
Module 12: Agentic AI and AI Systems

Design, evaluate and govern AI agents that use tools and take actions — with the controls that make them safe enough to ship.

Product Management Master Handbook
v0.1 · 23 October 2026

Built from the 55-lecture course study guide and extended to the 12-module Product Core curriculum.
Hypothetical examples are labelled; no real company results are invented.

Contents

12.1 What an AI agent is	3
12.2 Agents versus chatbots versus workflows	4
12.3 Autonomous versus human-supervised systems	5
12.4 Agent architecture	6
12.5 Planning and reasoning workflows	6
12.6 Tool calling and function calling	7
12.7 Memory systems	8
12.8 Multi-agent systems and orchestration	8
12.9 Workflow automation	9
12.10 RAG-enabled agents	9
12.11 Evaluation and observability for agents	10
12.12 Human approval checkpoints, guardrails and permission boundaries	11
12.13 Agent security and prompt injection	11
12.14 Latency and operating costs	12
12.15 Failure recovery	12
12.16 Agentic product design	12
12.17 AI automation opportunities in finance	13
12.18 AI product roadmaps and agentic business models	13
12.19 Case study: AI-powered customer-support assistant for a card issuer (hypothetical)	14
12.20 Exercises with model answers	19
12.21 Interview perspective	20
12.22 Key takeaways	20
12.23 Check your understanding	21
12.24 Recommended resources	22

MODULE 12

Agentic AI and AI Systems

Design, evaluate and govern AI agents that use tools and take actions — with the controls that make them safe enough to ship.

≈ 12 h study

About this module's examples

The financial-services assistant is a **hypothetical case** built for teaching. It describes no real bank's product, data or results. Regulatory references are general; consult qualified compliance and legal advisers for real decisions.

12.1 What an AI agent is

An **AI agent** is a system that uses a model to **pursue a goal by choosing actions** — calling tools, reading results, and deciding what to do next — within limits set by people or policy.

From your course notes (Picture lesson 'AI agents'; Companion ch. 14; Lecture 48)

An agent can work toward a goal by **choosing steps, using tools and looking at results**; unlike a fixed recipe it may choose its next step based on what happened. **Some tasks need only a simple fixed workflow, not an agent.** Key pieces: a goal, a way to choose a next step, access to approved tools, information returned by tools, and a **stopping rule**; a person or policy sets the limits. Before booking or spending money, it should ask for permission. Define the goal, allowed tools and data, boundaries, maximum actions, budget, time limit, logging and stop conditions; **test missed data, misleading tool output, retries, duplicate actions and prompt injection**; for consequential actions require human approval and a way to reverse or correct errors.



The agent loop. The loop repeats until a stop condition is met (goal reached, budget/time exhausted, or human needed). Every iteration costs tokens, time and risk.

12.2 Agents versus chatbots versus workflows

	Chatbot	Workflow (deterministic automation)	Agent
Who decides the steps?	User drives each turn	Designer, in advance	The model, at run time
Uses tools?	Rarely	Yes, fixed sequence	Yes, chosen dynamically
Predictability	High (just talks)	High	Lower — must be bounded
Best for	Q&A, drafting	Known, repeatable processes	Open-ended tasks where the path varies
Example	FAQ assistant	"When form submitted → validate → create CRM record → notify"	"Resolve this billing dispute using these systems"

Level 2 · Working PM rule of thumb

Use the **simplest** pattern that works: a single model call → a fixed workflow → an agent. Anthropic's guidance on building effective agents makes the same point: many successful systems are workflows built from simple, composable patterns, and full autonomy is added only when needed.

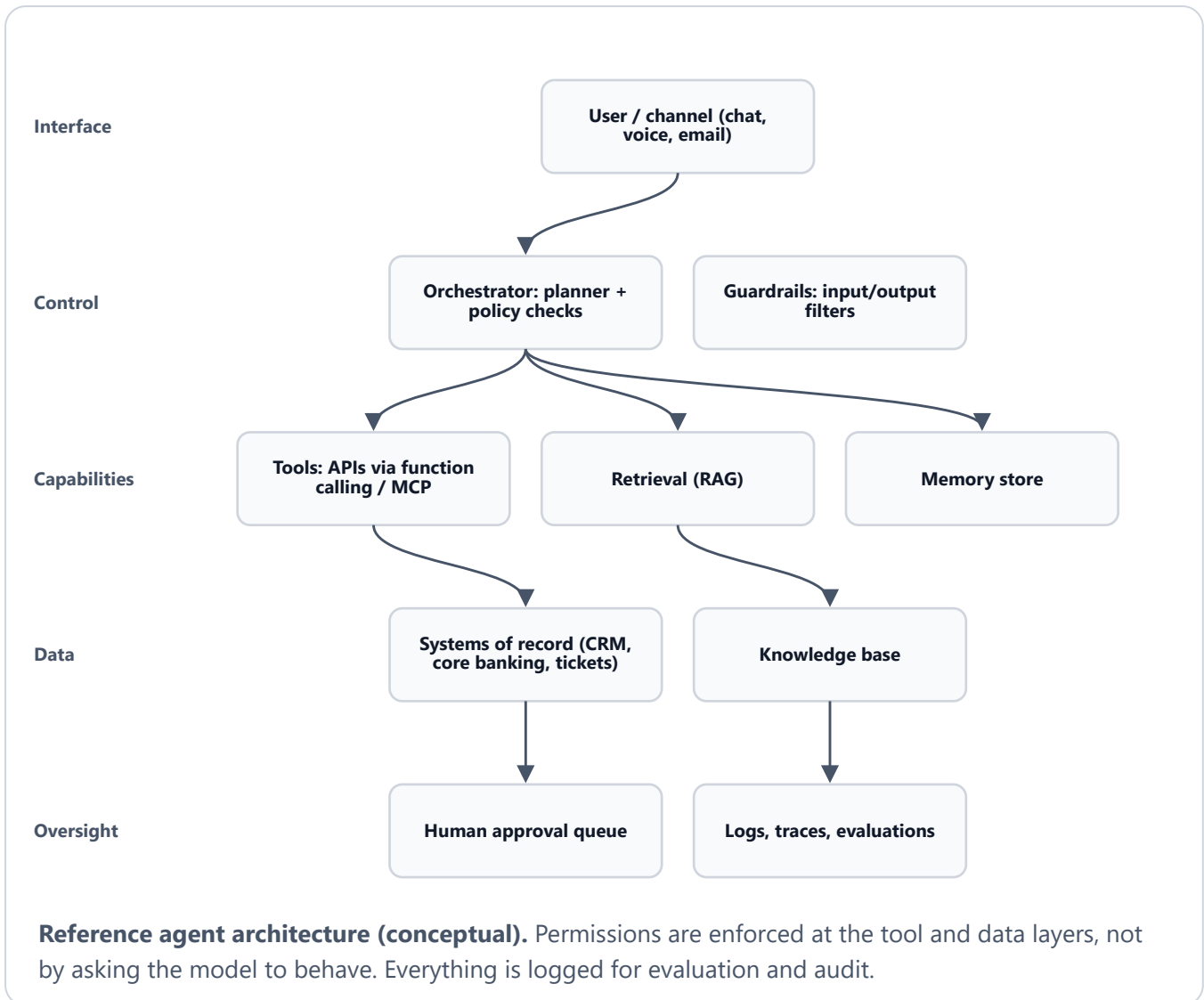
Common composable patterns (from Anthropic's "Building effective agents"): **prompt chaining** (step-by-step pipeline), **routing** (classify then send to a specialised path), **parallelisation** (split or vote), **orchestrator-workers** (a lead model delegates subtasks), **evaluator-optimiser** (one generates, another critiques), and **autonomous agents** (open-ended loops with tools).

12.3 Autonomous versus human-supervised systems



Levels of autonomy (a practical scale). Move up one level only when evaluation evidence, monitoring and reversibility justify it. Money movement, account changes and regulated decisions usually stay at L2–L3.

12.4 Agent architecture



12.5 Planning and reasoning workflows

- **ReAct** (reason + act): the model alternates thinking about the next step and calling a tool, using observations to update its plan.
- **Plan-then-execute**: draft a plan, check it (by rules, another model or a human), then execute steps.
- **Reflection / self-critique**: evaluate intermediate outputs before continuing.
- **Decomposition**: break a large goal into sub-goals with clear completion criteria.

PM implication: plans must be **observable** (logged) and **checkable** — reviewers should be able to see *why* the agent did something.

12.6 Tool calling and function calling

The model is given **tool definitions** (name, description, input schema). When useful, it outputs a structured **call**; the application executes it (with permission checks) and returns the result for the model to use.

```
{
  "name": "get_card_status",
  "description": "Return the status of the authenticated customer's card. Read-only.",
  "input_schema": {
    "type": "object",
    "properties": { "card_last4": { "type": "string", "pattern": "^[0-9]{4}$" } },
    "required": ["card_last4"]
  }
}
```

Design rules: small, single-purpose tools; clear descriptions (the model reads them); **separate read and write tools**; validate inputs server-side; return structured errors; make write tools **idempotent**; require approval for consequential writes.

12.6.1 MCP and APIs

From your course notes (Lecture 46; Companion ch. 15; Picture lesson 'API and MCP')

MCP (Model Context Protocol) is a standard way for an AI application to connect to resources and tools; **an API** is a broader term for software interfaces; an MCP server may expose a tool that itself calls an API. *MCP does not replace every API, does not automatically make a connection safe, does not grant access or guarantee an action succeeds.* Reading a calendar and creating an event are different permissions and may need different approval. Ask who runs the server, what it can access and do, how credentials are protected, whether actions are logged and how access is revoked. **Use least-privilege access, protect tokens as secrets and require approval for external actions.** The Figma demo in class did not establish end-to-end design creation; tool availability and pricing change.

12.7 Memory systems

Memory type	What it holds	Example	Risks
Working / short-term	Current conversation and tool results in the context window	The customer's last three messages	Context overflow; cost
Long-term semantic	Facts and preferences stored externally and retrieved	"Prefers Hindi; has two cards"	Stale or wrong facts; privacy
Episodic	Summaries of past interactions	"Raised a dispute on 3 March, resolved"	Over-reliance on summaries
Procedural	Learned instructions/playbooks	Updated escalation policy	Uncontrolled behaviour drift

Product rules: tell users what is remembered; let them view/delete; never store sensitive data without purpose and consent; expire memories; keep authoritative data in systems of record, not in the agent's memory.

From your course notes (Lecture 49)

Despite the LMS description, the session was mostly a guided workshop, not a detailed treatment of multi-agent memory. This section is independent reference material.

12.8 Multi-agent systems and orchestration

Multi-agent systems split work among specialised agents (e.g., a triage agent, a policy-lookup agent, a drafting agent, a reviewer agent) coordinated by an **orchestrator**. Benefits: specialisation, parallelism, separation of permissions. Costs: more tokens, latency, harder debugging, error propagation between agents.

From your course notes (Lecture 51)

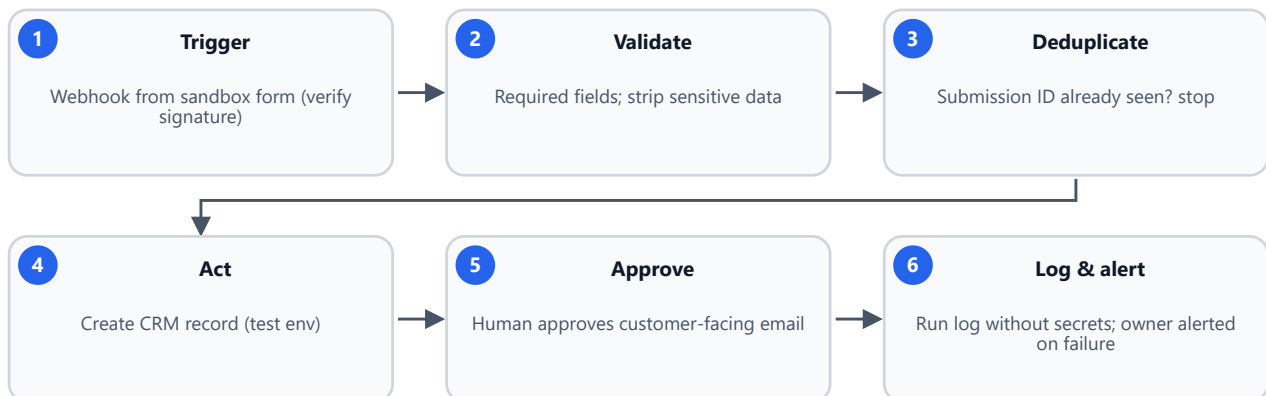
Orchestrate a workflow by breaking an outcome into steps, coordinating tools or specialised agents, passing information between them and handling failures — steps can be sequential or parallel (appointment-booking example). Add safeguards: access permissions, protected credentials, validated intermediate outputs and **human review appropriate to the risk** (a refund example used a human checkpoint before a consequential decision). *MCP and cloud IAM do not automatically secure data or prevent hallucinations.*

12.9 Workflow automation

Tools like **n8n**, Zapier or Make connect triggers and actions — often with AI steps inside a deterministic flow. This is frequently the right answer before a fully autonomous agent.

From your course notes (Companion ch. 25, 27)

In n8n a **trigger** starts a workflow, nodes process data and actions write or notify. Start with a manual trigger and dummy data; inspect each node's output. For a website form: validate fields, remove sensitive fields, **deduplicate using a submission ID**, store in an approved destination, notify a test inbox. Add error handling, retry limits, idempotency, logs without secrets, owner alerts and **an approval step before sending customer messages**. Before automating an existing website, identify the owner and existing integrations, use a sandbox, test missing email, repeat events, timeouts and invalid signatures, and get **explicit permission to activate**. Release checklist: security review; privacy and consent; scoped credentials; duplicate protection; rate limits; monitoring; backup and rollback; cost owner.



A safe automation (form → CRM → notify). Deterministic steps with one AI step (e.g., classify the enquiry) are easier to test and audit than a free-roaming agent.

12.10 RAG-enabled agents

An agent can call retrieval as a tool: decide *whether* to search, *what* to search, and whether results suffice. Benefits: answers grounded in policy; can combine knowledge with account data. Risks: retrieved text may contain **indirect prompt injection**; permissions must filter what is retrievable for this user (Module 10).

12.11 Evaluation and observability for agents

Evaluate at three levels:

1. **Outcome:** did the task get done correctly? (task success rate, correct resolution)
2. **Trajectory:** were the steps sensible? (right tools, right order, no unnecessary or unsafe calls, number of steps)
3. **Component:** retrieval quality, tool-call argument accuracy, refusal/escalation correctness.

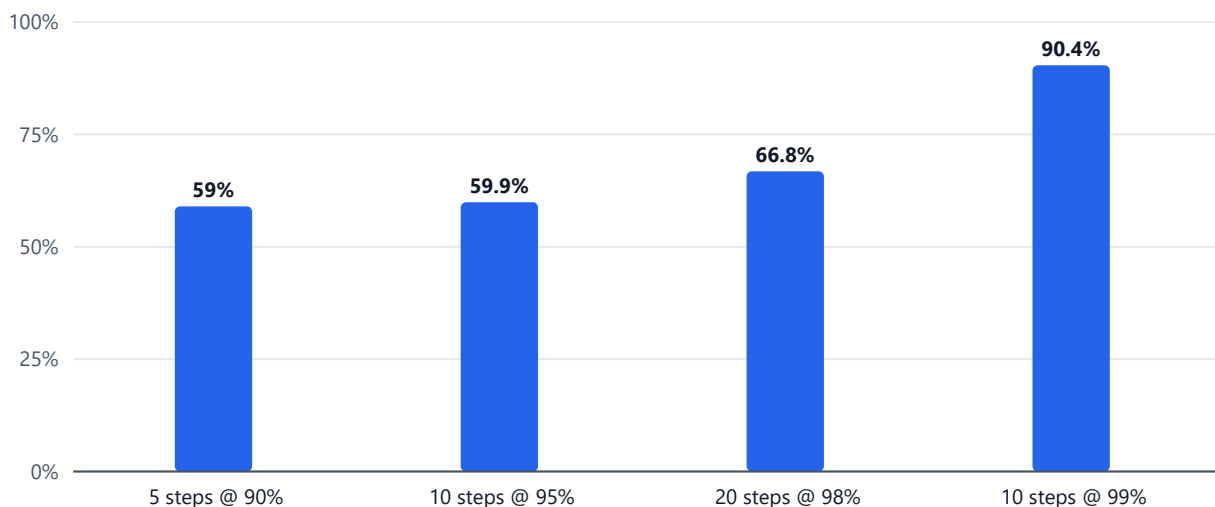
Observability: trace every run (prompts, tool calls, arguments, results, latency, tokens, costs, decisions); dashboards for success, escalation, errors, cost per task; sample reviews; alerting on anomalies (spikes in tool errors or refusals).

12.11.1 Reliability compounds

$P(\text{task succeeds}) \approx p_1 \times p_2 \times \dots \times p_n$ (if each step must succeed independently)

Worked example

10 steps each 95% reliable $\rightarrow 0.95^{10} \approx$ **60%** end-to-end. At 99% per step $\rightarrow 0.99^{10} \approx$ **90%**. 20 steps at 98% $\rightarrow$ **67%**. Fewer steps, more reliable tools, verification checkpoints and recovery paths matter more than a slightly better model.



End-to-end success when every step must succeed (calculated). Computed as per-step reliability raised to the number of steps (assumes independent steps without retries). Retries and verification can raise these figures; extra steps lower them.

12.12 Human approval checkpoints, guardrails and permission boundaries

- **Approval checkpoints:** before money moves, account details change, messages go to customers, or irreversible actions happen. Show the reviewer *what will happen, why, and the evidence*.
- **Guardrails:** input filters (PII, abuse, injection patterns), output filters (policy compliance, PII leakage), schema validation, allow-lists of actions, spending/time/step limits, rate limits, kill switch.
- **Permission boundaries:** least privilege per tool; scoped tokens; per-user authorisation enforced by the backend; separate read and write credentials; sandbox and test environments for development.

12.13 Agent security and prompt injection

Prompt injection is when untrusted text (a customer message, a web page, an email, a document) contains instructions that hijack the agent. **Indirect injection** hides them in content the agent retrieves.

The most dangerous combination — described by security researcher Simon Willison as the “lethal trifecta” — is an agent with **(1) access to private data, (2) exposure to untrusted content, and (3) the ability to communicate externally**. Remove at least one leg for sensitive workflows.

LIMIT CAPABILITY

- Least-privilege tools
- No external send without approval
- Separate agents for untrusted content

ISOLATE UNTRUSTED TEXT

- Delimit and label as data
- Never treat retrieved text as instructions
- Strip hidden/markup instructions

VALIDATE ACTIONS

- Allow-lists and schemas
- Policy checks before execution
- Human approval for consequential actions

DETECT & MONITOR

- Injection classifiers (imperfect)
- Anomaly alerts on tool use
- Red-team regularly

CONTAIN DAMAGE

- Scoped, short-lived tokens
- Idempotent, reversible actions
- Audit logs

TEST

- Adversarial eval cases in every release
- Include indirect injection via documents and emails

Prompt-injection defences (layered). No single defence is sufficient; prompt injection is not fully solved. Design so that a successful injection cannot cause serious harm.

12.14 Latency and operating costs

Worked example (hypothetical prices: \$3/M input, \$15/M output tokens)

An agent resolves a task in 8 model steps; context grows, averaging 4,000 input tokens per step and 300 output tokens. Tokens: 32,000 input, 2,400 output → cost = $32,000 \times 3/1M + 2,400 \times 15/1M = \$0.096 + \$0.036 = \mathbf{\$0.132 \text{ per task}}$ — about **12×** the single-call answer from Module 10 (\$0.0105). Latency: $8 \times (1.5 \text{ s model} + 0.5 \text{ s tool}) \approx \mathbf{16 \text{ s}}$. Users tolerate this for background tasks (with progress updates) but not for chat replies. Levers: fewer steps, smaller models for routing, caching, parallel tool calls, summarising context, and setting a step budget.

12.15 Failure recovery

- **Retries with back-off** for transient errors; **idempotency keys** so retries don't duplicate actions.
- **Checkpoints and state** so a long task resumes rather than restarts.
- **Timeouts and step budgets** to stop runaway loops.
- **Compensating actions / rollback** for multi-step changes (e.g., reverse a hold if a later step fails).
- **Graceful handoff** to a human with full context when the agent is stuck.
- **Post-incident review** feeding new eval cases.

12.16 Agentic product design

Design for **trust and control**:

- **Transparency**: show what the agent is doing ("Checking your last 3 transactions..."), what it used (sources), and what it will do next.
- **Control**: preview and approve; easy cancel; undo where possible.
- **Scoping**: tell users what it can and cannot do.
- **Progressive autonomy**: start with suggestions, earn the right to act.
- **Feedback loops**: thumbs, corrections, and escalation feed evaluation.
- **Accessibility and inclusion**: multiple languages, clear wording, alternative channels.

12.17 AI automation opportunities in finance

Area	Example opportunity	Typical autonomy	Key risk
Customer support	Answer FAQs, card status, statement help	L2–L3	Wrong policy answer; data leakage
Disputes / chargebacks	Gather evidence, draft case	L1–L2	Regulatory deadlines; fairness
KYC / onboarding	Document data extraction, completeness checks	L1–L2	Bias, false rejections
Collections / hardship	Identify hardship signals, offer options	L1–L2	Consumer-protection obligations
Fraud operations	Summarise alerts, propose actions	L1–L2	False positives freezing accounts
Finance back office	Reconciliation, invoice matching (maker–checker)	L2–L3	Payment errors
Internal knowledge	Policy Q&A for agents	L0–L1	Outdated policy

From your course notes (Lecture 55)

A logistics-document OCR prototype proposed extracting invoice, transport and tax fields, matching documents and flagging missing items with **maker–checker review**; a pilot should measure field accuracy, reconciliation errors and ROI. *Stated cost, savings and accuracy claims were projections, not established facts.*

12.18 AI product roadmaps and agentic business models

Roadmap shape: (1) internal copilot for staff (L0–L1) → (2) customer self-service for low-risk intents with RAG (L2) → (3) transactional actions within limits with approvals (L3) → (4) proactive agents (alerts, nudges) once trust and monitoring are proven. Each stage has evaluation gates.

Business models: per-seat (copilots), **usage-based** (per task/token), **outcome-based** (per resolved conversation or per successful action), tiered bundles. Outcome pricing aligns value but requires agreed definitions of “resolved” and robust measurement. Internally, ROI = (cost per human-handled contact – cost per AI-resolved contact) × resolved volume – platform and oversight costs — *including* the cost of errors.

12.19 Case study: AI-powered customer-support assistant for a card issuer (hypothetical)

12.19.1 Business background

A mid-sized (fictional) card issuer serves ~2 million cardholders through app chat, phone and email. Contact volume rises during sale seasons; customers wait long for simple answers (card block, statements, fees), while complex cases (disputes, fraud, hardship) need skilled agents.

12.19.2 Target users

- **Cardholders** seeking fast answers and simple actions.
- **Support agents** handling complex and sensitive cases.
- **Operations/compliance teams** who own policy and audit.

12.19.3 Customer problem

Customers wait for answers to simple questions and must repeat information when transferred. Agents spend time on routine queries and copying data between systems.

12.19.4 Business objective

Reduce cost per contact and wait times **without** increasing complaints, errors or compliance breaches.

12.19.5 Available evidence (hypothetical)

- Contact reasons for the last quarter (ticket tags): card block/unblock 14%, statement and fees 21%, payment status 12%, rewards 8%, disputes 9%, fraud 6%, hardship 3%, other 27%.
- Median chat wait 11 minutes at peak; 34% of chats transferred at least once.
- Policy documents exist but are inconsistent in places.

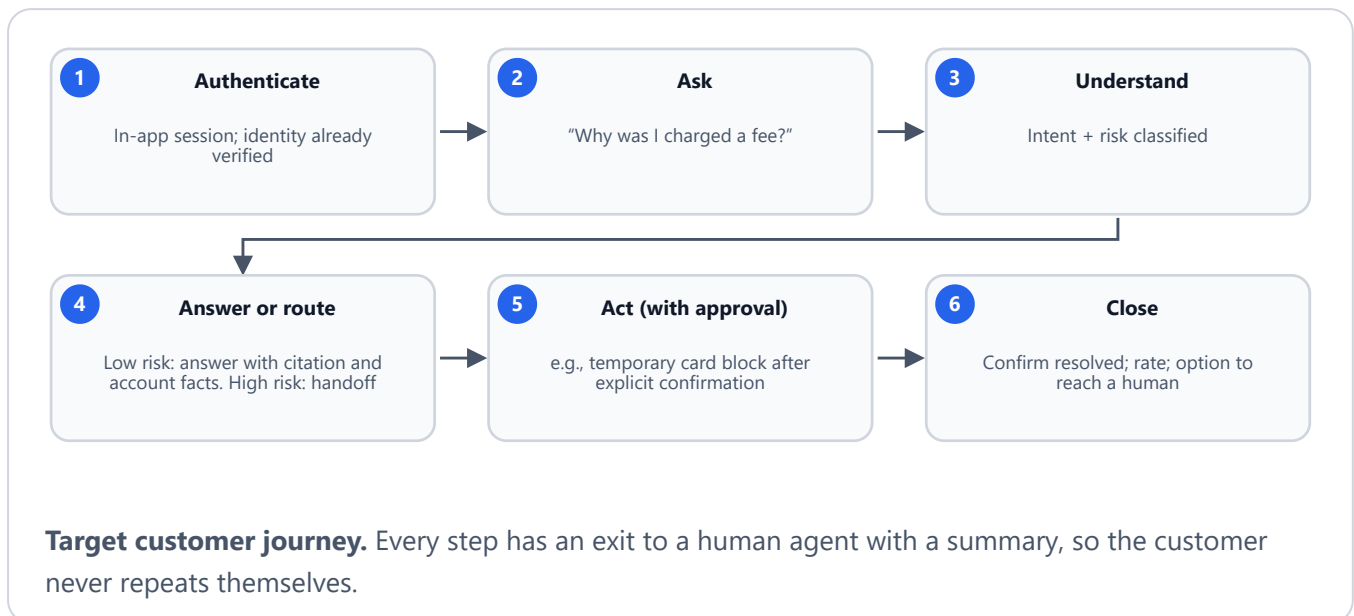
12.19.6 Assumptions to test

1. $\geq 40\%$ of contacts are low-risk informational intents answerable from policy + read-only account data.
2. Customers will accept an AI answer if it is fast, cited and offers a human.
3. The assistant can recognise high-risk intents (disputes, fraud, hardship) reliably enough to route them.

12.19.7 Product hypothesis

We believe an assistant that answers low-risk intents from approved policies and read-only account data, and routes high-risk intents to agents with a summary, will cut median wait time by 50% and resolve 35% of chats without an agent, while keeping CSAT and complaint rates at or better than baseline.

12.19.8 User journey (target state)



12.19.9 Candidate solutions

Option	Description	Pros	Cons
A · Better FAQ search	Improve help-centre search	Cheap, safe	Doesn't use account context; low resolution
B · RAG assistant (read-only)	Answers from policy + read-only account data; routes high risk	Strong value, controllable	Needs eval, permissioning
C · Agent with actions	B + actions (card block, address change, refunds under threshold)	Highest resolution	Higher risk; approvals needed
D · Agent copilot for staff	Drafts replies, summaries for agents	Low risk; fast ROI	Doesn't reduce customer wait directly

12.19.10 Prioritisation

Using a weighted score (value 35%, risk 25% inverse, feasibility 20%, time-to-value 20%), the team picks **D + B** first: the copilot builds evaluation data and agent trust; the read-only assistant addresses wait times. **C** follows with one low-risk action (temporary card block, reversible) behind explicit confirmation.

12.19.11 MVP

- Channels: in-app chat for authenticated customers in one language.
- Intents: card status/block (read-only status), statement location, fee explanations, rewards balance — **answer**; disputes, fraud, hardship, complaints — **route with summary**; off-topic financial advice — **refuse politely**.
- Grounding: approved policy pages via RAG with citations; account facts via read-only tools.
- Controls: permission-aware retrieval; no external sends; PII redaction in logs; human handoff button always visible; kill switch.
- Pilot: 5% of eligible chats for 4 weeks; agents review a daily sample.

12.19.12 KPIs

- **Primary:** containment / self-service resolution rate (resolved without agent, no repeat contact in 7 days).
- **Customer:** median time to answer; CSAT for bot-handled chats; complaint rate.
- **Quality & safety:** policy accuracy on sampled answers; correct routing of high-risk intents (target ≥ 98%); PII-leak incidents (target 0); injection success on red-team set (target 0).

- **Operational:** agent handle time for routed chats (summary benefit); cost per resolved contact.

12.19.13 Risks and constraints

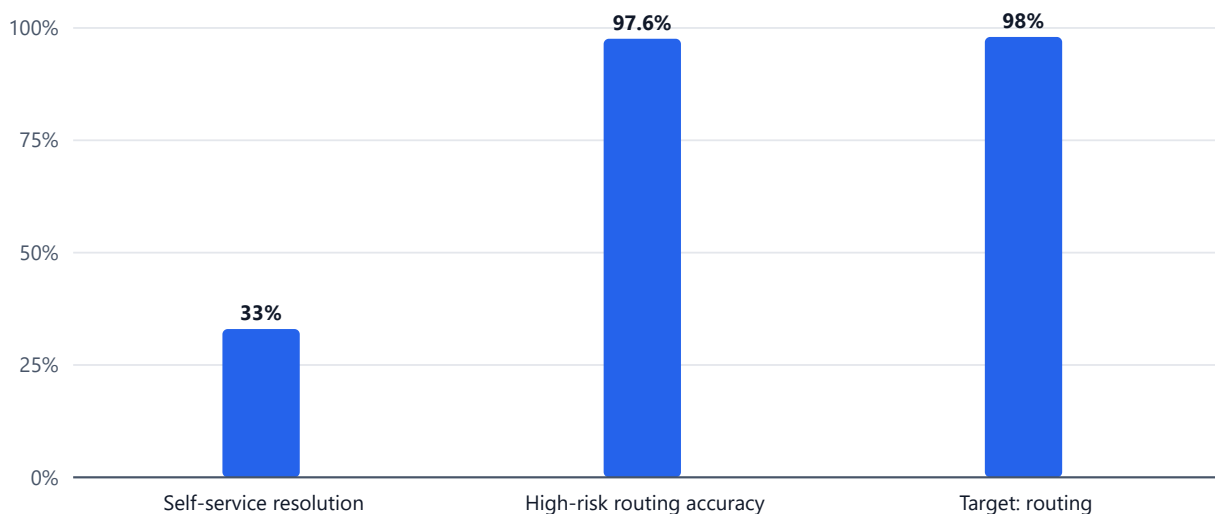
Regulatory obligations (complaints handling, disputes timelines, fair treatment, data protection); hallucinated policy; mis-routing vulnerable customers; prompt injection via customer messages; authentication and authorisation bugs; model/vendor changes; inconsistent policy sources; language and accessibility gaps.

12.19.14 Experiment plan

1. **Offline eval** on 60+ curated cases (see `datasets/support_eval_set.csv`) plus 500 anonymised historical chats labelled by agents; launch gate: routing accuracy $\geq 98\%$ on high-risk intents and zero severity-1 failures.
2. **Agent copilot pilot** (2 weeks) to compare handle time and agent satisfaction.
3. **Customer pilot:** randomised 5% of eligible chats vs control (current flow), 4 weeks; guardrails reviewed daily.
4. **Decision review** with compliance and support leadership.

12.19.15 Example metrics (hypothetical pilot readout)

Metric	Control	Assistant	Note
Median time to first answer	11 min	0.3 min	Low-risk intents
Self-service resolution	—	33%	Of all assistant chats
CSAT (1–5)	3.9	4.0	Difference not statistically tested in this readout
High-risk intents routed correctly	—	97.6%	Below 98% target → fix before scale
Complaints per 1,000 chats	6.1	6.4	Watch; investigate tags
PII incidents	0	0	—



Pilot readout — selected metrics vs targets (hypothetical). Hypothetical. Resolution is near the 35% hypothesis; routing accuracy misses its launch gate, so scaling waits for fixes.

12.19.16 Product recommendation

Do **not** scale yet. Fix routing misses (most involved hardship language such as “I can't pay”), add those phrasings to the eval set, investigate the complaint increase, then extend to 20% of traffic. Begin design of the first action (temporary card block) with explicit confirmation and a reversal path.

12.19.17 Potential outcomes

- **Best case:** 35–40% resolution, wait times collapse for simple queries, agents focus on complex cases.
- **Base case:** 25–30% resolution with strong agent-copilot gains.
- **Worst case:** policy errors or mis-routing cause complaints/regulatory exposure → kill switch, revert to copilot-only.

12.19.18 Key PM lessons

1. Start with **assistive, read-only** capabilities and earn autonomy.
2. **Routing high-risk intents** is a launch-blocking capability, not a nice-to-have.
3. **Evaluation sets and severity weighting** drive decisions — not demos.
4. **Permissions live in the backend**, never in the prompt.
5. Measure **customer outcomes and safety**, not just containment.

12.20 Exercises with model answers

Exercise 1 · Agent or workflow?

"When a customer uploads an invoice, extract fields, check against the PO, and post to the ERP if matched; otherwise send to finance." Agent or workflow? Why?

Model answer

A **workflow** with one AI step (field extraction): the path is known and auditable; matching and posting are deterministic rules. Add maker–checker review for exceptions and an approval for postings above a threshold. An agent adds unpredictability without benefit here.

Exercise 2 · Design approval checkpoints

An agent can issue refunds. Define thresholds and approvals.

Model answer (example)

Auto-approve refunds $\leq ₹500$ when policy criteria are met and the customer has no recent refunds; require agent approval for ₹500–₹5,000 or any repeat refund in 30 days; require supervisor approval above ₹5,000. All refunds idempotent (refund ID), logged with evidence, reversible within policy, and sampled weekly for audit. Kill switch if refund rate exceeds a daily threshold.

Exercise 3 · Lethal trifecta

Your email-triage agent reads external emails, can search internal customer records, and can send emails. What's the risk and how do you redesign it?

Model answer

It has all three legs: untrusted content, private data and external communication — an injected email could make it exfiltrate customer data. Redesign: remove autonomous sending (drafts only, human sends), or split into a reader agent with no data access and a separate drafting step with read access but no sending; restrict searchable fields; add injection tests.

12.21 Interview perspective

“Design an AI agent for customer support at a bank.” — structure

Clarify goals and constraints (regulation, channels, languages) → segment intents by risk → choose autonomy per intent (answer, route, act with approval) → architecture (RAG + read-only tools + approvals, permission-aware) → evaluation plan (offline set, severity, red-team, pilot A/B) → metrics (resolution, CSAT, routing accuracy, safety incidents, cost) → rollout and governance (kill switch, audits). *Mistakes*: full autonomy on day one; ignoring vulnerable-customer routing; trusting the model with access control.

“How do you evaluate an agent?” — 60-second answer

“At three levels. Outcome — did it complete the task correctly, measured on a curated test set and in a controlled pilot. Trajectory — did it use the right tools in a sensible order without unsafe or wasted steps, which I check from traces. Components — retrieval quality, tool-argument accuracy, routing and refusals. I weight failures by severity, red-team for prompt injection, and re-run the suite on every model, prompt or tool change. In production I trace every run and sample reviews weekly.”

12.22 Key takeaways

- Prefer **workflows** to agents unless the path genuinely varies; add autonomy **gradually** with evidence.
- Agents = model + tools + memory + control loop + **guardrails**; enforce permissions in the backend.
- **Reliability compounds** across steps; reduce steps, add verification and recovery.
- Defend against **prompt injection** in layers; avoid the private-data + untrusted-content + external-action combination.
- Evaluate **outcomes, trajectories and components**; trace everything.
- Agent costs and latency are multiples of single calls — **budget steps**.

12.23 Check your understanding

CHECK YOUR UNDERSTANDING

What most distinguishes an agent from a fixed workflow?

- A. It uses an LLM
- B. It chooses its next steps at run time based on results
- C. It has a chat interface
- D. It never needs humans

Why: Workflows follow predefined paths; agents decide steps dynamically within limits.

10 steps at 95% reliability each give end-to-end success of about...

- A. 95%
- B. 90%
- C. 60%
- D. 50%

Why: $0.95^{10} \approx 0.60$ when each step must succeed independently.

Which combination creates the highest prompt-injection risk?

- A. Private data + untrusted content + ability to communicate externally
- B. Read-only FAQ + no tools
- C. Offline batch job
- D. Human-only workflow

Why: Together they allow injected instructions to exfiltrate data.

Where should a customer's permission to see an account be enforced?

- A. In the system prompt
- B. In the backend tool/API authorisation
- C. By the model's judgement
- D. In the UI only

Why: Authorisation must be enforced by systems, not by model instructions.

MCP is best described as...

- A. A security product that makes integrations safe
- B. A standard protocol for connecting AI apps to tools and resources
- C. A replacement for all APIs
- D. A model

Why: MCP standardises connections; permissions and security still have to be designed.

12.24 Recommended resources

- [Anthropic: Building effective agents](#) (from your course references).
- [Model Context Protocol: Introduction](#) (course reference).
- [OWASP Top 10 for LLM Applications](#).
- [Simon Willison: The lethal trifecta for AI agents](#).
- [n8n: Build your first workflow](#) (course reference).
- [NIST AI Risk Management Framework](#).