

Agentic AI Project Planning

Worksheet

**A practical planning guide for defining, scoping, and preparing an
agentic AI use case before development begins.**

1. Project Overview

The project overview captures the basic identity and business purpose of the agentic AI initiative. It helps stakeholders quickly understand what the project is, why it matters, who owns it, and how success connects to a business outcome. This section should be completed before technical design begins so the team does not build an AI agent around an unclear or low-value problem.

1.1 Project Name

The project name should be short, memorable, and specific enough to distinguish the initiative from other automation or analytics work. A good name usually includes the process, department, or outcome being improved.

- **Worksheet prompt:** What should this agentic AI project be called?
- **Good example:** Finance Invoice Exception Resolution Agent.
- **Weak example:** AI Project 1, because it does not show the business process or intended outcome.

1.2 Problem or Opportunity

Describe the business pain point or opportunity that justifies the project. Agentic AI is most useful when a process requires reasoning, multi-step coordination, tool use, exception handling, or decisions that cannot be fully captured through simple rules. The

problem statement should explain what is happening today, who is affected, and why the current approach is insufficient.

- **Worksheet prompt:** What problem, bottleneck, missed opportunity, or customer pain should the agent address?
- **Example:** Support teams spend several hours each day reading incident tickets, checking knowledge-base articles, identifying likely causes, and drafting first responses. This creates delays during high-volume periods.
- **Tip:** Avoid framing the problem as “we need AI.” Instead, describe the operational issue that AI may help solve.

1.3 Business Goal

The business goal defines the measurable value the project should deliver. It should connect the AI agent to business outcomes such as faster cycle time, lower manual effort, better service quality, improved compliance, reduced rework, or increased revenue. Strong goals are specific enough to guide prioritization and later measurement.

- **Worksheet prompt:** What business result should improve if the agent succeeds?
- **Example goal:** Reduce average incident triage time by 30% within three months of pilot launch.

- **Possible measures:** cycle time, cost per transaction, service-level compliance, manual handoffs, rework rate, customer satisfaction, or employee productivity.

1.4 Project Owner

The project owner is the accountable person or function responsible for defining scope, securing stakeholder alignment, making trade-off decisions, and confirming whether the agent delivers acceptable business value. The owner does not need to be a technical expert, but they should understand the process and have authority to remove business blockers.

- **Worksheet prompt:** Who owns the business outcome and final approval for this project?
- **Example:** Head of IT Service Management, Finance Operations Manager, Procurement Process Lead, or HR Shared Services Director.
- **Include:** owner name, department, decision authority, escalation contact, and expected time commitment.

2. Use Case Definition

The use case definition translates the broad project idea into a clear operational scenario. It identifies the process to improve, the users involved, the expected outcome, and the challenges that must be solved. This section is important because agentic AI projects can become too broad if the team does not define the workflow boundaries early.

2.1 Process to Improve

Identify the workflow, task sequence, or decision process where the agent will operate. Capture the current steps at a practical level, including inputs, decisions, tools, handoffs, approvals, and outputs. A strong process definition helps the team decide whether the agent should assist, recommend, execute, monitor, or escalate.

- **Worksheet prompt:** Which process should the agent improve?
- **Example:** In incident management, the process may include ticket intake, impact classification, knowledge search, root-cause hypothesis, responder assignment, customer update drafting, and escalation.
- **Useful detail:** Note where humans spend the most time, where errors occur, and where information is scattered across systems.

2.2 Target Users

Target users are the people who will interact with, be supported by, or be affected by the AI agent. Defining users clearly helps shape the interface, level of automation, permissions, training needs, and human-in-the-loop controls. Different user groups may need different outputs from the same agent.

- **Worksheet prompt:** Who will use or rely on this agent?
- **Examples:** service-desk analysts, project managers, procurement specialists, finance operations teams, HR case managers, compliance reviewers, or customer-support agents.
- **Consider:** user skill level, approval rights, system access, risk tolerance, and whether the user needs recommendations, drafts, decisions, or completed actions.

2.3 Expected Outcome

The expected outcome describes what should be different after the agent is introduced. It should include both the immediate operational output and the broader business effect. For agentic AI, outcomes often include completed tasks, recommended next actions, generated summaries, prioritized cases, automated follow-ups, or coordinated actions across systems.

- **Worksheet prompt:** What should the agent produce, trigger, or improve?

- **Example:** For a project management agent, the outcome may be a weekly risk summary, updated task dependencies, flagged schedule conflicts, and recommended mitigation actions.
- **Success statement:** Users can complete the target workflow faster, with fewer manual checks, while maintaining appropriate human review for high-risk decisions.

2.4 Current Challenges

Current challenges explain why the existing process is difficult, slow, expensive, risky, inconsistent, or hard to scale. These challenges help determine whether the project requires an agentic AI approach or whether a simpler automation, dashboard, workflow rule, or knowledge-base improvement would be enough.

- **Worksheet prompt:** What makes the current process hard to execute well?
- **Common challenges:** fragmented data, manual copy-paste work, unclear ownership, inconsistent decision criteria, delayed approvals, lack of audit trail, high exception volume, or dependency on expert judgment.
- **Example:** A procurement team may struggle because vendor risk data sits in separate systems, policy checks are manual, and buyers are unsure when to escalate exceptions to legal or compliance.

3. Workflow Mapping

Workflow mapping converts the use case into a clear sequence of events, decisions, inputs, and outputs. For agentic AI projects, this step is especially important because the agent may not follow a single fixed path every time. It may interpret the goal, call tools, review intermediate results, branch to a different step, or pause for human approval when risk is high. A strong workflow map helps the team define where autonomy is useful and where control, monitoring, or escalation is required.

3.1 Workflow Trigger

The workflow trigger is the event, request, schedule, or condition that starts the agent's work. Triggers should be clearly defined so the agent does not act too early, too late, or in the wrong context. Common triggers include a new ticket being created, an email arriving, a file being uploaded, a scheduled reporting cycle, a threshold being breached, or a user asking the agent to begin a task.

- **Worksheet prompt:** What exact event should start the workflow?
- **Example:** A new high-priority incident ticket is created in the ITSM system and assigned to the service desk queue.
- **Trigger types:** user-initiated request, system event, scheduled run, data threshold, document upload, approval request, or exception alert.

- **Design tip:** Define trigger conditions tightly. If the trigger is too broad, the agent may waste effort on low-value tasks or create unnecessary alerts.

3.2 Key Workflow Steps

Key workflow steps describe what happens after the trigger. In an agentic workflow, steps may include goal interpretation, information gathering, reasoning, tool use, validation, human review, execution, and follow-up. Mapping these steps helps stakeholders understand what the agent is allowed to do independently and where a person must review or approve the result.

- **Worksheet prompt:** What are the main steps from trigger to completion?
- **Example workflow:** receive ticket, classify impact, retrieve relevant knowledge articles, identify likely cause, recommend assignment group, draft user update, request approval if customer-facing, and log final action.
- **Include:** decision points, handoffs, approval gates, exception paths, rework loops, and stopping conditions.
- **Design tip:** Separate deterministic steps from reasoning-based steps. Simple routing or data movement may not need AI, while ambiguous classification or recommendation may benefit from an agent.

3.3 Inputs Required

Inputs are the data, documents, context, instructions, and system records the agent needs to complete the workflow. Poor input definition is one of the most common reasons AI projects fail, because the agent cannot produce reliable outputs if the source information is incomplete, outdated, inaccessible, or ambiguous. Inputs should be mapped by source, format, owner, frequency, sensitivity, and quality level.

- **Worksheet prompt:** What information must the agent read, receive, retrieve, or ask for?
- **Examples:** ticket description, customer profile, system logs, policy documents, knowledge articles, previous cases, project plan, risk register, vendor master data, or approval history.
- **Input checklist:** source system, owner, access permission, refresh frequency, data format, sensitivity level, and known quality issues.
- **Design tip:** Mark inputs that require secure handling, audit logging, or human confirmation before use.

3.4 Expected Outputs

Expected outputs define what the agent should produce at the end of the workflow or at important intermediate points. Outputs may be recommendations, summaries, classifications, completed tasks, drafted communications, updated records, escalations,

or alerts. They should be clear enough that users can judge whether the agent's work is useful, accurate, and ready for action.

- **Worksheet prompt:** What should the workflow produce when it works correctly?
- **Examples:** triage summary, recommended next action, risk rating, draft email, updated workflow status, approved transaction, exception report, or escalation note.
- **Output quality criteria:** accurate, explainable, complete, timely, formatted for the target user, and traceable to source inputs.
- **Design tip:** Define which outputs are advisory only and which outputs can trigger system actions automatically.

4. Agent Roles & Responsibilities

This section defines what the agent is, what it is responsible for, and which tools it needs to perform its work. Clearly assigning agent roles prevents scope creep and reduces the risk of giving one agent too many responsibilities. In larger solutions, multiple agents may work together, each with a focused role such as researcher, planner, reviewer, executor, or escalation coordinator.

4.1 Agent Name

The agent name should make the agent's purpose easy to understand. A practical name usually reflects the business process, task, or function the agent supports. Avoid vague names that sound impressive but do not explain what the agent actually does.

- **Worksheet prompt:** What is the working name of the agent?
- **Examples:** Incident Triage Agent, Procurement Policy Check Agent, Project Risk Review Agent, Invoice Exception Agent, or HR Case Summary Agent.
- **Design tip:** Use the agent name consistently in requirements, workflow diagrams, test cases, and user training materials.

4.2 Agent Role

The agent role explains the primary function the agent performs in the workflow. The role should be narrow enough to test and govern, but broad enough to deliver

meaningful value. A well-defined role also clarifies whether the agent is assisting a human, recommending actions, coordinating tasks, executing approved actions, or monitoring for issues.

- **Worksheet prompt:** What role does this agent play in the workflow?
- **Role examples:** researcher, classifier, planner, reviewer, summarizer, coordinator, executor, monitor, or escalation assistant.
- **Example:** A Project Risk Review Agent reviews project updates, identifies emerging risks, compares them with the risk register, and recommends mitigation actions for the project manager.
- **Design tip:** Avoid assigning the same agent conflicting roles, such as both approving a transaction and auditing that same approval.

4.3 Responsibilities

Responsibilities define the specific tasks the agent is expected to perform and the boundaries it must respect. This section should separate what the agent can do independently, what it can recommend, what requires approval, and what must remain with a human. Clear responsibility boundaries are essential for accountability, auditability, and risk management.

- **Worksheet prompt:** What tasks is the agent responsible for, and what is outside its scope?

- **Example responsibilities:** gather context, summarize records, classify requests, recommend next steps, draft responses, update status fields, create follow-up tasks, detect exceptions, and escalate high-risk cases.
- **Boundary examples:** may draft a customer update but cannot send it without approval; may recommend a supplier risk rating but cannot override compliance policy; may update a project issue log but cannot close a critical issue without owner confirmation.
- **Design tip:** Document responsibility levels as “inform,” “recommend,” “draft,” “execute with approval,” or “execute automatically.”

4.4 Tools or APIs Required

Tools and APIs are the systems the agent must access to retrieve information, perform actions, or update records. Tool access should be designed carefully because it determines what the agent can actually do. Each tool should have a clear purpose, permission level, authentication method, logging requirement, and fallback process if the tool is unavailable.

- **Worksheet prompt:** Which systems, databases, applications, or APIs must the agent connect to?
- **Examples:** ServiceNow, Jira, Microsoft Teams, Outlook, SharePoint, ERP, CRM, knowledge base, document repository, logging tools, monitoring systems, approval workflows, or vendor databases.

- **Tool checklist:** tool name, business purpose, data accessed, action allowed, permission level, API owner, security requirement, usage limit, and audit trail.
- **Design tip:** Give the agent the minimum access needed for the workflow. Start with read-only access where possible, then expand permissions only after testing and approval.

5. AI vs. Automation

This section helps the project team decide which parts of the workflow genuinely require AI and which parts can be handled by regular automation, workflow rules, scripts, or integrations. Agentic AI should be used where reasoning, interpretation, uncertainty, language understanding, prioritization, or multi-step planning is needed. Routine, predictable, and repeatable actions should usually remain with traditional automation because they are easier to test, govern, and maintain.

5.1 Steps That Need AI

Steps that need AI usually involve judgment, ambiguity, natural language, incomplete information, or contextual reasoning. These steps are difficult to automate with fixed rules because the correct action depends on meaning, intent, exceptions, or changing business context. Identifying these steps helps the team focus AI investment where it provides unique value rather than applying it everywhere.

- **Worksheet prompt:** Which workflow steps require interpretation, reasoning, summarization, prioritization, or recommendation?
- **Examples:** understanding a poorly written incident description, summarizing a long customer email thread, identifying hidden project risks, recommending an escalation path, or interpreting whether a policy exception applies.

- **Good fit for AI:** classification with nuance, natural-language understanding, root-cause hypothesis generation, knowledge retrieval, decision support, summarization, and exception analysis.
- **Design tip:** Use AI where rules are too rigid, but require evidence and explanation for any recommendation that affects cost, compliance, customer commitments, or operational risk.

5.2 Steps That Can Use Regular Code

Steps that can use regular code are predictable, structured, and rule-based. These steps do not require an AI agent to reason about the situation; they simply need a reliable system action. Separating regular automation from AI keeps the solution simpler, cheaper, faster, and easier to audit.

- **Worksheet prompt:** Which steps can be completed using rules, scripts, workflow automation, or API calls?
- **Examples:** copying data from one system to another, sending a notification after approval, updating a status field, generating a scheduled report, checking whether a required field is blank, or routing a ticket based on priority code.
- **Good fit for regular code:** validation checks, calculations, structured data transfer, scheduled reminders, standard approvals, deterministic routing, and audit-log creation.

- **Design tip:** Prefer regular code when the logic can be written as “if this condition is true, then take this action.”

5.3 Human Approval Points

Human approval points define where a person must review, approve, correct, or reject the agent’s proposed action before it takes effect. These checkpoints are especially important when the agent’s output could affect customers, employees, financial records, legal obligations, compliance status, security posture, or operational continuity.

Approval points also create a feedback loop that helps improve the agent over time.

- **Worksheet prompt:** Where must a human approve the agent’s action before completion?
- **Examples:** sending an external customer message, closing a high-severity incident, approving a supplier exception, changing payroll data, escalating a legal issue, or publishing a project status report to leadership.
- **Approval criteria:** high financial value, regulatory impact, sensitive personal data, reputational risk, low model confidence, unclear evidence, or exception outside normal policy.
- **Design tip:** Define whether the human is “in the loop” before execution, “on the loop” after execution for review, or only alerted for exceptions.

5.4 Decision Rules

Decision rules define how the workflow should behave under specific conditions. They may be used by regular automation, the AI agent, or the human reviewer. Clear decision rules reduce ambiguity, improve consistency, and make it easier to test whether the agent follows business policy. They also help determine when to escalate, stop, request more information, or proceed automatically.

- **Worksheet prompt:** What rules, thresholds, confidence levels, or policies should guide the workflow?
- **Examples:** if confidence is below 80%, route to a human reviewer; if invoice variance is above 5%, create an exception case; if customer impact is high, escalate to the incident manager; if required evidence is missing, pause the workflow and request clarification.
- **Include:** business rules, risk thresholds, compliance requirements, exception logic, escalation paths, and fallback actions.
- **Design tip:** Write decision rules in plain language first, then convert them into technical logic during solution design.

6. Success Metrics

Success metrics define how the team will know whether the agentic AI project is delivering value. Metrics should cover business impact, operational performance, user adoption, quality, risk, and financial return. For agentic AI, it is not enough to track whether the system is active; the team should measure whether the agent completes useful work accurately, reduces human effort, improves speed, and produces outcomes that stakeholders trust.

6.1 Primary KPI

The primary KPI is the most important measure of project success. It should directly connect to the business goal defined earlier, not just to technical activity. A strong primary KPI tells leadership whether the agent improved the target workflow in a meaningful way.

- **Worksheet prompt:** What is the single most important success measure for this project?
- **Examples:** reduction in average triage time, percentage of cases resolved without escalation, improvement in SLA compliance, reduction in invoice exception backlog, or increase in project-risk detection rate.
- **KPI formula example:** Average triage time before AI minus average triage time after AI, divided by average triage time before AI.

- **Design tip:** Choose one primary KPI and a small number of supporting metrics.

Too many metrics can dilute accountability.

6.2 Time Saved

Time saved measures how much manual effort the agent removes or reduces. This may include time saved per transaction, per case, per report, per ticket, or per user each week. Time savings should be estimated before launch and measured again after deployment using real workflow data.

- **Worksheet prompt:** How much time should this agent save, and for whom?
- **Examples:** reduce ticket review time from 12 minutes to 6 minutes, reduce weekly project reporting effort from 4 hours to 1 hour, or reduce invoice exception research time by 40%.
- **Calculation:** time saved per item multiplied by number of items processed during the measurement period.
- **Design tip:** Separate time saved from time shifted. If the agent reduces analyst work but increases reviewer workload, measure both.

6.3 Cost Impact

Cost impact translates operational improvements into financial value. It may include labor savings, avoided rework, reduced outsourcing, fewer penalties, lower support costs, improved utilization, or faster revenue-related processes. Cost impact should

include both benefits and new costs such as licensing, development, integration, monitoring, governance, and support.

- **Worksheet prompt:** What financial benefit or cost reduction should the project create?
- **Examples:** lower average handling cost per case, reduce overtime during peak periods, avoid SLA penalties, reduce manual reconciliation effort, or decrease rework caused by incorrect routing.
- **Cost formula example:** total labor hours saved multiplied by loaded hourly cost, minus implementation and operating costs.
- **Design tip:** Be conservative in early estimates. Use pilot data to refine the business case before scaling.

6.4 Quality or Accuracy Target

Quality or accuracy targets define the minimum acceptable level of correctness, completeness, consistency, and trustworthiness for the agent's outputs. For agentic AI, quality should be measured not only by the final answer, but also by whether the agent used the right sources, followed the right steps, made valid recommendations, and escalated when appropriate.

- **Worksheet prompt:** What accuracy, quality, or reliability level must the agent achieve before launch or scale-up?

- **Examples:** 95% correct classification for low-risk tickets, less than 3% critical output error rate, 90% of summaries rated useful by reviewers, or 100% escalation for high-risk compliance exceptions.
- **Quality dimensions:** factual accuracy, completeness, source traceability, policy adherence, output clarity, decision consistency, and escalation reliability.
- **Design tip:** Define different quality targets for low-risk and high-risk tasks. High-risk decisions should require stricter validation and human oversight.

7. Risks & Guardrails

Risks and guardrails define how the project will prevent, detect, and respond to unsafe, inaccurate, unauthorized, or unintended agent behavior. Because agentic AI systems may use tools, access data, call APIs, and take multi-step actions, risk controls should be built into the workflow from the start rather than added after deployment.

7.1 Potential Risks

Potential risks are the ways the agent could fail, behave unexpectedly, or create harm if it is not properly designed and governed. These risks may be operational, financial, legal, reputational, security-related, or user-experience related.

- **Worksheet prompt:** What could go wrong if the agent misunderstands the task, uses the wrong data, or takes the wrong action?
- **Examples:** incorrect recommendations, hallucinated explanations, wrong routing, unauthorized system updates, missed escalations, poor customer communication, over-automation, or lack of accountability.
- **Risk categories:** data risk, model risk, process risk, compliance risk, security risk, user adoption risk, and operational continuity risk.
- **Design tip:** Prioritize risks based on likelihood, impact, detectability, and whether the agent can cause real-world consequences.

7.2 Security or Data Concerns

Security and data concerns focus on what information the agent can access, how that information is protected, and whether the agent might expose sensitive content through output, logs, memory, integrations, or downstream actions. Strong identity, permission, data-handling, and audit controls are essential for safe enterprise use.

- **Worksheet prompt:** What sensitive data, systems, permissions, or integrations are involved?
- **Examples:** customer data, employee information, financial records, contracts, incident logs, vendor risk data, payroll data, intellectual property, or regulated information.
- **Controls to consider:** role-based access, least-privilege permissions, encryption, secure API authentication, data masking, logging, retention rules, and approval gates for sensitive actions.
- **Design tip:** Start with read-only access wherever possible and expand privileges only after testing, approval, and monitoring are in place.

7.3 Guardrails Required

Guardrails are the controls that keep the agent within approved boundaries. They may operate before the agent runs, while it plans actions, before it calls tools, before it produces output, or after it completes a task. Effective guardrails combine technical controls with human oversight, policy rules, testing, and monitoring.

- **Worksheet prompt:** What controls are required to prevent unsafe, incorrect, or unauthorized actions?
- **Examples:** input validation, prompt-injection checks, source verification, confidence thresholds, restricted tool use, approval workflows, output review, audit logs, and escalation triggers.
- **Guardrail layers:** input controls, planning controls, tool-use controls, output controls, access controls, and monitoring controls.
- **Design tip:** Match guardrails to risk level. Low-risk drafts may only need review, while high-risk financial, legal, security, or customer-facing actions need stricter controls.

7.4 Fallback Actions

Fallback actions define what happens when the agent cannot complete a task safely or confidently. A well-designed fallback plan prevents the workflow from silently failing, producing low-quality output, or taking risky action when information is missing or uncertain.

- **Worksheet prompt:** What should happen when the agent is uncertain, blocked, or detects a high-risk condition?
- **Examples:** route to a human reviewer, pause the workflow, request missing information, create an exception case, use a safe default, notify the process owner, or revert to the previous manual process.

- **Fallback triggers:** low confidence, unclear instructions, missing data, failed API call, policy conflict, suspected prompt injection, unavailable system, or sensitive data risk.
- **Design tip:** Make fallback actions visible to users so they understand why the agent stopped or escalated.

8. Testing & Monitoring

Testing and monitoring ensure that the agent performs reliably before and after launch.

Testing checks whether the agent behaves correctly in known scenarios, while monitoring detects issues in live use. Together, they provide evidence that the agent is accurate, safe, useful, and operating within approved limits.

8.1 Test Scenarios

Test scenarios are realistic examples used to check how the agent performs across normal, edge, and high-risk situations. They should reflect the actual workflow, user language, data quality, system constraints, and exception cases the agent will face in production.

- **Worksheet prompt:** Which scenarios must the agent pass before it is released?
- **Examples:** standard request, incomplete request, conflicting information, high-priority incident, sensitive-data case, failed API call, policy exception, duplicate record, and user correction.
- **Include:** expected input, expected behavior, expected output, approval requirement, pass or fail criteria, and reviewer notes.
- **Design tip:** Test both successful paths and failure paths. An agent is not production-ready until it handles uncertainty and errors safely.

8.2 Quality Checks

Quality checks verify whether the agent's outputs are correct, complete, useful, policy-aligned, and traceable to reliable sources. These checks should be performed during development, user acceptance testing, pilot operation, and periodic review after launch.

- **Worksheet prompt:** How will reviewers decide whether the agent's output is acceptable?
- **Examples:** source accuracy, no unsupported claims, correct classification, clear explanation, appropriate tone, policy compliance, correct tool use, and proper escalation.
- **Review method:** sample-based review, side-by-side comparison with human output, checklist scoring, SME review, user feedback, and regression testing.
- **Design tip:** Create a quality checklist before testing begins so reviewers use consistent standards.

8.3 Monitoring Metrics

Monitoring metrics track how the agent behaves in live operation. These metrics should show whether the agent is useful, reliable, safe, cost-effective, and aligned with the expected business outcome. Monitoring also helps detect drift, misuse, system failures, rising costs, or declining output quality.

- **Worksheet prompt:** What live metrics should be tracked after launch?

- **Examples:** task completion rate, approval rate, escalation rate, error rate, override rate, average handling time, user satisfaction, model confidence distribution, tool failure rate, and cost per transaction.
- **Risk metrics:** sensitive-data incidents, policy violations, unauthorized tool attempts, hallucination reports, unresolved exceptions, and repeated fallback triggers.
- **Design tip:** Review monitoring data regularly and define thresholds that trigger investigation or rollback.

8.4 Error-Handling Plan

The error-handling plan explains how the agent and support team should respond when something goes wrong. It should cover technical errors, wrong outputs, failed integrations, unavailable systems, user complaints, policy breaches, and repeated low-confidence responses.

- **Worksheet prompt:** What should happen when an error occurs, and who is responsible for fixing it?
- **Examples:** show a clear error message, stop tool execution, log the issue, notify the owner, open a support ticket, retry after a delay, switch to manual processing, or roll back the last action.
- **Include:** severity levels, response time, owner, escalation path, communication template, rollback process, and post-incident review.

- **Design tip:** Treat serious agent failures like operational incidents. Capture root cause, corrective action, and prevention steps.

9. Implementation Roadmap

The implementation roadmap organizes the project into manageable phases from setup through post-launch review. It helps the team move from planning to delivery while keeping scope, risk, governance, testing, and business value visible. The roadmap should be realistic and should include both technical work and business-readiness activities.

9.1 Initial Setup

Initial setup confirms the business case, scope, ownership, data availability, security requirements, and delivery approach. This phase should establish the foundation before development begins so the team does not build against unclear requirements or unavailable systems.

- **Worksheet prompt:** What must be ready before development starts?
- **Activities:** confirm use case, define scope, identify users, assign project owner, map systems, document data sources, review security needs, and agree on success metrics.
- **Deliverables:** approved project brief, workflow map, risk assessment, initial backlog, access plan, and test strategy.
- **Design tip:** Do not begin development until the project owner confirms the decision rights and approval process.

9.2 Development

Development builds the agent workflow, integrations, prompts, tool connections, rules, guardrails, and user experience. The goal is to create a functional version that can be tested safely before broader release. Development should be iterative, with feedback from business users, technical teams, risk owners, and support teams.

- **Worksheet prompt:** What will be built in the first working version?
- **Activities:** configure agent instructions, connect approved tools, build workflow logic, define decision rules, create approval gates, implement logging, and set up fallback actions.
- **Deliverables:** prototype agent, tool integration list, prompt or instruction set, guardrail configuration, workflow automation, and development test results.
- **Design tip:** Build the smallest useful version first. Prove the workflow works before adding broad autonomy or advanced integrations.

9.3 Testing

Testing validates whether the agent behaves correctly, safely, and consistently across expected and unexpected scenarios. Testing should include functional testing, integration testing, security review, guardrail testing, user acceptance testing, and pilot evaluation.

- **Worksheet prompt:** How will the team confirm the agent is ready for users?

- **Activities:** run test scenarios, compare outputs with human benchmarks, test fallback behavior, check access controls, validate logs, test low-confidence cases, and collect pilot feedback.
- **Deliverables:** test report, issue log, quality scorecard, security sign-off, user acceptance feedback, and go/no-go recommendation.
- **Design tip:** Retest after every major change to the agent instructions, model, data source, tool, or workflow rule.

9.4 Deployment

Deployment releases the agent into a controlled production environment. Depending on risk, deployment may begin with a limited pilot, a small user group, read-only mode, or approval-only execution before expanding to wider use. Deployment should include user communication, support readiness, monitoring, and rollback planning.

- **Worksheet prompt:** How will the agent be released safely?
- **Activities:** confirm deployment approval, enable monitoring dashboards, brief users, train reviewers, publish support guidance, activate escalation paths, and track early usage.
- **Deliverables:** deployment checklist, release communication, production access approval, support plan, monitoring dashboard, and rollback plan.

- **Design tip:** Use phased deployment for higher-risk workflows so issues can be detected and corrected before full rollout.

9.5 Post-Launch Review

Post-launch review checks whether the agent is delivering the intended value and operating safely. This phase should compare actual performance against success metrics, review user feedback, assess incidents or errors, and decide whether to scale, improve, pause, or retire the agent.

- **Worksheet prompt:** What should be reviewed after launch, and when?
- **Activities:** review KPI performance, analyze time saved, audit quality scores, check security logs, review escalation patterns, collect user feedback, and update the roadmap.
- **Deliverables:** post-launch report, benefits realization summary, issue remediation plan, improvement backlog, governance review, and scale-up recommendation.
- **Design tip:** Schedule the first review within 30 to 60 days of launch, then continue periodic reviews based on risk and usage volume.

Conclusion

An agentic AI project should begin with a clear business problem, a well-defined workflow, measurable success metrics, and practical guardrails. This worksheet helps teams move from an idea to a structured implementation plan by clarifying the use case, mapping the workflow, separating AI from regular automation, defining responsibilities, identifying risks, and planning for testing, monitoring, and rollout. The strongest projects are not the ones that use the most AI; they are the ones that apply AI deliberately where reasoning and autonomy create real business value while keeping people, controls, and accountability in place.

AGENTIC AI FOUNDATION CERTIFICATION

AGENTIC AI FOUNDATION, BASED ON
THE PRINCIPLES OF ETHICS AND
RESPONSIBILITY, DRIVES AI
INNOVATION.



ABOUT GSDC CERTIFICATION



LIFETIME VALIDITY

GSDC Certification is an globally
accredited certification with lifetime
validity.



EBOOK

Extensive and exclusive Ebook created by
world's experts to help you with
understanding core concepts.



CREATED BY EXPERTS

GSDC certifications are created and
authored by world's leading experts in the
field.



LEARNING MATERIALS

Get access to learning materials such as videos,
ebooks, templates, and practice exams, which
will help you clear the certification exam.

LEARNING OBJECTIVE

- Access ready-to-implement templates for agentic AI solutions.
- Develop a deep understanding of agentic AI principles.
- Prepare for real-world challenges with agentic AI applications.

Enroll now with the
code **LEARN20** To
avail **20%** discount

Enroll Now



www.gsdcouncil.org