and on all platforms it should respect dark mode, notifications settings, etc., just like a human participant's messages.

3. Communicating Presence & Role: In productivity chats, the agent's presence is ideally **subtle yet transparent**. It might join a channel silently (no big splash announcing it's active) to maintain an "ambient" presence until needed. However, the first time it activates in a channel, a brief introduction can clarify its role: e.g. a one-line system message, *"AssistantBot is now in this channel to help with scheduling and notes (will only chime in when asked)."* This sets expectations without spamming. Thereafter, the agent typically speaks **explicitly only when triggered** – for instance, replying with "Sure, let me compile a summary..." when @mentioned. Its contributions should be framed as help, not commands. The bot might use first person plural tone ("We have 3 pending tasks from this chat") to sound collaborative. Importantly, it should avoid pretending to be human or hiding that it's a bot – clarity builds trust. An ambient presence might also include visual cues: for example, if the chat platform shows typing indicators, the agent could display

"Assistant is typing…" when preparing a long response, indicating it's actively working on a request. Outside of messages, the agent could appear in channel member lists (with a label like "BOT") so users know it's available. It might also show its **status** (online/offline) only when at least two users are present, reinforcing that it "wakes up" for group settings. In general, the agent stays **politely in the background** until invoked, then clearly indicates when it's speaking on record.

4. Customizations & Control Options: Work teams must be able to fine-tune the agent's behavior to suit their working style. Possible controls include:

- **Activation Mode:** Toggle between a *"Proactive" mode and a "Reactive" mode*. In proactive mode, the agent has more autonomy to offer unprompted suggestions (e.g. jumping in after long silence or when it detects a decision point), whereas in reactive mode it *only replies when directly addressed*. Users in studies often preferred the reactive setting to avoid interruptions [5] [6], so reactive could be the default with proactive as an opt-in.
- **Trigger Keywords & Sensitivity:** A settings panel can let users define keywords or phrases that summon the agent. For example, the team can add custom triggers like "@bot help us brainstorm." They can also adjust sensitivity for inactivity triggers (maybe configure "alert if no one responds for 5 minutes"). The agent Koala's design, for instance, exposed a *"contribution threshold"* setting – a confidence score the AI's answer must exceed before posting to ensure quality [7]. This prevents low-value chatter by the bot.
- **Response Frequency & Timing:** Controls to rate-limit the agent's interventions can help it match team pace. Users might set "at most 1 suggestion per 10 messages" or instruct the agent to *wait until humans have exchanged a few messages before chiming in* [8]. For instance, one might enable a rule that the bot shouldn't be the first to speak in a new discussion thread (to avoid undue influence [8]).
- **Content Style and Tone:** The agent's tone can be customized to fit team culture. Options might include formality (formal vs. casual language), brevity (concise bullet lists vs. detailed paragraphs), or even creativity level for idea generation. In brainstorming, users wanted to dial the bot's creativity up or down (from "conventional" to "crazy" ideas) [9], which could be implemented via a simple setting (e.g. a slider for how outside-the-box suggestions should be). Similarly, a toggle for including emojis or humor might exist – many professional teams prefer a straightforward style (in one study, users *"disliked emoticons and humor" from a team chatbot discussing serious topics* [10]).
- **Visibility and Location:** Users or admins can choose whether the agent posts in the main chat or in threads. For example, a **"respond in thread"** option could contain lengthy bot messages to a thread attached to the original question [11]. If enabled, the agent's answer appears as a threaded reply (perhaps with a summary snippet in the main chat), reducing noise. The team can experiment and configure this per use-case, since what reduces distraction in theory might also hide the bot's contributions too much [4]. The agent's responses could also be set to private for certain triggers – e.g. if one person asks the bot to draft a message, maybe the bot DM's that person the draft rather than posting publicly.
- **Permission & Admin Controls:** Not everyone in a group may have equal rights to tweak the bot. The system can allow workspace admins or channel owners to set global defaults (like turning the agent on/off in a channel, or restricting proactive mode). For instance, an admin might disable the agent in a sensitive channel (HR or leadership discussions) where its input isn't wanted. In Discord's design for the Clyde bot (gaming context), they *"provided a way to disable Clyde if the server admin did not want people using the feature"* [12] – a similar approach applies in enterprise settings. Additionally, the group might require consensus to change certain settings (to avoid one person silently making the

bot more intrusive for everyone). A clear "Agent Settings" panel accessible to all members (or just moderators) can display what rules are in place, ensuring transparency.

5. Behavior & Interface Differences (Productivity Context): In project management and collaborative work, the agent behaves *professionally, helpfully, and with context awareness of work tasks*. This differs from other contexts in a few ways. **Tone and personality**: The productivity agent typically uses a polite, formal tone (more like an office assistant than a jokey friend). It avoids slang and keeps messages succinct and on-topic. Users have indicated they prefer a human-like but *serious* persona for work bots [10] – meaning the bot can be friendly but should not use excessive jokes or emojis that might undermine its credibility. **Focus on structure**: In this context, the agent often provides structured outputs – e.g. task lists, meeting notes, or decision summaries – that help the team stay organized. It might automatically format its responses in bullet points or checklists for clarity. In contrast, a gaming bot might be freer in format. **Real-time vs asynchronous**: Productivity chats can be asynchronous (not everyone reads/replies instantly). The agent here might do things like *daily summaries or highlights* of what multiple people discussed (if someone was away, they could ask the agent for a catch-up summary). Slack's own AI features, for example, let users summarize channel discussions with a click [13]. Our agent could proactively offer a summary if a teammate returns after being offline, although likely only upon request to avoid noise. **Integration with tools**: A productivity-focused agent might integrate with calendars, to-do apps, or documents. For instance, if two people start scheduling a meeting in chat, the agent can notice and offer to create a calendar event (with a button to confirm). These integrations would appear in the interface as interactive cards or links (e.g. *"Meeting scheduled for 3 PM on Tue – added to Calendar [14] [15]"*). **Opt-in by default**: Generally, in work contexts the team might prefer the agent to be *opt-in (passive)* initially, only stepping forward when asked, to respect human autonomy [6]. This is slightly different from, say, a conflict mediation bot which might need to intervene on its own. By being an on-demand resource, the productivity agent supports the team without ever undermining the humans' sense of control over the workflow.

Gaming (Real-Time Team Play)

In gaming contexts (e.g. a Discord server for a multiplayer game or a guild chat), the agent serves as a **team sidekick** and coordinator, enhancing fun and strategy. This context is often fast-paced and informal, so the agent needs to deliver help quickly and playfully. Discord's experimental AI, *Clyde*, is a good example – it was designed to *"engage users in meaningful conversations"* while also keeping things light, even cracking jokes [16] [17]. The agent in a gaming scenario might help with game info, coordination of group actions, or just entertainment, but should never interrupt critical player communications (e.g. during intense gameplay moments).

1. Activation Conditions: The agent activates in a game chat when its input would be helpful or when players explicitly call on it. Likely triggers include:

- **Direct Invocations:** Players can summon the bot with a mention or command. For example, typing `@GameBot` or a shorthand like `/strategy` triggers it. In Discord, Clyde was designed such that users *"simply need to tag @Clyde in any channel to start chatting"* [18]. This direct call ensures the agent contributes only when someone wants it to. Common commands might be things like `@GameBot hint` (for a hint on the current puzzle/quest) or `@GameBot stats` (to retrieve team or enemy stats).
- **Game Event Triggers:** The agent could be integrated with game APIs or triggers. For instance, when a boss fight starts (and at least two guild members are in the chat or voice channel), the bot might

automatically post the boss's known weaknesses or the team's next objective. If a scheduled raid time arrives and multiple players are present in Discord, the bot could announce, *"Raid starting now – good luck team!"*. Similarly, if the team fails a level repeatedly, the bot might offer a tip unprompted (optionally): *"Noticed a struggle with Level 5. Want a hint?"*. These context-based triggers should be used sparingly and likely be configurable, as not all players want unsolicited advice mid-game.

- **Keywords from Chat:** The agent can listen for certain phrases among the group that suggest help is needed. For example, if one player types *"How do we solve this riddle?"* or *"I forget what the map says,"* the agent can respond with the relevant info (puzzle hints, map link, etc.). If someone asks *"Anybody know a good strategy for this boss?"* and no human answers promptly, the bot can jump in with a strategy guide snippet. Essentially, **question detection** can act as a trigger: when a question is posed to the group (and contains game-related terms the bot recognizes), the agent offers an answer, acting like an always-available guide.
- **Idle Social Triggers:** In more relaxed, social gaming chats, if conversation lags, the agent might throw in fun content to keep the group engaged (if enabled). For instance, after a period of silence, it could start a quick game trivia quiz or post a meme relevant to the game, but only in casual contexts. This keeps the chat lively and can strengthen the community feel. Of course, this should only happen in channels flagged for that level of playfulness (perhaps an "off-topic" channel for the guild).

2. User Experience & Interface (Web, Mobile, Integration): The agent's interface in gaming should be streamlined to avoid distracting from gameplay. On **desktop/web**, many gamers use Discord or similar overlays while playing. The bot would appear in the text channel feed just like any user. Its messages might include rich media – for example, an image of a game map or an embedded link to a wiki – to quickly convey info. Because timing is critical, the agent might use shorter messages or even one-word alerts (like "Incoming!" if an enemy event starts) accompanied by an emoji or color to grab attention. On **mobile**, where many gamers chat via apps, the bot's output should be minimalistic. It could use push notifications for important alerts (e.g. *"Match found – join now!"* if integrated with the game matchmaking). Within the app, a collapsible card UI could present info like quest details without flooding the small screen. For instance, tapping a bot message might expand a list of item drop rates or team assignments. In an **embedded platform like Discord**, which is common for gaming, the bot can use advanced Discord features: it might create threads automatically for side conversations (Discord's Clyde did this – after the first response in the main chat, *"Clyde would create a thread for you to continue the conversation without clogging up the main chat"* [19]). This means if players start bantering with the bot or asking it a series of questions, those messages move to a thread so as not to spam other members who are only watching main chat for callouts. The agent could also join voice channels in listen mode: for example, if players are on voice chat, the agent might **display an overlay** of real-time info (some games allow bots to post updates in an overlay or speak via TTS). Imagine a scenario where in a voice channel the agent detects the game state (like health low) and *quietly posts a text-to-speech message:* "Healing potion recommended now." The interface on voice could just be audio cues or subtle ping sounds, configurable by users. Overall, the design emphasizes quick consumption – the bot often uses **preformatted data** (like tables of stats, countdown timers, or progress bars) in the chat to convey info at a glance. For example, the bot might post: *"Team HP: [█████▒▒] 60% – stay cautious!"* as a single-line update rather than a verbose sentence.

3. Communicating Presence & Role: In gaming chats, the bot's persona can be more **overt and playful**. From the moment it's enabled on a server, it should be clear it's there to help and entertain. The first time it appears, it might introduce itself in a fun way – e.g. *"Hi, I'm GameBot, your AI teammate! Tag me anytime for tips or a joke ."* This could be an ephemeral message or pop-up that only new users/admins see, similar to how Discord's Clyde presented terms of use the first time it was invoked [20] . After that, the bot's presence is indicated by its active status and occasional friendly interjections. The agent typically communicates

explicitly when triggered (e.g., responding with an answer, making a joke when asked), but it can have an ambient feel in how it integrates with game events. For example, players might perceive it as part of the game environment if it automatically calls out events. To avoid confusion with human players, the bot should have a distinct name and avatar (often a mascot or game-themed icon). Clyde, for example, was given a unique avatar and even **animated expressions** (happy, normal, error states) to make its presence more relatable [21]. Our agent could similarly change its avatar or use emoticons in messages to convey *mood* (e.g. a when joking, or a when giving a tip about an objective). Importantly, the bot should not overpower human conversation; it acts more like an NPC side character. When it needs to share a lot of info, it might **signal its role** by formatting – e.g. using a different text color or block quote style for its big tips, so players instantly recognize "that's the bot talking". The presence is explicit in that everyone knows these messages are from the AI, but in spirit the bot is *one of the gang*. It might use inclusive language: *"We got this! Try flanking from the left."* This can boost morale and make the agent feel like a team mascot/coach. If the bot is not needed, it stays out of the way (possibly indicated by a quiet "listening" icon somewhere). Game server admins may choose to confine the bot to certain channels – e.g. only a #strategy channel – so its presence is context-specific. In summary, the agent's presence in gaming is **clearly signaled but friendly**: everyone sees it's there as a bot, and it actively acknowledges its helper role (maybe even with a bit of personality or backstory to amuse users).

4. Customizations & Controls: Gaming communities vary greatly, so the agent must be highly configurable to fit the server's style and the game's needs. Some customization options:

- **Enable/Disable and Channel Permissions:** Just as Discord allowed admins to turn Clyde on or off per server [12], our agent can be enabled only in desired channels. For example, an admin might enable the bot in #general-chat and #strategy, but keep it out of #matchmaking if they prefer no interference there. They could also restrict usage so only certain roles (like moderators or raid leaders) can call the bot with advanced commands (to prevent spam during crucial moments).
- **Trigger Settings:** Users can tailor what automatically triggers the bot. In a very competitive team, they might *disable all automatic tips*, opting to only use manual commands, so that the bot never distracts unless asked. In a casual guild, they might turn on features like automatic daily trivia or greeting new members. Perhaps a **"Hint Frequency"** slider can adjust how readily the bot offers unsolicited game hints. If set to low, the bot will remain quiet unless explicitly asked; if high, it might proactively drop hints if it detects players stuck on a puzzle for a long time.
- **Content and Persona:** A fun part of gaming bots is personalization. The agent could let the group **customize its persona or dialect**. For instance, users might choose a "wise mentor" personality vs. a "sarcastic companion" style, changing the tone of its messages. The Discord team experimented with letting users *change Clyde's personality, name, avatar, and even share custom personas across servers* [22] [23]. In our design, an options menu might allow editing the bot's name (maybe your guild calls it "Navigator"), selecting an avatar (perhaps a game character icon), and choosing from preset personalities (e.g. *"Cheerleader"* vs *"Tactical Commander"*). This control makes the agent feel like it belongs to the community.
- **Game Integrations:** The bot's settings can allow linking to game accounts or APIs. For example, connecting the bot to your game's API key so it can fetch live stats. Users can toggle which data it announces – one might turn on *kill/death ratio announcements* for a shooter game, but turn off *inventory suggestions*. If the agent supports multiple games, an admin could configure profiles for each game (with different trigger words and features).
- **Mediation of Bot Chat:** Given that game chats can get busy, another control is *how the bot's output is delivered*. As mentioned, automatically spawning a thread for extended bot conversations is one

method [19]. A server might enable "Bot threads" so that if two users start bantering with the bot, it will fork that into a thread after, say, 3 messages, to keep main chat clear. Alternatively, an admin could disable that if they prefer everything in one channel.

- **Opt-in Fun Features:** The agent could have mini-games or humor modules that are opt-in. For instance, a **"meme mode"** where if enabled, the bot responds with a meme image relevant to the chat (like a celebratory GIF when the team wins a round). These are off by default and only added if the group wants an extra layer of entertainment.
- **Safety and Moderation:** Although in gaming the bot is mostly a helper, it might double as a moderator assistant (overlap with conflict mediation context). Controls can allow it to flag toxic language or cheating discussions. For example, a filter setting might alert admins (or gently remind users) if someone's getting too heated or using slurs, leveraging Discord's AutoMod AI. This is related to conflict mediation, but in gaming, it ensures fun stays friendly. Admins can set the strictness of this filter or disable it entirely for a more unfiltered experience.

5. Behavior & Interface Differences (Gaming Context): In a real-time gaming environment, the agent's behavior is distinct from a workplace bot or mediator in several ways. **Pacing and brevity**: The gaming agent operates in real-time; its responses must be near-instant and concise. During a game session, there's no time for a long-winded explanation – the agent might output a one-liner strategy or use abbreviations common to the game. This contrasts with a productivity agent that might write a detailed multi-sentence summary. **Tone and personality**: The gaming agent is *more informal and witty*. It can use humor, memes, or game slang to bond with users. For example, if a player asks for a tip, the bot might reply with a bit of flair: *"Pro tip: flank the boss .*" Keeping things light can make the bot endearing. In fact, part of Clyde's appeal was its ability to *"keep the chat lively with humor"* [17]. In a productivity context that wouldn't be appropriate, but here it's a feature. **Role as a teammate vs tool**: In gaming, users might anthropomorphize the bot more, treating it like an extra party member. The interface could even show the bot in the player roster if the game allows (some games list bots as players). The agent might have a **persistent presence** during gameplay, whereas in productivity one might only consult it occasionally. For example, the bot could have a constant status message like "GameBot (online) – monitoring game state" which is very context-specific. **Interface elements**: The gaming bot might utilize visuals more heavily – posting images, using ASCII art for fun, or reacting with emojis automatically (like reacting with a trophy emoji when the team wins). In Slack or work chat, the bot wouldn't spam funny GIFs, but in gaming that could be delightful. **Opt-in vs passive**: Interestingly, a gaming agent might be allowed a bit more *proactivity* than a work agent. Gamers might appreciate the bot jumping in at the right moment (like a clutch tip in a boss fight). As such, the threshold for unprompted action might be set a bit lower in gaming context (depending on user preference). Because it's understood as part of the game environment, an unexpected interjection from the bot ("Shield up now!") might be seen as helpful rather than intrusive – provided it's well-timed. **Adaptation to context**: The agent's behavior can also change with game phase. During intense competitive play, it stays quiet unless absolutely necessary (maybe only critical alerts). During downtime or lobby chat, it can be chattier and initiate mini-games or discussions. This dynamic shifting is less relevant in other contexts that aren't as time-variable. **Comparison to other contexts**: Unlike a conflict mediator, the game bot isn't focusing on emotional tone or diplomacy; and unlike a project assistant, it isn't focused on formal record-keeping. Its primary goal is *team coordination and enhancing enjoyment*. It should **always prioritize not disrupting player concentration** – that's a golden rule in its design. Therefore, features like speaking only in designated channels, using non-intrusive notifications, or deferring to human leaders are key differences in how its interface operates relative to a productivity or mediation bot.

Conflict Mediation

In the conflict mediation context, the agent plays the role of an impartial facilitator or moderator in group chats that have become tense or hostile. Its purpose is to defuse conflicts, promote understanding, and keep the conversation respectful and productive. This is a delicate context – the agent must be **highly sensitive to tone and content**, and its interventions need to be tactful and unbiased. Unlike the productivity or gaming agents, a mediation agent often activates not for *task completion or fun*, but to manage human relationships and emotions. Think of it as an AI mediator that only steps in when needed, somewhat akin to a human moderator or counselor present in the group chat.

1. Activation Conditions: The agent should activate **only under conditions that indicate conflict or high tension** (or by explicit request for help). Key triggers might be:

- **Negative Sentiment & Toxic Language:** The primary signal is the emotional tone of messages. The agent can perform real-time sentiment analysis and toxicity detection on the group's chat. If it detects a spike in anger, insults, or aggressive language between participants (e.g. caps-lock shouting, profanities, personal attacks), it should prepare to intervene. For example, multiple messages like *"This is all your fault!"* or *"You're not listening!"* might trip a threshold. Indeed, experts note that AI can be trained to **"detect rising tension (via tone or language cues) and step in early."** [24] Using a service or model to gauge sentiment, the agent might monitor a moving window of recent messages and when the **anger/toxicity score exceeds a set limit**, that's a trigger.
- **Escalating back-and-forth:** If two or more people are rapidly exchanging heated replies – especially if it starts to go in circles – the bot may identify that pattern as a conflict needing mediation. For instance, if it sees rebuttal after rebuttal with no resolution (and perhaps others in the group have gone silent), it's a cue that the discussion is now a two-person argument dominating the chat.
- **Direct Summon for Mediation:** In some cases, a participant or bystander might explicitly ask the agent to step in. For example, someone could use a command like `/mediate` or say *"@ConflictBot, help us out here."* This opt-in trigger is important for **empowering users** – if one person feels uncomfortable with how the conversation is going, they can call the agent to assist. The agent would then immediately take a more active role upon this request.
- **Prolonged Inactivity After Conflict:** A subtle trigger could be if a conflict arises and then everyone goes silent (perhaps because feelings were hurt). A mediation bot might notice that after a spike of negativity, the chat has an unusual quiet. That could prompt it to gently reach out, either publicly or via private messages, to encourage resolution (e.g. *"It seems discussion paused after some tension. Would anyone like help clarifying viewpoints or taking a break?"*).
- **Context-specific cues:** Certain group contexts might have known conflict indicators. For example, in a work Slack, repeated disagreement on a decision thread might trigger the bot. In a community Discord, an admin flagging a conversation (via a reaction or a command) could signal the bot to intervene. The conditions can thus also include *human moderation signals*, where a moderator explicitly or implicitly asks the AI to help (maybe by reacting with an emoji like on a message to flag it).

It's worth noting that determining *when* exactly an AI mediator should jump in is tricky – doing it too soon might seem overbearing, too late might allow harm. Designers acknowledge that *when to contribute* is a key controllable aspect [3] . This agent likely errs on the side of caution: perhaps first giving a **private alert** to a moderator or even to the participants involved ("The AI senses conflict, ready to help if needed") rather than immediately broadcasting a message. As research on multi-party AI notes, deciding when the AI has the

agency to interject versus waiting for an explicit prompt is an open design question [6]. Our agent might combine approaches: passive observation until a threshold, then a gentle check-in query or suggestion before any stronger action.

2. User Experience & Interface (Web, Mobile, Embedded): The mediation agent's UI should be calming and supportive, providing tools for resolution. On **any platform (web or mobile)**, when the agent intervenes, its messages should be visually distinct to convey authority yet not alarm. For instance, its chat bubble might have a subtle highlight or a prefix like "[Moderator Bot]" to signal an official intervention. It could also use a neutral avatar (perhaps a scales of justice icon or a calm face) to set the tone. On web/desktop, the agent might post a **templated mediation message** in the channel, such as: *"I sense the conversation is getting heated. Let's all take a moment. I'm here to help clarify points or suggest a way forward."* This message might be collapsed or minimized by default (like Slack's *"see more"* format) so it doesn't take too much space unless the users choose to engage with it. On **mobile**, where space is tighter and immediate intervention might be jarring, the agent could use a less intrusive approach: possibly a banner or system notification in the chat. For example, a small banner reading " Mediation Bot is active in this discussion" could appear at the top of the chat, which users can tap to see its suggestions or to interact. This way, the presence is known but not forcing itself into the message flow unless needed.

For **integrated platforms like Slack or Discord**, the mediation agent can leverage special UI features: Slack could allow the bot to open a *modal dialogue* with conflict resolution options (accessible only to the involved users, perhaps). Imagine if the bot, upon detecting conflict, offers a button in-channel: " **Open Mediation Panel**". If users click, a sidebar or pop-up could appear with tools: e.g. it might list each person's main concerns (by analyzing their messages) and prompt them to confirm if that's accurate, essentially structuring the issues. This out-of-band interface lets the two parties work through a guided resolution without spamming the main chat. Discord, on the other hand, might use a thread or a private channel that the bot creates for conflict resolution. For example, the bot could say, *"I've started a private thread for the parties to discuss with me moderating. Others can continue the main conversation."* This splits the conflict off, much like taking two arguing coworkers into a private room to talk it out. The UI in that thread would then be just the bot and the conflicted users, where the bot can take turns giving each person space to speak (perhaps by asking each to write a summary of their view, which it then shares).

The agent can also use **ephemeral or DM messages** in service of mediation. If one user starts using harmful language, the bot might DM them privately: *"I noticed your last message might escalate conflict. Consider rephrasing or cooling down."* This one-on-one nudge (only visible to that user) can correct behavior without publicly shaming them. Likewise, it might DM an admin: *"User X and Y are in conflict in #channel. I'm intervening – here's a log link."* on Slack, to keep the human mods in the loop.

On all platforms, the mediation agent's **tone in the UI is neutral and calming**. It might use non-accusatory language and perhaps even gentle emoji (like a dove 🕊 or handshake) to set the mood. Any UI elements (buttons, prompts) should encourage positive action: e.g. "Agree on next steps" or "Request a cooling-off period" as options. Technically, the interface might also include quick feedback options for users to tell the bot if its intervention was helpful or not, which could appear after resolution (to improve future interactions).

3. Communicating Presence & Role: When the mediation agent activates, it should make its role explicit to avoid confusion. It effectively **switches from an invisible observer to an active mediator**. A possible approach is for the agent to post an introduction message like: *"I'm the group's AI mediator, here to help*

resolve the conflict and ensure respectful communication." This sets context that a neutral party is now "in the room." The presence can be communicated in an **ambient** way initially – for instance, some platforms allow setting a system notice or topic. The bot could change the channel topic to "Mediation in progress" or post a subtle header. However, given the sensitivity, an explicit message tends to work better so everyone is on the same page. The agent might also explicitly call for a small pause: *"Let's pause for a moment."* This pause sets the stage for mediation.

In terms of *ambient presence*, one idea is that the agent could have been present in the channel silently (maybe listed as a member named "MediatorBot"). If so, people might already know it could step in if needed. When it does step in, it transitions from silent to active. The agent should clarify it's not taking sides. It can say something like, *"I see both sides have concerns: I'm here to help clarify and find common ground."* Emphasizing neutrality is key – as one expert suggests, chatbots can act as *"neutral mediators"* [24]. The bot's messages should mirror a human mediator's approach: acknowledging each person's perspective and urging calm. For example: *"I understand UserA is frustrated about X, and UserB is concerned about Y. Correct me if I'm wrong."* This shows it's listening and aims to be fair.

Communication of presence also includes how the bot *exits* its role. After a conflict cools down or is resolved, the agent can post a concluding note and then "step back." E.g., *"Glad I could help. I'll step back now, but remember I'm here if needed."* That signals the end of the mediation mode, after which the bot goes quiet again. The interface might also remove any special indicators (like that topic change or banner) at this point, returning the chat to normal.

Whether the presence is ambient or explicit can sometimes be a user choice: for instance, a team might decide they want the bot *always visible* as a safeguard (explicit constant presence), or completely hidden until it speaks (ambient until triggered). Either way, once active, the agent should communicate transparently – possibly even explaining itself. For example, *"(The AI mediator detected a lot of angry sentiment, so it stepped in.)"* as a small italicized note. This helps users understand why it appeared, reducing any mystery or resistance to its involvement.

4. Customizations & Controls: Handling conflict is sensitive, so giving human admins and users control over the agent's mediation behavior is critical. Options might include:

- **Sensitivity Thresholds:** Users (or better, admins) can set how sensitive the bot is to negative language. A higher threshold means the bot tolerates more heated debate before intervening (useful in groups that value intense but respectful argument). A lower threshold means even mild sniping will summon the bot. For example, a slider from "Lenient – intervene only for harassment or slurs" to "Strict – intervene at early signs of tension" could be provided. This essentially adjusts the sentiment analysis cutoff for activation. Tools already exist that can gauge toxicity, and indeed research suggests using sentiment analysis to detect disputes in teams [25], so exposing that tuning makes sense.
- **Intervention Style:** The group could choose *how* the agent intervenes. One style might be **"Public mediator"** – the bot speaks in the channel to mediate openly. Another style could be **"Private coach"** – the bot mostly DMs advice to participants rather than public call-outs. In some cases, a combination: e.g. first try private nudges; if the conflict continues, escalate to a public mediation message. Giving an admin a toggle for "private vs public intervention" mode allows adaptation to group culture. Some groups might feel publicly called out by a bot is embarrassing, so they'd prefer

subtle guidance. Others might want the transparency of open mediation so everyone sees the resolution process.

- **Administrator Summon or Approval:** A setting could require that the bot only intervenes after a moderator approves. For instance, if the bot detects a conflict, it could notify mods "ready to mediate" and a mod clicks "Allow" for it to post. This prevents false positives or unwanted interference. Alternatively, any member might be allowed to say "Yes, please help" to the bot's query before it actively mediates. This *permission step* aligns with user suggestions that an agent should *"ask for permission instead of jumping into the discussion unprompted"* [26]. That control ensures humans still gatekeep the AI's involvement.
- **Mediation Playbook Preferences:** The agent could have options for what conflict resolution technique to apply. Some teams might prefer the bot to enforce a **cool-off period** (the bot could temporarily mute the conversation for 5 minutes and ask everyone to breathe). Others might prefer a **structured dialogue** approach (the bot asks each person to write a calm summary of their view). Or a **problem-solving approach** (bot asks what each party wants as an outcome). While the AI can try to do all, letting the group pick an approach from settings ensures the agent aligns with their conflict resolution philosophy. For example, a setting might be "When mediating, do you want the bot to: (a) encourage cooling off, (b) facilitate immediate discussion, (c) recommend involving a human mediator, (d) something else".
- **Ignore Topics or Roles:** There might be topics that are off-limits for the bot. For sensitive issues (politics, personal issues) the group might not want an AI mediator. They could list keywords or channels where the bot should *not interfere*. Conversely, they might specify that in certain channels (like a Debate club channel), the bot should never intervene unless a specific safe word is used. Role-based controls could also apply: e.g., do not intervene if two moderators are the ones arguing (perhaps they handle it themselves), but do intervene if regular members argue.
- **Logging and Privacy Controls:** In mediation, privacy is a concern. The agent likely logs the conversation to analyze it. Users or admins might control who can see those logs or summaries the bot makes. Maybe an option to automatically send a transcript of the mediated conversation to a private channel for record-keeping (some communities might want that, others not). Also, allow users to *opt out* of being analyzed by the mediation bot (though if they're in the chat it's hard, you could exclude their messages if they request). Having transparency about what the bot stores or shares is important, so these options should be available for trust – e.g., *"The mediator bot will notify an admin when it intervenes (yes/no)"*, etc.

These controls acknowledge that comfort with an AI mediator varies; the system should bend to the group's comfort level. In practice, one study found participants desired *various ways to control an AI agent's interactive behaviors* in group settings [2] – applying that here means offering as much control as feasible over when and how the bot engages in conflict resolution.

5. Differences in Behavior/Interface (Mediation Context): The mediation agent's behavior and interface are tailored to de-escalation and neutrality, which makes it quite different from the productivity or gaming agents. **Tone and language**: The mediator bot speaks in a *calm, measured, and empathetic tone*. It avoids any slang or harsh terms and often rephrases statements to remove accusatory language. For example, if one user says "You never do your part," the bot might restate it as *"It sounds like there's a feeling that contributions have been uneven."* This reframing is something AI can do to provide clarity without the emotional charge. In contrast, the gaming bot might amplify excitement and a work bot might use more directive language about tasks; the mediator is more like a counselor. As noted in research, such a bot should *mirror a human facilitator's tone while remaining neutral and precise* [27] – meaning it doesn't take sides or show frustration, and it carefully chooses words.

Content of communication: The mediation agent focuses on *process* over *topic*. That is, its messages often address *how* people are talking (tone, turn-taking) rather than the subject matter itself. For instance, it might set ground rules: *"Let's allow each person to speak without interruption."* Or encourage perspective-taking: *"Maybe each of you can express what you hope to achieve here."* It generally **does not contribute new ideas or task-related info**, whereas the other context agents do. Its contribution is meta-conversational – about the conversation itself. It may also propose solutions or compromises after hearing both sides, somewhat like a settlement suggestion. Studies have even found AI-generated mediation statements can be seen as informative and unbiased [28], so our agent might summarize the dispute and propose a middle ground in a clear, fact-focused way.

Visibility: Unlike the gaming agent which might be constantly visible and chatting, the mediator tries to **minimize presence until needed**, then becomes quite **prominent** during conflict. Its UI might even temporarily dominate the chat (for example, the channel could switch to a "moderation mode" where other messages are slowed or require confirmation, putting the bot's mediation in center). After conflict, it fades out again. This oscillation is unique to the mediation context – the productivity and gaming bots have a steady presence.

User perception and opt-in: In conflict scenarios, users might be more resistant or emotional towards an AI intervention. The agent's design must account for that – it behaves extremely *politely and helpfully*, acknowledging feelings. It might say things like, *"I understand this is a sensitive issue..."* to show empathy. It also should know when to step back if it's not helping; e.g., if users tell it to "go away" or ignore it, perhaps the bot apologizes and withdraws (and maybe notifies an admin as a backup). This adaptability differs from a productivity bot which wouldn't need to handle people possibly yelling at it.

Outcome orientation: The mediation bot might produce outputs like conflict summaries or agreed-upon resolutions. For instance, after mediating, it could post a brief *"Resolution summary"* listing what was agreed or what next steps to avoid future conflict. This is similar to how a human mediator would document a resolution. The interface could then offer to save that summary (maybe pin it or send to email). The other bots don't usually create this kind of outcome document (except maybe meeting notes in productivity, but those are factual, not about interpersonal resolution).

Comparison with other contexts: The mediation agent is essentially performing a *social function*. It has to manage human emotions and relationships. In doing so, it likely references **principles of fairness and empathy**. For example, it may enforce turn-taking – *"Let UserA finish their point, then UserB can respond."* It may also remind group norms or code of conduct: *"Let's keep the language respectful, as per our guidelines."* This aligns with moderation tasks. Indeed, Discord's AutoMod AI feature and similar are precursors, but those mostly alert mods; our agent actively engages. A productivity bot, conversely, wouldn't reprimand someone for language; a gaming bot might lightly moderate but not deeply. The mediation bot stands out by **actively managing human behavior**.

Additionally, success for the mediation agent is de-escalation. It measures things like reduction in negative sentiment after its intervention. This is a very different metric than the gaming bot (which might measure usage or response speed) or the productivity bot (which might measure task completion or user satisfaction with answers). So its behavior is governed by trying to maximize mutual understanding and minimize hostility. It might even suggest *"Maybe take a break and revisit later"* – effectively pausing the conversation, something other bots wouldn't do. This reflects how an AI mediator can *"calmly suggest resolutions, draft apologies, or reframe issues in a constructive way"* [29] to help people reconcile.

In the interface, one might see features like **anonymous mode**: the bot could allow participants to send it their feelings anonymously which it then shares without attribution to reduce personal friction. For example, two people might trust the bot with statements they wouldn't say directly. The bot could say *"One concern raised (anonymously) is that deadlines aren't clear"*. This feature, if used, drastically changes the conversation dynamic by removing direct confrontation. It's a tool unique to mediation context.

Finally, **opt-in vs passive**: While earlier we mentioned the mediation bot often steps in on its own when things get bad (passive monitoring -> active intervention), groups could also opt to only use it when someone triggers it (strict opt-in). The difference here is critical: In a public community, automatic mediation might be necessary to catch conflicts early (since many go unreported to human mods) [24]. In a small tight-knit team, they might prefer to call the bot only if they themselves can't resolve it. So the behavior can range from quietly watching like an algorithmic referee to being a summoned "digital moderator" when called. That flexibility is a difference in design philosophy not present in, say, a gaming bot that most often is just always available or a work bot that is mostly on-demand.

In conclusion, the conflict mediation agent acts with diplomacy and structure: it listens, intervenes tactfully based on defined triggers, presents a user experience that supports resolution (through careful messaging and perhaps private channels or modals), offers customization to align with the group's norms, and behaves in a contextually unique way focused on emotional intelligence and fairness. Through ambient monitoring and explicit guidance, it strives to transform a heated multi-person chat into a constructive dialogue, illustrating how an AI can facilitate better communication in group conflicts when designed with clear, functional UX patterns and respect for user control.

Sources: The design considerations above are informed by recent research on multi-party AI agents and real-world implementations. For instance, Houde *et al.* (2025) describe the importance of controlling when and how a group chat agent contributes (e.g., via direct triggers, inactivity, or asking permission) [3] [26]. Their work on the "Koala" Slack bot showed users prefer an AI that helps without overwhelming, underscoring the need for modes like reactive vs proactive [2]. In gaming, Discord's **Clyde** bot demonstrated how an AI can be a fun, helpful participant when explicitly invoked [18], and how moving extended bot interactions to threads can keep main chats clean [19]. The Clyde project also emphasized user customization – allowing personality tweaks and even full opt-out by admins [12] [22]. For conflict mediation, industry experts note that AI mediators can monitor tone and step in neutrally, offering suggestions or reframing issues calmly [24]. Overall, the design balances **opt-in vs ambient assistance** (a known challenge [6]) by giving users and admins clear controls over the agent's activation. By adapting its interface and behavior to each context – whether it's scheduling tasks, coordinating a raid, or defusing an argument – the chat-based agent can provide significant value while fitting naturally into the group conversation flow. The above design outlines strive to achieve that balance, ensuring the agent enhances collaboration and communication without ever becoming a nuisance or a hindrance to the human users.

---

[1] [6] AI's Missing Multiplayer Mode - by Charlie Guo
https://www.ignorance.ai/p/ais-missing-multiplayer-mode

[2] [2501.17258] Controlling AI Agent Participation in Group Conversations: A Human-Centered Approach
https://ar5iv.labs.arxiv.org/html/2501.17258v1

3 4 5 7 8 9 11 26 Controlling AI Agent Participation in Group Conversations: A Human-Centered Approach
https://arxiv.org/html/2501.17258v1

10 Frontiers | Designing a conversational agent to promote teamwork and collaborative practices using design thinking: An explorative study on user experiences
https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2022.903715/full

12 19 20 21 22 23 Clyde — Discord's AI Chatbot by John Avent on Dribbble
https://dribbble.com/shots/23447753-Clyde-Discord-s-AI-Chatbot

13 Guide to AI features in Slack | Slack
https://slack.com/help/articles/25076892548883-Guide-to-AI-features-in-Slack

14 15 AI in Slack apps overview | Slack Developer Docs
https://docs.slack.dev/ai/

16 17 18 Clyde AI: Revolutionizing Conversations on Discord - Oreate AI Blog
https://www.oreateai.com/blog/clyde-ai-revolutionizing-conversations-on-discord/c7478805fb8110d7413c0ffcb0620225

24 28 29 The AI Mediator: Can AI help with dispute resolution and peace keeping? — FRANKI T
https://www.francescatabor.com/articles/2025/6/3/the-ai-mediator-can-ai-help-with-dispute-resolution-and-peace-keeping

25 Leveraging AI for Conflict Resolution in Remote Teams - Bitrix24
https://www.bitrix24.com/articles/leveraging-ai-for-conflict-resolution-in-remote-teams.php

27 AI Agents in Mediation: Powerful, Proven Gains | Digiqt Blog
https://digiqt.com/blog/ai-agents-in-mediation/