Product Requirements Document: AI Development Cache

Project Name: AI CLI Session Caching

Version: 1.1

Prepared By: AI Agent

1.0 Executive Summary

The purpose of this document is to outline the requirements for a caching system that will persist AI development work done within a single working session for the Gemini and Claude command-line interfaces (CLIs). This project aims to significantly improve developer workflow efficiency by eliminating the need to re-run or re-generate previous outputs and to provide a persistent "memory" for each project, reducing token costs and accelerating iteration cycles. This project will be developed and maintained as an open-source tool on a public GitHub repository.

2.0 Project Goals & Business Objectives

2.1 Goals

- **Improve developer productivity:** Reduce the time and resources spent on repetitive AI prompts and tasks.
- **Lower operational costs:** Minimize API token usage for recurring prompts and conversations.
- **Enhance user experience:** Provide a seamless, stateful experience for developers using the AI CLIs, making the tools feel more like an integrated part of their workflow.

2.2 Objectives

- Implement a caching mechanism that can store and retrieve AI prompts, responses, and session context.
- Integrate the caching system with both the Gemini and Claude CLIs.
- Provide a simple, user-friendly way to manage the cache (e.g., clear, inspect, and enable/disable).
- Ensure the cache is persistent across multiple working sessions.

3.0 Scope and Key Features

3.1 In-Scope Features

- **Automatic Session Caching:** The system will automatically cache all user prompts,

AI-generated responses, and any associated context (e.g., file contents read by the CLI).
- **Persistence:** The cache will be stored locally on the user's machine (e.g., in a hidden directory like .ai-cache within the project folder) to persist between sessions.
- **CLI Integration:** A new sub-command or flag will be added to the Gemini and Claude CLIs to enable and manage the cache. For example: gemini --use-cache or claude --cache-clear.
- **Cache Management Commands:**
  - --cache-list: Displays a list of cached sessions or prompts.
  - --cache-clear: Clears the cache for the current project.
  - --cache-on / --cache-off: Toggles the caching feature for a session.
- **Project-Specific Caching:** The cache will be tied to the specific project directory, ensuring that cache data from one project does not interfere with another. This will likely be achieved by using a unique identifier based on the project path.

3.2 Out-of-Scope Features

- **Cloud Caching:** The initial version will not support cloud-based or shared caching between multiple users.
- **Cache Invalidation based on file changes:** The cache will not automatically invalidate if a file within the project is modified. The user will need to manually clear the cache if they want to ensure a fresh generation.
- **Version Control Integration:** The system will not automatically integrate with Git to commit or manage cache files. Users can choose to add the cache directory to their .gitignore file.

4.0 User Stories

- **As a developer,** I want all my AI prompts and responses to be automatically saved to a local cache so that I don't have to re-type them on my next working session.
- **As a developer,** I want to be able to re-run a previous command and have the cached response returned instantly, so that I can save time and API costs.
- **As a developer,** I want to be able to see a list of my previously cached prompts and their responses to review my work from a past session.
- **As a developer,** I want the cache to be specific to the project I am working on, so that my cached data for Project A doesn't accidentally show up when I am working on Project B.
- **As a developer,** I want to be able to easily clear the cache for a project when I want to start a new, clean session.
- **As a community developer,** I want to be able to easily contribute to the project by submitting pull requests and reporting issues.

5.0 Technical Requirements & Assumptions

5.1 Technical Requirements

- **Caching Mechanism:** A key-value store will be used to map a hash of the prompt and context to the AI's response. The key should be a hash of the user's prompt, the included context (e.g., filenames, file content), and the model's parameters (e.g., temperature, max tokens).
- **Storage:** The cache will be stored in a flat-file database or a simple file structure within the project's root directory. The directory name should be hidden (e.g., .ai-cache).
- **Supported CLIs:** The solution must work with the official Gemini and Claude CLIs. This may require creating a wrapper script or a plugin for each.
- **Data Structure:** The cached data should be stored in a structured format like JSON or YAML to be easily readable and extensible. The data should include the prompt, the response, a timestamp, the model used, and the session context.
- **Hash Function:** A fast and collision-resistant hashing algorithm (e.g., SHA-256) should be used to generate cache keys.

5.2 Assumptions

- The user has a single project directory where they use the CLIs.
- The project directory is stable and does not frequently change location.
- The user is comfortable interacting with a CLI and managing a local .ai-cache directory.
- There is no need for encryption on the locally stored cache data, as it is assumed to be non-sensitive.
- The project will be managed on a public GitHub repository using Git for version control.

6.0 Future Considerations (Out-of-Scope for V1)

- **Cloud-based caching:** Explore a future where the cache can be synced to a cloud service for multi-machine access.
- **Intelligent cache invalidation:** Implement a system to automatically invalidate cache entries when relevant project files are modified.
- **Cache expiration:** Add a time-to-live (TTL) to cache entries to prevent stale data and manage disk space.
- **Cache sharing:** Allow users to share specific cached prompts or sessions with teammates.

7.0 Open-Source Strategy & Community

The decision to open-source this project is driven by a desire to foster community collaboration, ensure transparency, and accelerate development. By making the source code public, we can benefit from the collective expertise of developers who want to improve their AI workflow.

7.1 Licensing

The project will be licensed under the **Apache 2.0 License**. This is a permissive license that allows for both commercial and non-commercial use, which is ideal for a developer tool. It

enables developers to use the code in their projects with minimal restrictions while still protecting the original contributors.

7.2 Contribution Guidelines

The project will be hosted on a public GitHub repository. To encourage and manage contributions, the following will be established:

- **Repository Structure:** The repository will include a clear directory structure with dedicated folders for source code, documentation, and tests.
- **README.md:** A comprehensive README.md will be created to explain the project's purpose, features, installation instructions, and basic usage.
- **CONTRIBUTING.md:** This document will provide detailed guidelines for how to contribute, including the process for opening issues, submitting pull requests, and adhering to coding standards.
- **Issue Tracking:** All bugs, feature requests, and discussions will be managed through GitHub Issues.
- **Pull Requests:** Contributions will be accepted via pull requests, which will be reviewed by maintainers to ensure quality and alignment with the project's goals.