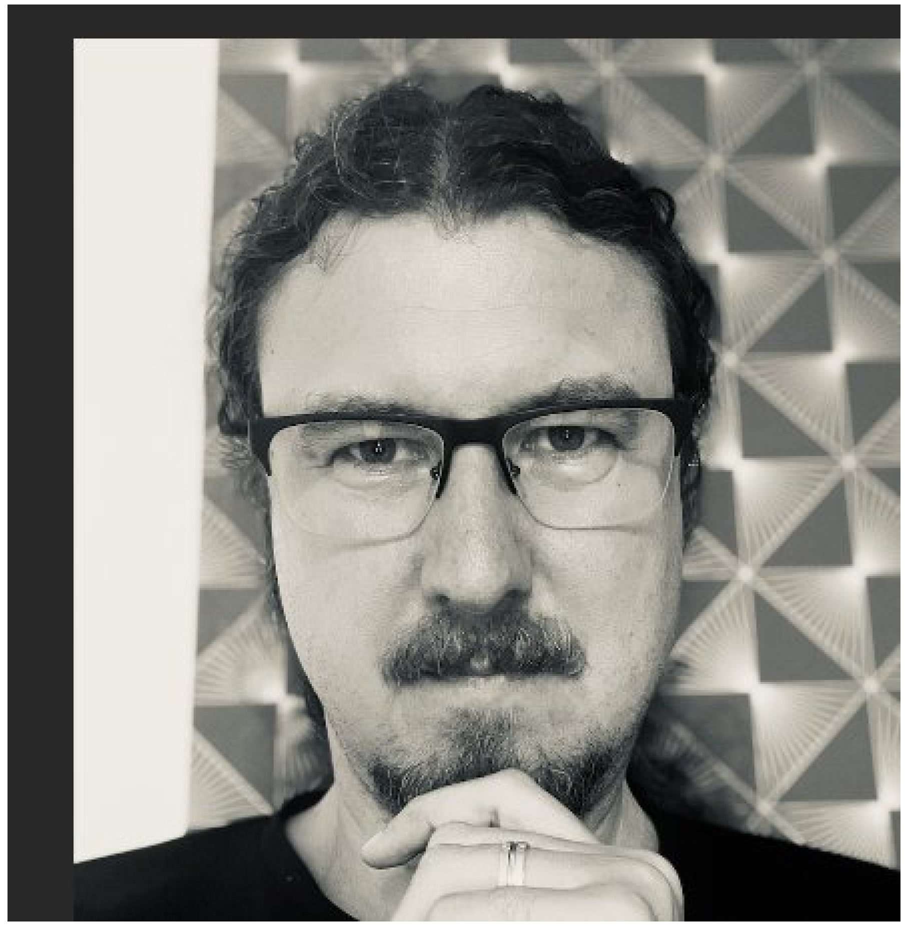
CONFIDENTIAL

OLAF
Open Lightweight Assistant Framework

by: Dorin Pomian
Alae Tamouh



Dorin Pomian

- Senior Software Engineer at **Amadeus**, driven by a passion for pushing the boundaries of refactoring tools and techniques
- Passionate about exploring and building Generative AI solutions for software engineers
- Favorite quote: "Two possibilities exist: either we are alone in the Universe or we are not. Both are equally terrifying - Arthur C. Clarke"

Alae Tamouh

- Software Engineer at **Amadeus**, focused on building tools and practices that help engineering teams work more efficiently
- Curious about all things computer science, from distributed systems to developer tools
- Passionate about Generative AI—how it can cut through repetitive work and give developers more room to focus on what really matters

CONFIDENTIAL

Agenda

- Introduction
  - context
  - problem space
- OLAF
  - Presenting key features
    - Structured prompts
    - Share and reuse prompts
    - Context rot
    - Meta-prompting
    - Interaction protocols
    - Targets various profiles (devs, qa, pdef…)
  - Demo / Use case walkthrough
  - Lessons & Best practices
    - Include Shortcomings & future evolutions
- Q&A

CONFIDENTIAL

How We Use AI Agents Today

- In today’s Generative AI landscape, many technical tasks can be delegated to agents such as GitHub Copilot, Windsurf Cascade, or Claude Code
- These agents act as a bridge between us as users and Large Language Models (LLMs)
- To increase answer relevancy and accuracy, agents enrich user prompts with context from memories, tools, MCP servers, and databases
- The agent also transforms raw LLM output back into something useful for the user (code edits, summaries, reports)
- This iterative loop of *prompt* → *answer* → *react* → *re-prompt* is powerful, but hard to manage in practice

CONFIDENTIAL

The Core Problem: Prompting Is Not a Process

- **Prompting ≠ Workflow**

- **Prompts are fragile**
  - Tiny wording changes → very different answers
  - “Good prompts” live in someone’s chat history, not in a repo
- **Difficult to be clear and complete**
  - We struggle to express intent, constraints, and edge cases concisely
  - Long prompts become unmaintainable walls of text
- **No repeatability**
  - A prompt that worked yesterday may not work tomorrow
  - Different people get different results for the “same” task

- Prompting alone doesn’t give us a reliable engineering process

CONFIDENTIAL

Why This Hurts Teams

- **Engineering work is process-driven**
  - Code review, testing, migrations, audits, documentation
- **We want consistent outcomes**
  - Same task → similar structure, level of detail, and checks
- **But with LLMs today**
  - Every engineer “freestyles” their own prompt
  - No review, versioning, or testing of prompts
  - Difficult to reuse across teams, projects, or models

- Teams don’t just need *better prompts*; they need **reliable AI workflows and consistent results**

CONFIDENTIAL

What is OLAF?

- **OLAF** is an **Open, adaptable, Lightweight prompting Assistant and Framework**
  - It helps you turn *ad-hoc prompting* into **structured, repeatable workflows**
- It is **model-agnostic** and **agent-agnostic**:
  - Works with different LLMs and coding assistants
- It focuses on **human–AI collaboration**:
  - Specifies how your team wants AI to work, step by step
  - Designed to streamline the creation, sharing, and execution of complex prompts, ensuring consistent behavior across different models and agents

- **Write your prompts once, use them everywhere!**

CONFIDENTIAL

OLAF Mental Model

- **Inputs**
  - Task description, constraints, project configuration
  - Context: codebase, documentation, tickets, memories
- **Workflow**
  - In OLAF terminology workflow is referred to as a competency
  - Sequence of **steps** (prompt + prompt templates + tools + data)
  - Each step has clear inputs/outputs and responsibilities
- **Outputs**
  - Structured artifacts: reports, diff analyses, refactor plans, PR comments, checklists, logs, prompts, etc

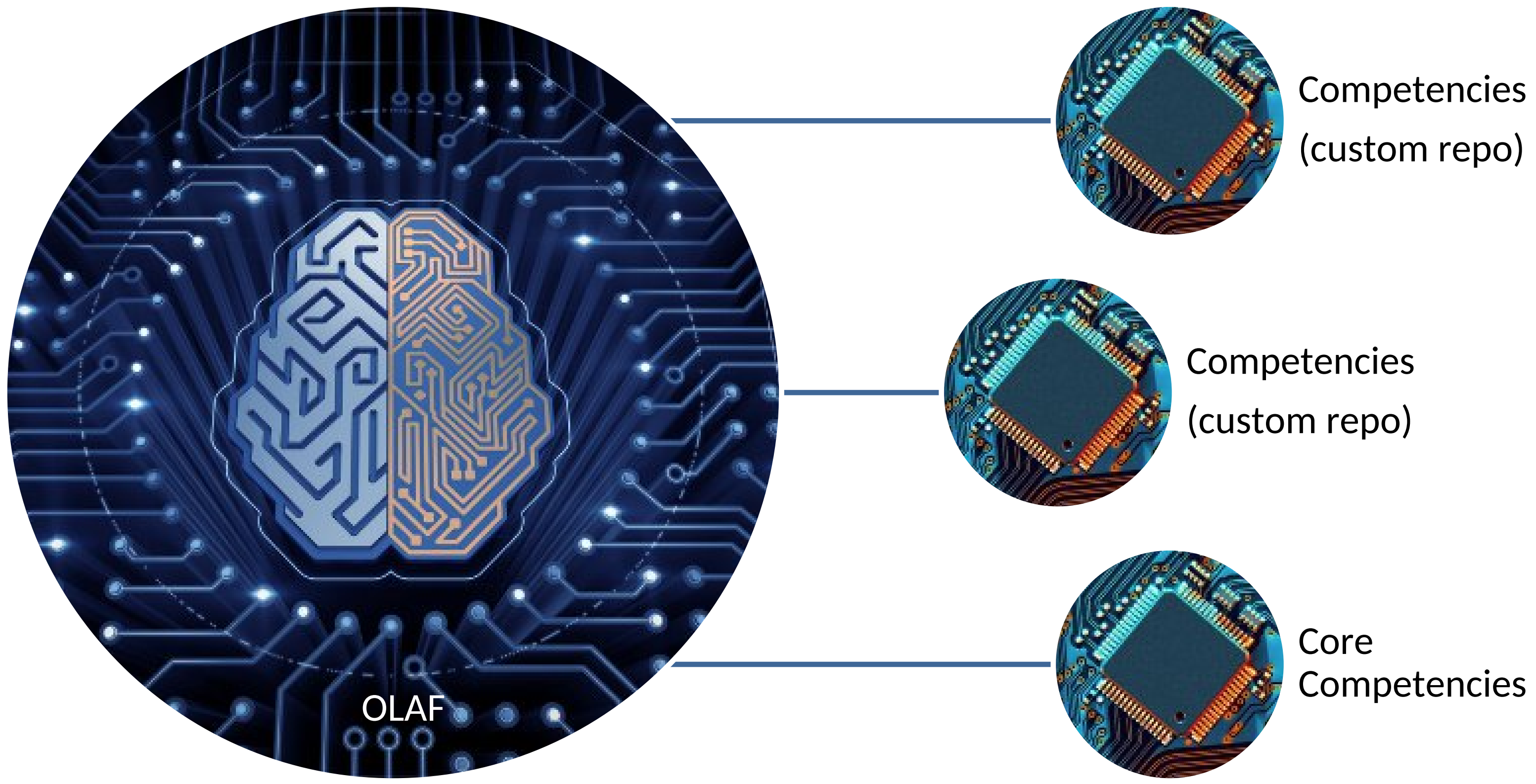
CONFIDENTIAL

OLAF
Architecture Overview

CONFIDENTIAL

OLAF High-Level Architecture

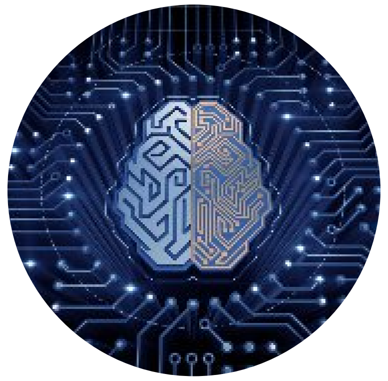
- Competencies
  - Specialized units for achieving a certain goal (e.g. *code review*, *generate technical documentation create prompts, etc.)*
  - Concrete set of prompts, templates, tools, and data needed for completing a certain task
  - Built-in competencies (e.g. *create prompt)*
  - Competencies from external repos
- OLAF framework
  - The “engine” that allows using a competency
  - Bootstrap for various IDEs (VSCode, Windsurf, Kiro)



CONFIDENTIAL

OLAF High-Level Architecture

- OLAF bootstrap and framework

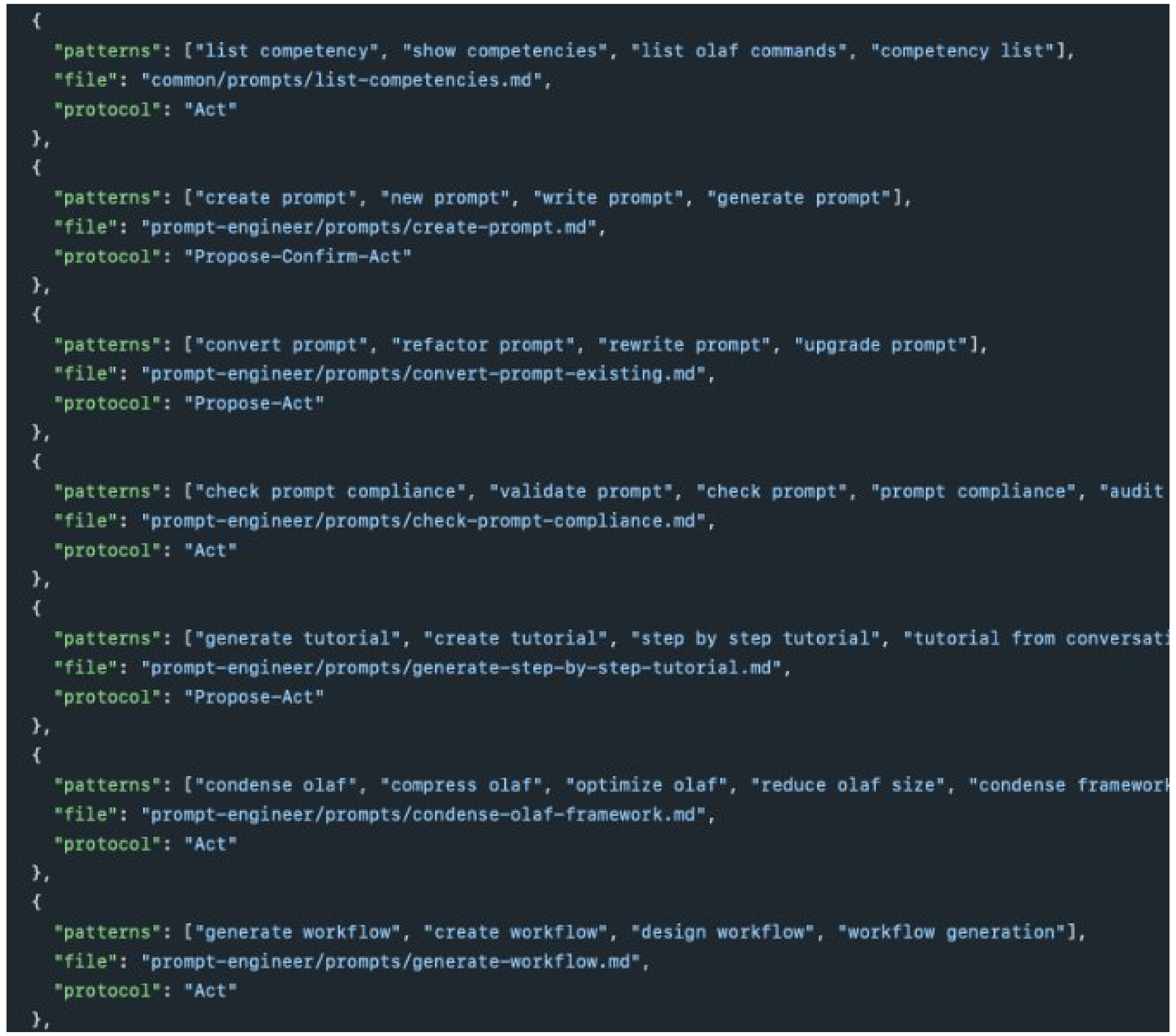


OLAF

- **Bootstrap**
  - IDE specific ”entry points” that instructs the coding agent when and how to load OLAF framework
    - VSCode *.github/copilot-instructions.md*
    - Windsurf - *.windsurf/rules/olaf-bootstrap.md*

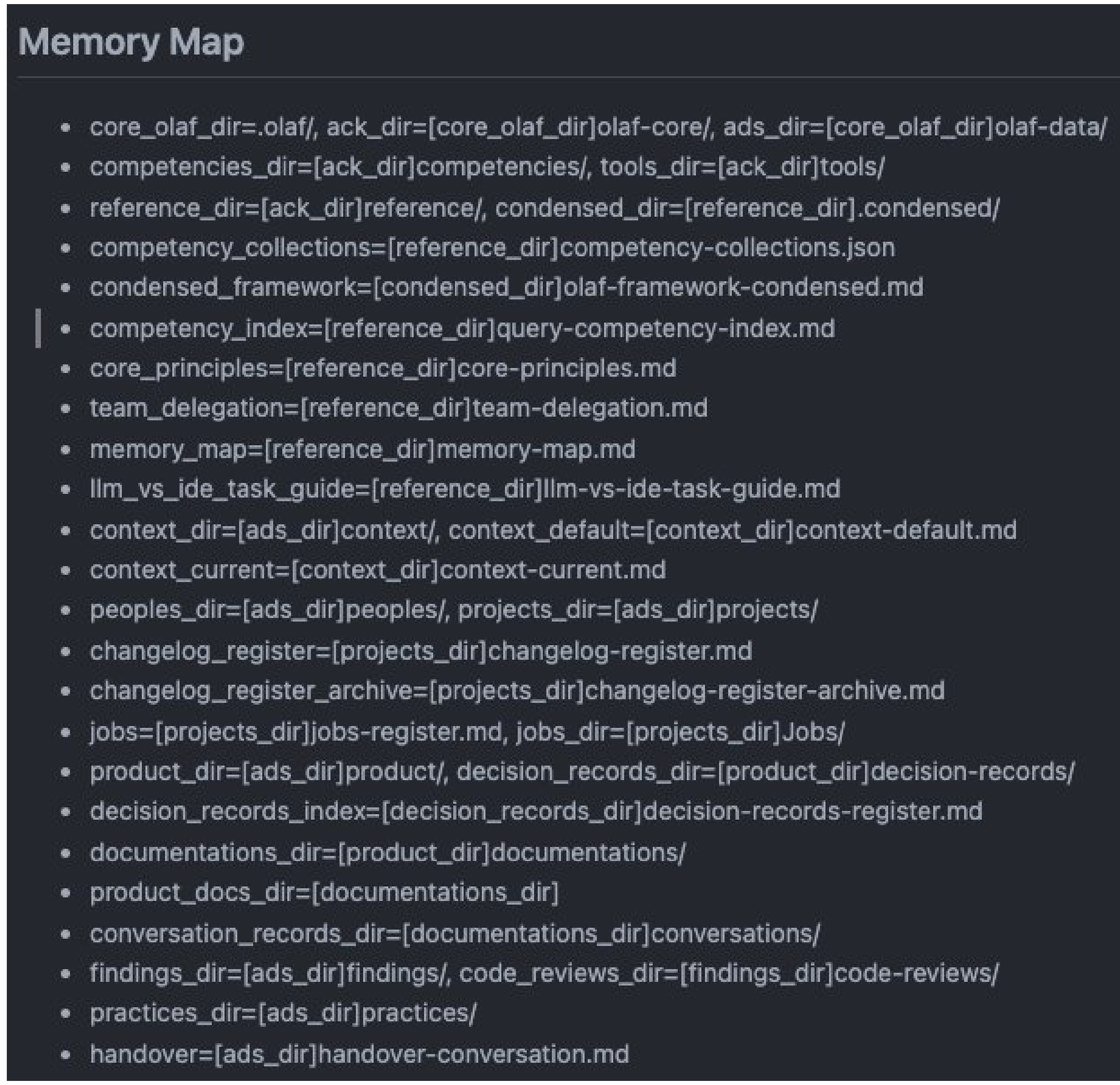
OLAF High-Level Architecture

- **OLAF framework**
    - A set of md files that represent the internal processes of OLAF
    - Describes how to run a competency
    - Describes the interaction protocols
    - Defines memory map
    - Naming and formatting conventions
    - Initial set of instructions



```
{
  "patterns": ["list competency", "show competencies", "list olaf commands", "competency list"],
  "file": "common/prompts/list-competencies.md",
  "protocol": "Act"
},
{
  "patterns": ["create prompt", "new prompt", "write prompt", "generate prompt"],
  "file": "prompt-engineer/prompts/create-prompt.md",
  "protocol": "Propose-Confirm-Act"
},
{
  "patterns": ["convert prompt", "refactor prompt", "rewrite prompt", "upgrade prompt"],
  "file": "prompt-engineer/prompts/convert-prompt-existing.md",
  "protocol": "Propose-Act"
},
{
  "patterns": ["check prompt compliance", "validate prompt", "check prompt", "prompt compliance", "audit
  "file": "prompt-engineer/prompts/check-prompt-compliance.md",
  "protocol": "Act"
},
{
  "patterns": ["generate tutorial", "create tutorial", "step by step tutorial", "tutorial from conversati
  "file": "prompt-engineer/prompts/generate-step-by-step-tutorial.md",
  "protocol": "Propose-Act"
},
{
  "patterns": ["condense olaf", "compress olaf", "optimize olaf", "reduce olaf size", "condense framework
  "file": "prompt-engineer/prompts/condense-olaf-framework.md",
  "protocol": "Act"
},
{
  "patterns": ["generate workflow", "create workflow", "design workflow", "workflow generation"],
  "file": "prompt-engineer/prompts/generate-workflow.md",
  "protocol": "Act"
},
```

OLAF Competency Index



Memory Map

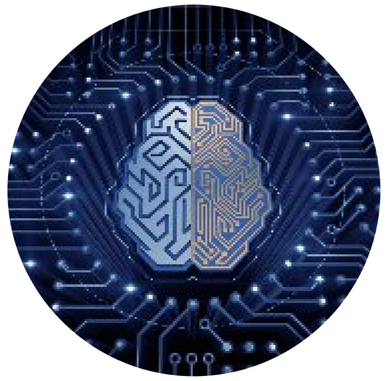
- core_olaf_dir=.olaf/, ack_dir=[core_olaf_dir]olaf-core/, ads_dir=[core_olaf_dir]olaf-data/
- competencies_dir=[ack_dir]competencies/, tools_dir=[ack_dir]tools/
- reference_dir=[ack_dir]reference/, condensed_dir=[reference_dir].condensed/
- competency_collections=[reference_dir]competency-collections.json
- condensed_framework=[condensed_dir]olaf-framework-condensed.md
- competency_index=[reference_dir]query-competency-index.md
- core_principles=[reference_dir]core-principles.md
- team_delegation=[reference_dir]team-delegation.md
- memory_map=[reference_dir]memory-map.md
- llm_vs_ide_task_guide=[reference_dir]llm-vs-ide-task-guide.md
- context_dir=[ads_dir]context/, context_default=[context_dir]context-default.md
- context_current=[context_dir]context-current.md
- peoples_dir=[ads_dir]peoples/, projects_dir=[ads_dir]projects/
- changelog_register=[projects_dir]changelog-register.md
- changelog_register_archive=[projects_dir]changelog-register-archive.md
- jobs=[projects_dir]jobs-register.md, jobs_dir=[projects_dir]Jobs/
- product_dir=[ads_dir]product/, decision_records_dir=[product_dir]decision-records/
- decision_records_index=[decision_records_dir]decision-records-register.md
- documentations_dir=[product_dir]documentations/
- product_docs_dir=[documentations_dir]
- conversation_records_dir=[documentations_dir]conversations/
- findings_dir=[ads_dir]findings/, code_reviews_dir=[findings_dir]code-reviews/
- practices_dir=[ads_dir]practices/
- handover=[ads_dir]handover-conversation.md

OLAF Memory Map

CONFIDENTIAL

OLAF High-Level Architecture

- OLAF Interaction protocol



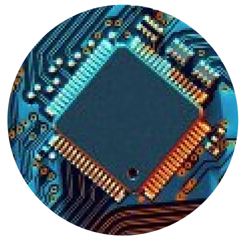
OLAF

- **Act**
  - The agent is encouraged to carry out an action without any user confirmation
- **Propose-Act**
  - The agent states the action it’s about to perform, then performs the action itself
- **Propose-Confirm-Act**
  - The agent is encouraged that for any action that alters files to propose the action and ask for user confirmation

CONFIDENTIAL

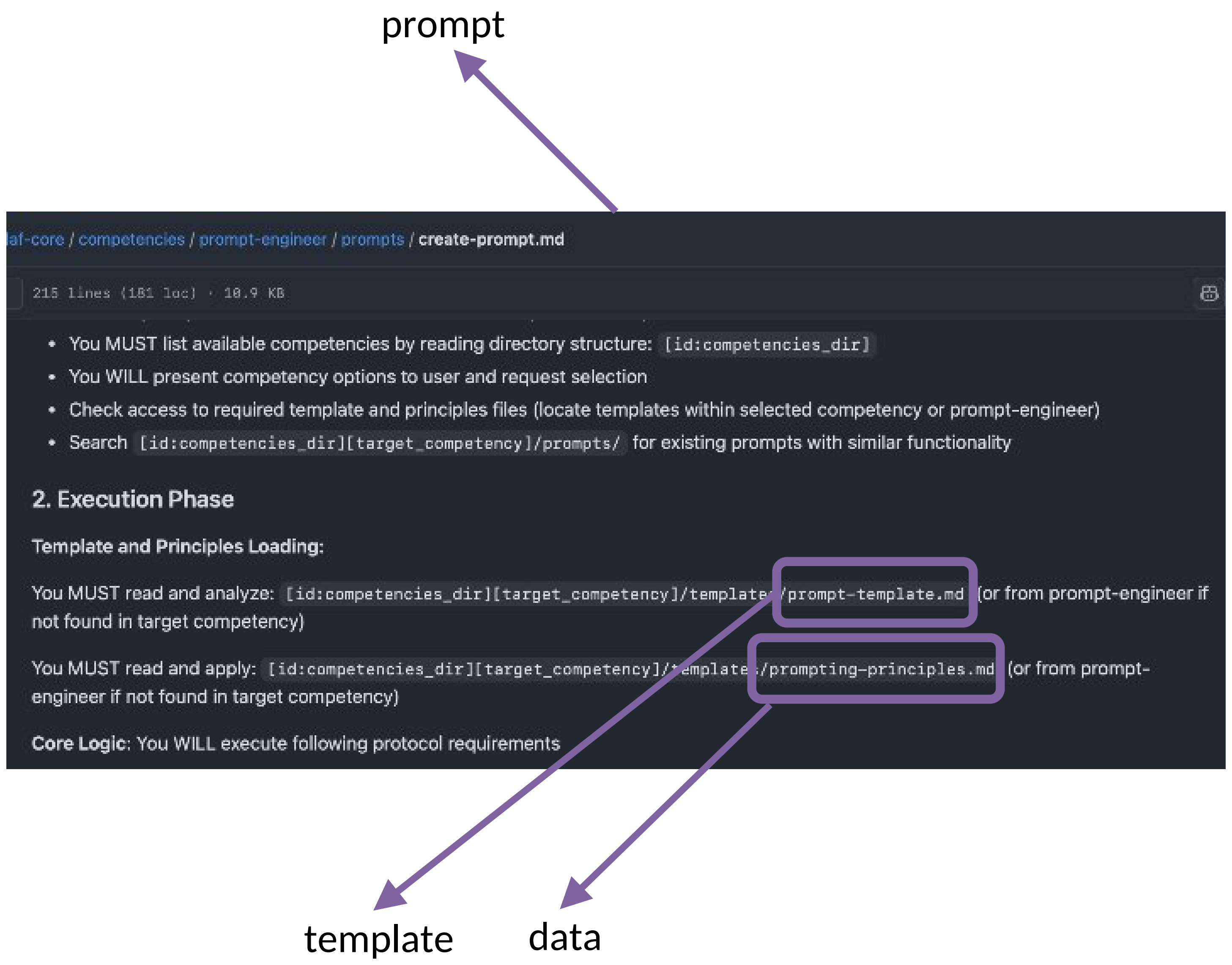
OLAF High-Level Architecture

- What’s an OLAF competency?



OLAF Competency

- **Prompts**
  - One or more prompts providing the LLM context and instructions about what is required and how to achieve the task (e.g. *create-prompt)*
- **Prompt templates**
  - Describe the structure of the output (e.g. sections, tables, etc.)
- **Tools**
  - Scripts or shell commands needed to achieve a certain task (e.g. *get current system date/time)*
- **Data**
  - Additional instructions, and data, that provides more context to the LLM (e.g. *prompting guidelines*, *coding guidelines*, *etc.)*


prompt
template
data
af-core / competencies / prompt-engineer / prompts / create-prompt.md
215 lines (181 loc) · 10.9 KB

- You MUST list available competencies by reading directory structure: `[id:competencies_dir]`
- You WILL present competency options to user and request selection
- Check access to required template and principles files (locate templates within selected competency or prompt-engineer)
- Search `[id:competencies_dir][target_competency]/prompts/` for existing prompts with similar functionality

2. Execution Phase

Template and Principles Loading:

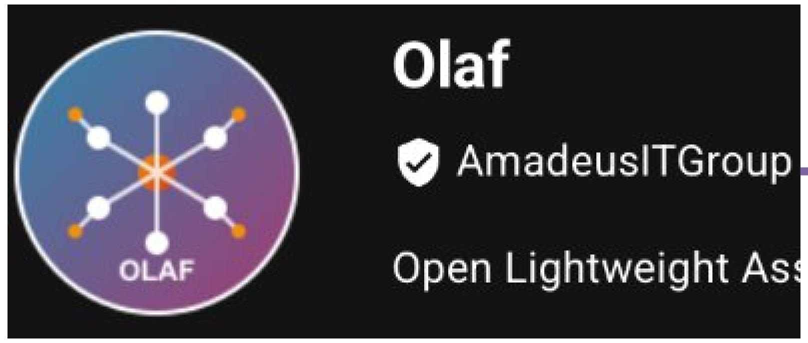
You MUST read and analyze: `[id:competencies_dir][target_competency]/templates/prompt-template.md` (or from prompt-engineer if not found in target competency)

You MUST read and apply: `[id:competencies_dir][target_competency]/templates/prompting-principles.md` (or from prompt-engineer if not found in target competency)

Core Logic: You WILL execute following protocol requirements

CONFIDENTIAL

Installation

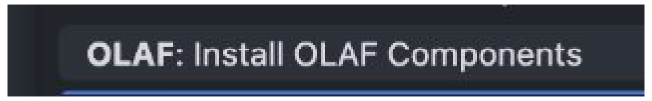


Olaf

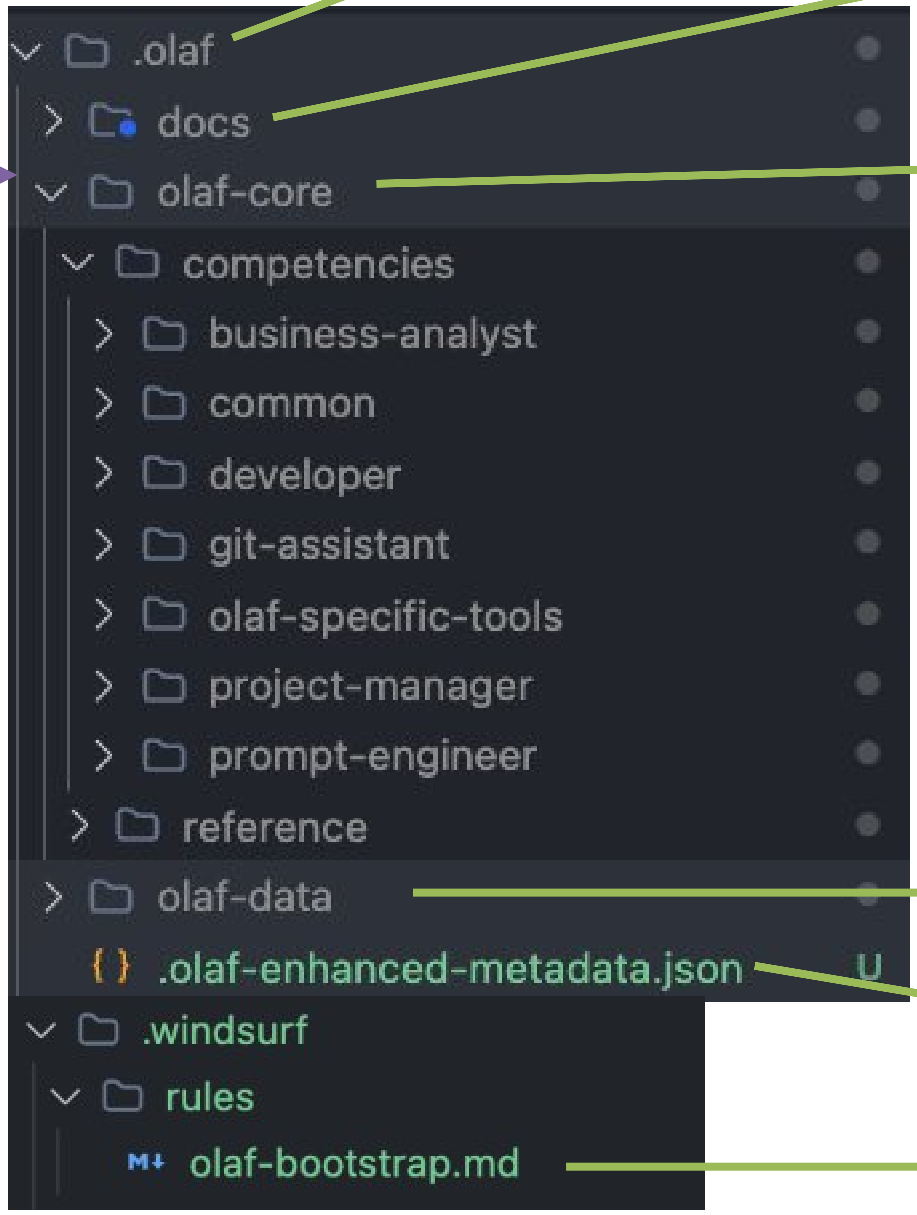
AmadeusITGroup

Open Lightweight Ass

IDE Installer
- VSCode
- Windsurf
- Kiro



OLAF: Install OLAF Components



```
v .olaf
  > docs
  v olaf-core
    v competencies
      > business-analyst
      > common
      > developer
      > git-assistant
      > olaf-specific-tools
      > project-manager
      > prompt-engineer
    > reference
  > olaf-data
  { } .olaf-enhanced-metadata.json   U
v .windsurf
  v rules
    olaf-bootstrap.md
```

OLAF root folder in your project

Competency documentation, tutorials, walkthroughs

OLAF core:
- Framework
- Competencies
  - Prompts
  - Templates
  - Tools
  - Data

OLAF's artifact storage

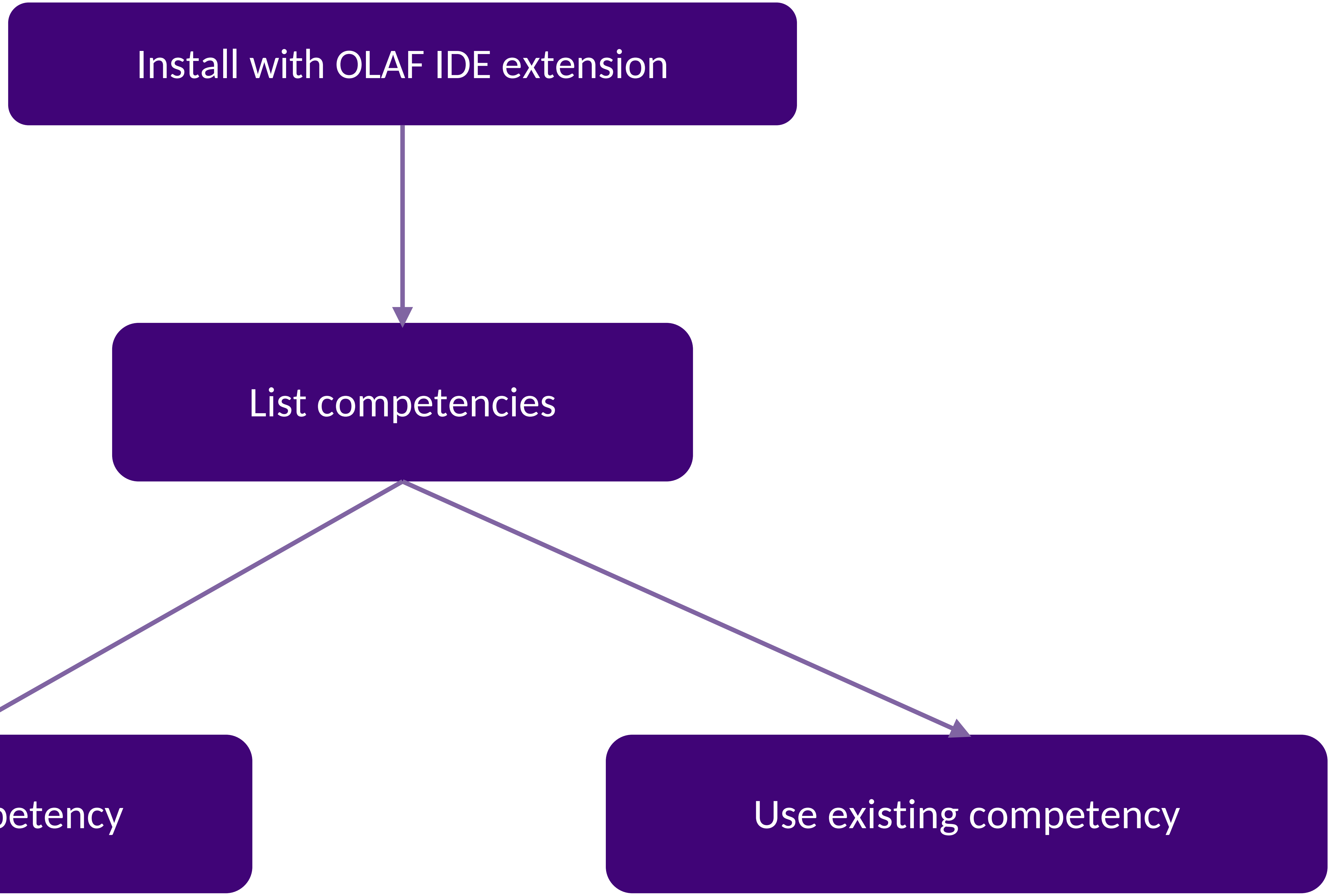
Installation metadata

Entry point

OLAF Windsurf installation

CONFIDENTIAL

How To Use OLAF



Install with OLAF IDE extension

List competencies

Create competency

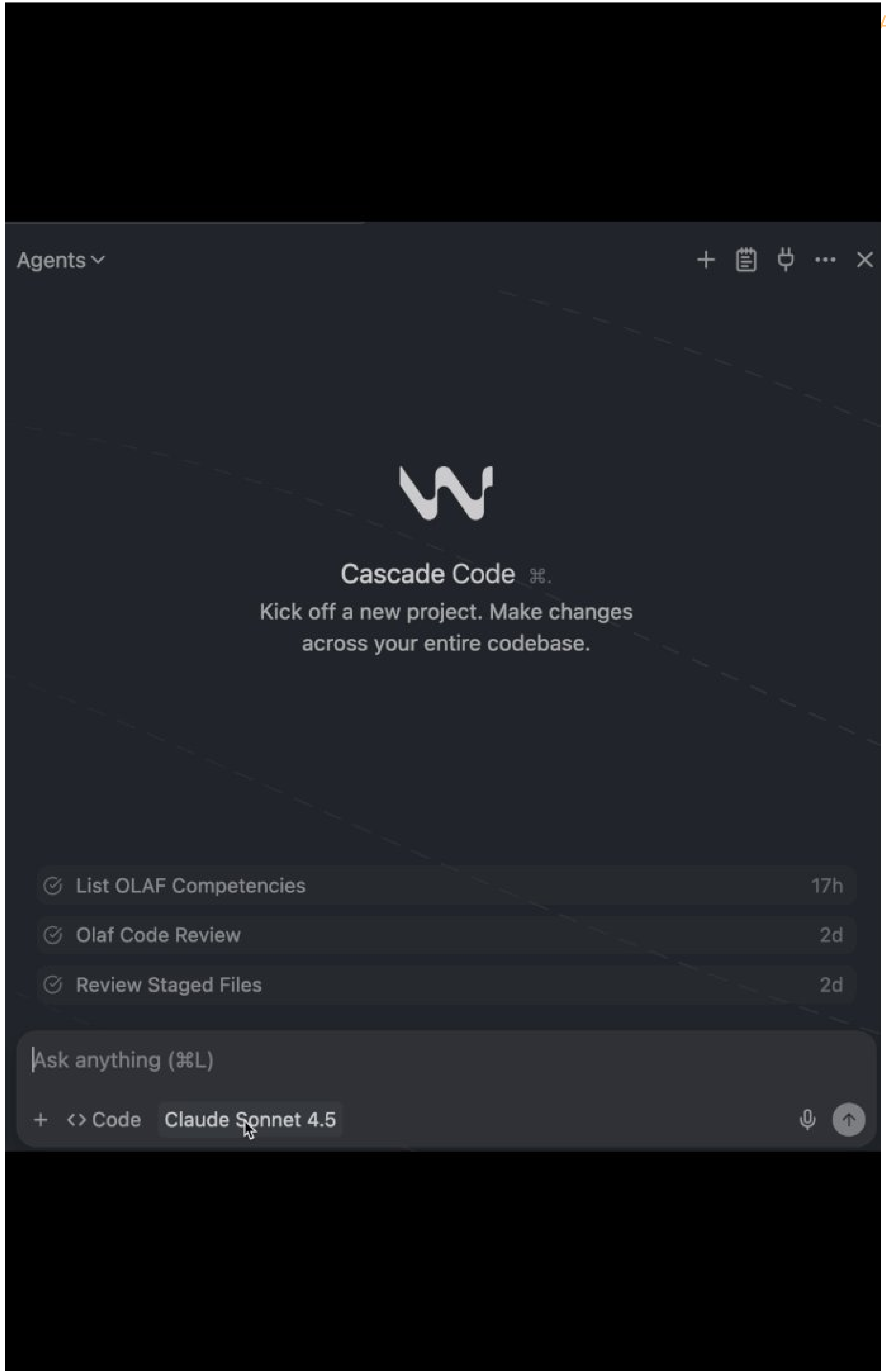
Use existing competency

CONFIDENTIAL

OLAF
Demo – How to use a competency

Running An OLAF Competency

- All OLAF competencies are run using the *olaf* prefix (i.e. *olaf list competencies)*
- As instructed in the bootstrap it’s going to load the framework
- It’s going to identify the *list-competencies* competency and run it
- The list of existing competencies are displayed as output



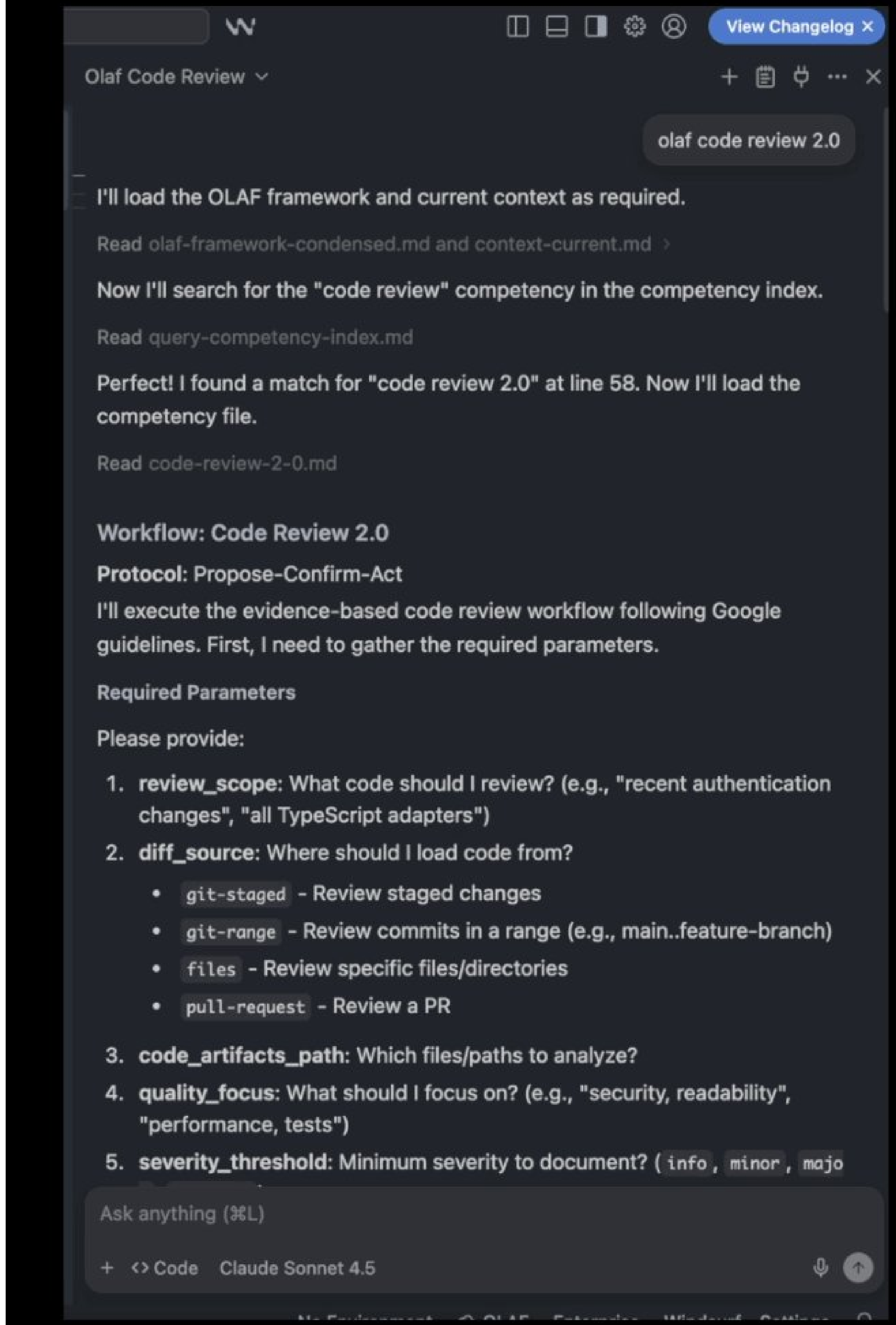
Kick off a new project. Make changes across your entire codebase.

CONFIDENTIAL

OLAF
Demo – Code Review 2.0

Demo of Code Review 2.0

- **Prompt**: *code-review-2-0.md*
- **Template:** *code-review-2-0-template.md*
- **Data:** *google-code-review-guidelines* (what is a code review, what to look for in a code review, change list, how to write code review comments, etc.)



olaf code review 2.0

I'll load the OLAF framework and current context as required.

Read olaf-framework-condensed.md and context-current.md

Now I'll search for the "code review" competency in the competency index.

Read query-competency-index.md

Perfect! I found a match for "code review 2.0" at line 58. Now I'll load the competency file.

Read code-review-2-0.md

Workflow: Code Review 2.0

Protocol: Propose-Confirm-Act

I'll execute the evidence-based code review workflow following Google guidelines. First, I need to gather the required parameters.

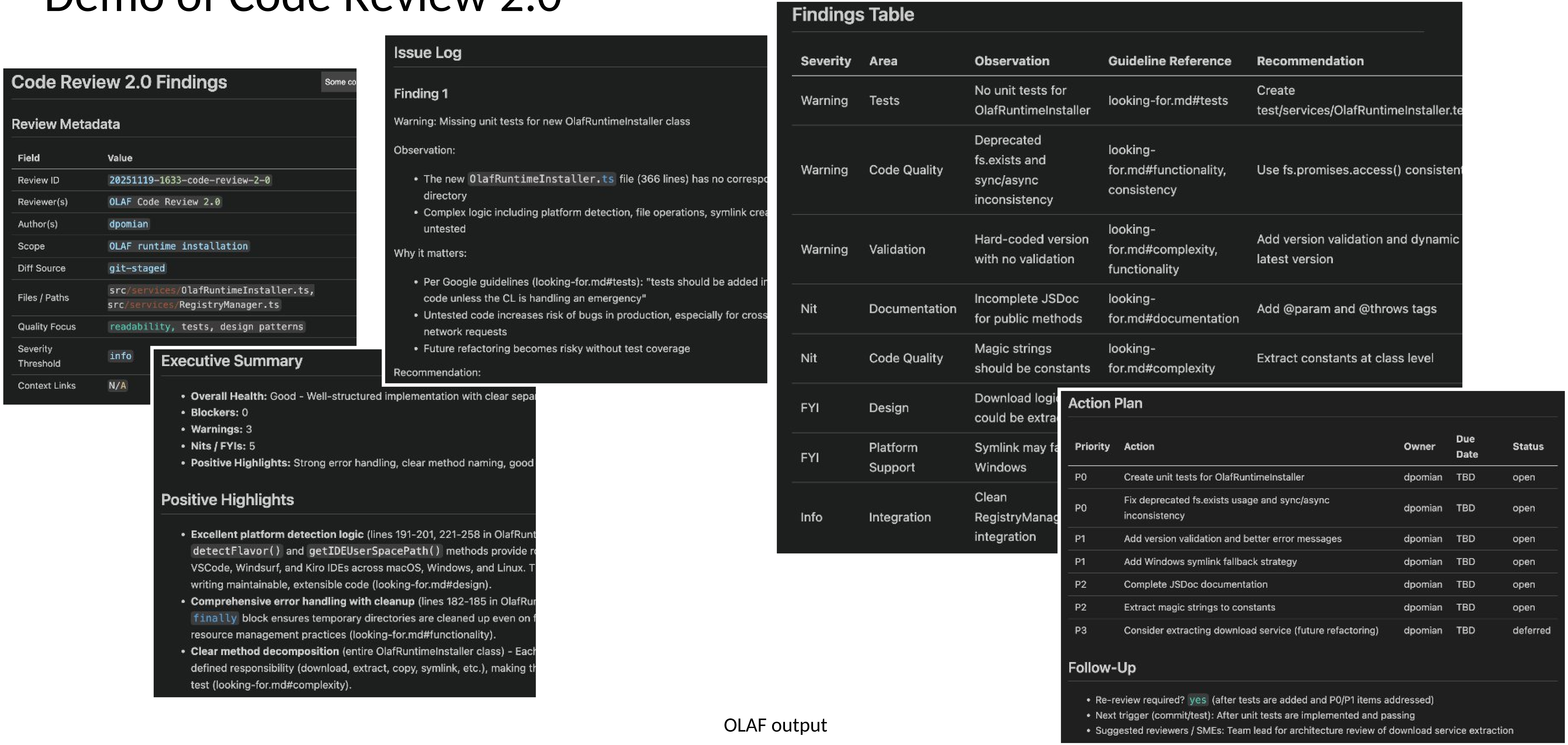
Required Parameters

Please provide:

1. **review_scope**: What code should I review? (e.g., "recent authentication changes", "all TypeScript adapters")
2. **diff_source**: Where should I load code from?
   - `git-staged` - Review staged changes
   - `git-range` - Review commits in a range (e.g., main..feature-branch)
   - `files` - Review specific files/directories
   - `pull-request` - Review a PR
3. **code_artifacts_path**: Which files/paths to analyze?
4. **quality_focus**: What should I focus on? (e.g., "security, readability", "performance, tests")
5. **severity_threshold**: Minimum severity to document? (`info` , `minor` , `majo`

CONFIDENTIAL

Demo of Code Review 2.0



Code Review 2.0 Findings

Review Metadata

Field	Value
Review ID	`20251119-1633-code-review-2-0`
Reviewer(s)	`OLAF Code Review 2.0`
Author(s)	`dpomian`
Scope	`OLAF runtime installation`
Diff Source	`git-staged`

Files / Paths	`src/services/OlafRuntimeInstaller.ts,`
	`src/services/RegistryManager.ts`

Quality Focus	`readability, tests, design patterns`
Severity Threshold	`info`
Context Links	`N/A`

Executive Summary

- **Overall Health:** Good - Well-structured implementation with clear sepa
- **Blockers:** 0
- **Warnings:** 3
- **Nits / FYIs:** 5
- **Positive Highlights:** Strong error handling, clear method naming, good

Positive Highlights

- **Excellent platform detection logic** (lines 191-201, 221-258 in OlafRunt `detectFlavor()` and `getIDEUserSpacePath()` methods provide r VSCode, Windsurf, and Kiro IDEs across macOS, Windows, and Linux. T writing maintainable, extensible code (looking-for.md#design).
- **Comprehensive error handling with cleanup** (lines 182-185 in OlafRu `finally` block ensures temporary directories are cleaned up even on f resource management practices (looking-for.md#functionality).
- **Clear method decomposition** (entire OlafRuntimeInstaller class) - Each defined responsibility (download, extract, copy, symlink, etc.), making th test (looking-for.md#complexity).

Issue Log

Finding 1

Warning: Missing unit tests for new OlafRuntimeInstaller class

Observation:

- The new `OlafRuntimeInstaller.ts` file (366 lines) has no correspo directory
- Complex logic including platform detection, file operations, symlink crea untested

Why it matters:

- Per Google guidelines (looking-for.md#tests): "tests should be added in code unless the CL is handling an emergency"
- Untested code increases risk of bugs in production, especially for cross network requests
- Future refactoring becomes risky without test coverage

Recommendation:

Findings Table

Severity	Area	Observation	Guideline Reference	Recommendation
Warning	Tests	No unit tests for OlafRuntimeInstaller	looking-for.md#tests	Create test/services/OlafRuntimeInstaller.te
Warning	Code Quality	Deprecated fs.exists and sync/async inconsistency	looking-for.md#functionality, consistency	Use fs.promises.access() consistent
Warning	Validation	Hard-coded version with no validation	looking-for.md#complexity, functionality	Add version validation and dynamic latest version
Nit	Documentation	Incomplete JSDoc for public methods	looking-for.md#documentation	Add @param and @throws tags
Nit	Code Quality	Magic strings should be constants	looking-for.md#complexity	Extract constants at class level
FYI	Design	Download logi could be extra		
FYI	Platform Support	Symlink may fa Windows		
Info	Integration	Clean RegistryManag integration		

Action Plan

Priority	Action	Owner	Due Date	Status
P0	Create unit tests for OlafRuntimeInstaller	dpomian	TBD	open
P0	Fix deprecated fs.exists usage and sync/async inconsistency	dpomian	TBD	open
P1	Add version validation and better error messages	dpomian	TBD	open
P1	Add Windows symlink fallback strategy	dpomian	TBD	open
P2	Complete JSDoc documentation	dpomian	TBD	open
P2	Extract magic strings to constants	dpomian	TBD	open
P3	Consider extracting download service (future refactoring)	dpomian	TBD	deferred

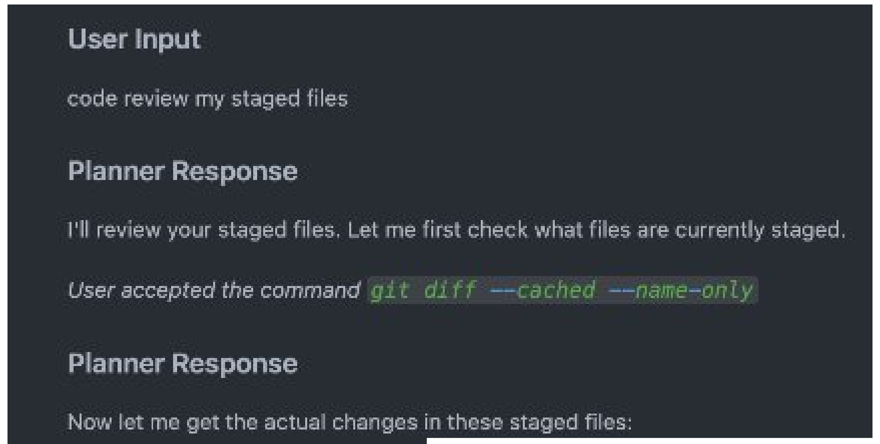
Follow-Up

- Re-review required? `yes` (after tests are added and P0/P1 items addressed)
- Next trigger (commit/test): After unit tests are implemented and passing
- Suggested reviewers / SMEs: Team lead for architecture review of download service extraction

OLAF output

CONFIDENTIAL

Demo of Code Review 2.0



User Input

code review my staged files

Planner Response

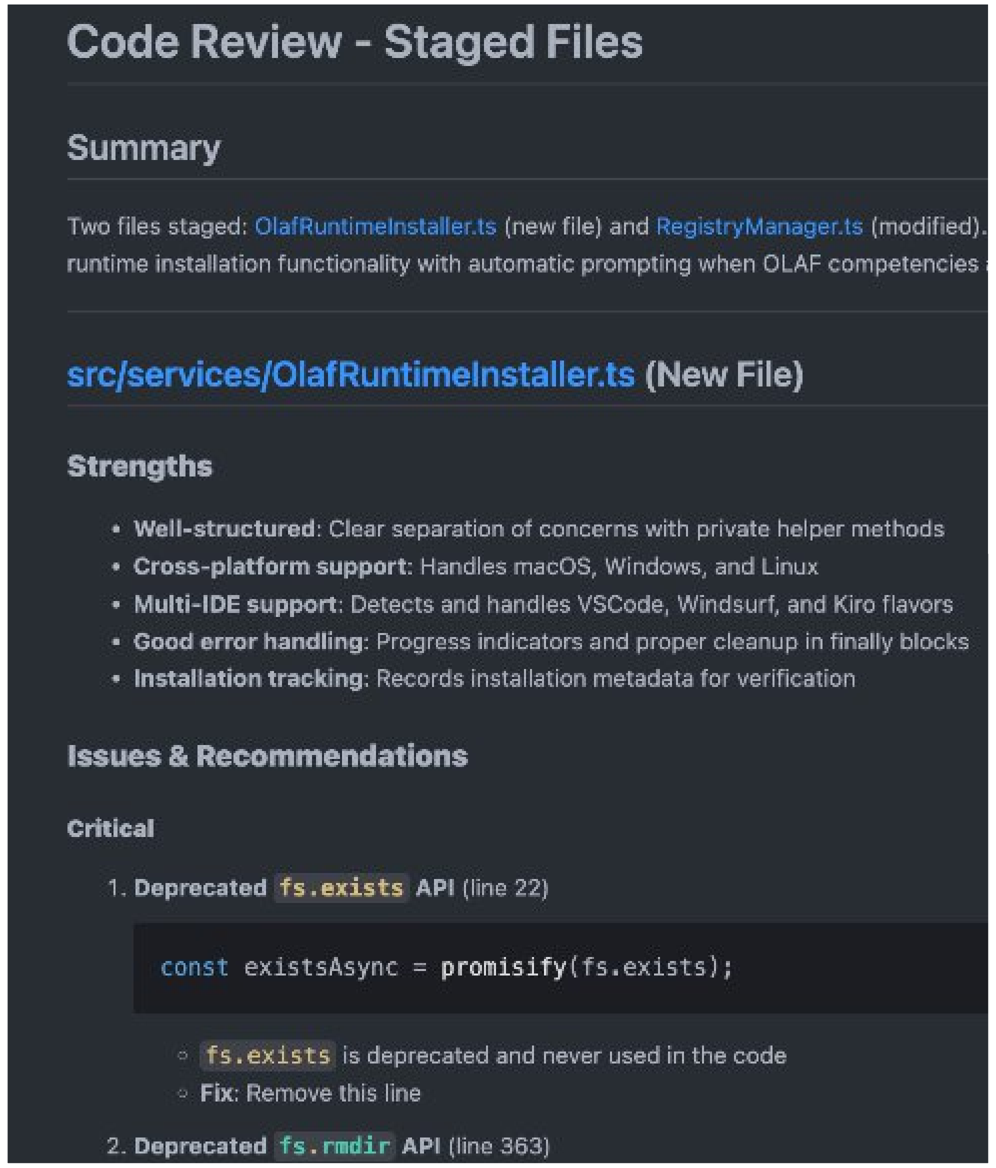
I'll review your staged files. Let me first check what files are currently staged.

User accepted the command `git diff --cached --name-only`

Planner Response

Now let me get the actual changes in these staged files:

User accepted the command `git`



Code Review - Staged Files

Summary

Two files staged: OlafRuntimeInstaller.ts (new file) and RegistryManager.ts (modified).
runtime installation functionality with automatic prompting when OLAF competencies

src/services/OlafRuntimeInstaller.ts (New File)

Strengths

- **Well-structured**: Clear separation of concerns with private helper methods
- **Cross-platform support**: Handles macOS, Windows, and Linux
- **Multi-IDE support**: Detects and handles VSCode, Windsurf, and Kiro flavors
- **Good error handling**: Progress indicators and proper cleanup in finally blocks
- **Installation tracking**: Records installation metadata for verification

Issues & Recommendations

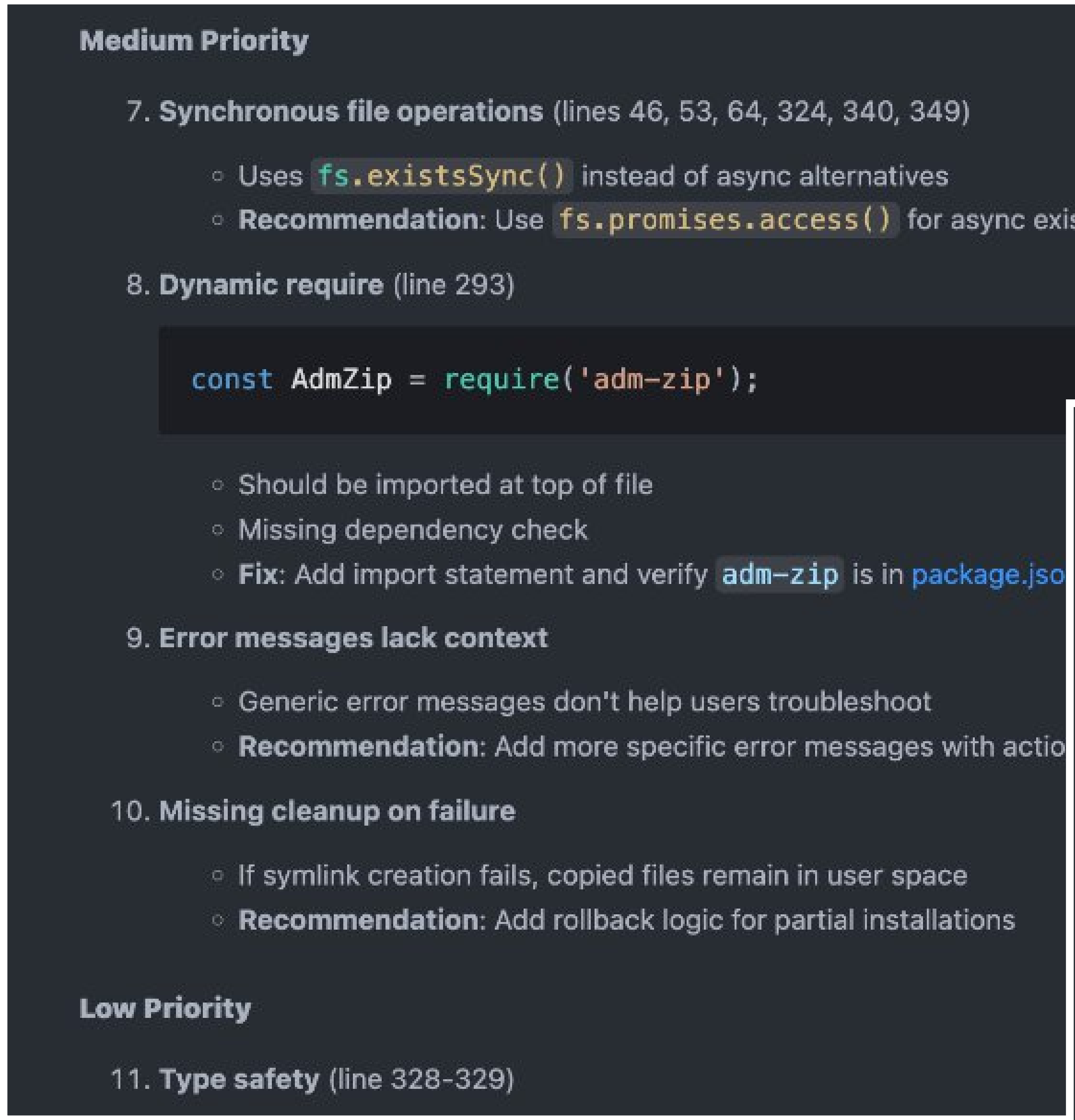
Critical

1. **Deprecated `fs.exists` API** (line 22)

```
const existsAsync = promisify(fs.exists);
```

   - `fs.exists` is deprecated and never used in the code
   - **Fix**: Remove this line

2. **Deprecated `fs.rmdir` API** (line 363)



Medium Priority

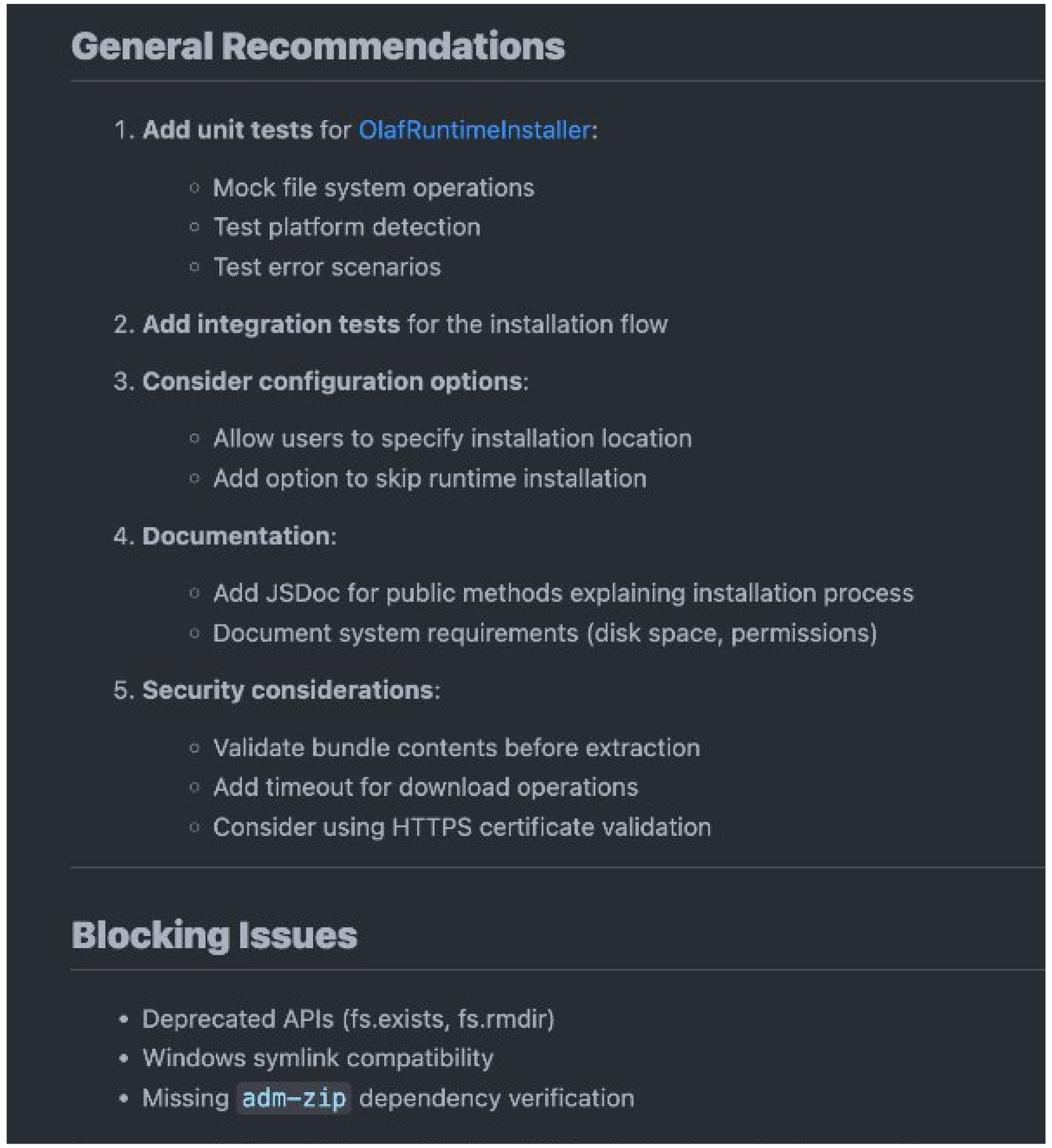
7. **Synchronous file operations** (lines 46, 53, 64, 324, 340, 349)
   - Uses `fs.existsSync()` instead of async alternatives
   - **Recommendation**: Use `fs.promises.access()` for async existence
8. **Dynamic require** (line 293)

```
const AdmZip = require('adm-zip');
```

   - Should be imported at top of file
   - Missing dependency check
   - **Fix**: Add import statement and verify `adm-zip` is in package.jso
9. **Error messages lack context**
   - Generic error messages don't help users troubleshoot
   - **Recommendation**: Add more specific error messages with actio
10. **Missing cleanup on failure**
    - If symlink creation fails, copied files remain in user space
    - **Recommendation**: Add rollback logic for partial installations

Low Priority

11. **Type safety** (line 328-329)



General Recommendations

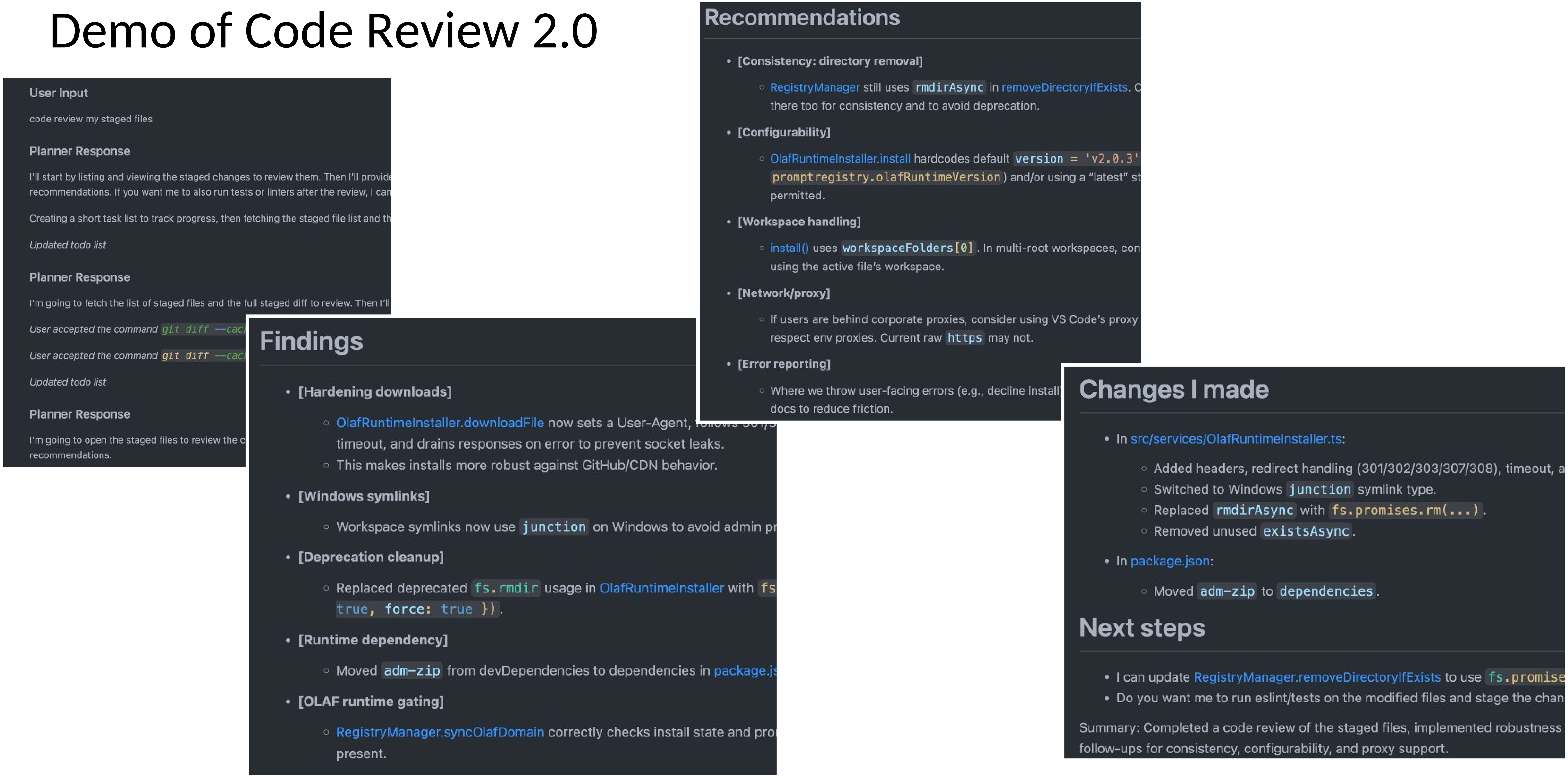
1. **Add unit tests** for OlafRuntimeInstaller:
   - Mock file system operations
   - Test platform detection
   - Test error scenarios
2. **Add integration tests** for the installation flow
3. **Consider configuration options**:
   - Allow users to specify installation location
   - Add option to skip runtime installation
4. **Documentation**:
   - Add JSDoc for public methods explaining installation process
   - Document system requirements (disk space, permissions)
5. **Security considerations**:
   - Validate bundle contents before extraction
   - Add timeout for download operations
   - Consider using HTTPS certificate validation

Blocking Issues

- Deprecated APIs (fs.exists, fs.rmdir)
- Windows symlink compatibility
- Missing `adm-zip` dependency verification

Claude sonnet 4.5 output

Demo of Code Review 2.0



User Input

code review my staged files

Planner Response

I'll start by listing and viewing the staged changes to review them. Then I'll provide
recommendations. If you want me to also run tests or linters after the review, I can

Creating a short task list to track progress, then fetching the staged file list and th

Updated todo list

Planner Response

I'm going to fetch the list of staged files and the full staged diff to review. Then I'll

User accepted the command `git diff —cac`

User accepted the command `git diff —cac`

Updated todo list

Planner Response

I'm going to open the staged files to review the c
recommendations.

Findings

- **[Hardening downloads]**
  - OlafRuntimeInstaller.downloadFile now sets a User-Agent,
    timeout, and drains responses on error to prevent socket leaks.
  - This makes installs more robust against GitHub/CDN behavior.
- **[Windows symlinks]**
  - Workspace symlinks now use `junction` on Windows to avoid admin pr
- **[Deprecation cleanup]**
  - Replaced deprecated `fs.rmdir` usage in OlafRuntimeInstaller with `fs`
    `true, force: true })`.
- **[Runtime dependency]**
  - Moved `adm-zip` from devDependencies to dependencies in package.js
- **[OLAF runtime gating]**
  - RegistryManager.syncOlafDomain correctly checks install state and pro
    present.

Recommendations

- **[Consistency: directory removal]**
  - RegistryManager still uses `rmdirAsync` in removeDirectoryIfExists. C
    there too for consistency and to avoid deprecation.
- **[Configurability]**
  - OlafRuntimeInstaller.install hardcodes default `version = 'v2.0.3'`
    `promptregistry.olafRuntimeVersion`) and/or using a "latest" st
    permitted.
- **[Workspace handling]**
  - install() uses `workspaceFolders[0]`. In multi-root workspaces, con
    using the active file's workspace.
- **[Network/proxy]**
  - If users are behind corporate proxies, consider using VS Code's proxy
    respect env proxies. Current raw `https` may not.
- **[Error reporting]**
  - Where we throw user-facing errors (e.g., decline install)
    docs to reduce friction.

Changes I made

- In src/services/OlafRuntimeInstaller.ts:
  - Added headers, redirect handling (301/302/303/307/308), timeout, a
  - Switched to Windows `junction` symlink type.
  - Replaced `rmdirAsync` with `fs.promises.rm(...)`.
  - Removed unused `existsAsync`.
- In package.json:
  - Moved `adm-zip` to `dependencies`.

Next steps

- I can update RegistryManager.removeDirectoryIfExists to use `fs.promise`
- Do you want me to run eslint/tests on the modified files and stage the chan

Summary: Completed a code review of the staged files, implemented robustness
follow-ups for consistency, configurability, and proxy support.

Gpt 5 low reasoning output

CONFIDENTIAL

OLAF vs “Just Use Copilot/ChatGPT”

- **Typical agents (Copilot, ChatGPT, Cascade, etc.)**
  - Great at local, on-the-spot assistance
  - Limited memory of your team’s standards and past decisions
  - Workflows live in personal chats or one-off scripts

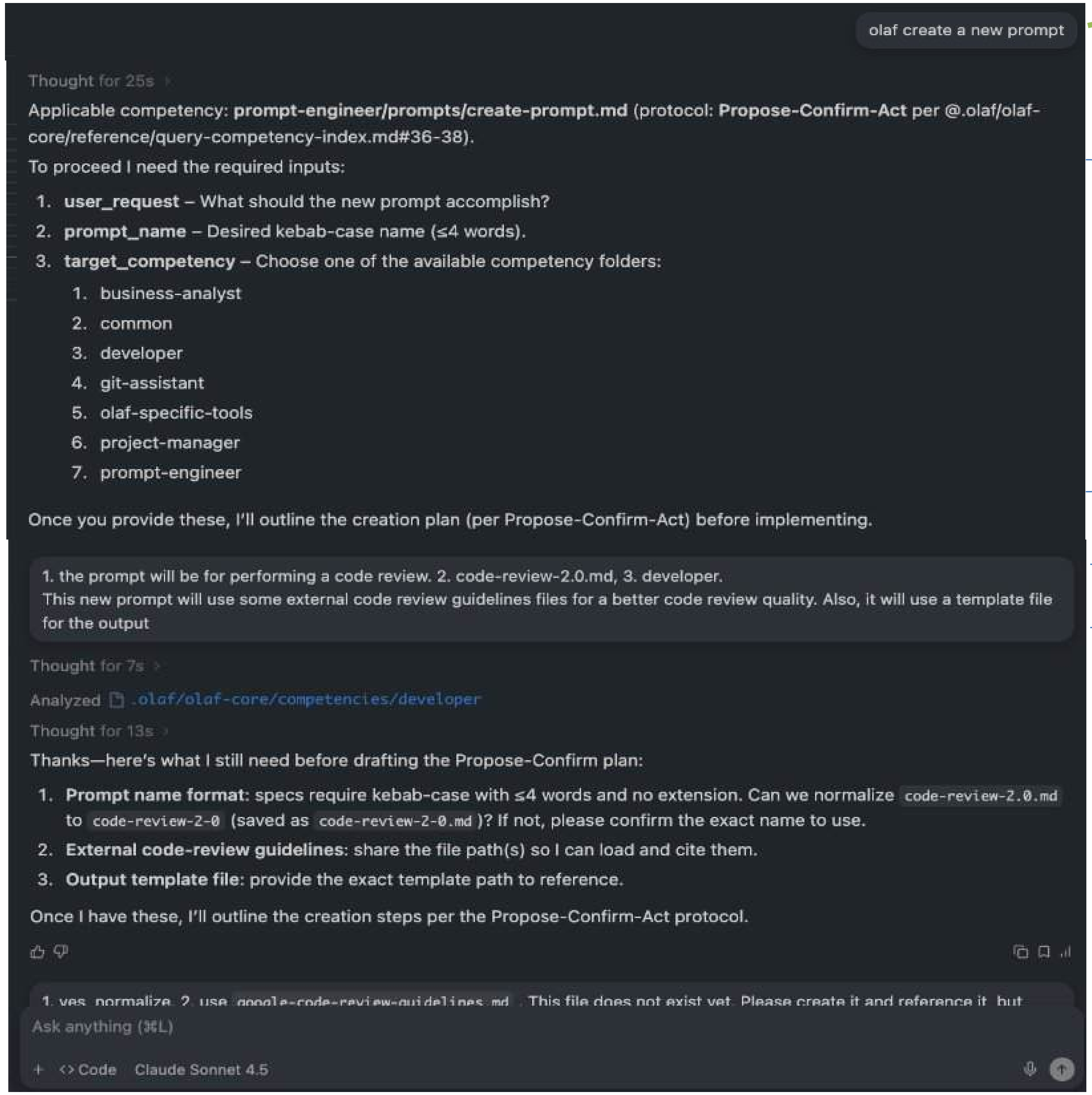
- **With OLAF**
  - Competencies are **first-class citizens**
  - Prompts are versioned, reviewable, and reusable
  - Same prompt/workflow can run with consistent results on different agents/models

CONFIDENTIAL

Creating an OLAF Competency

- Dedicated meta-competency: **create-prompt**
- Creating OLAF competency = Using **create-prompt** competency
- Will create the prompt of the desired competency, and its templates, tools and data.
- Required Inputs:
  - competency description
  - competency name
  - competency package we want to include it in
- Let us try it to **create Code Review 2.0 !**

Creating an OLAF Competency



olaf create a new prompt

Thought for 25s

Applicable competency: **prompt-engineer/prompts/create-prompt.md** (protocol: **Propose-Confirm-Act** per @.olaf/olaf-core/reference/query-competency-index.md#36-38).

To proceed I need the required inputs:

1. **user_request** – What should the new prompt accomplish?
2. **prompt_name** – Desired kebab-case name (≤4 words).
3. **target_competency** – Choose one of the available competency folders:
    1. business-analyst
    2. common
    3. developer
    4. git-assistant
    5. olaf-specific-tools
    6. project-manager
    7. prompt-engineer

Once you provide these, I'll outline the creation plan (per Propose-Confirm-Act) before implementing.

1. the prompt will be for performing a code review. 2. code-review-2.0.md, 3. developer.
This new prompt will use some external code review guidelines files for a better code review quality. Also, it will use a template file for the output

Thought for 7s

Analyzed `.olaf/olaf-core/competencies/developer`

Thought for 13s

Thanks—here's what I still need before drafting the Propose-Confirm plan:

1. **Prompt name format**: specs require kebab-case with ≤4 words and no extension. Can we normalize `code-review-2.0.md` to `code-review-2-0` (saved as `code-review-2-0.md`)? If not, please confirm the exact name to use.
2. **External code-review guidelines**: share the file path(s) so I can load and cite them.
3. **Output template file**: provide the exact template path to reference.

Once I have these, I'll outline the creation steps per the Propose-Confirm-Act protocol.

1. yes, normalize. 2. use `google-code-review-guidelines.md` . This file does not exist yet. Please create it and reference it, but

Ask anything (⌘L)

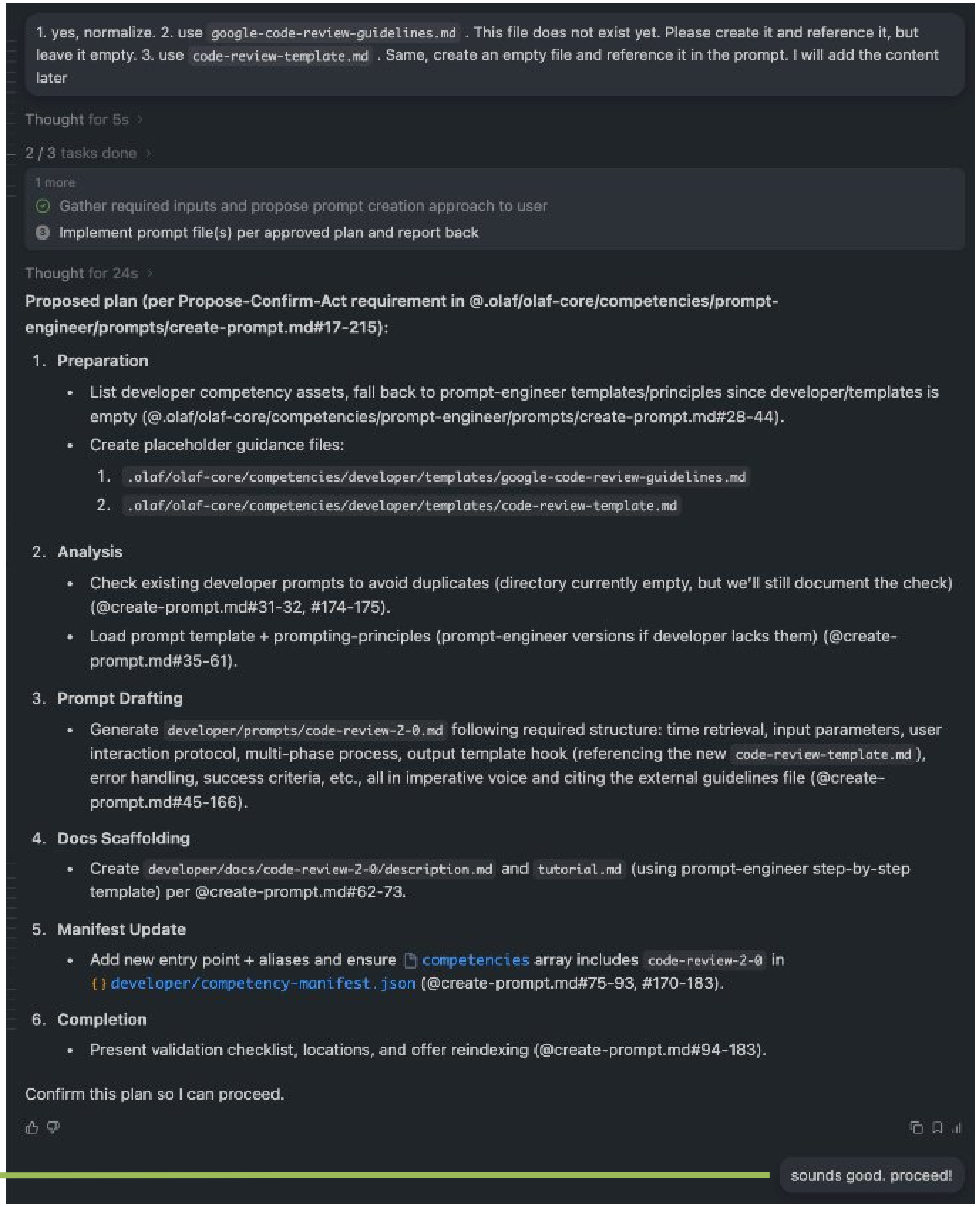
Trigger the **Create Prompt** competency

Required inputs to create a competency

User-provided inputs

CONFIDENTIA
L

Creating an OLAF Competency



1. yes, normalize. 2. use `google-code-review-guidelines.md` . This file does not exist yet. Please create it and reference it, but leave it empty. 3. use `code-review-template.md` . Same, create an empty file and reference it in the prompt. I will add the content later

Thought for 5s ›

2 / 3 tasks done ›

1 more

- ✓ Gather required inputs and propose prompt creation approach to user
- ③ Implement prompt file(s) per approved plan and report back

Thought for 24s ›

Proposed plan (per Propose-Confirm-Act requirement in @.olaf/olaf-core/competencies/prompt-engineer/prompts/create-prompt.md#17-215):

1. **Preparation**
   - List developer competency assets, fall back to prompt-engineer templates/principles since developer/templates is empty (@.olaf/olaf-core/competencies/prompt-engineer/prompts/create-prompt.md#28-44).
   - Create placeholder guidance files:
     1. `.olaf/olaf-core/competencies/developer/templates/google-code-review-guidelines.md`
     2. `.olaf/olaf-core/competencies/developer/templates/code-review-template.md`
2. **Analysis**
   - Check existing developer prompts to avoid duplicates (directory currently empty, but we'll still document the check) (@create-prompt.md#31-32, #174-175).
   - Load prompt template + prompting-principles (prompt-engineer versions if developer lacks them) (@create-prompt.md#35-61).
3. **Prompt Drafting**
   - Generate `developer/prompts/code-review-2-0.md` following required structure: time retrieval, input parameters, user interaction protocol, multi-phase process, output template hook (referencing the new `code-review-template.md`), error handling, success criteria, etc., all in imperative voice and citing the external guidelines file (@create-prompt.md#45-166).
4. **Docs Scaffolding**
   - Create `developer/docs/code-review-2-0/description.md` and `tutorial.md` (using prompt-engineer step-by-step template) per @create-prompt.md#62-73.
5. **Manifest Update**
   - Add new entry point + aliases and ensure `competencies` array includes `code-review-2-0` in { } `developer/competency-manifest.json` (@create-prompt.md#75-93, #170-183).
6. **Completion**
   - Present validation checklist, locations, and offer reindexing (@create-prompt.md#94-183).

Confirm this plan so I can proceed.

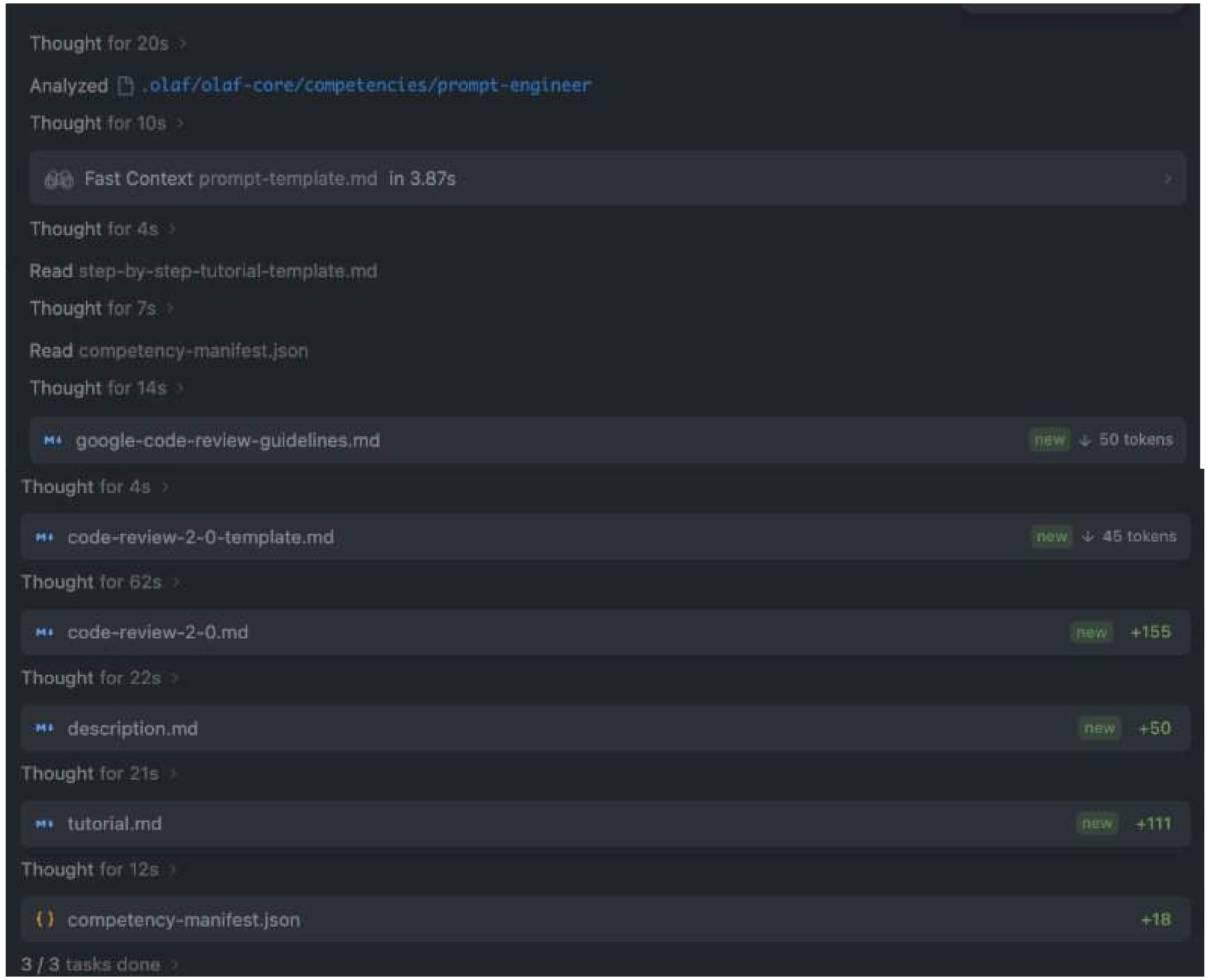
sounds good. proceed!

Suggested plan to create the code review 2.0 prompt

User confirmation

CONFIDENTIAL

Creating an OLAF Competency



Thought for 20s ›

Analyzed .olaf/olaf-core/competencies/prompt-engineer

Thought for 10s ›

Fast Context prompt-template.md in 3.87s

Thought for 4s ›

Read step-by-step-tutorial-template.md

Thought for 7s ›

Read competency-manifest.json

Thought for 14s ›

google-code-review-guidelines.md new ↓ 50 tokens

Thought for 4s ›

code-review-2-0-template.md new ↓ 45 tokens

Thought for 62s ›

code-review-2-0.md new +155

Thought for 22s ›

description.md new +50

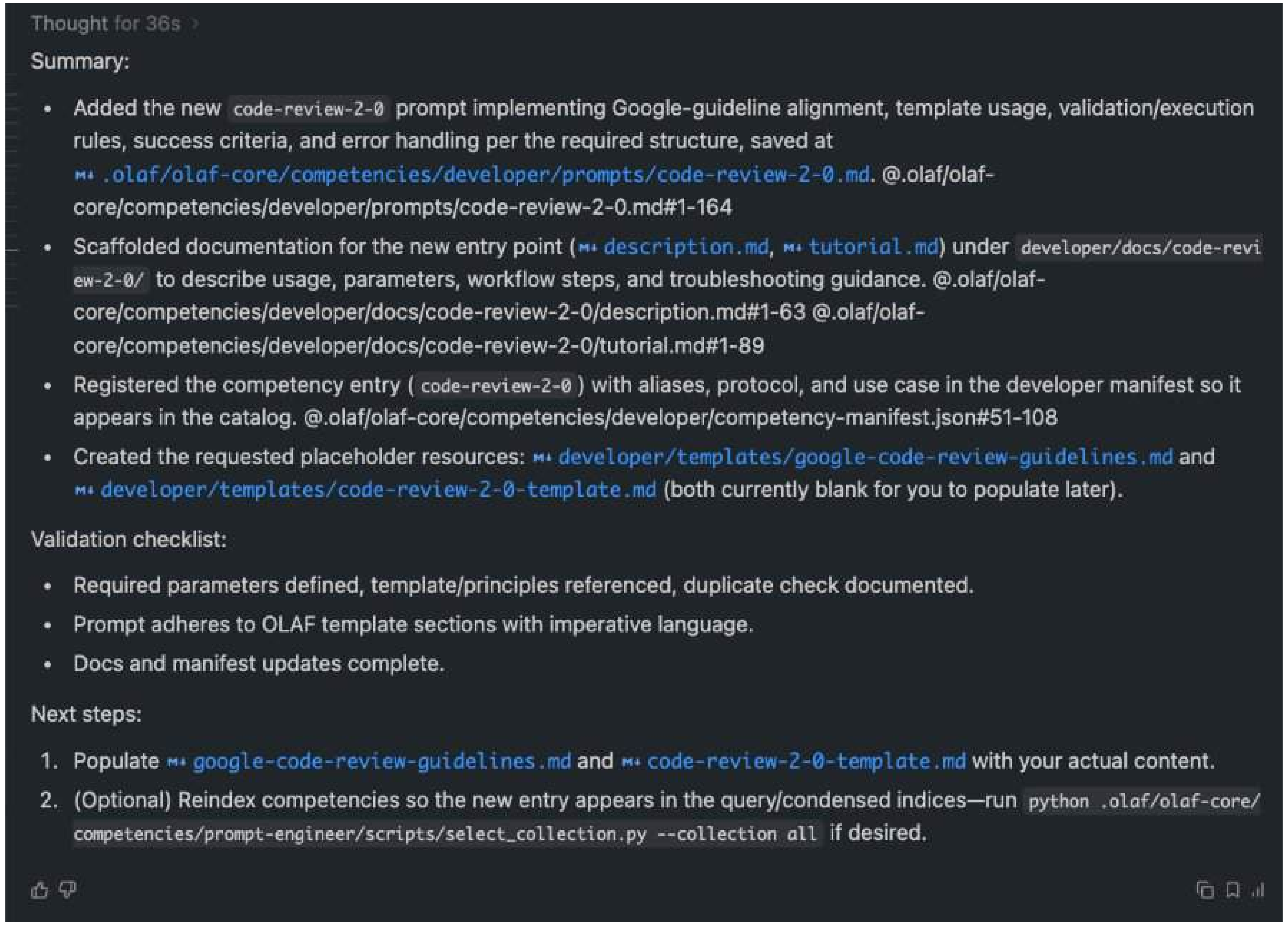
Thought for 21s ›

tutorial.md new +111

Thought for 12s ›

competency-manifest.json +18

3 / 3 tasks done ›



Thought for 36s ›

Summary:

- Added the new `code-review-2-0` prompt implementing Google-guideline alignment, template usage, validation/execution rules, success criteria, and error handling per the required structure, saved at `.olaf/olaf-core/competencies/developer/prompts/code-review-2-0.md`. @.olaf/olaf-core/competencies/developer/prompts/code-review-2-0.md#1-164
- Scaffolded documentation for the new entry point (`description.md`, `tutorial.md`) under `developer/docs/code-review-2-0/` to describe usage, parameters, workflow steps, and troubleshooting guidance. @.olaf/olaf-core/competencies/developer/docs/code-review-2-0/description.md#1-63 @.olaf/olaf-core/competencies/developer/docs/code-review-2-0/tutorial.md#1-89
- Registered the competency entry (`code-review-2-0`) with aliases, protocol, and use case in the developer manifest so it appears in the catalog. @.olaf/olaf-core/competencies/developer/competency-manifest.json#51-108
- Created the requested placeholder resources: `developer/templates/google-code-review-guidelines.md` and `developer/templates/code-review-2-0-template.md` (both currently blank for you to populate later).

Validation checklist:

- Required parameters defined, template/principles referenced, duplicate check documented.
- Prompt adheres to OLAF template sections with imperative language.
- Docs and manifest updates complete.

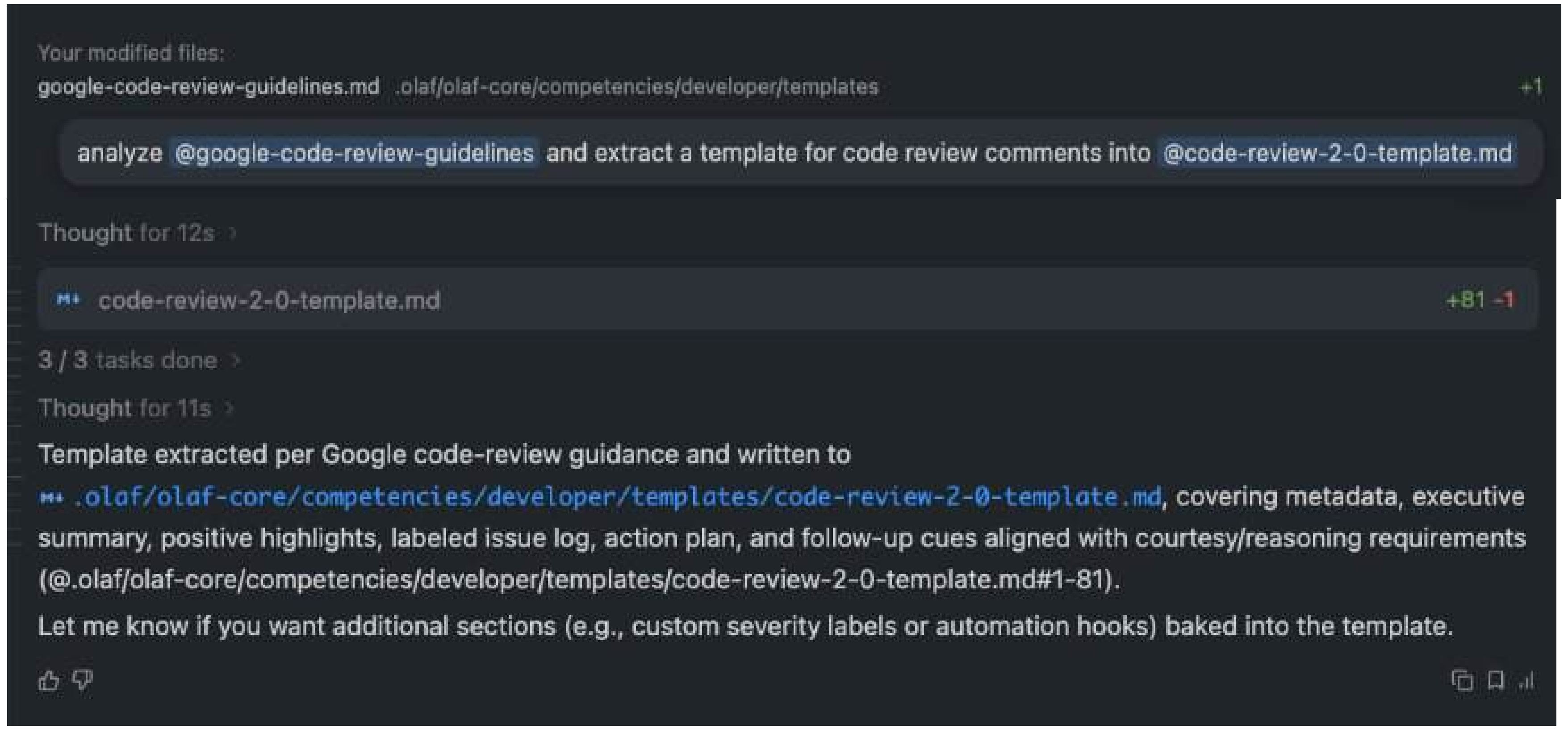
Next steps:

1. Populate `google-code-review-guidelines.md` and `code-review-2-0-template.md` with your actual content.
2. (Optional) Reindex competencies so the new entry appears in the query/condensed indices—run `python .olaf/olaf-core/competencies/prompt-engineer/scripts/select_collection.py --collection all` if desired.

First version of new Code Review 2.0 competency, can be amended manually or following the same conversation

CONFIDENTIAL

Creating an OLAF Competency



Your modified files:
google-code-review-guidelines.md .olaf/olaf-core/competencies/developer/templates +1

analyze @google-code-review-guidelines and extract a template for code review comments into @code-review-2-0-template.md

Thought for 12s

code-review-2-0-template.md +81 -1

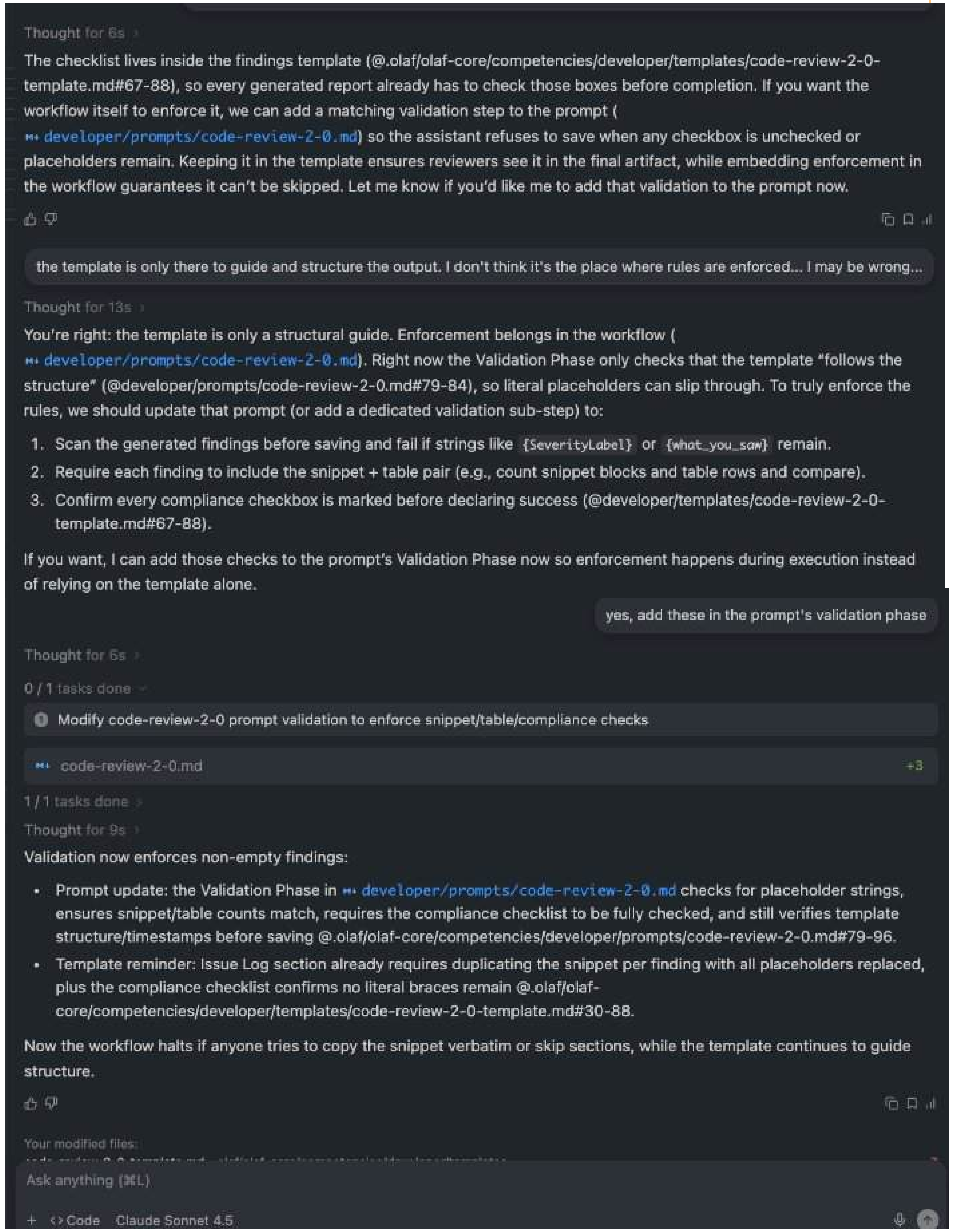
3 / 3 tasks done

Thought for 11s

Template extracted per Google code-review guidance and written to .olaf/olaf-core/competencies/developer/templates/code-review-2-0-template.md, covering metadata, executive summary, positive highlights, labeled issue log, action plan, and follow-up cues aligned with courtesy/reasoning requirements (@.olaf/olaf-core/competencies/developer/templates/code-review-2-0-template.md#1-81).

Let me know if you want additional sections (e.g., custom severity labels or automation hooks) baked into the template.

Amending the created MD files



Thought for 6s

The checklist lives inside the findings template (@.olaf/olaf-core/competencies/developer/templates/code-review-2-0-template.md#67-88), so every generated report already has to check those boxes before completion. If you want the workflow itself to enforce it, we can add a matching validation step to the prompt (developer/prompts/code-review-2-0.md) so the assistant refuses to save when any checkbox is unchecked or placeholders remain. Keeping it in the template ensures reviewers see it in the final artifact, while embedding enforcement in the workflow guarantees it can't be skipped. Let me know if you'd like me to add that validation to the prompt now.

the template is only there to guide and structure the output. I don't think it's the place where rules are enforced... I may be wrong...

Thought for 13s

You're right: the template is only a structural guide. Enforcement belongs in the workflow (developer/prompts/code-review-2-0.md). Right now the Validation Phase only checks that the template "follows the structure" (@developer/prompts/code-review-2-0.md#79-84), so literal placeholders can slip through. To truly enforce the rules, we should update that prompt (or add a dedicated validation sub-step) to:

1. Scan the generated findings before saving and fail if strings like `{SeverityLabel}` or `{what_you_saw}` remain.
2. Require each finding to include the snippet + table pair (e.g., count snippet blocks and table rows and compare).
3. Confirm every compliance checkbox is marked before declaring success (@developer/templates/code-review-2-0-template.md#67-88).

If you want, I can add those checks to the prompt's Validation Phase now so enforcement happens during execution instead of relying on the template alone.

yes, add these in the prompt's validation phase

Thought for 6s

0 / 1 tasks done

Modify code-review-2-0 prompt validation to enforce snippet/table/compliance checks

code-review-2-0.md +3

1 / 1 tasks done

Thought for 9s

Validation now enforces non-empty findings:

- Prompt update: the Validation Phase in developer/prompts/code-review-2-0.md checks for placeholder strings, ensures snippet/table counts match, requires the compliance checklist to be fully checked, and still verifies template structure/timestamps before saving @.olaf/olaf-core/competencies/developer/prompts/code-review-2-0.md#79-96.
- Template reminder: Issue Log section already requires duplicating the snippet per finding with all placeholders replaced, plus the compliance checklist confirms no literal braces remain @.olaf/olaf-core/competencies/developer/templates/code-review-2-0-template.md#30-88.

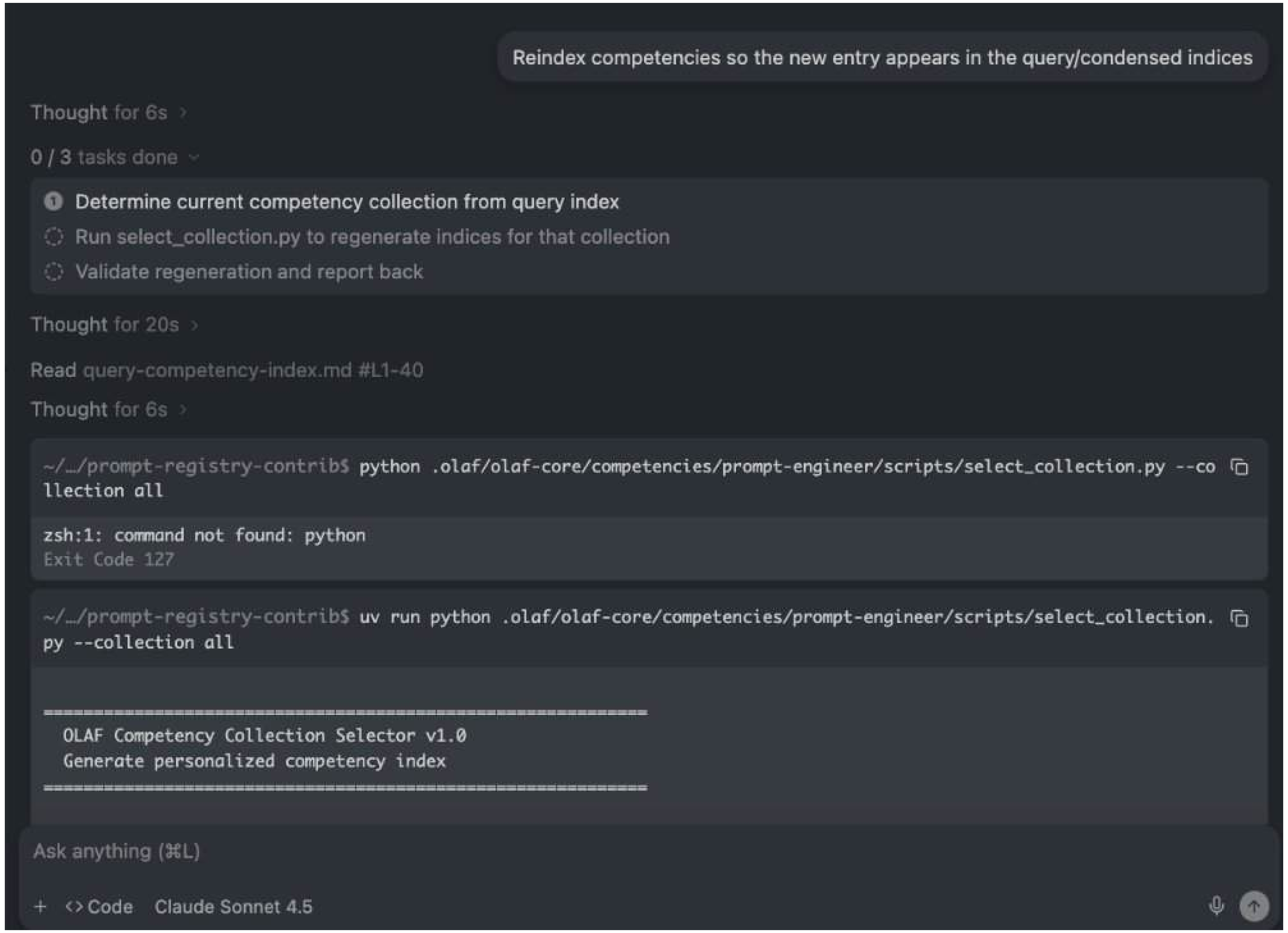
Now the workflow halts if anyone tries to copy the snippet verbatim or skip sections, while the template continues to guide structure.

Your modified files:

Creating an OLAF Competency

Competency list reindexation to make the new prompt recognized by OLAF in a new chat

(Python needed)



Reindex competencies so the new entry appears in the query/condensed indices

Thought for 6s ›

0 / 3 tasks done ⌄

- Determine current competency collection from query index
- Run select_collection.py to regenerate indices for that collection
- Validate regeneration and report back

Thought for 20s ›

Read query-competency-index.md #L1-40

Thought for 6s ›

```
~/…/prompt-registry-contrib$ python .olaf/olaf-core/competencies/prompt-engineer/scripts/select_collection.py --co
llection all
```

```
zsh:1: command not found: python
Exit Code 127
```

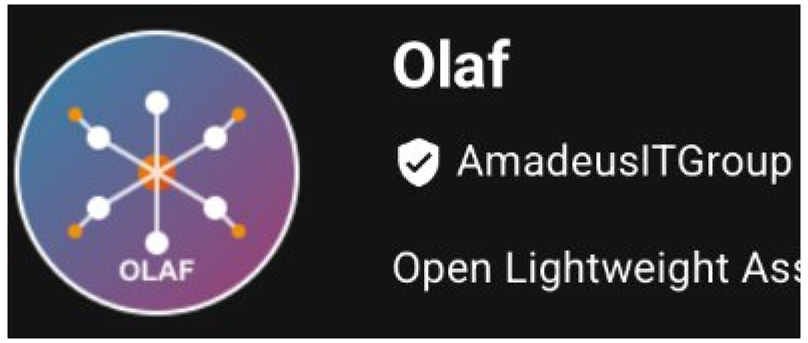
```
~/…/prompt-registry-contrib$ uv run python .olaf/olaf-core/competencies/prompt-engineer/scripts/select_collection.
py --collection all
```

```
============================================================
  OLAF Competency Collection Selector v1.0
  Generate personalized competency index
============================================================
```

CONFIDENTIAL

Sharing OLAF competencies

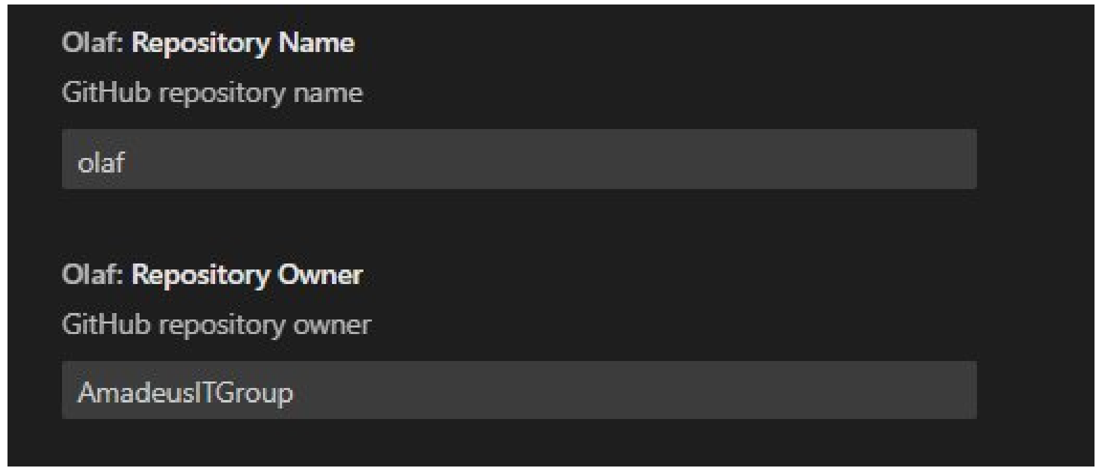
- Creating a competency locally is good, but sharing it is better
- Commit and push the created competency to your repository
- The OLAF extension can be configured to fetch OLAF from any GitHub repository



Olaf

AmadeusITGroup

Open Lightweight Ass



Olaf: **Repository Name**

GitHub repository name

olaf

Olaf: **Repository Owner**

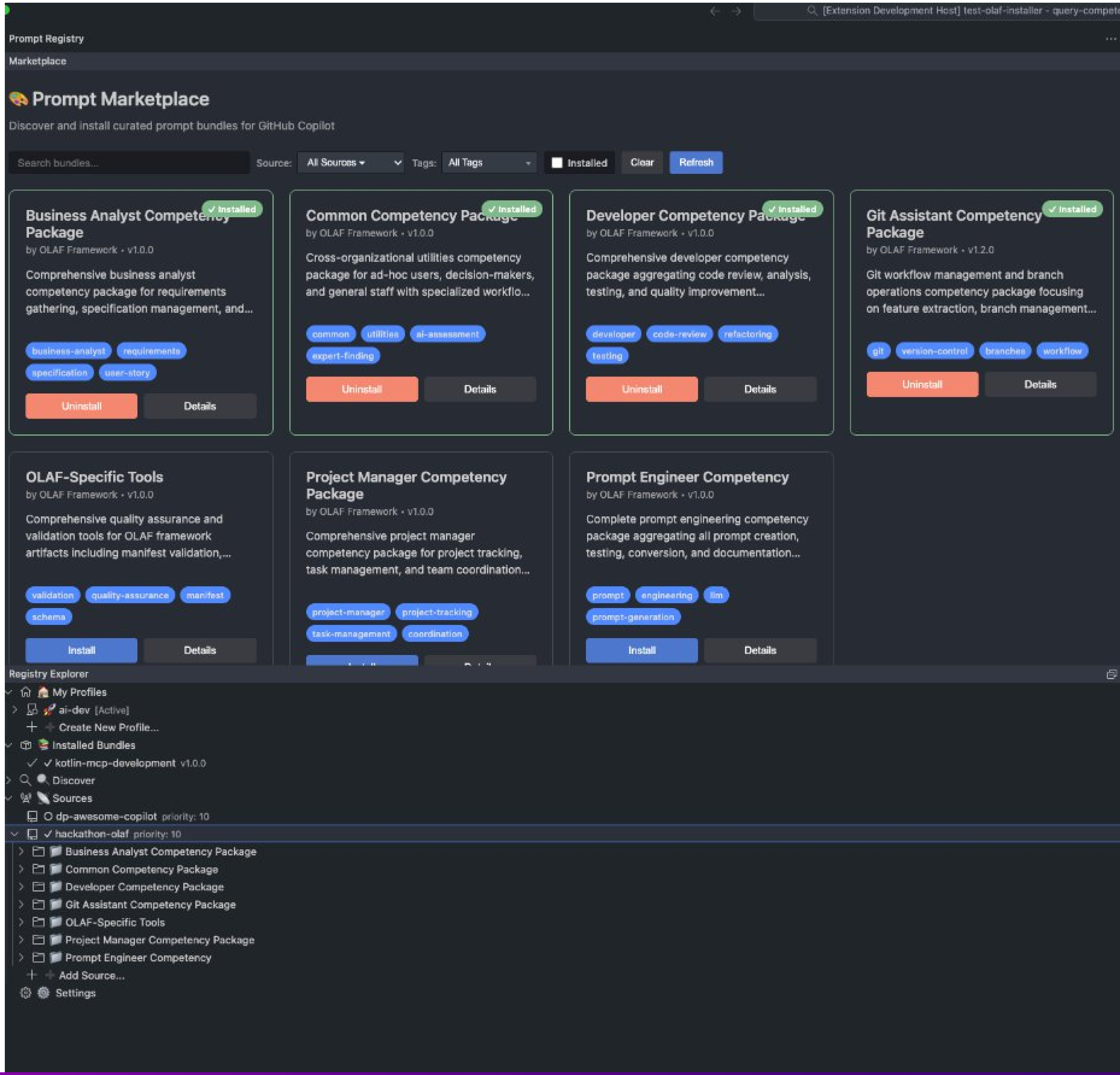
GitHub repository owner

AmadeusITGroup

CONFIDENTIAL

Next evolutions
Integration with *PromptRegistry*

- *PromptRegistry* – open source project to discover and manage your GitHub Copilot prompts
- Easy and granular installation of OLAF competencies
- Easy to contribute and share competencies across teams
- Easily add OLAF competency repositories as source



Prompt Marketplace

Discover and install curated prompt bundles for GitHub Copilot

Business Analyst Competency Package
by OLAF Framework • v1.0.0
Comprehensive business analyst competency package for requirements gathering, specification management, and...

Common Competency Package
by OLAF Framework • v1.0.0
Cross-organizational utilities competency package for ad-hoc users, decision-makers, and general staff with specialized workflo...

Developer Competency Package
by OLAF Framework • v1.0.0
Comprehensive developer competency package aggregating code review, analysis, testing, and quality improvement...

Git Assistant Competency Package
by OLAF Framework • v1.2.0
Git workflow management and branch operations competency package focusing on feature extraction, branch management...

OLAF-Specific Tools
by OLAF Framework • v1.0.0
Comprehensive quality assurance and validation tools for OLAF framework artifacts including manifest validation,...

Project Manager Competency Package
by OLAF Framework • v1.0.0
Comprehensive project manager competency package for project tracking, task management, and team coordination...

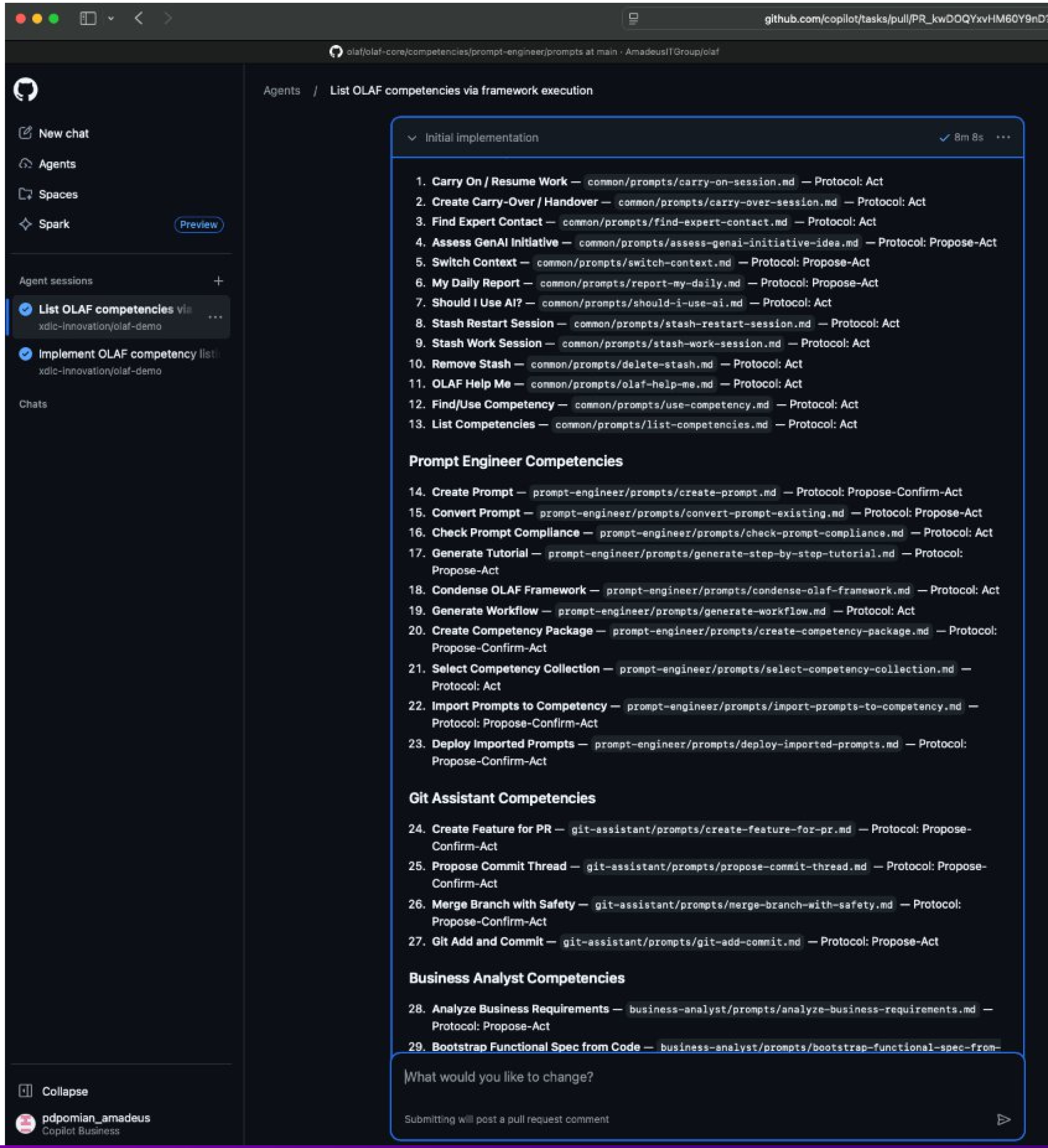
Prompt Engineer Competency
by OLAF Framework • v1.0.0
Complete prompt engineering competency package aggregating all prompt creation, testing, conversion, and documentation...

- My Profiles
  - ai-dev [Active]
  - Create New Profile...
- Installed Bundles
  - ✓ kotlin-mcp-development v1.0.0
- Discover
- Sources
  - ○ dp-awesome-copilot priority: 10
  - ✓ hackathon-olaf priority: 10
    - Business Analyst Competency Package
    - Common Competency Package
    - Developer Competency Package
    - Git Assistant Competency Package
    - OLAF-Specific Tools
    - Project Manager Competency Package
    - Prompt Engineer Competency
  - Add Source...
- Settings

CONFIDENTIAL

Next evolutions
Integration with *GH Agents*

- Trigger OLAF competencies withing GitHub Agents
- Create custom agents for various tasks directly in the CI pipeline (i.e. code review, vulnerability scans, etc.)


github.com/copilot/tasks/pull/PR_kwDOQYxvHM60Y9nD
olaf/olaf-core/competencies/prompt-engineer/prompts at main · AmadeusITGroup/olaf
New chat
Agents
Spaces
Spark
Preview
Agent sessions
List OLAF competencies via
xdic-innovation/olaf-demo
Implement OLAF competency list
xdic-innovation/olaf-demo
Chats
Collapse
pdpomian_amadeus
Copilot Business
Initial implementation
8m 8s

Agents / List OLAF competencies via framework execution

1. **Carry On / Resume Work** — `common/prompts/carry-on-session.md` — Protocol: Act
2. **Create Carry-Over / Handover** — `common/prompts/carry-over-session.md` — Protocol: Act
3. **Find Expert Contact** — `common/prompts/find-expert-contact.md` — Protocol: Act
4. **Assess GenAI Initiative** — `common/prompts/assess-genai-initiative-idea.md` — Protocol: Propose-Act
5. **Switch Context** — `common/prompts/switch-context.md` — Protocol: Propose-Act
6. **My Daily Report** — `common/prompts/report-my-daily.md` — Protocol: Propose-Act
7. **Should I Use AI?** — `common/prompts/should-i-use-ai.md` — Protocol: Act
8. **Stash Restart Session** — `common/prompts/stash-restart-session.md` — Protocol: Act
9. **Stash Work Session** — `common/prompts/stash-work-session.md` — Protocol: Act
10. **Remove Stash** — `common/prompts/delete-stash.md` — Protocol: Act
11. **OLAF Help Me** — `common/prompts/olaf-help-me.md` — Protocol: Act
12. **Find/Use Competency** — `common/prompts/use-competency.md` — Protocol: Act
13. **List Competencies** — `common/prompts/list-competencies.md` — Protocol: Act

Prompt Engineer Competencies

14. **Create Prompt** — `prompt-engineer/prompts/create-prompt.md` — Protocol: Propose-Confirm-Act
15. **Convert Prompt** — `prompt-engineer/prompts/convert-prompt-existing.md` — Protocol: Propose-Act
16. **Check Prompt Compliance** — `prompt-engineer/prompts/check-prompt-compliance.md` — Protocol: Act
17. **Generate Tutorial** — `prompt-engineer/prompts/generate-step-by-step-tutorial.md` — Protocol: Propose-Act
18. **Condense OLAF Framework** — `prompt-engineer/prompts/condense-olaf-framework.md` — Protocol: Act
19. **Generate Workflow** — `prompt-engineer/prompts/generate-workflow.md` — Protocol: Act
20. **Create Competency Package** — `prompt-engineer/prompts/create-competency-package.md` — Protocol: Propose-Confirm-Act
21. **Select Competency Collection** — `prompt-engineer/prompts/select-competency-collection.md` — Protocol: Act
22. **Import Prompts to Competency** — `prompt-engineer/prompts/import-prompts-to-competency.md` — Protocol: Propose-Confirm-Act
23. **Deploy Imported Prompts** — `prompt-engineer/prompts/deploy-imported-prompts.md` — Protocol: Propose-Confirm-Act

Git Assistant Competencies

24. **Create Feature for PR** — `git-assistant/prompts/create-feature-for-pr.md` — Protocol: Propose-Confirm-Act
25. **Propose Commit Thread** — `git-assistant/prompts/propose-commit-thread.md` — Protocol: Propose-Confirm-Act
26. **Merge Branch with Safety** — `git-assistant/prompts/merge-branch-with-safety.md` — Protocol: Propose-Confirm-Act
27. **Git Add and Commit** — `git-assistant/prompts/git-add-commit.md` — Protocol: Propose-Act

Business Analyst Competencies

28. **Analyze Business Requirements** — `business-analyst/prompts/analyze-business-requirements.md` — Protocol: Propose-Act
29. **Bootstrap Functional Spec from Code** — `business-analyst/prompts/bootstrap-functional-spec-from-`

What would you like to change?

Submitting will post a pull request comment

CONFIDENTIAL

Key takeaways

- The iterative loop of *prompt* → *answer* → *react* → *re-prompt* is powerful, but hard to manage in practice
- We struggle to express intent, constraints, and edge cases concisely
- **OLAF** is an **Open, adaptable, Lightweight prompting Assistant and Framework**
- **OLAF** helps turn *ad-hoc prompting* into **structured, repeatable workflows**
- Same prompt/workflow can run with consistent results on different agents/models

CONFIDENTIAL

Thank you!