Agentic Project Management

Manage complex projects with a team of AI Assistants, smoothly and efficiently.

Quick Start Guide for APM v0.4 - August 2025

Documentation Suite

CobuterMan

Contents

1 What is APM?

Agentic Project Management (APM) is a structured multi-agent workflow for managing complex projects in AI IDE environments.

APM uses chat sessions in your AI IDE as separate agent instances, each with its own context scope and memory. This enables more focused interactions, reduces the cognitive load on any single agent, and produces more consistent results.

1.1 The Problem APM Solves

Managing large projects with AI assistants presents systematic challenges. Extended conversations frequently lead to context degradation where the AI loses track of original requirements, produces contradictory suggestions, or generates inaccurate details.

These issues arise from fundamental limitations of LLMs: **Context Window Limits**.

This constraint feels "heavier" within AI IDEs, when often times Context Windows are shrunk even further to maintain profitable interactions with the model's provider. As conversations grow, the AI struggles to keep track of everything, leading to confusion, errors, and wasted time.

1.2 APM's Approach

The framework uses multiple AI agent instances, each with a specific role and clear responsibilities, coordinated through structured protocols and persistent memory management. The result is a workflow that feels more like working with a well organized team than wrestling with a single overloaded AI assistant.

> **Real World Analogy:** Think of APM like running a software development team. You have a project manager who understands the big picture, developers who focus on specific tasks, and clear documentation that keeps everyone aligned. Developers document each task execution, and the manager reviews the logs for coordination. The difference is that your "team members" are AI assistants in separate chat sessions.

1.3 About this Guide

This Quick Start Guide is designed to get you through your first complete APM session successfully. It assumes no prior APM knowledge and focuses on immediate action rather than comprehensive understanding.

For detailed optimization strategies, troubleshooting, and in-depth explanations, refer to the **User Guide** after completing your first session.

> **Note on Screenshots:** The screenshots in this quick start guide are taken from APM sessions in Cursor. However, most modern AI IDEs offer similar interfaces and workflows, so you can easily follow these steps in your preferred environment.

2 Prerequisites

Before starting your first APM session, ensure you have:

- **AI IDE Platform:** Access to an AI IDE supporting multiple chat sessions with tool access (Cursor, Windsurf, VS Code with AI extensions, etc.)

- **APM Assets:** Framework prompts and guides (see access options below)

- **Project Workspace:** Dedicated directory for your project files

2.1 Setting up APM Access

To access the APM framework assets, choose one of the following three options:

- **Option A: Clone the Upstream Repository (Default)**
  Clone the official APM repository directly from GitHub. This provides the latest, unmodified version of the APM assets.

  ```
  git clone https://github.com/sdi2200262/agentic-project-management
  ```

- **Option B: Use as Template (Recommended for Customization)**
   Use GitHub's "Use this template" feature to create your own copy of the APM repository. This allows you to customize prompts, guides, and workflows for your specific use case.

  ```
  git clone https://github.com/your-username/your-custom-apm-template
  ```

  > **Customization Tip:** Using a template enables you to tailor APM prompts, guides, and workflows for your project's or organization's needs. See Modifying APM in the documentation for practical customization strategies and examples.
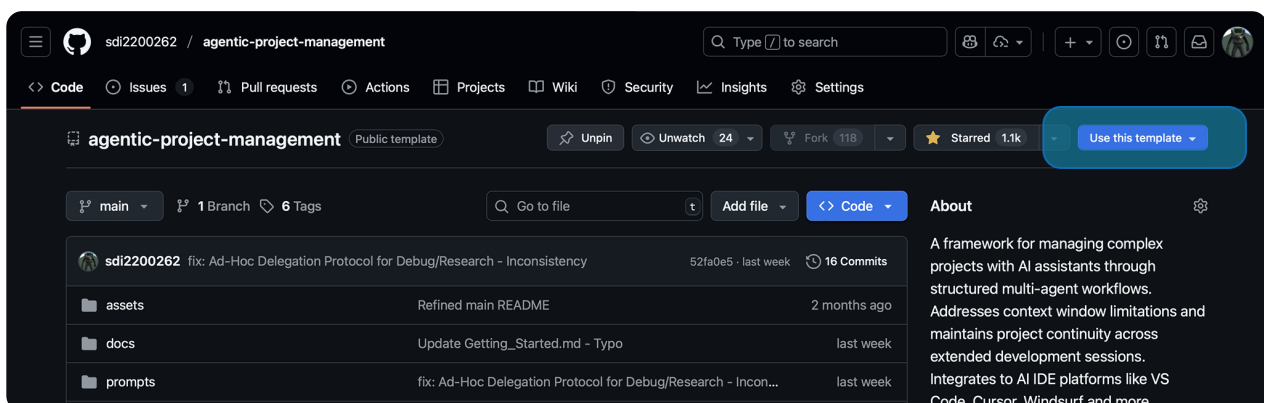

Figure 1: GitHub's "Use this template" feature - top right corner.

- **Option C: Manual Access - Not Recommended**
  Manually copy and paste prompt and guide contents from GitHub or other sources as needed. This is not recommended, as it undermines agentic workflow efficiency.

2.2 Adding APM Assets to Your Working Environment

Once you have chosen your preferred access method, you can integrate the APM assets into your working environment in several ways:

1. **Clone APM inside your project's directory**
   Clone the APM upstream or your custom template repository inside your project's directory. This creates an `apm/` directory at your project root for all APM assets.

   > **Git Implications:** You may wish to update your `.gitignore` to exclude the APM assets and `apm/` directory. If you prefer to track session progress, you can integrate commits in your task specifications.

2. **Set up a parent directory containing both your project and APM**
   Create a parent directory that will contain both your project's directory and the APM repo clone alongside it. **This avoids git implications within your project repository and keeps the `apm/` directory with session assets at the parent directory root.**
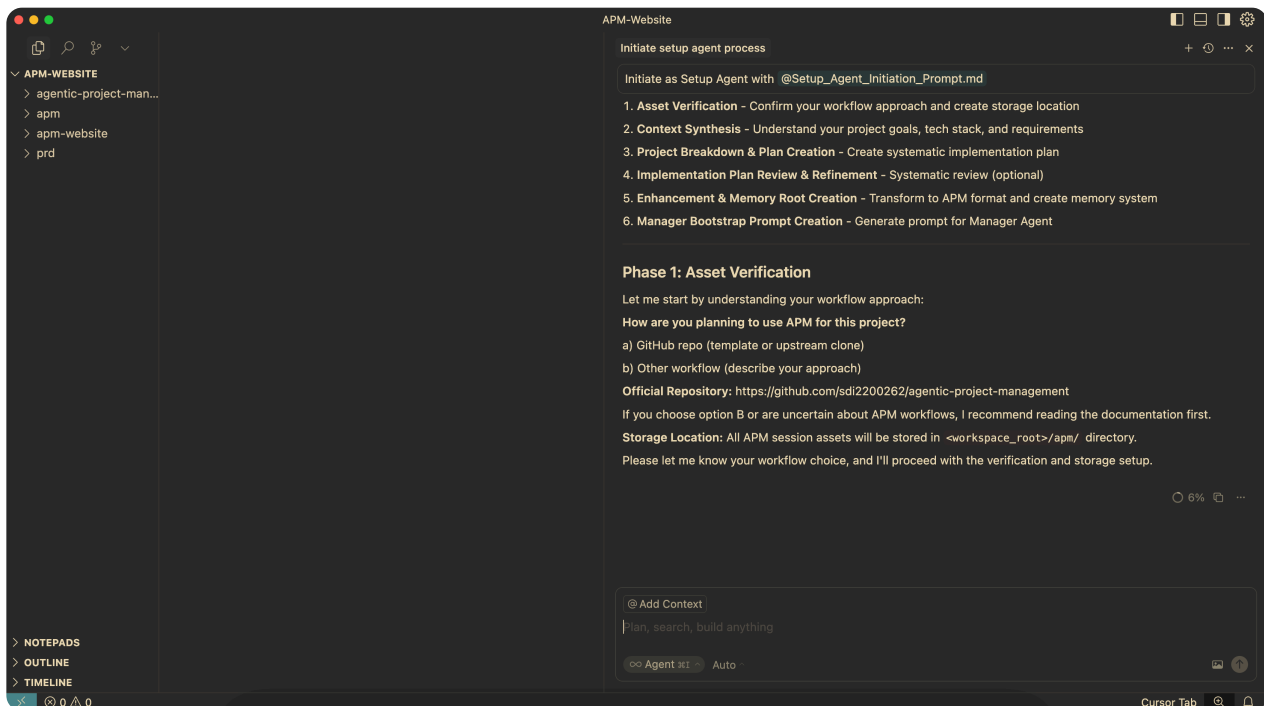
Figure 2: APM assets in a parent directory - bottom right corner.

3. **Clone APM in a shared location and add to your IDE workspace**
   Clone the APM directory in a location of your choice and add it to your IDE's workspace alongside your project. **This allows you to use the same APM assets across multiple projects by cloning it only once.** If you don't have project-specific assets, this is the 'tidiest' option.

   > **Codebase Indexing Implications:** This approach may have implications for codebase indexing in your IDE, as the `apm/` directory has no standard "root" to be stored in. If you use this method, be specific about where you want assets to be stored and how you intend to store the APM session assets within your workflow.

2.3 Agent Types and Model Selection

In APM, each chat session in your AI IDE serves as a dedicated agent instance, with its own context and memory. Each agent type has a clearly defined role and responsibility within the framework, allowing you to select models balancing performance and cost.

- **Setup Agent:** Initializes your session through project discovery, creates the Implementation Plan and Memory System, then hands off to the Manager Agent.

  Use **Claude Sonnet 4** for its breakdown, reasoning and agentic capabilities; if unavailable, choose premium "non-thinking" models with strong agentic features.

- **Manager Agent:** Coordinates the session, assigns tasks, reviews work, and makes project decisions.

  Prefer **Claude Sonnet 4** for advanced reasoning and coordination; reliable budget alternatives include Cursor Auto or Sonnet 3.7.

- **Implementation Agents:** Execute specific tasks in focused domains (e.g., Frontend, Backend), logging all work in the Memory System.

  For complex tasks, use premium models **Claude Sonnet 4, Gemini 2.5 Pro, or GPT-5**; for routine work, platform base models like Cursor Auto, Windsurf SWE-1, or Copilot GPT-4.1 are cost-effective and consistent.

- **Ad-Hoc Agents:** Temporary agents for isolated, context-intensive work (debugging, research, analysis etc.).

  Match model to task complexity: premium models for systemic debugging, mid-tier for web research or simple fixes.

> **Testing Note:** Claude Sonnet 4 delivered overall best performance during testing (August 2025). However, platform base models such as Cursor Auto, GitHub Copilot's GPT 4.1, and Windsurf's SWE 1 proved to be reliable and cost-effective alternatives for Manager and Implementation Agents.
>
> For practical strategies on model selection and optimizing token usage, see Token Consumption Tips in the documentation.

> **Model Switching Implications:** Switching models mid-session can lead to context loss or inconsistent behavior due to disruptancies in token caching. **For your first APM session, it is recommended that you stick with a single model throughout the entire session, preferably Claude Sonnet 4** if available.

3 Setup Phase - Initialize your first Setup Agent

The Setup Agent conducts comprehensive project discovery, decomposes the project into manageable tasks, and creates all necessary APM session assets. At each step of the Setup Phase, the Setup Agent will pause for your confirmation, allowing you to review, clarify, or request changes before proceeding. **This ensures you can iterate as needed before moving to the next step.**

1. **Create Setup Agent Session:** Open a new chat session in your AI IDE and name it clearly (e.g., "Setup Agent").

2. **Initialize Setup Agent:** Provide the full initiation prompt to the Setup Agent located at

   ```
   prompts/Setup_Agent/Setup_Agent_Initiation_Prompt.md
   ```

   via copy-paste, file upload, or your IDE's context tools, along with a clear instruction such as:

   ```
   Initialize as Setup Agent
   ```


Figure 3: Initializing a Setup Agent chat session in Cursor
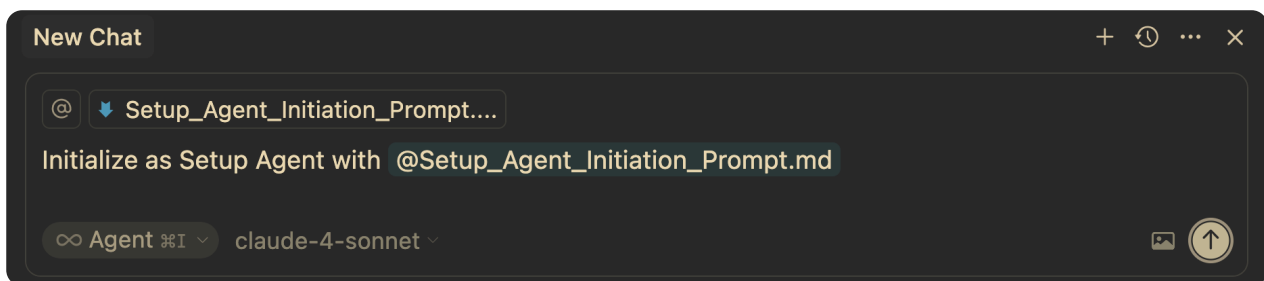
3.1 Asset Verification

The Setup Agent will state its six-step sequence and then ask how you plan to use APM for this project. Respond with your chosen access method: either **Option A** (GitHub template or upstream clone), or **Option B** (other workflow; describe your approach).

Figure 4: Setup Agent asking about workflow approach and creating apm/ directory

3.2 Context Synthesis

The Setup Agent guides you through structured project discovery across four phases:

- **Phase 1:** Existing materials, PRDs, project vision and scope
- **Phase 2:** Technical requirements, dependencies, constraints
- **Phase 3:** Process requirements, workflow preferences, quality standards
- **Phase 4:** Agent summarizes findings for user validation and prompts for preferred asset format (Markdown or JSON)

> **Context Synthesis Tip:** Timely input ensures the Setup Agent's project discovery works optimally, so provide all relevant information at the right phase and be prepared in advance. If you forget something, see Omitted Requirements During Context Synthesis.

Please proceed to Context Synthesis.

2. Context Synthesis Phase

Read Context_Synthesis_Prompt.md

Now I'll conduct the guided Context Synthesis Q&A to achieve a complete contextual understanding of your project and its requirements.

Phase 1: Existing Material and Vision (ITERATIVE)

Let me start by understanding your project foundation:

1. **What type of deliverable(s) are you creating?** (document, analysis, codebase, dataset, presentation, etc.)

2. **Do you have existing materials** such as PRD, requirements specs, user stories, roadmaps, architecture diagrams, code, research sources, or templates that I should review?

3. **What is your current plan or vision** for this project?

4. **Since I can see you have a `prd/` directory with Design-PRD.md and Implementation-PRD.md files** – are these the key documents that define your project requirements and scope?

Read Design-PRD.md

Read Implementation-PRD.md

I can see you have comprehensive PRD documents that define both the design specifications and implementation requirements for your APM Landing Page project. Let me continue with the Context Synthesis process to understand your specific needs and preferences.
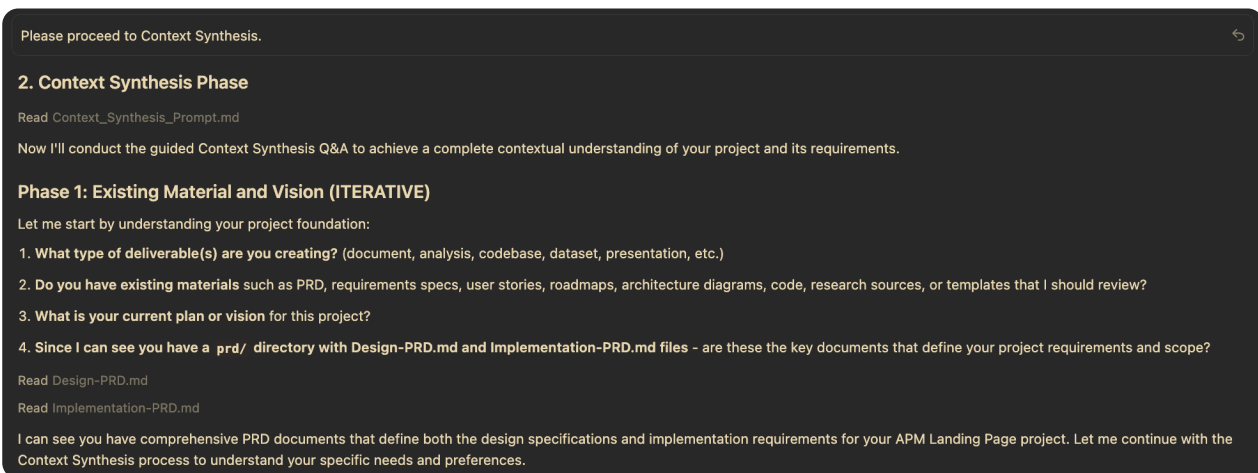
Figure 5: Setup Agent completing Phase 1 of Context Synthesis reading existing project materials

3.3 Project Breakdown

The Setup Agent transforms your project's requirements as gathered during Context Synthesis, into a structured Implementation Plan using systematic reasoning. The plan consists of phases, tasks, subtasks and agent assignments for your entire project.

> **Implementation Plan Review Point:** After Project Breakdown completes, carefully review the Implementation Plan. Request corrections or modifications for any mismatches with your requirements. This prevents costly downstream changes.

> **Project Breakdown Interruptions:** The Setup Agent's Project Breakdown sequence is engineered for reliable, systematic reasoning. On some platforms, interruptions may occur due to system prompts or model limitations, resulting in incomplete or fragmented breakdowns. For your first session, use **Claude Sonnet 4** as the Setup Agent to ensure the Project Breakdown sequence completes smoothly and without interruption.
>
> If you encounter issues, refer to the troubleshooting section in the User Guide for recovery.

3.4 Implementation Plan Review (Optional)

The Setup Phase includes this optional step, to help ensure the Implementation Plan is complete and accurate. If you opt for a systematic review, the Setup Agent highlights plan sections possibly needing extra attention for you to select. The user-selected sections are then analyzed and reviewed by the Setup Agent.

The agent-driven review targets AI-specific planning issues like task-packing, misclassified tasks, LLM errors, **not full project context**.

You should always conduct your own comprehensive review of the Implementation Plan instead of relying on the Agent's review.

> **Agent-Driven Review Tip:** Use this agent-driven review in your first APM session to help catch issues early. You'll learn to spot similar problems yourself in future sessions.

3.5 Enhancement & Memory Initialization

In this step, the Setup Agent transforms the reviewed Implementation Plan into a detailed APM artifact and initializes the Memory System accordingly.
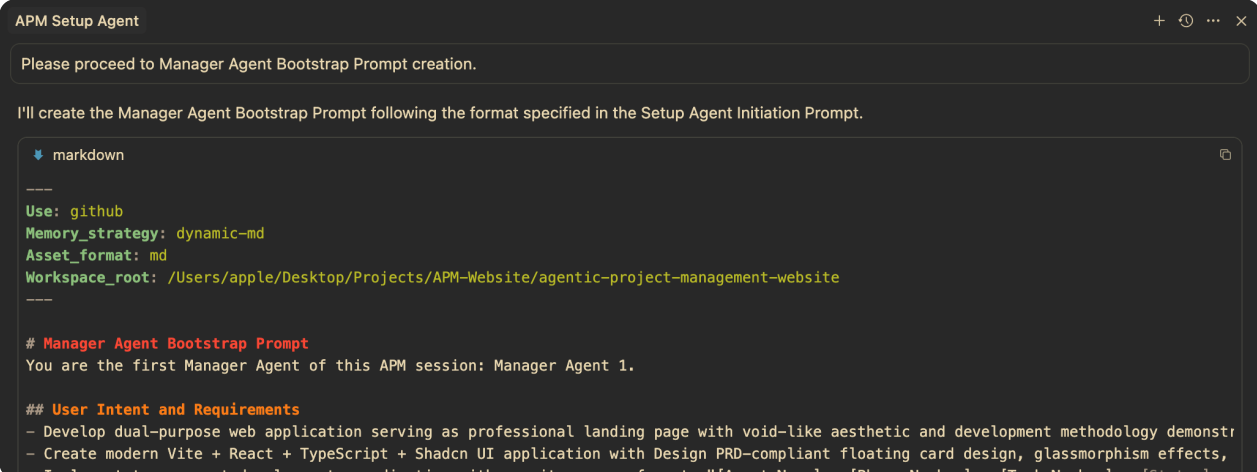
The transformation is completed in a single, structured response converting one phase at a time, while the Memory System is initialized based on the project complexity.

> **Implementation Plan Enhancement Warning:** Like Project Breakdown, this sequence may be interrupted for similar reasons. See the related troubleshooting section for recovery.

3.6 Manager Bootstrap Creation

At the final step of the Setup Phase, the Setup Agent generates the Bootstrap Prompt containing all key information for the Manager Agent to start the Task Loop Phase.

This Bootstrap Prompt is presented as a markdown code block for easy copy-paste. Store this prompt for use in the first Manager Agent initialization.


```markdown
---
Use: github
Memory_strategy: dynamic-md
Asset_format: md
Workspace_root: /Users/apple/Desktop/Projects/APM-Website/agentic-project-management-website
---

# Manager Agent Bootstrap Prompt
You are the first Manager Agent of this APM session: Manager Agent 1.

## User Intent and Requirements
- Develop dual-purpose web application serving as professional landing page with void-like aesthetic and development methodology demonstr
- Create modern Vite + React + TypeScript + Shadcn UI application with Design PRD-compliant floating card design, glassmorphism effects,
- Implement transparent development coordination with commit message format: "[Agent Name] — [Phase Number] — [Task Number] — [Status] —
```

Figure 6: Setup Agent presenting a Bootstrap Prompt in a markdown code block

4 Task Loop Phase - Initialize your first Manager & Implementation Agents

After Setup Phase completion, you'll work with Manager and Implementation Agents through the Task Loop Phase, coordinating work execution while maintaining clear documentation of progress and issues.

4.1 Initialize your first Manager Agent

The Manager Agent coordinates project execution, assigns tasks to Implementation Agents, and manages the Memory System throughout the Task Loop Phase.

1. **Create Manager Agent Session:** Open a new chat session in your AI IDE and name it clearly (e.g., "Manager Agent 1").

2. **Initialize Manager Agent:** Provide the full initiation prompt to the Manager Agent located at
   ```
   prompts/Manager_Agent/Manager_Agent_Initiation_Prompt.md
   ```
   via copy-paste, file upload, or your IDE's context tools, along with a clear instruction such as:
   ```
   Initialize as Manager Agent
   ```

3. **Deliver Bootstrap Prompt:** The Manager Agent will request a Bootstrap Prompt or a Handover Prompt. Paste the Bootstrap Prompt created by your Setup Agent as this is the first Manager Agent of the session.
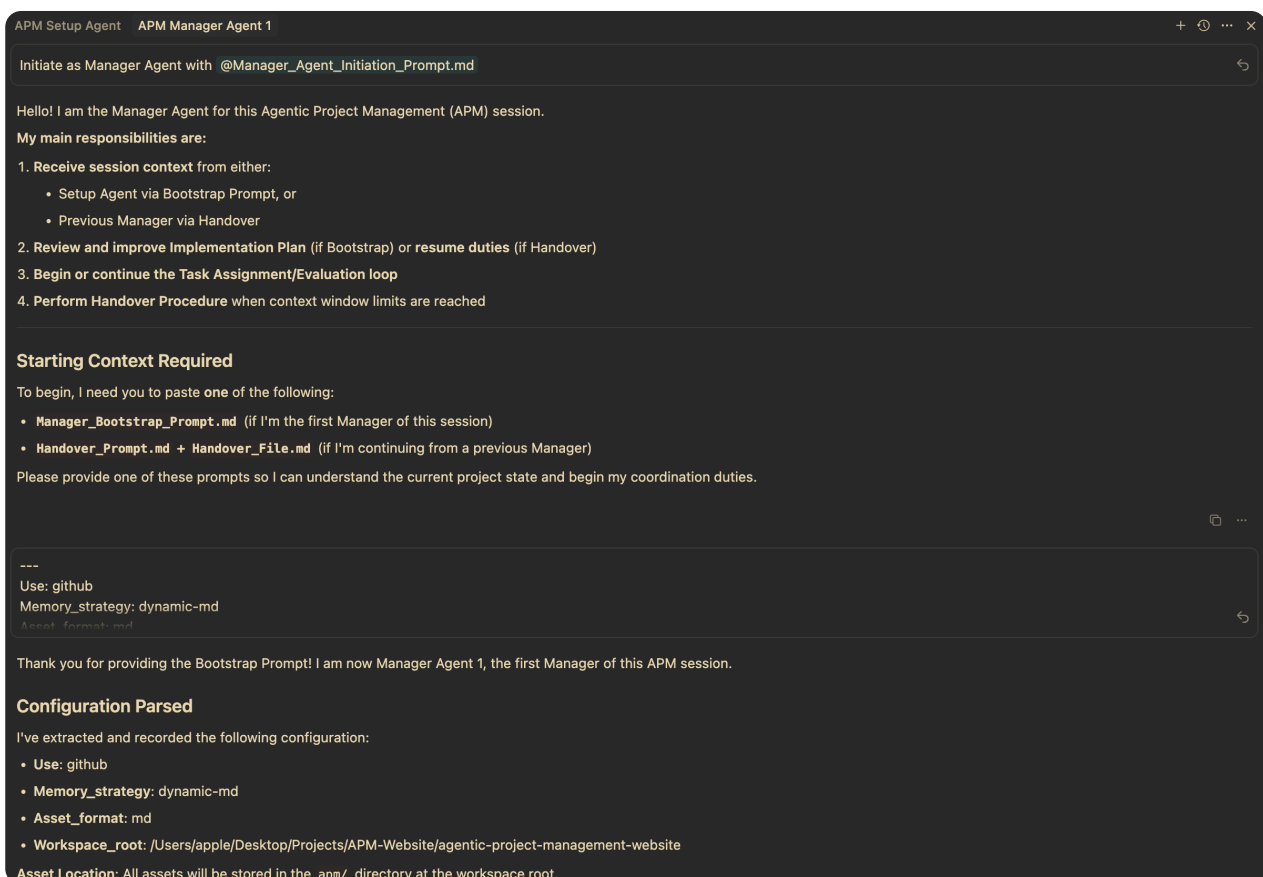

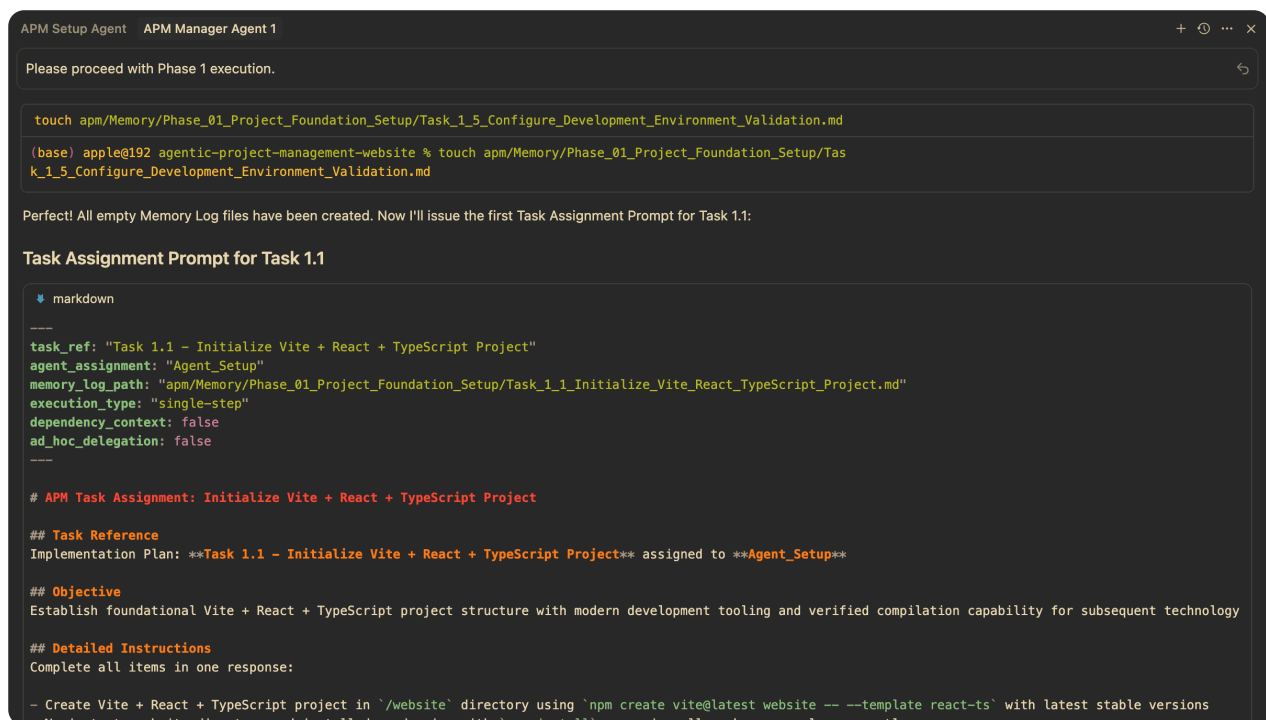Figure 7: Manager Agent initializing and processing Bootstrap Prompt

4.1.1 Bootstrap Prompt Processing

Upon reading the Bootstrap Prompt, the Manager Agent will:

- **Read Required Guides:** to understand the Manager Agent duties and responsibilities.

- **Read Implementation Plan & Memory System Root:** to understand the project plan and memory system strategy.

- **Present a Responsibility Summary:** to the User to confirm the initialization is complete.

- **Ask Approval:** to commence coordination duties.

4.1.2 First Task Assignment

Upon receiving approval, the Manager Agent will begin Phase 1 execution, initializing the Memory System for this phase and issuing **the first Task Assignment Prompt**.

Figure 8: Manager Agent beginning Phase 1 execution and issuing the first task assignment

4.1.3 Task Assignment Prompts

Task assignments are **presented as a markdown code block** for easy copy-paste. Each Task Assignment Prompt includes:

- **YAML Front matter:** Execution type and dependency flags

- **Task Reference:** Clear identification from Implementation Plan

- **Objectives:** One sentence summary of the task

- **Detailed Instructions:** Step-by-step guidance based on Implementation Plan

- **Expected Outputs:** Specific deliverables and success criteria

- **Memory Logging Instructions:** Where and how to record the work

4.2 Implementation Agent Initialization and Execution

Right after receiving your first Task Assignment Prompt, you will need to create a new chat session for the assigned Implementation Agent. You will have multiple Implementation Agents to manage, so you will need to create a new session for each one.

- **Create Implementation Agent Session:** Open a new chat session in your AI IDE and name it clearly using the assigned agent name from the Task Assignment Prompt (e.g., "Agent_Frontend").

- **Initialize Implementation Agent:** Provide the full initiation prompt to the Implementation Agent located at

```
prompts/Implementation_Agent/Implementation_Agent_Initiation_Prompt.md
```

  via copy-paste, file upload, or your IDE's context tools, along with a clear instruction such as:
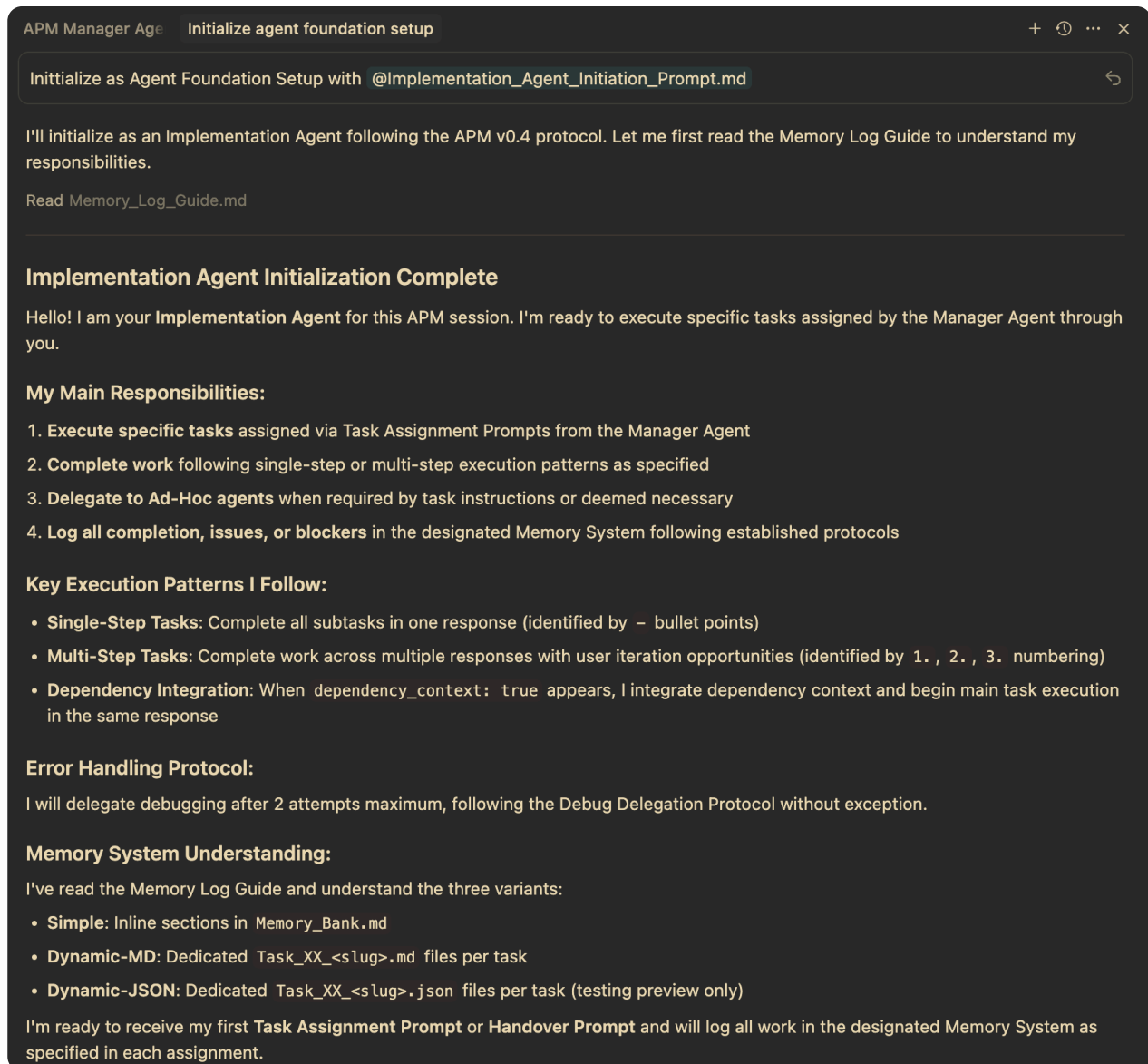
```
Initialize as Implementation Agent
```


Figure 9: Implementation Agent initialization

4.2.1 Task Execution

Implementation Agents execute tasks as per the Task Assignment Prompt, logging all work in the Memory System. A task execution is complete when the agent has delivered all expected outputs and logged the work as instructed. Tasks are classified into two types:

Single-Step Tasks: Implementation agents complete all subtasks and deliverables in one focused response for granular, well-scoped tasks, then directly log to Memory.

Multi-Step Tasks: For complex tasks with sequential dependencies, agents execute steps with user confirmation at each stage, allowing iteration and modifications before logging to Memory.

> **Step Combination Efficiency:** For straightforward or low-risk tasks, you can combine adjacent steps of multi-step tasks to minimize confirmation overhead and save tokens. For complex or high-risk work, keep steps separate to allow for thorough validation.
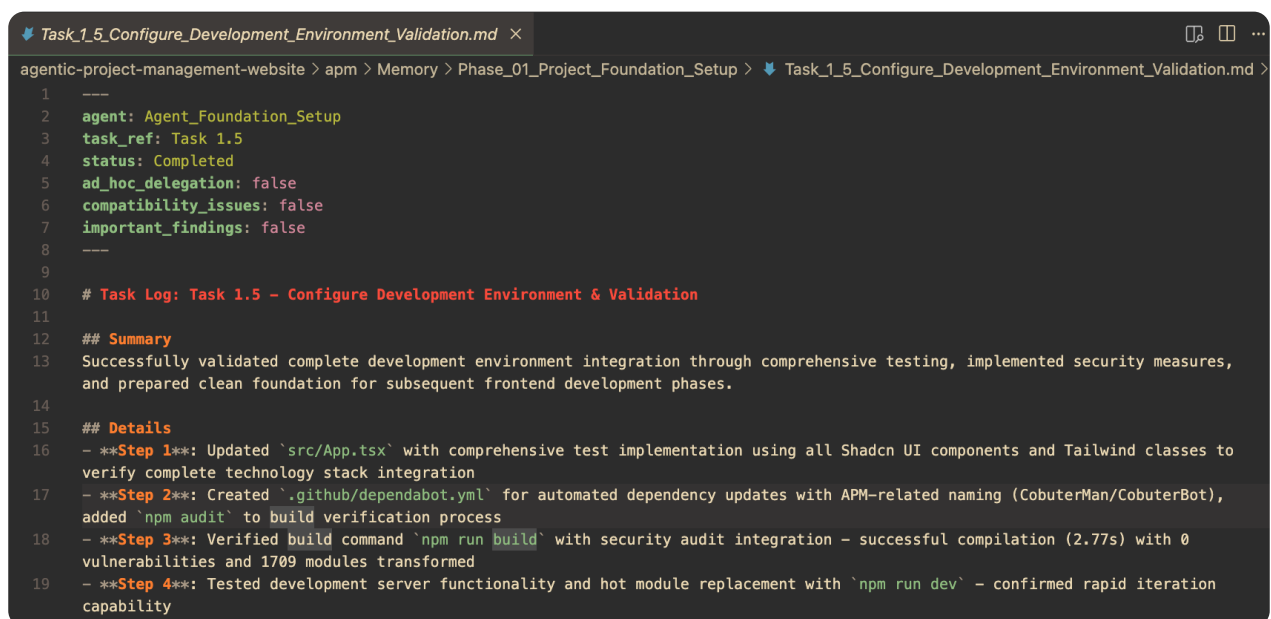>
> ```
> "Step 2 looks good. Please combine steps 3 and 4 in your next response."
> ```

4.2.2 Memory Logging

After task execution, the Implementation Agent will report completion status and outputs to the User, then request confirmation to log (or log immediately if the task was successful).

Memory Logs are appended inline for single-file Memory Banks, or saved as structured markdown files for Dynamic Memory Banks. Their structure includes:

- **YAML Front Matter:** Agent ID, Task ID, completion status, execution flags.

- **Execution Summary:** Brief summary of task execution.

- **Outputs and Deliverables:** List of outputs and deliverables.

- **Issues Encountered:** Summary of any issues during execution.

- **Next Steps:** Planned next actions.


```
Task_1_5_Configure_Development_Environment_Validation.md  ×

agentic-project-management-website > apm > Memory > Phase_01_Project_Foundation_Setup >  Task_1_5_Configure_Development_Environment_Validation.md >
  1    ---
  2    agent: Agent_Foundation_Setup
  3    task_ref: Task 1.5
  4    status: Completed
  5    ad_hoc_delegation: false
  6    compatibility_issues: false
  7    important_findings: false
  8    ---
  9
 10    # Task Log: Task 1.5 – Configure Development Environment & Validation
 11
 12    ## Summary
 13    Successfully validated complete development environment integration through comprehensive testing, implemented security measures,
       and prepared clean foundation for subsequent frontend development phases.
 14
 15    ## Details
 16    - **Step 1**: Updated `src/App.tsx` with comprehensive test implementation using all Shadcn UI components and Tailwind classes to
       verify complete technology stack integration
 17    - **Step 2**: Created `.github/dependabot.yml` for automated dependency updates with APM-related naming (CobuterMan/CobuterBot),
       added `npm audit` to build verification process
 18    - **Step 3**: Verified build command `npm run build` with security audit integration – successful compilation (2.77s) with 0
       vulnerabilities and 1709 modules transformed
 19    - **Step 4**: Tested development server functionality and hot module replacement with `npm run dev` – confirmed rapid iteration
       capability
```

Figure 10: Memory Log file entry after a multi-step task execution

4.2.3 Manager Review and Next Steps

After the Implementation Agent has logged the task execution results to the Memory System, you will return to the Manager Agent session for the review of the Memory Log and determination of the next action.

Provide clear instructions to the Manager Agent and additional context if necessary:

```
Task [X.Y] was completed. Please review the Memory Log and proceed accordingly.
```

The Manager Agent will:

- Review the Memory Log to receive context on the task execution

- Determine next action:

  – **Continue** with next task in the Implementation Plan
  – **Request corrections** via follow-up task assignment to the same agent
  – **Update the Implementation Plan** accordingly and continue with the next revised task

APM Manager Agent 1   Initialize agent fou   + ⊙ ··· ✕

Task 1.5 was completed, please review the log and proceed accordingly.   ↶

I'll review the Memory Log for Task 1.5 to assess completion status and proceed accordingly.

Read Task_1_5_Configure_Development_Environment_Validation.md

✅ **Task 1.5 Review Complete - Phase 1 Complete!**

Task Review Summary:
- **Status:** Completed Successfully ✅
- **All deliverables met:** Complete development environment validation with security measures implemented
- **Integration Testing:** Comprehensive test interface with all Shadcn UI components and Tailwind classes
- **Security Configuration:** Dependabot setup, npm audit integration, security verification scripts
- **Build Verification:** Production build successful (2.77s) with optimized assets, 0 vulnerabilities
- **Development Server:** Hot module replacement functional for rapid iteration
- **Clean Foundation:** Minimal App.tsx structure prepared for frontend development

Task Output: All deliverables met as specified in Implementation Plan
- Verified development environment ✅
- Security configurations implemented ✅
- Clean App.tsx foundation ✅
- Confirmed integration of all technology stack components ✅

◆◆ **Phase 1 Complete - All Tasks Successful!**
Phase 1 Summary:
- **5/5 Tasks Completed Successfully** ✅
- **Agent_Foundation_Setup** completed all foundation tasks
- **Technology Stack:** Vite + React + TypeScript + Tailwind CSS v4 + Shadcn UI
- **Security:** Dependabot, npm audit, security verification implemented
- **Development Environment:** Fully validated and ready for frontend development

Next Action: Phase 1 Summary & Phase 2 Preparation
Now I need to:
1. Create Phase 1 summary in Memory Root
2. Prepare Phase 2 Memory directory structure
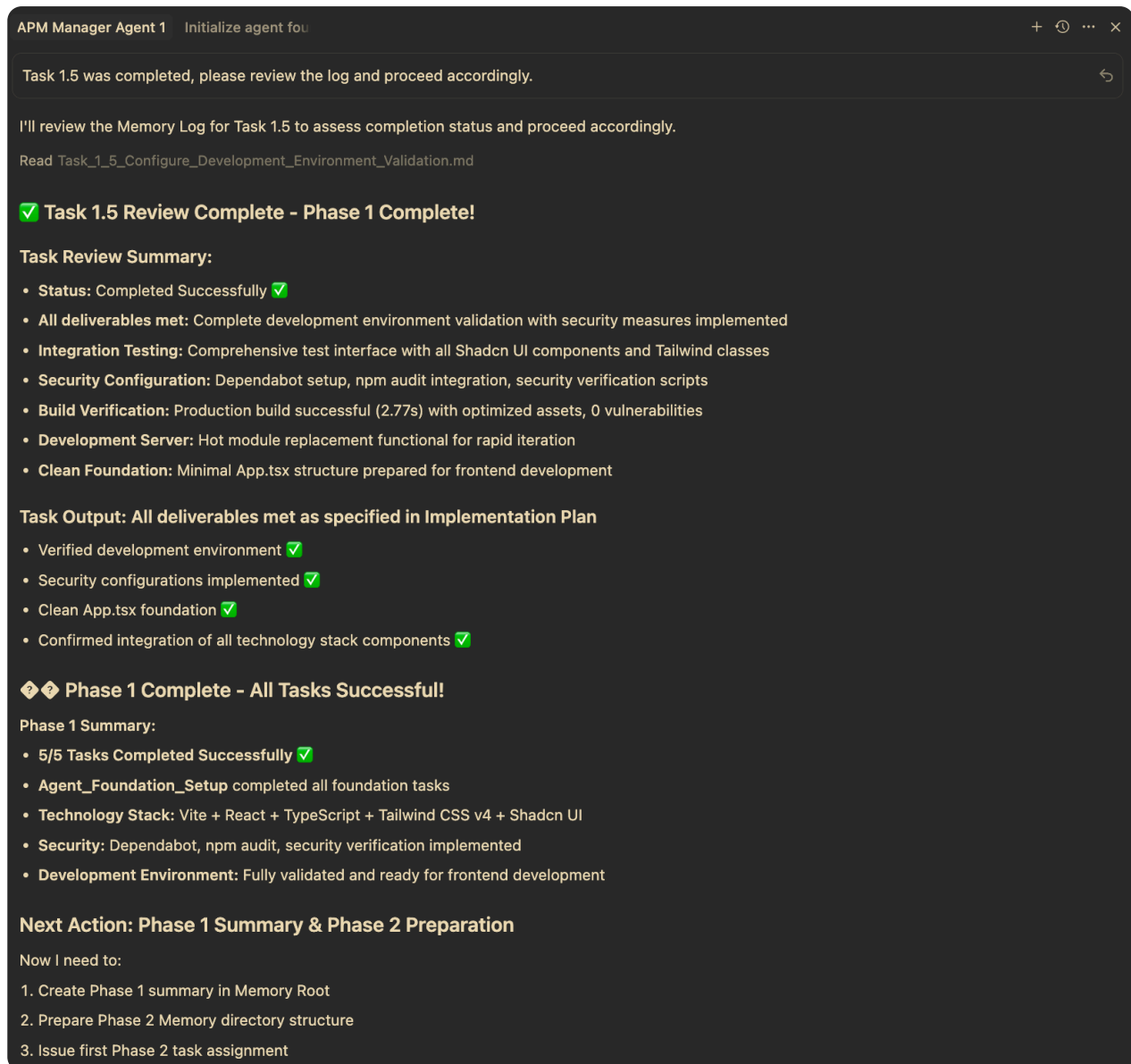3. Issue first Phase 2 task assignment

Figure 11: Manager Agent reviewing a task's Memory Log and determining next steps

4.3 Your First Ad-Hoc Delegation

Ad-Hoc Agents operate in an isolated workflow branch, without access to the Implementation Plan or Memory System. They focus solely on the delegated task and return their findings to the calling Implementation Agent. They are designed for handling complex or context-heavy issues that require focused, isolated attention.

In APM v0.4, we have two types of formal delegations:

- **Debugging:** When the Implementation Agent encounters complex issues that cannot be resolved locally.

  - **Minor Issues:** Implementation Agent debugs locally up to two attempts.
  - **Major Issues:** After two attempts or if the issue is complex/systemic, the Implementation Agent will delegate the task to an Ad-Hoc Agent.

- **Research:** When the task execution or requires web-based research which often consumes a significant portion of the context window. The Implementation Agent will delegate the task to an Ad-Hoc Agent.

4.3.1 Delegation Workflow

When there is a need to delegate work to an Ad-Hoc Agent, the process is as follows:

1. **Implementation Agent Creates Delegation Prompt:** The agent will read the appropriate delegation guide to create a structured prompt, which will be presented as a **markdown code block for easy copy-paste**.

2. **User Opens Ad-Hoc Session:** Create new chat session, name it clearly (e.g., "Ad-Hoc Debugging Session") and initialize Ad-Hoc Agent providing the:
   ```
   prompts/ad-hocAd-Hoc_Agent_Initiation_Prompt.md
   ```
   via copy-paste, file upload, or your IDE's context tools, along with a clear instruction such as:
   ```
   Initialize as Ad-Hoc Agent
   ```

3. **Ad-Hoc Agent Executes:** Follows 3-step process **(Assessment → Execution → Delivery)**. The Ad-Hoc Agent will deliver the findings in a **markdown code block for easy copy-paste**.

4. **User Returns Findings:** Deliver results back to the Implementation Agent for integration.

5. **Implementation Agent Integrates Findings:** Apply solution from findings and continue task execution, or escalate the situation to the Manager by documenting the issue in the Memory Log and requesting further guidance.

> **Escalating issues to Manager Agent:** If escalation is needed, the Manager Agent reviews the issue and determines the next steps, leveraging full project context for making the best decision. This may involve adjusting the Implementation Plan, or providing additional guidance to the Implementation Agent.

> **Understanding Ad-Hoc Agents:** Ad-Hoc Agents don't log to Memory System. They report findings via markdown code blocks for easy integration back into the main workflow.

5 Your First Handover

As agent sessions grow, you'll encounter context window limits. APM's Handover Protocol enables seamless context transfer to replacement agents.

5.1 When to Consider Handovers

Initiating a handover at the right moment is essential. The general principle is **"Better safe than sorry"**. Consider the following strategies:

- **Monitor context window usage:** Use your AI IDE's context window visualization tools to track how much of the available context is being used. Plan to initiate a handover when usage reaches 80–90% of the model's context window.

- **Monitor for declining agent performance:** Watch for warning signs like inconsistent decisions, fabricated details, or the agent forgetting recent instructions. These often signal that the context window is nearly full or has been exceeded.

> **Visual Context Window Indicators:** Many AI IDE platforms offer similar context window indicators, **while some may not provide this feature at all**. Be sure to familiarize yourself with the capabilities of the specific platform you are using.

> **If your AI IDE does not offer a context window indicator, use these general rules of thumb:**
>
> - **Implementation Agents:** Consider initiating a handover after **5-10 task cycles**, depending on task complexity and context depth. Task complexity can be estimated by the number of steps or instructions in the task and their own complexity. On some occasions, handovers may need to be even more frequent.
>
> - **Manager Agents:** Manager Agents do not execute tasks, therefore their cycles are not dependent on task complexity. Instead, they are dependent on next action assessment and decision making that happens after every task review. If a task has revealed a complex issue or significant insight, the Manager Agent will need to 'spend more context' assessing the situation and making a decision.
>
>   - For the **first Manager Agent instance** which parses the Bootstrap Prompt and initializes the Task Loop Phase, consider initiating a handover after **10-15 task cycles**.
>   - For **replacement Manager Agents** created after a previous handover, consider initiating a handover after **15-20 task cycles**.

> **Late Handover Risks:** Delaying handovers increases the risk of context corruption, which can negatively impact replacement agents. **Initiate handovers proactively to maintain workflow integrity.**

5.2 Executing a Handover Procedure

When you determine a handover is needed, **request one from the agent in their chat session by providing the Handover Guide for the specific agent type.** For example:

```
Please do a Handover Procedure via <Agent Type>_Handover_Guide.md.
```

where <Agent Type> is either Manager_Agent or Implementation_Agent.

The agent will do the following according it's Handover Guide:

1. **Check Handover Eligibility:** The agent will check if it's eligible for a handover. If a Handover is requested during a task cycle of an agent, it is not eligible for a handover.

2. **Context Synthesis:** The agent will synthesize current task execution state by reviewing it's previous Memory Logs and it's own working context.

3. **Produce Handover Artifacts:** The agent will create a Handover File with undocumented context and a Handover Prompt (markdown code block) with the onboarding and context integration instructions for replacement agent.

4. **Present for Review:** The agent will present the Handover Artifacts to the User for review and validation.
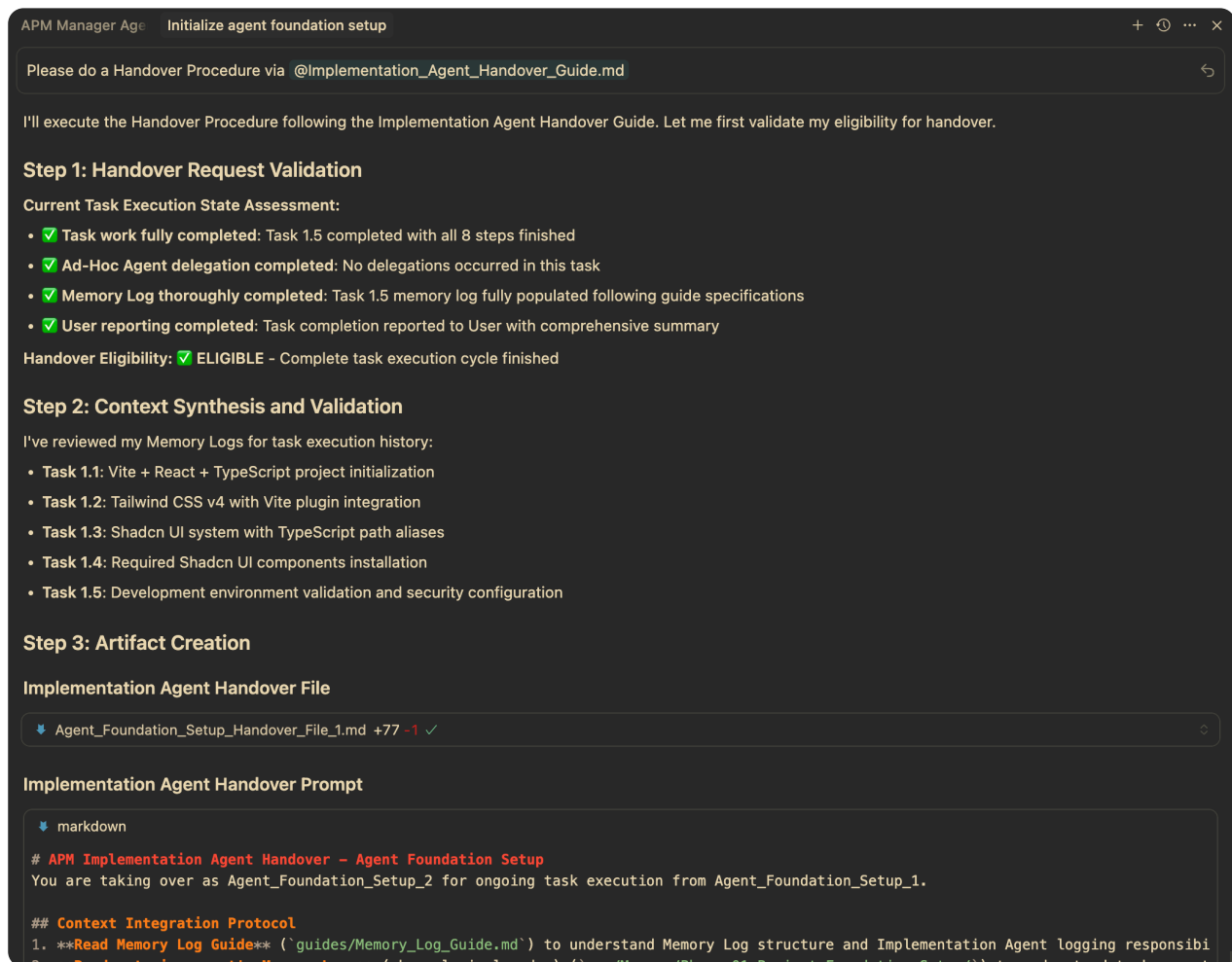
Figure 12: Implementation Agent executing a Handover Procedure

5.3 Completing a Handover Procedure

After the User has reviewed and validated the Handover Artifacts of the outgoing agent, the new agent, of the same type as the outgoing agent, can be initialized and onboarded by following the steps below:

1. **Open New Agent Session:** Create replacement session with clear naming (e.g., "Agent_Backend_2")

2. **Initialize New Agent:** Provide appropriate initiation prompt via copy-paste, file upload, or your IDE's context tools, along with a clear instruction such as: `Initialize as <Agent Type>`

3. **Deliver Handover Prompt:** Paste the outgoing agent's Handover Prompt as very first message after initialization

4. **Verify Understanding:** Carefully review the new agent's context understanding summary

5. **Authorize Operations:** Only proceed after confirming accurate context transfer

Figure 13: New Implementation Agent processing handover context and confirming understanding after initialization

6 Summary & What's Next

Completing your first APM session demonstrates how structured multi-agent workflows transform chaotic AI interactions into reliable project execution. The systematic coordination between agents, comprehensive memory management, and seamless context transfer provide confidence to tackle complex projects.

6.1 Key Takeaways

- **Structure Enables Flexibility:** The Setup Phase creates a foundation that makes execution highly efficient and adaptable

- **Context is Key:** Invest time in thorough Context Synthesis and Project Breakdown during Setup Phase; it prevents downstream problems. Comb through the Implementation Plan carefully before approval and request modifications or revisions as needed.

- **Proactive Management:** Initiate handovers before agents hit limits to maintain performance quality. Many agents can survive several handovers, but as projects progress context gets accumulated and transfer becomes inefficient.

- **You Are the Bridge:** Your role as communication coordinator between agents is crucial for project success. You are responsible of overseeing the interactions between agents, task execution and memory management, and are able to intervene when necessary.

6.2 Advanced Topics

For a deeper understanding, optimization strategies and troubleshooting common issues, refer to the **User Guide**. Each section of that document is self-contained, allowing you to use it as a reference manual as you become more familiar with APM.

For a complete understanding of all concepts, protocols, and workflows, explore the Documentation Suite in the GitHub repository.

For in-depth information on advanced prompt and context engineering, as well as the context and memory management techniques incorporated in APM, see these documents:

- Context and Memory Management

- Context and Prompt Engineering

6.3 Customization Opportunities

As you become more familiar with APM, consider customizing the framework to suit your specific project, team, or organizational needs. Creating your own template from the APM repository allows you to adapt prompts, guides, and workflows for optimal alignment with your objectives.

Happy project managing!