AI Agent Context Engineering

User Prompt

Primary trigger that initiates the agent's reasoning and context assembly

Contains intent, entities, constraints, and signals for behavior selection

Guides which tools, external prompts, MCP services, or memory to use

Evolves over turns, shaping the agent's plan and response

Example: Roofing Company Service Agent

User asks for a roof inspection and mentions storm damage

Agent infers intent (claim-related vs general inquiry)

Prompt signals what additional context to load

Follow-up prompts refine scope, urgency, and next actions

External System Prompts & Markdown Files

Externally stored instructions (often Markdown) pulled in at runtime

Allow agent behavior to change without redeploying code

Selected dynamically based on user intent, entities, or context

Define tone, rules, constraints, and domain-specific behavior

Example: Roofing Company Service Agent

Agent detects an insurance company name, adjuster ID, or claim number

Retrieves an insurance-specific instruction file (e.g. insurance-rules.md)

Switches to formal tone, insurance terminology, and compliance rules

Avoids consumer sales language and follows claim-handling workflows or

Passes the task to a claim agent.

User Interaction / Agent Memory

Persists information learned across interactions and sessions

Captures preferences, prior decisions, and confirmed facts

Informs personalization, continuity, and reduced repetition

Updated iteratively from user feedback and agent outcomes

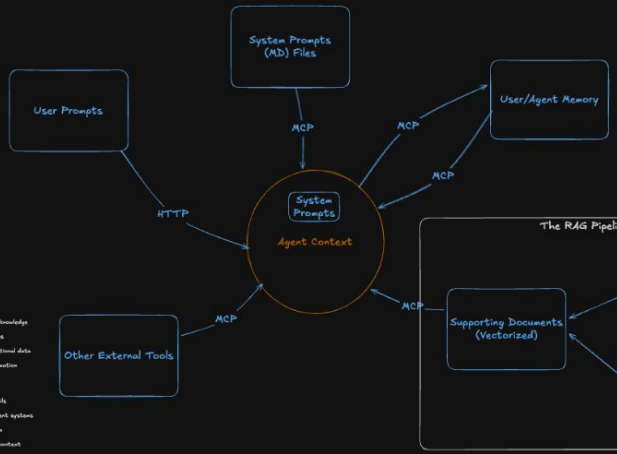
Example: Roofing Company Service Agent

User previously confirmed they are an insurance adjuster

Agent remembers preferred report format and terminology

Prior inspection details are reused in follow-up conversations

Agent avoids re-asking known information and maintains context



Tool-Based & External Data Retrieval

Provides real-time or system-of-record data beyond static knowledge

Accessed via tools, plugins, APIs, or MCP-connected services

Used when answers require freshness, authority, or transactional data

Enables the agent to take actions as well as retrieve information

Example: Roofing Company Service Agent

Queries a CRM to retrieve customer, adjuster, or claim details

Pulls project status and invoices from ERP or job management systems

Uses online search for weather events or recent storm data

Combines live system data with prompts, memory, and RAG context

Retrieval-Augmented Generation (RAG)

Supplies the agent with factual, source-backed context at runtime

Retrieves relevant knowledge based on the current user prompt

Grounds responses in company-specific or domain-specific content

Reduces hallucinations by anchoring outputs to real documents

Enables knowledge updates without retraining the model

RAG Pipeline (High Level)

Documents are ingested, chunked, embedded, and indexed

User prompt is embedded and used for similarity search

Top-ranked chunks are selected, scored, and filtered

Retrieved context is injected into the agent's context builder

Example: Roofing Company Service Agent

Agent retrieves roofing manuals, inspection checklists, and SOPs

Pulls insurance claim guidelines from Google Drive or SharePoint

Uses retrieved documents to answer policy-specific questions

References authoritative content when generating inspection reports

Less Is Better

- More context does not mean better understanding
Overloading an agent with excessive or irrelevant context increases noise, making it harder for the model to identify what actually matters for the current task or decision.

- Focused context improves reasoning quality
Carefully selected, high-signal inputs lead to clearer planning, better tool selection, and more accurate outputs than broad, unfocused context dumps.

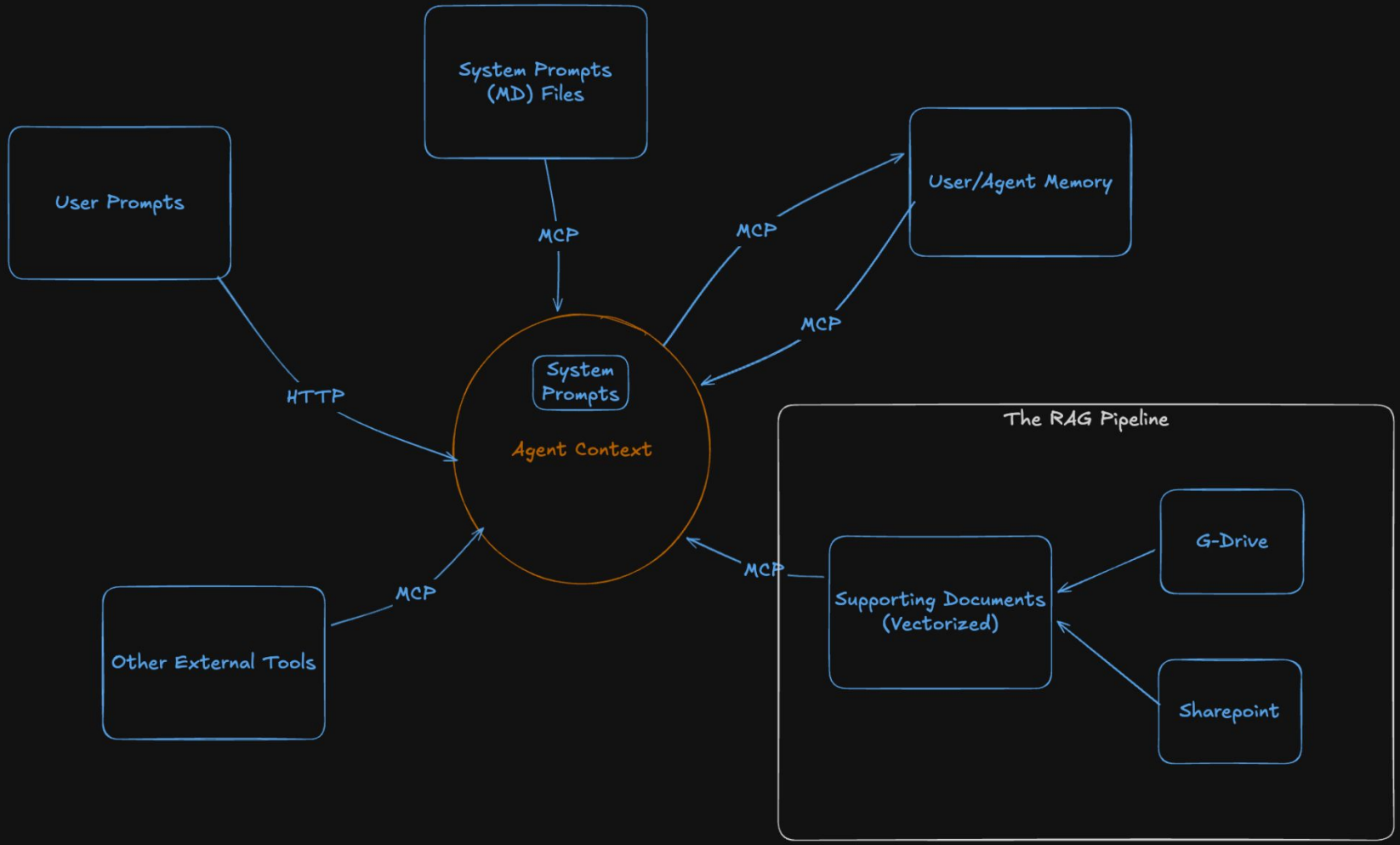
- Smaller context reduces unintended behavior
Unnecessary instructions, outdated documents, or conflicting rules can cause instruction drift, policy violations, or inconsistent responses.

- Lean context is more scalable and maintainable
Dynamically pulling only what is needed keeps systems faster, cheaper, and easier to evolve as prompts, tools, and knowledge sources grow.

- Intent-driven context mirrors human decision-making
Like a human expert, an effective agent reasons from the problem first, then consults only the most relevant references instead of reviewing everything at once.

TL;DR

Agent context is the selectively assembled set of instructions, memory, retrieved knowledge, and tool-based data an AI uses to reason and act for a specific user request. Effective agents prioritize relevant, minimal, and intent-driven context to improve reasoning quality, safety, and scalability.


System Prompts
(MD) Files
User Prompts
User/Agent Memory
MCP
MCP
MCP
HTTP
System
Prompts
Agent Context
The RAG Pipeline
G-Drive
MCP
Supporting Documents
(Vectorized)
MCP
Other External Tools
Sharepoint

User Prompt

Primary trigger that initiates the agent's reasoning and context assembly

Contains intent, entities, constraints, and signals for behavior selection

Guides which tools, external prompts, MCP services, or memory to use

Evolves over turns, shaping the agent's plan and response

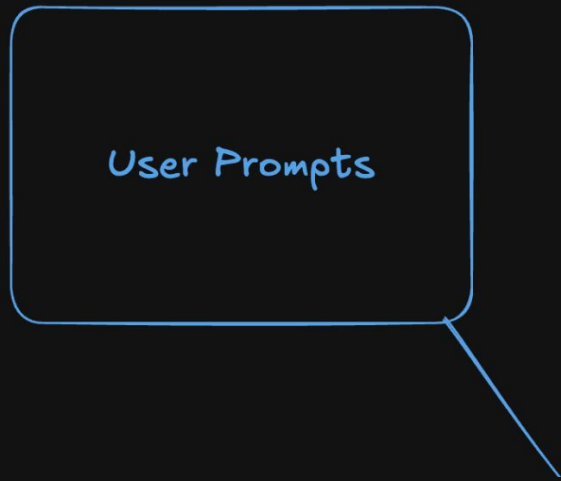
Example: Roofing Company Service Agent

User asks for a roof inspection and mentions storm damage

Agent infers intent (claim-related vs general inquiry)

Prompt signals what additional context to load

Follow-up prompts refine scope, urgency, and next actions



External System Prompts & Markdown Files

Externally stored instructions (often Markdown) pulled in at runtime

Allow agent behavior to change without redeploying code

Selected dynamically based on user intent, entities, or context

Define tone, rules, constraints, and domain-specific behavior

Example: Roofing Company Service Agent

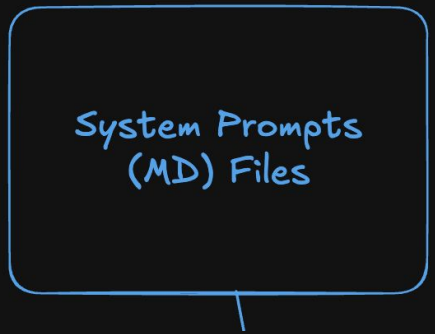
Agent detects an insurance company name, adjuster ID, or claim number

Retrieves an insurance-specific instruction file (e.g. insurance-mode.md)

Switches to formal tone, insurance terminology, and compliance rules

Avoids consumer sales language and follows claim-handling workflows or

Passes the puck to a claim agent.



User Interaction / Agent Memory

Persists information learned across interactions and sessions

Captures preferences, prior decisions, and confirmed facts

Informs personalization, continuity, and reduced repetition

Updated iteratively from user feedback and agent outcomes

Example: Roofing Company Service Agent

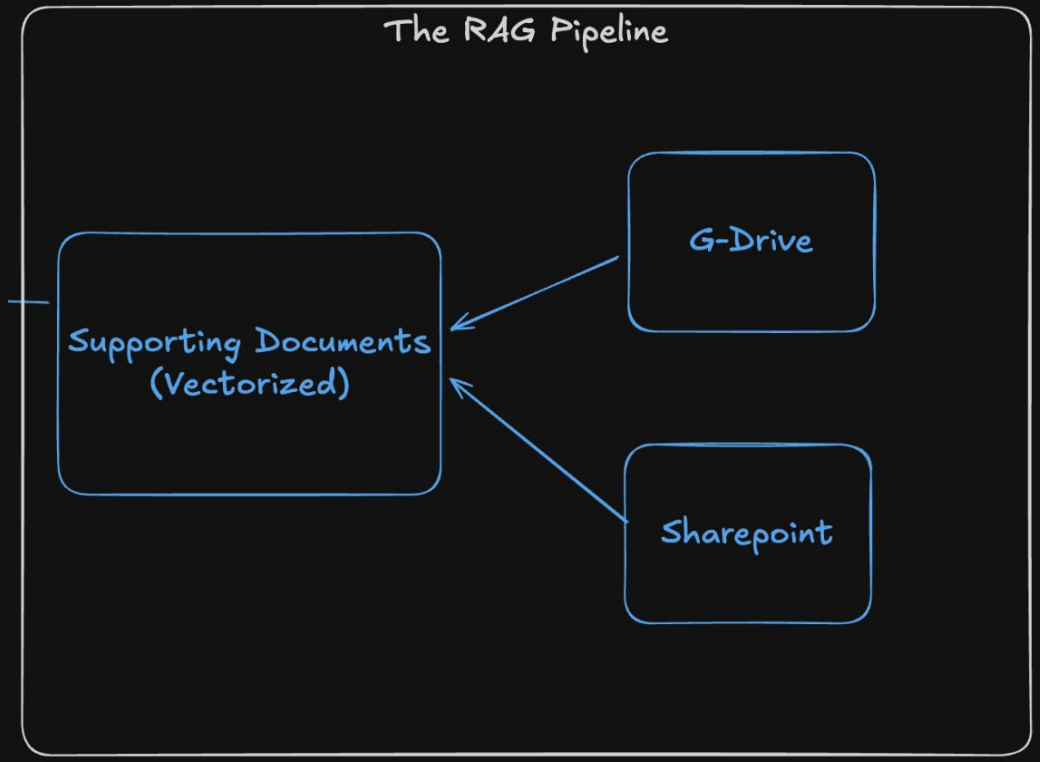
User previously confirmed they are an insurance adjuster

Agent remembers preferred report format and terminology

Prior inspection details are reused in follow-up conversations

Agent avoids re-asking known information and maintains context





Retrieval-Augmented Generation (RAG)

Supplies the agent with factual, source-backed context at runtime

Retrieves relevant knowledge based on the current user prompt

Grounds responses in company-specific or domain-specific content

Reduces hallucinations by anchoring outputs to real documents

Enables knowledge updates without retraining the model

RAG Pipeline (High Level)

Documents are ingested, chunked, embedded, and indexed

User prompt is embedded and used for similarity search

Top-ranked chunks are selected, scored, and filtered

Retrieved context is injected into the agent's context builder

Example: Roofing Company Service Agent

Agent retrieves roofing manuals, inspection checklists, and SOPs

Pulls insurance claim guidelines from Google Drive or SharePoint

Uses retrieved documents to answer policy-specific questions

References authoritative content when generating inspection reports

Tool-Based & External Data Retrieval

Provides real-time or system-of-record data beyond static knowledge

Accessed via tools, plugins, APIs, or MCP-connected services

Used when answers require freshness, authority, or transactional data

Enables the agent to take actions as well as retrieve information

Example: Roofing Company Service Agent

Queries a CRM to retrieve customer, adjuster, or claim details

Pulls project status and invoices from ERP or job management systems

Uses online search for weather events or recent storm data

Combines live system data with prompts, memory, and RAG context



Less Is Better

- More context does not mean better understanding
Overloading an agent with excessive or irrelevant context increases noise, making it harder for the model to identify what actually matters for the current task or decision.

- Focused context improves reasoning quality
Carefully selected, high-signal inputs lead to clearer planning, better tool selection, and more accurate outputs than broad, unfocused context dumps.

- Smaller context reduces unintended behavior
Unnecessary instructions, outdated documents, or conflicting rules can cause instruction drift, policy violations, or inconsistent responses.

- Lean context is more scalable and maintainable
Dynamically pulling only what is needed keeps systems faster, cheaper, and easier to evolve as prompts, tools, and knowledge sources grow.

- Intent-driven context mirrors human decision-making
Like a human expert, an effective agent reasons from the problem first, then consults only the most relevant references instead of reviewing everything at once.

TL:DR

Agent context is the selectively assembled set of instructions, memory, retrieved knowledge, and tool-based data an AI uses to reason and act for a specific user request. Effective agents prioritize relevant, minimal, and intent-driven context to improve reasoning quality, safety, and scalability.