Tasky Chat UI/UX, State, MCP/API Cheat Sheet

See also: State Management Diagrams for visual end-to-end flows.

A quick, practical reference for working on Tasky chat: components, state, MCP tools, IPC, data models, and end-to-end flows.

Components Overview

- ChatModule: Container orchestrating settings validation, AI streaming, MCP readiness, message timeline, composer, and confirmation.
  - File: `src/components/apps/ChatModule.tsx`
- MessageContainer: Builds ordered timeline from messages, tool events, inline confirmations, and adaptive card snapshots.
  - File: `src/components/chat/MessageContainer.tsx`
- MessageBubble: Renders user (plain text) and assistant (markdown) messages.
  - File: `src/components/chat/MessageBubble.tsx`
- MessageSkeleton: Placeholder when streaming assistant.
  - File: `src/components/chat/MessageSkeleton.tsx`
- AdaptiveCardRenderer: Displays tool results as cards; special cases for tasks/reminders lists and CRUD.
  - File: `src/components/chat/AdaptiveCardRenderer.tsx`
- InlineConfirmation: Inline confirmation UI for write/delete/execute tools.
  - File: `src/components/chat/InlineConfirmation.tsx`
- ToolCallDisplay: Live tool call monitor based on ToolEvent (start/done/error).
  - File: `src/components/chat/ToolCallDisplay.tsx`
- ChatComposer: Input (auto-resize), send/stop buttons, MCP Tools palette trigger.
  - File: `src/components/chat/ChatComposer.tsx`
- McpToolsHelper: MCP templates (create/list/update/delete/execute tasks; create/list/update/delete reminders).
  - File: `src/components/chat/McpToolsHelper.tsx`
- ai-elements: Presentational atoms for tool/task rendering.
  - Files: `src/components/ai-elements/*`

State Management

- useChatPersistence
  - State: `chatId: string|null`, `messages: ChatMessage[]`
  - IPC: `chat:create|list|load|save|delete|reset`
  - File: `src/components/chat/hooks/useChatPersistence.ts`
- useMcpTools
  - State: `toolEvents: ToolEvent[]`, `pendingConfirm`, `pendingResult`, `loadingTools: Set<string>`
  - Events: listens to `tasky:tool`, `tasky:tool:confirm`, produces result snapshots (`__taskyCard`)
  - File: `src/components/chat/hooks/useMcpTools.ts`
- useScroll
  - Refs/State: `scrollRef`, `showJumpToLatest`

    - Helpers: `isNearBottom`, `scrollToBottom`, `handleScroll`, `autoScrollIfNeeded`
    - File: `src/components/chat/hooks/useScroll.ts`
- Transient (ChatModule)
    - `input`, `busy`, `error`, `streamingAssistantMessage`, `mcpReady`, `systemPrompt`, `temperature`

Types (Chat Layer)

- ChatMessage: `{ role: 'user'|'assistant', content: string }`
- ToolEvent: `{ id, phase: 'start'|'done'|'error', name, args?, output?, error?, timestamp? }`
- AdaptiveCard: `{ kind: 'confirm'|'result', id?, name, args?, output? }`
- ConfirmState: `{ id, name, args } | null`
- Files: `src/components/chat/types.ts`

AI Service and Settings

- AIService
    - Methods: `streamText(messages, options, tools?)`, `generateText(messages, tools?)`
    - Providers: Google (tools supported), LM Studio (tools disabled)
    - File: `src/ai/index.ts`
- Settings
    - Manager: defaults, models, validation, normalization, provider capabilities, connectivity tests
    - Adapter: translates app `Settings` ⇄ AI settings, auto-fixes issues
    - Files: `src/ai/settings/index.ts`, `src/ai/settings/adapter.ts`, `src/ai/config/index.ts`, `src/ai/providers/index.ts`

MCP Tooling (Renderer)

- mcpCall tool (AI SDK tool)
    - Input: `{ name: string, args?: { title, description, id, message, time, days[], enabled, status, dueDate, tags[], oneTime } }`
    - Flow: auto-confirm list/get → inline confirm for write/delete → emit `tasky:tool` events → IPC to MCP → emit result → snapshot card
    - File: `src/ai/mcp-tools.ts`
- callMcpTool(name, args)
    - Programmatic invocation (used for inline JSON tool calls pasted by the model)
    - File: `src/ai/mcp-tools.ts`

Electron IPC Surface

- Preload: `window.electronAPI`
    - Chat: `chat:create|list|load|save|delete|reset`
    - MCP: `mcpToolsList`, `mcpToolsCall`
    - Tasks/Reminders + Notifications + Pomodoro
    - File: `src/preload.ts`
- Main handlers
    - `mcp:tools/list` → `sendMcpMessage`

  - `mcp:tools/call` → `sendMcpMessage` → notifications and UI refresh (`tasky:tasks-updated`, `tasky:reminders-updated`)
  - File: `src/main.ts`

MCP Server (Backend)

- Tools
  - Tasks: `tasky_create_task`, `tasky_list_tasks`, `tasky_update_task`, `tasky_delete_task`, `tasky_execute_task`
  - Reminders: `tasky_create_reminder`, `tasky_list_reminders`, `tasky_update_reminder`, `tasky_delete_reminder`
  - File: `tasky-mcp-agent/src/mcp-server.ts`
- Bridges
  - TaskBridge: SQLite `tasks`, `task_tags`; CRUD, list, execute (HTTP delegate to main app, else status update)
    - File: `tasky-mcp-agent/src/utils/task-bridge.ts`
  - ReminderBridge: SQLite `reminders`; defaults all days if none, relative time parsing support
    - File: `tasky-mcp-agent/src/utils/reminder-bridge.ts`
- Time Parser
  - Relative times: "in X minutes/hours/seconds/from now" → HH:MM; sets `oneTime=true` for relative inputs
  - File: `tasky-mcp-agent/src/utils/time-parser.ts`

End-to-End Flows

- User message → AI + tools

  1. `ChatModule.onSend` builds `[system, history, user]` and sets `tools={ mcpCall }` if MCP ready.
  2. Model may call `mcpCall(name,args)` during streaming.
  3. `mcpCall` emits `tasky:tool:confirm` (if needed) → `InlineConfirmation` UI via `useMcpTools`.
  4. On accept, `mcpCall` emits `tasky:tool` start → `electronAPI.mcpToolsCall` → MCP server.
  5. MCP returns `CallToolResult`; `mcpCall` emits `done` with output.
  6. `useMcpTools` snapshots assistant message containing `{"__taskyCard": {kind:'result',...}}`.
  7. `MessageContainer` detects and renders via `AdaptiveCardRenderer`.

- Example: List tasks

  - Tool: `tasky_list_tasks` (auto-confirm)
  - UI: `TaskDisplay` renders returned array.

- Example: Create reminder (relative time)

  - Tool: `tasky_create_reminder` args `{ message, time: "in 25 minutes", days: [] }`
  - Server: parses to HH:MM and forces one-time
  - UI: Reminder card with schedule badges and time

Data Models (MCP Results)

- Task (TaskBridge JSON)
  - `schema`: `{ id, title, description?, dueDate?, createdAt, updatedAt?, tags?, assignedAgent?, executionPath? }`
  - `status`: `'PENDING'|'IN_PROGRESS'|'COMPLETED'|'NEEDS_REVIEW'|'ARCHIVED'`
  - `humanApproved`, `result?`, `completedAt?`, `reminderEnabled?`, `metadata?`
- Reminder (ReminderBridge JSON)
  - `{ id, message, time, days: string[], enabled, oneTime? }`

Confirmation Rules

- Auto-confirm: list/get/read-only (`list_*`, `get_*`, `tasky_list_*`)
- Requires confirm: create/update/delete/execute; inline confirmation with contextual details

Troubleshooting Tips

- Provider/tool mismatch: LM Studio streams without tools; switch to Google for tool use.
- Function schema errors (400/Invalid schema): Chat retries without tools; verify AI SDK tool wiring and MCP server.
- Task execution needs main app: if `/execute-task` not reachable, MCP falls back to status update.
- Output parsing: AdaptiveCardRenderer attempts JSON extraction; raw strings still render in `ToolOutput`.

Key Files Index

- Chat container: `src/components/apps/ChatModule.tsx`
- Chat UI: `src/components/chat/*`
- Hooks: `src/components/chat/hooks/*`
- AI core: `src/ai/index.ts`, `src/ai/mcp-tools.ts`
- Settings: `src/ai/settings/*`, `src/ai/config/*`, `src/ai/providers/*`
- Preload/IPC: `src/preload.ts`, `src/main.ts`
- MCP server: `tasky-mcp-agent/src/mcp-server.ts`
- Bridges: `tasky-mcp-agent/src/utils/*`