
From Prompts to Production

Building Android Apps with Skills and MCP Tools

EMMANUELE VILLA

Why agents need project guidance



Same prompt, twice

"Expose the APOD UI state from the ViewModel to the composable screen."

```
class ApodViewModel : ViewModel() {
    private val _uiState =
        MutableLiveData<ApodUiState>()
    val uiState: LiveData<ApodUiState> = _uiState

    fun load() {
        _uiState.value = ApodUiState.Loading
        // ...
    }
}

@Composable
fun ApodScreen(vm: ApodViewModel) {
    val state by vm.uiState
        .observeAsState(ApodUiState.Loading)
    ApodContent(state)
}
```

Run 1: `LiveData` + `observeAsState`: imperative, lifecycle handled implicitly.

```
class ApodViewModel : ViewModel() {
    private val _uiState = MutableStateFlow(
        ApodUiState.Loading as ApodUiState)
    val uiState: StateFlow<ApodUiState> =
        _uiState.stateIn(
            viewModelScope,
            SharingStarted.WhileSubscribed(5000),
            ApodUiState.Loading
        )
}

@Composable
fun ApodScreen(vm: ApodViewModel) {
    val state by vm.uiState
        .collectAsStateWithLifecycle()
    ApodContent(state)
}
```

Run 2: same prompt, same repo: `StateFlow` + `stateIn`, lifecycle-aware collection.

What the agent won't check on its own

- Whether it followed the team's conventions.
- Whether it used the current version of an API.
- Whether the code meets the project's security requirements.



What is APOD?

- NASA's Astronomy Picture of the Day: a free public API, one photo and caption per day since 1995.
- Stable, well-documented schema: good material to build an app against, live.





Today

- We start from an empty Compose project.
- We end with the APOD app on the emulator, a reviewed PR, and a RAM analysis.
- Every tool is added at the moment we need it.

1. Setup



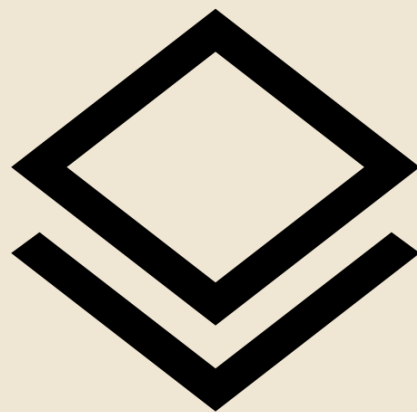
Prepared before the talk

- A GitHub repo and an empty Compose project from Android Studio.
- CLIs (`gh`, `android`), skills and MCP servers already installed and authenticated.



What's a skill?

- A folder with `SKILL.md`: name, description, instructions (and optional scripts).
- Auto-invoked when the task matches its description; can also be invoked explicitly by typing `/name`.
- Lives in the repo, reviewed like code; can also be disabled from model invocation (chapter 3).



compose-skill

```
class ApodViewModel : ViewModel() {
    var photo by mutableStateOf<Photo?>(null)
    var loading by mutableStateOf(false)

    fun load() {
        loading = true
        repository.fetch { result →
            photo = result
            loading = false
        }
    }
}

@Composable
fun ApodScreen(vm: ApodViewModel) {
    if (vm.loading) Loading()
    else vm.photo?.let { PhotoCard(it) }
}
```

Without the skill: mutable state exposed, fetch logic inline in the ViewModel.

```
sealed interface ApodIntent {
    data object Load : ApodIntent
}

class ApodViewModel : ViewModel() {
    private val _state =
        MutableStateFlow(ApodUiState())
    val state: StateFlow<ApodUiState> =
        _state.asStateFlow()

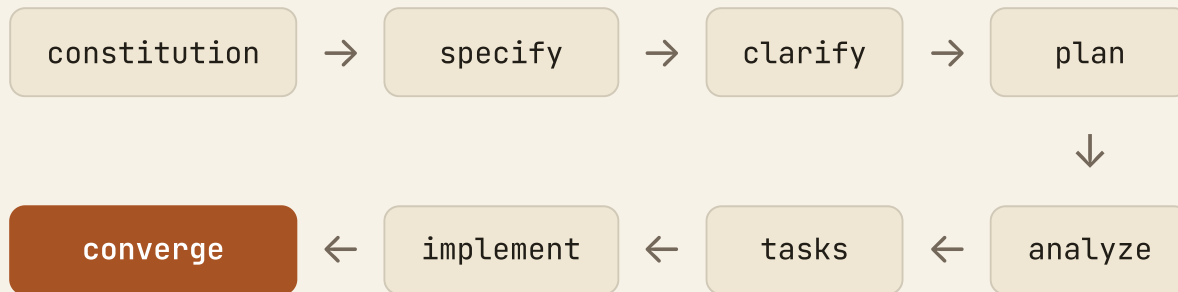
    fun onIntent(intent: ApodIntent) {
        when (intent) {
            is ApodIntent.Load → reduce()
        }
    }

    private fun reduce() { /* ... */ }
}
```

With the skill: StateFlow, intents, a reducer: one state holder per feature, MVI style.

What's **speckit**?

- GitHub's spec-kit: scaffolds a spec-driven workflow into a project.



Initialization

```
● ● ●  
$ specify init --here --integration claude  
$ claude  
> /speckit.constitution write the principles based  
  on the @.claude/skills/compose-skill best practices  
> /init
```

Full files shown live in Android Studio.

`constitution.md`

Written by `/speckit.constitution`:
the project's non-negotiable principles,
derived straight from `compose-skill`'s
best practices (architecture, state
management, testing, Material 3
theming), binding on every PR.

`CLAUDE.md`

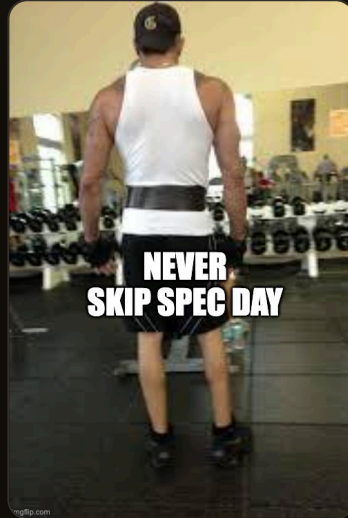
Written by `/init` from those principles:
project memory loaded in every session
(stack, commands, architecture rules,
spec-kit workflow).

Memory vs. skills

- `CLAUDE.md` is loaded in every session. Keep it short.
- Skills load only when the task matches their description.
- Both live in the repo and are reviewed like code.



2. The spec



Specify



```
> /speckit.specify  
APOD app, four screens:  
1. Today: image of the day, title, date,  
    short explanation, favorite button  
2. Detail: full text, credit, share  
3. Favorites: grid, empty state,  
    "local only" / "synced" indicator  
4. Login (optional): without it,  
    favorites stay on the device  
Fields: title, date, explanation,  
url/hdurl, media_type, copyright
```



```
150  ## Requirements *(mandatory)*
151
152  ### Functional Requirements
153
154  - **FR-001**: System MUST display, by default, the current day's astronomy picture: full-width
155    image (or video, per FR-002), title, date, and a 2-3 line preview of its explanation.
156  - **FR-002**: System MUST show a visible badge on the Today screen (and the Detail screen) when
157    the picture's `media_type` is "video" rather than an image.
158  - **FR-003**: Users MUST be able to expand the explanation preview on the Today screen to read the
159    full explanation in place.
160  - **FR-004**: Users MUST be able to mark the currently displayed picture as a favorite directly
161    from the Today screen.
162  - **FR-005**: Users MUST be able to open a Detail screen for the current picture showing its full
163    explanation, its credit/copyright when present, and a permalink to the original source.
164  - **FR-006**: Users MUST be able to save the picture to favorites and remove it from favorites
165    from the Detail screen, with the control reflecting the current saved state.
166  - **FR-007**: Users MUST be able to share the picture's title and permalink from the Detail screen
167    using the device's standard share mechanism.
168  - **FR-008**: System MUST provide a Favorites screen listing every picture the user has saved, each
169    shown with a thumbnail, title, and date, arranged as a grid.
```

User stories with priorities, and acceptance criteria for each screen.

3. Design



Live: design brief



Read the latest specs/*/spec.md.

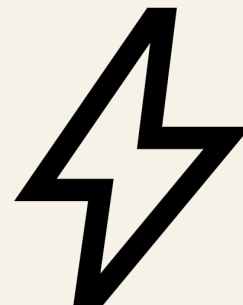
For every screen, define layout, content, and default/loading/empty/error states. Write design-brief.md. No code.

Avoid:

- Purple-blue gradients
- Inter-only typography
- Nested icon-heading cards
- Floating icon pills
- Generic shadows and spacing
- Generic dashboards, decorative cards, nested icon-heading cards, floating icon pills, purple-blue gradients, Inter-only typography, arbitrary shadows, and uniform spacing.

Design direction:

- Dark surfaces (#0B0D10), copper primary
- Serif display, monospace labels
- Asymmetric composition, hairline dividers
- Explicit pressed, saved, empty, and loading states



design-brief.md

```
# Design Brief: AP0D App – "Observatory Logbook"
```

```
**Source spec**: \[spec.md\]\(./spec.md\) (`001-apod-app`)
```

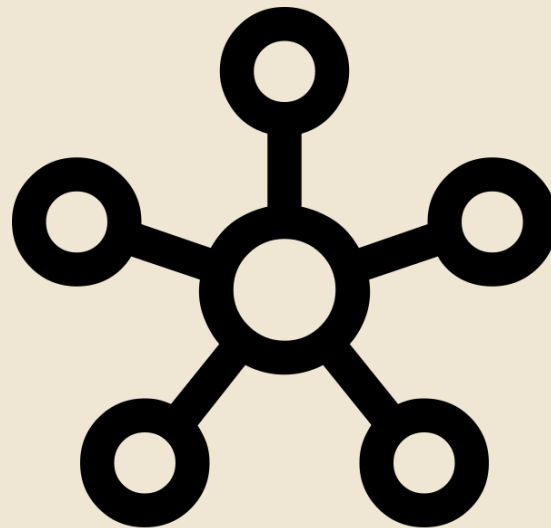
```
**Design direction source**: `.specify/memory/design-direction.md` was not found in this repository. This brief instead uses the direction supplied inline with this command, which it treats as the canonical "observatory logbook" direction until that memory file exists. Consider persisting the "Design Language" section below into `.specify/memory/design-direction.md` so future briefs don't have to restate it.
```

```
No code, no component library choices – this is layout, content, and state intent only.
```

Per-screen layout, content and states, following the design direction; still no code.

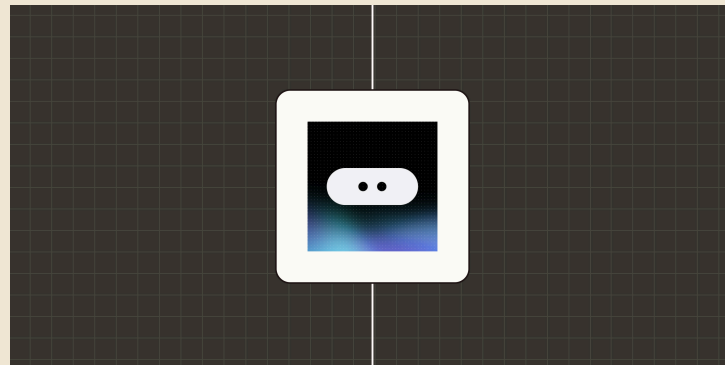
What's an MCP?

- Model Context Protocol: an external server (or not) that exposes tools/APIs an agent can call.
- Added per-project, with explicit authorization.
- Powers Context7, Firebase, GitHub and Stitch, all coming up in this chapter.



What is Stitch?

- Google Labs' AI UI designer: a text brief in, screens out.
- Generates visual designs, not application code.
- Also runs as an MCP server, so the agent can fetch the generated screens directly.

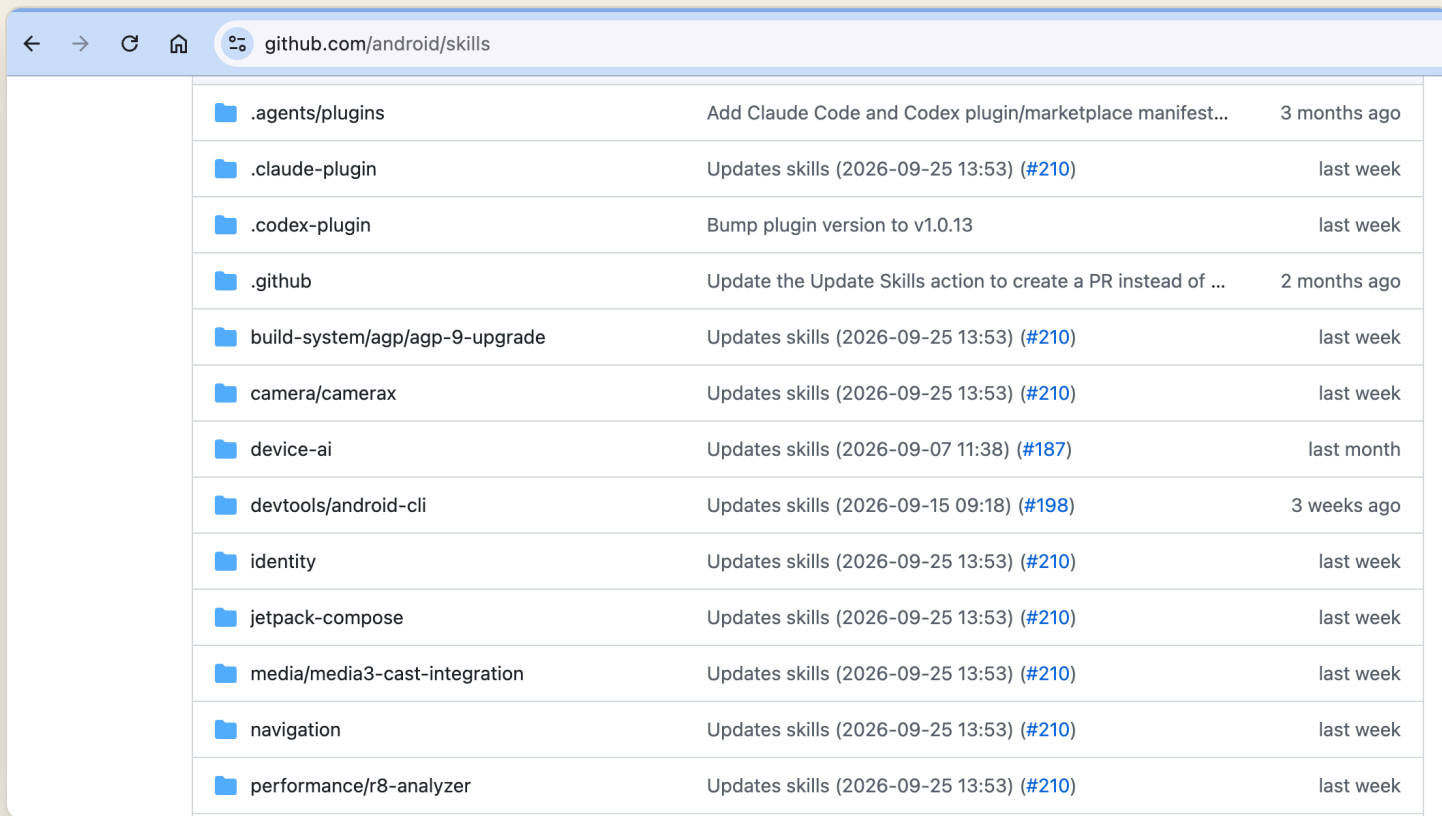


While Stitch generates

Stitch takes a few minutes. Meanwhile, the rest of the toolkit.



android/skills



← → ↻ 🏠	github.com/android/skills		
📁	.agents/plugins	Add Claude Code and Codex plugin/marketplace manifest...	3 months ago
📁	.claude-plugin	Updates skills (2026-09-25 13:53) (#210)	last week
📁	.codex-plugin	Bump plugin version to v1.0.13	last week
📁	.github	Update the Update Skills action to create a PR instead of ...	2 months ago
📁	build-system/agp/agp-9-upgrade	Updates skills (2026-09-25 13:53) (#210)	last week
📁	camera/camerax	Updates skills (2026-09-25 13:53) (#210)	last week
📁	device-ai	Updates skills (2026-09-07 11:38) (#187)	last month
📁	devtools/android-cli	Updates skills (2026-09-15 09:18) (#198)	3 weeks ago
📁	identity	Updates skills (2026-09-25 13:53) (#210)	last week
📁	jetpack-compose	Updates skills (2026-09-25 13:53) (#210)	last week
📁	media/media3-cast-integration	Updates skills (2026-09-25 13:53) (#210)	last week
📁	navigation	Updates skills (2026-09-25 13:53) (#210)	last week
📁	performance/r8-analyzer	Updates skills (2026-09-25 13:53) (#210)	last week

Official skills from Google, installed with `android skills add --all`: Compose, security, testing, profiler, `android` CLI...

mobile-security-skills (dweinstein)

Files

main

Go to file

skills

auth-assessment

code-quality-scan

crypto-review

masvs-checklist

mobile-pentest-plan

mobile-threat-model

network-security-check

platform-interaction-review

privacy-audit

resilience-assessment

secure-mobile-dev-guide

secure-storage-audit

README.md

mobile-security-skills / skills

Add file

...

dweinstein

Add some reverse-engineering tool guidance

cb7ce51 · 7 months ago

History

Name	Last commit message	Last commit date
..		
auth-assessment	Add some reverse-engineering tool guidance	7 months ago
code-quality-scan	Tighten stale and overly absolute skill guidance	7 months ago
crypto-review	Initial commit: MASTG skills collection	10 months ago
masvs-checklist	Tighten stale and overly absolute skill guidance	7 months ago
mobile-pentest-plan	Add some reverse-engineering tool guidance	7 months ago
mobile-threat-model	Initial commit: MASTG skills collection	10 months ago
network-security-check	Add some reverse-engineering tool guidance	7 months ago
platform-interaction-review	Tighten stale and overly absolute skill guidance	7 months ago
privacy-audit	Initial commit: MASTG skills collection	10 months ago
resilience-assessment	Add some reverse-engineering tool guidance	7 months ago
secure-mobile-dev-guide	Tighten stale and overly absolute skill guidance	7 months ago
secure-storage-audit	Tighten stale and overly absolute skill guidance	7 months ago

secure-storage-audit on a rehearsal build. One skill per MASVS area; you run only the check you need.

Skills only you can run

```
---  
name: masvs-checklist  
disable-model-invocation: true  
---
```

3 of the 12 security skills are set this way by their author: Claude never starts them on its own, only as slash commands.



Context7

```

// Coil 1.x (from training data)
Image(
    painter = rememberImagePainter(
        data = photo.url,
        builder = { crossfade(true) }
    ),
    contentDescription = photo.title
)
```

Without Context7: `rememberImagePainter`, removed in Coil 3.

```

// Coil 3.x (verified via Context7)
AsyncImage(
    model = ImageRequest.Builder(
        LocalContext.current)
        .data(photo.url)
        .crossfade(true)
        .build(),
    contentDescription = photo.title
)
```

With Context7: `AsyncImage`, the current API from today's docs.



Firestore MCP

- Official. Reads project config, security rules and SDK config.
- Can also create a project and register the app. We use that in chapter 5.
- Can also pull Crashlytics data, e.g. the top 10 most frequent crashes.

GitHub: MCP or CLI?



```
/context

System prompt      3.2k tokens
System tools       4.1k tokens
MCP: github        11.4k tokens
  (26 tools)
Memory files       1.8k tokens
Messages           0 tokens
-----
Free space         180k (90%)
```

`/context` with the GitHub MCP: its tool definitions take space in every session.



```
/context

System prompt      3.2k tokens
System tools       4.1k tokens
MCP                none
Memory files       1.8k tokens
  gh-cli skill: not loaded yet
Messages           0 tokens
-----
Free space         191k (95%)
```

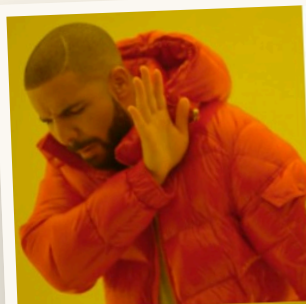
`/context` with `gh` and the `gh-cli` skill: it loads only when used.

Trade-off: `gh auth login` is usually broad, account-wide access, not scoped per repo like a fine-grained MCP token.

If you use the GitHub MCP, scope it

```
"github": {  
  "type": "http",  
  "url": "https://api.githubcopilot.com/mcp/",  
  "headers": {  
    "Authorization": "Bearer <PAT>",  
    "X-MCP-Toolsets": "repos,pull_requests"  
  }  
}
```

Fine-grained token for one repo, only the toolsets you need,
`X-MCP-Readonly` when it only has to read.

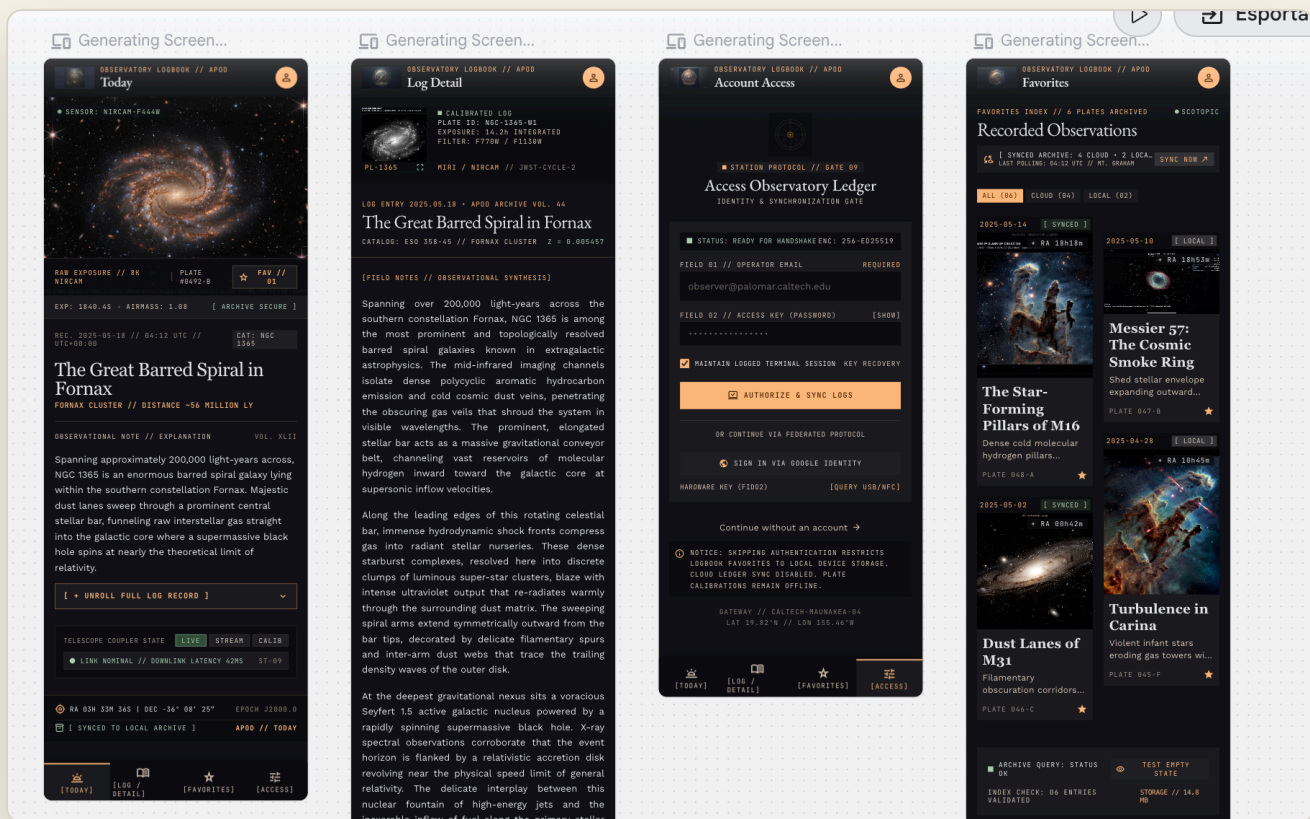


a PAT
with full
repo access



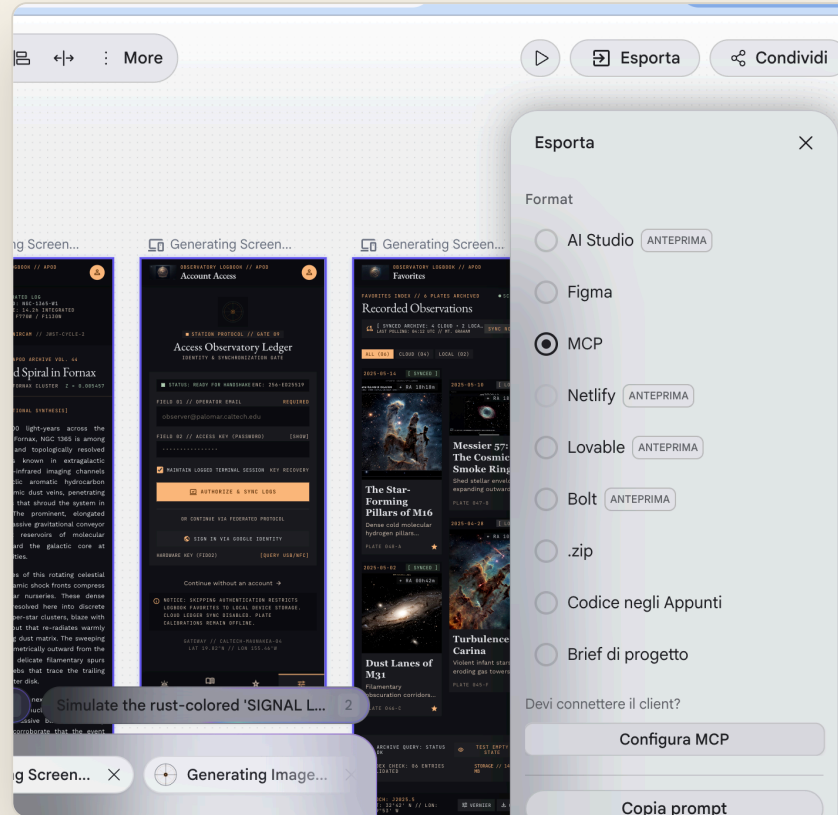
a fine-grained
token scoped
to one repo
and one toolset

What Stitch generated



Four screens from `design-brief.md`.

Installing the Stitch MCP



Added once per project; after this the agent can fetch the generated screens directly.

Does Stitch know your Compose components?

- No. It returns generic HTML/CSS, not your existing components (Figma with Code Connect does).
- HTML → Compose is not a trivial conversion; the generated scaffolding needs cleanup.
- No dark/light theme handling either: that information isn't in Stitch's output.
- Here we use it for structure; the visual pass comes in chapter 7.



4. Plan and tasks



Hooks vs. instructions

- The model can skip an instruction in a skill.
- A hook runs on every matching event: lint, format, or block (exit code 2).
- The output goes back to the agent, which fixes the problem.
- `PostToolUse` fires on every matching tool call; `Stop` fires once, when the agent believes it is done.



While it plans: a lint hook

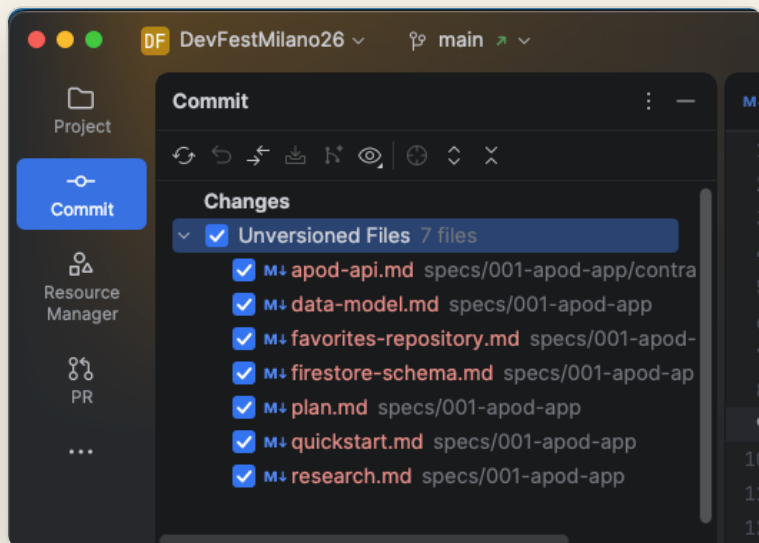
```

// .claude/settings.json
"hooks": {
  "Stop": [{
    "hooks": [{ "type": "command",
      "command": "./gradlew ktlintCheck -q" }]
  }]
}
```

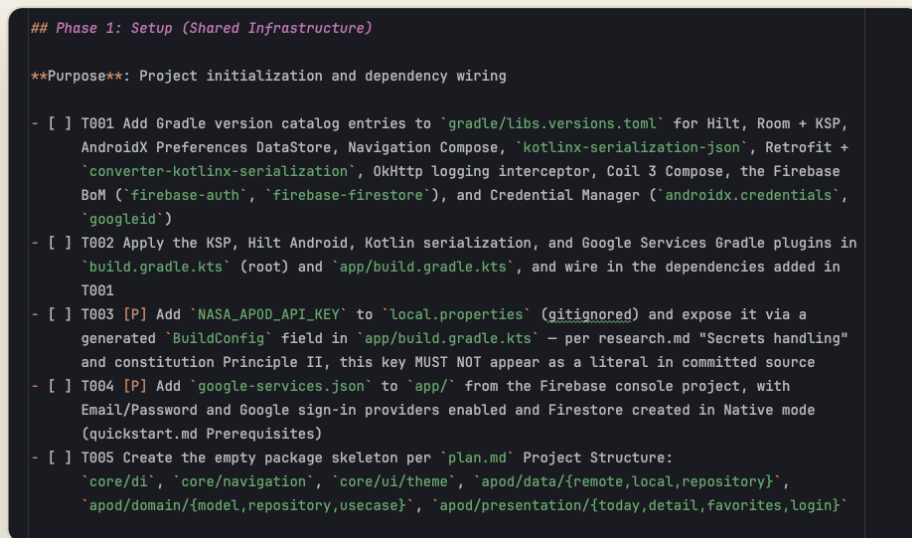
No **matcher**: **Stop** is not per-tool, it fires once when the agent is about to end its turn. Exit code 2 blocks the stop and feeds the failure back to the agent.



plan.md and tasks.md



Firebase, MVI, dependencies: the technical choices the spec deliberately left out.



Tasks grouped by user story, MVP first, each one naming the skill or MCP it uses.

5. Build



On a separate branch

```
• • •  
$ git checkout -b apod-mvp  
$ claude  
> /speckit.implement
```

A clean branch for the feature. If the run goes wrong, check out `main` and discard it.



What **implement** does



- | | |
|---------------------|----------------------|
| 1. Fetch screens | Stitch MCP |
| 2. Compose + MVI | compose-skill |
| 3. Firebase project | Firebase MCP |
| 4. Login + sync | security skills |
| 5. Unit tests | testing-setup |
| 6. Storage audit | secure-storage-audit |
| 7. API check | Context7 |
| 8. Open the PR | gh / GitHub MCP |



Creating the Firebase project

```
15 +NASA_APOD_API_KEY=DEMO_KEY
16 No newline at end of file

● Write(app/google-services.json)
  └ Wrote 29 lines to app/google-services.json
    1 {
    2   "project_info": {
    3     "project_number": "000000000000",
    4     "project_id": "devfestmilano26-placeholder",
    5     "storage_bucket": "devfestmilano26-placeholder.appspot.com"
    6   },
    7   "client": [
    8     {
    9       "client_info": {
   10         "mobilesdk_app_id": "1:000000000000:android:0000000000000000000000",
    ... +19 lines

  Read 1 file

● Update(.gitignore)
```

Step 3 asked before creating anything real: mock the SDK config instead? Said yes. No billing/organization selection to click through live.

Running the tests

Ran 2 shell commands

- Trovato un dispositivo fisico collegato via adb. Posso quindi eseguire davvero gli instrumented test e installare l'app per la validazione end-to-end (T059).

Running instrumented tests (Room DAO + Compose UI) on the connected device · 17s

└ \$./gradlew :app:connectedDebugAndroidTest --no-daemon 2>&1 | tail -200 (17s)
(ctrl+b to run in background)

· Hashing... (54m 31s · ↓ 201.4k tokens)

└ Tip: Use /clear to start fresh when switching topics and free up context

>

▶▶ auto mode on (shift+tab to cycle) · esc to interrupt · ← 1 agent

Thanks to the skills, Claude sees a device connected over `adb` and runs the tests on it autonomously.

How long did it take?

Cosa resta da fare prima che sia "reale"

- `app/google-services.json` è un placeholder (valori finti, gitignored) – avevi scelto questa opzione per non b con un progetto Firebase vero (e deployi `firestore.rules`).
- `NASA_APOD_API_KEY=DEMO_KEY` in `local.properties` – sostituiscila con una chiave vera da `api.nasa.gov` per evita
- `GOOGLE_WEB_CLIENT_ID` in `local.properties` è vuoto – va preso dal `google-services.json` reale.

24 file toccati (14 modificati, 10 nuovi gruppi). Non ho fatto commit – fammi sapere se vuoi che proceda.

* Cooked for 1h 0m 3s · done 3:48 PM

The full `/speckit.implement` run, start to finish: screens, Compose + MVI, Firebase, tests, security audit, API check.

The pull request

Implement APOD app: Today, Detail, Favorites, Login #1

 **Open** [EmmanueleVilla](#) wants to merge 9 commits into `main` from `001-apod-app`

Conversation 1

Commits 9

Checks 0

Files changed 193

+16.465

 **EmmanueleVilla** commented [now](#) Owner

Summary

- Full MVI + clean architecture implementation of all 60 tasks in `specs/001-apod-app/tasks.md`: Hilt DI, Room (favorites + today cache), DataStore session cache, Retrofit NASA APOD client, Coil image loading, and Navigation Compose with the persistent 4-tab bottom nav (Today / Log:Detail / Favorites / Access) matching the project's Stitch mockups.
- Firebase Auth (email/password + Google via Credential Manager) with Firestore sync: a live snapshot listener mirrors remote favorites into Room while signed in, and logging in migrates existing local-only favorites to the account (FR-015).
- Unit tests for every domain use case, repository, and ViewModel, plus instrumented Room + Compose UI tests — all passing, including a real run on a connected device (17/17 instrumented tests, manual walkthrough of loading/error/empty states and the login flow).
- Security hardening from an in-session MASVS review (`auth-assessment` , `secure-storage-audit` , `network-security-check`): redacts the NASA API key from debug HTTP logs, excludes the favorites DB and session DataStore from Android auto-backup, adds `firestore.rules` enforcing per-user access server-side, and an explicit `network_security_config.xml` blocking cleartext traffic.
- Mid-implementation correction: downloaded the project's actual Stitch screens and switched the nav from a "login-gate" assumption to the persistent bottom-nav bar they actually show, updating `research.md` / `tasks.md` accordingly.

Reviewers

 **Copilot** Balanced

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

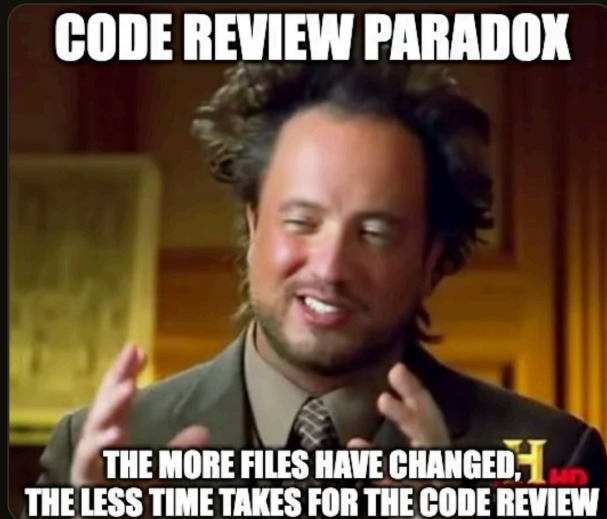
No milestone

Development

Successfully merging this pull request may close these issues.

The PR created by `/speckit.implement`.

6. Review

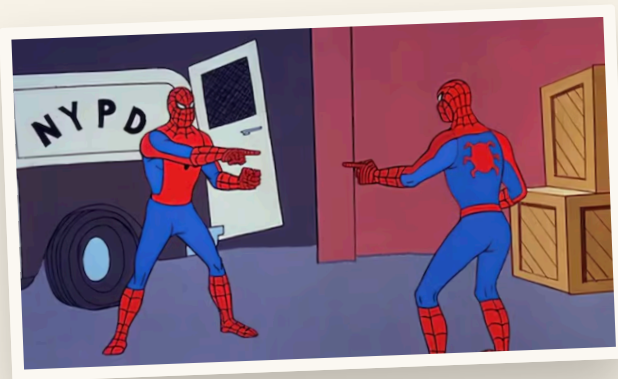


`/clear`, then review



```
> /clear  
> /code-review high --comment <PR#>
```

The session that wrote the code has its own reasoning in context and tends to agree with it. After `/clear` the reviewer sees only the diff, `CLAUDE.md` and an optional `REVIEW.md`.



Review comments

 **EmmanueleVilla** commented [now](#) [View reviewed changes](#)

✓

app/src/main/java/com/shadowings/devfestmilano26/apod/presentation/today/TodayViewModel.kt

55

+

favoriteStatus = null,

56

+

isExplanationExpanded = false,

57

+

)

58

+

observeFavoriteStatus(picture.date)

 **EmmanueleVilla** [now](#) Owner Author ...

Coroutine leak: every `load()` call stacks a new favorite-status collector. `load()` runs on `init` and again on every `TodayIntent.Retry`. Each invocation starts a fresh `observeFavoriteStatus(picture.date).onEach { ... }.launchIn(viewModelScope)` collector, but no reference to the previous `Job` is kept or cancelled. Repeated Retry taps (e.g. a user retrying on a flaky connection) leave N live collectors all writing `_state.value = current.copy(favoriteStatus = status)` for the `ViewModel`'s lifetime; a late emission from an older, stale collector can overwrite state produced by a more recent `load()`. Consider cancelling the prior favorite-status job before launching a new one, or deriving the whole `TodayState.Content` from a single `flatMapLatest`-based pipeline (as a sibling fix would also apply to `DetailViewModel`).



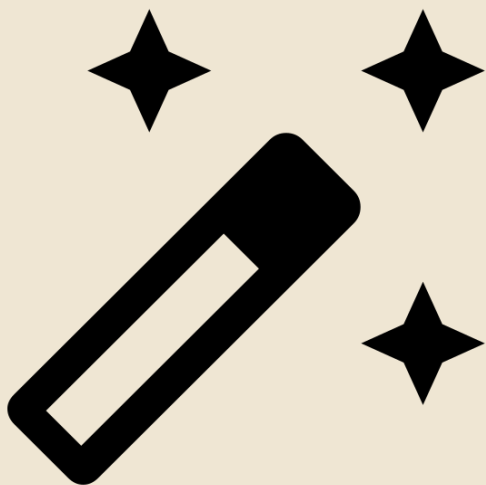
 Reply...

Resolve conversation

Inline comments on the PR. Nothing is approved or blocked; `--fix` applies them as a new commit.

7. Polish





impeccable (Paul Bakaus)

- A design skill with an Android reference based on Material 3.
- On Compose it reviews the source; its 61-rule detector only works on HTML.
- `/impeccable polish`: states, 48 dp targets, insets, system Back, color roles, type scale.

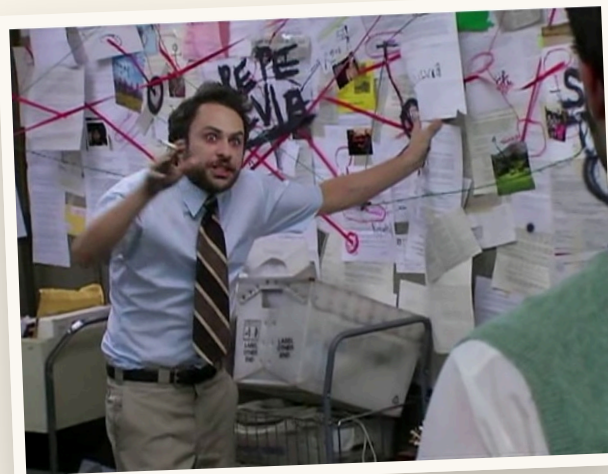
8. Run and profile



Two more skills from android/skills



1. Run the app on the emulator
(android-cli: android run)
2. Open Favorites, save 3 items,
go back and forth to Detail
3. Take a RAM dump
(android-profiler, Perfetto)
4. Is any Activity or ViewModel
retained? What fills the RAM?



RAM analysis

Memory profile — APOD app

Pixel 10a (Android 17) · com.shadowings.devfestmilano26 · October 5, 2026

PSS AT LAUNCH

197.0 MB

Today, first frame

PEAK PSS

227.6 MB

after login + 3 tabs

SESSION GROWTH

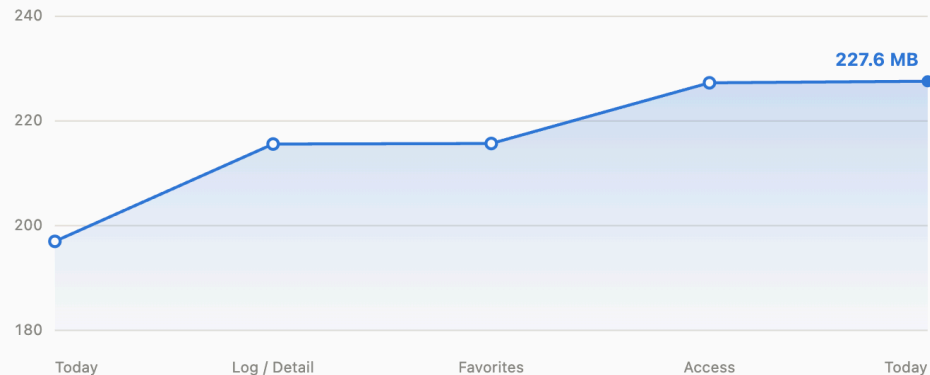
+30.6 MB

+15.5% over launch

PSS while navigating between sections

[View as table](#)

dumpsys meminfo · TOTAL PSS in MB, sampled on every tab switch



RAM analysis

The finished app



APOD



Screenshot after some polishing and fixing (e.g. dark theme).

What the project has now

Memory

`CLAUDE.md`, constitution

Skills

android/skills (incl. `android-cli`, `android-profiler`), compose-skill, mobile-security-skills, `gh-cli`, impeccable

CLIs

`specify`, `gh`, `android`, Perfetto

MCPs

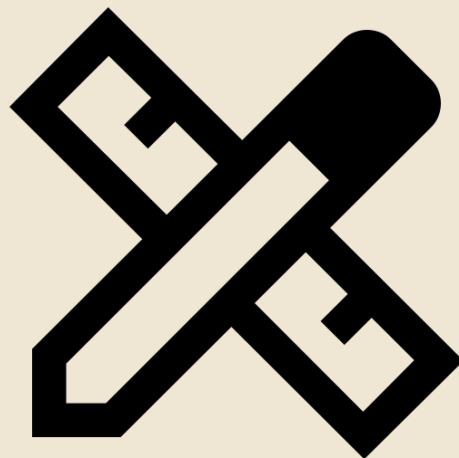
Stitch, Firebase, Context7

Hooks

ktlint after every edit

What worked, what didn't

- Separate steps gave me a point to review at each stage.
- Stitch output needed a design pass even with a detailed direction.
- On real projects I'd also run `clarify`, `checklist`, `analyze`.
- Worth it for new, sizable features; for small changes to existing code, just prompt directly.





Limits

- Skills need maintenance as libraries change.
- Every MCP adds attack surface: least-privilege tokens, minimal toolsets.
- Destructive actions should always require confirmation.
- A person still reviews and merges.



Sharing the setup

- Commit `CLAUDE.md`, `.claude/` (skills, hooks) and `.mcp.json`.
- Or package it as a plugin in a team marketplace.
- Everyone on the team then works with the same configuration.

Thank You & Questions

EMMANUELEVILLA.COM

What we covered

- 01: Project memory and a written spec
- 02: Skills, CLIs and MCPs, added when needed
- 03: Hooks, tests and review as checks

Questions?

