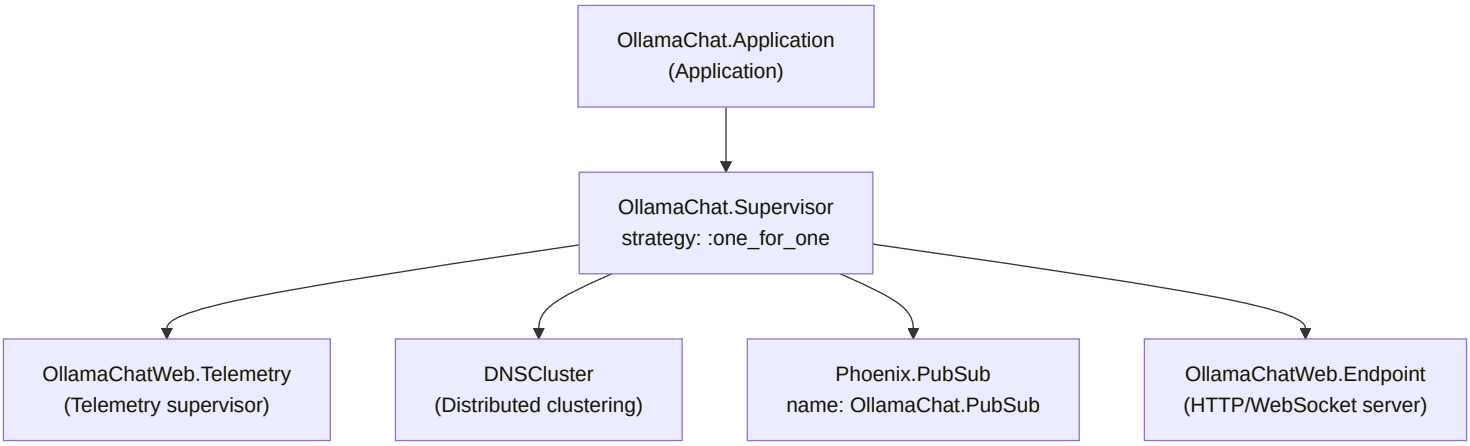
Ollama Chat — Architecture Diagrams

OTP Supervision Tree

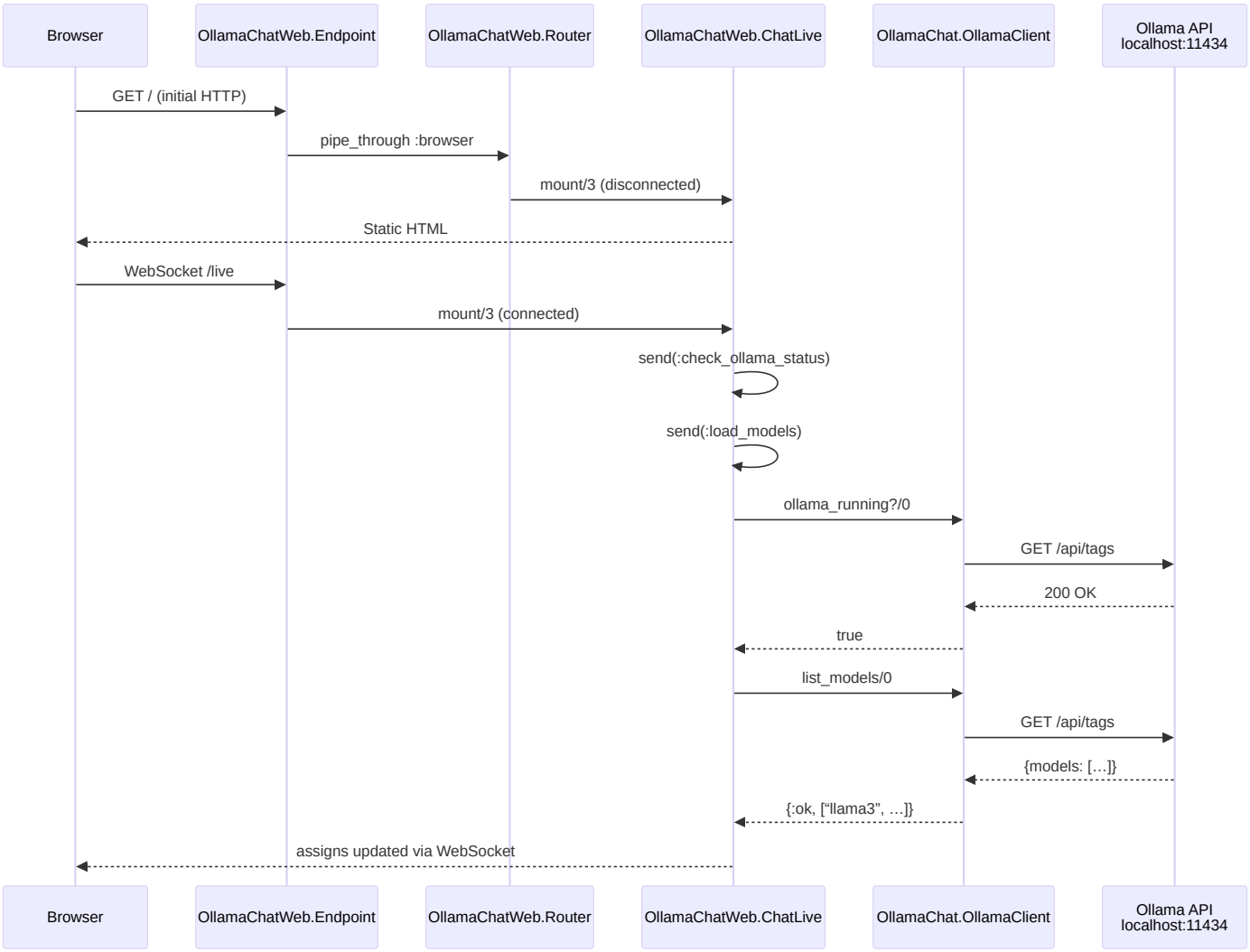
The application uses a `:one_for_one` supervision strategy — if any child crashes, only that child is restarted.



There is no database supervisor or GenServer processes — `OllamaClient` is a stateless module with plain functions, not a supervised process.

HTTP/WebSocket Request Flow

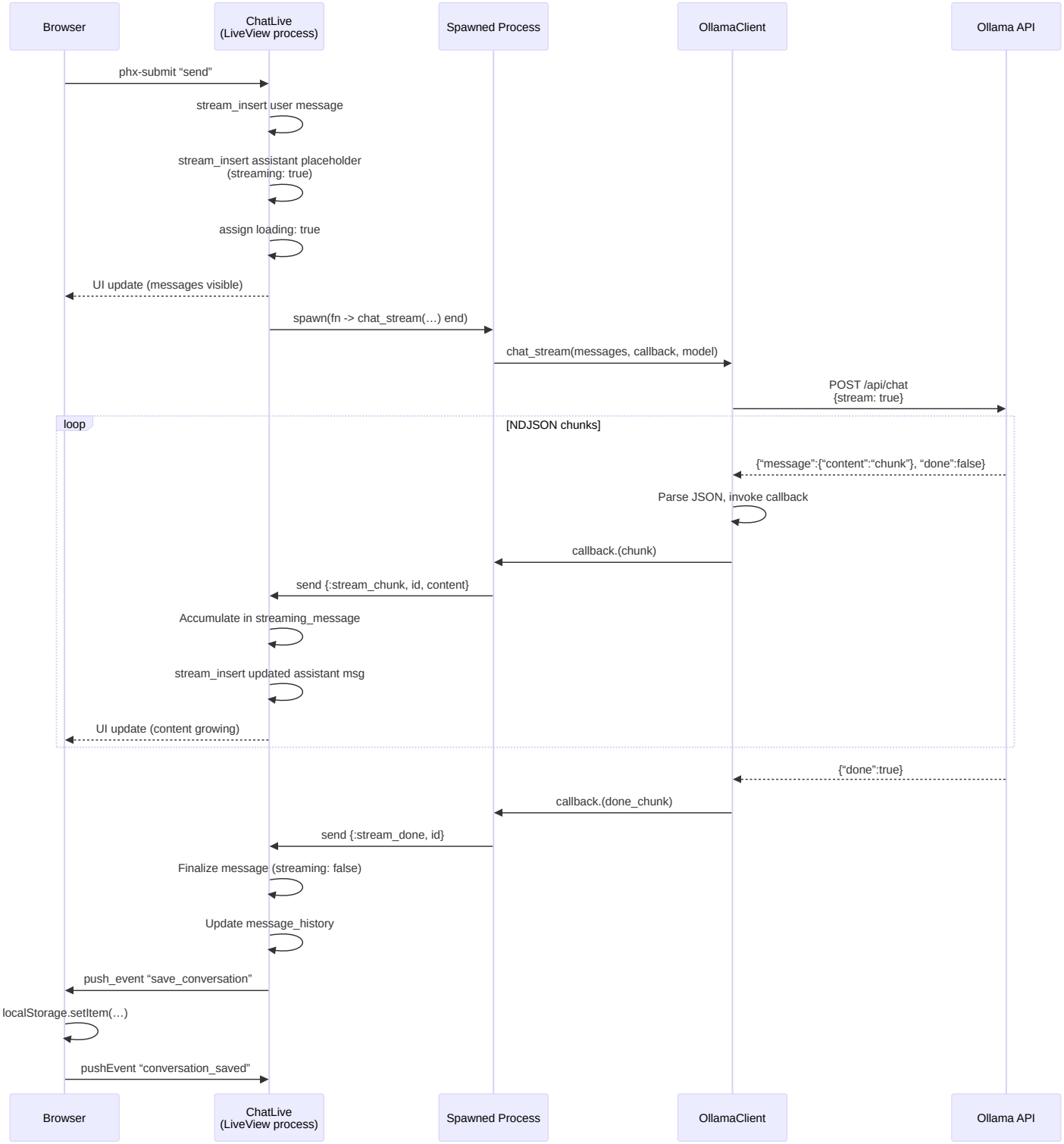
All user interaction goes through a single LiveView mounted at `/`. Static assets are served by `Plug.Static`.



The Endpoint plug pipeline: `Plug.Static` → `LiveReloader` (dev) → `RequestId` → `Telemetry` → `Parsers` → `MethodOverride` → `Head` → `Session` → `Router`.

Chat Streaming Flow

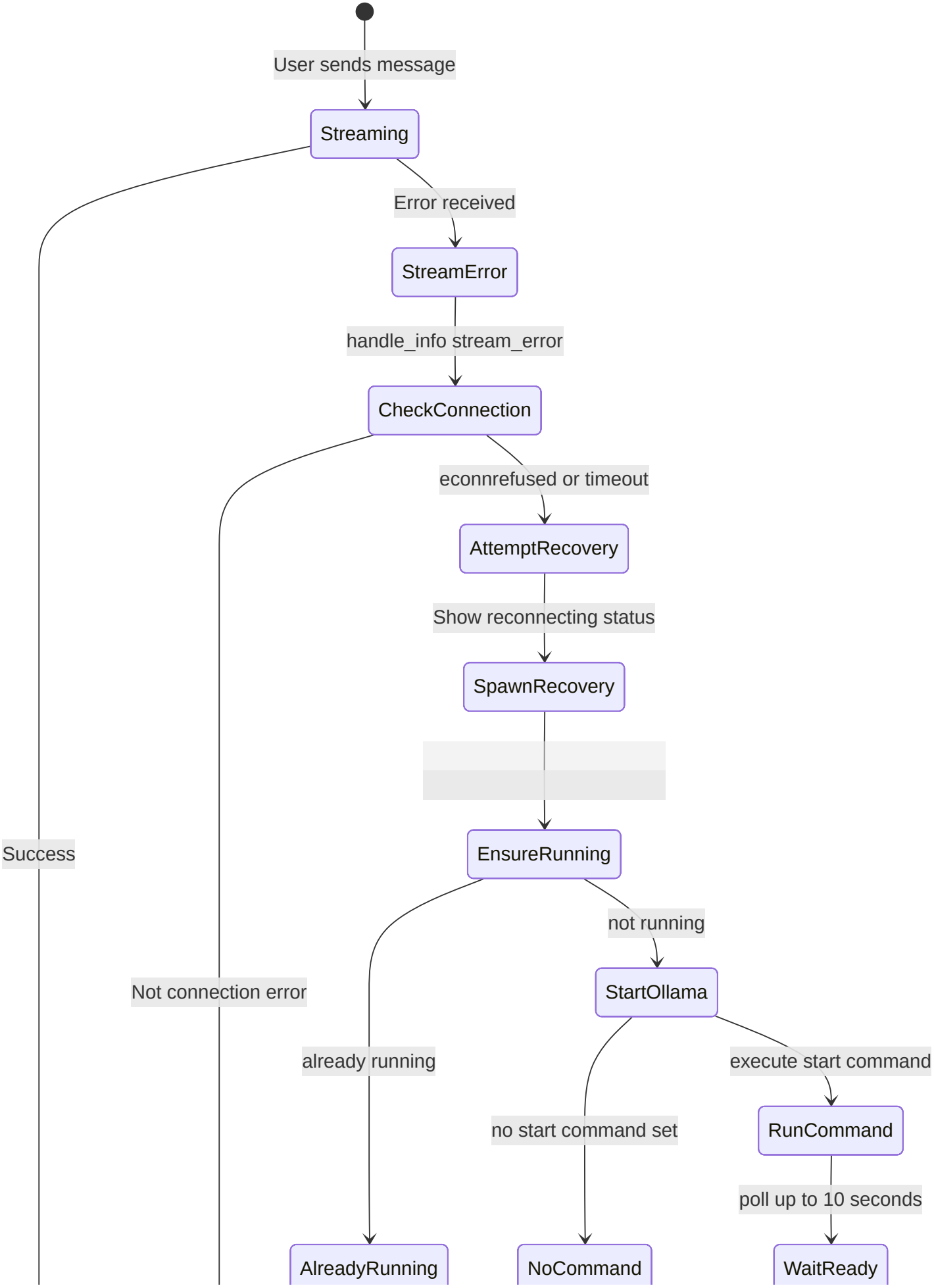
When the user sends a message, a spawned process streams NDJSON chunks from Ollama back to the LiveView via message passing.

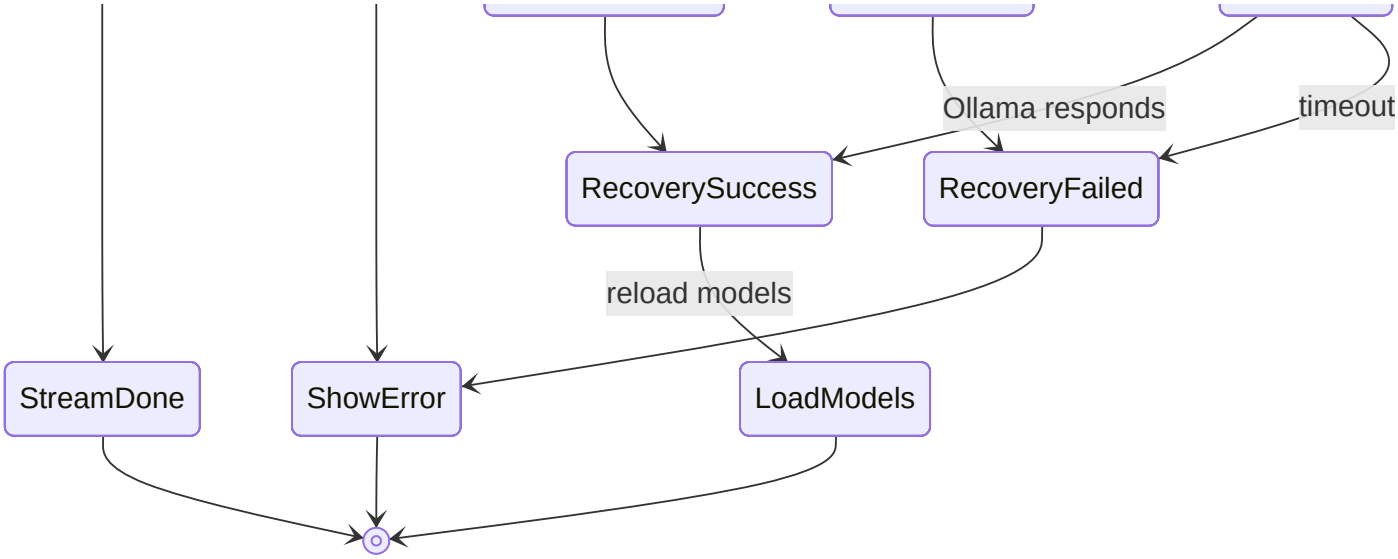


Key design decisions: - Streaming runs in a **spawned process** to avoid blocking the LiveView - Content is **accumulated** in `streaming_message` assign and re-inserted into the stream on each chunk - On completion, the conversation is **auto-saved to browser localStorage** via a `push_event`

Error Recovery Flow

Connection failures trigger automatic Ollama restart attempts when `OLLAMA_START_COMMAND` is configured.


User sends message
Streaming
Success
Error received
StreamError
handle_info stream_error
CheckConnection
Not connection error
econnrefused or timeout
AttemptRecovery
Show reconnecting status
SpawnRecovery
EnsureRunning
already running
AlreadyRunning
not running
StartOllama
no start command set
NoCommand
execute start command
RunCommand
poll up to 10 seconds
WaitReady



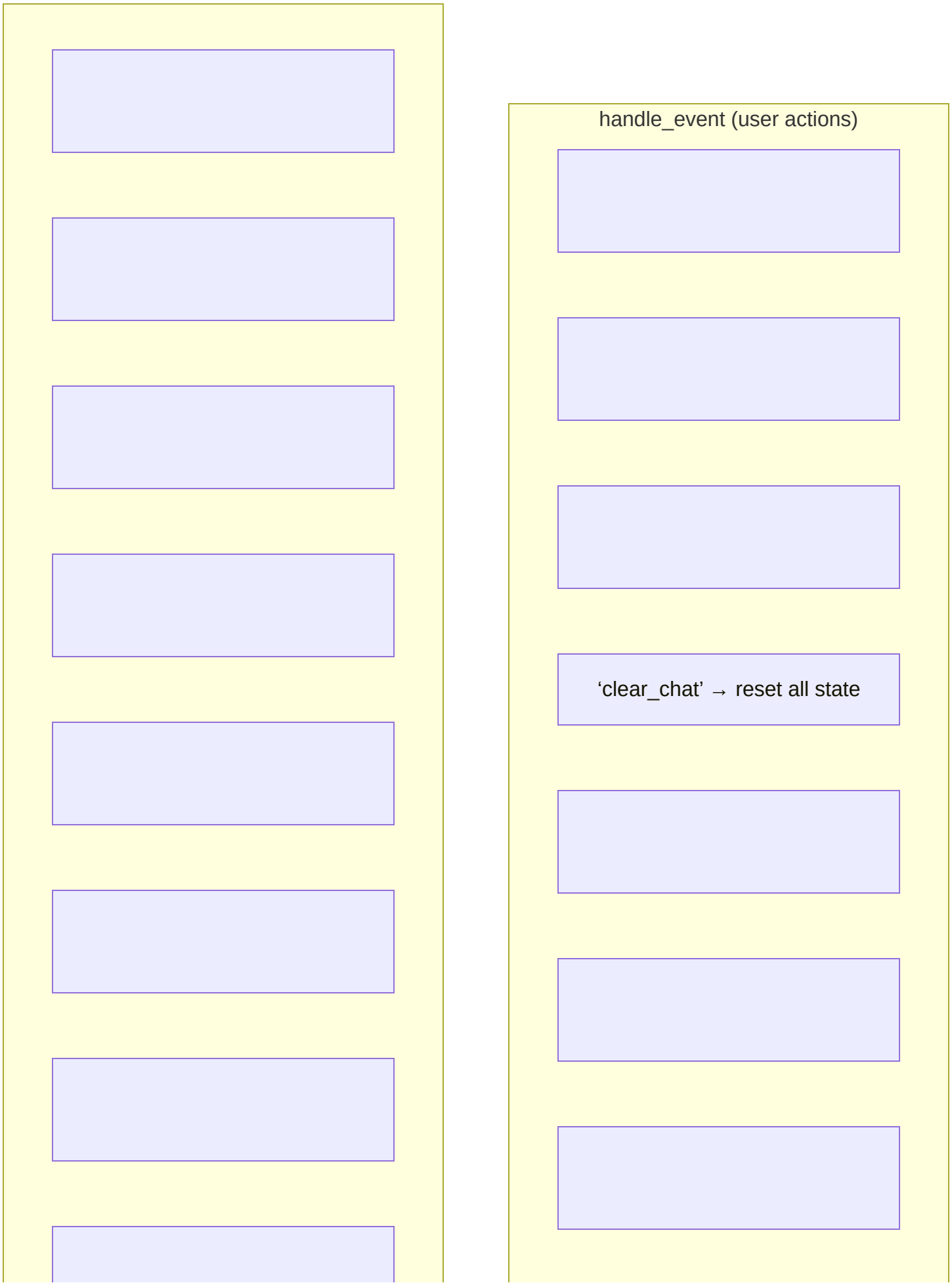
Recovery happens in two layers: 1. **OllamaClient level**: `chat/2` and `chat_stream/3` detect `:econnrefused`, call `ensure_ollama_running/0`, sleep 2s, then retry 2. **ChatLive level**: `{:stream_error, ...}` triggers `{:attempt_recovery, ...}` which spawns a recovery process and updates UI status

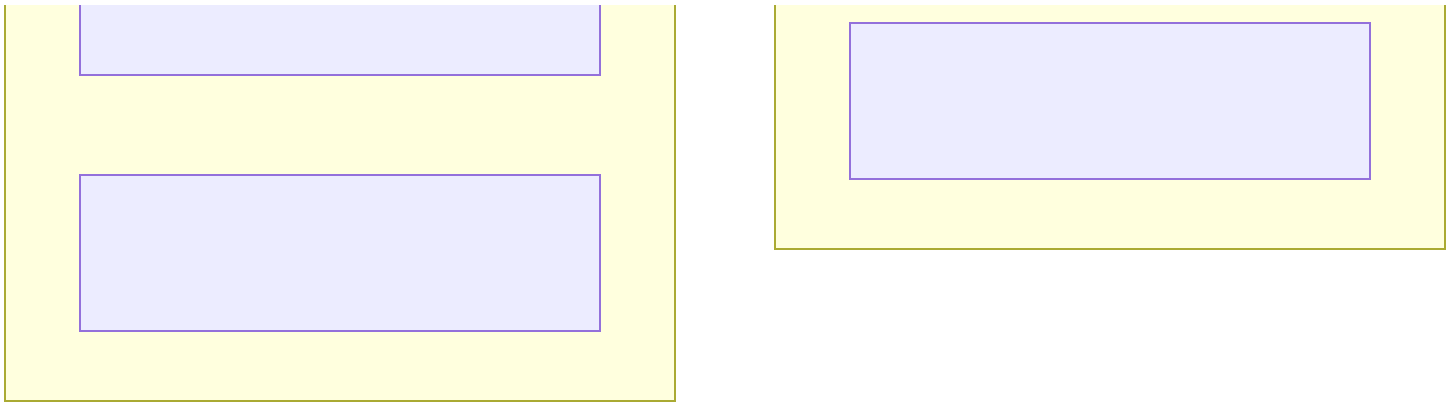
---

ChatLive State & Event Map

Socket Assigns

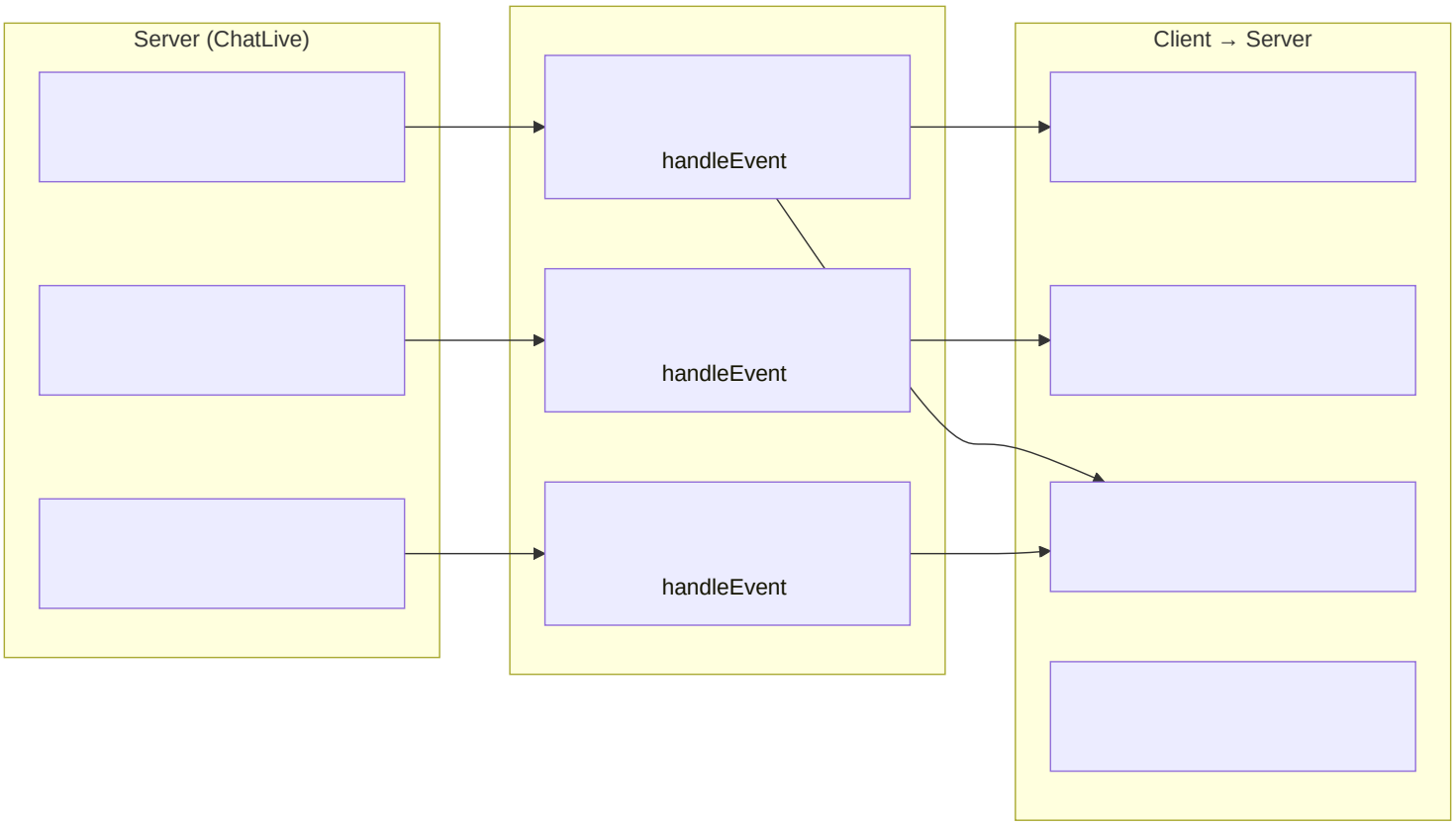
Event Handlers


handle_event (user actions)
‘clear_chat’ → reset all state

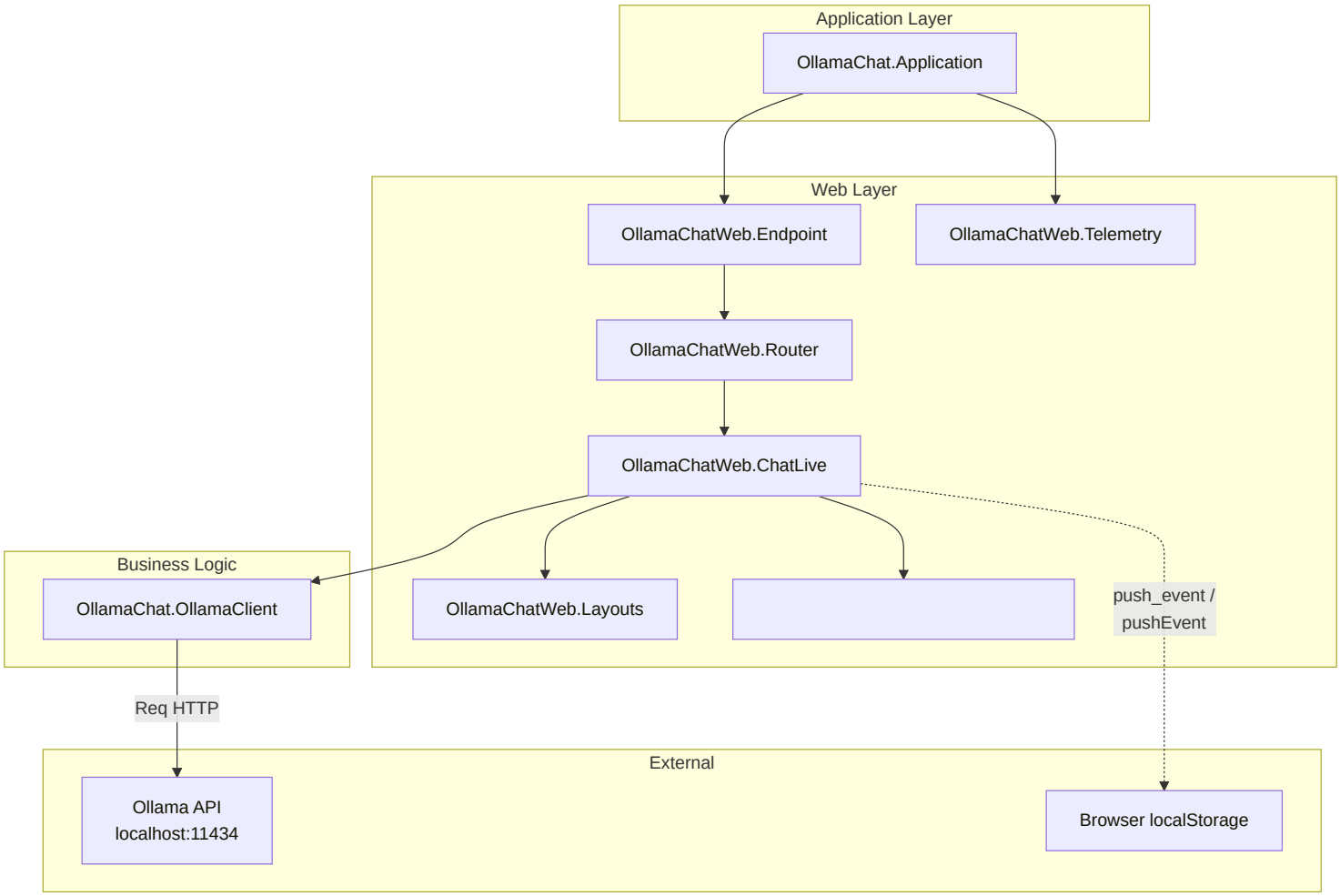


Client ↔ Server Event Bridge

Conversation persistence uses `push_event` / `pushEvent` to bridge LiveView and browser localStorage:



Module Dependency Graph



The dependency graph is intentionally flat — `ChatLive` is the only module that calls `OllamaClient`, and there are no intermediate service layers or GenServers.
