**ITERATION 1**  
PROJECT: GitRead Agent

OBJECTIVE: Build an AI agent that reads a GitHub repo and creates a clean, structured project doc using prompt chaining and knowledge from the Learn\_AI folder.

---

CONTEXT:

Available files:

- 01\_plan.md: Design plan using DX engineering guide principles

- 02\_architecture.mmd: Mermaid diagram of GitRead system

- 03\_docs.md: Behavior spec and prompt logic

- Mermaid\_chart.svg: Visual reference

- Learn\_AI/guide-to-ai-assisted-engineering.pdf: Use this to guide meta-prompting, chaining, modularization, unit prompting, and self-reflection

---

TASKS:

You are responsible for implementing GitRead Phase 1.

Steps:

1. Accept a GitHub repo URL as input

2. Clone the repo

3. Parse README, file tree, and dependencies

4. Generate a project doc using prompt chaining:

- Prompt 1: Generate outline

- Prompts 2...N: Fill sections (overview, usage, APIs, etc.)

5. Format the output as markdown

6. Save the result to outputs/project\_doc.md

---

RECOMMENDED FOLDER STRUCTURE (can be extended or improved):

GitRead/

├── main.py # Entrypoint CLI or script

├── agents/

│ ├── repo\_cloner.py # Clones repo (gitpython)

│ ├── parser.py # Parses README, deps, structure

│ ├── doc\_planner.py # Generates outline prompt

│ ├── section\_filler.py # Fills each section from outline

│ ├── formatter.py # Converts Markdown → PDF/HTML

├── prompts/

│ ├── meta\_prompt.txt # Role: doc engineer

│ ├── outline\_prompt.txt # Prompt to generate outline

│ ├── section\_prompt.txt # Prompt template per section

├── outputs/

│ └── project\_doc.md # Final documentation output

---

RULES:

- All output must be modular

- Use prompt chaining, not monolithic prompts

- Reason per-file and per-section (unit prompting)

- Use Learn\_AI PDF to improve your strategy

---

DELIVERABLES:

1. A generated markdown doc at outputs/project\_doc.md

2. A regeneration block you can use to hand off to another agent or rerun

---

REGENERATION\_BLOCK (template):

phase: GitRead – Phase 1 Complete

next\_goal: Add test generation for uncovered functions (DX Guide §2.4)

input\_used:

- outputs/project\_doc.md

- guide-to-ai-assisted-engineering.pdf

recommended\_next\_steps:

- Add coverage detection

- Generate test plan

- Save to agents/test\_generator.py

feedback: Good modularity. Extend prompt logic. Add error handling.

---

START:

Begin with main.py and repo\_cloner.py, then implement doc\_planner.py and section\_filler.py with outline and section prompts.  
  
**ITERATION 2**  
  
PROJECT: GitRead Agent

OBJECTIVE:

Build an AI agent that reads a GitHub repository and generates a clean, structured project documentation file (`project\_doc.md`). The agent should use prompt chaining, meta-prompting, and refer to prior AI knowledge stored in the `Learn\_AI/` folder.

---

AVAILABLE RESOURCES:

- Project Docs/01\_plan.md → Design blueprint (DX-style)

- Project Docs/02\_architecture.mmd → Mermaid system diagram

- Project Docs/03\_docs.md → Prompt formats and behavior guide

- Project Docs/Mermaid\_chart.svg → Visual logic reference

- Learn\_AI/guide-to-ai-assisted-engineering.pdf → Meta-prompting, chaining, test generation, scaffold philosophy

---

PHASE 1 GOAL:

1. Input GitHub URL

2. Clone repo using GitPython

3. Parse structure: README, dependencies, folder tree

4. Chain prompts to:

- Prompt 1: Generate outline

- Prompt 2-N: Fill doc sections (overview, APIs, setup, env vars, etc.)

5. Format final output (markdown, optional PDF)

6. Save result to: `outputs/project\_doc.md`

---

RULES:

- All code must be modular: separate files per task (see architecture below)

- Follow DX best practices: use meta-prompting, unit prompting, structured chaining

- Use `Learn\_AI/` content to inform prompt strategy and self-improvement

- Do not rely on a single monolithic prompt or file

---

RECOMMENDED FOLDER STRUCTURE:

GitRead/

├── main.py

├── agents/

│ ├── repo\_cloner.py # Uses GitPython to clone

│ ├── parser.py # Parses README, dependencies, structure

│ ├── doc\_planner.py # Generates outline from repo

│ ├── section\_filler.py # Fills sections using outline + chained prompts

│ ├── formatter.py # Converts markdown to PDF/HTML

├── prompts/

│ ├── meta\_prompt.txt

│ ├── outline\_prompt.txt

│ ├── section\_prompt.txt

├── outputs/

│ └── project\_doc.md

├── Learn\_AI/

│ └── guide-to-ai-assisted-engineering.pdf

├── Project Docs/

│ ├── 01\_plan.md

│ ├── 02\_architecture.mmd

│ ├── 03\_docs.md

│ ├── Mermaid\_chart.svg

---

DELIVERABLES:

1. `outputs/project\_doc.md` – clean, structured project documentation

2. Regeneration block for continued iterations or downstream agents

---

REGENERATION\_BLOCK (based on project\_doc.md):

phase: GitRead – Phase 1 Complete

status: ✅ Documentation successfully generated at `outputs/project\_doc.md`

source\_doc: outputs/project\_doc.md

context\_used:

- parsed code structure and README

- LLM-generated outline and sections

- agent behavior defined in 03\_docs.md

- prompting strategy inspired by guide-to-ai-assisted-engineering.pdf

recommended\_next\_goal: Add test generation and coverage detection (Guide §2.4)

next\_steps:

- Read `project\_doc.md` to identify key functions/modules

- Detect which are uncovered by tests

- Generate `test\_generator.py` to suggest or create basic unit tests

- Consider auto-detecting framework (pytest, unittest) from repo

- Add diagram generator if system structure is complex

output\_targets:

- agents/test\_generator.py

- outputs/test\_coverage.md

- outputs/enhanced\_project\_doc.md (includes tests + diagrams)

agent\_behavior\_upgrade:

- Use past project docs as memory context

- Improve prompt selection using themes in `guide-to-ai-assisted-engineering.pdf`

- Add fallback prompt logic for under-specified repos

---

START:

Begin by creating `main.py` and `agents/repo\_cloner.py`.

Then implement prompt chaining with `doc\_planner.py` → `section\_filler.py`.

Use `Learn\_AI/` and past `project\_doc.md` as guidance to iterate intelligently.

**ITERATION 3**

PROJECT: GitRead Agent v2 – DX-Aligned Upgrade

OBJECTIVE:

Expand GitRead to maximize documentation accuracy, flexibility, and maintainability. Apply best practices from guide-to-ai-assisted-engineering.pdf:

- Meta-prompting (explicit agent roles)

- Prompt chaining (outline → section → review → regenerate)

- Modular files (one task per agent/script)

- Unit prompting and test generation

- Regeneration block for downstream handoff

---

RESOURCES:

- Project Docs/01\_plan.md

- Project Docs/02\_architecture.mmd

- Project Docs/03\_docs.md

- Project Docs/Mermaid\_chart.svg

- Learn\_AI/guide-to-ai-assisted-engineering.pdf

- outputs/project\_doc.md

- outputs/regeneration\_block.md

---

PHASE 2 GOALS:

1. Improve prompt clarity with meta-prompts in each agent/script.

2. Use prompt chaining for doc generation and review.

3. Add agents/test\_generator.py for auto-generating and validating tests.

4. Add agents/review\_agent.py for output review, critique, and regeneration block management.

5. Update prompts/:

- Add system\_prompt.txt (overall rules)

- Add review\_prompt.txt (for critiquing/improving outputs)

6. Regeneration block: outputs/regeneration\_block.md

- Log what changed, TODOs, recommended next steps, and key outputs for next run.

---

FOLDER STRUCTURE (update as needed):

GitRead/

├── main.py

├── agents/

│ ├── repo\_cloner.py

│ ├── parser.py

│ ├── doc\_planner.py

│ ├── section\_filler.py

│ ├── formatter.py

│ ├── test\_generator.py # NEW

│ ├── review\_agent.py # NEW

├── prompts/

│ ├── meta\_prompt.txt

│ ├── outline\_prompt.txt

│ ├── section\_prompt.txt

│ ├── system\_prompt.txt # NEW

│ ├── review\_prompt.txt # NEW

├── outputs/

│ ├── project\_doc.md

│ ├── regeneration\_block.md # NEW

├── Learn\_AI/

│ └── guide-to-ai-assisted-engineering.pdf

├── Project Docs/

│ ├── 01\_plan.md

│ ├── 02\_architecture.mmd

│ ├── 03\_docs.md

│ ├── Mermaid\_chart.svg

---

REGENERATION\_BLOCK EXAMPLE:

phase: GitRead v2 - Iteration Complete

changes\_this\_pass:

- Meta-prompts added to all agent scripts

- Test generator enabled, creates unit tests

- Output review/critique agent run

- Regeneration block tracks agent outputs for next iteration

errors\_or\_todos:

- PDF/HTML export in progress

- Voice/image input not implemented

recommended\_next\_steps:

- Extend test\_generator for integration tests

- Enable adversarial model runs for critical outputs

- Refine system prompts based on feedback

inputs\_used:

- outputs/project\_doc.md

- outputs/regeneration\_block.md

- Learn\_AI/guide-to-ai-assisted-engineering.pdf

---

IMPLEMENTATION RULES:

- All agents/scripts must follow the guide-to-ai-assisted-engineering.pdf principles (meta-prompting, modularity, regeneration loops)

- Regeneration block must be written at every major output

- All prompts should be clear, modular, and specific

- Feedback/critique agent must be used after doc generation, before next phase

---

START:

1. Create agents/test\_generator.py and agents/review\_agent.py.

2. Add prompts/system\_prompt.txt and prompts/review\_prompt.txt (use DX guide language).

3. Run pipeline on a test repo. Save regeneration\_block.md and pass to next agent or team member.