Human-Centred AI Agent Design Checklist

25 questions to validate your AI agent before you build

Use this checklist to design AI agents that are reliable, transparent and trusted by the people who use them.

How to Use This Checklist

AI agents are no longer simple chatbots. They read requests, plan tasks, call tools, update records and make decisions on their own. That capability is useful, but it also means a poorly designed agent can make mistakes that customers and employees notice quickly. Once trust is lost, adoption drops, and even a technically strong agent ends up unused.

This checklist helps you design for trust from the start. It turns human-centred AI principles into 25 practical questions you can answer with your team before you write a single line of code.

Who it's for

- **Product managers** : defining what the agent should and should not do.
- **UX designers** : shaping how people see, approve and correct agent actions.
- **Business analysts** : mapping workflows, decision points and approval rules.
- **AI and automation professionals** : building, testing and monitoring the agent.

When to use it

- **During planning:** to shape scope and requirements before development begins.
- **Before development:** to confirm the design covers oversight, transparency and failure handling.
- **Before deployment:** as a final readiness review with stakeholders.

How to score it

Mark each question with one of three answers. Be honest: a 'Partially' today is far cheaper to fix than a production incident later.

Mark	Meaning
☐Yes	Fully addressed, documented and agreed by the team.
☐Partially	Discussed or started, but not complete, tested or documented.
☐No	Not yet considered or no evidence it is in place.

How the sections map to design thinking

Each section follows a stage of the design thinking process, so you move from understanding people to testing your solution in a natural order.

Stage	Checklist section	Core question
Empathise	1. Understand the people	Who uses the agent and what do they need?
Define	2. Define responsibilities	What exactly is the agent allowed to do?
Prototype	3. Transparency & 4. Oversight	How will people see and control the agent?
Test	5. Errors and unexpected situations	What happens when things go wrong?

Running example used in this guide

Throughout this checklist, we refer to a customer support agent that reads incoming messages, identifies the issue, retrieves information from company systems, recommends a solution, updates the support ticket and escalates complex cases to a human. Other examples cover invoicing, recruitment, sales and onboarding.

SECTION 1 | DESIGN THINKING STAGE: EMPATHISE

Understand the People Behind the Workflow

Most AI agent failures are not model failures. They happen because the agent was built around a task rather than around the people doing that task. Empathy work uncovers what users actually need, what they fear and where they want to stay in control.

Q1. Have you identified the primary users of the agent and their goals?

☐Yes ☐Partially ☐No

Why it matters: An agent built for 'everyone' ends up serving no one well. Different users have different goals, skills and tolerance for automation.

Example: *A customer support agent has at least two user groups: support executives who want fewer repetitive tickets, and customers who want fast, accurate answers. Their goals overlap but are not identical.*

What good looks like:

- A short list of named user groups, each with one or two clear goals.
- Simple personas (for example, 'Priya, Tier-1 support executive, handles 60 tickets a day').
- Agreement on which user group is the priority for the first release.

Q2. Have you mapped the current user journey the agent will change?

☐Yes ☐Partially ☐No

Why it matters: You cannot improve a workflow you do not understand. Journey mapping shows where time is lost, where errors happen and where an agent can genuinely help.

Example: *Mapping the refund journey reveals that executives spend 40% of their time looking up order history across three systems. That lookup step, not the customer reply, is the best place to start automating.*

What good looks like:

- A step-by-step map of the current process, including tools and handoffs.
- Pain points marked on the map (delays, rework, frustration).
- Clearly highlighted steps that the agent will take over or support.

Q3. Do you know which tasks users want automated and which they want to keep control of?

☐Yes ☐Partially ☐No

Why it matters: Users accept automation for tedious work but resist it for judgement-heavy decisions. Getting this wrong is one of the fastest ways to lose trust.

Example: *A sales team is happy for an agent to research prospects and update CRM fields, but wants to approve every discount offer and every email sent to a key account.*

What good looks like:

- A two-column list: 'Agent can do' versus 'Human must decide'.
- User input behind each item, not just the project team's assumptions.
- Notes on tasks users feel unsure about, to revisit after a pilot.

Q4. Have you validated these assumptions through interviews or observation?

☐Yes ☐Partially ☐No

Why it matters: Teams often design based on what they think users do. Watching real work regularly uncovers workarounds and edge cases no one mentioned in meetings.

Example: *While shadowing support executives, the team notices they keep a personal spreadsheet of 'VIP' customers who should never receive automated replies. That rule becomes a requirement for the agent.*

What good looks like:

- At least five to eight user interviews or observation sessions.
- Key insights recorded and shared with the full team.

- Assumptions that were proven wrong and the design changes they triggered.

Q5. Have you identified who else is affected by the agent's actions?

☐Yes ☐Partially ☐No

Why it matters: Agents rarely affect only their direct users. Customers, finance teams, compliance and partner teams may all feel the impact of an automated decision.

Example: *An agent that auto-closes resolved tickets also changes the support team's SLA reports and the quality team's review sample. Both teams need to be consulted.*

What good looks like:

- A stakeholder map showing direct users, indirect users and affected teams.
- Customers informed about where AI is used in their experience.
- Sign-off or input from compliance, legal or data owners where relevant.

Define the Agent's Responsibilities

A trustworthy agent is a predictable agent. Clear purpose, boundaries and ownership make it easy for users to know what the agent will do and, just as importantly, what it will never do.

Q6. Is the agent's purpose written down in one or two clear sentences?

☐Yes ☐Partially ☐No

Why it matters: If the team cannot explain the agent's purpose simply, users will not understand it either. A clear purpose statement also prevents scope creep.

Example: *"This agent triages incoming support emails, drafts replies for common delivery and order questions, and routes everything else to the right human team."*

What good looks like:

- A purpose statement that names the users, the task and the outcome.
- The statement is visible in project documents and in the agent's user-facing description.
- Every new feature request is checked against this statement.

Q7. Have you listed what the agent can do and what it must not do?

☐Yes ☐Partially ☐No

Why it matters: Explicit 'must not' rules are what stop an agent from taking well-intentioned but harmful actions.

Example: *A recruitment agent can screen applications against stated criteria and schedule interviews. It must not reject candidates, make offers or comment on salary.*

What good looks like:

- A written 'Can do / Must not do' list agreed by business owners.
- 'Must not' rules enforced technically (permissions, guardrails), not only in prompts.
- Rules reviewed whenever the agent gains new tools or data sources.

Q8. Does the agent have access only to the tools and data its role requires?

☐Yes ☐Partially ☐No

Why it matters: Least-privilege access limits the damage from mistakes, misuse or prompt injection. An agent cannot misuse a system it cannot reach.

Example: *The support agent needs read access to order history and write access to tickets. It does not need access to the payment gateway or to edit customer bank details.*

What good looks like:

- An access inventory listing every tool, API and data source the agent uses.
- Read-only access by default; write access only where clearly needed.
- Separate credentials for the agent, with activity logging enabled.

Q9. Is there a named owner accountable for the agent's outcomes?

☐Yes ☐Partially ☐No

Why it matters: When an automated action goes wrong, 'the AI did it' is not an answer. Users trust systems more when they know a person stands behind them.

Example: *The Head of Customer Support is named as the business owner, and a platform engineer is named as the technical owner responsible for monitoring and fixes.*

What good looks like:

- A named business owner and a named technical owner.

- A documented process for reporting and reviewing agent errors.
- Owners review performance at a regular cadence (for example, monthly).

Q10. Have you chosen the agent's autonomy level, and is the choice justified by task risk?

☐Yes ☐Partially ☐No

Why it matters: Giving an agent more independence just because it is technically possible is a common mistake. Autonomy should match the consequences of getting things wrong.

Example: *An email drafting assistant runs at Level 1 (suggestions only), while an internal scheduling agent runs at Level 3 (handles routine updates, humans handle exceptions).*

What good looks like:

- An autonomy level selected for each major task, using the table below.
- A written reason for the choice, based on risk and available safeguards.
- Clear criteria for moving to a higher level after proven reliability.

Autonomy level	Agent responsibility	Human involvement
Level 1: Assistive	Provides information and suggestions	Human makes all decisions
Level 2: Supervised	Performs tasks with regular approval	Human reviews important actions
Level 3: Conditional	Handles predefined tasks independently	Human intervenes in exceptions
Level 4: High autonomy	Manages approved workflows independently	Human monitors and handles escalations

Note: these levels are a practical design framework, not a universal industry standard.

Design for Transparency and Explainability

People trust what they can understand. Transparency means users can see what the agent is doing, why it is doing it and where its knowledge ends. This is where good UX design makes the biggest difference.

Q11. Do users know when they are interacting with an AI agent?

☐Yes ☐Partially ☐No

Why it matters: Hiding AI involvement damages trust once people discover it, and may breach regulations in some regions.

Example: *The support chat opens with: "Hi, I'm the virtual support assistant. I can help with orders and deliveries, and I'll connect you to our team for anything else."*

What good looks like:

- Clear AI disclosure at the start of every interaction.
- Agent-generated content labelled in internal tools (for example, 'Drafted by AI').
- An easy way to reach a human at any point.

Q12. Can the agent explain why it recommended or took an action?

☐Yes ☐Partially ☐No

Why it matters: Explanations help users judge whether to accept a recommendation, and they help teams debug errors.

Example: *When the agent marks a ticket as 'High priority', it shows: "Customer mentions a failed payment, order value is above ₹50,000, and this is the third contact in 48 hours."*

What good looks like:

- Short, plain-language reasons attached to key recommendations.
- Links to the source data or documents used.
- Explanations tested with real users to confirm they are understandable.

Q13. Are important actions logged and visible to users?

☐Yes ☐Partially ☐No

Why it matters: Logs create accountability and let users verify what happened, especially when outcomes are questioned later.

Example: *Every ticket shows an activity timeline: 'Agent retrieved order #4521', 'Agent drafted reply', 'Executive approved and sent reply'.*

What good looks like:

- An activity log for every action with timestamp, input and outcome.
- Logs visible to the relevant users, not just engineers.
- A retention period defined in line with data policies.

Q14. Can users see what the agent plans to do before it acts?

☐Yes ☐Partially ☐No

Why it matters: Previewing a plan gives users a chance to catch mistakes before they happen. This is the core of the Planner-Executor pattern.

Example: *A travel agent proposes: "1. Book flight AI-865 on Monday. 2. Reserve hotel near the client office. 3. Add itinerary to your calendar." The user approves or edits before any booking is made.*

What good looks like:

- Multi-step tasks shown as a plan before execution.
- Users can approve, edit or cancel individual steps.
- Plans written in business language, not technical jargon.

Q15. Are the agent's limitations clearly communicated?

☐Yes ☐Partially ☐No

Why it matters: Users who know where the agent is weak can rely on it appropriately, neither over-trusting nor ignoring it.

Example: *The HR policy agent states: "I answer using the current employee handbook. For individual cases such as medical leave, please contact HR directly."*

What good looks like:

- A short ‘What I can and can't help with’ statement in the interface.
- Confidence or source indicators where answers may be uncertain.
- Limitations updated as the agent's capabilities change.

SECTION 4 | DESIGN THINKING STAGE: PROTOTYPE

Keep Human Oversight at Key Decision Points

Oversight does not mean approving every action. That simply slows everyone down. Good oversight places human checkpoints only where the risk justifies them, so the agent stays fast on routine work and careful on important decisions.

Q16. Have you identified which decisions need human approval?

☐Yes ☐Partially ☐No

Why it matters: Without clearly defined checkpoints, teams either approve everything (slow and frustrating) or nothing (risky).

Example: *An invoice-processing agent verifies vendor details and matches invoices to purchase orders on its own, but sends payment release decisions to the finance manager.*

What good looks like:

- A list of decisions classified as 'auto', 'approve first' or 'human only'.
- Classification based on financial, legal, customer and reputational impact.
- Approvers named for each type of decision.

Q17. Are approval thresholds defined?

☐Yes ☐Partially ☐No

Why it matters: Thresholds turn vague rules ('big payments need approval') into clear, testable rules the agent can follow consistently.

Example: *Refunds under ₹2,000 are processed automatically; ₹2,000 to ₹20,000 need team lead approval; anything above ₹20,000 needs the support manager.*

What good looks like:

- Numeric or rule-based thresholds documented for each decision type.

- Thresholds enforced in the workflow logic, not left to the model's judgement.
- Thresholds reviewed periodically using real performance data.

Q18. Can users easily review, correct or reverse the agent's decisions?

☐Yes ☐Partially ☐No

Why it matters: People trust systems they can correct. Easy reversal also reduces the cost of any single mistake.

Example: *If the agent wrongly tags a ticket as 'Billing', the executive changes it with one click, and the correction is fed back into the agent's evaluation data.*

What good looks like:

- One-click edit, undo or override for agent actions where possible.
- Corrections captured and used to improve the agent.
- Reversal processes defined for actions that cannot be undone automatically (for example, sent emails).

Q19. Is the right design pattern selected for your workflow?

☐Yes ☐Partially ☐No

Why it matters: Design patterns give your workflow a proven structure for oversight. Choosing the wrong one creates either too much friction or too little control.

Example: *A procurement workflow uses Human-in-the-Loop for purchase approvals, while a policy Q&A assistant uses RAG so every answer is grounded in official documents.*

What good looks like:

- A pattern chosen deliberately, using the reference table below.
- Patterns combined where needed (for example, RAG plus Escalation).
- The choice documented with its reasoning.

Pattern	How it works	Best for
Human-in-the-Loop	Agent does routine work; human approves key actions	Financial, legal or reputational risk
Planner-Executor	Agent proposes a plan, then executes it	Multi-step tasks needing visibility

Supervisor-Agent	A lead agent coordinates specialist agents	Complex workflows across many systems
RAG	Agent retrieves approved sources before answering	Policy, product or knowledge Q&A
Escalation	Agent hands uncertain cases to humans	Customer-facing and variable requests

SECTION 5 | DESIGN THINKING STAGE: TEST

Design for Errors and Unexpected Situations

Every agent will eventually face missing data, confusing requests, broken integrations and situations it was never designed for. Trust is built not by avoiding all errors, but by handling them safely and visibly.

Q20. Does the agent recognise when it lacks information or confidence?

☐Yes ☐Partially ☐No

Why it matters: An agent that guesses confidently is more dangerous than one that admits uncertainty. Recognising gaps is the first step to safe behaviour.

Example: *When a customer writes "it's broken again" without an order number, the agent asks a clarifying question instead of guessing which order is meant.*

What good looks like:

- Clear rules for when the agent must ask, pause or escalate.
- Confidence checks on critical outputs (for example, identity or amount matching).
- Clarifying questions designed to be short and specific.

Q21. Is there a clear escalation path to a human for each failure type?

☐Yes ☐Partially ☐No

Why it matters: Escalation only builds trust if the handoff is smooth. Customers should not have to repeat themselves after being transferred.

Example: *If the agent cannot verify a customer's identity, it stops, transfers the chat to an authorised representative and passes along a summary of the conversation.*

What good looks like:

- An escalation matrix: failure type, who receives it, expected response time.
- Full context handed over with every escalation.
- Escalation rates tracked to spot recurring gaps.

Q22. Have you tested with incomplete, unexpected and edge-case inputs?

☐Yes ☐Partially ☐No

Why it matters: Agents that perform well on clean demo data often fail on real-world messages full of typos, mixed languages and missing details.

Example: *The test set includes emails in Hinglish, attachments with no text, duplicate requests, angry customers and attempts to make the agent ignore its instructions.*

What good looks like:

- A test suite covering normal, incomplete, unexpected and adversarial inputs.
- Tests that check behaviour (did it escalate correctly?) not just answer accuracy.
- Results reviewed with users before go-live.

Q23. Does the agent stop safely when a system or tool fails?

☐Yes ☐Partially ☐No

Why it matters: If an API times out halfway through a workflow, the agent must not leave records half-updated or retry endlessly.

Example: *If the CRM is unavailable while updating a lead, the agent pauses, logs the failure, notifies the owner and resumes only after the system is back, without creating duplicates.*

What good looks like:

- Defined fallback behaviour for every external tool.
- Safe defaults: when in doubt, stop and notify rather than continue.
- Retries limited and actions designed to avoid duplicates.

Q24. Are user data and privacy protected at every workflow stage?

☐Yes ☐Partially ☐No

Why it matters: Agents move data between systems, which creates new exposure points. A single privacy incident can undo months of trust building.

Example: *Before sending a ticket summary to an external model, the workflow masks phone numbers, email addresses and card details.*

What good looks like:

- Data flow mapped for each step, including any external services.
- Sensitive data masked, minimised or excluded where possible.

- Compliance with applicable regulations (for example, India's DPDP Act or GDPR).

Q25. Do you have metrics to track trust after launch?

☐Yes ☐Partially ☐No

Why it matters: Trust changes over time. Without measurement, you will not know whether people are relying on the agent appropriately or quietly working around it.

Example: *Three months after launch, the team notices the correction rate on billing tickets rising from 5% to 18%, which points to an outdated knowledge source.*

What good looks like:

- A trust dashboard using the metrics below.
- Targets and alert thresholds agreed with the business owner.
- Regular user feedback sessions alongside the numbers.

Trust metric	What it tells you
User adoption	Are people comfortable using the agent in daily work?

Task completion rate	Can the agent complete its assigned activities successfully?
Correction frequency	How often do users modify or reverse agent decisions?
Escalation rate	Does the agent recognise situations that need a human?
User confidence	Do users understand the agent's role and limits?
Appropriate reliance	Are people reviewing key decisions rather than blindly accepting or rejecting them?

Score Your Readiness

Count the number of questions you marked ‘Yes’. Treat each ‘Partially’ as a gap to close, not as half a point. Then use the table below to decide your next step.

Score (count of Yes)	Readiness	What it means
21–25	**Ready to build**	Strong human-centred foundation. Proceed, and keep

		monitoring trust metrics after launch.
15–20	**Needs refinement**	Close the gaps before development, especially any in Sections 4 and 5.
Below 15	**Revisit design**	Go back to user research and scope. Building now risks an agent people won't trust.

> **Tip: not all gaps are equal**
>
> A 'No' on Q8 (access), Q17 (thresholds), Q21 (escalation) or Q24 (privacy) is a blocker regardless of your total score. Fix these before any pilot with real users or data.

Your Top 3 Gaps to Fix

#	Gap (question no.)	Action to close it	Owner	Due date
1				
2				

3				

Example entry: *Gap: Q17, no refund thresholds. Action: agree ₹2,000 / ₹20,000 limits with finance and enforce in workflow. Owner: Support Lead. Due: end of sprint 2.*

YOUR NEXT STEP

Build AI Solutions People Actually Trust

This checklist gives you the questions. Design thinking gives you the method to answer them well. GSDC's Design Thinking Certification helps professionals build practical skills across the full design thinking process and apply them to real-world business and AI challenges.

What you'll learn

- **User research:** understand needs, behaviours and expectations before you build.
- **Problem definition:** frame the right problem and set clear boundaries.

- **Ideation:** generate and evaluate solution options with your team.
- **Prototyping:** turn ideas into testable workflows and experiences quickly.
- **Testing:** validate with real users and improve with evidence.

Who it's for

Product managers, UX designers, business analysts, AI professionals and anyone responsible for designing solutions that people need to adopt and trust.

GSDC
Global Skill Development Council

CERTIFIED DESIGN THINKING PROFESSIONAL

LEARN DIRECTLY FROM GLOBAL PRACTITIONERS, DESIGN THINKING EXPERTS, AND INDUSTRY LEADERS WHO ARE SHAPING THE FUTURE OF INNOVATION, CUSTOMER-CENTRIC DESIGN, AND PRODUCT EXPERIENCE STRATEGY.



ABOUT GSDC CERTIFICATION

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.

EBOOK

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

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

- Enhance professional credibility in design thinking.
- Expand career opportunities in innovation-driven roles
- Foster collaboration and creative problem-solving

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

Enroll Now

www.gsdcouncil.org