

Welcome back Choon Keat!

```
  *   █   █   █   *
 * █ █ █ █ █ █ *
 * █ █ █ █ █ █ *
```

Haiku 4.5 · Claude Max · choonkeat@gmail.com's
Organization
~/git/choonkeat/record-tui

Tips for getting started

Run /init to create a CLAUDE.md file with instructions for Claude

Recent activity

No recent activity

A gift for you

Your rate limits are 2x higher through 12/31. Enjoy the extra room to think!

> ls -l

```
● Bash(ls -l)
└─ total 32
   drwxr-xr-x  6 choonkeatchew  staff   192 31 Dec 14:57 bin
   drwxr-xr-x  3 choonkeatchew  staff    96 31 Dec 14:48 cmd
   ... +6 lines (ctrl+o to expand)
```

● Here's the directory structure. The project contains:

- bin/ - Executables/binaries
- cmd/ - Command entry points
- go.mod - Go module definitions
- internal/ - Internal Go packages
- Makefile - Build and task automation
- README.md - Project documentation
- research/ - Research notes
- tasks/ - Task tracking files

What would you like to do?

> read @research/ and @tasks/ and propose some ADRs

- └─ Listed directory **research/**
- └─ Listed directory **tasks/**

- Read(research/2025-12-29-1430-tools-landscape.md)
 - └─ Read **488** lines
- Read(research/2025-12-29-1530-terminal-recording-research.md)
 - └─ Read **444** lines
- Read(tasks/2025-12-29-160800-session-to-html-converters.md)
 - └─ Read **453** lines
- Read(tasks/2025-12-29-170500-session-to-html-playback.md)
 - └─ Read **513** lines
- Read(tasks/2025-12-31-120242-go-rewrite.md)
 - └─ Read **170** lines
- Read(tasks/2025-12-31-140606-record-tui-go.md)
 - └─ Read **265** lines
 - └─ **Interrupted** · What should Claude do instead?

> oh read our source code too of course.

- Search(pattern: "cmd/**/*.go")
 - └─ Found 1 file (ctrl+o to expand)

- **Search**(pattern: "internal/**/*.*.go")
 - └ Found **12** files (ctrl+o to expand)
- **Read**(cmd/record-tui/main.go)
 - └ Read **94** lines
- **Read**(internal/session/cleaner.go)
 - └ Read **56** lines
- **Read**(internal/html/template.go)
 - └ Read **151** lines
- **Read**(internal/record/recorder.go)
 - └ Read **64** lines
- **Read**(internal/record/converter.go)
 - └ Read **118** lines
- **Read**(internal/record/environment.go)
 - └ Read **23** lines
- **Read**(README.md)
 - └ Read **248** lines
- Based on the research, task tracking, and source code, here are the key architectural decisions to document as ADRs:

Proposed ADRs

1. ADR-001: Language Choice - Go over Node.js

Status: Implemented

Context:





- Started with Node.js/TypeScript (src/lib/* from earlier iteration)
- Rewrote complete implementation in Go

Decision: Build record-tui as standalone Go binary

Rationale:

- **Binary portability:** Single 2.7MB executable vs 100MB+ Node.js runtime
- **No runtime dependency:** Works anywhere without npm/Node.js
- **Performance:** Instant startup, minimal overhead
- **Distribution:** Can be distributed as single binary to ~/bin

Consequences:

-  Reduced binary size 37x
-  Simplified installation (no package manager needed)
-  Instant execution, no startup overhead
-  Smaller Go ecosystem compared to Node.js for HTML tools

2. ADR-002: Static Display vs Animated Playback

Status: Implemented (Static)

Context:

- Research explored animated playback with timing.log (xterm.js with requestAnimationFrame)
- Tasks show full playback sequencer implementation
- Current implementation uses only single frame






Decision: Use static display of final session output (one PlaybackFrame per session)

Rationale:

- **Simplicity:** Fewer moving parts, easier to debug
- **Instant viewing:** No animation overhead, content visible immediately

- **Sufficient for most use cases:** Users can review what happened
- **Reduced file size:** No need to store multiple frames
- **Browser compatibility:** Works in all browsers, no complex JavaScript required

Consequences:

-  Simpler codebase, fewer bugs
-  Smaller HTML output files
-  Faster generation
-  Cannot replay command execution in real-time
-  Loses timing/flow information from original session

Future Enhancement: If animated playback needed, refactor ConvertSessionToHTML to generate multiple frames from timing data

3. ADR-003: POSIX `script` Command vs Custom PTY Wrapper

Status: Implemented (POSIX script)

Context:







- Research shows asciinema and custom PTY wrapping approaches
- POSIX `script` command is built-in on macOS/Linux

Decision: Use OS `script` command via `exec.Command()`

Rationale:

- **Zero dependencies:** Built into POSIX systems
- **Proven:** Used for decades, battle-tested
- **ANSI preservation:** Captures full terminal data with color codes
- **No custom code needed:** Complex PTY handling done by OS
- **Simple integration:** Just `exec.Command("script", outputPath)`

Consequences:

-  Minimal code, maximum reliability
-  Inherits all script command features
-  No need to reimplement terminal emulation capture
-  Limited to POSIX systems (not Windows native)
-  Can't customize capture behavior
-  Limited to what script command supports

4. ADR-004: xterm.js via CDN vs Bundled

Status: Implemented (CDN)

Context:







- xterm.js needed for proper terminal emulation in HTML
- Could bundle as npm dependency or load from CDN

Decision: Load xterm.js from jsDelivr CDN

Rationale:

- **Standalone HTML:** No external dependencies, just fetch from browser
- **Browser cache:** CDN version cached across multiple users/sessions
- **Latest versions:** Easy to update by changing URL
- **File size:** Doesn't bloat the generated HTML itself
- **No build complexity:** No npm/webpack/bundler needed

Consequences:

-  Truly standalone HTML (can send via email, share via link)
-  No build/bundling complexity
-  Browser cache benefits
-  Requires internet connection to load
-  Privacy: User IP visible to CDN provider
-  Won't work in offline environments

Mitigation: Users concerned about privacy/offline can download xterm.js locally and edit HTML

5. ADR-005: Session Metadata Stripping Strategy

Status: Implemented (Line-based detection)

Context:






- `script` command adds header/footer lines
- Need to remove: "Script started on", "Command:", "Script done on", "Saving session"
- Could use regex or line-by-line parsing

Decision: Line-based detection with string matching (cleaner.go)

Rationale:

- **Simple:** Easy to understand and maintain
- **Robust:** Handles variations in script command output
- **Efficient:** Single pass through content
- **Debuggable:** Clear logic for what gets removed

Consequences:

-  Straightforward code
-  Works with real-world session.log files
-  Easy to extend with new patterns
-  Not formally specified (depends on script command output format)
-  May fail if script command changes its format

Risk Mitigation: Tests include real session.log files from ~/.record-tui/

6. ADR-006: Automatic Environment Setup (FORCE_COLOR, COLORTERM)

Status: Implemented (Automatic)

Context:






- Terminal applications may disable colors when detecting script/PTY mode
- Need colors to be captured for HTML playback

Decision: Automatically set FORCE_COLOR=1 and COLORTERM=truecolor before recording

Rationale:

- **Capture colors:** Ensures CLI apps output colors despite running in script PTY
- **Transparent to user:** Happens automatically, no user action needed
- **Common pattern:** Many Node.js tools follow this convention
- **Preserves intent:** User runs `record-tui` expecting colors to be captured

Consequences:

-  Colors always captured (when available)
-  No user confusion about missing colors
-  Matches user expectations
-  Globally modifies environment (affects all child processes)
-  Could theoretically affect unrelated commands if record-tui is interrupted

Mitigation: Only set for duration of recording process

7. ADR-007: Graceful Degradation - Recording > Conversion

Status: Implemented (in main.go)

Context:






- Recording and conversion are separate steps
- Conversion failure should not invalidate the recording

Decision: If conversion fails, show warning but keep session.log and exit 0 (success)

Rationale:

- **Recording is primary goal:** Conversion is enhancement
- **Don't lose user data:** session.log is valuable even without HTML
- **Clear feedback:** Warning message explains what happened
- **User can retry:** Can run session-to-html manually later

Consequences:

-  User data never lost
-  Partial success is communicated clearly
-  Conversion bugs don't prevent recording workflow
-  Could hide conversion problems
-  User might not realize conversion failed

8. ADR-008: Base64 Encoding for Frame Data

Status: Implemented (template.go)

Context:






- Frame content (with ANSI codes) needs to be embedded in HTML
- Could use direct string, JSON encoding, or base64

Decision: Encode frames as base64 in HTML, decode in JavaScript

Rationale:









- **Handles all characters safely:** Binary-safe encoding
- **Avoids escaping issues:** No need to escape quotes, backslashes
- **JavaScript compatible:** Standard atob/btoa functions
- **Preserves ANSI codes:** No accidental interpretation

Consequences:

-  No escaping issues
-  Robust for all session content
-  Simple decoding in JavaScript
-  Slightly larger HTML (base64 encoding overhead ~33%)
-  Extra decode step in browser

Performance Impact: Minimal - decode is fast, only happens once on page load

Summary Table

ADR	Decision	Status	Risk Level
001	Go over Node.js	 Implemented	Low (proven benefits)
002	Static display	 Implemented	Low (sufficient for MVP)
003	POSIX script	 Implemented	Medium (POSIX-only)
004	xterm.js CDN	 Implemented	Medium (privacy/offline)
005	Line-based metadata strip	 Implemented	Medium (format dependency)
006	Auto env setup	 Implemented	Low (standard practice)
007	Graceful degradation	 Implemented	Low (preserves data)
008	Base64 encoding	 Implemented	Low (proven approach)

These would be good candidates to create in `adr/` directory as you document them more formally.

> let's write them all down - succinctly. what's a good standard adr format that isn't too verbose.

- Here's a concise ADR format that's less verbose but captures what matters:

```
# ADR-NNN: Title





**Status**: Accepted

**Context**
Brief explanation of why this decision matters.
```

****Decision****

What we chose and the core rationale in 2-3 sentences.

****Consequences****

-  Benefit 1
-  Benefit 2
-  Tradeoff 1
-  Tradeoff 2

This is based on the standard but omits the "Alternatives" section (which you implied in context) and keeps each section tight. Let me write all 8 now:

- **Write**(adr/001-language-choice-go.md)
 - └ Wrote 18 lines to **adr/001-language-choice-go.md**
 - 1 # **ADR-001: Language Choice - Go**
 - 2 ****Status****: Accepted
 - 3 ****Context****
 - 4 Originally prototyped in Node.js/TypeScript. Need a language that produces portable, standalone binaries for distribution to ~/bin without runtime dependencies.
 - 5 ****Decision****
 - 6 Implement record-tui in Go. Single compiled binary handles all recording and HTML conversion tasks.... +8 lines (ctrl+o to expand)
- **Write**(adr/002-static-display-not-animated.md)
 - └ Wrote 19 lines to **adr/002-static-display-not-animated.md**
 - 1 # **ADR-002: Static Display Instead of Animated Playback**
 - 2 ****Status****: Accepted
 - 3 ****Context****
 - 4 Could display session as animated frames showing command execution in real-time (with timing.log), or as single static snapshot of final output.
 - 5 ****Decision****
 - 6 Use single PlaybackFrame with final session output. Renders instantly in browser without animation overhead.... +9 lines (ctrl+o to expand)
- **Write**(adr/003-posix-script-command.md)
 - └ Wrote 19 lines to **adr/003-posix-script-command.md**
 - 1 # **ADR-003: Use POSIX ``script`` Command for Recording**
 - 2 ****Status****: Accepted
 - 3 ****Context****
 - 4 Multiple approaches exist for capturing terminal sessions: custom PTY wrapper (like asciinema), system-specific recording, or built-in ``script`` command.
 - 5 ****Decision****
 - 6 Use POSIX ``script`` command via exec.Command() to record sessions. Captures terminal output with ANSI codes preserved.... +9 lines (ctrl+o to expand)
- **Write**(adr/004-xterm-js-cdn.md)
 - └ Wrote 19 lines to **adr/004-xterm-js-cdn.md**
 - 1 # **ADR-004: Load xterm.js from CDN**
 - 2 ****Status****: Accepted
 - 3 ****Context****
 - 4 Need proper terminal emulation in HTML to display ANSI colors and control sequences correctly. Could bundle as npm dependency or load from CDN.
 - 5 ****Decision****
 - 6 Load xterm.js@5.3.0 from jsDelivr CDN. Generated HTML is completely standalone except for this single external fetch.... +9 lines (ctrl+o to expand)
- **Write**(adr/005-metadata-stripping.md)
 - └ Wrote 19 lines to **adr/005-metadata-stripping.md**
 - 1 # **ADR-005: Line-Based Detection for Session Metadata Stripping**
 - 2 ****Status****: Accepted
 - 3 ****Context****
 - 4 ``script`` command adds header (Script started on, Command:) and footer (Script done on, Saving session) lines that need removal before displaying content.
 - 5 ****Decision****
 - 6 Parse session.log line-by-line, removing lines matching known patterns: "Script started on", "Command:", "Script done on", "Saving session".... +9 lines (ctrl+o to expand)
- **Write**(adr/006-auto-color-environment.md)
 - └ Wrote 18 lines to **adr/006-auto-color-environment.md**
 - 1 # **ADR-006: Automatically Set FORCE_COLOR and COLORTERM**
 - 2 ****Status****: Accepted

```
3 **Context**
4 Terminal applications (cargo, npm, etc.) detect PTY mode and disable colored output to avoid escape codes in pipes. Recording with `script` runs in PTY mode, losing colors
5 **Decision**
6 Automatically set FORCE_COLOR=1 and COLORTERM=truecolor before executing `script` command. Forces color output despite PTY detection.
... +8 lines (ctrl+o to expand)
```



● **Write**(adr/007-graceful-degradation.md)

```
└ Wrote 18 lines to adr/007-graceful-degradation.md
  1 # ADR-007: Graceful Degradation - Recording Over Conversion
  2 **Status**: Accepted
  3 **Context**
  4 Recording and HTML conversion are two separate steps. If conversion fails (e.g., template bug), should the entire operation fail?
  5 **Decision**
  6 Recording is the primary goal. If conversion fails, exit with warning but preserve session.log and exit code 0 (success). User can retry conversion manually.
... +8 lines (ctrl+o to expand)
```

● **Write**(adr/008-base64-frame-encoding.md)

```
└ Wrote 18 lines to adr/008-base64-frame-encoding.md
  1 # ADR-008: Base64 Encode Frame Data in HTML
  2 **Status**: Accepted
  3 **Context**
  4 Session content with ANSI codes needs to be embedded in HTML and decoded in JavaScript. Could use direct string, JSON, or binary-safe encoding.
  5 **Decision**
  6 Encode frames as base64 JSON in the HTML template. JavaScript decodes with atob()/Uint8Array before parsing.
... +8 lines (ctrl+o to expand)
```

● Perfect! Created all 8 ADRs in `/adr/` directory using a concise format:

- **Title** - descriptive, one-liner decision
- **Status** - Accepted (all are implemented)
- **Context** - why this decision matters (2-3 lines)
- **Decision** - what we chose (2-3 lines)
- **Consequences** -  benefits and  tradeoffs (bullet format)

The format is ~25-35 lines per ADR - enough detail to understand the decision without being verbose. Key decisions captured:

1. **Go** for portability & bundle size
2. **Static display** for simplicity
3. **POSIX script** to avoid custom PTY code
4. **xterm.js CDN** for standalone HTML
5. **Line-based parsing** to strip metadata
6. **Auto color env vars** for capturing colored output
7. **Graceful degradation** so conversion failures don't lose data
8. **Base64 encoding** for safe ANSI code embedding

You can reference these in code comments like `// See adr/001-language-choice-go.md` and they'll be available as project history when decisions need revisiting.