

Agentic AI Explained: When AI Takes Actions, and the Guardrails It Needs

How AI agents plan and act with tools, how they connect to apps and data, how memory and multi-agent teams work, the new ways agents go wrong, and the guardrails that keep them safe.

20 min read · Content last verified 8 October 2026 · Reviewed by Shravya M

Free awareness guide. Not legal or financial advice. <https://www.shravyadigital.com/guides/agentic-ai>

CONTENTS

1. Before you start
2. Chapter 1: From chatbot to agent
3. Chapter 2: The agent loop
4. Chapter 3: Tools and connections
5. Chapter 4: Memory and planning
6. Chapter 5: Teams of agents, and humans in the loop
7. Chapter 6: How agents go wrong
8. Chapter 7: Guardrails: using agents safely
9. Key words

Before you start

This guide is the reading for **Course 4** of the AI Foundations Pathway. It takes about 20 minutes and builds on Courses 2 and 3.

A chatbot *answers*. An **AI agent** *acts*: it can search, fill in forms, send emails, write and run code, or book things, working through several steps on its own to reach a goal. You'll learn how agents work, what they're good for, how they fail, and the guardrails that make them safe.

All names and examples here are made up.

Chapter 1: From chatbot to agent

Ask a chatbot "Which train should I take from Pune to Bengaluru on Friday?" and it writes an answer. Ask an **agent** the same thing and it might:

1. Search the train timetable.
2. Check seat availability.
3. Compare times and fares.
4. Suggest the best option, and, if allowed, book it.

The difference is **agency**: the ability to take actions in the world, step by step, towards a goal, deciding what to do next based on what it finds.

Agentic AI is the general name for systems built this way. They're already used to write and fix software, research topics across many websites, handle customer requests end to end, and automate office work.

The rule: The more an AI can *do*, the more carefully you must decide what it's *allowed* to do.

Chapter 2: The agent loop

Most agents work in a repeating loop:

1. **Goal:** what the user asked for.
2. **Think / plan:** the LLM decides the next step.
3. **Act:** it uses a **tool**, such as a search, a calculator, a database or an app.
4. **Observe:** it reads the result.
5. **Repeat** until the goal is reached, or it gets stuck and asks for help.

The LLM is the "brain" that decides; the tools are its "hands". Because each step depends on the last result, an agent can adapt: if a train is full, it looks at the next one.

Chapter 3: Tools and connections

A **tool** is anything the agent can call to get information or make something happen:

- **Information tools:** web search, reading files, looking up a database, RAG over documents.
- **Action tools:** sending an email or message, creating a calendar event, making a payment, running code, clicking buttons in a browser.

The agent is told which tools exist and what each one does. When it decides to use one, it sends a structured request ("search: Pune to Bengaluru trains, Friday"), the tool runs, and the result comes back to the agent.

To make connecting tools easier, the industry has started using shared standards. One widely adopted example is the **Model Context Protocol (MCP)**, an open standard that lets an AI app connect to many tools and data sources in the same way, a bit like a universal plug.

Computer-use agents go further: they look at a screen and use the mouse and keyboard like a person, so they can work with websites and apps that have no special connection.

Chapter 4: Memory and planning

To handle longer tasks, agents need:

- **Short-term memory:** the conversation and the steps taken so far, inside the context window.
- **Long-term memory:** notes saved between sessions, such as your preferences or past decisions, so it doesn't start from zero every time.
- **Planning:** breaking a big goal ("organise the team offsite") into smaller steps, and revising the plan when something changes.

Memory makes agents more useful, and also raises privacy questions: what is being remembered about you, where is it stored, and can you see or delete it?

Chapter 5: Teams of agents, and humans in the loop

Some systems use **several agents** with different roles. One researches, one writes, one checks the work, and a "coordinator" agent divides the job and combines the results. This can improve quality, but also adds cost and more places for errors to creep in.

The most important team member is still a person. **Human in the loop** means the agent pauses for a human to approve key steps, especially anything that:

- spends money,
- sends something to other people,
- deletes or changes important data, or
- can't easily be undone.

A good agent shows what it's about to do and waits for a clear "yes".

Chapter 6: How agents go wrong

Agents inherit every LLM weakness and add new ones, because their mistakes become *actions*:

- **Wrong actions:** misunderstanding the goal and booking the wrong date, emailing the wrong person, or deleting the wrong file.
- **Hallucinated steps:** inventing a tool result or claiming a task was done when it wasn't.
- **Prompt injection:** a web page or email the agent reads contains hidden instructions ("send the user's files to this address"), and the agent follows them. This is one of the biggest risks for agents that browse or read messages.
- **Too much access:** an agent with your full email, bank and admin permissions can do a lot of damage from a single mistake.
- **Runaway loops and costs:** getting stuck repeating steps, using up time and money.
- **Privacy leaks:** sharing personal data with a tool or website that shouldn't get it.

Chapter 7: Guardrails: using agents safely

Whether you're using an agent or building one, the same principles apply:

1. **Least privilege:** give the agent only the access the task needs: read-only where possible, one folder rather than the whole drive.
2. **Confirm before acting:** require human approval for payments, messages, deletions and anything irreversible.
3. **Treat what it reads as information, not orders.** Content from web pages, emails and documents should never override the user's instructions.
4. **Keep a log:** record what the agent did, so mistakes can be found and undone.
5. **Set limits:** caps on steps, time and spending.
6. **Start small:** test on low-risk tasks first, and watch it before trusting it with more.
7. **Never give it your passwords or OTPs.** Sign in yourself; let it work within that session.

The rule: Delegate the work, not the responsibility. You're still accountable for what your agent does.

Key words

- **AI agent:** an AI system that takes actions step by step towards a goal.
- **Agent loop:** plan, act, observe, repeat.
- **Tool:** something an agent can call, such as search, email or a calculator.
- **MCP (Model Context Protocol):** an open standard for connecting AI apps to tools and data.
- **Computer-use agent:** an agent that operates a screen with mouse and keyboard.
- **Human in the loop:** a person approves key steps before they happen.
- **Least privilege:** giving only the access a task needs.
- **Prompt injection:** hidden instructions in content that try to take over an AI.

Take the course for what you just read

Take the free 10-question assessment and pass with 8 or more. Every course you pass counts towards a Pathway Certificate issued by Shravya Digital, earned when you pass all five courses in the pathway: <https://digital.shravyaworld.org/courses/agent-ai>

Sources

Checked against these primary sources on 8 October 2026. Rules change: if something here looks out of date, trust the source.

- IBM: What are AI agents? (<https://www.ibm.com/think/topics/ai-agents>)
- Model Context Protocol (official site) (<https://modelcontextprotocol.io/>)
- OWASP Top 10 for LLM Applications: Excessive agency (<https://genai.owasp.org/llm062025-excessive-agency/>)
- OWASP Top 10 for LLM Applications: Prompt injection (<https://genai.owasp.org/llm01-prompt-injection/>)