/plan-build-test

Source: `~/.claude/skills/plan-build-test/SKILL.md`

---

name: plan-build-test version: "2.0" level: 3 trigger: "plan-build-test, full-cycle test, playwright, E2E testing, run tests" author: john updated: 2026-03-16 description: Orchestration skill for Plan?Build?Test development cycles. Runs Playwright CLI tests (NOT MCP) against local or remote web apps. Supports E2E testing, visual regression, and mobile viewport testing.

Plan-Build-Test Orchestration Skill

Automates the full development cycle: implement changes → build → test → fix → re-test.

CRITICAL: Playwright CLI ONLY — NEVER use MCP playwright tools. All testing via `npx playwright test` or `./scripts/test-runner.sh`.

Modes

Mode 1: Full Cycle (`\plan-build-test:full-cycle`)

Purpose: Implement feature/fix → build → test → fix failures → visual regression

Agent workflow:

1. **Read requirements**
   - Read task description and acceptance criteria
   - Identify files to change and expected test coverage
2. **Spawn builder subagent**
   - Use Task tool to spawn builder agent with clear file ownership
   - Wait for builder to complete implementation
   - Verify builder marked task as done
3. **Build verification**
   - Run build command: `npx next build` (or relevant for project)
   - Parse output for errors
   - If build fails → analyze errors → spawn builder to fix → re-build
   - Max 3 build iterations before escalating
4. **Start dev server (if testing locally)**
   - If TEST_BASE_URL not set, start dev server: `npx next dev &`
   - Wait for server ready (check http://localhost:3000)
   - If testing remote URL, skip this step
5. **Run E2E tests**
   - Execute: `./scripts/test-runner.sh [--project <project>] [--grep <pattern>]`
   - Parse JSON results from `/tmp/playwright-results.json`
   - Capture:
     - Total tests, passed, failed, skipped
     - Failure details (test title, error message)
     - Screenshot paths from `/tmp/playwright-screenshots/`
6. **Fix failures (if needed)**
   - If tests fail:
     - Analyze failure details and screenshots
     - Identify root cause
     - Spawn builder to fix issues
     - Re-run tests
   - Max 3 fix iterations before escalating

7. **Visual regression (optional)**
   - If changes affect UI:
     - Run: `./scripts/visual-regression.sh`
     - Compare against baseline
     - Report diff percentages
     - Show paths to diff images
   - If no baseline exists:
     - Capture baseline: `./scripts/visual-regression.sh --baseline`
     - Skip comparison (first run)
8. **Report summary**
   - Build status: pass/fail
   - Test results: X/Y passed
   - Failure details (if any) with screenshot references
   - Visual regression status (if run)
   - Next steps or completion confirmation

Variables:

- `{{TASK_DESCRIPTION}}` — What to implement
- `{{PROJECT_DIR}}` — Project root path
- `{{BASE_URL}}` — URL to test (default: http://localhost:3000)
- `{{MAX_ITERATIONS}}` — Max fix attempts (default: 3)

Example usage:

```
\plan-build-test:full-cycle
Task: Implement login form validation
Project: /Users/makinja/ALAI/products/Drop/src/drop-app
Base URL: http://localhost:3000
```

Mode 2: Test Only (`\plan-build-test:test-only`)

Purpose: Run tests against existing deployment (local or remote) without building

Agent workflow:

1. **Accept parameters**
   - URL to test (required, default: http://localhost:3000)
   - Project filter (optional, e.g., "mobile-iphone")
   - Test grep pattern (optional, e.g., "login")
2. **Run tests**

- Execute: `TEST_BASE_URL=<url> ./scripts/test-runner.sh [--project <project>] [--grep <pattern>]`
- Parse results from `/tmp/playwright-results.json`

3. **Report results**
   - Summary: X/Y tests passed
   - If failures:
     - Show failure details (test title + error message)
     - List screenshot paths from `/tmp/playwright-screenshots/`
   - Exit code: 0 = all pass, 1 = failures

Variables:

- `{{BASE_URL}}` — URL to test
- `{{PROJECT}}` — Project filter (optional)
- `{{GREP_PATTERN}}` — Test name filter (optional)

Example usage:

```
\plan-build-test:test-only
URL: https://staging.getdrop.no
Project: mobile-iphone
Pattern: login
```

Mobile testing:

- iPhone viewport: `--project mobile-iphone`
- Galaxy viewport: `--project mobile-galaxy`
- iPad viewport: `--project tablet-ipad`

Mode 3: Visual Check (`\plan-build-test:visual-check`)

Purpose: Capture screenshots and compare against baseline for visual regression detection

Agent workflow:

1. **Check baseline status**
   - Check if baseline exists: `ls tests/visual/baseline/*.png`
   - If no baseline → capture baseline mode
   - If baseline exists → comparison mode
2. **Capture baseline (first run)**
   - Execute: `./scripts/visual-regression.sh --baseline`
   - Saves screenshots to `tests/visual/baseline/`

- Report: "Baseline captured, X screenshots saved"
- Skip comparison (nothing to compare against)

3. **Run comparison (subsequent runs)**
   - Execute: `./scripts/visual-regression.sh [--threshold <percent>]`
   - Default threshold: 5% (customizable)
   - Compares current screenshots vs baseline
   - Generates diff images to `/tmp/visual-diffs/`
4. **Report results**
   - Per-page diff percentages
   - Overall status: pass (no diffs > threshold) or fail (diffs detected)
   - Paths to diff images for review
   - Recommendation: approve new baseline or fix regressions

Variables:

- `{{PROJECT_DIR}}` — Project root path
- `{{THRESHOLD}}` — Max diff percentage allowed (default: 5)

Example usage:

```
\plan-build-test:visual-check
Threshold: 10
```

Workflow:

1. First run: Capture baseline
2. Make UI changes
3. Run visual check → see diffs
4. Review diff images
5. If intentional → update baseline: `./scripts/visual-regression.sh --baseline`
6. If bugs → fix issues → re-run visual check

Key Constraints

1. **Playwright CLI ONLY**
   - NEVER use MCP playwright tools
   - All tests via `npx playwright test` or wrapper scripts
   - No browser automation except through Playwright CLI
2. **URL flexibility**
   - Support local dev: http://localhost:3000
   - Support staging: https://staging.example.com
   - Support production: https://example.com
   - Use TEST_BASE_URL env var to override default

3. **Mobile testing**
   - Use `--project` flag for mobile viewports
   - Available projects: mobile-iphone, mobile-galaxy, tablet-ipad
   - See playwright.config.ts for full project list
4. **JSON results parsing**
   - Always parse `/tmp/playwright-results.json` for structured data
   - Extract: total, passed, failed, skipped, failures[]
   - Reference screenshot paths from `/tmp/playwright-screenshots/`
5. **Screenshot evidence**
   - All failure screenshots saved to `/tmp/playwright-screenshots/`
   - Visual regression diffs saved to `/tmp/visual-diffs/`
   - Include paths in reports for manual review
6. **Iterative fixing**
   - Max 3 iterations for build fixes
   - Max 3 iterations for test fixes
   - After max iterations → escalate to human with detailed failure analysis
7. **Build before test**
   - Full cycle MUST run build before tests
   - Test-only mode assumes build already done
   - Visual check mode can run independently (screenshot capture doesn't require build)

File Locations

- **Test runner:** `./scripts/test-runner.sh`
- **Visual regression:** `./scripts/visual-regression.sh`
- **Playwright config:** `playwright.config.ts`
- **Test results:** `/tmp/playwright-results.json`
- **Screenshots:** `/tmp/playwright-screenshots/`
- **Visual diffs:** `/tmp/visual-diffs/`
- **Visual baseline:** `tests/visual/baseline/`

Example Outputs

Full Cycle Success

```
Task #1234 COMPLETE

Build: ✓ Passed
```

```
Tests: ✓ 15/15 passed
Visual regression: ✓ No changes detected (all diffs < 5%)

Ready for deployment.
```

Full Cycle with Failures

```
Task #1234 — Test failures detected

Build: ✓ Passed
Tests: ✗ 12/15 passed (3 failures)

Failures:
1. "login with valid credentials" — Error: Element not found: button[type="submit"]
   Screenshot: /tmp/playwright-screenshots/login-failure-1.png

2. "dashboard loads after login" — Error: Timeout waiting for selector: h1:has-
text("Dashboard")
   Screenshot: /tmp/playwright-screenshots/dashboard-timeout-2.png

Fix iteration 1/3 in progress...
```

Test Only (Remote)

```
Testing: https://staging.getdrop.no
Project: mobile-iphone
Results: ✓ 8/8 passed

All tests passed on mobile viewport.
```

Visual Check

```
Visual regression results:

✓ login.png — 0.2% diff (PASS)
✓ dashboard.png — 1.8% diff (PASS)
✗ profile.png — 12.5% diff (FAIL — exceeds 5% threshold)
```

```
Review diff: /tmp/visual-diffs/profile-diff.png


Action needed: Review profile page changes or update baseline if intentional.
```

? Operational Limits

- **MAX TURNS:** 30 (build) | 20 (validate) | 10 (lookup)
- Exit cleanly after completing. On 5+ failures: escalate to John with full error context.

Revision #6
Created 2026-02-18 08:42:06 UTC by John
Updated 2026-07-26 20:01:41 UTC by John