# Claude Code vs Codev: Round 1 Comparison

2026-02-13

## Contents

# Vibe Coding vs. SPIR: Todo Manager Comparison

**Date**: 2026-02-13 **PRs**: Vibe PR #1 | SPIR PR #1 **Reviewers**: Claude Opus 4.6, GPT-5.3 Codex, Gemini 3 Pro

## Methodology

Two Claude instances received the **identical prompt** to build a Todo Manager (Next.js 14+, TypeScript, localStorage, NL interface, Railway-ready). The only difference: the SPIR builder was told to use porch strict mode with 3-way multi-agent consultation at every checkpoint. Both ran with `--dangerously-skip-permissions` in fresh repos.

After both completed and submitted PRs, three independent AI reviewers (Claude, Codex, Gemini) reviewed each codebase blind — reading all source files, running bug sweeps, and rating seven dimensions.

---

## Scorecard

### Individual Reviewer Scores

| Dimension | Claude Vibe | Claude SPIR | Codex Vibe | Codex SPIR | Gemini Vibe | Gemini SPIR |
|---|---|---|---|---|---|---|
| Code Quality | 6 | 8 | 6 | 7 | 8 | 8 |
| Maintainability | 6 | 8 | 6 | 7 | 9 | 8 |
| Tests | 3 | 8 | 4 | 8 | 5 | 7 |
| Extensibility | 5 | 7 | 5 | 6 | 7 | 7 |
| NL Interface | 7 | 6 | 5 | 6 | 6 | 6 |
| Deployment | 5 | 7 | 4 | 8 | 9 | 9 |

### Averaged Scores

| Dimension | Vibe (avg) | SPIR (avg) | Delta |
|---|---|---|---|
| **Code Quality** | 6.7 | 7.7 | **+1.0** |
| **Maintainability** | 7.0 | 7.7 | **+0.7** |
| **Tests** | 4.0 | 7.7 | **+3.7** |
| **Extensibility** | 5.7 | 6.7 | **+1.0** |
| **NL Interface** | 6.0 | 6.0 | **0.0** |
| **Deployment** | 6.0 | 8.0 | **+2.0** |
| **Overall** | **5.9** | **7.3** | **+1.4** |

### Quantitative Comparison

| Metric | Vibe | SPIR |
|---|---|---|
| Source lines (excl. tests) | 916 | 1,596 |
| Test lines | 235 | 1,743 |
| Test-to-code ratio | 0.26:1 | 1.09:1 |
| Test files | 3 | 8 |
| Component tests | 0 | 288 lines |
| Integration tests | 0 | 196 lines |
| Git commits | 1 | 14 |
| Documentation artifacts | 0 | spec + plan + review + 30 consultation files |
| Dockerfile | No | Yes (multi-stage) |

---

## Bug Sweep Synthesis

All three reviewers independently reviewed the code. Below are the bugs **confirmed by 2+ reviewers** (high confidence) and those found by only one.

### Vibe Bugs (consensus)

| Bug | Severity | Found by | Description |
|---|---|---|---|
| **Broken `update` NL command** | Critical | Claude, Codex, Gemini | `nlParser.ts` fuzzy-matches the entire command body (including priority/date text) against todo titles. “update buy milk high priority” searches for “buy milk high priority” instead of “buy milk”. |
| **No localStorage error handling** | High | Claude, Codex, Gemini | `saveTodos()` in `storage.ts:18` does raw `setItem()` with no try/catch. `QuotaExceededError` crashes the app with no user feedback. |
| **No data validation on load** | Medium | Claude, Codex, Gemini | `loadTodos()` casts parsed JSON to `Todo[]` without schema checks. Corrupt/tampered data can break rendering. |
| **Aggressive prefix stripping** | Medium | Claude, Gemini | `stripIntentPrefix` removes command keywords. “Add task management” becomes an attempt to create “management” (or worse, interprets “update” in “Update the documentation” as a command). |
| **No delete confirmation** | Medium | Claude | Clicking delete immediately removes the todo. No undo, no confirmation dialog. |
| **Fuzzy match returns first, not best** | Medium | Claude, Gemini | `fuzzyMatch` returns the first substring match. “delete buy” with todos “buy groceries” and “buy milk” silently deletes whichever comes first. No disambiguation. |
| **No input validation** | Medium | Claude | No max-length on title/description. Users can paste arbitrary content. |

| Bug | Severity | Found by | Description |
|---|---|---|---|
| **No `test` script in package.json** | Low | Claude, Codex | `npm test` does nothing. CI/CD would fail. |
| **Font setup conflict** | Low | Codex | Layout imports Geist font but CSS overrides with Arial. |

**SPIR Bugs (consensus)**

| Bug | Severity | Found by | Description |
|---|---|---|---|
| **Date range filter not connected** | Critical | Codex, Gemini | NL commands like "show todos due this week" parse correctly but the `dateRange` filter is never applied to the UI. `NLInput` stores feedback only; `page.tsx` clears date range in `setFilters`. |
| **Data wipe on corrupt load** | High | Gemini | If `loadTodos()` fails (corrupt JSON), it returns empty array. `isLoaded` becomes true, triggering save — overwriting localStorage with `[]`. |
| **`cleanTitle` strips meaningful words** | Medium | Claude, Codex | `nl-parser.ts:101` strips "task", "todo" globally. "add task management" becomes "management". |
| **Limited date parsing** | Medium | Claude | Only supports "today", "tomorrow", weekdays, "this/next week". No specific dates ("March 15"), no relative dates ("in 3 days"). Vibe's `chrono-node` handles these. |
| **`updateTodo` silent no-op** | Medium | Codex | `useTodos.ts:103` returns `null` after map regardless of match. Updates to non-existent IDs report success. |
| **`onDelete` prop unused** | Low | Claude, Codex, Gemini | `TodoItem` receives `onDelete` but uses `onRequestDelete`. Dead code. |
| **ConfirmDialog hard-coded "Delete"** | Low | Codex, Gemini | Button text is always "Delete" even for non-delete confirmations. |

**Cross-cutting: Shared Weaknesses**

Both implementations share these problems: - **No multi-tab sync**: Neither listens for `storage` events. Opening two tabs causes stale state. - **Regex-based NL parsing**: Both use regex state machines that will become unmaintainable as commands grow. All three reviewers noted this. - **No accessibility testing**: Neither has ARIA compliance tests. - **No E2E browser tests**: Neither includes Playwright/Cypress tests.

---

## Architecture Comparison

### Vibe

- **State**: React Context + `useReducer` — clean, well-known pattern
- **NL**: Single `parseCommand()` function couples parsing AND execution (takes `todos[]`, returns `TodoAction`)
- **Storage**: Two bare functions (`loadTodos/saveTodos`), no error handling
- **Components**: 7 components, flat hierarchy, minimal prop drilling
- **Dependencies**: Uses `chrono-node` for date parsing (big advantage)

### SPIR

- **State**: Custom `useTodos` hook with validation, error surfacing, filtering, and sorting built in
- **NL**: Two-layer architecture — `parseNL` (pure parser) + `executeNL` (executor with deps injection). Independently testable.
- **Storage**: Typed errors (`StorageError`), availability checking, quota handling
- **Components**: 6 components + ConfirmDialog + EmptyState, more prop drilling but more testable
- **Dependencies**: No external date parser (limitation), but also no supply-chain risk

**Key architectural advantage of SPIR**: The parser/executor split. The parser is a pure function `string → NLCommand`. The executor takes `NLCommand + Dependencies → NLResult`. This makes both independently testable and swappable. The vibe version's `parseCommand` is a monolith that requires live todo state to produce results.

**Key architectural advantage of Vibe**: The `useReducer` pattern is naturally suited for undo/redo (state history stack). SPIR's `useState` in `useTodos` would need more refactoring for this.

---

## Test Quality Deep Dive

### Vibe (235 lines, 3 files)

- `nlParser.test.ts` (113 lines): 13 happy-path tests for add/complete/delete/show
- `storage.test.ts` (57 lines): 4 tests for round-trip, invalid JSON
- `todoReducer.test.ts` (65 lines): 5 tests for ADD/UPDATE/DELETE/TOGGLE/LOAD

**Not tested**: Components, date parsing, edge cases, filter behavior, NL update command (the broken one), error states.

**SPIR (1,743 lines, 8 files)**

- `nl-parser.test.ts` (224 lines): 20 tests including case variations, compound filters
- `nl-executor.test.ts` (248 lines): 12 tests including disambiguation, bulk delete
- `date-parser.test.ts` (93 lines): 11 tests with frozen time
- `storage.test.ts` (124 lines): 10 tests including quota exceeded, corrupted data
- `useTodos.test.ts` (558 lines): 28 tests covering filtering, sorting (6 scenarios), validation
- `components.test.tsx` (288 lines): 18 tests using Testing Library with user interactions
- `integration.test.ts` (196 lines): 8 end-to-end NL pipeline tests including multi-step workflow
- `setup.test.ts` (12 lines): Environment verification

**SPIR's standout**: The integration tests simulate the full NL pipeline from string input to store mutation, including a multi-step workflow test that creates, queries, updates, and deletes in sequence.

---

## NL Interface Comparison

| Capability | Vibe | SPIR |
|---|---|---|
| Add with priority | Yes | Yes |
| Add with flexible dates | **Yes** (chrono-node) | Limited (hardcoded keywords) |
| Complete by name | Yes (fuzzy) | Yes (substring) |
| Delete with confirmation | No | **Yes** |
| Bulk delete | No | **Yes** ("remove completed") |
| Change priority via NL | Broken | **Yes** |
| Set due date via NL | No | **Yes** |
| Filter by date range | No | Parsed but **not connected** |
| Disambiguation | No (first match) | **Yes** (shows options) |
| Mark as pending | No (toggle only) | **Yes** |
| Error with usage hints | No | **Yes** |

**Verdict**: SPIR has more NL features and better safety (disambiguation, confirmation). Vibe has significantly better date parsing via `chrono-node`. The ideal implementation would combine SPIR's architecture with Vibe's date parsing library.

---

## Deployment Readiness

| Aspect | Vibe | SPIR |
|---|---|---|
| `next.config.ts` standalone | Yes | Yes |
| Railway config | `railway.toml` (nixpacks) | No |

| Aspect | Vibe | SPIR |
|---|---|---|
| Dockerfile | No | **Yes** (multi-stage, non-root) |
| `.gitignore` | Yes | Yes |
| README | No | **Yes** |
| Health check | No | No |
| CI/CD | No | No |
| Test script in package.json | **No** | Yes |

SPIR's Dockerfile is production-grade: multi-stage build, non-root user (`nextjs:nodejs`), static assets separated, `HOSTNAME=0.0.0.0`. Vibe relies entirely on Railway's nixpacks, which is platform-locked.

---

## Reviewer Agreement Analysis

Where all three reviewers **agreed**: - SPIR has significantly better test coverage (unanimous) - Both NL parsers are regex-based and will become unmaintainable (unanimous) - Neither has XSS vulnerabilities (React auto-escaping protects both) - No real race conditions in either (single-tab, synchronous storage) - SPIR has better architecture (parser/executor split praised by all)

Where reviewers **disagreed**: - **Gemini rated Vibe maintainability 9/10** vs Claude/Codex at 6/10. Gemini emphasized the small codebase and simplicity; Claude/Codex penalized the lack of abstractions and NL coupling. - **Gemini rated Vibe deployment 9/10** vs Claude at 5/10 and Codex at 4/10. Gemini gave credit for Railway config; Claude/Codex penalized the missing Dockerfile and test script. - **Claude rated Vibe NL 7/10** vs Codex at 5/10. Claude gave credit for chrono-node's flexibility; Codex focused on the broken update command.

---

## Key Takeaways

1. **SPIR's biggest wins are in testing (+3.7) and deployment (+2.0)**. The structured methodology forced the builder to write comprehensive tests and a production Dockerfile. The vibe builder treated both as afterthoughts.
2. **NL Interface was a draw (6.0 vs 6.0)**. SPIR has more features (disambiguation, bulk ops, confirmation), but Vibe's `chrono-node` gives it significantly better date handling. Both are limited by regex parsing.
3. **SPIR costs ~74% more code** (1,596 vs 916 source lines) but delivers **7.4x more tests** (1,743 vs 235 test lines). The extra code buys validation, error handling, and safety features the vibe version lacks entirely.
4. **SPIR has a documentation trail** (spec + plan + review + 30 consultation artifacts). A new developer can understand *why* decisions were made. Vibe has a single commit with no rationale.
5. **Vibe's one clear advantage**: dependency choice. Using `chrono-node` for date parsing was a smart, pragmatic decision that SPIR's consultation-driven approach didn't produce. This

suggests CMAP excels at catching bugs and enforcing quality, but doesn't necessarily improve creative problem-solving or library selection.

6. **Both share the same fundamental limitation**: regex-based NL parsing. Neither reviewer panel suggested an LLM-based or proper NLP approach, which would be the real solution for production-quality natural language interfaces.

---

## Appendix: Raw Review Outputs

| Reviewer | Vibe | SPIR |
|---|---|---|
| Claude | Agent a4ff1d0 (deep comparison) | Agent a4ff1d0 (deep comparison) |
| Codex | `/tmp/codex-vibe-review.txt` | `/tmp/codex-spir-review.txt` |
| Gemini | `/tmp/gemini-vibe-review.txt` | `/tmp/gemini-spir-review.txt` |