term-journal

Use Case Specification Document

Use Case ID: 1

Version No.: 1

Warren Lazarraga
September 4, 2025

Contents

Use-Case Document: <Summary>

1 Summary

1.1 Use-case description

A private, always-there journal you SSH into. It fingerprints your SSH key to identify you, time-stamps entries to Postgres, and uses an LLM to ask one smart daily question that builds on yesterday's thoughts.

1.2 Actors

- **User:** Developers & engineers who live in the terminal and prefer keyboard-only workflows.

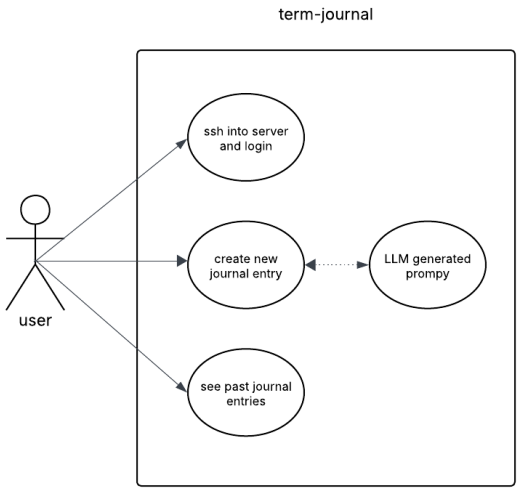

Figure 1: Use Case Diagram

2 Use Case 1 — SSH into Server and Log In

Primary Actor

User (developer)

Goal

Start a journal session securely from the terminal.

Scope

`term-journal` (SSH TUI + Postgres)

Preconditions

- Server is reachable; host key is known (or user accepts it on first connect).
- SSH public key is available on the client.
- Database is up.

Trigger

User runs `ssh journal.host` (optionally `-p <port>`).

Main Success Scenario

1. Server accepts the SSH connection and allocates a PTY.
2. System fingerprints the user's public key (e.g., `SHA256:...`).
3. If first time, a user record is created; otherwise the existing row is loaded.
4. The TUI banner renders today's date and today's question (if any).
5. Session lands at the command prompt (`>`).

Alternate/Exception Flows

- A1: Unknown/blocked key → show "access denied" and close session.
- A2: Host key mismatch → user is warned; connection refused until trust.
- A3: DB unavailable → graceful error message; retry suggestion.
- A4: No PTY (non-interactive) → instruct to connect with `ssh -t`.

Postconditions

- User is identified for this session by SSH fingerprint.
- A session audit record (connect time, IP, fingerprint) is logged.

Acceptance Checks

- With a valid key, TUI appears within <1s and shows today's question.
- With a revoked key, access is denied.

Non-Functional Notes

- Disable agent/X11/port forwarding on the public endpoint.
- Rate-limit attempts; capture minimal telemetry (no keystrokes).

3 Use Case 2 — Create a New Journal Entry (with LLM Prompt)

Primary Actor

User

Goal

Capture today's thoughts quickly and persist them with a timestamp.

Related

Uses yesterday's entry to generate tomorrow's question.

Preconditions

- User is authenticated (Use Case 1).
- `daily_questions(for_date=today)` exists or a default prompt is available.

Trigger

User starts typing at the prompt or invokes `:edit`.

Main Success Scenario

1. TUI shows today's question (e.g., "What energized you today?").
2. User types free-form text.
3. User finishes with `::save`.
4. System UPSERTs into `journal_entries` for `(user_id, entry_date=today)` with timestamp.
5. System calls the LLM with today's body to generate tomorrow's question ($\leq$140 chars).
6. System stores/upserts `daily_questions(for_date=tomorrow)`.
7. TUI confirms: "Saved. Queued tomorrow's question: ...".

Alternate/Exception Flows

- B1: Empty buffer $\rightarrow$ "Nothing to save." (no DB writes).
- B2: Duplicate same-day save $\rightarrow$ body replaced (UPSERT).
- B3: LLM unavailable $\rightarrow$ fallback to generic question, log error.
- B4: Connectivity blip during save $\rightarrow$ retry; on failure, keep buffer in memory.

Postconditions

- Exactly one row per `(user_id, entry_date)`.
- A question exists for next day (generated or fallback).

Data & Privacy

- Consider per-user encryption at rest for body.
- Log prompt+question pair (no PII) for observability.

Acceptance Checks

- Saving an entry → row exists with today's date and non-empty body.
- After save → tomorrow's question is present (generated or fallback).

Edge Cases

- Timezone boundaries: compute journal day with configured IANA TZ.
- Very long entries: truncate LLM context (first 4k chars), store full body.

4 Use Case 3 — See Past Journal Entries

Primary Actor

User

Goal

Review previous entries for reflection and continuity.

Preconditions

- User is authenticated (Use Case 1).
- At least one past entry exists (or UI shows "No history yet").

Trigger

User runs `:history` or `:view YYYY-MM-DD`.

Main Success Scenario

1. `:history` shows last N days (e.g., 7) with date + preview.
2. `:view YYYY-MM-DD` loads full body for specified date (read-only).
3. User can return to prompt and continue editing today's entry.

Alternate/Exception Flows

- C1: No history → "No history yet."
- C2: Invalid date format → "Usage: :view YYYY-MM-DD"
- C3: No entry for that date → "No entry for that date."
- C4: Large entry → paginate/scroll (future enhancement).

Postconditions

- No data modified; read-only access logged for analytics.

Optional Extensions

- `:search "term"` → substring search across entries.
- `:export md` → stream Markdown archive / one-time URL.
- `:stats` → streaks, entry counts, average length.

Acceptance Checks

- `:history` returns correct dates and previews for authenticated user only.
- `:view` displays stored text exactly (no truncation).