Solo-Git Features → Heaven UI Mapping

> Complete mapping of Solo-Git's unique features to UI requirements for the Heaven Interface

Last Updated: 2025-10-20
Status: Phase 4 Refinement - Comprehensive Feature Integration

---

Table of Contents

Core Philosophy

Solo-Git Principles

```
┌─────────────────────────────────────────┐
│  Tests are the review                   │
│  Trunk is king                          │
│  Workpads are ephemeral                 │
│  Auto-merge on green                    │
│  No branches, no PRs, no waiting        │
└─────────────────────────────────────────┘
```

UI Translation

- **No persistent clutter** - UI appears only when needed
- **Test results are primary feedback** - Not human approval
- **Linear history visualization** - No complex merge graphs
- **Fast-forward only merges** - Simple, clean git graph
- **Ephemeral workspaces** - Visual distinction from branches

---

Workpads (vs Branches)

What Solo-Git Does Differently

Traditional Git Branches:

```
feature/add-login (persistent, named by user, manual lifecycle)
└─> git checkout -b
└─> git merge
└─> git branch -d
```

Solo-Git Workpads:

```
pad-abc123 "add login" (auto-named, ephemeral, auto-lifecycle)
└─> evogitctl pad create
└─> Auto-promoted on green tests ✅
└─> Auto-deleted after merge 🗑️
```

UI Requirements

1. CommitTimeline Enhancements

```
interface WorkpadVisualization {
  // Workpads should look different from branches
  type: 'workpad' | 'trunk' | 'tag';

  // Visual indicators
  ephemeralIndicator: boolean;  // Fade/dotted line
  autoPromoted: boolean;        // Special icon (✨)
  testStatus: 'pending' | 'running' | 'green' | 'red';

  // Auto-lifecycle
  ttl?: number;                 // Days until auto-delete
  willAutoMerge: boolean;       // Show pending promotion
}
```

Visual Design:
- **Trunk commits**: Solid line, primary color (#61AFEF blue)
- **Workpad commits**: Dotted line, secondary color (#98C379 green)
- **Auto-promoted commits**: Sparkle icon (✨) next to commit
- **Pending promotion**: Pulsing indicator on workpad head
- **Test-gated**: Lock icon (🔒) if tests not yet passed

2. Workpad Status Panel

Location: Contextual panel (appears when workpad active)

Content:

```
┌──────────────────────────────────────────┐
│ 📝 Workpad: "add login feature"          │
│ 🔑 pad-abc123                            │
│                                          │
│ Status: 🧪 Tests Running...              │
│ ├─ unit-tests:      ✅ 42 passed         │
│ ├─ integration:     🔄 running...        │
│ └─ e2e-tests:       ⏳ pending           │
│                                          │
│ Auto-Promote: Enabled ⚡                 │
│ TTL: 7 days                              │
│                                          │
│ [View Diff] [Cancel Tests] [Promote]     │
└──────────────────────────────────────────┘
```

Behavior:
- Fades in when workpad created
- Auto-updates during test execution
- Shows live test output (streaming)
- Fades out 3 seconds after promotion
- Dismissible with Esc

AI Orchestration

Multi-Model Intelligence

Solo-Git uses **Abacus.ai RouteLLM API** with intelligent model routing:

```
┌─────────────────────────────────────────────┐
│  Planning Tier                              │
│  ├─ GPT-4, Claude 3.5 Sonnet                │
│  ├─ Architecture, complex logic             │
│  └─ Cost: $$$                               │
├─────────────────────────────────────────────┤
│  Coding Tier                                │
│  ├─ DeepSeek-Coder, CodeLlama               │
│  ├─ Patch generation, refactoring           │
│  └─ Cost: $$                                │
├─────────────────────────────────────────────┤
│  Fast Tier                                  │
│  ├─ Llama 3.1 8B, Gemma 2 9B                │
│  ├─ Simple edits, boilerplate               │
│  └─ Cost: $                                 │
└─────────────────────────────────────────────┘
```

UI Requirements

1. AI Activity Indicator

Location: Status bar (contextual, minimal)

States:

```
type AIActivityState =
  | 'idle'           // Faded, barely visible
  | 'planning'       // 🧠 Pulsing purple
  | 'coding'         // ✍️ Pulsing blue
  | 'reviewing'      // 👁️ Pulsing green
  | 'diagnosing'     // 🔬 Pulsing yellow
```

Design:
- Icon changes based on activity
- Subtle pulse animation
- Shows model tier being used
- Fades after 3 seconds of completion

2. AI Cost Tracker

Location: Contextual panel (Cmd+Shift+C to show)

Content:

```
┌─────────────────────────────────────────┐
│ 💰 AI Cost Tracker                      │
│                                         │
│ Today: $2.47 / $10.00 (24%)            │
│ ████░░░░░░░░░░░░░░░░                   │
│                                         │
│ Breakdown:                              │
│ ├─ GPT-4:          $1.20 (48%)         │
│ ├─ DeepSeek:       $0.85 (34%)         │
│ └─ Llama 8B:       $0.42 (18%)         │
│                                         │
│ Recent Operations:                      │
│ ├─ Plan "add login"    $0.15 (GPT-4)   │
│ ├─ Generate patch      $0.08 (DeepSeek)│
│ └─ Review code         $0.03 (Llama)   │
└─────────────────────────────────────────┘
```

Behavior:
- Hidden by default
- Alert (toast) when reaching 80% of budget
- Auto-close after 5 seconds of no interaction

3. AI Assistant Panel

Component: `AICommitAssistant.tsx`

Purpose: AI-powered commit message generation and code review

Layout:

```
┌──────────────────────────────────────────────┐
│ ✨ AI Assistant                              │
│                                              │
│ Analyzing changes...                         │
│                                              │
│ Suggested Commit Message:                    │
│ ┌──────────────────────────────────────┐     │
│ │ feat: Add passwordless login         │     │
│ │                                      │     │
│ │ - Implement magic link auth          │     │
│ │ - Add email verification             │     │
│ │ - Update user model                  │     │
│ └──────────────────────────────────────┘     │
│                                              │
│ Confidence: 95% 🟢                           │
│                                              │
│ [Accept] [Edit] [Regenerate]                 │
└──────────────────────────────────────────────┘
```

Features:
- Floating panel (Cmd+Shift+A to show)
- Analyzes git diff
- Suggests commit message following conventions
- Shows AI confidence score
- One-click accept, edit, or regenerate
- Fades away after commit

Auto-Merge Workflow

The Pair Loop

```
Prompt → Plan → Code → Test → Auto-Merge
  │        │       │       │          │
  1s      4s     10s      20s         1s
  └───────────────────────────────────┘
         Total: <1 minute
```

Workflow Steps

1. **Create Workpad** (auto-named)
2. **AI Plans** (GPT-4/Claude)
3. **AI Generates Patch** (DeepSeek)
4. **Apply Patch to Workpad**
5. **Run Tests** (parallel, sandboxed)
6. **Analyze Results** (AI if failures)
7. **Auto-Promote** (if green) ✅
8. **CI Smoke Tests** (post-merge)

UI Requirements

1. Workflow Progress Indicator

Component: `WorkflowPanel.tsx`

Layout:

```
┌─────────────────────────────────────┐
│ 🔄 Active Workflow: "add login"     │
│                                     │
│ ┌────┬────┬────┬────┬────┐          │
│ │ ✅ │ ✅ │ 🔄 │ ⏳ │ ⏳ │          │
│ └────┴────┴────┴────┴────┘          │
│  Plan Code Test Gate Merge          │
│                                     │
│ Current: Running Tests (20/30s)     │
│ ████████████░░░░░░                  │
│                                     │
│ [Cancel] [Skip Tests] [View Details]│
└─────────────────────────────────────┘
```

Features:
- Shows workflow stages as horizontal pipeline
- Progress bar for current stage
- Estimated time remaining
- Click stage to see details
- Auto-collapses when complete
- Appears only when workflow active

2. Promotion Gate Visualization

Purpose: Show why auto-merge was/wasn't triggered

Green State:

```
┌─────────────────────────────────────┐
│ ✅ Promotion Gate: APPROVED         │
│                                     │
│ All checks passed:                  │
│ ✅ All tests green (42 passed)      │
│ ✅ No merge conflicts               │
│ ✅ Trunk is up to date              │
│ ✅ No CI failures                   │
│                                     │
│ Auto-promoting to trunk... ⚡       │
└─────────────────────────────────────┘
```

Red State:

```
┌─────────────────────────────────────┐
│ ❌ Promotion Gate: BLOCKED          │
│                                     │
│ Failed checks:                      │
│ ❌ 3 tests failed                   │
│ ✅ No merge conflicts               │
│ ✅ Trunk is up to date              │
│ ⚠️  CI unstable (2 recent failures) │
│                                     │
│ Fix tests to auto-promote           │
│ [View Failures] [Force Promote]     │
└─────────────────────────────────────┘
```

Behavior:
- Toast notification on promotion decision
- Detailed panel available (Cmd+Shift+G)
- Shows all gate rules and their status

CI/CD Integration

Jenkins-like Smoke Tests

Solo-Git has a built-in CI orchestrator that runs smoke tests **after** promotion to trunk.

Workflow

```
Workpad Promoted → CI Triggered → Smoke Tests → Success/Rollback
                                                   │
                                                   ├─ unit-tests
                                                   ├─ integration-tests
                                                   ├─ e2e-tests
                                                   └─ security-scan
```

UI Requirements

1. Pipeline Visualization

Component: `PipelineView.tsx`

Layout:

```
┌─────────────────────────────────────────────────────────────┐
│ 🔬 CI Pipeline: Build #384                                  │
│                                                             │
│ ┌─────────┬─────────┬─────────┬─────────┬─────────┐         │
│ │   ✅    │   ✅    │   ✅    │   ✅    │   ✅    │         │
│ │  Build  │  Unit   │  Integ  │   E2E   │ Deploy  │         │
│ │   5s    │   30s   │   45s   │  120s   │   10s   │         │
│ └─────────┴─────────┴─────────┴─────────┴─────────┘         │
│                                                             │
│ Status: SUCCESS ✅                                          │
│ Duration: 3m 30s                                            │
│ Commit: a9b8c7d "feat: Add login"                           │
│                                                             │
│ [View Logs] [Retry] [History]                               │
└─────────────────────────────────────────────────────────────┘
```

States:
- **Running**: Animated spinner, pulse effect
- **Success**: Green checkmark, subtle fade
- **Failed**: Red X, persist until acknowledged
- **Unstable**: Yellow warning, flaky tests

Behavior:
- Appears as overlay when pipeline triggered
- Auto-dismisses on success after 5 seconds
- Persists on failure

- Click stage to see logs
- Retry failed stages
- Cancel running pipeline

2. Build Status in CommitTimeline

Purpose: Show CI status next to commits in git graph

Visualization:

```
┌────────────────────────────────────┐
│ * a9b8c7d feat: Add login    ✅ 🏗️ │
│ |                                  │
│ * b8c7d6e fix: Bug in auth   ❌ 🏗️ │
│ |                                  │
│ * c7d6e5f chore: Update deps ✅ 🏗️ │
└────────────────────────────────────┘
```

Icons:
- ✅ - CI passed (green)
- ❌ - CI failed (red)
- 🔄 - CI running (animated)
- ⚠️ - CI unstable (yellow)
- 🏗️ - Build number (hover for details)

3. Test Results Panel

Component: `TestResultsPanel.tsx`

Layout:

```
┌───────────────────────────────────────────┐
│ 🧪 Test Results: Build #384               │
│                                           │
│ Summary:                                  │
│ ✅ 69 passed                              │
│ ❌ 0 failed                               │
│ ⏭️  0 skipped                             │
│ ⏱️  Duration: 2m 15s                      │
│                                           │
│ Test Suites:                              │
│ ✅ unit/auth.test.ts       42 passed      │
│ ✅ api/session.test.ts     27 passed      │
│ ✅ ui/plan-pane.test.ts    11 passed      │
│                                           │
│ [View Details] [Filter] [Export]          │
└───────────────────────────────────────────┘
```

Features:
- Slide-in panel from right
- Expandable test suites
- Click test to see assertion details
- Filter by status (passed/failed/skipped)
- Export results as JSON
- Auto-hides after viewing (Esc to close)

Test Orchestration

Parallel Execution

Solo-Git runs tests in **sandboxed parallel execution**:

```
Test Suite
├─ unit-tests     (30s, parallel)
├─ integration    (60s, parallel)
├─ e2e-tests      (180s, sequential)
└─ security-scan  (120s, parallel)
```

UI Requirements

1. Live Test Output

Purpose: Stream test results as they execute

Layout:

```
┌─────────────────────────────────────────┐
│ 🧪 Running Tests...                      │
│                                         │
│ [unit-tests] ✅ 42/42 passed (30s)       │
│ ├─ auth.test.ts              ✅ 12/12    │
│ ├─ session.test.ts           ✅ 15/15    │
│ └─ validation.test.ts        ✅ 15/15    │
│                                         │
│ [integration] 🔄 15/27 passed (45s)      │
│ ├─ api.test.ts               ✅ 10/10    │
│ ├─ database.test.ts          🔄 5/10...  │
│ └─ redis.test.ts             ⏳ 0/7      │
│                                         │
│ [e2e-tests] ⏳ 0/11 pending               │
│                                         │
│ [Cancel Tests]                          │
└─────────────────────────────────────────┘
```

Features:
- Real-time updates (websocket/polling)
- Progress bars for each suite
- Collapsible test suites
- Click test to see output
- Cancel tests mid-execution
- Auto-scroll to failing tests

2. Test Failure Analysis

Purpose: AI-powered diagnosis of test failures

Layout:

```
┌──────────────────────────────────────────┐
│ 🔬 Test Failure Analysis                  │
│                                          │
│ Failed: integration/database.test.ts     │
│                                          │
│ Error:                                   │
│ ┌──────────────────────────────────┐     │
│ │ AssertionError: expected 200     │     │
│ │ to equal 201                     │     │
│ │                                  │     │
│ │ at test/database.test.ts:45      │     │
│ └──────────────────────────────────┘     │
│                                          │
│ 🤖 AI Analysis:                           │
│ This looks like a status code mismatch.  │
│ The test expects 201 (Created) but got   │
│ 200 (OK). Check if the endpoint is       │
│ returning the correct status for POST.   │
│                                          │
│ Suggested Fix:                           │
│ ┌──────────────────────────────────┐     │
│ │ return res.status(201).json(...) │     │
│ └──────────────────────────────────┘     │
│                                          │
│ [Apply Fix] [Dismiss] [Re-run Test]      │
└──────────────────────────────────────────┘
```

Features:
- AI analyzes stack trace
- Suggests fixes
- One-click apply suggested fix
- Re-run individual test
- Escalates to planning model for complex failures

Heaven Interface Modes

Solo-Git has **3 interface modes** (all share state via JSON):

1. Enhanced CLI

Rich formatting with Python Rich library:
- Colored output
- Panels and boxes
- ASCII commit graphs
- Progress bars
- Tables

2. Interactive TUI

Full-screen terminal with Textual framework:
- Keyboard-driven
- Command palette
- File tree

- Commit graph
- Live updates

3. Desktop GUI

Tauri app (Rust + React) - **This is what we're building!**

UI Architecture

```
┌──────────────────────────────────────────────┐
│                   Heaven GUI                  │
│                                              │
│                                             │
│  ┌─────────┬──────────────┬──────────────┐  │
│  │ Left    │   Center     │   Right      │  │
│  │ Rail    │   Stage      │   Rail       │  │
│  ├─────────┼──────────────┼──────────────┤  │
│  │ File    │   Monaco     │   AI         │  │
│  │ Tree    │   Editor     │   Assistant  │  │
│  │         │              │              │  │
│  │ Commit  │   Code       │   Test       │  │
│  │ Graph   │   (center    │   Results    │  │
│  │         │   stage)     │              │  │
│  └─────────┴──────────────┴──────────────┘  │
│                                             │
│  ┌──────────────────────────────────────┐  │
│  │             Bottom Panel              │   │
│  │  Logs, Terminal, Test Output         │   │
│  └──────────────────────────────────────┘  │
└──────────────────────────────────────────────┘
```

Design Tokens

From Heaven Interface Design System:

```css
/* Colors */
--heaven-bg: #1E1E1E;
--heaven-text: #DDDDDD;
--heaven-blue: #61AFEF;
--heaven-green: #98C379;
--heaven-red: #E06C75;
--heaven-yellow: #E5C07B;
--heaven-purple: #C678DD;

/* Typography */
--font-code: 'JetBrains Mono', 'SF Mono', monospace;
--font-ui: 'SF Pro', 'Roboto', sans-serif;

/* Spacing (8px grid) */
--space-1: 8px;
--space-2: 16px;
--space-3: 24px;

/* Animations */
--transition-fast: 150ms ease-in-out;
--transition-normal: 300ms ease-in-out;
```

UI Component Requirements

New Components Needed

1. `Toast.tsx` ✨ Priority: HIGH

- Fading notifications
- Auto-dismiss after 3-5s
- Stack vertically
- Types: success, error, warning, info

2. `AICommitAssistant.tsx` ✨ Priority: HIGH

- Floating panel
- AI-generated commit messages
- Confidence score
- Accept/edit/regenerate

3. `WorkflowPanel.tsx` ✨ Priority: HIGH

- Horizontal pipeline stages
- Progress indicator
- Real-time updates
- Contextual (appears only when active)

4. `PipelineView.tsx` ✨ Priority: HIGH

- Jenkins-like visualization
- Stage status indicators
- Click to see logs
- Retry/cancel actions

5. `TestResultsPanel.tsx` ✨ Priority: MEDIUM

- Slide-in from right
- Expandable test suites
- Filter by status
- Live updates

6. `AIActivityIndicator.tsx` ✨ Priority: MEDIUM

- Minimal status bar widget
- Shows model tier
- Pulse animation
- Fades when idle

7. `PromotionGatePanel.tsx` ✨ Priority: MEDIUM

- Shows gate rules
- Check status (✅/❌)
- Reason for approval/blocking

Component Modifications Needed

1. `CommitTimeline.tsx` 🔄 Priority: HIGH

- Distinguish workpads from trunk
- Show AI-assisted commits (✨)

- Show test status on commits
- Show CI build status
- Fade timeline to 10% opacity
- Auto-hide after 5s inactivity

2. `StatusBar.tsx` 🔄 Priority: HIGH

- Add AI activity indicator
- Add CI build status
- Make semi-transparent (80%)
- Contextual indicators only

3. `CommandPalette.tsx` 🔄 Priority: HIGH

- Add Solo-Git commands:
- `Pair: Start AI Pairing`
- `Workpad: Create`
- `Workpad: Promote`
- `Tests: Run`
- `CI: View Pipeline`
- `AI: Commit Message`

4. `FileExplorer.tsx` 🔄 Priority: MEDIUM

- Show git status on files
- Contextual search (Cmd+F)
- Minimal chrome (no borders)
- Hover to reveal actions

5. `CodeEditor.tsx` 🔄 Priority: MEDIUM

- Contextual header (show on hover)
- Fade minimap to 30%
- Hide line numbers until gutter hover

Feature Implementation Priority

Phase 1: Core “No UI” Refinement (Tasks 3-9)

Goal: Simplify existing components, harmonize design

1. ✅ Audit current UI for clutter
2. ✅ Simplify all components
3. ✅ Harmonize colors, spacing, shadows
4. ✅ Implement contextual visibility

Phase 2: Notification System (Tasks 10-12)

Goal: Replace persistent indicators with toasts

1. ✅ Create Toast component
2. ✅ Create notification manager
3. ✅ Replace all persistent indicators

Phase 3: Solo-Git Core Features (Tasks 13-16)

Goal: Integrate workpads, AI, git graph

1. ✅ AI Commit Assistant
2. ✅ Workflow Panel
3. ✅ Enhanced Git Graph
4. ✅ CommandPalette extension

Phase 4: CI/CD Visualization (Tasks 17-19)

Goal: Jenkins-like pipeline view

1. ✅ Pipeline View component
2. ✅ Build status integration
3. ✅ Test Results Panel

Phase 5: Contextual UI Patterns (Tasks 20-22)

Goal: Show-on-demand, hover-to-reveal, focus mode

1. ✅ useContextualVisibility hook
2. ✅ Hover patterns
3. ✅ Focus Mode

Phase 6: Polish & Testing (Tasks 23-30)

Goal: Animations, accessibility, performance, validation

1. ✅ Animation refinement
2. ✅ Accessibility
3. ✅ Performance optimization
4. ✅ Documentation
5. ✅ Testing and validation

Success Criteria

Visual Harmony ✅

- [ ] Consistent 8px spacing grid
- [ ] Unified color palette
- [ ] Harmonized shadows and borders
- [ ] Consistent typography

“No UI” Philosophy ✅

- [ ] No persistent clutter
- [ ] Contextual information only
- [ ] Smooth fade in/out
- [ ] Every element has clear purpose

Solo-Git Integration ✅

- [ ] Workpads visually distinct from branches

- [ ] AI operations tracked and visible
- [ ] Auto-merge workflow visualized
- [ ] Test-driven promotion clear

CI/CD Features ✅

- [ ] Pipeline visualization working
- [ ] Build status on commits
- [ ] Test results accessible
- [ ] Rollback mechanism visible

Performance ✅

- [ ] TypeScript checks pass
- [ ] Production build succeeds
- [ ] All animations < 150ms
- [ ] No layout thrashing

Next Steps

1. ✅ Complete this documentation
2. ➡️ Begin Phase 2: UI Audit and Simplification
3. ➡️ Implement notification system
4. ➡️ Integrate Solo-Git features
5. ➡️ Build CI/CD visualization
6. ➡️ Polish and validate

Notes:
- This is a living document - update as features evolve
- Solo-Git is in Phase 4 (beta prep) - features are stable
- Heaven GUI should showcase Solo-Git's unique workflow
- Focus on "tests as review" paradigm throughout UI

References:
- Solo-Git README (/home/ubuntu/code_artifacts/solo-git/README.md)
- Heaven Interface Design System (docs/HEAVEN_INTERFACE.md)
- Solo-Git CLI Reference (/home/ubuntu/code_artifacts/solo-git/sologit/cli/main.py)
- AI Orchestrator (/home/ubuntu/code_artifacts/solo-git/sologit/orchestration/ai_orchestrator.py)
- Auto-Merge Workflow (/home/ubuntu/code_artifacts/solo-git/sologit/workflows/auto_merge.py)