

AI Fundamentals and AI Development Skills

A practical beginner-to-intermediate guide for understanding core AI concepts, generative AI, large language models, prompt design, APIs, integrations, application building, and model selection.

1. AI Fundamentals

AI fundamentals provide the foundation for understanding how intelligent systems are designed, trained, evaluated, and used in real business contexts. Artificial intelligence is a broad field that includes systems capable of recognizing patterns, making predictions, generating content, supporting decisions, and automating tasks that normally require human intelligence. Before learners start building AI-powered solutions, they need to understand what AI can do, what it cannot do, and how to use it responsibly.

1.1 Understand Basic AI and Machine Learning Concepts

Artificial intelligence refers to the ability of a computer system to perform tasks that appear intelligent, such as understanding language, identifying objects in an image, recommending products, or predicting customer behavior. Machine learning is a major branch of AI where systems learn patterns from data instead of being programmed with every rule manually. For example, an email spam filter learns from thousands or millions of examples of spam and non-spam messages so it can classify new emails more accurately over time.

- **Artificial Intelligence:** A broad discipline focused on building systems that can perform tasks requiring reasoning, perception, prediction, or language understanding.
- **Machine Learning:** A method where models learn from data and improve their performance without being explicitly programmed for every scenario.

- **Deep Learning:** A machine learning approach that uses neural networks with many layers to process complex data such as images, speech, and text.
- **Training Data:** The examples used to teach a model how to recognize patterns and make predictions.
- **Model:** The trained system that receives input, processes it, and produces an output such as a prediction, classification, recommendation, or generated response.

Example: In a banking context, a machine learning model may analyze past transaction patterns to detect potential fraud. It does not “know” fraud the way a human investigator does; instead, it identifies statistical patterns such as unusual transaction amounts, unfamiliar locations, rapid repeated transactions, or behavior that differs from the customer’s normal activity.

1.2 Learn How Generative AI Works

Generative AI is a type of AI that creates new content such as text, images, code, audio, video, summaries, or structured documents. Rather than only classifying or predicting, generative AI produces outputs based on patterns learned from large volumes of training data. When a user gives a prompt, the model uses the prompt, its training, and any available context to generate a relevant response.

- **Input:** The user provides a prompt, instruction, question, document, image, or data set.

- **Context processing:** The system interprets the prompt along with any available context such as examples, documents, memory, tools, or conversation history.
- **Pattern generation:** The model predicts and generates the most likely useful output based on learned patterns.
- **Output:** The system produces text, code, images, summaries, classifications, or other requested content.
- **Review and refinement:** The user validates the output, corrects errors, adds missing context, and prompts again if needed.

Example: A trainer may ask a generative AI tool to create a lesson plan on incident management. If the prompt includes the audience level, duration, learning objectives, activity format, and assessment style, the AI can produce a structured training plan. If the prompt is vague, the output may be too generic or unsuitable for the learners.

1.3 Understand LLMs and Their Limitations

A large language model, or LLM, is a generative AI model designed to understand and generate human-like language. LLMs work by processing text as tokens and predicting likely sequences of tokens based on the prompt and context. They can summarize documents, draft emails, answer questions, generate code, translate text, and support brainstorming. However, they do not truly understand the world in the same way humans do, and they can produce confident but incorrect answers.

- **Hallucination risk:** LLMs may generate information that sounds plausible but is inaccurate or unsupported.
- **Context limits:** Models can only process a limited amount of information at one time, depending on their context window.
- **Knowledge freshness:** A model may not know recent events unless connected to a current data source or search tool.
- **Bias and fairness concerns:** Outputs can reflect biases in training data, prompts, or retrieval sources.
- **Reasoning limits:** LLMs can make mistakes in logic, calculations, compliance interpretation, and multi-step tasks.
- **Security and privacy risks:** Sensitive data should not be shared with AI systems unless approved controls are in place.

Example: If an LLM is asked to write an internal policy without access to the organization's approved policy library, it may produce a polished but non-compliant draft. A better approach is to provide the approved policy documents as context, ask the model to extract and summarize only from those documents, and have a subject matter expert review the final output.

1.4 Learn Basic AI Terminology

AI terminology helps learners communicate clearly with business teams, technical teams, vendors, and governance stakeholders. Knowing the vocabulary also helps learners evaluate AI tools more critically and avoid confusion between similar terms such as AI, machine learning, deep learning, generative AI, and LLMs.

- **Prompt:** The instruction or question given to an AI system.
- **Context:** Supporting information that helps the model produce a relevant answer.
- **Token:** A unit of text processed by a language model, such as a word, subword, or character segment.
- **Embedding:** A numerical representation of meaning used to compare and retrieve related content.
- **Fine-tuning:** Additional model training on a specific data set to adapt behavior for a specialized purpose.
- **RAG:** Retrieval-augmented generation, where an AI system retrieves relevant information from trusted sources before generating an answer.
- **Temperature:** A setting that influences how predictable or creative the model's output is.

- **Agent:** An AI-powered system that can follow goals, use tools, make decisions within constraints, and perform multi-step tasks.

Example: A customer support chatbot using RAG may retrieve the latest refund policy from a company knowledge base before answering a customer question. This is more reliable than asking the model to answer from memory because the response is grounded in approved content.

2. AI Development Skills

AI development skills focus on turning AI concepts into usable applications, workflows, automations, and intelligent products. A strong AI developer does not only write prompts or call APIs; they design context, connect tools, manage data securely, select appropriate models, test outputs, and build guardrails so AI systems are useful, reliable, and responsible.

2.1 Learn Prompt and Context Design

Prompt design is the skill of writing clear instructions that guide an AI system toward the desired output. Context design goes further by deciding what information, examples, constraints, documents, tool outputs, and user preferences the AI should receive before it responds. In production systems, context design is often more important than clever wording because the model needs the right environment to produce the right answer.

- **Define the role:** Tell the AI what perspective to use, such as trainer, developer, auditor, incident manager, or product owner.
- **State the task clearly:** Explain what output is required and what problem it should solve.
- **Provide context:** Include audience, objective, source material, constraints, examples, and expected level of detail.

- **Specify the format:** Ask for a table, checklist, report, lesson plan, JSON output, email draft, or step-by-step guide.
- **Add quality criteria:** Mention accuracy, tone, compliance boundaries, assumptions, exclusions, and review requirements.

Example prompt: “Act as an AI trainer. Create a 30-minute beginner lesson on generative AI for business analysts. Include learning objectives, simple examples, a short activity, and three reflection questions. Use a professional but easy-to-understand tone.” This prompt is stronger than “Explain generative AI” because it gives the model a role, audience, format, duration, and output expectations.

2.2 Work with AI APIs

AI APIs allow developers to connect applications to AI models through programmatic requests. Instead of using a chat interface manually, an application sends structured input to an AI service and receives structured output. This enables products such as document summarizers, support assistants, code copilots, content generators, workflow automations, and decision-support tools.

- **Authentication:** Use approved keys, tokens, or identity controls to access the AI service securely.
- **Request design:** Send prompts, system instructions, parameters, and context in a structured request.

- **Response handling:** Parse the model output and display, store, validate, or pass it to another service.
- **Error handling:** Plan for rate limits, timeouts, invalid inputs, unavailable services, and incomplete outputs.
- **Cost and usage monitoring:** Track token usage, latency, model calls, and budget impact.
- **Security controls:** Avoid exposing secrets, personal data, confidential records, or unapproved sensitive content.

Example: A project management application can use an AI API to convert meeting notes into action items. The application sends the notes, asks the model to identify owners, due dates, decisions, risks, and unresolved questions, then displays the structured output for human review before saving it to the project tracker.

2.3 Understand Tool Calling and Integrations

Tool calling allows an AI system to use external functions, applications, databases, or services to complete a task. Instead of only generating text, the AI can retrieve information, trigger workflows, look up records, create tickets, update calendars, or call APIs. Integrations are essential for building AI systems that can act on real business processes rather than simply answer questions.

- **Tool schema:** A clear definition of what the tool does, what inputs it needs, and what output it returns.

- **Permission control:** Rules that define when the AI is allowed to call a tool and what actions require approval.
- **Validation:** Checks to ensure the AI passes correct inputs and handles tool outputs safely.
- **Workflow integration:** Connections to systems such as CRM, ITSM, HRMS, ERP, email, calendar, data platforms, or ticketing systems.
- **Human-in-the-loop:** Approval steps for high-risk actions such as sending messages, changing records, approving payments, or closing incidents.

Example: An IT service desk AI agent may analyze an incoming incident, search the knowledge base, check service health, suggest a category and priority, and draft an initial response. If configured safely, it may create or update a ticket automatically, but escalation, closure, or customer communication may still require human approval.

2.4 Build Simple AI-Powered Applications

Building simple AI-powered applications helps learners move from theory to practice. A basic AI application usually includes a user interface, input capture, context preparation, model call, output display, validation, and feedback loop. The goal is not to build a complex system immediately, but to understand how AI becomes part of a working product or workflow.

- **Summarizer:** Upload or paste a document and generate a concise summary with key points and next steps.

- **Question-answer bot:** Ask questions over approved knowledge base content using retrieval-augmented generation.
- **Email drafting assistant:** Generate professional replies based on tone, audience, and purpose.
- **Code helper:** Explain code, generate test cases, or suggest improvements within defined constraints.
- **Checklist generator:** Convert a policy, process, or standard into an actionable checklist.

Example application flow: A learner builds an AI meeting-note assistant. The user pastes meeting notes, the system sends the notes with instructions to extract decisions and action items, the model returns a structured list, and the user reviews the list before exporting it to a task tracker. This teaches prompt design, API calls, structured output, validation, and user review in one small project.

2.5 Understand Basic Model Selection

Model selection is the process of choosing the most suitable AI model for a task based on capability, cost, speed, context length, accuracy, data sensitivity, integration needs, and risk level. The best model is not always the largest or most expensive one. A smaller model may be sufficient for classification, formatting, or routing, while a more advanced model may be needed for complex reasoning, long documents, coding, or multimodal analysis.

- **Task fit:** Match the model to the required output, such as text generation, classification, coding, image understanding, or embeddings.
- **Accuracy requirement:** Use stronger models for high-stakes tasks and add validation or human review.
- **Latency:** Choose faster models for real-time chat, support, and interactive applications.
- **Cost:** Balance model quality with token usage, call volume, and budget limits.
- **Context length:** Select a model that can handle the amount of information needed for the task.
- **Data governance:** Consider privacy, compliance, retention, auditability, and approved deployment options.
- **Evaluation results:** Test models against sample tasks before selecting one for production.

Example: For a simple email classification task, a lightweight model may be enough to label messages as urgent, routine, spam, or follow-up. For a legal contract review assistant, however, the organization may need a stronger model, retrieval from approved legal templates, strict guardrails, audit logs, and mandatory expert review.

3. Agentic AI Skills

Agentic AI skills help learners move beyond simple prompt-and-response systems toward AI solutions that can plan, use tools, make decisions within boundaries, and complete multi-step tasks. An AI agent is usually designed around a goal, available tools, memory or state, decision logic, and safety controls. The most important skill is not only making an agent act, but making sure it acts reliably, securely, and within approved limits.

3.1 Understand How AI Agents Work

An AI agent works by receiving a goal, deciding what step to take next, using tools or information sources when needed, observing the result, and then deciding whether to continue, change direction, or stop. Unlike a basic chatbot that usually responds to one prompt at a time, an agent may run a sequence of actions to complete a broader objective. For example, a support agent may read a customer issue, search the knowledge base, check account details, draft a response, and recommend whether the case should be escalated.

- **Goal:** The outcome the agent is expected to achieve, such as resolving a ticket or preparing a report.
- **Planning:** The agent breaks the goal into smaller steps and decides the next action.

- **Tool use:** The agent calls approved tools such as search, databases, APIs, calendars, ticketing systems, or code execution environments.
- **Observation:** The agent reads the output of a tool or previous action and updates its understanding.
- **State:** The agent keeps track of progress, decisions, tool results, and remaining work.
- **Stopping condition:** The agent knows when the task is complete or when it must ask for human help.

Example: In an ITSM workflow, an incident triage agent may receive a new incident, classify the affected service, compare it with known outages, suggest priority, draft a first response, and assign it to the correct resolver group. The agent should not close the incident automatically unless the organization has explicitly approved that level of autonomy.

3.2 Build a Simple AI Agent

Building a simple AI agent starts with selecting a narrow use case, defining the goal, listing the tools it can use, and setting clear boundaries. Beginners should avoid creating agents that can perform broad or risky actions. A good first agent has a predictable task, limited permissions, simple inputs, and a review step before any important action is taken.

- **Choose a focused use case:** Start with a task such as summarizing meeting notes, creating a checklist, reviewing FAQs, or drafting task updates.
- **Define the input:** Identify what the agent receives, such as a document, ticket, form, email, or chat message.
- **Define the output:** Specify exactly what the agent should produce, such as a summary, recommendation, classification, or draft response.
- **Add tools carefully:** Provide only the tools needed for the task and avoid unnecessary access.
- **Add a review step:** Let a human approve outputs before sending messages, updating systems, or making decisions.
- **Test with examples:** Run the agent on normal, unusual, incomplete, and risky scenarios before wider use.

Example: A simple project update agent can read weekly team notes and produce a draft status update with accomplishments, blockers, risks, next steps, and open questions. The agent should mark uncertain items clearly instead of inventing missing information. A project manager can then review, edit, and send the final update.

3.3 Connect Agents to External Tools

External tools turn an agent from a text generator into a workflow participant. Tools may include search engines, internal knowledge bases, CRM systems, ITSM platforms, project

management tools, databases, code repositories, spreadsheets, email, or calendar systems. Tool access must be designed with strong controls because an incorrect tool call can create real business impact.

- **Tool description:** Explain what each tool does and when the agent should use it.
- **Input validation:** Check that the agent sends complete and safe inputs to the tool.
- **Output validation:** Inspect tool results before using them in the next step or final response.
- **Access control:** Limit the agent to the data and actions needed for its role.
- **Audit logging:** Record which tools were used, what inputs were sent, and what actions were taken.
- **Fallback handling:** Define what the agent should do if a tool fails, times out, or returns unclear results.

Example: A procurement assistant agent may connect to supplier records, policy documents, risk ratings, and contract templates. It can gather evidence and recommend whether a supplier requires enhanced due diligence, but final approval should remain with an authorized procurement or risk professional.

3.4 Understand Multi-Agent Workflows

A multi-agent workflow uses more than one specialized agent to complete a larger task. Instead of one agent trying to do everything, different agents can handle research,

analysis, drafting, compliance review, testing, customer communication, or implementation. This separation can improve focus and control, but it also introduces coordination challenges such as hand-offs, duplicate work, conflicting outputs, and unclear accountability.

- **Sequential workflow:** Agents work one after another, such as research agent, drafting agent, reviewer agent, and publishing agent.
- **Parallel workflow:** Multiple agents work at the same time on different parts of a task and then combine results.
- **Supervisor workflow:** A coordinator agent assigns tasks, reviews outputs, and decides next steps.
- **Specialist workflow:** Each agent has a narrow responsibility, such as data extraction, compliance checking, code review, or customer response drafting.
- **Escalation workflow:** Agents pass the work to a human expert when confidence is low, risk is high, or policy requires approval.

Example: In a software release workflow, one agent may summarize feature requirements, another may generate code, another may create tests, another may check security risks, and a final reviewer agent may prepare a release note. A human release manager should approve the final deployment decision.

3.5 Set Goals, Permissions, and Guardrails

Goals, permissions, and guardrails define the safe operating space for an AI agent. The goal tells the agent what to achieve. Permissions define what the agent is allowed to access or do. Guardrails define the rules, checks, limits, and escalation paths that prevent harmful, unauthorized, or unreliable behavior. Effective guardrails should operate throughout the workflow, not only at the final response stage.

- **Clear goal boundaries:** Define the task outcome, scope, success criteria, and stopping point.
- **Least-privilege access:** Give the agent only the tools, data, and actions required for its role.
- **Approval gates:** Require human approval before high-impact actions such as sending external messages, changing records, executing code, or spending money.
- **Policy checks:** Validate actions against organizational rules, legal requirements, security standards, and compliance obligations.
- **Runtime limits:** Set limits for time, number of steps, cost, tool calls, retries, and autonomous decision depth.
- **Monitoring and auditability:** Capture logs, explanations, tool traces, and exception reports for review.

Example: A finance agent may be allowed to prepare invoice summaries and flag exceptions, but it should not approve payments, change vendor bank details, or release funds without authorized human approval. The guardrail design must reflect the risk of the action, not just the intelligence of the model.

4. AI Developer Tools

AI developer tools help software teams write, understand, test, improve, and document code more efficiently. These tools are useful as coding assistants, but they should not replace engineering judgment. Developers still need to review generated code, validate assumptions, run tests, check security implications, and ensure that the final implementation fits the architecture and business requirement.

4.1 Explore AI Coding Assistants

AI coding assistants are tools that support developers inside code editors, integrated development environments, terminals, repositories, or cloud workspaces. They can suggest code, explain unfamiliar logic, generate boilerplate, create tests, summarize pull requests, and help developers navigate large codebases. The best use of these assistants is as a collaborative support layer, where the developer remains responsible for design, quality, and correctness.

- **Inline completion tools:** Suggest code as the developer types.
- **Chat-based coding assistants:** Answer questions, explain code, and provide implementation guidance.
- **AI-native code editors:** Provide stronger repository-level context and multi-file editing support.

- **Terminal agents:** Help inspect projects, run commands, modify files, and troubleshoot issues.
- **Repository assistants:** Support pull request summaries, issue analysis, code review, and documentation updates.

Example: A developer working on a new API endpoint may ask an AI coding assistant to inspect the existing controller pattern, generate a draft endpoint, create unit tests, and explain where validation should be added. The developer should still review the generated code for security, maintainability, naming consistency, and architectural fit.

4.2 Use AI for Code Generation and Refactoring

AI can accelerate code generation by producing functions, classes, interface definitions, API routes, database queries, scripts, configuration files, and sample implementations from natural language instructions. It can also support refactoring by improving readability, reducing duplication, simplifying complex logic, extracting reusable functions, and aligning code with team standards. However, generated or refactored code must always be reviewed and tested because AI may miss hidden dependencies or business rules.

- **Generate boilerplate:** Create repetitive structures such as controllers, service classes, data models, or configuration files.
- **Translate requirements into code:** Convert simple user stories or acceptance criteria into an initial implementation draft.

- **Refactor safely:** Ask AI to preserve external behavior while improving readability and structure.
- **Modernize legacy code:** Suggest newer language features, framework patterns, or modular designs.
- **Explain trade-offs:** Compare different implementation options before making a change.

Example: A developer may ask AI to refactor a long function that calculates invoice totals. The assistant can suggest smaller functions for tax calculation, discount handling, validation, and rounding. The developer should then run regression tests to confirm that the refactor did not change the output for existing invoices.

4.3 Use AI for Testing and Debugging

AI can support testing by generating unit tests, integration test scenarios, edge cases, mock data, and negative test cases. It can support debugging by explaining error messages, tracing logic, identifying likely failure points, and suggesting diagnostic steps. This is especially useful when working with unfamiliar codebases or complex failures, but AI suggestions should be treated as hypotheses until verified by tests and logs.

- **Generate test cases:** Ask AI to cover normal paths, edge cases, invalid inputs, missing data, and boundary values.
- **Improve test coverage:** Use AI to identify untested branches or conditions.

- **Explain failures:** Provide stack traces or logs and ask for likely causes and next diagnostic steps.
- **Create mock data:** Generate realistic but non-sensitive data for local testing.
- **Debug systematically:** Ask AI to suggest a step-by-step investigation plan instead of jumping to conclusions.

Example: If a payment API fails only for certain currencies, a developer can ask AI to propose test cases for currency formats, decimal precision, rounding rules, exchange-rate lookups, and validation failures. The developer should then run those tests against the actual implementation and verify the root cause with logs.

4.4 Use AI for Documentation

AI can help developers create and maintain documentation that is consistent, readable, and easier to update. Documentation tasks include explaining code behavior, creating API documentation, writing setup guides, summarizing architecture decisions, generating release notes, and drafting onboarding guides. Good documentation should be accurate, current, and connected to the actual codebase or process it describes.

- **Code comments:** Explain complex logic, assumptions, and non-obvious decisions.
- **API documentation:** Describe endpoints, request parameters, response formats, errors, and authentication requirements.

- **README files:** Create setup instructions, project structure summaries, and local development steps.
- **Architecture notes:** Summarize components, dependencies, integration points, and design decisions.
- **Release notes:** Convert commit messages, tickets, or pull requests into user-friendly change summaries.

Example: After building a new reporting module, a developer can ask AI to draft API usage instructions, explain required permissions, list configuration steps, and create troubleshooting tips. The developer should validate every endpoint, parameter, and command before publishing the documentation.

4.5 Learn to Evaluate AI-Generated Code

Evaluating AI-generated code is a critical developer skill. AI may produce code that looks correct but contains security gaps, inefficient logic, hidden assumptions, licensing concerns, incomplete error handling, or incompatibility with the existing architecture. Developers should inspect AI-generated code with the same discipline used for code from any teammate: review, test, secure, document, and monitor it.

- **Correctness:** Does the code meet the requirement and handle expected inputs?
- **Security:** Does it avoid injection risks, hardcoded secrets, weak authentication, insecure dependencies, and unsafe data handling?

- **Maintainability:** Is the code readable, modular, consistent with naming standards, and easy to change?
- **Performance:** Does it avoid unnecessary loops, inefficient queries, excessive memory use, or slow external calls?
- **Error handling:** Does it handle failures, invalid inputs, timeouts, and unavailable services gracefully?
- **Testability:** Can the code be tested with clear unit, integration, and regression tests?
- **Compliance:** Does the code follow organizational, legal, privacy, and open-source usage policies?

Example: If AI generates a login function, the developer should check password handling, session management, account lockout behavior, logging, error messages, input validation, and authentication library usage. The function may appear simple, but security-sensitive code requires expert review and thorough testing before deployment.

5. AI Security & Governance

AI security and governance ensure that AI systems are used safely, responsibly, and in alignment with legal, ethical, and organizational expectations. As AI becomes embedded in business processes, teams must manage risks such as data leakage, unauthorized access, biased outputs, hallucinations, insecure integrations, and over-automation. Strong governance is not meant to slow innovation; it provides the structure needed to scale AI with trust.

5.1 Understand AI Security Risks

AI systems introduce new security risks because they process large amounts of data, interact with users through natural language, connect to external tools, and may influence automated decisions. Common risks include prompt injection, sensitive data exposure, model misuse, insecure plugins, data poisoning, hallucinated instructions, and excessive agent permissions. Developers and business owners must understand these risks before deploying AI into real workflows.

- **Prompt injection:** A malicious or accidental instruction attempts to override system rules or make the AI ignore controls.
- **Data leakage:** Sensitive information may be exposed through prompts, outputs, logs, integrations, or poorly configured access.
- **Unauthorized tool use:** An AI agent may call a tool or system action beyond its intended role.

- **Model hallucination:** The system may generate false information, incorrect recommendations, or unsupported claims.
- **Supply chain risk:** External models, libraries, plugins, and APIs may introduce vulnerabilities or compliance issues.
- **Over-reliance risk:** Users may trust AI outputs without review, especially when the answer appears confident and well-written.

Example: A chatbot connected to internal documents may be tricked into revealing confidential information if permissions and retrieval controls are weak. A secure design should enforce user-level access, filter sensitive content, log retrieval events, and refuse requests outside the approved scope.

5.2 Learn About Data Privacy

Data privacy focuses on protecting personal, confidential, regulated, and sensitive information throughout the AI lifecycle. AI systems may process user prompts, uploaded documents, logs, training data, retrieved records, and generated outputs. Teams must know what data is being used, where it is stored, who can access it, whether it is retained, and whether it is appropriate for the AI use case.

- **Data minimization:** Use only the data required for the task and avoid unnecessary collection.
- **Consent and purpose limitation:** Use data only for approved purposes and in accordance with applicable policies.

- **Masking and anonymization:** Remove or obscure personal and sensitive information where full detail is not needed.
- **Retention controls:** Define how long prompts, outputs, logs, and model interactions are stored.
- **Access control:** Ensure users and AI systems can only access information they are authorized to use.
- **Vendor and platform review:** Confirm whether external AI services meet privacy, security, and compliance requirements.

Example: If a healthcare team uses AI to summarize patient notes, the system must protect personal health information, restrict access to authorized users, and avoid storing sensitive prompts in unapproved environments. Privacy design should be part of the workflow from the beginning, not added after deployment.

5.3 Set Appropriate Permissions

Permissions define what an AI system, user, or agent is allowed to see, retrieve, create, update, delete, or execute. In AI systems, permissions are especially important because the model may connect to multiple tools and data sources. A safe design follows the principle of least privilege: give the AI only the access required to complete the task and nothing more.

- **Read-only access:** Use read-only permissions when the AI only needs to summarize, search, or answer questions.

- **Role-based access:** Match permissions to the user's function, department, and responsibility.
- **Action boundaries:** Clearly define whether the AI can draft, recommend, update, approve, delete, or execute.
- **Approval gates:** Require human confirmation before high-risk actions such as sending external messages or changing records.
- **Temporary access:** Use time-limited permissions for special tasks instead of permanent broad access.
- **Audit trails:** Record AI actions so administrators can review what happened and why.

Example: A sales assistant may be allowed to read customer interaction history and draft follow-up emails, but it should not change pricing, approve discounts, or send final messages without review. The permission model should reflect business risk and accountability.

5.4 Add Human Review Where Needed

Human review is essential when AI outputs affect people, money, legal obligations, customer communication, safety, compliance, or critical business operations. Human-in-the-loop design allows AI to assist with analysis and drafting while keeping decision accountability with an authorized person. The level of review should increase as the risk, uncertainty, or business impact increases.

- **Low-risk tasks:** AI may produce summaries, draft notes, or organize information with light review.
- **Medium-risk tasks:** AI recommendations should be reviewed before being shared or acted upon.
- **High-risk tasks:** Human approval should be mandatory before decisions are executed.
- **Uncertain outputs:** AI should flag low-confidence results and ask for review instead of guessing.
- **Exceptions:** Workflows should define when the AI must escalate to a specialist, manager, auditor, or security team.

Example: In recruitment, AI may help summarize candidate profiles or organize interview notes, but final hiring decisions should remain with trained human decision-makers who follow fair, transparent, and compliant hiring practices.

5.5 Understand Basic AI Governance

AI governance is the set of policies, roles, processes, standards, and controls used to guide how AI systems are selected, built, deployed, monitored, and retired. It helps organizations manage risk, assign accountability, protect data, maintain transparency, and demonstrate compliance. Good AI governance covers the complete lifecycle, from idea intake and risk assessment to testing, deployment, monitoring, incident response, and continuous improvement.

- **Accountability:** Identify who owns the AI system, its outcomes, and its risks.
- **Risk classification:** Assess the impact level of each AI use case before deployment.
- **Transparency:** Document data sources, model behavior, assumptions, limitations, and intended use.
- **Testing and evaluation:** Validate accuracy, safety, fairness, security, and reliability before launch.
- **Monitoring:** Track performance, drift, user feedback, misuse, security events, and business impact.
- **Change management:** Control updates to prompts, models, data sources, tools, and workflows.

Example: Before deploying an AI customer service assistant, an organization may require a use-case assessment, data privacy review, security review, model evaluation, user testing, escalation process, and monitoring dashboard. These governance steps help ensure the assistant is useful, safe, accountable, and aligned with business policy.

6. Practical Experience

Practical experience turns AI knowledge into visible capability. Employers, clients, and project sponsors are more convinced by working examples than by theory alone. Learners should build small but complete projects that demonstrate problem definition, data handling, prompt or model design, tool integration, testing, documentation, and user value.

6.1 Build 1–2 AI Projects

Building one or two focused AI projects helps learners apply the fundamentals in a realistic setting. The project does not need to be large, but it should solve a clear problem and include evidence of testing and iteration. Good beginner projects often involve summarization, classification, information retrieval, structured extraction, content generation, or workflow support.

- **Document summarizer:** Upload a long document and generate an executive summary, key risks, action items, and open questions.
- **Knowledge-base assistant:** Answer questions from approved documents using retrieval instead of model memory.
- **Resume or CV analyzer:** Compare a CV against a target role and suggest skill gaps.

- **Support-ticket classifier:** Categorize tickets by topic, urgency, affected service, and recommended resolver group.
- **Policy checklist generator:** Convert a policy document into a practical checklist for teams.

Example: A learner may build a document summarizer for project status reports. The tool accepts a report, extracts achievements, risks, blockers, decisions, and next steps, then produces a structured project update. This demonstrates prompt design, output formatting, validation, and practical business usefulness.

6.2 Build One Agentic AI Project

An agentic AI project should show that the learner can design an AI system that works toward a goal, uses tools, keeps track of progress, and follows guardrails. The project should be narrow enough to test properly but rich enough to demonstrate planning, tool calling, permissions, error handling, and human review.

- **Research assistant agent:** Search approved sources, summarize findings, identify gaps, and produce a brief with citations or source notes.
- **Incident triage agent:** Read an incident description, classify priority, suggest resolver group, and draft a first response for review.
- **Meeting action agent:** Extract decisions, action items, owners, dates, and follow-ups from meeting notes.

- **Code review assistant:** Inspect code changes, identify risks, suggest tests, and prepare a review summary.
- **Learning coach agent:** Recommend next learning steps based on a learner's current skills and project results.

Example: A learner can build an incident triage agent that receives a ticket, extracts affected application, impact, urgency, symptoms, and user group, checks a knowledge base, recommends priority, and drafts an escalation note. The agent should include a rule that prevents automatic closure and requires human approval before assignment changes.

6.3 Document Your Projects

Project documentation explains what was built, why it was built, how it works, and what was learned. Good documentation helps reviewers understand the problem, design choices, implementation approach, limitations, and future improvements. Documentation also shows communication skills, which are essential for AI roles because technical solutions must often be explained to non-technical stakeholders.

- **Problem statement:** Explain the business or user problem the project addresses.
- **Use case and audience:** Describe who will use the solution and what value it provides.
- **Architecture overview:** Show the main components, data flow, model calls, tools, and integrations.

- **Prompt or model design:** Explain how prompts, context, retrieval, or model selection were handled.
- **Testing approach:** Include sample test cases, edge cases, evaluation results, and known limitations.
- **Lessons learned:** Summarize what worked, what failed, and what you would improve next.

Example: For a knowledge-base assistant, the documentation should explain the source documents used, chunking approach, retrieval method, answer format, evaluation examples, privacy considerations, and limitations such as unsupported topics or outdated source material.

6.4 Add Projects to Your Portfolio

A portfolio turns learning into evidence. It should help someone quickly understand what you built, what problem it solves, what skills it demonstrates, and how to inspect the work. Each portfolio entry should be clear, honest, and specific. Avoid simply copying tutorials; instead, customize the project, explain your design choices, and show how you tested or improved it.

- **Project title:** Use a clear title that describes the outcome, such as “AI Incident Triage Assistant.”
- **Short summary:** Explain the project in two or three sentences.

- **Skills demonstrated:** List skills such as prompt design, RAG, APIs, tool calling, testing, evaluation, or governance.
- **Live demo or screenshots:** Provide evidence that the project works, where appropriate.
- **Repository link:** Include clean code, setup instructions, sample inputs, and documentation.
- **Reflection:** Explain limitations, trade-offs, and future improvements.

Example: A strong portfolio entry might state: “Built an AI support-ticket classifier that categorizes incidents, identifies missing details, and drafts a first-response message. The project demonstrates structured output, prompt design, evaluation with sample tickets, and human-review workflow design.”

6.5 Contribute to an AI or Open-Source Project

Contributing to an AI or open-source project helps learners gain experience with real collaboration, issue tracking, code review, documentation standards, and community practices. Contributions do not always need to be complex code changes. Beginners can start with documentation, examples, test cases, bug reports, tutorials, or small improvements.

- **Start with documentation:** Improve README files, setup instructions, examples, or troubleshooting notes.

- **Fix small issues:** Look for beginner-friendly issues such as typos, broken links, simple bugs, or missing tests.
- **Add test cases:** Contribute edge-case tests or sample input/output files.
- **Share examples:** Create practical examples that show how to use an AI tool or library.
- **Participate respectfully:** Follow contribution guidelines, communicate clearly, and respond professionally to feedback.

Example: A learner using an open-source RAG framework may contribute a beginner tutorial that shows how to connect documents, create embeddings, retrieve relevant passages, and evaluate answer quality. This contribution demonstrates both technical understanding and teaching ability.

7. Career Readiness

Career readiness connects learning, projects, credentials, and professional visibility. AI career growth depends not only on what a learner knows, but also on how clearly they can show applied skills, explain project decisions, communicate responsibly, and continue learning as tools and practices evolve.

7.1 Update Your CV with Relevant AI Skills

A CV should present AI skills in a practical, evidence-based way. Instead of only listing broad terms such as “AI” or “machine learning,” learners should describe tools used, project outcomes, types of problems solved, and measurable results where possible. The goal is to show that the learner can apply AI responsibly in real work contexts.

- **Skills section:** Add relevant skills such as prompt engineering, generative AI, AI APIs, RAG, agentic workflows, evaluation, and AI governance.
- **Project section:** Include 2–3 AI projects with problem, approach, tools, and result.
- **Experience section:** Describe how AI was used to improve productivity, automate work, support decisions, or build solutions.
- **Credential section:** Add relevant AI certifications, courses, or workshops.
- **Impact statement:** Use outcome-oriented language, such as “Built a prototype that reduced manual report summarization effort.”

Example: Instead of writing “Knowledge of AI tools,” a stronger CV bullet would be: “Built a document summarization prototype using generative AI prompts and structured output to extract key decisions, risks, and action items from project reports.”

7.2 Showcase AI Projects on LinkedIn or GitHub

LinkedIn and GitHub help learners make their AI work discoverable. LinkedIn is useful for explaining the business value and learning journey, while GitHub is useful for showing code, documentation, examples, and technical implementation. A strong profile connects both: a LinkedIn post can explain the project story, while the GitHub repository provides evidence of the work.

- **LinkedIn project post:** Explain the problem, solution, tools used, lessons learned, and next improvement.
- **GitHub README:** Include setup steps, project purpose, architecture, screenshots, sample outputs, and limitations.
- **Project tags:** Use relevant keywords such as generative AI, RAG, AI agents, model evaluation, or AI governance.
- **Visual evidence:** Add screenshots, workflow diagrams, sample outputs, or demo videos where possible.
- **Professional tone:** Be honest about what the project does and does not do.

Example: A LinkedIn post could say: “I built a small AI meeting-action assistant that extracts decisions, owners, due dates, and risks from notes. This helped me practice structured prompting, human review, and project documentation.” The GitHub repository would then include the code, sample notes, outputs, and README.

7.3 Write About What You Are Learning

Writing about learning helps build clarity, confidence, and professional credibility. Short posts, tutorial notes, case studies, and reflections show that the learner can explain AI concepts in practical language. Writing also helps identify gaps in understanding because explaining a topic clearly requires more than surface-level familiarity.

- **Concept notes:** Explain topics such as prompts, RAG, embeddings, agents, hallucinations, or model evaluation.
- **Project reflections:** Share what you built, what failed, and what you improved.
- **Tool comparisons:** Compare AI tools based on practical use, limitations, and learning value.
- **Mini tutorials:** Write step-by-step guides for small AI tasks or workflows.
- **Responsible AI reflections:** Discuss privacy, review, guardrails, and governance lessons learned from projects.

Example: After building a RAG-based assistant, a learner might write a short post titled “What I learned about retrieval quality while building my first AI knowledge-

base assistant.” The post could cover source quality, chunking, retrieval accuracy, hallucination prevention, and user testing.

7.4 Earn a Recognised AI Credential

A recognised AI credential can strengthen career readiness by showing structured learning, commitment, and baseline knowledge. Credentials are most valuable when combined with portfolio projects and practical examples. Learners should choose credentials that match their career direction, such as AI fundamentals, AI engineering, responsible AI, data science, cloud AI services, AI security, or agentic AI.

Conclusion

AI development is changing quickly, but you don't need to learn everything at once. Focus on building strong fundamentals, gaining hands-on experience, and developing the skills that matter most for working with AI and agentic systems.

Use this checklist to identify your gaps, choose your next few learning goals, and track your progress. The goal isn't to know every AI tool—it's to become a developer who knows how to use AI effectively, responsibly, and confidently.

AGENTIC AI DEVELOPER CERTIFICATION

**AGENTIC AI DEVELOPER
CERTIFICATION BASED ON REAL-
WORLD AGENT FRAMEWORKS TO
DESIGN, BUILD, AND DEPLOY
AUTONOMOUS AI SYSTEMS.**



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

- Improve problem-solving with real-world examples
- Accelerate learning through hands-on resources
- Prepare for advanced AI developer roles
- Boost credibility among AI employers

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

Enroll Now



www.gsdccouncil.org