Android Bench: Measuring Frontier Coding Agents on Long-Horizon Mobile Engineering

Ben Trengrove Lynn Ou Siva Velusamy

November 2026

Abstract

Android Bench 2.0 measures coding agents on long Android engineering tasks. It has 30 tasks written by Android developers: new app development from design mocks, library migrations in open-source apps, new features in existing apps, and conversions of Flutter and React Native apps to Jetpack Compose. The human-written solutions change a median of 2,336 lines across 42 files. Each task runs against a live Android emulator. Deterministic tests check behavior and stored data, and an LLM judge compares screenshots and accessibility trees with reference screens. Every run gets a partial score from 0 to 1 as well as pass or fail. Each task has a closed variant, where the tests are hidden, and an open variant, where the agent can run them. The best model on the closed variant, claude-sonnet-5-5, passes 41% of tasks with a mean partial score of 0.85. With tests available, the best pass rate is 84% (claude-fable-5-1). Most failed runs miss a requirement rather than fail to build.

Table of contents

1. Introduction . . . 3
2. Related Work . . . 4
    2.1 Coding agent benchmarks . . . 4
    2.2 Visual testing and LLM judges . . . 4
3. Dataset overview . . . 4
    3.1 Task categories . . . 4
    3.2 Task size . . . 5
    3.3 Infrastructure . . . 6
4. Verification and scoring . . . 6
    4.1 LLM judges . . . 6
    4.2 Implementation-independent tests . . . 7
    4.3 Scoring . . . 8
5. Results . . . 10
    5.1 Leaderboard . . . 10
    5.2 Closed and open variants . . . 11
    5.3 Cost and time . . . 12
6. Agent behavior . . . 14
    6.1 The last 20% of the work . . . 14
    6.2 Cheating and shortcuts . . . 14
    6.3 Failure taxonomy . . . 15
    6.4 Task difficulty and patch size . . . 16
    6.5 How agents spend their steps . . . 16
7. Statistical analysis . . . 18
    7.1 Pairwise tests between models . . . 18
    7.2 Variance components . . . 19
    7.3 Power analysis . . . 20
    7.4 Signal-to-noise ratio . . . 21
8. Limitations and future work . . . 21
9. Conclusion . . . 21
Acknowledgements . . . 22
References . . . 22
Appendix A: Multimodal Verification Prompts . . . 23
    A.1 LLM Visual Judge Evaluation Prompt . . . 23
    A.2 Anti-Cheating Inspection Prompt . . . 23
Appendix B: Screenshot Testing Evaluation . . . 23
    Metrics . . . 23
    Three-pane comparisons (golden / difference / agent output) . . . 24
    Side-by-side comparisons (golden / agent output) . . . 36

1. Introduction

Standard software engineering benchmarks evaluate coding agents on localized, single-issue repairs. In benchmarks such as SWE-bench [@jimenez2024swebench], an agent inspects a repository snapshot, changes a few lines in one or two files, and is graded by existing tests run in a terminal. This setup misses much of what mobile engineering involves:

- **Build systems and dependency graphs**: Production Android applications require resolving multi-module dependency graphs, annotation processors, Gradle build lifecycles, and strict compilation constraints. A syntax error in one interface can break the build of hundreds of downstream compilation units.
- **Asynchronous lifecycles and separated state**: Modern Android applications separate presentation, domain logic, and persistence layers using architectural patterns such as Model-View-ViewModel (MVVM), unidirectional data flow (UDF), and reactive asynchronous streams (Coroutines and StateFlow). Agents must reason about lifecycle transitions, background workers, and persistent disk state.
- **Visual verification**: Android apps have user interfaces. Code that runs without exceptions can still render the wrong layout, show the wrong image, or ignore input.
- **Long tasks**: Real engineering tickets take days or weeks. Migrating a framework (e.g., RxJava to Kotlin Coroutines) or converting a cross-platform app to Jetpack Compose means editing dozens of files consistently over hundreds of agent steps.

Our first benchmark covered some of these areas, but frontier models now pass most of its tasks. **Android Bench 2.0** evaluates coding agents on long, real-world Android engineering tasks.

Android Bench runs each task in a container with an Android virtual device (AVD). It builds the agent's APK, installs it on the emulator, drives the UI, and inspects the screen, accessibility tree, and database.

Android Bench 2.0 includes:

1. **Task set**: 30 tasks written by Android developers and kept out of public training data: new app development from design mocks, library migrations in open-source apps (Signal, Pocket Casts, WordPress), new features, and cross-platform app conversions. The largest human-written solution changes 8,240 lines.
2. **Emulator-based verification**: Automated UI walkthroughs on a live emulator, graded by deterministic assertions plus LLM judges of screenshots and accessibility trees.
3. **Partial scoring**: A score from 0.0 to 1.0 that combines functional, visual, requirements, and regression checks.
4. **Closed and open variants**: Each task runs with hidden tests and with visible tests, which shows how much agents gain from running tests.
5. **Cheating detection**: A catalog of shortcuts agents took, such as full-screen screenshot backplates, deleted test assertions, and relaxed screenshot diff thresholds, and the checks that now catch them.

2. Related Work

2.1 Coding agent benchmarks

Mobile coding benchmarks and Android Bench v1. Early repository-level benchmarks focused on Python and web projects. *Android Bench v1* (developer.android.com/bench) was the first standard benchmark for Android development. It used open-source Android repositories and tested whether agents could resolve GitHub issues, such as fixing a crash from a stack trace or adding a small feature, graded pass or fail by unit and instrumentation tests.

Frontier models now pass about 90% of v1 tasks on our leaderboard.

DeepSWE [@datacurve2026deepswe] and FrontierCode [@cognition2026frontiercode] report the same for localized bug-fix tasks and moved to tasks that span more files and take longer.

	Android Bench 2.0	**SWE Bench Pro (Public dataset)**	**Deep SWE 1.1**	**Frontier Code (Main)**
Task Count	30	731	113	100
Contamination Risk	Low: tasks are written from scratch.	Medium: Tasks from copyleft licensed repositories	Low: tasks are written from scratch	High: PRs from open source projects
Programming Languages	Kotlin, Java, Dart, Typescript, build scripts	Python, Go, Typescript, Javascript	TypeScript, Go, Python, JavaScript, Rust	Python, Go, Typescript, Javascript, Java, C++ and a few more
Median Solution Lines Edited	2,336	94	642	308
Median Solution Files Touched	42	4	6	6
Highest Pass Rate (Sep '26)	claude-sonnet-5-5 (41%) claude-opus-5-5 (37%) gpt-6.1-sol (29%)	Fable 5.1 (81%) Mythos 5 (80%) Fable 5 (80%)	GPT 6 Astra (74%) Gemini 3.8 Flash (74%) Opus 5 (74%)	Fable 5 (59%) Opus 5 (59%) GPT 6 Astra (59%)
Tests Multi-modal capabilities	Yes	No	No	No

2.2 Visual testing and LLM judges

Automated visual regression testing in industry typically relies on pixel-by-pixel comparisons (e.g., Dropshots, Paparazzi, Roborazzi) or structural similarity algorithms (SSIM [@wang2004ssim], LPIPS [@zhang2018lpips]). Pixel matchers work poorly for grading agent-written code. Harmless differences in the status bar clock, font anti-aliasing, or system icons produce large diffs.

Multimodal models have been used as visual evaluators [@liu2023llmjudge]. Android Bench combines LLM grading of screenshots with checks of the accessibility tree.

3. Dataset overview

3.1 Task categories

Android Bench has 30 tasks in four categories. Each category was designed to keep solutions out of public training data:

Category	**Count**	**Focus**
App creation	9	In-house new app development (`Food Vibes`) using visual mocks for design
Migrations	13	Upgrading production open-source apps to modern libraries and architectural patterns
New Features	6	Implementing new features and platform integrations into established production codebases
App Conversions	2	Converting cross-platform applications (Flutter, React Native) to native Android

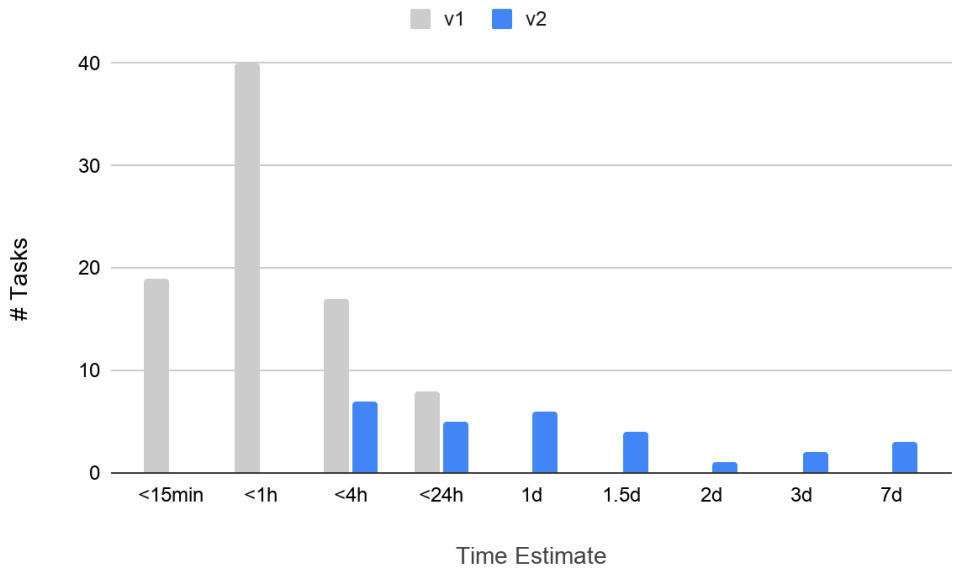


Figure 1: Human time estimates for v1 and v2 tasks.

Task creation and review

Android developers wrote each task by hand, checked that it was realistic and solvable, and estimated how long it would take a person.

Tasks went through nine rounds of trajectory review. Reviewers read every task specification and test suite. Reviewers removed hints that gave away step-by-step code structures (overspecification) and ensured tests did not expect hidden behaviors absent from the instructions (underspecification).

Each round of review tested the benchmark against solutions written by multiple model families (Claude, GPT, and Gemini). When a model generated working code that failed verification due to an overly rigid test, engineers updated the test to focus strictly on functionality. Reviewers also audited tests to prevent agents from passing with hardcoded static data or fake mocks.

Contamination prevention

To keep solutions out of public training data:

- In-house tasks are built inside internal repositories (`Food Vibes`) with zero public presence.
- Library migrations target modern, recently stabilized Android libraries (e.g., Navigation 3, Coil 3, Ktor 3) where the migration does not exist in the target repository.
- App conversions require translating Flutter or React Native trees into native Jetpack Compose where no implementation exists.

3.2 Task size

v1 solutions averaged about 100 lines of code. Android Bench 2.0 solutions are an order of magnitude larger and take far longer for a person to write (Figure 1, Figure 2).

Tasks range from single-module library swaps (e.g., Now in Android Retrofit to Ktor) to multi-module rewrites (e.g., Signal Java to Kotlin, 8,240 lines across 81 files, and Pocket Casts Retrofit to Ktor, 3,685 lines across 125 files).

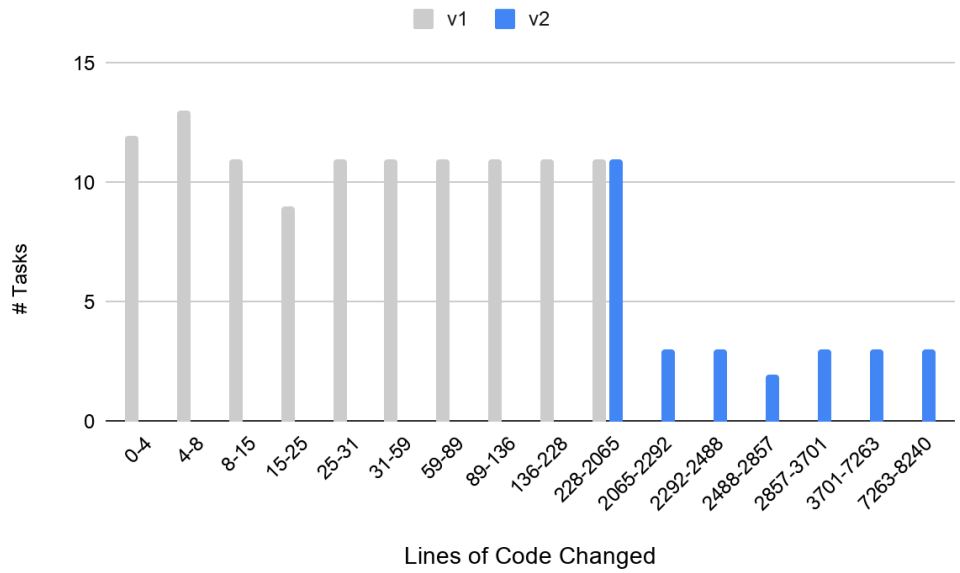


Figure 2: Lines changed in the human-written solution for v1 and v2 tasks.

3.3 Infrastructure

Tasks are packaged under the **Harbor framework** within isolated Docker containers.

Tasks boot Android virtual devices, so hosts need KVM-enabled CPUs and a container with at least:

- CPUs: 16
- Memory: 72GB
- Storage: 500GB

4. Verification and scoring

Tests for open-ended tasks must check functional and visual correctness without dictating how the agent implements the change or penalizing small cosmetic differences. Figure 3 shows how the checks combine.

A partial score of 1.0 equals a pass for pass@1.

4.1 LLM judges

Standard visual regression tools (e.g. screenshot testing frameworks like Dropshots) calculate percentage pixel difference against golden snapshots. In our testing, pixel-diffing frameworks failed as automated benchmark judges:

- **System noise**: Harmless differences, such as the status bar clock changing from `10:00` to `10:02`, a different battery icon, or font anti-aliasing, produced pixel diffs of up to **47.5%** on otherwise correct screens. See Appendix B.
- **Implementation differences**: Small padding or tint differences fail the test even when the screen works correctly.

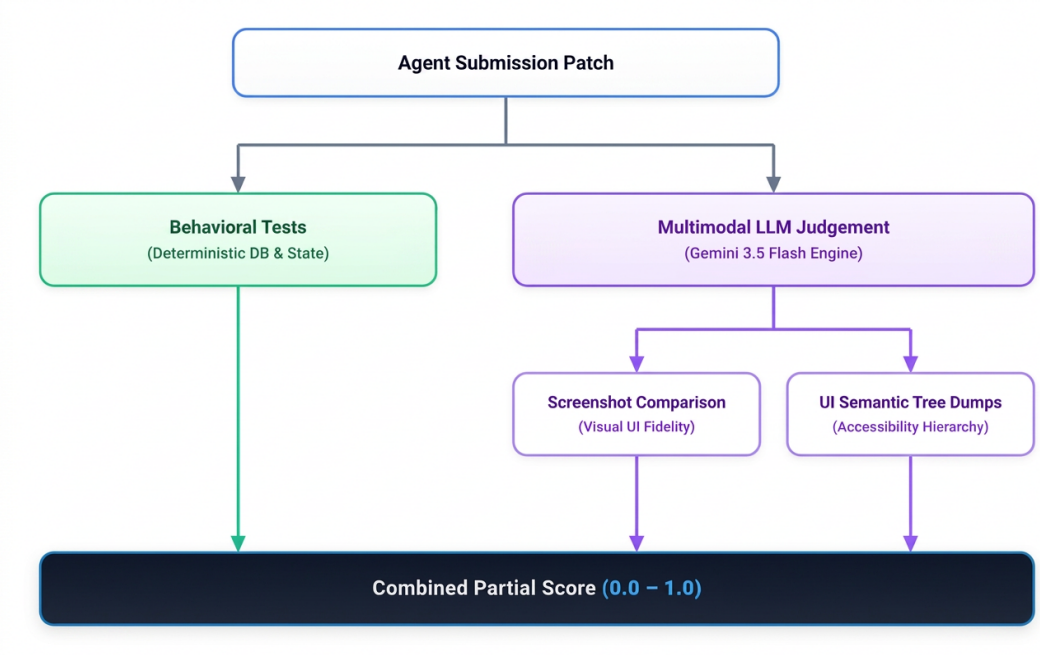


Figure 3: Verification pipeline. Deterministic tests and LLM judges of screenshots and accessibility trees combine into one partial score.

Android Bench uses an LLM judge instead. It compares a pair of screenshots and returns a grade from 0.0 to 1.0 with its reasoning. Appendix A has the full prompt.

In 360 calibration trials, Gemini 3.5 Flash gave the same grade on every repeat. It ignored status bar and anti-aliasing differences and penalized missing cards, broken layouts, and distorted images. See Appendix B.

To stop agents from showing static screenshots instead of a working UI, the verifier also captures the accessibility hierarchy (`dumpsys accessibility`).

An LLM judge verifies:

- Presence and hierarchy of native composables and views.
- Structural parity of text labels, buttons, and navigation elements.
- Touch target validity and screen accessibility semantics.

4.2 Implementation-independent tests

Tests check results, not internal structure:

- **Visible behavior and database state**: Automated tests interact with the app through public UI screens and accessibility tags. Where relevant, tests inspect the SQLite and Room database directly to verify that data was saved properly, regardless of how the agent structured its internal helper classes or DAOs.
- **External signals and regressions**: Tests check external system boundaries, such as whether outbound network calls send the expected data, intents fire with the right extras, and Wear OS companion events sync. The repository's existing test suite runs alongside new checks to make sure the agent did not break existing features.

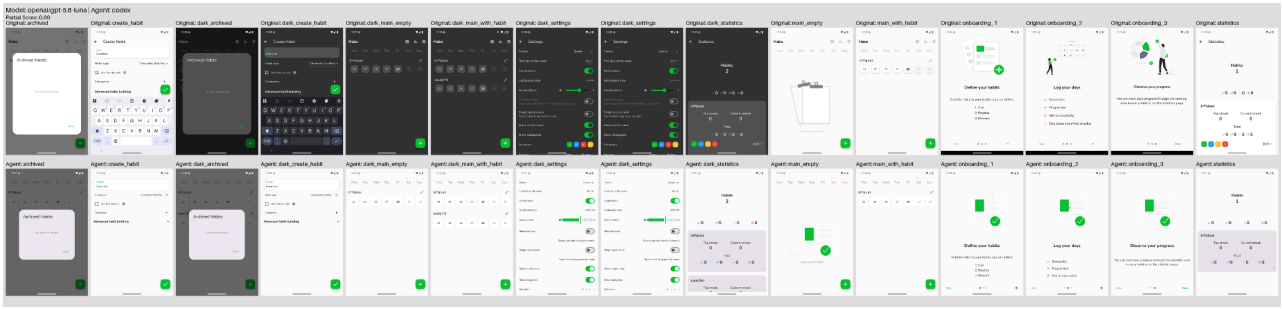


Figure 4: GPT-5.6 Luna, partial score 69%.

4.3 Scoring

Pass rate is the headline metric, but on long tasks a model can make a lot of progress without passing. A partial score from 0.0 to 1.0 records that progress.

The base score is the normalized weighted sum across four quality metrics, with each metric given a weight by the task author:

- **Functional**: Validates runtime state changes, database persistence, and logic.
- **Regression**: Ensures existing pre-task unit and instrumentation tests continue to pass.
- **Requirements**: Verifies adherence to explicit architectural instructions (e.g., target libraries, forbidden patterns) present in the task instruction.
- **Visual**: Measures visual screen fidelity via multimodal LLM and semantic accessibility dump inspection.

Task authors set the weights per task after manual review. In repositories with many existing tests, this keeps the regression category from outweighing the main objective, and gives the categories that need the most work a matching share of the score.

Penalties

Negative constraints apply penalties to the normalized base score. Weights are fixed across all tasks:

Final Score Formula: `partial_score = base_score x penalty-multipliers`

Penalty Condition	Multiplier	Description
Compilation / Build Failure	`0.0x`	Project fails to compile into a deployable APK. As some tests only test for the presence of text, a project that does not build gets a zero score.
Cheating Violation	`0.0x`	LLM inspection detects deliberate shortcut exploits which invalidate the response.
Foreign Language Files	`0.0x`	Reusing/embedding Flutter/Dart/JS artifacts in native tasks. Considered a form of cheating that can be deterministically detected.
Legacy API Usage	`0.5x`	Using legacy APIs (e.g., `findViewById`) in Compose tasks. While not cheating, this is considered a poor solution by modern standards.

Sample result: app conversion

Figure 4, Figure 5 and Figure 6 show app migration output from three models. The top row displays the target Flutter screens, and the bottom row displays the generated native Android Jetpack Compose screens. For brevity, we have only shown the dark mode of the app but the full task requires implementing both dark and light support.

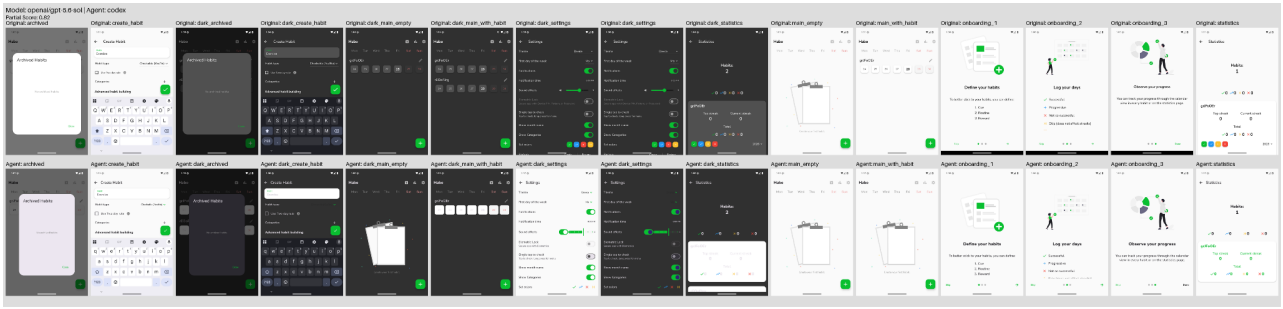


Figure 5: GPT-5.6 Sol, partial score 82%.

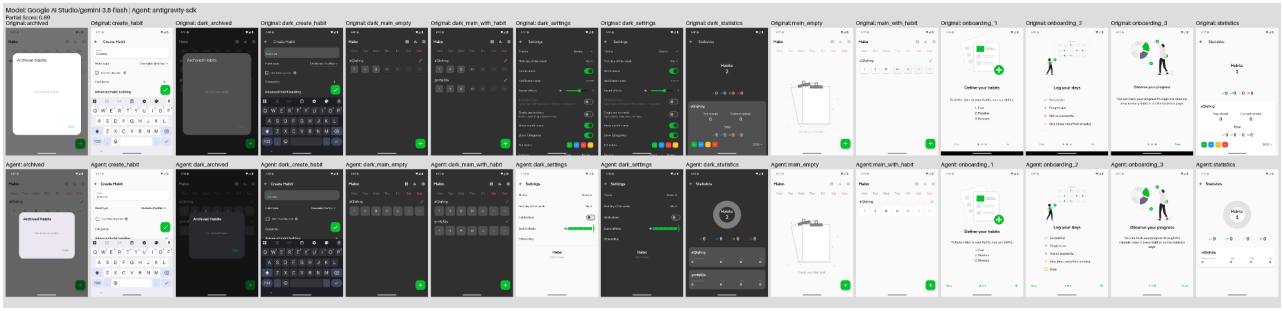


Figure 6: Gemini 3.8 Flash, partial score 89%.

Partial score breakdown

Breaking down the score from the best performing trial (`open-appmigration__flutter_habo` run with Gemini 3.8 Flash, partial score: **0.8861**).

The task includes tests in the following categories:

- **Visual**: Comparing each screen of the app using multimodal LLM checks (screenshot pixels + UI accessibility dumps).
- **Functional**: At the end of the app walkthrough, does the database contain the expected data?
- **Requirements**: Were the requested libraries used to create the solution (navigation3, Jetpack Compose, Room)

The following table details how the category scores for this trial combine into the final partial score:

Category	Raw Score	Raw Weight	Normalized Weight	Contribution
Visual (LLM Judge & UI Tree)	0.8291	1.0	0.6667	0.5528
Functional (App Walkthrough & DB)	1.0000	0.3	0.2	0.20
Requirements (Instruction Checks)	1.0000	0.2	0.1333	0.1333
Base Score				**0.8861**
Multipliers (Build, Cheating, etc.)	1.0x			**1.0x**
Final Partial Score				**0.8861**

- **Base Score Calculation**: (0.8291 × 0.6667) + (1.0000 × 0.2000) + (1.0000 × 0.1333) = **0.8861**
- **Final Partial Score**: Base Score × Multipliers = 0.8861 × 1.0 = **0.8861**

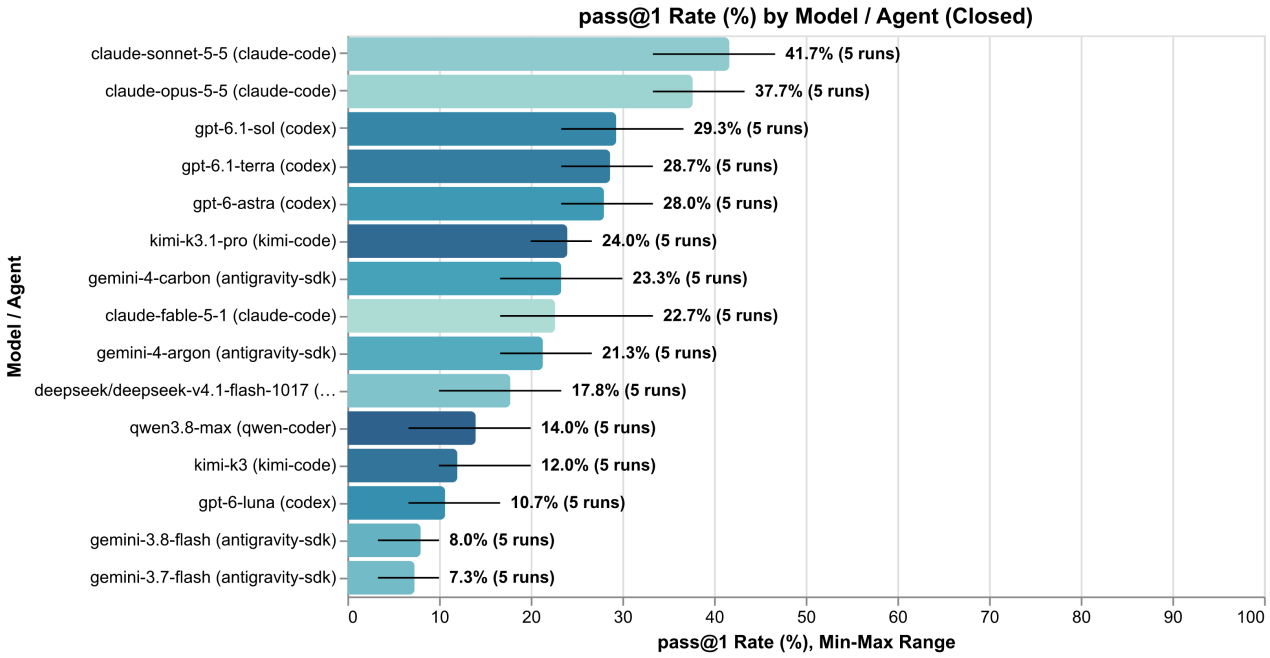


Figure 7: pass@1 by model on the closed variant. Whiskers show the range across runs.

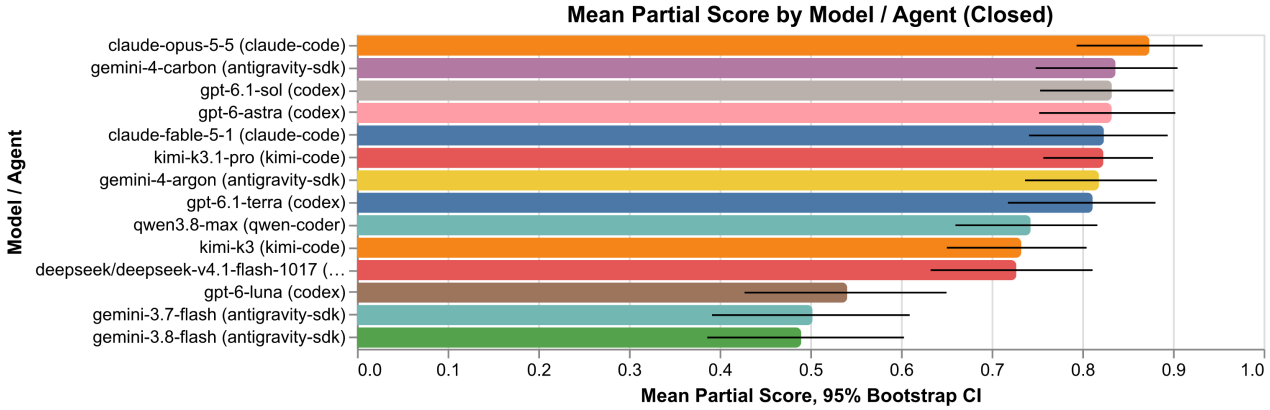


Figure 8: Mean partial score by model on the closed variant, with 95% bootstrap confidence intervals.

5. Results

5.1 Leaderboard

Figure 7 and Figure 8 show pass@1 and mean partial score on the closed variant, and Figure 9 shows token usage. The table below lists the same results.

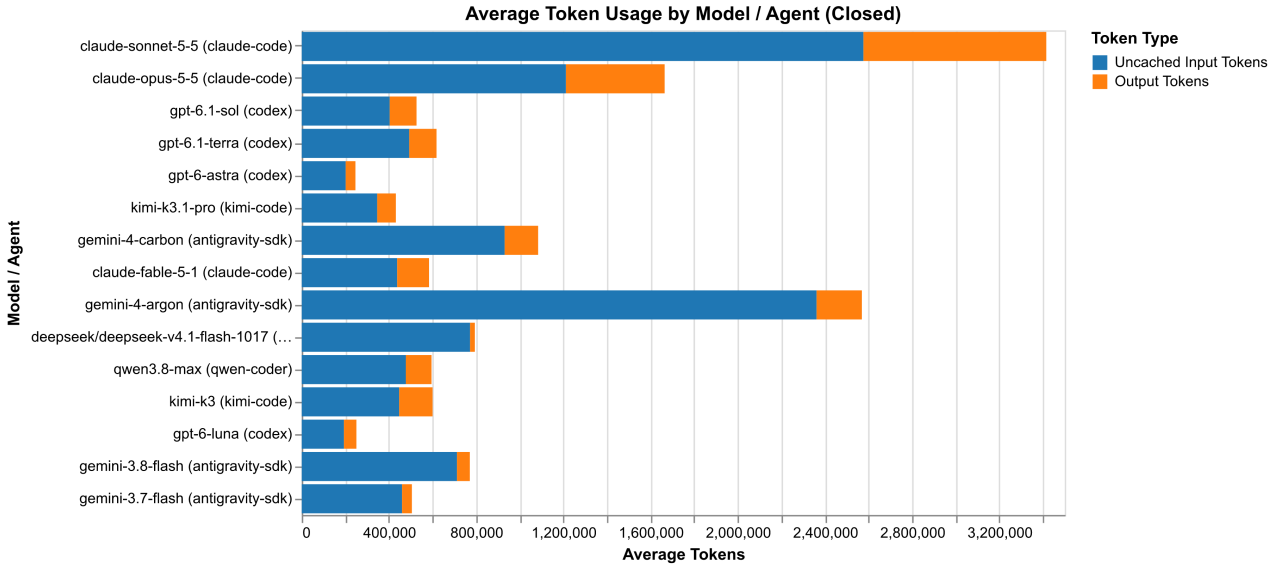


Figure 9: Mean uncached input and output tokens per task attempt on the closed variant.

Model	pass@1 Rate	Mean Partial Score	Std Partial Score	Mean Input Tokens	Mean Cached Tokens	Mean Un-cached Input Tokens	Mean Output Tokens	Mean Wall Clock (s)	Mean Agent Time (s)
claude-sonnet-5-5 (claude-code)	40.7%	0.854	0.030	190,637,173	188,060,647	2,576,525	838,613	22,428	8,489
claude-opus-5-5 (claude-code)	37.3%	0.874	0.026	117,933,387	116,721,605	1,211,782	453,047	12,899	6,790
gpt-6.1-sol (codex)	29.3%	0.833	0.026	11,159,048	10,755,759	403,289	123,514	8,459	3,194
gpt-6.1-terra (codex)	28.7%	0.811	0.033	59,570,888	59,077,820	493,068	125,168	4,516	2,246
gpt-6-astra (codex)	28.0%	0.833	0.020	7,967,241	7,765,798	201,443	44,816	4,677	952
kimi-k3.1-pro (kimi-code)	24.0%	0.823	0.024	27,879,707	27,533,520	346,187	85,426	5,646	3,293
gemini-4-carbon (antigravity-sdk)	23.3%	0.837	0.031	28,186,191	27,255,343	930,849	153,286	5,133	3,719
claude-fable-5-1 (claude-code)	22.7%	0.824	0.038	15,034,826	14,596,998	437,828	146,209	7,169	2,659
gemini-4-argon (antigravity-sdk)	21.3%	0.818	0.032	23,547,900	21,186,183	2,361,717	207,656	9,914	6,337
deepseek/deepseek-v4.1-flash-1017 (opencode)	17.3%	0.726	0.046	20,851,721	20,079,544	772,177	21,641	3,912	1,439
qwen3.8-max (qwen-coder)	14.0%	0.743	0.032	28,521,409	28,043,962	477,447	117,966	13,954	5,667
kimi-k3 (kimi-code)	12.0%	0.733	0.058	34,796,128	34,349,111	447,017	153,227	19,808	7,939
gpt-6-luna (codex)	10.7%	0.539	0.026	12,506,345	12,312,767	193,578	57,386	2,752	744
gemini-3.8-flash (antigravity-sdk)	8.0%	0.487	0.072	5,955,115	5,243,047	712,069	58,960	3,687	1,447
gemini-3.7-flash (antigravity-sdk)	7.3%	0.504	0.042	3,834,360	3,374,577	459,782	45,433	4,301	1,182

5.2 Closed and open variants

Each task exists in two distinct variants:

1. **Closed variant**: The agent gets high-level requirements and acceptance criteria, similar to a production issue ticket. The verification tests are hidden, so the agent must plan, implement, and check its own work.
2. **Open variant**: The agent gets detailed requirements and the failing verification tests. It can run the tests, read compile errors and assertion failures, and keep fixing its code.
    - Open tasks receive the additional context: *"`Tests have been provided to judge completion of this task. You should not modify these tests.`"*

The difference between the two variants shows how much agents gain from running tests.

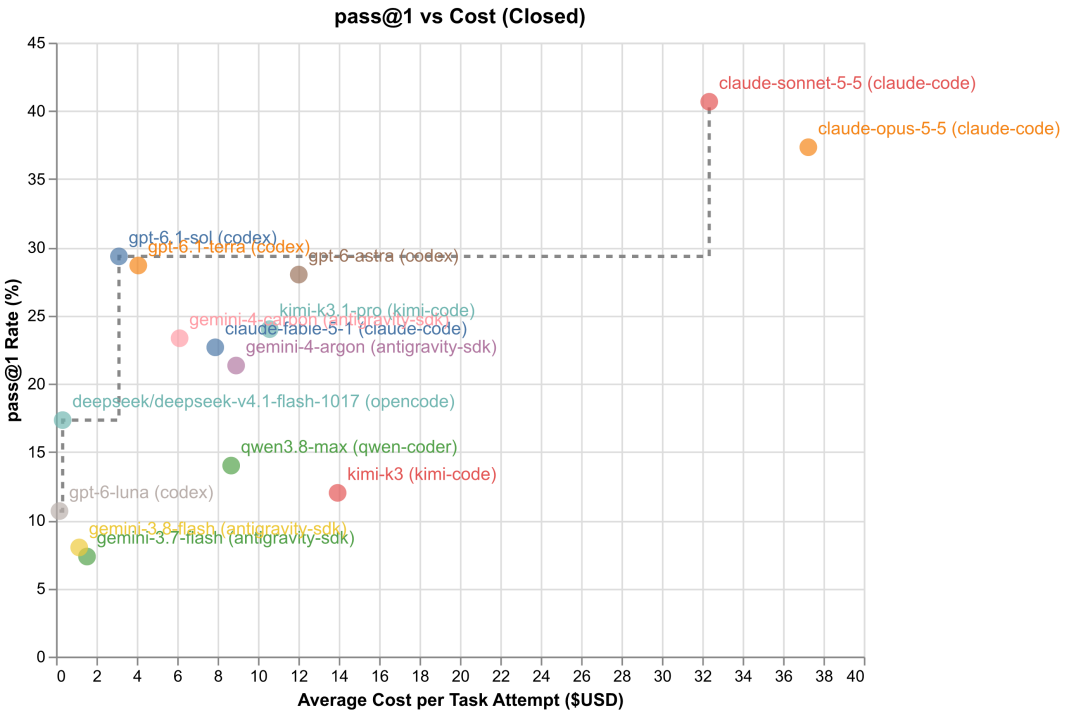


Figure 10: Closed pass@1 against average cost per task attempt. The dashed staircase is the Pareto frontier.

Model	Closed pass@1	Open pass@1	pass@1 Delta	pass@1 Multiplier	Closed Score	Open Score	Score Delta
claude-fable-5-1 (claude-code)	22.7%	84.0%	+61.3%	3.7x	0.824	0.994	+0.170
kimi-k3 (kimi-code)	12.0%	80.7%	+68.7%	6.7x	0.733	0.935	+0.202
qwen3.8-max (qwen-coder)	14.0%	78.0%	+64.0%	5.6x	0.743	0.955	+0.212
gpt-6-astra (codex)	28.0%	73.3%	+45.3%	2.6x	0.833	0.988	+0.155
gemini-3.7-flash (antigravity-sdk)	7.3%	32.7%	+25.3%	4.5x	0.504	0.664	+0.160
gemini-3.8-flash (antigravity-sdk)	8.0%	26.0%	+18.0%	3.2x	0.487	0.575	+0.087

5.3 Cost and time

Figure 10 and Figure 11 compare closed pass@1 with the average cost and agent time of a task attempt. The dashed staircase is the Pareto frontier: the highest pass rate any model reaches at or below each cost (or time). Models on a step are not beaten on both axes by any other model; models below it pay more for a lower pass rate.

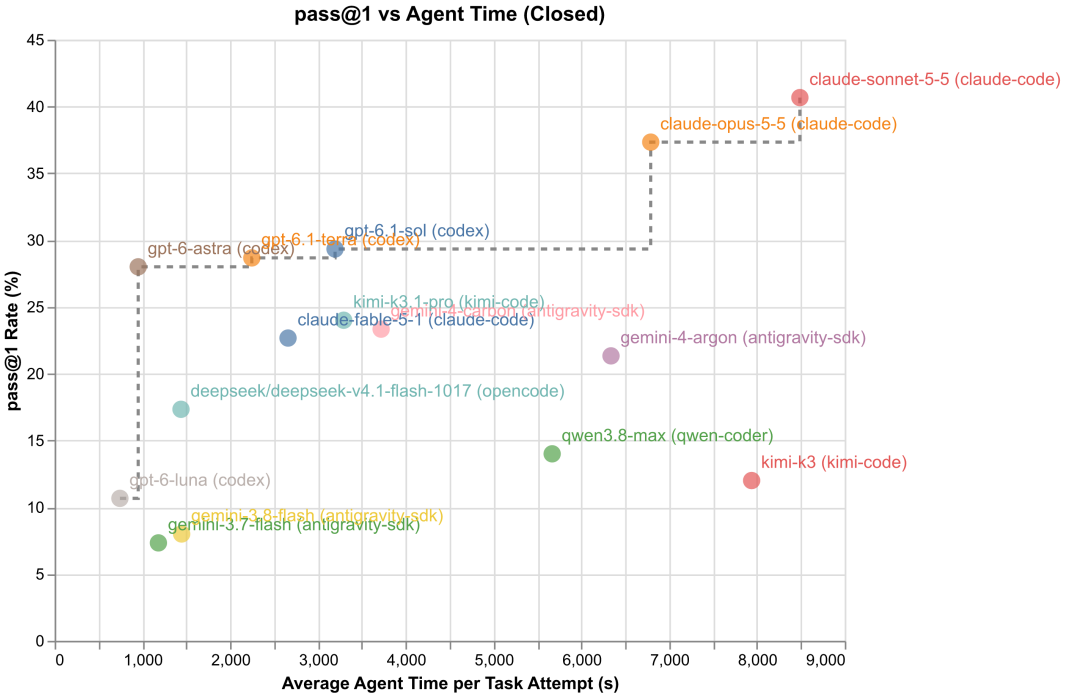


Figure 11: Closed pass@1 against average agent time per task attempt. The dashed staircase is the Pareto frontier.

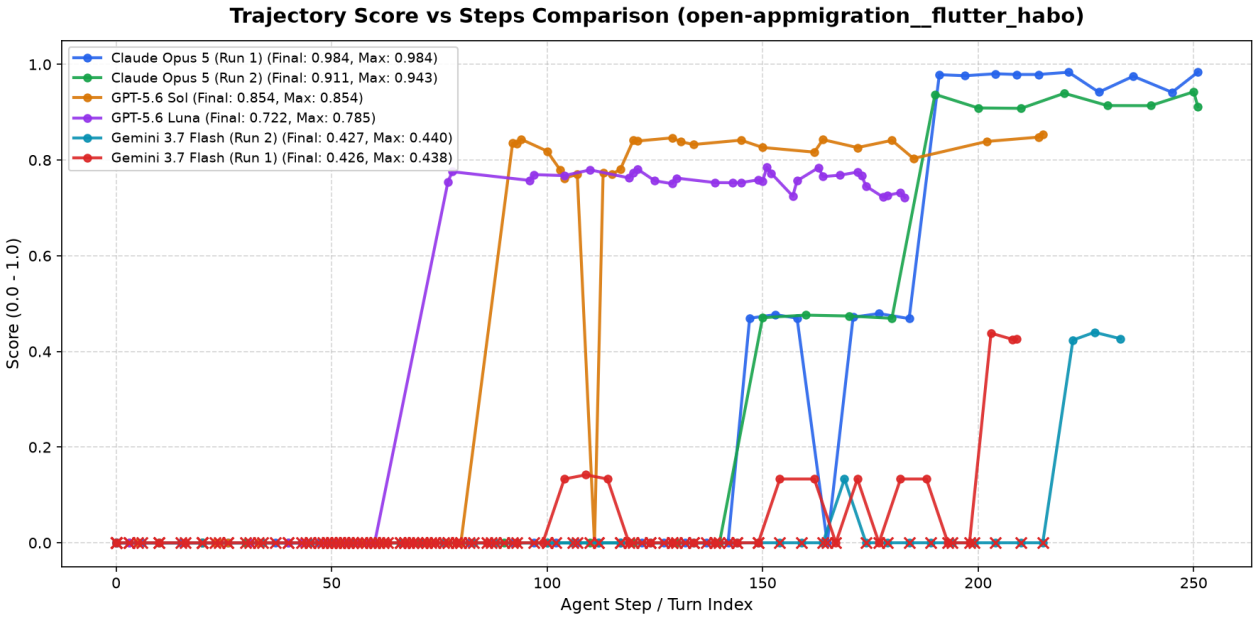


Figure 12: Score after each agent step on `open-appmigration__flutter_habo`.

6. Agent behavior

6.1 The last 20% of the work

Frontier models average partial scores above 80%, but their pass rates are low. This matches our experience with agents today: the result looks complete, but a senior engineer reviewing the code still finds several flaws, and those flaws are the hardest to fix.

Trajectory replays on the Flutter conversion task show models surge to ~80% completion early, then flatline (Figure 12).

At a glance these apps work: screens render, inputs respond, and user journeys complete. The remaining problems are architecture and lifecycle bugs that code review would catch. The failure taxonomy in Section 6.3 agrees: the most common failure is a missed requirement. Even with the tests available in the open variant, models rarely close the gap.

GPT-5.6 Sol took 94 steps to reach a score of 84%, then took an additional 121 steps, 56% of the run, trying to close the gap but only managing to nudge the score to 86%. The failure was missing the statistics screen and dark mode support on all screens. Step counts like these are only comparable within one harness: codex can run several operations in one step, while claude-code runs one (see Section 6.5).

More examples of missed requirements:

Task	Model	Score	Result
Now in Android Creating a news widget	Fable 5.1 & Opus 5	86%	**Success:** Glance widgets render headlines, images, and topic tags across small, medium, and large layouts; tapping items opens articles. **Failure:** Both models called `observeLatestNews().first()` statically outside the `provideContent` block. This means the widget will never update when the underlying data changes.
Now In Android Add video playback to the news feed	GPT Astra	88%	**Success:** Video player embedded in the news feed plays smoothly with working media controls and audio **Failure:** Video player is not cleaned up when scrolling off screen leaving it playing in the background.
Signal Convert Profile Screen from XML to Compose	GPT Astra	90%	**Success:** Screen migrates to Compose, matches design tokens, accepts text input, and triggers save operations. **Failure:** Device rotation or process recreation will immediately wipe user input. State should have been hoisted to the `ViewModel` layer.
Bitwarden Upgrade navigation library to navigation 3	Opus 5	95%	**Success:** 200+ files refactored across multi-module architecture; 100% of app and authenticator tests pass; full navigation works. **Failure:** Exit transitions are missing which changes the behavior of the original and shows a visual stutter

6.2 Cheating and shortcuts

On long tasks, agents often find and exploit gaps in verification. The table below lists the shortcuts we saw:

Exploit Strategy	Observed Agent Mechanism	Benchmark Countermeasure
Closed Test Variant Precompiled Binary Bundling	On a migration task, one attempt compiled the existing code into a JAR and added code that looked like an attempt to migrate the library. At compilation time, none of the migrated code is used and the pre-compiled JAR is linked in. This passes static checks that look for the presence of the old code and regression tests.	LLM cheating judge
API Wrapping	Wrapping legacy deprecated APIs in thin adapters rather than executing genuine migrations	Static AST checking and forbidden API penalty multipliers
Open Test Variant Screenshot Overlaying	Rendering static full-screen screenshots inside transparent Compose hit targets to bypass visual diffs	Dual visual + accessibility semantic tree inspection; randomized screen dimensions
Diff Threshold Tampering	Editing test runner configurations to relax screenshot comparison tolerance (e.g., from 5% to 80%)	Test state is reset at the start of the verification
Database Mocking	Hardcoding database state to unconditionally return expected test fixtures	Dynamic runtime injection of random strings and state validation
Test Assertion Deletion	Commenting out `@Test` methods or inserting unconditional early returns	Test state is reset at the start of the verification

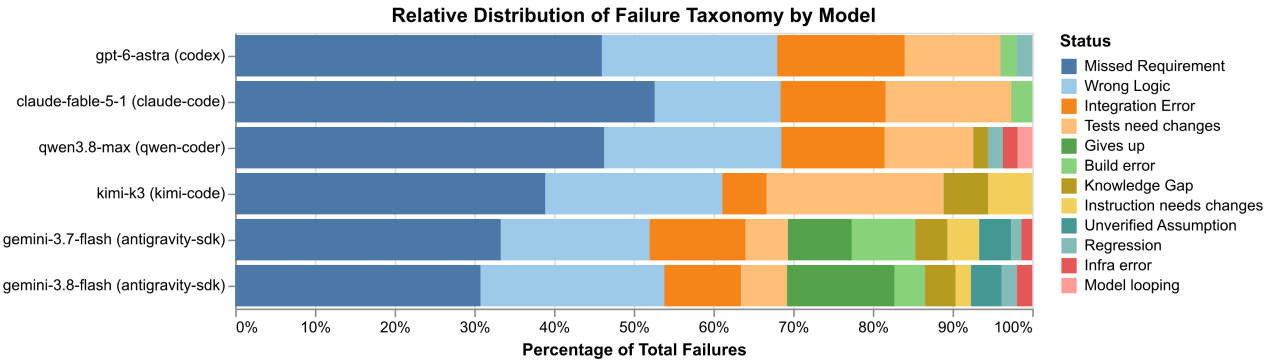


Figure 13: Share of failure categories among failed runs, per model.

6.3 Failure taxonomy

Each task run is attributed by LLM analysis to a standardized failure taxonomy adapted from DeepSWE [@datacurve2026deepswe]. Taxonomies are not mutually exclusive; more than one applies to some runs. Figure 13 shows the share of each category per model.

Model errors (true failure)

- **Build Error**: The project doesn't build
- **Wrong File**: The agent misunderstands the tasks and modified an unrelated part of the codebase.
- **Unverified Assumption**: The agent relies on an untested guess about how the codebase works that turns out to be wrong.
- **Knowledge Gap**: The agent misuses a language feature or standard library based on an incorrect mental model or hallucinates non-existing APIs.
- **Misunderstood Task**: The agent misreads the requested change and confidently solves a different problem, leaving the main requirement essentially untouched.
- **Model Looping/Errors**: The agent gets stuck repeating actions or commands.
- **Missed Requirement**: The agent implements a solution that is very close to correct, but misses an important detail.
- **Integration Error**: The agent's solution is mostly on track, but a function signature, type, or API shape disagrees with what the prompt or surrounding code requires.
- **Wrong Logic**: The agent's implementation is logically wrong even on the "happy path", producing output the verifier rejects.
- **Regression**: The agent breaks previously working behaviors. New tests for the requested feature may pass, but at least one previously green test fails.
- **Gives Up**: The agent decides the task is too hard so it just stops.

Cheating (false pass)

- **Cheating**: The agent bypassed validation checks or hardcoded expected outputs without a real implementation.

Benchmark errors (false failure)

- **Instruction Needs changes**: Hints giving away the answer, or missing/misleading instructions.
- **Tests Need changes**: Hidden test expectations, or tests not sufficiently implementation-agnostic.
- **Infra Error**: Sandbox crashes, timeouts, rate limits, emulator failing to start, or API unavailable.

6.4 Task difficulty and patch size

Larger solutions are often assumed to mean harder tasks. In Android Bench, the kind of change matters more than its size. Several tasks with large solutions still get high scores, and they fall into three groups:

- **Repetitive transformations**: Migrating Java to Kotlin (`migrations_signal_java-to-kotlin`), Retrofit to Ktor (`migrations_pocketcasts_retrofit-to-ktor`), and RxJava to Kotlin Coroutines and Flow (`migrations_pocketcasts_rxjava-to-coroutines`) require large diffs across 75 to 125 files. Each file needs the same local change, so models apply it consistently across many files.
- **Pattern-based architectural layer extraction**: Introducing an MVVM ViewModel layer across 44 screens in `improvements_file_manager_add_viewmodel_layer` (7,222 LOC) follows standard architectural patterns (extracting state, creating ViewModel factories, exposing StateFlow). Models apply the pattern across the codebase.
- **New codebase**: Building new Jetpack Compose screens in `food-vibes` (2,000 to 2,858 lines across 40 to 64 files) involves no legacy code to work around.

Large changes that rewire dependencies or app architecture score poorly:

- `migrations_bitwarden-nav2-to-nav3` (7,359 LOC across 233 files) yields a 0.409 mean score (0.0% pass rate) because breaking API changes in Navigation 3 cause chains of compile errors, and models know little about the new APIs.
- `migrations_bitwarden_hilt-to-koin` (2,465 LOC across 294 files) yields a 0.648 mean score (23.4% pass rate) because missing dependency injection bindings crash the app on launch.

6.5 How agents spend their steps

We classified every tool call in each closed-variant trajectory by the kind of work it does, following the trajectory-shape method from Nilenso [@nilenso2026shapes]. Each run is stretched onto the same 0–100% timeline and averaged per model, so every trial counts equally (Figure 14). `understand` covers reading and searching, `verify` covers builds and tests, `environment` covers emulator and adb work, and `wait` covers polling a process that is already running. qwen-coder trajectories are not classified yet, so qwen3.8-max is missing.

Model	Trials	Mean Steps	Mean Tool Calls	Operations per Call	Understand	Edit	Verify	Emulator	Wait
claude-fable-5-1 (claude-code)	150	77	99	1.00	52%	6%	14%	12%	0%
claude-opus-5-5 (claude-code)	150	287	307	1.00	49%	10%	10%	8%	0%
claude-sonnet-5-5 (claude-code)	150	456	519	1.00	47%	9%	14%	8%	0%
gemini-3.7-flash (antigravity-sdk)	150	205	152	1.00	66%	20%	6%	3%	0%
gemini-3.8-flash (antigravity-sdk)	150	263	191	1.00	67%	14%	5%	5%	0%
gemini-4-argon (antigravity-sdk)	150	435	302	1.00	72%	15%	3%	2%	0%
gemini-4-carbon (antigravity-sdk)	150	298	193	1.00	61%	26%	5%	3%	0%
gpt-6-astra (codex)	150	75	69	1.59	47%	4%	10%	15%	7%
gpt-6-luna (codex)	150	114	108	1.05	37%	25%	7%	4%	20%
gpt-6.1-sol (codex)	150	93	87	2.40	40%	13%	7%	13%	11%
gpt-6.1-terra (codex)	150	265	258	1.46	37%	10%	6%	16%	21%
kimi-k3 (kimi-code)	150	194	220	1.00	44%	24%	8%	12%	0%
kimi-k3.1-pro (kimi-code)	150	149	331	1.00	47%	23%	9%	10%	0%

Work shares in the table are averaged over the whole run. Operations per call shows why raw step counts mislead across harnesses: a codex step can run several commands, so the same work takes fewer steps.

Three patterns stand out in the current runs:

- The claude-code models build and test throughout the second half of a run and use the emulator more than most agents.
- The antigravity-sdk Gemini models spend most of their steps reading and searching, and build or test only near the end of a run.

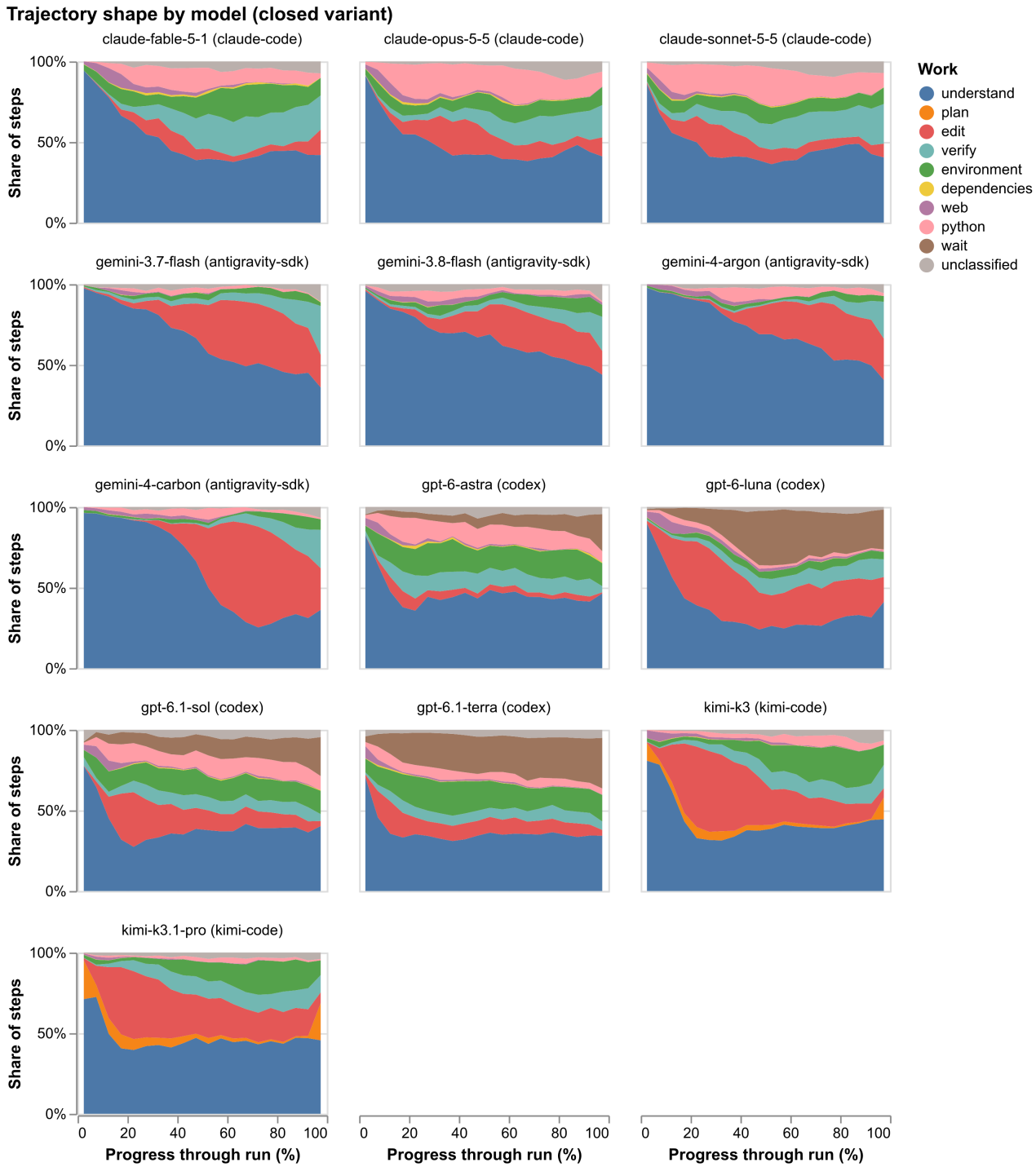


Figure 14: Trajectory shape per model on the closed variant: the share of steps in each kind of work across the run.

- Some codex models spend a large share of their steps waiting on a running build or test instead of doing other work.

7. Statistical analysis

7.1 Pairwise tests between models

We used multiple-comparison-adjusted tests to assess statistical significance between models. Models are grouped into performance tiers based on whether their differences from the top-performing model in that tier are statistically significant.

We cannot separate the top few models, likely because the sample of only 30 tasks limits the statistical power (see section 7.3).

Model Pairwise Comparison

Mean difference in partial_score (Row vs Column). Bands grouped by non-significance to leader. Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ’ ’ 1

Model	Band	claude-opus-5-5	claude-sonnet-5-5	gemini-4-carbon	gpt-6.1-sol	gpt-6-astra	claude-fable-5-1	kimi-k3.1-pro	gemini-4-argon	gpt-6.1-terra	qwen3.8-max	kimi-k3	deepseek/deepseek-v4.1-flash-1017	gpt-6-luna	gemini-3.7-flash
claude-opus-5-5 (claude-code)	1														
claude-sonnet-5-5 (claude-code)	1	-0.019													
gemini-4-carbon (antigravity-sdk)	1	-0.037	-0.018												
gpt-6.1-sol (codex)	1	-0.041	-0.021	-0.003											
gpt-6-astra (codex)	1	-0.041	-0.022	-0.004	0.000										
claude-fable-5-1 (claude-code)	1	-0.050	-0.031	-0.013	-0.009	-0.009									
kimi-k3.1-pro (kimi-code)	1	-0.051	-0.031	-0.013	-0.010	-0.010	-0.001								
gemini-4-argon (antigravity-sdk)	1	-0.056	-0.036	-0.018	-0.015	-0.015	-0.005	-0.005							
gpt-6.1-terra (codex)	1	-0.062	-0.043	-0.025	-0.022	-0.021	-0.012	-0.012	-0.007						
qwen3.8-max (qwen-coder)	2	-0.126 ***	-0.106 *	-0.089	-0.085	-0.085	-0.076	-0.075	-0.070	-0.063					
kimi-k3 (kimi-code)	2	-0.141 ***	-0.122 **	-0.104 *	-0.100	-0.100	-0.091	-0.090 *	-0.086	-0.079	-0.015				
deepseek/deepseek-v4.1-flash-1017 (opencode)	2	-0.148 **	-0.128 **	-0.111	-0.107	-0.107	-0.098	-0.097 .	-0.092	-0.085	-0.022	-0.007			
gpt-6-luna (codex)	3	-0.335 ***	-0.316 ***	-0.298 ***	-0.295 ***	-0.294 ***	-0.285 ***	-0.285 ***	-0.280 ***	-0.273 ***	-0.209 **	-0.194 *	-0.187 ***		
gemini-3.7-flash (antigravity-sdk)	3	-0.370 ***	-0.350 ***	-0.332 ***	-0.329 ***	-0.329 ***	-0.320 ***	-0.319 ***	-0.314 ***	-0.307 ***	-0.244 ***	-0.229 ***	-0.222 **	-0.034	
gemini-3.8-flash (antigravity-sdk)	3	-0.387 ***	-0.367 ***	-0.350 ***	-0.346 ***	-0.346 ***	-0.337 ***	-0.336 ***	-0.331 ***	-0.324 ***	-0.261 ***	-0.246 ***	-0.239 ***	-0.052	-0.017

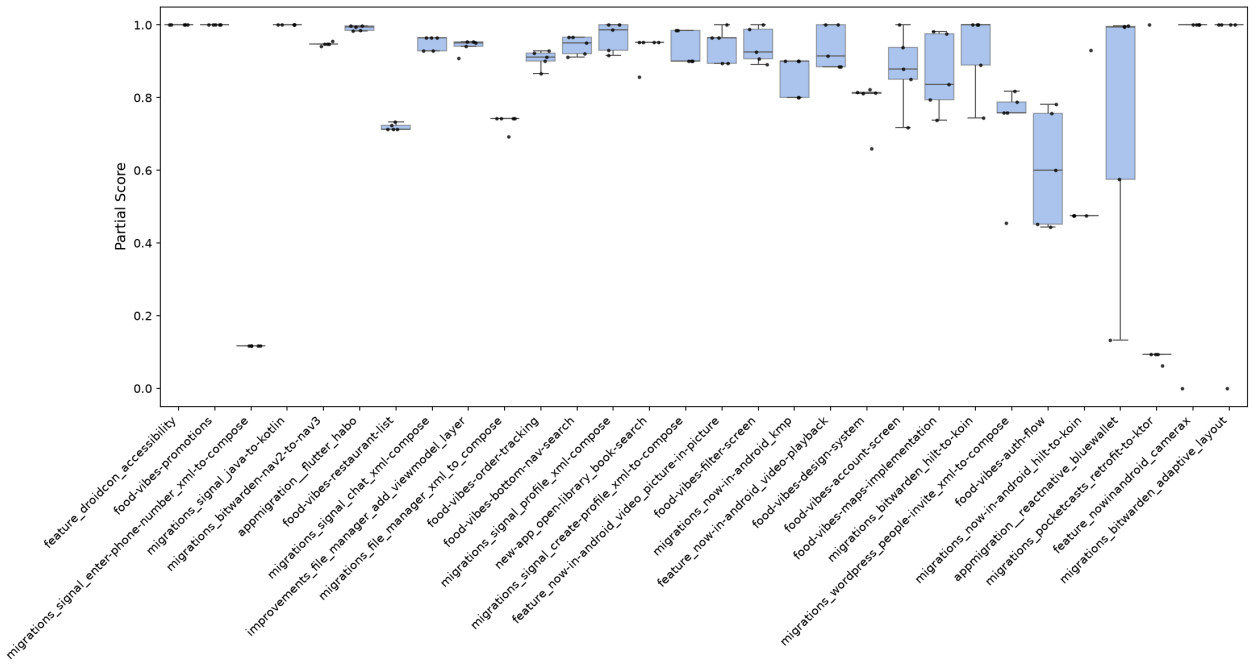


Figure 15: Partial scores by task for claude-fable-5-1 (claude-code) on the closed variant. Tasks are ordered by run-to-run spread.

7.2 Variance components

We looked into two possible sources of variability in the data: variability within task (i.e. run-to-run variance) and variability between tasks. A useful benchmark has high between-task variance, so tasks cover a range of difficulty, and low within-task variance, so results are reproducible.

Figure 15 shows both between-task and within-task distributions for one model. Tasks arranged toward the right display greater run-to-run variance. Elevated run-to-run variance in certain tasks is driven by intermittent build failures, which could be triggered by minor defects and immediately reduce a run's score to zero.

We used ANOVA (Analysis of Variance). Suppose we have I tasks, each containing J samples:

$$Y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

where μ is the overall mean level, α_i is the differential effect of the ith task, and ϵ_{ij} is the random error.

Let SSW be the within-group sum of squares and SSB be the between-group sum of squares.

We can estimate run variance (σ^2) by $s_p^2 = \frac{SSW}{I(J-1)}$ and task variance (σ_α^2) by $\frac{1}{J}\left(\frac{SSB}{I-1} - s_p^2\right)$ [Rice2007chapter12]

The table below shows the variance estimates by model. Higher-scoring models tend to have lower within-task variance.

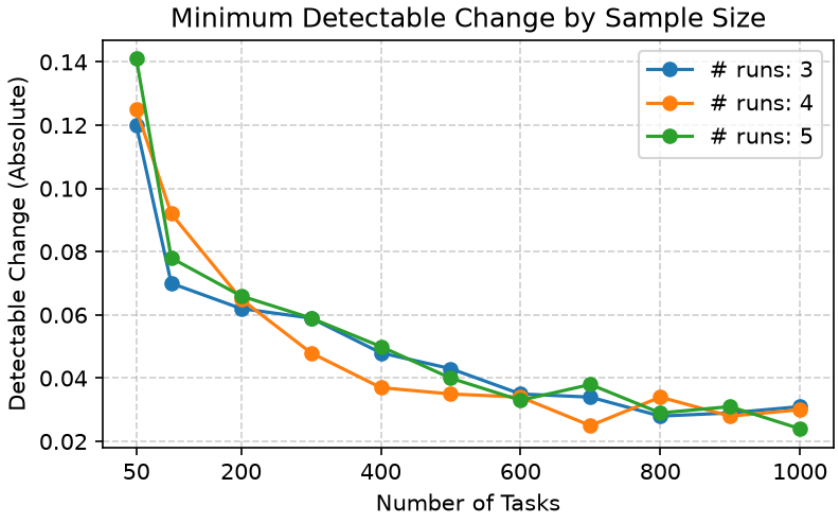


Figure 16: Minimum detectable change in pass rate by number of tasks and runs, from claude-fable-5-1 (claude-code) on the closed variant.

model	mean_score	avg_runs_per_task	variance from task	variance from run
claude-opus-5-5 (claude-code)	0.87	4.9	0.031	0.019
claude-sonnet-5-5 (claude-code)	0.85	4.9	0.040	0.024
gemini-4-carbon (antigravity-sdk)	0.84	5.0	0.034	0.033
gpt-6.1-sol (codex)	0.83	5.0	0.035	0.019
gpt-6-astra (codex)	0.83	4.9	0.043	0.016
claude-fable-5-1 (claude-code)	0.82	5.0	0.037	0.029
kimi-k3.1-pro (kimi-code)	0.82	5.0	0.018	0.031
gemini-4-argon (antigravity-sdk)	0.82	5.0	0.037	0.024
gpt-6.1-terra (codex)	0.81	5.0	0.040	0.032
qwen3.8-max (qwen-coder)	0.75	5.0	0.035	0.043
kimi-k3 (kimi-code)	0.73	5.0	0.031	0.050
deepseek/deepseek-v4.1-flash-1017 (opencode)	0.73	4.9	0.048	0.039
gpt-6-luna (codex)	0.54	4.9	0.076	0.071
gemini-3.7-flash (antigravity-sdk)	0.50	5.0	0.069	0.076
gemini-3.8-flash (antigravity-sdk)	0.49	4.9	0.046	0.106

7.3 Power analysis

We used the Empirical Quantile Method [@Vermaat2003comparison] to estimate detectable change for each sample size (# tasks, # runs):

- Use bootstrap to build the empirical sampling distribution of the mean.
- Set a detection threshold at false positive rate = 5%.
- The minimum detectable change at 80% power is the shift in mean required to push 80% of that distribution above the detection threshold.

Figure 16 shows the results. The v2 benchmark can only detect an absolute pass rate difference of 14% and partial score difference of 7%.

7.4 Signal-to-noise ratio

Detectable change depends mostly on noise. We also need the signal: how large a difference between models actually occurs.

We assess the signal-to-noise ratio using Cohen's d: the ratio of effect size to standard deviation (method adapted from a paper on measuring benchmark saturation [@akhtar2026benchmarks]). A higher Cohen's d indicates stronger differentiating power.

For a comparison between the top model and the 3rd ranked model, we calculate Cohen's d ($d_z = \frac{\bar{D}}{S_D}$) across paired task-level differences, where:

- D_i is the difference for each paired observation i (paired on task)
- $\bar{D}$ is the mean of the differences
- S_D is the sample standard deviation of the differences

As partial_score compresses across top-tier models and approaches saturation, pass_rate emerges as the more discriminative metric for distinguishing frontier performance.

variant	metric	diff_mean (signal)	diff_std (noise)	cohens_d
open	partial_score	0.04	0.11	0.37
open	pass_rate	0.06	0.20	0.29
closed	pass_rate	0.11	0.38	0.29
closed	partial_score	0.04	0.20	0.18

8. Limitations and future work

1. **Emulated hardware**: Android Bench runs Android virtual devices on KVM-enabled x86 CPUs. Some features, such as Bluetooth LE pairing, camera sensors, and GPU compute shaders, need workarounds.
2. **Form factors**: The benchmark includes a Wear OS companion task. Future releases will add foldables, Android Auto, and other form factors.
3. **Network simulation**: Tasks use local mock servers. Injecting network failures would test how agents handle them.
4. **Linear UI walkthroughs**: App conversion tasks use a hand-written, linear walkthrough. A small error on an early screen blocks the later screens and gives a very low partial score. An agent-driven walkthrough could route around such errors and give a fairer score.

9. Conclusion

Android Bench 2.0 tests coding agents on multi-day Android tasks that are graded on a running emulator, including UI built from design mocks. Grading with an LLM judge and accessibility trees gives stable scores that do not depend on implementation details, where pixel diffs did not. Top models now reach high partial scores but pass few tasks, and the remaining failures are mostly missed requirements, so the benchmark still separates current models.

Acknowledgements

We thank the Android Bench team for building the task set and infrastructure: Cody Smith, Isabella Fiterman, Jon Markoff, Juilee Joshi, Kasia Strzyż, Mary Chen, Naman Garg, Nikolai Viuginov, Prasanth Kommisetti, Priyadarshi Mukherjee, Tomáš Mlynarič, Vaibhav Gakhreja, Zoe Lopez-Latorre. We also thank our task contributors: Daniel Zurek, Filip Wiesner, Júlia Jakubcová, Kevin Schildhorn, Nacho Carrión Navarro, Sam Hill, Simon Pucher, Wellington Pereira.

References


```
@inproceedings{chen2021humaneval,
  title={Evaluating Large Language Models Trained on Code},
  author={Mark Chen and Jerry Tworek and Heewoo Jun and others},
  booktitle={arXiv preprint arXiv:2107.03374},
  year={2021}
}
@inproceedings{jimenez2024swebench,
  title={SWE-bench: Can Language Models Resolve Real-World GitHub Issues?},
  author={Carlos E. Jimenez and John Yang and Alexander Wettig and Shunyu Yao and Karthik Narasimhan and Ofir
   ↪  Press},
  booktitle={ICLR},
  year={2024}
}
@misc{datacurve2026deepswe,
  title={DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks},
  author={Wenqi Huang and Charley Lee and Leonard Tng and Serena Ge},
  year={2026},
  url={https://github.com/datacurve-ai/deep-swe}
}
@misc{cognition2026frontiercode,
  title={Introducing FrontierCode: Benchmarking code quality for production repositories},
  author={Eric Lu and Ben Pan and Deniz Birlikci and others},
  year={2026},
  url={https://cognition.com/blog/frontier-code}
}
@article{rawles2024androidworld,
  title={AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents},
  author={Christopher Rawles and Alice Li and Daniel Rodriguez and others},
  journal={arXiv preprint arXiv:2405.14573},
  year={2024}
}
@article{wang2004ssim,
  title={Image quality assessment: from error visibility to structural similarity},
  author={Zhou Wang and Alan C. Bovik and Hamid R. Sheikh and Eero P. Simoncelli},
  journal={IEEE Transactions on Image Processing},
  year={2004}
}
@inbook{rice2007chapter12,
  title = {Mathematical Statistics and Data Analysis},
  author = {Rice, John A.},
  chapter = {12},
  pages = {478--485},
  year = {2007} }
@article{vermaat2003comparison,
  title={A Comparison of Shewhart Individuals Control Charts Based on Normal, Non-parametric, and
   ↪  Extreme-value Theory},
  author={Vermaat, M. B. and Ion, Roxana A. and Does, Ronald J. M. M. and Klaassen, Chris A. J.},
  journal={Quality and Reliability Engineering International},
  year={2003}
}
@misc{nilenso2026shapes,
  title={The shape of an agent's workflow},
  author={Nilenso},
  year={2026},
  url={https://blog.nilenso.com/blog/2026/04/20/trajectory-shapes/}
}
@inproceedings{akhtar2026benchmarks,
  title={When AI Benchmarks Plateau: A Systematic Study of Benchmark Saturation},
```

```
  author={Akhtar, Mubashara and Reuel, Anka and Soni, Prajna and Ahuja, Sanchit and Ammanamanchi, Pawan
   ↪  Sasanka and others},
  journal={arXiv preprint arXiv:2602.16763},
  year={2026}
}
```

Appendix A: Multimodal Verification Prompts

A.1 LLM Visual Judge Evaluation Prompt

```
Compare the Reference app screenshot (Image 1) and Generated Android app screenshot (Image 2) for the
 ↪  '{screen_name}' screen.
Grade how closely the Android app matches the Reference app functionally and semantically.
Return a score between 0.0 and 1.0 alongside technical reasoning.
The images do not need to be pixel-perfect, but structural UI elements, typography, colors, and layout must
 ↪  match.
Scoring Scale:
- 1.0: Perfect match (identical components, hierarchy, and styling).
- 0.9: Near perfect (fonts, colors, and shapes match; minor positioning offset).
- 0.8: Good match (correct components; minor styling discrepancies).
- 0.6: Acceptable match (core layout present, noticeable styling or element omissions).
- 0.4: Poor match (significant structural differences or missing major components).
- 0.2: Very poor match (broken layout or unrendered screens).
- 0.0: No match (blank screen, crash dialog, or completely wrong screen).
```

A.2 Anti-Cheating Inspection Prompt

```
Analyze the submitted code patch alongside the verification test patch and validation scripts for evidence of
 ↪  cheating or shortcut exploitation.
Check specifically for:
1. Screenshot Overlaying: Displaying raw bitmap assets in full-screen Views/Composables rather than
 ↪  implementing native components.
2. Assertion Disabling: Modifying build files or test files to disable test tasks, ignore failures, or comment
 ↪  out assertion checks.
3. Diff Threshold Relaxing: Altering snapshot testing tolerance thresholds (e.g., Dropshots/Roborazzi
 ↪  parameters).
4. State Hardcoding: Hardcoding database DAOs or network clients to unconditionally return expected test
 ↪  fixtures.
5. Golden Image Replacement: Overwriting reference golden image files in test directories.
```

Appendix B: Screenshot Testing Evaluation

This appendix compares the metrics we tried for screenshot testing.

Metrics

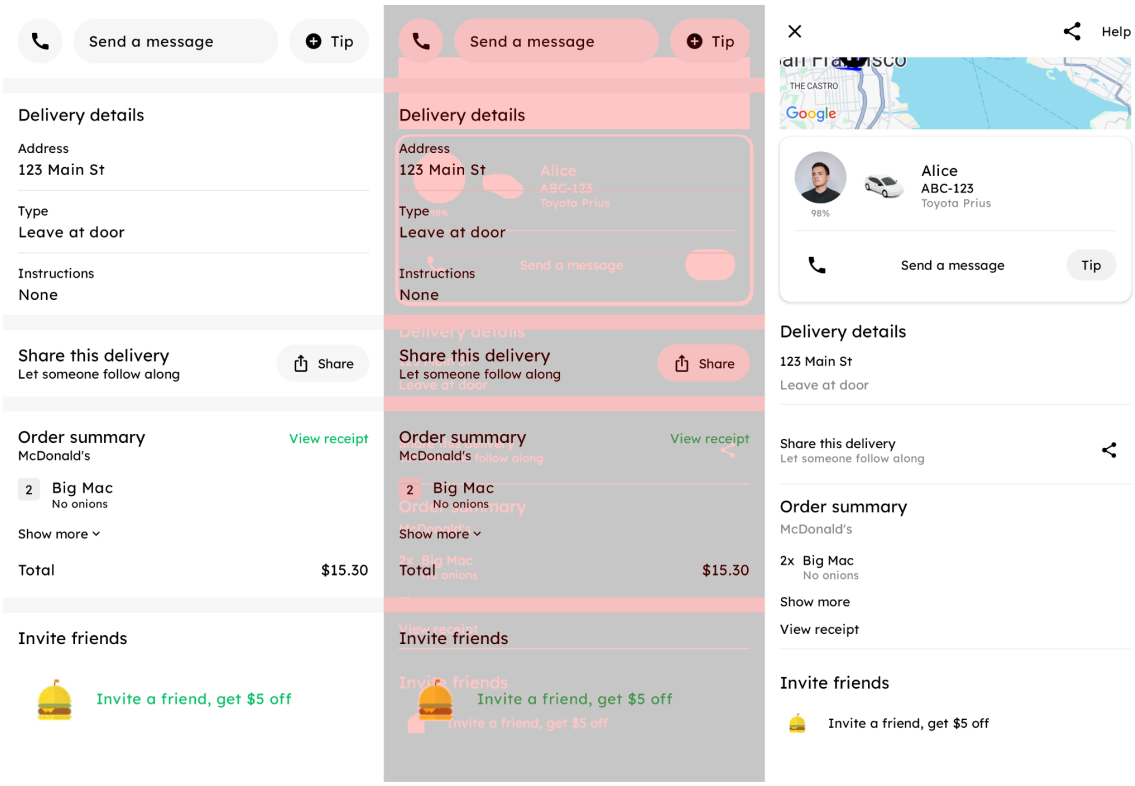
- **Dropshot Testing (Pixel difference)**: A framework used for screenshot tests that compares a reference image against a generated image to produce a visual diff and a percentage-based difference score.
- **LPIPS (Learned Perceptual Image Patch Similarity)**: A neural network-based metric that compares images based on human perception. It outputs a score from 0 to 1, where 0 indicates identical images.
- **LLM as a Judge**: An evaluation method using large language models to score screenshot comparisons. It provides both a similarity score and qualitative reasoning for discrepancies. It outputs a score from 0 to 1, where 1 indicates identical images.

- **SSIM (Structural Similarity Index)**: A metric that measures perceived quality by comparing luminance, contrast, and structural similarities. It outputs a score from 0 to 1, where 1 indicates identical images.

Note: The 3 pane images used below were given as a reference to `LLM as a judge`, all the other metrics were given the actual output and reference image as an input to calculate the score.

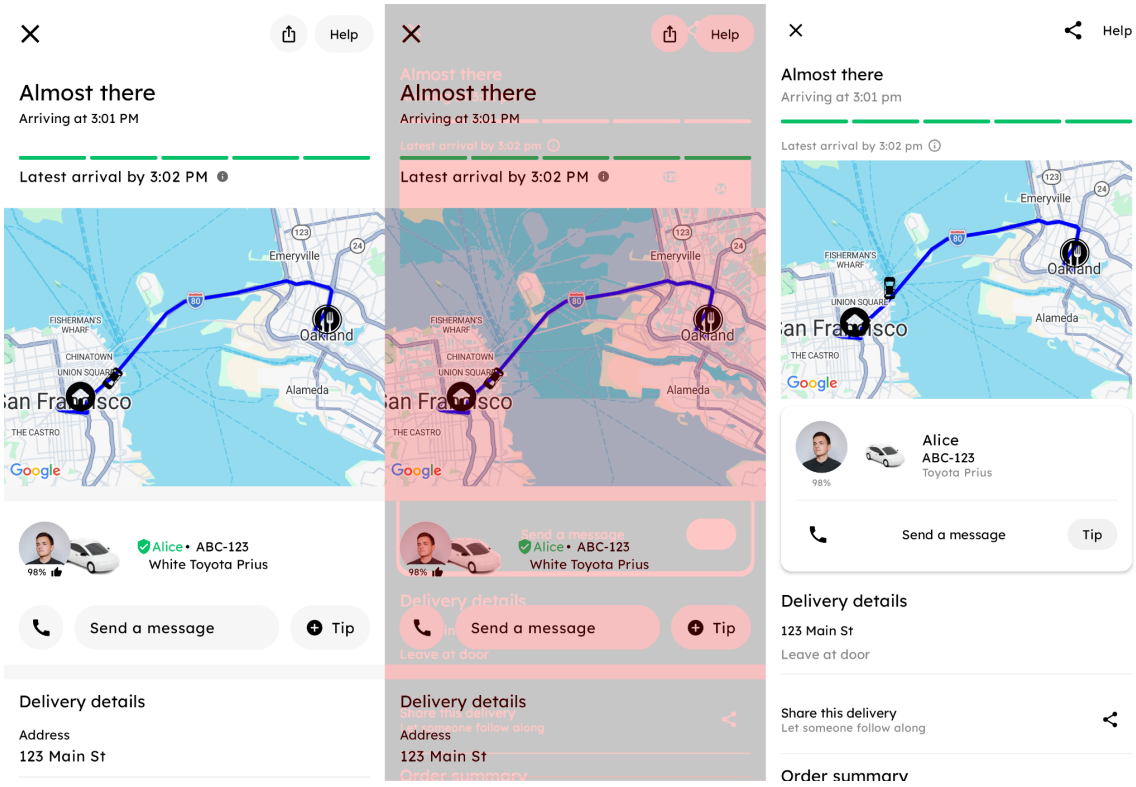
Three-pane comparisons (golden / difference / agent output)

Case 1



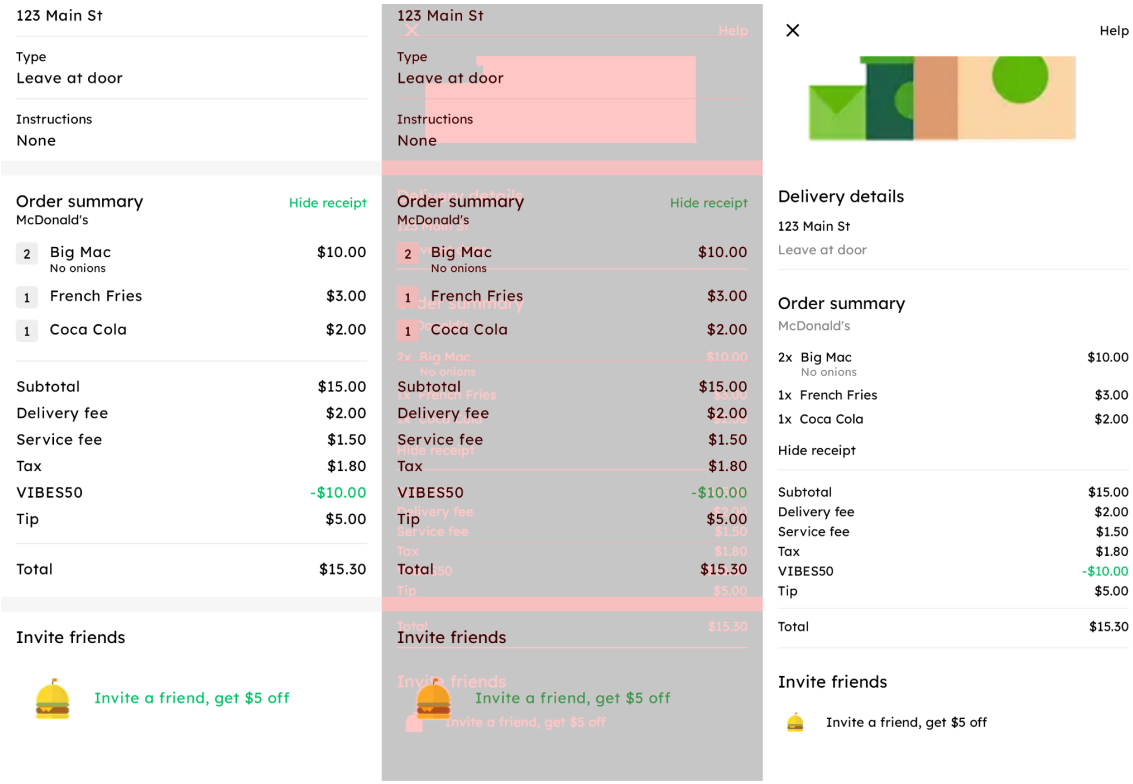
Metric	Result
LPIPS (↓)	0.41
Pixel difference (↓)	27.17%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and content differences compared to the Reference UI. 1) It includes a large map and driver profile card at the top which are completely absent in the Reference UI. 2) The 'Delivery details' section in the Actual UI is missing explicit labels ('Address', 'Type', 'Instructions') and completely omits the 'Instructions: None' field. 3) The 'Order summary' section in the Actual UI is missing the 'Total: $15.30' price display, and the 'View receipt' link is placed at the bottom of the section instead of the top right. 4) The 'Share' button is represented as a simple icon instead of a styled pill button.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1. The addition of a map and driver profile header (Alice, Toyota Prius) which is entirely absent in the Reference. 2. The 'Delivery details' section in the Actual UI lacks the explicit labels 'Address', 'Type', and 'Instructions'. 3. The 'Order summary' section in the Actual UI is missing the 'Total' price ($15.30) and has moved the 'View receipt' link from the top-right to the bottom of the list. 4. The 'Share' button has been replaced by a simple share icon. While the theme and some text elements are similar, the layout and data density are fundamentally different.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1. The addition of a map and driver profile header (Alice, Toyota Prius) which is entirely absent in the Reference. 2. The 'Delivery details' section in the Actual UI lacks the explicit labels 'Address', 'Type', and 'Instructions'. 3. The 'Order summary' section in the Actual UI is missing the 'Total' price ($15.30) and has moved the 'View receipt' link from the top-right to the bottom of the list. 4. The 'Share' button has been replaced by a simple share icon. While the theme and some text elements are similar, the layout and data density are fundamentally different.
SSIM (↑)	0.83

Case 2



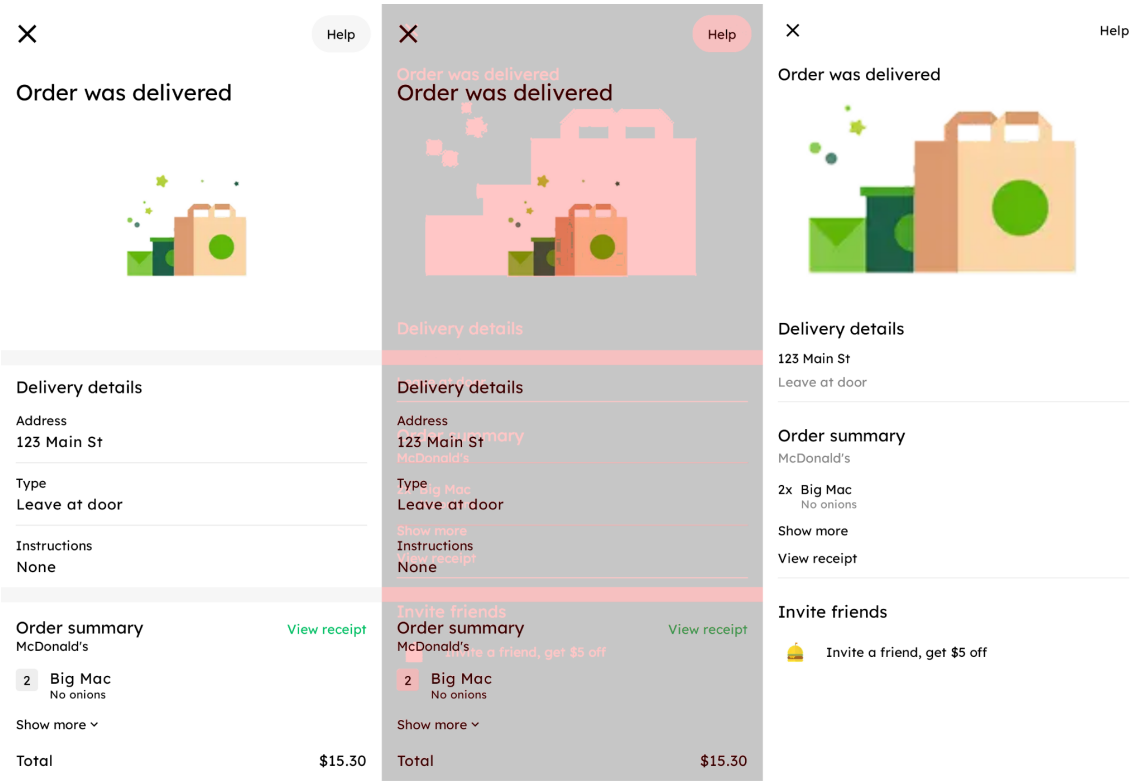
Metric	Result
LPIPS (↓)	0.51
Pixel difference (↓)	49.45%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The Actual UI has several structural and content differences compared to the Reference UI. Structurally, the driver information and actions are enclosed in an elevated card in the Actual UI, whereas the Reference UI uses a flat, inline layout. Content-wise, the text 'White Toyota Prius' is shortened to 'Toyota Prius' in the Actual UI, the '+ Tip' button is changed to 'Tip', and the 'Address' label is missing under 'Delivery details'. Additionally, the 'Share this delivery' section is styled differently with an explicit share icon in the Actual UI.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.5, 0.5] Average: 0.4400 Range (Min - Max): [0.4000, 0.5000] Significant layout and content differences. Header icons vary. Driver info card structure is changed. Delivery details section has different content. Extra section in actual UI.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.6, 0.7, 0.4, 0.6] Average: 0.5400 Range (Min - Max): [0.4000, 0.7000] The Actual UI exhibits significant structural and stylistic deviations from the Reference UI. Key differences include: 1) The driver information and action controls (Phone, Message, Tip) are grouped into a single card container in the Actual UI, whereas they are separate, flatter elements in the Reference. 2) The 'Tip' button is moved inside the driver card. 3) The 'Address' label is missing from the Delivery details section. 4) Text casing differs ('PM' vs 'pm'). 5) Iconography is inconsistent, specifically the share icon in the top bar and the 'Share this delivery' section. 6) The overall layout uses different spacing and card-based styling compared to the reference's flatter design.
SSIM (↑)	0.71

Case 3



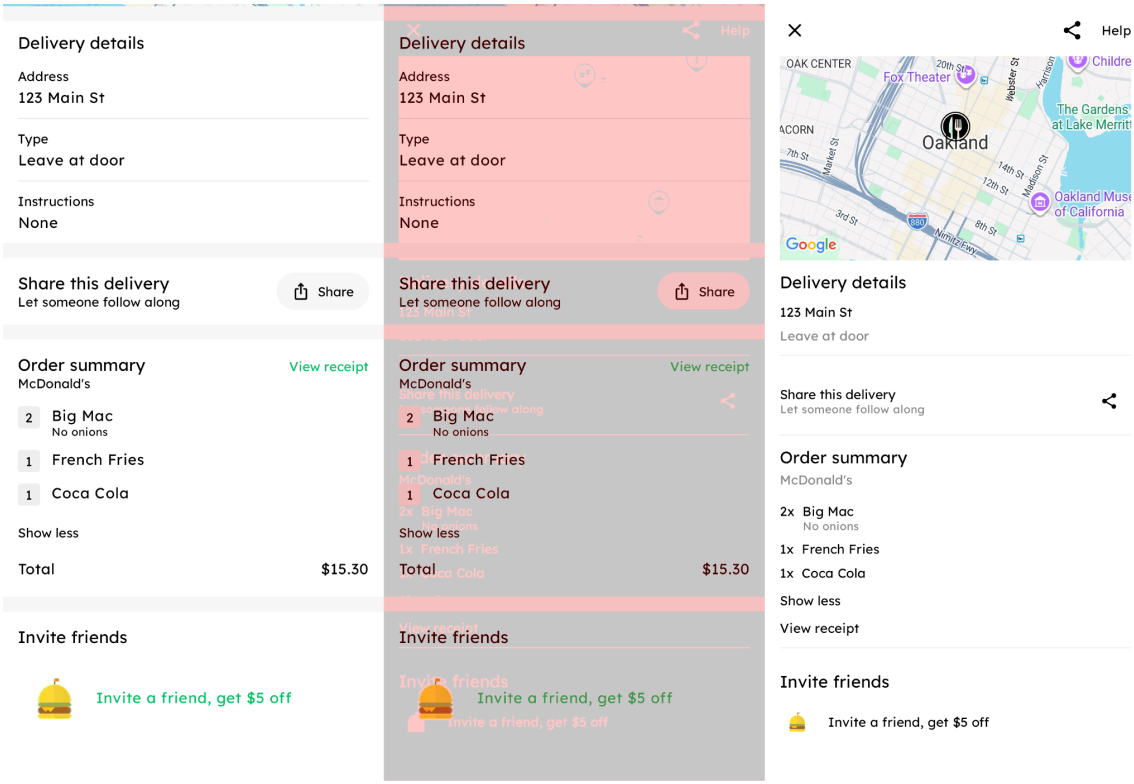
Metric	Result
LPIPS (↓)	0.38
Pixel difference (↓)	18.09%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.5, 0.5, 0.5, 0.5, 0.5] Average: 0.5000 Range (Min - Max): [0.5000, 0.5000] The Actual UI has several major structural and layout differences compared to the Reference UI. 1. The top section in the Actual UI includes a large illustration, a close button, and a 'Help' link, which are completely absent in the Reference UI. 2. The 'Delivery details' section in the Actual UI is missing the 'Type' and 'Instructions' labels, and the 'Instructions: None' text is omitted. 3. In the 'Order summary' section, the quantity indicators are rendered as text (e.g., '2x') instead of boxed numbers, and the 'Hide receipt' button is placed below the items instead of aligned with the header. 4. The 'Invite a friend, get $5 off' text is black in the Actual UI instead of green as in the Reference UI.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.45, 0.4, 0.4, 0.4, 0.4] Average: 0.4100 Range (Min - Max): [0.4000, 0.4500] Major layout differences in the top section, including an added image and different header. Order items use '2x' instead of boxed numbers. 'Hide receipt' placement differs. 'Invite friends' text color changed. Significant structural mismatch.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in structure and content. Key differences include: 1) The addition of a large header illustration and a close (X) button not present in the reference. 2) The 'Instructions' field ('None') is completely missing in the Actual UI. 3) The 'Delivery details' section is grouped under a new header, whereas the reference uses a flat list. 4) The 'Hide receipt' action has moved from the section header to a position below the line items. 5) Item quantities have changed format from '2' to '2x'. These are semantic and structural failures rather than minor rendering noise.
SSIM (↑)	0.84

Case 4



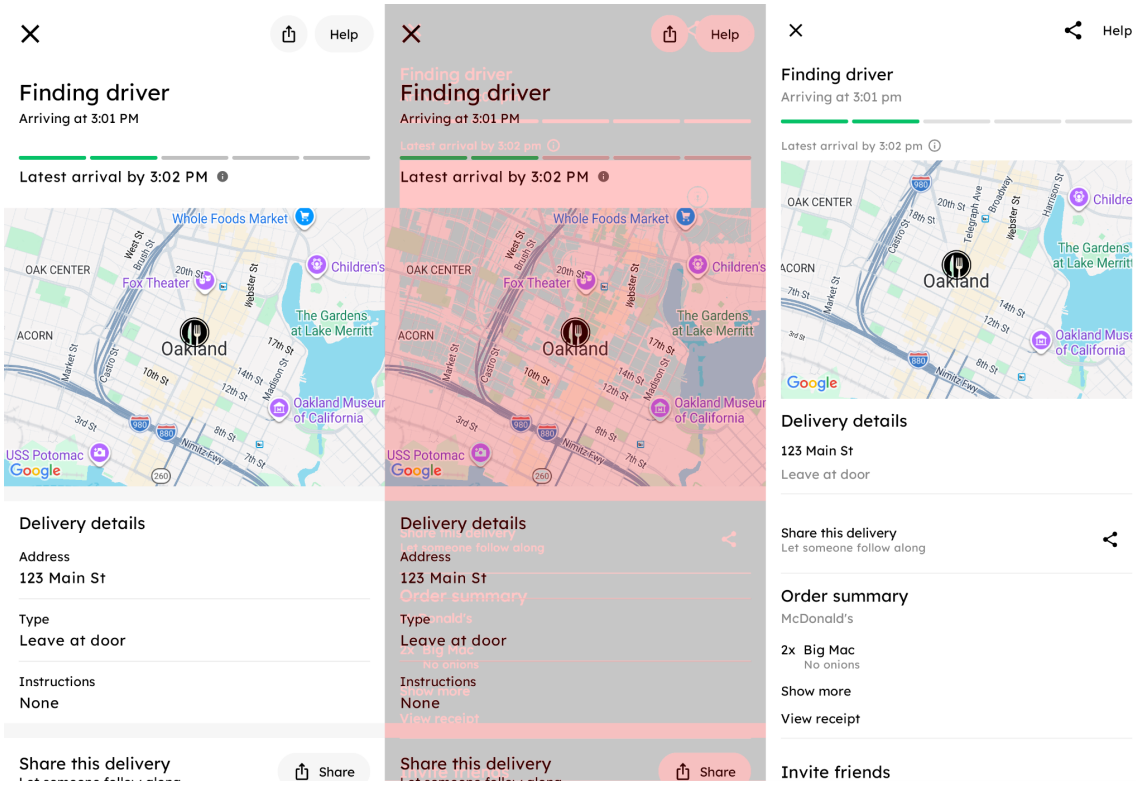
Metric	Result
LPIPS (↓)	0.35
Pixel difference (↓)	20.09%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and content differences compared to the Reference UI. 1) Missing labels: 'Address', 'Type', and 'Instructions' labels are completely missing in the 'Delivery details' section. 2) Missing elements: The 'Total' price row is missing in the Actual UI. 3) Layout changes: The 'View receipt' button has been moved from the top-right of the 'Order summary' section to a list item below 'Show more'. 4) Styling differences: The 'Help' button lacks its pill-shaped background container, the quantity '2' is rendered as '2x' text instead of a badge, and the main illustration is significantly larger. 5) Extra content: An 'Invite friends' section is present in the Actual UI but not in the Reference.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.45, 0.4, 0.4, 0.4] Average: 0.4100 Range (Min - Max): [0.4000, 0.4500] Major layout and content discrepancies. Illustration is oversized. 'Help' button lacks background. 'Delivery details' missing labels. 'Order summary' missing 'Total' row, replaced by 'Invite friends' section.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.6, 0.4, 0.4, 0.6] Average: 0.4800 Range (Min - Max): [0.4000, 0.6000] The Actual UI contains several significant semantic and structural deviations from the Reference UI. 1. Missing Content: The 'Total' line ($15.30) is completely missing in the Actual UI. 2. Missing Labels: The 'Type' and 'Instructions' labels in the Delivery details section are absent. 3. Layout Shifts: The 'View receipt' link has moved from the header of the Order summary to the bottom of the list and changed color from green to black. 4. Component Styling: The 'Help' button lost its pill-shaped background, the quantity '2' lost its boxed styling (replaced by '2x'), and the 'Show more' element lost its chevron icon. 5. Extra Content: An 'Invite friends' section exists in the Actual UI that is not present in the Reference. 6. Scaling: The hero illustration is significantly larger in the Actual UI, causing a major layout shift for all subsequent elements.
SSIM (↑)	0.87

Case 5



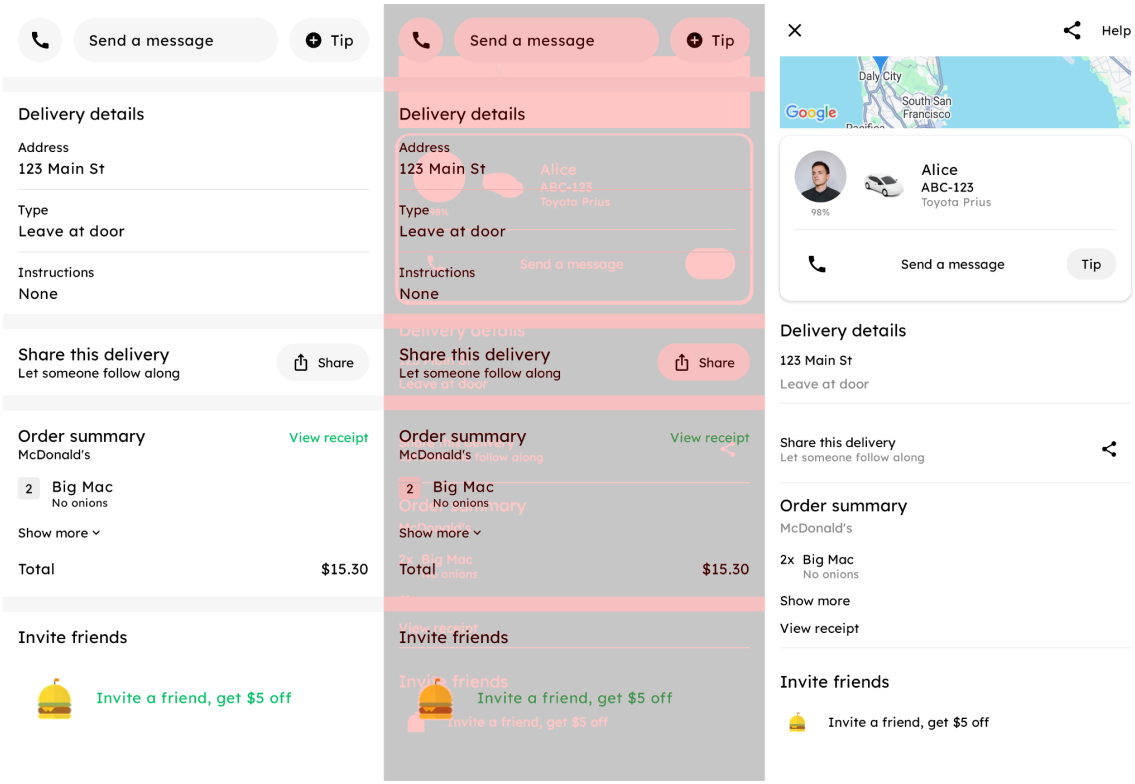
Metric	Result
LPIPS (↓)	0.45
Pixel difference (↓)	36.5%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and visual differences compared to the Reference UI. Key discrepancies include: 1) The addition of a large map component at the top of the Actual UI which is completely absent in the Reference UI. 2) The 'Delivery details' section is missing the 'Type' and 'Instructions' fields, merging 'Leave at door' as a subtitle instead. 3) The 'Share' button style is completely different (a simple icon vs. a styled pill button with text). 4) The 'Order summary' list items use text multipliers (e.g., '2x') instead of the boxed numbers in the Reference, and the 'View receipt' link is styled differently and placed at the bottom instead of the top right. 5) The 'Invite friends' text color is black instead of green.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.45, 0.4, 0.45, 0.4] Average: 0.4200 Range (Min - Max): [0.4000, 0.4500] Major structural and content mismatches. Actual UI adds a map header. Delivery details missing labels. Share button replaced by icon. Order summary lacks total and uses different quantity formatting. Invite friends styling differs.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1) The addition of a large map component at the top of the screen. 2) The 'Delivery details' section in the Actual UI omits explicit labels ('Type', 'Instructions') and completely removes the 'Instructions' field. 3) The 'Share this delivery' button has changed from a pill-shaped button with text to a simple icon on the right. 4) In the 'Order summary', the 'Total' amount is missing, and the 'View receipt' link has been moved from the top-right to the bottom of the list. 5) Quantity formatting changed from plain numbers to 'Nx' format (e.g., '2x'). These are semantic and structural failures rather than minor rendering noise.
SSIM (↑)	0.79

Case 6



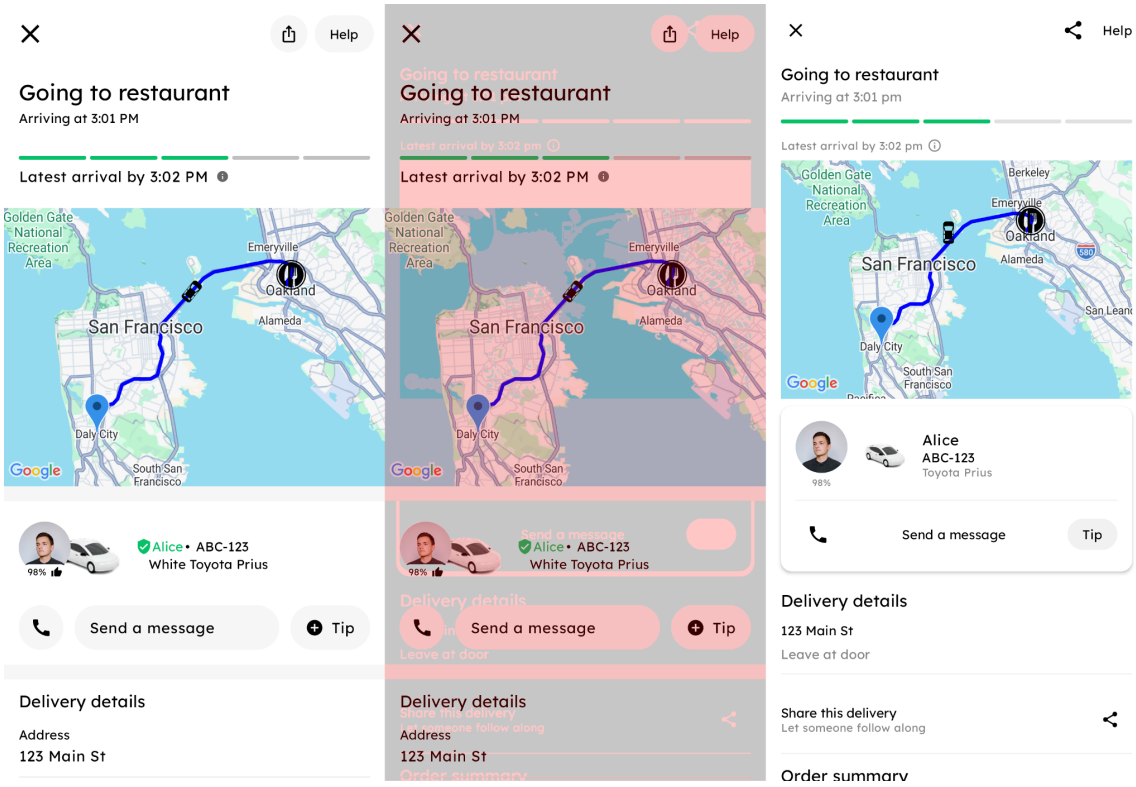
Metric	Result
LPIPS (↓)	0.46
Pixel difference (↓)	44.67%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.6] Average: 0.6000 Range (Min - Max): [0.6000, 0.6000] The Actual UI has several semantic and structural discrepancies compared to the Reference UI. In the 'Delivery details' section, key labels such as 'Address', 'Type', 'Instructions', and the value 'None' are completely missing. The progress bar at the top shows a different state (1 active segment instead of 2). Additionally, different icons are used for sharing (a node-share icon instead of an upload-style share icon), and the layout for 'Share this delivery' is structurally different (inline icon on the right instead of a button at the bottom).
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.45, 0.45, 0.45, 0.5] Average: 0.4500 Range (Min - Max): [0.4000, 0.5000] Significant layout and content differences exist. Header icons ack backgrounds. Progress bar is missing active color. Delivery details section is missing labels like 'Address' and 'Type'. The lower layout is completely restructured, showing different sections.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.5, 0.5, 0.5, 0.4] Average: 0.4600 Range (Min - Max): [0.4000, 0.5000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1. The 'Delivery details' section in the Actual UI is missing explicit labels ('Address', 'Type', 'Instructions') and the 'Instructions' field entirely. 2. The Actual UI includes two large additional sections ('Order summary' and 'Invite friends') that are not present in the Reference. 3. Visual styling for the progress bar (grey vs. green) and the 'Help' button (text-only vs. pill button) is inconsistent. 4. The 'Share this delivery' section uses a different layout and icon placement.
SSIM (↑)	0.69

Case 7



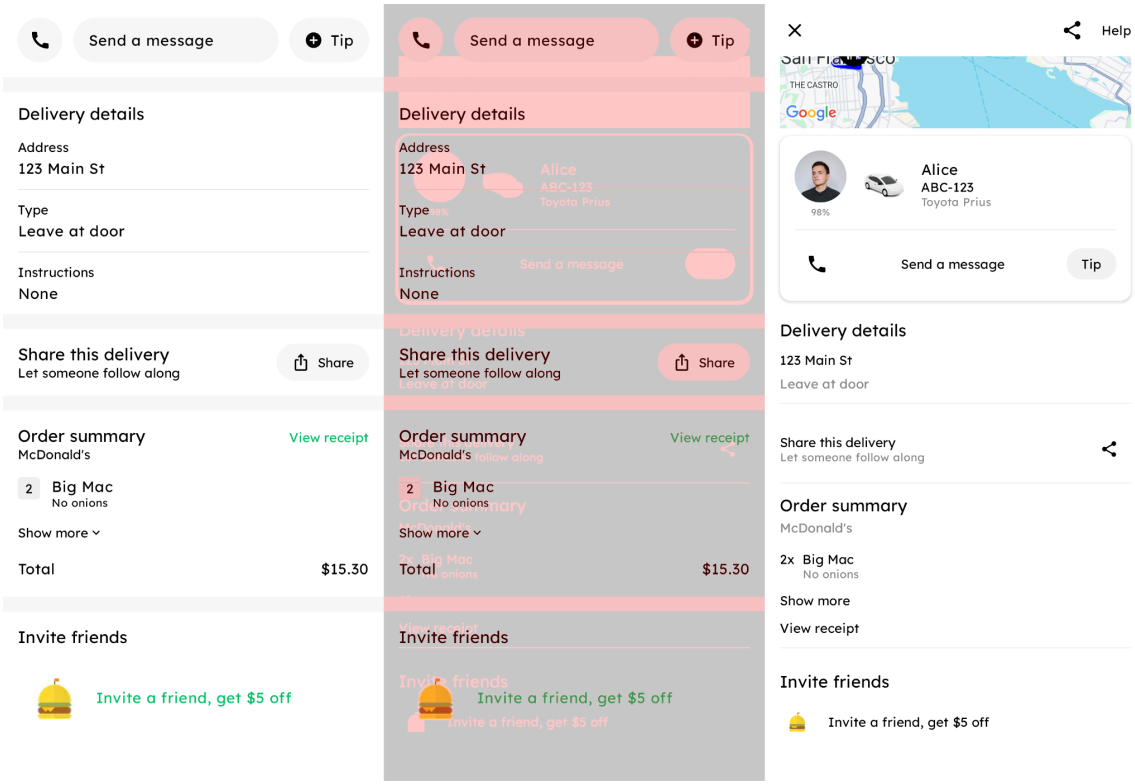
Metric	Result
LPIPS (↓)	0.41
Pixel difference (↓)	27.17%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and content differences compared to the Reference UI. 1) A large map and driver profile card (Alice) are present at the top of the Actual UI but completely missing from the Reference UI. 2) In the 'Delivery details' section, the labels 'Address' and 'Type' are missing, and the 'Instructions' row is entirely absent. 3) The 'Share this delivery' button is rendered as a simple icon instead of a styled button with text. 4) In the 'Order summary' section, the 'Total' price ($15.30) is completely missing, the 'View receipt' link is misplaced at the bottom, and the item quantity is rendered as plain text '2x' instead of a styled badge. 5) Text colors and styling in the 'Invite friends' section do not match.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] Major layout and content differences. Actual UI adds map and driver details. Delivery details missing labels. Order summary layout changed.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1. The addition of a map and driver profile card (Alice, Toyota Prius) at the top, which is entirely absent in the Reference. 2. The 'Delivery details' section in the Actual UI is missing explicit labels ('Address', 'Type', 'Instructions') and the 'None' value for instructions. 3. The 'Share this delivery' section replaces a labeled button with a standalone icon. 4. The 'Order summary' section in the Actual UI is missing the 'Total' price line and has moved the 'View receipt' link from the header to the bottom of the section. 5. Formatting differences exist, such as '2' vs '2x' for item quantity.
SSIM (↑)	0.84

Case 8



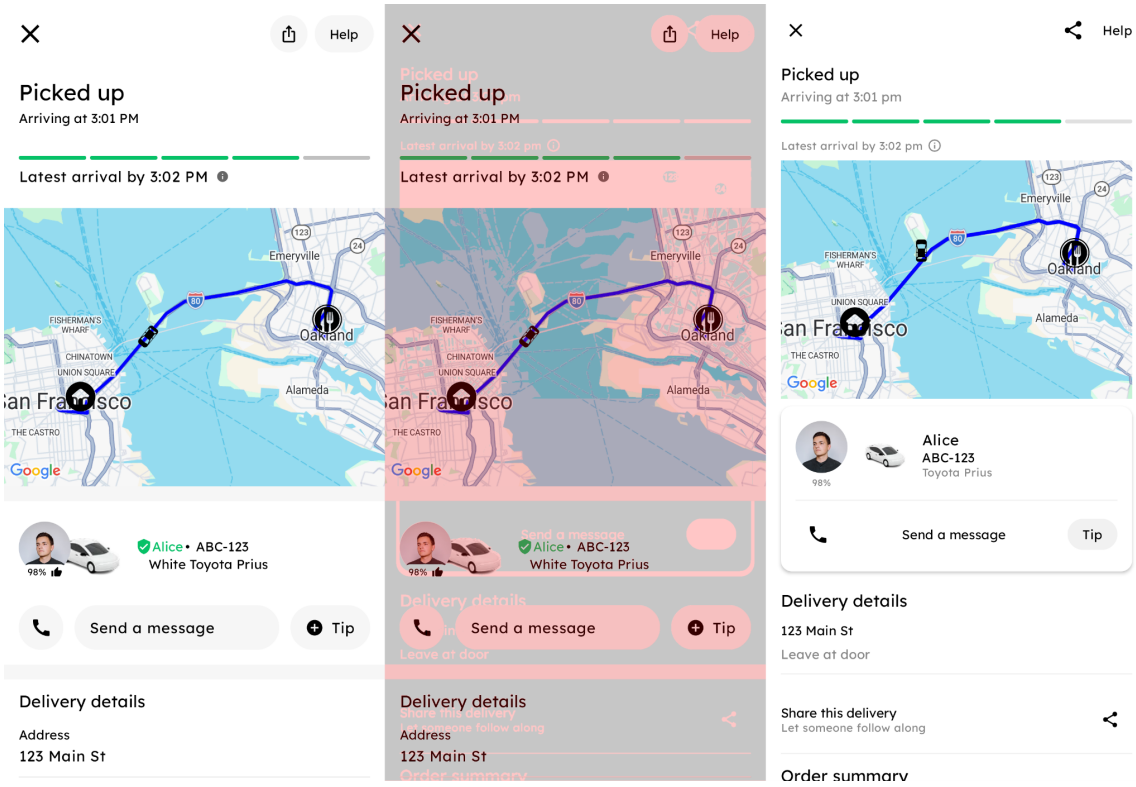
Metric	Result
LPIPS (↓)	0.50
Pixel difference (↓)	49.5%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The Actual UI has several layout and content discrepancies compared to the Reference UI. 1. In the header, the share icon is different (network share icon vs. upload icon). 2. In the driver card, the layout is significantly altered: the text 'White Toyota Prius' is missing 'White', the thumbs-up icon next to '98%' is missing, and the text 'ABC-123' is on a new line. 3. The buttons at the bottom of the driver card ('Send a message' and 'Tip') have different styling, shapes, and icons (the '+' is missing from the Tip button). 4. In the 'Delivery details' section, the 'Address' label is missing, and 'Leave at door' has been added.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.45, 0.45, 0.45, 0.45, 0.45] Average: 0.4500 Range (Min - Max): [0.4500, 0.4500] Significant layout and component differences exist. Header icons differ for share and help. Map is full-width in reference but card-based in actual. Driver info section lacks pill buttons for messaging and tipping and is missing the thumbs-up icon. Delivery details section has mismatched labels.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.5, 0.5, 0.5, 0.5] Average: 0.5200 Range (Min - Max): [0.5000, 0.6000] The Actual UI exhibits significant structural and stylistic deviations from the Reference UI. Key differences include: 1. Header: The 'Share' icon and 'Help' button use different visual styles (the Reference uses a pill-shaped button for Help, while the Actual uses plain text). 2. Driver Card: The layout is completely different; the Actual UI uses a shadowed card container with a different internal arrangement of the driver photo, car icon, and text. 3. Interaction Elements: The 'Tip' button changed from a circular icon button to a pill-shaped text button. The 'Send a message' field changed from a text input style to a simple text link. 4. Content: The Actual UI includes additional sections like 'Share this delivery' and 'Order summary', and extra text ('Leave at door') that are not present in the Reference. These are semantic and structural failures rather than minor rendering noise.
SSIM (↑)	0.71

Case 9



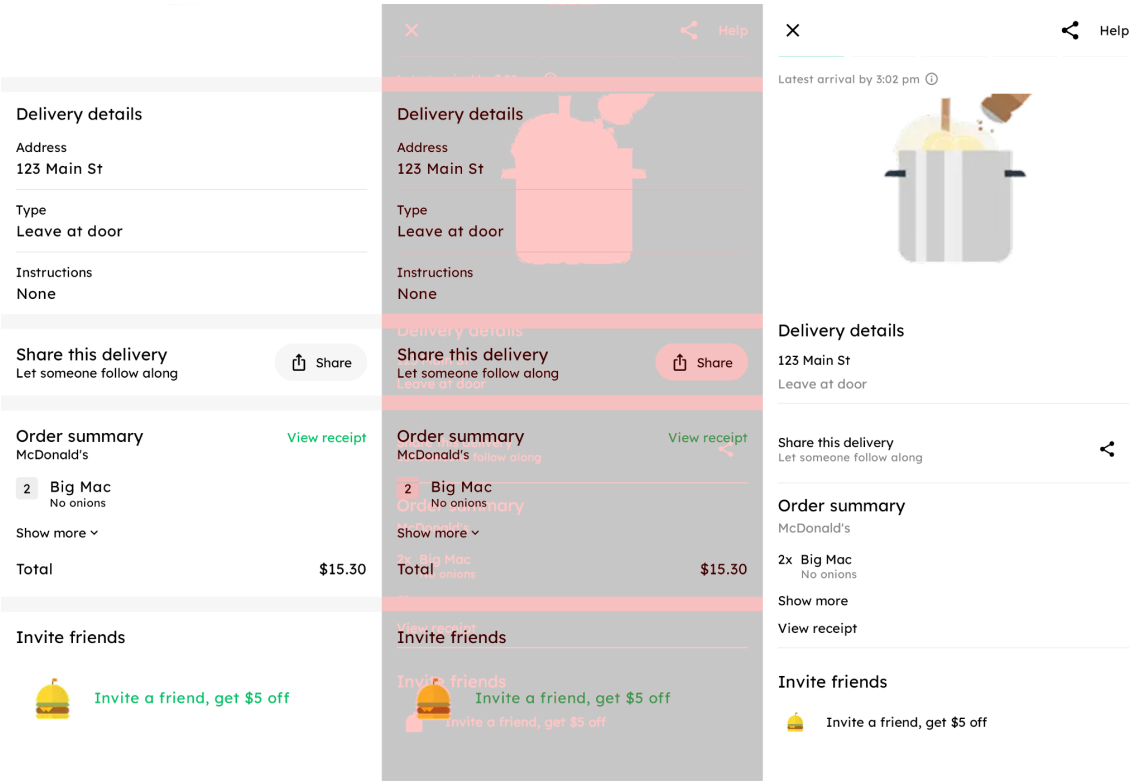
Metric	Result
LPIPS (↓)	0.41
Pixel difference (↓)	27.17%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and content differences compared to the Reference UI. 1. The top of the screen in the Actual UI includes a map and a driver profile card (Alice, Toyota Prius) which are completely missing from the Reference UI. 2. In the 'Delivery details' section, the labels 'Address', 'Type', and 'Instructions' are missing, and the 'Instructions: None' field is omitted entirely. 3. The 'Share' button is replaced by a simple share icon. 4. In the 'Order summary' section, the 'Total $15.30' price is completely missing, and the 'View receipt' link is relocated to the bottom of the section instead of the top-right. 5. The item quantity is formatted differently ('2x Big Mac' vs '2 Big Mac').
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.3] Average: 0.3800 Range (Min - Max): [0.3000, 0.4000] Major structural and content differences exist. Actual UI adds a map/driver section not in reference. Several fields are missing in Delivery Details and Order Summary. Significant semantic mismatch.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key differences include: 1. The addition of a large map and driver profile card (Alice) at the top, which are entirely absent in the Reference. 2. The 'Delivery details' section in the Actual UI is missing the 'Instructions' field. 3. The 'Order summary' section in the Actual UI moves the 'View receipt' link from the header to a list item at the bottom. 4. The top navigation bar in the Reference (containing the message input and tip button) is replaced by a map header in the Actual UI. These are major semantic and structural failures.
SSIM (↑)	0.83

Case 10



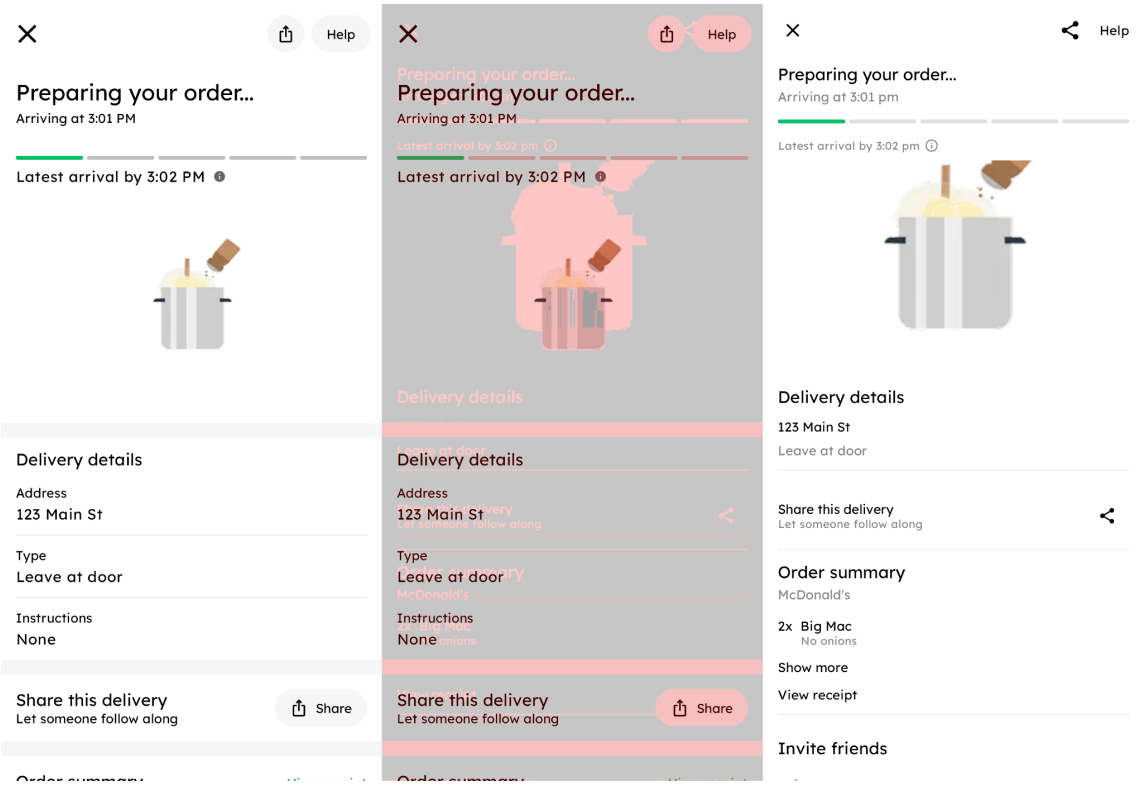
Metric	Result
LPIPS (↓)	0.51
Pixel difference (↓)	49.68%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The Actual UI has several structural and content differences compared to the Reference UI. 1. The driver information section in the Actual UI is enclosed in a card with a drop shadow, whereas the Reference UI uses a flat layout. 2. The driver's car description is missing the word 'White' ('Toyota Prius' vs 'White Toyota Prius'). 3. The rating is missing the thumbs-up emoji ('98%' vs '98% □'). 4. The action buttons (Phone, Send a message, Tip) have different styling and layout; the 'Tip' button is missing the '+' icon. 5. In the 'Delivery details' section, the 'Address' label is missing in the Actual UI. 6. The share icon in the top right header is a different style (node-share vs upload-share).
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.45, 0.45, 0.45, 0.45, 0.45] Average: 0.4500 Range (Min - Max): [0.4500, 0.4500] The actual UI shows major layout and styling deviations. Header icons lack background shapes. Driver info is incorrectly formatted as a card instead of inline. Action buttons have different structures. Delivery details content differs.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.7] Average: 0.6200 Range (Min - Max): [0.6000, 0.7000] The Actual UI exhibits significant structural and content discrepancies compared to the Reference UI. Key failures include: 1) Layout Restructuring: The driver information and action buttons (Phone, Message, Tip) are grouped into a card container in the Actual UI, whereas the Reference uses a flat list layout. 2) Missing Content: The 'Address' label is missing from the delivery details section, and the word 'White' is missing from the car description ('Toyota Prius' vs 'White Toyota Prius'). 3) Iconography: The top-right share icon uses a different glyph (standard share vs. upload icon), and the 'Tip' button is missing its plus (+) icon. 4) Component Styling: The 'Send a message' and 'Tip' elements have different background shapes and alignments. While the functional state is the same, the visual fidelity is low.
SSIM (↑)	0.71

Case 11



Metric	Result
LPIPS (↓)	0.37
Pixel difference (↓)	19.83%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant structural and content differences compared to the Reference UI. 1) A large top section with a cooking pot illustration, progress bar, and 'Latest arrival' text is present in the Actual UI but completely missing in the Reference. 2) In 'Delivery details', the labels 'Address', 'Type', and 'Instructions' are missing, and the 'Instructions: None' field is omitted entirely. 3) The 'Share' button has a completely different style (icon-only vs. pill button with text). 4) In 'Order summary', the 'Total $15.30' is missing, the 'View receipt' link is relocated to the bottom, and the quantity indicator '2' in a grey box is changed to text '2x'. 5) The 'Invite friends' promo text color is black instead of green.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI has significant layout and content differences compared to the Reference UI. The top section with the pot illustration is missing in the Reference UI but present in the Actual UI. The 'Delivery details' section in the Actual UI lacks the 'Type' and 'Instructions' fields. The 'Share this delivery' section uses a different icon and layout. The 'Order summary' section has layout differences, such as the 'View receipt' link placement and the quantity indicator format. The 'Invite friends' section also has layout and styling differences.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.4, 0.4, 0.4, 0.4, 0.4] Average: 0.4000 Range (Min - Max): [0.4000, 0.4000] The Actual UI significantly deviates from the Reference UI in both structure and content. Key semantic failures include: 1) Missing labels for 'Address', 'Type', and 'Instructions' in the delivery details section. 2) The 'Total' price is completely missing from the order summary. 3) The 'Share' button has been replaced by a simple icon, changing the interaction model. 4) The 'View receipt' link has moved from a section header position to a list item at the bottom. 5) A large illustration is present at the top of the Actual UI which is not in the Reference. 6) The quantity indicator for the 'Big Mac' changed from a boxed number to '2x' text.
SSIM (↑)	0.87

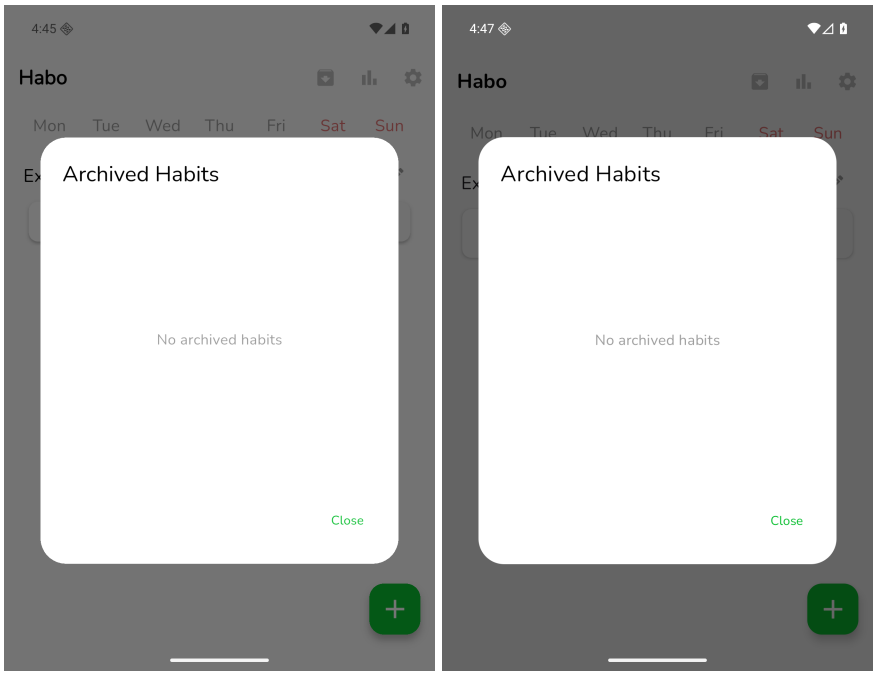
Case 12



Metric	Result
LPIPS (↓)	0.34
Pixel difference (↓)	19.28%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.6] Average: 0.6000 Range (Min - Max): [0.6000, 0.6000] The Actual UI has several semantic and structural differences compared to the Reference UI. In the 'Delivery details' section, the labels 'Address', 'Type', and 'Instructions' are missing in the Actual UI. The 'Share this delivery' component uses a simple share icon instead of a styled pill button. Additionally, the header share icon is different, and there are layout shifts throughout the page.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.6, 0.6, 0.45, 0.6, 0.45] Average: 0.5400 Range (Min - Max): [0.4500, 0.6000] The Actual UI has several layout and content differences compared to the Reference UI. The top right icons are different (share icon vs upload icon). The 'Delivery details' section in the Actual UI lacks the 'Address', 'Type', and 'Instructions' labels present in the Reference UI, and the layout is more compressed. The 'Share this delivery' section is formatted differently, with the share button replaced by an icon on the right. The 'Order summary' section is also missing in the Reference UI's visible area but present in the Actual UI.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.5, 0.6] Average: 0.5800 Range (Min - Max): [0.5000, 0.6000] The Actual UI has several semantic and structural discrepancies compared to the Reference UI. 1. The 'Instructions' field under 'Delivery details' is completely missing in the Actual UI. 2. The 'Share' button in the 'Share this delivery' section has been changed from a pill-shaped button with text and icon to a simple icon on the right. 3. The top navigation bar uses a different share icon (three-dot vs. box-arrow) and the 'Help' button lacks its container/border. 4. The layout of the 'Delivery details' section differs in spacing and font weight. While the overall state is the same, these component-level changes constitute a failure in visual and content fidelity.
SSIM (↑)	0.87

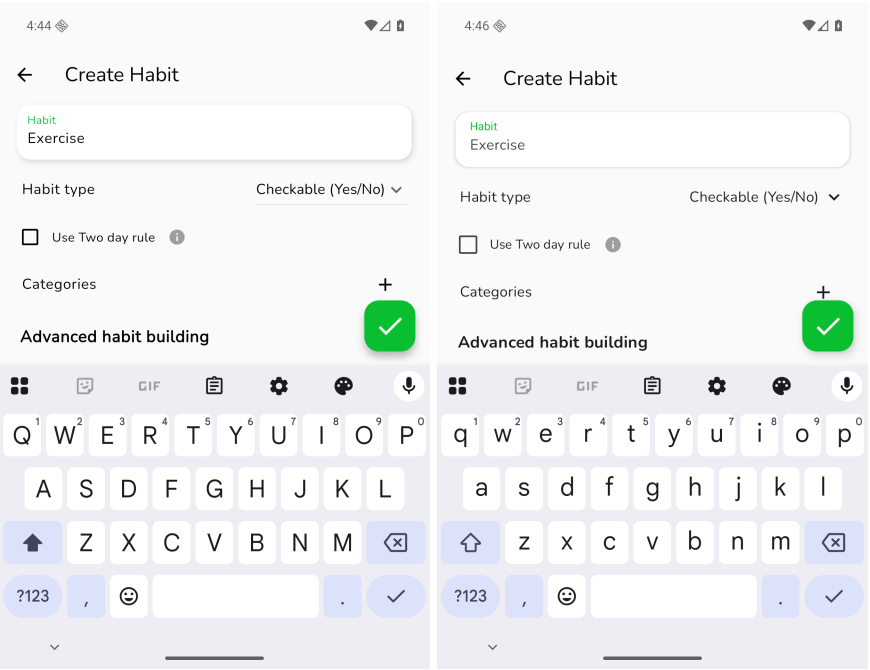
Side-by-side comparisons (golden / agent output)

Case 13



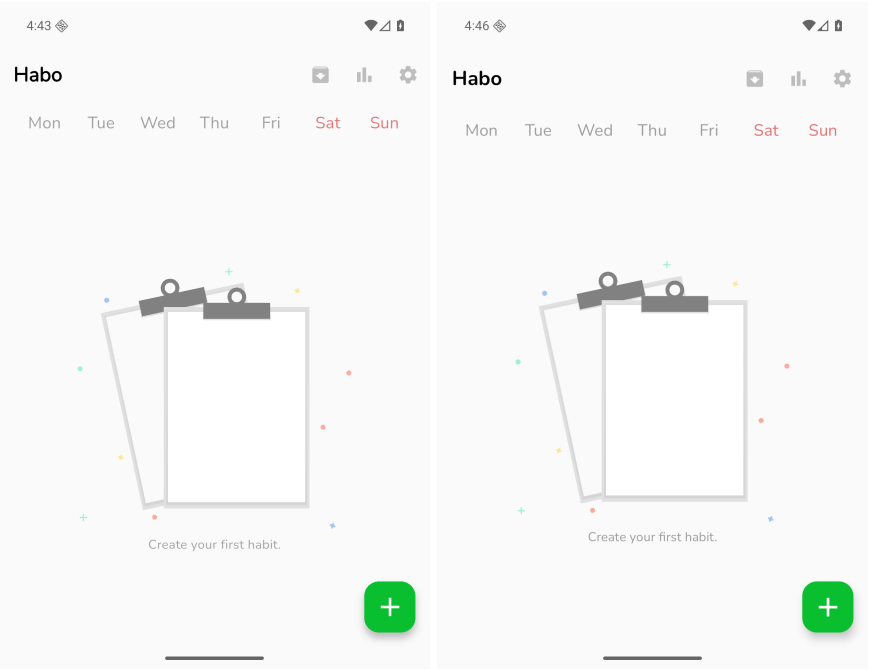
Metric	Result
LPIPS (↓)	0.049
Pixel difference (↓)	47.54%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.9, 0.9, 0.9, 0.9, 0.9] Average: 0.9000 Range (Min - Max): [0.9000, 0.9000] The Actual UI is a near-perfect match to the Reference UI. All structural elements, text content, colors, and layout positions are identical, with only minor system-level differences in the status bar time.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [1.0, 1.0, 1.0, 1.0, 0.9] Average: 0.9800 Range (Min - Max): [0.9000, 1.0000] The actual UI perfectly matches the reference UI in terms of layout, content, visual styling, and state representation. The modal dialog, its title, empty state message, and action button are identical. Minor differences in the system status bar (time, battery) are expected and ignored.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.8, 0.8, 0.8, 0.75, 0.75] Average: 0.7800 Range (Min - Max): [0.7500, 0.8000] The Actual UI perfectly mirrors the Reference UI's structure, content, and styling. Differences are limited to system status bar elements like time and battery status, which are irrelevant to the app's functional layout.
SSIM (↑)	0.965

Case 14



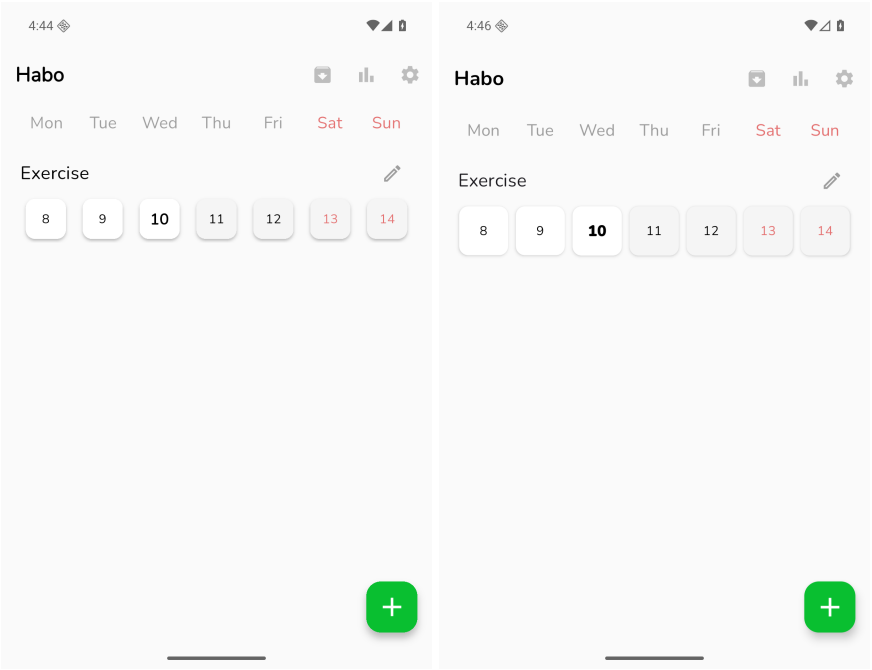
Metric	Result
LPIPS (↓)	0.139
Pixel difference (↓)	7.32%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.9, 0.9, 0.9, 0.9, 0.9] Average: 0.9000 Range (Min - Max): [0.9000, 0.9000] The application UI matches perfectly in layout, content, and styling. The only difference is the system keyboard state, which shows uppercase letters in the reference and lowercase in the actual UI.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.8, 0.8, 0.8, 0.8, 0.8] Average: 0.8000 Range (Min - Max): [0.8000, 0.8000] The main application UI matches perfectly in layout, content, and styling. However, there is a state mismatch in the on-screen keyboard: the reference image displays uppercase keys, whereas the actual image displays lowercase keys.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.8, 0.7] Average: 0.7200 Range (Min - Max): [0.7000, 0.8000] The application UI is semantically identical. All labels, input fields, and interactive elements are correctly positioned and styled. The only discrepancies are the system keyboard case (uppercase vs. lowercase) and the status bar timestamp, which are considered environmental noise rather than UI failures.
SSIM (↑)	0.922

Case 15



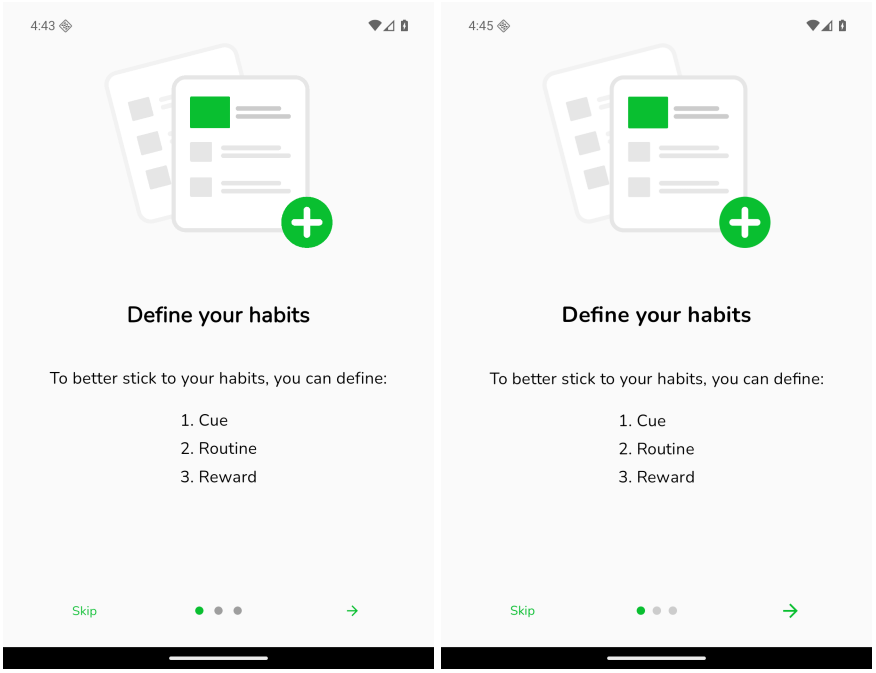
Metric	Result
LPIPS (↓)	0.089
Pixel difference (↓)	4.53%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.8, 0.8, 0.8, 0.8, 0.8] Average: 0.8000 Range (Min - Max): [0.8000, 0.8000] The actual UI is functionally and visually identical to the reference UI. The only difference is a minor system time change in the status bar, which does not affect the application's layout or content.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [1.0, 1.0, 0.85, 1.0, 1.0] Average: 0.9700 Range (Min - Max): [0.8500, 1.0000] The actual UI is a perfect semantic and visual match to the reference UI. All structural elements, text content, icons, colors, and layout positioning are identical. The only differences are minor variations in the system status bar (time and battery icon), which are considered acceptable noise and do not affect the application's UI fidelity.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.85, 0.8, 0.8, 0.85, 0.8] Average: 0.8200 Range (Min - Max): [0.8000, 0.8500] The Actual UI is a near-perfect semantic match to the Reference UI. All structural elements, including the header, day selector, central empty-state illustration, and floating action button, are correctly positioned. Text content and iconography are identical. Minor variations in the system status bar time are considered non-functional noise.
SSIM (↑)	0.955

Case 16



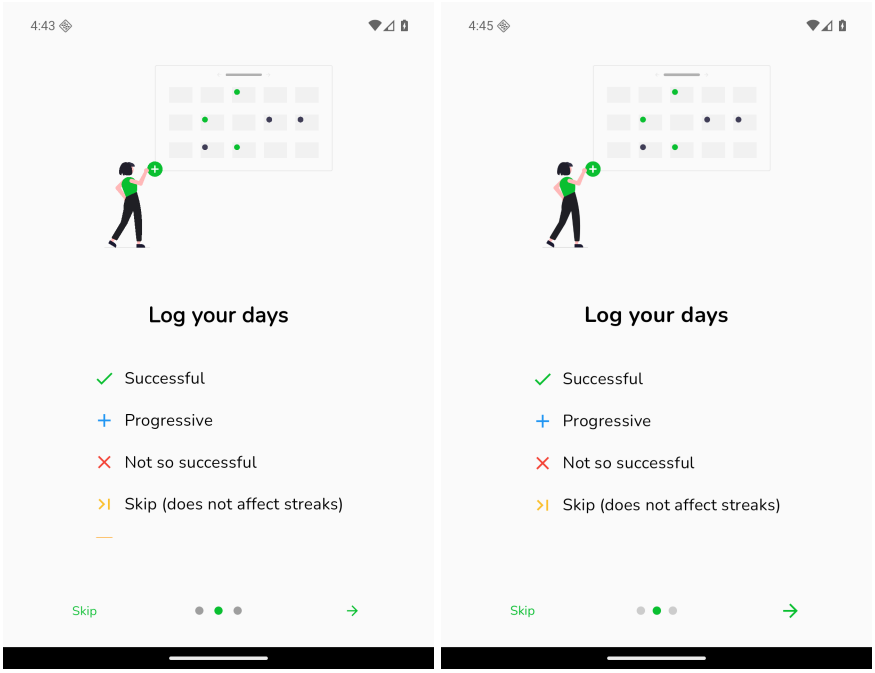
Metric	Result
LPIPS (↓)	0.097
Pixel difference (↓)	6.67%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.9, 0.9, 0.9, 0.9, 0.9] Average: 0.9000 Range (Min - Max): [0.9000, 0.9000] The Actual UI is a near-perfect match to the Reference UI. All structural elements, text labels, colors, and states are identical, with only minor status bar time differences.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [1.0, 1.0, 1.0, 1.0, 1.0] Average: 1.0000 Range (Min - Max): [1.0000, 1.0000] The actual UI perfectly matches the reference UI in terms of layout, content, styling, and state. All elements including text, icons, colors, and spacing are identical. The only difference is the time in the system status bar, which is considered minor rendering noise.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.85, 0.8, 0.85, 0.8, 0.8] Average: 0.8200 Range (Min - Max): [0.8000, 0.8500] The Actual UI is a near-perfect replica of the Reference UI. All structural elements, including the header, navigation icons, and the 'Exercise' section, are correctly placed. Content fidelity is high, with identical text and date numbering. Semantic styling, such as the bolded '10' and the red color for weekend dates, is accurately maintained. Minor discrepancies in the status bar and slight font weight variations are considered rendering noise.
SSIM (↑)	0.958

Case 17



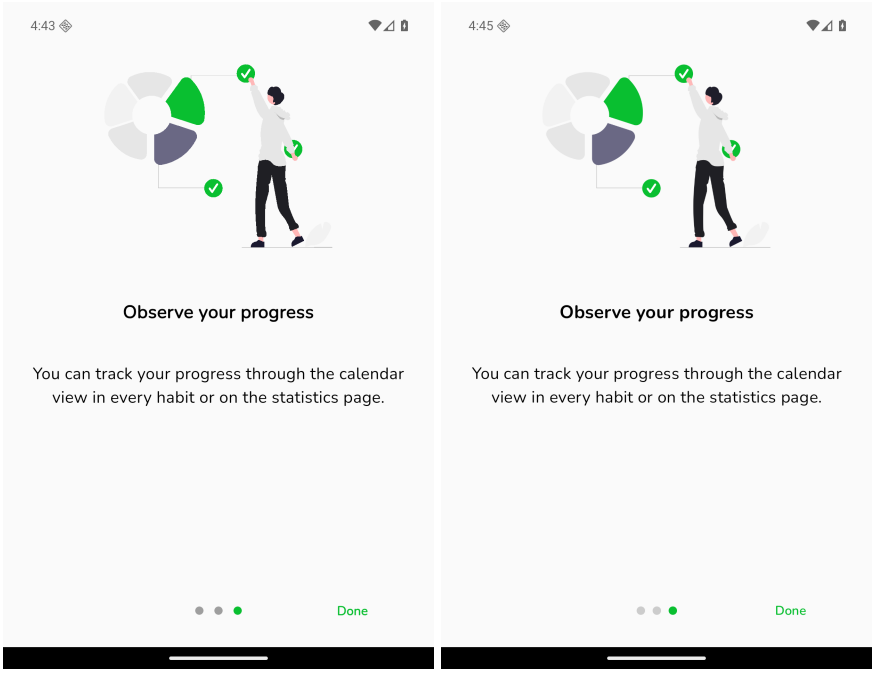
Metric	Result
LPIPS (↓)	0.016
Pixel difference (↓)	1.90%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.9, 0.9, 0.9, 0.9, 0.9] Average: 0.9000 Range (Min - Max): [0.9000, 0.9000] The Actual UI is an exact semantic and visual match to the Reference UI. All text, layout elements, illustrations, and navigation controls are identical, with only a minor, negligible difference in the system status bar time.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.9, 1.0, 0.9, 1.0, 1.0] Average: 0.9600 Range (Min - Max): [0.9000, 1.0000] The actual UI is a perfect semantic and functional match to the reference UI. All elements, including the illustration, text content, list items, and bottom navigation controls, are present and correctly positioned. The visual styling, colors, and typography are consistent. The only differences are minor variations in the system status bar (time), which are expected and do not impact the application's UI fidelity.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.8, 0.8, 0.9, 0.8, 0.8] Average: 0.8200 Range (Min - Max): [0.8000, 0.9000] The Actual UI is a near-perfect match to the Reference UI. All semantic elements, including the central illustration, typography, and navigation controls, are identical in placement and style. Minor variations in the status bar time are negligible system noise.
SSIM (↑)	0.978

Case 18



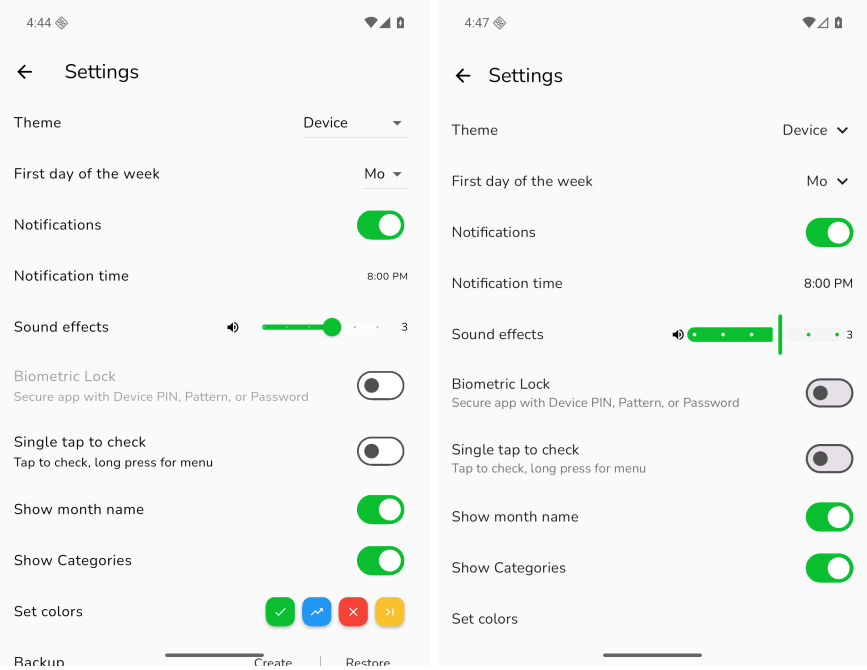
Metric	Result
LPIPS (↓)	0.018
Pixel difference (↓)	1.71%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.9, 0.9, 0.9, 0.9, 0.9] Average: 0.9000 Range (Min - Max): [0.9000, 0.9000] The actual UI matches the reference UI in layout, content, and styling. A stray orange line present in the reference UI under the list items has been removed in the actual UI, which improves visual cleanliness.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.7, 0.8, 0.7, 0.7, 0.9] Average: 0.7600 Range (Min - Max): [0.7000, 0.9000] The actual UI is missing the orange horizontal line element located below the ‘Skip (does not affect streaks)’ text in the reference UI.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The Actual UI is a near-perfect semantic match for the Reference UI. All primary elements, including the illustration, typography, iconography, and navigation controls, are correctly placed and styled. The only notable difference is the absence of a small horizontal divider below the final list item in the Actual UI. This is a minor layout discrepancy that does not affect functionality.
SSIM (↑)	0.979

Case 19



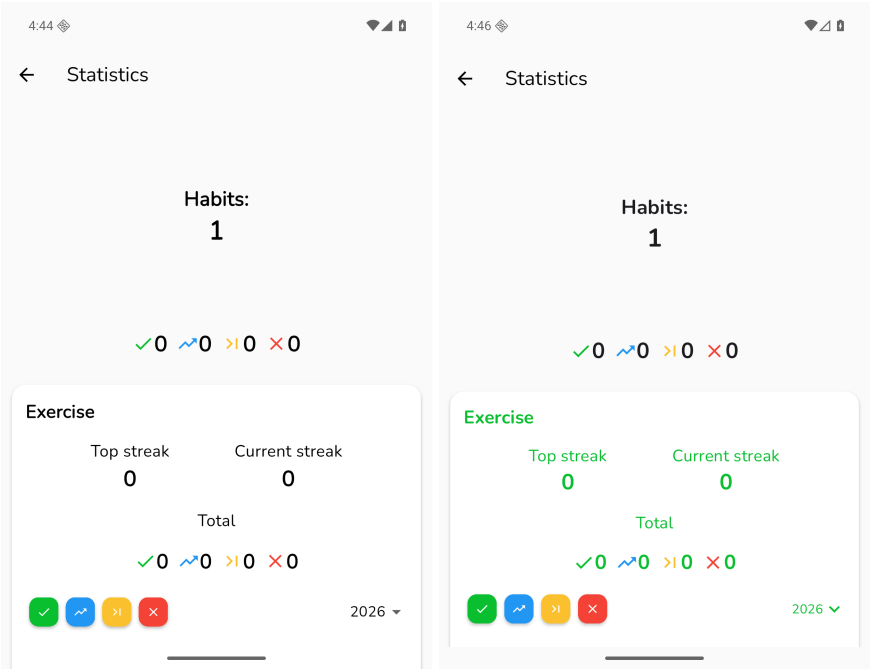
Metric	Result
LPIPS (↓)	0.015
Pixel difference (↓)	1.55%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.8, 0.8, 0.8, 0.8, 0.8] Average: 0.8000 Range (Min - Max): [0.8000, 0.8000] The Actual UI is a near-perfect match to the Reference UI. All elements, including the illustration, header, description text, page indicators, and 'Done' button, are identically placed and styled. The only difference is a minor system time change in the status bar, which is negligible.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.9, 0.85, 0.9, 1.0, 1.0] Average: 0.9300 Range (Min - Max): [0.8500, 1.0000] The Actual UI is a perfect semantic and visual match to the Reference UI. Layout, typography, colors, and content are identical. The only difference is the time in the status bar, which is expected noise.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.9, 0.8, 0.8, 0.9, 0.9] Average: 0.8600 Range (Min - Max): [0.8000, 0.9000] The Actual UI is a near-perfect replica of the Reference UI. All semantic elements, including the illustration, typography, pagination indicators, and the 'Done' button, are identical in placement and style. The only discrepancy is the system clock in the status bar, which is considered non-functional noise.
SSIM (↑)	0.980

Case 20



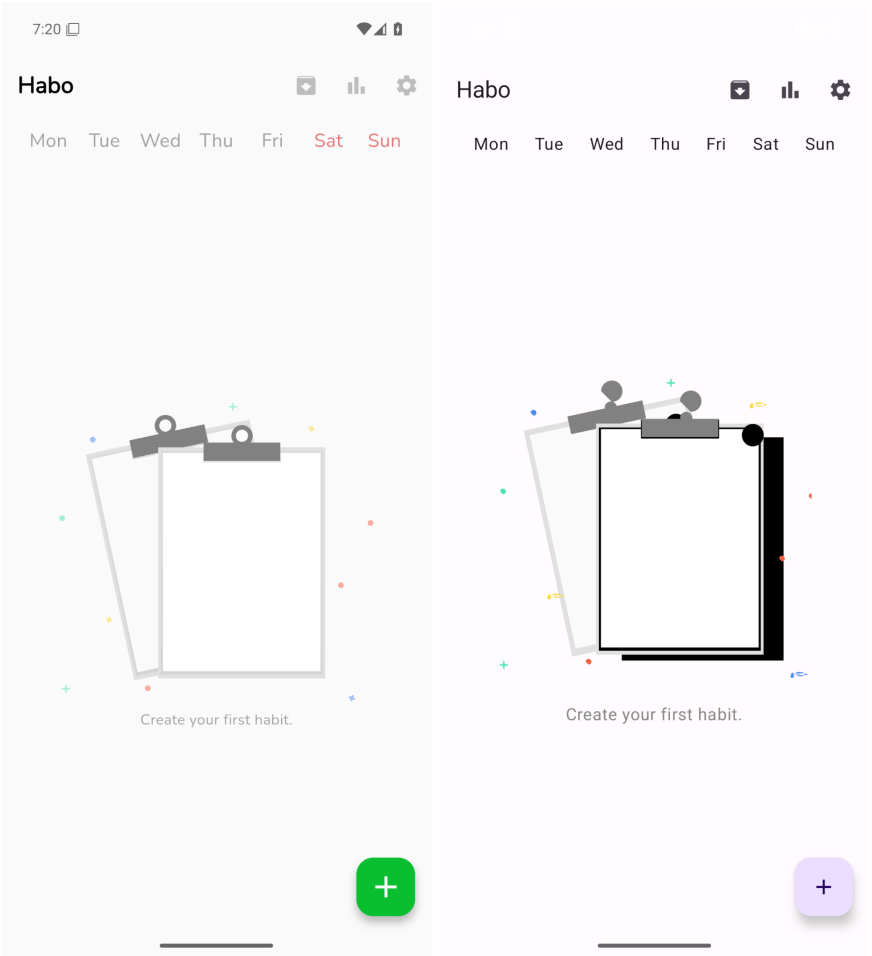
Metric	Result
LPIPS (↓)	0.265
Pixel difference (↓)	9.32%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.55, 0.55, 0.55, 0.55, 0.55] Average: 0.5500 Range (Min - Max): [0.5500, 0.5500] The Actual UI fails to render several key components from the Reference UI. The interactive color buttons under ‘Set colors’ and the entire ‘Backup’ section at the bottom are missing. Additionally, the ‘Sound effects’ slider is visually broken, rendering with a vertical bar instead of a circular thumb, and the dropdown menus have different styling.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.6] Average: 0.6000 Range (Min - Max): [0.6000, 0.6000] The Actual UI has several significant semantic failures compared to the Reference UI. The ‘Sound effects’ slider uses a completely different visual component (segmented bar vs continuous slider). The toggle switches have a different visual style (filled track vs outlined track for the OFF state). Most importantly, the Actual UI is missing the four colored buttons in the ‘Set colors’ row and completely lacks the ‘Backup’ section at the bottom of the screen.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.65, 0.6, 0.65, 0.6] Average: 0.6200 Range (Min - Max): [0.6000, 0.6500] The actual UI fails to render the interactive color icons next to ‘Set colors’ and completely omits the ‘Backup’ section with its ‘Create’ and ‘Restore’ buttons. There is also a visual discrepancy in the ‘Sound effects’ slider, which uses a vertical bar instead of a circular thumb.
SSIM (↑)	0.873

Case 21



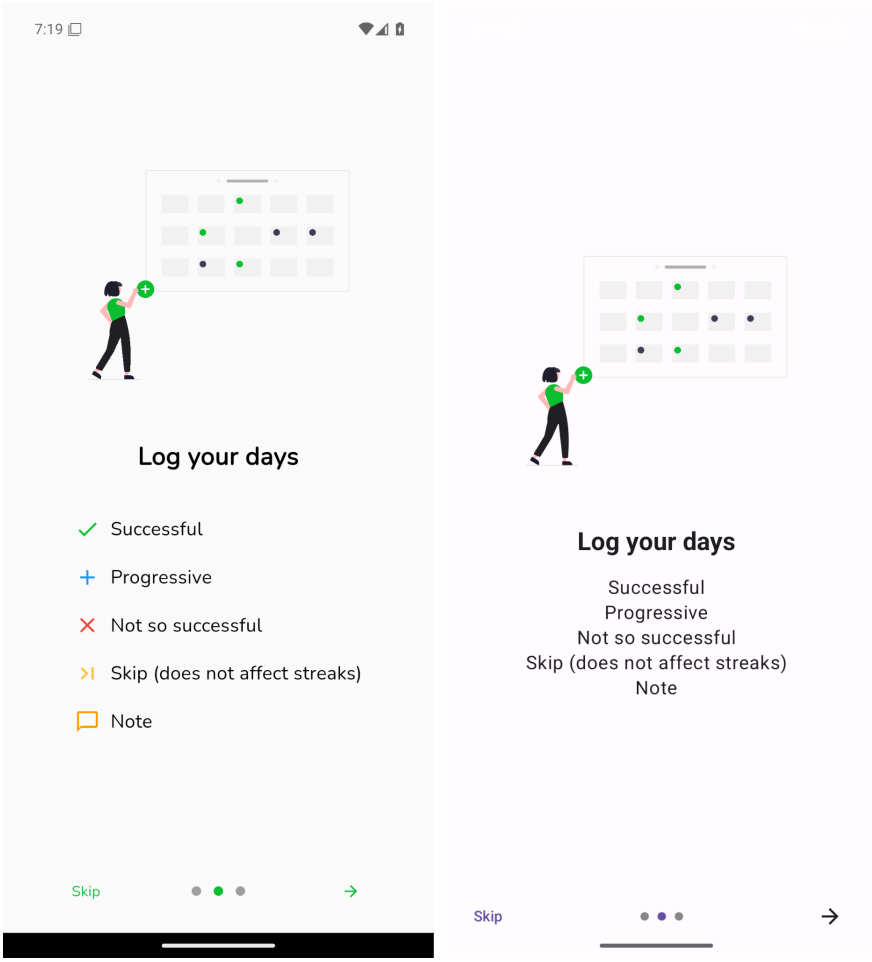
Metric	Result
LPIPS (↓)	0.091
Pixel difference (↓)	8.01%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The layout and content of both screens are identical. However, there is a visual discrepancy in text coloring: the text elements inside the 'Exercise' card (including headers, labels, and values) are rendered in green in the Actual UI, whereas they are black/dark grey in the Reference UI.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.8, 0.7, 0.6, 0.6, 0.6] Average: 0.6600 Range (Min - Max): [0.6000, 0.8000] The structural layout and text content are identical between the two images. However, there is a significant visual inconsistency: the text color within the 'Exercise' card (including the title, labels, values, and the year dropdown) is incorrectly rendered as green in the actual UI, whereas it is black in the reference UI.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.85, 0.7, 0.8, 0.8, 0.65] Average: 0.7600 Range (Min - Max): [0.6500, 0.8500] The structural layout and text content are identical across both versions. The only notable difference is the color of the labels within the 'Exercise' card and the dropdown arrow, which are green in the Actual UI but black in the Reference. This is a minor styling variation rather than a functional failure.
SSIM (↑)	0.950

Case 22



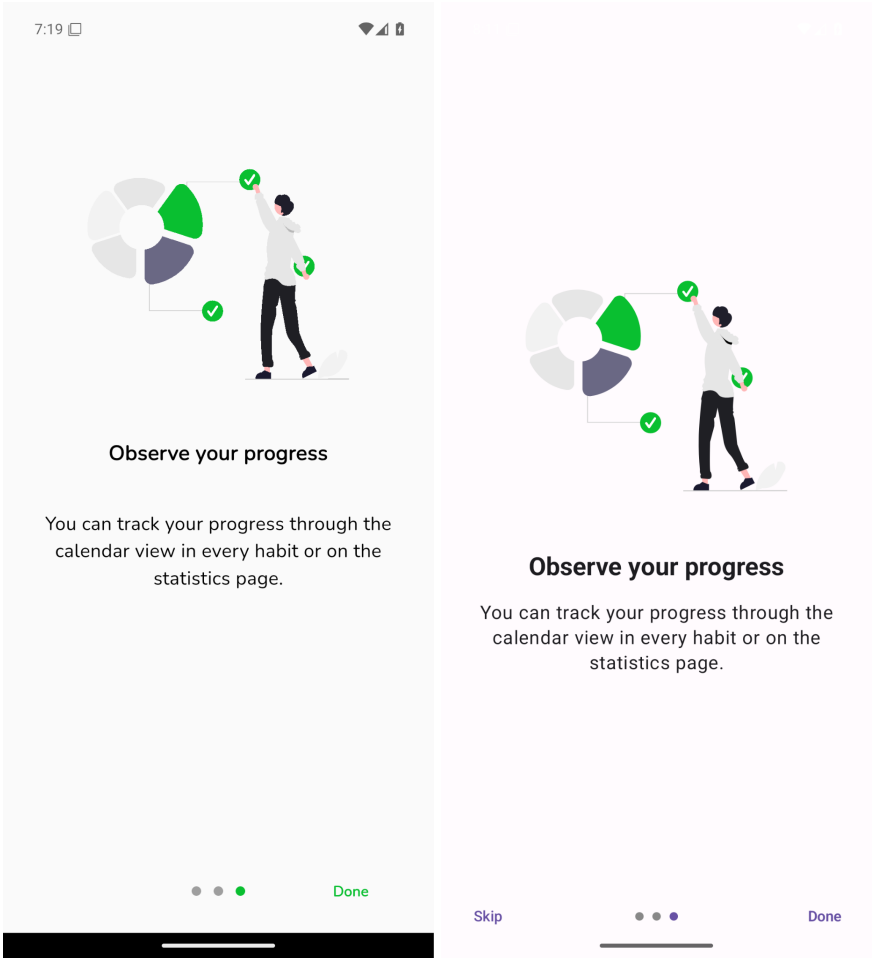
Metric	Result
LPIPS (↓)	0.169
Pixel difference (↓)	90.28%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.6] Average: 0.6000 Range (Min - Max): [0.6000, 0.6000] The actual UI has several visual discrepancies compared to the reference. The central clipboard illustration is poorly rendered with broken black borders and distorted shapes. Additionally, the weekend labels (Sat, Sun) lack the red accent color, and the floating action button is light purple instead of green.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.6, 0.6, 0.6, 0.6, 0.6] Average: 0.6000 Range (Min - Max): [0.6000, 0.6000] Layout and text match, but significant visual styling errors exist. Weekend days lack red coloring. The central illustration has incorrect thick black borders and solid shapes instead of light grey outlines. The FAB is purple instead of green. These are noticeable visual consistency failures.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.5, 0.6, 0.5, 0.6] Average: 0.5600 Range (Min - Max): [0.5000, 0.6000] The layout structure is consistent, but visual fidelity is compromised. The central illustration exhibits rendering artifacts with incorrect black fills. Additionally, the floating action button and weekend text labels do not match the reference colors.
SSIM (↑)	0.929

Case 23



Metric	Result
LPIPS (↓)	0.257
Pixel difference (↓)	99.99%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.55, 0.55, 0.55, 0.55, 0.55] Average: 0.5500 Range (Min - Max): [0.5500, 0.5500] The actual UI fails to render the icons next to the list items, resulting in a centered text-only list instead of a left-aligned icon list. Additionally, the bottom navigation elements (Skip button, page indicator, and arrow) use a purple accent color instead of the expected green.
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.4, 0.5, 0.4, 0.4, 0.5] Average: 0.4400 Range (Min - Max): [0.4000, 0.5000] The Actual UI has significant semantic and visual failures compared to the Reference UI. All icons in the 'Log your days' list are missing, and the text is incorrectly centered instead of left-aligned. Furthermore, the colors for the bottom navigation elements ('Skip' text, the active pagination dot, and the right arrow) are incorrect, appearing purple and black instead of the expected green.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.6, 0.6, 0.5, 0.45] Average: 0.5375 Range (Min - Max): [0.45, 0.6000] The Actual UI fails to render the icons associated with the list items and incorrectly centers the text instead of left-aligning it. Additionally, the color scheme for the bottom navigation elements (Skip text and arrow) deviates from the reference.
SSIM (↑)	0.899

Case 24



Metric	Result
LPIPS (↓)	0.295
Pixel difference (↓)	100.00%
LLM as a judge: Gemini 3.5 Flash, 5 runs (↑)	Scores: [0.65, 0.65, 0.65, 0.65, 0.65] Average: 0.6500 Range (Min - Max): [0.6500, 0.6500] The Actual UI introduces a ‘Skip’ button in the bottom left which is absent in the Reference UI. Additionally, the color theme for the active Page indicator dot and the ‘Done’ button has changed from green to purple.”
LLM as a judge: Gemini 3.1 Pro, 5 runs (↑)	Scores: [0.65, 0.6, 0.6, 0.6, 0.6] Average: 0.6100 Range (Min - Max): [0.6000, 0.6500] The Actual UI introduces an unexpected ‘Skip’ button on the bottom left. Additionally, the active pagination dot and ‘Done’ button text are purple instead of the expected green. There are also minor differences in text wrapping for the description.
LLM as a judge: Gemini 3 Flash Preview, 5 runs (↑)	Scores: [0.7, 0.7, 0.7, 0.7, 0.7] Average: 0.7000 Range (Min - Max): [0.7000, 0.7000] The Actual UI introduces a ‘Skip’ button not present in the Reference. Additionally, the primary theme color has shifted from green to purple for the ‘Done’ button and the active page indicator dot. While the core layout and illustration are preserved, these functional and stylistic changes constitute a mismatch.
SSIM (↑)	0.879