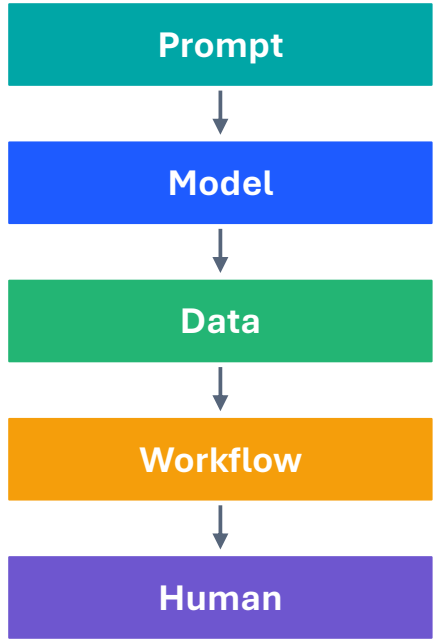
Module: Tools, Platforms & Practical AI Enablement

Session 1: AI for Personal & Professional Productivity

Using AI tools to improve speed, quality and thinking in everyday work



AI as my assistant: prompt, context, output, verification

Leadership with AI – Rajendra M Sonar, IIT Bombay

What we are going to learn

The session moves from concepts to a practical demonstration pattern.

🧭 Tool landscape

What general AI assistants, copilots and knowledge tools are good at — and where they are weak.

✍️ Prompt craft

A simple prompt framework: task, context, constraints, output format and quality criteria.

📎 Context engineering

How documents, examples, data and templates make outputs more reliable and useful.

🧠 Knowledge work

Summaries, briefs, analysis, comparisons, ideation, tables, checklists and decisions.

🖼️ Multimodal work

Working with documents, charts, screenshots, audio and images as inputs.

✅ Verification

How to review AI work: fact-checking, privacy, bias, hallucinations and accountability.

Some topics are applicable to other two sessions in this module.

[The 26 Best AI Tools of 2026 — Ranked]

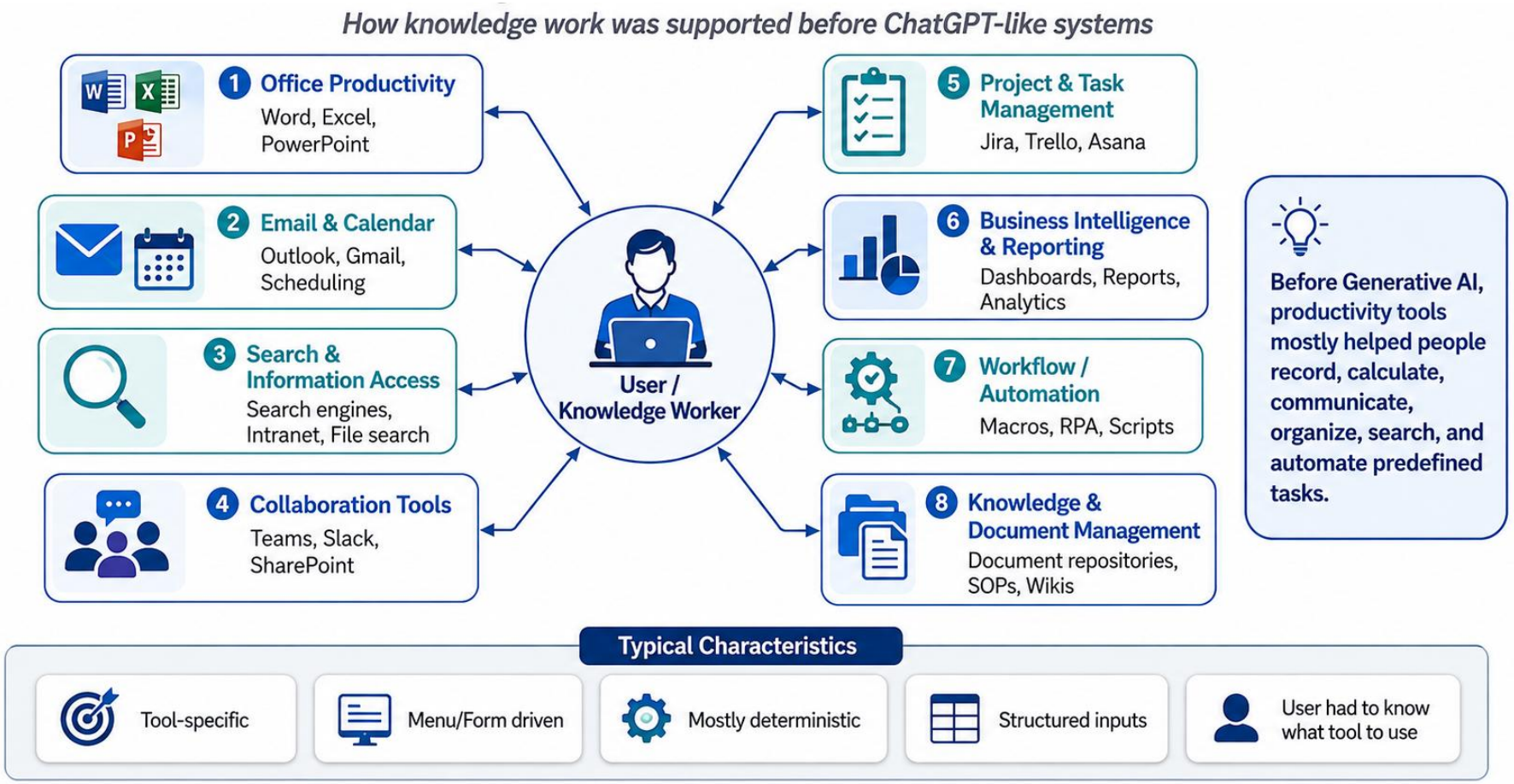
[10 Best AI Productivity Tools (Tested 2026)]

[The best AI productivity tools in 2026 | Zapier]

[Chatgpt/Codex Use cases]

[Money Machine: How 2 People Close 1,000 Enterprise Clients Every Month Using AI (Complete guide)]

Which tools were used before Gen AI tools?



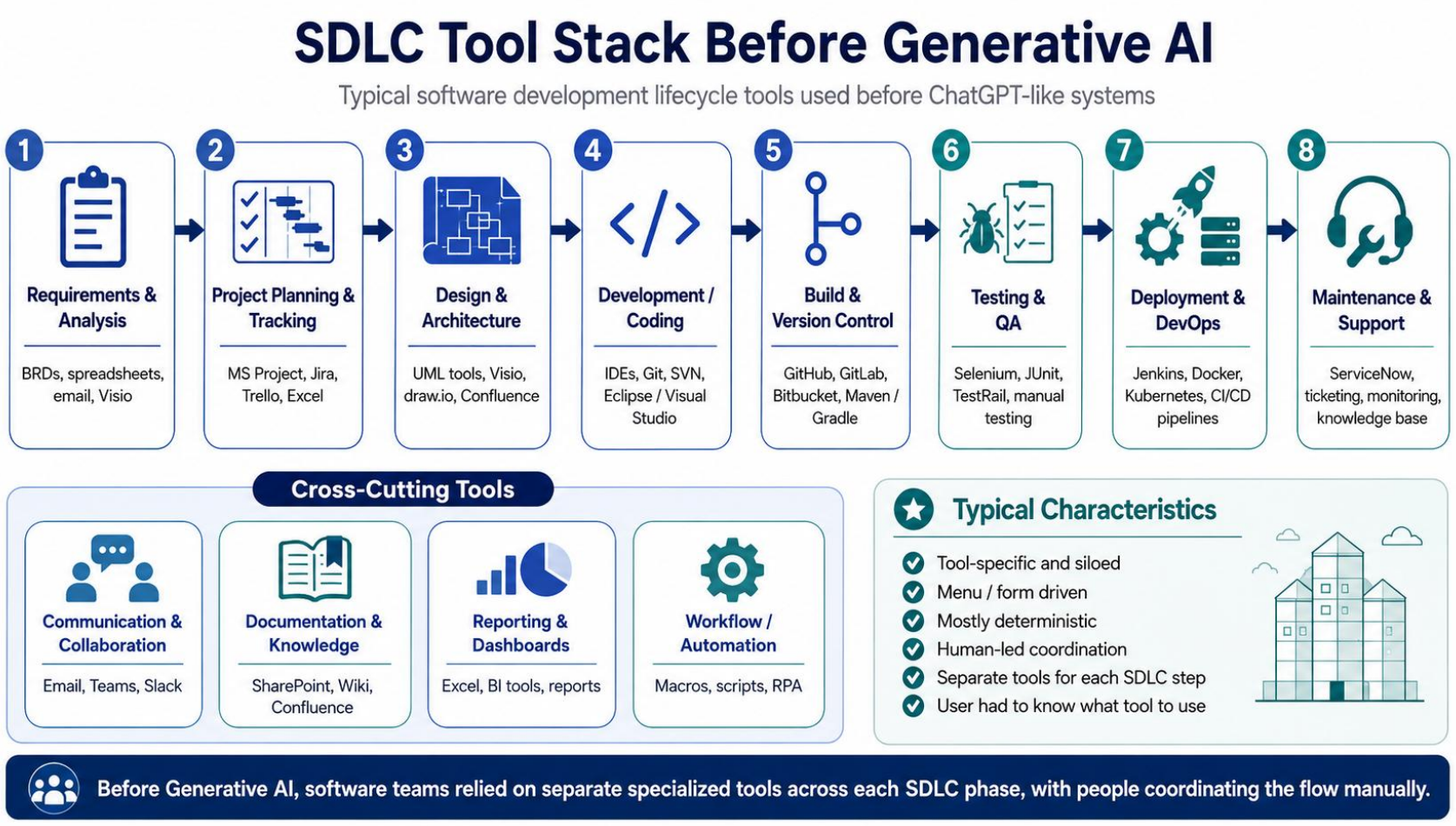
What fundamentally changed with ChatGPT-like systems?

Evolution of Productivity Tools: From Software Assistance to AI Collaboration using natural language

Instead of learning the software interface, users can increasingly express intent in natural language. The system can then:

- interpret the request,
- generate content,
- summarize,
- transform,
- reason across information,
- adapt output to context,
- work with unstructured information,
- and increasingly invoke tools and take actions.

Which tools were used before Gen AI tools? Example: task specific



Think over

1. How you can make these tools work this end-to-end?

2. Can there be a tool which takes care of many of these activities?

The shift

Earlier productivity tools	GenAI-era productivity tools
User operates functions	User expresses intent
Menu/form driven	Natural-language driven
Mostly deterministic	Generative and probabilistic
Structured inputs preferred	Handles unstructured/multimodal inputs
Executes predefined functions	Can synthesize and create new outputs
Tool-specific skills required	Conversational interaction lowers entry barrier
User integrates information	AI can synthesize across information
Separate tools for tasks	One assistant can span many tasks
Automation follows fixed rules	AI can interpret context before acting
Software is primarily a tool	AI increasingly behaves like a collaborator/copilot

Each user was limited by knowledge and skills the user knows and have worked on. Mostly one task a time!

Now world's knowledge is available at a fingertip.

Think over, if you already own the software, how you can make GenAI era backed, what way and what are the advantages?

Automation vs augmentation

Earlier productivity paradigm
"Make the existing task faster (efficiency)."

Examples:
- Excel formula calculates faster
- mail merge sends messages
- RPA copies data between systems
- PowerPoint templates speed slide creation

GenAI productivity paradigm
"Help perform the cognitive part of the task."

Examples:
- formulate the analysis itself
- interpret a report
- identify gaps
- generate options
- critique a proposal
- create a first draft
- tailor communication for different audiences

That is a much deeper shift.

Evolution and current landscape

Evolution of Productivity Technologies

- Productivity tools before Generative AI
- Office automation, search, collaboration, BI, workflow, and RPA
- From application-centric to intent-centric interaction
- From automation of tasks to augmentation of cognitive work
- Evolution from **tools → copilots → agents**

Understanding the Current AI Productivity Tool Landscape

- General-purpose AI assistants
- Enterprise copilots
- Research and knowledge tools
- Document and presentation tools
- Data-analysis tools
- Meeting assistants
- Coding assistants
- Multimodal AI tools

From “Using Software” to “Working with AI”

Traditional productivity software
User → selects application → selects function → provides structured input → receives predefined output

Traditional productivity software
User → selects application → selects function → provides structured input → receives predefined output

AI-enabled productivity
User → expresses intent → AI interprets context → generates/reasons → uses information/tools → produces/adapts output
And eventually:

Agentic productivity
User → defines goal → AI plans → invokes tools → coordinates steps → seeks approval → completes task
That one conceptual slide can become a very good bridge into the entire session.
A strong one-line takeaway for this part could be:

The productivity shift is from learning how to operate software to learning how to express intent, provide context, and collaborate with AI.

Revisit: Combining AI and human cognition

As AI absorbs tasks related to retrieval and summarization, the locus of human contribution shifts upward towards judgement, direction, expertise, and synthesis. AI is best understood as a powerful ship rather than an independent navigator: it can move faster and farther than any human, but without a *knowledgeable captain* who understands the vessel and the waters they are navigating, it is as likely to drift aimlessly as it is to arrive anywhere useful.

Cognitive workers must therefore supply *deep subject-matter* understanding to frame the right questions, identify meaningful trade-offs, and evaluate outputs critically.

AI *amplifies* the returns to prior knowledge and users who read widely and deeply are better able to steer models and push analysis beyond surface-level synthesis.

Cognitive workers must act as *system architects* rather than task executors. Productive AI use depends on the ability of the human to decompose complex problems, sequence inquiries, impose constraints, and define evaluation criteria. This is less about prompt engineering and more about structured thinking.

AI does not diminish the importance of cognitive workers, but rather it raises the threshold for what is considered *meaningful contribution*. For policymakers, this shift implies that skilling strategies must prioritize domain depth, analytical reasoning, structured problem solving, and continuous learning from an early age.

Example: User prompts and AI working behind the scene

1. A user (e.g. employee) is looking for something something? E.g. *tell me what earned leave rules?*
2. This information is not available in public domain and very specific to my organization.
3. Where and how to **get** the relevant knowledge based on what the user is looking for? Which knowledge source e.g. document(s), which part(s) of document etc.
4. Ohk! I **got** the relevant text from HR leave document! What to do next? How can I put that in the context of what user is looking for and **generate** answer?
5. Response is given to the user "*you can avail 30 days of earned leave in a year, you can carry over unused earned leave next year, maximum you can accumulate 300 days overall …* "
6. User now asking "*what about my earned leave status*"?
7. From where I can get data the user is asking for? What I need to do that? I checked user is logged in.
8. This information is available in my ERP system. Let me find out ***which API/DB Code*** would return me requested information. I got an API: *Get Leave Balance* to make a call.
9. API called, got response: 40. Let me create response to the user: "*you have 40 days earned leave in your leave account*"
10. Now user is asking "*I want to avail 10 days leave from 10.9.2026*". Let me check whether such operations are allowed or not.
11. ***Got an API/WF which does this operation***. Let me confirm with user whether the user wants to proceed. *Do you want to go ahead?* user says yes!
12. API is executed, it sent response." *Your 10 days leave from 10.9.2026 has been sent for approval to SK Raman …. Any thing else … I can help you*? "
13. User responds: "*No thanks*"

Understand user prompt. Understand what is required (context). Know where is what and how to get it. Make relevant, structured and faster search (retrieval) by drill down. Respond with relevant answer. How to do everything with least cost and efficiently?

Example: User prompts and AI working behind the scene

1. A user (e.g. employee) is looking for something something? E.g. *tell me what earned leave rules?*
2. This information is not available in public domain and very specific to my organization.
3. Where and how to **get** the relevant knowledge based on what the user is looking for? Which knowledge source e.g. document(s), which part(s) of document etc.
4. Ohk! I **got** the relevant text from HR leave document! What to do next? How can I put that in the context of what user is looking for and **generate** answer?
5. Response is given to the user "*you can avail 30 days of earned leave in a year, you can carry over unused earned leave next year, maximum you can accumulate 300 days overall …* "
6. User now asking "*what about my earned leave status*"?
7. From where I can get data the user is asking for? What I need to do that? I checked user is logged in.
8. This information is available in my ERP system. Let me find out ***which API/DB Code*** would return me requested information. I got an API: *Get Leave Balance* to make a call.
9. API called, got response: 40. Let me create response to the user: "*you have 40 days earned leave in your leave account*"
10. Now user is asking "*I want to avail 10 days leave from 10.9.2026*". Let me check whether such operations are allowed or not.
11. ***Got an API/WF which does this operation***. Let me confirm with user whether the user wants to proceed. *Do you want to go ahead?* user says yes!
12. API is executed, it sent response." *Your 10 days leave from 10.9.2026 has been sent for approval to SK Raman …. Any thing else … I can help you*? "
13. User responds: "*No thanks*"

Example: Use AI for AI, sounds interesting! It is reality!

Gemini example:

1. Can you convert this into professional diagram/PPT?
2. Another picture/PPT where every step, it should depict which AI technology e.g. RAG, Prompt Engineering, Context engineering, MCP etc. would play role etc.
3. Third picture/PPT it should show how various tools can be used to implement the steps etc.

Next step … ?

The 13-Step Enterprise AI Interaction Blueprint

PART 1: SCENARIO OVERVIEW

A complete enterprise interaction spans **Knowledge RAG**, **Database Querying**, and **Transactional Operations** with Human-in-the-Loop approval.

PHASE 1

Policy Discovery (RAG)

Unstructured document knowledge lookup for organization-specific leave rules.

Step 1: User prompt on leave rules
Step 2: Private domain validation
Step 3: Doc knowledge search
Step 4: Context building & RAG
Step 5: Grounded policy answer

PHASE 2

Data Retrieval (API)

Structured enterprise DB / ERP lookup for specific user leave balance.

Step 6: Prompt: leave balance status
Step 7: User identity/auth check
Step 8: ERP API function binding
Step 9: API call & balance summary

PHASE 3

Transactional Action

Executing leave application action with explicit user confirmation guardrail.

Step 10: Request to apply 10-day leave
Step 11: Policy check & confirm prompt
Step 12: Workflow trigger & approval

PHASE 4

Session Close

Dialogue finalization and session context clearing.

Step 13: User closure ("No thanks") & graceful thread termination

Part 1: Professional Workflow & Sequence Diagram

STEPS 1 TO 13 FLOW

1 User Inquiry
"Tell me what earned leave rules?"

2 Domain Check
Classified as private internal org knowledge

3 Knowledge Route
Identify HR policy documents & chunk sources

4 Rag Context
Retrieved HR text chunk; set generation context

5 Policy Response
"30 days/yr, carry forward max 300 days..."

↓ Transition From Unstructured Knowledge To Structured Data Query

6 User Data Query
"What about my earned leave status?"

7 Auth & Intent
Verify SSO logged-in session & employee ID

8 Api Discovery
Map intent to ERP endpoint GetLeaveBalance

9 Api Execution
API returns 40 days -> "You have 40 days..."

↓ Transition From Data Retrieval To Transactional Workflow Execution

10 Action Request
"Avail 10 days leave from 10.9.2026"

11 Policy & Confirm
Locate submit API; Prompt "Do you want to proceed?"

12 Workflow Trigger
Execute approval WF -> Sent to manager SK Raman

13 Session Close
User: "No thanks" -> Conversation complete

Part 2: Mapping AI Technologies Across the 13 Steps

AI TECH MAPPING

Step	User & System Interaction Step	AI Technology / Paradigm Operating	Technical Capability
1-2	User prompt: "What are leave rules?" & Domain Classification	PROMPT ENGINEERING, GUARDRAILS	Intent Classification, Input Moderation, Domain Boundary Detection
3-4	Locating knowledge source & retrieving HR document text	RAG, VECTOR SEARCH, CONTEXT ENG	Semantic Embedding Search, Chunk Retrieval, Context Window Construction
5	Generating policy response (30 days/yr, 300 max carry over)	GROUNDED LLM GENERATION	Hallucination Mitigation, Natural Language Summarization, Citation
6-7	"What about my leave status?" & Checking auth / logged-in state	MULTI-TURN MEMORY, CONTEXT MANAGEMENT	Session State Tracking, Identity Binding, Conversation History Indexing
8-9	Mapping to ERP GetLeaveBalance API & responding with 40 days	FUNCTION CALLING / TOOL USE, MCP PROTOCOL	Schema Binding, OpenAPI Dynamic Selection, JSON Response Parsing
10-11	"Avail 10 days leave" request & confirmation prompt	HUMAN-IN-THE-LOOP (HITL), AGENTIC PLANNING	Policy Validation Rule Engine, Dynamic Confirmation Safeguard
12-13	Executing Leave Workflow to SK Raman & Session Closure	ENTERPRISE WORKFLOW / MCP ACTION, SESSION END	Transactional API Mutation, Async Event Triggering, Memory Clearing

Deep Dive: Core AI Technologies in the Architecture

PART 2: TECHNOLOGY SPECTRUM

1. Retrieval-Augmented Generation (RAG)

Used in Steps 3–5 to fetch static policy rules from enterprise documents without model fine-tuning.

- **Embedding Models:** Converting HR text chunks into dense vector space.
- **Vector Search:** Cosine similarity matching user query against policy DB.
- **Context Injection:** Passing retrieved context directly into System Prompt.

2. Function Calling & Model Context Protocol (MCP)

Used in Steps 8–9 and Step 12 to query and mutate structured ERP database records.

- **Tool Schema Definition:** Exposing OpenAPI JSON schemas to LLM.
- **MCP Integration:** Standardized client-server protocol for tool discovery.
- **JSON Output Parsing:** Extracting parameters like emp_id and leave_days.

3. Human-in-the-Loop (HITL) & Guardrails

Used in Step 11 to ensure high-risk enterprise mutations require explicit user confirmation.

- **Action Confirmation:** Halting execution before write calls.
- **Auth Verification:** Ensuring user context is verified prior to execution.
- **Policy Enforcement:** Validating rules before submitting leave.

4. Context Engineering & Conversation Memory

Used in Steps 6–7 and Step 10 to maintain multi-turn context state across dialogue.

- **Windowed Chat History:** Tracking user intent across 13 turns.
- **Entity Resolution:** Remembering user reference ("leave balance").
- **State Management:** Storing intermediate tool execution results.

Part 3: Mapping Tools, Frameworks & Platforms

IMPLEMENTATION STACK

1. AI Orchestration Frameworks

Chaining prompts, memory, RAG, and tool calls.

LangChain / LangGraph
Stateful multi-turn agent graphs, memory, & tool routing (Steps 6-12).

LlamaIndex
Document ingestion, chunking, and vector index retrieval for HR policy (Steps 3-5).

Semantic Kernel / CrewAI
Enterprise agent plugin architecture & multi-agent delegation.

2. Knowledge & Protocol Stack

Databases, protocol servers, and enterprise systems.

Vector DBs (Pinecone / Qdrant)
Stores HR leave policy embeddings for rapid top-k similarity search (Step 3-4).

Model Context Protocol (MCP)
Standardized tool server exposing ERP endpoints like GetLeaveBalance (Step 8, 11).

SAP / Workday REST APIs
Enterprise ERP backend executing SQL database reads and workflow triggers (Step 9, 12).

3. Platform & Guardrail Layer

Foundation models, hosting platforms, and security.

LLM Engines (OpenAI / Claude / vLLM)
Core intelligence for reasoning, NLG, and native Function Calling (GPT-4o, Claude 3.5).

Guardrails (NeMo / LlamaGuard)
Checking query scope, redacting sensitive PII, and enforcing safety boundaries (Step 2).

Observability (LangSmith / Arize)
Tracing token usage, latency, and tool invocation accuracy across all 13 steps.

End-to-End Enterprise Technical Architecture

PART 3: UNIFIED SYSTEM VIEW

USER INTERFACE

SLACK / TEAMS BOT | STREAMLIT / REACT UI | SSO AUTHENTICATION (OAUTH2 / SAML)

AI AGENT ENGINE

LangGraph State Machine | Intent Router | Memory Store (Redis) | Guardrails & Moderation

UNSTRUCTURED RAG PIPELINE (Steps 3-5)

LlamaIndex + Embedding Model + Pinecone Vector Store (HR Leave Documents)

STRUCTURED TOOL / MCP LAYER (Steps 8-12)

MCP Server + OpenAPI Function Bindings + Human Approval Controller

BACKEND SYSTEMS

ENTERPRISE ERP (SAP/WORKDAY) | HR DATABASE (SQL) | WORKFLOW ENGINE (CAMUNDA/SERVICENOW)

Detailed Step-by-Step Implementation Matrix

TECHNICAL REFERENCE

Step	User / Agent Dialogue	AI Mechanism	Tool / Framework Used	Behind-The-Scenes Action
1-2	"Tell me leave rules?"	Prompt & Intent Routing	LangChain Router / Guardrails	Classifies query as org-specific policy inquiry.
3-4	Locate knowledge & fetch text	RAG Vector Search	LlamaIndex + Pinecone DB	Embeds query, searches vector index, extracts top chunks.
5	"You can avail 30 days..."	Grounded NLG Generation	OpenAI GPT-4o / Claude 3.5	Synthesizes response with zero hallucination.
6-7	"What about my status?"	State & Identity Binding	Redis Session Store + OAuth	Extracts user ID from authenticated session token.
8-9	Executes GetLeaveBalance	Function Calling / MCP	Model Context Protocol (MCP)	Calls ERP API; receives balance 40; formats output.
10-11	"Avail 10 days leave" & Confirm	Human-in-the-Loop (HITL)	LangGraph Interrupt / Human Node	Pauses graph execution; prompts user for explicit consent.
12-13	Leave sent to SK Raman & End	Transactional Action & Close	REST API / Enterprise Workflow	Triggers workflow mutation API & resets session state.

Key Agentic AI Patterns Highlighted in the Scenario

PATTERN ANALYSIS

PATTERN 1 (STEPS 1-5)

Knowledge-Augmented Pattern (RAG)

Read-only retrieval from static, unstructured knowledge bases.

Key Characteristics:
Deterministic chunk retrieval, no write side-effects, fast vector lookup, hallucination safeguard.

PATTERN 2 (STEPS 6-9)

Tool-Augmented Pattern (Function Calling / MCP)

Structured data retrieval from transactional SQL / ERP databases via APIs.

Key Characteristics:
Dynamic schema binding, exact numerical responses (e.g. 40 days), identity context aware.

PATTERN 3 (STEPS 10-13)

Action-Oriented Workflow Pattern (HITL)

Autonomous state mutation backed by explicit human confirmation guardrails.

Key Characteristics:
Stateful graph execution, transactional API side-effects, human approval pause node.

Strategic Takeaways for Enterprise AI Practitioners

LEADERSHIP SUMMARY

1. Hybrid Architecture is Mandatory

Real enterprise use cases require combining RAG (for unstructured documents) with Tool Calling/MCP (for structured live ERP data). Neither alone is sufficient.

2. MCP Standardizes Tool Integration

Model Context Protocol (MCP) decouples LLMs from custom API glue code, allowing seamless, secure connection to systems like SAP, Oracle, and ServiceNow.

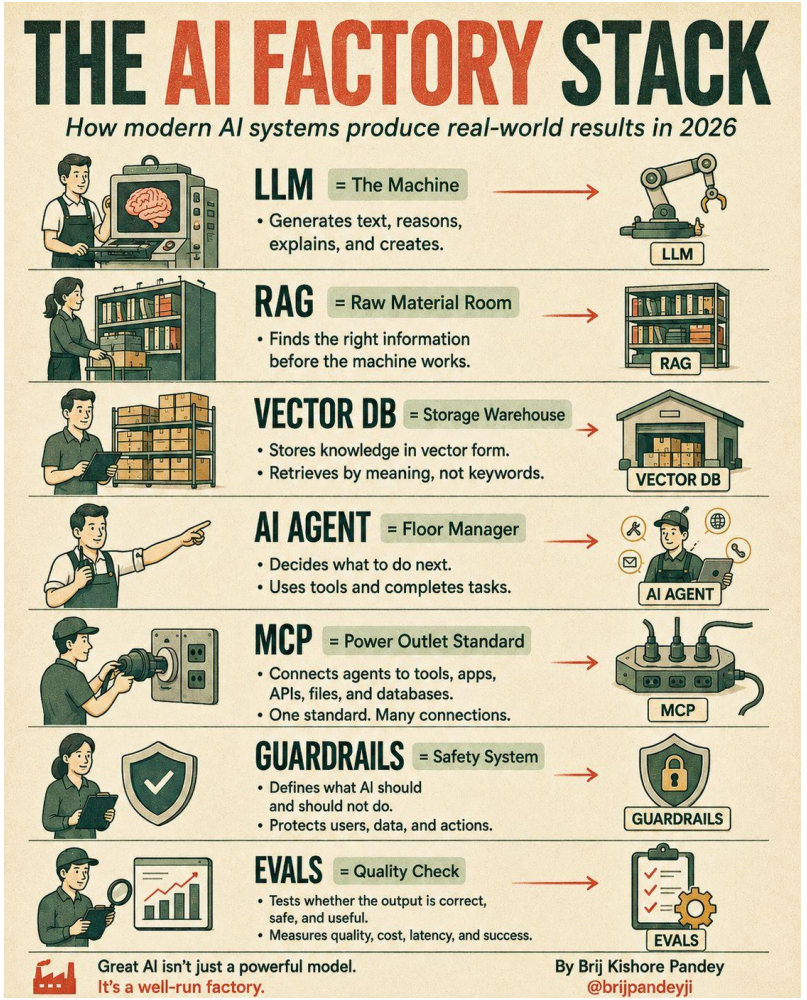
3. Guardrails & HITL Build Enterprise Trust

Never allow autonomous state mutations without Human-in-the-Loop approval nodes (as shown in Step 11 for leave application submission).

4. Choose Frameworks Based on State Complexity

Use LlamaIndex for RAG ingestion & search, LangGraph for stateful agent workflow loops, and standard OAuth for enterprise identity context binding.

Think over, what else you can do?



https://lnkd.in/p/dBv5b7cP

What else is missing in this picture?

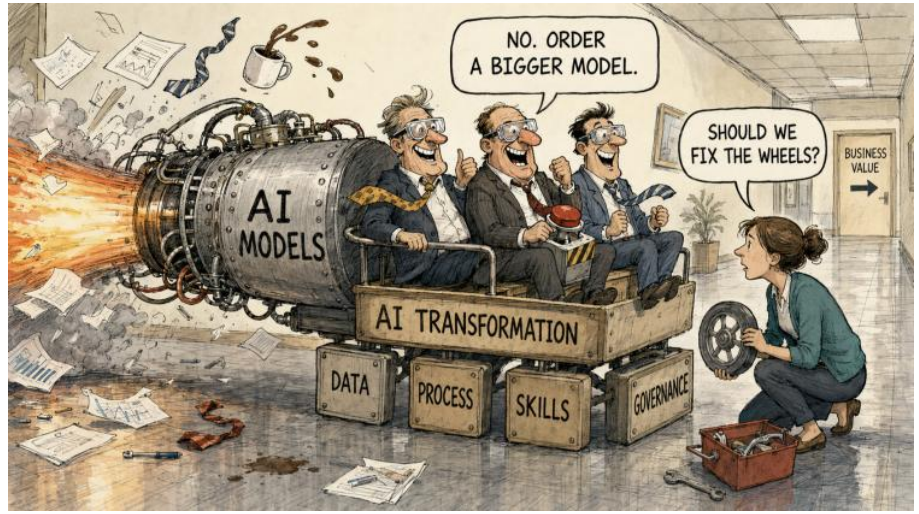
Revisit: Role of Knowledge sources and types & RAG

Sources

- Documents | Reports
- Internet
- Social Networks
- Sensors and devices
- Past solved cases
- Databases | Data (including multi-media Files
- People
- Automated Systems

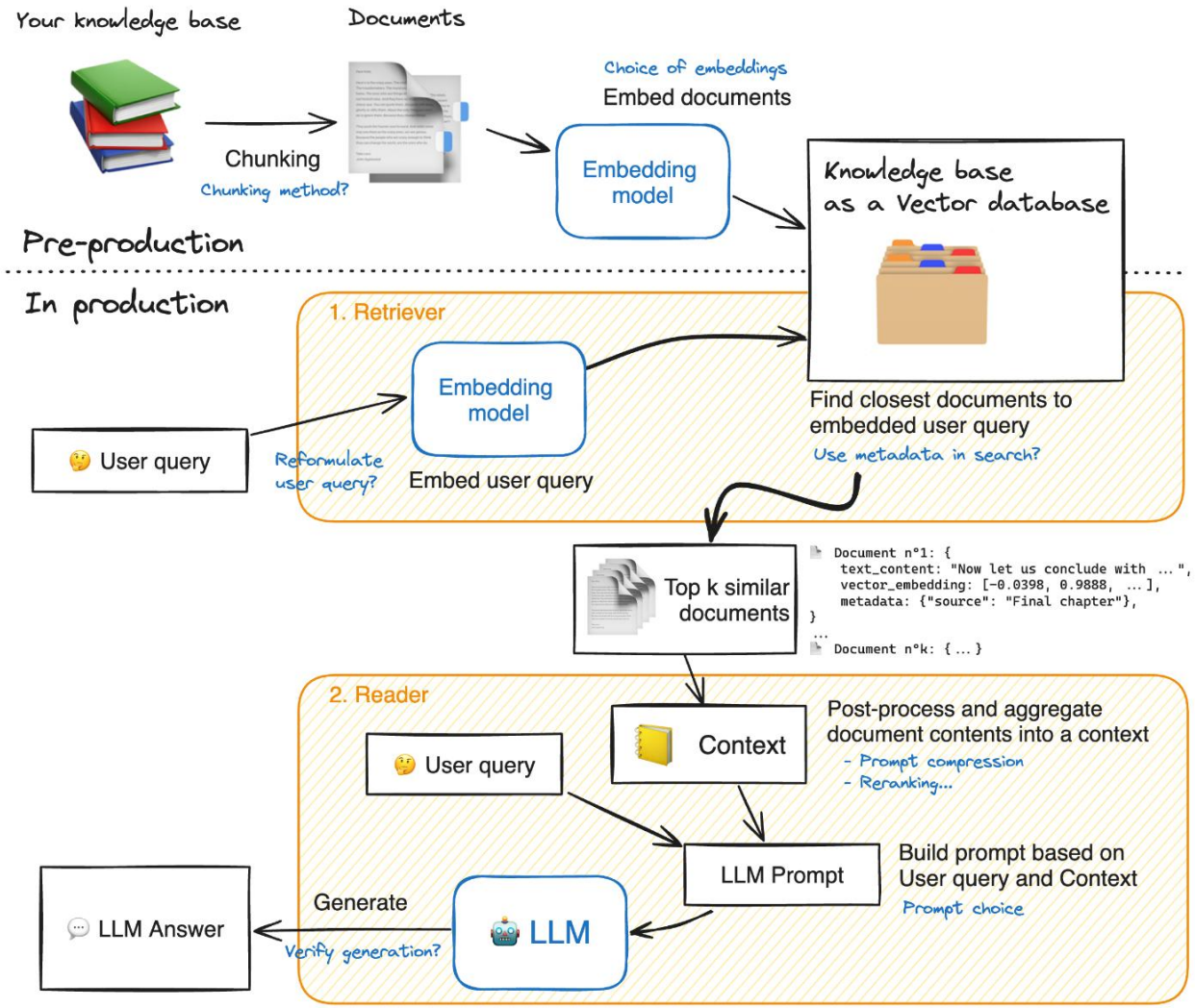
Types

- Master Data
- Transactional | Usage
- Documents
- Metadata
- External data
- Augmented data
- Geo, location, time data, etc.



What else is missing in this picture?

Revisit: Role of knowledge sources and types & RAG



Vector databases mostly deal with text data and not meant for numeric computational capabilities and searching capabilities (like search within data tables, decision tables, range tables etc.). Suggest not to include numeric (especially which are domain sensitive e.g. customer age in insurance, finance, health kind of domains) parameters. Use technologies like CBR for semantic retrieval.

Do not unnecessarily dump already structured data (like already stored in DBs unless lot of semantic and text data is involved) for the sake of it. Putting data in vector database limits search that is otherwise possible using complex SQL queries. Gen AI tools are good in converting user queries into SQL.

Context engineering can be challenging if lot of data resides in vector databases.

Think over, how would you manage effective retrieval from RAG for such documents?

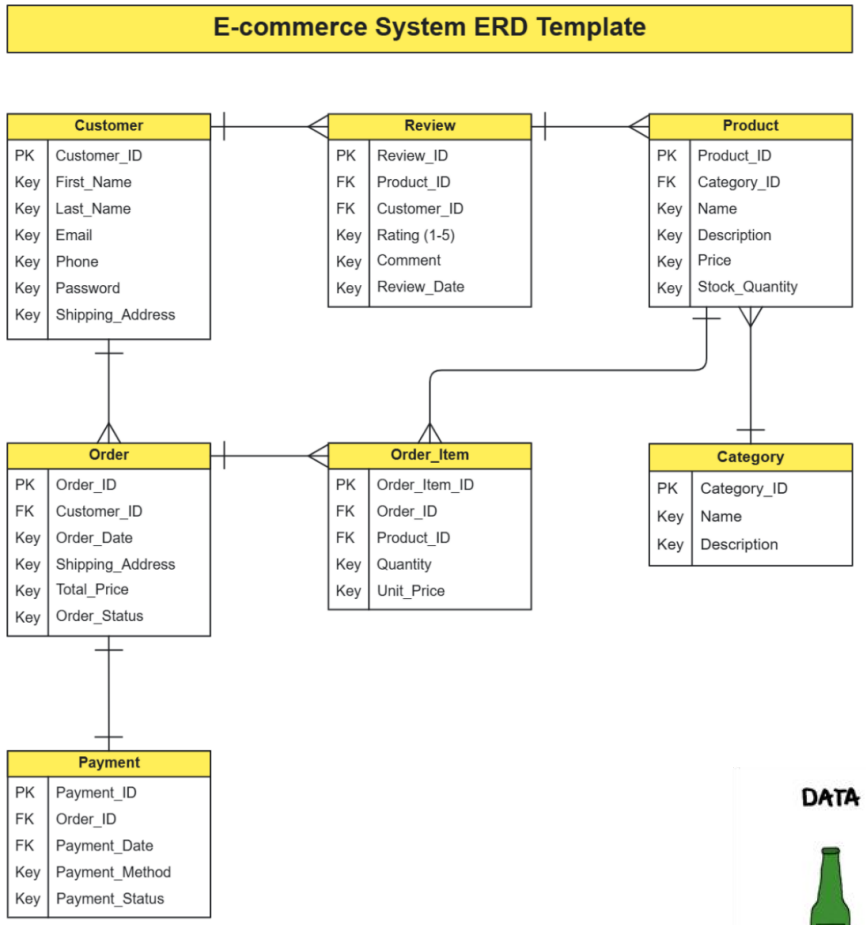
Policy Provisions of the Government of Maharashtra:

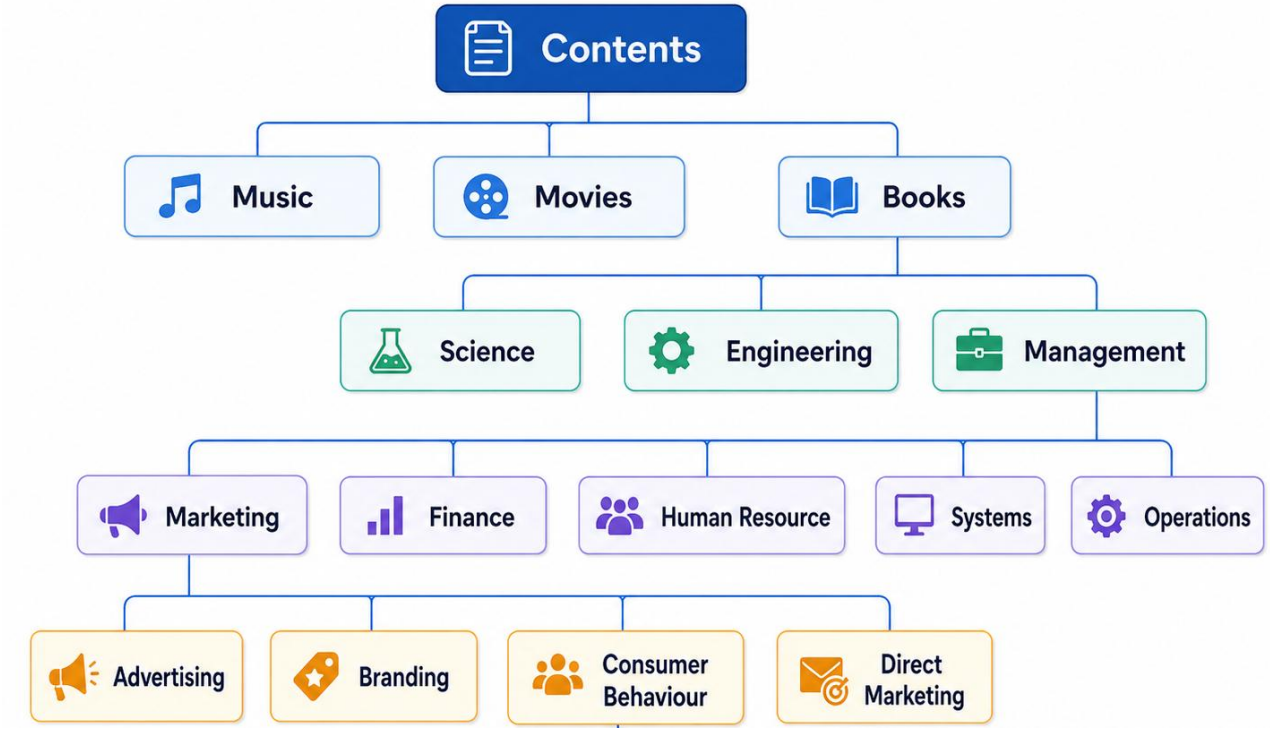
- Additional benefits under the package scheme of incentives: Coir industries in A and B zone will be eligible for benefits available to industries in C zone. Industries of C zone will get benefits of D zone and D zone industries will get benefits of D+ zone. Industries in D+ and no industrial zone will be eligible for benefits in naxal affected area. The new package scheme of incentives will be announced soon.
- Special Capital Incentives:

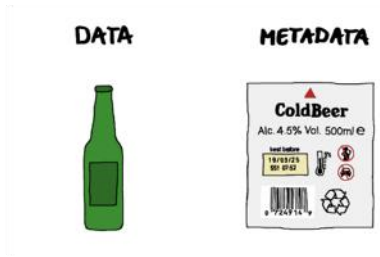
S. No.	Classification of Talukas	Capital Subsidy (% fixed capital investment)	Maximum ceiling for special capital subsidy (in Rs. lakhs)
1.	A and B	30	20
2.	C	35	30
3.	D	35	40
4.	D+	35	50
5.	Non-industry and Naxal affected area	35	50

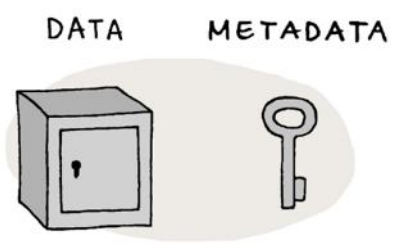
(Source: Maharashtra Coir Policy - 2018)

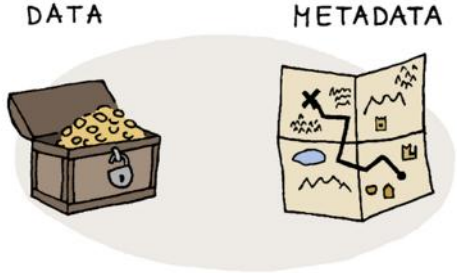
Revisit: Role of metadata, taxonomies, data models





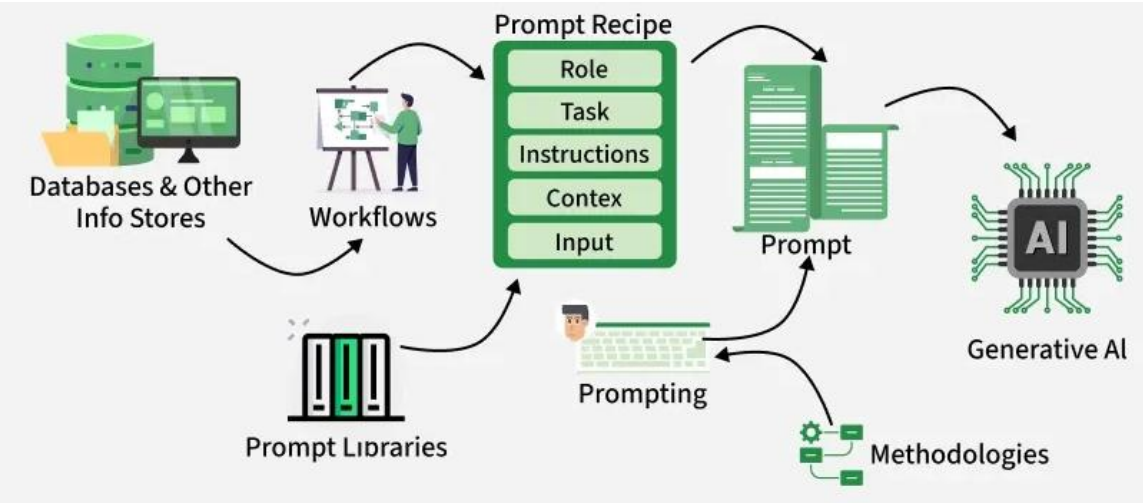


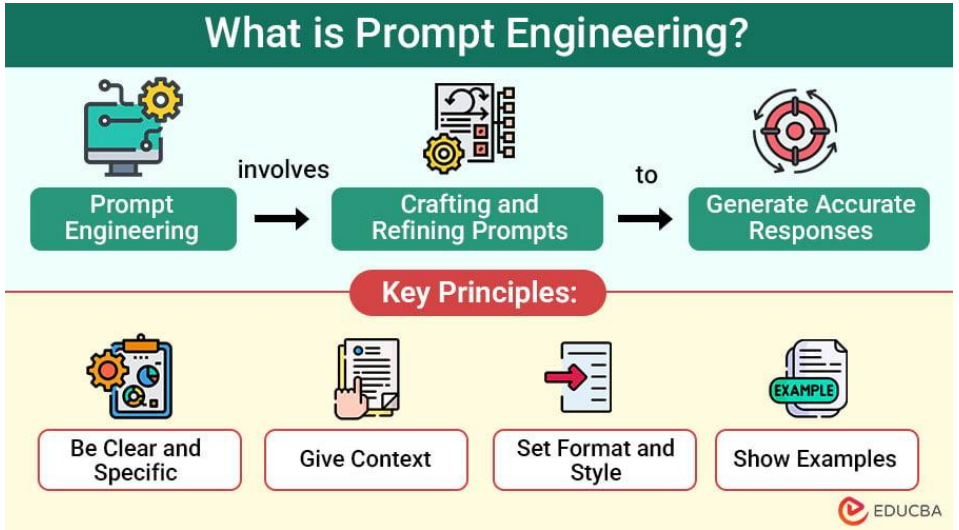


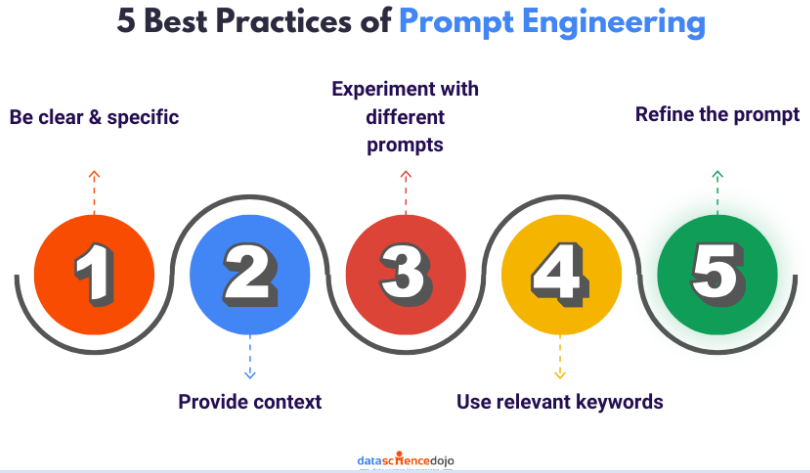


Revisit: Prompt Engineering

Prompt engineering is the practice of designing and refining input instructions (prompts) to guide generative AI models, like large language models (LLMs), to produce accurate, relevant, and desired outputs, essentially "programming" the AI in natural language to achieve specific goals, from writing code to creating content. It involves understanding model capabilities, providing context, specificity, and structure, often through techniques like few-shot or chain-of-thought prompting, to bridge the gap between human intent and machine understanding







[What-is-an-ai-prompt-engineering]

[Prompt-engineering-best-practices]

[Prompt-engineering]

[Prompt Engineering Types]

Prompts can change based on user is tech user, general user or domain specialist and business user. Prompts used (**context engineering!**) to automate LLM tasks (like in AI agents) using LLMs (mostly driven by APIs) are structured and dynamic; and need to be modelled differently especially with formatted input and output.

Revisit: Prompt Engineering

Content Generation

- Create contents (text, images, videos, docx, PPTs, etc.)
- Generate technical contents (code, database schema etc.)

Multi-modal tasks

- Convert content from one mode to another e.g. 1> converting audio to transcripts, 2> extracting text from image (e.g. OCR), 3> explaining pictures based on the context, 4> search contents, …

Language translation

- Translation from one language to another e.g English to Hindi

Conversion of data

- Converting from one format to another format (this can be simple plain or semantic conversion) e.g. 1> excel horizontal format (tedious to read) to vertical format word doc, 2> CSV to JSON, 3> CSV to database queries
- Convert from unstructured to technical formats like Json, XML, Rule, Code etc. e.g.
  Image to meta-data in attribute value pair,
  NLP text into JSON,
  HR leave rules into rules,
  NLP text into SQL (based on schema provided),
  NLP text into named entity pairs…

Template based extraction

- Extracts data from predefined templates e.g. PAN card to data
- Reading forms/printed outputs

Summarization and insights

- Get insights from documents (excel, docs, pdfs etc.)
- Showing contents using visualization

Research, Simulation and exploration

- Review kind of work (understand what has happened, what is happening, who has done/said what etc.)
- Market research on specific product, getting validations done and understanding current status (e.g. where your product fits, what are the competing products, how their features differ from your product's features?)
- Exploring how you can move from where are you right now to the target (e.g. 1> how you can reach to niche product X1 from your existing product X, 2> how you can enhance your skills to specific job)

To create prompt itself

- If you don't know a specific prompt for your task you can ask GenAI to create on for you and you can go on refining it

Prompt Engineering: Good prompt

Role:
Act as [relevant expert].

Task:
Do [specific task].

Context:
Use the following background information: [...]

Requirements:
- Include [...]
- Exclude [...]
- Consider [...]

Output:
Present the answer as [table, bullets, report, code, JSON, etc.].

Constraints:
Keep it within [length, tone, scope, technical level].

1. Be specific about the task

Weak:

```
Tell me about AI.
```

Better:

```
Explain artificial intelligence to MBA
students in approximately 300 words.
Cover its meaning, major capabilities,
business applications, and limitations.
```

Prompt Engineering: Good prompt

2. Provide relevant context

```
I am preparing a three-hour
introductory session for senior
managers who have
business experience but limited
technical knowledge.
```

Context helps the model select the right depth, language, and examples.

3. Define the intended audience

The same topic may need very different explanations for:

- school students,
- senior executives,
- software developers,
- researchers,
- customers.

Example:

```
Explain APIs to non-technical business
leaders using a restaurant analogy.
```

3. Define the intended audience

The same topic may need very different explanations for:

- school students,
- senior executives,
- software developers,
- researchers,
- customers.

Example:

```
Explain APIs to non-technical
business leaders using a restaurant
analogy.
```

Prompt Engineering: Good prompt

4. Specify the expected output

Instead of:

```
Analyse this proposal.
```

Use:

```
Analyse this proposal and return a table
with:
1. Strengths
2. Weaknesses
3. Risks
4. Missing information
5. Recommended improvements
```

5. State important constraints

```
Use simple language.
Avoid mathematical notation.
Limit the response to five bullets.
Do not recommend paid tools.
```

Constraints reduce irrelevant or overly broad answers.

6. Give examples when format matters

```
Write each rule in this format:

Rule name:
Condition:
Action:
Explanation:
```

Examples are especially useful for code, structured data, rules, reports, and classifications.

7. Ask the model to distinguish facts from assumptions

```
Clearly separate:
- facts provided in the prompt,
- assumptions you are making,
- recommendations.
```

This improves reliability.

8. Ask for gaps or ambiguities to be identified

```
Before giving the final recommendation,
identify any missing information or
ambiguous requirements that could materially
affect the answer.
```

This is useful for complex analysis, policies, system design, and planning.

Prompt Engineering: Good prompt

9. Break complex work into stages
Instead of asking for everything at once:

```
Step 1: Extract the major requirements.
Step 2: Group them into functional and non-functional requirements.
Step 3: Identify gaps and conflicts.
Step 4: Propose the system design.
```

This usually produces a more organized answer.

10. Avoid unnecessary instructions
Prompts do not need excessive politeness or repeated instructions.
Verbose:

```
Please carefully and thoroughly review the information below and provide a very clear and meaningful answer after considering every possible aspect.
```

Better:

```
Review the information below. Identify the main issues and recommend solutions.
```

A good prompt is clear about the objective, rich in relevant context, and precise about the expected output—but contains no unnecessary information.

Prompt Engineering: Good prompt

You are acting as: [role]

Objective:
[What I want to achieve]

Background:
[Essential context]

Task:
[Exact work to perform]

Requirements:
- [Requirement 1]
- [Requirement 2]
- [Requirement 3]

Output format:
[Bullets/table/report/code/JSON]

Audience:
[Who will use the answer]

Tone and level:
[Formal, simple, technical, executive, academic]

Constraints:
[Length, exclusions, technologies, assumptions]

Quality check:
Identify missing information, assumptions, and possible risks.

Act as an enterprise AI consultant.

Objective:
Help a manufacturing company identify suitable AI opportunities.

Background:
The company has production, quality-control, maintenance, inventory, and customer service functions. It has historical operational data but limited AI expertise.

Task:
Suggest ten practical AI use cases.

For each use case, provide:
- business problem,
- AI technique,
- required data,
- expected benefit,
- implementation difficulty,
- major risk.

Output:
Present the answer in a table.

Audience:
Senior business managers.

Constraints:
Use simple language. Avoid highly experimental use cases. Prioritize solutions that can be piloted within six months.

Prompt Engineering: Multi-modal prompts and key suggestions

For **multimodal prompts**—where you provide text along with images, audio, video, charts, screenshots, or documents—the same principles apply, with a few additional practices.

State what the model should inspect

- Do not simply attach an image and say "analyse this."

Better:

Examine the process diagram. Identify missing steps, unclear connections, and possible improvements.

Point to the relevant region or element

- Specify page, section, object, row, time range, or visual area.

Focus on the bottom-right section of the diagram.
Analyse pages 5–8 only.
Review the chart between 2:10 and 3:00 in the video.

Describe the relationship between modalities

- Explain how the text and image should be used together.

Use the policy text to validate whether the workflow shown in the diagram is complete.

Assign a purpose to each input

- This reduces confusion when several files are attached.

Image 1 shows the current design.
Image 2 shows the preferred style.
The document contains the functional requirements.

Ask the model not to infer unreadable content

- This is especially important for screenshots, scanned pages, and charts.

If any text is unclear, identify it rather than guessing.

Specify the expected output clearly

- Ask for annotations, comparison, extraction, explanation, redesign, or summary.

Return a table with: observed issue, evidence from the image, impact, and recommendation.

Separate observation from interpretation

- This improves accuracy.

First describe what is visible. Then provide your interpretation and recommendations.

Use reference labels

- When multiple images or files are involved, label them consistently.

Refer to them as Diagram A, Diagram B, and Document C.

Provide visual intent for image generation

- Mention composition, layout, style, audience, aspect ratio, and text requirements.

Create a clean executive diagram with three horizontal layers, minimal text, white background, and 16:9 layout.

Avoid overloading one prompt

- For complex multimodal tasks, use stages:

Step 1: Extract visible information.
Step 2: Compare it with the document.
Step 3: Identify gaps.
Step 4: propose a revised

Prompt Engineering: Multi-modal prompts and key suggestions

Inputs:
- Image A: current process diagram
- Document B: functional requirements
- Image C: preferred visual style

Task:
Compare Image A with Document B and identify missing or incorrect elements.

Focus:
- process steps
- decision points
- data flows
- user roles

Output:
1. Observations
2. Gaps
3. Recommended changes
4. Revised diagram description

Important:
- Use Image C only as a style reference.
- Do not assume unreadable text.
- Clearly distinguish visible evidence from inference.

Prompt Engineering: Reducing Size

Remove unnecessary context

- Include only information directly relevant to the current question.

Summarize earlier conversation

- Replace long chat history with a compact summary of decisions, constraints, and unresolved points.

Avoid repetition

- Do not repeat instructions, examples, background, or definitions already established.

Use concise structured formats

- Tables, JSON, key-value pairs, and bullet points usually require fewer tokens than descriptive paragraphs.

Retrieve only relevant information

- For large documents or knowledge bases, insert only the relevant passages rather than the entire source.

Use references or identifiers

- Define something once and later refer to it by a short label, such as RuleFormat-A or CustomerSchema.

Reduce the number of examples

- One strong example is often more useful than five similar ones.

Shorten instructions

- Replace verbose instructions such as:
- Please carefully analyse the information provided below and return your response in a clear and concise manner with:
- Analyse the following. Respond concisely.

Separate large tasks into stages

- First extract information, then summarize it, then use the compact result for the next task.

Move fixed instructions to reusable configuration

- Stable behaviour, tone, formatting rules, and output schemas should not be repeated in every prompt.

Moving from Tokenmaxxing to Valuemaxxing

Approach	Definition	The Problem / Limitation
Tokenmaxxing *(Old Approach)*	Maximizing AI and token consumption across teams under the assumption that heavier usage automatically equals better ROI.	Treats usage volume as a proxy for value; leads to uncontrolled costs with unclear returns as agentic workflows scale.
Token Minimization *(Knee-Jerk Reaction)*	Aggressively capping tokens, shrinking context windows, and restricting model access to cut visible expenses.	**"Costs do not disappear; they move."** Stripping context leads to ambiguous prompts, failed tool calls, retries, and human rework.
Valuemaxxing *(Strategic Focus)*	Aligning AI consumption directly with business outcomes and operational efficiency rather than raw token counts.	Requires dynamic model routing, outcome-based KPIs, and shared accountability across engineering and leadership.

Core Takeaways

Tokens are a Cost Signal, Not a Value Metric:

- Stop asking: *"How many tokens did we consume?"*
- Start asking: *"How many tasks were completed, vulnerabilities resolved, or engineering hours saved?"*

Systems Matter More Than Standalone Models:

- Access to frontier models is no longer the sole differentiator. Advantage comes from surrounding infrastructure: orchestration, context hygiene, memory, and governance.
- **Dynamic Model Routing:** Rather than defaulting to the most expensive frontier model, systems should dynamically route tasks to open-source or task-specific models (IDC predicts 70% of AI-driven enterprises will do this by 2028).

Shared Accountability:

- **Developers:** Treat AI usage like cloud compute/APIs (mastering context hygiene and cost vs. quality trade-offs).
- **Leadership:** Measure and reward concrete value creation, provide clear guardrails, and track ROI instead of usage leaderboards.

Tokens Measure Cost, Not Value: "Tokenmaxxing" (maximizing AI consumption) mistook volume for productivity. Moving forward, success is measured by business outcomes—developer hours saved, tasks completed, and cycle times reduced—not tokens burned.

Cost-Cutting Without Strategy Backfires: Aggressively slashing tokens ("token minimization") starves models of vital context, simply shifting costs from token bills into retries, failed tool calls, and developer rework.

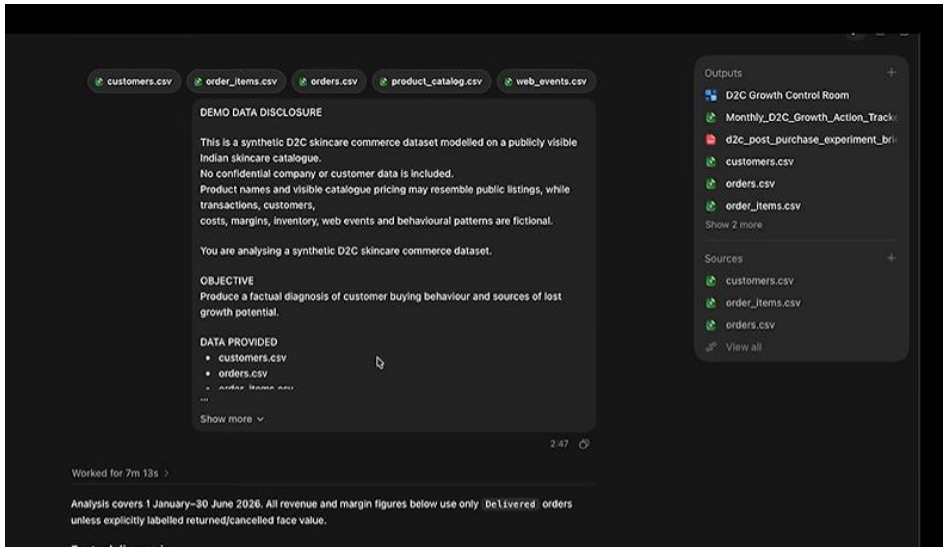
Competitive Advantage Lies in Orchestration: Peak ROI requires **Valuemaxxing**—leveraging intelligent model routing (directing simpler tasks to cost-effective models) and context governance rather than defaulting to the most expensive frontier models.

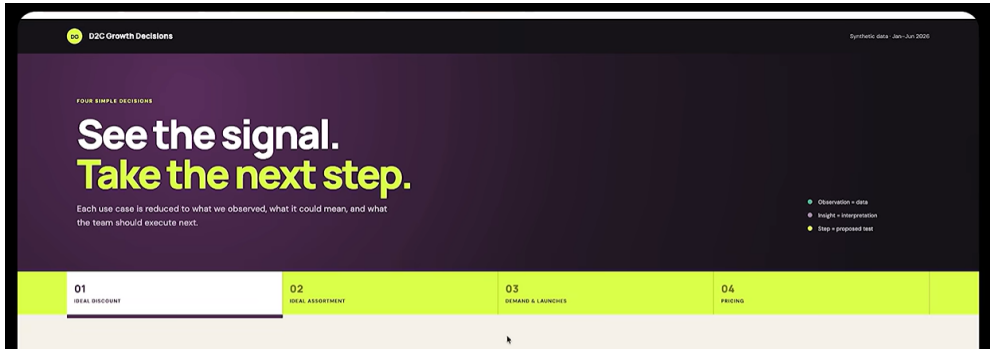
The experimentation phase of "use as much AI as possible" (tokenmaxxing) is ending. True AI maturity is **valuemaxxing**—optimizing workflow orchestration and model routing so every token spent translates directly to measurable business velocity and code quality.

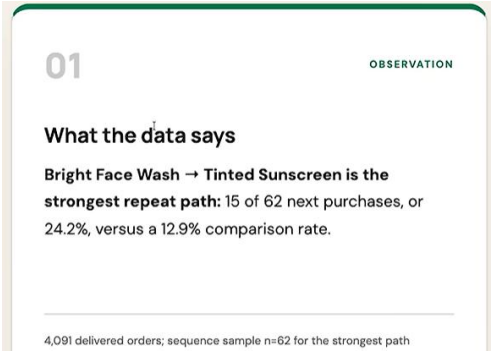
Shift AI governance from **consumption-driven** (volume) to **outcome-driven** (ROI) through smarter model orchestration and context management.

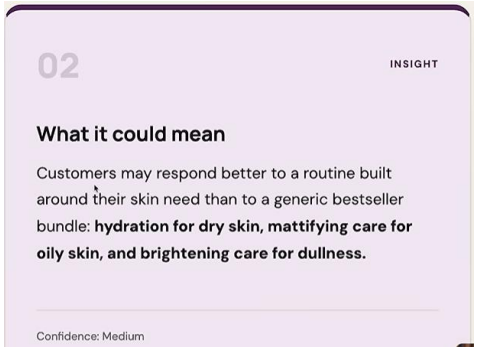
Tokenmaxxing dead long live valuemaxxing

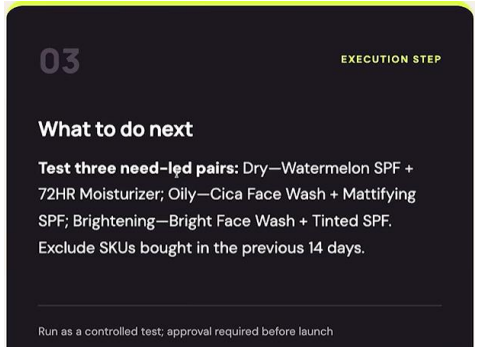
Automation of activities!











What AI/ML techniques/algos working behind the scenes that we have studied?

Do you want to share everything to Gen AI to get such insights?

Do you want to share data every time to get insights for new data?

What is your solution? To automate, save cost and keep your data private?

Chatgpt/Codex Use cases

From assistant to dependable work output

The practical maturity path for individual use.

Initial prompts are rarely the end product.

A professional workflow uses AI as a thinking partner: ask clearly, demand structure, supply context, refine the response, verify the result, then use it with accountability.

Demo anchor

Create an executive decision brief from rough notes, document excerpts and a small table of numbers.

A practical prompt framework

Use this as the default structure for most professional AI tasks.

1 Task	2 Context	3 Constraints	4 Output	5 Quality
What should the AI do?	What background should it know?	What rules, limits or tone apply?	What format should it produce?	How should it judge success?

Example: “Read this client note and produce a 1-page CXO brief with decision options, risks, missing data and recommended next steps.”

Tools / demo: Demo tools: ChatGPT / Gemini / Claude; compare weak vs structured prompt; request table or JSON output.

Context engineering: beyond a single prompt

Reliable outputs need relevant context, not only clever wording.

Documents

Policies, reports, proposals, manuals

Examples

Good outputs, templates, previous drafts

Data

Tables, metrics, transactions, logs

Definitions

Acronyms, terminology, business rules

Audience

CXO, client, team, regulator

Teaching point

Context engineering converts AI from a generic assistant into a task-aware collaborator. It is also the bridge to RAG and enterprise knowledge systems.

Tools / demo: Demo tools: ChatGPT with files, NotebookLM, Microsoft Copilot; attach policy/report + ask grounded questions.

Representative tools for personal productivity

Do not teach every product; teach categories and selection criteria.

Category	Representative tools	Best for
General AI assistant	ChatGPT, Gemini, Claude	Drafting, reasoning, analysis
Enterprise copilot	Microsoft 365 Copilot	Office work and meeting context
Research assistant	Perplexity, Gemini, ChatGPT	Current and cited research
Document knowledge	NotebookLM, ChatGPT with files	Study and Q&A on documents
Presentation / visual	PowerPoint Copilot, Canva, Gamma	Slides and visuals
Coding assistant	GitHub Copilot, Cursor	Code, scripts, technical reviews

Tools / demo: Use as tool map: ChatGPT, Copilot, Gemini, Claude, NotebookLM, Perplexity, PowerPoint/Excel Copilot, Canva/Gamma.

Demo 1: Turn messy inputs into an executive brief

A realistic classroom demonstration for leaders.

1 Input

Paste rough notes + attach document excerpts

2 Synthesize

Ask for issue summary, options and missing data

3 Structure

Request a table: risks, assumptions, decisions

4 Critique

Ask AI to critique its own brief

5 Finalize

Finalize a 1-page email or slide

Classroom tip: show a weak prompt first, then improve it using the framework.

Tools / demo: Demo tools: ChatGPT or Claude + PowerPoint Copilot/Canva; convert messy notes into CXO brief and slide outline.

Multimodal productivity: use more than text

Modern AI assistants can interpret many forms of business evidence.

Text

emails, policies, reports

Tables

budgets, project trackers

Images

screenshots, forms, diagrams

Audio

meeting recordings

Charts

interpretation and explanation

Key question for leaders

What evidence should be provided to the AI so that the output is grounded in reality rather than generic?

Tools / demo: Demo tools: ChatGPT Vision, Gemini, Claude, Copilot; analyse screenshot/table/chart and produce explanation.

Quality and safety checklist for personal AI use

Experienced users must verify, not blindly accept.

🔍 Accuracy

Can I verify the facts?

🧱 Completeness

What is missing?

🔒 Confidentiality

Am I exposing sensitive data?

⚖️ Bias

Is the output one-sided?

📜 Traceability

Can I explain how it was used?

👤 Accountability

Who owns the final decision?

Tools / demo: Practice tools: source links, browser search, file citations, checklists; show verification workflow before use.

Build a personal AI workbench

A lightweight operating model for individual adoption.

1. Approved assistant
2. Prompt templates
3. Document/context folders
4. Review checklist
5. Reusable outputs

Demo 2: reusable template

Build a prompt template for “meeting-to-decision brief”: agenda, context, participants, key discussion, decisions, risks, actions and follow-up mail.

Tools / demo: Build tools: Custom GPT / ChatGPT Projects, Copilot Studio, NotebookLM, prompt library in SharePoint/Confluence.

Leader’s practice lab for Session 1

Individual tasks participants can try during or after class.

1 Brief

Create a one-page brief from a report

2 Compare

Compare two vendor proposals

3 Extract

Convert free text into a table/JSON

4 Critique

Ask AI to find gaps and risks

5 Explain

Ask AI to teach an unfamiliar technical term

Tools / demo: Practice lab tools: ChatGPT, Copilot, NotebookLM, Perplexity, Excel Copilot, PowerPoint Copilot.

Key takeaways

What participants should remember and apply.

One-line takeaway

Personal AI productivity improves when prompts are structured, context is supplied, and outputs are verified before use.

1 Prompt clearly

State task, context, constraints, format and quality criteria.

2 Ground with context

Use documents, data, examples and terminology.

3 Use categories

Pick tools by task, not popularity.

4 Verify outputs

Check facts, gaps, bias and confidentiality.

5 Make reusable

Create templates, checklists and trusted workflows.

Source pointers: ChatGPT: chatgpt.com • Microsoft 365 Copilot: office.com • Hugging Face: huggingface.co